Міністерство освіти і науки України
Національний технічний університет України "КПІ"

# ВІСНИК

## НАЦІОНАЛЬНОГО ТЕХНІЧНОГО УНІВЕРСИТЕТУ УКРАЇНИ "КПІ"

## Інформатика, управління та обчислювальна техніка

Заснований у 1964 р.
Випуск 51

Київ "ВЕК+"2009

**УДК 004**

Вісник НТУУ "КПІ". Інформатика, управління та обчислювальна техніка: Зб. наук. пр. - К.: Век+, -2009. - №51. -240 С.

Рекомендований до друку Вченою радою факультету інформатики та обчислювальної техніки, протокол № 4 від 23.12.2009



Описано результати дослідження і створення компонентів обчислювальних й інформаційних систем і комплексів, пристроїв автоматики та передавання даних, систем автоматизації програмування, контролю й діагностики, штучного інтелекту тощо.
Для аспірантів, студентів, фахівців з обчислювальної техніки, систем керування, автоматизації програмування, штучного інтелекту та інших інформаційно-обчислювальних систем

ISSN 0201-744X
ISSN 0135-1729

Збірник наукових праць українською, англійською та російською мовами

Підп. до друку 26.10.2009. Формат 60×84 1/16. Гарнітура Times. Папір офсетний №1. Наклад 150 прим.
Надруковано в ЗАТ "ВІПОЛ", 03151, г.Київ, вул. Волинська, 60.

# З М І С Т

УДК 519.854.2

*ПАВЛОВ А.А.,*
*ИВАНОВА А.А.,*
*ШТАНЬКЕВИЧ А.С.,*
*ФЕДОТОВ А.П.*

# МОДИФИЦИРОВАННЫЙ МЕТОД АНАЛИЗА ИЕРАРХИЙ (ВЕРСИИ 2, 3)

В статье предлагаются модификации метода анализа иерархий. Модификации МАИ приведены для двухуровневой иерархии, затем полученные результаты адаптируются для общего случая. Приведены результаты статистических исследований, доказывающих эффективность данных модификаций по сравнению с классическим МАИ.

This article deals with the modifications of AHP. These modifications are developed for two levels hierarchy, then the results have been adapted for general case. The results of statistic researches, proving the effectiveness of the given modifications in comparison with classical AHP, are also proposed in the article.

В этой статье модели оптимизации [1-4] будут использованы для расширения области применения МАИ на случай большого количества альтернатив, (существенно превышающего их обычное количество, при котором применение МАИ считается корректным). Такая задача может возникнуть в двух случаях:

а) наилучшая альтернатива не выбирается из набора реально существующих альтернатив, а альтернативы генерируются искусственно для выбора наилучшей, после чего в реализацию этой альтернативы вкладываются существенные ресурсы;

б) искусственно генерируются альтернативы; с помощью МАИ находятся их результирующие веса, по которым строится аналитическое описание глобальной цели.

Корректное обоснование предлагаемых модификаций МАИ возможно в случае, когда задается формальная модель, которой отвечают эмпирические матрицы парных сравнений последнего уровня иерархий МАИ. В этом случае можно исследовать эмпирические (статистические) свойства алгоритмов, их эффективность, предлагать практические рекомендации к использованию.

Рассмотрим произвольные альтернативы $A_i$, $A_j$, которые сравниваются экспертом (экспертами) по эффективности относительного произвольного критерия предыдущего уровня иерархии МАИ.

В идеальном случае, т.е. в случае, когда предполагается, что на эксперта не действуют факторы, искажающие его решение (его компетентность, количество альтернатив, для которых строится эмпирическая матрица парных сравнений, неоднозначность качественного описания критерия, технология и последовательность заполнения эмпирической матрицы парных сравнений, психологические факторы и т.д.), значение эмпирического коэффициента $\gamma_{ij}$ ($\gamma_{ij}$ показывает во сколько раз вес объекта $A_i$ больше веса объекта $A_j$ по отношению к заданной цели) не зависит ни от количества альтернатив, из которых находится наилучшая, ни от их состава. Тогда $\gamma_{ij} = \dfrac{\omega_i}{\omega_j}$ в любой эмпирической матрице парных сравнений при парном сравнении альтернативы $A_i$ с альтернативой $A_j$, $\omega_i, \omega_j \geq 0$ интерпретируются как идеальные значения весов альтернатив $A_i$ и $A_j$.

Теперь рассмотрим случай, когда реально на решение эксперта действуют возмущающие факторы. Формально действие этих факторов предлагается описывать с помощью параметрического вероятностного распределения [5]. Закономерности, определяющие значения и изменение значений параметров вероятностного распределения (вероятностных распределений) как и само вероятностное распределение (вероятностные распределения) являются формальной моделью факторов, искажающих решение эксперта (экспертов).

*Примечание.* Использовалось параметрическое равномерное распределение, как одно из наиболее жестких распределений, для исследования эффективности предложенных моделей оптимизации.

Сначала модификацию МАИ приведем для двухуровневой иерархии, затем полу-

ченные результаты адаптируем для общего случая. Необходимо отметить, что двухуровневая иерархия имеет самостоятельное практическое значение.

## 1. Модифицированный метод анализа иерархий для двухуровневой структуры

Пусть дерево иерархий имеет вид:

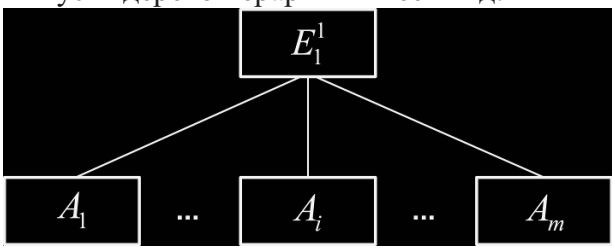



***Рис. 1. Дерево иерархии с двухуровневой структурой***

На рисунке 1 $m$ – количество альтернатив, достаточно большое число.

### 1.1 ММАИ (третья версия)

Искажение идеальных значений $\gamma_{ij}$ элементов эмпирической матрицы парных сравнений $\gamma_{ij}$* не зависит от ее размеров. (Предполагается, что она заполняется блочно подматрицами размерности не превышающей 7 ÷ 9). Искажение $\gamma_{ij}$* истинного значения $\gamma_{ij}$ $(\forall \gamma_{ij} \geq 1, i \neq j)$ тем больше, чем больше $\gamma_{ij}$ отличается от единицы.

*Примечание.* Для оценки весов $\omega_i, i = \overline{1,m}$; по прежнему используются только те элементы эмпирической матрицы парных сравнений для которых $\gamma_{ij}^* \geq 1\,(i \neq j)$.

Иными словами, чем больше альтернативы не сопоставимы между собой, тем больше ошибается эксперт. В этом случае веса альтернатив находятся по исходной эмпирической матрице парных сравнений (размерности $m \times m$) с помощью моделей оптимизации [1–4] с измененными функционалами: каждая составляющая функционала, соответствующая $\gamma_{ij}$, умножается на неотрицательный коэффициент, значение которого определяется из следующей посылки: величина его тем больше, чем меньше число $\gamma_{ij}$*–1 ($\gamma_{ij}$*≥1), взято из эмпирической матрицы парных сравнений.

*Примечание*. В этом случае используются модели 2 из [3] и модель из [4], которые в результате большого количества статистических экспериментов статистически значимо дают результаты лучше других моделей. Альтернатива с большим весом является наилучшей. Эффективными с аналогичным образом измененными функционалами также являются модели оптимизации, приведенные для эмпирических матриц парных сравнений с односторонними ограничениями [2].

Для эффективного использования ММАИ (третья версия) необходимо решить следующие задачи:

*Задача 1.* Определить и формализовать законы (в том числе вероятностные) искажения идеальных значений $\gamma_{ij}$ ($\gamma_{ij} \geq 1$, $i \neq j$), , зависящие от параметра $\gamma_{ij}-1$, соблюдая принцип: чем больше число $\gamma_{ij}-1$, тем больше искажение.

*Задача 2.* Для каждого типа закона искажения идеальных значений $\gamma_{ij}$ в результате статистического моделирования определить наилучшую (лучшие) модели оптимизации с взвешенными составляющими функционала; сформулировать правила определения значений взвешенных коэффициентов с учетом того, что они определяются по искаженным элементам $\gamma_{ij}$* (взятым из эмпирической матрицы парных сравнений), и тем больше, чем меньше число $\gamma_{ij}$* – 1($\gamma_{ij}$* ≥1, $i \neq j$).

*Пример.*

Имеется 40 альтернатив, истинные веса которых приведены в таблице 1. Предположим, что их веса не известны исследователю, веса альтернатив необходимо восстановить по эмпирической матрице парных сравнений, приведенной в таблицах 3-6. На практике эта матрица формируется экспертами путем проведения парных сравнений альтернатив. Но в данном примере мы сгенерировали матрицу автоматически из идеальной абсолютно согласованной матрицы (полученной с помощью весов из таблицы 1) путем вероятностного отклонения её элементов по принципу $\gamma_{ij}$*=$\gamma_{ij}$+$k_{ij}\gamma_{ij}$ со следующей особенностью: коэффициент $k_{ij}$ распределен равномерно в интервале $(-t,t)$, где $t$ зависит от величины истинного $\gamma_{ij}$ согласно таблице 2.

***Табл. 1. Веса альтернатив***

| № | Вес альтернативы | № | Вес альтернативы | № | Вес альтернативы | № | Вес альтернативы |
|---|---|---|---|---|---|---|---|
| 1 | 0,05379181 | 11 | 0,04082171 | 21 | 0,02289497 | 31 | 0,01168550 |
| 2 | 0,05366083 | 12 | 0,03101045 | 22 | 0,02284181 | 32 | 0,00757647 |
| 3 | 0,05357093 | 13 | 0,03100458 | 23 | 0,01769093 | 33 | 0,00757482 |
| 4 | 0,05341230 | 14 | 0,03089293 | 24 | 0,01762088 | 34 | 0,00755086 |
| 5 | 0,05340997 | 15 | 0,03087690 | 25 | 0,01754262 | 35 | 0,00731306 |
| 6 | 0,04114577 | 16 | 0,03082695 | 26 | 0,01740996 | 36 | 0,00729837 |
| 7 | 0,04107208 | 17 | 0,03073138 | 27 | 0,01191967 | 37 | 0,00308611 |
| 8 | 0,04093711 | 18 | 0,03071119 | 28 | 0,01185761 | 38 | 0,00298422 |
| 9 | 0,04089968 | 19 | 0,02299770 | 29 | 0,01181520 | 39 | 0,00297701 |
| 10 | 0,04088685 | 20 | 0,02296137 | 30 | 0,01178852 | 40 | 0,00294893 |

***Табл. 2. Значения параметра $t$ в зависимости от значения $\gamma_{ij}$***

| Интервал принадлежности $\gamma_{ij}$ | Значение $t$ |
|---|---|
| 1 | 0 |
| (1;1,1] | 0,003 |
| (1,1;1,3] | 0,009 |
| (1,3;1,6] | 0,018 |
| (1,6;2] | 0,03 |
| (2;2,5] | 0,045 |
| (2,5;3,1] | 0,063 |
| (3,1;3,8] | 0,084 |
| (3,8;4,6] | 0,108 |
| (4,6;5,5] | 0,135 |
| (5,5;6,5] | 0,165 |
| (6,5;7,6] | 0,198 |
| (7,6;8,8] | 0,234 |
| (8,8;10,1] | 0,273 |
| $(10,1;\infty)$ | 0,315 |

***Табл. 3. Эмпирическая матрица парных сравнений (столбцы 1-10)***

| **№/№** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** | **10** |
|---|---|---|---|---|---|---|---|---|---|---|
| **1** | 1,0000 | 1,0055 | 1,0071 | 1,0101 | 1,0102 | 1,3364 | 1,3388 | 1,3433 | 1,3446 | 1,3450 |
| **2** | 0,9946 | 1,0000 | 1,0047 | 1,0077 | 1,0077 | 1,3331 | 1,3355 | 1,3400 | 1,3412 | 1,3417 |
| **3** | 0,9929 | 0,9953 | 1,0000 | 1,0060 | 1,0060 | 1,3309 | 1,3333 | 1,3377 | 1,3390 | 1,3394 |
| **4** | 0,9900 | 0,9924 | 0,9940 | 1,0000 | 1,0030 | 1,3125 | 1,3293 | 1,3337 | 1,3349 | 1,3354 |
| **5** | 0,9899 | 0,9923 | 0,9940 | 0,9970 | 1,0000 | 1,3124 | 1,3292 | 1,3337 | 1,3349 | 1,3353 |
| **6** | 0,7483 | 0,7501 | 0,7514 | 0,7619 | 0,7619 | 1,0000 | 1,0048 | 1,0081 | 1,0091 | 1,0094 |
| **7** | 0,7469 | 0,7488 | 0,7500 | 0,7523 | 0,7523 | 0,9952 | 1,0000 | 1,0063 | 1,0072 | 1,0076 |
| **8** | 0,7444 | 0,7463 | 0,7475 | 0,7498 | 0,7498 | 0,9919 | 0,9937 | 1,0000 | 1,0039 | 1,0042 |
| **9** | 0,7437 | 0,7456 | 0,7468 | 0,7491 | 0,7491 | 0,9910 | 0,9928 | 0,9961 | 1,0000 | 1,0033 |
| **10** | 0,7435 | 0,7453 | 0,7466 | 0,7489 | 0,7489 | 0,9907 | 0,9925 | 0,9958 | 0,9967 | 1,0000 |
| **11** | 0,7423 | 0,7441 | 0,7454 | 0,7476 | 0,7477 | 0,9891 | 0,9909 | 0,9942 | 0,9951 | 0,9954 |
| **12** | 0,5529 | 0,5543 | 0,5552 | 0,5569 | 0,5569 | 0,7371 | 0,7385 | 0,7409 | 0,7416 | 0,7419 |
| **13** | 0,5528 | 0,5541 | 0,5551 | 0,5568 | 0,5568 | 0,7370 | 0,7383 | 0,7408 | 0,7415 | 0,7417 |
| **14** | 0,5507 | 0,5521 | 0,5531 | 0,5547 | 0,5548 | 0,7343 | 0,7356 | 0,7381 | 0,7388 | 0,7390 |
| **15** | 0,5505 | 0,5518 | 0,5528 | 0,5544 | 0,5545 | 0,7339 | 0,7353 | 0,7377 | 0,7384 | 0,7386 |
| **16** | 0,5496 | 0,5509 | 0,5519 | 0,5535 | 0,5535 | 0,7327 | 0,7340 | 0,7365 | 0,7372 | 0,7374 |
| **17** | 0,5478 | 0,5492 | 0,5501 | 0,5518 | 0,5518 | 0,7304 | 0,7317 | 0,7342 | 0,7349 | 0,7351 |
| **18** | 0,5475 | 0,5488 | 0,5498 | 0,5514 | 0,5514 | 0,7299 | 0,7313 | 0,7337 | 0,7344 | 0,7346 |
| **19** | 0,3993 | 0,4003 | 0,4010 | 0,4022 | 0,4022 | 0,5358 | 0,5367 | 0,5385 | 0,5390 | 0,5392 |
| **20** | 0,3986 | 0,3996 | 0,4003 | 0,4015 | 0,4015 | 0,5349 | 0,5359 | 0,5377 | 0,5382 | 0,5384 |
| **21** | 0,3975 | 0,3985 | 0,3991 | 0,4003 | 0,4004 | 0,5333 | 0,5343 | 0,5361 | 0,5366 | 0,5368 |
| **22** | 0,3965 | 0,3975 | 0,3982 | 0,3994 | 0,3994 | 0,5321 | 0,5331 | 0,5348 | 0,5353 | 0,5355 |

| №/№ | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|---|---|---|---|
| **23** | 0,2976 | 0,2983 | 0,2988 | 0,2997 | 0,2997 | 0,4016 | 0,4023 | 0,4037 | 0,4040 | 0,4042 |
| **24** | 0,2964 | 0,2971 | 0,2976 | 0,2985 | 0,2985 | 0,4000 | 0,4007 | 0,4020 | 0,4024 | 0,4025 |
| **25** | 0,2950 | 0,2957 | 0,2962 | 0,2971 | 0,2972 | 0,3982 | 0,3989 | 0,4002 | 0,4006 | 0,4007 |
| **26** | 0,2927 | 0,2935 | 0,2940 | 0,2949 | 0,2949 | 0,3951 | 0,3958 | 0,3971 | 0,3975 | 0,3976 |
| **27** | 0,1859 | 0,1863 | 0,1867 | 0,1872 | 0,1872 | 0,2533 | 0,2538 | 0,2546 | 0,2549 | 0,2549 |
| **28** | 0,1849 | 0,1854 | 0,1857 | 0,1862 | 0,1862 | 0,2520 | 0,2524 | 0,2533 | 0,2535 | 0,2536 |
| **29** | 0,1842 | 0,1847 | 0,1850 | 0,1856 | 0,1856 | 0,2510 | 0,2515 | 0,2523 | 0,2526 | 0,2527 |
| **30** | 0,1838 | 0,1843 | 0,1846 | 0,1851 | 0,1851 | 0,2505 | 0,2509 | 0,2518 | 0,2520 | 0,2521 |
| **31** | 0,1751 | 0,1826 | 0,1829 | 0,1835 | 0,1835 | 0,2482 | 0,2487 | 0,2495 | 0,2497 | 0,2498 |
| **32** | 0,1030 | 0,1032 | 0,1034 | 0,1037 | 0,1037 | 0,1479 | 0,1482 | 0,1487 | 0,1488 | 0,1488 |
| **33** | 0,1029 | 0,1032 | 0,1034 | 0,1037 | 0,1037 | 0,1479 | 0,1481 | 0,1486 | 0,1488 | 0,1488 |
| **34** | 0,1026 | 0,1029 | 0,1030 | 0,1033 | 0,1033 | 0,1474 | 0,1476 | 0,1481 | 0,1483 | 0,1483 |
| **35** | 0,0993 | 0,0995 | 0,0997 | 0,1000 | 0,1000 | 0,1366 | 0,1369 | 0,1374 | 0,1375 | 0,1375 |
| **36** | 0,0991 | 0,0993 | 0,0995 | 0,0998 | 0,0998 | 0,1364 | 0,1366 | 0,1371 | 0,1372 | 0,1373 |
| **37** | 0,0356 | 0,0357 | 0,0357 | 0,0358 | 0,0358 | 0,0467 | 0,0468 | 0,0469 | 0,0470 | 0,0470 |
| **38** | 0,0344 | 0,0345 | 0,0345 | 0,0347 | 0,0347 | 0,0451 | 0,0452 | 0,0454 | 0,0454 | 0,0454 |
| **39** | 0,0343 | 0,0344 | 0,0345 | 0,0346 | 0,0346 | 0,0450 | 0,0451 | 0,0453 | 0,0453 | 0,0453 |
| **40** | 0,0340 | 0,0341 | 0,0341 | 0,0342 | 0,0342 | 0,0446 | 0,0447 | 0,0448 | 0,0449 | 0,0449 |

***Табл. 4. Эмпирическая матрица парных сравнений (столбцы 11-20)***

| №/№ | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 |
|---|---|---|---|---|---|---|---|---|---|---|
| **1** | 1,3472 | 1,8087 | 1,8091 | 1,8157 | 1,8167 | 1,8197 | 1,8254 | 1,8266 | 2,5045 | 2,5086 |
| **2** | 1,3438 | 1,8042 | 1,8046 | 1,8112 | 1,8122 | 1,8152 | 1,8209 | 1,8221 | 2,4983 | 2,5023 |
| **3** | 1,3416 | 1,8012 | 1,8015 | 1,8081 | 1,8091 | 1,8121 | 1,8178 | 1,8190 | 2,4940 | 2,4981 |
| **4** | 1,3375 | 1,7957 | 1,7961 | 1,8027 | 1,8036 | 1,8066 | 1,8123 | 1,8135 | 2,4865 | 2,4905 |
| **5** | 1,3375 | 1,7957 | 1,7960 | 1,8026 | 1,8036 | 1,8065 | 1,8122 | 1,8135 | 2,4864 | 2,4904 |
| **6** | 1,0110 | 1,3566 | 1,3569 | 1,3618 | 1,3625 | 1,3648 | 1,3691 | 1,3700 | 1,8665 | 1,8695 |
| **7** | 1,0092 | 1,3541 | 1,3544 | 1,3594 | 1,3601 | 1,3623 | 1,3666 | 1,3675 | 1,8631 | 1,8661 |
| **8** | 1,0058 | 1,3496 | 1,3499 | 1,3548 | 1,3555 | 1,3578 | 1,3621 | 1,3630 | 1,8569 | 1,8598 |
| **9** | 1,0049 | 1,3484 | 1,3486 | 1,3536 | 1,3543 | 1,3565 | 1,3608 | 1,3617 | 1,8551 | 1,8581 |
| **10** | 1,0046 | 1,3480 | 1,3482 | 1,3531 | 1,3539 | 1,3561 | 1,3604 | 1,3613 | 1,8545 | 1,8575 |
| **11** | 1,0000 | 1,3458 | 1,3460 | 1,3510 | 1,3517 | 1,3539 | 1,3582 | 1,3591 | 1,8515 | 1,8545 |
| **12** | 0,7431 | 1,0000 | 1,0032 | 1,0068 | 1,0074 | 1,0090 | 1,0121 | 1,0128 | 1,3790 | 1,3812 |
| **13** | 0,7429 | 0,9968 | 1,0000 | 1,0066 | 1,0072 | 1,0088 | 1,0119 | 1,0126 | 1,3787 | 1,3809 |
| **14** | 0,7402 | 0,9932 | 0,9934 | 1,0000 | 1,0035 | 1,0052 | 1,0083 | 1,0090 | 1,3737 | 1,3759 |
| **15** | 0,7398 | 0,9927 | 0,9929 | 0,9965 | 1,0000 | 1,0046 | 1,0078 | 1,0084 | 1,3729 | 1,3751 |
| **16** | 0,7386 | 0,9911 | 0,9913 | 0,9949 | 0,9954 | 1,0000 | 1,0061 | 1,0068 | 1,3707 | 1,3729 |
| **17** | 0,7363 | 0,9880 | 0,9882 | 0,9918 | 0,9923 | 0,9939 | 1,0000 | 1,0037 | 1,3664 | 1,3686 |
| **18** | 0,7358 | 0,9874 | 0,9875 | 0,9911 | 0,9916 | 0,9933 | 0,9964 | 1,0000 | 1,3655 | 1,3677 |
| **19** | 0,5401 | 0,7252 | 0,7253 | 0,7280 | 0,7284 | 0,7296 | 0,7319 | 0,7323 | 1,0000 | 1,0046 |
| **20** | 0,5392 | 0,7240 | 0,7242 | 0,7268 | 0,7272 | 0,7284 | 0,7307 | 0,7312 | 0,9954 | 1,0000 |
| **21** | 0,5376 | 0,7219 | 0,7220 | 0,7247 | 0,7251 | 0,7263 | 0,7285 | 0,7290 | 0,9925 | 0,9941 |
| **22** | 0,5364 | 0,7202 | 0,7203 | 0,7230 | 0,7234 | 0,7245 | 0,7268 | 0,7273 | 0,9902 | 0,9918 |
| **23** | 0,4048 | 0,5470 | 0,5471 | 0,5491 | 0,5494 | 0,5503 | 0,5521 | 0,5524 | 0,7608 | 0,7620 |
| **24** | 0,4032 | 0,5448 | 0,5449 | 0,5469 | 0,5472 | 0,5481 | 0,5499 | 0,5502 | 0,7496 | 0,7508 |
| **25** | 0,4014 | 0,5424 | 0,5425 | 0,5445 | 0,5447 | 0,5456 | 0,5474 | 0,5477 | 0,7462 | 0,7474 |
| **26** | 0,3983 | 0,5382 | 0,5383 | 0,5403 | 0,5406 | 0,5415 | 0,5432 | 0,5435 | 0,7405 | 0,7417 |
| **27** | 0,2554 | 0,3489 | 0,3489 | 0,3502 | 0,3504 | 0,3510 | 0,3521 | 0,3523 | 0,4962 | 0,4970 |
| **28** | 0,2540 | 0,3470 | 0,3471 | 0,3484 | 0,3485 | 0,3491 | 0,3502 | 0,3505 | 0,4936 | 0,4944 |
| **29** | 0,2531 | 0,3457 | 0,3458 | 0,3471 | 0,3473 | 0,3478 | 0,3490 | 0,3492 | 0,4918 | 0,4926 |
| **30** | 0,2525 | 0,3449 | 0,3450 | 0,3463 | 0,3465 | 0,3470 | 0,3481 | 0,3484 | 0,4907 | 0,4915 |
| **31** | 0,2502 | 0,3419 | 0,3419 | 0,3432 | 0,3434 | 0,3439 | 0,3450 | 0,3453 | 0,4863 | 0,4871 |
| **32** | 0,1491 | 0,2054 | 0,2054 | 0,2062 | 0,2063 | 0,2066 | 0,2073 | 0,2074 | 0,2981 | 0,2986 |
| **33** | 0,1490 | 0,2053 | 0,2054 | 0,2061 | 0,2062 | 0,2066 | 0,2072 | 0,2074 | 0,2980 | 0,2985 |
| **34** | 0,1486 | 0,2047 | 0,2047 | 0,2055 | 0,2056 | 0,2059 | 0,2066 | 0,2067 | 0,2970 | 0,2975 |
| **35** | 0,1378 | 0,1981 | 0,1981 | 0,1989 | 0,1990 | 0,1993 | 0,1999 | 0,2001 | 0,2786 | 0,2791 |
| **36** | 0,1375 | 0,1977 | 0,1977 | 0,1984 | 0,1986 | 0,1989 | 0,1995 | 0,1996 | 0,2781 | 0,2785 |
| **37** | 0,0471 | 0,0655 | 0,0655 | 0,0658 | 0,0658 | 0,0659 | 0,0661 | 0,0662 | 0,0980 | 0,0981 |
| **38** | 0,0455 | 0,0602 | 0,0602 | 0,0604 | 0,0604 | 0,0605 | 0,0607 | 0,0608 | 0,0903 | 0,0904 |

| №/№ | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 |
|---|---|---|---|---|---|---|---|---|---|---|
| **39** | 0,0454 | 0,0600 | 0,0600 | 0,0602 | 0,0603 | 0,0604 | 0,0606 | 0,0606 | 0,0900 | 0,0902 |
| **40** | 0,0449 | 0,0594 | 0,0594 | 0,0597 | 0,0597 | 0,0598 | 0,0600 | 0,0600 | 0,0892 | 0,0893 |

***Табл. 5. Эмпирическая матрица парных сравнений (столбцы 21-30)***

| №/№ | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 |
|---|---|---|---|---|---|---|---|---|---|---|
| **1** | 2,5160 | 2,5219 | 3,3608 | 3,3744 | 3,3897 | 3,4160 | 5,3796 | 5,4084 | 5,4282 | 5,4407 |
| **2** | 2,5097 | 2,5157 | 3,3524 | 3,3660 | 3,3813 | 3,4075 | 5,3663 | 5,3949 | 5,4147 | 5,4272 |
| **3** | 2,5054 | 2,5114 | 3,3467 | 3,3603 | 3,3755 | 3,4017 | 5,3571 | 5,3857 | 5,4054 | 5,4179 |
| **4** | 2,4979 | 2,5038 | 3,3366 | 3,3501 | 3,3654 | 3,3915 | 5,3409 | 5,3694 | 5,3891 | 5,4015 |
| **5** | 2,4978 | 2,5037 | 3,3365 | 3,3500 | 3,3652 | 3,3913 | 5,3407 | 5,3692 | 5,3889 | 5,4013 |
| **6** | 1,8750 | 1,8794 | 2,4901 | 2,5002 | 2,5116 | 2,5310 | 3,9478 | 3,9689 | 3,9835 | 3,9927 |
| **7** | 1,8716 | 1,8760 | 2,4856 | 2,4957 | 2,5070 | 2,5264 | 3,9406 | 3,9617 | 3,9762 | 3,9854 |
| **8** | 1,8653 | 1,8697 | 2,4773 | 2,4873 | 2,4986 | 2,5180 | 3,9274 | 3,9484 | 3,9629 | 3,9720 |
| **9** | 1,8636 | 1,8680 | 2,4750 | 2,4850 | 2,4963 | 2,5156 | 3,9237 | 3,9447 | 3,9592 | 3,9683 |
| **10** | 1,8630 | 1,8674 | 2,4742 | 2,4842 | 2,4955 | 2,5148 | 3,9225 | 3,9434 | 3,9579 | 3,9670 |
| **11** | 1,8600 | 1,8644 | 2,4702 | 2,4802 | 2,4914 | 2,5108 | 3,9161 | 3,9370 | 3,9515 | 3,9606 |
| **12** | 1,3852 | 1,3885 | 1,8281 | 1,8355 | 1,8438 | 1,8581 | 2,8664 | 2,8818 | 2,8923 | 2,8990 |
| **13** | 1,3850 | 1,3882 | 1,8277 | 1,8351 | 1,8434 | 1,8577 | 2,8659 | 2,8812 | 2,8918 | 2,8985 |
| **14** | 1,3799 | 1,3832 | 1,8210 | 1,8284 | 1,8367 | 1,8509 | 2,8553 | 2,8706 | 2,8811 | 2,8878 |
| **15** | 1,3792 | 1,3824 | 1,8201 | 1,8274 | 1,8357 | 1,8499 | 2,8538 | 2,8691 | 2,8796 | 2,8863 |
| **16** | 1,3769 | 1,3802 | 1,8171 | 1,8244 | 1,8327 | 1,8469 | 2,8491 | 2,8643 | 2,8748 | 2,8815 |
| **17** | 1,3726 | 1,3758 | 1,8114 | 1,8187 | 1,8269 | 1,8411 | 2,8401 | 2,8553 | 2,8657 | 2,8724 |
| **18** | 1,3717 | 1,3749 | 1,8101 | 1,8175 | 1,8257 | 1,8398 | 2,8382 | 2,8533 | 2,8638 | 2,8704 |
| **19** | 1,0075 | 1,0099 | 1,3144 | 1,3341 | 1,3402 | 1,3505 | 2,0152 | 2,0259 | 2,0332 | 2,0379 |
| **20** | 1,0059 | 1,0083 | 1,3123 | 1,3320 | 1,3380 | 1,3483 | 2,0119 | 2,0226 | 2,0300 | 2,0346 |
| **21** | 1,0000 | 1,0053 | 1,3085 | 1,3137 | 1,3341 | 1,3444 | 2,0060 | 2,0167 | 2,0240 | 2,0287 |
| **22** | 0,9947 | 1,0000 | 1,3054 | 1,3106 | 1,3310 | 1,3412 | 2,0013 | 2,0119 | 2,0193 | 2,0239 |
| **23** | 0,7643 | 0,7660 | 1,0000 | 1,0070 | 1,0115 | 1,0192 | 1,5196 | 1,5277 | 1,5332 | 1,5367 |
| **24** | 0,7612 | 0,7630 | 0,9930 | 1,0000 | 1,0075 | 1,0152 | 1,5135 | 1,5215 | 1,5271 | 1,5306 |
| **25** | 0,7496 | 0,7513 | 0,9886 | 0,9926 | 1,0000 | 1,0107 | 1,5067 | 1,5147 | 1,5202 | 1,5237 |
| **26** | 0,7438 | 0,7456 | 0,9811 | 0,9850 | 0,9894 | 1,0000 | 1,4952 | 1,5031 | 1,5086 | 1,5120 |
| **27** | 0,4985 | 0,4997 | 0,6581 | 0,6607 | 0,6637 | 0,6688 | 1,0000 | 1,0083 | 1,0119 | 1,0142 |
| **28** | 0,4959 | 0,4970 | 0,6546 | 0,6572 | 0,6602 | 0,6653 | 0,9918 | 1,0000 | 1,0066 | 1,0089 |
| **29** | 0,4941 | 0,4952 | 0,6522 | 0,6549 | 0,6578 | 0,6629 | 0,9882 | 0,9934 | 1,0000 | 1,0053 |
| **30** | 0,4929 | 0,4941 | 0,6507 | 0,6534 | 0,6563 | 0,6614 | 0,9860 | 0,9912 | 0,9948 | 1,0000 |
| **31** | 0,4886 | 0,4897 | 0,6450 | 0,6476 | 0,6505 | 0,6555 | 0,9774 | 0,9825 | 0,9860 | 0,9883 |
| **32** | 0,2994 | 0,3001 | 0,4000 | 0,4016 | 0,4034 | 0,4066 | 0,6204 | 0,6237 | 0,6259 | 0,6274 |
| **33** | 0,2994 | 0,3001 | 0,3999 | 0,4015 | 0,4033 | 0,4065 | 0,6203 | 0,6235 | 0,6258 | 0,6272 |
| **34** | 0,2984 | 0,2991 | 0,3986 | 0,4002 | 0,4020 | 0,4052 | 0,6183 | 0,6215 | 0,6238 | 0,6252 |
| **35** | 0,2799 | 0,2806 | 0,3858 | 0,3874 | 0,3892 | 0,3922 | 0,5890 | 0,5922 | 0,5943 | 0,5957 |
| **36** | 0,2793 | 0,2800 | 0,3850 | 0,3866 | 0,3884 | 0,3914 | 0,5878 | 0,5910 | 0,5931 | 0,5945 |
| **37** | 0,0984 | 0,0987 | 0,1341 | 0,1346 | 0,1352 | 0,1363 | 0,2179 | 0,2191 | 0,2199 | 0,2204 |
| **38** | 0,0907 | 0,0909 | 0,1295 | 0,1301 | 0,1307 | 0,1317 | 0,2106 | 0,2117 | 0,2125 | 0,2130 |
| **39** | 0,0905 | 0,0907 | 0,1292 | 0,1297 | 0,1303 | 0,1314 | 0,2101 | 0,2112 | 0,2120 | 0,2124 |
| **40** | 0,0896 | 0,0898 | 0,1280 | 0,1285 | 0,1291 | 0,1301 | 0,2080 | 0,2091 | 0,2099 | 0,2104 |

***Табл. 6. Эмпирическая матрица парных сравнений (столбцы 31-40)***

| №/№ | 31 | 32 | 33 | 34 | 35 | 36 | 37 | 38 | 39 | 40 |
|---|---|---|---|---|---|---|---|---|---|---|
| **1** | 5,7112 | 9,7134 | 9,7155 | 9,7470 | 10,0704 | 10,0911 | 28,0964 | 29,0664 | 29,1376 | 29,4180 |
| **2** | 5,4760 | 9,6893 | 9,6914 | 9,7228 | 10,0454 | 10,0660 | 28,0272 | 28,9949 | 29,0659 | 29,3456 |
| **3** | 5,4666 | 9,6727 | 9,6748 | 9,7062 | 10,0282 | 10,0488 | 27,9797 | 28,9458 | 29,0167 | 29,2959 |
| **4** | 5,4501 | 9,6435 | 9,6456 | 9,6768 | 9,9979 | 10,0185 | 27,8959 | 28,8591 | 28,9298 | 29,2082 |
| **5** | 5,4499 | 9,6430 | 9,6452 | 9,6764 | 9,9975 | 10,0180 | 27,8947 | 28,8579 | 28,9286 | 29,2069 |
| **6** | 4,0286 | 6,7620 | 6,7635 | 6,7854 | 7,3180 | 7,3331 | 21,4171 | 22,1591 | 22,2135 | 22,4280 |
| **7** | 4,0213 | 6,7497 | 6,7512 | 6,7730 | 7,3046 | 7,3197 | 21,3782 | 22,1188 | 22,1732 | 22,3873 |
| **8** | 4,0078 | 6,7271 | 6,7285 | 6,7503 | 7,2801 | 7,2951 | 21,3069 | 22,0451 | 22,0993 | 22,3127 |
| **9** | 4,0040 | 6,7208 | 6,7223 | 6,7440 | 7,2733 | 7,2882 | 21,2871 | 22,0247 | 22,0788 | 22,2920 |
| **10** | 4,0028 | 6,7186 | 6,7201 | 6,7419 | 7,2709 | 7,2859 | 21,2803 | 22,0177 | 22,0718 | 22,2849 |
| **11** | 3,9962 | 6,7077 | 6,7092 | 6,7309 | 7,2591 | 7,2740 | 21,2459 | 21,9821 | 22,0361 | 22,2489 |

| №/№ | 31 | 32 | 33 | 34 | 35 | 36 | 37 | 38 | 39 | 40 |
|---|---|---|---|---|---|---|---|---|---|---|
| **12** | 2,9251 | 4,8691 | 4,8702 | 4,8860 | 5,0483 | 5,0587 | 15,2618 | 16,6231 | 16,6642 | 16,8258 |
| **13** | 2,9246 | 4,8681 | 4,8692 | 4,8850 | 5,0474 | 5,0578 | 15,2589 | 16,6199 | 16,6609 | 16,8225 |
| **14** | 2,9138 | 4,8502 | 4,8513 | 4,8670 | 5,0288 | 5,0391 | 15,2029 | 16,5589 | 16,5998 | 16,7608 |
| **15** | 2,9123 | 4,8476 | 4,8487 | 4,8645 | 5,0261 | 5,0365 | 15,1949 | 16,5502 | 16,5910 | 16,7520 |
| **16** | 2,9074 | 4,8396 | 4,8407 | 4,8564 | 5,0178 | 5,0282 | 15,1699 | 16,5229 | 16,5637 | 16,7244 |
| **17** | 2,8982 | 4,8243 | 4,8254 | 4,8410 | 5,0019 | 5,0122 | 15,1220 | 16,4707 | 16,5114 | 16,6715 |
| **18** | 2,8963 | 4,8211 | 4,8221 | 4,8378 | 4,9986 | 5,0089 | 15,1119 | 16,4597 | 16,5003 | 16,6604 |
| **19** | 2,0561 | 3,3549 | 3,3556 | 3,3665 | 3,5891 | 3,5965 | 10,2050 | 11,0791 | 11,1065 | 11,2144 |
| **20** | 2,0528 | 3,3495 | 3,3502 | 3,3610 | 3,5833 | 3,5906 | 10,1886 | 11,0612 | 11,0885 | 11,1963 |
| **21** | 2,0468 | 3,3396 | 3,3403 | 3,3511 | 3,5727 | 3,5800 | 10,1585 | 11,0285 | 11,0558 | 11,1633 |
| **22** | 2,0420 | 3,3317 | 3,3324 | 3,3432 | 3,5642 | 3,5715 | 10,1345 | 11,0024 | 11,0296 | 11,1368 |
| **23** | 1,5504 | 2,5001 | 2,5007 | 2,5088 | 2,5918 | 2,5971 | 7,4591 | 7,7195 | 7,7386 | 7,8138 |
| **24** | 1,5442 | 2,4901 | 2,4906 | 2,4987 | 2,5814 | 2,5867 | 7,4289 | 7,6882 | 7,7073 | 7,7822 |
| **25** | 1,5373 | 2,4788 | 2,4793 | 2,4874 | 2,5697 | 2,5750 | 7,3952 | 7,6534 | 7,6723 | 7,7469 |
| **26** | 1,5255 | 2,4597 | 2,4603 | 2,4682 | 2,5499 | 2,5552 | 7,3381 | 7,5942 | 7,6130 | 7,6871 |
| **27** | 1,0232 | 1,6119 | 1,6122 | 1,6174 | 1,6977 | 1,7012 | 4,5886 | 4,7490 | 4,7608 | 4,8071 |
| **28** | 1,0178 | 1,6034 | 1,6038 | 1,6089 | 1,6887 | 1,6922 | 4,5642 | 4,7237 | 4,7354 | 4,7815 |
| **29** | 1,0142 | 1,5976 | 1,5980 | 1,6031 | 1,6826 | 1,6860 | 4,5475 | 4,7064 | 4,7181 | 4,7640 |
| **30** | 1,0119 | 1,5940 | 1,5943 | 1,5994 | 1,6787 | 1,6821 | 4,5370 | 4,6955 | 4,7072 | 4,7530 |
| **31** | 1,0000 | 1,5799 | 1,5802 | 1,5853 | 1,6374 | 1,6672 | 4,3386 | 4,6536 | 4,6651 | 4,7105 |
| **32** | 0,6330 | 1,0000 | 1,0032 | 1,0064 | 1,0392 | 1,0413 | 2,6310 | 2,7957 | 2,8027 | 2,8299 |
| **33** | 0,6328 | 0,9968 | 1,0000 | 1,0062 | 1,0390 | 1,0411 | 2,6304 | 2,7951 | 2,8020 | 2,8293 |
| **34** | 0,6308 | 0,9936 | 0,9938 | 1,0000 | 1,0357 | 1,0378 | 2,6219 | 2,7861 | 2,7930 | 2,8202 |
| **35** | 0,6107 | 0,9622 | 0,9625 | 0,9655 | 1,0000 | 1,0050 | 2,5379 | 2,6261 | 2,6326 | 2,6581 |
| **36** | 0,5998 | 0,9603 | 0,9605 | 0,9636 | 0,9950 | 1,0000 | 2,5328 | 2,6208 | 2,6272 | 2,6527 |
| **37** | 0,2305 | 0,3801 | 0,3802 | 0,3814 | 0,3940 | 0,3948 | 1,0000 | 1,0373 | 1,0399 | 1,0498 |
| **38** | 0,2149 | 0,3577 | 0,3578 | 0,3589 | 0,3808 | 0,3816 | 0,9640 | 1,0000 | 1,0054 | 1,0150 |
| **39** | 0,2144 | 0,3568 | 0,3569 | 0,3580 | 0,3799 | 0,3806 | 0,9617 | 0,9946 | 1,0000 | 1,0126 |
| **40** | 0,2123 | 0,3534 | 0,3534 | 0,3546 | 0,3762 | 0,3770 | 0,9526 | 0,9852 | 0,9876 | 1,0000 |

Для восстановления весов альтернатив по плохо согласованной матрице парных сравнений воспользуемся следующими модификациями моделей 2 из [3] и модели из [4]:

Модифицированная модель 2 [3]

Модель представляет собой следующую задачу линейного программирования:

$$\min \sum \sum_{(ij)\in|A|} r_{ij} y_{ij} \tag{1}$$

$$\begin{gathered} -y_{ij} \le \omega_i - \gamma_{ij}^* \omega_j \le y_{ij}, \quad y_{ij} \ge 0 \\ {\scriptstyle (i,j)\in|A|} \\ a \le \omega_i, \quad i = \overline{1,n}, \end{gathered} \tag{2}$$

где $\alpha$ – заданное положительное число, $\alpha \ge 1$; $\omega_i, i = \overline{1,n}, \quad y_{ij}, \forall (i,j) \in A$ – переменные задачи линейного программирования; $r_{ij}$ – заданные весовые коэффициенты.

Модифицированная модель [4]

Представляет собой последовательность задач линейного программирования:

$$\min_{\substack{\Delta_{ij}^1, \Delta_{ij}^2 \\ \forall (ij)\in|A|}} \Big( \sum_{(ij)\in|A|} \sum r_{ij} \Delta_{ij}^1 - C_{lm} \sum_{(ij)\in|A|} \sum r_{ij} \Delta_{ij}^2 \Big) \tag{3}$$

$$\begin{gathered} \ln \gamma_{ij}^* + \Delta_{ij}^2 \le W_i - W_j \le \ln \gamma_{ij}^* + \Delta_{ij}^1, \\ 0 \le \Delta_{ij}^1 \le \ln(1 + l\Delta_1(l)), \quad 0 \ge \Delta_{ij}^2 \ge \ln(1 - m\Delta_2(m)), \\ W_i \ge 0, \quad i = \overline{1,n}, \end{gathered} \tag{4}$$

где $W_{ij}, \Delta^1_{ij}, \Delta^2_{ij}$ – переменные задачи линейного программирования; $l,m$ – натуральные числа, последовательно принимающие значения (1; 1), (1; 2), (2; 1), (2; 2) и т. д.; $\Delta_1(x)$, $\Delta_2(x)$ – заданные числовые скалярные функции натурального аргумента, принимающие неотрицательные значения; $C_{lm}$ – коэффициент, определяющийся из отношения:

$$\ln(1 + l \cdot \Delta_1(l)) = C_{lm} \ln \frac{1}{\left(1 - m \cdot \Delta_2(m)\right)};$$

$r_{ij}$ – заданные весовые коэффициенты.

При использовании задач (3)-(4) для поиска весов функции $\Delta_1(x)$, $\Delta_2(x)$ задаются таким образом, что значения $l\cdot\Delta_1(l)$ и $m\cdot\Delta_2(m)$ на каждой итерации (при каждой попытке решения задачи линейного программирования (3)-(4)) возрастают в небольшом соотношении к их предыдущему значению, а на первой итерации принимают минимально возможные значения. Итерации прекращаются при первом успешном решении задачи линейного программирования (3)-(4). После этого веса $\omega_i^*, i=\overline{1,n}$ объектов находятся из соотношения $\omega_i^* = e^{W_i^*}, i=\overline{1,n}$.

В функционалах (1) и (3) были добавлены весовые коэффициенты $r_{ij}$, значения которых задавались согласно таблице 7. Такой способ задания коэффициентов $r_{ij}$ соответствует особенностям эмпирической матрицы парных сравнений, элементы $\gamma_{ij}^*$ которой тем больше «зашумлены», чем больше значение $|\gamma_{ij}-1|$

($\gamma_{ij}$ – идеальное значение эмпирического коэффициента). Набор значений $r_{ij}$ в таблице не является единственно возможным и оптимальным для данного примера, однако, как будет показано далее, позволяет получить решение намного эффективнее, чем решение, полученное с помощью классического метода. Поиск эффективного варианта задания весовых коэффициентов $r_{ij}$ является следующей задачей статистического моделирования: необходимо для максимально широких допусков изменения параметров, определяющих статистический закон (статистические законы) искажения идеальных значений коэффициентов $\gamma_{ij}$ как функции параметра $|\gamma_{ij}-1|$, с помощью статистического моделирования определить усредненные значения весов $r_{ij}$ как функцию параметра $|\gamma_{ij}^*-1|$ ( не нарушая общности, можно работать только с $\gamma_{ij}^*\geq 1$; $\gamma_{ij}^*$ – элемент эмпирической матрицы парных сравнений, то есть искаженное значение коэффициента $\gamma_{ij}$). При этом необходимо предусмотреть методику опроса экспертов, позволяющую определить для каждого конкретного случая соответствующий диапазон изменения определенных выше параметров.

***Табл. 7. Значения весовых коэффициентов $r_{ij}$ в зависимости от $\gamma_{ij}^*$***

| **Интервал принадлежности $\gamma_{ij}^*$** | **Значение $r_{ij}$** |
|---|---|
| 1 | 1000,000 |
| (1;1,1] | 250,000 |
| (1,1;1,3] | 100,000 |
| (1,3;1,6] | 52,632 |
| (1,6;2] | 32,258 |
| (2;2,5] | 21,739 |
| (2,5;3,1] | 15,625 |
| (3,1;3,8] | 11,765 |
| (3,8;4,6] | 9,174 |
| (4,6;5,5] | 7,353 |
| (5,5;6,5] | 6,024 |
| (6,5;7,6] | 5,025 |
| (7,6;8,8] | 4,255 |
| (8,8;10,1] | 3,650 |
| (10,1;∞) | 3,165 |

Веса, найденные с помощью модифицированной модели 2 [3] (2)-(3), приведены в таблице 8. Угловое расстояние $d$ для найденного набора составляет 0,03395.

***Табл. 8. Веса альтернатив, найденные с помощью модифицированной модели 2 [3]***

| № | Вес альтернативы | № | Вес альтернативы | № | Вес альтернативы | № | Вес альтернативы |
|---|---|---|---|---|---|---|---|
| 1 | 0,03432438 | 11 | 0,02540454 | 21 | 0,01362760 | 31 | 0,00645089 |
| 2 | 0,03423995 | 12 | 0,01893249 | 22 | 0,01358675 | 32 | 0,00390892 |
| 3 | 0,03410050 | 13 | 0,01892862 | 23 | 0,01031444 | 33 | 0,00389652 |
| 4 | 0,03397955 | 14 | 0,01885955 | 24 | 0,01027286 | 34 | 0,00388402 |
| 5 | 0,03397747 | 15 | 0,01883784 | 25 | 0,01019712 | 35 | 0,00375023 |
| 6 | 0,02568419 | 16 | 0,01876416 | 26 | 0,01011917 | 36 | 0,00374254 |
| 7 | 0,02563750 | 17 | 0,01870522 | 27 | 0,00660095 | 37 | 0,00128182 |
| 8 | 0,02555155 | 18 | 0,01869276 | 28 | 0,00656583 | 38 | 0,00123567 |
| 9 | 0,02545385 | 19 | 0,01372079 | 29 | 0,00652721 | 39 | 0,00123567 |
| 10 | 0,02544574 | 20 | 0,01369876 | 30 | 0,00650858 | 40 | 0,00123567 |

Веса, найденные с помощью модифицированной модели из [4], приведены в таблице 9 (имеется в виду первое полученное оптимальное решение задачи линейного программирования (3)-(4)). Угловое расстояние $d$ для найденного набора составляет 0,02941.

***Табл. 9. Веса альтернатив, найденные с помощью модифицированной модели [4]***

| № | Вес альтернативы | № | Вес альтернативы | № | Вес альтернативы | № | Вес альтернативы |
|---|---|---|---|---|---|---|---|
| 1 | 0,05615910 | 11 | 0,04161281 | 21 | 0,02232223 | 31 | 0,01083834 |
| 2 | 0,05602195 | 12 | 0,03101129 | 22 | 0,02226921 | 32 | 0,00676883 |
| 3 | 0,05585637 | 13 | 0,03100534 | 23 | 0,01692297 | 33 | 0,00676735 |
| 4 | 0,05559492 | 14 | 0,03089221 | 24 | 0,01684726 | 34 | 0,00674554 |
| 5 | 0,05559250 | 15 | 0,03084095 | 25 | 0,01673181 | 35 | 0,00651328 |
| 6 | 0,04206997 | 16 | 0,03073600 | 26 | 0,01660359 | 36 | 0,00650016 |
| 7 | 0,04199362 | 17 | 0,03063946 | 27 | 0,01108935 | 37 | 0,00242244 |
| 8 | 0,04180634 | 18 | 0,03061922 | 28 | 0,01103145 | 38 | 0,00234223 |
| 9 | 0,04169403 | 19 | 0,02248891 | 29 | 0,01095936 | 39 | 0,00232957 |
| 10 | 0,04167940 | 20 | 0,02243891 | 30 | 0,01093418 | 40 | 0,00230753 |

Веса, найденные с помощью классического метода (поиск весов как элементов собственного вектора матрицы парных сравнений, отвечающего максимальному собственному числу), приведены в таблице 10. Угловое расстояние $d$ для найденного набора составляет 0,04390.

***Табл. 10. Веса альтернатив, найденные с помощью классического метода***

| № | Вес альтернативы | № | Вес альтернативы | № | Вес альтернативы | № | Вес альтернативы |
|---|---|---|---|---|---|---|---|
| 1 | 0,05726581 | 11 | 0,04208118 | 21 | 0,02204579 | 31 | 0,01033667 |
| 2 | 0,05705924 | 12 | 0,03109608 | 22 | 0,02199056 | 32 | 0,00627693 |
| 3 | 0,05695368 | 13 | 0,03108544 | 23 | 0,01652470 | 33 | 0,00627460 |
| 4 | 0,05675853 | 14 | 0,03096720 | 24 | 0,01644686 | 34 | 0,00625348 |
| 5 | 0,05674753 | 15 | 0,03094626 | 25 | 0,01636132 | 35 | 0,00597320 |
| 6 | 0,04247502 | 16 | 0,03089083 | 26 | 0,01623327 | 36 | 0,00595771 |
| 7 | 0,04236842 | 17 | 0,03078902 | 27 | 0,01057770 | 37 | 0,00214060 |
| 8 | 0,04222084 | 18 | 0,03076389 | 28 | 0,01052014 | 38 | 0,00203433 |
| 9 | 0,04217536 | 19 | 0,02215886 | 29 | 0,01048032 | 39 | 0,00202903 |
| 10 | 0,04215562 | 20 | 0,02212002 | 30 | 0,01045470 | 40 | 0,00200928 |

Таким образом, с помощью модифицированных моделей 2 из [3] и модифицированной модели из [4] нашли веса более эффективные по мере $d$, чем веса, найденные с помощью классического метода. Во всех найденных наборах победитель определен корректно – альтернатива под номером 1.

Статистические исследования показали рост эффективности данных методов по сравнению с классическим при увеличении количества альтернатив (полученное значение меры $d$ при использовании этих моделей в среднем в 3-4 раза меньше, чем при использовании классического метода при количестве альтернатив 70). Следует отметить, что во всех проведенных экспериментах при одинаковом способе задания коэффициентов $r_{ij}$ модифицированная модель из [4] давала более точное решение по мере $d$, чем модифицированная модель 2 из [3], но вследствие сложности модели из [4] на нахождение весов с её помощью времени требовалось в несколько раз больше, чем при использовании модели 2 из [3]. Можно сделать вывод, что использование в этом случае на практике модели из [4] вместо модели 2 из [3] обоснованно только тогда, когда точности восстановленных с помощью модели 2 из [3] весов не достаточно.

### 1.2 ММАИ (вторая версия).

Искажение $\gamma_{ij}$ зависит от двух факторов: размерности эмпирической матрицы парных сравнений и величины значения $\gamma_{ij}$ –1. Такое сочетание факторов искажения может иметь место, когда эксперт (эксперты) поэтапно блоками размерностью от 7×7 до 9×9 заполняют эмпирическую матрицу парных сравнений и ошибаются тем больше, чем из большего числа альтернатив выбираются произвольные 7÷9 альтернатив для парных сравнений.

ММАИ (вторая версия) реализуется в несколько этапов.

*Этап 1.*

По исходной эмпирической матрице парных сравнений с использованием моделей оптимизации, определенных при описании ММАИ (третья версия), находятся оценки весов $\hat{\omega}_i$ альтернатив $A_i, i=\overline{1,m}$.

Альтернативы упорядочиваются в соответствии с убыванием значений $\hat{\omega}_i$, $i=\overline{1,m}$.

*Этап 2.*

Выбираются наилучшие $m_2$ ($m_2 \leq 15$) альтернативы и для них эксперту (экспертам) предлагается заново построить эмпирическую матрицу парных сравнений.

*Примечание.* В известной авторам литературе число 15 – максимальное встречающееся число альтернатив, которое позволяет корректно решить практическую задачу выбора наилучшей альтернативы с помощью классического МАИ.

Далее для вновь построенной матрицы парных сравнений повторяется Этап 1.

*Этап 3.*

Для $m_3$ ($m_3 \leq 7 \div 9$) лучших альтернатив, определенных на Этапе 2 вновь экспертом (экспертами) определяется матрица парных сравнений. По этой матрице с помощью моделей оптимизации, определенных при описании ММАИ (третья версия), оцениваются веса альтернатив. Лучшей является альтернатива с наибольшим весом.

Задача статистически значимо решена правильно, если на этапе 3 эмпирическая матрица парных сравнений оказалась хорошо согласованной.

Для эффективного применения ММАИ второй версии должны быть решены задачи, аналогичные задачам 1, 2 ММАИ третьей версии.

*Примечание.* При использовании модифицированных моделей оптимизации (в частности 2 из [3] и модели из [4]) оценки весов по угловой мере могут быть допустимо близки к идеальным и при плохо обусловленной эмпирической матрице парных сравнений, когда число $\lambda^* - m$ является больше допустимого (m×m – размерность эмпирической матрицы парных сравнений, а $\lambda^*$ ее максимальное положительное собственное число). Тогда критерием произвольного решения задачи может быть статистический модифицированный критерий $M_1$ [3], в котором каждое слагаемое, определяемое $\gamma_{ij}^* \geq 1$ будет умножаться не на постоянный коэффициент, а на коэффициент зависящий от величины $\gamma_{ij}^* - 1$ (чем меньше $\gamma_{ij}^* - 1$, тем он больше).

Верхние границы допустимого значения модифицированного критерия находятся в результате статистического моделирования при заданных параметрических вероятностных законах искажения идеальных значений $\gamma_{ij}$. Результаты моделирования должны быть заданы в виде соответствующих таблиц, где верхние границы допустимого значения модифицированного критерия $M_1$ определяются параметрическим вероятностным законом, значения параметров которого находятся в заданных интервалах. Для эффективного использования этих таблиц (аналога определения хорошо согласованных эмпирических матриц парных сравнений) необходимо экспертным путем определить тип параметрического вероятностного распределения и верхние границы значений его параметров, определяющих его наиболее «жесткий» вариант.

## 2. Модифицированный метод анализа иерархий (общий случай)

Рассмотрим дерево иерархий.

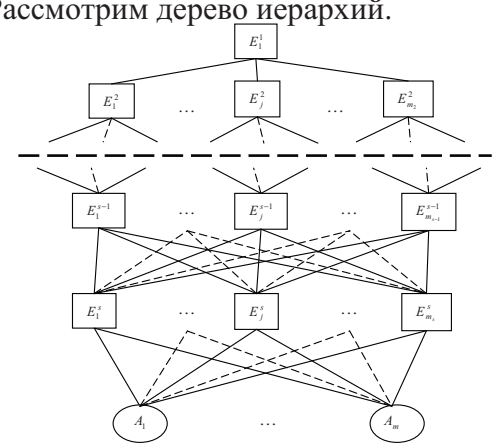



***Рис. 2. Дерево иерархии Саати***

Построение дерева иерархий [6], нахождение всех весов в виде $\omega_{E_t^{s-j}}^{E_e^{s-j+1}}$ реализуется в соответствии с общепризнанными требованиями к методу анализа иерархий [7-9]. Модификация метода относится к алгоритмам нахождения весов $E_j^s(A_i) = \omega_{E_j^s}^j, i = \overline{1,m}, j = \overline{1,m}$, для случая, когда число альтернатив достаточно велико. Рассмотрим адаптацию версий ММАИ, веденных для двухуровневой структуры.

В этом случае на нижнем уровне имеем $m_s$ эмпирических матриц парных сравнений, каждая из которых соответствует одному из критериев нижнего уровня $E_j^s, j = \overline{1,m_s}$. По каждой $j$-той ($j = \overline{1,m_s}$) эмпирической матрице парных сравнений необходимо найти веса $E_j^s(A_i), i = \overline{1,m}$, которые используются для нахождения результирующих весов $\omega_1^1(A_i), i = \overline{1,m}$, максимальный из которых определяет наилучшую альтернативу.

Обобщение всех версий ММАИ на общий случай является очевидным: в каждой версии модели оптимизации строятся не по од-

ной эмпирической матрице парных сравнений, а по каждой из эмпирических матриц парных сравнений.

При этом необходимо учесть следующее:

1. Для всех модификаций ММАИ на каждом этапе должна быть проведена соответствующая нормировка оценок весов по каждой эмпирической матрице парных сравнений.
2. Статистическая эффективность ММАИ всех модификаций для общего случая в целом ниже, чем для двухуровневой структуры, поскольку сказывается эффект наложения ошибок определения весов по всем эмпирическим матрицам парных сравнений.
3. ММАИ (вторая версия), эффективно решает задачу, если единая наилучшая альтернатива $A_i$ при реализации второго этапа оказалась в числе первых $m_3$ альтернатив, а на третьем этапе дерево иерархии оказалось интегрально хорошо согласованным (с эмпирическими матрицами парных сравнений размерности $m_3 \times m_3$ на $s$-том уровне иерархии).
4. Должна быть создана методика экспертного опроса, позволяющая обоснованно определить закон (законы) искажения эмпирических $\gamma_{ij}$ (элементов эмпирической матрицы парных сравнений), что позволит эффективно использовать ММАИ соответствующих модификаций.
5. Формальная модель каждой эмпирической матрицы парных сравнений последнего уровня иерархии с точностью до задания параметрического вероятностного распределения искажения ее элементов является одной и той же.

## Выводы

В данной статье были предложены модификации МАИ для двух случаев:

1) Искажение идеальных значений $\gamma_{ij}$ элементов эмпирической матрицы парных сравнений $\gamma_{ij}^*$ не зависит от ее размеров. Искажение $\gamma_{ij}^*$ истинного значения $\gamma_{ij}$ ($\forall \gamma_{ij} \geq 1, i \neq j$) тем больше, чем больше $\gamma_{ij}$ отличается от единицы.

2) Искажение $\gamma_{ij}$ зависит от двух факторов: размерности эмпирической матрицы парных сравнений и величины значения $\gamma_{ij} - 1$.

Модификации МАИ приведены для двухуровневой иерархии, затем полученные результаты адаптированы для общего случая.

В статье приведены результаты статистических исследований, доказывающих эффективность данных модификаций по сравнению с классическим МАИ.

## Список литературы


1. Павлов А.А., Кут В.И. Математические модели оптимизации для обоснования и нахождения весов объектов по неоднородным матрицам парных сравнений. Системні дослідження та інформаційні технології. №3, 2007р.
2. Павлов А.А., Кут В.И., Штанькевич А.С. Нахождение весов по матрице парных сравнений с односторонними ограничениями. Вісник НТУУ „КПІ” Інформатика, управління та обчислювальна техніка, Київ 2008р. №48.-122 с. – С.29-32.
3. Павлов А.А., Лищук Е.И., Кут В.И. Математические модели оптимизации для обоснования и нахождения весов в методе парных сравнений” Системні дослідження та інформаційні технології. №2, 2007р.
4. Павлов А.А., Лищук Е.И., Кут В.Н. Многокритериальный выбор в задаче обработки данных матрицы парных сравнений. Вісник НТУУ „КПІ” Інформатика, управління та обчислювальна техніка, Київ 2007р. №46.
5. Згуровский М.З., Павлов А.А., Штанкевич А.С. Модифицированный метод анализа иерархий. Інформаційні дослідження та інформаційні технології. №2. 2010 р. (принято к печати).
6. Андрейчиков А.В., Андрейчикова О.Н. Анализ, синтез, планирование решений в экономике.-Москва: Финансы и статистика.-2001.
7. Саати Т. Принятие решений. Метод анализа иерархий: Tomas Saaty. The Analytic Hierarchy Process. –Пер. с англ. Р.Г. Вачнадзе. – М.: Радио и связь, 1993. – 315 с.
8. Саати Т., Кернс К. Аналитическое планирование. Организация систем: Пер. с англ. Р.Г. Вачнадзе: Под ред. И.А. Ушакова. – М.: Радио и связь, 1991. – 223 с.
9. Saaty T.L. Multicriteria Detcision Making. The Analytik Hierarchy Process,.-New York:McGraw Hill International,1990.p.437.


Поступила в редакцию 30.11.2009

УДК 683.03

*КУЛАКОВ Ю.А.,*
*РУСАНОВА О.В.,*
*ШЕВЕЛО А.П.*

# ИЕРАРХИЧЕСКИЙ СПОСОБ ПЛАНИРОВАНИЯ ДЛЯ GRID

Статья посвящена повышению пользовательской производительности Grid систем за счет более эффективного способа планирования вычислений. Предложен усовершенствованный иерархический способ планирования, ориентированный на один из трех критериев оптимизации по выбору пользователя. Эффективность данного способа обеспечивается за счет разработанных авторами списочных алгоритмов для однородных и неоднородных узлов Grid систем, а также предложенных способов трансформации графов задач.

Paper is devoted to increasing of Grid systems real performance by effective scheduling method using. We propose an improved hierarchical scheduling method that uses one of the three optimization criterion determined by user. This method efficiency provides by creation of the new list scheduling algorithms for homogeneous and heterogeneous Grid system nodes and creation of task graph transformation.

## Введение

В настоящее время современные Grid системы все чаще используются при решении крупномасштабных проблем в науке, технике и бизнесе. Для обеспечения практической применимости Grid систем должна быть решена проблема обеспечения качества обслуживания. Качество обслуживания многоаспектное понятие, включающее: безопасность, надежность и доступность ресурсов; а также гибкость политики распределения ресурсов, которая непосредственно влияет на приемлемое и предсказуемое время ответа. Таким образом, для достижения высокого качества обслуживания и высокой пользовательской производительности Grid систем существенно важной является эффективная система планирования вычислений. К сожалению, способы планирования в традиционных параллельных и кластерных системах, не всегда могут быть использованы в Grid системах. Планирование в этих системах значительно сложнее из-за уникальных особенностей Grid систем [1-3]. Задача планирования в данных системах представляет интерес как самостоятельное исследовательское направление, которому посвящено достаточно большое количество публикаций.

В то же время, известно, что задачи планирования относятся к классу NP-полных и точного решения в общем случае не имеют. Это и определяет целесообразность разработки и исследования новых эффективных подходов к вопросам планирования в Grid системах.

Данная работа посвящена проблеме планирования в вычислительных интер Grid сетях (HTC) [1], которые состоят из множества суперкомпьютерных центров (кластеров и параллельных компьютерных систем различной сложности), рассредоточенных по всему миру и объединенных посредством Интернета.

## Основные способы планирования в Grid системах и их анализ

В настоящее время существует три основных способа планирования в Grid системе: *централизованный, децентрализованный* и *иерархический* [4].

При *централизованном* способе все пользовательские приложения посылаются централизованному планировщику. Существует единая очередь в централизованном планировщике для поступающих приложений. Как правило, локальный узел не имеет своей очереди, и не выполняет функцию планирования. Узел только получает задачи от центрального планировщика и выполняет их. Преимуществом такого способа является достаточно высокая эффективность планирования, связанная с тем, что планировщик в данном случае имеет информацию о всех доступных ресурсов и поступающих приложений. С другой стороны, централизованный планировщик может оказаться узким местом с точки зрения как надежности, так и производительности. Таким образом, централизованный способ планирования пригоден

лишь для Grid систем с ограниченным числом узлов.

При *децентрализованном* способе, функция планирования выполняется на каждом узле системы. В этом случае любой узел Grid системы работает и как планировщик, и как вычислительный ресурс. Пользовательские приложения передаются локальному планировщику Grid системы, где выполняются приложения. Локальный планировщик отвечает за планирование локальных приложений и обслуживает локальную очередь собственных приложений. Кроме того, он должен быть в состоянии ответить на запросы внешних приложений, принимая или отклоняя их. Так как ответственность планирования распределена, отказ отдельного планировщика не затрагивает работу других. Таким образом, децентрализованный способ обеспечивает лучшую отказоустойчивость и надежность, по сравнению с централизованным. Но нехватка глобального планировщика, который знает информацию обо всех приложениях и ресурсах, обычно приводит к низкой эффективности. Данный способ может быть применим для Grid систем с неограниченной масштабируемостью, однако эффективность планирования может быть достаточно низкой.

При *иерархическом* способе процесс планирования заданий распределен на двух уровнях: глобальном и локальном. На глобальном уровне управление заданиями осуществляет метапланировщик Grid, а на локальном – менеджер ресурсов. В отличие от централизованных метапланировщики Grid системы обычно не могут непосредственно управлять ресурсами сети, но работают как брокеры или агенты [5]. Информация о состоянии доступных ресурсов очень важна для метапланировщика Grid системы для того, чтобы выполнить эффективное планирование, особенно с учетом неоднородной и динамичной природы Grid сети. Роль информационной службы Grid системы (Grid information service (GIS)) – обеспечить такой информацией планировщиков Grid. GIS ответственна за сбор и предсказание информации о состоянии ресурса, такую как производительность процессора (процессоров) узлов, размер памяти, пропускная способность сети, готовность программного обеспечения и загрузка узла в определенный период. GIS отвечает на запросы для получения информации о ресурсе или передает информацию пользователям. Система мониторинга и контроля Globus (The Globus Monitoring and Discovery System (MDS)) [6] – пример GIS.

Помимо информации о ресурсе от GIS, свойства приложений (например, приблизительное количество команд, требования к памяти и хранению, зависимость подзадач в задании и объемы коммуникации) и производительность ресурса для различных видов приложений также необходимы для выполнения эффективного планирования. Получение указанных свойств может быть обеспечено компонентой профилирования приложения (Application profiling (AP)), а измерение производительности ресурса для данного типа задания – компонентой тестирования (analogical benchmarking (AB)) [7], [8]. На основе информации, полученной от AP и AB, а также используемой *модели производительности* [3] производится оценка стоимости планирования узлов-кандидатов на выполнение приложения, из которых планировщик выбирает оптимальный в соответствии с целевой функцией.

Модуль запуска и контроля (Launching and Monitoring LM) (также известный как "компоновщик" [9]) осуществляют передачу приложения на выбранные ресурсы, пересылая входные данные и исполняемые файлы, а в случае необходимости и контролирет выполнение приложений. LM – Globus GRAM (Grid Resource Allocation and Management) – пример такого модуля [10 ].

Локальный менеджер Ресурсов главным образом отвечает за выполнение внешних (полученных от метапланировщика) и локальных заданий, а также передает сообщение для GIS информациею о состоянии ресурсов. В пределах домена один или множество местных планировщиков работают с конкретной локальной политикой управления ресурсами. Примеры таких локальных планировщиков включают OpenPBS [11] и Condor [12]. Для сбора информации о локальном ресурсе LRM используют такие инструментальные средства, как Network Weather Service [13], Hawkeye [12] и Ganglia [14].

Таким образом иерархический способ планирования объединяет преимущества как централизованного, так и децентрализованного способов. Он обеспечивает высокую эффективность планирования, поддерживает высокую надежность и может быть приме-

ним для Grid систем с неограниченной масштабируемостью. Его основным недостатком является сложность. В данной работе предлагается именно иерархический способ планирования, поскольку это наиболее эффективный подход для вычислительных интер Grid систем.

### Анализ основных характеристик наиболее известных Grid систем

Рассмотрим основные характеристики наиболее известных Grid систем, приведенные в табл.1.

***Табл. 1. Основные характеристики Grid систем***

| Система | Разработчик | Ресурсы | Приложения | Политика планирования | Планирование |
|---|---|---|---|---|---|
| Condor | University of Wisconsin, Madison | Grid одного домена, не разделяемы во времени | Однозадачные Приложения | Оптимизация системной производительности, максимизирующая пропускную способность | Децентрализованная организация планировщика, динамическое |
| Condor/G | University of Wisconsin, Madison | Grid множества доменов,не разделяемы во времени | Однозадачные приложения | Оптимизация системной производительности, максимизирующая пропускную способность | Децентрализованная организация планировщика, динамическое |
| AppLeS | University of California, San Diego | Grid одного домена, разделяемы во времени | Приложения в виде параллельного потока подзадач | Максимизация пользовательской производительности | Децентрализованная организация планировщика, статическое |
| Legion | University of Virginia | Grid одного домена, разделяемы во времени | Приложения в виде параллельного потока подзадач | Максимизация пользовательской производительности , минимизирующая время выполнения | Децентрализованная организация планировщика, динамическое |
| Nimrod/G | Monash University, Australia | Grid множества доменов, разделяемы во времени | Приложение формируется в реальном времени | Оптимизация экономических затрат | Децентрализованная организация планировщика, статическое |
| EGEE | Европейский проект | Grid множества доменов,не разделяемы во времени | Приложения в виде параллельного потока подзадач | Оптимизация системной производительности, максимизирующая пропускную способность | Иерархическая организация планировщика, статическое |
| RDIG | Россия | Grid множества доменов,не разделяемы во времени | Приложения в виде параллельного потока подзадач | Оптимизация системной производительности, максимизирующая пропускную способность | Иерархическая организация планировщика, статическое |
| DEISA | Европейский проект | Grid одного домена, не разделяемы во времени | Приложения в виде параллельноого потока подзадач | Максимизация пользовательской производительности , минимизирующая время выполнения | Иерархическая организация планировщика, статическое |

Анализ данной таблицы позволяет сделать следующие выводы:

- современные Grid системы используют иерархическую организацию планировщика;
- современные Grid системы чаще используют статическое планирование;
- с точки зрения политики планирования, все Grid системы ориентированы на реа-

лизацию одного из возможных вариантов критерия оптимизации;

- практические Grid системы чаще всего используются для однозадачных приложений либо приложений в виде параллельного потока подзадач.

Таким образом, повышение эффективности способов планирования для Grid систем связано с их совершенствованием на основе следующих свойств:

1. Использования множества различных критериев оптимизации.
2. Возможностью реализации для приложений в виде параллельно-последовательного потока подзадач.
3. Иерархическая организация планировщика.
4. Использование эффективных субоптимальных алгоритмов статического планирования.

Поэтому в данной работе предложен способ планирования для вычислительных Grid систем с перечисленными выше свойствами.

### Общая постановка задачи планирования

Рассмотрим прежде всего исходные данные для выполнения задачи планирования. Как было указано выше, работа посвящена повышению эффективности многопроцессорного статического планирования в вычислительных Grid системах для приложений в виде параллельно-последовательных подзадач. В этом случае приложение может быть представлено ориентированным ациклическим графом задачи (directed acycled graph – *DAG*) $G_t(T,E_t)$ [15]. В таком графе $G_t$, множество вершин $T=\{t_1,..,t_m\}$ и множество дуг $E_t$ представляют собой, соответственно, подзадачи и информационные зависимости в виде связей между ними. При этом вес вершины $t_i$ в графе задачи определяется, как $w_i$ и представляет собой вычислительную сложность $i$-й подзадачи. Информационная зависимость между $i$-й и $j$-й подзадачами задается с помощью дуги между ними в графе задачи с весом $e_{ij,}$ и означает, что подзадача $t_j$ может начать выполнение лишь тогда, когда завершила свое выполнение подзадача $t_i$ . Кроме этого, перед выполнением, подзадача $t_j$ должна получить необходимый объем данных $e_{ij}$ от своей предшественницы (подзадачи $t_j$).

Любой высокопроизводительный узел Grid системы, который является кластером либо параллельной системой, может быть представлен неориентированным графом системы $G_p(P,E_p)$. В графе $G_p$, множество вершин $P=\{p_1,...,p_n\}$ и множество ребер $E_p$ представляют соответственно процессорные элементы (либо компьютеры кластера) и топологию каналов связи между ними. При этом вес вершины $p_i$ в графе системы определяется как $s_i$ и представляет собой производительность $i$-го процессора. А вес ребра, соединяющий две вершины в графе системы, например между $i$-ой и $j$-ой, определяется как $u_{ij}$, и представляет собой канал передачи данных (линк) между процессорами $i$ и $j$, который имеет скорость (пропускную способность), равную $u_{ij}$. Для однородных узлов Grid системы значения весов вершин и весов ребер в графе имеют равные значения, а для неоднородных кластеров значения таких весов могут различаться.

Результат планирования может быть представлен с помощью диаграммы Ганта. Однако традиционная диаграмма Ганта содержит не всю информацию о пересылаемых данных с одного процессора на другой. Поэтому предлагается использовать модифицированную диаграмму Ганта [16], в которой расписание работы каждого процессора включает не только очередь исполняемых задач, но и порядок пересылаемых данных всех его каналов.

Задача планирования заключается в следующем. Пусть имеется Grid система, состоящая из $K$ узлов. Каждый $i$–й узел состоит из $Pi$ процессоров (компьютеров). Задано приложение из $m$ подзадач с помощью *DAG* графа. Известны также топологии каждого из $K$ узлов системы, описанные с помощью неориентированных графов. Требуется найти $i$–й узел Grid системы, который обеспечивает оптимизацию (максимизирует эффективность) решения заданного приложения.

В данной работе будем решать следующие задачи оптимизации планирования.

1. Необходимо найти такой $i$–й узел Grid системы, который обеспечивает минимальное общее время выполнения заданного приложения $(T_i)$. Математическая модель этой задачи может быть записана следующим образом. Найти

$$\min_{i=1,R}\{T_i\} \qquad (1)$$

$$T_i = \sum_{i=1}^{m} \sum_{l=1}^{P_i} t_{jli} * X_{jli} + S_i + \max\{Tr, Tf_i\} \qquad (2)$$

где $t_{jli}$ – время выполнения $j$–й подзадачи в $l$–м процессоре $i$–го узла Grid системы;

$S_i$ – время доставки входных данных и результатов приложения в (из) $i$–го узла Grid системы;

$Tr$ – время готовности приложения к выполнению в узлах системы;

$Tf_i$ – время освобождения $i$–го узла Grid системы для выполнения заданного приложения в эксклюзивном режиме;

$X_{jli} = 1$, если $j$–й подзадача выполняется $l$–м процессоре $i$–го узла,

$X_{jli} = 0$ в противном случае.

2. Необходимо найти $i$–й узел Grid системы с минимальной стоимостью, который обеспечит выполнение заданного приложения за ограниченное время *(Tz)*. Математическая модель этой задачи может быть записана следующим образом. Найти

$$\min_{i=1,R} \{C_i\} \qquad (3)$$

при ограничении

$$T_i \le Tz (i = 1, K) \qquad (4)$$

В соотношении (3) $C_i$ – стоимость $i$–го узла при выполнении заданного приложения. В данном случае вначале определяется подмножество узлов, время выполнения приложения в которых соответствует ограничению (4). Затем среди них необходимо найти узел, в котором приложение выполняется с минимальной стоимостью (соотношение (3)).

3. Необходимо найти $i$–й узел Grid системы с минимальной стоимостью, который обеспечит минимальное общее время выполнение заданного приложения. Математическая модель этой задачи может быть записана следующим образом. Найти узел Grid системы, удовлетворяющий соотношениям (1) и (2) при ограничении

$$C_i \le C_{\min} \qquad (5)$$

В этом случае вначале в соответствии с соотношениям (1) и (2) определяется подмножество узлов, обеспечивающих выполнение приложения за минимальное время. Затем среди них в соответствии с ограничением (5) выбирается узел с минимальной стоимостью.

## Модифицированный иерархический способ планирования

Для повышения эффективности планирования приложений в вычислительных Grid системах предлагается модификация иерархического способа за счет:

- использования двух различных типов алгоритмов планирования: для однородных и для неоднородных узлов;
- разработки новых эффективных алгоритмов для однородных и неоднородных узлов;
- разработки и применения различных способов трансформации графа задачи, которые целесообразно применять в качестве предварительного этапа при решении задачи планирования;
- использования одного из трех критериев оптимизации по выбору пользователя из рассмотренных выше.

Известно, что одной из важнейших характеристик Grid систем является неоднородность как вычислительных ресурсов, так и коммуникационных сетей. Поэтому алгоритмы планирования для традиционных однородных параллельных систем неприемлемы для Grid систем. В то же время анализ существующих вычислительных Grid систем показывает, что несмотря на то, что такие системы действительно состоят из неоднородных узлов, каждый из них состоит из однородных процессоров. Так, например, система DEISA включает десять разнородных суперкомпьютеров и суперкластеров, каждый из которых состоит из однородных процессоров. Поэтому имеет смысл использовать при планировании два типа алгоритмов – однородные и неоднородные.

Как известно, повышение эффективности выполнения приложения в параллельных узлах Grid системы непосредственно связано с минимизацией числа пересылок. С этой целью предлагается использовать четыре способа трансформации графа задачи, которые целесообразно применять в качестве предварительного этапа при решении задачи планирования в метапланировщике.

При планировании вычислений в Grid системах чаще всего применяется один из следующих критериев оптимизации – максимизация пользовательской производительности, максимизация системной производительности или оптимизация экономических затрат. В предлагаемом подходе при плани-

ровании может быть применен один из следующих трех критериев по выбору пользователя. Первый из них относится к максимизации пользовательской производительности, второй к оптимизации экономических затрат, а третий к комбинации максимизации пользовательской производительности и оптимизации экономических затрат. Таким образом, предлагается более универсальный подход, который может быть применен для различных критериев оптимизации.

Рассмотрим основные этапы предлагаемого иерархического способа планирования для вычислительных Grid систем:

1. Пользователь посылает приложение в виде графа задачи и ресурсный запрос, учитывающий один из трех критериев оптимизации, описанных выше, метадиспетчеру. Метадиспетчер имеет следующую информацию об узлах: стоимость работы каждого узла за единицу времени, а также скорость доставки данных в каждый узел.
2. Метадиспетчер производит трансформацию графа задачи, затем посылает локальным менеджерам трансформированный граф задачи и получает в ответ возможное время запуска на каждом из них. Для определения этого времени на каждом из локальных ресурсов выполняется планирование заданного приложения с использованием одного из предлагаемых в данной работе алгоритмов: для однородных и для неоднородных узлов. Кроме того локальные менеджеры сообщают возможное время освобождения своих узлов для определения начального времени выполнения приложения.
3. В зависимости от заданного пользователем критерия оптимизации и имеющихся данных, метадиспетчер выбирает оптимальный узел Grid системы и посылает туда запрос на резервирование ресурсов, причем резервирование должно быть создано таким образом, чтобы воспользоваться зарезервированными ресурсами могло только это приложение.
4. Если резервирование выполнено, приложение посылается в очередь выбранного узла. В том случае если ресурсы, необходимые для запуска задания освободятся раньше времени начала резервирования, необходимо обеспечить разрешение запуска приложения.

## Предлагаемые списочные алгоритмы планирования для однородных и неоднородных узлов Grid системы

Рассмотрим эффективные алгоритмы планирования, применяемые на однородных и неоднородных локальных узлах Grid системы.

Для однородных узлов Grid системы предлагается использовать, разработанный авторами универсальный двухпроходный списочный эвристический алгоритм (TLHSA) [17], который имеет лучшие показатели эффективности в среднем на 12-25% по сравнению с широкоизвестными аналогичными алгоритмами, такими, как – "Mapping Algorithm" и "Mapping with Duplication Algorithm", разработанными El-Rewini и T. Lewis [18]. THLSA отличается от известных тем, что при определении величин вычислительной “срочности” подзадач (вершин) принимаются во внимание временные задержки на передачу данных. При определении величины вычислительной “срочности” учитываются следующие характеристики вершин графа задачи: ранний срок выполнения; поздний срок выполнения; резервное время выполнения; вес вершины; сумма весов входящих дуг. Значение вычислительной “срочности” каждой вершины в графе задачи определяется как разность между суммой весов входящих в нее дуг и резервным временем выполнения данной подзадачи. Очередь готовых к выполнению подзадач формируется в порядке убывания значений их вычислительной “срочности”. Назначение вершины выполняется на тот процессор, для которого время начала выполнения данной подзадачи на процессоре является минимальным. При этом не имеет значения, занят ли этот процессор в данный момент времени или свободен.

Для неоднородных узлов Grid системы предлагается использовать алгоритм, который является сочетанием TLHSA и известного гетерогенного алгоритма HEFT [19]. Для формирования очереди подзадач предлагается использовать подход подобный тому, который существует в TLHSA. Отличие заключается в способе определения позднего срока выполнения вершин. В алгоритме для однородных узлов время позднего срока выполнения определяется как разность между длиной критического пути графа задачи и длиной самого длинного пути, начиная от данной вершины и заканчивая конечной с

учетом весов вершин, но без учета весов дуг. В алгоритме же для неоднородных узлов целесообразно вместо весов вершин использовать среднее время выполнения вершин на процессорах с различной производительностью неоднородного узла Grid системы.

При принятии решения о назначении подзадач приложения на процессор узла Grid системы целесообразно применить подход, аналогичный тому, который используется в алгоритме HEFT. Такой подход позволяет получить достаточно точный результат. Однако предлагается усовершенствовать модель пересылки данных между процессорами в узле Grid системы при выполнении назначения по сравнению с алгоритмом HEFT. Предлагается при пересылке данных между процессорами использовать модель коммуникационной стоимости, в соответствии с которой в каждом процессоре имеется автономный контроллера ввода-вывода. Это означает, что процессор может одновременно вычислять задачу и обмениваться данными с другим процессором. Коммуникационные задержки между подзадачами, назначенными на один и тот же процессор приравниваются к нулю. Значение же коммуникационной задержки между двумя связанными подзадачами, выполняемыми на различных процессорах является функцией, зависящей от размера сообщения (веса дуги в графе задачи), маршрута и пропускной способности канала между процессорами. Кроме того, предполагается, что пересылка любого сообщения не прерывается и любой процессор узла Grid системы в текущий момент времени может обмениваться данными только с одним процессором. В процессе моделирования было проведено сравнение показателей эффективности предлагаемого алгоритма с известным алгоритмом HEFT. Результаты сравнения средних коэффициентов ускорения представлены на рис. 1.

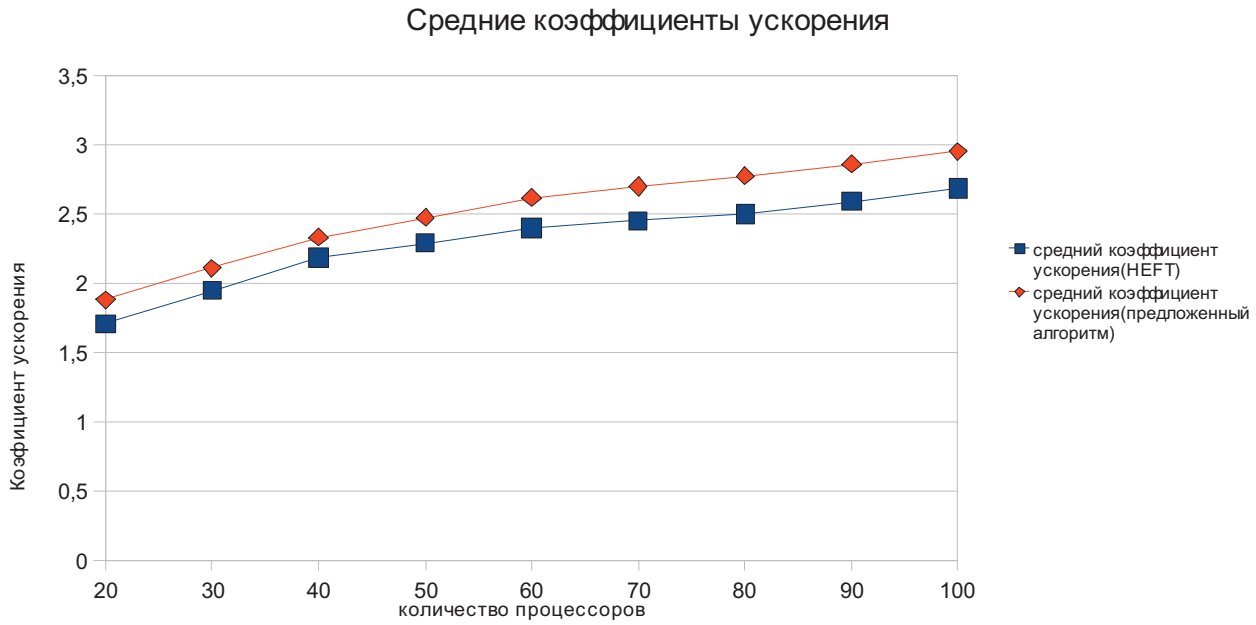



***Рис. 1.Сравнение средних значений коэффициентов ускорения при использовании предлагаемого алгоритма и HEFT***

Согласно полученным данным видно, что использовании предлагаемого алгоритма планирования для неоднородных узлов Grid системы приводит к увеличению коэффициента ускорения в среднем на 10% по сравнению с классическим алгоритмом HEFT.

## Способы трансформации графов задач

Как было указано выше, повышение эффективности выполнения приложения в параллельных узлах Grid системы непосредственно связано с разработкой и применением различных способов трансформации графа задачи, которые целесообразно применять в качестве предварительного этапа при решении задачи планирования. В работе[20] авторами данной статьи предложено следующие четыре способа трансформации графа задачи: дублирование вершины; объединение предшествующей и последующей вершин; объединение двух параллельных ветвей; объединение двух предшествующих ве-

ршин. Нами также была разработана программная модель для исследования различных алгоритмов статического планирования вычислений для Grid систем. С помощью данной модели был выполнен сравнительный анализ алгоритмов планирования без трансформации и алгоритмов с учетом пятнадцати различных сочетаний рассмотренных трансформаций графов задач. Наилучшие результаты по показателям эффективности были получены для алгоритмов с восьмым (сочетание первого и третьего способов), тринадцатым (сочетание первого, третьего и четвертого способов) и пятнадцатым (сочетание всех четырех способов) способами трансформации. Показатели эффективности этих алгоритмов в среднем на 25% выше, чем алгоритмов без трансформации графа задачи.

## Заключение

1. Проанализированы основные способы планирования в Grid системах. Определен иерархический способ планирования, как наиболее эффективный подход для вычислительных интер Grid систем.
2. Рассмотрена общая постановка задачи планирования, используемая в данной работе. Для этого в соответствии с теорией расписаний определены исходные данные, результаты планирования, а также описаны решаемые задачи оптимизации.
3. Предложен усовершенствованнный иерархический способ планирования вычислений для Grid систем, ориентированный на один из трех критериев оптимизации по выбору пользователя. Повышение эффективности предлагаемого подхода обеспечивается за счет использования двух новых эффективных алгоритмов планирования для однородных и неоднородных узлов, а также за счет разработанных способов трансформации графа задачи.

Предложенные алгоритмы доведены до практической реализации в виде программного продукта, который может быть использован при разработке иерархических планировщиков современных вычислительных Grid систем.

## Список литературы


1. Y. Zhu, A Survey on Grid Scheduling Systems, Department of Computer Science, Hong Kong University of science and Technology, 2003.
2. Technical Report No. 2006-504 Scheduling Algorithms for Grid Computing: State of the Art and Open Problems. Fangpeng Dong and Selim G. Akl. School of Computing,Queen's University Kingston, Ontario January 2006.
3. F. Berman, *High-Performance Schedulers*, chapter in The Grid: Blueprint for a Future Computing Infrastructure, Morgan Kaufmann Publishers, 1998.
4. V. Hamscher, U. Schwiegelshohn, A. Streit, R. Yahyapour, *Evaluation of Job-Scheduling Strategies for Grid Computing*, in Proc. of GRID 2000 GRID 2000, First IEEE/ACM International Workshop, pp. 191-202, Bangalore, India, December 2000.
5. F. Berman, R. Wolski, H. Casanova, W. Cirne, H. Dail, M. Faerman, S. Figueira, J.Hayes, G. Obertelli, J. Schopf, G. Shao, S. Smallen, N. Spring, A. Su and D.Zagorodnov, *Adaptive Computing on the Grid Using AppLeS*, in IEEE Trans. On Parallel and Distributed Systems (TPDS), Vol.14, No.4, pp.369-382, 2003.
6. K. Czajkowski, S. Fitzgerald, I. Foster, and C. Kesselman, *Grid Information Services for Distributed Resource Sharing*, in Proc. the 10th IEEE International Symposium on High- Performance Distributed Computing (HPDC-10), pp. 181-194, San Francisco,California, USA, August 2001.
7. A.Khokhar, V. K. Prasanna, M. E. Shaaban, and C. L. Wang, *Heterogeneous Computing: Challenges and Opportunities*, IEEE Computer, Vol. 26, No. 6, pp. 18-27,June 1993.
8. H. J. Siegel, H. G. Dietz, and J. K. Antonio, *Software Support for Heterogeneous Computing*, in ACM Computing Surveys, Vol.28, No.1, pp. 237 – 239, March 1996.
9. K. Cooper, A. Dasgupta, K. Kennedy, C. Koelbel, A. Mandal, G. Marin, M. Mazina,J. Mellor-Crummey, F. Berman, H. Casanova, A. Chien, H. Dail, X. Liu, A. Olugbile,O. Sievert, H. Xia, L. Johnsson, B. Liu, M. Patel, D. Reed, W. Deng, C. Mendes, Z.Shi, A. YarKhan and J. Dongarra, *New Grid Scheduling and Rescheduling Methods in the GrADS Project*, in Proc. of the 18th International Parallel and Distributed Processing Symposium (IPDPS'04), pp.199--206, Santa Fe, New Mexico USA, April 2004.
10. K. Czajkowski, I. Foster, N. Karonis, C. Kesselman, S. Martin, W. Smith, and S.Tuecke, *A Resource Management Architecture for Metacomputing Systems*, In D.G.Feitelson and L. Rudolph, editors, in Proc of the 4th Workshop on Job SchedulingStrategies for Parallel Processing, LNCS Vol. 1459 pp. 62–82, Orlando, Florida USA,March 1998.
11. http://www.openpbs.org
12. http://www.cs.wisc.edu/condor

13. R. Wolski, N. T. Spring and J. Hayes, *The Network Weather Service: A Distributed Resource Performance Forecasting Service for Metacomputing*, in the J. of Future Generation Computing Systems, Vol. 15, No. 5-6, pp. 757-768, January 1999.
14. F.D. Sacerdoti, M.J. Katz, M.L Massie and D.E. Culler, *Wide area cluster monitoring with Ganglia*, in Proc. of IEEE International Conference on Cluster Computing, pp.289 – 298, Hong Kong, December 2003.
15. Луцкий Г.М., Русанова О.В., "Проблемы отображения и планирования транспьютерных систем", (англ.) 1-ая Международная Конференция по Параллельной Обработке и Прикладной Математики - PRAM'94 R. Wyrzykowski and H. Piech eds., Czestochowa. 1994, pp.77-83.
16. G. Loutsky, O. Rusanova Scheduling Problems on the Parallel and Distributed Systems - an Overwiew,"Computer Systems and Networks: designing, application, utilization".- Poland, Rzeszow, 2000, tom 1, pp.101-105(Engl).
17. A.Korochkin, O.Rusanova Scheduling Problems for Parallel and Distributed Systems, ACM, SIGADA, Vol.XIX, Issue 3, New York, USA, 1999, P.195-201.
18. Блинова Т.А., Русанова О.В. Составление расписания работ для конвейерных вычислительных систем.Тез. докл. регион. семинара "Распределенная обработка информации" /Новосибирск, 1989. - С.24.
19. H. Topcuoglu, S. Hariri, M.Y. Wu, Performance-Effective and Low-Complexity Task Scheduling for Heterogeneous Computing, IEEE Transactions on Parallel and Distributed Systems, Vol. 13, No. 3, pp. 260 - 274, 2002.
20. Rusanova O.V., Shevelo A.P. List scheduling algorithm modification for MPP systems // Вісник НТУУ «КПІ».– Інформатика, управління та ОТ.-2006.-№45.-С.101-111.

Поступила в редакцию 18.12.2009

УДК 004.8

*ЛЮБЧЕНКО К. М.*

# НЕІНФОРМОВАНИЙ ПОШУК В ЕКСПЕРТНИХ СИСТЕМАХ

У статті розглянуто основні стратегії та методи неінформованого пошуку, на базі яких проектуються механізми виводу експертних систем.

In article the main strategies and methods of uninformed search on which base mechanisms of problem solving of expert systems are projected are considered.

## Вступ

Експертні системи (ЕС) є одним з типів систем штучного інтелекту. Вони виникли як значний практичний результат розвитку досліджень у галузі штучного інтелекту – сукупності наукових дисциплін, що вивчають методи розв’язування задач інтелектуального (творчого) характеру з використанням комп'ютерів.

Важливими особливостями експертних систем є можливості накопичення знань, збереження їх потенційно необмежений час, а також оновлення, що забезпечує відносну незалежність конкретного користувача від наявності в даний момент кваліфікованих фахівців. Крім того, накопичення знань дозволяє підвищувати кваліфікацію спеціалістів, що працюють в організації, використовуючи найкращі, перевірені рішення. На відміну від людини до будь-якої інформації експертні системи підходять об'єктивно, що покращує якість експертизи, що проводиться. При розв’язуванні задач, що вимагають обробки великого об'єму знань, можливість виникнення помилки при пошуку розв’язку дуже мала.

ЕС надають поради, проводять аналіз, виконують класифікацію, дають консультації і ставлять діагноз. Вони орієнтовані на розв’язування задач, що зазвичай вимагають проведення експертизи людиною-фахівцем. На відміну від машинних програм, що використовують процедурний аналіз, класичні ЕС вирішують задачі у вузькій предметній галузі (конкретній галузі експертизи) на основі дедуктивних міркувань. Такі системи часто виявляються здатними знайти розв’язок задач, які слабо структуровані і погано визначені. Вони справляються з відсутністю структурованості шляхом залучення евристик, тобто правил з досвіду експерта, що може бути корисним в тих системах, коли недолік необхідних знань або часу виключає можливість проведення повного аналізу.

Основними компонентами експертної системи є база знань, механізм виводу, система інтерфейсу користувача.

У даній статті розглянуто основи роботи механізму виводу. При його проектуванні необхідно визначитися зі стратегією та методом пошуку, на основі яких він здійснюється.

## Постановка задачі

Окремі автори і дослідники по-різному трактують зміст понять стратегії та методу пошуку. Наведемо декілька прикладів.

1. "Модуль розв’язування задачі може використовувати як стратегію пошуку на основі даних, так і на основі цілі" [1]. У даному випадку автор в якості стратегій пошуку визначає прямий та зворотній ланцюг міркувань. Однак пізніше стратегіями пошуку він називає пошук у глибину, у ширину та ітераційне заглиблення. В той же час, пошук у глибину та у ширину Дж. Люгер відносить до методів пошуку.
2. В. М. Вагін, О. Ю. Головіна, А. А. Загорянська, М. В. Фоміна фактично ототожнюють поняття стратегії і методу пошуку [2].
3. С. Рассел, П. Норвіг до стратегій пошуку відносять пошук у глибину, у ширину та з ітераційним заглибленням, а прямий і зворотній логічні виводи називають методами (алгоритмами) виводу [3].
4. Т. А. Гаврілова і В. Ф. Хорошевський по відношенню до пошуку користуються термінами "стратегія" і "метод" як синонімами, але зазначають, що "При розробці стратегії управління виводом важно визначити два питання:

1) яку точку у просторі станів прийняти за вихідну? Від вибору цієї точки залежить і

метод здійснення пошуку – у прямому або у зворотному напрямку;

2) якими методами можна підвищити ефективність пошуку розв'язку? Ці методи визначаються обраною стратегією перебору – глибину, у ширину, по підзадачам або по-іншому" [4].

Уникаючи вказаної неоднозначності у наведених публікаціях, будемо вважати:

- стратегіями пошуку – пошук у глибину, у ширину і т. п.;
- методами пошуку – пошук на основі даних (прямий), пошук від цілі (зворотній), абдукція і т. п.

Основною метою статті є аналіз та рекомендації щодо використання основних стратегій і методів неінформованого пошуку розв'язку в експертних системах.

### Виклад основного матеріалу

Розглянемо зазначені стратегії і методи пошуку з метою визначення їх взаємозв'язків і рекомендацій щодо застосування при розробці механізмів виведення експертних систем. Міркування будемо ілюструвати за допомогою графів.

Пошук у просторі станів відбувається шляхом побудови дерева пошуку у відповідності до обраної стратегії.

При пошуку у глибину після аналізу поточного стану здійснюється перехід до нового стану з наступним рівнем в ієрархії, який визначається методом пошуку. Наприклад, у діагностичній медичній експертній системі для підтвердження (або спростування) захворювання у пацієнта на кожному кроці будуть уточнюватися симптоми, що належать послідовно більш високому або низькому рівню їх ієрархії. Цей процес "занурення" буде здійснюватися доти, доки не буде досягнуто деякій стан у кінцевому рівні ієрархії. Після цього відбувається відкат до такого попереднього стану, який має альтернативу для подальшого пошуку.

Сутність пошуку у глибину зображено на рис. 1.

Стратегію пошуку у глибину найчастіше реалізують за принципом стеку або за допомогою рекурсії.

Пошук у глибину не потребує значної кількості пам'яті при збереженні тільки єдиного шляху від кореня дерева пошуку до листового вузла разом з сестринськими вузлами, які залишились нерозгорненими на поточному шляху.

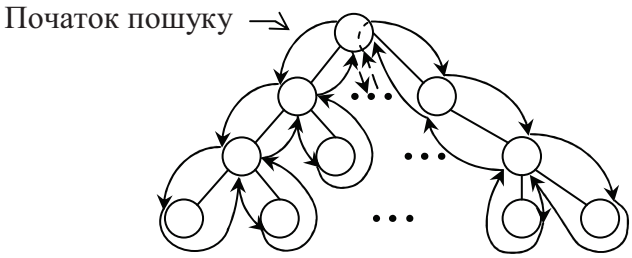



***Рис. 1. Пошук у глибину (повний обхід)***

Однак, суттєвим недоліком даної стратегії є можливість виникнення ситуації, коли пошук здійснюється по досить довгому (навіть нескінченому) шляху, адже розв'язок може бути недалеко від кореня дерева.

При пошуку у ширину аналізуються всі стани, що відносяться до поточного рівня ієрархії. Тому при даній стратегії діагностична медична експертна система спочатку буде здійснювати обробку симптомів одного рівня і, якщо не знайде задовільного рішення, перейде до наступного. Іншими словами, зазначений пошук відбувається пошарово.

Сутність пошуку у ширину зображено на рис. 2.

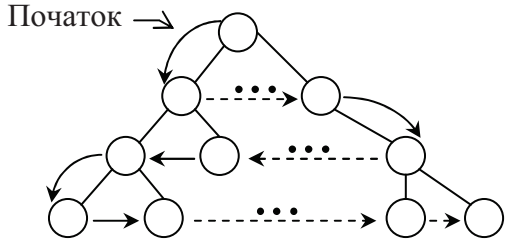



***Рис. 2. Пошук у ширину (повний обхід)***

Стратегію пошуку у ширину найчастіше реалізують за принципом черги.

На відміну від пошуку у глибину пошук у ширину потребує значної кількості пам'яті, оскільки зберігаються дані для кожного виробленого вузла. З іншого боку, дана стратегія дозволяє швидко знайти розв'язок, якщо він знаходиться недалеко від кореня дерева.

Одним з способів удосконалення пошуку у глибину є пошук у глибину з ітераційним заглибленням. Фактично він усуває деякі з зазначених проблем як пошуку у глибину, так і пошуку у ширину. Дана стратегія пошуку передбачає встановлення деякого значення глибини пошуку (кількості рівнів). При його реалізації здійснюється обхід обмеженого дерева з використанням стратегії "у глибину". Якщо розв'язок не знайдено, кількість рівнів збільшується на задане число і процес повторюється. При цьому вимоги до

пам'яті зменшуються у порівнянні з пошуком у ширину.

Отже, застосування пошуку у глибину з ітераційним заглибленням є кращою стратегією для неінформованого пошуку, якщо невідома глибина, на якій знаходиться розв'язок, а простір пошуку є досить великим.

Розробка механізму виводу для експертних систем базується на двох основних концепціях – прямому і зворотному ланцюжках міркувань. Кожна концепція визначає метод вибору в конкретних умовах.

Прямий вивід застосовується в тих ситуаціях, коли досить значне число потенційних рішень, а кількість даних, що визначають початковий стан проблеми, невелика. Прямий вивід називають виводом, що керується даними, або виводом, що керується антецедентами. Пошук починається від фактів. За допомогою правил (ходів) отримуються нові факти. Даний процес здійснюється до досягнення певної цілі.

Прямий вивід застосовується в системах планування і проектування.

Зворотній вивід доцільно використовувати у тих задачах, які мають незначну кількість розв'язків, але досить великий об'єм вхідної інформації. В цьому випадку поступово розглядається тільки один з можливих розв'язків, і механізм виводу виконує роботу по його підтвердженню або спростуванню. Зворотній вивід застосовується в системах діагностики (класифікації) і ремонту.

Одними з перших експертних систем були діагностичні, зокрема, медичні. Діагностика – це розділ медицини, що вивчає методи і принципи розпізнавання хвороб і постановки діагнозу [5]. Задачею діагностики є процес пошуку несправностей в обстежуваній системі (або визначення стадії захворювання в живій системі), заснований на інтерпретації даних, можливо зашумлених. Знаходження узгоджених і коректних інтерпретацій є основною вимогою в цієї задачі. Одна з необхідних умов досягнення результату – розуміння діагностом структурної організації обстежуваної області і механізмів взаємодії між різними підсистемами. Наприклад, завдання медичної діагностики полягає у виявленні захворювань на основі інтерпретації даних про поточний стан хворого, які виходять в результаті аналізу скарг пацієнта, його об'єктивного огляду, результатів лабораторних обстежень і аналізів. Цим визначається те, що при проектуванні механізму виводу для діагностичних, в тому числі медичних, експертних систем доцільно використовувати зворотній ланцюг міркувань, який є одним з ефективних методів виводу для даного типу задач.

На рис. 3-6 проілюстровано взаємозв'язок між основними стратегіями і методами виводу, які застосовуються при неінформованому пошуку (числами відображено порядок обходу простору станів).

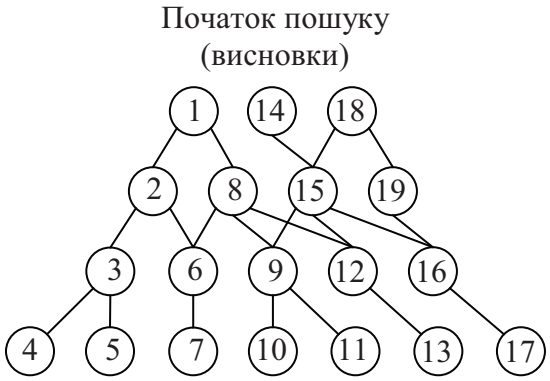



***Рис. 3. Зворотній вивід, пошук у глибину***

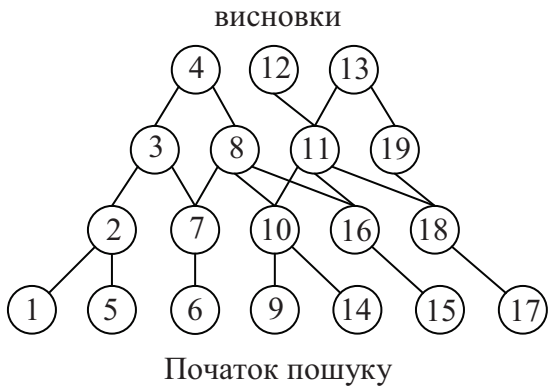



***Рис. 4. Прямий вивід, пошук у глибину***

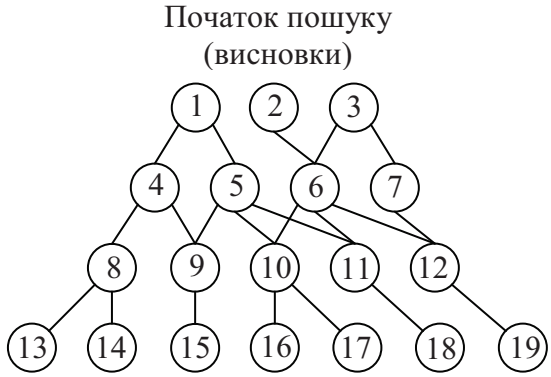



***Рис. 5. Зворотний вивід, пошук у ширину***

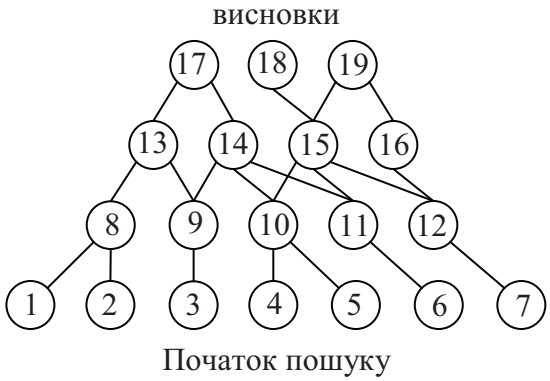



***Рис. 6. Прямий вивід, пошук у ширину***

## Висновки

Таким чином, розглянуті основні стратегії та методи неінформованого пошуку, які у різних комбінаціях і модифікаціях застосовуються в інтелектуальних, зокрема експертних, системах, мають певні переваги і недоліки – це може суттєво впливати на ефективність роботи відповідних програмних продуктів. Під ефективністю тут розуміється передусім швидкість пошуку розв'язку та об'єм пам'яті, який при цьому використовується. Тому при проектуванні механізму виводу зазначених систем необхідно враховувати їх галузь застосування, що дозволяє обрати ту чи іншу стратегію і метод виводу і, при необхідності, їх модифікацію.

## Список посилань

1. Люгер, Джордж, Ф. Искусственный интеллект: стратегии и методы решения сложных проблем, 4-е издание: Пер. с англ. – М.: Издательский дом "Вильямс", 2003. – 864 с.: ил.
2. Вагин В. Н., Головина Е. Ю., Загорянская А. А., Фомина М. В. Достоверный и правдоподобный вывод в интеллектуальных системах / Под ред. В. Н. Вагина, Д. А. Поспелова. – М.: ФИЗМАТЛИТ, 2004. – 704 с.
3. Рассел С., Норвиг П. Искусственный интеллект: современный подход, 2-е изд.: Пер. с англ. – М.: Издательский дом "Вильямс", 2006. – 1408 с.: ил.
4. Базы знаний интеллектуальных систем / Т. А. Гаврилова, В. Ф. Хорошевский – СПб: Питер, 2000. – 384 с.: ил.
5. Ефремова Т. Ф. Современный толковый словарь русского языка. В 3 томах. Том 1. А-Л. – М.: АСТ, 2006. – 1168 с.

Поступила в редакцию 15.12.2009

УДК 519.854.2

*ПАВЛОВ А.А.,*
*МИСЮРА Е.Б.,*
*МЕЛЬНИКОВ О.В.,*
*ГАНЗИНА Е.*

# РЕШЕНИЕ ЗАДАЧИ АГРЕГАЦИИ В ТРЕХУРОВНЕВОЙ МОДЕЛИ ПЛАНИРОВАНИЯ МЕЛКОСЕРИЙНОГО ПРОИЗВОДСТВА

Рассмотрен алгоритм работы блока агрегации на первом уровне модели планирования мелкосерийного производства.

The algorithm of aggregation unit on the first level of the three-levels model of small series manufactories planning is considered.

В современных условиях ключевым фактором, от которого в наибольшей степени зависит эффективность работы предприятия и полученная им прибыль, является эффективное планирование производственного процесса. Ключевой целью управления, вне зависимости от сферы применения, является оптимизация временных характеристик расписаний. С целью решения задачи оптимизации временной схемы мелкосерийного производства было разработано математическое обеспечение системы планирования и управления производством. В основу данной системы положена трехуровневая модель планирования мелкосерийного производства [1]. Согласно ей, построение производственной программы производится в три этапа: построение агрегированной модели, распределение производственной программы на плановый период и составление пооперационного плана функционирования подразделений предприятия с привязкой к оборудованию. Результаты расчета по формированию планов довольно громоздки, в то время как менеджеру необходима обобщенная информация о системе. Поэтому процесс формирования планов организован с учетом разработки агрегированной модели производства совокупности изделий, включенных в портфель заказов. Таким образом, агрегированная модель производства, полученная уже на первом этапе, содержит информацию, необходимую управленцу в принятии производственных решений.

В данной статье будет рассмотрен алгоритм работы блока агрегации в рамках первого уровня модели планирования.

## Алгоритмическое обеспечение блока агрегации

Необходимо изготовить *n* изделий (портфель заказов). Под изделием понимается серия однотипных изделий. Общая технология изготовления изделий задана в виде ацикличного ориентированного графа операций.

На первом этапе строится ацикличный технологический граф, вершинами которого являются ячейкокомплекты. Ячейкокомплект – это совокупность операций, выполняемых в одной производственной ячейке в рамках одного захода по одному изделию. Дуги интерпретируют связи между ячейкокомплектами, полученные в результате агрегации.

*Шаг 1.* Построение технологического графа. Рассмотрим пример (рис. 1).

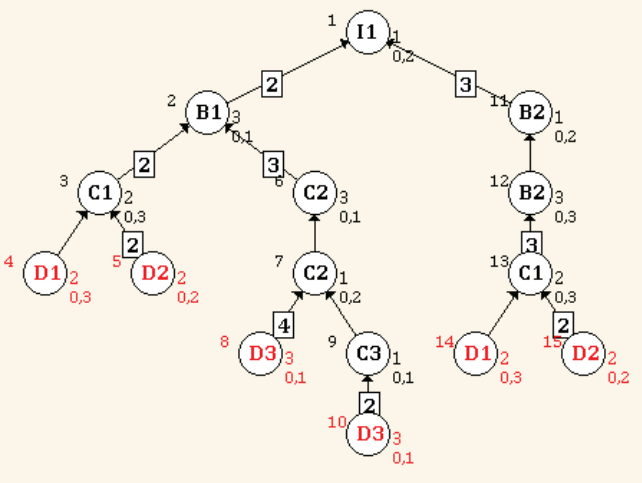



***Рис. 1. Исходный граф операций***

Каждой вершине графа соответствует часть операции, выполняемой в отдельной ячейке. Каждая операция имеет свое уникальное обозначение. Слева от вершины обозначен ее номер. Справа номер ячейки, в которой производится операция. В правом нижнем углу указано время выполнения операции

в одной производственной ячейке в рамках одного захода по одному изделию. Вес дуги соответствует количеству деталей, которые необходимо произвести на предыдущем этапе. Отсутствие обозначения веса означает, что требуется только одна деталь. В рамках одной операции обрабатывается одна и та же деталь.

*Шаг 2.* Введем ограничения, при которых решается задача:

1. Длительность изготовления каждого изделия (а также ячейкокомплекта) определяется его критическим путем;
2. Общие ячейкокомплекты разных изделий лежат на их критических путях и выполняются в одной ячейке;
3. Партия деталей не передается в другие ячейки до ее полного завершения.

Первый этап агрегации проводится, учитывая ограничение (3), таким образом. Время выполнения операции в отдельно взятой ячейке умножается на количество деталей, полученных с помощью выполнения этой операции в данной ветке графа. Таким образом, веса дуг становятся равными 1 (Рис. 2).

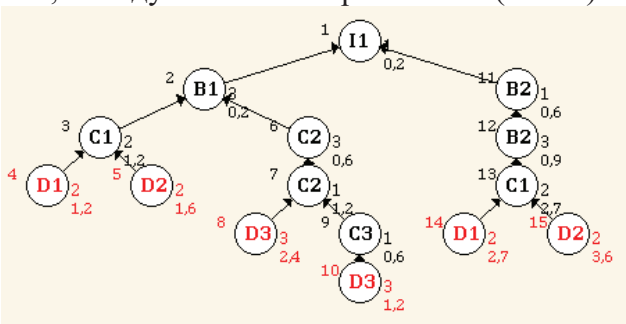



***Рис. 2. Сведение количества входящих к 1***

*Шаг 3.* Объединение смежных операций, выполняемых в одной ячейке в рамках одного захода. Если в производственном графе есть смежные операции, выполняемые в одних и тех же ячейках, то согласно ограничению (1) они могут быть объединены в один ячейкокомплект с длительностью выполнения, равной длине критического пути его операций.

*Шаг 4.* Так как в портфеле заказов указывается изготовление не одного экземпляра каждого изделия, а серии определенного размера, то этот факт тоже должен быть учтен в процессе построения агрегированной модели. Следует также уточнить, что в каждой производственной ячейке может быть несколько рабочих мест, на которых параллельно обрабатываются детали. Таким образом, время выполнения агрегированной операции по каждому ячейкокомплекту следует умножить на объем производимой серии и разделить на количество рабочих мест в ячейке. Следует также учесть тот факт, что полученный результат не должен быть меньше времени обработки одной детали одним работником в одной ячейке. Таким образом, получаем агрегированный граф ячейкокомплектов (Рис. 3).

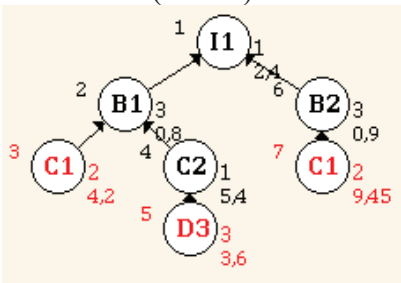



***Рис. 3. Агрегированный граф.***

На втором этапе для каждого изделия по агрегированному графу проводится поиск критического (наиболее длинного) пути, используя произвольный алгоритм поиска.

На этапе 3 некоторые ячейкокомплекты, для которых требуется общее, достаточно большое время на наладку оборудования, объединяются в общие вершины. Таким образом, система производства представляется в виде графа на критических путях изделий, что дает возможность решать задачи построения оптимального расписания ячейкокомплектов как задачи для одного станка.

Итак, нами рассмотрена схема реализации блока агрегации алгоритмического обеспечения адаптивной интегрированной системы планирования и управления производственным процессом для мелкосерийного производства.

## Список литературы

1. Павлов А.А., Теленик С.Ф. Информационные технологии и алгоритмизация в управлении. – К.: Техника.– 2002.– 344 с.

Поступила в редакцию 14.12.2009

УДК 004.633

*ЛІСНІЧЕНКО Д.І.,*
*СІМОНЕНКО А.В.*

# МЕТОД УНІФІКАЦІЇ РОБОТИ ІЗ ФАЙЛОВИМИ ІЄРАРХІЯМИ ШЛЯХОМ ЗАСТОСУВАННЯ ДЛЯ ЇХ ОБРОБКИ ПАРАМЕТРИЗОВАНИХ ОПЕРАЦІЙ НАД ГРАФАМИ

При традиційному підході до обробки ієрархій файлів, для кожної конкретної області застосування розробляється власне програмне забезпечення. У статті запропоновано метод обробки, що забезпечує більш гнучкий і універсальний підхід. Розглянуто окремі питання дизайну. Наведена ємнісна та обчислювальна складності. Намічено подальші шляхи оптимізації.

Traditional approach to file hierarchies processing requires development of specific software for every use case. In this article, it is proposed the processing method, which provides more flexible and universal approach. Some design problems are considered. Analysis of capacity and computing complexitшyi has been made. Further optimization ways are touched.

## Вступ

Оскільки найпоширенішою структурою файлових систем є ієрархічна, то її можна представити у вигляді дерева, вершинами якого є файли та директорії. В межах статті, буде використовуватись прийнятий у *nix-системах підхід, коли директорії трактуються як особливий тип файлів.

Пропонується представити файлову ієрархію у оперативній пам'яті у вигляді структури об'єктів і працювати із структурою, а не з файловою системою на диску. Позбавившись таким чином необхідності мати доступ до файлової системи, ми отримуємо можливість працювати із представленням локальних та віддалених файлових систем , (Дерево може бути отримане шляхом сканування файлової системи або може бути завантажене із попередньо згенерованого файлу опису). Потенційно, це дає нам можливість порівнювати файлові ієрархії на віддалених комп'ютерах, стани однієї й тієї ж файлової системи у різні періоди часу, виділяти змінені файли для збереження до інкрементальної резервної копії тощо.

Постає питання ефективної організації роботи із файловими ієрархіями. Можна застосувати традиційний підхід, коли для кожної моделі застосовування пишеться свій програмний комплекс, котрий реалізує жорстко закодований алгоритм: синхронізації, контролю версій, резервного копіювання. Однак такий підхід не забезпечує достатньої гнучкості і завжди існує вірогідність того, що конкретному користувачу не вистачатиме якоїсь функції і доведеться писати свій комплекс практично з нуля.

З іншого боку, оскільки файлова ієрархія представлена деревом, тобто графом, то природно було б застосовувати для роботи з нею **бінарні операції над графами**. Детальний аналіз такого методу обробки файлових ієрархій і наведений у статті.

## Операції над графами, застосовні для файлових ієрархій

Кожна операція виконується шляхом рекурсивного симетричного обходу обох графів і побудови графу-результату.

**Об'єднання $G_1$ □ $G_2$**

*У результуючому графі створюється посилання на кожен елемент з обох дерев. Однакові елементи зберігаються у одному екземплярі.*

**Перетин $G_1$ □ $G_2$**

*У результуючому дереві зберігаються тільки елементи, спільні для обох графів, причому у одному екземплярі.*

**Різниця $G_1 \setminus G_2$**

*У результуючому дереві зберігаються тільки елементи,присутні у дереві $G_1$ і відсутні у $G_2$.*

Введемо допоміжну операцію:

**Виключення $-G_1$**

*Із результуючого дерева видаляються елементи, що задовольняють певній умові.*

Комбінуючи вказані операції, можна описати операції вибору файлів для синхронізації, резервного копіювання, системи контролю версій (на рівні файлів) тощо.

Постає питання, на базі чого виконувати **порівняння елементів**. Для забезпечення достатньої гнучкості та універсальності, не достатньо використовувати тільки розмір файлу чи контрольну суму. Тому для кожної операції слід ввести параметр, з урахуванням якого вона буде здійснюватись.

### Структура запису про файл

Для однозначного представлення файла у пам'яті нам потрібно мати достатній набір характеристик файлів:

- Ім'я
- Розмір
- Файлові атрибути
- Час створення
- Час зміни
- Чи це файл, чи тека

Крім того, необхідно мати засіб, що дозволить розрізняти файли однакового розміру та з однаковими іменами, не потребуючи завантаження до оперативної пам'яті вмісту файлів. З цією метою зазвичай застосовуються контрольні суми (хеші). Суттєвий недолік лише один – теоретична ймовірність колізій. Але для використовуваних нині алгоритмів вірогідність виникнення колізій вкрай низька: наприклад, для 160-бітового SHA-1 вона оцінюється як 1 співпадіння на $2^{80}$ наборів вхідних даних [1]. Якщо навіть цього не достатньо, завжди можна ввести комбінацію двох чи більше контрольних сум, обчислених за різними алгоритмами (такий підхід використовується зокрема у протоколі Secure Sockets Layer (SSL) ). Оскільки на сучасних процесорах хеш-функції обчислюються набагато швидше, ніж йде зчитування з жорсткого диску, то суттєвого зниження швидкодії не відбудеться, зате опірність до колізій буде як мінімум не гірша за таку для найякіснішого із застосованих алгоритмів.

### Реалізація методу

Система, реалізована на базі описаного методу, працює подібно до бази даних: у систему завантажуються дані про файлові ієрархії, а потім надсилаються спеціально складені **запити**. Система виконує операції, описані у запитах, застосовуючи при цьому кешування та різноманітні оптимізації. Тобто різноманітна функціональність (наприклад, резервне копіювання або синхронізація) реалізується написанням порівняно простого скрипту без повторної компіляції програми. Причому скрипт може бути згенерований автоматично іншою програмою.

Описану вище функціональність можна також об'єднати у набір бібліотек – каркас (framework). Це дасть можливість повторного використання коду у інших проектах.

### Синтаксис запитів

Запити можуть описуватись, наприклад, таким чином:

Операції

& – об'єднання

^ – перетин

\ – різниця

- – виключення

Fx – файлова ієрархія з номером x, задається шляхом до файлової ієрархії на диску, шляхом до файлу опису або присвоєння результату виконання операцій.

# – Рядковий коментар

Приклади окремих запитів:

1. F3 = F1& F2 : hash;

Об'єднання файлових ієрархій F1 та F2 з виключенням дублікатів, що мають однаковий hash і знаходяться у паралельних рівнях ієрархії. Результат присвоюється F3.

2. F3 = -F2 : size > 100mb;

Присвоєння F3 дерева F2, окрім тих елементів, що мають розмір, більший за 100 мегабайт.

3. F3 = -F2: name = "*.tmp"

Виключити з F2 всі файли, ім'я яких закінчується на *.tmp, і присвоїти отриману файлову ієрархію F3. (застосовуються регулярні вирази).

Як і у випадку баз даних, швидкість виконання запиту у великій мірі залежить від оптимальності його складання.

Приклад скрипту

```
# Скрипт для складного інкрементного
резервного копіювання.
F1 = /media/data/;
F2 = /media/backups/bak44;
F3 = /media/backups/bak44/tmp/;
F4 = /media/backups/bak44/mail/;
#  Файли для відправлення по e-mail
F5 = /media/backups/bak44/index.fil;
F6 = /media/backups/bak43/index.fil;
#  Файл опису файлової ієрархії
# Вибираємо файли, яких не було у по-
передній резервній копії
# Це робиться з урахуванням хешу, тоб-
то вибираються і змінені файли.
F7 = F1 \ F6 : hash, name;
# Вибираємо всі тимчасові файли до
окремого дерева
F8 = -F7 : name != "*.tmp|*.bak|*.~" ;
# Вибираємо не тимчасові файли
```

```
F9 = F7 \ F8 : hash , name;
# Створюємо резервну копію даних
COPY  F9 F2 ;
# Тимчасові файли переміщуємо до іншої
резервної копії
MOVE F8 F3;
# Генерація файлу опису
LIST F5 ;
# Вибираємо особливо важливі файли,
які слід надіслати по
# e-mail для віддаленого зберігання
F10 = -F9 : name != “*-important.doc”;
COPY F10 F4;
```

Наведений вище скрипт може генеруватися із шаблону автоматично за допомогою допоміжних засобів (наприклад, у *nix-системах – це утиліта awk). При цьому іменем резервної копії може бути поточна дата тощо. Але детальний розгляд таких дій виходить за межі даної статті.

### Зауваження щодо дизайну системи

Вхідний скрипт може бути розібраний за допомогою табличного парсера. Запропонований синтаксис розроблявся з урахуванням такого використання.

Пропонується об’єкти типу File зберігати у пам’яті тільки у одному екземплярі, а у деревах зберігати тільки посилання на них (шаблон проектування flyweight [2]).

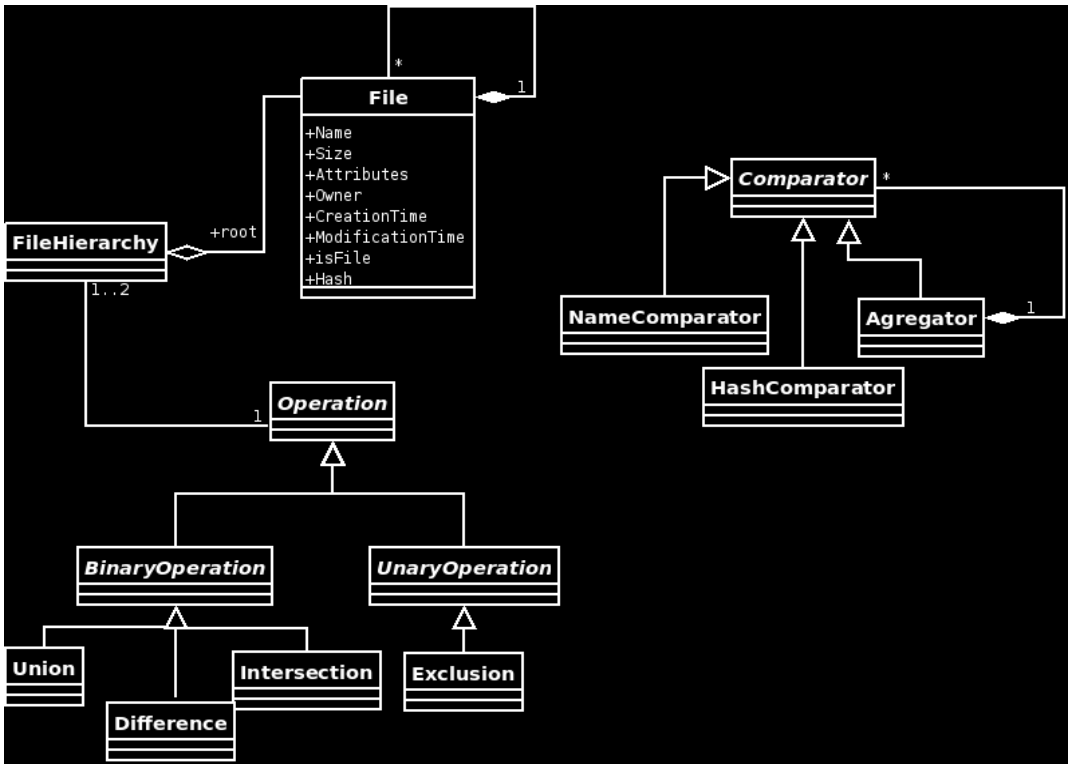



***Рис. 1 Запропонований дизайн системи***

Під час виконання операцій над деревами використовується лише операція перевірки рівності записів про файли за певним критерієм. Тому, підвищуючи рівень абстракції, можна ввести спеціальний абстрактний об’єкт-порівнювач: Comparator. Операції, що виконуються над деревами, параметризуються конкретними нащадками цього об’єкта. Таким чином, можна досягнути високого ступеня повторного використання коду і при цьому зберегти легкість розширення можливостей системи. Наприклад, щоб додати можливість роботи з файлами на основі їх власника, досить додати відповідне поле до об’єктів, що представляють файли, та оголосити нового нащадка класу Comparator. Якийсь Comparator може бути списком інших Comparator`ів, таким чином можна реалізувати обробку кількох операцій порівняння за раз (рис.1).

### Оцінка вимог до обчислювальних ресурсів

Можна обчислити обсяг пам’яті, необхідний для зберігання одного запису про файл. Зазвичай, використовуваними сьогодні операційними системами накладається обмеження на сумарну довжину шляху до файла та імені файла близько 512 байт [3]. Також, слід урахувати пам’ять, виділену під зберігання хешу (припустимо, 20 байтів), розміру файла (4 або 8 байтів), атрибутів, часу створення та модифікації, власника файлу, тощо. Сумарно на один запис про файл у навіть у граничному випадку буде достатньо 550-600 байтів. Тобто для зберігання у пам’яті мільйону записів потрібно до 600мб RAM.

Дерево вимагає ще порядку десятків мегабайт оперативної пам'яті на зберігання посилань на записи про файли, а також зв'язки між елементами (4 або 8 байтів на одне посилання для 32-х та 64-х розрядних систем відповідно).

Обчислювальна складність залежить квадратично від середньої кількості файлів у кожному каталозі (зростає кількість порівнянь) і росте лінійно пропорційно зростанню кількості файлів. Тобто обчислювальна складність поліноміальна .

Як бачимо, для обробки навіть великої файлової ієрархії достатньо можливостей персонального комп'ютера.

## Можливі оптимізації

Під час виконання бінарної операції із якимось параметром вперше, слід одночасно генерувати для поточного параметра дерево перетину та дерева різниць, якщо очікується повторна операція з цим самим параметром. Потенційне прискорення роботи від такої операції переважує накладні витрати пам'яті. Оскільки скрипт завантажується увесь одразу, то неважко передбачити, чи потрібні нам будуть ці дані.

Для операцій над файлами, можна ввести журналізацію на рівні операцій над цілим деревом. Журналізація використовується у базах даних та файлових системах типу ext3/ext4. І дозволяє у випадку раптового збою/вимкнення живлення уникнути втрати даних. Реалізується шляхом об'єднання виконуваних операцій у атомарні транзакції, кожна з яких або виконується цілком, або не виконується взагалі і реалізується за принципами **ACID** (*Atomicity, Consistency, Isolation, Durability*) – (Атомарність, узгодженість, ізоляція, стійкість). [4]

Користування програмою можна спростити, написавши графічну оболонку. У ній можна передбачити спрощений файловий менеджер для перегляду вмісту файлових ієрархій.

## Підсумок

У статті запропоновано універсальний метод обробки файлових ієрархій на противагу розробленим для конкретних потреб, використовуваним сьогодні. Він базується на представленні файлових ієрархій як графів та виконанні над ними бінарних операцій над графами за заданими параметрами. Метод не накладає обмежень на апаратне забезпечення, оскільки його коректна реалізація матиме поліноміальну обчислювальну складність та вимоги до обсягу оперативної пам'яті, адекватні кількості оброблюваних файлів. Також розглянуто окремі питання проектування, оптимізації та практичного використання описаного методу.

Реалізація запропонованого методу обробки файлових ієрархій у вигляді окремої бібліотеки або каркасу (framework) дасть можливість повторного використання коду завдяки застосуванню готових відтестованих та оптимізованих алгоритмів та спростить написання програм, що інтенсивно працюють з файловими ієрархіями.

## Список посилань

1. Bruce Schneier, Cryptanalysis of SHA-1 http: //www.schneier.com/ blog/archives/2005/02/ cryptanalysis_o.html
2. Гамма Э., Хелм Р., Джонсон Р., Влиссидес Дж. Приемы объектно-ориентированного проектирования. Паттерны проектирования. – СПб: Питер, 2001. – 368 с
3. http://en.wikipedia.org/wiki/Filename
4. Gray, Jim; and Reuter, Andreas; Distributed Transaction Processing: Concepts and Techniques, Morgan Kaufman, 1993 (ISBN 1558601902)

Поступила в редакцию 17.12.2010

УДК 656.7.052.071.13(022)

*ТЕЛЕНИК С.Ф.,*
*ГРІШИН І.Ю.*

# СПОСІБ ПІДВИЩЕННЯ ЕФЕКТИВНОСТІ ІНФОРМАЦІЙНОГО ЗАБЕЗПЕЧЕННЯ СИСТЕМИ УПРАВЛІННЯ ПОВІТРЯНИМ РУХОМ

У роботі проведений аналіз системи управління повітряним рухом як багаторівневої ієрархічної системи. Показано, що для підвищення якості оперативного управління, як найбільш складного і відповідального етапу, потрібно мінімізувати помилки оцінок параметрів траєкторій повітряних суден. Запропонований алгоритм оптимального управління інформаційними вимірювальними засобами.

The air-traffic control system analysis as multilevel hierarchical system is in-process carried out. It is shown, that for on-line control improvement of quality as the most difficult and responsible stage, it is necessary to minimise errors of parametres paths estimations of air courts. The algorithm of optimum control is offered by the informational measuring means.

## Вступ

На сьогодні галузь автоматизації процесів діяльності служби управління повітряним рухом (УПР) швидко розвивається. Серед важливих чинників впливу варто виділити необхідність дотримання вимог безпеки, досягнення регулярності польотів, економічність, висока динаміка процесів. Ефективне УПР неможливе без АСУ повітряним рухом (ПР).

В цих системах підсистема оперативного управління посідає особливе місце, оскільки тут зазначені чинники виявляються найповніше. Відомо, що можливості АСУ ПР, як людино-машинних систем, обмежуються можливостями диспетчерів. Одним із напрямів зменшення впливу зазначених обмежень є удосконалення інформаційної підтримки роботи диспетчерів, забезпечення їх не надлишковою, точною і адекватною тим задачам, які вони вирішують, інформацією. Тому проблеми підвищення ефективності інформаційного забезпечення АСУ ПР є дуже актуальними.

В статті розглядається спосіб підвищення ефективності інформаційного забезпечення АСУ ПР при управлінні потоком повітряних суден (ПС) за рахунок мінімізації помилок оцінок параметрів їх траєкторій, що отримуються за даними радіолокаційних систем (РЛС).

## Особливості управління РЛС в службах повітряного руху

Завданнями служби УПР є встановлення і підтримки порядку в повітряному русі, попередження небезпечних зближень і зіткнень ПС одне з іншим і з наземними перешкодами, забезпечення інформацією і консультаціями, необхідними для безпечного виконання польотів і повідомлення відповідних органів про ПС, що потребують порятунку, і надання їм необхідної допомоги [1].

Служба руху має трирівневу ієрархічну організаційну структуру, для підтримки процесів її діяльності створені АСУ ПР, типова функціональна структура яких наведена на рисунку 1. На вищому рівні управління вирішуються завдання організації повітряного руху і забезпечення польотів на всіх етапах руху ПС, побудови органів управління, а також взаємодії між центрами УПР різних відомств і службами забезпечення польотів. На середньому рівні вирішуються завдання довгострокового і попереднього планування повітряного руху і безпосереднього забезпечення виконання цього плану. На нижньому рівні вирішуються завдання оперативного управління повітряним рухом на всіх етапах польотів ПС.


***Рис. 1. Функціональна структура АСУ ПР***

Оперативне управління повітряним рухом призначене для забезпечення руху кожного ПС відповідно до прийнятого плану його польоту, критеріїв безпеки, регулярності і економічності, а також для забезпечення безпеки польотів сукупності ПС, тобто недопущення їх зближення до інтервалів, менших ніж задані критеріями безпеки, на всіх етапах польоту ПС шляхом контролю управління рухом. У зонах з великою щільністю повітряного руху до системи оперативного УПР висовуються жорсткі вимоги до точності дотримання визначених інтервалів проходження ПС.

Точна витримка інтервалів, яка заснована на своєчасному отриманні достовірної інформації про місцеположення ПС, є основою безпеки руху.

Управління може здійснюватися за допомогою одного з двох основних методів: за траєкторією поточного плану або з екстраполяцією за даними радіолокаційних вимірювань. Вибір методу управління залежить від того, в якій зоні виконується політ, і визначається специфікою польотів в цих зонах. З усіх видів діяльності служби руху найбільш складним є оперативне управління, оскільки, по-перше, саме тут найповніше виявляються специфічні особливості повітряного руху – динамічність процесів і неможливість їх припинення, і, по-друге, інтенсифікація процесів управління, неминуча при збільшенні щільності руху, обмежується можливостями диспетчера.

У роботах [2, 4] показано, що якість виконання диспетчером завдань оперативного управління функціонально залежить від похибок визначення параметрів траєкторій ПС радіолокаційними засобами, що можна представити у вигляді виразу

$$K_D = f\left(\sum_{i=1}^{N}\sum_{j=1}^{M} h_{ij}\sigma_{ij}^2\right) = f\left(\sum_{i=1}^{N} tr\left(\mathbf{h}_j^{\mathrm{T}}\mathbf{\Psi}_j\right)\right) \qquad (1)$$

де $N$ – кількість ПС, які обслуговуються диспетчером; $M$ – кількість параметрів трає-

кторій, необхідних для характеризації руху ПС; $\sigma_{ij}^2$ – дисперсії помилок оцінок відповідних параметрів траєкторій ПС; $tr\left(\mathbf{h}_j^{\mathrm{T}}\mathbf{\Psi}_j\right)$ – зважений слід коваріаційної матриці помилок оцінок параметрів траєкторії $j$-го ПС; $\mathrm{h}_j$ – матриця вагових коефіцієнтів для обліку важливості точності оцінок окремих параметрів траєкторії $j$ - го ПС.

Відомо [3], що функція $f(\bullet)$ найчастіше має такий вигляд

$$K_D = \exp\left[-\left(\sum_{i=1}^{N} tr\left(\mathbf{h}_j^{\mathrm{T}}\mathbf{\Psi}_j\right)\right)\right] \qquad (2)$$

Таким чином, для підвищення якості роботи диспетчерів з управління потоком ПС необхідно забезпечити мінімізацію помилок оцінок параметрів їх траєкторій, що отримуються за даними радіолокаційних вимірювань.

## Огляд існуючих рішень

У праці [5] вперше в АСУ ПР задача управління РЛС була розглянута з системних позицій, було обґрунтовано необхідність узгодження критерію якості управління РЛС з критерієм системи більш високого рівня. Для синтезу алгоритмів управління було запропоновано використовувати інформаційний підхід, за яким головним критерієм є ентропія інформації про цілі на різних етапах роботи РЛС.

Проте такий підхід, враховуючи сучасний стан теорії оптимального управління, не дозволяє отримати ефективних рішень для РЛС УПР в умовах складної повітряної обстановки. У праці [6] розглянуті підходи до управління РЛС, засновані на вирішенні поточних завдань управління. Але вони не враховують специфіку УПР і ефективність за достатньо тривалий проміжок часу.

У роботі [7] запропонований підхід до оптимізації роботи РЛС зенітних ракетних систем, проте не врахована специфіка роботи УПР.

## Постановка проблеми

Нехай для підсистеми оперативного управління АСУ ПР визначені основні методи управління. Диспетчери управляють щільним потоком ПС, що визначає суттєвий вплив можливих помилок оцінок параметрів траєкторій ПС, які одержуються за даними радіолокаційних вимірювань, на якість прийнятих диспетчерами рішень.

Необхідно розробити метод оптимального управління РЛС АСУ ПР, що враховує особливості цієї системи та властиві їй обмеження.

## Розробка методу управління РЛС АСУ ПР

Будемо дотримуватися такого плану досліджень: спочатку визначимо показник якості управління, а потім перейдемо до управління процесом вимірювання координат при вирішенні задачі оцінювання параметрів траєкторій ПС.

Розпочнемо з визначення показника якості управління. Використовуючи відомі методи координації багаторівневих ієрархічних систем, не важко довести, що показник якості управління етапу супроводження ПС, узгоджений з показником ефективності системи наступного за ієрархією рівня, має такий вигляд [8]:

$$J = \sum_{i=1}^{N}\sum_{t=1}^{T} tr\left[\mathbf{h}_i^{\mathrm{T}}(t)\mathbf{\Psi}_i(t)\right] \qquad (3)$$

Тут $tr\left[\mathbf{h}_i(t)\mathbf{\Psi}_i(t)\right] = \sum_{l=1}^{k} h_i^{ll}(t)\Psi_i^{ll}(t)$ – слід твору матриць $\mathbf{h}_i(t)$ і $\mathbf{\Psi}_i(t)$, що характеризує зважену величину похибки фільтрації координат $i$-го об'єкту. При цьому вагова матриця $\mathbf{h}_i(t)$ визначається залежно від способу супроводу і допошуку ПС в процесі супроводу, а також від способу прийняття рішення на управління повітряним рухом.

Структура матриць $\mathbf{\Psi}_i(t)$ і $\mathbf{h}_i(t)$ у разі оцінки параметрів траєкторій ПС в прямокутній системі координат, оцінок координат і їх похідних, що є взаємно некорельованими, може бути такою:

$$\mathbf{\Psi}_i(t) = \begin{pmatrix} \sigma_{x_i}^2 & 0 & 0 & 0 & 0 & 0 \\ 0 & \sigma_{\dot{x}_i}^2 & 0 & 0 & 0 & 0 \\ 0 & 0 & \sigma_{y_i}^2 & 0 & 0 & 0 \\ 0 & 0 & 0 & \sigma_{\dot{y}_i}^2 & 0 & 0 \\ 0 & 0 & 0 & 0 & \sigma_{z_i}^2 & 0 \\ 0 & 0 & 0 & 0 & 0 & \sigma_{\dot{z}_i}^2 \end{pmatrix},$$

$$\mathbf{h}_i(t)=\begin{pmatrix} h_{x_i}^2 & 0 & 0 & 0 & 0 & 0 \\ 0 & h_{\dot{x}_i}^2 & 0 & 0 & 0 & 0 \\ 0 & 0 & h_{y_i}^2 & 0 & 0 & 0 \\ 0 & 0 & 0 & h_{\dot{y}_i}^2 & 0 & 0 \\ 0 & 0 & 0 & 0 & h_{z_i}^2 & 0 \\ 0 & 0 & 0 & 0 & 0 & h_{\dot{z}_i}^2 \end{pmatrix}.$$

При цьому у разі вирішення задачі оперативного управління повітряним рухом за траєкторіями поточних планів необхідно мінімізувати поточну кульову помилку оцінки параметрів траєкторії $\sigma_{ш}^2 = h_{x_i}^2\sigma_{x_i}^2 + h_{y_i}^2\sigma_{y_i}^2 + h_{z_i}^2\sigma_{z_i}^2$, тому величини $h_{x_i}^2$, $h_{y_i}^2$, $h_{z_i}^2$ повинні приймати значення $w_i$ ($0 \le w_i \le 1$) залежно від важливості величини помилки по конкретних координатах, а величини $h_{\dot{x}_i}^2$, $h_{\dot{y}_i}^2$, $h_{\dot{z}_i}^2$ дорівнюють нулю. Якщо ж вирішується задача оперативного управління повітряним рухом за траєкторіями поточних планів з екстраполяцією, то величини $h_{\dot{x}_i}^2$, $h_{\dot{y}_i}^2$, $h_{\dot{z}_i}^2$ набувають значення пропорційно інтервалу часу екстраполяції, оскільки екстрапольоване значення координати у разі лінійної моделі руху ПС обчислюється відповідно до виразів

$$x_i(t+t_э) = x_i(t) + \dot{x}_i(t)t_э;$$

$$y_i(t+t_э) = y_i(t) + \dot{y}_i(t)t_э;$$

$$z_i(t+t_э) = z_i(t) + \dot{z}_i(t)t_э,$$

де $t_э$ – часовий інтервал екстраполяції положення ПС для прийняття рішення на управління повітряним рухом.

Величина квадрата кульової помилки в екстрапольованій точці дорівнює

$$\sigma_{шэ}^2 = \sigma_{x_{i-1}}^2 + \sigma_{\dot{x}_{i-1}}^2 t_э^2 + \sigma_{y_{i-1}}^2 + \sigma_{\dot{y}_{i-1}}^2 t_э^2 + \sigma_{z_{i-1}}^2 + \sigma_{\dot{z}_{i-1}}^2 t_э^2$$

Таким чином, $h_{x_i}^2$, $h_{y_i}^2$, $h_{z_i}^2$ набувають значення рівні одиниці, а $h_{\dot{x}_i}^2$, $h_{\dot{y}_i}^2$, $h_{\dot{z}_i}^2$ дорівнюють $t_э^2$.

Тепер зосередимося на управлінні процесом вимірювання координат при вирішенні задачі оцінювання параметрів траєкторій ПС. Задача оптимальної фільтрації частково спостережуваної послідовності $\{\boldsymbol{\upsilon}_i(t),\ \mathbf{Y}_i(t)\}$, $i = 1,\ 2,\ldots,N;\ t = 1,\ 2,\ldots,T-1$, полягає у побудові оптимальних (у середньоквадратичному сенсі) оцінок неспостережних значень векторів $\upsilon_i(t)$ за спостереженнями

$$(\mathbf{Y}_i)_0^t = \{\mathbf{Y}_i(0),\ \mathbf{Y}_i(1),\ldots,\ \mathbf{Y}_i(t)\},$$
$$t = 1,\ 2,\ldots,T-1.$$

Модифікований вираз рекурентної фільтрації з урахуванням керованих спостережень за цілями для оптимальних оцінок $\hat{\boldsymbol{\upsilon}}_i(t)$ і коваріаційних матриць $\boldsymbol{\Psi}_i(t)$ має такий вигляд [8]:

$$\begin{aligned}
&\hat{\boldsymbol{\upsilon}}_{эi}(t+1) = \boldsymbol{\Phi}_i(t)\hat{\boldsymbol{\upsilon}}_i(t);\\
&\boldsymbol{\Psi}_{эi}(t+1) = \boldsymbol{\Phi}_i(t)\boldsymbol{\Psi}_{n-1}\boldsymbol{\Phi}_i^{\mathrm{T}}(t);\\
&\mathbf{K}_i(t+1) = \boldsymbol{\Psi}_{эi}(t+1)\mathbf{H}_i^{\mathrm{T}}(t,\alpha(t))\times\\
&\times\left(\mathbf{H}_i(t,\alpha(t))\boldsymbol{\Psi}_{эi}(t+1)\mathbf{H}_i^{\mathrm{T}}(t,\alpha(t)) + \mathbf{B}(t)\mathbf{B}^{\mathrm{T}}(t)\right);\\
&\hat{\boldsymbol{\upsilon}}_i(t+1) = \mathbf{C}_i(t) + \hat{\boldsymbol{\upsilon}}_{эi}(t+1) + \mathbf{K}_i(t+1)\times\\
&\times\left(\mathbf{Y}_i(t+1) - \mathbf{C}_i(t,\alpha(t)) - \mathbf{H}_i(t,\alpha(t))\hat{\boldsymbol{\upsilon}}_{эi}(t+1)\right);\\
&\boldsymbol{\Psi}_i(t+1) = \boldsymbol{\Psi}_{эi}(t+1) + \mathbf{b}_i(t)\mathbf{b}_i^{\mathrm{T}}(t) -\\
&-\mathbf{K}_i(t+1)\mathbf{H}_i(t,\alpha(t))\boldsymbol{\Psi}_{эi}(t+1),\\
&\boldsymbol{\Psi}_i(0) = \boldsymbol{\Psi}_{i0}.
\end{aligned} \tag{4}$$

Логічно припустити, що до початку спостережень всі управління дорівнюють нулю ($\alpha(-1) = 0$). Врахуємо при цьому «вартості» проведення вимірювання, а також перемикання каналів. Тоді якість управління спостереженнями в задачі фільтрації можна охарактеризувати таким функціоналом [7]:

$$\begin{aligned}
J = \sum_{i=1}^{N}\Big\{&\sum_{t=1}^{T} tr\left(\mathbf{h}_i^{\mathrm{T}}(t)\boldsymbol{\Psi}_i(t)\right) +\\
&+\sum_{j=1}^{M}\sum_{t=0}^{T}\Big[\alpha^{ij}(t)\boldsymbol{\Gamma}_j(t) +\\
&+\alpha^{ij}(t)\left(1-\alpha^{ij}(t-1)\right)\gamma_j(t)\Big]\Big\}.
\end{aligned} \tag{5}$$

Тут $tr\left(\mathbf{h}_i^{\mathrm{T}}(t)\boldsymbol{\Psi}_i(t)\right) = \sum_{\nu=1}^{k} h_i^{\nu\nu}(t)\Psi_i^{\nu\nu}(t)$ характеризує зважену величину похибок фільтрації координат $i$-го об'єкту, причому можливість зміни елементів діагональних матриць $\mathrm{h}_i(t)$ дозволяє виділяти ті координати, точність оцінювання яких бажано підвищити. Решта членів у виразі (5) враховують «вартість» спостережень і перемикання каналів.

Таким чином, завдання управління спостереженнями системою РЛС УПР при супроводженні ПС полягає в знаходженні параметрів процесу

$$\begin{aligned} &\alpha(0),\ \alpha(1),\ \ldots,\ \alpha(T-1); \\ &\Psi_i(0),\ \Psi_i(1),\ \ldots,\ \Psi_i(T-1), \\ &\Psi_i(T),\ i=1,\ 2,\ldots,\ N, \end{aligned} \qquad (6)$$

що задовольняє заданим обмеженням, для яких виконана початкова умова $\Psi_i(0)=\Psi_{i0},\ i=1,\ 2,\ldots,\ N$, і при цьому функціонал (5) приймає найменше можливе значення.

Для будь-якого $T < \infty$ множина різних значень, які може набувати функціонал (5), скінчена і не перевищує величину

$M^T \times (N+1)^T$ [8].

Тому рішення поставленої задачі завжди існує.

Після знаходження параметрів процесу (6) необхідно здійснити відповідні підстановки в системі рекурентних рівнянь (4). В результаті отримаємо систему для знаходження оцінок $\hat{\upsilon}_i(t)$ неспостереженого вектора процесу $\upsilon_i(t),\ t=0,\ 1,\ldots,\ T$ (для кожного $i$ = 1,2,…, $N$) при оптимальному в сенсі критерію (5) управлінні процесом спостереження.

Отже, нами виконані математичні побудови для формалізації задачі оптимального управління системою РЛС УПР і встановлено, що рішення задачі у наведеній постановці завжди існує. Для практичної реалізації наведених побудов можна застосувати алгоритм вирішення одержаної оптимізаційної задачі, наведений в праці [8].

## Результати моделювання

Моделювання проводилося з метою оцінки ефективності розробленого методу і алгоритму оптимального управління системою РЛС УПР в режимі супроводження ПС. У якості базового методу, використаного для порівняння, був застосований метод рівномірного зондування (розподілу енергетичних ресурсів) РЛС, який на сьогодні є основним в реальних радіолокаційних комплексах в умовах складної повітряної обстановки.

При застосуванні розробленого методу для РЛС УПР виграш в точності оцінки параметрів супроводжуваних об'єктів в порівнянні з базовим методом склав 17 – 51 %. Оптимальні плани супроводження ПС відрізняються істотною нерівномірністю.

## Висновки

В результаті виконаної роботи розроблений новий спосіб підвищення ефективності інформаційного забезпечення АСУ ПР за рахунок оптимального управління системою РЛС УПР. Запропонований метод оптимального управління системою РЛС УПР відрізняється від відомих високою ефективністю.

В ході подальшої роботи запропонований спосіб буде вдосконалений для урахування специфіки супроводження ПС різного класу і конкретних АСУ ПР.

## Список посилань


1. Управление воздушным движением / Т.Г. Анодина, С.В. Володин, В.П. Куранов, В.И. Мокшанов. – М.: Транспорт, 1988. – 228 с.
2. Анодина Т.Г., Кузнецов А.А., Маркович Е.Д. Автоматизация управления воздушным движением. М. : Транспорт, 1992. – 279 с.
3. Автоматизированные системы УВД: Справочник / В.И. Савицкий, В.А. Василенко, Ю.А. Владимиров, В.В. Тагилов. М.: Транспорт , 1986. – 412 с.
4. Дубровский В.И., Крыжановский Г.А., Солодухин В.А. Организация радиотехнического обеспечения в системе УВД. М.: Транспорт, 1985. – 164 с.
5. Конторов Д.С., Голубев-Новожилов Ю.С. Введение в радиолокационную системотехнику. – М.: Сов. радио, 1971. – 368 с.
6. Кузьмин С.З. Основы проектирования систем цифровой обработки радиолокационной информации. – М.: Радио и связь, 1986. – 352 с.
7. Гришин И.Ю., Можар М.К., Решетник В.М. Проблемы управления зенитными ракетными комплексами // Наука і оборона, - 1994, вип. 3. с. 27-32.
8. Гришин И.Ю. Актуальные проблемы оптимизации управления в технических и экономических системах. – Ялта: РИО КГУ, 2009. – 286 с.


Поступила в редакцию 25.09.2009

УДК 004.032

*СЕРАЯ О.В.,*
*КАТКОВА Т.И.,*
*ФИЩУКОВА Н.В.*

# НЕЧЕТКАЯ БАЙЕСОВА ЭКСПЕРТНАЯ СИСТЕМА

Предложена методика расчета функций принадлежности апостериорных вероятностей состояний контролируемого объекта для нечеткой байесовой экспертной системы.

The calculation method of membership functions of a posteriori probabilities of the states of the controlled object is offered for fuzzy Bayes expert system.

## Постановка проблемы, анализ последних публикаций

Экспертные системы (ЭС) – один из наиболее эффективных инструментов оценивания состояния контролируемых объектов. Такая система преобразует набор измеренных значений $x_1, x_2, ..., x_n$ контролируемых параметров объекта в значение $y$ – параметра, оценивающего состояние этого объекта. Если при этом механизм логического вывода системы – продукционный, то процедура преобразования сводится к применению совокупности правил вида

$$\begin{aligned} &ЕСЛИ\ \ x_1\ \ это\ \ A_1, \\ &\qquad x_2\ \ это\ \ A_2, \\ &\qquad \ldots\ldots\ldots\ldots\ldots\ldots, \\ &\qquad x_n\ \ это\ \ A_n, \\ &ТО\ \ \ \ \ \ y\ \ это\ \ B. \end{aligned} \tag{1}$$

Продукционные системы очень удобны для использования, поскольку логика вывода в такой системе аналогична той, какую использует в подобной ситуации человек. Точность оценивания состояния в такой системе может быть сделана как угодно высокой и ограничивается только числом контролируемых параметров, точностью их измерения и правильностью заключений, образующих правила (1). Неопределенность, неизбежно сопровождающая все этапы процедуры оценивания состояния объектов с использованием ЭС, приводит к появлению и все более широкому использованию нечетких ЭС. Одной из наиболее известных таких систем является система нечеткого вывода Мамдани – Заде [1]. Применительно к задаче оценки состояния объекта эта система работает следующим образом. Для каждого из возможных состояний объекта $S_1, S_2, ..., S_p$ формируются функции принадлежности контролируемых параметров $\mu^{(k)}(x_j)$, $k = 1,2,...,p$, $j = 1,2,...,n$, диапазону возможных своих значений, определяемому состоянием. В соответствии с этим при получении конкретного набора измеренных значений параметров $X^{(0)} = \{x_1^{(0)},...,x_n^{(0)}\}$ осуществляется вычисление значений функций принадлежности $\mu^{(k)}(X_j^{(0)})$, $k = 1,2,...,p$, $j = 1,2,...,n$. Далее значения функций принадлежности, относящихся к каждому из состояний, агрегируются (чаще всего с использованием операции логического суммирования). При этом получают

$$\begin{aligned} &\mu^{(k)}(X^{(0)}) = \\ &= max\{\mu^{(k)}(x_1^{(0)}), \mu^{(k)}(x_2^{(0)}), \\ &...,\mu^{(k)}(x_n^{(0)})\},\ \ k = 1,2,...,p. \end{aligned} \tag{2}$$

Завершающей является операция дефуззификации выполняемая, например, следующим образом:

$$\hat{k} = \frac{\sum_{k=1}^{p} \mu^{(k)}(x^{(0)}) \cdot k}{\sum_{k=1}^{p} \mu^{(k)}(x^{(0)})}. \tag{3}$$

Недостатки описанной процедуры достаточно очевидны: снижение точности диагностики состояния за счет агрегирования, а также неоднозначность трактовки результата операции дефуззификации.

Другой подход, не требующий дефуззификации, реализован в модели нечеткого вывода Такаги-Сугено [2]. В этой модели для каждого из возможных состояний объекта рассчитывается уравнение регрессии

$$y_k = a_{k_0} + a_{k_1} x_1 + a_{k_2} x_2 + ... + a_{k_n} x_n, \tag{4}$$
$$k = 1,2,...,p,$$

связывающее некоторый результирующий параметр $y$ с результатами непосредственных измерений параметров $x_1, x_2, ..., x_n$. При этом коэффициенты $(a_{k_j})$ в (4) оцениваются статистически. С другой стороны, тот же что и в модели Мамдани – Заде набор функций принадлежности $\mu^{(k)}(x_j)$ используется для формирования весовых коэффициентов $W_k$, $k = 1,2,..., p,$ по правилу

$$W_k = min\{\mu^{(k)}(x_j)\}, \quad k = 1,2,..., p. \qquad (5)$$

Теперь с учетом этих коэффициентов рассчитывается оценка состояния объекта

$$\hat{k} = \sum_{k=1}^{p} \frac{W_k}{\sum_{k=1}^{p} W_k} \cdot y_k. \qquad (6)$$

Слабые звенья этой процедуры: необоснованный выбор структуры уравнения регрессии (4) и использование операции логического умножения (5) при расчете весовых коэффициентов.

Принципиально другая идея реализуется в предложенной в [3] нечеткой байесовой ЭС. В этой системе нечеткость исходных данных отображается в описании с помощью функций принадлежности $\mu^{(k)}(x_j)$ нечетких значений априорных вероятностей наблюдения значений контролируемых параметров $x_j$ при условии, что объект находится в состоянии $S_k$. Байесова система преобразует контролируемый набор параметров $X^{(0)}$ с учетом совокупности $\{\mu^{(k)}(x_j^{(0)})\}$ в набор апостериорных вероятностей $P\left(S_k / X^{(0)}\right)$, $k$=1,2,…$p$. Понятно, что нечеткость исходных данных навязывает нечеткость результата. В [3] нечеткие априорные вероятности описаны треугольными функциями принадлежности. Исчерпывающий результат состоял бы в получении функций принадлежности апостериорных вероятностей состояний. К сожалению, в [3] содержится лишь паллиативное решение задачи: получены формулы для расчета носителей нечетких чисел, описывающих апостериорное распределение вероятностей состояний.

## Цель

Разработка технологии расчета функций принадлежности для апостериорных вероятностей состояний объекта.

## Постановка задачи

Пусть по результатам предварительной обработки реальных данных или экспертного оценивания получены основные статистические характеристики (математическое ожидание и дисперсия) значений априорных вероятностей наблюдения каждого из контролируемых параметров $x_j$, $j = 1,2,...,n,$ при условии, что объект находится в каждом из возможных состояний $S_1, S_2, ..., S_p$, то есть имеются наборы $\left(m_1^{(k)}, m_2^{(k)}, ..., m_n^{(k)}\right)$, $\left(D_1^{(k)}, D_2^{(k)}, ..., D_n^{(k)}\right)$, $k = 1,2,..., p$.

В соответствии с формулой Байеса набор апостериорных вероятностей состояний объекта в ситуации, когда в результате контроля получен вектор значений параметров $X^{(0)} = \left(x_1^{(0)}, ..., x_n^{(0)}\right)$, отыскивается в результате последовательного применения формул

$$P\left(S_k / x_1^{(0)}\right) = \frac{P\left(x_1^{(0)} / S_k\right) \cdot P(S_k)}{\sum_{k=1}^{p} P\left(x_1^{(0)} / S_k\right) \cdot P(S_k)}, \qquad (7)$$

$$P\left(S_k / x_1^{(0)}, x_2^{(0)}\right) = \frac{P\left(x_2^{(0)} / S_k\right) \cdot \hat{P}_1(S_k)}{\sum_{k=1}^{p} P\left(x_2^{(0)} / S_k\right) \cdot \hat{P}_1(S_k)}, \qquad (8)$$

$$\hat{P}_1(S_k) = P\left(S_k / x_1^{(0)}\right), \quad k = 1,2,..., p, \qquad (9)$$

$$P\left(S_k / x_1^{(0)}, x_2^{(0)}, x_3^{(0)}\right) = \frac{P\left(x_3^{(0)} / S_k\right) \cdot \hat{P}_2(S_k)}{\sum_{k=1}^{p} P\left(x_3^{(0)} / S_k\right) \cdot \hat{P}_2(S_k)}, \qquad (10)$$

$$\hat{P}_2(S_k) = P\left(S_k / x_1^{(0)}, x_2^{(0)}\right),, \quad k = 1,2,..., p \qquad (11)$$

и т.д.

Поставим задачу расчета функций принадлежности нечетких значений апостериорных вероятностей состояний объекта

$$P\left( S_k \Big/ x_1^{(0)}, x_2^{(0)}, \ldots, x_n^{(0)} \right), \; k = 1,2,\ldots,p.$$

## Основные результаты

Предварительно рассмотрим следующую вспомогательную задачу. Пусть заданы математическое ожидание $m \neq 0$ и дисперсия $D$ случайной величины $z$. Требуется оценить математическое ожидание и дисперсию случайной величины $v = \frac{1}{z}$.

Понятно, что $M[v] = M\left[\frac{1}{z}\right] = \frac{1}{m}$. Получим теперь оценку дисперсии случайной величины $v$. В соответствии с неравенством Чебышева вероятность того, что отклонение случайной величины $z$ от ее математического ожидания $m$ меньше заданного положительного $\varepsilon$ не меньше, чем $1 - \frac{D}{\varepsilon^2}$, то есть

$$P(m - \varepsilon < z < m + \varepsilon) \geq 1 - \frac{D}{\varepsilon^2}.$$

Зададим требуемую вероятность $\gamma$ того, что диапазон $[m - \varepsilon, m + \varepsilon]$ накрывает $z$ и найдем значение $\varepsilon$. Имеем

$$1 - \frac{D}{\varepsilon^2} = \gamma.$$

Отсюда

$$\varepsilon = \sqrt{\frac{D}{1-\gamma}} = \frac{\sigma}{\sqrt{1-\gamma}}.$$

Таким образом, с вероятностью $\gamma$ границы диапазона определяются значениями

$$z_{min} = m - \frac{\sigma}{\sqrt{1-\gamma}} > 0, \; z_{max} = m + \frac{\sigma}{\sqrt{1-\gamma}}.$$

Тогда диапазон возможных значений $v$ определяется интервалом

$$\left[\frac{1}{z_{max}}, \frac{1}{z_{min}}\right] = \left[\frac{1}{m + \frac{\sigma}{\sqrt{1-\gamma}}}, \frac{1}{m - \frac{\sigma}{\sqrt{1-\gamma}}}\right].$$

Длина этого интервала равна

$$l = \frac{1}{m - \frac{\sigma}{\sqrt{1-\gamma}}} - \frac{1}{m + \frac{\sigma}{\sqrt{1-\gamma}}} = 2\frac{\frac{\sigma}{\sqrt{1-\gamma}}}{m^2 - \frac{\sigma^2}{1-\gamma}}.$$

При этом, в соответствии с тем же неравенством Чебышева

$$1 - \frac{D[v]}{l^2} = \gamma,$$

откуда

$$D[v] = l^2(1-\gamma) = \frac{4D}{\left(m^2 - \frac{D}{1-\gamma}\right)^2}.$$

Вернемся к основной задаче. Будем последовательно оценивать математическое ожидание и дисперсию апостериорных вероятностей, задаваемых в соответствии с (7)-(11). Рассчитаем математическое ожидание и дисперсию числителя и знаменателя в соотношении (7).

$$M\left[P\left(x_1^{(0)} \Big/ S_k\right) \cdot P(S_k)\right] = m_1^{(k)} \cdot P(S_k), \quad (12)$$

$$D\left[P\left(x_1^{(0)} \Big/ S_k\right) \cdot P(S_k)\right] = D_1^{(k)} \cdot P^2(S_k), \quad (13)$$

$$M\left[\sum_{k=1}^{p}\left(x_1^{(0)} \Big/ S_k\right) \cdot P(S_k)\right] = \sum_{k=1}^{p} m_1^{(k)} \cdot P(S_k), \quad (14)$$

$$D\left[\sum_{k=1}^{p}\left(x_1^{(0)} \Big/ S_k\right) \cdot P(S_k)\right] = \sum_{k=1}^{p} D_1^{(k)} \cdot P^2(S_k), \quad (15)$$

Тогда:

$$M\left[P\left(S_k \Big/ x_1^{(0)}\right)\right] =$$

$$= M\left[\frac{P\left(x_1^{(0)}/S_k\right)\cdot P(S_k)}{\sum_{k=1}^{p} P\left(x_1^{(0)}/S_k\right)\cdot P(S_k)}\right] =$$

$$= \frac{m_1^{(k)}\cdot P(S_k)}{\sum_{k=1}^{p} m_1^{(k)}\cdot P(S_k)}.$$

Для оценки дисперсии величины $P\left(S_k/x_1^{(0)}\right)$ используем следующее известное соотношение: для независимых случайных величин $X$ и $Y$

$$D[XY] = D[X]D[Y] +$$
$$+ M^2[X]\cdot D[Y] + M^2[Y]\cdot D[X].$$

Тогда

$$D\left[P\left(S_k/x_1^{(0)}\right)\right] =$$

$$= D\left[P\left(x_1^{(0)}/S_k\right)\cdot P(S_k)\times\right.$$

$$\left.\times\frac{1}{\sum_{k=1}^{p} P\left(x_1^{(0)}/S_k\right)\cdot P(S_k)}\right] =$$

$$= D^2\left[P\left(x_1^{(0)}/S_k\right)\cdot P(S_k)\right] +$$

$$+ D^2\left[\frac{1}{\sum_{k=1}^{p} P\left(x_1^{(0)}/S_k\right)\cdot P(S_k)}\right] +$$

$$+ M^2\left[P\left(x_1^{(0)}/S_k\right)\cdot P(S_k)\right]\times \qquad (16)$$

$$\times D\left[\frac{1}{\sum_{k=1}^{p} P\left(x_1^{(0)}/S_k\right)\cdot P(S_k)}\right] +$$

$$+ M^2\left[\frac{1}{\sum_{k=1}^{p} P\left(x_1^{(0)}/S_k\right)\cdot P(S_k)}\right]\times$$

$$\times D\left[P\left(x_1^{(0)}/S_k\right)\cdot P(S_k)\right].$$

В соответствии с решением вспомогательной задачи оценка величины

$$D\left[\frac{1}{\sum_{k=1}^{p} P\left(x_1^{(0)}/S_k\right)\cdot P(S_k)}\right]$$

с заданной надежностью имеет вид

$$D\left[\frac{1}{\sum_{k=1}^{p} P\left(x_1^{(0)}/S_k\right)\cdot P(S_k)}\right] =$$

$$= \frac{4D_1^{(k)}\cdot P^2(S_k)}{\left[\left(m_1^{(k)}\cdot P(S_k)\right)^2 - \frac{D_1^{(k)}\cdot P^2(S_k)}{1-\gamma}\right]^2} = \qquad (17)$$

$$= \frac{4D_1^{(k)}}{P^2(S_k)\left[\left(m_1^{(k)}\right)^2 - \frac{D_1^{(k)}}{1-\gamma}\right]^2}.$$

Подставляя теперь (12)-(15), а также (17) в (16) получим

$$D\left[P\left(S_k/x_1^{(0)}\right)\right] =$$

$$= \left(D_1^{(k)}\right)\cdot P^2(S_k)\times$$

$$\times\frac{4\left(D_1^{(k)}\right)}{P^2(S_k)\left[\left(m_1^{(k)}\right)^2 - \frac{D_1^{(k)}}{1-\gamma}\right]^2} +$$

$$+\left(m_1^{(k)}\right)^2 \cdot \frac{4D_1^{(k)}}{\left[\left(m_1^{(k)}\right)^2 - \frac{D_1^{(k)}}{1-\gamma}\right]^2} +$$

$$+\frac{1}{\left(m_1^{(k)}\right)^2} \cdot D_1^{(k)}. \qquad (18)$$

Проведенные вычисления с учетом (9) повторим для соотношений (8), (10) и т.д.

Таким образом, в конце концов будут получены наборы значений математического ожидания $\left(m_A^{(1)}, m_A^{(2)}, ..., m_A^{(p)}\right)$ и дисперсии $\left(D_A^{(1)}, D_A^{(2)}, ..., D_A^{(p)}\right)$ для всех компонентов апостериорного распределения вероятностей состояний объекта. С использованием этих наборов для описания функций принадлежности соответствующих нечетких чисел естественно выбрать гауссовы модели

$$\mu_A\left(P(S_k)\right) = exp\left\{-\frac{\left(P(S_k) - m_A^{(k)}\right)^2}{2D_A^{(k)}}\right\},$$

$k = 1,2,...,p$.

Поставленная задача решена.

## Выводы

Таким образом, получены соотношения для описания функций принадлежности апостериорных вероятностей состояний объекта в нечеткой байесовой экспертной системе, в которой неопределенность входа задается значениями статистических характеристик априорных вероятностей.

## Список литературы


1. Zadeh L.A. The concept of linguistic variable and its application to approximate reasoning / L.A. Zadeh // Information Sciences, 1975. – Vol.4. – pp 199-249.
2. Takagi T. Fuzzy identifications of systems and its application to modeling and control / T. Takagi, M. Cugeno // IEEE Trans. SMC, 1985. – pp 116-132.
3. Серая О.В. Модели и информационные технологии оценки и прогнозирования состояния многомерных динамических объектов в условиях нечетких входных данных: дис. канд. техн. наук: 05.13.06; защищена 17.01.02; утв. 13.03.02 / Серая Оксана Владимировна; НТУ «ХПИ». – Х., 2001. – 251 с.


Поступила в редакцию 21.10.2009

УДК 519.5:681:513

*ПАВЛОВ О.А.,*
*ЛІЩУК К.І.,*
*ШТАНЬКЕВИЧ О.С.,*
*ІВАНОВА Г.А.,*
*ФЕДОТОВ О.П.*

# МОДИФІКОВАНИЙ МЕТОД АНАЛІЗУ ІЄРАРХІЙ (ВЕРСІЯ 1,2)

В статті розглядаються дві версії модифікованого методу аналізу ієрархій для випадку великої кількості альтернатив. Наведений приклад, який демонструє правильність запропонованої модифікації МАІ.

In article two version of modification of the Analytic Hierarchy Process by T.Saaty (AHP), which may used for big amount of alternative is considered. An example, which demonstrate the adequacy of proposed technique is given.

## Вступ

Метод аналізу ієрархій, запропонований Т.Сааті [1-3], в теперішній час є одним із найбільш використовуваних методів вирішення задач багатокритеріального вибору. Однак, класична реалізація даного методу має декілька недоліків та обмежень, а саме:

а) велика кількість експертної інформації, яка представляє собою множину оцінок переваг, отриманих в результаті попарних порівнянь альтернатив та критеріїв;

б) обмеження на кількість альтернатив, які одночасно порівнюються – не рекомендується більше 9;

в) матриці попарних порівнянь, з якими працює МАІ повинні бути повністю узгодженими, однак, найчастіше отримані матриці попарних порівнянь є не повністю узгодженими в силу впливу на експерта різних факторів та властивостей оцінюваних альтернатив.

В [4,5] запропоновані та обґрунтовані моделі оптимізації для знаходження ваг об‘єктів по емпіричним матрицям попарних порівнянь, які дозволяють знаходити ваги об‘єктів по неповністю узгодженим матрицям попарних порівнянь. Тому актуальною залишається проблема розширення області застосування МАІ на випадок великої кількості альтернатив (істотно перевищуючих їх звичайну кількість, при якому застосування МАІ вважається коректним). Подібна задача може виникнути в двох випадках:

а) найкраща альтернатива не обирається з набору реально існуючих альтернатив, а альтернативи генеруються штучно для вибору найкращої, після чого в реалізацію цієї альтернативи вкладаються істотні ресурси;

б) штучно генеруються альтернативи; за допомогою МАІ знаходяться результуючі ваги, по яким будується опис глобальної мети.

Коректне обґрунтування пропонованих модифікацій МАІ можливо лише в тому випадку, коли задається формальна модель, якій відповідають емпіричні матриці попарних порівнянь останнього рівня ієрархії МАІ. В цьому випадку можна досліджувати емпіричні (статистичні) властивості алгоритмів, їх ефективність, пропонувати практичні рекомендації їх використання.

## Модель емпіричних матриць попарних порівнянь останнього рівня ієрархії МАІ

Розглянемо наступну задачу багатокритеріального вибору: маємо глобальну мету та ряд альтернатив $A_1,...A_m$. За допомогою МАІ необхідно знайти ваги $\omega_1,...\omega_m$ об‘єктів (альтернатив, критеріїв) по відношенню до деякої властивості, мети (критерію). Ваги визначаються за емпіричною матрицею попарних порівнянь $(\gamma_{ij})_1^m$, яка задається експертом (експертами). Число $\gamma_{ij}$ задається експертом та показує у скільки разів вага об‘єкту $A_i$ більше ваги об‘єкту $A_j$ по відношенню до заданої мети (критерію). Розглянемо довільні альтернативи $A_i$, $A_j$, які порівнюються експертом (експертами) за ефективністю відносно довільного критерію попереднього рівня ієрархії МАІ.

В ідеальному випадку, тобто в випадку, коли припускається, що на експерта не впливають фактори, які перетворюють його рішення (його компетентність, кількість альтернатив, для яких будується матриця попарних порівнянь, неоднозначність якісного опису критерію, технологія та послідовність заповнення емпіричної матриці попарних

порівнянь, психологічні фактори, тощо), значення емпіричного коефіцієнту $\gamma_i$ не залежить не від кількості альтернатив, з яких знаходиться найкраща, а ні від їх складу. Тоді $\gamma_{ij} = \frac{\varpi_i}{\varpi_j}$ в будь-якій емпіричній матриці попарних порівнянь при попарному порівнянні альтернативи $A_i$ з альтернативою $A_j$, $\omega_i, \omega_j \geq 0$ інтерпретуються як ідеальні значення ваг альтернатив $A_i$ та $A_j$.

Тепер розглянемо випадок, в якому реально на рішення експерта діють обурюючи фактори. Формально дія цих факторів пропонується описувати за допомогою параметричного імовірнісного розподілу. Закономірності, які визначають значення та зміну параметрів імовірнісного розподілу (імовірнісних розподілів) як й сам імовірнісний розподіл (імовірнісні розподіли) є формальною моделлю факторів, які перетворюють рішення експерта (експертів).

*Примітка.* Для дослідження ефективності запропонованих моделей оптимізації, використаний параметричний рівномірний розподілу, як один з найбільш жорстких розподілів. Всі версії та їх реалізації модифікованого МАІ можуть біти використані для цієї моделі емпіричної матриці попарних порівнянь останнього рівня ієрархії МАІ.

Спочатку модифікацію МАІ наведемо для дворівневої ієрархії, потім отримані результати адаптуємо для загального випадку. Необхідно відмітити, що дворівнева ієрархія має самостійне практичне значення.

### Модифікований метод аналізу ієрархій для дворівневої структури

Нехай дерево ієрархій має наступний вигляд:

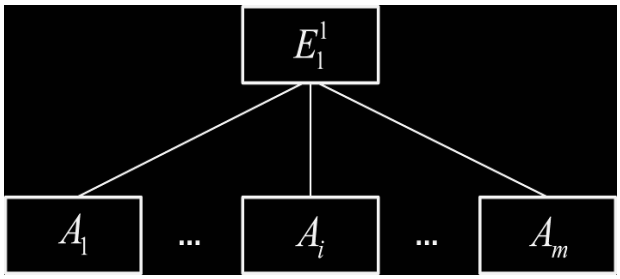



***Рис. 1. Дерево ієрархії з дворівневою структурою***

На Рис. 1 $m$ – кількість альтернатив, достатньо велике число.

### ММАІ (перша версія)

Відповідає випадку, коли яка-небудь інформація про закономірності перекручування емпіричних значень $\gamma_{ij}$ відсутня. Маємо одну емпіричну матрицю попарних порівнянь розміру $m \times m$.

Застосовувати в цьому випадку класичний МАІ неможна. Пропонується оцінки ваг виконувати з використанням моделей оптимізації, а результуючий вибір проводити у відповідності зі значеннями критеріїв $M_1, M_2$ (детально процедура описана у статтях [4,5]). Результати статистичного аналізу показують, що використання моделей оптимізації:

а) дозволяє використовувати частково заповнені матриці попарних порівнянь;

б) отримувати оцінки ваг (для великих $m$) в середньому краще, ніж класичний МАІ;

в) в залежності від значень критеріїв $M_1$ та $M_2$ виконувати обґрунтований вибір найкращої альтернативи (значення $M_1$ та $M_2$ невеликі) або використовувати отримані результати в якості попереднього аналізу ефективності альтернатив з заданої множини.

### ММАІ (друга версія)

Помилка у визначенні $\gamma_{ij}$ зростає зі збільшенням розмірності емпіричної матриці попарних порівнянь.

ММАІ (друга версія) реалізується в декілька етапів.

*Етап 1.*

По вихідній емпіричній матриці попарних порівнянь з використання моделей оптимізації по значенням критеріїв $M_1, M_2$ або за допомогою моделей, які наведені в [4], знаходяться оцінки ваг $\omega_i$ альтернатив $A_i, i = \overline{1, m}$.

*Примітка.* Враховуючи результати статистичного моделювання можна безпосередньо використовувати моделі 4, 5, 7.

Альтернативи впорядковуються у відповідності з убуванням значень $\varpi_i, i = \overline{1, m}$.

*Етап 2.*

Обираються найкращі $m_2$ ($m_2 \leq 15$) альтернативи та для них пропонується експерту (експертам) знову побудувати емпіричну матрицю попарних порівнянь.

*Примітка.* Число 15 – це максимальна кількість альтернатив, яка зустрічається у відомій авторам літературі, яка дозволяє коректно розв‘язати практичну задачу вибору найкращої альтернативи за допомогою класичного МАІ.

Далі для заново побудованої матриці попарних порівнянь повторюється Етап 1.

*Етап 3.*

Для $m_3$ ($m_3 \leq 7 \div 9$) найкращих альтернатив, які визначені на Етапі 2 знову експертом (експертами) визначається матриця попарних порівнянь. По цій матриці за допомогою класичного МАІ або за допомогою моделей оптимізації, які наведені в [4] оцінюються ваги альтернатив. Найкращою є альтернатива з найбільшою вагою.

Задача статистично значимо вирішена вірно, якщо на Етапі 3 емпірична матриця попарних порівнянь виявилась добре узгодженою.

При проведенні статистичних досліджень емпірична матриця попарних порівнянь генерувалась з еталонної (ідеально узгодженої) шляхом імовірнісного відхилення її елементів за принципом:

$$\gamma_{ij}^{*} = \gamma_{ij} + k_{ij}\gamma_{ij} \qquad (1)$$

де модулі коефіцієнтів $k_{ij}$ задавалися параметричним рівномірним розподілом, границі якого залежали від розмірності емпіричної матриці попарних порівнянь згідно Табл. 1, а знак $k_{ij}$ задавався рівноймовірно.

В результаті статистичного моделювання найкраща альтернатива в кожному випадку знаходилась вірно. При моделюванні довільна емпірична матриця попарних порівнянь $7 \times 7$ (на Етапі 3) задавалась добре узгодженою (з малим рівнем помилок експертів).

***Табл. 1. Границі рівномірного розподілу $|k_{ij}|$ в залежності від розміру матриці попарних порівнянь***

| Кількість альтернатив | Границі рівномірного розподілу $\|k_{ij}\|$ |
|---|---|
| 20 | (0; 0,15) |
| 30 | (0; 0,20) |
| 40 | (0,05; 0,25) |
| 50 | (0,10; 0,30) |
| 60 | (0,15; 0,35) |
| 70 | (0,2; 0,4) |

## Приклад використання модифікованого методу аналізу ієрархій для дворівневої структури

Маємо 70 альтернатив, серед яких необхідно визначити переможця відносно деякого глобального критерію. Нехай ці альтернативи мають ваги, нормовані значення (сума дорівнює одиниці) яких наведені в Табл. 2 (природно припускається, що ці ваги на практиці невідомі).

***Табл. 2. Нормовані еталони ваги альтернатив***

| № | Вага | № | Вага | № | Вага |
|---|---|---|---|---|---|
| 1 | 0,02382481 | 25 | 0,00306957 | 49 | 0,00682838 |
| 2 | 0,00955014 | 26 | 0,00324009 | 50 | 0,00169459 |
| 3 | 0,02249443 | 27 | 0,02348073 | 51 | 0,02326341 |
| 4 | 0,01402535 | 28 | 0,01173888 | 52 | 0,01879676 |
| 5 | 0,02113500 | 29 | 0,02191203 | 53 | 0,01589583 |
| 6 | 0,02329004 | 30 | 0,00687466 | 54 | 0,01305656 |
| 7 | 0,01112283 | 31 | 0,01160676 | 55 | 0,02043728 |
| 8 | 0,01696852 | 32 | 0,01951981 | 56 | 0,00256201 |
| 9 | 0,01539390 | 33 | 0,02355131 | 57 | 0,02367316 |
| 10 | 0,01084301 | 34 | 0,00163997 | 58 | 0,02313647 |
| 11 | 0,01235783 | 35 | 0,01933938 | 59 | 0,01081000 |
| 12 | 0,01379027 | 36 | 0,01659109 | 60 | 0,00115500 |
| 13 | 0,00467520 | 37 | 0,01935597 | 61 | 0,00345337 |
| 14 | 0,00972133 | 38 | 0,02289772 | 62 | 0,02306069 |
| 15 | 0,00817688 | 39 | 0,01275461 | 63 | 0,00835715 |
| 16 | 0,01904103 | 40 | 0,01143995 | 64 | 0,01408617 |
| 17 | 0,01626562 | 41 | 0,01260273 | 65 | 0,02048480 |
| 18 | 0,01673642 | 42 | 0,02041830 | 66 | 0,00246058 |
| 19 | 0,00579612 | 43 | 0,01738300 | 67 | 0,00688988 |
| 20 | 0,01792782 | 44 | 0,01653572 | 68 | 0,02350822 |
| 21 | 0,02256418 | 45 | 0,00948436 | 69 | 0,01217243 |
| 22 | 0,00922108 | 46 | 0,02305618 | 70 | 0,02036092 |
| 23 | 0,01278091 | 47 | 0,00717709 | | |
| 24 | 0,01002018 | 48 | 0,02046352 | | |

Позиції при впорядкуванні еталонних ваг за убуванням: 1, 57, 33, 68, 27, 6, 51, 58, 62, 46, 38, 21, 3, 29, 5, 65, 48, 55, 42, 70, 32, 37, 35, 16, 52, 20, 43, 8, 18, 36, 44, 17, 53, 9, 64, 4, 12, 54, 23, 39, 41, 11, 69, 28, 31, 40, 7, 10, 59, 24, 14, 2, 45, 22, 63, 15, 47, 67, 30, 49, 19, 13, 61, 26, 25, 56, 66, 50, 34, 60. Таким чином, дійсним переможцем є альтернатива під номером 1.

Альтернативи порівнюються експертом (експертами), в результаті чого формується емпірична матриця попарних порівнянь, яка є погано обумовленою. В нашому прикладі емпіричну матрицю будуємо на основі ідеально узгодженої (елементи якої – відношення ваг з табл. 2) шляхом імовірнісного відхилення (зашумлення) її елементів за принципом (1), де $k_{ij}$ – імовірнісний коефіцієнт, такий що модуль $|k_{ij}|$ розподілений рівномірно в інтервалі $(0,2; 0,4)$, а знак $k_{ij}$ визначається випадково з рівною імовірністю. Побудована емпірична матриця попарних порівнянь наведена в Табл.3 - 7.

***Табл. 3. Емпірична матриця попарних порівнянь (стовпці 1-14)***

| № / № | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **1** | 1,0000 | 1,4013 | 1,4230 | 0,8725 | 0,7684 | 1,4321 | 2,8311 | 1,8276 | 0,8356 | 2,8568 | 1,1272 | 2,2620 | 2,7341 | 3,2395 |
| **2** | 0,7136 | 1,0000 | 0,2435 | 0,8920 | 0,2657 | 0,2270 | 0,5432 | 0,3290 | 0,8074 | 1,2125 | 1,1214 | 0,9215 | 2,9185 | 1,2958 |
| **3** | 0,7027 | 4,1070 | 1,0000 | 1,0892 | 0,6019 | 0,5170 | 2,8365 | 0,7044 | 0,9296 | 2,8854 | 1,0289 | 2,3209 | 2,5156 | 3,1113 |
| **4** | 1,1461 | 1,1210 | 0,9181 | 1,0000 | 0,4381 | 0,3253 | 1,8151 | 1,1894 | 0,4641 | 1,7008 | 1,5550 | 0,6411 | 4,4737 | 1,9317 |
| **5** | 1,3014 | 3,7631 | 1,6615 | 2,2826 | 1,0000 | 1,2457 | 2,6093 | 0,7814 | 0,7793 | 1,3064 | 2,3724 | 0,9263 | 6,3948 | 1,0994 |
| **6** | 0,6983 | 4,4046 | 1,9343 | 3,0739 | 0,8028 | 1,0000 | 1,3168 | 0,7964 | 2,1696 | 1,4570 | 1,1395 | 0,9163 | 2,5978 | 3,5927 |
| **7** | 0,3532 | 1,8409 | 0,3526 | 0,5509 | 0,3832 | 0,7594 | 1,0000 | 0,8896 | 0,4436 | 1,4053 | 1,2151 | 1,1763 | 1,5446 | 1,6636 |
| **8** | 0,5472 | 3,0396 | 1,4197 | 0,8408 | 1,2797 | 1,2556 | 1,1241 | 1,0000 | 0,6625 | 0,9198 | 0,8438 | 0,8288 | 4,7884 | 2,3579 |
| **9** | 1,1968 | 1,2385 | 1,0757 | 2,1548 | 1,2831 | 0,4609 | 2,2541 | 1,5094 | 1,0000 | 0,7746 | 1,8681 | 1,5830 | 4,4098 | 2,3579 |
| **10** | 0,3500 | 0,8248 | 0,3466 | 0,5880 | 0,7655 | 0,6864 | 0,7116 | 1,0872 | 1,2911 | 1,0000 | 1,1511 | 0,4917 | 1,5727 | 1,6575 |
| **11** | 0,8872 | 0,8917 | 0,9720 | 0,6431 | 0,4215 | 0,8776 | 0,8230 | 1,1851 | 0,5353 | 0,8688 | 1,0000 | 1,2501 | 3,9297 | 1,7016 |
| **12** | 0,4421 | 1,0852 | 0,4309 | 1,5599 | 1,0795 | 1,0913 | 0,8501 | 1,2065 | 0,6317 | 2,0338 | 0,7999 | 1,0000 | 1,6081 | 0,7751 |
| **13** | 0,3658 | 0,3426 | 0,3975 | 0,2235 | 0,1564 | 0,3849 | 0,6474 | 0,2088 | 0,2268 | 0,6359 | 0,2545 | 0,6219 | 1,0000 | 0,6570 |
| **14** | 0,3087 | 0,7717 | 0,3214 | 0,5177 | 0,9096 | 0,2783 | 0,6011 | 0,4241 | 0,4241 | 0,6033 | 0,5877 | 1,2902 | 1,5220 | 1,0000 |
| **15** | 0,2555 | 1,3645 | 0,2703 | 0,9704 | 0,2752 | 0,2410 | 1,3736 | 0,3320 | 0,3994 | 0,5268 | 0,4813 | 0,9151 | 1,2007 | 1,4772 |
| **16** | 1,1850 | 3,3611 | 0,6430 | 0,9295 | 1,3015 | 1,4986 | 1,3064 | 0,8369 | 1,7799 | 3,1031 | 1,0705 | 2,3053 | 2,7232 | 3,1623 |
| **17** | 0,9787 | 2,7456 | 0,5475 | 0,7802 | 0,5328 | 0,5331 | 0,9875 | 1,6188 | 0,8064 | 1,0840 | 0,9058 | 1,7707 | 6,2528 | 2,8839 |
| **18** | 0,4966 | 1,3036 | 1,3924 | 1,9041 | 1,2191 | 0,5008 | 2,7035 | 0,7494 | 0,7644 | 1,1141 | 1,0008 | 2,0000 | 5,9152 | 1,3170 |
| **19** | 0,4068 | 0,4651 | 0,1751 | 0,8222 | 0,4546 | 0,3884 | 0,3849 | 0,4954 | 0,2716 | 1,0286 | 0,9073 | 0,2890 | 1,8577 | 0,8957 |
| **20** | 1,1118 | 3,0950 | 1,3093 | 2,2406 | 1,3166 | 0,5697 | 1,1962 | 1,5625 | 0,8219 | 1,2289 | 1,0764 | 0,9022 | 2,8922 | 2,9104 |
| **21** | 1,7374 | 1,6542 | 1,7545 | 2,8616 | 1,7812 | 1,6462 | 2,9445 | 2,4953 | 2,4422 | 3,8456 | 3,2306 | 1,1949 | 8,0950 | 3,8014 |
| **22** | 0,2594 | 1,7875 | 0,6224 | 1,2207 | 0,3044 | 0,6148 | 0,5812 | 0,3940 | 1,0206 | 1,5436 | 1,2606 | 0,4473 | 1,4141 | 0,6514 |
| **23** | 0,8884 | 0,9683 | 0,4157 | 1,3763 | 1,1085 | 0,9650 | 2,1968 | 1,3097 | 1,6456 | 0,8291 | 0,7518 | 1,8194 | 2,0094 | 1,9585 |
| **24** | 0,2993 | 1,6556 | 0,3037 | 0,5287 | 0,8582 | 0,8204 | 0,6089 | 0,4446 | 1,1515 | 0,6307 | 1,5188 | 1,2646 | 3,2124 | 0,7536 |
| **25** | 0,2263 | 0,5130 | 0,0955 | 0,3497 | 0,1044 | 0,2123 | 0,2122 | 0,2894 | 0,3495 | 0,1894 | 0,1883 | 0,3640 | 0,4789 | 0,2409 |
| **26** | 0,0984 | 0,5435 | 0,2458 | 0,1658 | 0,2616 | 0,2217 | 0,2119 | 0,1409 | 0,1407 | 0,4586 | 0,4156 | 0,1580 | 0,5236 | 0,2472 |
| **27** | 0,7229 | 3,9492 | 1,5160 | 2,6316 | 0,7807 | 0,7160 | 1,5796 | 0,9948 | 2,7508 | 1,4567 | 1,2680 | 3,0392 | 3,6950 | 4,4181 |
| **28** | 0,8877 | 0,8600 | 0,8458 | 1,4064 | 0,4037 | 0,3503 | 1,9538 | 1,1002 | 0,5190 | 0,8305 | 1,6450 | 0,6247 | 4,7406 | 1,9068 |
| **29** | 1,4642 | 1,5969 | 0,6873 | 1,1508 | 1,5429 | 1,3528 | 1,3468 | 2,0079 | 2,2181 | 1,5196 | 2,5664 | 1,0683 | 6,6997 | 1,6978 |
| **30** | 0,2200 | 1,1145 | 0,5518 | 0,3449 | 0,5279 | 0,5328 | 0,4200 | 0,2947 | 0,6871 | 1,0546 | 0,4052 | 0,3422 | 2,8159 | 1,3280 |
| **31** | 0,3365 | 2,1981 | 0,8850 | 1,2518 | 0,3671 | 0,3463 | 0,7504 | 1,0773 | 1,2235 | 0,7277 | 0,6786 | 0,5721 | 4,3688 | 2,0389 |
| **32** | 1,2789 | 1,5171 | 0,6536 | 2,4404 | 0,6729 | 0,5973 | 1,3227 | 1,7175 | 1,9150 | 1,3212 | 2,9502 | 2,7783 | 6,3662 | 1,4268 |
| **33** | 0,6855 | 1,6448 | 0,7977 | 1,2778 | 0,7577 | 0,6809 | 4,1594 | 1,0344 | 2,4986 | 4,1066 | 1,4377 | 3,0487 | 7,9210 | 4,2086 |
| **34** | 0,0499 | 0,1221 | 0,0489 | 0,0842 | 0,0518 | 0,1348 | 0,1117 | 0,1543 | 0,1692 | 0,2803 | 0,2071 | 0,1781 | 0,5813 | 0,3207 |
| **35** | 1,1783 | 1,4515 | 1,3029 | 0,9655 | 1,4326 | 0,6001 | 1,2672 | 1,9217 | 2,1798 | 3,1834 | 1,0509 | 0,9358 | 6,2821 | 1,3503 |
| **36** | 0,5327 | 1,2289 | 1,2396 | 2,2939 | 1,3755 | 1,1804 | 1,0901 | 1,6814 | 0,7570 | 2,2313 | 1,0163 | 1,9598 | 2,4059 | 1,2738 |
| **37** | 1,1923 | 1,3881 | 1,3732 | 0,9822 | 1,6531 | 0,5698 | 3,0091 | 2,2343 | 0,9443 | 1,3410 | 2,4415 | 2,3977 | 7,1489 | 3,5358 |
| **38** | 0,7122 | 4,0912 | 0,7173 | 1,1123 | 0,7709 | 1,5047 | 3,1753 | 0,9936 | 1,0411 | 3,1818 | 3,3697 | 3,0051 | 3,4219 | 4,4046 |
| **39** | 0,4011 | 0,9221 | 0,4110 | 0,6423 | 0,4249 | 0,9055 | 0,8608 | 1,2769 | 0,6143 | 0,8124 | 0,6898 | 1,4253 | 4,6076 | 2,3541 |
| **40** | 0,7895 | 0,8605 | 0,9351 | 0,6105 | 0,9699 | 0,9798 | 0,7244 | 0,5037 | 0,5388 | 0,7550 | 1,3398 | 1,6086 | 1,6890 | 1,7687 |
| **41** | 0,8546 | 1,8926 | 0,9063 | 0,6200 | 0,4117 | 0,8061 | 0,8660 | 1,3759 | 0,6077 | 0,8487 | 1,5720 | 1,3999 | 4,2372 | 0,9936 |
| **42** | 0,6459 | 3,3198 | 0,6395 | 2,1862 | 1,5697 | 0,5879 | 1,2294 | 1,9478 | 0,9004 | 2,8163 | 2,3978 | 2,5109 | 6,6685 | 3,7489 |
| **43** | 0,5324 | 1,2240 | 1,4294 | 1,7977 | 1,2803 | 0,5396 | 1,1071 | 1,7183 | 0,8056 | 1,1125 | 1,0189 | 0,9490 | 2,5647 | 3,0956 |
| **44** | 1,3407 | 3,4394 | 0,5596 | 0,8080 | 1,4270 | 0,4845 | 2,7813 | 0,6571 | 0,8017 | 2,6895 | 2,2344 | 0,8511 | 5,1225 | 1,2113 |
| **45** | 0,5942 | 1,7987 | 0,3229 | 1,2064 | 0,8593 | 0,7870 | 0,6338 | 0,4065 | 0,4434 | 1,3867 | 0,5890 | 0,5288 | 3,9407 | 0,7206 |
| **46** | 0,6548 | 3,5867 | 1,6552 | 1,1880 | 0,8139 | 0,7039 | 3,2413 | 1,9793 | 1,1031 | 1,5088 | 1,3894 | 1,1175 | 3,5381 | 1,5855 |
| **47** | 0,4567 | 1,4847 | 0,2422 | 0,7794 | 0,2574 | 0,5507 | 0,4314 | 0,3068 | 0,3235 | 0,4631 | 1,0853 | 0,9133 | 1,1806 | 0,5188 |
| **48** | 1,3825 | 3,6457 | 1,4809 | 2,4782 | 0,7185 | 1,5954 | 2,8973 | 0,8733 | 0,9775 | 1,3662 | 2,9360 | 2,4642 | 3,2683 | 1,5282 |
| **49** | 0,5360 | 0,5335 | 0,2163 | 0,3343 | 0,2351 | 0,5824 | 0,4614 | 0,2911 | 0,3060 | 0,4665 | 0,7957 | 0,3397 | 2,4289 | 0,4824 |
| **50** | 0,0477 | 0,1356 | 0,0548 | 0,0823 | 0,0538 | 0,1438 | 0,1054 | 0,0693 | 0,1940 | 0,1199 | 0,0936 | 0,2149 | 0,2667 | 0,3192 |
| **51** | 0,6933 | 3,6847 | 0,7093 | 2,7375 | 0,8378 | 1,5691 | 1,4085 | 2,0285 | 2,6845 | 1,5056 | 3,5700 | 2,8690 | 8,8249 | 1,8260 |
| **52** | 1,1536 | 1,4884 | 1,3026 | 2,0384 | 1,4080 | 1,2059 | 1,1772 | 0,7954 | 0,8769 | 1,2120 | 1,0960 | 2,1504 | 2,8656 | 3,7677 |
| **53** | 1,3325 | 1,2298 | 0,4951 | 0,8636 | 1,1425 | 0,4920 | 1,0266 | 0,6420 | 0,7243 | 2,5520 | 0,8603 | 1,9351 | 2,4727 | 1,2401 |
| **54** | 0,4074 | 0,9829 | 0,4325 | 1,6541 | 1,0076 | 0,3928 | 1,9802 | 1,3971 | 1,4442 | 0,9241 | 1,7839 | 0,7153 | 1,9928 | 1,9709 |
| **55** | 0,6073 | 1,5059 | 1,4154 | 2,6644 | 1,5522 | 0,6235 | 1,3653 | 0,8395 | 2,3816 | 3,6954 | 1,1949 | 1,1211 | 3,1221 | 4,0441 |
| **56** | 0,0746 | 0,1842 | 0,0805 | 0,1232 | 0,0893 | 0,1576 | 0,3564 | 0,1104 | 0,3012 | 0,1782 | 0,1438 | 0,2869 | 0,3685 | 0,1860 |

| № / № | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **57** | 0,7174 | 1,7760 | 0,7213 | 2,9030 | 0,8217 | 0,7001 | 3,6656 | 1,0710 | 1,1708 | 1,6459 | 1,4182 | 1,1570 | 3,4277 | 1,6981 |
| **58** | 1,8003 | 1,7266 | 0,7167 | 2,9286 | 2,0503 | 1,4308 | 1,5961 | 1,0339 | 1,0953 | 4,0620 | 1,2763 | 2,6475 | 8,3389 | 3,4836 |
| **59** | 0,3093 | 1,7359 | 0,7039 | 1,1139 | 0,3565 | 0,3190 | 0,7094 | 0,4684 | 1,2962 | 1,4604 | 1,4993 | 0,5451 | 4,3490 | 0,7620 |
| **60** | 0,0703 | 0,1909 | 0,0346 | 0,0602 | 0,0802 | 0,0353 | 0,1737 | 0,0489 | 0,1086 | 0,1995 | 0,0628 | 0,0642 | 0,3653 | 0,0849 |
| **61** | 0,2825 | 0,2469 | 0,2347 | 0,3642 | 0,2883 | 0,2264 | 0,5015 | 0,1441 | 0,4247 | 0,2238 | 0,1889 | 0,1731 | 0,5467 | 0,6815 |
| **62** | 0,6457 | 1,8085 | 1,9917 | 3,2630 | 0,7535 | 1,6436 | 3,2168 | 2,0184 | 1,0125 | 4,1507 | 1,3991 | 1,2441 | 3,4378 | 3,8223 |
| **63** | 0,2423 | 0,6186 | 0,5766 | 0,4020 | 0,6088 | 0,2554 | 1,3754 | 0,3669 | 0,3746 | 1,3935 | 0,4830 | 0,9819 | 3,2659 | 0,6124 |
| **64** | 1,0474 | 1,0091 | 0,4223 | 1,4714 | 1,2769 | 0,9672 | 1,9155 | 0,5730 | 1,7818 | 0,8974 | 0,7733 | 2,0068 | 2,2501 | 2,3902 |
| **65** | 1,4020 | 1,5462 | 1,5613 | 1,1007 | 0,6474 | 1,5233 | 1,3010 | 0,8775 | 2,2446 | 1,3548 | 3,1995 | 2,8507 | 7,6490 | 1,4444 |
| **66** | 0,1534 | 0,1860 | 0,0760 | 0,3133 | 0,1712 | 0,0762 | 0,1486 | 0,2458 | 0,2481 | 0,3712 | 0,1459 | 0,1191 | 0,9080 | 0,4135 |
| **67** | 0,5573 | 0,5118 | 0,4765 | 0,3375 | 0,6411 | 0,4809 | 0,9011 | 0,7244 | 0,3245 | 1,1128 | 0,4085 | 0,7693 | 2,5399 | 0,5264 |
| **68** | 0,6828 | 4,5995 | 1,8818 | 1,2649 | 2,0878 | 1,9143 | 1,5609 | 2,1864 | 1,0764 | 1,5987 | 1,4506 | 1,2695 | 3,4149 | 1,7096 |
| **69** | 0,3914 | 0,9771 | 0,8233 | 1,2878 | 0,3868 | 0,3960 | 1,5634 | 0,5095 | 1,4223 | 1,8566 | 0,7198 | 1,6284 | 1,9276 | 0,9360 |
| **70** | 0,5932 | 3,5462 | 1,3309 | 2,2060 | 0,6977 | 1,3745 | 3,1751 | 1,9345 | 0,9517 | 3,3671 | 3,2855 | 1,0580 | 7,8736 | 3,5411 |

***Табл. 4. Емпірична матриця попарних порівнянь (стовпці 15-28)***

| № / № | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **1** | 3,9145 | 0,8439 | 1,0217 | 2,0139 | 2,4581 | 0,8995 | 0,5756 | 3,8555 | 1,1256 | 3,3413 | 4,4184 | 10,1587 | 1,3834 | 1,1265 |
| **2** | 0,7329 | 0,2975 | 0,3642 | 0,7671 | 2,1501 | 0,3231 | 0,6045 | 0,5594 | 1,0327 | 0,6040 | 1,9493 | 1,8399 | 0,2532 | 1,1628 |
| **3** | 3,7003 | 1,5552 | 1,8265 | 0,7182 | 5,7118 | 0,7637 | 0,5700 | 1,6067 | 2,4053 | 3,2930 | 10,4734 | 4,0691 | 0,6596 | 1,1823 |
| **4** | 1,0305 | 1,0759 | 1,2817 | 0,5252 | 1,2163 | 0,4463 | 0,3495 | 0,8192 | 0,7266 | 1,8916 | 2,8596 | 6,0317 | 0,3800 | 0,7111 |
| **5** | 3,6339 | 0,7684 | 1,8769 | 0,8203 | 2,1996 | 0,7595 | 0,5614 | 3,2851 | 0,9022 | 1,1652 | 9,5742 | 3,8227 | 1,2808 | 2,4769 |
| **6** | 4,1495 | 0,6673 | 1,8757 | 1,9967 | 2,5749 | 1,7554 | 0,6075 | 1,6265 | 1,0363 | 1,2189 | 4,7095 | 4,5104 | 1,3967 | 2,8547 |
| **7** | 0,7280 | 0,7655 | 1,0126 | 0,3699 | 2,5982 | 0,8360 | 0,3396 | 1,7207 | 0,4552 | 1,6422 | 4,7128 | 4,7197 | 0,6331 | 0,5118 |
| **8** | 3,0122 | 1,1950 | 0,6177 | 1,3345 | 2,0187 | 0,6400 | 0,4008 | 2,5381 | 0,7635 | 2,2492 | 3,4556 | 7,0980 | 1,0053 | 0,9089 |
| **9** | 2,5038 | 0,5618 | 1,2400 | 1,3082 | 3,6817 | 1,2167 | 0,4095 | 0,9798 | 0,6077 | 0,8685 | 2,8610 | 7,1084 | 0,3635 | 1,9270 |
| **10** | 1,8983 | 0,3223 | 0,9225 | 0,8976 | 0,9722 | 0,8138 | 0,2600 | 0,6478 | 1,2062 | 1,5854 | 5,2796 | 2,1807 | 0,6865 | 1,2041 |
| **11** | 2,0775 | 0,9342 | 1,1039 | 0,9992 | 1,1021 | 0,9290 | 0,3095 | 0,7932 | 1,3301 | 0,6584 | 5,3093 | 2,4060 | 0,7887 | 0,6079 |
| **12** | 1,0928 | 0,4338 | 0,5648 | 0,5000 | 3,4598 | 1,1084 | 0,8369 | 2,2358 | 0,5496 | 0,7907 | 2,7472 | 6,3285 | 0,3290 | 1,6007 |
| **13** | 0,8328 | 0,3672 | 0,1599 | 0,1691 | 0,5383 | 0,3458 | 0,1235 | 0,7072 | 0,4977 | 0,3113 | 2,0880 | 1,9100 | 0,2706 | 0,2109 |
| **14** | 0,6770 | 0,3162 | 0,3468 | 0,7593 | 1,1165 | 0,3436 | 0,2631 | 1,5351 | 0,5106 | 1,3269 | 4,1520 | 4,0457 | 0,2263 | 0,5244 |
| **15** | 1,0000 | 0,6132 | 0,3250 | 0,6490 | 0,8610 | 0,2812 | 0,2375 | 0,5122 | 0,9393 | 1,0734 | 1,5727 | 3,3751 | 0,4717 | 0,4235 |
| **16** | 1,6308 | 1,0000 | 1,6868 | 0,7597 | 1,9684 | 1,5870 | 0,5012 | 2,8962 | 0,9933 | 2,5892 | 3,3175 | 8,2359 | 0,5226 | 2,3127 |
| **17** | 3,0771 | 0,5929 | 1,0000 | 1,2724 | 1,4472 | 0,4799 | 0,4689 | 2,5420 | 1,7817 | 1,0455 | 2,7409 | 6,9876 | 0,4224 | 0,7793 |
| **18** | 1,5409 | 1,3163 | 0,7859 | 1,0000 | 4,2419 | 0,4958 | 0,5180 | 2,4067 | 1,8834 | 2,5040 | 3,5927 | 7,5718 | 0,9872 | 1,9976 |
| **19** | 1,1615 | 0,5080 | 0,6910 | 0,2357 | 1,0000 | 0,4686 | 0,1309 | 0,8676 | 0,6693 | 0,3304 | 0,9509 | 0,9479 | 0,1592 | 0,7182 |
| **20** | 3,5558 | 0,6301 | 2,0836 | 2,0170 | 2,1340 | 1,0000 | 1,1865 | 1,3101 | 1,8350 | 1,2397 | 8,0958 | 3,8187 | 0,4079 | 2,2163 |
| **21** | 4,2098 | 1,9954 | 2,1327 | 1,9307 | 7,6370 | 0,8428 | 1,0000 | 1,2747 | 2,5656 | 3,0537 | 9,6686 | 9,4506 | 0,5502 | 1,2310 |
| **22** | 1,9522 | 0,3453 | 0,3934 | 0,4155 | 1,1525 | 0,7633 | 0,7845 | 1,0000 | 0,5039 | 0,4679 | 2,0340 | 1,6299 | 0,5501 | 0,4415 |
| **23** | 1,0647 | 1,0067 | 0,5612 | 0,5310 | 1,4941 | 0,5450 | 0,3898 | 1,9844 | 1,0000 | 0,7398 | 2,6736 | 5,4301 | 0,3053 | 1,5523 |
| **24** | 0,9316 | 0,3862 | 0,9564 | 0,3994 | 3,0268 | 0,8066 | 0,3275 | 2,1371 | 1,3517 | 1,0000 | 1,8535 | 4,3339 | 0,2603 | 1,2313 |
| **25** | 0,6359 | 0,3014 | 0,3648 | 0,2783 | 1,0516 | 0,1235 | 0,1034 | 0,4917 | 0,3740 | 0,5395 | 1,0000 | 1,3462 | 0,0698 | 0,1395 |
| **26** | 0,2963 | 0,1214 | 0,1431 | 0,1321 | 1,0549 | 0,2619 | 0,1058 | 0,6135 | 0,1842 | 0,2307 | 0,7428 | 1,0000 | 0,0868 | 0,4054 |
| **27** | 2,1201 | 1,9135 | 2,3676 | 1,0130 | 6,2832 | 2,4519 | 1,8176 | 1,8180 | 3,2752 | 3,8424 | 14,3199 | 11,5256 | 1,0000 | 2,9121 |
| **28** | 2,3611 | 0,4324 | 1,2833 | 0,5006 | 1,3923 | 0,4512 | 0,8123 | 2,2650 | 0,6442 | 0,8121 | 7,1667 | 2,4665 | 0,3434 | 1,0000 |
| **29** | 1,8832 | 1,6719 | 2,6629 | 2,3899 | 2,5912 | 2,0118 | 0,6484 | 3,3962 | 2,7582 | 3,1560 | 13,9606 | 5,1692 | 1,6167 | 1,3626 |
| **30** | 0,6165 | 0,2771 | 0,2943 | 0,6130 | 1,9480 | 0,6637 | 0,2141 | 0,5518 | 0,3872 | 1,3283 | 4,0896 | 1,5913 | 0,5154 | 0,9703 |
| **31** | 0,9748 | 1,1752 | 1,0669 | 0,5202 | 3,4398 | 1,0815 | 0,3853 | 0,9639 | 1,6571 | 0,8885 | 5,6367 | 5,5071 | 0,3790 | 0,6609 |
| **32** | 1,7308 | 1,4908 | 1,7414 | 0,8493 | 2,2987 | 2,1041 | 1,7166 | 1,4177 | 1,0270 | 2,9353 | 11,0462 | 4,2200 | 0,5859 | 1,2376 |
| **33** | 4,4690 | 1,8761 | 2,6394 | 2,2211 | 3,0263 | 2,0651 | 1,6666 | 1,7159 | 2,7078 | 1,6079 | 5,3997 | 10,8520 | 0,6771 | 3,4740 |
| **34** | 0,3799 | 0,0639 | 0,1874 | 0,1466 | 0,2162 | 0,0674 | 0,0495 | 0,2959 | 0,0972 | 0,1197 | 0,8654 | 0,3893 | 0,1396 | 0,2512 |
| **35** | 1,7716 | 1,9834 | 1,9828 | 1,8933 | 2,2712 | 0,7520 | 0,5949 | 1,4547 | 2,4913 | 2,7708 | 4,4465 | 4,4858 | 0,5693 | 1,1783 |
| **36** | 2,8989 | 1,2949 | 0,7598 | 0,7127 | 5,3830 | 1,6590 | 0,5102 | 3,5079 | 1,8665 | 2,8624 | 7,7647 | 8,0890 | 0,5341 | 2,4053 |
| **37** | 1,7025 | 1,4667 | 1,8759 | 0,8355 | 2,5595 | 0,7680 | 1,5178 | 3,2787 | 2,6558 | 2,8142 | 9,1354 | 10,0716 | 1,2427 | 2,3867 |
| **38** | 5,3454 | 2,1037 | 1,0059 | 2,4768 | 6,4849 | 2,0053 | 0,7443 | 1,7747 | 1,2067 | 4,2110 | 11,3174 | 12,5038 | 0,7236 | 1,3104 |
| **39** | 2,9803 | 1,2603 | 0,5435 | 1,3442 | 4,0365 | 0,5462 | 0,9325 | 0,9424 | 0,7394 | 0,9053 | 2,8551 | 6,6521 | 0,3624 | 2,0984 |
| **40** | 0,9800 | 0,8976 | 1,2881 | 1,2428 | 3,6479 | 0,4515 | 0,9571 | 1,9189 | 0,6492 | 2,1747 | 2,7585 | 6,5016 | 0,3312 | 1,6075 |
| **41** | 1,1233 | 0,4441 | 1,4673 | 1,3543 | 1,4566 | 1,3174 | 0,8450 | 0,9823 | 1,9389 | 2,4741 | 2,8985 | 2,8625 | 0,3728 | 0,7931 |

| № / № | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **42** | 4,2181 | 0,7464 | 1,8945 | 1,9830 | 5,1572 | 1,8919 | 1,5523 | 1,5062 | 1,1702 | 3,6699 | 4,9302 | 12,5034 | 1,2753 | 3,0363 |
| **43** | 1,5512 | 0,6796 | 0,7339 | 1,9856 | 2,2153 | 0,7063 | 1,4500 | 1,4445 | 1,0062 | 2,5471 | 8,8388 | 3,6347 | 1,0929 | 2,5878 |
| **44** | 3,8441 | 0,6605 | 0,6967 | 1,4970 | 4,1961 | 1,5363 | 1,1978 | 1,2007 | 2,2602 | 1,2210 | 9,6111 | 3,5372 | 1,2796 | 1,0402 |
| **45** | 0,8130 | 0,7501 | 0,3929 | 0,8285 | 1,1086 | 0,8274 | 0,7146 | 0,7003 | 1,3426 | 1,8440 | 5,0524 | 2,0626 | 0,2782 | 1,5919 |
| **46** | 2,0606 | 1,9297 | 0,9871 | 0,9314 | 2,7172 | 0,8713 | 1,9000 | 4,0683 | 1,3011 | 4,3126 | 5,2522 | 4,7849 | 1,4500 | 1,4779 |
| **47** | 1,3370 | 0,2783 | 0,6853 | 0,2958 | 2,0919 | 0,5999 | 0,2209 | 0,5709 | 0,3744 | 0,5198 | 4,0566 | 3,7606 | 0,4837 | 0,9231 |
| **48** | 3,7229 | 0,7462 | 0,9167 | 2,0313 | 2,5735 | 0,8263 | 0,6694 | 3,7624 | 2,7667 | 1,4552 | 4,9350 | 10,6808 | 1,2792 | 2,5611 |
| **49** | 1,4479 | 0,2618 | 0,3212 | 0,7335 | 2,3282 | 0,7093 | 0,2048 | 1,2647 | 0,3656 | 1,2749 | 3,2581 | 1,4099 | 0,2207 | 1,0058 |
| **50** | 0,4109 | 0,0634 | 0,0760 | 0,0676 | 0,2212 | 0,0641 | 0,0575 | 0,3509 | 0,0942 | 0,1263 | 1,0552 | 0,9356 | 0,1433 | 0,2065 |
| **51** | 4,6061 | 2,2534 | 2,2247 | 2,6142 | 7,6826 | 0,9734 | 0,7103 | 1,7925 | 1,3329 | 3,9456 | 11,6156 | 5,3838 | 0,7327 | 3,5101 |
| **52** | 4,3619 | 0,7151 | 2,1908 | 2,1390 | 5,3831 | 1,9764 | 0,5954 | 1,4148 | 1,0745 | 1,3211 | 11,6556 | 4,2136 | 1,2596 | 2,6853 |
| **53** | 1,3521 | 0,6055 | 1,4753 | 1,3898 | 4,4019 | 1,6955 | 1,1377 | 1,2621 | 1,8344 | 2,5909 | 8,9887 | 3,6925 | 0,4682 | 0,9066 |
| **54** | 1,0666 | 0,5011 | 0,5597 | 0,5794 | 1,6345 | 1,3726 | 0,4082 | 2,3409 | 0,7330 | 2,5093 | 2,9396 | 7,0483 | 0,4156 | 0,7432 |
| **55** | 1,8626 | 1,8228 | 2,2506 | 0,8736 | 2,5497 | 0,8287 | 0,6083 | 1,6018 | 1,1734 | 1,5578 | 9,9910 | 10,3906 | 1,2779 | 1,2411 |
| **56** | 0,4928 | 0,1992 | 0,2425 | 0,3053 | 0,6866 | 0,0967 | 0,2145 | 0,2106 | 0,3484 | 0,3944 | 0,6343 | 1,1945 | 0,1805 | 0,1667 |
| **57** | 2,1009 | 0,8782 | 1,0172 | 2,7664 | 2,8543 | 1,9732 | 1,9181 | 1,7816 | 3,4646 | 1,7419 | 5,4621 | 5,4107 | 1,8178 | 1,5463 |
| **58** | 4,4431 | 0,9248 | 0,9702 | 2,0527 | 2,7768 | 0,9680 | 0,7112 | 1,8480 | 2,7979 | 1,6705 | 12,5461 | 12,6344 | 0,7169 | 3,1569 |
| **59** | 2,5616 | 0,9361 | 0,4932 | 0,4368 | 1,3295 | 0,4593 | 0,6999 | 0,8014 | 1,2641 | 0,7282 | 2,4454 | 2,2828 | 0,7756 | 1,6351 |
| **60** | 0,2380 | 0,0971 | 0,0505 | 0,1244 | 0,1391 | 0,1271 | 0,0385 | 0,2144 | 0,1399 | 0,0802 | 0,2532 | 0,6277 | 0,0767 | 0,0747 |
| **61** | 0,2985 | 0,2928 | 0,4182 | 0,3413 | 0,8677 | 0,2857 | 0,1040 | 0,5430 | 0,3935 | 0,6323 | 1,9782 | 0,7748 | 0,2311 | 0,2107 |
| **62** | 4,8906 | 0,9116 | 2,5692 | 0,9470 | 6,5163 | 0,9334 | 0,7773 | 1,7596 | 2,7195 | 4,5932 | 5,5370 | 13,4736 | 1,7781 | 1,3842 |
| **63** | 1,5528 | 0,3322 | 0,3610 | 0,8929 | 2,1551 | 0,7197 | 0,5387 | 1,5787 | 0,4364 | 1,3385 | 1,9416 | 1,7993 | 0,6792 | 0,5023 |
| **64** | 1,2394 | 0,5270 | 1,2786 | 1,4176 | 4,7236 | 1,3388 | 1,0687 | 1,1220 | 1,5840 | 1,0094 | 3,0761 | 6,3973 | 0,4079 | 1,8635 |
| **65** | 4,9181 | 1,8738 | 0,8677 | 1,9711 | 2,4861 | 2,1287 | 0,6513 | 4,2138 | 2,4092 | 3,5403 | 11,1106 | 4,2491 | 0,6607 | 1,2029 |
| **66** | 0,2032 | 0,2477 | 0,1076 | 0,1036 | 0,6357 | 0,0954 | 0,1982 | 0,1874 | 0,3250 | 0,4285 | 0,5486 | 0,5084 | 0,1548 | 0,1579 |
| **67** | 0,6256 | 0,2741 | 0,3048 | 0,2749 | 1,8059 | 0,5793 | 0,2037 | 0,5527 | 0,8188 | 1,0250 | 1,6960 | 3,8827 | 0,2125 | 0,8434 |
| **68** | 2,0266 | 1,9134 | 2,3052 | 1,0411 | 7,5351 | 2,5891 | 1,8555 | 4,8566 | 3,0442 | 3,8607 | 5,6035 | 5,0716 | 1,5116 | 1,4627 |
| **69** | 1,0067 | 1,0729 | 0,5003 | 1,3298 | 1,6038 | 0,5161 | 0,8163 | 0,9407 | 0,6893 | 0,8974 | 7,1176 | 6,7116 | 0,8500 | 0,7612 |
| **70** | 3,8994 | 0,7764 | 1,8372 | 2,0995 | 2,5603 | 0,7657 | 1,4130 | 1,6084 | 1,0792 | 1,5313 | 5,0778 | 4,2721 | 1,4408 | 1,3297 |

***Табл. 5. Емпірична матриця попарних порівнянь (стовпці 29-42)***

| № / № | 29 | 30 | 31 | 32 | 33 | 34 | 35 | 36 | 37 | 38 | 39 | 40 | 41 | 42 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **1** | 0,6830 | 4,5459 | 2,9721 | 0,7819 | 1,4589 | 20,0400 | 0,8487 | 1,8772 | 0,8387 | 1,4041 | 2,4934 | 1,2667 | 1,1702 | 1,5482 |
| **2** | 0,6262 | 0,8972 | 0,4549 | 0,6591 | 0,6080 | 8,1896 | 0,6889 | 0,8137 | 0,7204 | 0,2444 | 1,0844 | 1,1622 | 0,5284 | 0,3012 |
| **3** | 1,4550 | 1,8122 | 1,1299 | 1,5301 | 1,2536 | 20,4419 | 0,7675 | 0,8067 | 0,7282 | 1,3941 | 2,4328 | 1,0694 | 1,1034 | 1,5637 |
| **4** | 0,8690 | 2,8992 | 0,7988 | 0,4098 | 0,7826 | 11,8775 | 1,0358 | 0,4359 | 1,0181 | 0,8991 | 1,5570 | 1,6381 | 1,6129 | 0,4574 |
| **5** | 0,6481 | 1,8942 | 2,7237 | 1,4862 | 1,3197 | 19,3060 | 0,6980 | 0,7270 | 0,6049 | 1,2972 | 2,3534 | 1,0310 | 2,4292 | 0,6371 |
| **6** | 0,7392 | 1,8769 | 2,8876 | 1,6742 | 1,4687 | 7,4162 | 1,6665 | 0,8472 | 1,7549 | 0,6646 | 1,1043 | 1,0206 | 1,2405 | 1,7009 |
| **7** | 0,7425 | 2,3811 | 1,3326 | 0,7560 | 0,2404 | 8,9543 | 0,7891 | 0,9174 | 0,3323 | 0,3149 | 1,1616 | 1,3805 | 1,1548 | 0,8134 |
| **8** | 0,4980 | 3,3934 | 0,9282 | 0,5822 | 0,9667 | 6,4809 | 0,5204 | 0,5947 | 0,4476 | 1,0064 | 0,7831 | 1,9855 | 0,7268 | 0,5134 |
| **9** | 0,4508 | 1,4554 | 0,8173 | 0,5222 | 0,4002 | 5,9114 | 0,4588 | 1,3210 | 1,0590 | 0,9605 | 1,6279 | 1,8560 | 1,6455 | 1,1106 |
| **10** | 0,6581 | 0,9482 | 1,3742 | 0,7569 | 0,2435 | 3,5672 | 0,3141 | 0,4482 | 0,7457 | 0,3143 | 1,2309 | 1,3244 | 1,1783 | 0,3551 |
| **11** | 0,3897 | 2,4680 | 1,4735 | 0,3390 | 0,6955 | 4,8277 | 0,9516 | 0,9840 | 0,4096 | 0,2968 | 1,4496 | 0,7464 | 0,6361 | 0,4170 |
| **12** | 0,9361 | 2,9226 | 1,7479 | 0,3599 | 0,3280 | 5,6144 | 1,0686 | 0,5103 | 0,4171 | 0,3328 | 0,7016 | 0,6217 | 0,7143 | 0,3983 |
| **13** | 0,1493 | 0,3551 | 0,2289 | 0,1571 | 0,1262 | 1,7203 | 0,1592 | 0,4156 | 0,1399 | 0,2922 | 0,2170 | 0,5921 | 0,2360 | 0,1500 |
| **14** | 0,5890 | 0,7530 | 0,4905 | 0,7009 | 0,2376 | 3,1181 | 0,7406 | 0,7851 | 0,2828 | 0,2270 | 0,4248 | 0,5654 | 1,0064 | 0,2667 |
| **15** | 0,5310 | 1,6221 | 1,0259 | 0,5778 | 0,2238 | 2,6321 | 0,5645 | 0,3450 | 0,5874 | 0,1871 | 0,3355 | 1,0204 | 0,8902 | 0,2371 |
| **16** | 0,5981 | 3,6090 | 0,8509 | 0,6708 | 0,5330 | 15,6599 | 0,5042 | 0,7723 | 0,6818 | 0,4753 | 0,7934 | 1,1141 | 2,2517 | 1,3398 |
| **17** | 0,3755 | 3,3980 | 0,9373 | 0,5742 | 0,3789 | 5,3373 | 0,5043 | 1,3161 | 0,5331 | 0,9942 | 1,8398 | 0,7763 | 0,6815 | 0,5278 |
| **18** | 0,4184 | 1,6314 | 1,9224 | 1,1775 | 0,4502 | 6,8199 | 0,5282 | 1,4032 | 1,1968 | 0,4037 | 0,7439 | 0,8047 | 0,7384 | 0,5043 |
| **19** | 0,3859 | 0,5134 | 0,2907 | 0,4350 | 0,3304 | 4,6259 | 0,4403 | 0,1858 | 0,3907 | 0,1542 | 0,2477 | 0,2741 | 0,6865 | 0,1939 |
| **20** | 0,4971 | 1,5066 | 0,9246 | 0,4753 | 0,4842 | 14,8301 | 1,3298 | 0,6028 | 1,3021 | 0,4987 | 1,8308 | 2,2148 | 0,7591 | 0,5286 |
| **21** | 1,5422 | 4,6708 | 2,5952 | 0,5825 | 0,6000 | 20,1825 | 1,6810 | 1,9599 | 0,6588 | 1,3436 | 1,0724 | 1,0448 | 1,1834 | 0,6442 |
| **22** | 0,2944 | 1,8122 | 1,0374 | 0,7054 | 0,5828 | 3,3798 | 0,6874 | 0,2851 | 0,3050 | 0,5635 | 1,0611 | 0,5211 | 1,0181 | 0,6639 |
| **23** | 0,3626 | 2,5829 | 0,6035 | 0,9737 | 0,3693 | 10,2887 | 0,4014 | 0,5358 | 0,3765 | 0,8287 | 1,3524 | 1,5403 | 0,5157 | 0,8546 |
| **24** | 0,3169 | 0,7529 | 1,1255 | 0,3407 | 0,6219 | 8,3567 | 0,3609 | 0,3494 | 0,3553 | 0,2375 | 1,1046 | 0,4598 | 0,4042 | 0,2725 |
| **25** | 0,0716 | 0,2445 | 0,1774 | 0,0905 | 0,1852 | 1,1555 | 0,2249 | 0,1288 | 0,1095 | 0,0884 | 0,3502 | 0,3625 | 0,3450 | 0,2028 |
| **26** | 0,1935 | 0,6284 | 0,1816 | 0,2370 | 0,0921 | 2,5690 | 0,2229 | 0,1236 | 0,0993 | 0,0800 | 0,1503 | 0,1538 | 0,3493 | 0,0800 |
| **27** | 0,6185 | 1,9403 | 2,6389 | 1,7067 | 1,4769 | 7,1646 | 1,7566 | 1,8722 | 0,8047 | 1,3819 | 2,7594 | 3,0189 | 2,6822 | 0,7842 |
| **28** | 0,7339 | 1,0306 | 1,5132 | 0,8080 | 0,2879 | 3,9814 | 0,8487 | 0,4157 | 0,4190 | 0,7631 | 0,4765 | 0,6221 | 1,2608 | 0,3293 |

| № / № | 29 | 30 | 31 | 32 | 33 | 34 | 35 | 36 | 37 | 38 | 39 | 40 | 41 | 42 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **29** | 1,0000 | 4,4920 | 1,1813 | 1,5214 | 1,2715 | 8,4261 | 0,6406 | 1,8894 | 0,7788 | 0,5453 | 2,3937 | 2,8568 | 1,0841 | 0,6937 |
| **30** | 0,2226 | 1,0000 | 0,2965 | 0,4587 | 0,1913 | 2,8894 | 0,1851 | 0,2498 | 0,5154 | 0,4389 | 0,7380 | 0,8992 | 0,3775 | 0,2131 |
| **31** | 0,8465 | 3,3725 | 1,0000 | 0,8604 | 0,7340 | 3,6410 | 0,3807 | 0,9180 | 0,7837 | 0,2643 | 0,4798 | 0,6554 | 1,2374 | 0,8252 |
| **32** | 0,6573 | 2,1803 | 1,1622 | 1,0000 | 0,5357 | 16,0270 | 1,5050 | 1,7115 | 1,3259 | 1,1271 | 2,2256 | 0,9137 | 1,0640 | 1,3124 |
| **33** | 0,7865 | 5,2265 | 1,3625 | 1,8666 | 1,0000 | 19,4770 | 0,8439 | 2,0935 | 1,5889 | 0,6256 | 2,5324 | 2,7840 | 1,2862 | 1,6567 |
| **34** | 0,1187 | 0,3461 | 0,2746 | 0,0624 | 0,0513 | 1,0000 | 0,1267 | 0,0567 | 0,0513 | 0,0362 | 0,1849 | 0,0888 | 0,1932 | 0,1176 |
| **35** | 1,5609 | 5,4032 | 2,6269 | 0,6644 | 1,1849 | 7,8923 | 1,0000 | 0,7395 | 0,5610 | 1,1001 | 2,2621 | 2,2648 | 2,2926 | 0,5701 |
| **36** | 0,5293 | 4,0025 | 1,0893 | 0,5843 | 0,4777 | 17,6261 | 1,3522 | 1,0000 | 1,2779 | 1,0659 | 0,7400 | 1,8881 | 1,8433 | 1,2166 |
| **37** | 1,2840 | 1,9403 | 1,2760 | 0,7542 | 0,6294 | 19,4958 | 1,7825 | 0,7825 | 1,0000 | 1,2227 | 0,7876 | 2,2290 | 2,2474 | 0,5024 |
| **38** | 1,8338 | 2,2786 | 3,7837 | 0,8873 | 1,5984 | 27,6523 | 0,9090 | 0,9382 | 0,8179 | 1,0000 | 1,1380 | 1,0220 | 2,5648 | 0,7042 |
| **39** | 0,4178 | 1,3551 | 2,0844 | 0,4493 | 0,3949 | 5,4082 | 0,4421 | 1,3513 | 1,2696 | 0,8787 | 1,0000 | 1,6355 | 1,3440 | 0,3739 |
| **40** | 0,3500 | 1,1121 | 1,5258 | 1,0944 | 0,3592 | 11,2558 | 0,4415 | 0,5296 | 0,4486 | 0,9784 | 0,6114 | 1,0000 | 0,6251 | 0,3877 |
| **41** | 0,9224 | 2,6490 | 0,8082 | 0,9399 | 0,7775 | 5,1750 | 0,4362 | 0,5425 | 0,4450 | 0,3899 | 0,7440 | 1,5999 | 1,0000 | 0,8039 |
| **42** | 1,4415 | 4,6928 | 1,2119 | 0,7619 | 0,6036 | 8,5016 | 1,7541 | 0,8220 | 1,9905 | 1,4201 | 2,6748 | 2,5793 | 1,2440 | 1,0000 |
| **43** | 1,2356 | 1,8610 | 1,0511 | 1,6251 | 1,2616 | 21,0966 | 1,3373 | 0,7260 | 0,6394 | 0,5622 | 2,1126 | 2,7865 | 0,9552 | 1,6661 |
| **44** | 0,5149 | 1,6507 | 2,0438 | 0,6023 | 0,5330 | 14,6255 | 0,6090 | 0,7523 | 0,6516 | 1,1163 | 0,9479 | 1,0207 | 1,8818 | 0,5598 |
| **45** | 0,3122 | 0,9482 | 1,5664 | 0,8076 | 0,2742 | 3,8569 | 0,3659 | 0,4158 | 0,3678 | 0,7190 | 0,5220 | 0,6372 | 0,5355 | 0,3570 |
| **46** | 0,7512 | 2,4046 | 2,9004 | 0,8631 | 0,7238 | 10,2830 | 0,9124 | 1,0058 | 2,2760 | 0,7122 | 1,3250 | 3,7367 | 2,8839 | 1,9709 |
| **47** | 0,2330 | 2,0701 | 1,0418 | 0,6008 | 0,4481 | 6,3502 | 0,2537 | 0,6673 | 0,2813 | 0,2135 | 1,0196 | 1,0987 | 0,3981 | 0,2441 |
| **48** | 1,6370 | 4,3635 | 1,2245 | 0,7656 | 0,6172 | 8,5714 | 1,9598 | 0,8324 | 0,7549 | 0,6566 | 2,7163 | 1,2329 | 1,2426 | 0,6692 |
| **49** | 0,2098 | 0,6674 | 1,0996 | 0,2376 | 0,2225 | 8,2162 | 0,5510 | 0,2832 | 0,5557 | 0,4922 | 0,3597 | 0,9985 | 0,3868 | 0,2435 |
| **50** | 0,1327 | 0,4021 | 0,2373 | 0,0586 | 0,0517 | 1,8026 | 0,0630 | 0,1462 | 0,1403 | 0,0502 | 0,0894 | 0,2173 | 0,2400 | 0,1327 |
| **51** | 0,7425 | 2,3824 | 3,2660 | 2,1503 | 0,6917 | 9,8251 | 0,9060 | 0,9845 | 1,8561 | 1,6858 | 2,9832 | 1,3620 | 3,2029 | 0,7993 |
| **52** | 0,6154 | 5,3246 | 2,9516 | 0,6548 | 0,5921 | 8,3606 | 1,4671 | 0,8066 | 0,6476 | 0,5868 | 1,0125 | 1,1350 | 2,1779 | 0,6303 |
| **53** | 1,0464 | 3,4324 | 0,9707 | 1,4626 | 1,0516 | 6,7228 | 0,5935 | 0,6852 | 0,5663 | 0,5257 | 2,3495 | 2,2103 | 2,1231 | 1,3112 |
| **54** | 1,0885 | 3,4639 | 0,8195 | 1,0435 | 0,4054 | 13,0938 | 1,1784 | 0,5785 | 1,2870 | 0,4268 | 1,6287 | 0,7895 | 2,0171 | 0,4843 |
| **55** | 1,3986 | 2,1601 | 1,1850 | 1,5939 | 1,4106 | 8,5696 | 0,7408 | 0,8786 | 0,7629 | 1,7544 | 2,9367 | 2,7203 | 1,1257 | 1,7480 |
| **56** | 0,0864 | 0,2861 | 0,3513 | 0,0958 | 0,0789 | 1,1320 | 0,0903 | 0,2612 | 0,1973 | 0,1756 | 0,3811 | 0,1610 | 0,1545 | 0,1865 |
| **57** | 1,7838 | 6,0216 | 1,4661 | 2,2967 | 0,6942 | 10,8444 | 2,3923 | 2,2304 | 0,9297 | 0,7642 | 2,9057 | 3,1041 | 1,4448 | 1,7605 |
| **58** | 0,8052 | 2,2714 | 3,0618 | 0,8696 | 1,5080 | 20,7756 | 0,9063 | 2,0065 | 0,8948 | 1,5847 | 1,2408 | 1,3831 | 1,3900 | 0,8661 |
| **59** | 0,3481 | 1,1981 | 0,6993 | 0,8815 | 0,3360 | 4,7548 | 0,9747 | 1,1123 | 0,8557 | 0,6753 | 0,6422 | 1,3657 | 0,6000 | 0,3907 |
| **60** | 0,0824 | 0,2788 | 0,0690 | 0,0872 | 0,0887 | 0,5115 | 0,1047 | 0,0467 | 0,0431 | 0,0372 | 0,1355 | 0,0773 | 0,1660 | 0,0828 |
| **61** | 0,1141 | 0,7808 | 0,5565 | 0,2750 | 0,1000 | 3,0399 | 0,2974 | 0,1555 | 0,1304 | 0,1019 | 0,1819 | 0,5633 | 0,2051 | 0,3335 |
| **62** | 0,7765 | 6,6394 | 1,4451 | 0,7998 | 0,7390 | 22,7376 | 0,8003 | 1,0285 | 2,2204 | 1,9290 | 1,2771 | 3,7609 | 3,5099 | 0,7622 |
| **63** | 0,6491 | 0,9315 | 1,4334 | 0,8172 | 0,5416 | 9,3132 | 0,3170 | 0,3499 | 0,6277 | 0,2700 | 0,9709 | 1,0489 | 1,0451 | 0,8105 |
| **64** | 0,4398 | 1,4523 | 1,8448 | 1,3752 | 1,0200 | 13,1566 | 1,4063 | 0,5887 | 1,1127 | 1,0503 | 2,1221 | 1,9114 | 2,0282 | 0,4777 |
| **65** | 0,6794 | 2,0414 | 1,1771 | 1,6125 | 0,6620 | 9,1965 | 1,6749 | 0,8332 | 0,7401 | 0,6302 | 1,1208 | 1,2651 | 1,1896 | 0,7121 |
| **66** | 0,0844 | 0,2612 | 0,1419 | 0,0841 | 0,1646 | 2,7889 | 0,0964 | 0,0999 | 0,2067 | 0,0795 | 0,2931 | 0,1541 | 0,1406 | 0,0904 |
| **67** | 0,2366 | 1,5424 | 0,9091 | 0,2714 | 0,2204 | 6,1640 | 0,2650 | 0,6131 | 0,2527 | 0,2007 | 0,8422 | 0,8807 | 0,3932 | 0,2405 |
| **68** | 1,8810 | 2,3212 | 3,2237 | 0,8548 | 0,6860 | 10,4490 | 0,9297 | 2,5925 | 1,7715 | 1,8222 | 1,2771 | 1,5726 | 1,3256 | 0,8521 |
| **69** | 0,8080 | 1,2516 | 0,7317 | 0,4789 | 0,7531 | 5,1065 | 1,2401 | 0,5390 | 1,1323 | 0,7918 | 0,7320 | 0,7185 | 1,7466 | 0,4570 |
| **70** | 0,6984 | 5,5659 | 3,2793 | 1,5635 | 1,3270 | 9,1653 | 0,7119 | 1,7853 | 0,7519 | 0,6424 | 2,7976 | 3,1283 | 1,1475 | 1,8963 |

***Табл. 6. Емпірична матриця попарних порівнянь (стовпці 43-56)***

| № / № | 43 | 44 | 45 | 46 | 47 | 48 | 49 | 50 | 51 | 52 | 53 | 54 | 55 | 56 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **1** | 1,8782 | 0,7459 | 1,6830 | 1,5273 | 2,1897 | 0,7233 | 1,8656 | 20,9522 | 1,4423 | 0,8669 | 0,7505 | 2,4544 | 1,6468 | 13,4102 |
| **2** | 0,8170 | 0,2908 | 0,5560 | 0,2788 | 0,6736 | 0,2743 | 1,8746 | 7,3735 | 0,2714 | 0,6719 | 0,8132 | 1,0174 | 0,6641 | 5,4284 |
| **3** | 0,6996 | 1,7871 | 3,0966 | 0,6042 | 4,1286 | 0,6753 | 4,6240 | 18,2522 | 1,4098 | 0,7677 | 2,0198 | 2,3119 | 0,7065 | 12,4172 |
| **4** | 0,5563 | 1,2376 | 0,8289 | 0,8417 | 1,2830 | 0,4035 | 2,9918 | 12,1558 | 0,3653 | 0,4906 | 1,1580 | 0,6046 | 0,3753 | 8,1180 |
| **5** | 0,7811 | 0,7008 | 1,1637 | 1,2287 | 3,8845 | 1,3918 | 4,2535 | 18,5888 | 1,1936 | 0,7102 | 0,8753 | 0,9925 | 0,6442 | 11,2015 |
| **6** | 1,8533 | 2,0639 | 1,2706 | 1,4206 | 1,8160 | 0,6268 | 1,7169 | 6,9549 | 0,6373 | 0,8292 | 2,0324 | 2,5456 | 1,6038 | 6,3467 |
| **7** | 0,9032 | 0,3595 | 1,5778 | 0,3085 | 2,3180 | 0,3451 | 2,1675 | 9,4899 | 0,7100 | 0,8494 | 0,9741 | 0,5050 | 0,7324 | 2,8062 |
| **8** | 0,5820 | 1,5218 | 2,4600 | 0,5052 | 3,2596 | 1,1451 | 3,4350 | 14,4217 | 0,4930 | 1,2573 | 1,5577 | 0,7157 | 1,1912 | 9,0558 |
| **9** | 1,2414 | 1,2473 | 2,2552 | 0,9065 | 3,0912 | 1,0230 | 3,2679 | 5,1537 | 0,3725 | 1,1404 | 1,3806 | 0,6924 | 0,4199 | 3,3202 |
| **10** | 0,8989 | 0,3718 | 0,7212 | 0,6628 | 2,1591 | 0,7320 | 2,1436 | 8,3428 | 0,6642 | 0,8251 | 0,3918 | 1,0821 | 0,2706 | 5,6130 |
| **11** | 0,9815 | 0,4475 | 1,6978 | 0,7197 | 0,9214 | 0,3406 | 1,2567 | 10,6868 | 0,2801 | 0,9124 | 1,1624 | 0,5606 | 0,8369 | 6,9540 |
| **12** | 1,0538 | 1,1749 | 1,8910 | 0,8949 | 1,0949 | 0,4058 | 2,9442 | 4,6543 | 0,3485 | 0,4650 | 0,5168 | 1,3981 | 0,8920 | 3,4857 |
| **13** | 0,3899 | 0,1952 | 0,2538 | 0,2826 | 0,8470 | 0,3060 | 0,4117 | 3,7499 | 0,1133 | 0,3490 | 0,4044 | 0,5018 | 0,3203 | 2,7135 |
| **14** | 0,3230 | 0,8256 | 1,3877 | 0,6307 | 1,9274 | 0,6544 | 2,0728 | 3,1330 | 0,5477 | 0,2654 | 0,8064 | 0,5074 | 0,2473 | 5,3768 |
| **15** | 0,6446 | 0,2601 | 1,2300 | 0,4853 | 0,7480 | 0,2686 | 0,6907 | 2,4335 | 0,2171 | 0,2293 | 0,7396 | 0,9376 | 0,5369 | 2,0291 |

| № / № | 43 | 44 | 45 | 46 | 47 | 48 | 49 | 50 | 51 | 52 | 53 | 54 | 55 | 56 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **16** | 1,4715 | 1,5140 | 1,3332 | 0,5182 | 3,5931 | 1,3402 | 3,8200 | 15,7779 | 0,4438 | 1,3985 | 1,6516 | 1,9957 | 0,5486 | 5,0201 |
| **17** | 1,3626 | 1,4352 | 2,5453 | 1,0131 | 1,4593 | 1,0908 | 3,1134 | 13,1652 | 0,4495 | 0,4565 | 0,6778 | 1,7868 | 0,4443 | 4,1245 |
| **18** | 0,5036 | 0,6680 | 1,2070 | 1,0737 | 3,3806 | 0,4923 | 1,3633 | 14,7994 | 0,3825 | 0,4675 | 0,7195 | 1,7258 | 1,1447 | 3,2751 |
| **19** | 0,4514 | 0,2383 | 0,9020 | 0,3680 | 0,4780 | 0,3886 | 0,4295 | 4,5202 | 0,1302 | 0,1858 | 0,2272 | 0,6118 | 0,3922 | 1,4565 |
| **20** | 1,4158 | 0,6509 | 1,2087 | 1,1478 | 1,6670 | 1,2103 | 1,4099 | 15,5983 | 1,0274 | 0,5060 | 0,5898 | 0,7286 | 1,2068 | 10,3371 |
| **21** | 0,6897 | 0,8349 | 1,3994 | 0,5263 | 4,5264 | 1,4939 | 4,8825 | 17,3861 | 1,4078 | 1,6794 | 0,8790 | 2,4498 | 1,6439 | 4,6614 |
| **22** | 0,6923 | 0,8328 | 1,4280 | 0,2458 | 1,7517 | 0,2658 | 0,7907 | 2,8498 | 0,5579 | 0,7068 | 0,7924 | 0,4272 | 0,6243 | 4,7493 |
| **23** | 0,9939 | 0,4424 | 0,7448 | 0,7686 | 2,6711 | 0,3614 | 2,7353 | 10,6137 | 0,7503 | 0,9307 | 0,5451 | 1,3643 | 0,8522 | 2,8704 |
| **24** | 0,3926 | 0,8190 | 0,5423 | 0,2319 | 1,9237 | 0,6872 | 0,7844 | 7,9204 | 0,2534 | 0,7569 | 0,3860 | 0,3985 | 0,6419 | 2,5354 |
| **25** | 0,1131 | 0,1040 | 0,1979 | 0,1904 | 0,2465 | 0,2026 | 0,3069 | 0,9477 | 0,0861 | 0,0858 | 0,1113 | 0,3402 | 0,1001 | 1,5764 |
| **26** | 0,2751 | 0,2827 | 0,4848 | 0,2090 | 0,2659 | 0,0936 | 0,7093 | 1,0688 | 0,1857 | 0,2373 | 0,2708 | 0,1419 | 0,0962 | 0,8371 |
| **27** | 0,9150 | 0,7815 | 3,5940 | 0,6897 | 2,0675 | 0,7817 | 4,5317 | 6,9769 | 1,3649 | 0,7939 | 2,1358 | 2,4061 | 0,7825 | 5,5403 |
| **28** | 0,3864 | 0,9613 | 0,6282 | 0,6766 | 1,0833 | 0,3905 | 0,9942 | 4,8426 | 0,2849 | 0,3724 | 1,1030 | 1,3456 | 0,8057 | 5,9988 |
| **29** | 0,8093 | 1,9422 | 3,2031 | 1,3313 | 4,2920 | 0,6109 | 4,7659 | 7,5375 | 1,3468 | 1,6249 | 0,9557 | 0,9187 | 0,7150 | 11,5704 |
| **30** | 0,5373 | 0,6058 | 1,0546 | 0,4159 | 0,4831 | 0,2292 | 1,4984 | 2,4870 | 0,4197 | 0,1878 | 0,2913 | 0,2887 | 0,4629 | 3,4956 |
| **31** | 0,9514 | 0,4893 | 0,6384 | 0,3448 | 0,9599 | 0,8167 | 0,9094 | 4,2145 | 0,3062 | 0,3388 | 1,0302 | 1,2203 | 0,8439 | 2,8469 |
| **32** | 0,6154 | 1,6604 | 1,2382 | 1,1586 | 1,6644 | 1,3061 | 4,2087 | 17,0695 | 0,4650 | 1,5271 | 0,6837 | 0,9583 | 0,6274 | 10,4399 |
| **33** | 0,7926 | 1,8761 | 3,6467 | 1,3816 | 2,2314 | 1,6203 | 4,4937 | 19,3472 | 1,4457 | 1,6890 | 0,9509 | 2,4666 | 0,7089 | 12,6707 |
| **34** | 0,0474 | 0,0684 | 0,2593 | 0,0972 | 0,1575 | 0,1167 | 0,1217 | 0,5547 | 0,1018 | 0,1196 | 0,1487 | 0,0764 | 0,1167 | 0,8834 |
| **35** | 0,7477 | 1,6420 | 2,7327 | 1,0960 | 3,9414 | 0,5103 | 1,8147 | 15,8636 | 1,1037 | 0,6816 | 1,6848 | 0,8486 | 1,3500 | 11,0754 |
| **36** | 1,3774 | 1,3293 | 2,4053 | 0,9943 | 1,4985 | 1,2013 | 3,5313 | 6,8401 | 1,0157 | 1,2397 | 1,4593 | 1,7288 | 1,1382 | 3,8292 |
| **37** | 1,5641 | 1,5347 | 2,7191 | 0,4394 | 3,5549 | 1,3247 | 1,7996 | 7,1282 | 0,5388 | 1,5442 | 1,7660 | 0,7770 | 1,3107 | 5,0681 |
| **38** | 1,7787 | 0,8958 | 1,3908 | 1,4041 | 4,6849 | 1,5229 | 2,0318 | 19,9207 | 0,5932 | 1,7042 | 1,9024 | 2,3431 | 0,5700 | 5,6949 |
| **39** | 0,4733 | 1,0549 | 1,9157 | 0,7547 | 0,9808 | 0,3681 | 2,7799 | 11,1893 | 0,3352 | 0,9876 | 0,4256 | 0,6140 | 0,3405 | 2,6243 |
| **40** | 0,3589 | 0,9797 | 1,5694 | 0,2676 | 0,9101 | 0,8111 | 1,0015 | 4,6012 | 0,7342 | 0,8811 | 0,4524 | 1,2666 | 0,3676 | 6,2130 |
| **41** | 1,0469 | 0,5314 | 1,8675 | 0,3467 | 2,5120 | 0,8048 | 2,5854 | 4,1668 | 0,3122 | 0,4592 | 0,4710 | 0,4958 | 0,8883 | 6,4745 |
| **42** | 0,6002 | 1,7863 | 2,8014 | 0,5074 | 4,0960 | 1,4943 | 4,1072 | 7,5376 | 1,2511 | 1,5864 | 0,7627 | 2,0648 | 0,5721 | 5,3621 |
| **43** | 1,0000 | 1,3949 | 2,7157 | 0,4413 | 3,6251 | 0,5422 | 1,5912 | 5,2637 | 1,1193 | 0,5153 | 1,5519 | 0,8075 | 0,5739 | 4,2869 |
| **44** | 0,7169 | 1,0000 | 1,0209 | 0,4715 | 3,3920 | 0,4055 | 3,4102 | 6,0943 | 0,3643 | 0,5689 | 0,5344 | 1,7787 | 1,1492 | 8,8044 |
| **45** | 0,3682 | 0,9795 | 1,0000 | 0,2287 | 1,9699 | 0,2932 | 0,7004 | 3,2810 | 0,5715 | 0,3350 | 0,3915 | 0,9951 | 0,2372 | 5,2929 |
| **46** | 2,2659 | 2,1210 | 4,3719 | 1,0000 | 2,1230 | 1,6748 | 4,9841 | 19,8784 | 1,3794 | 1,7337 | 1,9161 | 1,0033 | 1,6108 | 6,1362 |
| **47** | 0,2759 | 0,2948 | 0,5076 | 0,4710 | 1,0000 | 0,2295 | 0,6998 | 2,7531 | 0,4085 | 0,5456 | 0,2887 | 0,7258 | 0,1934 | 1,6230 |
| **48** | 1,8444 | 2,4660 | 3,4102 | 0,5971 | 4,3571 | 1,0000 | 4,2152 | 17,4952 | 1,3111 | 1,4221 | 0,7233 | 2,2161 | 1,4591 | 4,3519 |
| **49** | 0,6285 | 0,2932 | 1,4277 | 0,2006 | 1,4289 | 0,2372 | 1,0000 | 2,3642 | 0,1787 | 0,5098 | 0,3003 | 0,7552 | 0,4671 | 1,3966 |
| **50** | 0,1900 | 0,1641 | 0,3048 | 0,0503 | 0,3632 | 0,0572 | 0,4230 | 1,0000 | 0,1022 | 0,1281 | 0,0712 | 0,0797 | 0,1127 | 0,4172 |
| **51** | 0,8934 | 2,7447 | 1,7498 | 0,7250 | 2,4477 | 0,7627 | 5,5956 | 9,7803 | 1,0000 | 0,8659 | 2,1424 | 2,3980 | 0,5830 | 13,5877 |
| **52** | 1,9406 | 1,7577 | 2,9852 | 0,5768 | 1,8328 | 0,7032 | 1,9614 | 7,8045 | 1,1548 | 1,0000 | 0,7419 | 0,8783 | 0,5383 | 9,9695 |
| **53** | 0,6444 | 1,8711 | 2,5545 | 0,5219 | 3,4636 | 1,3825 | 3,3297 | 14,0456 | 0,4668 | 1,3479 | 1,0000 | 0,6942 | 1,1256 | 3,9904 |
| **54** | 1,2383 | 0,5622 | 1,0049 | 0,9967 | 1,3777 | 0,4513 | 1,3242 | 12,5499 | 0,4170 | 1,1385 | 1,4405 | 1,0000 | 0,3697 | 3,4449 |
| **55** | 1,7423 | 0,8702 | 4,2159 | 0,6208 | 5,1719 | 0,6853 | 2,1408 | 8,8761 | 1,7151 | 1,8579 | 0,8884 | 2,7047 | 1,0000 | 11,2910 |
| **56** | 0,2333 | 0,1136 | 0,1889 | 0,1630 | 0,6161 | 0,2298 | 0,7160 | 2,3968 | 0,0736 | 0,1003 | 0,2506 | 0,2903 | 0,0886 | 1,0000 |
| **57** | 2,2884 | 2,1492 | 3,7675 | 1,7673 | 2,4771 | 0,7769 | 5,3827 | 10,5131 | 1,4625 | 2,2066 | 1,0673 | 1,3910 | 0,7746 | 6,4798 |
| **58** | 2,2730 | 2,0177 | 1,7603 | 1,5573 | 6,3786 | 1,9969 | 5,1613 | 22,2095 | 1,4741 | 2,1398 | 2,8656 | 3,4356 | 0,8195 | 17,3815 |
| **59** | 0,8960 | 1,0194 | 0,8548 | 0,8870 | 2,3185 | 0,9732 | 1,2113 | 4,2828 | 0,3195 | 0,8398 | 1,1839 | 0,5936 | 0,3575 | 6,1085 |
| **60** | 0,0494 | 0,1391 | 0,2271 | 0,0382 | 0,3147 | 0,0402 | 0,1252 | 1,0606 | 0,0372 | 0,0450 | 0,0556 | 0,1324 | 0,0820 | 0,3388 |
| **61** | 0,1404 | 0,1448 | 0,5790 | 0,1049 | 0,3218 | 0,1227 | 0,3509 | 1,3874 | 0,1026 | 0,2797 | 0,3988 | 0,4434 | 0,2978 | 0,9895 |
| **62** | 0,9263 | 2,5122 | 1,8449 | 1,7937 | 2,1690 | 0,7966 | 5,1905 | 22,8663 | 1,8986 | 1,8833 | 0,9849 | 2,8978 | 0,8061 | 13,6316 |
| **63** | 0,8920 | 0,8572 | 0,6398 | 0,6494 | 2,1741 | 0,3124 | 0,9367 | 3,7564 | 0,2633 | 0,7646 | 0,9731 | 0,4420 | 0,6982 | 2,3269 |
| **64** | 0,5440 | 1,2994 | 2,4010 | 0,9382 | 1,4802 | 0,4997 | 3,4337 | 5,9903 | 0,4636 | 1,3063 | 0,6467 | 2,0619 | 0,4900 | 3,6896 |
| **65** | 2,2918 | 0,8633 | 4,2211 | 1,3183 | 5,1175 | 0,7278 | 2,2218 | 19,3616 | 0,6127 | 1,9023 | 2,2287 | 2,2991 | 0,7632 | 5,8454 |
| **66** | 0,0995 | 0,2824 | 0,3709 | 0,1766 | 0,2539 | 0,1921 | 0,2539 | 2,1887 | 0,0810 | 0,0905 | 0,1042 | 0,1375 | 0,1855 | 1,8325 |
| **67** | 0,2684 | 0,2864 | 1,0736 | 0,5564 | 1,5116 | 0,6053 | 0,6863 | 5,9269 | 0,4828 | 0,2637 | 0,6258 | 1,0170 | 0,6161 | 4,9206 |
| **68** | 1,9981 | 1,0911 | 3,8944 | 0,7384 | 6,0431 | 0,7701 | 5,6069 | 9,7283 | 0,6899 | 0,8428 | 1,0720 | 1,3418 | 0,7983 | 16,8541 |
| **69** | 1,0647 | 0,5451 | 1,8621 | 0,3930 | 3,0990 | 1,0810 | 1,2494 | 4,9763 | 0,3508 | 0,4391 | 1,3593 | 1,5117 | 0,4575 | 6,8965 |
| **70** | 0,8339 | 2,0937 | 4,2171 | 1,5763 | 1,9171 | 0,7178 | 4,9400 | 8,6370 | 1,2508 | 0,7407 | 1,9216 | 1,1286 | 1,4717 | 15,4269 |

***Табл. 7. Емпірична матриця попарних порівнянь (стовпці 57-70)***

| № / № | 57 | 58 | 59 | 60 | 61 | 62 | 63 | 64 | 65 | 66 | 67 | 68 | 69 | 70 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **1** | 1,3939 | 0,5555 | 3,2328 | 14,2167 | 3,5403 | 1,5487 | 4,1272 | 0,9548 | 0,7133 | 6,5187 | 1,7943 | 1,4646 | 2,5550 | 1,6859 |
| **2** | 0,5631 | 0,5792 | 0,5761 | 5,2384 | 4,0499 | 0,5530 | 1,6164 | 0,9910 | 0,6468 | 5,3757 | 1,9540 | 0,2174 | 1,0234 | 0,2820 |
| **3** | 1,3864 | 1,3952 | 1,4207 | 28,9150 | 4,2616 | 0,5021 | 1,7342 | 2,3682 | 0,6405 | 13,1618 | 2,0984 | 0,5314 | 1,2146 | 0,7514 |
| **4** | 0,3445 | 0,3415 | 0,8977 | 16,6137 | 2,7456 | 0,3065 | 2,4873 | 0,6796 | 0,9085 | 3,1922 | 2,9629 | 0,7906 | 0,7765 | 0,4533 |
| **5** | 1,2171 | 0,4877 | 2,8051 | 12,4626 | 3,4687 | 1,3271 | 1,6425 | 0,7832 | 1,5446 | 5,8408 | 1,5599 | 0,4790 | 2,5852 | 1,4332 |
| **6** | 1,4284 | 0,6989 | 3,1345 | 28,3424 | 4,4176 | 0,6084 | 3,9155 | 1,0340 | 0,6565 | 13,1217 | 2,0794 | 0,5224 | 2,5252 | 0,7275 |
| **7** | 0,2728 | 0,6265 | 1,4097 | 5,7560 | 1,9940 | 0,3109 | 0,7270 | 0,5221 | 0,7687 | 6,7277 | 1,1098 | 0,6406 | 0,6396 | 0,3150 |
| **8** | 0,9337 | 0,9673 | 2,1349 | 20,4433 | 6,9420 | 0,4955 | 2,7253 | 1,7452 | 1,1396 | 4,0691 | 1,3804 | 0,4574 | 1,9625 | 0,5169 |
| **9** | 0,8541 | 0,9130 | 0,7715 | 9,2082 | 2,3547 | 0,9876 | 2,6693 | 0,5612 | 0,4455 | 4,0310 | 3,0813 | 0,9290 | 0,7031 | 1,0508 |
| **10** | 0,6076 | 0,2462 | 0,6847 | 5,0117 | 4,4690 | 0,2409 | 0,7176 | 1,1143 | 0,7381 | 2,6936 | 0,8987 | 0,6255 | 0,5386 | 0,2970 |
| **11** | 0,7051 | 0,7835 | 0,6670 | 15,9216 | 5,2948 | 0,7147 | 2,0706 | 1,2932 | 0,3125 | 6,8532 | 2,4480 | 0,6894 | 1,3892 | 0,3044 |
| **12** | 0,8643 | 0,3777 | 1,8344 | 15,5738 | 5,7756 | 0,8038 | 1,0185 | 0,4983 | 0,3508 | 8,3951 | 1,2998 | 0,7877 | 0,6141 | 0,9452 |
| **13** | 0,2917 | 0,1199 | 0,2299 | 2,7372 | 1,8290 | 0,2909 | 0,3062 | 0,4444 | 0,1307 | 1,1014 | 0,3937 | 0,2928 | 0,5188 | 0,1270 |
| **14** | 0,5889 | 0,2871 | 1,3124 | 11,7772 | 1,4674 | 0,2616 | 1,6328 | 0,4184 | 0,6923 | 2,4182 | 1,8998 | 0,5849 | 1,0683 | 0,2824 |
| **15** | 0,4760 | 0,2251 | 0,3904 | 4,2019 | 3,3500 | 0,2045 | 0,6440 | 0,8069 | 0,2033 | 4,9202 | 1,5984 | 0,4934 | 0,9933 | 0,2564 |
| **16** | 1,1387 | 1,0813 | 1,0682 | 10,2976 | 3,4158 | 1,0970 | 3,0104 | 1,8976 | 0,5337 | 4,0363 | 3,6480 | 0,5226 | 0,9321 | 1,2880 |
| **17** | 0,9831 | 1,0307 | 2,0277 | 19,8084 | 2,3913 | 0,3892 | 2,7701 | 0,7821 | 1,1524 | 9,2948 | 3,2809 | 0,4338 | 1,9986 | 0,5443 |
| **18** | 0,3615 | 0,4872 | 2,2895 | 8,0405 | 2,9302 | 1,0560 | 1,1200 | 0,7054 | 0,5073 | 9,6540 | 3,6379 | 0,9605 | 0,7520 | 0,4763 |
| **19** | 0,3503 | 0,3601 | 0,7522 | 7,1880 | 1,1525 | 0,1535 | 0,4640 | 0,2117 | 0,4022 | 1,5732 | 0,5537 | 0,1327 | 0,6235 | 0,3906 |
| **20** | 0,5068 | 1,0330 | 2,1774 | 7,8674 | 3,4999 | 1,0713 | 1,3895 | 0,7470 | 0,4698 | 10,4838 | 1,7263 | 0,3862 | 1,9375 | 1,3060 |
| **21** | 0,5214 | 1,4061 | 1,4287 | 25,9701 | 9,6167 | 1,2865 | 1,8562 | 0,9357 | 1,5353 | 5,0453 | 4,9087 | 0,5389 | 1,2251 | 0,7077 |
| **22** | 0,5613 | 0,5411 | 1,2478 | 4,6636 | 1,8417 | 0,5683 | 0,6334 | 0,8913 | 0,2373 | 5,3372 | 1,8094 | 0,2059 | 1,0631 | 0,6217 |
| **23** | 0,2886 | 0,3574 | 0,7910 | 7,1478 | 2,5412 | 0,3677 | 2,2916 | 0,6313 | 0,4151 | 3,0765 | 1,2214 | 0,3285 | 1,4507 | 0,9266 |
| **24** | 0,5741 | 0,5986 | 1,3733 | 12,4628 | 1,5815 | 0,2177 | 0,7471 | 0,9907 | 0,2825 | 2,3335 | 0,9756 | 0,2590 | 1,1143 | 0,6530 |
| **25** | 0,1831 | 0,0797 | 0,4089 | 3,9502 | 0,5055 | 0,1806 | 0,5150 | 0,3251 | 0,0900 | 1,8228 | 0,5896 | 0,1785 | 0,1405 | 0,1969 |
| **26** | 0,1848 | 0,0791 | 0,4381 | 1,5931 | 1,2907 | 0,0742 | 0,5558 | 0,1563 | 0,2353 | 1,9669 | 0,2576 | 0,1972 | 0,1490 | 0,2341 |
| **27** | 0,5501 | 1,3950 | 1,2894 | 13,0301 | 4,3263 | 0,5624 | 1,4722 | 2,4518 | 1,5135 | 6,4579 | 4,7050 | 0,6616 | 1,1764 | 0,6941 |
| **28** | 0,6467 | 0,3168 | 0,6116 | 13,3813 | 4,7457 | 0,7225 | 1,9908 | 0,5366 | 0,8313 | 6,3351 | 1,1857 | 0,6837 | 1,3137 | 0,7520 |
| **29** | 0,5606 | 1,2419 | 2,8730 | 12,1353 | 8,7634 | 1,2879 | 1,5406 | 2,2738 | 1,4719 | 11,8532 | 4,2260 | 0,5316 | 1,2375 | 1,4319 |
| **30** | 0,1661 | 0,4403 | 0,8347 | 3,5870 | 1,2807 | 0,1506 | 1,0735 | 0,6886 | 0,4898 | 3,8282 | 0,6483 | 0,4308 | 0,7990 | 0,1797 |
| **31** | 0,6821 | 0,3266 | 1,4301 | 14,4860 | 1,7969 | 0,6920 | 0,6976 | 0,5421 | 0,8495 | 7,0452 | 1,0999 | 0,3102 | 1,3666 | 0,3049 |
| **32** | 0,4354 | 1,1500 | 1,1344 | 11,4672 | 3,6362 | 1,2503 | 1,2237 | 0,7272 | 0,6202 | 11,8914 | 3,6841 | 1,1699 | 2,0881 | 0,6396 |
| **33** | 1,4405 | 0,6631 | 2,9758 | 11,2679 | 9,9986 | 1,3532 | 1,8463 | 0,9804 | 1,5107 | 6,0740 | 4,5373 | 1,4577 | 1,3278 | 0,7536 |
| **34** | 0,0922 | 0,0481 | 0,2103 | 1,9549 | 0,3290 | 0,0440 | 0,1074 | 0,0760 | 0,1087 | 0,3586 | 0,1622 | 0,0957 | 0,1958 | 0,1091 |
| **35** | 0,4180 | 1,1034 | 1,0259 | 9,5478 | 3,3628 | 1,2495 | 3,1543 | 0,7111 | 0,5971 | 10,3769 | 3,7741 | 1,0756 | 0,8064 | 1,4047 |
| **36** | 0,4484 | 0,4984 | 0,8990 | 21,4248 | 6,4321 | 0,9723 | 2,8578 | 1,6987 | 1,2002 | 10,0095 | 1,6311 | 0,3857 | 1,8553 | 0,5601 |
| **37** | 1,0756 | 1,1176 | 1,1687 | 23,1801 | 7,6711 | 0,4504 | 1,5931 | 0,8987 | 1,3512 | 4,8381 | 3,9575 | 0,5645 | 0,8832 | 1,3300 |
| **38** | 1,3086 | 0,6311 | 1,4808 | 26,8740 | 9,8143 | 0,5184 | 3,7042 | 0,9521 | 1,5869 | 12,5791 | 4,9837 | 0,5488 | 1,2629 | 1,5566 |
| **39** | 0,3442 | 0,8059 | 1,5570 | 7,3825 | 5,4978 | 0,7830 | 1,0300 | 0,4712 | 0,8922 | 3,4119 | 1,1873 | 0,7830 | 1,3661 | 0,3574 |
| **40** | 0,3222 | 0,7230 | 0,7322 | 12,9399 | 1,7754 | 0,2659 | 0,9534 | 0,5232 | 0,7904 | 6,4891 | 1,1354 | 0,6359 | 1,3919 | 0,3197 |
| **41** | 0,6922 | 0,7194 | 1,6666 | 6,0247 | 4,8763 | 0,2849 | 0,9568 | 0,4931 | 0,8406 | 7,1135 | 2,5433 | 0,7544 | 0,5725 | 0,8715 |
| **42** | 0,5680 | 1,1547 | 2,5592 | 12,0706 | 2,9984 | 1,3120 | 1,2338 | 2,0933 | 1,4042 | 11,0593 | 4,1580 | 1,1736 | 2,1880 | 0,5273 |
| **43** | 0,4370 | 0,4399 | 1,1160 | 20,2515 | 7,1208 | 1,0795 | 1,1211 | 1,8384 | 0,4363 | 10,0549 | 3,7258 | 0,5005 | 0,9393 | 1,1992 |
| **44** | 0,4653 | 0,4956 | 0,9810 | 7,1916 | 6,9040 | 0,3981 | 1,1666 | 0,7696 | 1,1584 | 3,5412 | 3,4917 | 0,9165 | 1,8344 | 0,4776 |
| **45** | 0,2654 | 0,5681 | 1,1699 | 4,4028 | 1,7272 | 0,5420 | 1,5629 | 0,4165 | 0,2369 | 2,6958 | 0,9314 | 0,2568 | 0,5370 | 0,2371 |
| **46** | 0,5658 | 0,6421 | 1,1274 | 26,2088 | 9,5350 | 0,5575 | 1,5398 | 1,0659 | 0,7586 | 5,6628 | 1,7974 | 1,3544 | 2,5448 | 0,6344 |
| **47** | 0,4037 | 0,1568 | 0,4313 | 3,1779 | 3,1074 | 0,4610 | 0,4600 | 0,6756 | 0,1954 | 3,9392 | 0,6615 | 0,1655 | 0,3227 | 0,5216 |
| **48** | 1,2872 | 0,5008 | 1,0276 | 24,8538 | 8,1468 | 1,2553 | 3,2007 | 2,0012 | 1,3741 | 5,2061 | 1,6521 | 1,2986 | 0,9251 | 1,3932 |
| **49** | 0,1858 | 0,1937 | 0,8255 | 7,9886 | 2,8502 | 0,1927 | 1,0676 | 0,2912 | 0,4501 | 3,9388 | 1,4571 | 0,1784 | 0,8004 | 0,2024 |
| **50** | 0,0951 | 0,0450 | 0,2335 | 0,9429 | 0,7208 | 0,0437 | 0,2662 | 0,1669 | 0,0516 | 0,4569 | 0,1687 | 0,1028 | 0,2010 | 0,1158 |
| **51** | 0,6837 | 0,6784 | 3,1300 | 26,8644 | 9,7467 | 0,5267 | 3,7972 | 2,1572 | 1,6322 | 12,3384 | 2,0712 | 1,4494 | 2,8507 | 0,7995 |
| **52** | 0,4532 | 0,4673 | 1,1908 | 22,2126 | 3,5755 | 0,5310 | 1,3079 | 0,7655 | 0,5257 | 11,0498 | 3,7916 | 1,1866 | 2,2771 | 1,3500 |
| **53** | 0,9369 | 0,3490 | 0,8447 | 17,9766 | 2,5075 | 1,0153 | 1,0277 | 1,5463 | 0,4487 | 9,5972 | 1,5978 | 0,9329 | 0,7357 | 0,5204 |
| **54** | 0,7189 | 0,2911 | 1,6846 | 7,5539 | 2,2552 | 0,3451 | 2,2625 | 0,4850 | 0,4350 | 7,2721 | 0,9833 | 0,7453 | 0,6615 | 0,8861 |
| **55** | 1,2910 | 1,2202 | 2,7970 | 12,2012 | 3,3581 | 1,2405 | 1,4322 | 2,0409 | 1,3104 | 5,3894 | 1,6230 | 1,2527 | 2,1857 | 0,6795 |
| **56** | 0,1543 | 0,0575 | 0,1637 | 2,9512 | 1,0106 | 0,0734 | 0,4298 | 0,2710 | 0,1711 | 0,5457 | 0,2032 | 0,0593 | 0,1450 | 0,0648 |
| **57** | 1,0000 | 0,5903 | 3,2015 | 29,6375 | 9,2518 | 0,5771 | 4,1220 | 1,1725 | 1,6663 | 13,7737 | 2,3722 | 0,5569 | 1,0346 | 1,6591 |
| **58** | 1,6941 | 1,0000 | 2,8786 | 11,2840 | 9,9013 | 1,3132 | 3,7915 | 0,8880 | 0,6276 | 6,0859 | 2,0139 | 1,3152 | 2,5703 | 1,6044 |
| **59** | 0,3124 | 0,3474 | 1,0000 | 13,2116 | 1,7565 | 0,6864 | 1,9337 | 1,0356 | 0,7833 | 2,5116 | 0,8177 | 0,2508 | 0,5454 | 0,3476 |

| № / № | 57 | 58 | 59 | 60 | 61 | 62 | 63 | 64 | 65 | 66 | 67 | 68 | 69 | 70 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **60** | 0,0337 | 0,0886 | 0,0757 | 1,0000 | 0,1836 | 0,0683 | 0,0734 | 0,0442 | 0,0331 | 0,6900 | 0,0838 | 0,0709 | 0,1240 | 0,0348 |
| **61** | 0,1081 | 0,1010 | 0,5693 | 5,4459 | 1,0000 | 0,0990 | 0,2471 | 0,3306 | 0,1144 | 1,9191 | 0,6683 | 0,0798 | 0,1759 | 0,0863 |
| **62** | 1,7328 | 0,7615 | 1,4570 | 14,6493 | 10,1009 | 1,0000 | 3,8294 | 2,4300 | 1,4874 | 5,7991 | 4,5434 | 0,5572 | 1,2440 | 1,6491 |
| **63** | 0,2426 | 0,2637 | 0,5171 | 13,6159 | 4,0470 | 0,2611 | 1,0000 | 0,3335 | 0,5355 | 1,7076 | 0,8390 | 0,2392 | 0,9455 | 0,2361 |
| **64** | 0,8529 | 1,1261 | 0,9656 | 22,6295 | 3,0244 | 0,4115 | 2,9988 | 1,0000 | 0,9996 | 3,6998 | 2,9675 | 0,3880 | 0,6594 | 0,9901 |
| **65** | 0,6001 | 1,5933 | 1,2766 | 30,1718 | 8,7432 | 0,6723 | 1,8674 | 1,0004 | 1,0000 | 5,1190 | 4,2669 | 0,5147 | 0,8782 | 0,5110 |
| **66** | 0,0726 | 0,1643 | 0,3982 | 1,4493 | 0,5211 | 0,1724 | 0,5856 | 0,2703 | 0,1954 | 1,0000 | 0,2323 | 0,1551 | 0,1013 | 0,1802 |
| **67** | 0,4215 | 0,4965 | 1,2229 | 11,9292 | 1,4964 | 0,2201 | 1,1919 | 0,3370 | 0,2344 | 4,3047 | 1,0000 | 0,4019 | 0,7899 | 0,4458 |
| **68** | 1,7956 | 0,7604 | 3,9877 | 14,1033 | 12,5351 | 1,7947 | 4,1806 | 2,5771 | 1,9430 | 6,4493 | 2,4884 | 1,0000 | 1,2212 | 0,6487 |
| **69** | 0,9666 | 0,3891 | 1,8335 | 8,0652 | 5,6849 | 0,8038 | 1,0577 | 1,5166 | 1,1387 | 9,8762 | 1,2660 | 0,8189 | 1,0000 | 0,7982 |
| **70** | 0,6027 | 0,6233 | 2,8769 | 28,7393 | 11,5853 | 0,6064 | 4,2351 | 1,0100 | 1,9570 | 5,5482 | 2,2430 | 1,5414 | 1,2528 | 1,0000 |

Слідуючи Етапу 1 по емпіричній матриці попарних порівнянь необхідно знайти оцінки ваг альтернатив. Для цього застосуємо модель 4(Б) для знаходження ваг альтернатив (детально модель наведена в [4]). Знайдені за допомогою моделі 4(Б) ваги наведені в Табл. 8.

***Табл. 8. Нормовані ваги альтернатив, які знайдені по емпіричній матриці попарних порівнянь за допомогою Моделі 4(Б)***

| № | Вага | № | Вага | № | Вага | № | Вага | № | Вага |
|---|---|---|---|---|---|---|---|---|---|
| 1 | 0,00144766 | 15 | 0,00031204 | 29 | 0,00114382 | 43 | 0,00083060 | 57 | 0,00159426 |
| 2 | 0,00037944 | 16 | 0,00083471 | 30 | 0,00025662 | 44 | 0,00079131 | 58 | 0,00155123 |
| 3 | 0,00126948 | 17 | 0,00062781 | 31 | 0,00048708 | 45 | 0,00036231 | 59 | 0,00044781 |
| 4 | 0,00050961 | 18 | 0,00061805 | 32 | 0,00119469 | 46 | 0,00144512 | 60 | 0,00008366 |
| 5 | 0,00105840 | 19 | 0,00021209 | 33 | 0,00139451 | 47 | 0,00026650 | 61 | 0,00013951 |
| 6 | 0,00130596 | 20 | 0,00086244 | 34 | 0,00008366 | 48 | 0,00111671 | 62 | 0,00161298 |
| 7 | 0,00048521 | 21 | 0,00136563 | 35 | 0,00101458 | 49 | 0,00027970 | 63 | 0,00036377 |
| 8 | 0,00079213 | 22 | 0,00035521 | 36 | 0,00089732 | 50 | 0,00008366 | 64 | 0,00074364 |
| 9 | 0,00061243 | 23 | 0,00051499 | 37 | 0,00098515 | 51 | 0,00141943 | 65 | 0,00137463 |
| 10 | 0,00039240 | 24 | 0,00035975 | 38 | 0,00124854 | 52 | 0,00093027 | 66 | 0,00009777 |
| 11 | 0,00044613 | 25 | 0,00011556 | 39 | 0,00048293 | 53 | 0,00077948 | 67 | 0,00028323 |
| 12 | 0,00046922 | 26 | 0,00011745 | 40 | 0,00044196 | 54 | 0,00055383 | 68 | 0,00174522 |
| 13 | 0,00017444 | 27 | 0,00133260 | 41 | 0,00048681 | 55 | 0,00128495 | 69 | 0,00056788 |
| 14 | 0,00034843 | 28 | 0,00042731 | 42 | 0,00109378 | 56 | 0,00010355 | 70 | 0,00154059 |

Позиції при впорядкуванні знайдених ваг за убуванням: 68, 62, 57, 58, 70, 1, 46, 51, 33, 65, 21, 27, 6, 55, 3, 38, 32, 29, 48, 42, 5, 35, 37, 52, 36, 20, 16, 43, 8, 44, 53, 64, 17, 18, 9, 69, 54, 23, 4, 31, 41, 7, 39, 12, 59, 11, 40, 28, 10, 2, 63, 45, 24, 22, 14, 15, 67, 49, 47, 30, 19, 13, 61, 26, 25, 56, 66, 34, 50, 60. Дійсний переможець в наведеному впорядкуванні зайняв 6 місце.

Якщо на Етапі 1 для визначення оцінок ваг використовувати класичний метод (пошук ваг альтернатив як елементів власного вектору матриці попарних порівнянь, який відповідає максимальному власному числу матриці), то отримаємо ваги, які наведені в Табл. 9.

***Табл. 9. Нормовані ваги альтернатив, які знайдені по емпіричній матриці попарних порівнянь за допомогою класичного методу***

| № | Вага | № | Вага | № | Вага | № | Вага | № | Вага |
|---|---|---|---|---|---|---|---|---|---|
| 1 | 0,02183031 | 15 | 0,00748299 | 29 | 0,02259041 | 43 | 0,01710684 | 57 | 0,02350067 |
| 2 | 0,00934289 | 16 | 0,01797268 | 30 | 0,00693438 | 44 | 0,01572453 | 58 | 0,02569871 |
| 3 | 0,02036421 | 17 | 0,01495228 | 31 | 0,01136776 | 45 | 0,00895401 | 59 | 0,01092521 |
| 4 | 0,01275600 | 18 | 0,01477740 | 32 | 0,01900451 | 46 | 0,02174720 | 60 | 0,00120868 |
| 5 | 0,01939620 | 19 | 0,00570069 | 33 | 0,02498934 | 47 | 0,00705689 | 61 | 0,00367391 |
| 6 | 0,02029092 | 20 | 0,01658224 | 34 | 0,00170339 | 48 | 0,02182543 | 62 | 0,02521008 |
| 7 | 0,01114681 | 21 | 0,02370294 | 35 | 0,01928216 | 49 | 0,00690983 | 63 | 0,00905019 |
| 8 | 0,01615891 | 22 | 0,00927783 | 36 | 0,01880474 | 50 | 0,00178982 | 64 | 0,01598368 |

| № | Вага | № | Вага | № | Вага | № | Вага | № | Вага |
|---|---|---|---|---|---|---|---|---|---|
| 9 | 0,01531115 | 23 | 0,01214221 | 37 | 0,02056576 | 51 | 0,02480749 | 65 | 0,02068810 |
| 10 | 0,01033382 | 24 | 0,00923552 | 38 | 0,02357755 | 52 | 0,01854666 | 66 | 0,00257934 |
| 11 | 0,01233080 | 25 | 0,00304845 | 39 | 0,01244670 | 53 | 0,01630315 | 67 | 0,00785267 |
| 12 | 0,01220317 | 26 | 0,00304870 | 40 | 0,01160145 | 54 | 0,01309741 | 68 | 0,02471132 |
| 13 | 0,00439212 | 27 | 0,02296763 | 41 | 0,01278566 | 55 | 0,02124138 | 69 | 0,01322168 |
| 14 | 0,00853823 | 28 | 0,01157322 | 42 | 0,02248972 | 56 | 0,00264224 | 70 | 0,02293906 |

Позиції при впорядкуванні знайдених ваг за убуванням: 58, 62, 33, 51, 68, 21, 38, 57, 27, 70, 29, 42, 1, 48, 46, 55, 65, 37, 3, 6, 5, 35, 32, 36, 52, 16, 43, 20, 53, 8, 64, 44, 9, 17, 18, 69, 54, 41, 4, 39, 11, 12, 23, 40, 28, 31, 7, 59, 10, 2, 22, 24, 63, 45, 14, 67, 15, 47, 30, 49, 19, 13, 61, 26, 25, 56, 66, 50, 34, 60. Дійсний переможець в наведеному впорядкуванні зайняв лише 13 місце.

На даному етапі видно, що в умовах погано узгодженої матриці попарних порівнянь великої розмірності Модель 4(Б) справилась з поставленою задачею набагато ефективніше класичного методу.

Виходячи з результатів роботи Моделі 4(Б) для Етапу 2 відбираються альтернативи з номерами: 68, 62, 57, 58, 70, **1**, 46, 51, 33, 65, 21, 27, 6, 55, 3. Присвоїмо даним альтернативам нові порядкові номери відповідно: 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15 (дійсний переможець тепер має номер 6). Ці альтернативи повторно порівнюються експертом (експертами), після чого формується нова емпірична матриця попарних порівнянь розмірності 15×15. В нашому прикладі ця матриця генерується за принципом (1) з тою відмінністю, що модуль $|k_{ij}|$ розподілений рівномірно в інтервалі $(0,1; 0,2)$. Зменшення рівня шуму в попарних порівняннях та покращення узгодженості матриці відповідає ситуації на практиці, коли експерти набагато менше помиляються при порівнянні малої кількості альтернатив. Емпірична матриця попарних порівнянь 15 кращих альтернатив наведена в Табл. 10.

***Табл. 10. Емпірична матриця попарних порівнянь 15 кращих альтернатив***

| № / № | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **1** | 1,0000 | 1,1799 | 0,8348 | 1,1754 | 0,9835 | 1,1181 | 1,2148 | 1,1544 | 1,1712 | 1,0082 | 0,9318 | 0,8896 | 1,1356 | 1,3002 | 1,1970 |
| **2** | 0,8475 | 1,0000 | 0,8598 | 1,1476 | 1,3307 | 1,1186 | 0,8297 | 1,1277 | 0,8067 | 0,9411 | 0,8914 | 1,1486 | 0,7956 | 0,9382 | 1,1493 |
| **3** | 1,1978 | 1,1630 | 1,0000 | 1,1958 | 1,3692 | 0,8139 | 1,1499 | 0,8757 | 1,1758 | 1,3435 | 1,1677 | 1,1445 | 1,1490 | 1,3850 | 1,2070 |
| **4** | 0,8508 | 0,8714 | 0,8362 | 1,0000 | 1,0210 | 0,7898 | 1,1870 | 1,1548 | 1,1023 | 0,9751 | 1,1698 | 0,8696 | 1,1265 | 1,2583 | 0,8458 |
| **5** | 1,0168 | 0,7515 | 0,7304 | 0,9795 | 1,0000 | 0,7462 | 0,7532 | 1,0307 | 1,0171 | 1,1404 | 0,7601 | 0,7018 | 1,0430 | 0,8661 | 0,7692 |
| **6** | 0,8944 | 0,8940 | 1,2287 | 1,2662 | 1,3401 | 1,0000 | 1,2236 | 1,1359 | 0,8516 | 1,2844 | 1,2300 | 0,8554 | 1,1850 | 1,0474 | 0,9232 |
| **7** | 0,8232 | 1,2052 | 0,8696 | 0,8424 | 1,3277 | 0,8173 | 1,0000 | 0,8520 | 1,1452 | 0,9569 | 0,9008 | 0,8832 | 0,7988 | 0,9182 | 0,8538 |
| **8** | 0,8662 | 0,8868 | 1,1420 | 0,8659 | 0,9702 | 0,8804 | 1,1737 | 1,0000 | 0,8265 | 1,2633 | 0,9101 | 0,8011 | 1,1408 | 0,9901 | 0,8411 |
| **9** | 0,8538 | 1,2396 | 0,8505 | 0,9072 | 0,9832 | 1,1742 | 0,8732 | 1,2099 | 1,0000 | 0,9200 | 0,8785 | 1,1413 | 0,8449 | 1,2809 | 0,8494 |
| **10** | 0,9919 | 1,0625 | 0,7443 | 1,0256 | 0,8769 | 0,7786 | 1,0450 | 0,7916 | 1,0869 | 1,0000 | 1,0782 | 0,7688 | 0,7780 | 0,8480 | 0,8129 |
| **11** | 1,0732 | 1,1219 | 0,8564 | 0,8549 | 1,3157 | 0,8130 | 1,1101 | 1,0987 | 1,1382 | 0,9274 | 1,0000 | 0,8572 | 1,1597 | 1,3174 | 1,1216 |
| **12** | 1,1241 | 0,8706 | 0,8737 | 1,1500 | 1,4249 | 1,1691 | 1,1322 | 1,2483 | 0,8762 | 1,3007 | 1,1666 | 1,0000 | 1,1848 | 1,3330 | 1,2115 |
| **13** | 0,8806 | 1,2569 | 0,8704 | 0,8877 | 0,9587 | 0,8439 | 1,2519 | 0,8765 | 1,1836 | 1,2853 | 0,8623 | 0,8441 | 1,0000 | 1,0097 | 0,8684 |
| **14** | 0,7691 | 1,0659 | 0,7220 | 0,7947 | 1,1545 | 0,9548 | 1,0891 | 1,0100 | 0,7807 | 1,1792 | 0,7591 | 0,7502 | 0,9904 | 1,0000 | 1,0327 |
| **15** | 0,8354 | 0,8701 | 0,8285 | 1,1824 | 1,3000 | 1,0831 | 1,1712 | 1,1890 | 1,1772 | 1,2301 | 0,8916 | 0,8254 | 1,1516 | 0,9683 | 1,0000 |

За допомогою Моделі 4(Б) знаходимо оцінки ваг (наведені в Табл. 11).

***Табл. 11. Нормовані ваги альтернатив, які знайдені по емпіричній матриці попарних порівнянь 15 кращих альтернатив за допомогою Моделі 4(Б)***

| № | Вага | № | Вага | № | Вага | № | Вага | № | Вага |
|---|---|---|---|---|---|---|---|---|---|
| 1 | 0,033062089 | 4 | 0,030903278 | 7 | 0,031607605 | 10 | 0,029477005 | 13 | 0,033248924 |
| 2 | 0,031715613 | 5 | 0,028064139 | 8 | 0,031315967 | 11 | 0,032073308 | 14 | 0,02859486 |
| 3 | 0,032986573 | 6 | 0,033913828 | 9 | 0,033205796 | 12 | 0,033372433 | 15 | 0,032809336 |

Позиції при впорядкуванні знайдених ваг за убуванням: 6, 12, 13, 9, 1, 3, 15, 11, 2, 7, 8, 4, 10, 14, 5. Дійсний переможець в наведеному впорядкуванні зайняв 1 місце.

Виходячи з результатів роботи Моделі 4 для Етапу 3 обираються альтернативи з номерами: 6, 12, 13, 9, 1, 3, 15. Присвоїмо даним альтернативам нові порядкові номери відповідно: 1, 2, 3, 4, 5, 6, 7 (дійсний пере-

можець тепер має номер 1). Ці альтернативи повторно порівнюються експертом (експертами), після чого формується нова емпірична матриця попарних порівнянь розмірності $7\times7$. В нашому прикладі ця матриця знову генерується за принципом (1) з тою відмінністю, що модуль $|k_{ij}|$ розподілений рівномірно в інтервалі $(0;0,03)$. Емпірична матриця попарних порівнянь 7 найкращих альтернатив наведена в Табл. 12.

***Табл. 12. Емпірична матриця попарних порівнянь 7 кращих альтернатив***

| № / № | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|
| **1** | 1,0000 | 0,9962 | 1,0503 | 0,9978 | 0,9885 | 1,0250 | 1,0884 |
| **2** | 1,0038 | 1,0000 | 1,0206 | 0,9953 | 0,9745 | 0,9877 | 1,0501 |
| **3** | 0,9521 | 0,9798 | 1,0000 | 0,9751 | 1,0159 | 0,9898 | 1,0093 |
| **4** | 1,0022 | 1,0048 | 1,0256 | 1,0000 | 1,0076 | 1,0039 | 1,0517 |
| **5** | 1,0116 | 1,0262 | 0,9844 | 0,9924 | 1,0000 | 0,9828 | 1,0558 |
| **6** | 0,9756 | 1,0124 | 1,0103 | 0,9961 | 1,0175 | 1,0000 | 1,0689 |
| **7** | 0,9188 | 0,9523 | 0,9907 | 0,9508 | 0,9472 | 0,9355 | 1,0000 |

За допомогою класичного методу (пошук ваг альтернатив як елементів власного вектору матриці попарних порівнянь, який відповідає максимальному власному числу матриці) знаходимо оцінки ваг (наведені в Табл. 13).

Позиції при впорядкуванні знайдених ваг за убуванням: 1, 4, 6, 5, 2, 3, 7. Дійсний переможець в наведеному впорядкуванні зайняв 1 місце.

Таким чином, по завершенню Етапу 3 переможець веред вихідного набору з 70 альтернатив знайдений вірно.

***Табл. 13. Нормовані ваги альтернатив, які знайдені по емпіричній матриці попарних порівнянь 7 кращих альтернатив за допомогою класичного методу***

| № | Вага |
|---|---|
| 1 | 0,14573088 |
| 2 | 0,14344353 |
| 3 | 0,14120601 |
| 4 | 0,14476148 |
| 5 | 0,14387277 |
| 6 | 0,14442723 |
| 7 | 0,1365581 |

## Висновок

Незважаючи на розвиток наука та автоматизації методів прийняття рішення, всі відповідальні рішення приймає людина. В сучасній економічній ситуації перед впровадженням будь-якого проекту (будь-то побудова корабля, впровадження АСУ, тощо) виникає питання вибору однієї з альтернатив. В цьому випадку найбільш логічним рішенням є генерація множини альтернатив (проектів) під заздалегідь відомі критерії. Для подальшого вибору одного найкращого проекту може використовуватись запропонований в даній статті модифікований метод аналізу ієрархій.

## Список посилань


1. Saaty, T.L. Multycriteric Decision Making. The Analytic Hierarchy Process, McGraw Hill International. – New York, 1980. Translated to Russian, Portuguese, and Chinese. Revised edition, Paperback. – Pittsburgh, PA: RWS Publications, 1990, 1996.
2. Саати, Т. Принятие решений. Метод анализа иерархий: Tomas Saaty. The Analytic Hierarchy Process. – Пер. с англ. Р.Г. Вачнадзе. – М.: Радио и связь, 1993. – 315 с.
3. Саати, Т. Аналитическое планирование. Организация систем / Саати Т., Кернс К.; пер. с англ. Р.Г. Вачнадзе; под ред. И.А. Ушакова. – М.: Радио и связь, 1991. – 223 с.
4. Павлов, А.А. Математические модели оптимизации для обоснования и нахождения весов объектов в методе парных сравнений [Текст] / А.А. Павлов, Е.И. Лищук, В.Н. Кут // Системні дослідження та інформаційні технології. – 2007р. – №2. – С. 13 – 21.
5. Павлов, А.А. Многокритериальный выбор в задаче обработки данных матрицы парных сравнений [Текст] / А.А. Павлов, Е.И. Лищук, В.Н. Кут // Вісник НТУУ „КПІ” Інформатика, управління та обчислювальна техніка. – 2007. – №46. – С. 48 – 52.


Поступила в редакцию 7.12.2009

УДК 519.854.2

*ПАВЛОВ А.А.,*
*МИСЮРА Е.Б.,*
*МЕЛЬНИКОВ О.В.,*
*ГАНЗИНА Е.*

# ПРОГРАММНЫЙ ПРОДУКТ «РЕШЕНИЕ ЗАДАЧИ АГРЕГАЦИИ ТЕХНОЛОГИЧЕСКОГО ГРАФА»

Рассмотрен программный продукт, предназначенный для построения агрегированной модели производства, созданный в рамках разработки системы планирования мелкосерийного производства.

The software product designed for constructing the aggregated model of manufacturing process is considered which was created in the framework of the small series manufactories planning system development.

В рамках разработки (совершенствования) системы планирования производства мелкосерийного типа (СПУМП) – диалогового пакета программ, предназначенных для решения задач планирования и управления функционированием мелкосерийного производства [1], был создан программный продукт «Решение задачи агрегации технологического графа». СПУМП основывается на трехуровневой модели планирования и предназначается для формирования производственной программы предприятия для всех уровней управления предприятием. Построение производственной программы согласно трехуровневой модели планирования производится в несколько этапов: построение агрегированной модели, распределение производственной программы на плановый период и составление пооперационного плана функционирования подразделений предприятия с привязкой к оборудованию. В программном продукте «Решение задачи агрегации технологического графа» реализован первый этап «Построение агрегированной модели» трехуровневой модели планирования производственной программы предприятия.

Программный продукт обладает удобным, графическим интерфейсом и наглядными средствами визуализации данных и результатом расчетов. Программный продукт был разработан с учетом потребностей пользователей, которыми являются менеджеры по планированию производственной процесса. Представителям высшего менеджмента при принятии решений неудобно оперировать громоздкими расчетами, поэтому им необходима информация о системе в обобщенном, сжатом и доступном виде.

Программный продукт предоставляет удобный интерфейс для введения и модификации данных, которые могут вводиться как оператором напрямую через диалоговое окно, так и считываться программным продуктом из заранее подготовленных файлов данных специального формата (Рис. 1, 2).

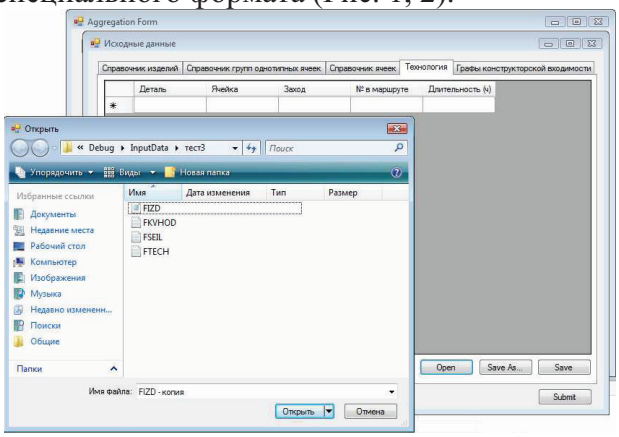



***Рис. 1. Интерфейс введения данных в систему***

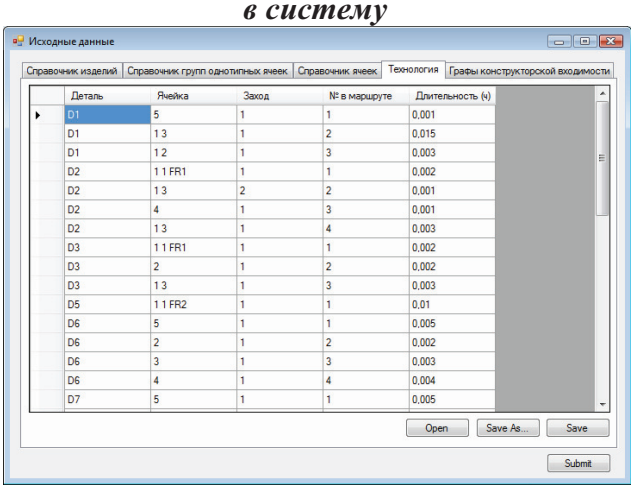



***Рис. 2. Интерфейс модификации данных***

Программный продукт решает следующие задачи:

1. Построение ациклического ориен-тированного технологического графа взаимосвязи ячейкокомплектов (Рис. 3). Вершинами

данного графа являются ячейкокомплекты (совокупность операций, выполняемых в одной производственной ячейке в рамках одного захода по одному изделию). Дугами обозначается степень вхождения данной детали в конечный продукт, а также порядок выполнения технологического процесса.

На данном этапе программный продукт позволяет увидеть, какие группы ячейкокомплектов будут агрегированы на следующем этапе.

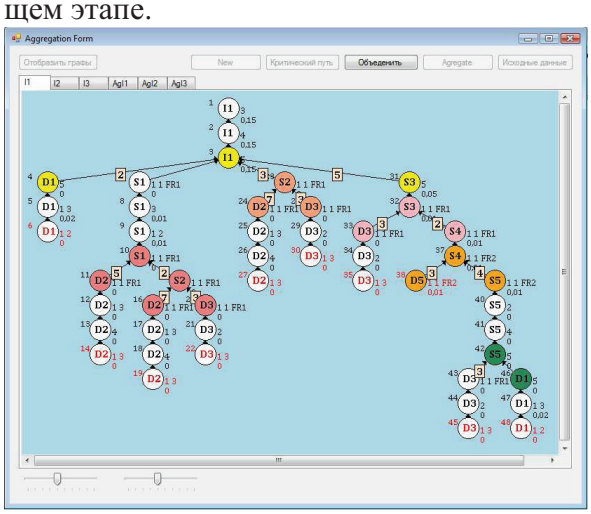



***Рис. 3. Граф ячейкокомплектов по одному изделию***

2. Агрегация технологического графа изделия выполняется с учетом следующего ограничения: длительность изготовления каждого изделия (а также ячейкокомплекта) определяется его критическим путем. Для поиска критического пути изделия используется модифицированный метод ветвей и границ для поиска наиболее длинного пути на графе.

3. Поиск критических путей изделий. В результате объединения смежных ячейкокомплектов и расчета показателей по всему объему партии получаем агрегированные графы по каждому отдельно взятому изделию из портфеля заказов. На данном этапе для каждого изделия по агрегированному графу проводится поиск критического пути, используя модифицированный метод ветвей и границ для поиска наиболее длинного пути на графе (Рис. 4).

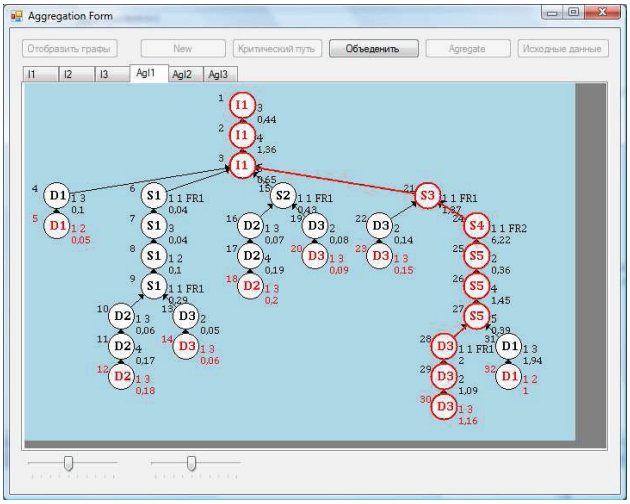



***Рис. 4. Критический путь одного изделия***

4. Построение графа на критических путях изделия. В результате выполнения предыдущих пунктов получены критические пути всех изделий. На основе полученной агрегированной информации строится граф на критических путях изделий.

Таким образом, программный продукт «Решение задачи агрегации технологического графа» позволяет наглядно отобразить ход построения агрегированной модели при планировании мелкосерийного производства. Продукт может использоваться менеджерами производства для получения обобщенной информации о производственном процессе, выбора уровня агрегации, построения агрегированной модели производства и последующего решения оптимизационных задач на полученном графе ячейкокомплектов.

## Список литературы

1. Павлов А.А., Теленик С.Ф. Информационные технологии и алгоритмизация в управлении. – К.: Техника.– 2002.– 344 с.

Поступила в редакцию 14.12.2009

УДК 004.2

*МАРКОВСКИЙ А.П.,*
*АБУ-УСБАХ А.Н.,*
*РОМАНЕЦ Н.Н.*

# МЕТОД ДООПРЕДЕЛЕНИЯ ЧАСТИЧНО-ЗАДАННЫХ БУЛЕВЫХ ФУНКЦИЙ ДЛЯ ОБЕСПЕЧЕНИЯ ИХ ЛАВИННЫХ СВОЙСТВ

В статье предложен метод построения булевых балансных функций, которые соответствуют критерию строго лавинного эффекта. Особенность решаемой задачи состоит в том, что формируемая функция частично задана. Суть проектирования состоит в доопределении частично заданной функции таким образом, чтобы она удовлетворяла критерию лавинного эффекта. Подробно описана формализованная процедура конструирования балансных функций, обладающих строгим лавинным эффектом. Приведен пример синтеза функции.

In this paper the method for designing of Boolean balanced function that satisfies the Strict Avalanche Criterion (SAC) are presented. A peculiaruty of solving task is that designed function is partially given. The goal of designing consist of finish building of partially difined function such way that it will satisfies the Strict Avalanche Criterion (SAC). The formalized procedure for construction partial derivative balanced SAC-functions is described in detail. Examples of function design are given.

## Введение

Интенсивное расширение информационной интеграции на основе компьютерных сетей является одним из основных фактором повышения эффективности управления во всех областях человеческой деятельности.

Развитие информационной интеграции в значительной мере зависит от эффективности реализации функций защиты данных и контроля прав доступа к информационным ресурсам. Это определяет необходимость постоянного развития и совершенствования систем защиты информации.

К настоящему времени в основе большинства таких систем лежат криптографические механизмы, базирующиеся на аналитически неразрешимых математических задачах теории чисел, эллиптических кривых и булевых функций [1]. Использование последних играет особенно важную роль в современных системах защиты информации, поскольку вычисление булевых преобразований выполняется на 3-4 порядка быстрее по сравнению с сложными мультипликативными операциями модулярной арифметики, выполняемыми над числами, длина которых на порядок превышает разрядность процессоров.

Наиболее важным, с точки зрения практического использования, является свойство строгого лавинного эффекта (Strict Avalanche Criterion), которое характеризуется максимальным значением дифференциальной энтропии изменения значения функции при инвертировании любой из переменных, на которых определена эта функция.

Булевы функции, обладающие свойством строгого лавинного эффекта играют ключевую роль при создании широкого класса алгоритмов защиты информации, поскольку это свойство обеспечивает устойчивость к нарушению защиты дифференциальным и линейным криптоанализом [2] .

Важной проблемой практического использования булевых функций, обладающих свойством лавинного эффекта является получение таких функций.

Таким образом, задача получения булевых функций, обладающих важными для криптографических средств защиты информации свойствами максимума полной и дифференциальной энтропии, является важной и актуальной.

## Обзорный анализ методов получения SAC-функций

Булева функция $f(x_1,...,x_n)$ от $n$ переменных определена на $2^n$ возможных наборах значений, принадлежащих множеству $Z$, и принимает значения на множестве $\{0,1\}$. Функция соответствует критерию максимума полной энтропии, или, иными словами, является балансной, если она с одинаковой вероятностью принимает значения нуля и единицы:

$$\sum_{x_1,...,x_n \in Z} f(x_1,...,x_n) = 2^{n-1} \quad (1)$$

Булева функция $f(x_1,...,x_n)$ соответствует критерию максимума условной энтропии, или критерию строго лавинного эффекта (SAC), если изменение значения какой либо одной из $n$ переменных приводит к изменению значения функции с вероятностью $0.5$:

$\forall x_j, j = 1,...,n$:

$$\sum_{x_1,...,x_n \in Z} f(x_1,...,x_j,...,x_n) \oplus f(x_1,...,\bar{x}_j,...,x_n) = 2^{n-} \quad (2)$$

Наиболее важными критериями качества методик получения балансных булевых SAC-функций с точки зрения их практического применения следует считать:

- затраты вычислительных ресурсов (машинного времени и памяти) на реализацию процесса синтеза;
- количество функций от $n$ переменных, которые могут быть синтезированными ( до сегодняшнего дня проблема определения общего количества балансных SAC-функций для $n$ переменных есть открытой [1]);

Принимая во внимание практическую важность проблемы автоматизированного синтеза балансных SAC–функций для современных средств защиты информации, за последние 15 лет предложен ряд подходов по решению этой проблемы.

Все известные методы получения SAC–функций ориентированы на их синтез “с нуля”, то есть исходят из того, что при построении булевой функции не накладывается дополнительных условий на их значения.

Однако, в последние годы началось активное использование булевых функций для решения задач защиты информации, традиционно решаемых с использованием технологий теории чисел. В частности, появились работы, в которых булевы преобразования используются для реализации криптографической концепции “нулевых знаний”, аутентификации абонентов, несимметричной криптографии [3]. Очевидным преимуществом использования булевых функций для перечисленных задач является существенный (на 2-3 порядка) выигрыш в скорости реализации функций защиты информации.

Для большинства из приведенных задач процесс построения булевых функций включает два этапа:

- определение значений функции на части наборов значений переменных исходя из условий, накладываемых криптографическим преобразованием;
- доопределение функции на оставшихся наборах переменных так, чтобы она удовлетворяла определенным криптографическим критериям, основными из которых является максимум полной и дифференциальной энтропии.

При реализации второго этапа возникает задача доопределения частично-заданных булевых функций таким образом, чтобы они были балансными и удовлетворяли критерию лавинного эффекта (SAC).

Высоконелинейные балансные SAC-функции могут быть получены путем деконкатенации bent-функций [2], тем не менее получение самых bent-функций от большого количества переменных представляет собой довольно сложную проблему, решение которой требует затрат значительных вычислительных ресурсов, как в плане затрат машинного времени, так и в плане нужных объемов памяти.

Значительно меньших ресурсов требует реализация рекурсивного получения булевых балансных SAC-функций от $n$ переменных с использованием порождающих четырех SAC-функций (не балансных) от $n-1$ переменных [1]. Однако этот метод также не пригоден для доопределения частично-определенных SAC-функций, поскольку на порождающие функции также накладывается условие частичной определенности.

Наиболее совершенный на сегодняшний день метод получения балансовых SAC-функций предложен К.Куросава и Т.Сатохом [2].

Анализ показывает, что метод разрешает получать лишь незначительную часть балансных SAC-функций от их общего количества.

Метод не пригоден для доопределения частично-заданных функций, поскольку при этом накладываются нелинейные условия на компоненты вектора Q, которые могут быть выполнены только в процессе перебора.

Таким образом, приведенный выше краткий обзор известных на сегодняшний день подходов к проблеме синтеза балансных SAC-функций показывает, что они не позво-

ляют решать новую задачу – построение SAC-функций при наличии ограничений в виде предварительного задания значений функций на части наборов переменных. Для решения этой задачи необходимо разработать новый метод.

Целью работы является создание метода доопределения значений частично-заданных булевых функций так, чтобы они удовлетворяли условию строгого лавинного эффекта.

### Способ доопределения таблицы истинности частично-заданной функции для обеспечения ее лавинных свойств

Исходным для решения поставленной задачи является булева функция $f(x_1,...,x_n)$ от $n$ переменных, определенная на некотором количестве $M$ наборов.

Вполне очевидно, что в зависимости от значения $M$, задача доопределения функции так, чтобы она удовлетворяла SAC, может быть решена с некоторой вероятностью. Причем, чем ближе значение $M$ к максимальному числу наборов – $2^n$, на которых определена функция, тем меньше упомянутая вероятность.

Исходя из этого, основная цель разрабатываемого подхода – получение функций, в наибольшей степени удовлетворяющих критерию SAC, а также исследование зависимости лавинного эффекта от наперед заданного $M$.

В основу разрабатываемой методики положен тот факт, что функция $f(x_1,...,x_n)$ удовлетворяет критерию SAC, если она и ее частичные производные $df/dx_1,...,df/dx_n$ являются балансными, т.е. количество нулей и единиц в таблице истинности функции с ее производными одинаковое.

Исходной для задачи синтеза является заполненная на $M$ наборах таблица истинности функции.

Суть предлагаемой методики состоит в целенаправленном заполнении таблицы истинности функции $f(x_1,...,x_n)$ так, чтобы в максимальной степени сбалансировать (уравнять) число единичных и нулевых значений как самой функции, так и каждой из ее частичных производных $df/dx_1,...,df/dx_n$.

Предлагаемый способ доопределения функции с целью обеспечения ее лавинных свойств сводится к следующей последовательности действий:

1) По известным значениям функции $f(x_1,...,x_n)$ вычисляются значения частичных производных $df/dx_1,...,df/dx_n$. Для определения производной $df/dx_j = \varphi(x_1,..,x_{i-1},x_{i+1},..x_n)$ по переменной $x_j$, $j\in\{1,..,n\}$ анализируются все $2^{n-1}$ наборов переменных $x_1,..,x_{i-1},x_{i+1},..,x_n$. Каждому из этих наборов соответствует два набора всех $n$ переменных: $x_1,..,x_{i-1},0,x_{i+1},..,x_n$ и $x_1,..,x_{i-1},1,x_{i+1},..,x_n$. Если, при расчете производной, функция $f(x_1,...,x_n)$ определенна на обоих наборах и имеет одинаковые значения, то значение производной принимает нулевое значение, если функция имеет разные значения, то значение производной принимает единичное значение. В случае, когда функция определена только на одном из двух наборов, соответствующее значение производной учитывается как частично определенное значение. Для обозначения частично определенного значения производной вводится символ '*'. Если функция не определена на обоих наборах, то и значение производной – не определенное и обозначается как '-'.

2) Для оценки возможных вариантов значений функции $f(x_1,...,x_n)$ каждое не определенное значение функции $f(x_1,...,x_n)$ заменяется нулевым и единичным значениями, и вычисляется оценка влияния такой замены на свойства сбалансированости как самой функции, так и ее производных.

3) Функция доопределяется значением на наборе, интегральная оценка которых имеет максимальное значение.

4) Производится коррекция значений функций и ее производных с учетом вновь определенного значения функции на выбранном наборе. Если значения функции определены не на всех наборах – возврат на повторное выполнение пп.2., иначе – конец.

Изложенное иллюстрируется следующим примером. Пусть задана частично-определенная булева функция от четырех переменных: $f(x_1,x_2,x_3,x_4)$ (далее $f$), которая определена на $M$=8 наборах. Таблица истинности частично-заданной функции представлена в виде последовательности ее значений на наборах переменных от 0000 до 1111: $f$={0, -, 0, 1, -, -, -, 0, 0, 0, -, -, 1, -, 1, -}, где символом '-' обозначены не определенные значения функции.

Следуя методике, поэтапно доопределяется частично-заданная булева функция так, чтобы она удовлетворяла критерию SAC.

Этап 1: Выполняется вычисление производных по заданным значениям функции *f*.

***Табл. 1. Таблица истинности частично-определенной функции f и ее производных***

| $x_1 x_2 x_3 x_4$ | $f$ | $df/dx_1$ | $df/dx_2$ | $df/dx_3$ | $df/dx_4$ |
|---|---|---|---|---|---|
| 0000 | 0 | 0 | * | 0 | * |
| 0001 | - | * | - | * | * |
| 0010 | 0 | * | * | 0 | 1 |
| 0011 | 1 | * | 1 | * | 1 |
| 0100 | - | * | * | - | - |
| 0101 | - | - | - | * | - |
| 0110 | - | * | * | - | * |
| 0111 | 0 | * | 1 | * | * |
| 1000 | 0 | 0 | 1 | * | 0 |
| 1001 | 0 | * | * | * | 0 |
| 1010 | - | * | * | * | - |
| 1011 | - | * | - | * | - |
| 1100 | 1 | * | * | 0 | * |
| 1101 | - | - | - | - | * |
| 1110 | 1 | * | * | 0 | - |
| 1111 | - | * | 1 | - | - |

Как указывалось выше, определенные значения производных обозначаются ’0‘, если соответствующие значения функции одинаковы, или ’1‘, если значений функции разные. Частично определенные значения производных обозначаются символом ’*‘, неопределенные – ’-‘. Значения производных для функции *f* приведены в таблице 1.

Этап 2: итерационно, вместо каждого не определенного значения функции подставляются нуль и единица (пробные значения) и вычисляются производные, значения для которых изменились в результате подстановки (таблица 2).

Затем оцениваются результаты замены. Для этого подсчитывается количество единиц и нулей функции и ее производных. В функции *f* определено три единицы и пять нулей, то есть на текущем шаге итерации при выборе следующего значения функции необходимо отдать преимущество единице.

Для производной $df/dx_1$ количество нулей равно двум, а количество единиц равно нулю, значит, приемлемым есть единичное значение. По аналогии определяется, что для производной $df/dx_2$ требуемое значение нуль, для $df/dx_3$ – единица, а для $df/dx_4$ – допустимо любое значение, так как количество единиц и нулей равно. В результате изложенного, формируется таблица 3 оценок пробных значений функции и производных.

***Табл. 2. Таблица пробных значений функции f и ее производных***

| $x_1 x_2 x_3 x_4$ | $f$ | $df/dx_1$ | $df/dx_2$ | $df/dx_3$ | $df/dx_4$ |
|---|---|---|---|---|---|
| 0001 | 1 | 1 | * | 0 | 1 |
| | 0 | 0 | * | 1 | 0 |
| 0100 | 1 | 0 | 1 | * | * |
| | 0 | 1 | 0 | * | * |
| 0101 | 1 | * | * | 1 | * |
| | 0 | * | * | 0 | * |
| 0110 | 1 | 0 | 1 | * | 1 |
| | 0 | 1 | 0 | * | 0 |
| 1010 | 1 | 1 | 0 | 1 | * |
| | 0 | 0 | 1 | 0 | * |
| 1011 | 1 | 0 | * | 1 | * |
| | 0 | 1 | * | 0 | * |
| 1101 | 1 | * | 1 | * | 0 |
| | 0 | * | 0 | * | 1 |
| 1111 | 1 | 1 | * | * | 0 |
| | 0 | 0 | * | * | 1 |

В этой таблице приемлемые значения обозначаются символом ’+‘, неприемлемые – ’-‘, допустимые – ’×‘.

Этап 3: Анализируя значения таблицы 3, определяется набор, на котором наибольшее количество требуемых значений. Количество ’+‘ в первой строчке равно двум, во второй – единице и так далее. Наибольшее количество приемлемых значений в девятой строчке для варианта f(1010)=1, то есть выбранное значение функции *f* записывается в таблицу истинности и считается определенным вариантом.

Этап 4: выполняется коррекция значений функций и ее производных с учетом определенного значения функции на выбранном наборе.

Перечисленные этапы повторяются до тех пор, пока в функции *f* не будет определенна на всех $2^n$ наборах. Результатом выполнения примера есть следующая булева функция $f$ ={0, 1, 0, 1, 1, 0, 0, 0, 0, 0, 1, 1, 1, 0, 1, 1}, которая удовлетворяет критерию SAC.

***Табл. 3. Таблица оценок пробных значений функции f и ее производных***

| $x_1\,x_2\,x_3\,x_4$ | $f$ | $df/dx_1$ | $df/dx_2$ | $df/dx_3$ | $df/dx_4$ |
|---|---|---|---|---|---|
| 0001 | + | + | × | - | × |
| | - | - | × | + | × |
| 0100 | + | - | - | × | × |
| | - | + | + | × | × |
| 0101 | + | × | × | + | × |
| | - | × | × | - | × |
| 0110 | + | - | - | × | × |
| | - | + | + | × | × |
| 1010 | + | + | + | + | × |
| | - | - | - | - | × |
| 1011 | + | - | × | + | × |
| | - | + | × | - | × |
| 1101 | + | × | - | × | × |
| | - | × | + | × | × |
| 1111 | + | + | × | × | × |
| | - | - | × | × | × |

## Анализ влияния меры определенности функции на лавинные свойства

Важным аспектом практического использования разработанной методики доопределения частично-заданной булевой функции для обеспечения максимума полной и дифференциальной энтропии играет установление зависимости между мерой априорной определенности функции и возможностью решения указанной выше задачи.

В качестве меры определенности значений функции выступает ξ – отношение числа $M$ наборов, на которых функция задана к общему их числу – $2^n$, выраженное в процентах: $\xi = M \cdot 2^{-n} \cdot 100\%$.

Ясно, что чем больше значение ξ, тем меньше вероятность $P_{SAC}$ того, что возможно доопределение функции такое, что она будет балансной и удовлетворять критерию *SAC*. Очевидным, также является и то, что указанная вероятность зависит и от числа $n$ переменных, на которых определена функция.

К настоящему времени не существует аналитического выражения для исчисления количества булевых SAC-функций от $n$ переменных [1].

В качестве грубой оценки их числа используют мультипликативную модель, которая предполагает, что свойства лавинности по каждой из переменных не зависимы. Исходя из упомянутой модели, вероятность $P_n$ того, что функция $f(x_1,x_2,\ldots,x_n)$ удовлетворяет SAC по всем $n$ переменным определяется как произведение вероятностей того, что функция удовлетворяет SAC по каждой из переменных:

$$P_n = P_{x1} \cdot P_{x2} \cdot \ldots \cdot P_{xn} = P_{x1}{}^n \qquad (4)$$

Число $K_1$ функций, удовлетворяющих SAC по одной переменной, например $x_i$, $i \in \{1,..,n\}$ достаточно просто вычислить, используя известные формулы комбинаторики. Действительно, можно рассмотреть $2^{n-1}$ возможных значений $n$-1 переменной, исключая $x_i$. Для того, чтобы функция была балансной и удовлетворяла SAC по переменной $x_i$, необходимо, чтобы на половине ($2^{n-2}$) этих пар наборов функция при изменении $x_i$ меняла свое значение на противоположное, на одной четвертой ($2^{n-3}$) пар наборов принимала строго единичное значение вне зависимости от $x_i$ и на оставшихся ($2^{n-3}$) наборах принимала нулевое значений. Выбор пар наборов, на которых функция меняет свое значение может быть произведен $C_{2^{n-1}}^{2^{n-2}}$ способами, среди оставшихся $2^{n-2}$ пар наборов выбор $2^{n-3}$ на которых функция принимает единичное значение, можно выполнить $C_{2^{n-2}}^{2^{n-3}}$ способами. Кроме того, на каждом из пары $2^{n-2}$ наборов возможно 2 варианта значения функции. Таким образом, общее количество $K_1$ балансных SAC-функций по одной переменной определяется формулой:

$$K_1 = C_{2^{n-1}}^{2^{n-2}} \cdot C_{2^{n-2}}^{2^{n-3}} \cdot 2^{n-2} \approx \frac{2^{3 \cdot n + 2.5}}{\pi} \qquad (5)$$

Соответственно, вероятность $P_{x1}$ того, что балансная функция удовлетворяет SAC по одной фиксированной переменной равно:

$$P_{X1} = \frac{K_1}{2^{2^n}} \approx \frac{1}{\pi} \cdot 2^{-2^n + 3 \cdot n - 2.5} \qquad (6)$$

Из формул (4) и (6) следует, что с ростом числа $n$ переменных вероятность $P_n$ того, что функция $f(x_1,x_2,\ldots,x_n)$ удовлетворяет SAC по всем $n$ переменным уменьшается.

Зависимость вероятности $P_{SAC}$ того, что в результате предложенной процедуры доопределения частично-заданная на $M = \xi \cdot 2^n$ наборах функция станет балансной и будет удовлетворять *SAC*, от значения ξ носит достаточно сложный характер. Это обусловлено тем, что на успешность доопределения функции значительное влияние оказывает не только число $M$ наборов, на которых функция задана, но и от выбора указанных наборов. Ввиду указанных обстоятельств зависимость $P_{SAC}$ от ξ исследовалась экспериментально. Результаты этих исследований приведены в таблице 4.

***Таблица 4. Зависимость успешности (в процентах) разработанной методики от влияющих на нее факторов.***

| ξ | δ | Результативность в процентах | | |
|---|---|---|---|---|
| | | *n*=4 | *n*=6 | *n*=8 |
| 30 | 0 | 85.5 | 49,8 | 30.4 |
| | 1 | 85.9 | 55.5 | 54.9 |
| | 2 | 86.4 | 62.2 | 88.3 |
| | 3 | 87.3 | 69.2 | 100 |
| | 4 | 87.6 | 75.7 | 100 |
| | 5 | 88.1 | 82.4 | 100 |
| 40 | 0 | 83.7 | 49.1 | 25.6 |
| | 1 | 84.2 | 55.7 | 53.7 |
| | 2 | 84.6 | 62.4 | 87.3 |
| | 3 | 85.2 | 69.3 | 100 |
| | 4 | 85.8 | 75.6 | 100 |
| | 5 | 86.3 | 82.2 | 100 |
| 50 | 0 | 72.2 | 47,4 | 23.4 |
| | 1 | 72.9 | 54.2 | 52.6 |
| | 2 | 73.7 | 61.1 | 86.2 |
| | 3 | 74.8 | 67.8 | 100 |
| | 4 | 75.7 | 74.6 | 100 |
| | 5 | 76.7 | 81.4 | 100 |
| 60 | 0 | 47.5 | 43.1 | 22.4 |
| | 1 | 49.1 | 50.3 | 54.7 |
| | 2 | 50.6 | 57.4 | 90.1 |
| | 3 | 52.2 | 64.6 | 100 |
| | 4 | 53.8 | 71.7 | 100 |
| | 5 | 55.3 | 78.9 | 100 |
| 70 | 0 | 34.3 | 25,4 | 20.3 |
| | 1 | 35.1 | 30.5 | 53.5 |
| | 2 | 36.8 | 36.9 | 86.6 |
| | 3 | 38.6 | 43.4 | 100 |
| | 4 | 40.4 | 49.9 | 100 |
| | 5 | 42.2 | 56.3 | 100 |
| 80 | 0 | 16.3 | 7,5 | 14.4 |
| | 1 | 18.1 | 12.2 | 38.0 |
| | 2 | 20.2 | 17.2 | 61.7 |
| | 3 | 22.1 | 22.1 | 85.6 |
| | 4 | 23.9 | 27.3 | 90.1 |
| | 5 | 25.7 | 32.2 | 100 |

При нарушении защиты данных, построенной на основе булевых SAC-функций, используются методы линейного и дифференциального криптоанализа, базирующиеся на статистической обработке. Это делает целесообразным учет при анализе результативности разработанной процедуры доопределения функций не только строго SAC-функций, но и функций, близких к ним, и статистически не отличимых от них.

Для оценки степени близости булевой функции $\varphi(x_1,\ldots,x_n)$ к SAC-функции используется мера δ, численно равная отношению минимального числу $R$ наборов, изменение значений функции $\varphi(x_1,\ldots,x_n)$ на которых делает ее удовлетворяющей критерию SAC к общему числу возможных наборов $2^n$:

$$\delta = R\cdot 2^{-n}\cdot 100\%. \qquad (7)$$

Результаты исследований – зависимости выраженной в процентах статистической успешности (то есть получения функции, удовлетворяющей SAC с погрешностью δ) предложенной методики доопределения булевых функций от меры ξ предварительной ее заданности, числа $n$ переменных и погрешности δ приведены в таблице 4.

## Выводы

В результате проведенных исследований разработана методика итеративного доопределения частично-заданных булевых функций, обеспечивающая их балансность и лавинные свойства, важные с точки зрения использования таких функций в системах защиты информации.

Предложенная методика доведена до уровня готовых к практическому использованию программных продуктов. Экспериментальными и теоретическими исследованиями доказано, что предложенная методика обеспечивает эффективное решение новой задачи автоматизированного проектирования компонент систем защиты информации, позволяет на 1-2 порядка уменьшить время синтеза частично-заданных лавинных функций по сравнению с перебором.

Разработка может быть использована для создания перспективных средств защиты информации повышенной производительности.

## Список литературы


1. Марковский А.П., Эль-Хами И., Рябуха Л.Р. “Метод одержання булевих балансних функцій, які задовольняють критерію чіткого лавинного ефекту“, Наукові Вісті НТУУ “КПІ“, 2001, № 2, с.31-40.
2. Kurosawa K., Satoh T. Design of SAC/PC(l) of Order k Boolean Functions and Three Other Cryptographic Criteria. // Advanced in Cryptology – Eurocrypto’97 Proceeding, Lecture Notes in Computer Science 1233 – 1997-P.433-449.
3. Самофалов К.Г., Марковский А.П. Комбинаторный подход к получению булевых функций, обладающих строгим лавинным эффектом // Электронное моделирование.- 2004,- Том. 26, - № 3, - с.27-40


Поступила в редакцию 8.12.2009

УДК 681.3.06

*КУЗНЕЦОВ А.В*

# СПОСОБ РЕКОНСТРУИРОВАНИЯ ПРЕДЛОЖЕНИЙ ЕСТЕСТВЕННОГО ЯЗЫКА ПО ГРАММАТИЧЕСКИМ ПРИЗНАКАМ

Предложен подход к реконструированию предложений естественного языка в Прологе с использованием специализированного словаря, включающего допустимые формы частей речи с набором грамматических признаков. Способ реконструирования ориентирован на системы анализа флективных языков. В частности, приводимый пример демонстрирует обработку предложений для русского языка.

Approach to reconstructing of natural language sentences in Prolog using specialized dictionary that include possible form of words with set of grammatical signs is offered. The reconstructing method is oriented to systems for analysis of inflexional languages. In particular, made an example demonstrates treatment of Russian sentences.

## Останов логической регрессии в DCG

В стандартном Прологе реконструирование предложений естественного языка, определяемого формализмом DCG (Definite Clause Grammar), осуществляется простым вызовом обратных действий (далее – *логическая регрессия*) решения задачи контекстного анализа. Такой контекстный анализ предполагает непосредственную проверку словарной формы текущей лексемы на допустимость в предложении, что является приемлемым решением для аналитических языков, имеющих ограниченное количество словарных форм. Иначе обстоит дело с флективными языками, включающих большое разнообразие словарных форм для основных частей речи (существительных, прилагательных и глаголов), что делает неприемлемым эффективный контекстный анализ на основе существующих решений DCG.

В [1] была рассмотрена упрощенная *модель контекстов* лексем на основе типовой декларативно-процедурной семантики Пролога и продемонстрирован пример обработки согласованных предложений при помощи *смысловых агентов*, внедряемых в систему грамматического разбора DCG для русского языка. Эта контекстная модель содержит *базовые формы* лексем, объединенных в допустимые *смысловые отношения*, а остальные формы тех же лексем вместе со своими грамматическими признаками помещены в специализированный словарь. Применение смысловых отношений позволяет компактно решать задачу контекстного анализа предложений с предусмотренным порядком следования частей речи (как это принято в DCG) на основе единой грамматической сигнатуры. Но, в этом случае возникает проблема реконструирования предложений, для которой, в силу особенностей данного подхода к контекстному анализу, требуется самостоятельное решение.

Суть проблемы состоит, с одной стороны – в организации специализированного словаря для анализа флективного естественного языка, а с другой – в использовании смысловых отношений, которые хранят лексемы в базовой форме. Тогда, в ходе анализа предложения для проверки допустимости лексемы в текущем контексте она должна быть приведена к своей базовой форме, после чего соответствующий смысловой агент попытается отыскать такую лексему в заданном отношении. И, наоборот, при реструктурировании предложения необходимо найти подходящую лексему в допустимом смысловом отношении, привести к требуемой согласованной форме и подставить в синтезируемый текст.

Таким образом, **целью статьи** является представление результата разработки специального предиката `'возвр_предлож'`, используемого для эффективного реконструирования предложений русского языка по грамматическим признакам.

Ниже приведен пример из [1], в котором несколько расширен словарь и определен дополнительный предикат `'возвр_предлож'` вместе со вспомогательными предикатами. В этом примере задачу преобразования частей речи из согласованной формы в базовую и обратно решают предикаты с функторами `'существит_индекс'`,`'прилагат_индекс'`, `'прех_глаг_индекс'`, `'неперех_глаг_индекс'`. В каждом из этих предикатов определено по четыре параметра. Первый из них унифицируется с лексемой в согласованной форме. Этот параметр задается при вызове предиката, когда необходимо выполнить преобразование лек-

семы к базовой форме. Тогда в качестве второго параметра возвращаются грамматические признаки лексемы, которые извлекаются из словаря, третий параметр игнорируется, а возвращенное значение четвертого параметра представляет собой индекс в списке текущих форм той части речи, к которой относится анализируемая лексема. По этому индексу отыскивается подходящая лексема в списке базовых форм.

```
%интерпретатор Amzi!Prolog 8.0.6
%проверьте следующие запросы
s1:-'предлож'(['смешная','рыжая','обезьяна',
'держит','банан'],[]).
s2:-'предлож'(['знаменитый','талантливый',
'пародист','рассмешил','публику'],[]).
s3:-'предлож'(['легкомысленный','халатный',
'охранник','прозевал','грабителя'],[]).
s4:-'предлож'(['опасный','вооруженный',
'гангстер','грабит','банки'],[]).
s5:-'предлож'(['обезьяна','держит','банан'],[]).
s6:-'предлож'(['опасный','вооруженный',
'гангстер','угрожает','гражданину'],[]).
s7:-'предлож'(['опасный','вооруженный',
'гангстер','угрожает','гражданам'],[]).
s8:-'предлож'(['опасный','вооруженный',
'гангстер','грабит','инкассаторов'],[]).
s9:-'предлож'(['легкомысленный','халатный',
'охранник','читает','книгу'],[]).
%проверьте запросы на реконструирование предложений
rq1(S):-rs((2,'мужск':'единств':'именит'),
'лицо'('третье'):'прош':'соверш':'действит',
'мужск':'единств':'винит',S).
rq2(S):-rs((2,'мужск':'единств':'именит'),
'лицо'('третье'):'наст':'несоверш':'действит',
'средн':'множеств':'винит',S).
rq3(S):-rs((2,'женск':'единств':'именит'),
'лицо'('третье'):'наст':'несоверш':'действит',
'мужск':'единств':'винит',S).
rq4(S):-rs((1,'женск':'единств':'именит'),
'лицо'('третье'):'наст':'несоверш':'действит',
'мужск':'единств':'винит',S).
rq5(S):-rs((1,'мужск':'единств':'именит'),
'лицо'('третье'):'наст':'несоверш':'действит',
'средн':'множеств':'винит',S).
rq6(S):-rs((2,'мужск':'единств':'именит'),
'лицо'('третье'):'наст':'несоверш':'действит',
          'женск':'единств':'винит',S).
rq7(S):-rs((2,'мужск':'единств':'именит'),
'лицо'('третье'):'наст':'несоверш':'действит',
'мужск':'единств':'дател',S).
rq8(S):-rs((2,'мужск':'единств':'именит'),
'лицо'('третье'):'наст':'несоверш':'действит',
       'средн':'множеств':'дател',S).
:- op(500,xfy,:).
rs((Dn,G:Q:A),R:T:V:D,Ga:Qa:Aa,S):-Obj=
'объект'(_,_,_), bagof(Obj,Obj,St),
'принадлежать'(Sc,St),
'возвр_предлож'((Dn,G:Q:A),R:T:V:D,Ga:Qa:Aa,S,Sc).
'возвр_предлож'((Nd,G:Q:A),R:T:V:D,Ga:Qa:Aa,S,Sc):-
Sc='объект'(N,'определитель'(Db),'функция'(F)),
'длина_списка'(Db,Dn),( Nd > Dn -> fail;
('существит_индекс'(N,_:'единств':'именит':C:GBn,_,
In),'существит_индекс'(_,G:Q:A:C:GBn,Kn,In),
'возвр_функц'(F,Fc,Lo),(Aa='винит'->
('перех_глаг_индекс'(Fc,_,_,Iv,CL),
'перех_глаг_индекс'(_,G:Q:R:T:V:D:C,Kv,Iv,CL));
('неперех_глаг_индекс'(Fc,_,_,Iv,CL),
'неперех_глаг_индекс'(_,G:Q:R:T:V:D:C,Kv,Iv,CL))),
'принадлежать'(Ar,Lo),
'существит_индекс'(Ar,_:'единств':'именит':Ca:GBa,_,
Ia),'существит_индекс'(_,Ga:Qa:Aa:Ca:GBa,Ka,Ia),
(Nd =:= 0 -> S= [Kn,Kv,Ka];
('возвр_опред'(Db,G:Q:A:C,[],Ld),(Nd=:=Dn->
'конкатенация_списков'(Ld,[Kn,Kv,Ka],S);
('принадлежать'(Ld1,Ld),
'конкатенация_списков'([Ld1],[Kn,Kv,Ka],S)))))).
'возвр_функц'(F,Fc,Lo):-
'принадлежать'(Fs,F),Fs=..[Fc,Lo].
'возвр_опред'([],_:_:_:_,Ld,Ld):- !.
'возвр_опред'([D|Dt],G:Q:A:C,Dc,Ld):-
'прилагат_индекс'(D,_,_,I),
'прилагат_индекс'(_,G:Q:A,Kd,I),Dg = [Kd|Dc],
'возвр_опред'(Dt,G:Q:A:C,Dg,Ld).
'предлож'-->'фраза_существит'(G,Q,'именит',C,F),
       'фраза_глаг'(G,Q,_,_,_,_,C,F).
'фраза_существит'(G,Q,A,C,F)-->[W1],[W2],[W3],
{'прилагат_индекс'(W1,G:Q:A,_,I1),
'прилагат_индекс'(W2,G:Q:A,_,I2),
'существит_индекс'(W3,G:Q:A:C:GB,_,I3),
'прилагат_индекс'(_,'мужск':'единств':'именит',K1,I1),
'прилагат_индекс'(_,'мужск':'единств':'именит',K2,I2),
'существит_индекс'(_,GB:'единств':'именит':C:_,K3,I3),
'агент_о'([K1,K2],S),'агент_с'(K3,S,F)}.
'фраза_существит'(G,Q,A,C,F) -->[W],
{'существит_индекс'(W,G:Q:A:C:GB,_,I),
'существит_индекс'(_,GB:'единств':'именит':C:_,K,I),
((is_list(F)->'агент_фо'(K,F);Obj='объект'(K,_,_),
bagof(Obj,Obj,S),'агент_с'(K,S,F)))}.
'фраза_глаг'(G,Q,_,_,_,_,C,F)-->[W],
{'перех_глаг_индекс'(W,G:Q:'лицо'('третье'):_:V:
'действит':C,_,I,Lc),
'перех_глаг_индекс'(_,'неопфм':'неопфм':'неопфм':
'неопфм':V:'неопфм':C,K,I,Lc),'агент_ф'(K,F,Fo)},
'фраза_существит'(_,_,'винит',_,Fo).
'фраза_глаг'(G,Q,_,_,_,_,C,F)-->[W],
{'неперех_глаг_индекс'(W,G:Q:'лицо'('третье'):_:V:
'действит':C,_,I,Lc),
'неперех_глаг_индекс'(_,'неопфм':'неопфм':'неопфм':
'неопфм':V:'неопфм':C,K,I,Lc),'агент_ф'(K,F,Fo)},
'фраза_существит'(_,_,'дател',_,Fo).
'существит_индекс'(W,G:Q:A:C:GB,K,I):-
atomic(W),atomic(I),!,fail.
'существит_индекс'(W,G:Q:A:C:GB,K,I):-
(atomic(W)->('словарь_существит'(G:Q:A:C:GB,L),
'элемент'('принадлежать'(W,L)),'индекс'(W,L,I));
(atomic(I),atomic(G),atomic(Q),atomic(A),
'словарь_существит'(G:Q:A:C:GB,L),
'искать_слово'(K,L,I))),!.
'прилагат_индекс'(W,G:Q:A,K,I):-atomic(W),atomic(I),
!,fail.
'прилагат_индекс'(W,G:Q:A,K,I):-(atomic(W)->
('словарь_прилагат'(G:Q:A,L),
'элемент'('принадлежать'(W,L)),'индекс'(W,L,I));
(atomic(I),atomic(G),atomic(Q),atomic(A),
'словарь_прилагат'(G:Q:A,L),'искать_слово'(K,L,I))),!.
'местоим_индекс'(W,G:Q:R:A,K,I):-atomic(W),
atomic(I),!,fail.
'местоим_индекс'(W,G:Q:R:A,K,I):-(atomic(W)->
('словарь_местоим'(G:Q:R:A,L),
'элемент'('принадлежать'(W,L)),'индекс'(W,L,I));
(atomic(I),atomic(G),atomic(Q),atomic(R),atomic(A),
'словарь_местоим'(G:Q:R:A,L),'искать_слово'(K,L,I))),
!.
'перех_глаг_индекс'(W,G:Q:R:T:V:D:C,K,I,Lc):-
atomic(W),atomic(I),!,fail.
```

```
'перех_глаг_индекс'(W,G:Q:R:T:V:D:C,K,I,Lc):-
(atomic(W)->('перех_глаг'(G:Q:R:T:V:D:C,L,Lc),
'элемент'('принадлежать'(W,L)),'индекс'(W,L,I));
(atomic(I),atomic(G),atomic(Q),(R=='неопфм' ->
true;(R=..[P,Vp],atomic(P),atomic(Vp))),
atomic(T),atomic(V),atomic(D),atomic(C),
'перех_глаг'(G:Q:R:T:V:D:C,L,Lc),
'искать_слово'(K,L,I))),!.
'неперех_глаг_индекс'(W,G:Q:R:T:V:D:C,K,I,Lc):-
atomic(W),atomic(I),!,fail.
'неперех_глаг_индекс'(W,G:Q:R:T:V:D:C,K,I,Lc):-
(atomic(W)->('неперех_глаг'(G:Q:R:T:V:D:C,L,Lc),
'элемент'('принадлежать'(W,L)),'индекс'(W,L,I));
(atomic(I),atomic(G),atomic(Q),(R=='неопфм'-> true;
(R=..[P,Vp],atomic(P),atomic(Vp))),
atomic(T),atomic(V),atomic(D),atomic(C),
'неперех_глаг'(G:Q:R:T:V:D:C,L,Lc),
'искать_слово'(K,L,I))),!.
'длина_списка'([],0):-!.
'длина_списка'([X|Xs],Ln):- 'длина_списка'(Xs,L1),
Ln is L1+1.
'конкатенация_списков'([],L,L).
'конкатенация_списков'([X|L1],L2,[X|L3]):-
'конкатенация_списков'(L1,L2,L3).
'элемент'( P ) :- P, !.
'подмножество'(L,[]).
'подмножество'(L,[X|T]):-
'элемент'('принадлежать'(X,L)),
'подмножество'(L,T).
'принадлежать'( X, [ X|_ ] ).
'принадлежать'( X, [ _|T ] ):-'принадлежать'( X, T ).
'индекс'( X,     [ ], I ) :- I is 0.
'индекс'( X, [ X|T ], I ) :- !, I is 1.
'индекс'( X, [ _|T ], I ) :- 'индекс'( X, T, Ic ),
                  (Ic =:= 0->I is Ic;I is Ic + 1).
'искать_слово'( X, [ _|_ ], 0 ):- !, fail.
'искать_слово'( X, [ X|_ ], 1 ).
'искать_слово'( X, [ _|T ], I ) :- Ic is I - 1,
                 'искать_слово'( X, T, Ic ).
'агент_о'([K1,K2],S):-
Obj='объект'(N,'определитель'(L),F),
bagof(Obj,(Obj,'подмножество'(L,[K1,K2])),S).
'агент_с'(Ls,[],_):-!,fail.
'агент_с'(Ls,S,F):-'функция_субъект'(Ls,S,F).
'функция_субъект'(Ls,[],[]).
'функция_субъект'(Ls,['объект'(Ls,_,'функция'(F))|_],
F).
'функция_субъект'(_,['объект'(_,_,'функция'(_))|St],
_):-'функция_субъект'(Ls,St,F).
'агент_ф'(Lv,F,Fo):-'аргум_функция_субъект'(Lv,F,Fo).
'аргум_функция_субъект'(Lv,[],_):-!,fail.
'аргум_функция_субъект'(Lv,Lf,Fo):- Lf=[Lfh|Lft],
   Lfh=..[Fn,Fco],(Lv=Fn-> Fo=Fco;
   'аргум_функция_субъект'(Lv,Lft,Fo)).
'агент_фо'(Lo,Fo):-'элемент'('принадлежать'(Lo,Fo)).
'объект'('гангстер','определитель'(['вооруженный',
'опасный']),'функция'(['грабить'(['банк','инкассатор',
'казино','склад']),'угрожать'(['гражданин'])])).
'объект'('обезьяна','определитель'(['смешной',
'рыжий']),'функция'(['держать'(['банан','палка'])])).
'объект'('пародист','определитель'(['знаменитый',
'талантливый']),
'функция'(['рассмешить'(['публика'])])).
'объект'('охранник','определитель'(['легкомысленный',
'халатный']),'функция'(['прозевать'(['грабитель']),
'читать'(['книга'])])).
'словарь_существит'('мужск':'единств':'именит':'люди':
'мужск',['гангстер','грабитель','гражданин',
'инкассатор','охранник','пародист']).
'словарь_существит'('мужск':'единств':'родит':'люди':
'мужск',['гангстера','грабителя','гражданина',
'инкассатора','охранника','пародиста']).
'словарь_существит'('мужск':'единств':'дател':'люди':
'мужск',['гангстеру','грабителю','гражданину',
'инкассатору','охраннику','пародисту']).
'словарь_существит'('мужск':'единств':'винит':'люди':
'мужск',['гангстера','грабителя','гражданина',
'инкассатора','охранника','пародиста']).
'словарь_существит'('мужск':'единств':'творит':'люди':
'мужск',['гангстером','грабителем','гражданином',
'инкассатором','охранником','пародистом']).
'словарь_существит'('мужск':'единств':'предлож':
'люди':'мужск',['гангстере','грабителе','гражданине',
'инкассаторе','охраннике','пародисте']).
'словарь_существит'('средн':'множеств':'именит':
'люди':'мужск',['гангстеры','грабители','граждане',
'инкассаторы','охранники','пародисты']).
'словарь_существит'('средн':'множеств':'родит':'люди':
'мужск',['гангстеров','грабителей','граждан',
'инкассаторов','охранников','пародистов']).
'словарь_существит'('средн':'множеств':'дател':'люди':
'мужск',['гангстерам','грабителям','гражданам',
'инкассаторам','охранникам','пародистам']).
'словарь_существит'('средн':'множеств':'винит':'люди':
'мужск',['гангстеров','грабителей','граждан',
'инкассаторов','охранников','пародистов']).
'словарь_существит'('средн':'множеств':'творит':
'люди':'мужск',['гангстерами','грабителями',
'гражданами','инкассаторами','охранниками',
'пародистами']).
'словарь_существит'('средн':'множеств':'предлож':
'люди':'мужск',['гангстерах','грабителях','гражданах',
'инкассаторах','охранниках','пародистах']).
'словарь_существит'('женск':'единств':'именит':'люди':
'женск',['публика']).
'словарь_существит'('женск':'единств':'родит':'люди':
'женск',['публики']).
'словарь_существит'('женск':'единств':'дател':'люди':
'женск',['публике']).
'словарь_существит'('женск':'единств':'винит':'люди':
'женск',['публику']).
'словарь_существит'('женск':'единств':'творит':'люди':
'женск',['публикой']).
'словарь_существит'('женск':'единств':'предлож':
'люди':'женск',['публике']).
'словарь_существит'('женск':'единств':'именит':
'животные':'женск',['обезьяна']).
'словарь_существит'('женск':'единств':'родит':
'животные':'женск',['обезьяны']).
'словарь_существит'('женск':'единств':'дател':
'животные':'женск',['обезьяне']).
'словарь_существит'('женск':'единств':'винит':
'животные':'женск',['обезьяну']).
'словарь_существит'('женск':'единств':'творит':
'животные':'женск',['обезьяной'] ).
'словарь_существит'('женск':'единств':'предлож':
'животные':'женск',['обезьяне']).
'словарь_существит'('мужск':'единств':'именит':
'растения':'мужск',['банан']).
'словарь_существит'('мужск':'единств':'родит':
'растения':'мужск',['банана']).
'словарь_существит'('мужск':'единств':'дател':
'растения':'мужск',['банану']).
'словарь_существит'('мужск':'единств':'винит':
'растения':'мужск',['банан']).
'словарь_существит'('мужск':'единств':'творит':
'растения':'мужск',['бананом'] ).
'словарь_существит'('мужск':'единств':
```

```
'предлож':'растения':'мужск',['банане']).
'словарь_существит'('мужск':'единств':'именит':
'учереждения':'мужск',['банк','склад']).
'словарь_существит'('мужск':'единств':'родит':
'учереждения':'мужск',['банка','склада']).
'словарь_существит'('мужск':'единств':'дател':
'учереждения':'мужск',['банку','складу']).
'словарь_существит'('мужск':'единств':'винит':
'учереждения':'мужск',['банк','склад']).
'словарь_существит'('мужск':'единств':'творит':
'учереждения':'мужск',['банком','складом'] ).
'словарь_существит'('мужск':'единств':'предлож':
'учереждения':'мужск',['банке','складе']).
'словарь_существит'('средн':'множеств':'именит':
'учереждения':'мужск',['банки','склады']).
'словарь_существит'('средн':'множеств':'родит':
'учереждения':'мужск',['банков','складов']).
'словарь_существит'('средн':'множеств':'дател':
'учереждения':'мужск',['банкам','складам']).
'словарь_существит'('средн':'множеств':'винит':
'учереждения':'мужск',['банки','склады']).
'словарь_существит'('средн':'множеств':'творит':
'учереждения':'мужск',['банками','складами'] ).
'словарь_существит'('средн':'множеств':'предлож':
'учереждения':'мужск',['банках','складах']).
'словарь_существит'('средн':'единств':'именит':
'учереждения':'средн',['казино']).
'словарь_существит'('средн':'единств':'родит':
'учереждения':'средн',['казино']).
'словарь_существит'('средн':'единств':'дател':
'учереждения':'средн',['казино']).
'словарь_существит'('средн':'единств':'винит':
'учереждения':'средн',['казино']).
'словарь_существит'('средн':'единств':'творит':
'учереждения':'средн',['казино'] ).
'словарь_существит'('средн':'единств':'предлож':
'учереждения':'средн',['казино']).
'словарь_существит'('средн':'множеств':'именит':
'учереждения':'средн',['казино']).
'словарь_существит'('средн':'множеств':'родит':
'учереждения':'средн',['казино']).
'словарь_существит'('средн':'множеств':'дател':
'учереждения':'средн',['казино']).
'словарь_существит'('средн':'множеств':'винит':
'учереждения':'средн',['казино']).
'словарь_существит'('средн':'множеств':'творит':
'учереждения':'средн',['казино']).
'словарь_существит'('средн':'множеств':'предлож':
'учереждения':'средн',['казино']).
'словарь_существит'('женск':'единств':'именит':
'литература':'женск',['книга']).
'словарь_существит'('женск':'единств':'родит':
'литература':'женск',['книги']).
'словарь_существит'('женск':'единств':'дател':
'литература':'женск',['книге']).
'словарь_существит'('женск':'единств':'винит':
'литература':'женск',['книгу']).
'словарь_существит'('женск':'единств':'творит':
'литература':'женск',['книгой'] ).
'словарь_существит'('женск':'единств':'предлож':
'литература':'женск',['книге']).
'словарь_прилагат'('мужск':'единств':'именит',
['вооруженный','знаменитый','легкомысленный',
'опасный','рыжий','смешной','талантливый',
'халатный']).
'словарь_прилагат'('мужск':'единств':'родит',
['вооруженного','знаменитого','легкомысленного',
'опасного','рыжего','смешного','талантливого',
'халатного']).
'словарь_прилагат'('мужск':'единств':'дател',
['вооруженному','знаменитому','легкомысленному',
'опасному','рыжему','смешному','талантливому',
'халатному']).
'словарь_прилагат'('мужск':'единств':'винит',
['вооруженного','знаменитого','легкомысленного',
'опасного','рыжего','смешного','талантливого',
'халатного']).
'словарь_прилагат'('мужск':'единств':'творит',
['вооруженным','знаменитым','легкомысленным',
      'опасным','рыжим','смешным','талантливым',
'халатным']).
'словарь_прилагат'('мужск':'единств':'предлож',
['вооруженном','знаменитом','легкомысленном',
'опасном','рыжем','смешном','талантливом',
'халатном']).
'словарь_прилагат'('женск':'единств':'именит',
['вооруженная','знаменитая','легкомысленная',
'опасная','рыжая','смешная','талантливая',
'халатная']).
'словарь_прилагат'('женск':'единств':'родит',
['вооруженную','знаменитую','легкомысленную',
'опасную','рыжую','смешную','талантливую',
'халатную']).
'словарь_прилагат'('женск':'единств':'дател',
['вооруженной','знаменитой','легкомысленной',
'опасной','рыжей','смешной','талантливой',
'халатной']).
'словарь_прилагат'('женск':'единств':'винит',
['вооруженную','знаменитую','легкомысленную',
'опасную','рыжую','смешную','талантливую',
'халатную']).
'словарь_прилагат'('женск':'единств':'творит',
['вооруженной','знаменитой','легкомысленной',
'опасной','рыжей','смешной','талантливой',
'халатной']).
'словарь_прилагат'('женск':'единств':'предлож',
['вооруженной','знаменитой','легкомысленной',
'опасной','рыжей','смешной','талантливой',
'халатной']).
'перех_глаг'('неопфм':'неопфм':'неопфм':'неопфм':
'несоверш':'неопфм':'люди',['грабить','читать'],
['люди']).
'перех_глаг'('неопфм':'неопфм':'неопфм':'неопфм':
'несоверш':'неопфм':C,['держать','зевать','смешить'],
Lc):-['животные','люди'] = Lc,
'элемент'('принадлежать'(C,Lc)).
'перех_глаг'('неопфм':'неопфм':'неопфм':'неопфм':
'соверш':'неопфм':C,['продержать','прозевать',
'рассмешить'],Lc):-['животные','люди'] = Lc,
'элемент'('принадлежать'(C,Lc)).
'неперех_глаг'('неопфм':'неопфм':'неопфм':'неопфм':
'несоверш':'неопфм':C,['угрожать'],Lc):-
['животные','люди']=Lc, 'эл-
емент'('принадлежать'(C,Lc)).
'неперех_глаг'('неопфм':'неопфм':'неопфм':'неопфм':
'соверш':'неопфм':C,['проугрожать'],Lc):-
['животные','люди'] = Lc,
'элемент'('принадлежать'(C,Lc)).
'перех_глаг'('мужск':'единств':'лицо'('третье'):
'наст':'несоверш':'действит':'люди',['грабит',
'читает'],['люди']).
'перех_глаг'('мужск':'единств':'лицо'('третье'):
'наст':'несоверш':'действит':C,['держит','зевает',
'смешит'],Lc):-['животные','люди']=Lc,
'элемент'('принадлежать'(C,Lc)).
'перех_глаг'('женск':'единств':'лицо'('третье'):
'наст':'несоверш':'действит':C,['держит','зевает',
'смешит'],Lc):-['животные','люди']=Lc,
```

```
'элемент'('принадлежать'(C,Lc)).
'перех_глаг'('мужск':'единств':'лицо'('третье'):
'прош':'соверш':'действит':'люди',['ограбил'],
['люди']).
'перех_глаг'('мужск':'единств':'лицо'('третье'):
'прош':'соверш':'действит':C,['продержал','прозевал',
'рассмешил'],Lc ):-['животные','люди']= Lc,
'элемент'('принадлежать'(C,Lc)).
'перех_глаг'('женск':'единств':'лицо'('третье'):
'прош':'соверш':'действит':C,['продержала',
'прозевала','рассмешила','пригрозила'],Lc ):-
['животные','люди']= Lc,
'элемент'('принадлежать'(C,Lc)).
'неперех_глаг'('мужск':'единств':'лицо'('третье'):
'наст':'несоверш':'действит':C,['угрожает'],Lc):-
['животные','люди']=Lc,
'элемент'('принадлежать'(C,Lc)).
'неперех_глаг'('мужск':'единств':'лицо'('третье'):
'прош':'соверш':'действит':C,['пригрозил'],Lc ):-
['животные','люди']= Lc,
'элемент'('принадлежать'(C,Lc)).
'неперех_глаг'('женск':'единств':'лицо'('третье'):
'прош':'соверш':'действит':C,['пригрозила'],Lc ):-
['животные','люди']=Lc,
'элемент'('принадлежать'(C,Lc)).
```

В свою очередь, для преобразования лексемы из базовой формы, извлекаемой из смыслового отношения, в необходимую согласованную форму, при вызове упомянутых предикатов первый параметр игнорируется. Во втором параметре задаются грамматические признаки, которым должна соответствовать требуемая части речи, а в четвертом – индекс, который подсчитывается по списку базовых форм. Тогда в качестве третьего параметра возвращается форма части речи, запрошенная в соответствии с грамматическими признаками. Таким образом, в данном случае работа со словарем основана на доступе к требуемой форме части речи по индексу, значение которого равно номеру позиции лексемы в соответствующем списке форм словаря.

Заметим, что при ином подходе к работе со словарем флективного языка, перебор огромного количества сочетаний из различных форм лексем в ходе работы механизма DCG в направлении логической регрессии, неизбежно ведет к крайне неэффективному решению задачи реконструирования предложений. По этой причине в разработанных предикатах преобразования форм предусмотрена проверка наличия значений параметров, содержащих либо форму лексемы, либо индекс. В случае их отсутствия (если запросить реконструирование предложений по правилам DCG в виде свободной переменной и пустого списка), ход логической регрессии блокируется предикатом анализа типа atomic.

Работа примера в соответствии с заданными контекстными моделями дает следующие результаты:

```
?- rq1(S).

S = [опасный, вооруженный, гангстер, ограбил, банк] ;
S = [опасный, вооруженный, гангстер, ограбил, инкассатора] ;
S = [опасный, вооруженный, гангстер, ограбил, склад] ;
S = [халатный, легкомысленный, охранник, прозевал, грабителя] ;
no

?- rq2(S).

S = [опасный, вооруженный, гангстер, грабит, банки] ;
S = [опасный, вооруженный, гангстер, грабит, инкассаторов] ;
S = [опасный, вооруженный, гангстер, грабит, казино] ;
S = [опасный, вооруженный, гангстер, грабит, склады] ;
S = [халатный, легкомысленный, охранник, зевает, грабителей] ;
no

?- rq3(S).

S = [рыжая, смешная, обезьяна, держит, банан] ;
no

?- rq4(S).

S = [рыжая, обезьяна, держит, банан] ;
S = [смешная, обезьяна, держит, банан] ;
no
?- rq5(S).

S = [опасный, гангстер, грабит, банки] ;
S = [вооруженный, гангстер, грабит, банки] ;
S = [опасный, гангстер, грабит, инкассаторов] ;
S = [вооруженный, гангстер, грабит, инкассаторов] ;
S = [опасный, гангстер, грабит, казино] ;
S = [вооруженный, гангстер, грабит, казино] ;
S = [опасный, гангстер, грабит, склады] ;
S = [вооруженный, гангстер, грабит, склады] ;
S = [халатный, охранник, зевает, грабителей] ;
S = [легкомысленный, охранник, зевает, грабителей] ;
no

?- rq6(S).

S = [талантливый, знаменитый, пародист, смешит, публику] ;
S = [халатный, легкомысленный, охранник, читает, книгу] ;
no

?- rq7(S).

S = [опасный, вооруженный, гангстер, угрожает, гражданину] ;

no
?- rq8(S).

S = [опасный, вооруженный, гангстер, угрожает, гражданам] ;
no
?- quit.
```

Здесь вспомогательные предикаты с функтором типа rql используются для упрощения запросов. Эти предикаты обеспечивают вызов предиката с функтором rs, в котором в качестве параметров принимаются значения грамматических признаков частей

речи. Затем, при помощи встроенного предиката bagof в базе знаний собираются все допустимые смысловые отношения, после чего предикат 'возвр_предлож' осуществляет их обработку и синтезирует согласованные предложения.

### Особенности применения грамматических признаков

Для вызова предиката 'возвр_предлож' используются пять параметров, первые три из которых представляют собой группы. Т.е. этот предикат задан определением формулы:

'возвр_предлож'((Dn,G:Q:A),R:T:V:D,Ga:Qa:Aa,S,Sc).

Первая группа параметров в виде структуры без функтора включает следующие признаки:

Dn – количество определителей (прилагательных) в группе существительного DCG;

G:Q:A – род, число и падеж для согласования группы существительного в DCG;

R:T:V:D – лицо, время, вид и залог глагола в DCG;

Ga:Qa:Aa – род, число и падеж существительного в глагольной группе DCG;

S – свободная переменная, используемая для возврата реконструированного предложения;

Sc – переменная, куда при вызове предиката помещен список допустимых смысловых отношений, на основе которых выполняется реконструирование.

По приведенным в предыдущем разделе результатам, в соответствии с запросами, можно наблюдать изменение грамматических признаков в синтезированных и согласованных предложениях. Так, в запросе rq1 группа существительного выводится с двумя прилагательными единственного числа, в именительном падеже мужского рода, а глагол – в третьем лице, прошедшем времени, совершенном виде и действительном залоге. Для существительного глагольной группы заданы: мужской род, единственное число и винительный падеж, так как глагол 'грабить' применяется в качестве переходного и по правилам русского языка требует существительного в винительном падеже. В запросе rq2 изменены грамматические признаки глагольной группы. Для глагола используются: третье лицо, настоящее время, несовершенный вид, действительный залог, а для существительного глагольной группы – средний род, множественное число и винительный падеж.

Аналогичные изменения и согласования частей речи можно наблюдать в синтезированных предложениях, соответствующих остальным запросам, где заданы другие грамматические признаки. Кроме того, в запросах rq4 и rq5 задан вывод одного, а не двух определителей-прилагательных группы существительного, т.е. параметр Dn унифицирован с единицей. В результате – в ходе реконструирования были выведены предложения с одним из возможных определителей группы существительного. Такие предложения синтезированы с учетом однократного предшествования прилагательного во всех вариантах, предусмотренных в смысловых отношениях, что обеспечено работой механизма перебора с возвратом.

Отметим также, что предикат 'возвр_предлож' легко модифицируется. Так, в запросах rq7 и rq8 существительное глагольной группы выводится в дательном падеже в соответствии с логикой применения непереходного глагола 'угрожать'. Вывод дательного падежа предусмотрен в правиле, соответствующем предикату 'возвр_предлож'. Подобная гибкая модификация правил позволяет учитывать тонкости языковой грамматики для синтеза предложений на основе грамматических признаков.

### Роль смысловых отношений

Смысловые отношения представляют собой основу организации логической связи слов при реконструировании предложений, в отличие от правил грамматики естественного языка, которые обеспечивают только грамматическое согласование. Прежде всего, обратим внимание на различие в содержаниях выводимых предложений, хотя все они могут обрабатываться едиными правилами DCG и в своих смысловых отношениях используют один и тот же шаблон для организации структур. Другими словами, использование ограниченного набора правил DCG и небольшого количества смысловых отношений позволяет существенно влиять на содержание как анализируемых, так и реконструируемых предложений в пределах одной и той же грамматической сигнатуры. Для этого достаточно использовать специализированный словарь с грамматическими признаками частей речи и обеспечить обработку лексем естественного языка с учетом значений этих признаков, как при анализе, так и при реконструировании предложений.

Для лучшего представления о назначении смысловых отношений рассмотрим работу предиката 'возвр_предлож' более детально.

Все смысловые отношения определены в виде структур, которые имеют общий функтор 'объект', одинаковую арность и назначе-

ние аргументов [1]. Перед обращением к предикату 'возвр_предлож' в предикате rs вводится переменная Obj, которая унифицируется со структурой 'объект', что обеспечивает создание списка всех таких структур предикатом bagof и унификацию этого списка с переменной St. Далее, предикатом 'принадлежать' из списка St извлекается текущая обрабатываемая структура и унифицируется с переменной Sc, которая вместе с заданными грамматическими признаками передается предикату 'возвр_предлож'.

Работа предиката 'возвр_предлож' начинается с того, что из структуры Sc извлекаются элементы реконструируемого предложения, которые представлены в базовой форме:

N – существительное (субъект действия);

Db – список определителей группы существительного;

F – список структур, представляющих глагольную группу, в которых функтор соответствует глаголу, а в качестве аргумента представлен список существительных (объектов действия).

Тогда, дальнейшая обработка смыслового отношения сводится к преобразованию лексем из базовой формы в согласованную форму по заданным грамматическими признакам и объединение в готовое предложение.

Заметим, что порядок обработки лексем в данном случае не имеет значения. Так, в приведенном примере сначала обрабатывается объект-существительное, затем лексемы глагольной группы и после этого идет перебор возможных определителей-прилагательных. Результат представляет собой реконструированное и согласованное предложение в виде списка лексем, который получен предикатом 'конкатенация_списков'.

## Выводы

1. Эффективность работы механизма DCG, при стандартном подходе к реконструированию предложений флективного естественного языка, резко снижается из-за большого разнообразия форм основных частей речи.

2. Решение рассмотренной задачи упрощается за счет работы со специализированным словарем, который содержит образцы форм частей речи с грамматическими признаками, а также за счет введения допустимых смысловых отношений.

3. Использование грамматических признаков позволяет гибко изменять согласующиеся формы частей речи, учитывая особенности грамматики естественного языка.

4. Применение смысловых отношений дает возможность управлять содержанием реконструируемых предложений, которые представлены единой грамматической сигнатурой в правилах DCG.

## Список литературы

1. Кузнецов А.В. Смысловая аморфность в грамматических сигнатурах предложений естественного языка //Вісник НТУУ «КПІ». Інформатика, управління та обчислювальна техніка: Зб. наук. пр. - К.: Век+. - 2007. - №47.- С.242-254.).

Поступила в редакцию 17.12.2009

УДК 004.2

*ЖАБИН В.И.*

# МЕТОДЫ ПОВЫШЕНИЯ ЭФФЕКТИВНОСТИ ОБМЕНА ДАННЫМИ В МУЛЬТИПРОЦЕССОРНЫХ СИСТЕМАХ

Предлагаются методы ускорения обмена данными в мультипроцессорных системах с общей магистралью, основанные на множественном оконном доступе к локальной памяти процессоров. Рассматривается возможность синхронизации процедур обмена по шине данных и шине управления.

Methods of data transaction acceleration in global-bus multi-processor systems based on multiple window processor local memory access are proposed. Possibility of transaction synchronization through data and control buses is considered.

## Введение

Широкое внедрение автоматических систем управления техническими комплексами и технологическими процессами требует ускоренного проектирования вычислительных средств для решения различных по сложности задач обработки информации в режиме реального времени.

Перспективным направлением создания промышленного ряда систем для управления различными объектами и технологическими процессами является использование масштабируемых параллельных вычислительных систем, построенных на однотипных процессорных модулях.

Модульный принцип построения программных и аппаратных средств позволяет ускорить разработку систем и снизить их стоимость. При таком подходе к организации систем путем масштабирования может быть обеспечена необходимая производительность систем, которая определяется конкретными условиями применения.

Возможность масштабирования наиболее эффективно реализуется на основе магистральной топологии, когда в качестве коммуникационной среды для обмена информацией между процессорными модулями используется общая магистраль. Изменение числа процессорных модулей в этом случае не требует модификации коммуникационной среды системы.

Время обработки информации в параллельных системах зависит не только от эффективности распараллеливания задачи, скорости обработки параллельных ветвей, но и от эффективности процедур обмена данными между процессорными модулями.

Следовательно, важной проблемой повышения эффективности параллельной обработки информации является создание методов и средства ускорения обмена данными между процессорными модулями в мультипроцессорных системах с общей магистралью.

## Архитектура систем с общей магистралью

Наиболее простыми с точки зрения технической реализации являются мультипроцессорные системы с общим адресным пространством [1-7]. Системы с такой архитектурой, построены на базе одинаковых процессоров, то есть являются симметричными мультипроцессорами (symmetric multipro-cessor) или SMP-системами. Общая память используется не только для хранения системной информации и обмена между процессорами, но и для хранения программ процессоров. Процессоры обращаются к памяти за командами, что приводит к непроизводительным затратам времени на ожидание доступа и разрешение конфликтов. Наличие в каждом процессорном модуле кэш-памяти уменьшает число конфликтов, однако в этом случае возникают трудности с обеспечением когерентности данных, что усложняет взаимодействие процессоров.

В системах с такой организацией памяти используется небольшое число процессоров (например, SMP-серверы обычно строятся на основе 2, 4 или 8 процессоров), то есть возможность масштабирования весьма ограничена.

Уменьшение числа конфликтных ситуаций может быть достигнуто за счет использования в каждом процессорном модуле собственной локальной памяти, в которой хранится программа данного процессора [2, 5,

8]. Процессор взаимодействует со своей памятью через локальную магистраль, а с общей памятью через системную (общую) магистраль. Такие системы относятся к классу систем с неоднородным доступом к памяти (NUMA-системы).

Для ускорения обмена данными между процессорами в системах может быть предусмотрен непосредственный доступ каждого процессора к определенной области адресного пространства локальной памяти других процессоров, называемой коммуникационной памятью [8]. За счет организации интерфейса, доступ к такой памяти возможен как со стороны локальной, так и со стороны системной магистрали.

К недостаткам систем с коммуникационной памятью следует отнести то, что с увеличением числа процессорных модулей все больший объем адресного пространства локальной памяти каждого процессора выделяется на коммуникационную память. Сокращение объема индивидуальной локальной памяти в каждом процессорном модуле является одним из важных факторов, ограничивающих число процессоров в системе.

Для устранения указанного недостатка может быть использован оконный механизм доступа к локальной памяти каждого процессора со стороны системной магистрали [2]. Устройство доступа, реализующее механизм окна, подключается между системной и локальной магистралями. В адресном пространстве общей памяти для каждой локальной памяти выделяется адрес для регистра адреса и регистра данных устройства доступа. В регистр адреса записывается начальный адрес локальной памяти, к которой осуществляется доступ со стороны системной магистрали. Передача данных (чтение или запись) осуществляется через регистр данных. При этом адрес в регистре адреса модифицируется (например, прибавляется или вычитается константа).

Для сравнительной оценки систем с разной архитектурой на функциональном уровне будем учитывать коэффициент эффективности использования системной магистрали, который определяется по формуле

$$K_{\text{СМ}} = \frac{N_S + N_D}{M}, \qquad (1)$$

где $N_S$ – число обращений к системной магистрали для инициализации и синхронизации процедур обмена (непроизводительные затраты времени); $N_D$ – число обращений к системной магистрали для передачи непосредственно данных; $M$ – число передаваемых слов.

Коэффициент $K_{\text{СМ}}$ определяет среднее число обращений к системной магистрали для передачи одного слова между компонентами системы. При равной скорости обработки параллельных ветвей алгоритмов система с меньшим значением $K_{\text{СМ}}$ будет затрачивать меньше времени на решение задач за счет более быстрого межпроцессорного обмена.

Определение $K_{\text{СМ}}$ в общем случае является нетривиальной задачей и требует разработки программ (если известна система команд) или, по крайней мере, алгоритмов синхронизации и пересылки данных на этапе проектирования системы.

Системы с оконным доступом позволяют эффективно использовать адресное пространство при обмене данными между процессорными модулями, но требуют непроизводительных затрат времени на подготовку обмена. К ним относятся затраты времени на синхронизацию обмена, захвата и инициализацию окна. Для синхронизации применяют флажки в локальной памяти процессоров, а для захвата окна (ограждения от других процессоров на время пересылки данных) используют семафоры в общей памяти.

Исследуя алгоритм обмена данными для систем с оконным доступом в соответствии с (1) можно получить

$$K_{\text{СМ}} = 1 + \frac{N_{\text{ОК}} + 4}{M}, \qquad (2)$$

где $N_{\text{ОК}}$ – суммарное число обращений к магистрали для проверки и захвата семафоров каждым из процессоров.

При большом числе процессоров и большой связности графа обмена данными $N_{\text{ОК}}$ может достигать больших значений и, следовательно, существенно влиять на время решения задач.

Ниже показана возможность сокращения непроизводительных затрат времени обмена данными для систем с оконным доступом к памяти.

## Методы ускорения обмена данными

Для повышения эффективности межпроцессорного обмена в децентрализованных системах можно применить механизм множественного оконного доступа к локальной памяти.

Для описания систем будем использовать MSBC-модель [9], которая учитывает распределение памяти и особенности доступа к адресному пространству процессоров.

Организация системы на функциональном уровне показана на рис. 1. В адресном пространстве общей памяти GM каждому процессору $P_i$ выделено по два адреса для обращения к каждой локальной памяти: $WA_{ij}$ – адрес регистра начального адреса массива, $WD_{ij}$ – адрес регистра данных ($j$ – номер локальной памяти $LM_j$, к которой осуществляется обращение).

К локальной памяти $LM_i$ подключено одно окно с множественным доступом, на которое выделяется $W=2(n-1)$ адресов в общей памяти системы. Число окон с множественным доступом в системе равно числу $n$ процессоров. Укрупненная структурная схема окна с множественным доступом показана на рис. 2.

Регистры начальных адресов образуют блок памяти начальных адресов (ПНА), выполненный в виде СОЗУ.

Перед началом обмена с процессором $P_j$ процессор $P_i$ записывает по адресу $WA_{ji}$ начальный адрес локальной памяти $LM_j$, с которого начинается пересылка массива. Обмен выполняется посредством стандартных циклов обращения процессора $P_i$ к $WD_{ij}$.

Данные при этом пересылаются через блок данных БД. При каждом обращении соответствующий адрес в ПНА увеличиваться или уменьшается, что автоматически делает доступным очередной адрес $LM_j$. Аналогичным образом с $LM_j$ могут одновременно и без предварительного захвата окна работать и любые другие процессоры, так как возникающие конфликты обращения к системной магистрали разрешаются стандартными аппаратными средствами арбитража.

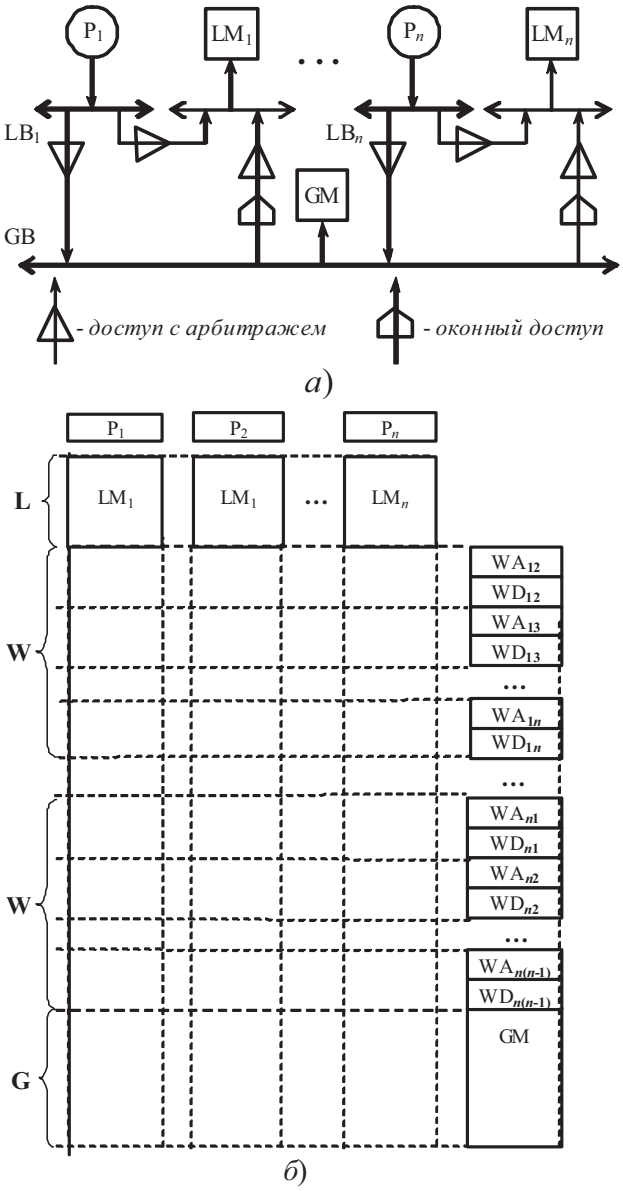



***Рис. 1. Граф управляющих потоков (а) и карта распределения памяти (б) системы с множественным оконным доступом***

Доступ к локальной памяти процессоров не требует предварительного захвата окон. Синхронизацию можно осуществить с помощью флажков, находящихся в локальной памяти процессоров. Процессор, который подготовил в своей локальной памяти массив данных для пересылки настраивает окно процессора, который должен принять данные, захватывая магистраль на один цикл. В следующем цикле он производит установку флажка. Проверка флажка осуществляется процессором в своей локальной памяти, то есть системная магистраль для этого не используется.

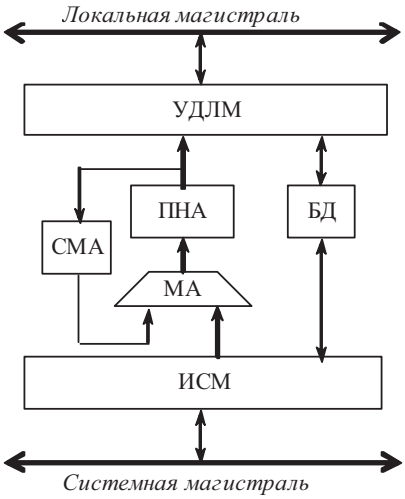



***Рис. 2. Устройство множественного доступа: УДЛМ – устройство доступа к локальной магистрали; ИСМ – интерфейс системной магистрали; БД – буфер данных; ПНА – память начальных адресов; СМА – схема модификации адресов; МА – мультиплексор адреса.***

Если флажок установлен, процессор настраивает необходимое окно (за один цикл), а затем пересылает массив данных в свою локальную память. Для пересылки одного слова системная магистраль захватывается один раз. Завершив пересылку массива, процессор устанавливает флажок в памяти другого процессора. Если флажок записывается по адресу, который автоматически установился в окне, то для этого нет необходимости лишний раз выполнять инициализацию окна. При такой организации обмена $N_S$=4. Действительно, магистраль два раза используется для настройки окна и два раза для установки флажков. Для пересылки данных к магистрали необходимо обратиться $N_D$=$M$ раз.

С учетом (1) коэффициент эффективности использования магистрали можно записать как

$$K_{\text{СМ}}(1) = 1 + \frac{4}{M}. \qquad (3)$$

Сравнивая выражения (2) и (3), видно, что предложенный метод повышает скорость межпроцессорного обмена по сравнению с простым оконным доступом, поскольку отпадает необходимость захвата окон. С увеличением длины передаваемого массива среднее число обращений к системной магистрали для пересылки одного слова стремится к минимальному значению – одному обращению. Однако при пересылке небольших массивов информации непроизводительные расходы в большей степени сказываются на скорость обмена.

Уменьшить непроизводительные расходы можно с помощью следующего метода синхронизации обмена по шине управления. Организация системы поясняется рис. 3.

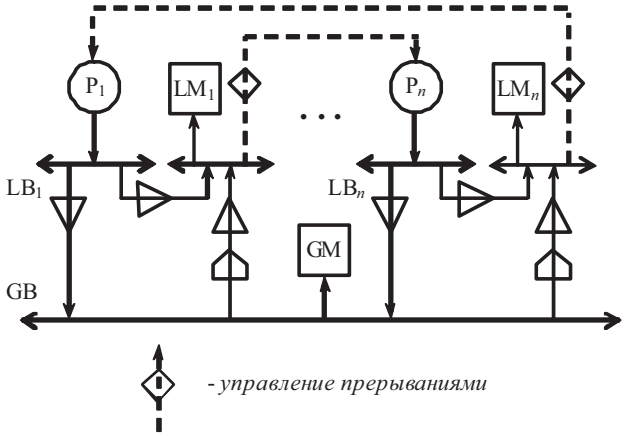



***Рис.3. Система с синхронизацией обмена по шине управления***

Для установки флажков синхронизации системная магистраль в данном случае не используется. Процессор $P_i$ для установки флажка синхронизации в локальной памяти процессора $P_j$ формирует сигнал требования прерывания для этого процессора. Подпрограмма обработки соответствующего прерывания в процессоре $P_j$ устанавливает (сбрасывает) флажок в своей локальной памяти. Состояние флажка проверяется в нужный момент времени основной программой обмена данными. Аналогично процессор $P_j$ изменяет состояние флажка в локальной памяти процессора $P_i$.

Для формирования сигнала требования прерывания выделяются адреса в локальной памяти процессоров. Запись слова по данному адресу через соответствующий интерфейс запускает контроллер прерывания, который реализует процедуру инициализации прерывания.

Таким образом, для пересылки массива только один раз используется системная магистраль для инициализации окна, после чего непосредственно пересылается массив данных. Следовательно, для данного метода получим

$$K_{\text{СМ}}(2) = 1 + \frac{1}{M}. \qquad (4)$$

Очевидно, что количество обращений к системной магистрали в данном случае меньше, чем для предыдущего метода.

## Сравнение эффективности методов

При сравнении выражений (2), (3) и (4), очевидно, что предложенные методы выигрывают в эффективности обмена по сравнению с известным, поскольку при большой связности графа межпроцессорного обмена $N_{ОК}$ в выражении (2) может достигать больших значений.

Сравнивая предложенные методы (см. табл. 1), можно сделать вывод, что синхронизация по шине управления более эффективна при передаче между процессорами небольших массивов данных (примерно до 32 слов). Действительно, для передачи массива из 8 слов системная магистраль используется меньше приблизительно на 30%, а для 16 слов – на 18%.

С увеличением длины передаваемых массивов различия между методами относительно использования системной магистрали становятся незначительными. В этом случае следует отдать предпочтение первому из предложенных методов, поскольку его реализация требует меньших аппаратурных затрат.

***Табл. 1. Сравнение методов***

| $M$ | **4** | **8** | **16** | **32** |
|---|---|---|---|---|
| $K_{СМ}(1)/K_{СМ}(2)$ | 1,6 | 1,33 | 1,18 | 1,1 |

## Выводы

Предложенные методы обмена данными, как показано выше, обеспечивают уменьшение числа обращений к системной магистрали. Это создает предпосылки для ускорения вычислений и, как следствие, расширения области применения мультипроцессорных систем, повышения качества управления и моделирования в режиме реального времени.

## Список литературы


1. Белецкий В.Н. Многопроцессорные и параллельные структуры с организацией асинхронных вычислений. – К.: Наукова думка, 1988. – 240 с.
2. Валях Е. Последовательно-параллельные вычисления. – М.: Мир, 1985. – 456 с.
3. Воеводин В.В., Воеводин Вл.В. Параллельные вычисления. – СПб.: БХВ-Петербург, 2002. – 608 с.
4. Головкин Б.А. Параллельные вычислительные системы. – М: Наука, 1980. – 519 с.
5. Корнеев В.В. Параллельные вычислительные системы. – М.: “Нолидж”, 1999. – 320 с.
6. Майерс Г. Архитектура современных ЭВМ: В 2-кн. Кн. 1. Пер. с англ. – М.: Мир, 1985. – 364 с.
7. Мультипроцессорные системы и параллельные вычисления./ Под. ред. Ф.Г.Энслоу. – М.: Мир, 1976. – 383 с.
8. Siewiorek D. P. A Case Study of C. mmp, Cm* and C.vmp, Part I: Experience with Fault-Tolerance in Microprocessor Systems // Proceedings of the IEEE. – 1978. – Vol. 66, No. 10. – P. 1178-1199.
9. Жабин В.И. Графическое описание архитектуры вычислительных систем // Вісник Національного технічного університету України “Київський політехнічний інститут”. Інформатика, управління та обчислювальна техніка. – К: “ВЕК+”, 2001. – № 36. – С. 80–88.


Поступила в редакцию 8.12.2009

УДК 004.415.2.043

*САЛАПАТОВ В.І.*

# ОСОБЛИВОСТІ ПІДВИЩЕННЯ ЯКОСТІ ПРОГРАМ У СИГНАЛЬНИХ ПРОЦЕСОРІВ

У статті розглядається загальна методика створення ефективних кодів програм для цифрових сигнальних процесорів. Розглядаються архітектурні відмінності від звичайних процесорів наявністю відокремленої пам'яті для програм і для даних, а також наявністю кількох груп регістрів загального призначення для швидкого перемикання контекстів програм.

This article deals with the common methods of the program code creation for digital signal processors. Architectures divisions from usual processors due to existing program memory and data memory and existing groups common registers for quick context program commutation are looked.

Цифрові сигнальні процесори (ЦСП) мають певні архітектурні особливості, які суттєво впливають на оптимальність коду програм. Це, перш за все гарвардська архітектура та одночасне звернення до пам'яті команд та даних. При цьому читання чергового коду команди та адрес операндів виконується в одному машинному циклі, застосовується дворівнева конвеєризація виконання команд. По-друге, закладена у командах апаратна підтримка щодо реалізації циклів, і нарешті, по-третє, в архітектурі ЦСП реалізовано перемикання контексту задач у вигляді примірників груп регістрів загального призначення на кшталт сегментів стану задачі (TSS) у багатозадачних операційних системах [1]. Останнє дозволяє швидке перемикання на підпрограми обслуговування у разі виникнення переривань. Якість програм для ЦСП надзвичайно важлива, оскільки переважна їхня більшість виконується у реальному часі. Тому на рівні мовних засобів створення програм важливо враховувати усі особливості ЦСП для породження якісного коду. Поки що це можливо на рівні Асемблера, що викликає певні труднощі програмування при створенні програм. Поєднання легкості програмування на мові високого рівня з оптимізацією на рівні машинного коду є важливою науково-технічною задачею. Оскільки для систем на базі ЦСП існують жорсткі обмеження на розмір та швидкість виконання програм, то згадана задача є актуальною як з теоретичної, так і з практичної точок зору.

На даний час в системах підготовки програм для цифрових сигнальних процесорів існує загальна методика породження ефективних кодів програм за допомогою мови С, а критичні ділянки програм створювати на мові Асемблер. Така технологія на даний час офіційно застосовується для ЦСП багатьма фірмами, зокрема Analog Device [2], і не є зручною з точки зору швидкої розробки ефективних програм. Задача полягає у розробці методики ефективного використання архітектурних особливостей ЦСП. Якщо за критерій ефективності обрати мінімальне використання пам'яті при максимальній швидкості виконання програм [3], то необхідно врахувати перелічені особливості ЦСП. Таким чином, важливо, щоб при породженні машинного коду компілятори у повній мірі використовували особливості архітектури та системи команд ЦСП.

З урахуванням викладеного задачею даної роботи є внесення доповнень у структуру даних компіляторів, які дозволять здійснити машинно-залежну оптимізацію коду програм. Ці дані дозволять обрати оптимальний код програми та урахувати особливості архітектури та системи команд ЦСП.

У попередніх роботах [3, 4, 5] було зазначено, як за допомогою застосування реляційної моделі даних [6] можна формалізувати процес породження оптимального коду програм. При цьому дані, які зберігаються у програмно доступних регістрах, у процесі компіляції програм можуть відслідковуватись під час створення об'єктного коду [4] і зберігатись у спеціальній базі даних (БД) поточних інформаційних ресурсів. Під час компіляції при породженні машинних кодів програм ці дані мають використовуватись в максимальній мірі, тобто багаторазово, щоб уникати, таким чином, надмірності об'єктного коду. З урахуванням особливостей архітектури ЦСП до бази даних (БД) опису архітектури та системи команд необхідно додати

атрибути типу пам'яті та номеру програмного контексту. Структура відношення опису формату команд при цьому доповнюється атрибутом типу пам'яті MT і набуває наступного вигляду (мал. 1).

| FORMAT | |
|---|---|
| GF * | Номер формату |
| AD* | Тип адресації до операнду |
| VD | Вид операнду |
| NP | Місце в операторі |
| MT | Тип пам'яті |
| TM | Додавання часу |
| LN | Додавання довжини |

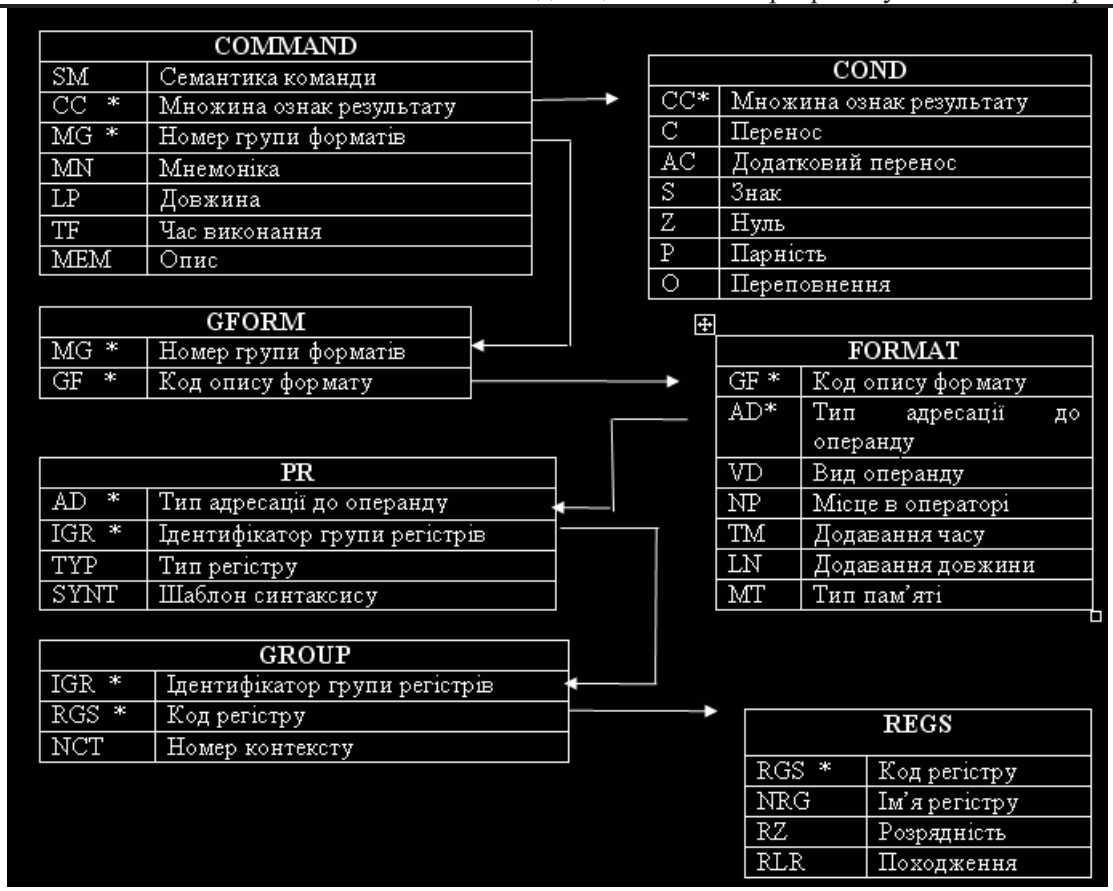
***Мал. 1. Структура відношення FORMAT опису формату команд***

Справа у тому, що за гарвардською архітектурою необхідно враховувати розміщення даних у ЦСП, які можуть розміщуватись, як у пам'яті команд, так і у пам'яті даних, що впливає га швидкість виконання програми. Крім того, при описі програмно-доступних регістрів у відношення опису групи регістрів GROUP треба додати атрибут номеру контексту – NCT, як це зображено на малюнку 2.

| GROUP | |
|---|---|
| IGR * | Група регістрів |
| RGS * | Код регістру |
| NCT | Номер контексту |

***Мал. 2. Структура відношення GROUP опису регістрів***

Значення параметру NCT визначає групу регістрів процесора, тобто поточний номер фрагменту програми – зазвичай, підпрограми переривання. Тоді повна структура опису системи команд та особливостей архітектури процесорів, у тому числі ЦСП із введеними поправками має вигляд, що представлений на малюнку 3. Зірочками позначені атрибути-зв'язки. Для подання інформації про інформаційні ресурси внесені зміни враховуються автоматично, оскільки БД, яка їх описує в частині архітектури, перетинається з базою даних опису системи команд та особливостей архітектури процесора. Для створення загальної БД, яка поряд із описом системи команд та особливостей архітектури процесора, а також описом інформаційних ресурсів (ІР) програм повинна мати БД опису внутрішнього подання програм. Залишилось лише врахувати дані про поточний фрагмент програми, тобто для ЦСП буде додано ще поточний номер контексту. Цей атрибут має вказувати на поточний програмний контекст для коректного використання ІР у програмі [6]. Оскільки йдеться про контекст фрагменту задачі, то зміни тут будуть мінімальні і кожна верхівка синтаксичного дерева програми має бути доповнена атрибутом номера контексту. При цьому номер контексту опису внутрішнього подання програми NCT як зв'язок буде коректно вказувати на ІР конкретного програмного фрагменту. Маючи пов'язані усі три БД компілятор буде породжувати ефективний програмний код. Усі наведені БД приведені до нормальної форми Бойса-Кодда, оскільки з одного боку кожен детермінант відношення є потенційним ключем, а з іншого боку – усунено транзитивні замикання залежності атрибутів. Структура загальної БД для породження ефективного машинного коду представлена на малюнку 4.




***Мал. 3. Структура БД опису системи команд та особливостей архітектури процесора***

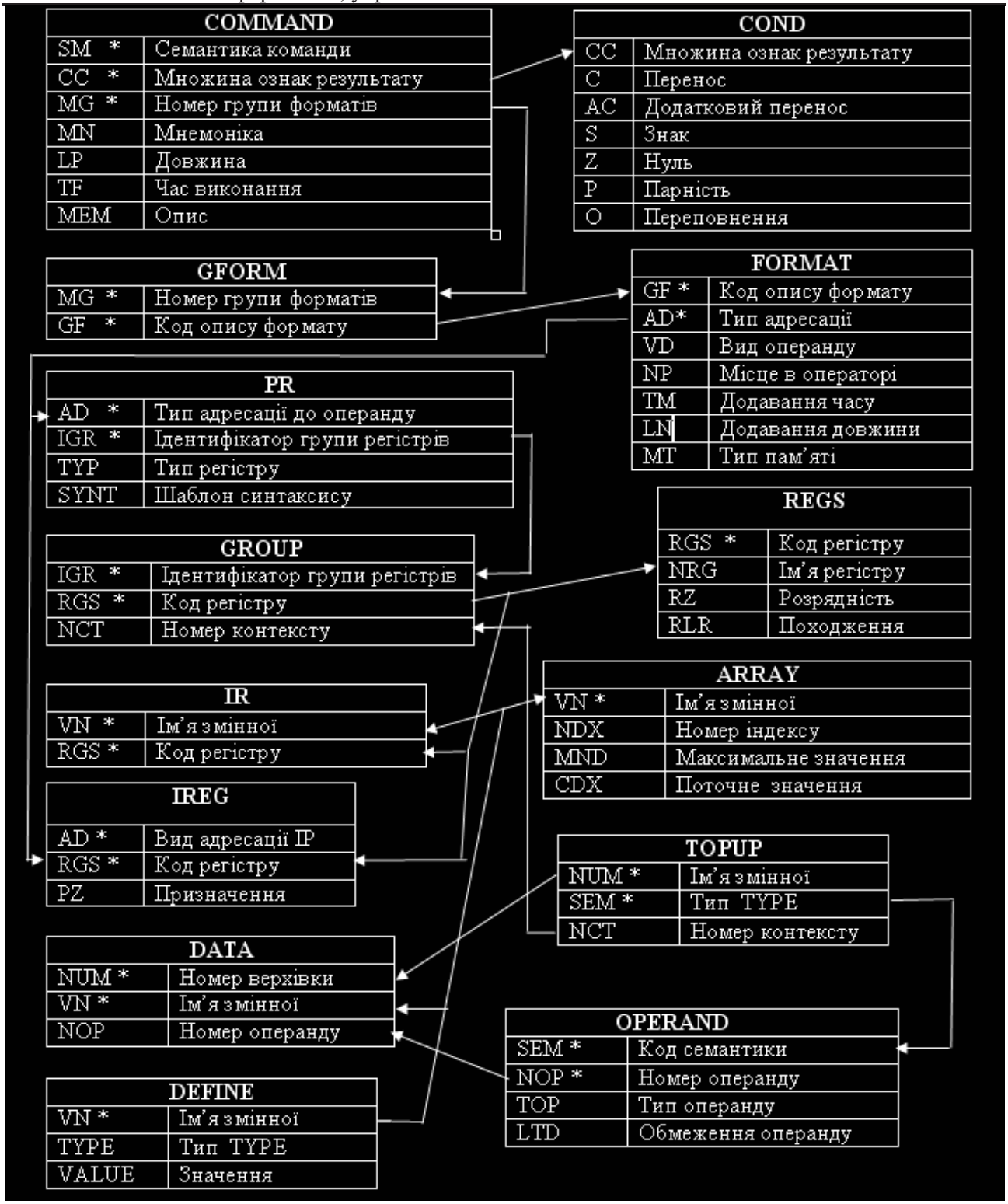
Оскільки кожна мовна конструкція програми розбивається на елементарні оператори, що за своєю семантикою мають наближуватись до відповідних операторів машинних команд, то подальший синтез коду програм зводиться, по суті, до використання в максимальній мірі інформаційних ресурсів. Такий порядок обробки виключає породження зайвих підготовчих машинних команд, а також дозволяє обирати оптимальний формат для машинних команд [4], які безпосередньо реалізують певну мовну конструкцію. В процесі синтезу коду у відношення з інформаційними ресурсами будуть вноситись зміни. В разі переповнення наявних фізичних ресурсів (програмно доступних регістрів) вирішується задача витиснення ресурсів з цього відношення та їхнє заміщення на ті, що потрібні для синтезу коду програми на кшталт аналогічних дій при сторінковій організації пам'яті та сегментації при реалізації віртуальної пам'яті, коли сторінка або сегмент, до яких здійснюється звернення, відсутні у оперативній пам'яті.

Як відомо, основну надмірність коду програм у процесі компіляції складають непотрібні дублюючі команди пересилок даних, а також їхнє збереження та відновлення.

Запропонований метод синтезу об'єктних кодів при компіляції програм передбачає постійне оновлення БД з інформаційними ресурсами за рахунок часткової інтерпретації програми (тобто збереження інформації про певні данні у певних регістрах).

У даній роботі запропонована принципово нова схема та структура даних для породження машинних кодів програм, зокрема, для цифрових сигнальних процесорів, а також мікроконтролерів, яка дозволяє здійснювати машинно-залежну оптимізацію кодів програм у компіляторах за обраним критерієм ефективності.

Для подання даних обрано реляційну модель, яка дозволяє застосовувати над даними операції реляційної алгебри і програмно реалізовувати їх через певні програмні конструкції. На даний час за такою методикою створено компілятор мови С для мікроконтролера AVR [8], який підтвердив свою високу ефективність і знаходиться у дослідній експлуатації.




***Мал. 4. Загальна структура БД схеми компіляції ефективного коду***

## Список посилань


1. А.В.Гордеев, А.Ю.Молчанов. Системное программное обеспечение. – СПб., Питер, 2001. – 736 с.
2. ADSP-218x DSP Instruction Set Reference. Analog Devices, Inc. One Technology Way Norwood, Mass. 02062-9106 Revision 2.0, November 2004. Part Number 82-002000-01.
3. Салапатов В.І. Адаптація процесу синтезу кодів програм під систему команд цільової ЕОМ. Вісник національного технічного університету України «КПІ», Інформатика, управління та обчислювальна техніка. № 43. Київ. 2005 с.
4. Салапатов В. І. Подання даних для синтезу коду у нормалізованому вигляді. Вісник Черкаського інженерно-технологічного інституту, 2001, №3, с.96-99.

5. Салапатов В. І. Структура данных для описания семантики и синтаксиса команд целевой ЭВМ. Вісник національного технічного університету України «КПІ», Інформатика, управління та обчислювальна техніка. № 41. Київ. 2004 с. 191-198.
6. Салапатов В. І. Використання реляційної моделі даних для внутрішнього уявлення програми. Вісник національного технічного університету України «КПІ», Інформатика, управління та обчислювальна техніка. № 42. Київ. 2004 с. 191-198.
7. Э. Таненбаум. Архитектура компьютера. – СПб.; Питер, 2002. - 704 с. 8. Трамперт В. AVR-RISC микроконтроллеры. - К.: МК-Пресс, 2006. – 464 с.

Поступила в редакцию 14.12.2009

УДК 004.413:338.5

*ЛУЦЬКИЙ М.Г.,*
*РЯБОКІНЬ Ю.М.*

# МЕТОД СТВОРЕННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ПУЛЬТА ІНСТРУКТОРА АВІАЦІЙНОГО ТРЕНАЖЕРУ

Пропонується метод для створення програмного забезпечення візуального інтерфейсу пульта інструктора авіаційного тренажеру. Метод базується на понятті ролі при формуванні інтерфейсу, шаблонному проектуванні при створенні програмного забезпечення пульта.

The method for creating software visual interface instructor operating stations of flight simulator is proposed. The method on the concept of interface design role, and patterns design for software development is based.

## Вступ

Авіаційна техніка – коштовна складова транспортної галузі та Збройних сил України. Внаслідок обмеженого фінансування та кризових явищ зараз є значні труднощі щодо розробки та впровадження нової техніки, тому проблема підтримки придатності авіаційної техніки є актуальною [1, 2].

Одним із основних елементів авіаційної техніки є авіаційний тренажер [3]. Використання авіаційних тренажерів для зменшення вартості льотної підготовки, покращення екології набуває особливої актуальності в умовах економічної кризи країни. В зв'язку з цим, зараз до авіаційних тренажерів висуваються більш жорсткі вимоги щодо гнучкості, мобільності та вартісних складових.

Невід'ємною частиною авіаційного тренажера є пульт інструктора, який відображає інформацію про політ та стан тренажеру (рис. 1). Існуючі пульти більшості успадкованих тренажерів [4] реалізовані апаратно, морально застарілі та фізично зношені в процесі тривалої експлуатації і, найчастіше не підлягають ремонту із-за припинення виробництва їх компонентів і запасних частин. Це є причиною заміни апаратного пульта інструктора на нові апаратно-програмні засоби на основі сучасних комп'ютерних технологій [5].

В якості апаратного забезпечення використовуються персональні або промислові комп'ютери загального призначення. Програмне забезпечення пульта включає системне програмне забезпечення (операційні системи, протоколи обміну, драйвери) і прикладне програмне забезпечення, яке реалізує функціональність пульта [6].

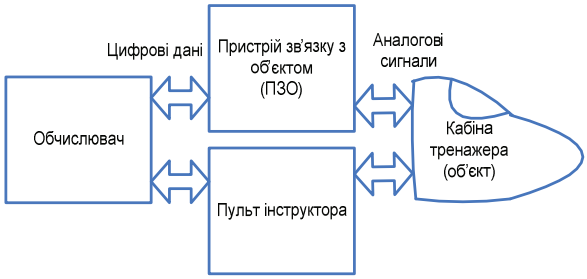



***Рис. 1. Склад авіаційного тренажеру***

При розробці програмного забезпечення пульта інструктора авіаційного тренажера створення програмного забезпечення візуального інтерфейсу є однією із основних задач. Для вирішення цієї задачі в статті пропонується метод, сутність якого полягає в використанні поняття ролі при формуванні інтерфейсу, представленні інтерфейсних елементів як композиції елементів та шаблонному проектуванні при створенні програмного забезпечення пульта.

## Аналіз існуючих підходів щодо створення пультів інструктора авіаційного тренажера

Навчання пілотів на тренажері управляється і контролюється інструктором, в обов'язки якого входить наступне: установка початкових умов польоту, контроль якості пілотування, імітація взаємодії пілотів з диспетчером, управління імітацією відмов літака, розбір помилок і особливостей пілотування (рис. 2) [7]. Для оцінки якості пілотування і розбору польотів інструктор отримує значення відповідних параметрів з фіксацією їх на паперовому чи електронному носіях.

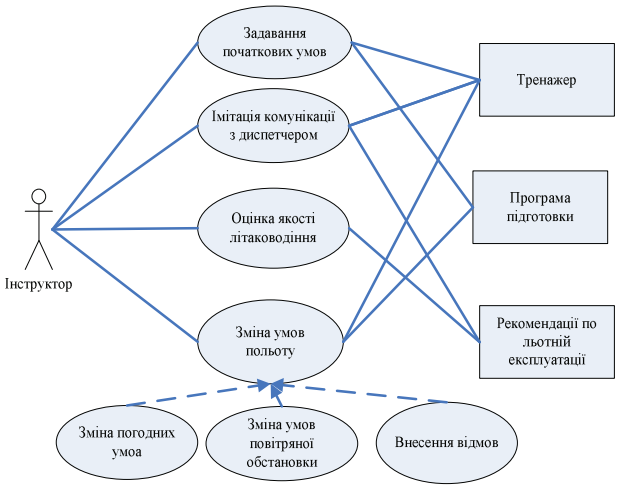



***Рис. 2. Функції інструктора***

Навчання пілотів відбувається відповідно до програм підготовки, які формуються інструкторами в залежності від конкретних задач навчання, і передбачають проведення серійних учбових польотів на тренажерах в заданих умовах. Програми підготовки можуть змінюватися в період експлуатації літака з врахуванням набутого досвіду пілотування. Тому, для ефективного виконування навчальних дій та швидкого аналізу даних інструктору необхідно володіти детальною інформацією, представленою в доступному для огляду виді [3].

На даний час застосовуються, як тренажери поколінь 60-80-х років, так і сучасні тренажери. В процесі експлуатації тренажерів розробки 60-80-х років, уточнити або скоригувати перелік необхідних параметрів дуже складно. До того ж, інформація, що виводиться на індикатори і види управляючих впливів, що здійснюються з пульту жорстко (апаратно) фіксуються при розробці тренажеру [3]. Крім цього, такі пульти незручні у роботі через те, що введення відмов та початкових умов польоту здійснюється вручну за допомогою кнопок, ручок та тумблерів на пульті, а вся інформація надається інструктору одночасно, без врахування того, потрібна вона йому в процесі відпрацювання конкретної вправи чи ні. Це призводить до того, що інструктор змушений досить багато часу витрачати на багатокрокові процедури введення відповідних умов польоту та на відбір інформації, яка йому необхідна на той чи інший момент часу. Отже, внаслідок обмежених технічних можливостей в ергономічному сенсі діяльність інструктора організована не оптимально. Засоби відображення інформації належним чином не пристосовані до його функцій, а будь-які пристрої для обробки цієї інформації відсутні. Тому актуальною є задача принципової зміни інтерфейсу пульта інструктора [8].

Сучасні пульти інструктора (SLZ-242, Ми-8/Ми-17) реалізуються як апаратно-програмні системи на основі комп'ютерів, що включають засоби візуалізації інформації інтерактивної взаємодії з користувачем і інформаційного обміну з іншими компонентами тренажеру. Використання таких систем дозволяє відмовитися від дорогих спеціалізованих пристроїв відображення, вводу і фіксації інформації, що застосовувалися раніше (спеціалізовані індикатори, карти-планшети) і використати комп'ютерні і програмні компоненти [5]. Такі пульти інструктора складаються з одного чи декількох моніторів, миші, клавіатури та принтера. На моніторах за допомогою спеціалізованого програмного забезпечення імітується візуалізація інтерфейсу пульта інструктора (формується зображення індикаторів приладової дошки і відтворюється їх функціонування). До сучасного спеціалізованого програмного забезпечення тренажерів, яке може використовуватися при створенні інтерфейсу пульта інструктора відносять [9]: системи автоматизованого проектування Pro/ENGINEER Wildfire, DeskSim, Segambis.

Інформація на моніторах представляється інтерфейсом виду «вказати-натиснути» (вказати об'єкт на екрані і обрати відривним натиском кнопки миші). За допомогою таких пультів інструктор може швидко створювати звичайні та незвичайні стани в авіаційному тренажері (ланцюжок із загорання двигуна, збої в роботі механізму вивантаження, електричні дефекти, слизька злітно-посадкова смуга, збої в навігаційній системі), зміна погоди в процесі польоту (буря, вітер, блискавка), задання пори року та часу доби, виконання польотів при реальній погоді, якщо комп'ютер підключений до Інтернет, миттєве переміщення літака в будь-яку точку (етап) польоту, переміщення повітряного судна в будь-яку точку земної кулі.

## Підхід до створення інтерфейсу пульта інструктора

Зараз, для створення сучасних пультів інструкторів використовують підхід – «фіксо-

вана функціональність», сутність якого полягає в тому, що склад і функціональність пульта визначаються на основі відомих типових задач, що виконуються інструктором при навчанні пілотів [10]. Основним недоліком такого підходу є те, що після створення пульта, види і форми інформації, що відображається та вводиться не змінюються. Для внесення змін необхідна наявність розробника. Тобто, кожен елемент, що міститься на пульті інструктора, розташовується у чітко визначеному місці і не може бути видалений у разі, якщо він не потрібний за даних умов (наприклад, при відпрацюванні певної польотної ситуації).

В роботі [10] пропонується апробований підхід для подолання зазначеного недоліку. Кожен інтерфейсний елемент інтерфейсу пульта інструктора тренажера (кнопка, індикатор тощо) реалізований у вигляді окремого програмного модулю, який динамічно підключається, та представляє собою .Net-збірку, що реалізує певну функціональність. Це дозволяє з окремих інтерфейсних елементів збирати певний інтерфейс [11].

Як розвиток зазначеного підходу для підвищення гнучкості побудови інтерфейсних елементів та настроювання інтерфейсу під різні види задач в статті пропонується застосування шаблонного проектування [12] і узагальнення інформації на основі поняття ролі.

В авіаційному прикладному домені роль визначається як сукупність функціональних обов'язків, властивих авіаційному персоналу. Авіаційний персонал – це особовий склад авіаційних підприємств, організацій, підрозділів і учбових закладів, що складаються із спеціалістів за відповідними професійними ознаками (члени екіпажу повітряного судна, командно-управляючий, інструкторський, диспетчерський склад і т.п.) [4, 13]. Для кожного спеціаліста визначені типові посадові функціональні обов'язки, вимоги до знань і кваліфікації, тобто його роль [14].

В процесі своєї діяльності авіаційний спеціаліст отримує і забезпечує визначену інформацію через технічні системи. Сукупність видів інформації і способів її відображення, властивій одній авіаційній ролі будемо називати рольовим представленням. Таку інформацію можна представити як набір окремих інформаційних елементів, які групуються в логічно і ергономічно пов'язані групи. Останні будемо називати панелями представлень. Наприклад, до панелей представлення інструктора можна віднести навігаційну приладову панель, панель управління силовою установкою, навколишнє візуальне оточення кабіни літака (рис. 3).

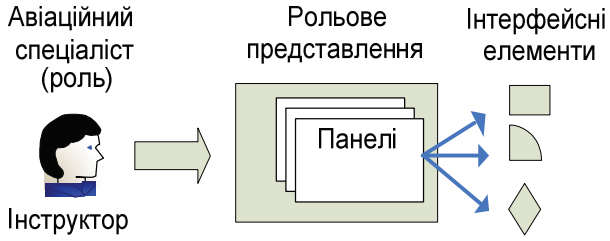



***Рис. 3. Роль, панелі представлення і інтерфейсні елементи***

Індикатори і органи управління пульта тренажера складають користувальницький інтерфейс інструктора, через який він отримує інформацію про політ і управляє тренажером. Кожний індикатор відображає один параметр польоту. Множина індикаторів, доступних інструктору складає реалізацію його рольового представлення і повинна визначатися підмножиною всіх параметрів польоту, і способами їх представлення, прийнятими в авіації, з врахуванням конкретного типу літака. Органи управління пульта визначаються потребами управління тренажером.

Набір і вид інтерфейсних елементів індивідуальні для тренажера літака кожного типу, однак суттєво схожі. Кожний інтерфейсний елемент $ie = (it, pf)$ визначається двома складовими – видом інформації (параметром) $it$ і формою її представлення $pf$. Рольове представлення $rv$ можна описати як множину інтерфейсних елементів, що входять в нього $RV = \{ie_i \mid ie_i \in RV\}$, а роль $R$ в свою чергу – як множину панелей представлення $R = \{RV \mid RV \in R\}$. Вся множина інтерфейсних елементів домену $IE = \{ie_i \mid ie_i \in IE, i = 1..d\}$ визначається як множина двійок $ie = (it, pf)$ всіх його підмножин $IE_r$, визначених для ролей: $R(IE) = \{IE_r \mid IE_r \subseteq IE\}$.

Інтерфейс для ролі інструктора синтезується шляхом вибору та настройки визначених інтерфейсних елементів:

$$r_{instr} = RV_{instr} \subseteq \{ie_i = (it, pf) \mid i = 1..l\}.$$

### Створення інтерфейсних елементів

Інтерфейсні елементи пульта інструктора за призначенням умовно поділяють на два класи управляючі – для вводу інформації

(перемикачі режимів, регулятори, кнопки вводу значень) та індикатори, призначені для її відображення (індикатори швидкості, висоти, зображення карт). В свою чергу, індикатори приймають вигляд шкали-стрілки, цифрових табло та візирів [15].

Кожний інтерфейсний елемент можна розглядати як композицію простих елементів: стрілок, шкал, діапазонів величин, маркування, кожний із яких характеризується певними властивостями і поведінкою (рис. 4).

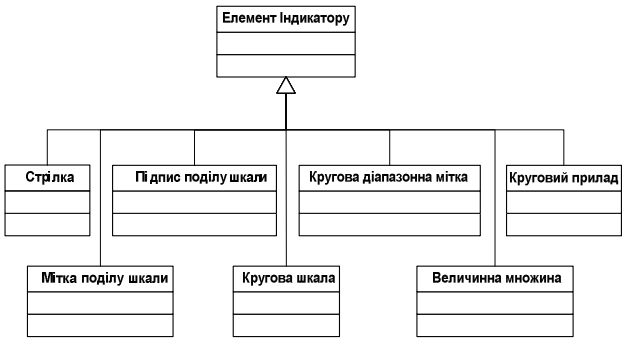
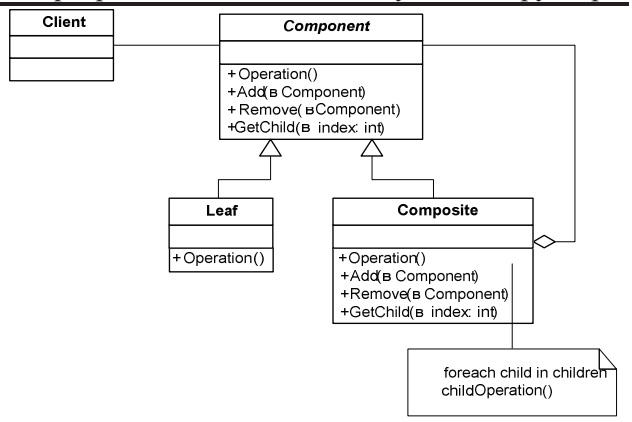


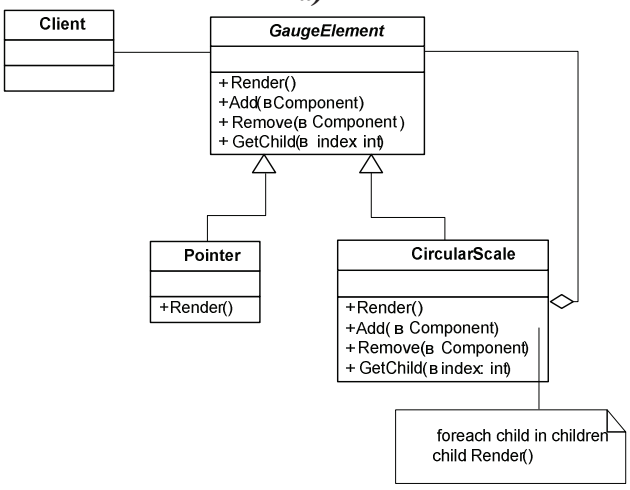

***Рис. 4. Композиція елементів індикатору***

Елементи індикатору представляються класами, що характеризуються певними властивостями. Атрибути класу відображають властивості елементів, які можуть бути спільними (наприклад, колір, радіус, діапазон величин), і індивідуальними (наприклад, довжина стрілки).

Для побудови індикаторів різного виду був використаний шаблон проектування «компонувальник» (composite) [16]. Він об'єднує об'єкти в деревоподібну структуру для представлення ієрархії від окремого до цілого та дозволяє клієнтам звертатися до окремих об'єктів і до їх груп однаково. На рис. 5 (а, б) представлена UML діаграма шаблону і відповідно, окремий випадок індикаторного приладу. GaugeElement (індикатор) є абстрактним класом, що включає дочірні елементи Pointer (стрілка) та CircularScale (кругова шкала). Даний шаблон дає можливість здійснювати звернення до функціональності індикатору в цілому, а також до простих і складних елементів однаково. Так, метод прорисовки Render викликає послідовно цей метод у всіх елементів композиції (Pointer, CircularScale), проходячи по ієрархії класів.




***а)***




***б)***

***Рис. 5. UML діаграма шаблону «компонувальник» для індикаторів***

В запропонованому підході індикатори будуються як композиції елементів (наприклад, круговий прилад включає в себе кругову шкалу, стрілку, мітку та підпис поділу шкали) (рис. 6).

Для того, щоб створити певний інтерфейсний елемент необхідно визначити склад елементів і задати значення їх атрибутів.

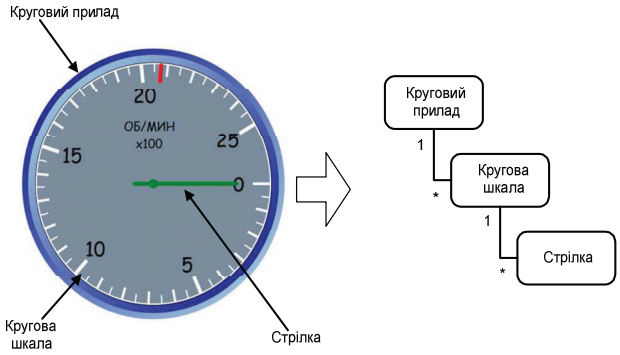



***Рис. 6. Представлення інтерфейсного Елемента***

### Реалізація бібліотеки класів

На основі класифікації і специфікування властивостей елементів в роботі побудована ієрархія класів, на базі якої розроблена бібліотека. Шаблон «компонувальник», викори-

стовується як базова архітектурна концепція ієрархії (рис. 7).

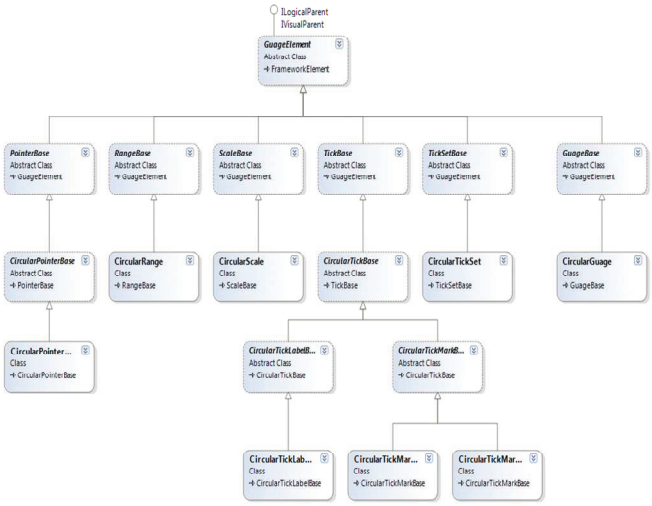



***Рис. 7. Архітектура бібліотеки класів індикатору***

Всі класи бібліотеки наслідуються від базового класу GaugeElement, який має можливість композиції собі подібних елементів.

Бібліотека класів реалізована з використанням технології Windows Presentation Foundation, яка входить в склад .NET Framework 3.0. Для створення інтерфейсних елементів пульта застосовується мова розмітки XAML, що входить до складу Windows Presentation Foundation. Використання Windows Presentation Foundation сумісно з XAML дає можливість змінювати масштабованість індикаторів без втрати якості зображення, динамічно створювати нові індикатори і їх оформлення. Розробник може будувати індикатори відповідного типу, включаючи до композиції різноманітні елементи із бібліотеки.

Створюваний індикатор має деревоподібну структуру піделементів. Наприклад, структура приладу «тахометр» (рис. 8):

1. Круговий прилад (Circular Gauge);
1.1 Кругова шкала (Circular Scale);
1.1.1. Величинна множина (Tick Set);
1.1.1.1. Мітка поділу шкали (Tick Mark);
1.1.1.2. Підпис поділу шкали (Tick Label);
1.1.1.3. Стрілка(Pointer);

Використання XAML робить створення приладів більш зрозумілим і відповідно швидшим, порівняно з застосуванням C# [15].

```
<ctrls:CircularGuage Radius="100" >
 <ctrls:CircularGuage.Scales>
 <ctrls:CircularScale Radius="99" StartAngle="-225" SweepAngle="55" >
    <ctrls:CircularScale.TickSets>
       <ctrls:CircularTickSet Minimum="0" Maximum="5.25" >
         <ctrls:CircularTickSet.Ticks>
        <ctrls:CircularTickMarkMajor ScalePlacement="Inside"          TickMarkType="Rectangle" />
           <ctrls:CircularTickLabelMajor      StartValue="0"    EndValue="5.2"    ValueSign="Positive"
            Interval="1"  ScalePlacement="Inside" />
             </ctrls:CircularTickSet.Ticks>
              <ctrls:CircularTickSet.Pointers>
          <ctrls:CircularPointerNeedle  Background="Green"  Value="{Binding  ElementName=UcElement,
          Path=Value}   />
               </ctrls:CircularTickSet.Pointers>
       </ctrls:CircularTickSet>
    </ctrls:CircularScale.TickSets>
  </ctrls:CircularScale>
 </ctrls:CircularGuage.Scales>
</ctrls:CircularGuage>
```

***Рис. 8. Створення приладу за допомогою мови XAML***

## Висновок

Визначено задачу вдосконалення технології створення програмного забезпечення пультів інструкторів авіаційних тренажерів в напрямку створення можливостей адаптації інтерфейсу до особливостей навчально-тренувального процесу.

Для створення візуального інтерфейсу пульта інструктора запропоновано метод, особливістю якого є можливість динамічної адаптації безпосередньо в процесі експлуатації, що дозволяє підвищити ефективність використання комп'ютерних авіаційних тренажерів в навчально-тренувальному процесі. Метод базується на використанні поняття ролі при формуванні інтерфейсу, класифікації інтерфейсних елементів, специфікуванні їх властивостей та побудові програмного середовища за допомогою шаблону проектування «компонувальник» (composite).

Запропонований метод реалізовано у вигляді бібліотеки класів, на основі якої побудовано гнучку систему індикаторів пульта інструктора тренажеру ТЛ-410М. Аналіз практичного використання такого пульта показав ефективність запропонованого підходу з точки зору якості підготовки авіаційних спеціалістів та перспективність подальших розробок у цьому напрямку.

## Список посилань

1. Луцький М.Г. Підтримка придатності програмного забезпечення при модернізації та створенні авіаційної техніки // Матеріали науково-технічної конф. «Створення та модернізація озброєння і військової техніки МОУ в сучасних тумовах». – Ф. – 2009.

2. Луцький М.Г., Сидоров М.О., Зіатдінов Ю.К. Підтримка придатності інформаційно-програмного забезпечення авіаційної техніки // Матеріали науково-практичної конф. «Актуальні проблеми розвитку авіаційної техніки». – К. – 2009.
3. Меерович Г.Ш., Годунов А.И., Ермолов О.К. Авиационные тренажеры и безопасность полетов. – М.: Воздушный транспорт. 1990. – 343 с.
4. Сидоров Н.А., Недоводеев В.Т., Хоменко В.А., Сердюк И.П., Сидоров Е.Н. Реинженерия программного обеспечения информационно-моделирующих тренажерных комплексов. // Управляющие системы и машины. – К. - 2008. – № 4. – С.68-74.
5. Design of a flight simulator software architecture. Göran Ancker, Jan Wallenberg. – School of Mathematics and Systems Engineering, Växjö University. – 2002. – 91 p.
6. Сидоров М.О., Іванова Л.М., Хоменко В.А. Методологічні принципи реінженерії програмного забезпечення успадкованих авіаційних тренажерів // Матеріали VIII Міжн. наук-техн. конф. АВІА-2007 – К. – 2007. – Т.1. – с. 13.119 – 13.122.
7. Хоменко В.А., Сидоров Є.М. Повторно используемое решение для программного обеспечения пульта инструктора авиационного тренажера// Матеріали науково-практичної конф. «Актуальні проблеми розвитку авіаційної техніки». – К. – 2009. – с. 118.
8. Іваськевич І. О. Ергономіка: Навчальний посібник // - Т.: «Економічна думка», 2002. –165 с.
9. Желонкин В.И. Система поддержки исследований по выбору и оптимизации видов электронной индикации // Вестник Международной Академии проблем человека в авиации и космонавтике. – М.– 2007. №3(26). – с. 33-39.
10. Сидоров Е.Н. Архитектура програмного обеспечения пульта инструктора авиационного тренажера // Матеріали міжнародної науково-практичної конф. аспірантів і студентів «Інженерія програмного забезпечення 2008». – К. – 2009. – с. 45-49.
11. Хоменко В.А., Сидоров Є.М., Артєєв Т. КП «Програмне забезпечення робочого місця інструктора успадкованого авіаційного тренажеру ТЛ-410М». Авторське свідоцтво № 28163.
12. Хоменко В.А, Сидоров Е.Н, Мендзебровский И.Б. Шаблон программного обеспечения устройств связи с объектом авиационных тренажеров. // Проблемы программирования. – К. – 2008. – С. 30 –40.
13. Повітряний кодекс України. Розділ 5. – К. – 1993.
14. Сидоров М.О. Групова динаміка і комунікації. – К.:НАУ, 2009. – 87с.
15. Хоменко В.А., Авраменко Е.А., Рябоконь Ю.Н. Подход к созданию визуального интерфейса пульта инструктора авиационного тренажера. // Матеріали науково-практичної конф. «Актуальні проблеми розвитку авіаційної техніки». – К. – 2009. – с. 117.
16. Ларман К. Применение UML и шаблонов проектирования. – М.:«Вильямс», 2002. – 624 с.

Поступила в редакцию 16.12.2009

УДК 004.415.2.043

*ПУСТОВАРОВ В.І.,*
*КОВАЛЕНКО С.Ю.*

# ОРГАНІЗАЦІЯ СХОВИЩ ДАНИХ ДЛЯ МОВ ОПИСУ ТА МАНІПУЛЯЦІЙ З ІНФОРМАЦІЙНИМИ СУТНОСТЯМИ

Проаналізовані поширені структури даних, методи та засоби накопичення інформаційних сутностей в комп'ютерних ресурсах розв'язання задач інженерії знань. Запропоновані універсальні структури зберігання сховищ аналітичних і управляючих даних та узагальнена архітектура баз знань та їх елементів для управління маніпуляціями при виконанні логічного доведення. Обґрунтовані переваги архітектури з аналітичними компонентами в гнучкості і швидкості при розв'язанні зв'язаних комплексів задач на графових моделях різноманітних проблемних галузей з різними рівнями деталізації інформаційних сутностей.

Methods and tools for storage of informational entities are analyzed for computer recourses of computer task decision. Structures for unified warehouses of analytic and control data and generalized knowledge base architecture are proposed for logical deduction manipulation. Analytical component architecture advantages in flexibility and speed are shown for complex task decision for various problem area graph models with different information entity implementation.

## Вступ

На сьогоднішній день поширюється використання різноманітних середовищ розробки баз знань та комп'ютерних ресурсів обробки, що забезпечують розв'язання комплексів задач проектування та аналізу для заданої проблемної галузі. Разом з тим, сучасні засоби розробки програм [1] спрямовуються на розв'язання задач, які стосуються не лише вузько спеціалізованої проблемної галузі, а також торкаються суміжних галузей і задач. До них зазвичай відносять базові напрями синтезу і аналізу елементів технічних проектів, наприклад, розробку апаратних і програмних ресурсів розв'язання комплексів задач (РРКЗ) в заданій проблемній галузі. Виходячи з перспективного використання таких ресурсів та їх специфікацій [5] в суміжних галузях, доцільним є еволюційний розвиток узагальненого подання інформаційних сутностей об'єктів в комп'ютерних сховищах [4], що передбачає можливість модифікації методів створення, керування та відображення об'єктів.

Для вирішення цього питання потрібно подолати проблему різноманіття форматів представлення та зберігання платформенонезалежних даних, які входять до складу створюваних проектів [2, 3]. На сьогоднішній день практично кожна фірма-розробник РРКЗ має низку власних форматів даних навіть у деяких випадках в рамках однієї проблемної галузі. Таке різноманіття форматів приводить користувачів до вибору між двома шляхами застосування пакетів РРКЗ. Перший варіант полягає у використанні деякого базового пакету РРКЗ з усіма його перевагами і недоліками; інший – у використанні декількох програмних пакетів та часто повторюваних процедурах конвертування різних форматів даних. Крім того, взагалі відсутні механізми взаємодії між середовищами розробки, які використовуються для різних проблемних галузей, не зважаючи на те, що розробки комплексних та узагальнених рішень стають все більш популярними. До того ж, значна різноманітність апаратних та програмних платформ і використовуваних базових продуктів є причиною виникнення величезної кількості програм, необхідних для роботи на базі різних платформ реалізації програм та обладнання. За умов досить швидкого розвитку апаратних і програмних платформ очевидно, що кількість таких продуктів буде збільшуватися і надалі [4].

Все це обґрунтовує необхідність розробки єдиного стандартного або уніфікованого механізму для збереження маніпулювання і представлення інформаційних сутностей, що мають можливість відобразити все різноманіття сутностей в найбільш універсальному і найчастіше неповному, розрідженому поданні. Необхідні уніфіковані подання конструкцій комп'ютерних мов повинні відпові-

дати наступному комплексу вимог, що забезпечують:

- абстрагування скороченням та/або деталізацію додаванням інформаційних сутностей в моделях предметної галузі та їх об'єктах;
- створення узагальненої схеми збереження даних про інформаційні сутності, їх зв'язки, типізацію та іменування їх атрибутів для довільної проблемної галузі;
- полегшення перетворень зв'язків, вхідних та цільових змінних і критеріїв задач на обчислювальні формули та оператори програм;
- розв'язання кінцевої множини задач аналізу, настройки або оптимізації та синтезу комплексів об'єктів проблемної галузі для досягнення поставленої прагматичної мети;
- визначення баз даних і знань для зберігання інформаційних сутностей РРКЗ та результатів їх обґрунтування для будь-яких конфігурацій і комбінацій платформ проектування.

### Постановка задачі

При створенні ефективних сховищ необхідно узагальнити вхідні комп'ютерні мови проектування РРКЗ і внутрішнє подання їх конструкцій та інформаційних сутностей. Більш чітке окреслення цілей та вимог до вдосконалених комп'ютерних мов полягає в розробці універсальних та зручних засобів для подання і перетворення різноманітних даних.

Основними задачами при розширеннях існуючих мов є спрощення сприйняття створюваного сценарію та перебудова орієнтуючої концепції елементів опису мови і керування даними таким чином, щоб включити в систему об'єкт-відображення як універсальний елемент взаємозв'язків. Такі описові елементи мови забезпечать компактне подання моделей та використання спеціальних вбудованих пакетів РРКЗ для зберігання та маніпуляцій даними. Крім того, платформенно-незалежний спосіб взаємодії між внутрішніми частинами сховища та зовнішніми клієнтами забезпечить зменшення кількості програм цільової обробки і спростить механізми взаємодії з даними.

Виходячи з існуючої ситуації, поставимо за мету обґрунтування доцільності реалізації абстрактної мови опису та маніпуляції інформаційними сутностями та програмного пакету для обробки інформаційних сутностей будь-якого типу і виду, в тому числі, управляючих і доказових сутностей. Наявність такого пакету РРКЗ дозволить як більш ефективно використовувати існуючі на сьогодні пакети проектування, так і створювати нові універсальні пакети для розробки програм, моделей, будь-яких інформаційних сутностей, які мають можливість взаємодіяти між собою, використовуючи уніфіковану організацію сховищ.

### Пропозиції щодо розв'язання задачі

При розробці спеціалізованих галузевих програмних комплексів для моделювання об'єктів, середовищ їх існування, тощо детально розглядаються різноманітні об'єкти, їх проблемно-спрямовані описи різних рівнів деталізації, які набувають все більш спеціалізованих форм з ускладненням комплексу. Найбільш логічним шляхом розв'язання цієї проблеми є використання графових структур, поданих в форматі реляційних баз даних.

Для графових структур таке узагальнення не несе в собі суттєвих ускладнень механізму обробки даних, в той час як для реляційних (табличних) баз даних воно призводить до значного ускладнення структур даних або до зменшення ефективності роботи системи. Використання виключно графового подання даних призводить до ускладнення сприйняття та зменшення ефективності роботи сховища при доступі до даних. Тому доцільно знайти компроміс між швидкодією при використанні таблиць та універсальністю графового подання і деталізації даних.

**Концепція сховищ подання інформаційних сутностей.** Концепцію розробки та розвитку будь-якої мови має формувати саме цільове спрямування мови. В нашому випадку основна мета створення узагальненого внутрішнього представлення – це універсалізація описів, реалізація та настроювання РРКЗ. Виходячи з цього, запропоновані сховища мають забезпечувати максимальну гнучкість та багатоваріантність їх використання. Їх організація має бути найпростішою для будь-яких комп'ютерних мов.

Крім того, форма внутрішнього представлення має забезпечити можливість написан-

ня послідовностей дій над інформаційними сутностями та відображеннями. Послідовності кодів, які являють собою цільові обробники, матимуть можливість працювати з даними в середині сховища, не використовуючи зовнішніх ресурсів та сторонніх засобів програмування.

Базові виконавчі оператори комп'ютерних мов включають в себе блочні оператори структурного програмування зі специфічними щодо мов модифікаціями синтаксису і можуть відображуватися графами [2]. Приклад графа підлеглості операторів і операцій для оператора циклу з передумовою мови C++ наведено на рис. 1:

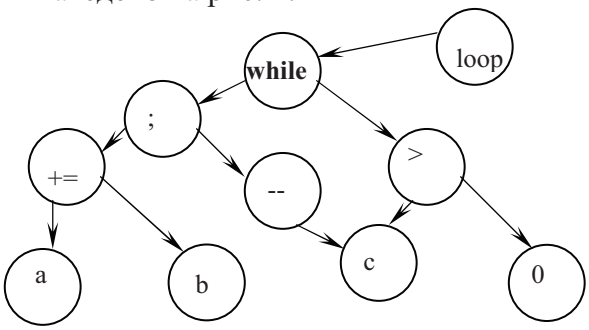



***Рис. 1. Граф підлеглості оператора*** **do**{a+=b-c; --c;}**while**(c>0);

Як видно з рис. 1 узагальнення ациклічного графу підлеглості для побудови структур умовних операторів та операторів циклів повинно включати єдиний кореневий вузол для представлення кожного оператора. Зворотні семантичні зв'язки в графах операторів структурного програмування умовно опускаються на період синтаксичної та пошукової обробки, а послідовності операторів організуються так, щоб представлення першого з них знаходилося в найвіддаленішій лівій гілці графа.

Основу ідентифікації будь-якої інформаційної сутності складає заголовок її опису. В загальному випадку інформаційну сутність складають будь-які об'єкти даних, специфікацій [5] та управління обчисленнями. Ациклічний граф, наведений на рис.1, можна узагальнити для відображення типів та примірників даних, виділивши в них ідентифікаційний заголовок та конкретизацію сутності. На рис. 2 показано приклад опису класу мови C++ з метапозначеннями проміжних конструкцій та ідентифікАцією РРКЗ через вузол *ім'я*.

Організація описів та зв'язків даних у вигляді ациклічних графів дає перевагу у вигляді максимальної універсальності опису структур даних. Разом з тим вона породжує ряд складнощів у забезпеченні цілісності динамічних даних. Вони вирішуються шляхом створення графів спеціальних списків власників об'єктів та їх відображень.

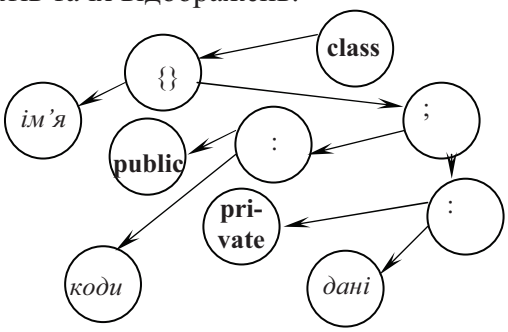



***Рис. 2. Приклад дерева визначення класу об'єктів мови C++*** `class` *ім'я*{**public**:*коди*;**private**:*дані*;}

Подібні структури надають можливість формувати комплексні (складені або складні) групи об'єктів. При цьому як нетермінальні вузли графів використовуються дужки ("{}") та роздільники (":") групування, а також роздільники переліку ("," і ";"). Для конкретизації об'єктів використовуються спеціальні графи описів, що посилаються на попередньо визначений тип користувача. Приклад такої структури показаний на рис. 3, має вигляд ациклічного графа, в якому термінальні вузли вміщують ідентифікацію та характеристики примірників об'єктів.

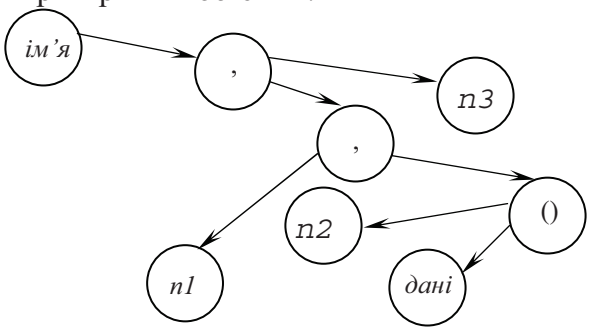



***Рис. 3. Приклад дерева визначення примірників об'єктів в мові C++*** *ім'я n1,n2*(*дані*),*n3*;

За аналогією з описом класів та примірників об'єктів можна будувати і такі ж складні графи описів типів даних користувача з переліком полів та примірників даних. Для того щоб довести суперечність або надлишковість описаних даних, подібні представлення слід використати для специфікацій [5] зв'язків типів і примірників об'єктів.

Графові представлення специфікацій на додаток до елементів групування і переліку

[3, 5] включають теоретико-множинні і звичайні математичні унарні та бінарні операції для визначення доменів, структур і зв'язків. Це дозволяє також зберігати відношення або обчислювальні формули, виклики функцій та коди для комп'ютерних обчислень.

Послідовності перетворень [3], що використовуються для доведень і формування розрахункових формул, включають ланцюжки операцій еквівалентних та прямих або зворотних імплікативних перетворень. Такі доведення та обґрунтування доцільно зберігати в базах знань, що управляють роботою зі створення, використання і супроводження бібліотек РРКЗ для заданої проблемної галузі. Вони зберігають базові інформаційні сутності для наступних доведень і формальної верифікації розроблюваних РРКЗ.

Пошук елементів бібліотек РРКЗ з синтаксичними та синтаксичними ознаками і зберігати послідовності доведення коректності. Заголовки елементів бібліотек повинні включати ідентифікацію функцій та версій елементів, покажчик на записи специфікації і про доведення відповідності специфікації, а також про оцінки критеріїв продуктивності та необхідних витрат для експлуатації РРКЗ.

Таблиці ідентифікації інформаційних сутностей доцільно побудувати як індекси з використанням покажчиків на інформаційні поля графів. Таким чином, загальне використання графово-табличного представлення обґрунтовується універсальністю логічного представлення, яку вони забезпечують.

**Набір структур даних для представлення сутностей**, який використовується в узагальненому форматі сховища, являє собою поєднання послідовного представлення даних на нижньому рівні та представлення структурованих груп даних у вигляді ациклічних графів на верхньому. Ациклічні графи, приклади яких показані на попередніх прикладах, обираються як інструмент структуризації даних, тому що вони надають максимальну гнучкість та універсальність у поданні інформаційних та управляючих сутностей. Поля вузлів графа дозволяють мінімізувати зібрану інформацію, яка може бути опущена при відсутності необхідності її використання.

Для обґрунтування такого вибору використані наступні міркування. Найбільш популярним варіантом структур даних на сьогоднішній день являються реляційні табличні бази даних. Вони є найбільш зручними та ефективними для описання порівняно простих та одноманітних об'єктів. Але об'єкти, що використовуються у програмних пакетах для моделювання та розробки не являються простими та одноманітними. це призводить до виникнення значної кількості таблиць для всеохоплюючого подання інформації, як наслідок, зниження ефективності їх використання.

Натомість при використанні ациклічних графів введення нових об'єктів, зміна структури інформаційних сутностей вже існуючих об'єктів та створення додаткових відображень (найбільш поширена операція в системах проектування) не потребують значних додаткових витрат ресурсів. В той самий час нерідко виникає ситуація, коли при зміні структур даних інформаційних сутностей доводиться заново проектувати табличну базу даних.

Структура даних, що пропонується для використання в удосконаленому сховищі має дуальний вигляд і складається з 2 частин: декларативної та інформаційної, що поєднуються спільним кореневим вузлом графа.

Декларативна частина структури даних представлена окремою структурою даних, в якій зібрані описи для відображень, що використовуються в програмному пакеті. В ній міститься інформація про тип і зміщення кожного поля даних у відображенні відносно адреси початку області пам'яті що займає відображення. Це необхідно для надання можливості оперувати окремими полями даних та збереження інформації про наявні види відображень. Декларативна частина створюється як універсальна для всіх об'єктів РРКЗ, представлених у системі.

Також окремо визначається група узагальнених описів, призначених для створення визначень, об'єктів та відображень, їх редагування та видалення. Її основна задача – дозволяти редагувати об'єкти та структуру інформаційних сутностей засобами мови. До неї входять наступні команди визначення даних і графів описів. За аналогією з мовою SQL це – Create Instance <Ім'я об'єкта> is <Ім'я визначення>: <Визначення об'єкта з присвоєнням попередньо заданих значень>; Create Reflection <Ім'я відображення> : <Визначення відображення з присвоєнням попе-

редньо визначених значень> – команди для створення відображення.

**Набір процедур обробників для перетворення сутностей** мають містити арифметичні та логічні оператори, оператори зчитування та модифікації полів відображень, деякі стандартні оператори виводення та оператори для управління відображеннями та об'єктами.

Кожне відображення містить дані для певної проблемної галузі і певного рівня деталізації у вигляді таблиці. При цьому в ній не міститься даних, що мають обов'язкові відношення до інших галузей або рівнів деталізації. Відношення між даними реалізуються зв'язками відображень або спільними відображеннями. Таке представлення даних відповідає асоціативності та проблемній орієнтованості сприйняття і проектування структур даних і забезпечує оптимальне універсальне представлення даних будь-якої структури.

Крім того, запропоноване узагальнене представлення має надавати можливість описати будь-яку кількість відображень інформаційної сутності об'єкта. Відображення доцільно орієнтувати на задану проблемну галузь. Відображення в свою чергу можуть включати в себе відображення наступного рівня, орієнтовані на роботу з даними, що відносяться до певного більш вузького напряму заданої проблемної області. Вони також зберігаються як ациклічний граф, що забезпечує можливість створювати сутності даних з інформацією, відповідальною за зв'язок даних з різних проблемних областей.

**Набір операцій для роботи з даними графів.** Всі операції, необхідні для забезпечення вищевказаних функцій, можна розділити на декілька функціональних груп. Відповідно до них сховище деталізується як база для реалізації інтерпретатора запропонованих розширень мови SQL і швидкісних перетворень настроювання.

Перший тип операцій – операції доступу до даних та обробників. Вміщує в себе операції для отримання інформації з відображення предмету та запуску обробників. Також в цю групу доцільно включити операції встановлення фільтрів для відображень. На додаток до команд мови маніпуляції даними SQL група додаткових команд вибірки включає команди: встановлення фільтру для відображення (SetFilter); встановлення фільтру на певне відображення певного предмета (SetIndividualFilter); одержання інформації про об'єкт або його частину (Get (Sub)Object); пошук схожого підграфа за заданою мірою схожості (FindSimilar); одержання інформації про відображення відповідно встановленого фільтра (GetDefinition); старт обробника відображень (RunReflector).

Команди корекції, що додаються до мови SQL, включають додавання додаткового підграфа до заданого вузла (AddSubTree); вилучення підграфа, починаючи з заданого вузла (AddSubTree); перетворення графа для розрахунку заданого цільового вузла (ConvToGoal).

Відповідно до функцій сховища даних, друга група команд має дозволяти працювати з обробки правил операції занесення корекцій графів. Наразі для роботи з постійними даними використовуються функції файлової системи. Тому для збереження та перенесення даних на постійних носіях потрібно забезпечити можливість створювати та видаляти папки та зберігати сутності та групи сутностей в певних файлах, а також відновлювати їх в процесі завантаження файлу.

## Висновки

Створення сховищ для середовищ проектування РРКЗ, орієнтованих на збереження даних забезпечує гнучко та універсально інфраструктуру для необмежених структур даних. Вона орієнтована на роботу з інформаційними сутностями, що мають різноманітну структуру даних. Її методи та оператори дозволяють виконувати дії з даними як з допомогою зовнішніх скрипкових сценаріїв, так і на стороні сервера.

Архітектура сховища даних, описана в статті, дозволяє зв'язати робочий простір сховища з іншими робочими просторами для інших програм. Дана взаємодія на файловому рівні доволі сильно спрощує деякі ключові моменти роботи середовищ розробки.

Додатковою можливістю взаємодії є варіант реалізації сховища як надбудови на існуючі варіанти реляційних СУБД. Це дозволяє змінювати дані в інтерактивному режимі. Крім того, за допомогою трансакцій в

базах даних можлива взаємодія на рівні безконфліктного зчитування даних для будь-якої кількості програм.

Ще одним варіантом перетворень, при якому атомарною є взаємодія з записами бази знань, є обробка специфікацій програм та сумісна обробка специфікацій та програмних кодів. Вони дозволяють досить зручно відображати структури даних, які використовуються у сховищі. Проте робота з ними потребує гігантських обсягів обчислень, і тому допускається лише при невеликому робочому завантаженні.

Виходячи з цього найбільш ефективною є реалізація сховища у складі окремого програмного пакету з можливостями управління паралельною обробкою даних. Описаний в статті варіант графової архітектури та методів перетворень сховища є доволі гнучким для забезпечення обробки на рівні перетворення даних для різних форматів, а прикладний програмний інтерфейс забезпечує можливість безпосереднього звертання програм до сховища даних. Такий метод взаємодії з базою знань про бібліотеки РРКЗ є найбільш безпечним і ефективним при настроюванні програм з усіх методів збереження інформаційних сутностей, що використовуються у вбудованих базах даних і знань.

## Список посилань

1. Ноутон П., Шилдт Г. Java™ 2: Пер. с англ. – СПб.: БХВ-Петербург, 2007. – 1072 с.
2. Пустоваров В.І. Побудова баз знань для сумісного створення програм і обладнання та їх формальної верифікації.: Вісник Національного технічного університету “Київський політехнічний інститут”. Інформатика, управління та обчислювальна техніка. К.: «Век+». – 2008.– 49.– с. 41 – 46.
3. Буханцов С.Н., Пустоваров В.І., Стіренко С.Г., Желудкова Т.В. Доказове управління контролю генерацією та модифікацією програмних модулів для інформаційно-аналітичних систем.: Вісник Національного технічного університету “Київський політехнічний інститут”. Інформатика, управління та обчислювальна техніка. К.: «Век+». – 2008.– 48.– с. 49 – 53.
4. Inmon W. H. Building the Data Warehouse. Wiley Publishing, Inc. 2005 – 543p.
5. Woodcock J., Davies J. Using Z. Specification, Refinement, and Proof. C.A.R. Hoare Series editor, 1995 – 390 p.

Поступила в редакцию 8.12.2009

УДК 004.052.42

*МАРКОВСКИЙ А.П.,*
*САИДРЕЗА МАХМАЛИ,*
*ТУРЧЕНКО Ю.А.,*
*САКУН В.Н.*

# ИСПОЛЬЗОВАНИЕ ВЗВЕШЕННЫХ КОНТРОЛЬНЫХ СУММ ДЛЯ ИСПРАВЛЕНИЯ “ПАЧЕК” ОШИБОК ПЕРЕДАЧИ ДАННЫХ

Статья посвящена решению проблемы повышения эффективности обнаружения и исправления “пачек” ошибок передачи данных, вызванных внешними помехами. Для исправления “пачек” ошибок предложено использовать специальную модификацию взвешенной контрольной суммы. Разработан алгоритм обнаружения и коррекции “пачек” ошибок. Показано, что предложенная технология обеспечивает большую эффективность обнаружения и исправления ошибок по сравнению с известными корректирующими кодами.

Paper is dedicated to solving the problem of increasing the efficiency of detecting and correcting the data transmission burst errors brought on by external noises. For burst errors coorrection the utilization of special weighed check sum modification has been proposed. Algorithm for burst error detection and correction has been worked out. It has been shown that proposed techniques ensure superior data transformation error detecting and correcting effectiveness in compare to known correcting code.

## Введение

К началу XXI-го века информационная интеграция стала одним из наиболее весомых факторов развития человеческой цивилизации. В свою очередь, процесс расширения информационной интеграции базируется на совершенствовании средств и технологии передачи цифровых данных. Ныне основными направлениями такого совершенствования являются повышение скорости передачи и расширение использования беспроводных каналов.

Важнейшим фактором совершенствования технологии передачи данных является обеспечение высокого уровня надежности. Это достигается как за счет повышения собственно передачи в канале, так и за счет совершенствования специальных средств обнаружения и коррекции ошибок, возникающих при передаче данных. Поступательное развитие техники передачи данных постоянно ставит новые требования к средствам обнаружения и коррекции ошибок передачи. Так, повышение скорости передачи данных ужесточает требования к быстродействию средств контроля ошибок. Расширение использования беспроводных систем передачи данных имеет следствием рост числа ошибок, вызванных внешними помехами. Характерно, что внешние помехи, как правило, вызывают искажение группы смежных бит передаваемого сообщения. Такие групповые искажения бит сообщения получили название “пачек” ошибок [1].

Вследствие возникновения “пачек” ошибок кратность искаженных битов сообщения значительно превышает число битовых ошибок, вызванных гауссовым шумом или межсимвольной интерференцией. Для коррекции битовых искажений большой кратностей необходимы новые методы и средства контроля ошибок.

Таким образом, проблема повышения эффективности обнаружения и коррекции ошибок, вызванных внешними помехами, в современных условиях является актуальной и имеющей важное практическое значение.

## Анализ известных способов коррекции “пачек” ошибок

Доминирующими причинами возникновения ошибок в современных каналах передачи цифровых данных являются межсигнальная интерференция, гауссов шум и внешние помехи [1]. Характер ошибок, обусловленных первыми двумя причинами существенно отличается от тех, которые вызваны внешним шумом. Так, явления межсигнальной интерференции и гауссова шума вызывают независимые искажения отдельных битов информационного, причем зависимость вероятности появления ошибок от их кратности носит биномиальный характер, так, что для описания таких ошибок применима теоретическая модель двоичного симметричного канала [2].

Для описания ошибок, вызванных внешними помехами используется другая модель [2]. Внешние помехи, как правило, по дли-

тельности соответствуют времени передачи группы из $h$ битов. Соответственно, битовые искажения, вызванные внешней помехой ограничены указанной группой битов. Это означает, что искажению могут подвергнуться каждый из $h$ смежных битов сообщения. В дальнейшем, будем считать, что число смежных бит, передаваемых во время действия помехи и, соответственно, подверженных риску искажения не превышает $m$, то есть $h \leq m$. Таким образом, в рамках рассмотренной модели возникновения ошибок, однократная внешняя помеха может вызвать искажение любого числа из $m$ смежных битов сообщения. При этом группа из $m$ смежных битов сообщения, подверженных искажению под действием внешней помехи получила название “пачки” ошибок [1].

Для обеспечения надежности передачи данных применяется широкий арсенал средств. Однако, развитие технологий передачи цифровой информации имеет следствием постоянное смещение акцентов в оценке эффективности средств обнаружения и коррекции ошибок. Основными критериями средств обеспечения надежности передачи данных являются:

- Надежность контроля ошибок, которая характеризуется классом исправляемых и выявляемых ошибок.
- Надежность классификации класса корректируемых ошибок.
- Временные характеристики: возможность вычисления контрольного кода в темпе передачи данных, время $t_1$ выявления ошибки, время $t_2$ коррекции ошибок.
- Число передаваемых контрольных разрядов.

С ростом скорости передачи значимость временных характеристик возрастает, а важность количества передаваемых контрольных разрядов уменьшается.

К настоящему времени для обеспечения надежности передачи цифровых данных применяются две базовые технологии:

- контроль и исправление выявленных ошибок с использованием корректирующих кодов
- выявление ошибок с использованием специальных кодов и исправление ошибок путем повторной передачи информационного блока.

Выбор базовой технологии определяется характеристиками канала передачи данных и особенностями использования. Так, в проводных и кабельных линиях передачи данных интенсивность возникновения ошибок достаточно мала и, поэтому, здесь чаще используется технология обнаружения ошибок с повторной передачей. Напротив, в подверженных внешним помехам беспроводных каналах, в которых интенсивность возникновения ошибок относительно велика, чаще применяют корректирующие коды [3].

При использование корректирующих кодов их эффективность существенно падает с ростом кратности исправляемых ошибок, поскольку и число необходимых для коррекции контрольных разрядов и вычислительная сложность операций контроля экспоненциально зависят от кратности корректируемых разрядов [3]. При этом корректирующие коды не учитывают особенностей ошибок, возникающих при воздействии внешних помех.

Значительная вычислительная сложность получения синдрома ошибок при их большой кратности создает существенные трудности при реализации контроля в темпе передачи данных для скоростных каналов передачи цифровой информации. Важным недостатком корректирующих кодов является то, что объем вычислений для получения синдрома ошибок постоянен и не зависит от того, произошли они или нет [3]. Вместе с тем, для кабельных линий передачи данных интенсивность возникновения “пачек” ошибок относительно мала и выполнение большого объема вычислений является избыточным.

Существенно меньшее количество контрольных разрядов используется для обнаружения “пачек” ошибок. Наиболее распространенным средством контроля таких ошибок являются циклические избыточные коды (CRC). При использовании CRC “пачка” ошибок гарантированно обнаруживается, если степень $n$ образующего полинома CRC больше или равна длине “пачки” [3]. При этом CRC не гарантирует обнаружения двух и более “пачек” ошибок. Для многих приложений повторная передача блока является неприемлемой в силу того, что это требует значительного времени и сопряжено с прерыванием потока данных. Такая проблема возникает при передаче данных с использованием кабельных линий в системах реального времени [3].

Таким образом, для достаточно широкого круга приложений, связанных, в первую очередь, с обработкой информации в реальном времени, известные методы не обеспечивают эффективного решения задачи коррекции “пачек” ошибок передачи данных.

Целью исследований является разработка новых, эффективных средств коррекции вызванных внешними помехами “пачек” ошибок в

высокоскоростных проводных и кабельных каналах.

### Коррекция “пачек” ошибок с использованием взвешенных контрольных сумм

Известно, что наиболее быстрым способом контроля ошибок являются контрольные суммы. Существует ряд их модификаций, наиболее эффективными из которых являются взвешенные контрольные суммы [4]. Важным их достоинством является возможность адаптации к особенностям ошибок доминирующего типа.

Исходя из этого, поставленная цель может быть достигнута путем создания модификации взвешенной контрольной суммы, адаптированной для обнаружения и исправления “пачек” ошибок, вызванных одной или двумя внешними помехами.

Контролируемый блок $B=\{b_1,b_2,\ldots,b_n\}$, $\forall l\in\{1,\ldots,n\}$: $b_l\in\{0,1\}$, логически разделяется на $m$-битовые символы: $B=\{X_1,X_2,\ldots,X_q\}$, $q=n/m$, $\forall j\in\{1,\ldots,q\}$: $X_j=\{b_{(j-1)\cdot m+1}, b_{(j-1)\cdot m+2},\ldots,b_{j\cdot m}\}$. Тогда блок $B$ можно рассматривать как матрицу, строки которой соответствуют символам, а столбцы – одноименным битам символов.

Контрольный код $C$ информационного блока $B$ предлагается формировать в виде $C=\{S_0, S_1,S_2,\ldots,S_m, P\}$, где $P=\{p_1,p_2,\ldots,p_m\}$ $m$-битовый вектор, каждый $i$-тый ( $i\in\{1,\ldots,m\}$ ) бит которого равен сумме по модулю 2 $i$-тых битов всех символов:

$$\forall i\in\{1,\ldots,m\}: p_i = x_{1i}\oplus x_{2i}\oplus\cdots\oplus x_{qi} = \\ = b_i\oplus b_{m+i}\oplus\cdots\oplus b_{n-q+i}$$

$S_0$ – $k$-битовый ( $k=\log_2 q$ ) код суммы по модулю 2 произведений битов четности символов информационного блока на весовой коэффициент $W_j$, зависящий от порядкового номера символа в блоке:

$$S_0=\bigoplus_{j=1}^{q} W_j\cdot(x_{j1}\oplus x_{j2}\oplus\cdots\oplus x_{jm}) \qquad (1)$$

В простейшем случае, весовой коэффициент $W_j$ $j$-го символа совпадает с $k$-битовым номером этого символа : $W_j=j$. При формировании $m$-разрядных компонент контрольного кода: $S_1,S_2,\ldots,S_m$ используются наборы весовых коэффициентов $U_1,U_2,\ldots,U_m$. Каждый $i$-тый набор $U_i$ включает в себя $q$ $(k+1)$-разрядных весовых коэффициентов: $U_i=\{u_{i1},u_{i2},\ldots,u_{iq}\}$, обладающих определенными свойствами. Каждая $i$-тая компонента $S_i$ контрольного кода $C$ представляет собой $(k+1)$-разрядный код суммы по модулю 2 произведений $i$-тых битов всех символов блока на соответствующий весовой коэффициент $i$–то набора $U_i$

$$\forall i\in\{1,\ldots,m\}: S_i=\bigoplus_{j=1}^{q} u_{ij}\cdot x_{ji} \qquad (2)$$

Коэффициенты наборов $U_1,U_2,\ldots,U_m$ подбираются таким образом, чтобы выполнялись следующие условия:

1. В рамках одного набора все весовые коэффициенты должны быть отличными друг от друга: $\forall i\in\{1,\ldots,m\}$, $\forall j,e\in\{1,\ldots,q\}$, $j\neq e$: $u_{ij}\neq u_{ie}$.

2. Для каждой пары $U_i$ и $U_v$ наборов, $\forall i,v\in\{1,\ldots,m\}$, сумма по модулю 2 двух любых пар весовых коэффициентов должна быть отличной друг от друга: $\forall j,e,z,y\in\{1,\ldots,q\}$, $j\neq e$, $j\neq z$, $j\neq y$, $e\neq z$, $e\neq y$, $z\neq y$: $(u_{ij}\oplus u_{ie})+2^{k+1}\cdot(u_{vj}\oplus u_{ve})\neq(u_{iz}\oplus u_{iy})+2^{k+1}\cdot(u_{vz}\oplus u_{vy})$. Другими словами, недопустимой является совпадение сумм по модулю 2 двух пар весовых коэффициентов для двух наборов.

В таблице 1, в качестве примера, приведены значения наборов $U_1,U_2,U_3,U_4$ весовых коэффициентов, удовлетворяющие приведенным условиям для $m=4$ и $q=10$.

Из свойства 2 следует, что полученные наборы $U_1,U_2,\ldots,U_q$ весовых коэффициентов обеспечивают однозначную локализацию двух символов, в которых искажена одноименная пара бит. В частности, если для приведенных в таблице 1 наборов весовых коэффициентов рассматривать искажения 2-го и 3-го битов в паре 4-разрядных символов, то для любой пары $r,s\in\{1,..,10\}$ символов пара значений сумм соответствующих коэффициентов $<u_{2,r}\oplus u_{2,s}, u_{3,r}\oplus u_{3,s}>$ позволяют однозначно идентифицировать одну из 45 возможных пар $<r, s>$. Этот факт иллюстрируется таблицей 2.

***Табл. 1. Наборы весовых коэффициентов***

| Номер символа | Наборы весовых коэффициентов | | | |
|---|---|---|---|---|
| | $U_1$ | $U_2$ | $U_3$ | $U_4$ |
| 1 | 2 | 2 | 2 | 2 |
| 2 | 3 | 1 | 1 | 1 |
| 3 | 4 | 4 | 4 | 4 |
| 4 | 5 | 3 | 5 | 6 |
| 5 | 6 | 7 | 3 | 8 |
| 6 | 7 | 8 | 8 | 3 |
| 7 | 8 | 5 | 6 | 9 |
| 8 | 9 | 9 | 11 | 5 |
| 9 | 10 | 10 | 15 | 10 |
| 10 | 11 | 14 | 13 | 15 |

Общий объем $V_T$ памяти таблиц соответствия пары сумм весовых коэффициентов

паре номеров коэффициентов определяется в виде:

$$V_T = \frac{n \cdot (n - m - q + 1)}{2} \cdot \log_2 q \qquad (3)$$

Предлагаемый способ формирования контрольного кода иллюстрируется следующим примером. Пусть передается блок данных, состоящий из 40 бит. Пусть длительность воздействия помехи не превышает времени передачи 4-х символов. Тогда, разрядность $m$ символа принимается равной 4-м. Соответственно блок состоит из 10-ти ($q$=10) 4-х битовых символов: $X_1$={1001}, $X_2$={0011}, $X_3$={1000}, $X_4$={1111}, $X_5$={1011}, $X_6$={0001}, $X_7$={1100}, $X_8$={0110}, $X_9$={0101}, $X_{10}$={0010}. Если при формировании контрольного кода используются наборы весовых коэффициентов $U_1$,$U_2$,$U_3$ и $U_4$, приведенные в таблице 1, то компоненты контрольного кода вычисляются в следующем виде: $S_0$ = {1010}, $S_1$={1101}, $S_2$={0101}, $S_3$={0001}, $S_4$={0100}, $P$={0101}.

***Табл. 2. Соответствие сумм коэффициентов $U_2$ и $U_3$ номерам пары символов, в которых искажены 2-й и 3-й биты.***

| $u_{2,r} \oplus u_{2,s}, u_{3,r} \oplus u_{3,s}$ | **r, s** | $u_{2,r} \oplus u_{2,s}, u_{3,r} \oplus u_{3,s}$ | **r, s** |
|---|---|---|---|
| 1 - 2 | 3,7 | 9 – 9 | 2,6 |
| 1 - 3 | 6,8 | 9 – 10 | 4,9 |
| 1 - 7 | 1,4 | 9 – 14 | 5,10 |
| 2 - 4 | 2, 4 | 10 – 9 | 3,10 |
| 2 - 5 | 5,7 | 10 – 10 | 1,6 |
| 2 - 7 | 6,9 | 10 – 14 | 4,8 |
| 3 - 3 | 1,2 | 11 – 9 | 1,8 |
| 3 - 4 | 8,9 | 11 – 11 | 7,10 |
| 3 - 7 | 3,5 | 11 – 13 | 4,6 |
| 4 - 2 | 9,10 | 11 – 14 | 2,9 |
| 4 - 6 | 4,5 | 12 – 12 | 3,6 |
| 4 - 7 | 2,7 | 12 – 13 | 7,8 |
| 5 - 1 | 1,5 | 12 – 15 | 1,10 |
| 5 - 5 | 2,3 | 13 – 8 | 4,10 |
| 6 - 2 | 2,5 | 13 – 12 | 5,9 |
| 6 - 3 | 4,7 | 13 – 14 | 6,7 |
| 6 - 5 | 6,10 | 13 – 15 | 3,8 |
| 6 - 6 | 1,3 | 14 – 8 | 5,8 |
| 7 - 1 | 3,4 | 14 – 11 | 3,9 |
| 7 - 4 | 1,7 | 15 – 9 | 7,9 |
| 7 - 6 | 8,10 | 15 – 11 | 5,6 |
| 8 - 10 | 2,8 | 15 – 12 | 2,10 |
| 8 - 13 | 1,9 | | |

## Исправление одной “пачки” ошибок

Если под воздействием внешней помехи возникла “пачка” ошибок, то это означает, что в рамках компактной группы из $m$ битов подверглись искажению от одного до $m$ бит, принадлежащих одному или двум смежным символам. В этой ситуации принципиально исключено искажение двух битов с одинаковыми номерами. Соответственно, ситуация классифицируется путем анализа полей $\Delta S_1,\ldots,\Delta S_m$ и $\Delta P$ разности контрольных кодов передатчика и приемника: если одноименные коды $\Delta S$ и компоненты $\Delta P$ оба не равны нулю или оба равны нулю, то считается, что при передаче блока произошла одиночная “пачка” ошибок. Формально: если $\forall g \in \{1,\ldots,m\}$ справедливо одно из двух условий: $\Delta S_g \neq 0$ и $\Delta p_g \neq 0$ или $\Delta S_g = 0$ и $\Delta p_g = 0$, то “пачка” ошибок классифицируется как одиночная, искажающая один или пару смежных символов блока.

Если “пачка” ошибок локализована в одном символе, например $X_r$, где $r \in \{1,\ldots,q\}$, то на приемнике будут искажены биты, номера которых принадлежат множеству $\vartheta \subseteq \{1,2,\ldots,m\}$. Соответственно, биты в коде $\Delta$ разности контрольных кодов приемника и передатчика биты $\Delta p$ поля $P$ равны единице, если их номера принадлежат $\vartheta$. Аналогично, коды $\Delta S_1$, $\Delta S_2$, …, $\Delta S_m$ принимают значение нуля, если их номер не принадлежит $\vartheta$ и равны $r$-тому весовому коэффициенту соответствующего набора, если их номера принадлежат $\vartheta$ : $\forall g \in \vartheta$: $\Delta S_g = u_{gr}$. Код $\Delta S_0$ может быть равным нулю или единице, в зависимости от того четное или нечетное количество элементов содержит множество $\vartheta$.

Описанный тип ошибки достаточно просто исправляется: номер $r$ искаженного символа определяет любой ненулевой код $\Delta S_1$, $\Delta S_2$, …, $\Delta S_m$, а искаженные в этом символе биты выделяют единичные компоненты $\Delta P$.

Если “пачка” ошибок локализована в паре смежных символов, например $X_r$ и $X_{r+1}$ где $r \in \{1,\ldots,q\}$, то на приемнике, в символе $X_r$ будут искажены биты, номера которых принадлежат множеству $\Theta \subseteq \{d,\ldots,m\}$, а в символе $X_{r+1}$ будут искажены биты, номера которых принадлежат множеству $\Omega \subseteq \{1,\ldots,d-1\}$, так, что общее количество искаженных битов в обоих символах не превышает $m$. Локализация искаженных символов осуществляется следующим образом: ненулевые компоненты $\Delta S_d$, …, $\Delta S_m$ позволяют определить номер $r$ первого из искаженных символов, а ненулевые компоненты $\Delta S_1$, …, $\Delta S_{d-1}$ через соответствующие весовые коэффициенты определяют значение $r$+1. При этом значение $d$ определяется по кодам $\Delta S_1$,

$\Delta S_2$, …, $\Delta S_m$: их ненулевые значения образуют две компактные группы: первая образована весовыми коэффициентами ($r$+1)-го символа, а вторая – весовыми коэффициентами $r$-го символа.

Описанная процедура коррекции одной "пачки" ошибок может быть иллюстрирована следующим образом в рамках изложенного выше примера передачи блока из 40 бит.

Пусть под воздействием помехи исказились два смежных символа с номерами 8 и 9, так, что на приемнике получены следующие их значения: $X_8'$={011**1**}, $X_9'$={**10**01}. Контрольный код на приемнике вычисляется в следующем виде: $S_0'$ = {0010}, $S_1'$={0111}, $S_2'$={1111}, $S_3'$={0001}, $S_4'$={0001}, $P'$={1110}. Соответственно, компоненты разности контрольных кодов приемника и передатчика равны: $\Delta S_0$ = {1000}, $\Delta S_1$={1010}, $\Delta S_2$={1010}, $\Delta S_3$={0000}, $\Delta S_4$={0101}, $\Delta P$={1101}. Поскольку $\Delta p_1$=1, то первый бит искажен только в одном символе. Так как $\Delta S_1$= 10, а в наборе $U_1$ имеется только один код, равный 10: $u_{19}$=10, то первый бит искажен в 9-м символе. Аналогично, поскольку $\Delta p_2$=1, то второй бит искажен в символе, весовой коэффициент которого в наборе $U_2$ равен $\Delta S_2$= 10. Поскольку только $u_{29}$=10, то второй бит искажен в 9-м символе. Так, как $\Delta p_3$=0 и $\Delta S_3$ = 0, то 3-й бит не искажен ни в одном из передаваемых символов. Так как $\Delta p_4$=1, то номер символа, в котором искажен 4-й бит находится исходя из значения $\Delta S_4$=5. Среди $U_5$ только $u_{48}$=5=$\Delta S_4$. Следовательно, 4-й бит искажен в 8-м символе.

Таким образом, предложенная модификация взвешенной контрольной суммы позволяет корректировать любые битовые искажения, вызванные однократным воздействием внешней помехи.

### Исправление двух "пачек" ошибок

При воздействии двух помех, длительность каждой из которых не превышает времени передачи одного символа, в передаваемом блоке данных может исказиться два, три или четыре символа.

При искажении пары символов, например, $r$-го $X_r$ и $s$-го – $X_s$, $r,s\in\{1,\ldots,q\}$ их локализация осуществляется следующим образом. Единичные биты компоненты $\Delta P$ разности контрольных кодов приемника и передатчика выделяют биты, которые искажены только в одном из пары символов $X_r$ и $X_s$. Пусть, например, бит $\Delta p_i$=1, $i\in\{1,\ldots,m\}$. Это означает, что искажению подвергся $i$-тый бит $r$-того символа $X_r$. Численное значение номера $r$ искаженного символа определяется исходя из того, что компонента $\Delta S_i = u_{ir}$, то есть $r$-тому коду набора $U_i$ весовых коэффициентов. По значению $u_{ir}$ однозначно определяется номер $r$ искаженного символа. В случае, если существует $\Delta p_g$=1, $g\in\{1,\ldots,m\}$ такое, что $\Delta S_g = u_{gs}$, то по описанному выше способу определяется номер $s$ второго из искаженных символов – $X_g$. Если такого единичного бита $\Delta P$ не существует, то находится бит $e$, $e\in\{1,\ldots,m\}$ в котором искажены оба символа $X_r$ и $X_g$. Этот бит находится исходя из следующих условий: $\Delta p_e$=0, $\Delta S_e\neq 0$. Очевидно, что $\Delta S_e = u_{er} \oplus u_{eg}$. Код $u_{er}$ известен, поскольку известны коды $e$ и $r$. Код весового коэффициента $u_{eg} = \Delta S_e \oplus u_{er}$. По коду $u_{eg}$ находится номер $g$ второго из искаженных символов – $X_g$.

Наконец, если в обоих символах $X_r$ и $X_g$ искажены одни и те же биты, то все элементы вектора $\Delta P$ равны нулю. Пусть пара бит, например, с номерами $i$ и $e$ искажены в обоих символах $X_r$ и $X_g$. Тогда $\Delta S_i = u_{ir}\oplus u_{ig}$ и $\Delta S_e = u_{er}\oplus u_{eg}$. Согласно свойству 2 весовых коэффициентов пара значений суммы двух элементов наборов $U_i$ и $U_e$ однозначно определяет номера $r$ и $g$ суммируемых элементов этих наборов или, что то же самое, – номера искаженных символов блока. Ясно, что если в обоих символах $X_r$ и $X_g$ искажено более 2-х одноименных бит, то номера символов определяются аналогичным способом.. Если в каждом из символов $X_r$ и $X_g$ искажен один и тот же единственный бит, например с номером $i$, то:

$$\Delta S_0 = r \oplus g, \Delta S_i = u_{ir} \oplus u_{ig} \qquad (4)$$

Очевидно, что система линейных уравнений (4) позволяет однозначно определить номера $r$ и $g$ искаженных символов.

Номера искаженных битов в символах $X_r$ и $X_g$ определяются следующим образом. Бит с номером $i$ в символе $X_r$ считается искаженным, если выполняется одно из двух условий: $\Delta p_i$ =1 при $\Delta S_i = u_{ir}$ или $\Delta p_i$ =0 при $\Delta S_i \neq u_{ir}$.

Если две помехи вызвали искажение трех или четырех символов, то при условии, что длительность воздействия одной помехи не превышает времени передачи одного симво-

ла, искажению могут быть подвержены не более 2-х одноименных битов в символах.

Пусть, в результате воздействия первой помехи искажены символы $X_r$ и $X_{r+1}$, а второй – символы $X_g$ и $X_{g+1}$. При этом, бит с номером $i$ может быть искажен либо в одном из символов $X_r$ или $X_{r+1}$, и независимо – в одном из символов $X_g$ или $X_{g+1}$. Если $i$–тый бит искажен только в символе $X_r$, то $\Delta p_i$=1 и $\Delta S_i$=$u_{ir}$, только в символе $X_{r+1}$ то $\Delta p_i$=1 и $\Delta S_i$=$u_{i,r+1}$, только в символе $X_g$ то $\Delta p_i$=1 и $\Delta S_i$=$u_{ig}$, только в символе $X_{g+1}$ то $\Delta p_i$=1 $\Delta S_i$=$u_{i,g+1}$. Таким образом, если $i$-тый бит искажен только в одном символе, локализация последнего осуществляется достаточно просто.

Если $i$-тый бит искажен в паре символов, то возможно две ситуации. Первая состоит в том, что существует еще один бит который искажен в паре символов. Вторая ситуация возникает, если не существует другого символа, искаженного в паре символов.

В первой из означенных ситуаций локализация искаженных символов производится следующим образом. Пусть бит с номером $i$, $i\in\{1,\ldots,m\}$ искажен в символе $X_r$ (или $X_{r+1}$) и символе $X_g$ (или $X_{g+1}$). Существует также бит с номером $e$, $e\in\{1,\ldots,m\}$, который искажен в символе $X_r$ (или $X_{r+1}$) и символе $X_g$ (или $X_{g+1}$). В этом случае $\Delta p_i$=0, а компонента $\Delta S_i$ разности взвешенных контрольных сумм может принимать одно из 4-х значений: $\Delta S_i\in\{u_{ir}\oplus u_{ig}, u_{ir}\oplus u_{i,g+1}, u_{i,r+1}\oplus u_{i,g}, u_{i,r+1}\oplus u_{i,g+1}\}$. Аналогично, $\Delta p_e$=0 и $\Delta S_e\in\{u_{er}\oplus u_{eg}, u_{er}\oplus u_{e,g+1}, u_{e,r+1}\oplus u_{e,g}, u_{e,r+1}\oplus u_{e,g+1}\}$. В силу свойства 2 весовых коэффициентов пара значений $\Delta S_i$ и $\Delta S_e$ однозначно определяет компоненты наборов $U_i$ и $U_e$, а значит и номера символов, в которых искажены $i$-тый и $e$-тый биты.

Таким образом, предложенная модификация взвешенной контрольной суммы гарантирует исправление 2-х, произвольно локализованных в передаваемом блоке “пачек” ошибок, длиной не более $m$ бит.

## Оценка эффективности

Основным достоинством предложенной модификации взвешенных контрольных сумм по сравнению с корректирующими кодами является существенно большая производительность, позволяющая контролировать ошибки в темпе передачи данных. При использовании корректирующих кодов контроль ошибок производится путем вычисления синдрома. Вне зависимости от того, произошли ошибки или нет, это время постоянно и равно $t_s$. При этом код синдрома указывает на локализацию искаженных битов в блоке [3]. Таким образом, при использовании корректирующих кодов объем вычислений, обеспечивающий локализацию возможных ошибок производится всегда, вне зависимости от того, есть они или нет. В силу того, объем вычислений экспоненциально зависит от кратности исправляемых ошибок, время $t_s$ вычисления синдрома достаточно велико [3].

При использовании разработанной модификации взвешенной контрольной суммы время контроля (обнаружения) – $t_r$ ошибок и время их коррекции – $t_k$ существенно различаются: $t_k$>>$t_r$. Фактически, в отличие от корректирующих кодов, вычисление контрольного кода производится посимвольно непосредственно в темпе передачи. При этом несколько символов могут обрабатываться параллельно, так, что темп их обработки определяется задержкой вносимой выполнением логической операции XOR. Это определяет малую временную сложность контроля ошибок в предлагаемой разработке. Вместе с тем, временная сложность исправления ошибок в модификации взвешенной контрольной суммы превышает сложность выполнения аналогичной операции при применении корректирующих кодов: $t_k > t_s$. Однако, для кабельных линий передачи данных интенсивность возникновения ошибок достаточно мала, поэтому необходимость в коррекции ошибок возникает достаточно редко. На практике вероятность корректирования ошибок при передаче блока на 1-2 порядка меньше вероятности того, что блок передан без искажений [3]. С учетом этого, очевидно, что среднее значение временной сложности коррекции ошибок в предложенном варианте взвешенной контрольной суммы существенно меньше, чем при использовании корректирующих кодов. Результаты моделирования показали, что при интенсивности возникновения ошибок $10^{-2}$ на блок, использование предложенной модификации взвешенной контрольной суммы обеспечивает в 5-8 раз меньшую временную сложность обработки ошибок по сравнению с кодом Рида-Соломона.

По числу передаваемых контрольных разрядов предложенная модификация взвешенной контрольной суммы также эффективнее

корректирующих кодов. Это обусловлено тем, что, в отличие от корректирующих кодов, которые допускают произвольную локализацию искаженных битов, взвешенная контрольная сумма ориентирована на то, что искажения имеют вид “пачек”, то есть специализирована на частном виде ошибок передачи данных.

Количество $h$ разрядов контрольного кода в предложенной модификации взвешенных контрольных сумм определяется формулой:

$$h = m \cdot (\log_2 q + 3) = m \cdot (\log_2 \frac{n}{m} + 3) \qquad (5)$$

При использовании корректирующих кодов, способных исправить все возможные битовые искажения, вызванные 2-мя внешними помехами, или, другими словами, две $m$-битовые “пачки” ошибок число $H$ контрольных разрядов должно обеспечивать возможность локализации $2 \cdot m$ искаженных битов внутри $n$-разрядного кода. Поскольку число $G$ возможных размещений $2 \cdot m$ искаженных битов внутри $n$-разрядного кода определяется формулой:

$$G = \binom{2 \cdot m}{n} = \frac{n!}{(n - 2 \cdot m)! \cdot (2 \cdot m)!} \approx n^{2 \cdot m} \qquad (6)$$

то число $H$ контрольных разрядов равно:

$$H = \log_2 G \approx 2 \cdot m \cdot \log_2 n \qquad (7)$$

Очевидно, что $h<H$. Например, при передаче блока размером в 1 Кбайт ($n=2^{13}$) и если символ равен байту ($m=8$), число разрядов для коррекции 2-х “пачек” битовых ошибок при использовании корректирующих кодов составляет $H=208$, а взвешенной контрольной суммы – $h=106$.

Таким образом, для данного примера, число контрольных разрядов для решения одной и той же задачи контроля ошибок при использовании предложенной модификации взвешенной контрольной суммы в 1.96 раз меньше, чем в корректирующих кодах.

## Выводы

В результате проведенных исследований разработан способ коррекции одной или двух “пачек” битовых ошибок в последовательных линиях передачи данных. В основе предложенного способа лежит модификация взвешенной контрольной суммы.

Разработана организация вычисления контрольного кода, классификации и исправления одной и двух “пачек” битовых искажений. Сформулированы требования к выбору весовых коэффициентов для вычисления взвешенных контрольных сумм.

Теоретически и экспериментально показано, что по сравнению с корректирующими кодами реализация предложенного способа имеет меньшую временную сложность, что позволяет выполнять эффективный контроль ошибок в темпе передачи данных для высокоскоростных каналов передачи данных.

Доказано, что решение задачи коррекции двух “пачек” битовых ошибок достигается в предложенном способе с меньшим числом контрольных разрядов, чем в корректирующих кодах.

Предложенный способ коррекции пачек ошибок на основе взвешенных контрольных сумм может быть эффективно использован для обеспечения надежной передачи цифровых данных в скоростных каналах в условиях помех.

## Список литературы

1. Ирвин Дж., Харль Д. Передача данных в сетях: инженерный подход. СПб.: БХВ-Петербург, 2002.- 448 с.
2. Скляр Б. Цифровая связь. Теоретические основы и практическое применение. М.: Издательский дом ”Вильямс”.- 2004.- 1104 с.
3. Морелос-Сарагоса Р. Искусство помехоустойчивого кодирования. Методы, алгоритмы, применение. М.: Техносфера - 2005.- 319 с.
4. Самофалов К.Г., Марковский А.П., Мулки Яссин Ахмед Ал Бадайнех. Обнаружение и исправление ошибок передачи данных с использованием взвешенных контрольных сумм // Проблеми інформатизації та управління. Збірник наукових праць: Випуск 3(14).-К.,НАУ.- 2008.- С.121-128.

Поступила в редакцию 7.12.2009

УДК 681.301

*ЗАЙЦЕВ В.Г.,*
*ЛАН ЧУНЬЛИНЬ*

# СПОСОБЫ ПОВЫШЕНИЯ ЭФФЕКТИВНОСТИ КЛАССИФИКАЦИИ ДОКУМЕНТОВ ДЛЯ КОНЕЧНОГО МНОЖЕСТВА ЯЗЫКОВ

В работе рассматриваются две основные проблемы классификации документов для конечного множества языков: анализ текста в обработке естественного языка; нахождение наилучшего алгоритма машинного обучения. Описывается предлагаемый эффективный метод машинного обучения классификации текстов для конечного множества языков.

The paper analysis on two major problems for the classification documents in multilingual environment: analysis of the text in natural language processing; choose the efficient algorithm methods for machine learning. The efficient algorithm methods for machine learning are described.

## Введение

Задача разработки информационных систем, таких как интеллектуальные системы документооборота является одной из самых актуальных на сегодняшний день. Она рассматривается в контексте создания хранилищ данных и их систематизации с целью облегчения поиска необходимой информации. Современные информационные системы должны быть способны решать весь комплекс задач, связанных с управлением потоком входящих данных – автоматическую классификацию текстов для конечного множества языков. Действует целый ряд систем классификации больших объемов текстовой информации, в основе которых лежат технологии компьютерной лингвистики и алгоритмов распознавания образов. Обычные системы автоматической классификации текстов решают задачу автоматической классификации текстов на одном языке. Повышение эффективности классификации документов можно достичь изменением методов машинного обучения. Для конечного множества языков, это не только зависит от метода машинного обучения системы, а также зависит от обработки естественного языка (Natural Language Processing, NLP). Обработка естественного языка – общее направление искусственного интеллекта и математической лингвистики. Оно изучает проблемы компьютерного анализа и синтеза естественных языков. Применительно к искусственному интеллекту анализ означает понимание языка, а синтез – генерацию грамотного текста. Решение этих проблем будет означать создание более удобной формы взаимодействия компьютера и человека[1].

Таким образом, повышение эффективности автоматической классификации документов для конечного множества языков связано с нахождением наилучшего алгоритма машинного обучения и наилучшую обработку естественного языка с целью анализа исходных данных.

## Анализ текста – одна из самых главных задач обработка естественного языка

Распознавание слов и сегментация фраз на слова являются важными шагами во многих приложениях обработки естественного языка. В западных и европейских языках слова в предложениях разделяются пробелами. Поэтому слова довольно просто определяются как человеком так и компьютером. В восточных языках, таких как китайский, корейский и японский языки, проблемы распознавания слов и сегментации становятся более сложными. Слово может являться одним слогом либо комбинацией слогов, расположенных вместе в предложении. Проблемы распознавания китайских слов и сегментации китайских предложений на слова не могут быть полностью решены из-за следующих причин: не существует алгоритма, который сегментирует китайскиое предложение на слова точно в соответствии с его смыслом, если предложение считается изолированным. Отсутствие алгоритма сегментации на слова, который работал бы в предложении, объясняется тем, что каждое слово может быть частью разных слов. Его смысл не может быть определен без контекста.

В контексте автоматической обработки текстовой информации наиболее очевидной является проблема делимитации слова в китайском языке. Технически сложность определения границ слова заключается в том, что в китайской системе письменности не принято отделять слова пробелами. Например, в упомянутых европейских языках проблема определения слова ограничивается нахождением его физических границ, и, как правило, все системы автоматического анализа письменного текста в качестве базовой единицы принимают именно графическое слово – последовательность знаков алфавита, отделённых друг от друга пробелами или знаками препинания. Проблема делимитации слова в китайском языке осложняется также и тем, что слова в китайском языке почти лишены формальных признаков морфологического уровня. В словообразовании доминирует корнесложение, флективные формы встречаются нерегулярно, словоизменение почти отсутствует, что в совокупности делает невозможным выделение хотя бы основных лексических единиц путём определения их границ по каким-либо морфологическим маркерам. Многие исследователи пытаются решить проблему так называемой сегментации китайского текста (разбиение текста на слова). Для этого используются три основных способа: словарный (обычно с применением алгоритма максимального соответствия), статистический и комбинированный, сочетающий в себе оба предыдущих. По сообщениям авторов им удаётся правильно сегментировать текст на 95-99 процентов, это проблема почти решена[2].

В западных языках и европейских языках существует проблема стемминга. Стемминг (stemming) – это процесс нахождения основы слова для заданного исходного слова. Стеммер Портера (Stemmer Porter) – алгоритм стемминга. Впоследствии Мартин создал проект «Snowball» и, используя основую идею алгоритма, написал стеммеры для распространённых индоевропейских языков, в том числе для русского. Алгоритм не использует баз основ слов, а лишь, применяя последовательно ряд правил, отсекает окончания и суффиксы, основываясь на особенностях языка[3].

Результаты исследований показали, что существует хорошее программное обеспечение для обработки естественного языка: General Architecture for Text Engineering (GATE); LingPipe; Natural Language Toolkit (NLTK). Для решения проблемы делимитации слов китайского языка рекомендуют использовать ICTCLAS (Institute of Computing Technology, Chinese Lexical Analysis System)[4]. Для европейских языков для решения проблемы стемминга рекомендуют использовать Snowball[5].

### Классификация документов

Классификация документов – одна из задач информатики, заключающаяся в отнесении документа к одной из нескольких категорий на основании содержания документа. Используются методы информационного поиска и машинного обучения[6].

### Постановка задачи классификации

Определим задачу формально. Пусть задано некоторое конечное множество категорий $C = \{c_1, ..., c_{|c|}\}$, конечное множество документов $D = \{d_1, ..., d_{|D|}\}$ и некоторая вначале неизвестная целевая функция *Φ*, которая для каждой пары <документ, категория> определяет, соответствуют ли они друг другу: $\Phi: D \times C \rightarrow \{0,1\}$. Задача состоит в том, чтобы найти максимально близкую к функции *Φ* функцию *Φ′*. Функцию *Φ′* называют классификатором. Машинное обучение основывается на начальном множестве документов $\Omega = \{d_1, ..., d_{|\Omega|}\} \subset D$. При этом, значение целевой функции *Φ* известно для каждой пары $< d_j, c_i > \in \Omega \times C$. Документы из *Ω* разделяют на два непересекающиеся множества:

"Учебное" множество $T_r = \{d_1, ..., d_{|T_r|}\}$ и множество документов, с помощью которой создается классификатор *Φ′*. *Φ′* обучается индуктивно, основываясь на замеченных характеристиках этих документов.

"Тестовое" множество $T_e = \{d_{|T_r+1|}, ..., d_{|\Omega|}\}$ и множество документов, на котором тестируется эффективность построенного классификатора. Каждый "тестовый" документ подается на вход классификатора *Φ′*, а затем сравнивается результат классификатора $\Phi'(d_j, c_i)$ с известным значением функции $\Phi(d_j, c_i)$. Классификатор считается тем эф-

фективнее, чем чаще эти значения совпадают.

Документ $d \in \Omega$, называется положительным или отрицательным примером для категории *c*, если значение функции *Ф(d,c)* равно *1* или *0*, соответственно[7].

### Методы машинной классификации

Существует несколько наиболее известных методов классификации:

1. Метод Байеса.

Метод Байеса основан на анализе совместных распределений признаков документа и категорий [8]. Документу $D = < d_1, d_2, ..., d_n >$ сопоставляется наиболее вероятная апостериорная категория, определяемая по формуле:

$$c^* = \arg\max_{c \in C} P(c \mid x_1 = d_1, x_2 = d_2, ..., x_n = d_n) \quad (1)$$

Апостериорная вероятность принадлежности документа некоторой категории вычисляется по формуле Байеса, связывающей априорную вероятность с апостериорной:

$$P(c \mid x_1 = d_1, x_2 = d_2, ..., x_n = d_n) = \frac{P(x_1 = d_1, x_2 = d_2, ..., x_n = d_n \mid c) \cdot P(c)}{P(x_1 = d_1, x_2 = d_2, ..., x_n = d_n)} \quad (2)$$

Подставляя (2) в (1), получаем:

$$c^* = \arg\max_{c \in C} \frac{P(x_1 = d_1, x_2 = d_2, ..., x_n = d_n \mid c) \cdot P(c)}{P(x_1 = d_1, x_2 = d_2, ..., x_n = d_n)} \quad (3)$$

Так как знаменатель не зависит от категории, его можно исключить:

$$c^* = \arg\max_{c \in C} P(x_1 = d_1, x_2 = d_2, ..., x_n = d_n \mid c) \cdot P(c) \quad (4)$$

Условные вероятности $P(x_1 = d_1, x_2 = d_2, ..., x_n = d_n \mid c)$ можно вычислить в предположении условной независимости переменных $x_1, x_2, ..., x_n$. В этом случае, формула для определения наиболее вероятной категории будет выглядеть следующим образом:

$$c^* = \arg\max_{c \in C} P(c) \cdot \prod_{i=1..n} P(x_i = d_i \mid c) \quad (5)$$

Для множества обучающих документов вероятности $P(x_i = d_i \mid c)$ вычисляются по формуле [9,10,11]:

$$P(x_i = d_i \mid c) = \frac{|\{D \in Ex \mid c \in Rub(D) \wedge D_i = d_i\}| + 1}{|\{D \in Ex \mid c \in Rub(D)\}|} \quad (6)$$

2. Метод опорных векторов SVM (Support Vector Machines).

Метод опорных векторов [12,13,14] разработан на основе принципа структурной минимизации риска – одновременного контроля количества ошибок классификации на множестве для обучения и «степени обобщения» обнаруженных зависимостей.

Нахождение оптимальной плоскости разделения множеств методом SVM сводится к решению оптимизационной задачи с линейными ограничениями типа равенств и неравенств [12]:

$$L_D(\alpha) = \sum_{i=1}^{l} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{n} \alpha_i \alpha_j y_i y_j K(x_i, x_j) \to \max, \quad 0 \le \alpha_i \le C, \sum_{i=1}^{l} \alpha_i y_i = 0 \quad (7)$$

Здесь $K(x_i, x_j)$ – функция ядра SVM, которая в простейшем случае равна евклидову скалярному произведению векторов $x_i$ и $x_j$. Для решения задачи (7) предложены эффективные методы решения [15,16].

3. Метод ближайших соседей.

Метод ближайших соседей – простейший метрический классификатор, основанный на оценивании сходства объектов. Метод ближайшего соседа является, пожалуй, самым простым алгоритмом классификации. Классифицируемый объект *x* относится к тому классу $y_i$, которому принадлежит ближайший объект обучающей выборки $x_i$. Метод *k* ближайших соседей. Для повышения надёжности классификации объект относится к тому классу, которому принадлежит большинство из его соседей – *k* ближайших к нему объектов обучающей выборки $x_i$. В задачах с двумя классами число соседей берут нечётным, чтобы не возникало ситуаций неоднозначности, когда одинаковое число соседей принадлежат разным классам.

Основная формула: Пусть задана обучающая выборка пар «объект–ответ» $X^m = \{(x_1, y_1), ..., (x_m, y_m)\}$. Пусть на множестве объектов задана функция расстояния

$\rho(x,x')$. Эта функция должна быть достаточно адекватной моделью сходства объектов. Чем больше значение этой функции, тем менее схожими являются два объекта $x,x'$. Для произвольного объекта $u$ расположим объекты обучающей выборки $x_i$ в порядке возрастания расстояний до $u$: $\rho(u,x_{1;u}) \le \rho(u,x_{2;u}) \le ... \le \rho(u,x_{m;u})$, где через $x_{i;u}$ обозначается тот объект обучающей выборки, который является $i$-м соседом объекта $u$. Аналогичное обозначение введём и для ответа на $i$-м соседе: $y_{i;u}$. Таким образом, произвольный объект $u$ порождает свою перенумерацию выборки. В наиболее общем виде алгоритм ближайших соседей есть:

$$a(u) = \arg\max_{y \in Y} \sum_{i=1}^{m} [x_{i;u} = y] \omega(i,u) \qquad (8)$$

где $\omega(i,u)$ – заданная весовая функция, которая оценивает степень важности $i$-го соседа для классификации объекта $u$. Естественно полагать, что эта функция неотрицательна и не возрастает по $i$. По-разному задавая весовую функцию, можно получать различные варианты метода ближайших соседей, $\omega(i,u) = [i = 1]$ – простейший метод ближайшего соседа, $\omega(i,u) = [i \le k]$ – метод $k$ ближайших соседей[17].

### Оценка качества классификации

С целью определения наилучшего метода были предложены следующие критерии: полнота; точность. Для оценки качества классификации используются метрики из информационного поиска:

Полнота (recall): отношение количества найденных документов из категории к общему количеству документов категории.

Точность (precision): Определяется как отношение числа релевантных документов, найденных ИПС (информационно-поисковые системы), к общему числу документов.

$$recall = \frac{|D_{rel} \cap D_{retr}|}{|D_{rel}|} \qquad (9)$$

$$precision = \frac{|D_{rel} \cap D_{retr}|}{|D_{retr}|} \qquad (10)$$

где $D_{rel}$ – это множество релевантных документов в базе, а $D_{retr}$ – множество документов, найденных системой[18].

### Тестовые коллекции для классификации

Существует несколько наиболее известных коллекции для экспериментов.

Коллекции для английского языка: Reuters-21578 Text Categorization Collection (http://kdd.ics.uci.edu/databases/reuters21578/reuters21578.html); OHSUMED Test Collection (http://ir.ohsu.edu/ohsumed/); 20 Newsgroups Data (http://people.csail.mit.edu/jrennie/20Newsgroups).

Коллекции для китайского языка: SogouT (http://www.sogou.com/labs/dl/t.html).

Коллекции для русского языка: Тестовые коллекции РОМИП (Российский семинар по Оценке Методов Информационного Поиска) (http://romip.ru/ru/collections/index.html).

### Результаты сравнение методов машинного обучения

В статьях [7,8,10,11] сравниваются различные методы машинного обучения. Проведенные эксперименты показали, что метод SVM имеет преимущество перед другими методами машинного обучения, поскольку обеспечивают самую высокую точность и полноту. Метод k ближайших соседей обеспечивает наилучшее время но самую низкую точность и полноту. Метод Байеса даёт средние временные характеристики, а также точность и низкую полноту. Поэтому можно рекомендовать для классификации использован SVM метод.

### Выводы

Выбранный метод и результаты работы подтверждают возможность создания эффективной системы автоматической классификации документов для конечного множества языков по критерию принадлежности к определенной области знания, используя современные средства вычислительной техники.

## Список литературы


1. http://ru.wikipedia.org/wiki/Обработка_естественного_языка.
2. Загибалов Т. Е. Автоматический анализ текстов на китайском языке. Проблема выбора базовой единицы. // Труды международной конференции "Диалог 2005".
3. http://ru.wikipedia.org/wiki/Стеммер_Портера.
4. http://www.ictclas.org/
5. http://snowball.tartarus.org/
6. http://ru.wikipedia.org/wiki/Классификация_документов.
7. Юрий Лифшиц. Курс "Алгоритмы для Интернета" РАН 2006.
8. Yang Y., Liu X. A re-examination of text categorization methods. // Proc. of Int. ACM Conference on Research and Development in Information Retrieval (SIGIR-99), 1999 — p. 42-49.
9. Joachims T. A Probabilistic Analysis of the Rocchio Algorithm with TFIDF for Text Categorization. // Proceedings of ICML-97, 14th International Conference on Machine Learning. — 1996. — p. 2-13.
10. Yang Y. An Evaluation of Statistical Approaches to Text Categorization. / Journal of Information Retrieval, 1999 — V.1 — p. 67--88.
11. Joachims T. Text Categorization with Support Vector Machines: Learning with Many Relevant Features. // Proceedings of ECML-98, 10th European Conference on Machine Learning — 1998. — p. 6-18.
12. Vapnik V. The Nature of Statistical Learning Theory. — Springer-Verlag — New York, 1995. — p. 123-167.
13. Burges C.J.C. A tutorial on support vector machines for pattern recognition. // Data Mining and Knowledge Discovery, 1998. — p. 955-974,
14. Вапник В.Н. Восстановление зависимостей по эмпирическим данным. — М.: Наука, 1979. —с. 223-255.
15. Joachims T. Making Large-Scale SVM Learning Practical. Advances in Kernel Methods // Support Vector Learning, Burges C., Smola A. (ed.), — MIT-Press, 1999. —p. 5-12.
16. Joachims T. Estimating the Generalization Performance of a SVM Efficiently. // Proceedings of the International Conference on Machine Learning, — Morgan Kaufman, 2000. —p. 5-22.
17. http://www.machinelearning.ru/wiki/index.php?title=Метод_ближайших_соседей.
18. http://ru.wikipedia.org/wiki/Информационный_поиск.


Поступила в редакцию 17.12.2009

УДК 519.6

*ЧЕРТОВ О.Р.*

# ПОЛІНОМИ КУНЧЕНКА ДЛЯ РОЗПІЗНАВАННЯ ОБРАЗІВ

В статті розглянута задача розпізнавання образів за допомогою застосування поліномів наближення Кунченка. Викладення ведеться на прикладі виділення шаблону (зразка) стереотипної поведінки зловмисника, що займається мереживим шахрайством типу «склікування». Запропоновано новий підхід до аналізу часових рядів, що містять дані про «склікування». Показано, що це дозволяє визначити певні типи недійсних переходів по рекламним посиланням. Обґрунтовано перспективність застосування вказаного підходу до дослідження моделей в області статистики та соціології.

This paper concerns the task of pattern recognition by using Kunchenko's approximating polynomials. We take a stereotyped behavior template matching of intruder during click fraud as an example. New approach for clicks' time series analyzing is proposed. It has been shown, that with the help of proposed techniques it is possible to define some invalid clicks. The prospects for further research in area of statistics and sociology are confirmed.

## Вступ

Класифікація об'єктів (образів) за певними категоріями чи класами називається розпізнаванням образів. Задачі, які відносяться до розпізнавання образів, виникають в багатьох областях – від медичної діагностики, статистики, геології до сканування паперових документів чи дактилоскопії. Виділяють чотири групи методів розпізнавання образів [1]:

1) порівняння зі зразком (template matching), коли застосовується геометрична нормалізація та рахується відстань від певного зразка;
2) статистична класифікація (statistical classification), коли для кожного класу будується свій розподіл і здійснюється класифікація за правилом Байєса;
3) синтаксичне чи структурне порівняння (syntactic or structural matching), коли об'єкт ділиться на елементи та класифікується в залежності від того – містить він чи ні окремі елементи або їх послідовності;
4) нейронні мережі (neural networks), коли вибирається певний тип мережі та налагоджуються його коефіцієнти.

В статті ми зупинимося на першій з вказаних груп методів і дослідимо випадок, коли в початковому сигналі необхідно розпізнати заздалегідь відомий зразок (шаблон) [2]. Такі задачі типові при статистичному аналізі даних, в радіолокації, електрозв'язку, сейсморозвідці тощо, коли при моделюванні складної системи довільної природи виникає потреба в поліноміальному наближенні функції.

Наразі найширше застосування при розв'язанні вказаних задач знаходять класичні поліноми Тейлора (чи Маклорена) та поліноми (ряди) Фур'є. Загальновідомо, що поліном Тейлора є поліномом найкращого наближення функції в околі заданої точки. Але функція, що наближується, повинна мати в цьому околі похідні відповідного порядку. Значно слабші умови накладаються при розвиненні функції в ряд Фур'є по системі незалежних ортогональних чи ортонормованих функцій, що утворюють базис відповідного простора зі скалярним добутком. Проте з точки зору розпізнавання в вихідному сигналі якогось певного зразка більш перспективним представляється використання так званих поліномів Кунченка [3]. Пов'язано це з тим, що поліноми Кунченка будуються в особливому підпросторі евклідова чи гільбертова простора, який має породжувальний елемент. В якості такого елементу можна взяти саме шуканий зразок. Таким чином, задача співставлення зі зразком зводиться до більш простої задачі пошуку найближчого (в певній метриці) апроксимаційного полінома Кунченка.

В даній роботі зазначений підхід демонструється шляхом використання поліномів наближення Кунченка, насамперед, для боротьби зі «склікуванням» (click fraud) – одним із різновидів мережевого шахрайства, що має місце в сфері Інтернет-реклами. На даний момент не має одностайного погляду на реальні масштаби цього виду шахрайства, але, наприклад, дослідження спеціалізованої компанії Clickforensics [4] свідчить про те,

що на кінець 2009 р. не менше 15 % всіх переходів за рекламними об'явами в Інтернеті були здійснені або помилково, або з шахрайською метою. Тому пошук нових методів розпізнавання «склікування», зокрема, за допомогою поліномів Кунченка є актуальним.

### Огляд останніх досліджень і публікацій

Найбільш повне викладення теорії поліномів Кунченка наведено в [3]. В цій роботі, зокрема, були виділені чотири основні відмінності вказаних просторів та поліномів від інших, які також застосовуються при розв'язанні різних науково-технічних проблем:

1) базис вихідного лінійного простору складається лише з однієї (породжувальної) функції;

2) сам вказаний простір містить лише лінійно незалежні та, в загальному випадку, не ортогональні породжені функції;

3) довільна функція із множини породжених функцій може бути наближена лінійною комбінацією (тобто поліномом, який і називається поліномом Кунченка) із будь-яких інших (додаткових) породжених функцій;

4) оскільки функція, що наближується, та функції, що наближують, є лінійно незалежними, то за будь-якого скінченного порядку полінома наближення не існує таких його коефіцієнтів, при котрих він співпаде з наближуємою функцією.

Але в роботах професора Ю.П. Кунченка та його учнів, зокрема, в [5] і [6], основна увага приділяється застосуванню поліномів наближення для фільтрації негауссівських випадкових процесів та величин в радіотехніці, радіолокації та зв'язку. В статті [7], найбільш близькій за тематикою до даної, будується оптимальний за певним критерієм якості алгоритм, що дозволяє по апріорному моментно-кумулянтному опису сигналу вказати образ, тобто сигнал, якому він відповідає. Але, знову ж таки, фактично розглядається задача розпізнавання постійних радіосигналів на тлі асиметричних та ексцесних негауссівських завад.

В даній роботі ми пропонуємо застосувати поліноми Кунченка в принципово іншій проблемній області, а саме: для розпізнавання в часових рядах, що містять інформацію про переходи користувача за рекламними Інтернет-посиланнями, тих шаблонів (зразків), що, скоріш за все, відповідають поведінці зловмисника, який намагається під час демонстрації контекстної реклами завдати фінансової шкоди.

Контекстна реклама є одним із найбільш ефективних та рентабельних видів реклами, бо тематика рекламних об'яв підбирається у відповідності до прогнозуємих інтересів користувача. Наприклад, об'ява може бути показана разом з результатами пошуку за ключовим словом, що було введене відвідувачем. Таким чином, забезпечується максимально ефективне використання коштів, які вкладаються в рекламну кампанію, оскільки рекламна об'ява демонструється лише потенційним клієнтам. Більше того, за допомогою таргетингу (налагодження показу об'яв за часом та місцерозташуванням відвідувачів) можна ще більше звузити коло клієнтів, відкинувши явно неперспективних.

Проте, суттєвим недоліком контекстної реклами є проблема «склікування» – одного із різновидів мережевого шахрайства, коли імітується «клік», тобто перехід реального користувача за рекламним оголошенням, для зняття з відповідного рекламодавця плати за здійснений перехід. Загалом, всі «кліки» користувача по рекламних оголошеннях можна розділити на дві групи: дійсні, тобто ті, що зроблені користувачем з метою переходу на сайт, який рекламується, і недійсні, тобто такі, що зроблені помилково чи з метою шахрайства.

Існує декілька методів, що наразі застосовуються для боротьби зі «склікуванням».

Найбільш поширений метод полягає в порівнянні поточної активності відвідувачів з їх активністю в минулому. За наявності суттєвих відхилень вважається, що рекламна об'ява «зазнала» нападу зловмисників. Але використовуючи даний метод, неможливо врахувати різке збільшення зацікавленості до пропонуємої продукції. Бо воно може бути викликане не тільки розгортанням запланованої рекламної кампанії, але й зовнішніми обставинами. Наприклад, якщо в якомусь місті трапилася серйозна аварія на електростанції, то з великою ймовірністю зросте попит на продукцію компанії, що займається реалізацією автономних джерел живлення.

Другий метод базується на алгоритмах, що керуються правилами («rules-based algorithms») [8]. Суть методу визначається наяв-

ністю таких певних умов, коли «клік» вважається недійсним. Кожен перехід відвідувача проходить через систему фільтрації цими алгоритмами, в результаті чого приймається рішення про те, чи є відповідний «клік» дійсним або недійсним. Підготовкою таких алгоритмів і їх налагодженням (підбором параметрів) займаються відповідні фахівці.

Третій метод використовує для аналізу передісторію індивідуальної активності користувача. Зі збільшенням накопичуємої бази даних стає простіше визначити дійсні переходи відвідувача, ніж недійсні. Даний метод базується на двох припущеннях: в попередніх моделях поведінки користувача відсутні недійсні «кліки» і ці моделі однозначно визначають майбутню поведінку відвідувачів.

Четвертий метод включає ряд алгоритмів, що базуються на підрахунку та врахуванні кількості показів між послідовними «кліками» [9].

Але всі ці методи фактично є чисто статистичними, в них не враховується поведінка можливого зловмисника.

### Постановка задачі

Мета роботи полягає в дослідженні принципової можливості застосування поліномів наближення Кунченка для вирішення задачі співставлення зі зразком на прикладі побудови та аналізу моделей поведінки можливих учасників процесу «склікування», які потенційно можна в подальшому використовувати для визначення частини недійсних «кліків» та їх попередження.

### Взаємодія учасників процесу надання контекстної реклами

В процесі створення та проведення рекламної кампанії в Інтернеті, як правило, задіяні наступні учасники: рекламодавець, рекламне агентство, пошукова мережа, рекламний майданчик і клієнт (людина, якій адресовані рекламні повідомлення). Характер їх взаємодії наведено на рис. 1.

Якщо рекламодавець вирішує розмістити свої об'яви в Інтернеті, то він може це зробити самостійно, скориставшись автоматизованими продуктами, котрі надаються, наприклад, такою рекламною мережею як Google. Але в цьому випадку йому потрібно буде входити в особливості налагодження рекламних кампаній.

Альтернативний варіантом є використання послуг рекламних агентств. Перевагою такого підходу є також те, що спеціалізовані агентства мають відповідну кваліфікацію та зможуть забезпечити проведення рекламної кампанії на високому рівні.

Сплата послуг розміщення реклами здійснюється в залежності від обраної моделі рекламної кампанії. Модель CPC (cost per click) передбачає сплачування за кожен перехід за об'явою. Обираючи модель CPM (cost per mille), рекламодавець платить за тисячу показів реклами. Модель CPA (cost per action) гарантує, що оплачуватися буде тільки якась наперед визначена дія відвідувача.

В залежності від налагоджень рекламної кампанії об'ява буде розміщуватися в якості результатів пошуку по ключовим словам, заздалегідь визначеним рекламодавцем, або на сайтах, суміжних за тематичним спрямуванням. Власник сайту, на якому розміщується рекламне повідомлення, отримує певну суму коштів за переходи по об'явам чи за тривалість розміщення останніх на своєму сайті.

Здійснювати певні неправомірні дії, тобто виконувати «склікування» («клікати» по об'явах при моделі CPC) або «споказувати» (генерувати штучні покази при моделі CPM) можуть різні групи учасників. Наприклад, власник сайту – з метою підвищення своїх доходів, рекламодавець – для розтрачання рекламного бюджету своїх конкурентів тощо.

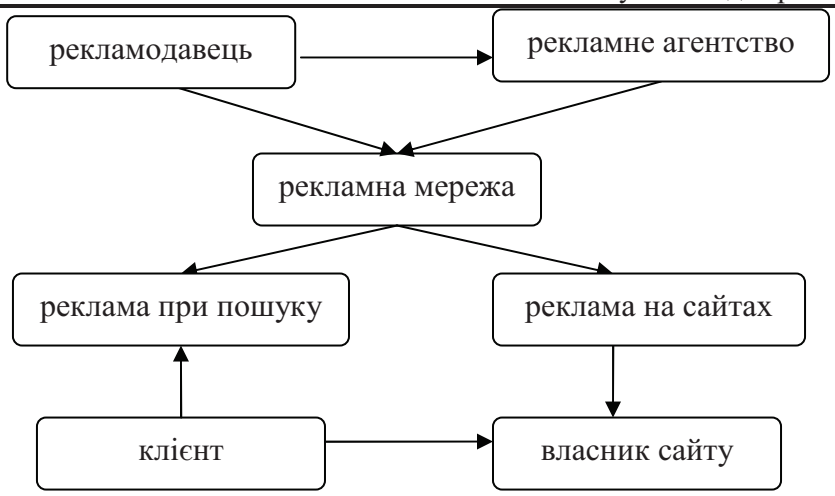



***Рис. 1. Взаємодія учасників проблемної області***

### Часові ряди та шаблони поведінки потенційних зловмисників під час «склікування»

Системи моніторингу Google AdWords/AdSense, Yahoo! Search Marketing чи Microsoft adCenter перевіряють кожен «клік» за такими параметрами як IP-адреса, час виконання тощо. В даній роботі будемо використовувати такі параметри, як кількість переходів за рекламними об'явами на день та щогодини впродовж деякого проміжку часу (тиждень, місяць). Братимемо до уваги лише ті «кліки», що будуть визнаватися дійсними відповідними системами моніторингу.

Розглянемо детально спочатку найбільш розповсюджену ситуацію, коли зловмисник-рекламодавець «склікує» об'яви свого головного конкурента, щоб підірвати його рекламну кампанію. Стратегічна задача зловмисника – «склікати» об'яву таким чином, щоб вона перестала відображатися серед результатів пошуку. Це відбудеться за вичерпання добового бюджету рекламної кампанії (суми, котру рекламодавець згоден витратити на рекламу протягом доби).

Для опису моделі поведінки цього учасника можна скористатися моделлю інформаційної атаки [10]. В нашому випадку інформаційна атака включає такі фази як «фоновий шум», «спроба», «затишшя», «атака». Ці фази відповідають наступним етапам процесу «склікування»:

- природний рівень «кліків», тобто «кліки», що зроблені реальними відвідувачами до тих пір, поки сайт ще не став жертвою «склікування»;
- перші спроби «склікати» об'яву – зловмисник ще не знає напевне, яким чином буде реагувати система моніторингу пошукової мережі, яким є розмір добового бюджету (тобто скільки «кліків» потрібно зробити, щоб об'ява перестала відображатися серед результатів пошуку);
- шахрайський вплив на рекламну об'яву призупиняється на певний час, щоб знизити ймовірність розкриття шахрайства та прийняти рішення про момент нападу;
- власне, атака, в результаті котрої рекламний добовий бюджет об'яви буде витрачений протягом короткого проміжку часу.

Відповідні елементи відобразяться у вигляді локальних та глобальних максимумів на часовому ряді кількості «кліків» (див. рис. 2). Екстремуми відповідають фазам «спроби» і «атаки». При цьому другий максимум буде більшим за перший. Різниця між цими максимумами може бути досить суттєвою, бо в результаті збільшення кількості «кліків» підніметься значення показнику CTR (click-through rate), а, відповідно, вартість одного «кліку» зменшиться. Отже, фазі «атаки» буде відповідати інтервал з найбільш високою щільністю «кліків» на протязі дня.

Як для рекламодавця, що бореться шахрайськими методами зі своїми конкурентами, так і для інших учасників процесу проведення рекламної кампанії в Інтернеті, можні виділити інші стереотипні моделі (шаблони) поведінки.

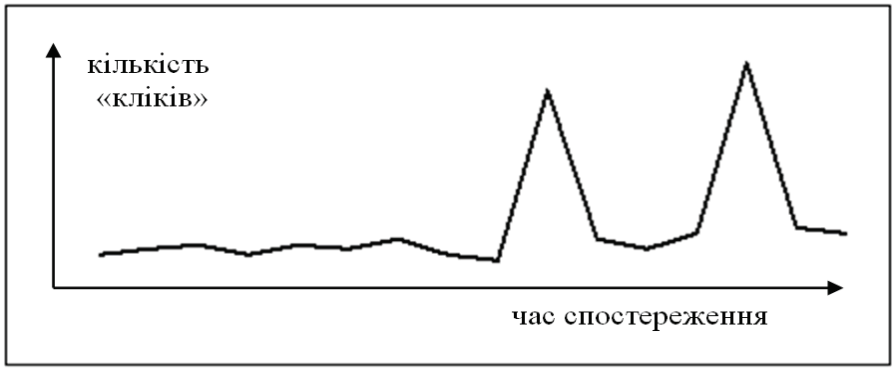



***Рис. 2. Модель «склікування» типу інформаційної атаки***

Наприклад, «склікувати» рекламні об'яви можуть також зловмисники з метою заподіяння максимальних збитків пошуковим мережам. В цьому випадку дії зловмисників сконцентруються на найдорожчих рекламних оголошеннях. При цьому буде відсутній взаємозв'язок між оголошеннями, а саме: місце розміщення та продукція, що рекламується можуть суттєво відрізнятись. Кількість «кліків» по кожному з оголошень не буде пов'язана жодною залежністю, проте збільшення активності відвідувачів матиме місце в один і той же час. На часовому ряді це матиме вигляд приблизно такий, як на рис. 3.

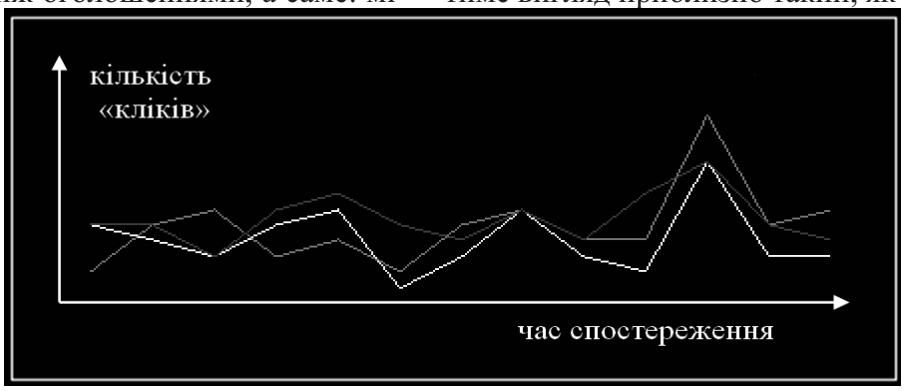



***Рис. 3. Модель поведінки зловмисників при «склікуванні» найдорожчих об'яв***

### Побудова поліномів Кунченка

Виділені в попередньому розділі (чи аналогічні їм) моделі поведінки зловмисників як певні шаблони (зразки) функціональної залежності кількості «кліків» від часу спостереження можна взяти в якості єдиного базисного елементу вихідного лінійного простору, тобто в якості його породжувального елементу $e$.

Тоді як лінійну комбінацію лінійно-незалежних перетворень $f_1(e), f_1(e), \ldots, f_n(e)$ відповідного породжувального елементу можна побудувати поліном $P_n$ наближення $n$–го порядку до (частини) вихідного сигналу $f_s(e)$:

$$P_n = \sum_{\substack{k=0,\\ k\neq s}}^{n} c_k f_k(e),$$

де коефіцієнти $c_k$ знаходяться із умови забезпечення мінімуму відстані між будуємим поліномом та вихідним сигналом. Як показано в [3, с. 95-96] при цьому

$$c_0 = \frac{< f_s(e), f_0(e) > - \sum\limits_{\substack{k=1,\\ k\neq s}}^{n} c_k < f_k(e), f_0(e) >}{< f_0(e), f_0(e) >}$$

а інші коефіцієнти $c_k$ знаходяться як розв'язок системи лінійних рівнянь:

$$\sum_{\substack{k=1,\\ k\neq s}}^{n} c_k F_{i,k} = F_{i,s},\ \overline{i=1,n},\ i \neq s,$$

де центровані корелянти $F_{i,k}$ також рахуються за допомогою скалярних добутків відповідних перетворень:

$$F_{i,k} \equiv < f_i(e), f_k(e) > - \\ - \frac{< f_i(e), f_0(e) > \cdot < f_k(e), f_0(e) >}{< f_0(e), f_0(e) >}.$$

Чисельною характеристикою, яку можна застосовувати в критеріях якості співставлення сигналу з виділеним шаблоном, тобто як міру наближення полінома Кунченка $P_n$ до (частини) вихідного сигналу $f_s(e)$ є коефіцієнт ефективності $d_n$:

$$d_n = \frac{\sum\limits_{\substack{k=1,\\ k\neq s}}^{n} c_k < f_k(e), f_s(e) >}{< f_s(e), f_s(e) >}.$$

## Застосування поліномів Кунченка при дослідженні моделей в статистиці та соціології

Розглянутий метод розпізнавання певних зразків за допомогою побудови простору з породжувальним елементом та пошуку коефіцієнтів відповідного полінома Кунченка може бути використаний в будь-якій проблемній області, в якій можна апріорі в часовому чи просторовому ряді виділити певні характерні шаблони. Особливо перспективним представляється його застосування до так званих «м'яких» моделей (в термінології Рене Тома). При побудові таких моделей виходять із припущень та гіпотез про суть явищ та процесів, які описуються, роблять висновки з цих гіпотез та уточнюють самі гіпотези. «М'які» моделі характерні, насамперед, для статистики і соціології.

Наприклад, маючи такий інструментарій як інформаційно-аналітична система обробки даних Всеукрїнського перепису населення 2001 р. [11], можна досить просто відшукати певні закономірності розподілу респондентів за територією, сімейним станом, освітою тощо. Перевірку ж того, наскільки віднайдена гіпотеза справедлива для генеральної сукупності, можна здійснити за допомогою описаного метода.

## Висновки

Побудувавши типові моделі поведінки можливих мережевих зловмисників під час надання контекстної реклами і створивши шаблони на їх основі, можна застосовувати метод на базі поліномів Кунченка для визначення (і попередження) можливого нападу. Для цього потрібно в якості породжувального елементу відповідного лінійного простору взяти виділений шаблон поведінки зловмисника та, скориставшись коефіцієнтами ефективності, виділити в часовому ряді кількості «кліків» даний шаблон як зразок (в розумінні теорії розпізнавання образів).

В роботі також обґрунтована перспективність застосування вказаного підходу до дослідження моделей в області статистики та соціології.

## Список посилань


1. Jain A.K., Duin P.P.W., Mao J. Statistical Pattern Recognition: A Review // IEEE Transactions on pattern analysis and machine intelligence. – 2000. – Vol. 22, № 1. – P. 4-37.
2. Brunelli R. Template matching techniques in computer vision: theory and practice. – Chippenham: Wiley, 2009. – 348 p.
3. Кунченко Ю.П. Приближения в пространстве с порождающим элементом. – К.: Наук. думка, 2003. – 243 с.
4. Click Fraud Index [Electronic resource]. – Режим доступу: http://www.clickforensics.com/resources/click-fraud-index.html
5. Кунченко Ю.П., Заболотній С.В. Поліноміальні оцінки параметрів близьких до гауссівських випадкових величин. Частина II. Оцінка параметрів близьких до гауссівських випадкових величин. – Черкаси: ЧІТІ, 2001. – 251 с.
6. Лега Ю.Г., Гончаров Ю.Г., Філіпов В.В. Асимпотичні властивості оцінок параметра постійного сигналу при усіченому оцінюванні дисперсії асиметричної завади другого типу першого виду // Вісник Черкаського державного технологічного університету. – 2008. – № 3.– С. 3-8.
7. Палагін В.В., Жила О.М. Поліноміальне вирішення задач розпізнавання випадкових сигналів // Вісник Черкаського державного технологічного університету. – 2008. – № 2. – С. 31-35.
8. Matin S. Click Ahoy! Navigation Online Advertising in a Sea of Fraudulent Clicks // Berkeley Technology Low Journal, Annual Review 2007. – Vol. 22, № 1. – P. 533-554.
9. Immorlica N., Jain K., Mahdian M., Talwar K. Click Fraud Resistant Methods for Learning Click-Through Rates // Technical Report. – 2005. – Vol. 3828. – P. 34-45.
10. Фурашев В.М., Ланде Д.В. Практичні засади прогнозування можливих загроз та ризиків шляхом аналізу взаємозв'язку подій з інформаційним простором // Открытые информационные и компьютерные интегрированные технологии: сб. науч. трудов. Вып. 42. – Харьков: Нац. аэрокосм. ин-т «ХАИ», 2009. – С. 194-203.
11. Чертов О.Р. Система многомерного анализа данных Всеукраинской переписи населения 2001 года // Россияне в зеркале статистики: Всероссийская перепись населения 2002 года: Международный симпозиум, 30-31 марта 2004 г.: труды симп. – М.: Изд-во Федеральной службы государственной статистики, 2004. – С. 234-238.


Поступила в редакцию 4.12.2009

УДК 004.71

*ОРЛОВА М.М.,*
*ШАНДИБА Д.В.*

# РЕАЛІЗАЦІЯ МЕРЕЖЕВИХ ПРОТОКОЛІВ ЯК НЕЗАЛЕЖНИХ ПРОЦЕСІВ

В статье рассматривается подход к реализации сетевых протоколов как независимых процессов. Использование предлагаемого метода позволяет реализовывать логику каждого сетевого протокола в стеке полностью независимо от других протоколов, что значительно облегчает применение формальных методов для решения задачи синтеза протоколов. В результате исследования сформулирована и опробована методика формального описания обобщенных моделей поведения сетевых протоколов, а также внешних механизмов их взаимодействия.

The article reviews the approach to network protocols implementation as independent processes. Usage of the proposed method allows implementation of each network protocol in the stack completely independently from other protocols, that significantly simplifies the task of formal methods application for protocols synthesis. As a result of the research there was proposed and practically tried out a methodology of formal specification of generalized models of network protocols behavior, as well as of external mechanism of their interaction.

## Вступ

Формальні методи представлення мережевих протоколів є важливим напрямом розвитку автоматизації розробки програмних технологій створення та підтримки ефективних мережевих систем. Модульне представлення поведінкової логіки мережевих протоколів є однією з підзадач розв'язання якої наближає нас до вирішення загальної проблеми формального подання мережевих протоколів. Використання модульності (зокрема у вигляді загальних абстракцій, таких як кінцеві автомати) в поведінковій логіці протоколів на практиці дозволяє перенести значний об'єм описів поведінкової логіки мережевих протоколів у формалізовані форми, які, в свою чергу, надають можливості автоматизації проектування, реалізації, верифікації та контролю за роботою мережевих протоколів. Пошуки вирішення кожної із цих задач займають важливе місце в сфері мережевого програмування.

## Існуючі дослідження і публікації

Загалом дослідження у сфері формального представлення мережевих протоколів виконуються досить давно і набули свого розвитку у трьох головних напрямках, загальний огляд яких наведено у [1]:

1. використання формальних методів із метою аналізу характеристик протоколів ще до їх реалізації;
2. використання формальних методів із метою контролю за роботою вже реалізованих протоколів, вирішення задач мережевої безпеки;
3. використання формальних методів із метою синтезу (реалізації) протоколів.

Розробки останньої групи, в свою чергу, виконуються двома основними шляхами: шляхом створення нових мов програмування вузького застосування (domain-specific languages), із, відповідно, розробкою специфікацій цих мов та повної інфраструктури їх підтримки [8, 9, 10], а також, рідше, шляхом модифікації (розширення) існуючих мов програмування [2]. В обох випадках практичне застосування отриманих реалізацій потребує наявності додаткових інструментів розробки (компіляторів або трансляторів), а часто також і додаткових дій, необхідних для інтеграції отриманих реалізацій у програмні продукти, що мають ці реалізації використовувати. Необхідність цього є суттєвим недоліком при практичному використанні таких методів: фактично формалізація процесу розробки протоколів вищезазначеними способами дає ресурсний виграш при реалізації правил протоколу, проте потребує додаткового часу та зусиль для реального використання цих можливостей. Тож проблема формалізації протоколів шляхом, привабливим з практичної точки зору, все ще потребує вирішення.

Окремою задачею є проблема формального представлення даних, якими оперують протоколи. На відміну від представлення поведінкової логіки, вона більше досліджена, і

результатами цих досліджень є численні мови опису даних та супутні технології, розглянуті в [9, 11].

### Мета

Метою дослідження, викладеного в даній статті, є розробка моделі реалізації поведінкової логіки мережевих протоколів, яка б з одного боку могла бути безпосередньо виражена в термінах популярних сьогодні мов програмування виробничого маштабу (таких як C/C++, Java, .Net мови) без застосування додаткових інструментів, а з іншого – мала б характеристики, що дозволяли б її опис, формування (створення) та аналіз із застосуванням формальних методів.

### Пропонований підхід

Традиційний підхід до реалізації поведінкової логіки мережевих протоколів (як із використанням звичайного програмування, так і формальних методів) полягає у використанні ієрархічності природи мережевих протоколів, тобто коли протоколи вищого рівня безпосередньо використовують можливості протоколів нижчого рівня. Це дозволяє уникнути дублювання коду, а також досягти вищих рівнів абстрагування при розробці наступних (вищих) рівнів протоколів. В даному випадку поняття "використовують", як правило, означає безпосередні виклики функцій інтерфейсів протоколів нижчого рівня протоколами вищого рівня.

У більшості випадків така архітектура протоколів є повністю виправданою. Втім, вона істотно погіршує модульні властивості окремих протоколів у стеку, і цей факт набуває особливої гостроти при дослідженні способів формального представлення протоколів. В більшості існуючих підходів до формального представлення поведінкової частини протоколів безпосередній зв'язок між протоколами реалізується через інкапсульовані інтерфейси протоколів нижчого рівня у виклики підсистеми представлення протоколів. Тобто бібліотечний виклик заманюється на формальний літерал, який цей виклик закапсулює. В реалізаціях, основаних на умовно псевдо-незалежних моделях [10], процеси хоча і мають суттєві ознаки відокремленості, проте вони все одно тісно пов'язані між собою. Загальним негативним результатом використання таких підходів є "монолітність" кінцевих реалізацій. Зміна будь–якої поведінкової частини протоколу нижчого рівня потребує перегляду всієї реалізації на предмет виникнення можливих логічних помилок. У випадку існуючих способів формального представлення поведінки протоколів додається ще один суттєвий недолік: внесення можливостей протоколів нижчого рівня в інфраструктуру системи формального визначення протоколів фактично робить створювані системи асиметричними з точки зору маштабованості: інтерфейс створеного, нового протоколу не виглядає таким самим чином, як виглядає інтерфейс протоколу, внесеного в інфраструктуру системи. Це робить такі системи придатними для порівняно ефективного створення лише "фінальних" протоколів, на основі яких важко створювати протоколи вищих рівнів тим самим шляхом, яким було створено даний протокол. Наприклад, у [8] викладено успішну методику створення мережевих протоколів транспортного рівня, яка мало придатна для протоколів прикладного і вищого рівнів через відсутність інтерфейсної сумісності між отримуваними реалізаціями.

Ідеальною моделлю представлення поведінкової логіки мережевого протоколу є абсолютна незалежність рівнів протоколу. Це вирішувало б існуючі проблеми, проте така модель є або принципово недосяжною (при збереженні ідеї повторного використання коду), або недоцільною, внаслідок втрати переваги від повторного використання коду.

Для поєднання з одного боку переваг ідеальної моделі для легкості формалізації протоколів, а з іншого – існуючих практик реалізації протоколів запропоновано розглядати протоколи в одному стеку як окремі незалежні процеси, взаємодія між якими виконується шляхом використання відокремленої сутності, подібної до паттерну прикладного програмування "посередник" (mediator) [6]. Таким чином, реалізація мережевих протоколів стає подібною до реалізації розподілених систем із відповідною специфікою. Кожен з протоколів пропонується реалізовувати відповідно лише до правил власне цього протоколу, а взаємодію між протоколами організовувати завдяки окремому прошарку, винесеному в окрему абстракцію, тобто в певний модуль, який би впливав на поведінку обох протоколів, а більш точно –корегував

би її для кожного протоколу на основі сигналів (повідомлень або викликів), що генеруються кожним із цих протоколів (Рис. 1a). Для більш ніж двох протоколів підхід масштабується або шляхом застосування спільного керуючого модуля для всіх протоколів (Рис. 1b), або шляхом використанням декількох окремих модулів (Рис. 1c).

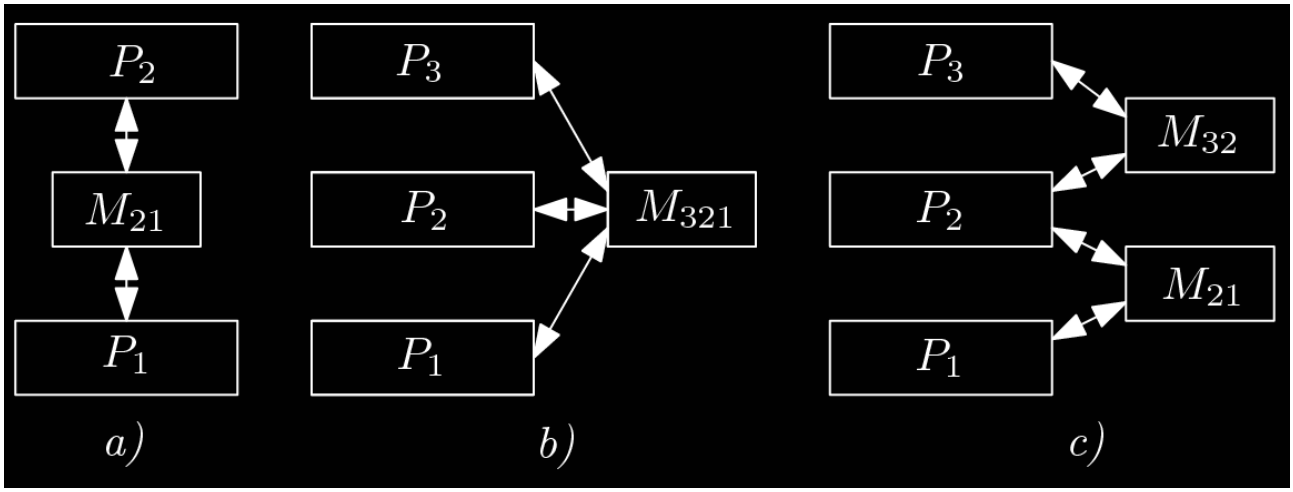



***Рис. 1. Використання посередників для зв'язку між протоколами***

Для реалізації запропонованого підходу необхідно виконувати ряд вимог:

1. Кожен протокол має повністю задекларувати (описати) свої публічні інтерфейсні стани, сигнали та дані, якими він оперує.
2. Будь-які внутрішні (непублічні) процеси та стани протоколу не можуть суперечити його публічному станові.
3. Кожен протокол має бути реалізований в термінах лише цього протоколу.

Завдяки цим правилам опис поведінки кожного протоколу в системі концентрується лише на власному стані протоколу, без залежності від станів інших протоколів, які реалізовані у стеку. Це дозволяє досягти того рівня абстрагування, за якого опис кожного окремого протоколу стає відокремленою задачею, і може бути виконаний довільним запропонованим формальним методом, так як семантично і синтаксично зникають залежності від протоколів інших рівнів, які часто стають на заваді створенню містких зручних формалізацій (специфікацій).

Розглянемо детальніше роль медіатору. Для простоти прикладу оперуватимемо медіатором для двох протоколів. Роль медіатору полягає в тому, щоб на основі інтерфейсних сигналів, що генеруються розроблюваними протоколами, реалізувати залежність станів та переходів протоколів вищого рівня від станів та переходів протоколів нижчого рівня. Фактично, медіатор реалізує ту частину поведінкової логіки протоколу, яка пов'язана із використанням цим протоколом можливостей іншого протоколу (протоколу нижчого рівня). Для рис. 1a, пара $\{P_2, M_{21}\}$ повністю виражає поведінкову логіку протоколу $P_2$, причому $P_2$ описує протокол "як такий", а $M_{21}$ виражає його поведінковий зв'язок із протоколом $P_1$.

На практиці безпосередню реалізацію протоколів та медіатору можна повністю закапсулювати, визначивши наступні частини:

1. Формальний публічний інтерфейс протоколів ($I(P_n)$):

- множина публічних сигналів, що генеруються кожним протоколом ($SG(P_n)$);
- множина публічних сигналів, що приймаються кожним протоколом ($SR(P_n)$);
- опис (перелік) публічних станів протоколів ($T(P_n)$);
- опис публічних даних, якими оперують протоколи ($D(P_n)$).

2. Способи визначення логіки медіатору, в термінах інтерфейсів обох протоколів (процедурне, декларативне або процедурно–декларативне представлення) ($R(M_{nm})$).

Таким чином, схема взаємодії протоколів і медіатору набуває вигляду, представленого на рис. 2. Зауважимо, що кожен із протоколів взаємодіє лише з елементами, які визначені в ньому, і жодним чином не торкається логіки, станів, сигналів чи даних, описаних в іншому протоколі. Задачу цієї "інтеграції" виконує медіатор. Аналогічним чином виконується розміщення клієнтського коду, тобто коду, що в решті решт користується можливостями протоколу, що специфікується: адже

клієнтський код – це, за своєю суттю, той самий медіатор, який тільки поєднує між собою не 2 (або більше) різні рівні протоколів, а програмну бізнес–логіку із протоколом. Така уніфікованість дозволяє використовувати пропоновані засоби моделювання протоколів навіть за межами частин програмного забезпечення, що торкаються специфікації протоколів. А також значно полегшує заміну реалізацій протоколів в кінцевому продукті.

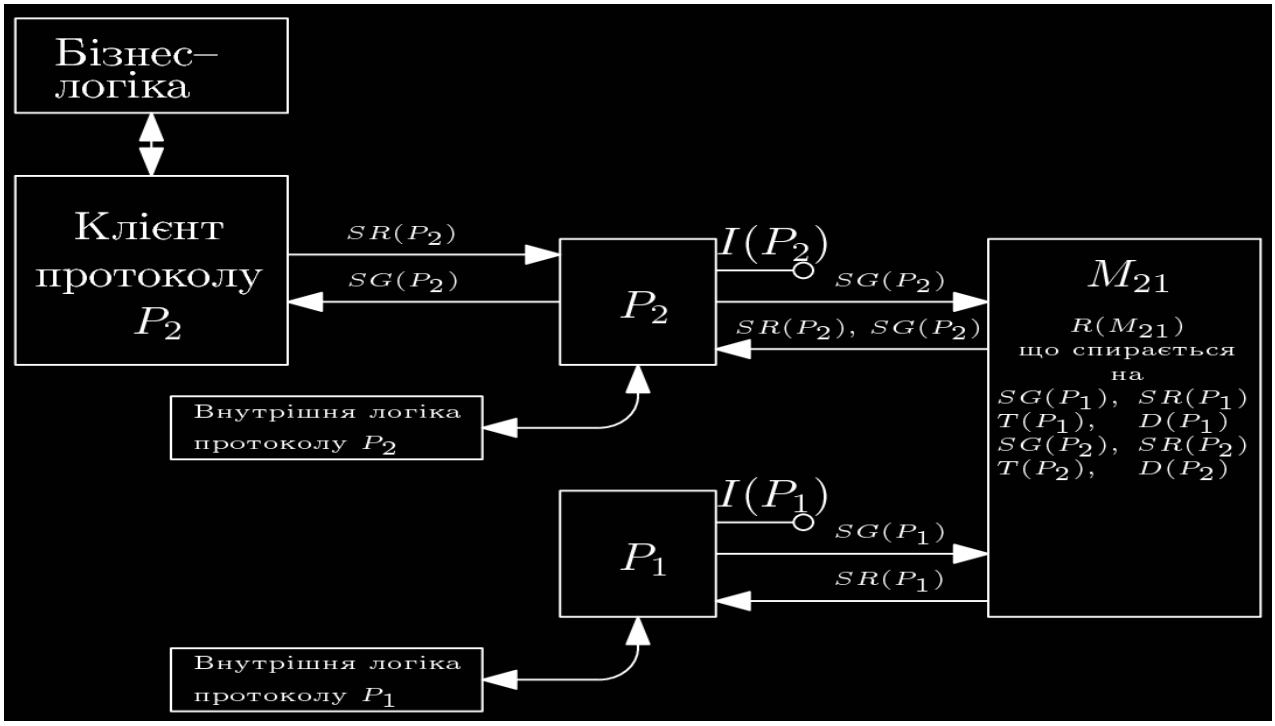



***Рис. 2. Взаємодія елементів системи протоколів***

В залежності від характеру протоколу може змінюватися об'єм та роль коду, що відповідає за внутрішню логіку протоколу. В залежності від обраного способу представлення протоколів може бути доцільно реалізовувати внутрішню логіку в термінах інтерфейсних станів та сигналів протоколу. За певних обставин внутрішня логіка протоколу може взагалі реалізуватися виключно в термінах публічного інтерфейсу протоколу.

Таким чином, за умови визначення цих частин, питання внутрішньої реалізації протоколу та медіатору теоретично не є принциповим. Втім, метою даного дослідження є знаходження зручних та функціональних формальних способів представлення поведінки протоколів. Тому пропонується звернутися до ряду відомих абстракцій, які б підійшли для представлення поведінковий моделі протоколу та медіатору. Такими абстракціями можуть бути кінцеві автомати. Перевага їх використання для вирішення даної задачі полягає в тому, що по–перше, їх властивості дуже добре досліджені в тому числі в сфері реалізації мережевих протоколів і розподілених систем [7], по–друге, кінцеві автомати різною мірою часто вже використовуються в існуючих текстових специфікаціях протоколів, а по–третє, вони дуже добре дозволяють формалізувати свої моделі в загальному вигляді.

Виходячи із цього, формальне представлення протоколу $P_n$ зводиться до формального представлення кінцевого автомату $A(P_n)$, що описує даний протокол. А отже, формальне представлення повної поведінки протоколу $P_2$ в стеку над протоколом $P_1$ є послідовністю формального представлення автоматів $A(P_2)$ та $A(P_1)$, а також –опису взаємодії між ними $R(M_{21})$.

Питання формального представлення $R(M_{21})$ носить інший характер. Фактично, взаємодію автоматів $P_2$ та $P_1$ можна описати процесом перетворення вхідних кортежів $<t(P_2), t(P_1), [sg(P_2),] [sg(P_1),] [d(P_2),] [d(P_1)]>$ у кортежі $<[sg(P_2),] [sr(P_2),] [sr(P_1)]>$. Дані перетворення є ключовими у визначенні поведінкової логіки протоколів. Завдяки їм автомати, що відповідають за різні рівні протоколів, отримують можливість "співпрацювати": тобто виконувати власні внутрішні переходи на основі сигналів іншого автомату, генерувати сигнали у відповідь на сигнали іншого автомату, передавати дані від і до

іншого автомату у відповідності до описаних форматів даних.

В практичній площині застосування пропонованого підходу вимагає або наявності в системі вже готових базисних реалізацій протоколів, на які б могла спиратися розробка нових протоколів вищого рівня, або вимагатиме реалізацію базисного шару самим розробником. В тому чи іншому випадку в системі існуватиме один або більше протоколів, які з одного боку, будуть реалізовані в умовно вільній формі, а з іншого – матимуть інтерфейс у тому вигляді, у якому він може використовуватись запропонованим декоративним шляхом. До цього шару доцільно віднести протоколи, вже доступні користувачеві системи у традиційному вільному вигляді, реалізувавши "обгортки–адаптери" цих протоколів для використання у системах, створених за запропонованою методикою (Рис. 3).

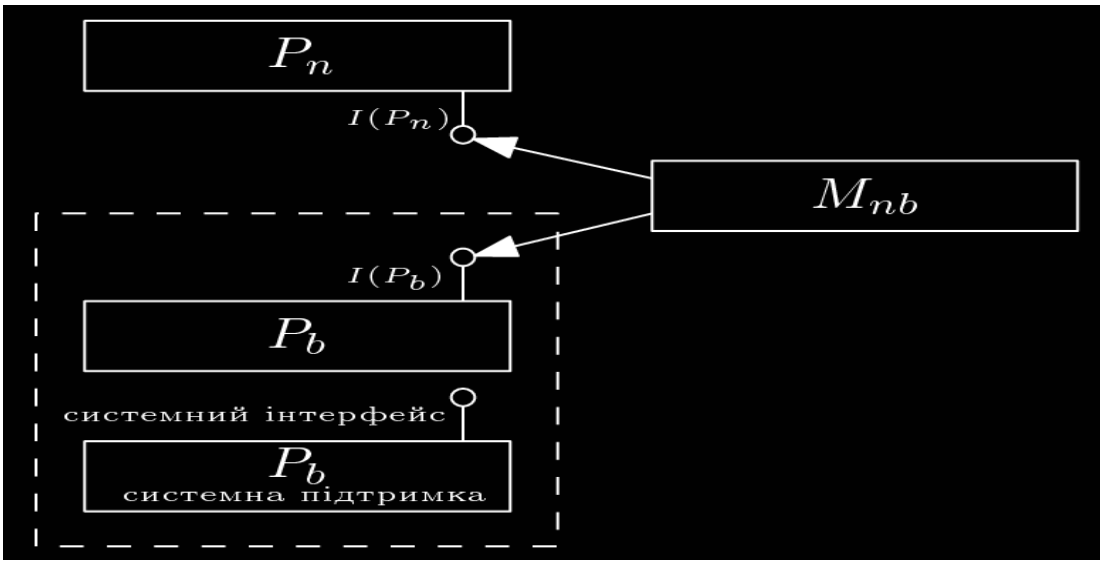



***Рис. 3. Приклад взаємодії протоколу $P_n$ із протоколом $P_b$, що належить базисному шарові***

### Отримані результати

Запропонована методика дозволила на практиці створити формальні описи і перетворити їх у подальші реалізації ряду простих мережевих протоколів, що знаходяться у стеку безпосередньо над протоколом TCP: клієнтську та серверну частину протоколу daytime, спрощений варіант протоколу HTTP, а також клієнтську та серверну частину двох нових протоколів, що знаходяться в стеку один над одним і в свою чергу базуються на протоколі UDP. В якості початкового текстуального опису протоколів для daytime та HTTP використовувались відповідні документи RFC: RFC–867 [4], RFC–2068 [5], для нових протоколів було створено стислі описи правил протоколів у вигляді списку. В якості представлення формального подання описів протоколів використовувались 2 підходи: вільна форма опису кінцевих автоматів із застосуванням таблиць опису поведінкової взаємодії протоколів із подальшим перетворення у програний код, а також безпосереднє представлення кінцевих автоматів та правил їх взаємодії із використанням можливостей мета–програмування мови C++. Окрім того, кожен із цих протоколів було розроблено класичним шляхом на мові C++ із використання BSD Sockets в бібліотеці Boost [3]. У порівняні із класичним підходом значно більший час було витрачено на створення "інфраструктури" TCP, що для способу формального подання протоколів означало опис і реалізацію кінцевого автомату TCP. Однак як тільки вона була створена один раз, більше не було потреби в її модифікації. Для класичного підходу для вирішення цієї задачі безпосередньо використовувались можливості бібліотеки TCP в бібліотеці Boost. Одночасний опис і створення автоматів протоколів із застосуванням можливостей декларативного програмування C++ і програмування логіки цих протоколів засобами класичного програмування зайняли приблизно однаковий час. Використання двоетапного підходу: спочатку опису, а потім його перетворення у реалізацію, зайняло

приблизно в півтори рази більше часу,. Найбільш цікаві результати були отримані при необхідності зміни логіки роботи протоколів (при навмисному внесенні змін в текстуальні описи протоколів): в цьому випадку зміна декларативних описів протоколів на C++ зайняла найменший час, зміна описів автоматів та їх реалізацій була знову приблизно в півтори рази довшою. Зміна класичного програмного коду зайняла найбільше часу – більше ніж у 2 рази довше у порівняння із декларативним C++ підходом.

## Висновки

В результаті проведеного дослідження було, по–перше, доведено доцільність використання методик формального подання поведінкової логіки мережевих протоколів, а по–друге було запропоновано одну з таких методик, можливість використання якої було перевірено практично. Такою методикою було розглянуто спосіб подання протоколів у вигляді незалежних процесів, основаних на кінцевих автоматах із описом взаємодії між ними. Практичні результати показали, що найбільш ефективним з точки зору часових затрат на первинне створення і подальшу підтримку реалізацій протоколів є їх реалізація в декларативному вигляді на кінцевій мові програмування (використовувалась мова C++). Втім, як враховуючи, що далеко не всі популярні мови мають елементи декларативного або функціонального програмування, ефективним також можна вважати підхід, при якому описи елементів протоколу виконуються у певній формальній формі, після чого вони максимально автоматично перетворюються у кінцеві реалізації. Класичне програмування за дотримання високих стандартів створення коду дозволяє найбільш швидко розробляти первинні реалізації відносно нескладних протоколів. Проте при роботі із більш складними протоколами та при необхідності змінювати існуючі реалізації класичний підхід програє підходу формального визначення поведінкової логіки протоколів. Окрім цього, формальний підхід автоматично дозволяє виконувати моніторинг роботи протоколу на рівні не абстрактних об'єктів мови програмування, а реальних складових частин цього протоколу, а отже дозволяє пришвидшити пошук та локалізацію помилок в реалізації протоколу, оцінювати його продуктивність тощо.

Втім, важливим є розробка ефективного та зручного оточення для створення формальних описів (будь–яким способом: безпосереднім чим двоетапним). Таким чином, з великою обачністю має обиратися конкретний спосіб подання кінцевих автоматів: їх використання в будь–якому випадку збільшує архітектурну складність кінцевого коду, а отже впливає на продуктивність системи, що є однією з проблем запропонованої методики. Окрім того, отримані результати залишають відкритими ряд питань для подальшого дослідження і вдосконалення, такі як: вибір ефективної методики реалізації запропонованої моделі в термінах конкретного середовища розробки (окрім дослідженої мови C++); створення оточення та бібліотек для підтримки цих методів; питання оптимальності отриманих реалізацій: наскільки вони можуть бути наближені до реалізацій із використанням класичних підходів; питання трансформації правил протоколів в правила поведінки процесів, а також їх формального подання; питання представлення блоків даних.

Сукупність отриманих результатів і перспективи розв’язання існуючих проблем роблять дану тематику актуальною в сфері сьогоденного мережевого програмування.

## Список посилань

1. Babich F. and Deotto L. Formal methods for specification and analysis of communication protocols // IEEE Communications Surveys & Tutorial. –2002. –Vol. 4, Q3. –P. 2–20.
2. Basu A., Morrisett G. and Von Eicken T. Promela++: a language for constructing correct and efficient protocols // INFOCOM '98. Seventeenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE. –1998. –Vol. 2. –P. 455–462.
3. C++ Бібліотека Boost [електронний ресурс]. – Режим доступу: http://www.boost.org. – Boost C++ Library.
4. Daytime protocol (Request For Comments 867) [електронний ресурс] / Postel J. – Режим доступу: http://www. rfc-editor.org/rfc/rfc867.txt — Request for Comments: 867.
5. Hypertext Transfer Protocol – HTTP/1.1 (Request For Comments 2068) [електронний ресурс] /

Fielding R. – Режим доступу: http://www. rfc-editor.org/rfc/rfc2068.txt – Request for Comments: 2068.
6. Gamma. E., Helm R., Johnson R., Vlassides J.M. Design Patterns: Elements of Reusable Object-Oriented Software / Eddison–Wesley Professional. –1994. –416 p.
7. Killian C.E., Anderson J.W., Braud R., Jhala R. and Vahdat A.M. Mace: language support for building distributed systems // Proceedings of the 2007 ACM SIGPLAN conference on Programming language design and implementation. –San Diego, California, USA, ACM, 2007. –P. 179–188.
8. Kohler E., Kaashoek M.F. and Montgomery D.R. A readable TCP in the Prolac protocol language // Proceedings of the conference on Applications, technologies, architectures, and protocols for computer communication. –Cambridge, Massachusetts, USA; ACM, 1999. –P. 3–13.
9. Madhavapeddy A., Ho A., Deegan T., Scott D. and Sohan R. Melange: creating a "functional" internet // Proceedings of the 2nd ACM SIGOPS / EuroSys European Conference on Computer Systems. – 2007.
10. Vuong S., Lau, C. and Chan R.I. Semiautomatic Implementation of Protocols Using an Estelle-C Compiler // Software Engineering, IEEE Transactions on. –1988. –Vol. 14. –P. 384–393.
11. Warth A. and Piumarta I. OMeta: an object-oriented language for pattern matching // Proceedings of the 2007 symposium on Dynamic languages. –Montreal, Quebec, Canada; ACM, 2007. –P. 11–19.

Поступила в редакцию 14.12.2009

УДК 004.75

*КОШЕЛЬ М.О.,*
*СИМОНЕНКО А.В.*

# БАЛАНСУВАННЯ ЗАВАНТАЖЕННЯ ВЕБ-СЕРВІСІВ, З ВИКОРИСТАННЯМ МОБІЛЬНИХ АГЕНТІВ

В даній роботі описані методи балансування завантаження розподілених веб-сервісів. Розглянуті переваги та недоліки таких методів, та запропонований метод з використанням технології мобільних агентів.

In this work methods of load balancing of the distributed web-services are described. All advantages and lacks of these methods are considered and the optimal method with use of mobile agents is offered.

## Вступ

На даний час веб-сервісам знаходять все більш широке застосування. Вони використовуються в різних випадках і ситуаціях в Інтернеті [1, 2, 3, 4]. Але швидке збільшення кількості веб-сервісів і користувачів цих сервісів потребує підвищення продуктивності веб-серверів, для зменшення часу відгуку на запити, які, в свою чергу можуть з'явитися в будь-який момент часу. При цьому необхідно забезпечити ще й надійність таких веб-серверів.

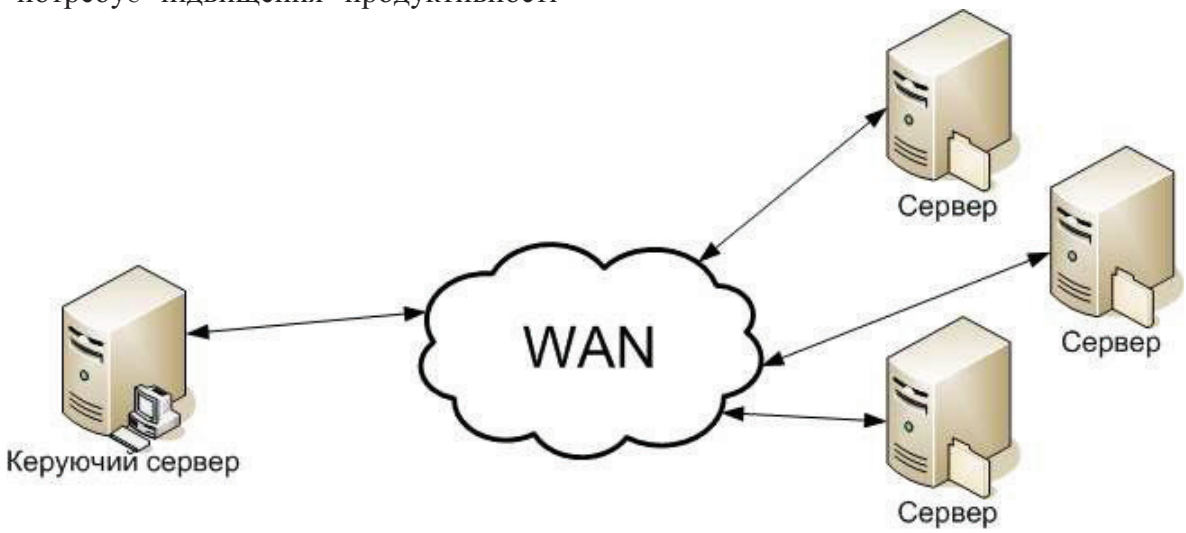



***Рис.1. Структура розподілених веб-сервісів***

Для реалізації високопродуктивних і надійних веб-серверів використовують розподілені веб-сервери. Розподілені веб-сервери представляють собою набір (N-ну кількість серверів) веб-серверів. Це продубльовані ресурси для одночасного надання послуг багатьом користувачам. Вхідні запити можуть бути розподілені між серверами згідно певних стратегій розподілення завантаження, і тому ці запити можуть бути опрацьовані в певних часових межах (час відгуку). Розподілені веб-сервери (мал. 1) можуть бути організовані різними способами:

- вони можуть бути інтегровані в кластер веб-серверів, з'єднаних через локальну обчислювальну мережу, щоб працювати як один потужний сервер;
- вони можуть використовуватись в різних географічних місцях через глобальну обчислювальну мережу;

Розподілені веб-сервери можуть легко розширюватися, та мають високу степінь масштабованості. Кількість їх може бути збільшено простим додаванням нового сервера в локальну мережу.

Для забезпечення доброї масштабованості високопродуктивних веб-серверів потрібно проведення балансування завантаження усіх веб-серверів. Вхідні запити від користувачів повинні бути розподілені відповідно до стратегії навантаження між серверами, щоб ко-

ристувач отримував відповідь на запит в певних часових межах. Як і у випадках з розподіленими обчисленнями та розподіленим моделюванням, роботу з перевантажених серверів необхідно перемістити на не завантажені, що сприяє підвищенню пропускної здатності системи. В іншому випадку може бути ситуація, коли запит користувача буде знаходитись в черзі безкінечно довго. В таких випадках сервер може відхилити запит користувача. В цій роботі описані методи, що допоможуть уникнути таких ситуацій.

**Методи балансування завантаження, що базуються на технології клієнт-сервер**

Виділяють наступні категорії методів балансування навантаження [1,2]:

- Клієнтські
- Основані на DNS
- Диспетчерські
- Серверні


***Рис.2. Клієнтський підхід балансування завантаження***

Клієнтський підхід (мал. 2) реалізує вибір сервера на стороні клієнта. Клієнти можуть вибрати один з доступних веб-серверів випадковим чином, або вибрати сервер, що найбільш їм підходить, використовуючи механізми інтелектуального вибору.


***Рис. 3. Підхід, що базується на DNS***

Наприклад, браузер Netscape Navigator, використовує клієнтський підхід для доступу до своїх сайтів. Коли користувач переглядає домашню сторінку Netscape, браузер випадковим чином вибирає один із серверів і направляє йому запит користувача. Однак випадковий вибір не може гарантувати балансування навантаження всіх серверів та доступність вибраного сервера. Інтелектуальний вибір сервера може бути реалізований з використанням Java апплетів, що запущені на стороні клієнт для визначення стану серверів і затримок мережі. В такому випадку може бути вибраний сервер, що найбільше підходить, і запит користувача буде направлений саме йому. *Недолік такого методу полягає в великій часовій затримці, що викликана визначенням станів серверів.*

Підхід с використанням DNS – це прийняття рішення на стороні DNS-сервера, що обробляє запити по трансляції імен (мал. 3). Щоб перетворити ім’я, клієнт відправляє запит на DNS-сервер для перетворення імені веб-сервіса на адресу веб-сервера. Він, згідно

з стратегією балансування, вибирає адресу сервера, та відправляє його клієнту. Найпростіша стратегія – RoundRobin, а більш складна включає моніторинг завантаження, облік адміністративних особливостей мережі і т.д.

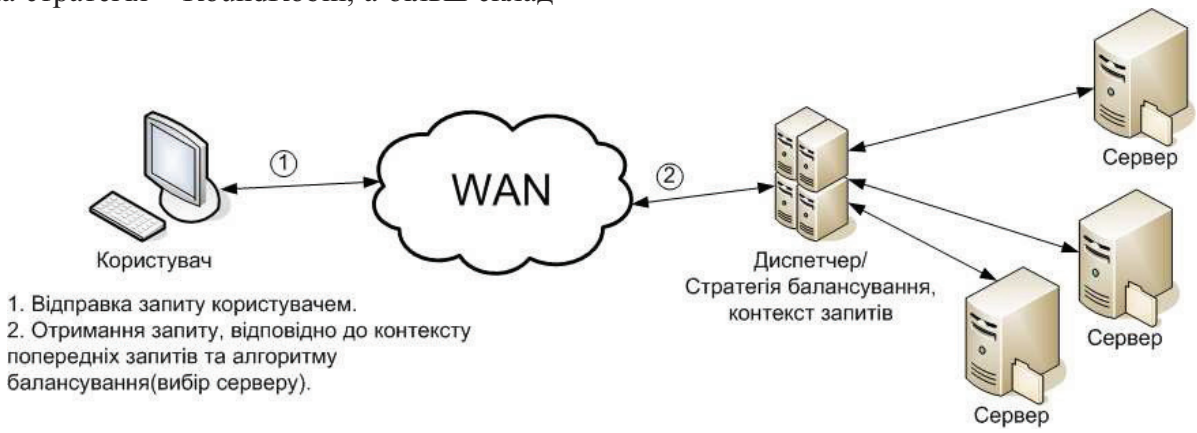



***Рис. 4. Диспетчерський підхід балансування завантаження***

Однак, DNS-сервер може стати вузьким місцем в процесі маршрутизації. Існують програмні продукти для розподілення навантаження між багатьма географічно розподіленими серверами, в яких DNS-сервер визначає і доступність серверів, і часову затримку в мережі для вибору найбільш оптимального сервера, що використовує технологію клієнт-сервер.

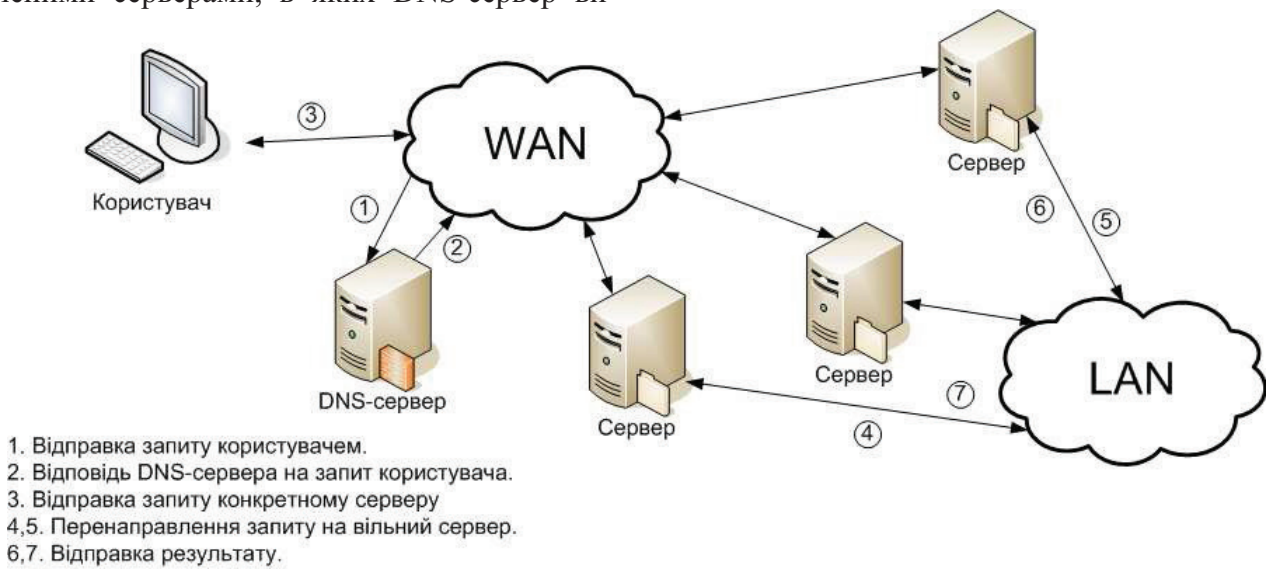



***Рис. 5. Серверний підхід балансування завантаження***

Диспетчерський підхід (мал. 4) виконує відображення адрес на рівні IP. Це потрібно для того, щоб отримати повний контроль над маршрутизацією клієнтських запитів. Він визначає групу веб-серверів, що має однотипну віртуальну IP адресу, яка є IP адресою диспетчера. Диспетчер в свою чергу діє в якості центрального планувальника і має повний контроль над маршрутизацією запитів. Диспетчер ідентифікує кожний сервер з частковою адресою і, якщо потрібно, то перенаправляє клієнт-серверний пакет, переписавши IP адресу. В основі цього підходу лежить механізм на основі TCP-маршрутизатора. TCP-маршрутизатор в цьому підході, виконує роль диспетчера. Основним недоліком цього підходу є накладні витрати на зміну адреси. Однак існує інший спосіб перенаправлення пакетів – перенаправлення пакетів на рівні HTTP. Тут диспетчер розподіляє вхідні запити між серверами через механізми перенаправлення, що надає протокол HTTP. Диспетчер може перенаправити запит через відповідь клієнта, при цьому в заголовку відповіді вказана адреса сервера. В цьому випадку, клієнт перенаправляє запит, але вже напряму на сервер.

Серверний підхід (мал. 5) використовує механізм дворівневої диспетчеризації. DNS-сервер спочатку, визначає сервер, на який йде клієнтський запит. Після цього, кожен сервер може перенаправити запит на інший сервер. Це децентралізована стратегія перерозподілення завантаження, в якій всім сер-

верам дозволена участь в процесі балансування завантаження. Існує реалізація такого підходу на основі веб-сервера Apache.

***Таблиця 1. Переваги та недоліки методів балансування завантаження, що використовують технологію клієнт-сервер.***

| Підходи | Переваги | Недоліки |
|---|---|---|
| Клієнтський | Простота розроблення програмного забезпечення. | Велика часова затримка, викликана визначенням станів серверів. |
| DNS | | Велика часова затримка, викликана великою чергою запитів на DNS-сервері, коли в один момент часу приходить велика кількість запитів. |
| Диспетчерський | | Збільшення накладних витрат на зміну адреси. |
| Серверний | | Велика часова затримка, при пере направленні сервером запитів, на інший. Може виникнути безкінечний цикл. |

Ця реалізація дозволяє виконати однорідне пере розподілення HTTP запитів від серверів до мало використовуваних. Запит передається по ланцюжку функцій прийняття рішень, що називаються функціями-кандидатами. Кожна функція-кандидат пере впорядковує набір доступних серверів і вибирає метод пере розподілення запиту (HTTP-пере направлення або через HTTP-проксі), базуючись на інформації про ресурси.

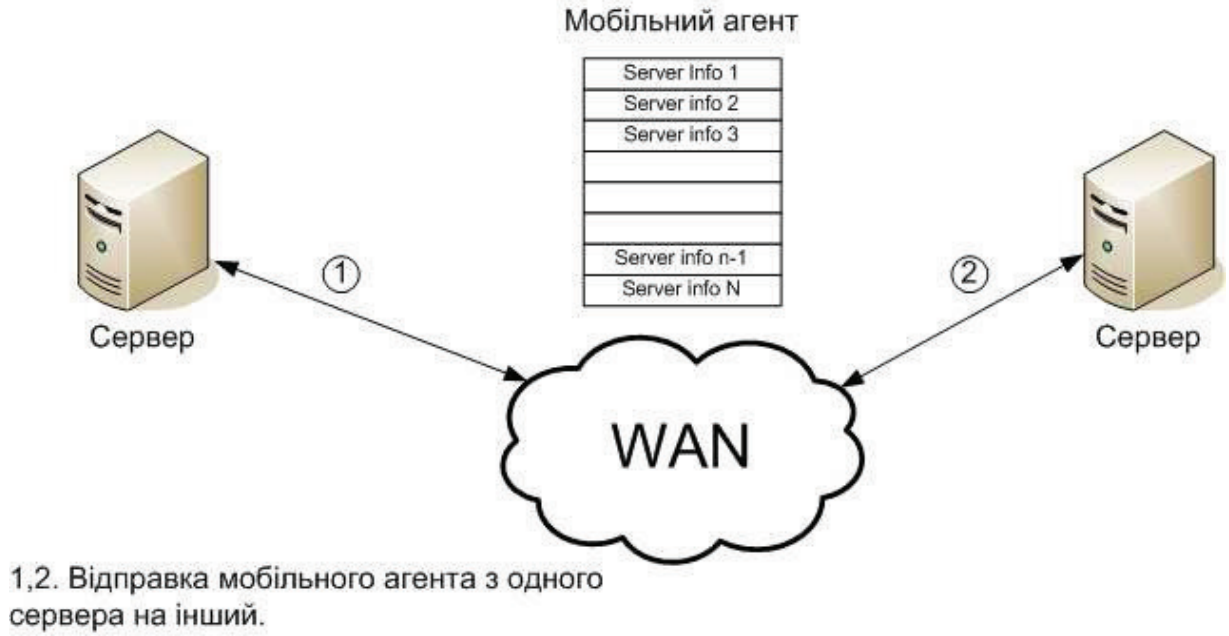



***Рис. 6. Балансування завантаження з використанням мобільних агентів***

В таблиці 1 представлені переваги та недоліки методів балансування завантаження, що базуються на технології клієнт сервер. Запропонований в статті метод балансування навантаження, з використанням технології мобільних агентів не має вказаних недоліків.

### Метод балансування завантаження з використанням мобільних агентів

Мобільних агент – це програмний компонент, який може автоматично пересуватись з одного вузла мережі на інший, разом зі своїм станом і виконавчим кодом і виконувати різноманітні операції на цих вузлах.

Агенти можуть розділити функціональність при проектуванні веб-систем. В традиційному підході балансування завантаження, який базується на відправці повідомлень, модулі серверних сервісів змішують основну функціональність веб-сервіса з підтримкою таких функцій, як балансування завантаження. Коли вводиться нова політика балансування завантаження, може знадобитись переписати серверний модуль. З іншої сторони

при використанні мобільних агентів функції підтримки можуть бути відділені від сервісних модулів і реалізовані порізно в мобільних агентах. А це свідчить про те, що спосіб балансування завантаження, що базується на мобільних агентах, являється більш гнучким при підключенні нових політик балансування завантаження для різноманітних веб-серверних систем.

Агенти використовують мало трафіку. В методах, які базуються на передачі повідомлень, веб-сервери повинні періодично обмінюватись повідомлення з інформацією про завантаження серверів, для того щоб приймати рішення балансуванню завантаженням. Обмін повідомленнями хоча і знижує пропускну здатність веб-системи, але передається лише службова інформація, тому це незначно впливає на продуктивність мережі. Взаємодія на місці використання усуває прямий обмін повідомленнями, що відбувається між серверами в обох напрямках. З цього слідує, що можуть бути значно зменшені мережевий трафік і затрати на комунікацію.

Мобільні агенти підтримують асинхронні і автономні операції. Сервери можуть запускати мобільні агенти, які незалежно мандрують між серверами (мал. 6), і виконують різного роду операції. Мобільний агент може виконувати алгоритми, що містять різноманітні стратегії балансування завантажень і приймають рішення по розподіленню завантаження на льоту, згідно поточному стану сервера.

Використання мобільних агентів підвищує надійність веб-систем, так як вони можуть переносити запроси клієнтів від несправних серверів до активних(ті що працюють).

## Список посилань

1. Таненбаум Е. Комп'ютерні мережі. –СПб.:Пітер. – 2002.
2. Таненбаум Е. Сучасні комп'ютерні системи. –СПб.:Пітер. – 2008.
3. Таненбаум Е., Ван Стеєн М. Розподілені системи. – СПб.:Пітер, 2003.
4. Coulouris G., Dollimore J., Kindberg T. Distributed Systems. Concepts and Design. - Addison-Wesley Publishing Company, 1995.

Поступила в редакцию 18.12.2009

УДК 004.04 (043.3)

*КУЛАКОВ Ю.А.,*
*ДЕРЕВЯНЧУК А.О.*

# БЕЗОПАСНАЯ ПЕРЕДАЧА ИНФОРМАЦИИ НА ОСНОВЕ МНОГОПУТЕВОЙ МАРШРУТИЗАЦИИ

В данной работе приведен краткий обзор протоколов безопасной маршрутизации для мобильных сетей, а так же приведено новое решение научной задачи, которая использует преимущества распределенных беспроводных сетей вместе с секретным разделением сообщения и применением многопутевой маршрутизации.

In this paper we present a survey of secure routing protocols for mobile wireless network and give a new solution to a scientific problem which is to take advantage of the distributed nature of wireless networks and combine the secret sharing scheme and multipath routing.

## 1. Введение

Многопутевая маршрутизация является одним из наиболее важных направлений в области маршрутизации. Данная маршрутизация основана на однопутевой маршрутизации между узлом источником и узлом назначения, где с большой вероятностью выбирается путь с минимальность стоимостью, хотя по различным ценовым показателям могут быть различные пути. Таким образом, в хорошо связанной сети может существовать несколько путей между парой узлов источника и назначения. Смысл многопутевой маршрутизации состоит в том, чтобы предоставить узлу источнику выбор одного из нескольких маршрутов в любое время к конкретному узлу назначения, используя преимущество избыточной связности а основной сети. Несколько путей могут использоваться как поочередно (трафик проходит по одному из путей в одно время), так и одновременно (трафик проходит одновременно по нескольким путям).

Мобильные ad hoc сети (MANETs) в последнее время привлекли много внимания. MANET – это набор узлов, которые могут свободно перемещаться и сообщаться друг с другом используя беспроводные устройства. MANET характеризуется динамическими изменениями топологии, ограниченной пропускной способностью, а также ограниченной мощностью батарей (аккумуляторов) в узлах. В последнее время многопутевая маршрутизация широко используется в сетях MANETs. За счет плотности расположения узлов в мобильной сети многопутевая маршрутизация используется как естественная и перспективная технология решения проблемы частых изменений топологии. Так же многопутевая маршрутизация используется в целях повышения надежности доставки данных, балансировки нагрузки трафика, балансировки потребления мощности между узлами, уменьшения сквозных задержек, повышения частоты нахождения маршрутов, а так же для повышения безопасности сети.

Безопасность и надежность связи являются двумя важными аспектами в любой сети, которые с первого взгляда противоречат использованию дополнительной избыточности. С одной стороны, надежность может быть достигнута за счет отправления избыточной информации по нескольким путям. С другой стороны, избыточная информация дает противнику больше шансов для перехвата информации. Для решения этой задачи предложен новый протокол безопасности по надежной доставки данных (SPREAD, Security Protocol for REliable dAta Delivery), который повышает как безопасность, так и надежность. Данный протокол был исследован в мобильных ad hoc сетях. Цель предлагаемого протокола состоит в обеспечении дополнительной защиты передаваемых данных, в частности, в снижении вероятности того, что секретное сообщение будет утеряно в то время, как оно будет передаваться по ненадежной сети. Основная идея состоит в разделении секретного сообщения на несколько частей по секретной схеме разделения и в последующей отправки этих частей по нескольким независимым путям к источнику. Таким образом, если

даже какое-то небольшое количество узлов, используемых для передачи сообщения, будет скомпрометировано, то секретное сообщение в целом не будет рассекречено.

Целью статьи является ознакомление с концепцией и технологией многопутевой маршрутизации в ad hoc сетях и получением значительной выгоды при ее использовании.

## 2. Надежность

Надежность – вероятность того, что сообщение, сгенерированное в одном месте сети будет доставлено в узел назначения. Надежность является сложной задачей в сетях MANETs/WSNs из-за того, что есть большая вероятность потери пакетов в связи с частыми изменениями топологии, различными помехами, которые влияют на корректность кодирования беспроводных сигналов в беспроводных трансиверах.

Многопутевая маршрутизация в MANET была изначально разработана как средство обеспечения защиты маршрута от сбоев. Например, протокол динамической маршрутизации от источника (Dynamic Source Routing – DSR) [1] может запоминать несколько маршрутов к определенному узлу назначения. Когда возникают проблемы на основном маршруте, будут использоваться альтернативные маршруты для того, чтобы доставить пакет к узлу назначения. Так же было предложено "многопутевое" расширение некоторых протоколов, которые первоначально использовали для маршрутизации один путь, как, например AODV-BR, APR (Alternative Path Routing) и SMR (Split Multipath Routing), что значительно улучшило однопутевые протоколы за счет предложения альтернативных путей. В этом случае, разные маршруты используются не одновременно. Трафик проходит по одному из путей. Остальные маршруты хранятся в качестве резервных для случая, если на используемом маршруте возникнут проблемы. Когда все известные маршруты сталкиваются с проблемами, запускается новая процедура поиска маршрутов. Выбор маршрута осуществляется на канальном уровне, когда доступно несколько последующих хопов, и пакет посылается по маршруту с наилучшим состоянием канала [2].

Другой способ использования многопутевой маршрутизации – одновременно отправлять поток данных по нескольким путям. Параллельно многопутевая маршрутизация в MANET была разработана для повышения пропускной способности, надежности и балансировки нагрузки.

В [3] и [4] за счет объедения пороговой схемы разделения секрета и многопутевой маршрутизации мы предложили новый протокол – SPREAD, который обеспечивает более высокую надежность и более безопасную доставку данных в MANET. Алгоритм порогового (T;N) разделения секрета делит и всю информацию на N сегментов. Исходную информацию можно восстановить из любого T, состоящего из N сегментов. Кроме того, схема разделения секрета является достаточно безопасной, то есть, менее чем из T частей, невозможно узнать никакой информации, тем более восстановить первоначальное сообщение. Таким образом, в ad hoc сетях повысилась надежность доставки, так как при использовании протокола SPREAD допускается потеря определенного количества пакетов или путей, в то время как надежность каждого пути в целом не повышается.

## 3. Качество обслуживания

Одной из основных целей многопутевой маршрутизации является качество обслуживания, а именно, уменьшение задержек, избежание или снижение перегрузок, а так же повышение пропускной способности. Было доказано, что многопутевая маршрутизация повышает качество обслуживания за счет уменьшения задержек при доставки пакетов. На уменьшение задержек влияет несколько факторов. Задержка – это время, за которое пакет проходит от узла источника к узлу назначения. Кроме обычной задержкой передачи, задержкой распространения и задержкой в очереди, которые присутствуют во всех IP сетях, еще существует 2 типа задержек, которые встречаются только в ad hoc протоколах маршрутизации по требованию. Первый вид задержек – это время поиска первого маршрута к узлу назначения. Многопутевая маршрутизация существенно снижает время поиска маршрута, поэтому время такой задержки снижается. Второй вид задержек связан с восстановлением маршрута в случае нарушений в маршруте. В этом случае задержка – это

сумма трех задержек (время, которое потратил пакет при прохождении маршрута до обнаружения поломок; время обнаружения поломки; время, затраченное на прохождение сообщения об ошибке к узлу источнику). Многопутевая маршрутизация помогает избежать или уменьшить возникновение ошибок в маршруте, следовательно, снижается и этот вид задержек.

### 4. Безопасность

Для повышения безопасности сети предложено несколько способов на основе многопутевой маршрутизации. Многопутевая маршрутизация в данном случае часто применяется с шифрованием секрета. При пороговом разделении (T, N) секрет делится на N частей, которые называются часть (share) или тень (shadow). Секрет может быть восстановлен не менее чем из T частей (share). Таким образом, при объединении многопутевой маршрутизации с методом порогового разделения секрета происходит следующее – разделение секрета и доставка его частей при использовании многопутевой маршрутизации. Тем самым, при распределении по нескольким путям сети, система становится более устойчивой к атакам противника.

Разделение секрета было предложено для управления ключами в системе информационной безопасности. Управление ключами является сложной задачей, когда мы говорим о безопасности в MANET или WSN. Возможность использования других сервисов безопасности, как, например, конфиденциальность и аутентификация, зависит от эффективного и рационального управления ключами. В [5] автора используют репликацию и пороговое шифрование и, таким образом, они смогли построить более защищенную и более доступную систему управления ключами для отклонения атак в MANET. Идея состоит в том, чтобы распределить функции центра сертификации по управлению ключами на несколько серверов (доверенных узлов). Таким образом повышается безопасность центра сертификации. В предложенной модели пороговое шифрование используется для разделения системного секрета на части, каждая часть будет удерживаться на своем сервере; сервера вместе выполняют такие функции, как подписание сертификата и обновление распределенных ключей. Многопутевая маршрутизации подразумевает под собой маршрутизацию частей от нескольких серверов к одному объединителю (сумматору). Данный подход был исследован в [6], где центры сертификации локализированы путем распределения серверов по сети таким образом, что совместные криптографические операции могли бы выполняться соседями запрашивающих узлов. Другой подход к управлению ключами на основе многопутевой маршрутизации – это вероятностный подход с установлением парных секретных ключей [7, 8]. Из-за вычислительной сложности вычисления, основанные на алгоритмах открытого ключа, являются достаточно дорогими для сетей MANET/WSN. Проектирование данного подхода основывается на вероятностном разделении ключа и на методе порогового разделения секрета. При вероятностном разделении ключа, каждый узел сети будет предварительно содержать начальный ключ. С большой вероятностью, любая пара узлов будет обладать общими ключами. Затем, используя общие ключи в качестве начальных, узел источник генерирует новый секретный ключ и разбивает его на несколько частей используя секретную схему разделения. После этого части секретного ключа доставляются к узлу назначения при использовании многопутевой маршрутизации. Многопутевая маршрутизация в такой схеме логична – части ключа могут проходить по одинаковым физическим каналам, но при этом каждый будет зашифрован различными открытыми ключами. Схема SPREAD аналогична подходам описанным выше – схема базируется на объединение многопутевой маршрутизации и разделении секрета. Однако, схема SPREAD направлена на защиту доставляемого по незащищенной сети потока данных, предполагая, что сквозное шифрование является небезопасным и ненадежным. Многопутевая маршрутизация в схеме SPREAD использует физически разделенные пути и предлагает алгоритм поиска заданного пути. Схема SPREAD может быть использована для доставки ключей вместо потока данных. Тем не менее, при доставке потока данных, схема SPREAD повышает не только безопасность, но и надежность, что

является большой проблемой в сетях MANETs/WSNs.

## 5. Протокол SPREAD

Впервые схема SPREAD была предложена в [8], а затем была изучена в качестве дополнительного механизма повышения безопасности доставки данных в MANET в [3]. Основная идея и работа SPREAD представлена на рис. 1.

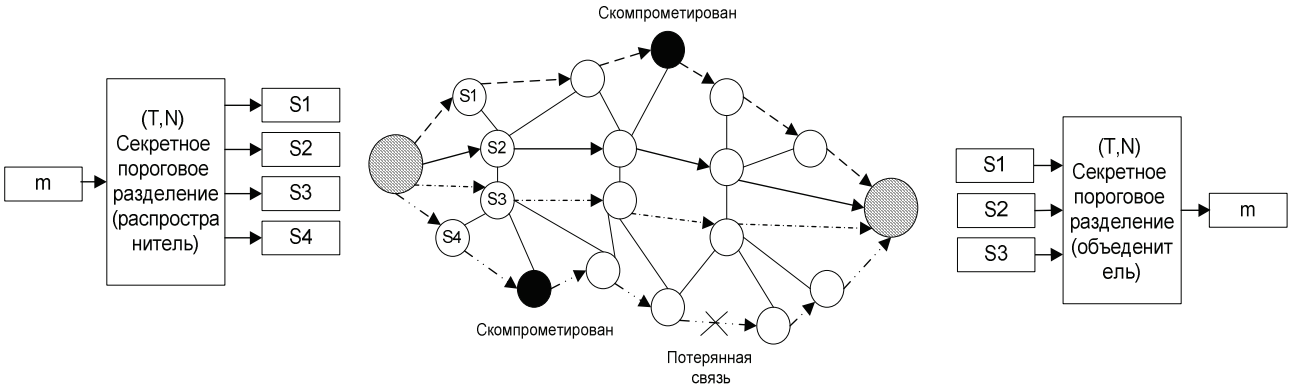



***Рис. 1 Принцип работы SPREAD***

Секретное сообщение m разделяется на несколько частей – S1, S2,..., по секретной схеме разделения, а затем, доставляются к узлу назначения по разным независимым путям.

Из-за характерных особенностей секретного разделения и доставки сообщения по разным путям, алгоритм SPREAD показал хорошую устойчивость при отказах узлов, а именно, даже если отказало небольшое количество узлов, частей сообщения или путей, целостность не нарушалась. В целях повышения надежности могут использоваться различные схемы кодирования для разделения трафика при многопутевой маршрутизации. Примерами могут служить код Рида-Соломона, разнесенный код и др. В нашей схеме SPREAD мы использовали пороговую секретную схему для обеспечения дополнительно безопасности.

Секретная схема (T,N) разделения делит секретное сообщение на N частей, называемых частями. Особенность такого разделения состоит в том, что меньше чем из T частей невозможно что-либо узнать о сообщении. В то же время при использовании соответствующего алгоритма можно восстановить секретное сообщение из любого числа *T* (или больше) частей. Затем, используя многопутевую маршрутизацию, части сообщения доставляются к адресату по N различным путям, где у *T* путей нет пересекающихся узлов. Процесс разделения на части очень простой – вычисление многочлена степени *(T-1)*.

$$f(x) = \left(a_0 + a_1 x + \ldots + a_{T-1} x^{T-1}\right) \bmod p$$

в точке $x = i$ получаем *i*–ю часть:

$$S(i) = f(i)$$

где $a_0, a_1, a_2, \ldots, a_{T-1}$ – части секретного сообщения, а $p$ – большое простое число, которое больше любого из коэффициентов и которое является открытым.

Согласно основной теореме алгебры, при известный *T* значениях многочлена степени *(T - 1)* можно решить многочлен (т.е. определить все его коэффициенты). Таким образом, зная значение любых T частей, можно восстановить секретное сообщение. Эффективность $\left(O\left(T \log^2 T\right)\right)$ алгоритма была исследована для определения многочлена и интерполяции [7]. Кроме того, в зависимости от количества доступных путей, значение (T,N) в SPREAD будет небольшим. Для практического применения достаточно даже обычных квадратичных алгоритмов.

Многопутевая маршрутизация является перспективной технологией в мобильных сетях с ненадежной передачей данных. Такая маршрутизация хорошо применима из-за плотного расположения узлов в сетях MANETs. SPREAD – это новая схема, которая включает в себя многопутевую маршрутизацию и технологию разделения сообщения с целью обеспечения надежности

и безопасности. Применение SPREAD схемы позволяет одновременную маршрутизацию по нескольким путям и обеспечивает более безопасную передачу данных при прохождении данных по незащищенной сети. Избыточность не влияет на безопасность за счет оптимального расположения частей на каждом выбранном пути.

## 6. Повышение эффективности алгоритма SPREAD

Алгоритм SPREAD будет работать эффективно в том случае, когда получает не меньше чем *T* частей сообщения. Для уменьшения вероятности потери пакетов могут использоваться дополнительные буферы в каждом из узлов. При таком подходе необходимо, чтобы в каждом узле поддерживалось по 2 небольших буфера – один для маршрутов, а второй – для сообщений.

Кэш маршрутов содержит несколько маршрутов к активному адресату, а кэш сообщений содержит последний отправленный пакет. Таким образом, в случае отказа канала связи или узла по основному маршруту и при наличии в узле альтернативного маршрута, сообщение может быть передано повторно.

## 7. Выводы

Многопутевая маршрутизация – перспективный метод в сетях MANET. Преимущества при использовании многопутевой маршрутизации: повышение отказоустойчивости, безопасности, надежности, эффективности, уменьшение потерь при маршрутизации, балансировка нагрузки по потоку сообщений и потреблению энергии, снижение задержек.

## Список литературы


1. D. B. Johnson, D. A. Maltz, Y-C. Hu, J. G. Jetcheva, “The dynamic source routing protocol for mobile ad hoc networks”, IETF Internet Draft, draft-ietfmanet-dsr-06.txt, Nov 2001
2. S. Jain, Yi Lv, S. R. Das, “Exploiting path diversity in the link layer in wireless ad hoc networks,” Technical Report, SUNY at Stony Brook, CS department, WINGS Lab, July 2003.
3. W. Lou, W. Liu, Y. Fang, “SPREAD: Enhancing data confidentiality in mobile ad hoc networks”, IEEE INFOCOM 2004, HongKong, China, March 2004
4. W. Lou, Y. Zhang, W. Liu, Y. Fang, “A multipath protocol for secure and reliable data collection in wireless sensor networks”, technical report, ECE department, Worcester Polytechnic Institute, June 2004
5. J. Kong, P. Zerfos, H. Luo, S. Lu and L. Zhang, “Providing robust and ubiquitous security support for manet,” Proceedings of the 9th IEEE International Conference on Network Protocols(ICNP), pp. 251 - 260, 2001.
6. H. Chan, A. Perrig, D. Song, “Randon key predistribution schemes for sensor networks”, IEEE Symposium on Security and Privacy (SP’03), Oakland, CA, May 2003
7. S. Zhu, S. Xu, S. Setia, S. Jajodia, “Establishing pairwise keys for secure communication in ad hoc networks: A probabilistic approach”, 11th IEEE International Conference on Network Protocols (ICNP’03), Atlanta, GA, November 2003
8. W. Lou, Y. Fang, “A multipath routing approach for secure data delivery”, IEEE Military Communications Conference (MILCOM 2001), Mclean, VA, USA, Oct 2001


Поступила в редакцию 4.12.2009

УДК 004.04 (043.3)

*КУЛАКОВ О.Ю.,*
*БРОЛІНСЬКИЙ С.М.,*
*АШАЄВ Ю.М.*

# ДИНАМІЧНЕ СТВОРЕННЯ ВІРТУАЛЬНИХ GRID СИСТЕМ ДЛЯ ВИРІШЕННЯ РОЗПОДІЛЕНИХ ЗАДАЧ НА ОСНОВІ МЕНЕДЖЕРА РЕСУРСІВ

У даній статті розглядається метод застосування віртуальних мереж для Grid систем з динамічним створенням, модифікацією та знищенням віртуальних мереж. Даний метод дозволяє об'єднати кілька віддалених Grid систем чи їх частин в одну віртуальну Grid систему чи розбити локальну Grid систему на кілька віртуальних, що дозволяє ефективно використовувати ресурси Grid системи, виділяючи для задачі необхідну для її вирішення кількість ресурсів. За створення, модифікацію і видалення віртуальної мережі відповідальний VDHCP сервер. Створення черги задач, що очікують на вирішення, а також визначення кількості необхідних ресурсів – одні з основних задач VDHCP сервера, для вирішення яких він, якщо це можливо, використовує попередній досвід.

In this article the method of application of virtual networks for Grid systems with dynamic creation, modification and destruction of virtual networks is overviewed. This method allows integration of several remote Grid systems or their parts in a single virtual Grid system or split a local Grid into several virtual system that allows efficient use of Grid resources, emphasizing the need amount of resources for the problem. For creating, modifying and deleting virtual network is that, what VDHCP server is responsible for. Create a queue of tasks waiting on the decision and determine the number of required resources – some of the main tasks of VDHCP server, for which it, if possible, using previous experience.

## 1. Вступ

Основна задача Grid обчислень – розподіл ресурсів мережі між користувачами. В багатьох випадках необхідним є обмін даними між учасниками Grid системи, який нерідко може бути ускладнений високим ступенем асиметричності каналів зв'язку між ними [2]. Часто зв'язок між процесами A та B може бути встановлено тільки у тому випадку, якщо його ініціює процес A. Розв'язати задачу симетричності каналів зв'язку покликані віртуальні мережі. Основними двома технологіями, які сприяють асиметричності є приватні мережі та фаєрволи. Через це часто комп'ютери стають недосяжними з мережі Інтернет. Водночас вони досяжні з комп'ютерів приватної мережі, деякі з яких можуть бути досяжними з мережі Інтернет. Якщо об'єднати ці комп'ютери в одну віртуальну мережу, можна зробити досяжними з мережі Інтернет будь-які з них.[1].

При використанні Grid систем важливо правильно розподіляти ресурси між завданнями, щоб вони не простоювали. Вирішити цю проблему покликаний VDHCP сервер, який, використовуючи алгоритм планування завдань і свій минулий досвід, займається створенням віртуальних Grid систем і наданням їх ресурсів користувачеві.

## 2. Постановка задачі

Існує декілька комп'ютерів (10-15), наприклад у малому офісі, ресурси кожного окремо взятого комп'ютера повною мірою не використовуються, навантаження на локальну мережу невелике. Необхідно забезпечити користувачеві можливість використання ресурсів мережі для вирішення складних задач без потреби налаштування складного ПЗ. При цьому необхідно враховувати минулий досвід вирішення даного типу задач і подолати проблему асиметричності каналів зв'язку [1].

## 3. Існуючі рішення

Globus Toolkit – інструментарій, розроблений американськими вченими, який став де-факто світовим стандартом. Він включає в себе, зокрема, спеціальний протокол на основі HTTP для використання обчислювальних ресурсів GRAM (Grid Resource Allocation Management); розширену версію протоколу для передачі файлів GridFTP; службу безпеки GSI (Grid Security Infrastructure); розподілений доступ до інформації на основі протоколу LDAP ; віддалений доступ до даних через інтерфейс GASS (Globus Access to Secondary Storage) [3]. Во-

лодіє високою надійністю і швидкодією, але не вирішує проблему асиметричності каналів зв'язку і не враховує минулий досвід.

gLite – наступне покоління middleware програмного забезпечення для систем Grid. gLite істотно спирається на досвід низки великих європейських проектів: EDG, LCG, Alien, Nordugrid і створюється колективно – в його розробці беруть участь більше 80 фахівців з 11 дослідницьких центрів. gLite повинен стати основним ГПЗ проекту EGEE, прийшовши на зміну комплексу LCG-2 [3]. Основна відмінність комплексу gLite в тому, що крім інструментальних засобів, до нього входить ширший набір служб, але також не вирішує проблему асиметричності каналів зв'язку і не враховує минулий досвід.

Обидва рішення є досить великими і складними в обслуговуванні, що значно ускладнює їх використання для вирішення завдань, наприклад, в мережі малого офісу. Для вирішення даної проблеми нами була розроблена система балансування навантаження на мережеву обчислювальну систему з застосуванням віртуалізації каналів зв'язку та урахуванням попереднього досвіду (VDHCP). До її переваг можна віднести:

- Відсутність необхідності в адмініструванні.
- Вирішення проблеми асиметричності каналів зв'язку за рахунок використання віртуалізації даних каналів [1].
- Використання планувальника завдань, який ґрунтується на минулому досвіді

## 4. Застосування

Часто задачі, які вирішуються комп'ютерами малого офісу досить однотипні і вимагають для свого рішення приблизно однакову кількість ресурсів, тому доцільно використовувати для планування завдань планувальник, який враховує досвід вирішення задач даного типу. У той же час в малому офісі ресурси кожного окремо взятого комп'ютера рідко бувають повністю задіяні, а утримання системного адміністратора, для забезпечення роботи мережевої обчислювальної системи, занадто витратно, тому в якості можливого застосування можна привести приклад використання даної технології в мережі малого офісу, коли при використанні високошвидкісних каналів зв'язку між комп'ютерами, кожен окремо взятий комп'ютер не має достатнього кількість ресурсів для вирішення завдання, або час виконання завдання занадто великий. Застосувавши дану технологію стане можливо використовувати незадіяні ресурси на комп'ютерах в мережі для більш швидкого або якісного вирішення завдання. Також стане можливо об'єднати декілька віддалених малих офісів в єдину віртуальну мережу, або навпаки розбити одну фізичну мережу на кілька віртуальних для якіснішого використання ресурсів мережі.

## 5. Фізична топологія мережі

В якості фізичної мережі для створення віртуальної мережі може виступати будь-яка топологія мережі, наприклад така як на рис. 1. Деякі комп'ютери можуть бути недоступні через використання NAT або фаєрвола, але дана проблеми вирішується за допомогою віртуалізації каналів зв'язку [1].

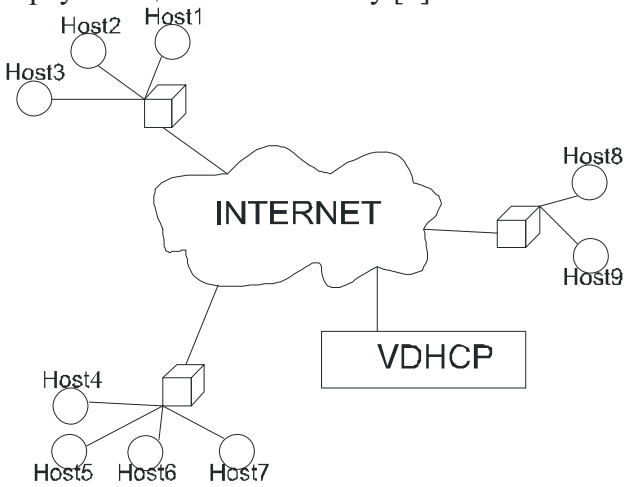



***Рис. 1. Фізична топологія мережі***

## 6. Керування віртуальними Grid системами

В якості контролера віртуальних Grid систем виступає, розроблений нами, VDHCP сервер. VDHCP сервер – сервер, відповідальний за створення віртуальних Grid мереж і розподіл навантаження на кожну з них. Основні завдання VDHCP сервера:

- Видача IP адреси, маски підмережі, IP адреси шлюзу віртуальної мережі.
- Використання планувальника задач на основі графа, що відображає топологію мережі (рис. 3).
- Примусова зміна IP адреси хоста для участі у віртуальній мережі.
- Зберігання списку віртуальних мереж і незадіяних комп'ютерів.
- Балансування навантаження на віртуальну

мережу.

- Створення переліку вирішених завдань.

VDHCP використовує планувальник завдань для прийняття рішення про створення нової віртуальної мережі. Часто завдання, які необхідно опрацювати, досить однотипні, отже для їх обробки потрібно приблизно однакову кількість ресурсів. VDHCP використовує цю особливість для розв'язання задачі планування – він використовує минулий досвід для прийняття рішення про створення нової віртуальної мережі і корегує його в процесі роботи, ґрунтуючись на коефіцієнті завантаження ресурсів віртуальної мережі.

### 6.1. Внутрішня структура мережі

Для VDHCP мережа представлена в узагальненому варіанті, де є хости і абстрактні зв'язки між ними. Приклад на рис. 2

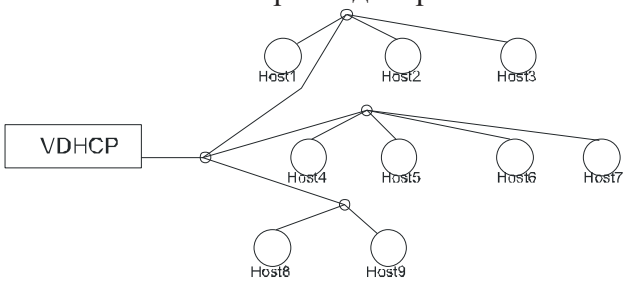



***Рис. 2. Внутрішня структура мережі***

Ґрунтуючись на даному поданні а також графі, що відображає топологію мережі (рис. 3), VDHCP приймає рішення про створення нової Grid мережі, реорганізації або видалення вже існуючої.

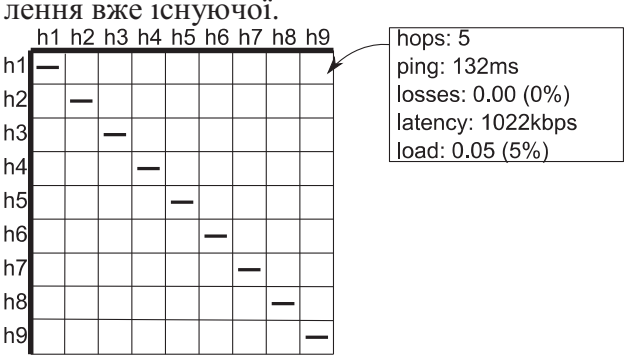



***Рис. 3. Граф топології мережі***

Для користувача, який виконав запит, в результаті роботи VDCHP, кластер виглядає, як реальна Grid мережа (рис. 4), завдяки чому не потрібно додаткові налаштування або перероблення програмного забезпечення.

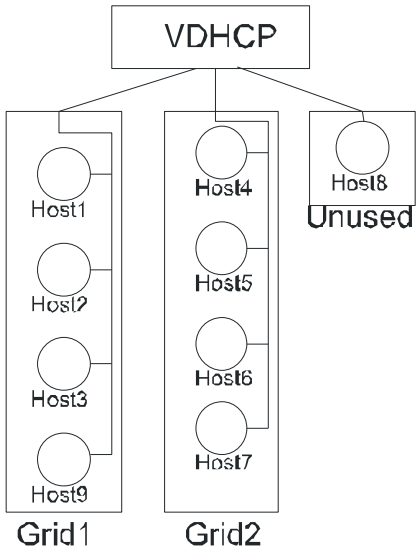



***Рис. 4. Результат роботи VDHCP***

### 6.2. Граф топології мережі

На рис. 3 відображена матриця, де записані дані про віртуальну топології мережі, дані записані у вигляді структур, як це показано в табл. 1

Розмір одного запису у матриці становить 4 + 8 + 8 + 8 + 8 = 36 байт. Отже хосту-учаснику віртуальної мережі необхідно відправляти $N^2$ по 36 байт інформації, що для мережі в 40 комп'ютерів складе 57600 байт або 56.25 Кбайт. VDHCP серверу ж доведеться приймати $N^3$ по 36 байт, що в свою чергу для мережі в 40 комп'ютерів складе 230400 байт або 2.197 Мбайт даних. У постійно змінюваному середовищі дані про топологію мережі можуть застаріти, тому VDHCP сервер запитує у хостів нові дані з певною періодичністю, яка задана в конфігурації сервера, так наприклад з періодичністю в 1 хвилину і розмірі мережі в 40 комп'ютерів VDHCP серверу доведеться використовувати мінімум 30 Кбіт/с з зі свого каналу на оновлення даних про стан мережі.

***Таблиця 1. Структура даних в матриці топології мережі***

| Поле | Відображення | Формат даних | Значення |
|---|---|---|---|
| hops | Число | int | Число проміжних елементів мережі між двома комп'ютерами |
| ping | Мілісекунди | double | Час передачі пакету розміром в 32 байта і очікування відповіді |
| losses | Відсотки | double | Кількість втрат на каналі при передачі даних |
| latency | Кілобіти на секунду | double | Доступна швидкість передачі між двома комп'ютерами |
| load | Відсотки | double | Завантаження каналу |

### 6.3. Планувальник задач

Планувальник завдань є невід'ємною частиною VDHCP сервера. Він приймає рішення, про те які комп'ютери будуть включені у віртуальну мережу для вирішення даної задачі. Для прийняття рішення планувальник враховує наступне:

1. Попередній досвід, якщо такий є
2. Ширину і якість каналів зв'язку між комп'ютерами
3. Кількість ресурсів, доступну для майбутньої мережі

Для врахування всіх цих факторів планувальник виконує пошук по графу, що відображає топологію мережі (рис. 3) і списку доступних комп'ютерів. Структура планувальника завдань VDHCP сервера відображена на рис. 5

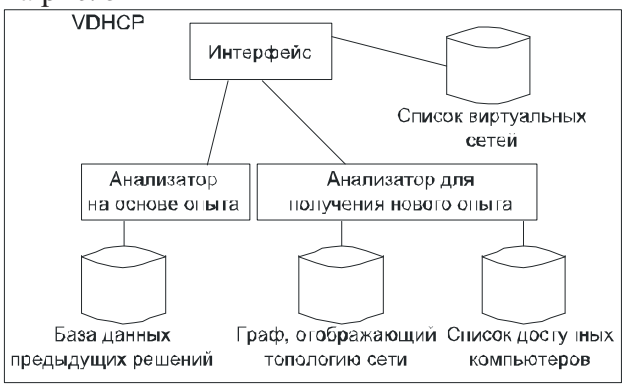



***Рис. 5. Структура планувальника завдань VDHCP сервера***

### 7. Моделювання

Для наглядного відображення роботи VDHCP сервера нами була розроблена програма для моделювання усіх його компонентів. Графічний інтерфейс користувача відображено на рис. 6.

Кожна задача моделювання запускається в окремому потоці, що відповідає реальній поведінці задач у системі. Задача представляє собою масив з чотирьох коефіцієнтів, які відповідають ступені складності задачі за кожним з чотирьох показників: вимогливість до швидкості каналів зв’язку між комп’ютерами, вимогливість до розміру ОЗУ, розмірів дискового простору і потужності центрального процесору. Спочатку задача передається до планувальника задач, який виділяє віртуальну мережу, за умови наявності ресурсів, або заблокує задачу до моменту, коли необхідні ресурси будуть доступними. Потім виконується моделювання задачі за допомогою ланцюгів Маркова, таким чином, що кожен тип ресурсів (ОЗУ, диск і процесор) кожного комп’ютера, включеного в віртуальну мережу має 2 стани – вільний і зайнятий. Вірогідності переходу між вершинами прямо пропорційні відношенням доступних ресурсів до необхідних. При моделюванні рахується час знаходження кожного ресурсу у цих станах, а коефіцієнт завантаженості ресурсу дорівнює часу, який ресурс перебував у стані «зайнятий» до загального часу моделювання.

$$\eta = \frac{T_{зайнятий}}{T_{загальний}}$$

Після чого відбувається корекція бази знань планувальника задач на основі отриманих результатів, при цьому планувальник задач намагається утримувати коефіцієнт навантаження на рівні 60-90% щоб уникнути перенавантаження і простоїв системи.

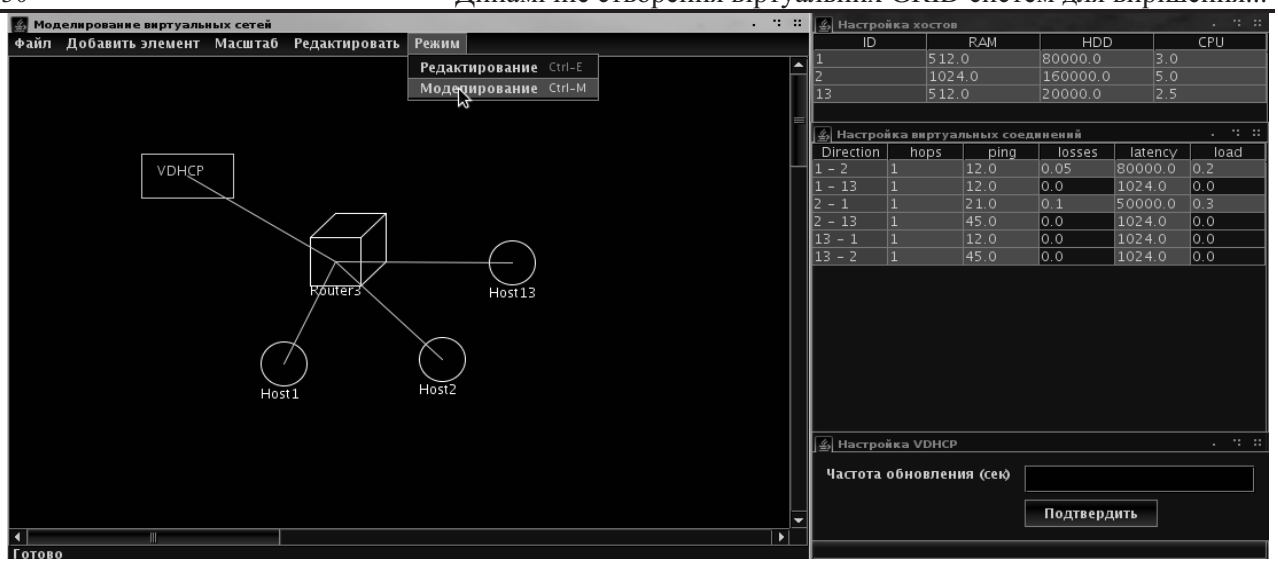



***Рис. 6. Графічний інтерфейс користувача програми моделювання***

Для відтворення функції корекції бази знань було проведено моделювання з використанням сімох однакових задач у якості вхідних даних, результат цього моделювання показаний на рис. 7. Для відтворення результатів моделювання у вигляді графіку було використано ПЗ gnuplot.

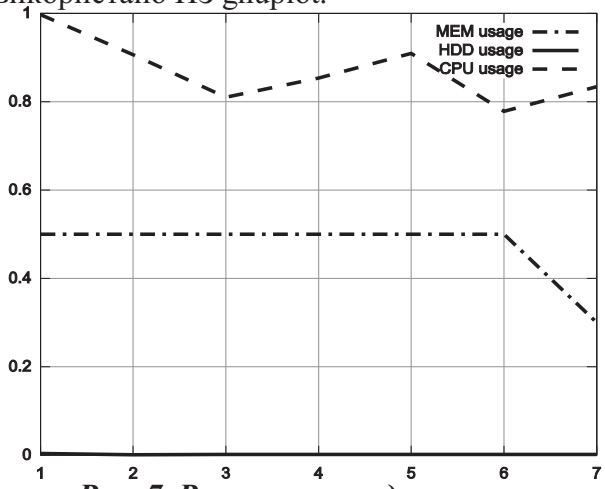



***Рис. 7. Результат моделювання***

### 7.1. Аналіз результатів моделювання

Як видно з рис. 7, вузьким місцем системи є центральний процесор, що доволі характерно для обчислювальних задач. З самого початку (1-2 задача) коефіцієнт навантаження центрального процесора кожного комп'ютера близький до одиниці, що означає, що система перевантажена, але вже починаючи з 3-го завдання коефіцієнт навантаження центрального процесора не перевищує позначки 0.9, що свідчить про успіх процесу наповнення бази знань.

## 8. Висновки

Загалом виділення віртуальних мереж для вирішення задачі показує гарні результати коефіцієнта навантаження на систему і дозволяє використовувати незадіяні ресурси мережі, що може зменшити час виконання великих обчислювальних задач, а відсутність потреби у адмініструванні дозволяє використовувати систему навіть у малому офісі.

Для подальшого розвитку технології, з метою поліпшення продуктивності, необхідно реалізувати дану технологію в якості модуля операційної системи, що дозволить зменшити час прийняття рішення про створення віртуальної мережі та зменшити час, необхідний для оновлення інформації про поточний стан мережі. Також важливою частиною подальшого розвитку є використання нових, більш ефективних, способів наповнення бази даних попередніх рішень, що значною мірою зменшить час визначення необхідного кількість ресурсів для нового завдання.

## Список посилань

1. Maur´ıcio Tsugawa, and Jos´e A. B. Fortes. A Virtual Network (ViNe) Architecture for Grid Computing. Parallel and Distributed Processing Symposium, 2006.
2. Foster, C. Kesselman and S. Tuecke. The Anatomy of the Grid: Enabling Scalable Virtual Organizations. International J. Supercomputer Applications, 15(3), 2001.
3. http://grid.kpi.ua/index.php?option=com_content&task=view&id=28&Itemid=57

Поступила в редакцию 17.12.2009

УДК 004.75

*МАСЛЯНКО П.П.,*
*РЯБУШЕНКО А.В.*

# СИСТЕМНЕ КОНСТРУЮВАННЯ ТА МОДЕЛЬ РОЗГОРТАННЯ РОЗПОДІЛЕНОЇ СИСТЕМИ УПРАВЛІННЯ ІНВЕСТИЦІЙНИМ ПОРТФЕЛЕМ

В роботі представлена модель розгортання розподіленої системи управління інвестиційним портфелем цінних паперів з використанням розподіленої нереляційної бази данних Cassandra. Проаналізовані переваги використання розподілених баз у порівнянні з реляційними базами даних.

In this paper deployment model of distributed investment portfolio management system is presented with application of distributed non-relational database Cassandra. Benefits of distributed databases are analyzed in comparison with relational databases.

## Вступ

Застосування прикладного системного аналізу і системної інженерії проектів інформатизації організаційних систем дозволяє суттєво вдосконалити процес створення систем фінансово-інвестиційної діяльності. Завдяки формалізації бізнес процесів та застосуванню компонентного процесу розробки вдається суттєво підвищити ефективність бізнес процесів управління інвестиційним портфелем, знизити ризики і зменшити повну вартість володіння системою [1].

З метою залучення закордонного капіталу в Україні з'явилась потреба створювати системи управління інвестиційним портфелем цінних паперів, які відповідають міжнародним стандартам: BASEL II, Sarbanes-Oxley та IFRS [2,3]. Створення подібних систем потребує строго формалізованого підходу до аналізу, проектування, конструювання та впровадження системи, який би відповідав би міжнародним вимогам. Компонентний процес розробки, що використовують автори, відповідає міжнародних нормам в області software engineering та system engineering [4].

В роботах [5-7] запропонована компонентна модель системи управління фінансово-інвестиційною діяльністю, що розроблена у відповідності міжнародним стандартам з використанням компонентного процесу розробки.

В цій роботі досліджується проблема системного впровадження системи управління інвестиційним портфелем [5-7] у корпоративній гетерогенній мережі банку UBS, що входить до п'ятірки найбільших банків світу. Пропонується метод системного впровадження у вигляді розподіленої системи.

Розподілені системи у фінансово-інвестиційній сфері дозволяють підвищити надійність та продуктивність, забезпечити масштабованість та розширюваність, саме тому впровадження розподілених систем є надзвичайно актуальною проблемою для підвищення ефективності функціонування систем управління фінансово-інвестиційної діяльності.

## Постановка задачі

Мета дослідження полягає в застосуванні системної інженерії проектів інформатизації організаційних систем для реалізації фази системного конструювання розподіленої системи управління інвестиційним портфелем і пошуку ефективних методів системного впровадження.

Об'єкт дослідження – розподілені обчислення, як метод впровадження системи управління інвестиційним портфелем цінних паперів.

Предмет дослідження – системна інженерія фінансово-інвестиційної діяльності.

## Системне конструювання розподіленої системи управління інвестиційним портфелем

Ідея застосування розподілених систем у фінансово-інвестиційній та банківській сфері досліджується вже доволі давно. Результати багаточисельних досліджень підсумовані у [8,9]. Розподілені системи останні 10 років вважаються перспективним напрямком досліджень у фінансовому секторі економіки,

але їх впровадження у корпоративних мережах просувається дуже повільно.

Розподілені системи мають ряд переваг: масштабованість, підвищена надійність, заміна апаратного забезпечення без зупинки надання сервісів, можливість використання обчислювальних потужностей вже закуплених серверів, які, як показує статистика задіяні менш ніж на 10%, тим самим зменшити витрати на апаратне забезпечення та багато інших.

Але розподілені системи мають також ряд недоліків. Не кожну задачу можна ефективно вирішувати за допомогою розподілених обчислень. Задача повинна піддаватись розпаралелюванню, тобто містити в собі окремі функціональні блоки, що можуть виконуватись незалежно на різних апаратних вузлах. Розробка ефективних розподілених алгоритмів це складна та трудомістка задача [10]. Існуюче програмне забезпечення має бути переписаним з використання розподілених технологій, що потребує значний коштів та часу, саме тому компанії не квапляться упроваджувати нові розподілені технології.

Математичні методи управління ризиками, що використовуються в UBS, детально описані у [11]. Проаналізувавши математичні методи моделювання можна зробити висновок, що вони є сильно зв'язаною задачею, що погано піддається розпаралелюванню. Тому впровадження розподілених обчислень у системі управління інвестиційним портфелем спіткає ряд серьозних труднощів. Тим не менш, навіть в таких умовах розподілені обчислення можуть привнести додаткові переваги.

Стандартний метод впровадження, що найбільш поширений серед фінансових компаній – це трьох рівнева архітектура (клієнт, сервер, база даних). В такій архітектурі доволі часта саме база данних стає вузьким місцем. Реляційні БД погано масштабуються, тому саме вони часто стають причиною низької швидкодії та точкою єдиного виходу з ладу.

Автори пронуються використати розподілену базу данних заміст реляційної БД з метою підвищення надійності, швидкодіїї та відходу від реляційної моделі пердставлення інформаціїї.

## Компонента модель системи управління інвестиційним портфелем

Компонентна модель системи управління інвестиційним портфелем та її специфікація представлена у [5] та розширена в роботах [6,7]. Система складається з наступних компонентів (рис. 1): оптимізація інвестиційного портфеля, управління ризиками, прогнозування, оцінка вартості похідних фінансових інструментів, сек'юритизація, алгоритмічний трейдинг, арбітраж, статистичний арбітраж, фінансова інженерія, прийняття стратегічних рішень. Основне призначення системи – надати інвестиційним компаніям інтегровану систему управління, що дозволить проводити моделювання інвестиційного портфеля для сформування портфеля з оптимальними характеристиками, прискорення прийняття інвестиційного рішення, ідентифікації та оцінки ризиків.

На рис. 2 наведено суперклас системи, що є структурним мета-представленням сутностей розподіленої системи управління інвестиційним портфелем. Суперклас представлений у нотації UML [12] і визначає сутність проблеми управління інвестиційний портфелем, його інтеграцію з зовнішніми системами. Таке представлення системи управління дозволяє на етапі аналізу виділити задіяні зовнішні системи та інтерфейси їх взаємодії. Система управління інвестиційним портфелем спрямована на використання інституційними інвесторами для управління великими портфелями цінним паперів. Клієнтам, які бажають приймати участь у управлінні їхньої частини портфеля надається інтерфейс користувача. Система повинна оперувати як на біржовому ринку, так і на позабіржовому ринку (Over-the-counter), який по об'єму торгів значно перевищує біржовий. Також, до класу позабіржових систем входить важливий тип ринків "Чорні пули ліквідності" [13], який став грати важливу роль з зниженням ліквідності на світових інвестиційних ринках, що викликано фінансовою кризою.

Інформаційні агенції надають інформацію, яка необхідна для прийняття інвестиційного рішення. Інформаційні агенції можна умовно розділити на три групи: провайдери фінансових нових, такі як Reuters, Bloomberg; провайдери фінансових звітів емітентів, наприклад Bloomberg, FSA та про-

вайдери аналітичних звітів, наприклад JPMorgan Chase, Morgan Stanley.

### Засоби реалізації

На рис. 3 наведено пакет засобів реалізації системи управління інвестиційним портфелем у профілі Еріксона-Пенкера [14]. Засоби реалізації розподілені на програмі та апаратні.

До апаратних відносять елементи корпоративної мережі: сервери додатків, сервери баз даних, маршрутизатори та мережеве обладнення. Згідно з внутрішніми корпоративними стандартами UBS, в якості серверів баз даних використовуються сервери SUN з процесорами UltraSparc. Для серверів додатків був зроблений виняток – використовуються сервери SUN з процесорами AMD Opteron.

Згідно з внутрьошнімі стандартми UBS до програмних засобів відносять: операційну систему – Solaris 10, файлову систему – ZFS, реляційну базу даних – Oracle 10g.

Взаємодія з зовнішніми системами угод відбувається за стандартним протоколом FIX (Financial Information eXchange). Для систем з підвищеними вимогами до продуктивності застосовується протокол FAST (FIX Adapted for STreaming). Для котирувань складних похідних фінансових інструментів використовується додаткові протоколи: MDDL (Market Data Description Language) та FpML (Financial Products Markup Language). З зовнішніми інформаційними системами взаємодія відбувається за стандартними протоколами, що базуються на XML: для обміну фінансовими новинам – NewsML (News Markup Language), фінансовою звітністю емітентів цінних паперів – XBRL (eXtensible Business Reporting Language), аналітичними звітами – RIXML (Research Information eXchange Markup Language) [15].

### Системне розгортання розподіленої системи управління інвестиційним портфелем

На рис. 4 наведена модель розгортання системи управління інвестиційним портфелем. Система включає в себе: основну та резервну реляційні бази даних, що виконують роль центрального сховища для обробки аналітичних запитів, розподілювач навантаження з фільтрацією запитів. Для збереження критично важливої бізнес інформації забезпечується її архівація на магнітну стрічку. Загрузка даних із зовнішніх джерел реалізована окремими компонентами, які завантажують інформацію у реляційну БД.

Система складається з необхідної кількості типових обчислювальних вузлів з типовим програмним забезпеченням (рис. 4). Апаратна реалізація вузлів може бути різною.

В якості розподіленої БД використовується нереляційна БД – Cassandra. Кожен обчислювальних вузол має свій екземпляр розподіленої БД. База даних Cassandra поєднує масштабованість, надмірність та надійність такої повністю розподіленої бази данних, як Dynamo [16] та баз данних розділенних таблиць (partitioned table) BigTable і Hadoop [17]. Cassandra не підтримує мови запитів SQL, так як вона не є реляційною базою даних. Замість SQL використовується мова запитів, що схожа на MapReduce [18]. Сховище Cassandra орієнтовано на зберігання масивів (column-oriented storage) [19,20], що має кращу швидкодію у порівнянні зі зберіганням окремих значень (value-oriented storage), таких розподілених БД, як Memcache та Dynamo; та row-oriented storage реляційних БД.

База даних Cassandra забезпечує надійність завдяки автоматичній реплікації розподілених БД, а також підтримує прозоре введення нових обчислювальних вузлів у кластер без зупинки обслуговування. Нереляційна БД Cassandra, на відміну від реляційної БД, забезпечує значне підвищення продуктивності обчислень та дозволяє ефективно оперувати різними типами фінансових інструментів, які не можна ефективно представили у реляційній алгебрі. Погана структурованість фінансових інструментів та дуже великі їх відмінності один від одного не дозволяють спроектувати ефективну БД фінансових інструментів у реляційній алгебрі.

Обчислювальні задачі виконуються на тому вузлі, який має необхідну інформацію у власному екземплярі БД, що суттєво знижує навантаження на комунікаційну мережу. Результати обчислень спочатку зберігаються в нереляційній БД, а потім в асинхронному режимі передаються в реляційну БД.

Для прискорення обчислювально-складних алгоритмів в системі задіяні відеокарти.

Сучасні відеокарти мають від 500-1200 процесорів обробки чисел з плаваючою крапкою одинарної точності. Це величезна обчислювальна потужність, яку можна задіяти, якщо алгоритм піддається розпаралелюванню. Для цього застосовуються GPGPU бібліотеки [21], які дозволяють використовувати відеокарти в тому числі і для математичних обчислень.

## Висновки

Застосування системної інженерії для реалізації фази системного конструювання систем управління інвестиційним портфелем цінних паперів забезпечує необхідні значення ключових властивостей ІКС, а саме: продуктивності, масштабованості, розширюваності, надійності, безпеки, прозорості, низької повної вартості володіння системою.

Необхідні значення ключових властивостей ІКС досягаються завдяки використанню розподіленої нереляційної БД Cassandra. Крім цього, відмова від реляційного принципу представлення даних, дозволило ефективно оперувати фінансовими інструментами максимально наблизити їх використання до реального масштабу часу.

Використання процесорів відео карт та GPGPU бібліотек [21] дозволило суттєво підвищити ефективність використання наявних обчислювальних ресурсів для складних алгоритмів, наприклад непараметричне Монте Карло [7], генерація багатовимірних випадкових чисел [22] та генетичного алгоритму оптимізації [23].

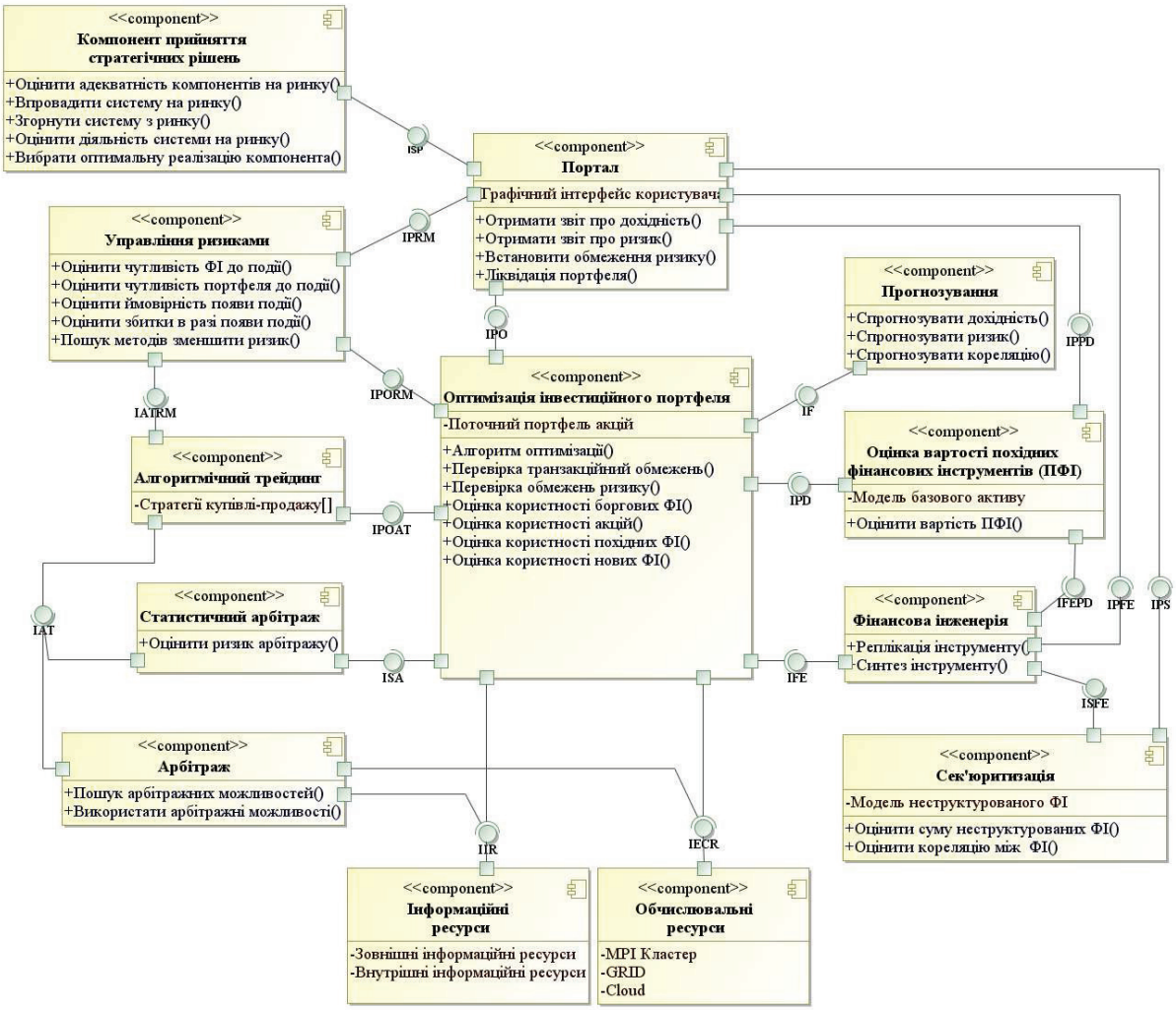



***Рис. 4. Модель системи управління фінансово-інвестиційною діяльністю. Діаграма компонентів в нотації UML***


***Рис. 2. Суперклас системи управління інвестиційним портфелем. Діаграма класів у нотації UML***

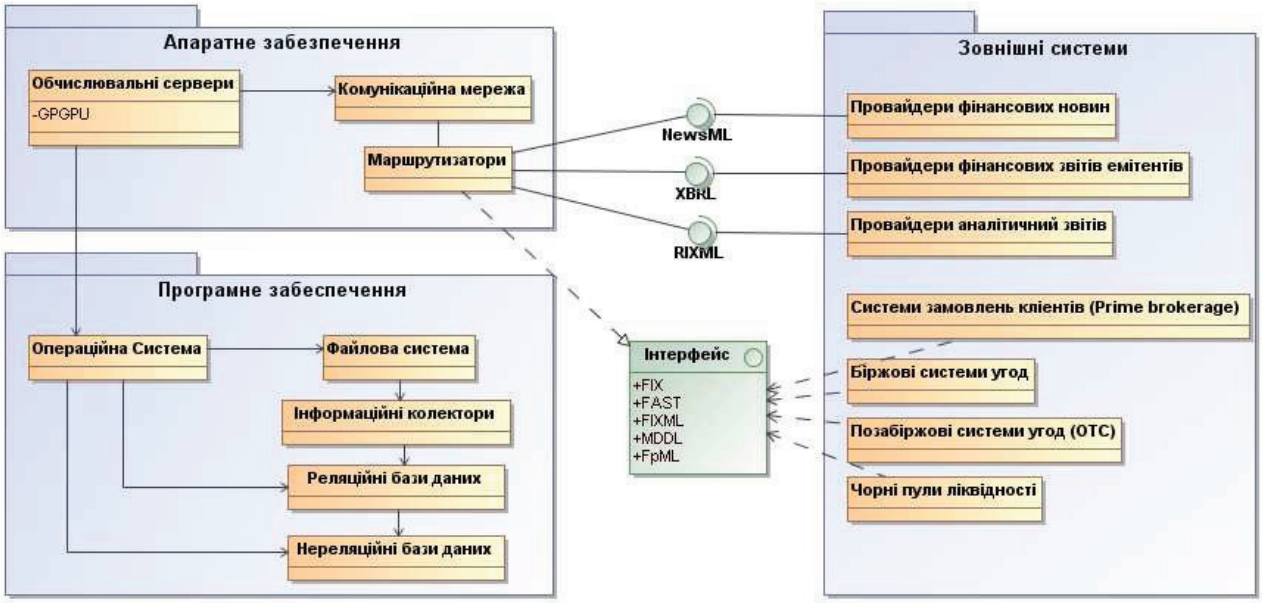



***Рис. 3. Засоби реалізації системи управління інвестиційним портфелем. Пакет класів у профілі Еріксона-Пенкера***

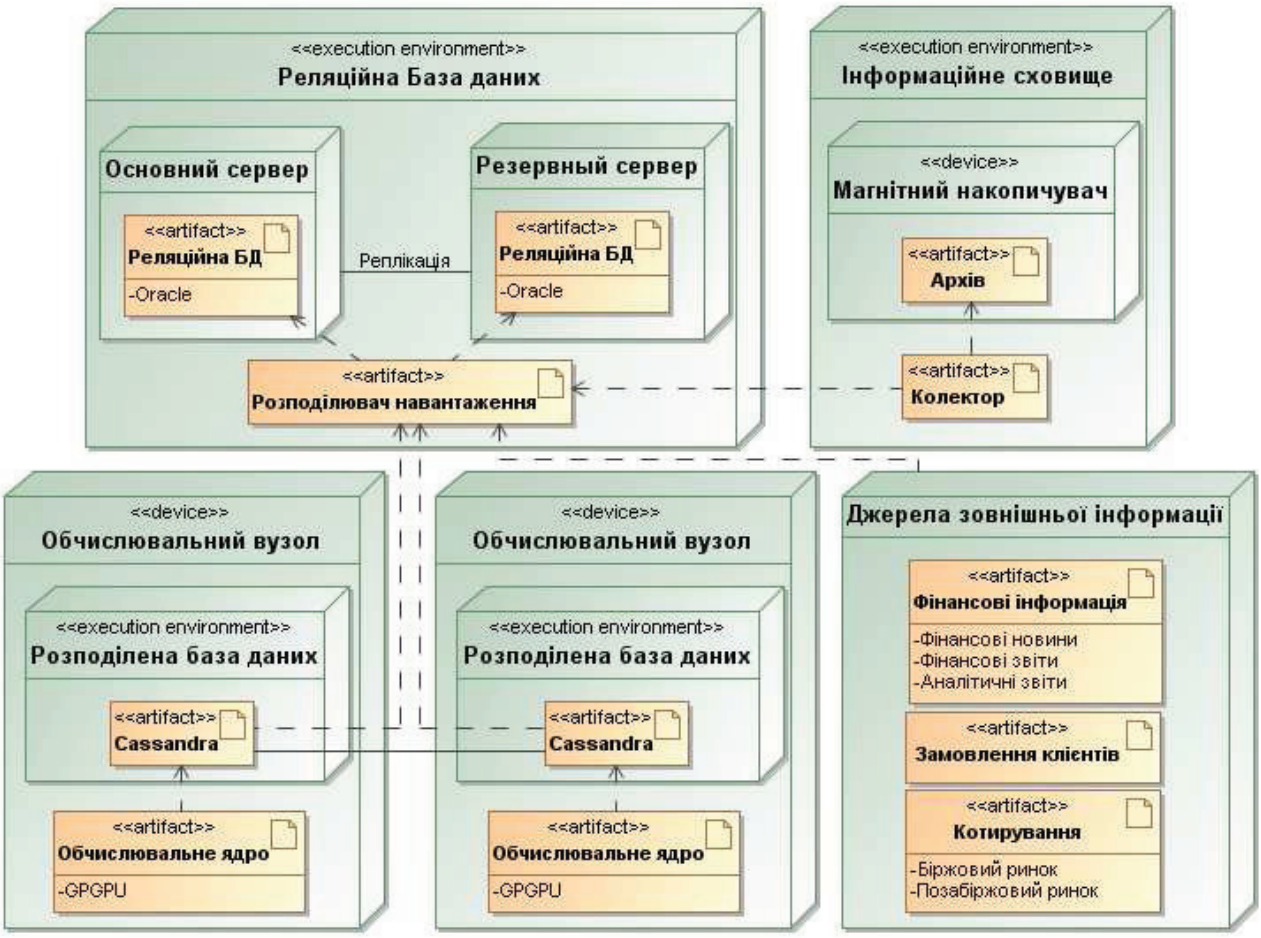



***Рис. 4. Модель розгортання системи управління інвестиційним портфелем. Діаграма впровадження у нотації UML***

## Список посилань


1. Маслянко П.П. Концепція інформатизації корпоративних структур. Наукові вісті НТУУ “КПІ” 2001, № 3 - С.45-50.
2. Basel Committee on Banking Supervision. International Convergence of Capital Measurements and Capital Standards. A Revised Framework. 2005. Bank of international settlements - 284 pp.
3. Tarantino A. Manager's Guide to Compliance: Sarbanes-Oxley, COSO, ERM, COBIT, IFRS, BASEL II, OMB's A-123, ASX 10, OECD Principles, Turnbull Guidance, Best Practices, and Case Studies // Wiley, 2006. – 336pp.
4. Маслянко П.П. Основні положення методологій системного проектування інформаційно-комунікаційних систем. Наукові вісті НТУУ “КПІ” 2007, № 6 - С.54-60.
5. Маслянко П. П., Рябушенко А. В. Компонентна модель інформаційно-аналітичної системи та генетичний алгоритм формування оптимального портфеля акцій // Наукові вісті НТУУ “КПІ” 2009, № 1 - С.36-46.
6. Маслянко П. П., Рябушенко А. В. Створення компонента стратегічного планування системи управління фінансово-інвестиційною діяльністю // Наукові вісті НТУУ “КПІ” 2009, № 4 - С.53-65.
7. Маслянко П. П., Рябушенко А. В. Підсистема управління ризиками фінансово-інвестиційної діяльності // Вісник східноукраїнського національного університету ім. Володимира Даля. – 2009. – Т.2, №1. – С.370–378.
8. Tanenbaum A. S., Van Steen M. Distributed Systems: Principles and Paradigms. – Prentice Hall, 2006. – 2nd Edition. – 704 pp.
9. Dollimore J., Kindberg T., Coulouris G. Distributed Systems: Concepts and Design Addison. – Wesley, 2005. – 4th Edition. – 944 pp.
10. Ghosh S. Distributed Systems: An Algorithmic Approach. – Chapman & Hall/CRC Computer, 2006. – 424 pp

11. Cesari G., Aquilina J., Charpillon N. та ін. Modelling, Pricing and Hedging Counterparty Credit Exposure // Springer finance, 2009. – 246 pp.
12. Unified Modeling Language Specification, Version 2.0. Object Management Group, Framingham, Mass., 2004.
13. Banks E. Dark Pools: The Structure and Future of Off-Exchange Trading and Liquidity. – Palgrave Macmillan, 2010. – 240 pp.
14. Ericsson H.E., Penker M. Business Modeling with UML: Business Patterns at work // Wiley Computer Publishing, 2000. – 350 pp.
15. Martin S. MDDL and the quest for a market data. – Elsevier, 2007. – 320 p.
16. Hastorun D., Jampani M., Kakulapati G. та ін. Dynamo: amazon's highly available key-value store // In Proc. SOSP, 2007. – P.205-220.
17. Chang F., Dean J., Ghemawat S. Bigtable: A Distributed Storage System for Structured Data // OSDI'06: Seventh Symposium on Operating System Design and Implementation, 2006. – P. 205–218.
18. J Dean, S Ghemawat. MapReduce: Simplified data processing on large clusters // Communications of the ACM, 2008. – Vol. 51, No. 1. – P. 107-113.
19. Stonebraker M., Abadi D. J., Batkin Ad. та ін. C-store: a column-oriented DBMS // VLDB '05: Proceedings of the 31st international conference on Very large data bases. – VLDB Endowment, 2005. – p. 553-564
20. Abadi D., Madden S., Ferreira M. Integrating compression and execution in column-oriented database systems // SIGMOD '06: Proceedings of the 2006 ACM SIGMOD international conference on Management of data. – ACM, 2006. – p.671-682
21. Pharr M., Fernando R. GPU Gems 2: Programming Techniques for High-Performance Graphics and General-Purpose Computation // Addison-Wesley Professional, 2005. – 880 pp.
22. Маслянко П. П., Рябушенко А. В. Непараметричне Монте Карло: вдосконалений метод моделювання ризиків на фінансовому ринку // Всеукраїнський математичний конгрес. – К.: Інститут математики НАН України, 2009. http://www.imath.kiev.ua/~congress2009/Abstracts/Maslyanko_Ryabushenko.pdf
23. Маслянко П.П., Рябушенко А.В. Генетичний алгоритм оптимізації портфеля акцій на фондовому ринку // Системний аналіз та інформаційні технології. – К.: ННК "ІПСА" НТУУ "КПІ", 2009. – С.347.

Поступила в редакцию 3.12.2009

УДК 681.322

*СЕРГИЕНКО А.М.*

# ПРОСТРАНСТВЕННЫЙ ГРАФ СИНХРОННЫХ ПОТОКОВ ДАННЫХ

В статье рассмотрена модель пространственного графа синхронных протоков данных, предложенная для задания периодических алгоритмов и их отображения в вычислительные средства. Модель позволяет одновременно выполнять как составление расписания выполнения операций, так и поиск структуры вычислительного устройства, что улучшает процесс оптимизации решения.

A space synchronous dataflow graph computational model is considered, which is useful for periodical algorithm composition and mapping into computers. This model provides to implement simultaneously both operator scheduling and structure deriving, which improves the optimization process.

## Введение

Персональные компьютеры, средства цифровой обработки сигналов (ЦОС) и другие вычислительные системы (ВС) реального времени выполняют периодические алгоритмы, обрабатывающие потоки данных и/или команд. Для реализации ЦОС чаще всего используются заказные СБИС, сигнальные микропроцессоры и программируемые логические интегральные схемы (ПЛИС).

Сейчас для программирования многопроцессорных параллельных ВС применяется парадигма многопотоковой обработки (multithreading), которая поддерживается как на уровне матобеспечения, так и на аппаратном уровне современных компьютеров. Но многопотоковая модель не гарантирована от блокировок. Поэтому вместо нее в работе [1] предложено использовать модель графа синхронных потоков данных (ГСПД) и его модификации, гарантирующие безопасность реализации алгоритма.

Современные задачи ЦОС не могут быть решены без разработки методов и средств отображения алгоритмов в СБИС и ПЛИС, программирования многопроцессорных ВС. В статье предлагается модель пространственного ГСПД, которая позволяет проектировать эффективные аппаратные устройства для реализации периодических алгоритмов.

## Модели потоков данных

По определению Поста и Тьюринга, алгоритм – это вычислительный процесс, который выполняется моделью вычислителя, сконструированной в рамках точных математических понятий [2]. Модель вычислителя, которая реализует функциональный алгоритм, принято представлять в виде графа. В такой модели вершины графа означают операторы или локальные процессы алгоритма, а дуги – каналы передачи данных, зависимости по данным и управлению. Реализация алгоритма на такой модели представляет собой передачу данных в направлении дуг и обход вершин в сответствии с их инцидентностью, наличием данных на их входах или по другим правилам.

На рис.1 показан граф классификации наиболее известных графовых вычислительных моделей. Анализ различных графовых моделей с точки зрения детерминизма алгоритма, возможностей и удобства задания алгоритма для обработки потоков данных и отображения его в аппаратные средства приведен в [3,4]. Далее будет рассмотрена модель ГСПД, как модель, наиболее приближенная к заданию аппаратно реализованных алгоритмов.

Вершины – акторы графа потоков данных (ГПД), как и производного от него ГСПД, представляют операции, а дуги – потоки передачи данных. Акторы потребляют со своих входов данные, называемые метками или токенами и выдают метки на свои выходы.

Корректность задания алгоритма на модели вычислителя может быть проверена или аналитически (если это возможно), или путем составления расписания выполнения вычислительного процесса на ней. Расписание также составляется при выполнении алгоритма в конкретной ВС.

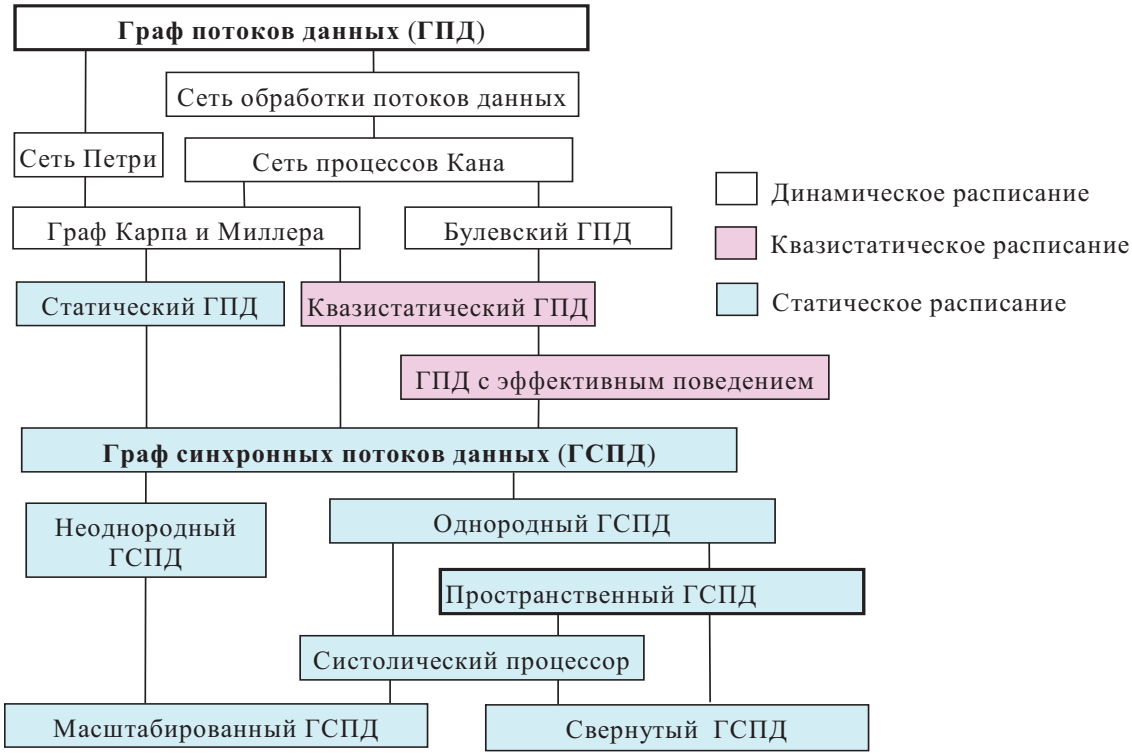



***Рис.1. Классификация графов потоков данных***

Если структура ВС отвечает графу модели, то составление расписания состоит в определении порядка выполнения операторов (акторов, процессов). Если структура ВС произвольная, то дополнительно выполняют назначение операторов на процессоры этой ВС.

*Статическое расписание* составляется до выполнения алгоритма в ВС, т.е. во время компиляции, а динамическое расписание можно определить только при исполнении алгоритма в ВС при обработке конкретных данных. Согласно виду расписания, различают *статические и динамические модели*. Если поведение и параметры модели можно предвидеть на стадии компиляции (максимальный период вычислений, размер буферов потоков и т.п.), то такую модель называют *моделью с квазистатическим расписанием*. Следует отметить, что при отображении алгоритма в аппаратуру СБИС и ПЛИС допускаются алгоритмы только со статическим расписанием.

Таким образом, ГПД представляе собой естественную модель для задания алгоритмов обработки потоков данных. Она имеет широкие возможности выразительного задания большого множества алгоритмов на высоком уровне абстракции. Платой за это является необходимость динамического составления расписания, и в результате – недетерминированность поведения, которая угрожает наличием блокировок [1,3]. Из-за этого они используются, в основном, в ВС с динамическим расписанием и не приспособлены для синтеза аппаратных средств вычислительной техники.

*Поток данных* – это средство передачи упорядоченных данных между компонентами модели. Данные в потоке, упорядоченные по номерам ассоциированных с ними тегов, например, временных меток, называют *сигналом*. Сигналы и соответствующие им потоки считаются *синхронными*, если теги пар сигналов имеют взаимно-однозначное соответствие. Модель считается *синхронной*, если все сигналы в ней – синхронные, иначе – это *асинхронная модель*.

Синхронный сигнал производится *регулярным актором*, т.е. актором, который использует и производит некоторое количество меток, которое стабильно при выполнении алгоритма и может быть определено во время компиляции. В отличие от него, *динамический актор* выдает различное число меток в зависимости от данных в его входных потоках, т.е. генерирует асинхронный сигнал.

Граф Карпа и Миллера, в котором все акторы – регулярные – это *граф синхронных потоков данных* (ГСПД, synchronous data flow, SDF). Если количество меток, использованных каждым входом и сгенерированных каждой вершиной в одном цикле, одинаково и равно константе, то такой граф на-

зывают *однородным* ГСПД (homogeneous SDF). А если оно различно, то это *неоднородный* (многоскоростной) ГСПД (multirate SDF).

Неоднородный ГСПД – это удачная модель для компактного задания периодических алгоритмов. Но его сложнее анализировать в сравнении с однородным ГСПД. Кроме того, его моделирование усложняется из-за возможности блокировок. Поэтому для анализа и синтеза на его основе ВС неоднородный ГСПД часто преобразуют в эквивалентный однородный ГСПД. Модель неоднородного ГСПД получила большое распространение, например, в пакете Matlab-Simulink.

ГСПД – это статическая модель. Для неоднородного ГСПД отсутствие блокировок несложно определяется при решении системы уравнений баланса $\Gamma \boldsymbol{r} = 0$, где $\Gamma$ – топологическая матрица графа, $\boldsymbol{r}$ – вектор повторений срабатывания акторов. Однородный ГСПД никогда не блокируется при наличии меток начального состояния в буферах, входящих в циклы графа. Таким образом, среди большого разнообразия потоковых моделей алгоритмов, ГСПД является наиболее подходящей моделью для задания алгоритмов, отображаемых в аппаратные средства.

## Применение ГСПД для проектирования аппаратных средств

Однородный ГСПД взаимно однозначно отвечает сигнальному графу алгоритма ЦОС или вычислительной схеме с периодом вычислений один такт. Существуют такие его разновидности, как *масштабированный* ГСПД (scalable SDF), *многомерный* ГСПД, *цикло-статический* ГСПД (cyclo-static dataflow), *параметрический* ГСПД, блоковый ГПД (Blocked Data Flow), предназначенные для задания алгоритмов решения современных задач ЦОС. В [3] дан обзор методов отображения ГСПД в многопроцессорные ВС на основе программной реализации акторов.

Согласно общепринятой методологии, отображение ГСПД в ВС выполняют в три этапа. На первом этапе выбирают множество аппаратных ресурсов вичислителя, ассортимент и количество которых отвечают набору операторов алгоритма и требуемой производительности. На втором – составляется расписание исполнения акторов на выбранных ресурсах. И на третьем этапе – этапе назначения – акторы распределяются среди ресурсов, данным назначаются элементы памяти, определяется структура ВС, планируются пересылки данных, синтезируется управляющий автомат. Оптимизация ВС состоит в формировании множества альтернативных решений и выборе наиболее предпочтительной ВС по заданному критерию качества.

Этапы отображения преследуют противоречивые цели – выбор ресурсов минимизирует аппаратные затраты, а составление расписания – минимизирует время вычислений за счет увеличения аппаратных ресурсов. Поэтому такое отображение ГСПД часто приводит к решениям, далеким от оптимальных. Окончательная оценка аппаратных затрат выполняется на последнем этапе, когда уже готово расписание. И для оптимизации необходимо многократно повторять этапы отображения.

Составление расписания сводится к построению на основе ГСПД графа зависимости по данным (ГЗД) и отображению его топологической сортировки в пространство событий. При этом точное решение этой задачи (целочисленное линейное программирование, метод ветвей и границ) возможно только для числа вершин не более двух-трех десятков. Такой эффективный эвристический метод, как силовое планирование, также не учитывает факторы, возникающие на этапе размещения.

Таким образом, модель ГСПД, обладая выразительностью, отсутствием блокировок, имеет некоторые трудности отображения в аппаратные ресурсы. Во многом, это связано с тем, что в ВС отображается не сам ГСПД, а соответствующий ему ГЗД (составление расписания) и множество его вершин (назначение на ресурсы), а межпроцессорные связи ВС, внутренняя структура процессорных плементов (ПЭ), управляющий автомат формируются искусственно на последнем этапе.

В статье предлагается вершинам и дугам ГСПД назначить теги с многомерными векторами, благодаря которым ГСПД отображается непосредственно и одновременно как в структуру ВС, так и ее расписание.

Важно отметить, что известный граф систолического процесора, по существу, является однородным ГСПД. Причем его вершины

и дуги уже имеют теги из многомерных векторов. Однако, в отличие от отображения ГСПД, систолические процессоры синтезируются путем отображения ГЗД [5].

### Пространственный ГСПД

Любой граф алгортима $G_A$, в том числе ГСПД и соответствующий ему ГЗД, задается матрицей инцидентности

$$A = ||a_{i,j}||_{N*M},$$

где $a_{i,j} = 1$, если дуга $j$ графа $G_A$ концом инцидентна $i$-й вершине или $a_{i,j} = -1$, если началом инцидентна $i$-й вершине оператора, и $a_{i,j} = 0$ – при отсутствии инцидентности, $N$ и $M$ – число вершин и дуг графа $G_A$.

Данная матрица инцидентности – нетрадиционная – она имеет инверсные знаки $a_{i,j}$. Это связано с тем, что здесь разность координат дуг, умноженных на $a_{i,j}$, указывает на положительные задержки операндов.

В общем случае, ГСПД представим в $n$-мерном пространстве $\mathbf{Z}^n$. Векторы $\boldsymbol{K}_i \in \mathbf{Z}^n$ отождествляются с вершинами графа $G_A$, т.е. играют роль тегов. Координаты векторов $\boldsymbol{K}_i$, могут иметь разную интерпретацию. Часть координат векторов $\boldsymbol{K}_i$ – это координаты $\boldsymbol{K}_{Si}$ ПЭ, в котором выполняется $i$-й оператор, а другая часть $\boldsymbol{K}_{Ti}$ – кодирует момент времени выполнения оператора $\boldsymbol{K}_i = \left(\boldsymbol{K}_{Si}^{\mathrm{T}}, \boldsymbol{K}_{Ti}^{\mathrm{T}}\right)^{\mathrm{T}}$. При отображении алгоритма в виде гнезда циклов вектор $\boldsymbol{K}_i$ является набором индексов переменных.

По другим координатам $\boldsymbol{K}_i$ может вестись отсчет, например, уровней иерархии алгоритма и структуры, типа оператора и ПЭ, задержки выполнения оператора, кодироваться информация об особенностях операторов и ПЭ, таких, как необходимость конвейерной реализации, заданный момент выполнения и т.п. Для простоты изложения далее будет рассматриваться размерность пространства $n = 3$. При изображении ГСПД алгоритма в 3-мерном целочисленном пространстве $\mathbf{Z}^3$ $i$-й оператор отождествляется с радиусом-вектором $\boldsymbol{K}_i = (s, p, t)^{\mathrm{T}}$, $(i = 1...N)$, где $p$ – тип операции, $s$ – пространственная координата, $t$ – временная координата. Более подробно о семантике координат векторов будет изложено при рассмотрении отображения ГСПД.

Обычно вершины ГСПД генерируют метки, отвечащие одиночным переменным $x_j$. В других случаях, когда вершина выдает различные переменные в разные дуги, следует расщепить оператор вершины на несколько операторов, которые генерируют одиночные результаты $x_j$.

Аналогично, каждая переменная $x_j$ из $M$ переменных алгоритма отождествляется с $n$-мерным вектором $\boldsymbol{D}_j$; $(j = 1...M) \in \mathbf{Z}^3$, причем

$$\boldsymbol{D}_j = \boldsymbol{K}_l - \boldsymbol{K}_i, \qquad (1)$$

где $\boldsymbol{K}_i$ соответствует оператору с результатом $x_j$, а $\boldsymbol{K}_l$ – оператор, у которого $x_j$ – исходный операнд, т.е. $\boldsymbol{K}_i$ непосредственно предшествует $\boldsymbol{K}_l$.

Множество векторов $K = \{\boldsymbol{K}_i\}$ и $D = \{\boldsymbol{D}_j\}$ взаимосвязаны точно так же, как множества вершин и дуг алгоритма. Таким образом, представление алгоритма в виде пространственного ГСПД означает его кодирование в виде матриц векторов $\boldsymbol{K}_i$, $\boldsymbol{D}_j$, а также матрицы инцидентности алгоритма. Для того, чтобы задать ГСПД с помощью этих множеств векторов, необходимы следующие определения:

*Конфигурацией алгоритма* (КА) является тройка $C_A = (K,D,A)$, где $K = (\boldsymbol{K}_1, ..., \boldsymbol{K}_N)$ – матрица координат векторов-вершин $\boldsymbol{K}_i = (s_i, p_i, t_i)^{\mathrm{T}} \in \mathbf{Z}^3$; $D = (\boldsymbol{D}_1, ..., \boldsymbol{D}_{M+1})$ – матрица координат векторов-дуг $\boldsymbol{D}_j = (s_j, p_j, t_j)^{\mathrm{T}} \in \mathbf{Z}^3$; $A = ||a_{i,j}||_{N*(M+1)}$ – матрица инцидентности графа $G_A$, которая дополнена $M$+1-м столбцом с элементом $a_{p,M+1} = 1$, которому отвечает базисный вектор $\boldsymbol{D}_{M+1} = \boldsymbol{D}_B$, соединяющий начало системы координат пространства $\mathbf{Z}^3$ с некоторым произвольным вектором-вершиной $\boldsymbol{D}_B = \boldsymbol{K}_p \in K$ [6].

Название „конфигурация” происходит от понятия комбинаторной конфигурации, т.е. одного из элементов множества, составленого из групп элементов другого множества. Так же, как и комбинаторная конфигурация, КА представляет собой элемент множества возможных представлений алгоритма в многомерном пространстве, т.е. элементом множества решений задачи отображения алгоритма.

Векторы $\boldsymbol{K}_i$ КА занимают в пространстве ресурсы – время (с координатами $s,t$) площадь некоторой фигуры, покрываемой прямоугольником со сторонами $S_m$ и $T_m$. При максимальной загруженности ПЭ структуры, площадь прямоугольника $Q = S_m T_m$ принимает минимальное значение. При различных

вариантах отображения можно получить разное время выполнения алгоритма $T_m$ и количество ПЭ $S_m$. Причем, при условии максимальной загруженности ПЭ площадь прямоугольника $Q$ остается приблизительно постоянной, т.е. получается масштабируемая структура ВС.

### Применение пространственного ГСПД для синтеза конвейерных ВС

Конвейерная ВС – это многоступенчатый аппаратный вычислитель, который обрабатывает потоки данных. Пространственный ГСПД дает возможность формального построения конвейерных вычислителей на основе метода его отображения, описанного ниже.

Если ГСПД представлен в пространстве $\mathbf{Z}^n$, то в соответствии с условием инъективности отображения алгоритма, граф структуры ВС и моменты выполнения операторов должны быть представлены в подпространствах $\mathbf{Z}_S$ и $\mathbf{Z}_T$, прямая сумма которых равна исходному пространству $\mathbf{Z}^n = \mathbf{Z}_S \oplus \mathbf{Z}_T$. Для того, чтобы упростить процесс выделения этих подпространств, целесообразно выбирать их с ортонормированными базисами, которые совпадают с осями ординат, т.е. $\mathbf{Z}_S = \mathbf{Z}^m$ и $\mathbf{Z}_T = \mathbf{Z}^{n-m}$.

*Конфигурацией структуры* (КС) является тройка $C_S = (K_S, D_S, A)$, где $K_S = (\boldsymbol{K}_{S1},...,\boldsymbol{K}_{SN})$, и $D_S = (\boldsymbol{D}_{S1}, ..., \boldsymbol{D}_{S(M+1)})$ – матрицы координат векторов-вершин $\boldsymbol{K}_{Si} = (k_{Si,1},...,k_{Si,m})^{\mathrm{T}} \in \mathbf{Z}^m$ и векторов-дуг $\boldsymbol{D}_{Sj} = (d_{Sj,1},...,d_{Sj,m}) \in \mathbf{Z}^m$, которые равняются координатам ПЭ, в которых выполняется $i$-й оператор и относительным координатам линий связи структуры, по которой передается $j$-я переменная, соответственно; $A$ – матрица инцидентности ГСПД.

В простом случае, $\boldsymbol{K}_{Si} = (s_i, p_i)^{\mathrm{T}} \in \mathbf{Z}^2$.

*Конфигурацией предшествования* (КП) является тройка $C_T = (K_T, D_T, A)$, где $K_T = (\boldsymbol{K}_{T1},...,\boldsymbol{K}_{TN})$, и $D_T = (\boldsymbol{D}_{T1}, ..., \boldsymbol{D}_{T(M+1)})$ –матрицы векторов $\boldsymbol{K}_{Ti}=(k_{Ti,1},...,k_{Ti,(n-m)})^{\mathrm{T}} \in \mathbf{Z}^{n-m}$, $\boldsymbol{D}_{Tj} =(d_{Tj,1},...,d_{Tj,(n-m)})^{\mathrm{T}} \in \mathbf{Z}^{n-m}$, которые отвечают моментам срабатывания $i$-го оператора и относительной задержке передачи $j$-й переменной, $A$ – матрица инцидентности ГСПД или ГЗД.

В простейшем случае, $\boldsymbol{K}_{Ti} = (t_i) \in \mathbf{Z}$. Тогда матрице $K_T$ отвечает *вектор временной развертки* $\boldsymbol{T} = (t_1,...,t_N)^{\mathrm{T}}$, который указывает, в какие моменты времени срабатывают определенные операторные вершины.

На рис.2 показаны пример КА (а) и соответствующих ей КС (б) и КП (в).

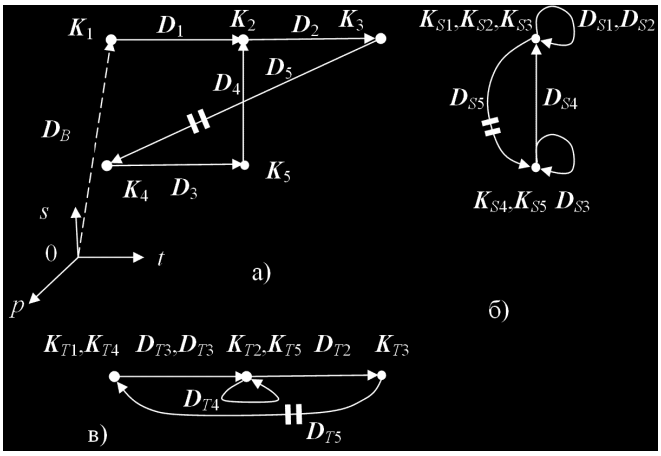



***Рис.2. Пример КА и соответствующих ей КС и КП***

КА является *корректной*, если в матрице $K$ нет двух одинаковых векторов, т.е.

$$\forall\ \boldsymbol{K}_{\mathrm{i}},\boldsymbol{K}_{\mathrm{j}}\ (\boldsymbol{K}_{\mathrm{i}} \neq \boldsymbol{K}_{\mathrm{j}},\ i \neq j\ ). \qquad (2)$$

Если в ГСПД есть циклы, то должна быть равна нулю сумма векторов-дуг $\boldsymbol{D}_{\mathrm{j}}$, входящих в любой из циклов графа, т.е. для $i$-го цикла

$$\sum b_{\mathrm{i,j}}\, \boldsymbol{D}_{\mathrm{j}} = 0, \qquad (3)$$

где $b_{\mathrm{i,j}}$ – элемент $i$-й строки цикломатической матрицы ГСПД.

КП $C_T$ для ГСПД – (*строго*) *корректна* относительно временной функции $R$, если

$$\boldsymbol{K}_{Tr} - \boldsymbol{K}_{Ti} = \boldsymbol{D}_{Tj} \in D_T \Rightarrow$$

$$\begin{cases} R\,(\boldsymbol{K}_{Tr}) \geq R\,(\boldsymbol{K}_{Ti});\ \text{– для ненагруженных дуг} \\ R(\boldsymbol{K}_{Tr}) < R(\boldsymbol{K}_{Ti}).\ \text{– для дуг, нагруженных задержками} \end{cases} \qquad (4)$$

$$i,r = 1,...,N;\ j = 1,...,M+1;$$

(для строгой корректности – неравенство строгое), т.е. если вершины связаны дугой, то момент срабатывания оператора предшествующей вершины всегда меньше момента срабатывания следующей. На рис.2 дуга $\boldsymbol{D}_5$ нагружена двумя задержками, что означает что в ней операнд задерживается на 2 итерации.

*Расписание* для КА $C_A$, выполняемое с периодом $L$ тактов – *корректно*, если операторы, отображаемые в один и тот же ПЭ, выполняются в различных тактах, т.е.

$$\forall \boldsymbol{K}_i, \boldsymbol{K}_j \in K^{\upsilon}(k_{i,r} = k_{j,r};\ r=1,\ldots,n-1) \Rightarrow$$

$$k_{i,n} \equiv k_{j,n} \bmod L - (\boldsymbol{K}_i, \boldsymbol{K}_j) - \quad (5)$$

дуга, нагруженная задержками

$$k_{i,n} \not\equiv k_{j,n} \bmod L$$ – в остальных случаях

КА линейным отображением своих компонентов может быть преобразованная в КС и КП, так что:

$$K = \begin{pmatrix} K_S \\ K_T \end{pmatrix};\ D = \begin{pmatrix} D_S \\ D_T \end{pmatrix}, \quad (6)$$

если КА, КП и расписание являются корректными, т.е. удовлетворяют условиям (2) – (5), причем полученная модель ВС будет выполнять заданный алгоритм в конвейерном режиме с периодом $L$ тактов.

Однотипные операторы следует отображать в ПЭ того же типа, т.е.

$$\boldsymbol{K}_i, \boldsymbol{K}_j \in K_{p,q}(k_i = k_j = p,\ s_i = s_j = q),\ |K_{p,q}| \le L,$$

где $K_{p,q}$ – множество векторов-вершин операторов $p$-го типа, отображаемых в $q$-й ПЭ $p$-го типа ($q$=1,2,… $q^{p}_{max}$).

Эффективную КА ищут в два этапа. На первом этапе вершины ГСПД вместе с дугами размещаются в трехмерном пространстве как множества векторов $\boldsymbol{K}_i$ и $\boldsymbol{D}_j$ с учетом условий, приведенных выше, т.е. формируется начальная КА. Минимизируется число ПЭ в искомой структуре путем выполнения требования $|K_{p,q}| \to L$, т.е. число вершин, отображаемых в один ПЭ, стремится к $L$. Также возможна перестановка вершин относительно оси времени *ot*, которая отвечает ресинхронизации ГСПД.

На втором этапе КА уравновешивается. При этом рассматривается ациклический подграф ГСПД без обратных дуг $\boldsymbol{D}_{Bj}$. Во все его дуги включаются промежуточные вершины операторов задержки (регистров). В уравновешенной КА все векторы-дуги, кроме нагруженных дуг, равны $\boldsymbol{D}_j = \langle a_j, b_j, 1\rangle$ или $\boldsymbol{D}_j = \langle a_j, b_j, 0\rangle$. При этом вершины-операторы образуют ярусы, расстояние между которыми по координате времени *ot* равно 1 такт. Уравновешенная КА оптимизируется путем взаимных перестановок векторов-вершин, принадлежащих одному ярусу с минимизацией числа регистров и входов мультиплексоров. Применяются и другие стратегии, например, алгоритм левого края.

Структуру ВС и расписание выполнения алгоритма можно получить, расщепив КА $K_G$ на КС $K_S$ и КП, которые имеют ту же матрицу $A$, а векторы матрицы $K_S$ координат ПЭ и матрицы моментов срабатывания $K_T$ равны координатам векторов матрицы $K$, т.е. $\langle k_i, s_i\rangle$ и $\langle t_i\rangle$, т.е. согласно (6). КС и КП представляют собой отображение КА в подпространства структур и времени, которое выполняется элементарным образом.

Данный метод позволяет целенаправленно выполнять поиск как структуры ВС, так и расписания выполнения алгоритма в ней. В [7] метод усовершенствован с целью минимизации аппаратных затрат ПЛИС, в которой реализуется конвейерная ВС. При этом благодаря тому, что результатом отображения выступает описание ВС на языке VHDL, метод дает возможность не строить собственно структуру ВС и расписание выполнения алгоритма, а переложить это задание на компилятор-синтезатор проекта для ПЛИС.

Генетический метод оптимизации систем характеризуется высокой эффективностью и независимостью от предметной области. Однако его использование ограничено выбором кодирования генома варианта решения. Так как в КА в компактной форме закодирован некоторый корректный вариант реализации ВС, она может эффективно использоваться как геном при синтезе структур генетическим методом.

Пространственный динамический ГПД с эффективным поведением отображается в конвейерную ВС методом, предложенным в [8]. Этот метод позволяет формально отображать алгоритмы с операторами управления в структуру конвейерных ВС с заданным периодом вычислений, которые имеют минимизированные аппаратные затраты и высокую тактовую частоту. Методы проверены при синтезе ряда конвейерных устройств ЦОС, таких как цифровые фильтры, процессоры быстрого преобразования Фурье, дискретного косинусного преобразования и многих других, которые имеют оптимизированное отношение производительность – аппаратные затраты.

Методы также эффективно использовались для программирования многопроцессорных ВС [9], а также такой SIMD-архитектуры, как Intel MMX [10].

## Выводы

Граф потоков данных (ГПД) – это естественная модель для задания алгоритмов обработки таких потоков. Динамические ГПД имеют широкие возможности для задания алгоритмов, но их анализ усложнен и они могут иметь блокировки. Из-за этого они используются, в основном, в системах с динамическим расписанием и не приспособлены для синтеза конвейерных вычислительных систем.

Конвейерные ВС следует проектировать путем отображения графов синхронных потоков данных (ГСПД) или ГПД с эффективным поведением и квазистатических ГПД, которые имеют ряд свойств, таких же, как у ГСПД.

Меньшая выразительность и большая трудоемкость представления алгоритма на модели однородного ГСПД компенсируется тем, что при представлении такого ГСПД в многомерном пространстве в виде пространственного ГСПД его отображение в конвейерную структуру выполняется формально с получением минимизированных аппаратных затрат при заданной производительности.

Пространственный ГСПД позволяет применять различные известные методы комбинаторной оптимизации ВС, начиная с метода ветвей и гнаниц и кончая генетическими алгоритмами. При этом оптимизация становится направленной как на минимизацию аппаратных затрат, так и на повышение отношения производительность – стоимость.

## Список литературы


1. Lee E. A. The Problem with Threads // IEEE Computer. –2006. —V39. –№5. –P.33-42.
2. Котов В.Е. Введение в теорию схем программ. –Новосибирск: Наука. —1978. –258 с.
3. Bhattacharyya S.S., Leupers R. Marwedel P. Software Synthesis and Code Generation for Signal Processing Systems // IEEE Trans. on Circuits and Systems, Part II, Analog and Digital Signal Processing. –2000. –V.47. –№ 9. –p. 849–875.
4. Сергиенко А.М., Симоненко В.П. Алгоритмические модели обработки потоков данных // Электрон. моделирование.–2008. – Т.30, № 6.–С. 49–60.
5. Каневский Ю.С. Систолические процессоры. –Киев:Техніка. —1991. —172 с.
6. Каневский Ю.С., Овраменко С.Г., Сергиенко А.М. Отображение регулярных алгоритмов в структуры специализированных процессоров //Электрон. моделирование.–2002.–Т.24.–№2.–С. 46–59.
7. Симоненко В.П., Сергиенко А.М. Отображение периодических алгоритмов в программируемые логические интегральные схемы //Электрон. моделирование. – 2007. – Т.29. №2. –С.49–61.
8. Сергиенко А.М. Синтез структур для выполнения периодических алгоритмов с операторами управления // Вістн. Націон. Техн. Університету України „КПІ”. Сер. Інформатика, управління і обчислювальна техніка. —2008. №47. —С.62 —68.
9. Сергиенко А.М. Отображение алгоритма QR-разложения Гивенса в многопроцессорную систему // Электрон. моделирование. – 2004. – Т.26. –№5. – С.43–53.
10. Sergiyenko A., Kaniewski J., Arefjev A., Kortshev D. A Method for Mapping DSP Algorithms into Pentium MMX™ Architecture. //Proc 3-d Int. Conf. On Parallel Processing and Applied Mathematics, PPAM'99. –Kazimierz Dolny, Poland. –Sept. 14-17. –1999. –p.348–356.


Поступила в редакцию 14.12.2009

УДК 004.94

*ДЕМЧИНСКИЙ В.В.,*
*ДОРОГОЙ Я.Ю.,*
*ДОРОШЕНКО Е.С.*

# ВИРТУАЛИЗАЦИЯ СЕТЕЙ ПЕРЕДАЧИ ДАННЫХ В DYNAMIPS

В статье рассмотрены основные аспекты особенностей методов виртуализации на базе программного обеспечения Dynamips/Dynagen/GNS3, полученные в ходе практических исследований, как подходов моделирования, так и вариантов построения сложных конвергентных корпоративных сетей на базе оборудования Cisco/Juniper.

The article describes the main aspects of virtualization features of the methods based on software Dynamips/Dynagen/GNS3, obtained during the practical research as modeling approaches and options for building a complex converged enterprise networks based on equipment of Cisco / Juniper.

## Введение в проблему

Виртуализация отдельного сетевого оборудования и целых сетевых топологий находит свое применение не только в учебных целях, но и в задачах проектирования крупномасштабных сетей, например в случае капитального строительства современных бизнес центров или расчета затрат для провайдера услуг связи. При этом не требуется физического наличия весьма дорогостоящего оборудования, а виртуальная природа системы позволяет легко менять топологию связей и выбирать оборудование с необходимой функциональностью.

С развитием технологий виртуализации, в последние годы появилась программная эмуляция сетевого оборудования Cisco (маршрутизаторы 1700, 2600, 3600, 3700, 7200 серий, PIX, ASA, IDS и тд.) и Juniper (серия J).

## Анализ существующих решений

Сейчас существуют такие пакеты для проектирования виртуальных корпоративных сетей:

**Opnet** – мощный пакет моделирования, позволяющий промоделировать основные аспекты поведения сети при использовании широкого круга технологий.

**Boson NetSim** – еще один пакет моделирования, позволяющий моделировать сети, построенные на оборудовании Cisco. Данный пакет имеет достаточно широкий диапазон поддерживаемых технологий и является достаточно удобным для изучения оборудования на базовом уровне.

**Cisco Packet Tracer** – бурно развивающийся современный пакет моделирования сетей на базе оборудования Cisco. Позволяет моделировать виртуальные сети средней сложности, изучать аспекты проходящих в сети процессов обмена трафика.

Общими недостатками для всех рассмотренных пакетов, является:

- невозможность совмещения реальной сети с виртуализированной;
- медленное пополнение набора доступных технологий для изучения;
- полное отсутствие возможности распределить смоделированную сеть на несколько ПК с целью увеличения ее масштаба и уменьшения ресурсной нагрузки на один ПК.

Все эти проблемы решены в пакете Dynamips в связке с Dynagen и GNS3.

## Скрытые проблемы виртуализации

Как известно, за счет виртуализации можно существенно уменьшить количество оборудования и необходимую площадь датацентра, а также существенно снизить стоимость необходимого оборудования. Однако при всех известных преимуществах виртуализации после ее реализации возникает целый ряд специфичных проблем, о которых поначалу иной раз просто не задумываются.

*Проблема 1.* На невиртуализованной платформе усредненный серверный процессор работает с нагрузкой 10–15%. В случае виртуализации этот показатель возрастает примерно до 70–80%.

*Проблема 2.* При виртуализации падает скорость обмена информацией между устройствами.

*Проблема 3.* Не все можно виртуализировать.

## Цель работы

В данной статье будут рассмотрены аспекты применения Dynamips [1] в задачах моделирования, наряду с возникающими в процессе моделирования проблемами и возможными их решениями.

## Виртуализация при помощи Dynamips

Поскольку продукт написан на Python, это позволяет переносить его на любую платформу, для которой существует интерпретатор языка.

Естественно, что сетевая производительность виртуализированного в Dynamips оборудования отстает от реального в сотни раз, что ограничивает его применение. Та же причина затрудняет использование эмулятора для исследования механизмов, требующих realtime-производительности при высокой нагрузке (например, QoS и VoIP).

Однако, можно констатировать, что к 2008 году продукты Dynamips достигли зрелого состояния, достаточного для применения в практических целях.

Наряду с ранее перечисленными сериями устройств, Dynamips эмулирует около 20 разнообразных сетевых модулей (NM) и WIC – карт.

Следует сказать, что эмуляция коммутационного оборудования (коммутаторы Ethernet, Frame Relay, ATM) в моделирующей среде почти отсутствует. Это объясняется закрытым характером архитектур специальных плат коммутации (ASIC) фирмы Cisco. Исключением является базовая коммутация второго уровня. Больший набор функций коммутаторов LAN обеспечивает эмулируемый Dynamips сетевой модуль **NM-16 ESW**.

Остальные виды коммутации потоков пакетов между маршрутизаторами обеспечиваются самой средой (фактически, без эмуляции коммутаторов). Поэтому Dynamips обеспечивает основные возможности коммутаторов Ethernet (с VLAN), Frame Relay, ATM. Возможность захвата потока пакетов с одного или нескольких каналов топологии будет полезна при подробном исследовании сетевых протоколов.

Второй продукт из связки (Dynagen) [2] позволяет экспортировать / импортировать как настройки одного виртуального устройства, так и всей топологии в целом (настройки топологии плюс настройки маршрутизаторов в виде одного файла).

При перегрузке физического процессора моделирующего компьютера могут наблюдаться эффекты, аналогичные перегрузкам физического оборудования или аппаратным проблемам (эмуляция аппаратных ошибок) – interface flapping (перманентный переход интерфейса между включенным и выключенным состоянием, приводящий к сбросу логических соединений между соседними маршрутизаторами и повторному установлению соединения протоколом маршрутизации и последующему пересчету маршрутной информации, т.е. к нестабильности канала и усугублению перегрузки эмулируемого процессора). Другое следствие перегрузки физического процессора – нарушение пакетов, принимаемых виртуальными устройствами (не совпадение контрольной суммы).

При использовании Dynamips/Dynagen полезным будет придерживаться нескольких рекомендаций:

- образы IOS обязательно нужно распаковать перед использованием – это уменьшит нагрузку на физический процессор;
- использовать idlepc – параметр, обеспечивающий оптимальную загрузку процессора путем вычисления среднего времени простоя и функционирования;
- использовать ghostios – параметр, использование которого заставит эмулятор хранить в оперативной памяти только одну копию IOS (для каждого используемого feature set );
- использовать sparsemem – параметр, позволяющий оптимизировать использование физической памяти путем выделения количества памяти только в размере, который используется операционной системой устройства. В случае, когда данный параметр отключен для каждого устройства выделяется столько памяти, сколько указано в настройках. Зачастую такого рода распределение является не оптимальным.

Количество оперативной памяти физического компьютера должно позволять полностью разместить данные памяти всех виртуальных устройств, поскольку использование

подкачки критически замедляет производительность моделирующей системы. Таким образом, требуется не допустить вытеснения страниц памяти, содержащей данные виртуальных устройств.

Использование библиотек libcap (Linux) и winpcap (Windows) позволяет связать модель сети в Dynamips с другими хостами физической (или аналогичной виртуальной) сети т.е. "подключать" один из ethernet-портов виртуальных устройств к физической сети. Это свойство необходимо при использовании распределенных топологий моделируемых сетей (части одной топологии выполняются на нескольких Dynamips-серверах), а также при подключении к виртуальным маршрутизаторам других физических или виртуальных (VMWare, Virtual PC, Virtual Box) компьютеров или внешнего ПО (в частности, SDM – Security Device Manager – web - утилита администрирования маршрутизаторов), а также реального оборудования CISCO.

Однако только один из портов эмулируемых маршрутизаторов может быть подключен к физическому сетевому порту компьютера. Это ограничивает построение распределенных сетевых топологий на нескольких Dynamips-серверах. Естественно, можно задействовать несколько сетевых карт на физическом компьютере. Однако существует более простое решение – подключить все порты, которые необходимо вывести во внешнюю сеть, к одному виртуальному коммутатору, а затем еще один порт данного коммутатора подключить во внешнюю сеть. Аналогично подключаются порты в других частях распределенной топологии.

Третий пакет из связки (GNS3) [3] представляет собой средство для построения топологии виртуальной сети при помощи интуитивно-понятного дружелюбного графического интерфейса. Работа связки этих программных пакетов выглядит приблизительно так:

- при помощи пакета GNS3 создается топология виртуализированной сети;
- GNS3 создает по топологии специальную подпрограмму «скрипт», который автоматически выполняется в Dynagen с использованием Dynamips-сервера.

Для полноценной работы набора также необходимы образы операционных систем используемого оборудования.

## Эксперименты и результаты

С целью исследования работы данного набора была собрана следующая топология сети (рис. 1).

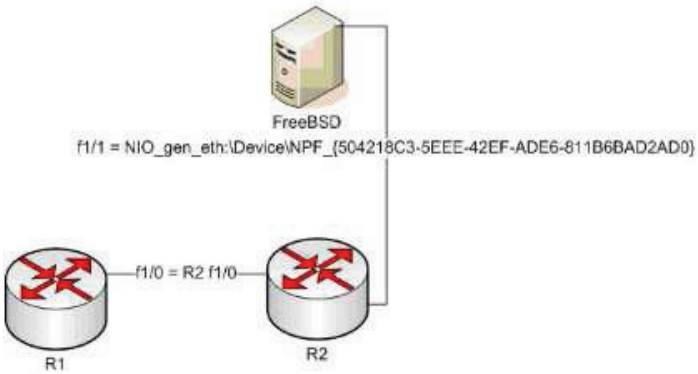



***Рис. 1. Топология виртуализированной сети***

Устройства R1 и R2 представляют собой два маршрутизатора Cisco 3640, причем эмуляция каждого из них запущена на разных ПК. Сервер FreeBSD является реальным физическим устройством.

В результате эксперимента была построена сеть с использованием виртуальных и физических устройств. Поведение смоделированной сети полностью соответствует ожидаемому (т.е, можно передавать трафик между устройствами, настраивать политики доступа и тд).

## Выводы

Таким образом, средства Dynamips / Dynagen / GNS3 обеспечивают широчайший набор возможностей для изучения сетевого оборудования и технологий в учебных целях и при моделировании реальных систем, например, на производственных площадках, где для проверки поведения сети в определенных ситуациях необходимо иметь возможность собрать недорогой макет.

## Список литературы

1. http://www.ipflow.utc.fr/index.php/Cisco_7200_Simulator
2. http://dynagen.org
3. http://gns3.org

Поступила в редакцию 16.12.2009

УДК 004.75

*ТЕЛЕНИК С.Ф.,*
*РОЛІК О.І.,*
*БУКАСОВ М.М.,*
*ЛАБУНСЬКИЙ А.Ю.*

# МОДЕЛІ УПРАВЛІННЯ ВІРТУАЛЬНИМИ МАШИНАМИ ПРИ СЕРВЕРНІЙ ВІРТУАЛІЗАЦІЇ

В статті проводиться аналіз методів управління розподілом ресурсів при серверній віртуалізації. Розроблені моделі та методи управління ресурсами для випадку живої міграції віртуальних машин. Для здійснення управління реалізовано метод гілок і меж. Наводяться опис та результати експериментальних досліджень.

In the article the analysis of methods of resource management for server virtualization is realized. The models and methods of resource management for case of virtual machines live migration are developed. Implementation of the branch and bound algorithm for realization of management is proposed. Description and results of experiments are described.

## Вступ

Останнім часом при створенні інформаційних систем спостерігається тенденція підвищення вимог до якості і надійності функціонування сервісів інформаційно-телекомунікаційної інфраструктури, як основи ведення бізнесу. Найпростішим вирішенням цього питання протягом тривалого часу було розміщення найбільш критичних застосувань на виділених серверах чи кластерах [1]. Таке рішення на порядок підвищувало надійність функціонування інформаційної системи, але в той же час значно збільшувало сукупну вартість володіння серверною складовою інформаційної інфраструктури за рахунок вартості додаткових серверів та ліцензій на комерційне програмне забезпечення, витрат на електроживлення, кондиціювання, адміністрування, обслуговування тощо.

За статистичними даними компанії VMware, зазвичай завантаження виділених під окремі задачі серверів не перевищує 15 – 20 % від їх повної потужності. Тому в останні роки стало популярним скорочення кількості фізичних серверів для підтримки різноманітних сервісів шляхом впровадження технології віртуалізації серверів. Такий підхід за рахунок більшого завантаження серверів дозволяє не тільки зменшити їх кількість, але й скоротити витрати на їх утримання [2]. До того ж наявність функції «живої міграції» віртуальних машин (ВМ) між фізичними серверами дозволяє отримати показники готовності сервісів, близькі до кластерних рішень, при значно менших витратах.

Тому дуже актуальною стає задача управління розподілом ресурсів між ВМ та розташування ВМ по серверам таким чином, щоб при обмежених обчислювальних ресурсах у першу чергу забезпечити функціонування застосувань, пов’язаних з найбільш важливими бізнес-процесами, навіть за рахунок менш важливих [3], а при надлишку ресурсів розташувати ВМ таким чином, щоб вивільнити сервери з метою економії енергоспоживання. Реалізація подібних систем управління розподілом ресурсів потребує розроблення відповідних моделей та методів чи адаптації вже відомих [4] з урахуванням віртуалізації серверів та «живої міграції» ВМ.

У праці [3] запропоновано ресурсний підхід до управління IT-інфраструктурою підприємства. Пропонується розглядати діяльність підприємства як сукупність бізнес-процесів, що споживають визначену кількість ресурсів певних типів. Розглянуто дискретний випадок, коли окремі бізнес-процеси підтримуються в повному обсязі чи не підтримуються зовсім, та неперервний випадок, коли можлива часткова підтримка бізнес-процесів.

Цей підхід розвинений у праці [4], де крім ресурсних обмежень додавались обмеження по надійності підтримки бізнес-процесів.

У праці [5] запропоновані моделі і методи управління IT-інфраструктурою на основі ресурсного підходу на всіх етапах життєвого

циклу – від планування при створенні до експлуатації та подальшого розвитку.

У праці [6] розглянуто питання управління ресурсами, застосуваннями та навантаженням для випадку віртуалізації серверів, але без можливості «живої міграції» ВМ.

Питання управління ресурсами IT-інфраструктури розглянуті досить детально, але використання для цього «живої міграції» ВМ, можливість якої в більшості системах віртуалізації з'явилася не так давно, все ще потребує додаткових досліджень.

## Мета дослідження

Метою статті є розробка моделей і методів управління розподілом ресурсів IT-інфраструктури з використанням «живої міграції», при якому забезпечується як підвищення доступності найважливіших сервісів в умовах нестачі обчислювальних ресурсів, так і економія енергоспоживання в умовах їх надлишку.

## Моделі та методи управління ресурсами при живій міграції

Нехай є декілька фізичних серверів $S_i$, $i = 1, ..., n$, на яких під управлінням гіпервізорів функціонують ВМ $V_j, j = 1, ..., m$ (рис. 1).

Кожна з ВМ в залежності від потоку клієнтських запитів використовує певну кількість ресурсів типу $R_k$, $k = 1, ..., l$ (пам'ять, процесорний час, дисковий простір, пропускна спроможність підсистеми вводу-виводу, зовнішніх каналів зв'язку тощо).

Введемо необхідні для побудови моделей позначення:

$r_{ki}$ – кількість ресурсу типу $R_k$, що встановлена на сервері $S_i$;

$p_{kj}$ – потреби ВМ $V_j$ у ресурсах типу $R_k$ для забезпечення заданого рівня SLA;

$x_{ij}$ – булева змінна, яка визначає, чи встановлена ВМ $V_j$ на сервері $S_i$.

Оскільки кожна ВМ одночасно розташована не більше, ніж на одному сервері, має виконуватись умова:

$$\sum_{i=1}^{n} x_{ij} \le 1,\ j = 1,...,m. \qquad (1)$$

До того ж, для нормального функціонування ВМ повинні бути забезпечені достатнім обсягом ресурсів серверів, на яких вони розташовані:

$$\sum_{j=1}^{m} x_{ij} p_{kj} \le r_{ki},\ k = 1,...,l;\ i = 1,...,n. \qquad (2)$$

Розглянемо дві задачі розташування ВМ для випадків нестачі та надлишку ресурсів.

**Задача 1**. У разі, якщо внаслідок збільшення клієнтських запитів потреба окремих ВМ у ресурсах певного типу збільшилась настільки, що стає неможливим забезпечити усі ВМ необхідною кількістю ресурсів, природним виходом стає задача підтримки найбільш важливих бізнес-процесів шляхом забезпечення ресурсами тих ВМ, на яких працюють пов'язані з ними сервіси, за рахунок менш важливих.

Позначимо через $w_j$, $j = 1, ..., m$, важливість застосувань, встановлених на ВМ $V_j$.

Тоді, задачу можна сформулювати таким чином:

$$\sum_{j=1}^{m} \sum_{i=1}^{n} x_{ij} w_j \to \max, \qquad (3)$$

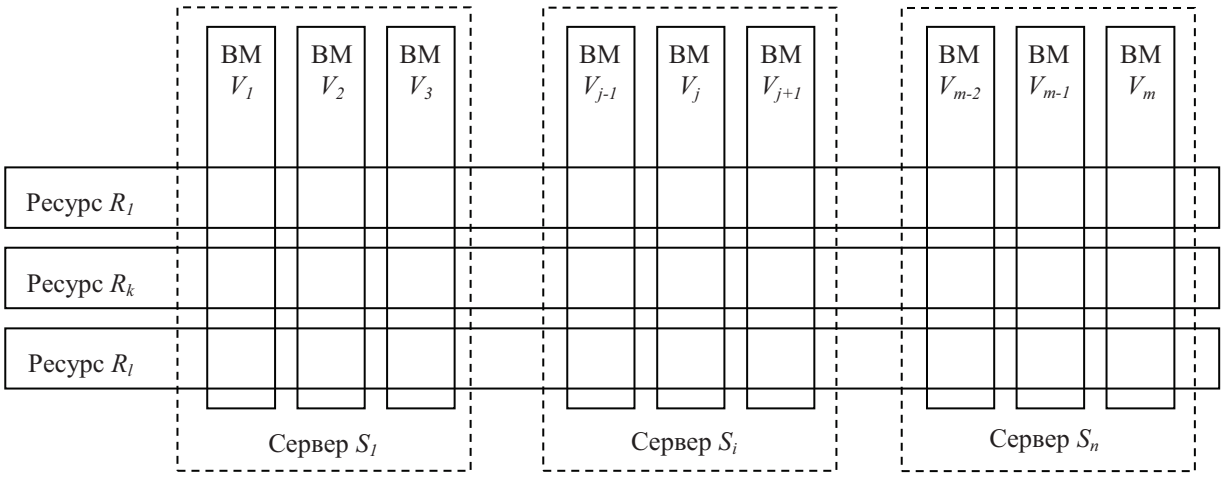



***Рис. 1. Розподіл ресурсів серверів між ВМ***

при обмеженнях (1), (2).

Ця задача становить собою лінійну задачу булевого програмування. Для її вирішення скористаємося методом гілок та меж. Але спочатку покажемо, як можна, враховуючи специфіку задачі, зменшити початкову кількість можливих комбінацій, яка складає $2^{mn}$.

По-перше, згідно з обмеженням (1), кожна ВМ може бути розташована не більше, ніж на одному сервері, тобто замість перебору $2^n$ комбінацій для кожної ВМ $V_j$, можна обмежитись розглядом лише $n$+1 варіантів: $y_j$=0, ..., $n$, де $y_j$ – номер сервера, на якому встановлена ВМ $V_j$ ($y_j$=0 у випадку, якщо внаслідок нестачі ресурсів ВМ $V_j$ не встановлена на жодному сервері або ресурси для неї виділяються по залишковому принципу).

По-друге, враховуючи те, що вимоги ВМ у ресурсах $p_{ij}$ є невід'ємними, якщо на якомусь з етапів побудови дерева варіантів розміщення ВМ по серверам одне з обмежень (2) перестає виконуватись, продовжувати побудову цієї гілки немає сенсу, оскільки для усіх її вузлів це обмеження також не буде виконуватись.

По-третє, слід врахувати те, що дуже часто при побудові серверних ферм використовують сервери з однаковою конфігурацією. Якщо, наприклад, є два ідентичних сервера, кількість можливих варіантів розміщення ВМ можна скоротити в 2 рази, оскільки перенесення усіх ВМ з першого сервера на другий, а з другого на перший ніяк не вплине на об'єм ресурсів, які використовуються. Тобто перестановка рядків у матриці $X = ||x_{ij}||$, які відповідають однаковим стовпцям у матриці $R = ||r_{ki}||$, ніяк не впливає ані на виконання обмежень (1), (2), ані на значення критерію (3). Іншими словами, для серверів з однаковою конфігурацією важливе не абсолютне розміщення ВМ на цих серверах, а їх розташування одна відносно іншої. При збільшенні кількості серверів з однаковою конфігурацією кількість «зайвих» комбінацій значно зростає від ($n$-1) для випадку розміщення усіх ВМ на одному сервері до ($n$!-1) при розміщенні ВМ на різних серверах.

Так, наприклад, для випадку трьох ВМ і трьох ідентичних серверів з можливих 27 комбінацій розташування ВМ лише 5 є унікальними, а решта – 22 комбінації – «дзеркальними» до них. Тобто, навіть для таких незначних значень $m$ і $n$ початкову множину варіантів можна скоротити більше, ніж у 5 разів.

Алгоритм перебору та його результати для цього випадку проілюстровані на рис. 2. Для наочності буквами («А», «В», «С») позначені ВМ, цифрами (1, 2, 3) – номера серверів, комбінацією букви з цифрою – розміщення ВМ на сервері. Поруч з кінцевими результатами на рисунку перелічені комбінації, які є «дзеркальними» до них, тобто призводять до таких же результатів і можуть бути виключені з перебору. Алгоритм перебору при побудові дерева розташування ВМ полягає у наступному: на черговому рівні чергової гілки кількість серверів, що розглядаються, для перебору приймається на 1 більше, ніж максимальна для попередніх рівнів цієї ж гілки, але не більше ніж $n$.

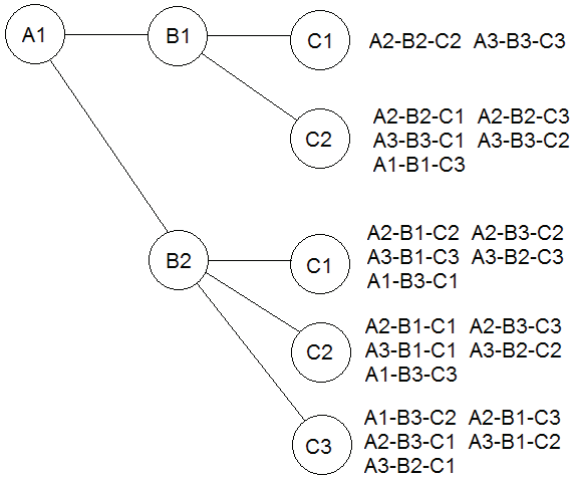



***Рис. 2. Схема перебору комбінацій для ідентичних серверів***

Так, на рівні «А» розглядається лише випадок розташування ВМ «А» на сервері №1, а на серверах №2 та №3 не розглядається, оскільки сервери №1-3 є повністю ідентичними і розташування ВМ «А» на них призведе до тих самих результатів. При розв'язанні задачі в реальних умовах, з метою зменшення кількості міграцій ВМ, при виборі довільного сервера обирається той, на якому ця ВМ вже встановлена. На рівні «В» розглядаються два варіанти «В1» – розміщення ВМ «В» на тому ж сервері, що і ВМ «А», та «В2» – розміщення ВМ «В» на іншому сервері, у цьому випадку №2. Варіант «В3» не розглядається, оскільки він нічим принципово не відрізняється від варіанту «В2» і призведе до тих самих результатів (аналогічно, якщо ВМ «В» уже встановлена на сервері №3, слід використовувати саме його, а не сервер №2).

Якщо у наявності є сервери з іншою конфігурацією, до схеми перебору додаються відповідні вузли на кожному рівні. Сюди також відносяться випадки, коли внаслідок нестачі ресурсів дозволяється низькопріоритетні ВМ не розміщувати на жодному сервері, чи виділяти ресурси для них за залишковим принципом. Формально це описується шляхом розміщення таких ВМ на неіснуючому сервері №0 (це можливо у задачі 1, неможливо у задачі 2). В цьому випадку на першому рівні буде вже декілька вузлів (рис. 3).

Крім визначення процедури розгалуження метод гілок і меж передбачає визначення процедур оцінки верхньої та нижньої меж за допомогою наближених, але швидких алгоритмів.

При вирішенні задачі 1 (задача максимізації) для визначення оцінки зверху можна використати оптимістичний прогноз, який полягає в припущенні того, що на черговому кроці алгоритму усі ВМ, не розподілені між серверами, можуть бути розподілені таким чином, що жодна з них не отримує відмови в обслуговуванні (не буде розміщена на неіснуючому сервері №0) за умови виконання ресурсних обмежень (2).

Щоб одержати оцінку знизу, пропонується використати жадібний алгоритм, упорядкувавши ВМ за зменшенням вимог до того ресурсу, нестача якого стала причиною переходу від старого до нового плану розміщення ВМ.

Процедура відсікання гілок полягає у тому, що гілка, для якої оцінка верхньої межі менша, ніж оцінка нижньої межі хоча б однієї з інших гілок, розглядається як безперспективна та виключається з дерева рішень. Враховуючи те, що вимоги ВМ до ресурсів є невід'ємними, ця процедура може бути доповнена відсіканням тих гілок, для яких припиняють виконуватись обмеження (2).

У разі, якщо на черговому кроці буде знайдене рішення, при якому усі ВМ успішно розміщені, тобто жодна ВМ не розмішена на сервері №0, пошук можна припинити, оскільки глобальний оптимум знайдений і критерій (3) набуває максимального значення. Але, за умови наявності часових можливостей, продовження пошуку може допомогти знайти рішення, що забезпечить меншу кількість міграцій ВМ при реалізації нового плану розташування, ніж уже знайдене рішення та його «дзеркальні» рішення.

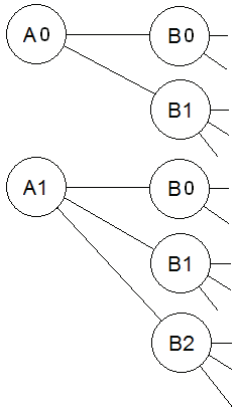



***Рис. 3. Схема перебору комбінацій для неідентичних серверів***

**Задача 2**. У разі, якщо кількість наявних обчислювальних ресурсів суттєво перевищує потреби ВМ, природною стає задача розподілити ВМ між серверами таким чином, щоб зменшити енергоспоживання за рахунок вимкнення живлення серверів, які не використовуються на цей час.

Оскільки відключати сервери для економії електроенергії має сенс лише у тому випадку, коли у наявності є вільні ресурси (випадок, коли економічний ефект від бізнес-процесів не перевищує енерговитрати на живлення серверів не розглядається), модифікуємо обмеження (1), щоб гарантувати, що кожна ВМ буде забезпечена ресурсами одного і лише одного сервера:

$$\sum_{i=1}^{n} x_{ij} = 1,\ j = 1,...,m\,. \qquad (4)$$

Позначимо через $e_i$ енергоспоживання сервера $S_i$ коли на ньому не працює жодної ВМ.

Ознаку того, що на сервері $S_i$ не працює жодної ВМ виразимо наступним чином:

$$d_i = \prod_{j=1}^{m} \overline{x_{ij}},\ j = 1,...,m\,. \qquad (5)$$

Тоді задачу мінімізації енергоспоживання можна сформулювати як задачу:

$$\sum_{i=1}^{n} d_i e_i \to max\,, \qquad (6)$$

при обмеженнях (2), (4), (5).

Для вирішення задачі 2 пропонується той же алгоритм, але зі спрощеною процедурою перебору, оскільки згідно з обмеженням (4) неможливі випадки, коли якась ВМ не розміщена на жодному сервері.

### Реалізація запропонованих методів

Зазначені моделі були апробовані для управління ВМ, які працювали під управлінням гіпервізора VMware ESX на двох серверах.

Схема макету наведена на рис. 4. Кожен з серверів мав два мережних інтерфейси: один – для взаємодії через мережу, другий – для передачі стану пам'яті ВМ у процесі міграції ВМ. «Жива міграція» здійснювалася за допомогою утиліти VMware vMotion, яка є складовою пакету VMware vSphere.

Управляюча програма, яка є частиною системи управління IT-інфраструктурою (СУІ) SmartBase ITS Control [7], була написана мовою C#.

Для взаємодії з програмним забезпеченням (ПЗ) VMware (моніторинг споживання ресурсів, управління розподілом ресурсів та міграцією ВМ) використовувалася бібліотека vSphere SDK for .NET.

На всіх ВМ була встановлена операційна система Linux. На ВМ «А» був встановлений веб-сервер Apache, для якого мовою PHP було створено ПЗ, яке генерувало динамічний контент у відповідь на запити, що надходили від клієнтів. На ВМ «В» та «С» були запущені обчислювальні процеси для створення постійного фонового завантаження ВМ.

У результаті проведених експериментів було виявлено, що за даних умов проведення експерименту з наявних типів ресурсів (процесор, пам'ять, пропускна здатність мережі, дискова підсистема) вузьким місцем в системі виявився процесор. Тому далі на графіках зображені лише завантаження процесорів, моніторинг яких здійснювався засобами VMware, та статистика обробки клієнтських запитів, яка збиралася та оброблялася за допомогою написаного ПЗ.

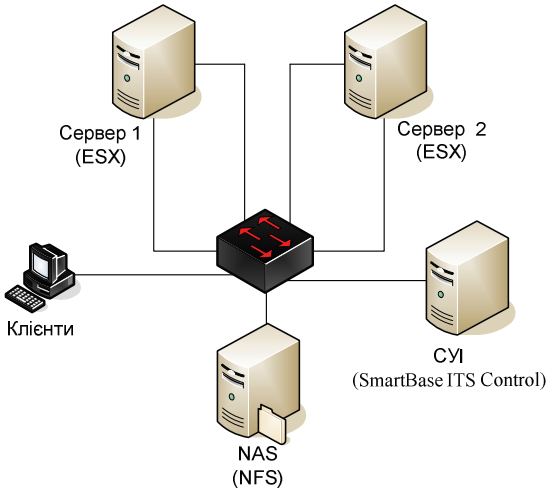



*Рис. 4. Схема макету*

Потік клієнтських запитів поступово збільшувався. На графіках (рис. 5–7) видно, що після того, як один з параметрів (у цьому експерименті – завантаження процесора на сервері 1), протягом заданого часу перевищував заданий поріг, СУІ прийняла рішення щодо необхідності перенесення ВМ «А» на сервер 2, що дозволило ліквідувати вузьке місце та відновити динаміку обробки запитів. Порогові значення та максимальна тривалість їх перевищення задається в конфігурації СУІ окремо для кожного з параметрів, виходячи з вимог, оговорених у SLA. У цьому експерименті рішення про необхідність міграції ВМ приймалось у разі, якщо середнє завантаження процесора перевищувало 90% протягом трьох хвилин.

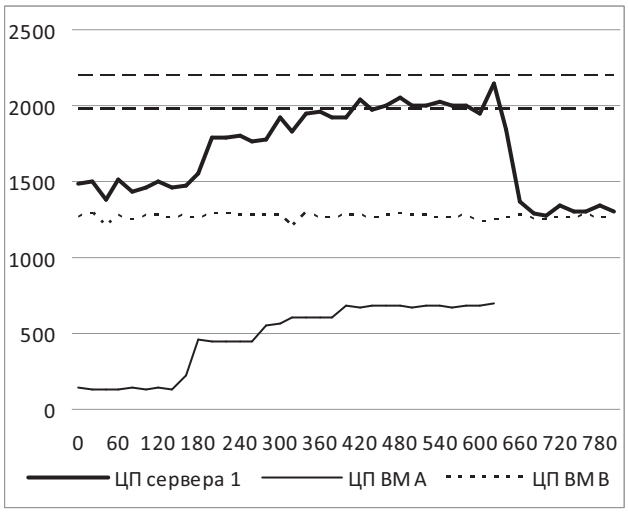



*Рис. 5. Завантаження ЦП сервера 1*

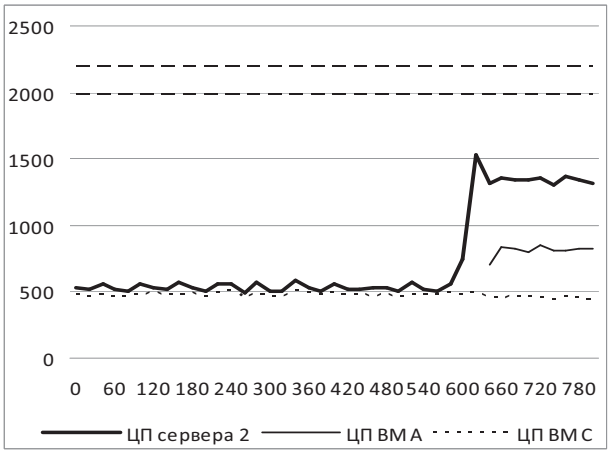



*Рис. 6. Завантаження ЦП сервера 2*

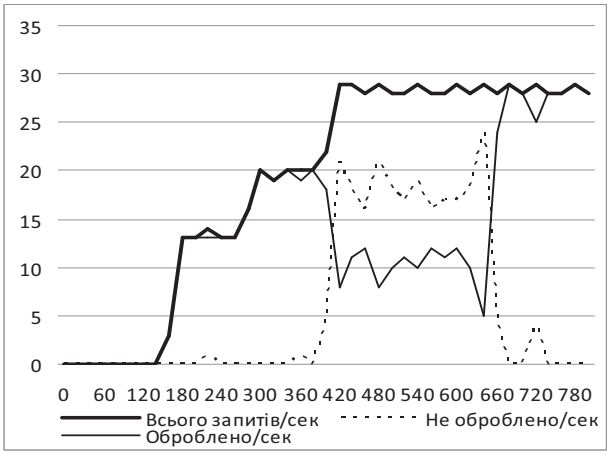



***Рис. 7. Надходження та обробка запитів***

## Висновки

У статті розглянуті питання управління розміщенням ВМ при серверній віртуалізації з можливістю здійснення «живої міграції» ВМ. Запропоновані моделі управління міграцією ВМ та розподілом ресурсів між ними для випадків нестачі та надлишку обчислювальних ресурсів. Ці моделі можуть бути застосованими для вирішення задач балансування навантаження між серверами шляхом міграції ВМ, для забезпечення високого рівня доступності найбільш важливих сервісів, та для зниження експлуатаційних витрат, а саме економії електроенергії шляхом відключення електроживлення окремих серверів з попереднім перенесенням ВМ з них на інші сервери. Для вирішення зазначених задач запропонована модифікація алгоритму гілок і меж. Запропоновані моделі і методи реалізовані в СУІ SmartBase ITS Control. Наведені результати виконаних експериментів, які підтвердили працездатність та ефективність зазначених моделей і методів.

## Список посилань

1. Теленик С.Ф. Управління навантаженням і ресурсами центрів оброблення даних при виділених серверах / С.Ф. Теленик, О.І. Ролік, М.М. Букасов, Р.В. Римар, К.О. Ролік // Автоматика. Автоматизація. Електротехнічні комплекси та системи. — 2009. — №2 (24). — С. 122—136.
1. David Marshall, Wade A. Reynolds, Dave McCrory. Advanced Server Virtualization: VMware and Microsoft Platforms in the Virtual Data Center. — Auerbach Publications, 2006. — 760 p.
2. Теленик С.Ф., Ролік О.І., Букасов М.М. Моделі управління розподілом обмежених ресурсів в інформаційно-телекомунікаційній мережі // Вісник НТУУ «КПІ». Інформатика, управління та обчислювальна техніка. — К.: Екотех. — 2006. — №44. — С. 243—246.
3. Теленик С.Ф., Ролік О.І., Терещенко П.І., Букасов М.М. Забезпечення процесів діяльності з визначеним рівнем надійності в ІТС спеціального призначення // Зб. наук. праць ВІТІ НТУУ «КПІ». — №3. — К., 2007. — С. 134—138.
4. Теленик С.Ф., Ролік О.І., Букасов М.М. Технологія управління ІТ-інфраструктурою на основі ресурсного підходу // Вісник ЖДТУ. — 2008.— №4(47). — С. 180—189.
5. Теленик С.Ф., Ролік О.І., Букасов М.М. Моделі і методи розподілу ресурсів в інформаційних системах з віртуалізацією серверів// Проблеми й перспективи розвитку ІТ-індустрії: Матеріали 1-й Міжнародної науково-практичної конференції. — Харків: ХНЕУ, 2009. — С. 46—47.
6. Теленик С.Ф., Ролік О.І., Букасов М.М., Соколовський Р.Л. Система управління інформаційно-телекомунікаційною системою корпоративної АСУ// Вісник НТУУ «КПІ». Інформатика, управління та обчислювальна техніка. — К.: «ВЕК+», — 2006. — № 45.— С. 112—126.

Поступила в редакцию 3.12.2009

УДК 681.3.07

*КВІТКО О.С.,*
*ДОРОШЕНКО К.С.,*
*ПОЛТОРАК В.П.*

# ДОСЛІДЖЕННЯ МЕТОДІВ ДИНАМІЧНОГО ПРОГРАМУВАННЯ У КОРПОРАТИВНИХ МЕРЕЖАХ

Розглянуто можливості використання методів динамічного програмування в мережах TCP/IP. Наведено результати експериментальних досліджень використання алгоритму динамічного програмування для вирішення задачі пошуку оптимального шляху (маршруту) в корпоративній мережі.

The article describes the usage of dynamic programming methods in TCP/IP networks. There are results of experimental using a dynamic programming algorithm in solving the optimal path (route) search on the corporate network.

## Постановка задачі в загальному вигляді та її актуальність

Сучасні корпоративні мережі, побудовані на IP-протоколах, використовуються не тільки для класичної передачі даних, але й для обміну звуковими даними, проведення мультимедійних конференцій, оперативного управління, прослуховування музичних записів, перегляду відео кліпів, мережевих ігор та інших прикладних програм реального часу. Тому все актуальнішою стає проблема пошуку оптимального шляху в процесі маршрутизації, та мінімізація ресурсів, необхідних для пошуку оптимального шляху. Дивлячись із цієї точки зору, дослідження методу моделювання процесу маршрутизації на основі методу динамічного програмування заслуговує на прискіпливішу увагу.

## Мета дослідження

Довести доцільність використання методу динамічного програмування для моделювання процесу пошуку оптимального шляху в мережі.

## Викладення основного матеріалу досліджень

Як відомо існує кілька параметрів оцінювання протоколів маршрутизації, серед них масштабування, швидкість обміну інформацією та ресурсоємність. Існує також параметр для оцінки самого алгоритму, який лежить в основі протоколу динамічної маршрутизації, що має назву О-нотація. Асимптотична нотація великого О, відома також як нотація Ландау, – розповсюджена математична нотація для формального запису асимптотичної поведінки функцій. Вона широко вживається в теорії складності обчислень, інформатиці та математиці.

Теорія складності обчислень – підрозділ теоретичної інформатики, який займається дослідженням складності алгоритмів для розв'язання задач на основі формально визначених моделей обчислювальних пристроїв. Складність алгоритмів вимірюється за необхідними ресурсами, в основному, це тривалість обчислень або необхідний об'єм пам'яті. В окремих випадках досліджуються інші міри складності, такі як розмір мікросхем, або кількість процесорів, необхідна для роботи паралельних алгоритмів.

Основна задача досліджень в теорії складності обчислень полягає у класифікації всіх розв'язуваних задач. Зокрема, робляться спроби відокремити множину задач з ефективними алгоритмами розв'язання від множини важко розв'язуваних задач.

Нехай A – алгоритм розв'язання деякого класу задач, а N – розмірність окремої задачі цього класу. N може бути, наприклад, розмірністю оброблюваного масиву, числом вершин оброблюваного графа тощо. Позначимо $f_A(N)$ функцію, що дає верхню межу максимального числа основних операцій (додавання, множення і т. д.), які повинен виконати алгоритм A, вирішуючи задачу розмірності N. Говоритимемо, що алгоритм A поліноміальний, якщо $f_A(N)$ зростає не швидше, ніж деякий поліном від N. В іншому разі A – експоненціальний алгоритм.

Виявляється, що між цими класами алгоритмів є істотна різниця: при великих розмі-

рностях задач (які, зазвичай, найцікавіші на практиці), поліноміальні алгоритми можуть бути виконані на сучасних комп'ютерах, тоді як експоненціальні алгоритми для практичних розмірностей задач, як правило, не можуть виконуватися повністю. Зазвичай розв'язання задач, що породжують експоненціальні алгоритми, пов'язано з повним перебором всіх можливих варіантів, і зважаючи на практичну неможливість реалізації такої стратегії, для їх розв'язання розробляються інші підходи.

Наприклад, якщо навіть існує експонентіальний алгоритм для знаходження оптимального розв'язку деякої задачі, то на практиці застосовуються інші, ефективніші поліноміальні алгоритми для знаходження не обов'язково оптимального, а лише прийнятного розв'язку (наближеного до оптимального).

Поліноміальні алгоритми також можуть істотно розрізнятися залежно від ступеня полінома, що апроксимує залежність $f_A(N)$. Розглянемо оцінювання часової складності алгоритму. Така оцінка проводиться із застосуванням згаданої вище О – нотації. Будемо говорити, що $f_A(N)$ зростає як $g(N)$ для великих N і записувати це $f_A(N)=O(g(N))$, якщо існує така додатна константа $a>0$, що $\lim_{N\to\infty} f_A(N)/g(N)=a$. Тоді оцінка $O(g(N))$ називається асимптотичною часовою складністю алгоритму.

Оцінка $O(g(N))$ для функції $f_A(N)$ застосовується, коли точне значення $f_A(N)$ невідоме, а відомий лише порядок зростання часу, затрачуваного на розв'язання задачі розмірністю N за допомогою алгоритму А. Точні значення $f_A(N)$ залежать від конкретної реалізації, тоді як $O(g(N))$ є характеристикою самого алгоритму.

В наступній таблиці наведено поширені асимптотичні складності, типові для задач управління мережами.

Для алгоритму Беллмана-Форда складність можна записати так O(m*n). Фактично це є часова складність алгоритму, що означає, що за час O(m*n) алгоритм знаходить найкоротші маршрути від однієї вершини до решти вершин. Щодо алгоритму Дейкстри, то його складність значною мірою залежить від способу реалізації цього алгоритму. В найпростішому випадку, коли для пошуку вершин переглядається вся підмножина вершин, а для збереження величин використовується масив, час роботи алгоритму буде $O(n^2 + m)$.

| **Складність** | **Коментар** | **Приклади** |
|---|---|---|
| O(1) | Сталий час роботи не залежно від розміру задачі | Очікуваний час пошуку в хеші |
| O(log n) | Логарифмічне зростання – подвоєння розміру задачі збільшує час роботи на сталу величину | Обчислення xn; двійковий пошук в масиві з n елементів |
| $O(n^2)$ | Квадратичне зростання – подвоєння розміру задачі вчетверо збільшує необхідний час | Елементарні алгоритми сортування |
| $O(n^3)$ | Кубічне зростання – подвоєння розміру задачі збільшує необхідний час у вісім разів | Звичайне множення матриць |
| O(cn) | Експоненціальне зростання – збільшення розміру задачі на 1 призводить до с-кратного збільшення необхідного часу; подвоєння розміру задачі підносить необхідний час у квадрат | Деякі задачі комівояжера, алгоритми пошуку повним перебором |

Щоб визначити функцію складності досліджуваного нами алгоритму динамічного програмування для початку необхідно проаналізувати цей алгоритм. Основною частиною роботи даного алгоритму є процедура розширення найкоротших шляхів, які відомі на даний момент, шляхом додавання в них по ще одному ребру. Вибір ребра, що додається, здійснюється на основі доведення, що всі часткові маршрути найкоротшого маршруту також вважаються найкоротшими маршрутами. Таким чином, для матриці розмірністю n*n мінімальна вага будь-якого шляху з вершини i в вершину j, що включає не більше m ребер визначається так:

$$l_{ij}^{(m)} = \min_{1\le k\le n}\left\{ l_{ik}^{(m-1)} + w_{kj} \right\},$$

де k – проміжна вершина, що входить в оптимальний маршрут, $w_{kj}$ – вага ребра, що веде від вершини k до кінцевої вершини j.

Оскільки алгоритм включає три вкладені цикли, доцільно говорити, що час його роботи $O(n^3)$. Після аналізу графіку функції (рис. 1),

що відповідає даному порядку складності, було прийнято рішення ввести в формулу складності деякий коефіцієнт q, що показуватиме зміну кількості ітерацій даного алгоритму в залежності від кількості ребер у графі.

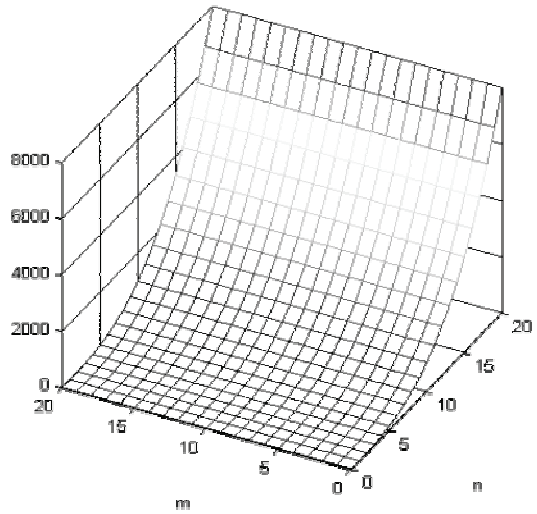



***Рис.1 Графік функції O(n³).***

Виразимо цей коефіцієнт відношенням $q = \frac{n}{m}$, тоді час роботи алгоритму $O(qn^3)$.

Після аналізу графіка функції, що показує складність алгоритму з урахуванням коефіцієнта q (рис. 2), можна помітити, що даний коефіцієнт може суттєво впливати на показник складності алгоритму. Якщо 0<q<1 очевидно, що показник складності алгоритму може істотно зменшуватись при $q \to 0$.

Відношення $\frac{n}{m}$ є показником, що характеризує ступінь заповненості матриці сполучень, інакше кажучи, дане відношення характеризує фізичну структуру досліджуваної мережі. Наприклад, якщо у мережі n вузлів, то мінімальна кількість ребер, необхідних щоб з'єднати ці вузли так, щоб до кожного вузла призначення було два маршрути, становить m=n.

В такому випадку значення $q = \frac{n}{m} = \frac{n}{n} = 1$. А максимальна кількість ребер, що дозволить організувати повністю зв'язану топологію (full-mesh topology), становить m=n(n-1)/2, що приведе до того, що відношення матиме вигляд $q = \frac{n}{m} = \frac{2n}{n(n-1)} = \frac{2}{n-1}$.

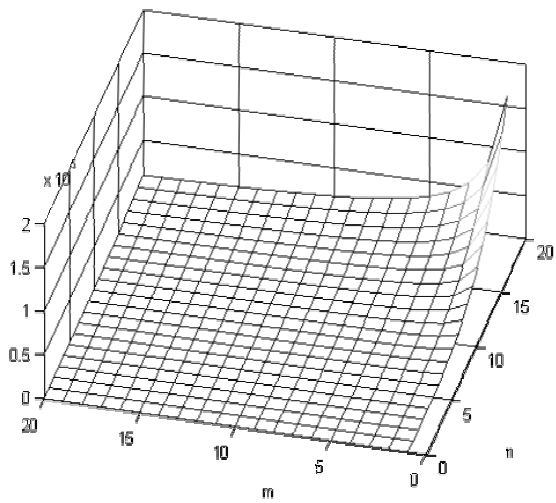



***Рис.2 Графік функції O(qn³).***

Керуючись тим фактом, що в реальному житті побудова повністю зв'язаної топології мережі надто дороге рішення, будемо говорити, що m<n(n-1)/2. А також тим, що при q=1 складність даного алгоритму досягатиме значення $O(n^3)$. А отже визначимо діапазон значень коефіцієнта q, при яких алгоритм буде мати найменшу складність – $\frac{2}{n-1} < q < 1$.

Оскільки в дослідженні ми прив'язуємось саме до даної реалізації алгоритму, щоб оцінити його здатність адекватно моделювати процес маршрутизації, тобто вирішувати задачу пошуку оптимального маршруту, необхідно оцінити його складності з практичної точки зору. Для цього було проведено ряд досліджень аналітичного характеру, по результатам яких ми отримали дані, що унаочнюють залежність часу роботи алгоритму від топології досліджуваної мережі (рис.3).

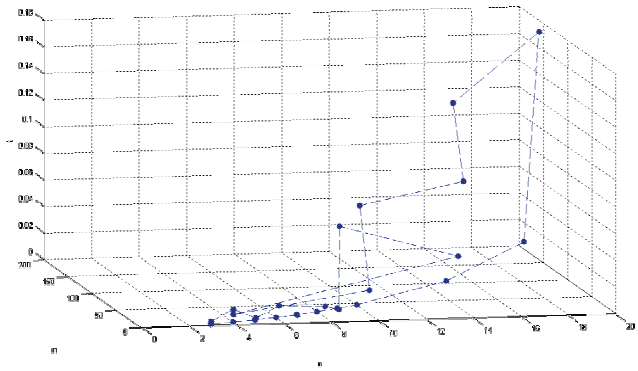


***Рис.3 Графік залежності часу виконання від структури мережі***

Аналізуючи результати дослідження можна прослідкувати деяку схожість з теоретичними даними. Завдяки проведеним розрахункам і побудові графіка, що зображений

вище, ми довели, що час роботи програми залежить не лише від кількості вузлів у топології, але й від кількості ребер, що з'єднують ці вузли.

### Огляд існуючих рішень

Метод динамічного програмування широко відомий і потужний математичний метод сучасної теорії управління. Динамічне програмування дозволяє вирішувати задачі, розбиваючи їх на підзадачі, аналогічні початковій задачі, і об'єднуючи в подальшому рішення цих підзадач. Тобто оптимальне рішення підзадач меншого розміру може бути використано для рішення початкової задачі. Ключовою умовою для застосування динамічного програмування є наявність підзадач, що перекриваються.

Для кожного кінцевого користувача важливо, щоб мережа виконувала функції важливі саме для нього. Тому існує велика кількість критеріїв оцінювання роботи мережі передачі даних. Для проведення моделювання необхідно виконати вибір критерію якості для оцінювання роботи алгоритму. При виборі критерію оцінювання роботи мережі передачі даних перевага була надана показнику, що з нашої точки зору найбільш точно покаже швидкість та надійність функціонування досліджуваного алгоритму. В даному дослідженні критерієм оцінювання функціонування мережі обрано час відгуку кінцевого вузла на запит вузла-відправника. Для цього було використано загально відому утиліту Ping, що використовується для перевірки з'єднань в мережах на основі TCP/IP. Якщо не вказано інакше, Ping пересилає пакети встановленого розміру, як правило, не великі у 32 байта. Утиліта відправляє запити (ICMP Echo-Request) протоколу ICMP [2] вказаному вузлу мережі і фіксує отримані відповіді (ICMP Echo-Reply). Час між відправкою запиту і отриманням відповіді (RTT, від англ. Round Trip Time) дозволяє визначити як двосторонні затримки (RTT) по маршруту, так і частоту, з якою пакети втрачаються.

Таким чином метою, що була поставлена в процесі дослідження, є моделювання ситуації за якої в мережі відбувається мінімізація критерію оптимізації, а саме часу, що витрачається на передачу вказаного пакету з пункту А в пункт В.

За основу даного дослідження було взято статтю [3], в якій було представлено метод моделювання процесу маршрутизації на основі методу динамічного програмування. Власне там же можна знайти алгоритм функціонування програми та результати досліджень автора. Для реалізації цього алгоритму було використано засоби середовища MATLAB, в тому числі мова програмування MATLAB. В даній праці було побудовано математичну модель мережі, із наперед визначеною конфігурацією, процес маршрутизації в ній було реалізовано на основі алгоритму динамічного програмування. Моделювання процесу маршрутизації виконано засобами програмного середовища MATLAB 7.0. В даній роботі було використано вже існуючу програмну реалізацію даного алгоритму, проте адаптовано його під роботу з іншими вхідними даними.

**Дослід №1**

Моделювання телекомунікаційної мережі, в нашому випадку, відбувалося на рівні Network моделі OSI, та, на відміну від моделі розглянутої у статті [3], в процесі даного дослідження нами було побудовано модель телекомунікаційної мережі, яка складалась з десяти вузлів (маршрутизаторів), які з'єднані між собою каналами різної перепускності (bandwidth).

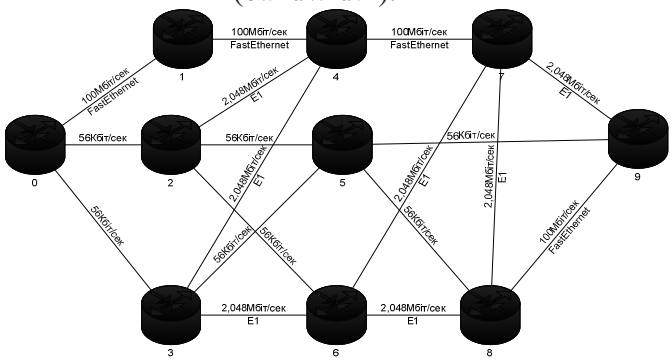



***Рис. 4. Схема мережі***

Результати роботи методу динамічного програмування порівнювалися із традиційними методами такими, як протоколи маршрутизації OSPF, RIP та їх комбінації.

В залежності від технології Канального рівня (Network Access layer) моделі TCP/IP значення перепускності (bandwidth) міняється. В таблиці 1. представлено значення перепускності (bandwidth) в залежності від різних технологій Канального рівня, що використовувались при розробці даної моделі.

***Таблиця 1.***

| Найменування технології | Формат | Перепускність (Bandwidth) |
|---|---|---|
| 100BaseT | Ethernet кадр | 100 Мб/сек |
| PPP_E1 | ip3 дейтаграма | 2.048 Мб/сек |
| PPP_56K | ipv4 дейтаграма, ipx пакет | 56Кб/сек |

Після перетворення величин пропускних здатностей у ціни каналів схема набуде наступного вигляду:

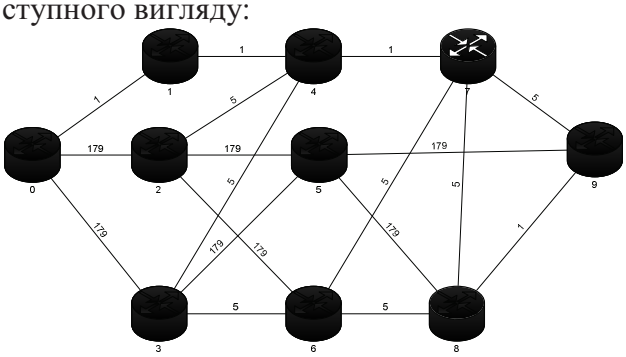



***Рис. 5. Схема мережі, представлена у вигляді графа***

Для розв'язання задачі маршрутизації інформаційного пакету з вузла 0 до вузла 9, потрібно описати отриманий граф матрицею сполучень:

$$C = \begin{pmatrix} 999 & 1 & 179 & 179 & 999 & 999 & 999 & 999 & 999 & 999 \\ 1 & 999 & 999 & 999 & 1 & 999 & 999 & 999 & 999 & 999 \\ 179 & 999 & 999 & 999 & 5 & 179 & 179 & 999 & 999 & 999 \\ 179 & 999 & 999 & 999 & 5 & 179 & 5 & 999 & 999 & 999 \\ 999 & 1 & 5 & 5 & 999 & 999 & 999 & 1 & 999 & 999 \\ 999 & 999 & 179 & 179 & 999 & 999 & 999 & 999 & 179 & 179 \\ 999 & 999 & 179 & 5 & 999 & 999 & 999 & 5 & 5 & 999 \\ 999 & 999 & 999 & 999 & 1 & 999 & 5 & 999 & 5 & 5 \\ 999 & 999 & 999 & 999 & 999 & 179 & 5 & 5 & 999 & 1 \\ 999 & 999 & 999 & 999 & 999 & 179 & 999 & 5 & 1 & 999 \end{pmatrix},$$

де числами 999 позначені неіснуючі сполучення у графі.

Подавши таку матрицю на вхід програми, отримаємо:

result =
CurrentTime: 1
CurrentVertex: 1
NextVertex: 2
Cost: 1

result =
CurrentTime: 2
CurrentVertex: 2
NextVertex: 5
Cost: 1

result =
CurrentTime: 3
CurrentVertex: 5
NextVertex: 8
Cost: 1

result =
CurrentTime: 4
CurrentVertex: 8
NextVertex: 10
Cost: 5

Отриманий результат буде занесений у таблицю маршрутизації (Routing Table) на вузлі 0 і буде застосовуватись для передачі всіх пакетів, адресованих у вузол 9. Аналогічні розрахунки будуть виконані маршрутизатором для всіх інших вузлів призначення, які наявні в його Topology Database (база даних в памяті маршрутизатора, що містить інформацією про всі маршрути). Різниця між Topology Database і Routing Table полягає у тому, що перша містить інформацію про всі маршрути з вузла 0 в вузол 9, а друга зберігає тільки оптимальні маршрути з вузла 0 в вузол 9. По результатам роботи програми в середовищі MATLAB 7.0 можна зробити висновки про оптимальний шлях з вузла 0 в вузол 9: 0 – 1 – 4 – 7 – 9.

Для перевірки результату, отриманого за допомогою програми, було використано середовище моделювання телекомунікаційних мереж OPNet, оскільки даний пакет моделювання широко розповсюджений і відомий різноманітністю технологій, що можуть бути промодельовані в даному середовищі [4]. Модель мережі наведена на наступному рисунку:

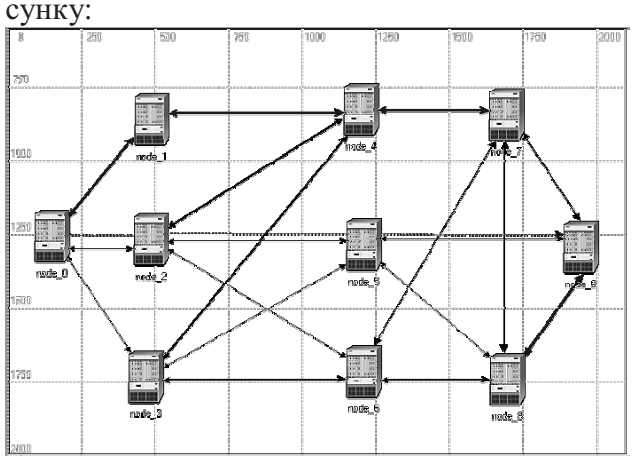



***Рис.6. Модель мережі в пакеті OPNet***

Для моделювання вузлів мережі були використані узагальнені блоки ethernet4_slip8_gtwy, які можуть моделювати роботу довільного IP-маршрутизатора. Зв'язки між вузлами мережі промодельовані за допомогою каналів PPP (Point-to-Point Protocol). Для задання ціни каналу, яка враховується при моделюванні процесу прийняття рішення про маршрутизацію пакетів,

використовується атрибут Reference Bandwidth. На основі побудованої топології мережі в середовищі OPNet було здійснено настройку існуючих протоколів маршрутизації трьома різними способами. В першому випадку було настроєно протокол маршрутизації OSPF (Open Shortest Path First), який враховує значення перепускності кожного з'єднання для пошуку оптимального маршруту. Для знаходження оптимального (у сенсі швидкості передачі даних) шляху між вузлами «node_0» та «node_9», які відповідають вершинам 0 та 9 графа, зображеного на Рис. 2, було задано напрям руху трафіку ICMP (Ping Traffic) від вузла «node_0» до вузла «node_9».

List of traversed IP interfaces:

| IP Address | Hop Delay | Node Name |
|---|---|---|
| 192.0.14.2 | 0,00000 | Enterprise Network.node_0 |
| 192.0.13.1 | 0,00304 | Enterprise Network.node_1 |
| 192.0.2.1 | 0,00401 | Enterprise Network.node_4 |
| 192.0.12.2 | 0,00285 | Enterprise Network.node_7 |
| 192.0.18.1 | 0,00215 | Enterprise Network.node_9 |

Якщо ICMP пакети проходять через ці вузли, очевидно, що шлях node_0 -> node_1 -> node_4 -> node_7 -> node_9 обрано оптимальним для даної мережі на проміжку між вузлом 0 та вузлом 9. А отже результат моделювання в середовищі OPNet підтвердив результати моделювання з допомогою MATLAB 7.0.

**Висновок**: програма написана по алгоритму динамічного програмування підходить для вирішення задачі маршрутизації в більших мережах, тобто добре масштабується на більші мережі. Крім того, даний алгоритм підходить для вирішення задачі маршрутизації з урахуванням перепускності лінка як показника якості для вибору оптимального маршруту.

**Дослід №2**

В другому випадку процес пошуку оптимального шляху в моделі здійснювався по протоколу динамічної маршрутизації RIP (Routing Information Protocol). Даний протокол для своєї роботи використовує досить просту метрику, що вимірюється кількістю проміжних вузлів від відправника до адресата. Як відомо протокол маршрутизації RIP не враховує значення перепускності (bandwidth) для вибору оптимального маршруту в телекомунікаційній мережі, рішення про оптимальність маршруту приймається на основі метрики, що вимірюється в кількості проміжних вузлів від відправника до одержувача. А отже можна сказати, що показником якості для роботи такої мережі буде відстань, що обчислюється в хопах (кількість кроків до одержувача), і значенням перепускності (bandwidth) можна знехтувати. Тому модель такої мережі можна представити у вигляді показаному на рисунку 7.

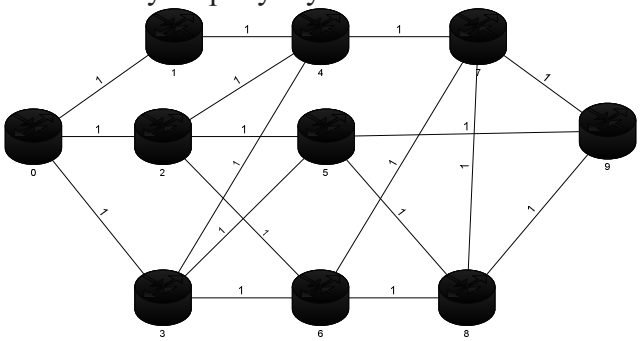



***Рис. 7. Схема мережі у вигляді графа з одиничними вагами зв'язків між вузлами***

Для розв'язання задачі маршрутизації інформаційного пакету з вузла 0 до вузла 9, потрібно описати отриманий граф матрицею сполучень:

$$C = \begin{pmatrix} 999 & 1 & 1 & 1 & 999 & 999 & 999 & 999 & 999 & 999 \\ 1 & 999 & 999 & 999 & 1 & 999 & 999 & 999 & 999 & 999 \\ 1 & 999 & 999 & 999 & 1 & 1 & 1 & 999 & 999 & 999 \\ 1 & 999 & 999 & 999 & 1 & 1 & 1 & 999 & 999 & 999 \\ 999 & 1 & 1 & 1 & 999 & 999 & 999 & 1 & 999 & 999 \\ 999 & 999 & 1 & 1 & 999 & 999 & 999 & 999 & 1 & 1 \\ 999 & 999 & 1 & 1 & 999 & 999 & 999 & 1 & 1 & 999 \\ 999 & 999 & 999 & 999 & 1 & 999 & 1 & 999 & 1 & 1 \\ 999 & 999 & 999 & 999 & 999 & 1 & 1 & 1 & 999 & 1 \\ 999 & 999 & 999 & 999 & 999 & 1 & 999 & 1 & 1 & 999 \end{pmatrix},$$

де 1 – вузли з'єднані безпосередньо, 999 – безпосередній зв'язок між вузлами відсутній.

Подавши таку матрицю на вхід програми, отримаємо:

```
result =
    CurrentTime: 2
  CurrentVertex: 1
     NextVertex: 3
           Cost: 1
result =
    CurrentTime: 3
  CurrentVertex: 3
     NextVertex: 6
           Cost: 1
result =
    CurrentTime: 4
  CurrentVertex: 6
     NextVertex: 10
           Cost: 1
```

Отже оптимальний шлях з вузла 0 в вузол 9, вирахуваний програмою на основі алгоритму динамічного програмування: 0 -> 2 -> 5 -> 9. Для перевірки цих даних в середовищі OPNet необхідно налаштувати для даної топології протокол маршрутизації RIP (Routing Information Protocol). В результаті виконаних налаштувань найбільш оптимальним був обраний шлях:

0 -> 2 -> 5 ->9.

List of traversed IP interfaces:

| IP Address | Hop Delay | Node Name |
|---|---|---|
| --- | --- | --- |
| 192.0.11.1 | 0,00000 | Enterprise Network.node_0 |
| 192.0.10.1 | 0,11449 | Enterprise Network.node_2 |
| 192.0.28.2 | 0,02117 | Enterprise Network.node_5 |
| 192.0.18.1 | 0,02153 | Enterprise Network.node_9 |

Отже, результат моделювання підтверджено в пакеті OPNet. Для даного випадку вибір шляху 0-3-5-9 як оптимального також має сенс, оскільки по обраному критерію якості обидва маршрути рівнозначні.

**Висновок:** Програма, написана по алгоритму динамічного програмування придатна для розрахунку процесу маршрутизації на основі такого показника якості, як кількість проміжних вузлів від відправника до одержувача.

**Дослід №3**

Щоб максимально приблизити дане дослідження до реальності, було вирішено промоделювати роботу даної топології з двома різними протоколами динамічної маршрутизації. У випадку, коли маршрутизатор працює по кільком різним протоколам, для прийняття рішення про оптимальність маршруту використовується додатковий критерій, що називається адміністративною відстанню АВ (Administrative distance), який характеризує ступінь довіри до того чи іншого джерела інформації про маршрут. Адміністративна дистанція виражається числовим параметром, який може приймати значення від 0 до 255. Кожен протокол маршрутизації має своє значення адміністративної дистанції, при чому чим менше значення АД, тим більш достовірною вважається інформація про маршрут.

***Таблиця 2.***

| Назва протоколу | Значення АД |
|---|---|
| Мережа безпосередньо з’єднана з даним вузлом | 0 |
| Статичний маршрут | 1 |
| Маршрут отриманий по OSPF | 110 |
| Маршрут отриманий по RIP | 120 |
| Походження маршруту невідоме | 255 |

Для моделювання процесу маршрутизації по двом критеріям необхідно в алгоритм програми додати блок, що дозволить враховувати значення АД. Вхідними даними для програми будуть дві матриці: матриця сполучень (як і в випадку пошуку оптимального маршруту по одному критерію) та матриця адміністративних дистанцій (щоб мати змогу оцінювати ступінь довіри до протокола).

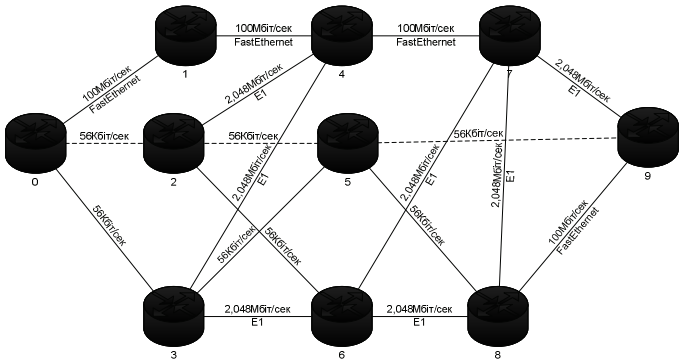



***Рис. 8. Модифікована схема мережі***

В даній праці для перевірки роботи програми з урахуванням двох критеріїв якості, необхідно модифікувати схему мережі, наведену на рис. 4 таким чином, щоб ділянки, які в попередніх дослідах були обрані оптимальними по протоколу OSPF або RIP, продовжували працювати по тому ж протоколу. В модифікованій мережі було використано комбінацію протоколів RIP та OSPF, таким чином, що на ділянці 0 – 2 – 5 – 9 настроєно RIP, а на всіх інших – OSPF.

Якщо врахувати особливості роботи модифікованого алгоритму програми (мається на увазі, що для вибору оптимального маршруту два критерії якості для кожного з'єднання додаються і уже по мінімальному значенню суми обирається оптимальний маршрут), цілком логічно було припустити, що результати досліду будуть відрізнятись від тривіального рішення.

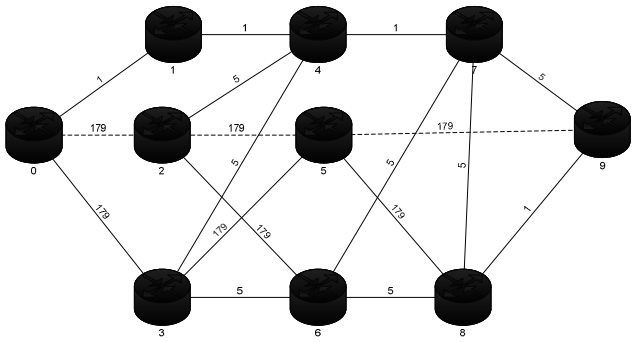



***Рис. 9. Модифікована схема мережі у вигляді графу***

Вхідними даними для моделювання в середовищі MATLAB 7.0, як уже було сказано вище, є дві матриці.

$$C = \begin{pmatrix} 999 & 1 & 179 & 179 & 999 & 999 & 999 & 999 & 999 & 999 \\ 1 & 999 & 999 & 999 & 1 & 999 & 999 & 999 & 999 & 999 \\ 179 & 999 & 999 & 999 & 5 & 179 & 179 & 999 & 999 & 999 \\ 179 & 999 & 999 & 999 & 5 & 179 & 5 & 999 & 999 & 999 \\ 999 & 1 & 5 & 5 & 999 & 999 & 999 & 1 & 999 & 999 \\ 999 & 999 & 179 & 179 & 999 & 999 & 999 & 999 & 179 & 179 \\ 999 & 999 & 179 & 5 & 999 & 999 & 999 & 5 & 5 & 999 \\ 999 & 999 & 999 & 999 & 1 & 999 & 5 & 999 & 5 & 5 \\ 999 & 999 & 999 & 999 & 999 & 179 & 5 & 5 & 999 & 1 \\ 999 & 999 & 999 & 999 & 999 & 179 & 999 & 5 & 1 & 999 \end{pmatrix}$$

$$D = \begin{pmatrix} 999 & 110 & 120 & 110 & 999 & 999 & 999 & 999 & 999 & 999 \\ 110 & 999 & 999 & 999 & 110 & 999 & 999 & 999 & 999 & 999 \\ 120 & 999 & 999 & 999 & 110 & 120 & 110 & 999 & 999 & 999 \\ 110 & 999 & 999 & 999 & 110 & 110 & 110 & 999 & 999 & 999 \\ 999 & 110 & 110 & 110 & 999 & 999 & 999 & 110 & 999 & 999 \\ 999 & 999 & 120 & 110 & 999 & 999 & 999 & 999 & 110 & 120 \\ 999 & 999 & 110 & 110 & 999 & 999 & 999 & 110 & 110 & 999 \\ 999 & 999 & 999 & 999 & 110 & 999 & 110 & 999 & 110 & 110 \\ 999 & 999 & 999 & 999 & 999 & 110 & 110 & 110 & 999 & 110 \\ 999 & 999 & 999 & 999 & 999 & 120 & 999 & 110 & 110 & 999 \end{pmatrix}$$

В результаті застосування програми, що написана по алгоритму динамічного програмування і модифікована для роботи з двома критеріями якості, було отримано наступний маршрут:

```
result =
    CurrentTime: 2
  CurrentVertex: 1
     NextVertex: 4
           Cost: 289
result =
    CurrentTime: 3
  CurrentVertex: 4
     NextVertex: 6
           Cost: 289
result =
    CurrentTime: 4
  CurrentVertex: 6
     NextVertex: 10
           Cost: 299
```

Отже, по результатам моделювання в середовищі MATLAB 7.0 маршрут 0 – 3 – 5 – 9 було обрано найоптимальнішим.

Щоб перевірити ці дані необхідно модифікувати і модель в середовищі OPNet як показано на рисунку 10.

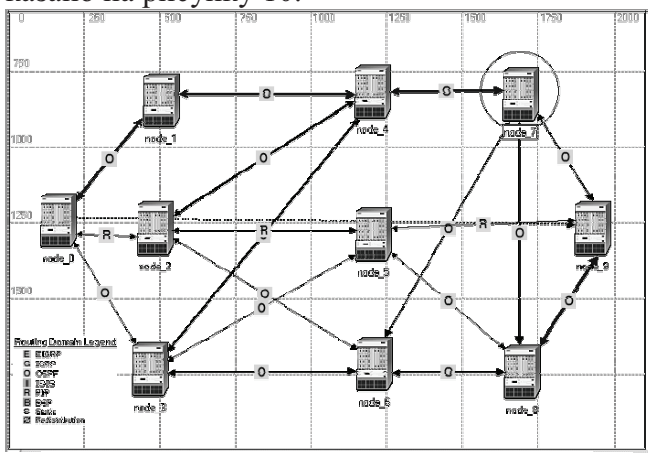



***Рис. 10. Модель модифікованої мережі в OPNet***

В результаті виконаних налаштувань найбільш оптимальним був обраний шлях:

List of traversed IP interfaces:

| IP Address | Hop Delay | Node Name |
|---|---|---|
| 192.0.14.2 | 0,00000 | Enterprise Network.node_0 |
| 192.0.13.1 | 0,00304 | Enterprise Network.node_1 |
| 192.0.2.1 | 0,00403 | Enterprise Network.node_4 |
| 192.0.12.2 | 0,00281 | Enterprise Network.node_7 |
| 192.0.18.1 | 0,00221 | Enterprise Network.node_9 |

Моделювання процесу маршрутизації на основі двох критеріїв якості, принесло не зовсім очікувані результати, оскільки оптимальний маршруту 0-1-4-7-9 вирахуваний засобами середовища OPNet відрізнявся від отриманого в середовищі MATLAB, так як був отриманий на основі роботи алгоритму динамічного програмування, що не повторює собою застосовувані на даний момент алгоритми маршрутизації мережевого рівня.

**Висновок:** Отримана таким чином інформація, що описує стан мережі вказаними вище методами, надає можливість отримати альтернативне оптимальне рішення задачі з двома критеріями якості, завдяки чому можна знаходити множину оптимальних маршрутів по першому з показників якості (кількість проміжних вузлів) і уже з цієї множини обирати оптимальне рішення по другому показнику якості (пропускна здатність сполучення). За допомогою методу динамічного програмування, застосованому для моделювання процесу маршрутизації, можна знизити ймовірність утворення черг в телекомунікаційних мережах, а отже і зменшити ймові-

рність втрати пакетів даних. Даний метод надає можливість організації оптимального розподілення мережевих ресурсів, за рахунок перерозподілу навантаження на найбільш оптимальних маршрутах, за рахунок використання сполучень, що зазвичай не використовуються для передачі даних в стандартних алгоритмах.

Окрім того застосування даного методу зменшує час, що витрачається на процес вибору маршруту за інших рівних умов, оскільки часткові рішення задачі пошуку оптимального шляху зберігаються у буфері, то при визначенні повної задачі є можливість використовувати вже існуючі, розраховані часткові рішення, що в поєднанні дають рішення початкової задачі. Підтвердження даної гіпотези стає можливим завдяки процесу емуляції програмної реалізації математичних методів на спеціалізованому технічному обладнанні, що розроблене для моделювання та тестування мережевих рішень рівня Network моделі OSI.

В нашому випадку основною складністю та недоліком у процесі моделювання на основі методу динамічного програмування являється процес складання матриці з'єднань, оскільки матриці з'єднань складаються вручну. В подальшому планується написання спеціального блоку підпрограм, що вирішує дану проблему.

## Список посилань

1. Чураков Е.П. Оптимальные и адаптивные системы : учеб. пособие для вузов / Чураков Е.П. – М. : Энергоатомиздат, 1987. – 256 с.
2. J. Postel. RFC792 - Internet Control Message Protocol. 1981.
3. Репнікова Н.Б., Дорошенко К.С., Жуковський О.О. Метод моделювання процесу маршрутизації в IP-мережі за допомогою мови програмування MatLab /«Штучний інтелект» 2'2009. – С.63-68.
4. T. Svensson, A. Popescu –Development of laboratory exercises based on OPNET Modeler / Blekinge Institute of Technology, 2003.
5. Карачун В.Я., Бех П.О., Гульчук Г.Г., Карачун О.О., Коряченко В.Г., Прохур Ю.З., Черненко І.А, Чиж С.М. Російсько-українсько-англійський науково-технічний словник – Київ «Техніка»,1997

Поступила в редакцию 8.12.2009

УДК 004.415.25

*ЛІСНІЧЕНКО Д. І.,*
*СИМОНЕНКО А.В.*

# СИСТЕМА ОПЕРУВАННЯ ФАЙЛОВИМИ ІЄРАРХІЯМИ

При традиційному підході до обробки ієрархій файлів, для кожної конкретної області застосування розробляється власне програмне забезпечення. У статті описано об'єктну модель існуючої системи, що забезпечує більш гнучкий і універсальний підхід. Розглянуто й проаналізовано окремі деталі дизайну системи.

Traditional approach to file hierarchies processing requires development of specific software for every use case. In this article, object model of more flexible and universal system is proposed. Some design details are considered and analyzed.

## Вступ

Під час обробки файлових ієрархій особливу складність складає коректна реалізація рекурсивних операцій над ними. Програмне забезпечення, що виконує ці операції, як правило, реалізує один незмінний алгоритм, наприклад, синхронізації, контролю версій, резервного копіювання. Однак такий підхід не забезпечує достатньої гнучкості та повторного використання коду, тому завжди існує вірогідність того, що конкретному користувачу не вистачатиме якоїсь функції і доведеться розробляти власний комплекс практично з нуля.

Запропонована система оперування файловими ієрархіями призначена для виконання списків рекурсивних операцій над файловими ієрархіями, що знаходяться на одному чи кількох комп'ютерах. Файлова ієрархія представляється у оперативній пам'яті у вигляді структури об'єктів і з'являється можливість працювати із структурою, а не з файловою системою на диску. Позбувшись таким чином необхідності мати доступ до файлової системи, ми отримуємо можливість швидкої однотипної обробки представлень локальних та віддалених файлових систем. Дерево може бути отримане шляхом сканування файлової системи або може бути завантажене із попередньо згенерованого файлу опису. Відповідно, можна порівнювати файлові ієрархії на віддалених комп'ютерах, стани однієї й тієї ж файлової системи у різні моменти часу [1], виділяти змінені файли для збереження до інкрементальної резервної копії [2] тощо. Завдяки публічно доступному API, вбудованому інтерпретатору скриптів та форматі службових файлів, що базується на XML, забезпечується висока гнучкість та налаштовуваність системи.

## Структура системи

Структура системи схематично зображена на рис 1.

Система складається з:

- Ядра. Функції ядра: виконує операції над файловою системою, завантажує списки файлів у віртуальні дерева, виконує над ними операції, експортує списки файлів і формує службові пакети. Службові файли зберігаються у форматі XML, що забезпечує кросплатформенність та уніфікацію. Можливості ядра розширюються за допомогою модулів розширення.
- Публічного API (Application programming interface), що надає інтерфейс для задання списку та аргументів операцій, які мають бути виконані ядром.
- Інтерпретатора скриптів і сторонніх програм, котрі використовують API для виконання високорівневих задач. Інтерпретатор може бути викликаний із системних скриптів і відповідно надає можливість пакетного виконання. Сторонні програми можуть використовувати API для безпосереднього керування процесом обробки.
- Надбудов, які надають зручний інтерфейс користувача. Наприклад, конструктор скриптів дозволяє у графічному режимі створювати скрипти і виконувати їх попередній аналіз.

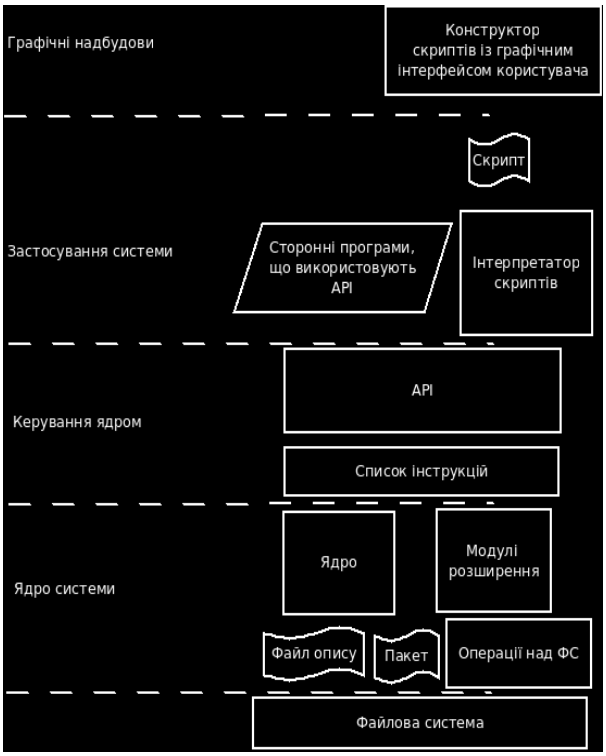



***Рис. 1. Структура системи***

### Внутрішнє представлення файлів

Усередині системи файлова ієрархія представляється у вигляді дерев відповідно до поширеного шаблону проектування Composite [3]. Для оптимального використання пам'яті застосовується відділення метаданих файлів (поля нащадків AbstractFile) від елементів дерев, над якими виконуються маніпуляції (Node) використовуються шаблони проектування Memento і FlyWeight [3]. Таким чином, під час виконання операцій над деревами створювані об'єкти займають мінімальний об'єм пам'яті і вдається уникнути створення і подальшого видалення допоміжних об'єктів (наприклад, String) (рис 2).

Вся інформація про файл представлена у системі у вигляді метаданних (Property) (рис 3). Відповідно до шаблону проектування Comparator [3], класи Property, що описують метадані, водночас надають інструменти для порівняння цих даних. Для додання нового класу метаданних до системи достатньо оголосити нащадка абстрактного класу Property і додати поля для зберігання інформації про новий тип метаданих до класу PropertyData (шаблон Memento [3]). Безперечно, більш прямолінійним підходом було б зберігання пар "назва метаданних – значення " у HashMap. При такому підході не потрібно було б модифікувати клас PropertyData, а досить було б додати новий клас до відповідного пакету. Але, оскільки система призначена для обробки мільйонів і, можливо, десятків мільйонів файлів, було вирішено принести суворо об'єктний підхід у представленні даних у жертву швидкості роботи.

### Фільтри та критерії

Операції над файловими ієрархіями виконуються з урахуванням лише метаданих. Для цього задаються критерії (Criteria) та фільтри (Filter).

*Критерії* – список Property, які слід приймати до уваги при виконанні операції (рис 3). Наприклад, для операції перетину список Criteria визначає, ідентичність яких метаданих для двох AbstractFile свідчить ідентичність самих об'єктів.

*Фільтри* – умови, що дозволяють проігнорувати певні елементи під час виконання операції. Абстрактний клас Filter реалізує шаблони проектування Interpreter та Template Method [3] і дозволяє задавати складні вкладені умови. Умови можуть включати як логічні оператори, так і операції порівняння значень метаданих із конкретними значеннями. (рис 4)

### Внутрішнє представлення операцій

Аналогічним чином побудована ієрархія об'єктів, що представляють операції над файловими ієрархіями (рис 5). AssignCommand є інструкцією, у ході виконання якої виконується вираз Expression, що складається із операцій Operation та змінних Variable. Операції можуть містити змінні та вкладені операції. Така гнучкість забезпечена завдяки застосуванню шаблонів Interpreter та Template Method [3].

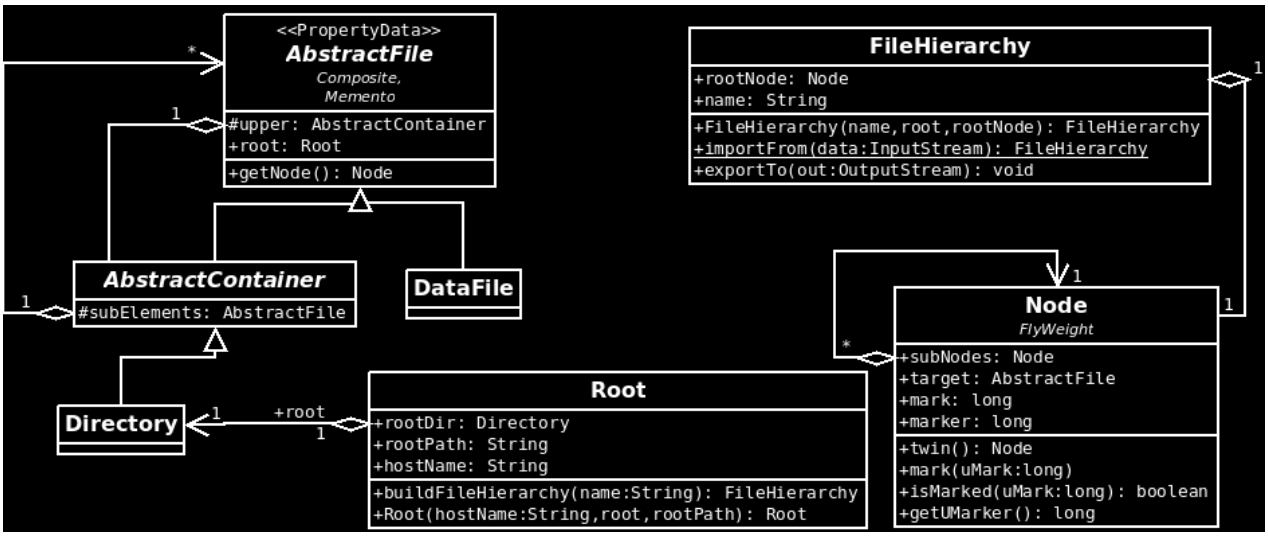



*Рис. 2 Внутрішнє представлення файлів*

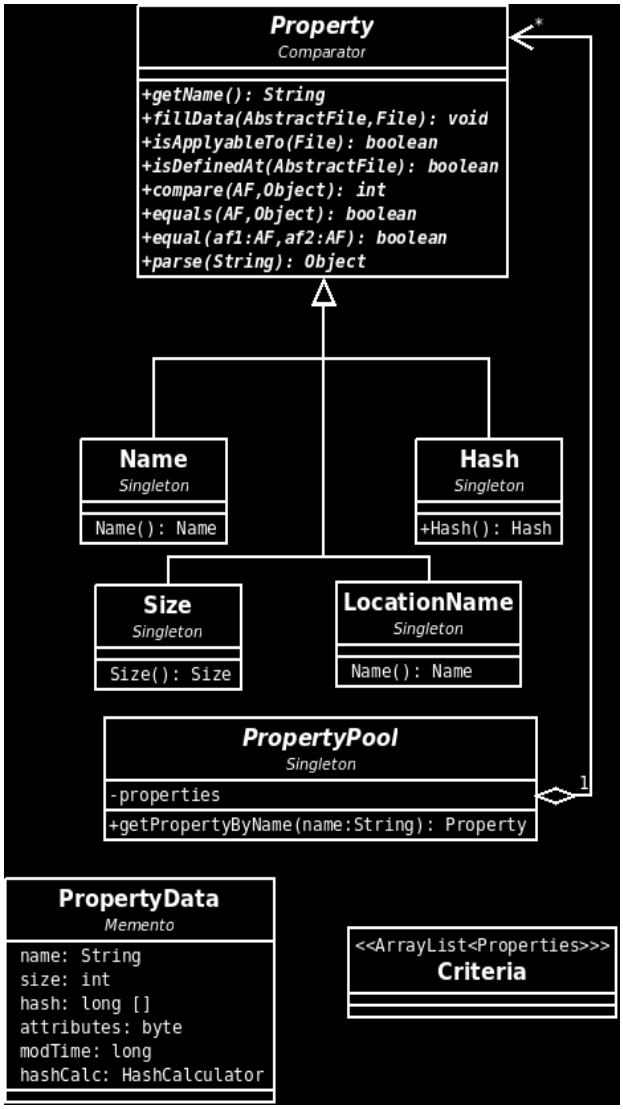



*Рис 3. Представлення метаданих*

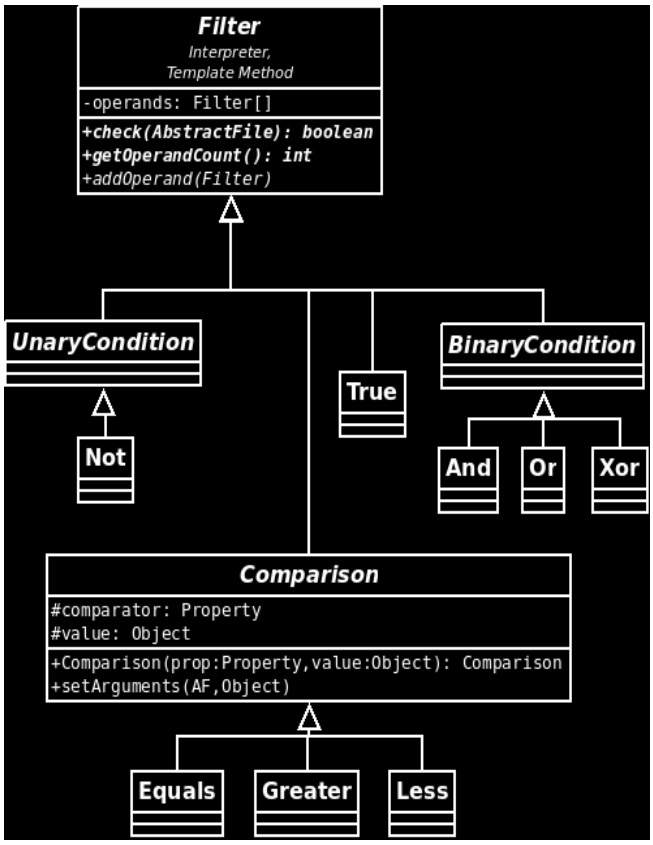



*Рис 4. Фільтри*

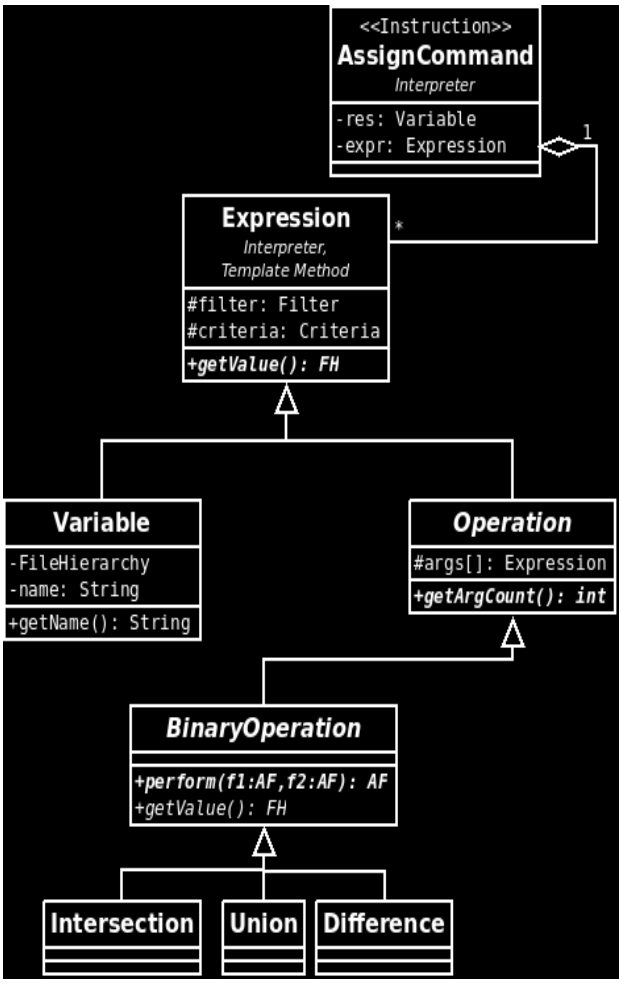



***Рис 5 Внутрішнє представлення операцій***

## Підсумок

У статті розглянуто об'єктну модель системи оперування файловими ієрархіями. Для забезпечення гнучкості, універсальності, можливості підтримки та повторного використання коду система побудована із застосуванням поширених шаблонів проектування. У результаті система може бути легко розширена (наприклад, шляхом реалізації нових типів метаданих) і інтегрована через відкритий API у сторонні програми.

Система надає можливість винесення алгоритму обробки файлових ієрархій до високорівневих скриптів, які можна легко модифікувати відповідно до реальних потреб.

Оскільки базові операції реалізовані безпосередньо на рівні системи, протестовані і відповідно можуть вважатись досить надійними, зменшується вірогідність виникнення помилок. Це важливо під час обробки файлових ієрархій, будь-які помилки, що виникають у процесі обробки, є загрозою цілісності даних.

## Список посилань

1. http://ru.wikipedia.org/wiki/Система_управления_версиями
2. http://en.wikipedia.org/wiki/Incremental_backup
3. Гамма Э., Хелм Р., Джонсон Р., Влиссидес Дж. Приемы объектно-ориентированного проектирования. Паттерны проектирования. – СПб: Питер, 2001. – 368 с

Поступила в редакцию 14.12.2009

УДК 004.27

*КЛИМЕНКО И.А.,*
*ЖАБИНА В.В.*

# ОБЕСПЕЧЕНИЕ ОТКАЗОУСТОЙЧИВОСТИ ПОТОКОВЫХ СИСТЕМ НА ОДНОТИПНЫХ ВЫЧИСЛИТЕЛЬНЫХ МОДУЛЯХ

Предлагаются методы автоматической реконфигурации потоковых вычислительных систем с различной организацией сред формирования команд. Показана возможность исправления ошибок, вызванных отказом вычислительных модулей, на аппаратном уровне без дополнительного расхода команд. Процесс вычислений продолжается, пока в системе остается хотя бы один работоспособный вычислительный модуль.

Methods of automatic reconfiguration for data-flow systems with different instruction generation environment configuration are proposed. Possibility to fix errors caused by computation unit failure without additional instruction use at hardware level is shown. Computation process is active until there is at least one operable computation unit.

## Введение

Для систем управления процессами и объектами в реальном времени важными характеристиками является быстродействие и надежность. Как известно, возможным подходом к ускорению обработки информации является распараллеливание вычислительных процессов на различных уровнях. В системах реального времени зачастую возникает необходимость реализации алгоритмов с мелкозернистой структурой. К ним относятся, например, алгоритмы интерполяции функций, расчета траектории объектов, преобразования координат в многомерном пространстве. Ускорения вычислений в этом случае можно добиться распараллеливанием вычислений на уровне операций. Использование вычислительных систем, в которых распределение заданий между вычислительными узлами осуществляется под управлением программы с использованием статических средств программирования, неэффективно ввиду большого числа обмена данными между узлами, выполняющими отдельные операции. Объединение операций, связанных по данным, в программный модуль уменьшает число пересылок данных, но снижает возможности распараллеливания вычислений.

В работе [1] показано, что эффективным подходом к ускорению реализации мелкозернистых алгоритмов является использование динамических средств распределения операций между вычислительными узлами, основанных на потоковой модели вычислений (вычислений, управляемых потоком данных). В потоковых системах, реализующих указанную модель вычислений, распределение операций осуществляется динамически в процессе вычислений на аппаратном или микропрограммном уровне. При подготовке задачи нет необходимости в прямой форме описывать процедуры синхронизации процессов и обмена данными между ветвями алгоритмов. Все это создает предпосылки для ускорения обработки информации.

Эффективным подходом к проблеме повышения надежности систем и достоверности результатов обработки информации является обеспечение отказоустойчивости систем. Наиболее экономичными с точки зрения аппаратуры являются программные способы повышения отказоустойчивости. Однако для систем реального времени зачастую временные затраты при таком подходе оказываются неприемлемыми. В данном случае более эффективными являются методы динамической реконфигурации систем при отказе оборудования. Проблеме обеспечения отказоустойчивости систем посвящено множество публикаций, в том числе, монографий [2-4]. Однако, эта проблема не разработана для систем, управляемых потоком данных, ввиду специфики организации вычислительных процессов в таких системах.

Таким образом, разработка методов и средств обеспечения отказоустойчивости вычислительных потоковых систем, в которых обеспечивается динамическое распределение заданий между вычислительными узлами и динамическая реконфигурация при отказе оборудования является актуальной задачей.

## Модель вычислений, управляемых потоком данных

К первым работам в области организации систем, управляемых потоком данных, можно отнести публикации [5-8]. Несмотря на особенности архитектуры потоковых систем, они объединены общей концепцией, связанной со способом формирования и порядком выполнения команд.

В системах, управляемых потоком данных, команды выполняются не в заданной программой последовательности, а по мере поступления полной информации о команде и операндах, то есть определяющим для выполнения команды является доступность данных, а не заданный программой порядок выполнения команд.

Данные для вычислений могут быть подготовлены на основе потокового графа, вершины которого определяют операции, а дуги – потоки данных. Таким образом, связь между командами осуществляется только по данным.

Операции описывается информационным словом, которое называют актором (*actor*). В общем случае он может быть представлен в виде следующего кортежа

$$A = \langle I, F, N, P \rangle, \quad (1)$$

где $I$ – идентификатор актора; $F$ – функция преобразования данных; $N$ – имя актора, для которого передается результат в качестве операнда, а $P$ – совокупность признаков данного операнда.

В свою очередь, данные описываются в следующем виде

$$D = \langle I, Q, N, P \rangle, \quad (2)$$

где $Q$ – значение операнда.

На аппаратном уровне потоковая вычислительная система состоит из среды формирования команд, нескольких вычислительных модулей и блока управления. Связь между структурными компонентами и вычислительной системой более высокого уровня осуществляется через коммутационную среду. Различные аппаратные решения определяют различный способ формирования и активизации команд. В систему поступают акторы и данные. С использованием определенных средств и алгоритмов формируются команды, которые выполняются в свободном вычислительном модуле.

## Постановка задачи

Несмотря на общую модель вычислений, потоковые системы имеют существенные отличия, связанные со способом активизации команд. Существуют различные подходы к структурной организации СФК. Известны методы формирования команд в СФК на основе ассоциативной памяти (АП) [7] и памяти с произвольным адресным доступом к ячейкам (ППД) [9]. Целью работы является исследование возможности обеспечения отказоустойчивости потоковых систем на однотипных ВМ, использующих разные механизмы формирования команд.

## Автоматическая реконфигурация потоковых систем с АП

В потоковых системах подготовка задачи осуществляется без учета количества вычислительных модулей. Это создает предпосылки в случае отказа ВМ продолжать вычисления до тех пор, пока в системе не останется хотя бы один работоспособный ВМ. Учитывая, что методы обеспечения отказоустойчивости памяти разработаны достаточно хорошо (см., например, [10]), основное внимание уделяется аппаратным средствам автоматической реконфигурации системы при отказе ВМ. При этом основной проблемой является восстановление информации о команде, утерянной в связи с отказом ВМ.

Пусть для определенности ВМ выполняют двуместные операции (для одноместных может быть введен фиктивный операнд). Тогда команды из акторов (1) и данных (2) формируются в формате $\langle A, D_1, D_2 \rangle$.

Организация системы поясняется укрупненной структурной схемой (рис. 1). В состав системы входят вычислительные модули (ВМ), среда формирования команд (СФК) на базе ассоциативной памяти (АП), регистры для временного хранения информации (РВ), блок управления (БУ) буферная память (БП) и коммутационные среды (КС).

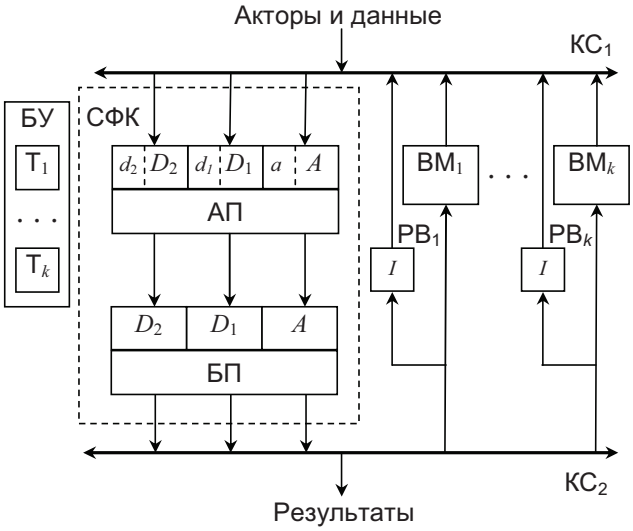



***Рис. 1. Организация системы с АП***

Формирование команд производится следующим образом. Акторы и данные в любом порядке поступают через $КС_1$ в СФК. С использованием процедуры адресной записи акторы и данные для $i$–й команды записываются в одну ячейку АП по адресу $I_i$, который в соответствии с (1) и (2) совпадает для всех объектов одной команды. Одновременно с записью актора и данных устанавливаются признаки их наличия в соответствующих разрядах ячейки памяти ($d_2, d_1, a$). Значения признаков $d_2 d_1 a = 111$ указывает на наличие в АП готовой команды для ВМ. Готовая команда считывается из АП с помощью процедуры ассоциативного чтения и переписывается в БП типа FIFO. Цикл безадресного чтения данных из БП значительно меньше цикла ассоциативного чтения из АП, что дает возможность ускорить вычисления.

Свободный $ВМ_j$ принимает команду из БП и приступает к ее выполнению. Адрес $I_i$ на время выполнения команды сохраняется в $РВ_j$. Одновременно с этим сбрасываются в нулевое состояние признаки в соответствующей ячейке АП и блокируется запись новой информации в указанную ячейку. Пока признаки снова не будут установлены в единицу, команда считается неготовой и не может быть считана повторно другим ВМ. Вместе с этим в БУ запускается таймер $Т_j$ на время, которое достаточно для выполнения самой продолжительной команды. Число таймеров должно быть равно числу ВМ.

После успешного выполнения команды результат из $ВМ_j$ через $КС_1$ записывается в АП по адресу $N_i$. Вместе с этим сбрасывается таймер $Т_j$ и разблокируется адресная запись в ячейку с адресом $I_i$, то есть эта ячейка может быть использована для формирования другой команды с соответствующим именем.

В случае отказа $ВМ_j$ срабатывается таймер $Т_j$ (заканчивается лимит времени на выполнения команды). $ВМ_j$ отключается от КС и команда активируется путем установки признаков $d_2 d_1 a = 111$ в ячейке АП с адресом $I_i$, который сохранялся в $РВ_j$. Свободный ВМ считывает эту команду и она выполняется повторно, как указано выше.

Поскольку подготовка алгоритма к реализации для потоковой системы выполняется без учета конкретного числа ВМ в системе, то имеется возможность продолжения вычислений до тех пор, пока в системе будет оставаться хотя бы один работоспособный ВМ.

Основным недостатком системы является сложность ассоциативной памяти и большая длительность цикла ассоциативного поиска данных, что ограничивает объем памяти и снижает производительность системы.

Трудность реализации ассоциативных запоминающих устройств большого объема приводит к необходимости их эмуляции с применением других технических средств, например, специализированных процессоров [6], что позволяет увеличить объем памяти, но существенно снижает производительность, поскольку процесс эмуляции требует больших временных затрат. Кроме того, при такой организации системы невозможно использовать ячейки АП после считывания команды для других команд, что требуется, например, при итерационных вычислениях и вычислениях в конвейерном режиме. Все это снижает эффективность параллельных вычислений.

### Реконфигурация систем с ППД

Уменьшение объема оборудования для активизации акторов и ускорения циклов обращения к памяти для формирования команд может быть достигнуто с использованием СФК на основе механизма адресного чтения и записи [1, 9] с использованием ППД. Возможны различные варианты организации систем.

*Первый вариант.* Укрупненная структура системы с ППД показана на рис. 2. Рассмотрим особенности активации команд для такой системы. Команда начинает формироваться в ячейках ППД, два разряда которых ($d$ и $a$) являются признаками, показывающими наличие в ППД одного операнда $D_1$ и актора. Запись указанных объектов для $i$–й команды производится по адресу $I_i$. При поступлении из $КС_1$ второго операнда $D_2$ для данной команды, он записывается непосредственно в соответствующие разряды регист-

ра буферной памяти (БП) типа *FIFO*, куда одновременно из ППД переписываются $A$ и $D_1$.

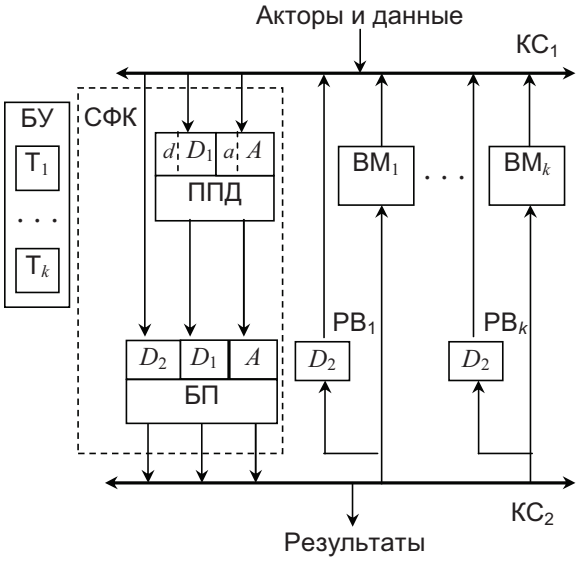



***Рис. 2. Организация системы с ППД по первому варианту***

Таким образом, в БП формируется готовая для выполнения команда, которая затем через $КС_2$ передается в свободный ВМ. При этом содержимое соответствующей ячейки ППД сохраняется для обеспечения повторного формирования команды в случае отказа ВМ. Одновременно с поступлением команды $\langle A, D_1, D_2 \rangle$ в $ВМ_j$ ее часть $D_2$, которая не сохраняется в ППД, записывается в регистр временного хранения $РВ_j$ на входной шине соответствующего вычислителя. При успешном выполнении операции, то есть когда время ее выполнения не превысило лимит времени выполнения команды в $ВМ_j$ установленный соответствующим таймером $Т_j$ блока управления БУ, результат операции поступает через $КС_1$ в ППД по адресу $N_i$, а содержимое соответствующей ячейки ППД по адресу $I_i$ обнуляется.

Если при выполнении операции произошел отказ $ВМ_j$, то есть лимит времени выполнения операции, заданный соответствующим $Т_j$, был превышен, $ВМ_j$ считается отказавшим и блокируется (отключается от $КС_1$ и $КС_2$), а значение $D_2$ из $РВ_j$ снова передается в СФК. Поскольку значения $A$ и $D_1$ утеряны не были (соответствующая ячейка ППД не была изменена), то команда повторно записывается в БП, что дает возможность реализовать следующую успешную попытку выполнения команды в исправном ВМ.

К недостаткам описанного метода следует отнести необходимость хранения компонентов команды в ячейках ППД до полного завершения выполнения данной команды, в том числе, с учетом времени реконфигурации системы при отказе ВМ и повторного выполнения команды. Это замедляет вычисления, если соответствующую ячейку памяти необходимо использовать для других команд, например, для очередной итерации или очередной реализации алгоритма в конвейерном режиме.

*Второй вариант.* Для устранения указанного недостатка предлагается в РВ записывать не только операнд $D_2$, а всю команду из БП (рис. 3) [11]. При этом принцип формирования команды остается прежним. После записи данных в БП освобождается место в ППД для записи следующих данных. По аналогии с предыдущим вариантом, при отказе $ВМ_j$ он будет блокирован, а данные из $РВ_j$ будут переданы обратно непосредственно в БП, что даст возможность реализовать следующую попытку выполнения команды.

Таким образом, при таком подходе ресурсы системы используются более эффективно. Нет необходимости сохранять фрагменты команды в СФК. Ячейки ППД могут использоваться для других алгоритмов или другой итерации. Это в свою очередь разрешает совмещать вычислительные процессы не только на уровне команд, но и на уровне алгоритмов, что обеспечивает сокращение времени преобразования информации.

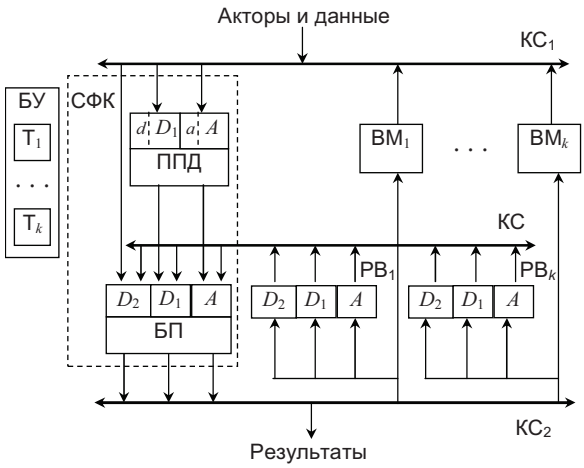



***Рис. 3. Организация системы с ППД по второму варианту***

Тем не менее, с точки зрения затрат времени на реконфигурацию системы, при отказе ВМ данный подход, как и рассмотренные ранее, является недостаточно эффективным. Действительно, время ожидания срабатывания таймера может быть гораздо больше, чем время, реально необходимое для выполнения команды. Это вносит дополнительную временную задержку при реконфигурации системы.

*Третий вариант.* Для решения вышеуказанной проблемы формат команды, передаваемой в ВМ, преобразуется к виду: $\langle A, D_1, D_2, T \rangle$, где $T$ – время ожидания результата выполнения конкретной команды (рис. 4). Фактически $T$ и $A$ могут рассматриваться как один информационный объект. Часть РВ выполнена в виде таймера, который запускается непосредственно с записью команды в РВ и ВМ. При этом необходимость таймеров в БУ отпадает.

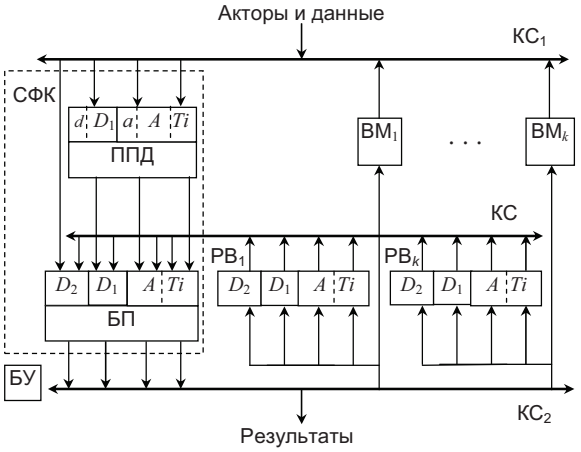



***Рис. 4. Организация системы с ППД по третьему варианту***

## Сравнительная оценка систем с разной организацией СФК

Механизмы повторного выполнения команды при отказе ВМ, описанные во втором и третьем варианте организации систем с ППД, могут быть применены и для системы с АП (рис. 1), то есть системы с АП и ППД имеют сравнимые функциональные возможности. В связи с этим целесообразно сравнить системы по быстродействию и аппаратурным затратам.

В рассматриваемых системах используются разные структуры СФК. Поскольку БП могут иметь одинаковую организацию, то основное отличие СФК заключается в организации АП и ППД. Как известно, АП уступает в быстродействии ППД и требует для построения больших аппаратурных затрат. Это объясняется тем, что кроме адресных операций АП должна выполнять ассоциативный поиск ячеек по признакам. Это, в свою очередь, требует более сложной организации как структуры памяти в целом, так и запоминающих ячеек. Из-за этого увеличивается и цикл обращения к памяти.

Таким образом, системы с ППД содержат меньше оборудования и являются более быстродействующими, чем системы с АП.

## Выводы

Предложенные методы реконфигурации и восстановления систем при отказе ВМ реализуются на аппаратном уровне без расхода дополнительных команд. Особенности архитектуры потоковых систем не требуют замены отказавшего модуля резервным путем его подключения в определенное место системы. Если все модули имеют одинаковые функциональные возможности, то команда, которая не выполнилась в отказавшем ВМ, повторно выполняется в другом ВМ, работающем в обычном режиме. В системе сохраняется полная информация для осуществления повторного выполнения команды. Таким образом, реконфигурация системы после отказа ВМ в основном сводится к отключению отказавшего модуля, что не требует больших затрат времени.

При построении систем может быть использован основополагающий принцип построения отказоустойчивых систем – принцип расширяемого ядра, защищенного аппаратными методами повышения надежности [12]. В качестве защищенного ядра в рассматриваемом случае может служить среда формирования команд. Данный компонент системы реализован на основе запоминающих устройств, для которых хорошо разработаны аппаратные методы контроля и исправления ошибок [10]. С учетом указанного предложенные методы автоматической реконфигурации потоковых вычислительных систем для разных сред формирования команд дают возможность продолжать вычисления до тех пор, пока в системе остается хотя бы один работоспособный модуль. Это повышает достоверность результатов вычислений, что является важным фактором, определяющим эффективность применения вычислительных систем.

Таким образом, для реализации алгоритмов с мелкозернистой структурой в реальном времени могут быть эффективно использованы потоковые вычислительные системы, обеспечивающие автоматическое распределение операций между вычислительными модулями и автоматическую реконфигурацию системы при отказах оборудования.

## Список литературы


1. Жабин В.И. Архитектура вычислительных систем реального времени. – К.: ВЕК +, 2003. – 176 с.
2. Погребинский С.Б., Стрельников В.П. Проектирование и надежность моногопроцессорных ЭВМ. – М.: Радио и связь, 1988. – 168 с.
3. Диллон Б., Сингх И. Инженерные методы обеспечения надежности систем. – М: Мир, 1983. – 318 с.
4. Коваленко А.Е., Гула В.В. Отказоустойчивые микропроцессорные системы. – К.: Техніка, 1986. – 150 с.
5. Dennis J. B., Missunas D. P. A preliminary architecture for basic data flow processor// Proc. 2nd Annual Symp. Comput. Stockholm, May 1975. N. Y. IEEE. – 1975. – P. 126 – 132.
6. Johnson D. Data flow machines threaten the program counter// Electronic Design. – 1980. – N 22. – P. 255 – 258.
7. Silva J.G.D., Wood J.V. Design of processing subsystems for Manchester data flow computer // IEEE Proc. N.Y. – 1981. – Vol. 128, N 5. – P. 218 – 224.
8. Watson R., Guard J. A practical data flow computer // Computer. – 1982. – Vol. 15, N 2.– P. 51 – 57.
9. Жабин В.И.. Организация вычислений в системах, управляемых потоком данных // Вісник Національного технічного університету Украіни “Київський політехнічний інститут”, Інформатика, управління та обчислювальна техніка: Зб. наук. пр. – К.: Век+. – 1998. – №31. – С. 44-51.
10. Огнев И.В., Сарычев К.Ф. Надежность запоминающих устройств. – М.: Радио и связь, 1988. – 223 с.
11. Дек. пат. №7727 України, МКВ G06F 15/16, 15/76. Обчислювальній пристрій / І.А. Жуков, В.І. Жабін, І.А. Клименко, В.В. Ткаченко (Україна). – №20040907712: заявлено 22.09.2004; опубл. 15.07.2005. Бюл. №7. – 9 с.
12. Авиженис А. Отказоустойчивость – свойство, обеспечивающее постоянную работоспособность цифровых систем // ТИИЭР (пер. с англ.). – 1978. – Т. 66, №10. – С. 5-25.


Поступила в редакцию 14.12.2009

УДК 004.237 : 621.337.2

*КЛИМЕНКО І.А.,*
*САКАРА Н. А.,*
*ФЕДОРЧУК М.В.*

# РЕАЛІЗАЦІЯ КОНТРОЛЕРА ПРІОРИТЕТНИХ ПЕРЕРИВАНЬ ДЛЯ ОБЧИСЛЮВАЛЬНОЇ СИСТЕМИ З ВІДКРИТОЮ АРХІТЕКТУРОЮ

Стаття присвячена рішенню проблеми підвищення продуктивності обчислювальних систем реального часу за рахунок вдосконалення процедур обробки переривань на апаратному рівні. З метою гарантованого обслуговування переривань від великої кількості зовнішніх пристроїв за певній проміжок часу запропоновано використання розподіленого контролера пріоритетних переривань з різними типами дисциплін динамічної обробки запитів. За результатами експериментальних досліджень доведено, що застосування запропонованого підходу усуває недоліки систем з географічними абсолютними пріоритетами, та сприяє вирішенню проблеми підвищення продуктивності систем реального часу з відкритою архітектурою.

The paper is dedicated to solving the problem of increasing productivity real time computing systems by improving the procedures for handling interrupts in hardware. To guarantee interrupt service from a large number of external devices for a certain period of time requested for the new priority interrupt controller with various types of disciplines dynamic query processing. Experimental studies proved that the application of the proposed approach eliminates the disadvantages of geographical systems an absolute priority, and to address the problem of increasing the productivity of real-time systems with open architecture.

## Вступ

Підвищення продуктивності обчислювальних систем (ОС) реального часу є важливою задачею в області застосування автоматичних систем управління. Особливістю таких систем є функціонування в режимі реального часу, що вимагає забезпечення гарантій виконання жорстких часових вимог. Складність виконуваних задач підвищують вимоги до сучасних ОС реального часу і обумовлюють розробку нових апаратних та програмних засобів для підвищення ефективності обробки та обміну даними.

Огляд літератури в даній області показав, що підвищення продуктивності паралельних обчислювальних систем реального часу досягається застосуванням методів і засобів зменшення витрат часу на ініціалізацію і синхронізацію процесів обміну даними. Так в роботі [1] запропоновані методи автоматичного динамічного розпаралелювання обчислень на рівні команд, програмних модулів і програм у системах з різною архітектурою, метою яких є підвищення реакції систем управління на зовнішні події та розширення області їх застосування. В роботі [2] були запропоновані формули апаратно-орієнтованих алгоритмів швидких перспективних перетворень, адаптовані для програмно-апаратної реалізації процесорів реального часу, що задовольняють критеріям точності та швидкодії. Окрім того, однією з функціональних особливостей систем управління реального часу є необхідність обробки переривань від великої кількості зовнішніх пристроїв. Це потребує ефективної реалізації системи переривань. На програмному рівні зазвичай питання обробки переривань вирішуються за допомогою спеціалізованих операційних системи реального часу, які базуються на системі пріоритетів процесів і алгоритмів планування [3].

В даній роботі проблема підвищення продуктивності систем реального часу вирішується за рахунок модифікації архітектури системи переривань, що забезпечить гарантоване обслуговування переривань від великої кількості зовнішніх пристроїв на протязі фіксованого часу циклу управління.

## Постановка задачі

Цілі дослідження: забезпечення гарантованого обслуговування переривань від кожного зовнішнього пристрою та уникнення тупикових ситуацій, характерних застосуванню географічного пріоритету, за рахунок чого підвищення продуктивності обчислювальних систем реального часу.

Реалізація розподіленого контролера пріоритетних переривань (КПП) з фіксованими та динамічними рівнями пріоритетів запитів

на ПЛІС, виконання моделювання в САПР Quartus II.

### Особливості реалізації КПП з фіксованими та динамічними пріоритетами в ОС

З точки зору швидкодії обробка переривань на апаратному рівні є більш ефективною в ОС розглянутого класу [3]. Для обробки переривань на апаратному рівні використовуються спеціалізовані пристрої – централізовані та розподілені КПП. Перевагами розподілених контролерів переривань є невелика кількість ліній зв'язку у шині управління та простота нарощування зовнішніх пристроїв. Недоліками такої системи є велика частота звернень процесора до системної магістралі під час ініціалізації системи, а також використання фіксованих рівнів пріоритетів запитів, які не забезпечують гарантованого обслуговування заявок від зовнішніх пристроїв. Запити з низьким рівнем пріоритету, за великої інтенсивності запитів, можуть не виконуватися тривалий час, що може призвести до сповільнення обчислювального процесу, а іноді – до тупикової ситуації.

Досліджена обчислювальна система, структурна схема якої зображена на рис. 1. Система складається з процесора *P* (*Processor*) і зовнішніх пристроїв *PU* (*Peripheral Unit*), пов'язаних між собою загальною шиною *GB* (*Global Bus*). У системі реалізована обробка зовнішніх векторних переривань за допомогою розподіленого контролера переривань, у склад кожного *PU* входить блок обробки переривань *IB* (*Interrupt Block*). Готовий *IB* до обміну даними видає сигнал запиту переривання *IR* (*Interrupt Request*) на загальну лінію вимоги переривань *ID* (*Interrupt Demand*). Відповідний сигнал процесора підтвердження переривання *IA* (*Interrupt Acknowledgement*) поширюється послідовно через усі *IB*, що утворюють так називаний пріоритетний «ланцюжок» (*Daisy Chain*), пріоритети обробки запитів переривань від зовнішніх пристроїв залежать від їх географічного розташування по відношенню до процесора. Таким чином найвищий пріоритет має зовнішній пристрій, що розташований найближче до процесора, а самий останній у ланцюзі – має найнижчий пріоритет. Така система не забезпечує гарантованого обслуговування заявок від зовнішніх пристроїв на визначеному відрізку часу. Заявки з низьким рівнем пріоритету за великої інтенсивності заявок можуть не виконуватися тривалий час. Цього можна уникнути за допомогою забезпечення динамічних пріоритетів. В КПП з послідовним передаванням пріоритету в кожному такті максимальний пріоритет, тобто початок ланцюга, передається наступному у ланцюгу процесору. За цього максимальний час очікування обслуговування дорівнює ($n$ – 1) тактів, де $n$ – кількість зовнішніх пристроїв. За такого способу реалізації динамічних пріоритетів максимальний пріоритет (початок ланцюга) отримує зовнішній пристрій, наступний за тим, що був обслугований у поточному такті.

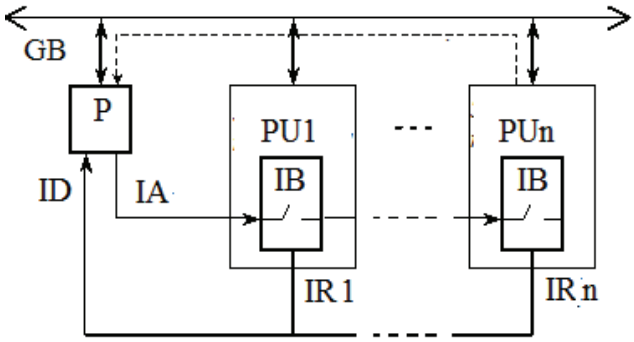



***Рис. 1. Структурна схема обчислювальної системи з фіксованими пріоритетами***

Обчислювальна система (рис. 2) містіть процесор *P*, *n* зовнішніх пристроїв *PU*, загальну шину *GB*, до якої підключені процесор *P* та зовнішні пристрої *PU*. До складу кожного зовнішнього пристрою входить блок формуванняпереривань *IB*, тригер *D,* елемент І (&) та елемент АБО (*I* ) (апаратура зовнішніх пристроїв, яка не стосується реалізації переривань на рис. 2 умовно не показана). Вихід тригера *D* підключений до першого входу елемента І, вихід якого зв'язаний з першим входом елемента АБО, який своїм виходом підключений до входу блока переривань *IB*. Вихід *NEXT_TR* кожного тригеру *D* приєднаний до інформаційного входу *DIN* тригера наступного зовнішнього пристрою, при цьому тригер *D* останнього зовнішнього пристрою *PU* підключений до входу першого зовнішнього пристрою *PU*. Таким чином, тригери *D* об'єднані у кільце. Вихід *NEXT_PU* кожного блоку переривань приєднаний до входу елемента АБО наступного зовнішнього пристрою, при цьому блок переривань останнього зовнішнього пристрою *PU* підключений до входу першого зовнішнього пристрою *PU*. Таким чином, блоки переривань *IB* через елементи АБО об'єднані у кільце. Виходи *IR* блоків переривань *IB* об'єднані у єдину лінію і підключені до входу вимоги переривань (*ID*) процесора *P*, вихід підтвердження переривання (*IA*) якого

підведений до керуючих входів тригерів *D* та других входів елементів *I*.

Обчислювальна система працює наступним чином. У кожний момент часу тільки в одному із тригерів *D* записана одиниця (під час ініціалізації – у тригері першого в пріоритетному ланцюзі *PU*). Всі інші тригери встановлені в нуль. Найвищій пріоритет має зовнішній пристрій *PU,* тригер пріоритету якого встановлений в одиницю. Готовий до обміну інформацією з процесором *P* будь-який зовнішній пристрій *PU* формує сигнал запиту переривання (*IR*). За наявності такого сигналу розривається пріоритетний ланцюжок між входом та виходом *NEXT_PU* блоку переривань *IB*. Якщо є хоч один сигнал на виходах *IR,* формується загальний сигнал вимоги переривань на вході *ID* процесора. Після закінчення чергової команди процесор у відповідь на сигнал *ID* формує сигнал підтвердження переривання на виході *IA*. Цей сигнал потрапляє у пріоритетний ланцюжок, замкнутий у кільце. При цьому початок ланцюжка визначає логічний елемент І, який відкривається високим рівнем сигналу в тригері збереження пріоритету, як вже було зазначено в такому стані знаходиться тільки один зовнішній пристрій системи. Сигнал *IA* розповсюджується по ланцюжку до першого на його шляху блока переривань *IB,* який виставив сигнал запиту переривання *IR*. В цьому блоці *IB* формується сигнал на виході *NEXT_PU*. За зрізом сигналу *NEXT_PU* формується сигнал дозволу видачі вектору переривання, який через загальну шину *GB* надходить в процесор *P*. Процесор переходить на виконання програми обслуговування переривання після чого знімає сигнал *IA*. Після зняття процесором сигналу *IA* і за фронтом сигналу *IA* одиниця з виходу *NEXT_TR* зовнішнього пристрою *PU* з найвищим пріоритетом переписується у тригер *D* наступного *PU*. У цьому випадку пріоритети передаються послідовно від одного зовнішнього пристрою до іншого. Що гарантує обробку переривань від всіх зовнішніх пристроїв на визначеному проміжку часу.

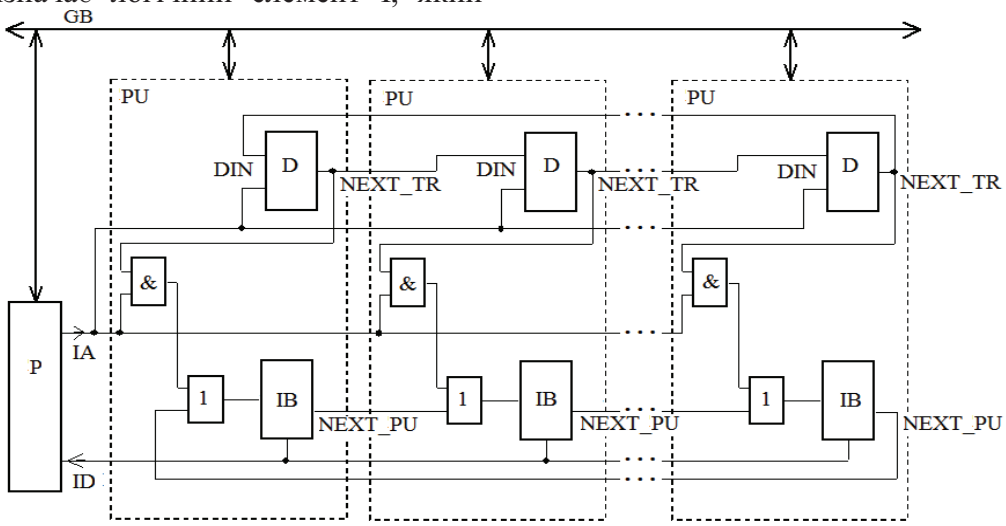



***Рис. 2. Обчислювальна система з послідовним передаванням пріоритету***

В обчислювальній системі з динамічним передаванням пріоритету (рис. 3) на відміну від обчислювальної системи з послідовним передаванням пріоритету, кожний блок переривань має третій вихід *NEXT_TR,* який приєднаний до інформаційного входу *DIN* тригера наступного зовнішнього пристрою, при цьому блок переривань останнього зовнішнього пристрою *PU* підключений до входу першого зовнішнього пристрою *PU*. Таким чином, блоки переривань *IB* через елементі АБО об'єднані у кільце. Виходи *IR* блоків переривань *IB* об'єднані у єдину лінію і підключені до входу вимоги переривань (*ID*) процесора *P*, вихід підтвердження переривання (*IA*) якого підведений до керуючих входів тригерів *D* та других входів елементів *I*.

Особливістю роботи обчислювальної системи з динамічним передаванням пріоритету є формування сигналу *NEXT_TR* та вектору переривання на виході блоку *IB*. Одиниця з виходу *NEXT_TR* переписується у тригер *D* наступного *PU*. Таким чином забезпечується перенос початку пріоритетного ланцюжка. У цьому випадку зовнішній пристрій, що обслуговується, одержує мінімальний пріоритет, а максимальний рівень пріоритету одержує наступний зовнішній пристрій.

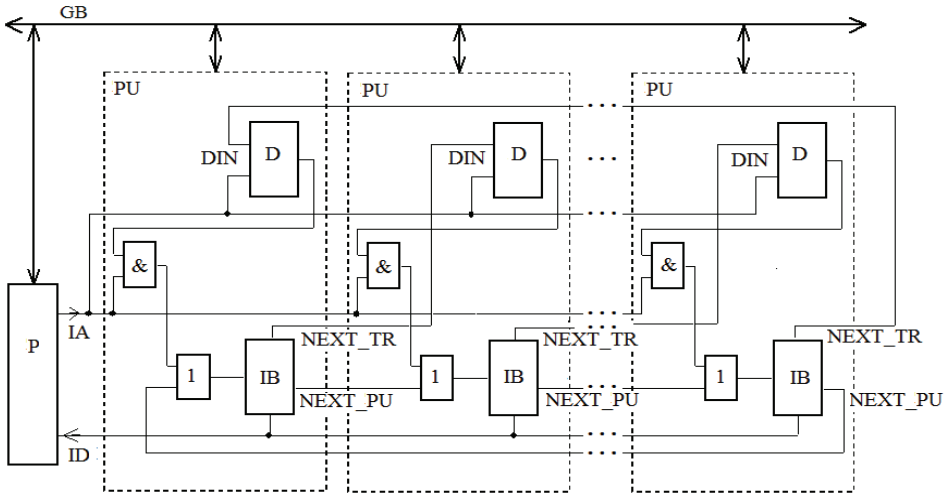



***Рис. 3. Обчислювальна система з динамічним передаванням пріоритету***

### Опис досліджуваних ОС на мові VHDL

Для моделювання роботи обчислювальної системи створений спеціальний «*test-bench*» рівень коду, який генерує вхідні параметри для проекту. *Test-bench* рівень представлений у вигляді наступних компонент: *PU_N* – блок, що емулює роботу п'яти зовнішніх пристроїв, *generator* – генератор запитів від зовнішніх пристроїв, що випадковим чином видає сигнали вимоги переривань, *save_request* – компонент для збереження поточного рівня запитів від зовнішніх пристроїв, proc – процесор.

Блок розподіленого контролеру переривань, що входить у склад кожного зовнішнього пристрою складається з наступних компонент: *d_triger* – тригер *D*, що зберігає максимальний пріоритет; *and*1 – елемент І, який управляє початком ланцюжка залежно від стану тригеру пріоритету, що надходить по лінії *next_tr*; *or* – елемент АБО, *and*2 та *notand*2 – блоки переривань відповідно до способу реалізації динамічних пріоритетів. Особливість функціонування системи з послідовним передаванням пріоритету: максимальний пріоритет передається послідовно з виходу тригера зберігання пріоритету поточного ЗП на вхід тригера зберігання пріоритету наступного у пріоритетному ланцюзі ЗП (лінія *next_tr*) і фіксується там за управління сигналу вимоги переривань (*signal_id*), незалежно від того який ЗП був обслуговуваний.

В системі з динамічним передаванням пріоритету в структурі компоненту *PU_N* максимальний пріоритет формується в блоці переривань зовнішнього пристрою, що був обслугований у поточному такту, та за лінією *next_tr* поступає на тригер зберігання пріоритету наступного у ланцюзі зовнішнього пристрою.

Розподілені контролери переривань синтезовані на мові *VHDL*.

### Результати моделювання

Виконано моделювання розподілених контролерів пріоритетних переривань з послідовним передаванням пріоритету та динамічним передаванням пріоритету в САПР *ModelSim*. В якості результати досліджень наведені часові діаграми роботи контролерів переривань.

Моделювання модулів КПП показало, що за наявності сигналу запиту переривання *IR* зовнішній пристрій *PU* отримує сигнал підтвердження переривання *IA*. Якщо на тригері *D* встановлено нульове значення, зовнішній пристрій *PU* переходить у стан очікування за чергою.

### Висновки

Застосування розподіленого КПП відповідає модульному принципу організації ОС і ефективно вирішує задачу масштабування обчислювальних систем з відкритою архітектурою, що до кількості зовнішніх пристроїв.

Застосування системи з динамічними пріоритетами гарантує обслуговування переривань від кожного зовнішнього пристрою на визначеному проміжку часу, що надає можливість уникнути тупикових ситуацій та простоїв і підвищити швидкодію системи.

Застосування контролерів з послідовним передаванням пріоритетів ефективно в обчислювальних системах з однорідним циклом функціонування. Тут кожний зовнішній пристрій на протязі ($n$-1) тактів роботи системи гарантовано отримає обслуговування. В сис-

темах що вирішують різного роду задачі управління в тому числі і траєкторні задачі, цикли управління характеризуються своєю неоднорідністю. Таким чином у визначений момент часу виконується опитування та обробка переривань від певної групи ЗП, що впливають на стратегію управління. Інші ж ЗП знаходяться у пасивному стані. В таких системах застосування КПП з послідовним передаванням пріоритетів буде обумовлювати затримку початку обслуговування переривань, що буде визначатись довжиною пріоритетного ланцюжка. Застосування КПП з динамічним передаванням пріоритету дозволяє підвищити ефективність обробки переривань в системах з неоднорідним циклом управління і великою кількістю зовнішніх пристроїв за рахунок „обминання” зовнішніх пристроїв, які не приймають участі на даному етапі управління.

Що до перспектив подальшого дослідження планується проведення наступного етапу розроблення розподілених контролерів переривань з динамічними пріоритетами для масштабованих обчислювальних систем у САПР *Quartus* II з урахуванням реальних параметрів мікросхеми, завантаження отриманого пристрою в ПЛІС та визначення реальних характеристик системи.

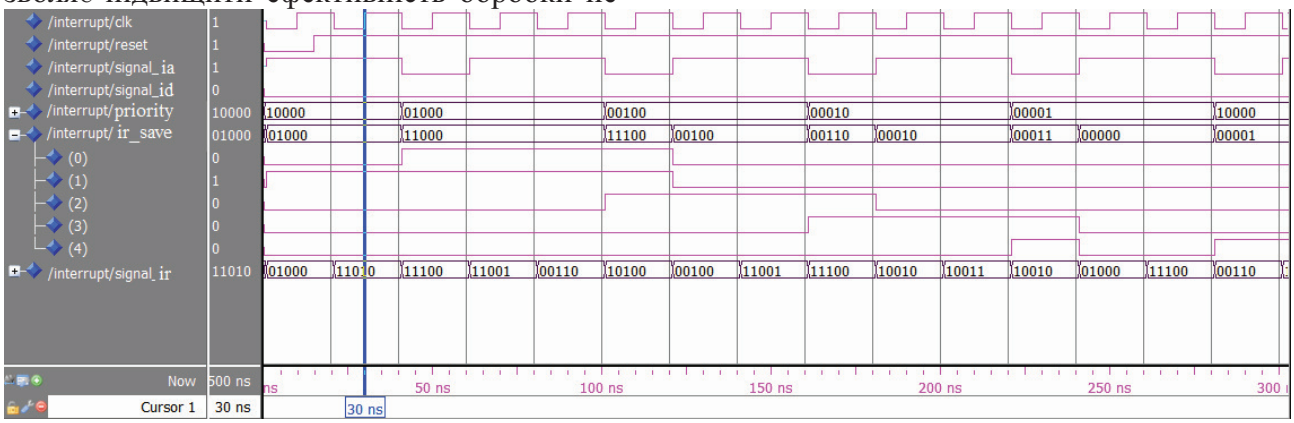



***Рис. 4. Часова діаграма обчислювальної системи з послідовним передаванням пріоритету***

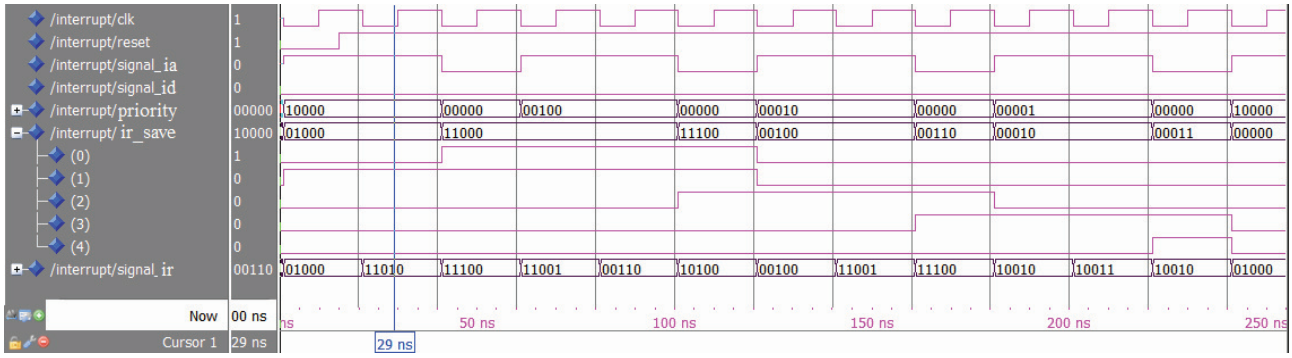



***Рис. 5. Часова діаграма обчислювальної системи з динамічним передаванням пріоритету***

## Список посилань

1. Автореф. дис. д-ра техн. наук: 05.13.13 / В.І. Жабін; Нац. техн. ун-т України "Київ. політехн. ін-т". — К., 2006. — 36 с. — укр.
2. Автореф. дис. канд. техн. наук: 05.13.13 / М.В. Васильєв; Харк. нац. ун-т радіоелектрон. — Х., 2003. — 19 с.: рис. — укр.
3. Жабин В.И. Архитектура вычислительных систем реального времени. – К.: ВЕК +, 2003. – 176 с.

Поступила в редакцию 7.12.2009

УДК 681.3.016

*ТЕЛЕНИК С.Ф.,*
*АМОНС О.А.,*
*ШКАБУРА О.Ю.,*
*ПОДРИГАЙЛО Н.О.*

# ПОШУК І РЕФЕРУВАННЯ В СИСТЕМІ ЕЛЕКТРОННОГО ДОКУМЕНТООБІГУ

Робота присвячена проблемі пошуку документів у масиві за атрибутами та на основі повнотекстового пошуку. Представлено модифікований метод рубрикації та метод реферування на основі рубрикації. Показано переваги використання цього підходу на прикладі системи електронного документообігу SmartBase.SEDO.

This work deals with the problem of document search in arrays by attributes and uses full-text search technology. Modification of rubrication method is presented and abstracting rubrication-based method is developed. The advantages of this conception usage is demonstrated on the electronic documents circulation system SmartBase.SEDO.

## Вступ

Сьогодні обсяги інформації зростають швидкими темпами, причому левова частка надходжень забезпечується за рахунок неструктурованої інформації. Склалася ситуація, за якої фахівці більше часу витрачають на пошук інформації, ніж на її використання за призначенням. Щоб зарадити цій ситуації в системах автоматичного та автоматизованого оброблення великих масивів неструктурованої інформації широко використовуються моделі та алгоритми рубрикації та реферування інформації. Такі ж проблеми притаманні системам електронного документообігу, які поступово стають незамінними у міністерствах, відомствах та великих корпораціях. До того ж у цих системах час пошуку документів, які можуть бути корисними для прийняття рішень, часто суттєво обмежений, причому апріорі відомою може бути лише тема документу.

У статті розглядається проблема пошуку інформації у системах електронного документообігу. У цих системах необхідно підтримувати пошук документів за атрибутами, на основі ключових слів та повнотекстовий пошук. Для прискорення процесу повнотекстового пошуку інформації в системах електронного документообігу пропонуються модифікації методу опорних векторів для рубрикації та методу реферування документів на основі рубрикації.

Експериментальні результати, одержані за допомогою реалізованої на основі запропонованих методів пошукової системи у складі системи електронного документообігу SmartBase.SEDO, підтвердили їх ефективність.

## Системи електронного документообігу, реферування і рубрикація

Документ – це матеріальний носій із зафіксованою на ньому інформацією. Таким носієм може бути папір, диск комп'ютера, фотоплівка і т.п. Діловий документ служить для фіксування адміністративної (управлінської) інформації. У нашому випадку це електронний документ (ЕД). Вважатимемо, що ЕД – це сукупність вмісту і службової інформацію щодо документа у вигляді електронної реєстраційно-контрольної картки документа (ЕРКК). Інформація ЕРКК документа представлена в системі окремим програмним класом і зберігається в відповідній таблиці бази даних. Для ефективної автоматизації функцій опрацювання ЕД необхідно закріпити за документом чітку формалізовану структуру у вигляді шаблону. Надалі шаблоном будемо називати об’єкт, який описує сталі частини інформації документів, створених за даним шаблоном, та поля змінної інформації з описанням методів і механізмів їх заповнення.

Об’єкт шаблон містить загальну інформацію, вміст (так звана статична інформація), перелік усіх елементів (тобто поля змінної інформації) та набір методів і механізмів, які заповнюють елементи документа. Уся наведена інформація представлена відповідними класами і зберігається в організаційній базі даних. Шаблон документа можна розглядати

як заздалегідь підготовлений макет документа. Використання шаблонів дозволяє заощаджувати час на формування та оброблення документів і досягти певного рівня уніфікації і стандартизації документів.

Рубрикація документів – це розповсюджена технологія впорядкування інформації щодо вмісту документа. Процес рубрикації полягає в описанні змісту документа шляхом використання елементів визначеного скінченого переліку тем (рубрикатора).

Тема – це сукупність ключових слів чи словосполучень з вказаними для них деякими параметрами. Вага теми – це числове значення, яке присвоюється кожній темі в залежності від таких критеріїв:

- кількість входжень слів чи словосполучень в текст;
- входження слів чи словосполучень в запит користувача;
- входження слів чи словосполучень в тему документа;
- входження слів чи словосполучень в поле «Короткий зміст документа».

Система електронного документообігу оперує n·1000000 документів на різні теми, різного обсягу та складності. Для ознайомлення с таким обсягом інформації необхідно витратити велику кількість часу. Ось чому проблемам автоматичного реферування та рубрикації приділяється так багато уваги.

### Огляд існуючих рішень

За останні роки з'явилося багато публікацій, в яких розглядаються проблеми автоматичного реферування. Навіть саме поняття «Автоматичного реферування» стало більш об'ємним і тепер стосується не лише текстової інформації а й мультимедіа у цілому. Поряд з традиційними задачами створення рефератів для окремих документів з'явилося нове поле діяльності, пов'язане з багатомовним реферуванням багатьох документів. В оглядовій статті [1] наведений детальний огляд стану автоматичного реферування. На сайті постійно діючої конференції DUC [2] можна побачити останні публікації в області автоматичного реферування, порівняльні характеристики та результати тестування різних систем. Методи, які використовуються для реферування текстової інформації, можна поділити на два напрямки: квазіреферування та генерування рефератів. Перший напрям включає в себе методи, які базуються на виділенні з тексту найбільш інформативних частин (речень), що передають головний зміст тексту документа. Другий напрямок націлений на виділення найбільш інформативної частини тексту з наступним генеруванням на її основі нового тексту. Практично усі сучасні системи реферування відносяться до напрямку квазіреферування, хоча останнім часом з'явилися цікаві роботи, в яких розвиваються технології генерування рефератів [3].

Більшість існуючих методів обох підходів до реферування базується на запропонованій Г.Луном концепції [4]. Сутність зазначеної концепції полягає у послідовному виділенні у тексті слів, яким певним чином приписується вага, визначенні ваги речень шляхом сумування ваги слів, що входять до його складу, та включенні в реферат речень з найбільшою вагою. Для сучасних методів реферування характерною ознакою є використання традиційного підходу з деякими модифікаціями. Наприклад, в якості значущих елементів вибираються не слова, а словосполучення [5], вводяться додаткові критерії відбору значущих слів, наприклад вага слова збільшується в залежності від його знаходження в заголовку, в першому чи останньому реченні або у запиті користувача [6] тощо.

У праці [7] пропонується концепція фрактального реферування для подачі контенту на мобільні прилади, які звичайно характеризуються малими розмірами екрану та обчислювальною потужністю, низькою швидкістю передачі даних. Запропонований підхід використовує традиційні методи виділення речень, але додатково враховує інформацію про ієрархічну структуру документу та потрібний «рівень абстракції» представленого документу.

Методи симетричного реферування, описані у працях [8], враховують, насамперед, зв'язки між реченнями, причому важливішими вважаються речення, які містять багато зв'язків. Зв'язки між реченнями виявляються за допомогою готового словника термінів предметної області, що дещо обмежує застосування цих методів.

Підхід на основі різноманіття запропонований у праці [9]. Ідея підходу полягає у тому, щоб спочатку знайти тематичні кластери документа (тобто групи речень, які належать

до однієї підтеми документа). Після цього важливі речення виділяються з кожного кластера шляхом застосування традиційних методик. Кластеризація речень виконується за допомогою модифікованого методу $k$ – середніх. Цей метод відносить кожне навчальне спостереження до одного з $k$ кластерів (де $k$ – число радіальних елементів) таким чином, щоб кожен кластер був представлений центроїдом відповідних спостережень, а кожне спостереження знаходилося на меншій відстані від центроїду свого кластера, ніж відстань до центроїда всіх інших кластерів.

Відомі також підходи до реферування на основі попередньо виконаної тематичної кластеризації документа з подальшим виділенням ключових речень з кожного кластера [9]. У відповідності з попереднім розбиттям документа на частини з урахуванням структури документа, на основі кластеризації здійснюється побудова реферату для кожної з частин і відбір найбільш важливих з них [10].

Важливою складовою задачі реферування є оцінювання якості одержаного реферату. Цій проблемі присвячено багато праць, наприклад [11,12]. Окрім традиційних методик, пов'язаних з експертними оцінками якості рефератів, останніми роками розвиваються автоматичні методи оцінювання. Наприклад, у праці [12] пропонується оцінювати якість рефератів за наявністю в них частотних словосполучень з оригіналу і близькістю розподілів частот появи цих словосполучень в документі та рефераті.

Недоліком існуючих методів реферування, побудованих на застосуванні рубрикації, є рознесення цих процесів у окремі процедури. Оскільки процеси рубрикації і реферування містять у собі спільні процедури оброблення текстів, наприклад, морфологічний чи концептуальний аналіз, можна суттєво пришвидшити процес реферування за рахунок його тісної інтеграції з процесом рубрикації. Було б доцільним зберігати теми разом з документом і використовувати у подальшому для визначення тем в документах, створених на основі цього ж шаблону, оскільки кожний шаблон значно звужує тематику документів створених на його основі.

## Постановка проблеми

Нехай заданий масив з $n$ документів $D = \{d_i,\ i = 1,.....n\}$. Необхідно розробити підсистему пошуку документів у масиві за атрибутами, на основі ключових слів та на основі повнотекстового пошуку. Для вдосконалення пошуку необхідно розробити систему автоматичного реферування. Повнотекстовий пошук має здійснюватися на основі рефератів. Реферування нових документів повинно виконуватися паралельно з рубрикацією на основі існуючого рубрикатора. Результатом цього процесу повинні бути реферати нових документів і оновлений рубрикатор.

## Загальний опис підходу

Підхід, який пропонується, базується на ідеї тісної інтеграції процесів реферування та рубрикації. Процес реферування, відповідно і процес пошуку, суттєво пришвидшується за рахунок раціонального використання спільних процедур оброблення текстів зазначених процесів, наприклад, морфологічного чи концептуального аналізу. Так, тему пропонується зберігати разом з документом і використовувати у подальшому для визначення тем в документах, створених на основі цього ж шаблону. Саме застосування шаблонів, традиційне для систем електронного документообігу, підвищує ефективність пошуку, значно звужуючи тематику документів, створених на їх основі.

Для виконання рубрикації документів пропонується використовувати модифікацію методу опорних векторів (SVM).

Після виконання процедури рубрикації масиву документів $D$ кожному документу приписується множина пар, першим елементом яких є номер теми, а другим – її вага в документі. Кожна з тем множини $T$ визначається множиною ключових слів та словосполучень, які містяться у тексті документу і в описі теми, і частотами появи цих слів та словосполучень і описується відповідним масивом двійок, першим елементом яких є слово або словосполучення з документу, яке визначає тему, другим елементом – частота появи в документі першого елементу.

При створенні нового документа, у ньому визначаються теми, що дозволяє одночасно будувати реферат документу і поповнювати рубрикатор.

### Модифікація метода опорних векторів

Реферування виконується паралельно з рубрикацією. Для виконання рубрикації документів, як зазначено вище, будемо використовувати модифікацію методу SVM.

Модифікацію виконуємо з метою підвищення точності рубрикації, оскільки звичайний метод SVM виконує класифікацію лише до того моменту поки не отримає позитивного класу ознак. Модифікація полягає у тому, щоб відділяти одну рубрику від усіх інших. Тобто для класифікації по 5 рубрикам буде проводитися навчання відразу 5 рубрикаторів. Це дозволить нам відносити тему не до першої позитивної рубрики, а до рубрики з найбільшою релевантністю.

Алгоритм модифікованого методу рубрикації:

1. Аналізується новий документ;
2. Для кожного з уже існуючих кластерів, перебираються теми, які до нього належать. Для кожної теми будується вектор ознак.
3. Виконується класифікація векторів ознак. У випадку, якщо вектор відноситься до позитивного класу, нова тема відноситься до того кластеру, до якого належить тема, для якої був отриманий вектор ознак. Якщо тема не була віднесена до жодного з існуючий кластерів, то на основі цієї теми будується новий кластер.

сервиса VoIP.

Після виконання процедури рубрикації масиву з $n$ документів $D = \{d_i\}$, $i = 1,.....n$, кожному документу $d_i$ приписується множина пар $TW_i = \{(t_1,w_1),\ldots,(t_j,w_j),...,(t_m,w_m)\}$, де $t_i$, $w_i$, $j = 1,...,m$, – відповідно номер теми і її вага в документі. Нехай $T$ – набір тем документа $d_i$, $W$ – набір ваг тем в документі $d_i$ (сумарних частот появи в тексті словосполучень, які містяться у описі теми з набору $T$).

У свою чергу, кожна з тем $t_j \square T$ визначається множиною ключових слів та словосполучень, які містяться у тексті документу $d_i$ і в описі теми $t_j$ і частот появи цих слів та словосполучень. Кожна тема $t_j$ описується масивом двійок PF = $\{(p^{(1)}_{ij},f^{(1)}_{ij}),...,(p^{(k)}_{ij},f^{(k)}_{ij}),\ldots,(p^{(l)}_{j\,ij},f^{(l)}_{j\,ij})\}$, де $p^{(k)}_{ij}$ – слово або словосполучення з документу $d_i$, яке визначає тему $t_j$; $f^{(k)}_{ij}$ – частота появи в документі $d_i$ слова чи словосполучення $p^{(k)}_{ij}$; $l_j$ – кількість слів чи словосполучень, які описують тему $t_j$.

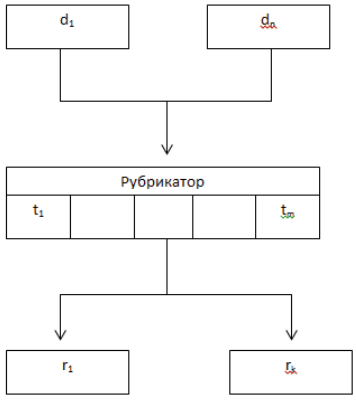



***Рис. 1. Схематичне представлення роботи механізму***

При створенні нового документа, у ньому визначаються теми. Якщо чергова визначена тема відсутня в рубрикаторі, то останній поповнюється, а тема з відповідною вагою приписується документу. Якщо ж чергова тема вже зафіксована у рубрикаторі, то останній не поповнюється, але тема з її вагою приписується документу.

### Методи реферування на основі рубрикації

Вибір найбільш інформативних речень з тексту документа $d_i$ пропонується здійснюється за допомогою такого модифікованого алгоритму:

Крок 1. Для кожної теми формується список речень, який її характеризує. Ця інформація береться з рубрикатора.

Крок 2. Усі речення списку оброблюються з метою вилучення стоп-слів з використанням словника стоп-слів, який містить службові частини мови, а також неінформативних слів та словосполучень.

Крок 3. Визначається вага кожного речення шляхом підсумовування частот появи у ньому слів та словосполучень, які визначають тему. Вага залежить від місця, де це речення вживається – на початку, кінці абзацу чи у ключових частинах, наприклад у висновках чи вступі.

Крок 4. У кожному з відібраних на кроці 1 речень видаляються примітки (слова у дужках) та деякі звороти за допомогою спеціального словника зворотів (різниця між словником зворотів та словником стоп-слів полягає у тому, що неінформативні слова повністю задаються списком, а в словнику зворотів присутня лише початкова частина речення, наприклад “Як повідомляється…”).

Після розпізнавання звороту з речення тексту вилучається не лише та частина, яка наведена в словнику, а й увесь зворот до наступного розділового знака.

Крок 5. Усі речення після оброблення на попередніх кроках (вилучення стоп-слів, приміток чи зворотами), які були кандидатами на включення до реферату, послідовно перевіряються на тотожність. Речення вважаються близькими, якщо вони мають 80% спільних слів. З усіх близьких речень в реферат включаються речення з найбільшою вагою.

### Пошук

У системі електронного документообігу SmartBase.SEDO реалізовані такі типи пошуку:

1. атрибутивний пошук: пошук за різними комбінаціями відомих значень полів ЕРКК документу, який потрібно знайти, наприклад, дата створення документу, назва документу, відправник; автор документу тощо;
2. пошук по ключовим словам: дозволяє проводити пошук за відомими особі, яка здійснює пошук, ключовим словам у обсязі реферату;
3. повнотекстовий пошук: дозволяє проводити пошук за відомими особі, яка здійснює пошук, змістовними елементами документу (темами).

Метою цієї статі було розробити такий алгоритм пошуку, який би дозволяв здійснювати швидкий пошук, відштовхуючись від інформації, що вже міститься у системі. Саме ці особливості пошуку важливі для електронного документообігу. Оскільки повнотекстовий пошук є трудомістким, а тематика, за якою потрібно здійснювати пошук, може бути зовсім не освітленою в контенті конкретної системи, то звичайні методи пошуку вимагають не виправдано великих ресурсів і не забезпечують вимог систем електронного документообігу до оперативності. Пошукова підсистема для електронного документообігу, розроблена на основі попереднього реферування та рубрикації, дозволяє досить ефективно розв'язати ці проблеми.

Щоб перейти до реалізації запропонованого підходу розглянемо вимоги до функціональності пошукової підсистеми з боку користувача. Користувач, який здійснює пошук з використанням запропонованого підходу, повинен отримати у своє розпорядження такий функціонал:

- можливість вибору користувачем у рубракаторі тем, що його цікавлять;
- створення рефератів для документів, які попадають у систему електронного документообігу;
- відбір документів за вибраними користувачем темами і видача користувачеві рефератів усіх відібраних документів (у визначеному користувачем порядку);
- можливість надання користувачу повного тексту документа після ознайомлення з рефератом (на вимогу користувача).

### Реалізація мехазму

Для програмної реалізації механізму пошуку документів у системі електронного документообігу на основі реферування та рубрикації документів розроблена узагальнена об'єктна модель, наведена на рис. 1. Вона не прив'язана до конкретного середовища розроблення або мови програмування. На основі цієї моделі створено відповідні таблиці в системі управління базами даних (СУБД) Oracle і програмні класи. Шаблони документів та самі документи зберігаються в базі даних (БД), що дозволяє працювати з ними територіально відокремленим групам користувачів.

Основу механізму роботи з шаблонами документів складають вісім класів, які забезпечують базову функціональність пошукової підсистеми: «Тип документа», «Організація», «Електронна реєстраційно-контрольна картка», «Шаблон документа», «Опис елемента», «Статичний вміст шаблону документа», «Елемент документа», «Джерело даних для заповнення елемента».

Клас «Тип документа» розрізняє всі документи і шаблони документів за призначенням, а клас «Організація» визначає організацію, до якої належить користувач. За допомогою цих двох класів визначаються шаблони документів, доступних користувачеві.

Клас «Шаблон документа» використовується для описання і оброблення шаблонів документів. Кожний шаблон має унікальний ідентифікатор, а також набір полів, наприклад тип, назва, додаткова інформація тощо. Цей клас забезпечує збереження шаблону документа у БД і генерацію документа за вибраним шаблоном. Елементи шаблону,

спільні для всіх документів, створюваних за цим шаблоном, зберігаються у БД. Усі дані шаблону, які будуть унікальними для конкретних документів, описуються відповідними графічно-програмними елементами. Ці елементи програмно закріплюються в шаблоні документа і зберігаються в його структурі за допомогою класу «Опис елемента». Кожний елемент шаблона пов'язаний з класом «Шаблон документа» за допомогою унікального ідентифікатора шаблона. У системі реалізовано такі типи елементів як текстове поле, дата, випадаючий список та ін. Тип елемента визначається класом «Опис елемента».

Кожний елемент шаблону може бути заповнений автоматично або залишений незаповненим. Заповнення елементів шаблону можна визначати за допомогою логічних правил. Наприклад, це дуже зручно для введення дати. Для реалізації цієї ідеї розроблений механізм автоматичного заповнення елементів за допомогою класу «Джерело даних для заповнення елемента». Автоматично перевіряються типи елементів та функцій.

Окрім готових функцій можна використовувати формули заповнення. Логічний апарат оброблення формул, виконання необхідних розрахунків і заповнення елементів результатами обчислень описано в окремому класі «Менеджер формул». Формули корисні у випадку необхідності виконати певні математичні операції над визначеними даними в документі. Типовим прикладом може бути сума усіх значень визначених полів таблиці.

Клас «Статичний вміст шаблона документа» відповідає за збереження шаблону документа в БД. Вся статична структура шаблону документа створюється в форматі MS Word, архівується й упаковується в байтовий формат. В упакованому вигляді весь шаблон зберігається в одному полі БД.

Після створення шаблону документа, на його базі можна створювати електронні документи за допомогою класу «Електронна реєстраційно-контрольна картка». Користувачу необхідно лише вибрати відповідний шаблон і визначити потрібні опції, наприклад електронний цифровий підпис, затвердження, тощо. Елементи електронного документа описуються класом «Елемент документа», який також відслідковує всі зміни елементів конкретного документа після його редагування і відповідає за заповнення елементів документа.

Для реалізації механізмів реферування та рубрикації в систему вводяться додаткові класи «Анотація» та «Кластер». Перший з них забезпечує аналіз змісту документа та його збереження у БД разом з самим документом. Другий клас забезпечує збереження всіх тем документів, створених на основі визначеного шаблону. До кожної теми також зберігаються слова та їх ваги.

Розглянемо роботу механізму більш детально. У процесі створення електронного документа спочатку формуються загальні відомості про документ, насамперед тип, назва й опис. На основі аналізу ряду показників, насамперед організації користувача і типу документа, формується відповідний список шаблонів документа. Після вибору шаблону документа користувачем він завантажується з БД, при цьому всі елементи, що мають активний метод заповнення, заповнюються відповідними даними. Генерований таким чином електронний документ зберігається у БД, над ним виконуються дії відповідно до визначених користувачем опцій. Варто зазначити, що користувач має можливість у будь-якому документі відповідно до своїх прав створювати нові елементи або вилучати непотрібні елементи, змінювати методи їх заповнення. Створення електронного документу завершується його автоматичним анотуванням (із застосуванням рубрикації) і реферуванням.

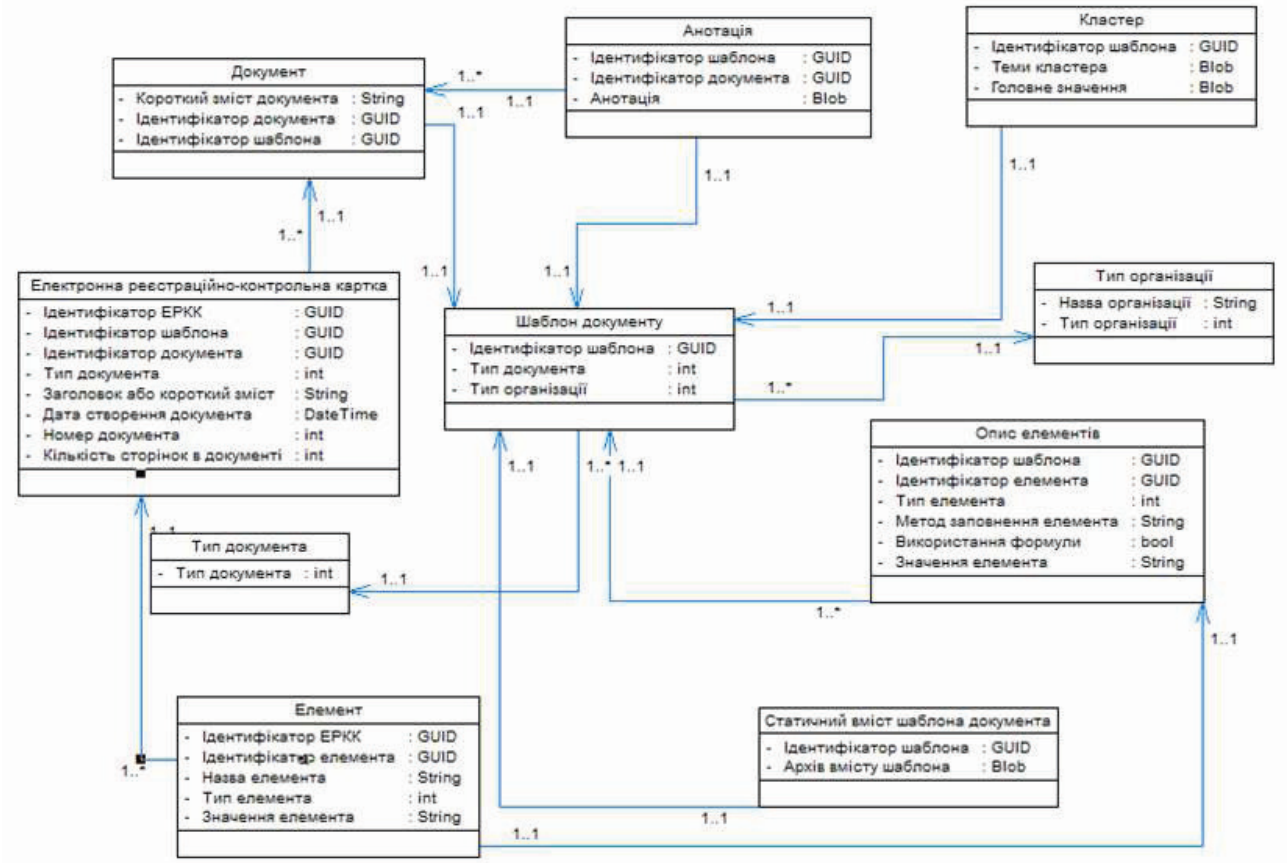



***Рис.2. Об'єктна модель механізму анотування та реферування***

## Результати експеремента

Для визначення впливу механізму реферування на ефективність пошуку в системі електронного документоообігу були проведені його експериментальні дослідження. Вони дозволили одержати цікаві результати. Визначимо вихідні та результуючі параметри дослідження. Вихідними параметрами будуть основні компоненти електронного документообігу – множина документів і структура шаблону. В якості результуючих характеристик будемо відслідковувати час, витрачений на пошук з використанням реферування, та час реферування.

У цьому дослідженні використовувався пошук документів у системі електронного документообігу з використанням механізму попереднього реферування і без використання попереднього реферування. Результати наведені на рис. 3.

Експериментальні дослідження виконувалися на тестових документах, створених для тестування механізмів і підсистем системи електронного документообігу.

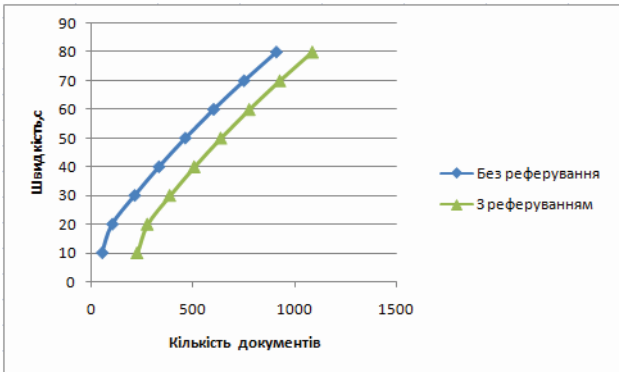



***Рис. 3. Діаграма дослідження швидкості при пошуку документів***

Наведені на рисунку результати дослідження дозволяють зробити висновок, що пошук з використанням реферування працює швидше – система за той самий час спроможна обробити більшу кількість документів. І чим більша кількість документів, що обробляються, тим більше часу при цьому економиться. Про це свідчить діаграма залежності часу реферування від кількості документів, наведена на рис. 4.

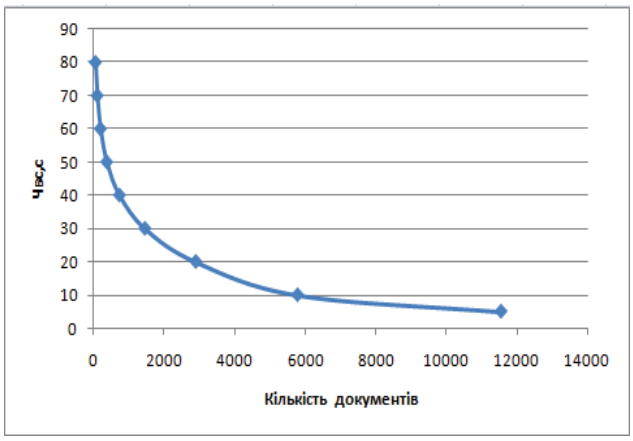



***Рис. 4. Часова діаграма роботи механізму реферування***

Це ще одне підтвердження ефективності використання механізму реферування. Досвід впровадження SmartBase.SEDO у системах електронного документообігу великих міністерств і відомств, де накопичуються сотні тисяч різноманітних документів, продемонстрував високу ефективність пошуку на основі реферування. Чим більше документів вже пройшли процес реферування, тим менше часу йтиме на реферування інших типових документів.

## Висновки

Запропонований у статті метод пошуку може бути з успіхом застосований у сучасних інформаційних та інформаційно-аналітичних системах для оброблення великих масивів документів. Попереднє автоматичне реферування документів зменшує витрати у перерахунку на один документ і дозволяє значно скоротити час пошуку документів.

## Список посилань

1. Hahn U., Mani I. The Challenges of Automatic Summarization // Computer.- 2000.- vol.33.- №11.- P. 29–36.
2. Document Understanding Conferences (DUC) Web site, 2008. http://duc.nist.gov.
3. Алыгулиев Р.М. Автоматическое реферирование документов с извлечением информативных предложений // Вычислительные технологии. – 2007. – Т. 12, № 5. – С. 5–15.
4. Luhn H. The automatic creation of literature abstracts // IBM Journal of Research and Development.- Vol. 2(2).- 1958.- P. 159–165.
5. Белоногов Г.Г., Калинин Ю.П., Хорошилов А.А. Компьютерная лингвистика и перспективнse информационные технологии. – М.: Русский мир, 2004. – 246 с.
6. Браславский П.И., Колычев И.С. Автоматическое реферирование веб-документов с учетом запроса // Интернет-математика-2005. – М.: Яндекс, 2005. – С. 485–501.
7. Yang, Ch. C. , Wang, F. L. Fractal Summarization for Mobile Devices to Access Large Documents on the Web // Proc. of the WWW2003, May 20-24, 2003, Budapest, Hungary. P. 26–31 .
8. Ступин В. С. Система автоматического реферирования методом симметричного реферирования // Компьютерная лингвистика и интеллектуальные технологии. Труды международной конференции «Диалог'2004». (Верхневолжский, 2 – 7 июня 2004 г.). – М.: Наука, 2004. – С. 579-591.
9. Nomoto T., Matsumoto Y. The diversity-based approach to open-domain text summarization // Information Processing & Management.- 2003.- №39.- P. 363–389.
10. Губин М.В., Меркулов А.И. Эффективный алгоритм формирования контекстно-зависимых аннотаций // Компьютерная лингвистика и интеллектуальные технологии : Труды международной конференции «Диалог'2005» (Звенигород, 1 – 6 июня 2005 г.). – М. : Наука, 2005. – С. 116–120.
11. Harman D., Over P. The effects of human variation in DUC summarization evaluation // Text summarization branches out workshop at ACL'2004.
12. Tait J. Making Better Summary Evaluations // Proc. of the Internatinal Workshop "Crossing Barriers in Text Summarisation Research" / Horacio Saggion and Jean-Luc Minnel (eds.).- Borovets, Bulgaria, Septemeber 2005, P. 26–31.

Поступила в редакцию 16.12.2009

УДК 004.056

*ЗЮЗЯ А.А.*

# СПОСОБ ПРОТИВОДЕЙСТВИЯ РЕКОНСТРУКЦИИ КЛЮЧЕЙ БЛОКОВЫХ АЛГОРИТМОВ ЗАЩИТЫ ИНФОРМАЦИИ АНАЛИЗОМ ДИНАМИКИ ПОТРЕБЛЯЕМОЙ МОЩНОСТИ

В статье рассмотрены методы реконструкции ключей криптоалгоритмов с помощью анализа динамики потребляемой мощности. Проведен анализ существующих программных контрмер для таких видов атак. Предложен новый способ противодействия анализу динамики потребляемой мощности, основанный на организации псевдопараллельной обработки блоков в режиме разделения времени. Способ отличается стохастическим переключением между обрабатываемыми блоками для обеспечения случайной временной локализации этапов обработки, что значительно снижает эффективность реконструкции ключа с помощью анализа динамики потребляемой мощности. Предложенный способ реализован для алгоритма ГОСТ 28.147-89. При этом падение производительности меньше, чем при использовании других способов противодействия анализу динамики потребляемой мощности.

The article studies power analysis key reconstruction methods for cryptographic algorithms. The analysis of existing software countermeasures for such types of attacks was done. A new power analysis obstruction method was proposed. It is based on pseudo parallel data processing in time sharing mode. The method is characterized by a stochastic switching between proceeding blocks of data for random time location of block processing stages. It significantly reduces effectiveness of the power analysis key reconstruction methods. Proposed method was implemented in software implementation of GOST 28.147-89 cryptographic algorithm. Decrease in performance of such modified implementation is less then via using other power analysis obstruction methods.

## Введение

Рост интеграции информационных ресурсов на основе компьютерных технологий к началу 21-го века стал одним из решающих факторов поступательного развития всех сфер деятельности современного общества. Оборотной стороной расширения информационной интеграции является увеличение числа угроз нарушения конфиденциальности и безопасности данных. Для нейтрализации этих угроз широко используются системы защиты информации, в основе большинства из которых лежат криптографические алгоритмы. Современные алгоритмы криптографической защиты являются полностью открытыми (закрытым элементом является только ключ) и базируются на аналитически неразрешимых математических задачах. Соответственно, в большинстве случаев, для нарушения защиты данных необходимо узнать ключ.

Традиционно, для нарушения защиты – то есть реконструкции ключа используется перебор возможных его значений, при этом, для уменьшения объема перебора используются алгебраические свойства криптографических алгоритмов. Такие технологии нарушения защиты анализируют лишь математические аспекты алгоритма, абстрагируясь от вычислительной платформы, на которой реализуется система защиты данных.

В последнее десятилетие появились и активно развиваются новые технологии получения несанкционированного доступа к ключевой информации криптографических алгоритмов. Эти технологии базируются на получении информации о ключе криптографического алгоритма посредством анализа параметров технической реализации на конкретной вычислительной платформе. В литературе [1] такие технологии получения незаконного доступа к ключу алгоритмов защиты информации получили название атак по сторонним каналам. В отличии от традиционных методов, атаки по сторонним каналам учитывают особенности конкретной аппаратной реализации криптосистемы, а не только математического алгоритма, используемого в ней. Такое сужение специализации позволяет таким атакам достигать значительно более высокой эффективности при работе с конкретной криптографической системой [1].

Наиболее распространенной технологией реконструкции ключа алгоритма по параметрам его технической реализации является измерение и анализ динамики потребления мощности (АДПМ) вычислительным ус-

тройством, на котором выполняется алгоритм.

Для эффективного применения АДПМ важным является то, чтобы на устройстве реализовался только один вычислительный процесс. Это условие выполняется для микроконтроллеров, встроенных микропроцессоров и смарт-карт. Именно эти вычислительные устройства
являются основными объектами для применения методов реконструкции ключей криптографических алгоритмов с помощью АДПМ.

Учитывая, расширяющееся использование указанного класса вычислительных устройств в современных информационных технологиях, задача создания эффективных средств противодействия широкое несанкционированного доступа к ключам защиты посредством АДПМ является актуальной и важной.

## Технологии реконструкции ключей на основе АДПМ и анализ методов противодействия

Впервые метод реконструкции ключей криптографических алгоритмов с использованием АДПМ был предложен Полом Кочером [2] и получил широкое распространение благодаря комбинации таких факторов, как высокая эффективность, низкая стоимость и невозможность обнаружения системой защиты информации того, что она стала объектом атаки (ввиду пассивности и неинвазивности). АДПМ базируется на различии в потребляемой мощности устройством при выполнении одних команд на различных данных. В основе большинства известных технологий реконструкции ключей криптографических алгоритмов с использованием АДПМ лежит модель утечки, называемая ”моделью Хеммингового веса” (Hamming weight model) [1]. Согласно этой модели, потребляемая мощность $S$ (сигнал утечки) в момент обработки кода $D$ определяется следующим образом:

$$S = \delta + \beta \cdot H(D) + \eta \qquad (1)$$

где $\delta$ – постоянное смещение, $\beta$ – действительное число, $H(X)$ – Хеммингов вес кода $X$, $\eta$ – шум с математическим ожиданием 0 и среднеквадратичным отклонением σ.

Согласно этой модели потребление мощности пропорционально числу единиц в обрабатываемых данных (Хэммингов вес).

Приведенная модель – наиболее простая. существуют и другие модели описания зависимости мгновенного значения потребляемой мощности от обрабатываемого кода. Так на практике часто используется “модель дифференциала Хеммингова веса”, согласно которой мгновенное значение потребляемой мощности пропорционально числу изменяемых при обработке разрядов кода. Ясно, что степень адекватности той или иной модели существенным образом зависит от используемой при изготовлении микроконтроллера интегральной технологии. Поэтому оправданным представляется создание модели утечки для каждого конкретного типа микроконтроллера.

На сегодняшний день существует ряд методов использования АДПМ для реконструкции ключей криптографических алгоритмов. Эти методы разделяются на две группы [3]:

- простой АДПМ (SPA – Simple Power Analysis) и его развитие – профильный АДПМ (Profiling Analysis) основаны не непосредственном сопоставлении динамики потребления мощности с командами обрабатываемого криптографического алгоритма; в большинстве случаев простой АПМД позволяет установить последовательность выполнения команд и если она зависит от битов ключевого кода – восстановить его.
- дифференциальный АДПМ или Д-АДПМ (DPA – Differential Power Analysis) и его развитие Д-АДПМ высоких порядков (HO-DPA – High Order Differential Power Analysis), основанные на статистическом анализе данных о динамике потребления мощности для ряда вычислений алгоритма при меняющихся данных и постоянном ключе.

Для противодействия методам реконструкции ключей криптографических алгоритмов с помощью АДПМ используют программные и аппаратные способы. Суть аппаратных способов заключается в том, чтобы сделать неотличимой разницу в потребляемой мощности для различных данных путем использования специальных схемных решений, таких как балансные схемы или шумящие схемные компоненты [2].

Существующие к настоящему времени программные средства противодействия реконструкции ключей АДПМ-технологиями основаны на использовании случайных и псевдослучайных элементов при реализации

алгоритмов криптографической защиты. Базовыми критериями эффективности средств противодействия АДПМ-технологиями являются достигаемый ими уровень защищенности и объем дополнительно затрачиваемых вычислительных ресурсов. Последнее особенно важно с учетом того, что вычисления производятся на малоразрядных микроконтроллерах и смарт-картах, обладающих низкой производительностью и ограниченными ресурсами памяти. Для оценки достигаемого уровня защищенности от АДПМ-технологий в большинстве случаев используют упомянутые выше модели зависимости потребляемой мощности от обрабатываемого кода.

Основными направлениями при создании программных средств противодействия АДПМ-технологиям являются:

- маскирование – наложение на обрабатываемые данные и ключи случайных компонент перед обработкой и выполнение специальных вычислений для получения истинного результата – “снятие маски” после обработки. Маскирование применяют, преимущественно, для защиты ключей симметричных блоковых алгоритмов, таких как DES, Rijndael, ГОСТ 28.147-89 [5], в которых используются относительно простые операции. Существенным недостатком маскирования является возможность исключения влияния случайных масок путем специальной статистической обработки;
- случайное изменение порядка выполнения независимых по данным команд алгоритма (полиморфная реализация), что затрудняет сопоставление участков программы диаграмме потребляемой мощности. Полиморфная реализация алгоритма является эффективным средством противодействия простому АДПМ и наиболее часто используется для защиты ключей алгоритмов на основе мультипликативных операций модулярной арифметики – RSA, DSA, El-Gamal. Недостатком полиморфной реализации алгоритмов является необходимость значительных вычислительных ресурсах, требующихся для анализа зависимости по данным команд и реализации случайного их выполнения. Это существенно замедляет реализацию алгоритмов криптографической защиты данных на микроконтроллерах;
- разделение ключей, используемых в алгоритмах на несколько частей случайным образом с последующей раздельной их обработкой и компоновкой результата. Подход эффективен только для алгоритмов типа DES, IDEA, в которых используются только логические операции. Недостатком является также необходимость не менее, чем двукратном увеличении времени реализации криптографического алгоритма.
- случайное перемешивание выполнения операций над реальными и фиктивными данными; действенность подхода зависит от эффективности средств генерации фиктивных команд и данных ими обрабатываемых;
- использование сбалансированных кодов с постоянным числом единиц.

Указанные подходы могут использоваться как отдельно, так и комбинированно.

Эффективность этих способов в значительной степени зависит от вида АДПМ в используемом методе реконструкции ключей. Так, например, маскирование неэффективно против Д-АДПМ высоких порядков, если порядок анализа выше числа масок. Для обхода перемешивания используются интегральные модификации различных видов АДПМ. Общим недостатком рассматриваемых программных способов противодействия АДПМ-методам является снижение производительности криптографического алгоритма пропорционально числу масок при маскировании, либо фиктивных переменных при перемешивании.

Проведенный анализ показал, что наиболее действенным, с точки зрения достигаемого уровня защищенности от АДПМ, является динамичная модификация программной реализации криптографического алгоритма во время выполнения (полиморфизм кода). При этом последовательность команд для реализации криптографического алгоритма каждый раз будет различной, как и динамика потребления мощности. Это позволяет противодействовать как профильному АДПМ, так и различным Д-АДПМ (в том числе высоких порядков). При этом, эффективность применения такого механизма определяется общим числом возможных модификаций кода и величиной разницы временной локализации этапов обработки данных для различных модификаций.

Эффективность полиморфной реализации существенно ограничивается зависимостью команд по данных в большинстве криптографических алгоритмов, а также от степени ветвления алгоритма. Очевидно, что наиболее эффективно использование полиморфизма для реализации маловетвящихся алгоритмов, к числу которых относятся блоковые алгоритмы шифрования, такие как ГОСТ 28.147-89, DES, IDEA, Rijndael.

Целью исследований является повышение эффективности защиты от АДПМ-технологий в рамках полиморфной реализации криптографических алгоритмов блочного типа за счет увеличения временного диапазона выполнения его операций.

## Способ организации псевдопараллельной обработки блоков в криптографических алгоритмах

В качестве способа организации полиморфного исполнения для блочных криптографических алгоритмов предлагается использовать одновременную псевдопараллельную, в режиме разделения времени, обработку нескольких блоков данных.

Блочные алгоритмы состоят из $k$ циклов обработки данных $R_i(D,K_i)$, $i=1..k$, в каждом из которых производится некоторое преобразование блока $D$ данных с участием ключа $K_i$. В зависимости от типа блочного шифра цикл обработки может рассматриваться как некоторая макрооперация для сети Фейстеля либо последовательность макроопераций для SP-сети. В любом случае для преобразования блока данных выполняется конечная последовательность $P$ таких макроопераций, длина $K$ которой равна (сеть Фейстеля) либо кратна (SP-сеть) числу $k$ циклов обработки данных.

Для противодействия АДПМ необходимо реализовать полиморфное исполнение алгоритма, при котором указанная последовательность макроопераций будет меняться при каждом выполнении алгоритма. Предлагается использовать помирфизм на двух уровнях: на уровне макроопераций и на уровне цикла обработки данных (псевдопараллельная обработка алгоритмом нескольких блоков данных).

Полиморфизм макроопераций блоковых алгоритмов шифрования основан на том, что длина блока существенно превышает разрядность обрабатываемого фрагмента. При этом, каждый из фрагментов может обрабатываться независимо. Так, в большинстве блочных алгоритмов нелинейное преобразование выполняется с использованием ряда табличных преобразователей (S-блоков), каждый из который преобразует относительно небольшое число разрядов кода. Так в DES используется 16 S-блоков, каждый из которых независимо обрабатывает 4 разряда. В силу наличия независимо обрабатываемых фрагментов существует возможность менять очередность их обработки, то есть организовать полиморфную обработку фрагментов. Выбор очередности обработки фрагментов может выполняться статически или динамически. В первом случае предварительно формируются и сохраняются в памяти ряд вариантов программной реализации цикла преобразования, отличающихся последовательностью обработки фрагментов. Выбор варианта выполняется случайно в начале каждого выполнения цикла криптографического преобразования. При динамическом полиморфизме программные фрагменты-заготовки имеют меньшую длину (вплоть до команды) и их выбор осуществляется непосредственно во время выполнения цикла преобразования

Полиморфная, на уровне макроопераций, реализация криптографического алгоритма потенциально уязвима для интегральной разновидности АДПМ в которой интервал интегрирования равен длине макрооперации. Следовательно для нейтрализации такого вида анализа необходимо обеспечить различную временную локализацию макроопераций из P на разных циклах обработки блоков данных – полиморфизм уровня цикла обработки данных. Для этого предлагается использовать псевдопараллельную организацию обработки нескольких блоков данных в криптографическом алгоритме.

Возможность такой организации выполнения блоковых алгоритмов основана на том, что обработка блоков шифруемого (дешифруемого) сообщения выполняется независимо. Соответственно, сущность полиморфизма на уровне блоков сообщения состоит в том, что организуется одновременно-поочередная обработка нескольких блоков сообщения, причем выбор длина программного фрагмента обработки блока выбирается случайным образом. Псевдопараллельная организация подразумевает одновременную обработку $m$ блоков данных на меньшем числе вычислительных устройств (ВУ) – чаще всего на одном в режиме разделения времени. При условии, что ВУ только одно, такая ор-

ганизация вычислений не приводит к повышению производительности (традиционная цель распараллеливания), однако позволит смешивать выполнение $m$ последовательностей $P_j$, $j=1..m$ обработки различных блоков данных $D_j$, $j=1..m$.

При псевдопараллельном, в режиме разделения времени, исполнении последовательностей $P_j$, $j=1..m$ происходит их разделение на $q$ подпоследовательностей $P_{j,l}$, $l=1..q$ различной длины, которые могут чередоваться различными способами с сохранением порядка расположения в $P_j$. Пример такого псевдопараллельного выполнения блоков $P_1$, $P_2$ и $P_3$ графически показан на рисунке 1. При полиморфном выполнении криптографического алгоритма разбиение на подпоследовательности и их чередование осуществляется стохастически.

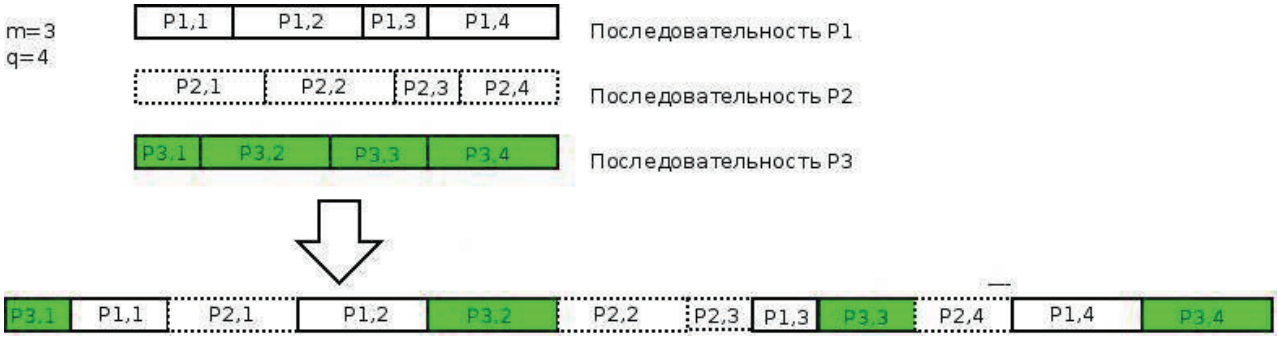



***Рис.1. Пример чередования при выполнении трех последовательностей***

Полиморфизм псевдопараллельной реализации обработки $m$ блоков данных обеспечивается на этапах:

- случайного выбора длин $q$ подпоследовательностей $P_{j,l}$ $l=1..q$ – при этом по случайному закону выбираются ($q$-1) длин подпоследовательностей, а последняя длина является дополнением суммы остальных длин до $K$;
- стохастического планирования выполнения подпоследовательностей $P_{j,l} j=1..m, l=1..q$ – во время работы алгоритма. Из $m$ готовых к выполнению подпоследовательностей случайно выбирается одна, по окончании её исполнения ситуация повторяется; в конце обработки число готовых $P_{j,l}$ сокращается за счет завершения обработки все большего числа блоков данных; признаком окончания обработки всех $m$ блоков данных является отсутствие готовых к выполнению подпоследовательностей.

Следует отметить, что оба этапа могут быть как статическими (выполняться перед обработкой данных) так и динамическими (во время обработки данных).

Для реализации псевдопараллельной обработки с разделением по времени необходимо обеспечить механизм сохранения/восстановления состояния ВУ при переключении между $P_j$. Реализация такого механизма требует дополнительных ресурсов, что снижает производительность алгоритма на величину пропорциональную числу переключений. Значительных временных затрат требует также программная генерация псевдослучайного кода. Исходя из этого, общее время $t^m_{par}$ псевдопараллельной обработки $m$ блоков данных определяется формулой :

$$t^m_{par} = m \cdot t^1 + (m \cdot q - 1) \cdot (t_{state} + t_g) \qquad (2)$$

где $t^1$ – время последовательной обработки одного блока данных, $t_{state}$ – время, затрачиваемое на сохранение/восстановление состояния при переключении, $t_g$ – время программной генерации псевдослучайного числа.

Таким образом, для уменьшения негативного влияния предложенной псевдопараллельной реализации на производительность необходимо использовать малые значения $q$ и минимизировать объем данных при сохранении состояния. Это достигается за счет рационального выбора размеров макроопераций и их атомарного выполнения – переключение возможно только между макрооперациями.

Вполне очевидно, что максимальное значение вариации $\Delta t_m$ времени выполнения команды определяется формулой:

$$\Delta t_m = (m-1) \cdot t^{m-1}_{par} + \frac{(q-k) \cdot t^1}{q \cdot k} \qquad (3)$$

Среднее значение времени $\Delta t_a$ вариации момента выполнения команды вычисляется следующим образом:

$$\Delta t_a = \frac{m \cdot t^1}{k} \cdot (\frac{q}{k} + 1) \qquad (4)$$

Анализ формулы (3) показывает, что среднее время вариации момента выполнения команды при использовании предлагаемого способа выполнения криптографических алгоритмов увеличивается в $m$ раз.

Важным фактором производительности полиморфной реализации алгоритмов криптографической защиты данных является способ получения случайных чисел. Учитывая технологические сложности измерения физически случайных величин, для большинства применений терминальных микроконтроллеров в качестве источника случайных чисел можно использовать встроенный таймер вместо случайных чисел применять превдослучайные числа.

### Полиморфная реализация алгоритма ГОСТ 28.147-89

Предложенный способ организации псевдопараллельной обработки конкретизирован для модификации реализации сертифицированного на Украине алгоритма ГОСТ 28.147-89. В качестве базовой использована реализация указанного алгоритма блочного шифрования для 32-х битных процессоров архитектуры Intel x86, разработанная Винокуровым А.Ю.[4].

Объектом модификации является макрооперация – цикл обработки данных алгоритма ГОСТ 28.147-89, реализованный в процедуре _gost32. В этой процедуре элементами наиболее уязвимыми для анализа динамики потребления мощности являются 4 операции табличного преобразования (xlat) после наложения ключа на данные. В этих операциях данные, содержащие часть ключевой информации выставляются на шину, для обращения к памяти. Остальные операции (кроме наложения ключа) регистровые и потребление мощности при их выполнении существенно меньше, чем при работе с памятью. Следовательно, основным элементом при получении множества эквивалентных реализаций процедуры _gost32 должны быть именно эти операции. Поскольку операции табличного преобразования независимы по данным, порядок их выполнения может быть произвольным, что позволило получить 24 варианта реализации процедуры _gost32 на каждом из 32-х циклов.

Для псевдопараллельной обработки в режиме разделения времени использовались следующие параметры $m$=2, $q$=4·32=128. При этом вместо случайного планирования выполнения подпоследовательностей реализовано циклическое переключение со случайным выбором стартовой подпоследовательности. В процедуре _gost32 единственный свободный регистр EDI использован для хранения длин подпоследовательностей $P_{j,l}$ $l=1..q=4, j=1..2$. Поскольку общее число подпоследовательностей равно 8, то на хранение длины отводится 4 бита, что позволяет хранить значение до 15 (из 32 возможных). Объем данных которые необходимо сохранить при переключении последовательностей достаточно мал – текущий блок данных (EDX:EAX) и указатель на текущее смещение в развернутом ключе (ESI). Ввиду отсутствия свободных регистров сохранение ведется в стек. Таким образом, для сохранения и восстановления состояния при переключении достаточно 4 и 6 команд соответственно. Значение $t_{state}$ для произведенной реализации равно:

$$t_{state} = 3 \cdot t_{PUSH} + 3 \cdot t_{POP} + t_{SUB} + t_{MOV} + \\ + t_{AND} + t_{JMP} = 10 \cdot t_C \qquad (5)$$

где $t_{PUSH}$ , $t_{POP}$ , $t_{SUB}$, $t_{MOV}$ , $t_{AND}$ , $t_{JMP}$ – время выполнения команд PUSH, POP, SUB, MOV, AND и JMP соответственно, а $t_C$ – время выполнения цикла процессора Intel Pentium.

Для реализации псевдослучайного генератора случайных чисел, используемого для выбора обрабатываемого цикла и подпоследовательности в нем, наиболее целесообразным с точки зрения времени реализации представляется использование сдвигового регистра с линейной функцией обратной связи (LFSR – Linear Feedback Shift Register), выбираемой случайно из некоторого наперед заданного набора. Для генерации псевдослучайного числа требуется: вычислить значение линейной функции обратной связи (команда AND – для наложения маски обратной связи, 3 команды MOV и команда сдвига для копирования флага четности в флаг переноса) и сдвинуть регистр (одна команда ROL) – всего 6 команд.

При этом полный цикл обработки блока данных в реализации процедуры _gost32 равен: $t^1 = 1408 t_C$. Численное значение времени $t_{par}^2$ выполнения 2-х блоков, вычисленное по формуле (2) составляет: $t_{par}^2 = 2 \cdot 1408 t_C + 127 \cdot (10+6) \cdot t_C = = 4848 \cdot t_C$.

Таким образом, при псевдопараллельной обработке блоков в алгоритме ГОСТ 28.147-89 соотношение времени выполнения пары

блоков – $t_{par}^2$ к времени $2 \cdot t^1$ их последовательного выполнения равно: $t_{par}^2/(2 \cdot t^1)$ =4848/2816=1.721.

Полученное значение подтверждается результатами экспериментальных исследований для $m$=2, $q$=128. При этом, определяющим фактором уменьшения производительности реализации криптографического алгоритма является величина $q$.

## Выводы

Методы реконструкции секретных ключей криптографических алгоритмов посредством анализа динамики потребляемой мощности (АДПМ) представляют на сегодняшний день одну из наиболее серьезных угроз для средств защиты информации с одним исполняющим блоком (смарт-карты, встраиваемые системы и др.). Аппаратные способы противодействия АДПМ-технологиям требуют дорогостоящей специализированных средств. Использование традиционных программных средств противодействия имеет следствием деградацию производительности алгоритмов и малоэффективно против некоторых видов атак (например Д-АДПМ высоких порядков).

В качестве эффективной контрмеры АДПМ-технологиям несанкционированного доступа к ключам криптографических алгоритмов предложен способ их полиморфной реализации. Основной эффект противодействия достигается за счет того, что каждый раз криптографическое преобразование производится с помощью различной последовательности команд. Это препятствует сбору корректных образцов-профилей для профильного АДПМ и использованию статистических методов усреднения в различных Д-АДПМ. Отличительной особенностью предложенного способа является организация пседопараллельной криптографической обработки $m$ блоков данных, что обеспечивает в $m$ раз увеличение среднего значения вариации момента времени выполнения каждой команды алгоритма. Это значительно снижает эффективность существующих технологий реконструкции ключа с помощью анализа динамики потребляемой мощности.

Способ может быть эффективно использован для защиты ключей алгоритмов как симметричного шифрования: DES, Rijndael, ГОСТ 28.147-89, так и несимметричного типа RSA.

Предложенный способ конкретизирован в виде программной реализации стандартизированного в Украине алгоритма шифрования ГОСТ 28.147-89. Экспериментально доказано увеличение роста вариации момента времени выполнения команд алгоритма при снижении производительности в 1.7 раз, что меньше деградации производительности при использовании таких альтернативных способов противодействия АДПМ-технологиям, как маскирование и чередование [6].

Предложенный способ и его реализация для алгоритма ГОСТ 21147-89 могут быть использованы для повышения эффективности защиты информации на смарт-картах и встроенных терминальных устройствах компьютерных сетей.

## Список литературы

1. Rivain M., Prouff E., Doget J. Higher-order masking and shuffling for software implementations of block ciphers // Cryptographic Hardware and Embedded Systems - CHES 2009. of Lecture Notes in Computer Science. - Springer, Heidelberg.- 2009.- Vol. 5747. - P. 171-188.
2. Kocher P., Jafft J., Jub B. Differential power analysis // Advances in Cryptology - CRYPTO '99 Lecture Notes in Computer Science. - Springer, 1999. - Vol. 1666 . - P. 388—397.
3. Chari S., Jutla C., Rao J., Rohatgi P. Towards Sound Approaches to Counteract Power-Analysis Attacks // Advances in Cryptology – CRYPTO ’99 Lecture Notes in Computer Science. - Springer, 1999. - Vol. 1666 . - P. 398–412.
4. Винокуров А. Алгоритм шифрования ГОСТ 28.147-89, его использование и реализация для компьютеров платформы Intel x86 [Электронный ресурс]. - Работа на правах рукописи, доступна на веб-сайте http://www.enlight.ru/crypto/articles/vinokurov/gost_i.htm
5. Марковский А.П., Абабне О.А., Ияд Мохд Маджид Ахмад Шахрури. Способ защиты ключей алгоритма ГОСТ 28.147-89 от реконструкции анализом динамики потребляемой мощности // Вісник національного технічного університету України ”КПІ”. Інформатика, управління та обчислювальна техніка. – 2007.- № 46.- С.128-138.

Поступила в редакцию 18.12.2009

УДК 004.942

*ЛАВРЕНЮК С.И.,*
*КОПЫЧКО С.Н.,*
*ГОРДИЕНКО Р.А.*

# ОЦЕНКА ПАРАМЕТРОВ ЗАГРУЗКИ УЗЛОВ GRID-СИСТЕМЫ ДЛЯ ОПТИМИЗАЦИИ ЕЕ ПРОИЗВОДИТЕЛЬНОСТИ

Статья обосновывает необходимость оценки параметров, которые определяют узлы в планировщике программного обеспечения узла GRID-системы для оптимального распределения нагрузки в системе. Был сделан обзор и выбор необходимых параметров для оценки. Предложены методы оценки надежности узлов на основе вероятности. Опубликованы результаты прогнозирования выбранных параметров грид-узлов в УАГ и EGEE с помощью временных рядов. Предложены перспективы дальнейшей работы.

The paper describes needs of estimation of parameters which identify nodes in Grid middleware schedulers' for right distribution of load in system. Sampling of main parameters are made. Proposed method for estimate a reliability of node on the base of probability. Described result of forecasting sampled parameters of nodes in UAG and EGEE with the time series analyses. The prospects for further research and modification are proposed.

## Актуальность проблемы

Необходимость использования распределенных вычислений связана с «очень быстрым» ростом объемов задач в научной сфере деятельности. Грид-системы являются новым поколением распределенных систем, популярность которых растет с каждым днем. На сегодняшний день Грид-системы в основном используются в научных вычислениях. По мере развития самих систем возникают проблемы правильного распределения задач в таких системах, что являются основным фактором, влияющим на производительность. Основной проблемой большой распределенной системы является вовлечение всех ее ресурсов в процесс вычислений. Обычно задача посылается на случайный узел, не зная, таким образом, о сроках простоя в очереди и не имея уверенности в надежности и стабильности работы данного узла. Для правильного распределения нагрузки необходимо оценивать каждый из узлов по его параметрам, и выбирать наиболее оптимальный для каждой конкретной задачи.

С развитием Грид-технологий в Украине связано создание Украинского Академического Грида (УАГ). На сегодняшний день он не насчитывает и 30-ти узлов, но в будущем планируется активное его развитие и внедрение в украинскую науку, как мощный аппарат для вычислений. Для подготовки этого этапа необходимо внедрение технологии доверия в УАГ для увеличения производительности без дополнительных материальных затрат на аппаратное обеспечение.

## Обзор существующих исследований и публикаций

Впервые описанную выше технологию в Грид-систему внедрили в работе [1], где кратко изложили необходимость введения и структуру системы с внедренными оценочными коэффициентами. Анализ результатов их работы говорит о незначительном (0.3-1%) повышении использования машинных ресурсов и хорошем (на 30%) уменьшением времени выполнения задач. В том же году была опубликована структура аналитической системы оценивания [2], основанная на использовании динамически поставляемых данных о работе узлов. Успешно примененный метод прогнозирования для Грид-систем, был опубликован в статье [3] и основывался на применении теории нечетких множеств для оценки параметров. Наиболее удачный подбор параметров, по которым можно оценить узел системы, был подобран в [7].

Было опубликовано множество других статей по технологии доверия и применении ее к Грид-системам, но в любом из разработанных методов имеются существенные недостатки, потому проблемы разработки кон-

цепции оценки узлов для Грид-систем остаются актуальными на сегодняшний день. В основном проблемой большинства разработанных подходов является стремление выбрать наиболее лучший узел во всей системе, что приводит к не использованию ненадежных узлов, даже для небольших задач. Оценивание узла проводится комплексно по сумме всех критериев, в то время когда нам может понадобиться не мощный, но надежный узел, который получит небольшую оценку в системе по причине невысоких показателей производительности. Основной проблемой всех ранее разрабатываемых концепций является существенное различие критериев, по которым определяется коэффициент готовности в каждой системе. Разработки по внедрению такой концепции будут проводиться для каждой Грид-системы отдельно до тех пор, пока каждая конкретная распределенная система не будет универсальной, а будет существовать для выполнения определенного вида вычислений, как, например, EGEE(Enabling Grid for Electronic science) большинство ресурсов выделяет для обработки данных и проведения расчетов для Большого Андронного Коллайдера.

## Постановка задачи

Данная работа имеет целью выделить параметры, необходимые для целесообразной оценки узлов в УАГ и спрогнозировать оптимальным, с точки зрения погрешности прогноза, методом по этим параметрам их работу в системе для последующего создания модели функционирования Грид-системы. По причине небольшого размера УАГ, также для сравнения также был проведен анализ эмпирических данных по Европейской Грид-системе EGEE. Она насчитывает около 500 узлов и десятки сервисов, один из который (Grid Observatory) предоставляет различные данные о работе системы, предшествующей настоящему времени.

Нашей первостепенной задачей являлся выбор общих параметров узлов, которые мы могли отслеживать в следах работы EGEE и собирать с помощью активных экспериментов в УАГ. Главным параметром для любого узла в любой системе является доступность, в соответствие которой мы поставили параметр Status. Также были выбраны такие параметры, как количество доступных логических процессоров (FreeJobSlots или FreeCPU's для УАГ), среднее время простоя задачи в очереди (EstimatedResponseTime), приоритет узла (Priority). Прогнозирование проводилось и по другим параметрам, но перечисленные выше являются ключевыми для принятия решения о направлении задачи на узел.

На основе разработанной в работе [4] концепции сбора данных в УАГ было проведено активное тестирование узлов, результаты которого были записаны в таблицы данных. Далее мы остановимся только на нескольких узлах.

Также отдельно была разработана и создана реляционная база данных, включающая в себя информацию о значении вышеописанных и других параметров с узлов EGEE на основе работ [5] и [6], а также разработано программное обеспечение, позволяющее проводить выборку из базы и использовать полученные данные в прогнозировании.

## Определение степени доступности узла

Перед тем как прогнозировать какие-либо параметры узла, необходимо узнать будет ли он работать в то время, когда мы захотим направить на него задачу. Узел может работать в режиме реального времени, т.е. принимать задачи и одновременно выполнять их, а может сначала собирать задачи в очередь, а затем выполнять их, в нужном порядке, используя наиболее оптимальный с точки зрения времени выполнения и важности конкретного задания алгоритм планирования. Остальные состояния, в которых узел не доступен или не принимает, а выполняет задачи, нас не интересуют. Назовем состояние, в котором узел принимает задания, но не начинает их выполнение – Queueing (составление очереди), а состояние в котором узел принимает и выполняет задания – Production(выполнение),. В УАГ редко используется такой прием, как установление для узла статуса Queueing с целью накопления задач, но он повсеместно применяется в европейской Грид-системе. Для определения наилучшего узла вычислялся $К_{готовности}=2*P_{production}+P_{queueing}$, где $К_{готовности}$ – коэффициент готовности, $P_{production}$ – вероятность того, что узел будет пребывать в состоянии Production, а $P_{queueing}$ – вероятность того, что узел будет пребывать в состоянии Queueing. Вероятность вычислялась отношением временных отрезков, на которых узел пребывал в заданном состоянии к общему количеству временных отрезков, на которых исследовалась работа узла. Таким образом, чем выше у узла коэффициент $К_{готовности}$, тем

стабильнее будет его работа в последующие интервалы времени. Проведенные исследования показывают, что лишь в 1 из 10-ти случаев коэффициент неоправданно высок у узла, который почти всегда находится в недоступном состоянии.

### Обзор методов прогнозирования

Был сделан обзор методов прогнозирования численных характеристик узла для выбора оптимального метода, с погрешностью не более 20%. Эксперименты для сравнения методов проводились, используя данные о значениях параметра FreeJobSlots (или FreeCPU's для УАГ). Обычная экстраполяция дала очень неточные результаты. Попытка оценить распределение простой аналитической функцией, как то: полиномиальная, экспоненциальная, логарифмическая, в среднем давали погрешности от 30 до 70%. Это обусловлено наличием большого количества скачков для всех параметров Грид-системы. Затем была рассмотрена оценка, основанная на концепции временных рядов. Единицей времени был выбран интервал в 15 минут. Таким образом, час составлял 4 единицы, а сутки 96. Был проведен прогноз для трех узлов по параметру FreeJobSlots для узла EGEE и по FreeCPU's для УАГ по формуле $Q_t=T_t+S_t+I_t$. В процессе оценки не был учтен фактор сезонности ($S_t=0$), т.к. в распределенной системе он отсутствует. Был выбран упрощенный способ, в котором учитывался только тренд (направление распределения), который строился на основе аппроксимации данных, собранных на последних 10 отрезках времени, аналитической функцией, и отклонение от направления в каждый момент времени, как случайная величина, которая строится на основе анализа погрешности прогноза предыдущих 30 значений.

### Описание экспериментов и их результатов

Эксперименты проводились с использованием данных о четырех узлах. Три из них принадлежат УАГ и один EGEE. Результаты приведены в виде графиков на Рис. 1-4. Узлы были выбраны исходя из данных по их загруженности. Первые два узла (грид-узел НТУУ «КПИ» и грид-узел Института Теоретической физики им. М.М. Боголюбова) активно используются в системе, постоянно имея задачи для выполнения. Процессоры на этих узлах почти никогда не простаивают. Узел energrid.ipme.kiev.ua наоборот редко используется в системе, его загрузка очень невелика и не постоянна. Промежуток времени, на котором проводился прогноз, составляет 3 дня. Узел atlas-ce-01.roma1.infn.it европейской Грид-системы имеет редкие скачки в своей производительности, но в целом работает стабильно и быстро восстанавливается после сбоев. Он рассматривался в течении суток.


***Рис.1 Эмпирическое распределение и оценка временным рядом параметра FreeCPU's (узел УАГ: nordu.hpcc.ntu-kpi.kiev.ua)***

Погрешность оценки первого узла видна на Рис.1 и достигает 13%, но в среднем равна 4%, что является хорошим результатом.

Загрузка второго узла была спрогнозирована с средней относительной погрешностью в 8% (Рис.2).

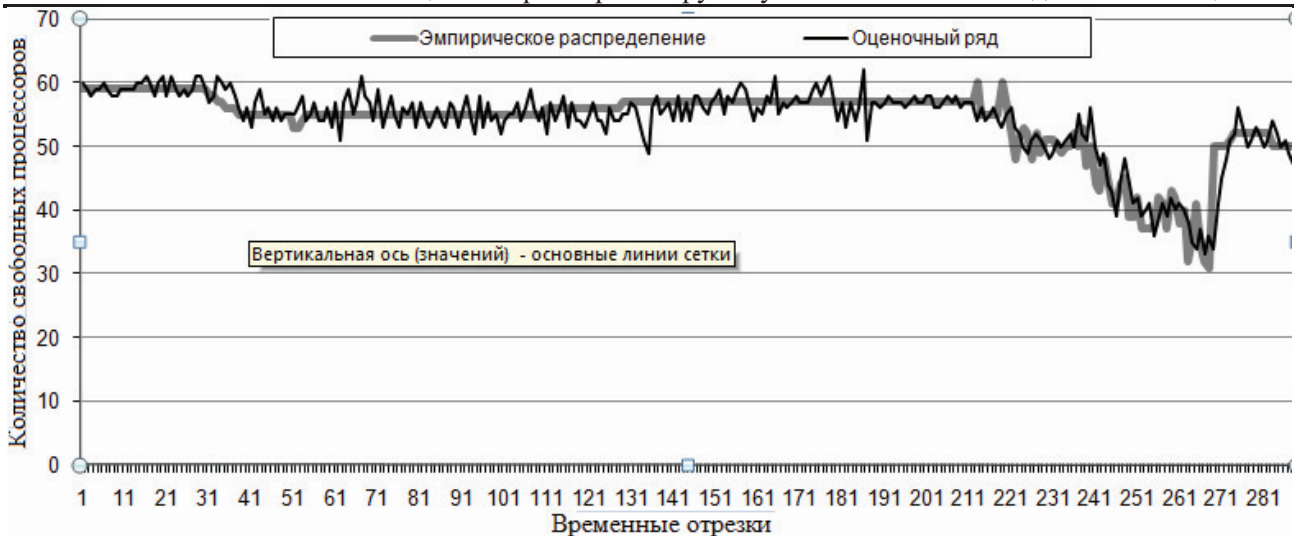



***Рис.2 Эмпирическое распределение и оценка временным рядом параметра FreeCPU's (узел УАГ: nordug.bitp.kiev.ua)***


***Рис.3 Эмпирическое распределение и оценка временным рядом параметра FreeCPU's (узел УАГ: energrid.ipme.kiev.ua)***

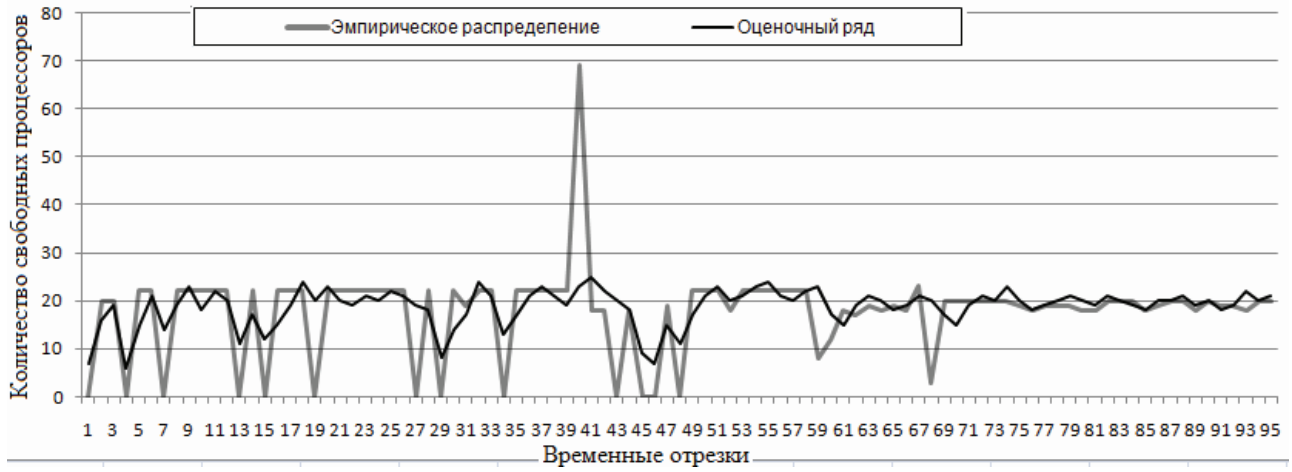



***Рис 4. Эмпирическое распределение и оценка временным рядом параметра FreeJobSlots (узел EGEE: atlas-ce-01.roma1.infn.it)***

Для третьего узла погрешность достигала 80% и в среднем была равна 27%. Четвертый узел имел погрешность в спрогнозированной загрузке в 18%.

## Анализ результатов и перспективы дальнейшей работы

По результатам проделанной работы можно сделать следующие выводы. Оценка

описанными выше рядами хорошо работает в случае стабильной работы узла и стабильной его загруженности. При редких сбоях в работе (несколько в день) и быстром восстановлении узла, как на atlas-ce-01.roma1.infn.it, оценка дает вполне допустимую погрешность, но даже на примере этого узла из графика на Рис.1 видна невозможность выбранным методом оценить резкие скачки в значениях.

Те же результаты дает нам прогнозирование параметров украинских узлов. В случае бесперебойной работы узла и постоянной загрузки его процессоров можно очень хорошо (с погрешностью не более 13%) оценить параметр загруженности, на основе данных всего одного ряда. Если на узле бывают частые сбои, и по этой причине его работа прерывается на длительный срок, а при восстановлении его загрузка изменяется скачкообразно, то оценка временным рядом не даст нам возможности спрогнозировать работу такого узла с заданной точностью. Решением в данной ситуации может служить применение более сложных моделей для имитации работы Грид-системы, приведенных в [7], а также применение для оценки большего количества рядов и нахождения оптимального прогноза на основе нескольких построенных рядов.

Дальнейшая работа будет направлена на подбор рядов и их зависимость между собой, которые дадут возможность создать более качественный прогноз для нестабильных узлов и в своей оценке охватить и учесть скачкообразное поведение значений параметров, а так же определение оптимального количества таких рядов для модели, с точки зрения скорости построения модели и точности создаваемого прогноза.

## Список литературы


1. Farag Azzedin and Muthucumaru Maheswaran, Integrating Trust into Grid Resource Management Systems, 2007
2. Zheng Yan and Silke Holtmanns, Trust Modeling and Management: from Social Trust to Digital Trust, 2007
3. Куссуль Н. Н., Шелестов А.Ю., Робастное оценивание состояния узла GRID-системы методом нечетких эллипсоидов, 2008
4. Лавренюк С.И., Лавренюк А.Н., Грипич Ю.А, Построение базы данных мониторинга состояния узлов Grid-инфраструктуры // Материалы трудов шестой международной конференции «Теоретические и прикладные аспекты построения программных систем» – TAAPSD' 2009 Киев
5. Sergio Andreozzi, Stephen Burke and others, GLUE Schema Specification version 1.3 Final, 16.01.2007
6. Cecile Germain-Renaud, Alain Cady, Grid observatory technical documentation, 11.05.2008
7. Лавренюк С.И., Шелестов А.Ю., Идентификация моделей и оценка состояния Grid-систем // Кибернетика и системный анализ. — 2009. — № 6. — С. 42-50.


Поступила в редакцию 14.12.2009

УДК 004.75

*ЗАЙЧЕНКО Ю.П.,*
*МАЛИХЕХ ЕСФАНДИЯРФАРД,*
*ОВИ НАФА АГАИ АГ ГАМИШ*

# АНАЛИЗ МОДЕЛИ ОПТИМИЗАЦИИ НЕЧЕТКОГО ПОРТФЕЛЯ

В статье рассмотрены задачи портфельной оптимизации в нечетких условиях. Построена математическая модель данной задачи. Исследована зависимость «оптимальная доходность – риск» для нечеткого портфеля. Определены достаточные условия при которых эта зависимость будет монотонно-убывающей. Приводятся результаты экспериментальных исследований, подтверждающие теоретические результаты.

The fuzzy portfolio optimization problem is considered. The mathematical model is constructed and the dependence “optimal benefits – risk” is investigated. The sufficient conditions for this dependence to be monotonously decreasing are obtained. The results of experimental investigations are presented.

## Введение

Задача оптимизации инвестиционного портфеля в условиях неопределенности в последние годы вызывает значительный интерес. Для решения этой проблемы был предложен аппарат нечетких множеств в работах [1,2], согласно которому доходности акций и в целом доходность инвестиционного портфеля рассматриваются как нечеткие числа с заданной функцией принадлежности, а риск трактуется как возможность (субъективная вероятность) ситуации, когда реальная доходность портфеля оказывается ниже ожидаемой доходности нечеткого портфеля. С целью более обоснованной оценки доходности акций на фондовом рынке по предыстории в задаче портфельной оптимизации был предложен метод прогнозирования, применение которого позволило улучшить результаты [3]. В ходе экспериментальных исследований разработанного метода были построены зависимости «оптимальная доходность-риск» для нечеткого портфеля, которые во многих случаях имели прямо противоположный вид по сравнении с классической моделью Марковица-Тоббина. Целью настоящей работы является получение аналитических условий, при которых зависимость «оптимальная доходность – риск» для нечеткого портфеля будет монотонно убывающей и их экспериментальная проверка.

Задача нечеткой портфельной оптимизации базируется на допущении, что доходность портфеля является нечетким числом (НЧ), описываемым тройкой параметров:

$$r = (r_{\min} = \sum_{i=1}^{N} x_i r_i ; \tilde{r} = \sum_{i=1}^{N} x_i \tilde{r}_i ; r_{\max} = \sum_{i=1}^{N} x_i r_{i2}),$$

где $(r_{1i}, \tilde{r}_i, r_{2i})$ – доходность i-*той* ценной бумаги. $x_i$-доля i-*той бумаги в портфеле.*

Для того, чтобы определить структуру портфеля, который обеспечит максимальную доходность при заданном уровне риска, требуется решить следующую задачу оптимизации:

$$\tilde{r} = \sum_{i=1}^{N} x_i \tilde{r}_i \to \max \qquad (1)$$

$$\beta = const \qquad (2)$$

$$\sum_{i=1}^{N} x_i = 1 \quad x_i \geq 0 \quad i = \overline{1, N} \qquad (3)$$

$$0 < \beta < 1$$

где β- уровень риска.

В работе [1] было показано, что этот случай возможен когда

$$r_{\min} = \sum_{i=1}^{N} x_i r_{i1} \leq r^* < \sum_{i=1}^{N} x_i \tilde{r}_i$$

либо когда

$$\sum_{i=1}^{N} x_i \tilde{r}_i \leq r^* < \sum_{i=1}^{N} x_i r_{i2} = r_{\max}$$

Пусть $\sum_{i=1}^{N} x_i r_{i1} \leq r^* < \sum_{i=1}^{N} x_i \tilde{r}_i$.

Тогда используя результаты работы [3], задача (1)-(3) сводится к следующей задаче нелинейного программирования:

$$\tilde{r} = \sum_{i=1}^{N} x_i \tilde{r}_i \to \max \qquad (4)$$

$$\frac{1}{\sum_{i=1}^{N} x_i r_{i2} - \sum_{i=1}^{N} x_i r_{i1}} * \tag{5}$$

$$* \left( \left( r^* - \sum_{i=1}^{N} x_i r_{i1} \right) + \ln \left( \frac{\sum_{i=1}^{N} x_i \tilde{r}_i - r^*}{\sum_{i=1}^{N} x_i \tilde{r}_i - \sum_{i=1}^{N} x_i r_{i1}} \right) \right) = \beta$$

$$\sum_{i=1}^{N} x_i r_{i1} \le r^* \tag{6}$$

$$\sum_{i=1}^{N} x_i \tilde{r}_i > r^* \tag{7}$$

$$\sum_{i=1}^{N} x_i = 1 \quad x_i \ge 0 \quad i = \overline{1, N} \tag{8}$$

Учитывая вид для $r_{\min}$, $\tilde{r}$, и $r_{max}$, а также формулу (1), данную задачу нахождения интервала доходности для нечеткого портфеля с n ЦБ можно свести к задаче с двумя ЦБ ($x_1$,$x_2$) с минимальной и максимальной доходностями.

Пусть первая ЦБ имеет минимальную доходность в портфеле, описываемую НЧ $r_1 = [r_{11}, \tilde{r}_1, r_{12}]$, а вторая ЦБ – максимальную доходность в портфеле $r_2 = [r_{21}, \tilde{r}_2, r_{22}]$. Пусть также критериальное значение описывается НЧ $r^* = [r^*_1, r^*, r^*_2]$.

При этом допустим $r_{11}<r_{21}$; $\tilde{r}_1 < \tilde{r}_2$.

Тогда $\tilde{r} = \tilde{r}_1 x_1 + \tilde{r}_2 x_2; x_1 + x_2 = 1$. Причем $x=x_2$, тогда $x_1=1-x$.

Доходность портфеля описывается в этом случае НЧ вида $r= [r_{min},\tilde{r},r_{max}]$, где:

$$r_{\min} = r_{11}(1-x) + r_{12}x;$$
$$\tilde{r}_1 = \tilde{r}(1-x) + \tilde{r}_2 x;$$
$$r_{\max} = r_{21}(1-x) + r_{22}x;$$

Требуется найти такое $x$, что

$$\tilde{r} = \tilde{r}_1(1-x) + \tilde{r}_2 x \to \max \tag{9}$$

при условиях:

$$\beta(x) \le \beta_{зад} \tag{10}$$

где выражение β(x) для случая 0<β<1 и $\tilde{r}_1 < \tilde{r}_2$ при подстановке $x_1=1-x$; $x_2=x$; $n=2$ в выражение (5) дает:

$$\beta(x) = \frac{1}{(1-x)(r_{12} - r_{11} + x(r_{22} - r_{21}))} *$$
$$* \{ r^* - ((1-x)r_{11} + x r_{21}) + ((1-x)\tilde{r}_1 + x\tilde{r}_2 - r^*) *$$
$$* \ln \frac{\tilde{r}_1(1-x) + \tilde{r}_2 x - r^*}{x(\tilde{r}_1 - r_{11}) + (1-x)(\tilde{r}_2 - r_{21})} \le \beta$$

Достаточные условия монотонно убывающего характера зависимости $(\tilde{r}_{опт}\beta)$ являются следующие

$$\frac{\partial \tilde{r}_{опт}}{\partial x} \ge 0; \frac{\partial \beta}{\partial x} \ge 0 \tag{11}$$

Рассмотрим сначала $\frac{\partial \tilde{r}}{\partial x} > 0$.

Очевидно $\frac{\partial \tilde{r}}{\partial x} = \tilde{r}_2 - \tilde{r}_1$ (12) и т.к., по предположению, $\tilde{r}_1 < \tilde{r}_2$, то $\frac{\partial \tilde{r}}{\partial x} > 0$.

Рассмотрим теперь зависимость $\frac{\partial \beta}{\partial x}$.

Для нахождения сложной производной $\frac{\partial \beta}{\partial x}$, введем следующие обозначения:

$$A(x) = r^* - ((1-x)r_{11} + x r_{12});$$
$$B(x) = (1-x)\tilde{r}_1 + x\tilde{r}_2 - r^* \tag{13}$$
$$C(x) = (1-x)(\tilde{r}_1 - r_{11}) + x(\tilde{r}_2 - r_{21}) \tag{14}$$
$$D(x) = (1-x)(r_{12} - r_{11}) + x(r_{22} - r_{21}) \tag{15}$$

Тогда

$$\beta(x) = \left[ A(x) + B(x) \ln \frac{B(x)}{C(x)} \right] \frac{1}{D(x)} \tag{16}$$

Найдем производную $\frac{\partial \beta}{\partial x}$

$$\frac{\partial \beta}{\partial x} = \{ [A'(x) + B'(x) \ln \frac{B(x)}{C(x)} +$$
$$+ \frac{B'(x)C(x) - B(x)C'(x)}{C(x)} ] D(x) - \tag{17}$$
$$D'(x)[A(x) + (x \ln \frac{B(x)}{C(x)})] \} \frac{1}{D^2(x)}$$

Выясним условия, при которых $\frac{\partial \beta}{\partial x} < 0$.

Для этого необходимо, чтобы выражения в фигурных скобках {} в (17) было меньше 0.

Очевидно

$$A'(x) = r_{11} - r_{21}; \; B'(x) = -\tilde{r}_1 + \tilde{r}_2$$
$$C'(x) = -(\tilde{r}_1 - r_{11}) + (\tilde{r}_2 - r_{21})$$
$$= (\tilde{r}_2 - \tilde{r}_1) - (r_{21} - r_{11}) \tag{18}$$
$$D'(x) = (r_{22} - r_{21}) - (r_{12} - r_{11}).$$

Подставим (18) в (17) и после несложных преобразований в (17) получим следующее условие

$$C(x)D(x)\left[A'(x)+B'(x)+B'(x)\ln\frac{B(x)}{C(x)}\right]-$$
$$DC'(x)B(x)-C(x)D'(x) \qquad (19)$$
$$(A(x)+B(x)\ln\frac{B(x)}{C(x)})<0$$

Подставляя выражение для $A(x); A'(x); B(x); B'(x); C(x); C'(x); D(x); D'(x)$ в (19), получим

$$\begin{aligned}
&[(1-x)(r_{12}-r_{11})+x(r_{22}-r_{11})]*\\
&*[(1-x)(\tilde{r}_1-r_{11})+x(\tilde{r}_2-r_{21})]*\\
&[(r_{11}-r_{21})+(\tilde{r}_2-\tilde{r}_1)+(\tilde{r}_2-\tilde{r}_1)*\\
&*\ln\frac{(1-x)\tilde{r}_1+x\tilde{r}_2-r^*}{(1-x)(\tilde{r}_1-r_{11})+x(\tilde{r}_2-r_{21})}]-\\
&-[(1-x)\tilde{r}+x\tilde{r}_2-r^*][(\tilde{r}_2-r_{21})-\\
&-(\tilde{r}_1-r_{11})]*\\
&*[(1-x)(r_{12}-r_{11})+x(r_{22}-r_{21})]-\\
&-[(1-x)(\tilde{r}_1-r_{11})+x(\tilde{r}_2-r_{21})]*\\
&*[(r_{22}-r_{21})-(r_{12}-r_{11})]*\\
&(r^*-(1-x)r_{11}-xr_{21}+\\
&+[(1-x)\tilde{r}+x\tilde{r}_2-r^*]*\\
&\ln\frac{(1-x)\tilde{r}_1+x\tilde{r}_2-r^*}{(1-x)(\tilde{r}_1-r_{11})+x(\tilde{r}_2-r_{21})})
\end{aligned} \qquad (20)$$

Пусть функции принадлежности являются треугольными симметричными. Тогда

$$\tilde{r}_1-r_{11}=r_{12}-\tilde{r}_1=\Delta_1;\ \tilde{r}_2-r_{21}=r_{22}-\tilde{r}_2=\Delta_2$$

и, очевидно, $r_{12}-r_{11}=2\Delta_1;\ r_{22}-r_{21}=2\Delta_2$.

Подставляя эти выражения в (20) получим

$$\begin{aligned}
&[(1-x)2\Delta_1+x2\Delta_2][(1\Delta_1x)2\Delta_2+x\Delta_2*\\
&[(\Delta_2-\Delta_1)+(\tilde{r}_2-\tilde{r}_1)*\\
&*\ln\frac{(1-x)\tilde{r}_1+x\tilde{r}_2-r^*}{(1-x)(\tilde{r}_1-r_{11})+x(\tilde{r}_2-r_{21})}]-\\
&-[(1-x)\tilde{r}_1+x\tilde{r}_2-r^*]*[\Delta_2-\Delta_1]*\\
&[(1-x)2\Delta_1+x2\Delta_2]*[(1\Delta_1x)2\Delta_2+x\Delta_2]*\\
&*[\Delta_2-\Delta_1]*(r^*-(1-x)r_{11}-xr_{21})+\\
&+[(1-x)\tilde{r}_1+x\tilde{r}_2-r^*]*\\
&*\ln\frac{(1-x)\tilde{r}_1+x\tilde{r}_2-r^*}{(1-x)(\tilde{r}_1-r_{11})+x(\tilde{r}_2-r_{21})}
\end{aligned} \qquad (21)$$

Преобразуем правую часть выражения (21)

$$\begin{aligned}
&-[(1-x)\tilde{r}_1+x\tilde{r}_2-r^*](\Delta_2-\Delta_1)*\\
&*[(1-x)2\Delta_1+x2\Delta_2]-[(1-x)\Delta_1+x\Delta_2]*\\
&*(\Delta_2-\Delta_1)\{r^*-(1-x)r_{11}-xr_{21}\}=\\
&=-2[(1-x)\Delta_1+\Delta_2](\Delta_2-\Delta_1)*\\
&*[(1-x)\tilde{r}_1+x\tilde{r}_2-r^*+r^*-(1-x)r_{11}-xr_{21}]=\\
&=-2[(1-x)\Delta_1+\Delta_2](\Delta_2-\Delta_1)[(1-x)\Delta_1+x\Delta_2]=\\
&=2[((1-x)\Delta_1+\Delta_2)^2(\Delta_2-\Delta_1)]
\end{aligned} \qquad (22)$$

Анализируя выражение (22) нетрудно проверить, что

$$C(x)D(x)[A'(x)+B'(x)]=$$
$$D(x)C'(x)B(x)+C(x)D'(x)A(x),$$

поэтому после сокращения подобных членов в (22) останутся члены

$$C(x)D(x)B'(x)\ln\frac{B(x)}{C(x)}-$$
$$C(x)D'(x)B(x)\ln\frac{B(x)}{C(x)} \qquad (23)$$

И подставляя выражение для $C(x); D(x); B(x); B'(x)$ в (23) получим

$$\begin{aligned}
&[(1-x)(r_{12}-r_{11})+x(r_{22}-r_{21})]*\\
&*[(1-x)(r_{12}-r_{11})+x(r_{22}-r_{21})*\\
&*(\tilde{r}_2-\tilde{r}_1)-(1-x)\tilde{r}+x\tilde{r}_2-r^**\\
&(\tilde{r}_2-\tilde{r}_1)(r_{22}-r_{21})*\\
&*\ln\frac{(1-x)(\tilde{r}_1+x\tilde{r}_2-r^*)}{(1-x)(\tilde{r}_1-r_{11})+x(\tilde{r}_2-r_{21})}]
\end{aligned} \qquad (24)$$

Откуда

$$\begin{aligned}
&2[(1-x)\Delta_1+x\Delta_2]*\\
&*\ln\frac{(1-x)\tilde{r}_1+x\tilde{r}_2-r^*}{(1-x)(\tilde{r}_1-r_{11})+x(\tilde{r}_2-r_{21})}]*\\
&*[(1-x)2\Delta_1+x2\Delta_2(\tilde{r}_2-\tilde{r}_1)-\\
&(\Delta_2-\Delta_1)][(1-x)\tilde{r}_1+x\tilde{r}_2-r^*]
\end{aligned} \qquad (25)$$

Заметим, что

$$\begin{aligned}
&(1-x)(\tilde{r}_1-r_{11})+x(\tilde{r}_2-r_{21})=\\
&=(1-x)\tilde{r}_1+x\tilde{r}_2-(1-x)r_{11}-\\
&xr_{21}=(1-x)\tilde{r}_1+x\tilde{r}_2-r_{\min}
\end{aligned}$$

и так как $r^*>(1-x)r_{11}+xr_{21}=r_{\min}$, то

$$\frac{B(x)}{C(x)}=\frac{(1-x)\tilde{r}_1+x\tilde{r}_2-r^*}{(1-x)(\tilde{r}_1-r_{11})+x(\tilde{r}_2-r_{21})}<1,$$

$$\ln\frac{B(x)}{C(x)}<0 \qquad (26)$$

Следовательно, для того, чтобы выражение (25) было меньше нуля необходимо и достаточно, чтобы

$$(1-x)2\Delta_1 + x2\Delta_2(\tilde{r}_2 - \tilde{r}_1) - (\Delta_2 - \Delta_1)][(1-x)\tilde{r}_1 + x\tilde{r}_2 - r^*]$$

было больше 0.

Исследуем условия, при которых

$$\{2[(1-x)\Delta_1 + x\Delta_2](\tilde{r}_2 - \tilde{r}_1) - - (\Delta_2 - \Delta_1)][(1-x)\tilde{r}_1 + x\tilde{r}_2 - r^*\} > 0 \quad (27)$$

а) неравенства

$\tilde{r}_2 - \tilde{r}_1 > \Delta_2 - \Delta_1 = (\tilde{r}_2 - \tilde{r}_1) - (r_{21} - r_{11})$ выполняются, если $r_{21} > r_{11}$.

б) неравенства

$2[(1-x)\Delta_1 + x\Delta_2] > (1-x)\tilde{r}_1 + x\tilde{r}_2 - r^*$ выполняется, если

$$\begin{aligned} &2(1-x)(\tilde{r}_1 - r_{11}) + 2x(\tilde{r}_2 - r_{21}) - \\ &- (1-x)\tilde{r}_1 - \tilde{r}_2 + r^* = (1-x) * \\ &* (\tilde{r}_1 - r_{11}) + x(\tilde{r}_2 - r_{21}) + r^* = \\ &= (1-x)(\tilde{r}_1 - x_{11}) + x(\tilde{r}_2 - r_{21}) + \\ &+ r^* - [(1-x)x_{11}] + xr_{21} = \\ &= (1-x)\Delta_1 + x\Delta_2 + r^* - r_{\min} > 0 \end{aligned} \quad (28)$$

и (28) выполняется, если $r^* - r_{\min} > 0$.

Таким образом, мы получили следующие достаточные условия для (см. рис 1)

а) $\tilde{r}_1 < r^* < \tilde{r}_2$;

б) $r_{11} < r_{21}$;

в) $r^* > r_{\min} = (1-x)r_{11} + x_1 r_{21}$;

г) $r^* < \tilde{r} = (1-x)\tilde{r}_1 + x_1\tilde{r}_2$;

При выполнении этих условий, зависимость «оптимальная доходность-риск» будет иметь монотонно-убывающий характер.

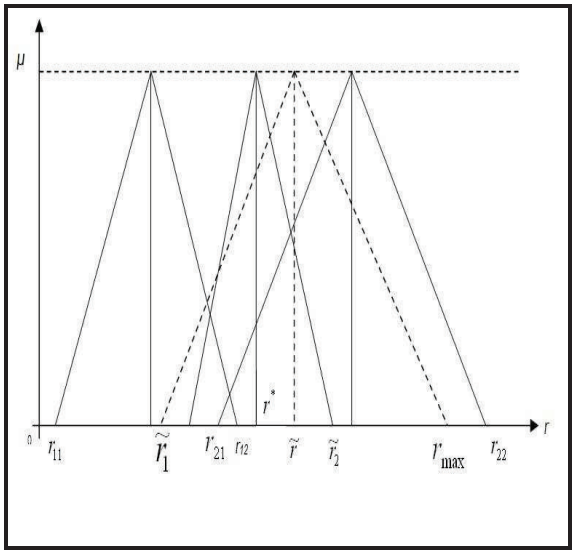



***Рис .1. Графическая интерпретация достаточных условий***

## Экспериментальные исследования зависимости «оптимальная доходность - риск»

Проведем экспериментальные исследования зависимости «оптимальная доходность-риск» для нечеткого инвестиционного портфеля. При этом рассмотрим случаи двух ЦБ с малой доходностью и высокой доходностью с различными волатильностями.

При этом будем варьировать величину критериального значения.

Исследуем случаи, когда эта зависимость будет иметь монотонно- убывающий характер.

Случай 1. Когда первой ЦБ была задана высокая волатильность и низкая доходность, а другой ЦБ средняя волатильность и высокая доходность, то зависимость доходность-риск носила монотонно – убывающий характер.Это иллюстрируется графиками на рис.2.

Ниже приводятся различные примеры взаимного расположения ФП доходностей акций и критериального значения и соответствующие построенные зависи -мости « оптимальная доходность-риск»
(рис. 2-10).

Пример 1

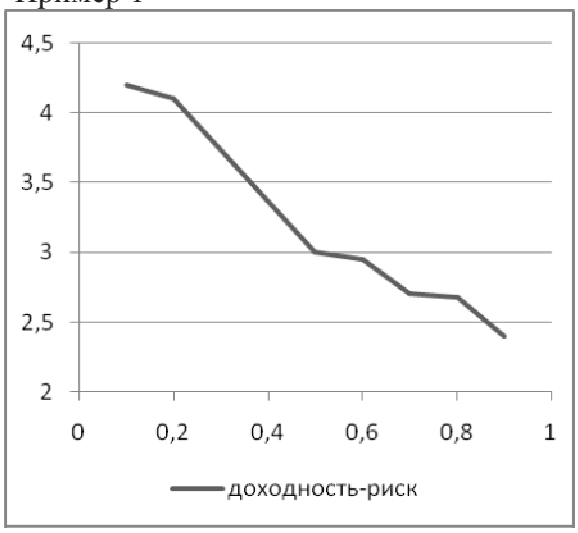



***Рис .2. График зависимости доходность- риск нечеткого портфеля***

Пример 2.

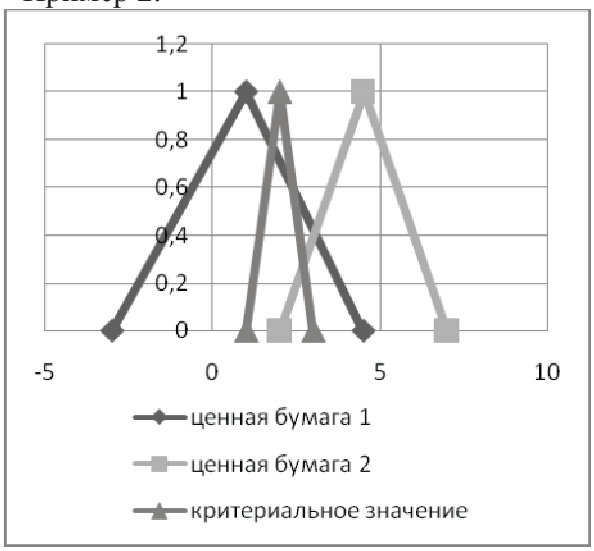



***Рис. 3. Взаимное расположение нечетких доходностей ЦБ и критериального значения для треугольных ФП***

Пример 3.

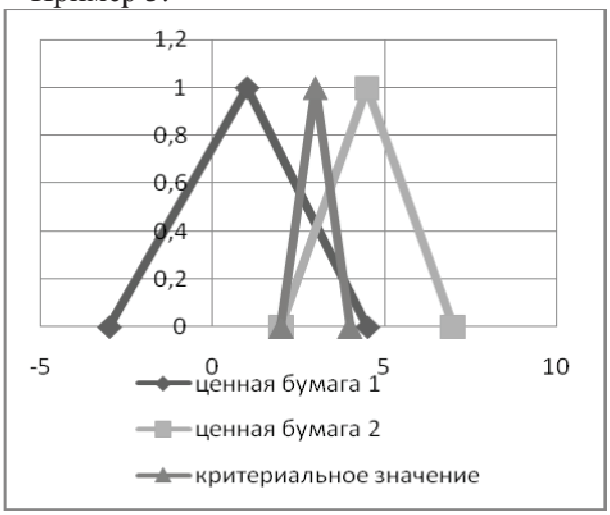



***Рис. 4. Взаимное расположение нечетких доходностей ЦБ и критериального значения.***

Пример 4.

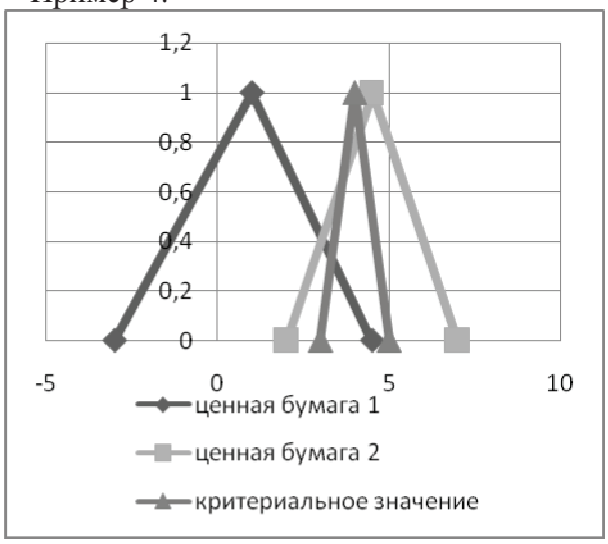



***Рис. 5. Взаимное расположение нечетких доходностей ЦБ и критериального значения.***

Пример 5.

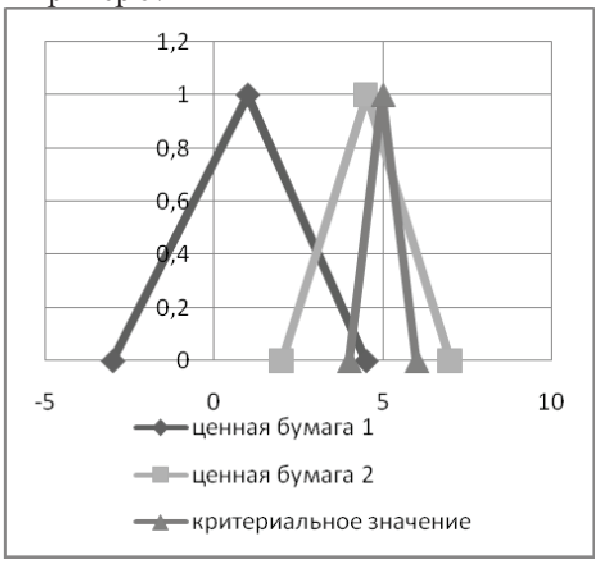



***Рис. 6. Взаимное расположение нечетких доходностей ЦБ и критериального значения.***

Пример 6.

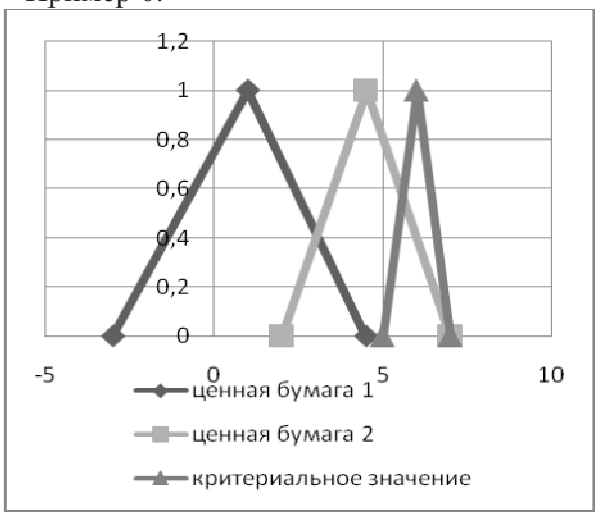



***Рис. 7. Взаимное расположение нечетких доходностей ЦБ и критериального значения для треугольных ФП.***

2. Случай монотонно- возрастающей зависимости «доходность- риск»

При задании первой ЦБ низкой волатильности и низкой доходности, а второй ЦБ-высокой волатильности и высокой доходности и при задании критериального значения меньше ожидаемой доходности менее доходной ЦБ зависимость «доходность- риск» носит возрастающий характер (см.рис. 8-10, примеры 7,8).

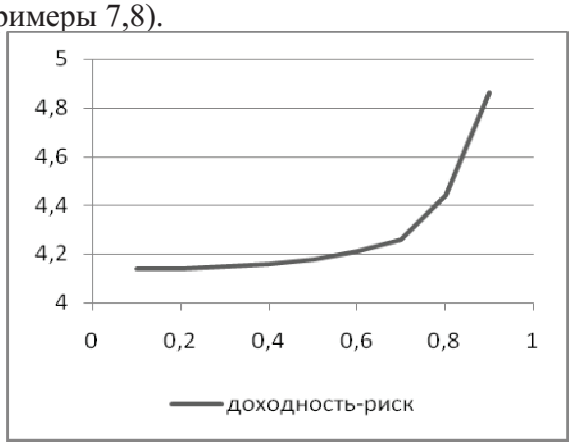



***Рис 8. График зависимости «доходность- риск»***

Пример 7.

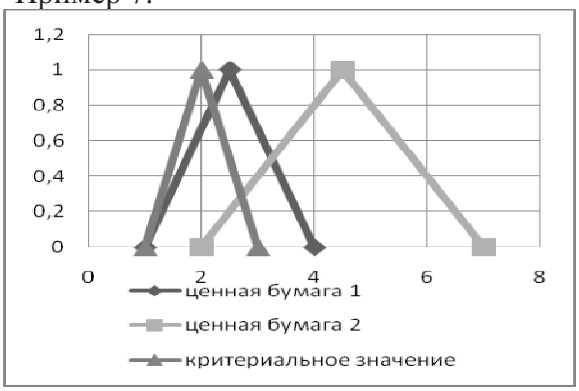



***Рис. 9. Взаимное расположение нечетких доходностей ЦБ и критериального значения***

Пример 8.

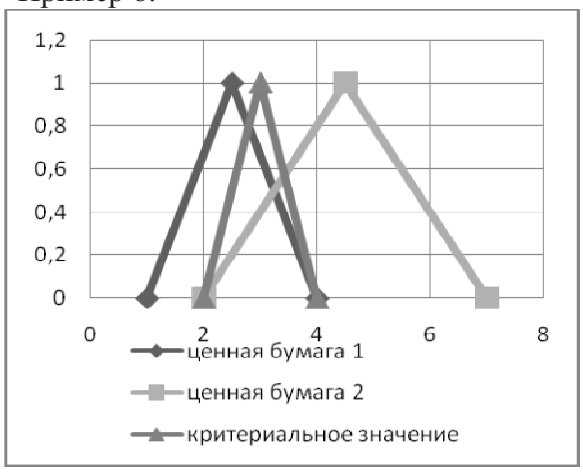



***Рис. 10. Взаимное расположение нечетких доходностей ЦБ и критериального значения***

## Выводы

В работе исследована модель задачи оптимизации нечеткого инвестиционного портфеля. Установлены достаточные условия, при которых зависимость «оптимальная доходность-риск» будет иметь убывающий характер, прямо противоположный зависимости для классической модели Марковица-Тоббина.

Проведены экспериментальные исследования условий и оптимальных решений задачи портфельной оптимизации для убывающей и возрастающей зависимостей «доходность-риск», которые полностью подтвердили теоретические результаты.

Зависимость « доходность- риск » нечеткого портфеля носит преимущественно убывающий характер в случаях, когда менее доходная ЦБ имеет высокую или среднюю волатильность, а более ценная ЦБ имеет высокую доходность при малой и средней волатильности, а при этом критериальное значение находится между доходностями обоих ЦБ.

Зависимость «доходность-риск» носит возрастающий характер, в случаях, когда менее доходная ЦБ имеет малую или среднюю волатильность, а более доходная ЦБ имеет высокую волатильность, а критериальное значение меньше доходности менее ценной ЦБ или расположено между ожидаемыми доходностями обоих ЦБ.

## Список литературы

1. Зайченко Ю.П., Малихех Есфандиярфард. Анализ инвестиционного портфеля для различных видов функций принадлежности // Системні дослідження та інформаційні технології. - 2008.- №2.- С. 59-76.
2. Зайченко Ю.П., Малихех Есфандиярфард. Нечеткий метод индуктивного моделирования для прогнозирования курсов акций в задачах портфельной оптимизации // Вісник Черкаського державного технологічного університету. – 2008. - № 1.- С. 9-14.
3. Зайченко Ю.П., Малихех Есфандиярфард, Заика А.И. Анализ инвестиционного портфеля на основе прогнозирования курсов акций // Вісник національного технічного університету України «КПІ». «Інформатика, управління та обчислювальна техніка». – К.: ТОО «ВЕК+», 2007. - № 47. - С. 168-179.
4. Zaychenko Yurii, Esfandiyarfard Maliheh. Optimization of the investment portfolio in the conditions of uncertainty // International Journal "Information Technologies and Knowledge" Volume 2. - 2008. – Vol.2. - Number 3. - P. 225-233.

Поступила в редакцию 16.12.2009

УДК 004.4'242

*БУЗОВСКИЙ О.В.,*
*АЛЕЩЕНКО А.В.,*
*ПОДРУБАЙЛО А.А.*

# СИСТЕМА АВТОМАТИЧЕСКОЙ ГЕНЕРАЦИИ КОДОВ ПО ГРАФИЧЕСКИМ СХЕМАМ АЛГОРИТМОВ

В статье рассматриваются построение графических схем алгоритмов, процесс генерации кодов по графическим схемам алгоритмов. Как практическая сторона реализована система построения графических схем, предназначенных для дальнейшей генерации программных кодов. Программа позволяет создавать графические схемы, редактировать, сохранять и открывать их, а также генерировать соответствующие схемам программные коды и выдавать результаты их выполнения.

In article construction of algorithm graphical scheme, process of generation code of a program by graphical scheme is treated. As the practical side system of building graphical scheme for generation code of a program is realized. The software product makes it possible to create algorithm graphical schemes, edit, save and open they, to generate code of a program and to show results of their execution.

Целью работы является разработка приложения, реализующего построение графического представления программных функций [1] и последующую генерацию кода в заданном подмножестве языков программирования (в частности Java, Pascal, C). Основными функциями приложения являются следующие:

1. Построение изображения графических схем алгоритмов (ГСА).
2. Структурный, синтаксический и частично семантический контроль.
3. Генерация программного кода.

При проектировании редактора в качестве аналогов рассматривались следующие программные средства: графический редактор в Microsoft Office Word [2], Microsoft Office Visio, Kivio [3], Блоксхематор [4], Редактор блок-схем алгоритмов [5], Редактор блок-схем [6]. Основными недостатками данных программных средств являются:

1. Отсутствие возможности генерации кодов (кроме двух последних средств).
2. Отсутствие контроля ГСА.
3. Отсутствие специализации (Word, Visio, Kivio).

Перечисленные недостатки делают актуальным разработку собственного редактора.

При генерации кода основная сложность связана с проблемами типизации переменных. Данная проблема может рассматриваться в двух аспектах:

- отображение типа данных в окне редактора,
- автоматическое связывание [7].

В системе в данный момент реализовано, как отображение данных в окне редактора, так и частично, автоматическое связывание. Для выполнения функций автоматического связывания используется анализ кода (парсер), описанный ниже.

## Внутреннее представление графической схемы алгоритма. Абстрактный компонент граф-схемы

Для унификации различных операций, проводимых над всеми компонентами граф-схем алгоритмов, как то прорисовка, сохранение, выделение компонент, была выделена абстракция, содержащая общие для различных типов элементов ГСА свойства и методы их обработки. В разработанной программной системе такому абстрактному компоненту соответствует абстрактный класс AbstractComponent (см. UML-диаграмму на рис. 1).

Свойство serialVersionUID присутствует для полного соответствия требованиям интерфейса Serializable, что в свою очередь необходимо для корректного сохранения объектного представления ГСА в файл и дальнейшего его восстановления.

Поле blockProperties предназначено для хранения смысловых свойств компонент граф-схемы. Класс Properties, использованный в данном случае, позволяет хранить любые объекты, при этом для доступа исполь-

зуются объекты-ключи. В разработанной системе было решено использовать в качестве ключей строки с названиями свойств. Строки определяются в качестве закрытых неизменяемых полей (в данном классе – это свойство SELECTED), а доступ к хранимым значениям осуществляется с помощью соответствующих методов чтения и записи (таких, как isSelected и setSelected). Таким образом значительно снижается вероятность выполнения каких-либо некорректных действий с данными, хранимыми внутри blockProperties.

Абстрактный метод draw добавлен для дальнейшей реализации прорисовки конкретного элемента в классах-наследниках.

Класс AbstractComponent используется в методах, ориентированных на обработку всех составляющих блок-схемы, как блоков, так и связей между ними.

### Абстрактный блок

Данная абстракция выделена для унификации операций и свойств различных типов блоков граф-схемы алгоритма. Узлы граф-схемы различных типов имеют значительно больше общих между собой свойств, нежели те же узлы и соединения между ними. Соответственно класс AbstractBlock, реализующий в программном продукте абстракцию блока, имеет значительное число полей и методов доступа к ним.

Каждый блок, помимо свойства выделяемости, определенного в классе AbstractComponent, имеет абсциссу и ординату центра, длину, ширину, строку содержащегося в нем оператора и два соединения с другими блоками – входящее и исходящее. Также предусмотрено хранение структуры данных, образованной в результате лексического анализа содержащегося значения. Все вышеперечисленные свойства хранятся в закрытых полях, и доступ к ним возможен только с помощью специально определенных в этом классе методов по такому же механизму, как и тот, который был описан в классе AbstractComponent. Также в классе описаны методы, позволяющие, исходя из этих, непосредственно хранимых, свойств, вычислить координаты вершин прямоугольника, содержащего в себе данный блок (т.е. касающийся блока всеми четырьмя сторонами).

Абстрактный блок содержит два метода перемещения: moveTo, перемещающий центр узла в точку с абсолютно заданными координатами, и moveByOffSet, выполняющий перемещение блока в соответствии с заданным смещением.

Так же в этом классе реализованы специфицированные в абстракции компонента методы contains и isSelected. Первый проверяет, принадлежит ли точка с абсолютно заданными координатами данному элементу блок-схемы, второй определяет, попадает ли данный блок в прямоугольную область выделения, задаваемую в аргументах. Также реализован вариант метода contains, определяющий, принадлежит ли точка данной фигуре с некоторым допуском.

Абстрактным в данном классе остается лишь метод draw, призванный определять прорисовку конкретного компонента.

Диаграмма классов внутреннего представления ГСА приведена на рис. 2.

### Модель алгоритма

Модель алгоритма – это внутреннее представление графической схемы алгоритма в целом. Класс AlgorithmModel, реализующий абстракцию модели ГСА, содержит ссылки на все экземпляры компонент, входящих в данную блок-схему. Класс соответствует требованиям интерфейса Serializable, что позволяет его запись в бинарные объектные потоки с последующим сохранением в файл.

Модель алгоритма обладает широкой функциональностью, позволяющей обеспечить все необходимые для работы с ГСА операции.

Диаграмма классов внутреннего представления ГСА приведена на рис. 1 и рис.2.

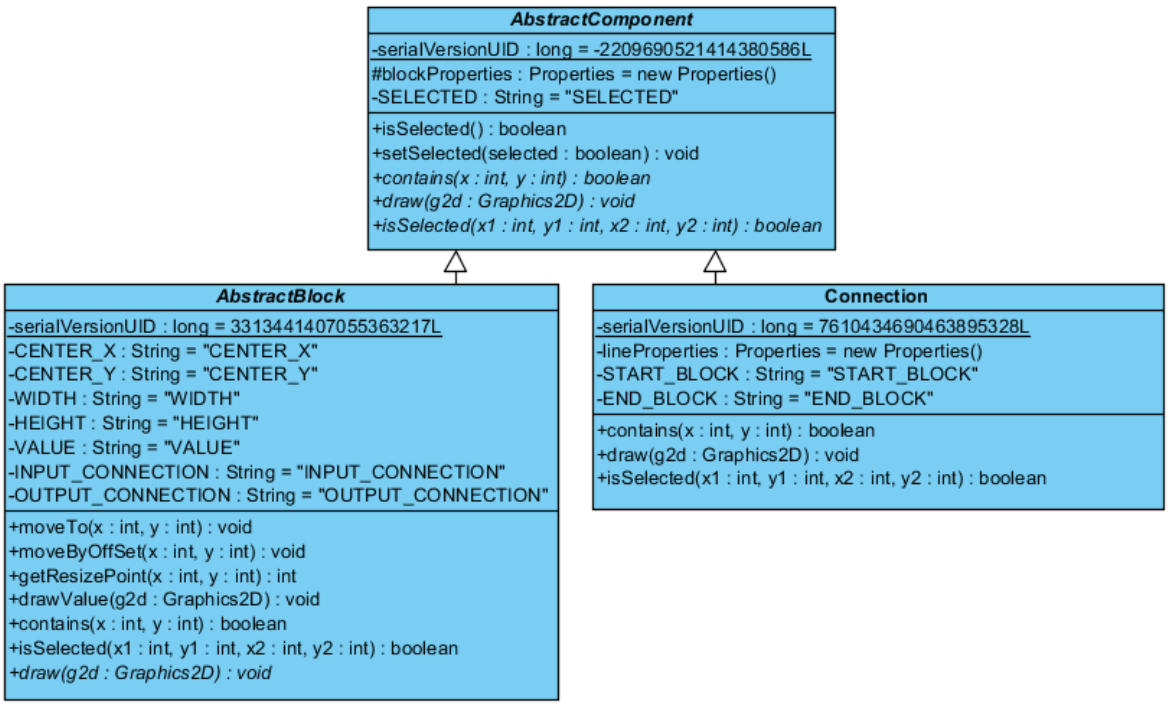



*Рис. 1. Диаграмма классов-абстракций блоков и связей между ними*

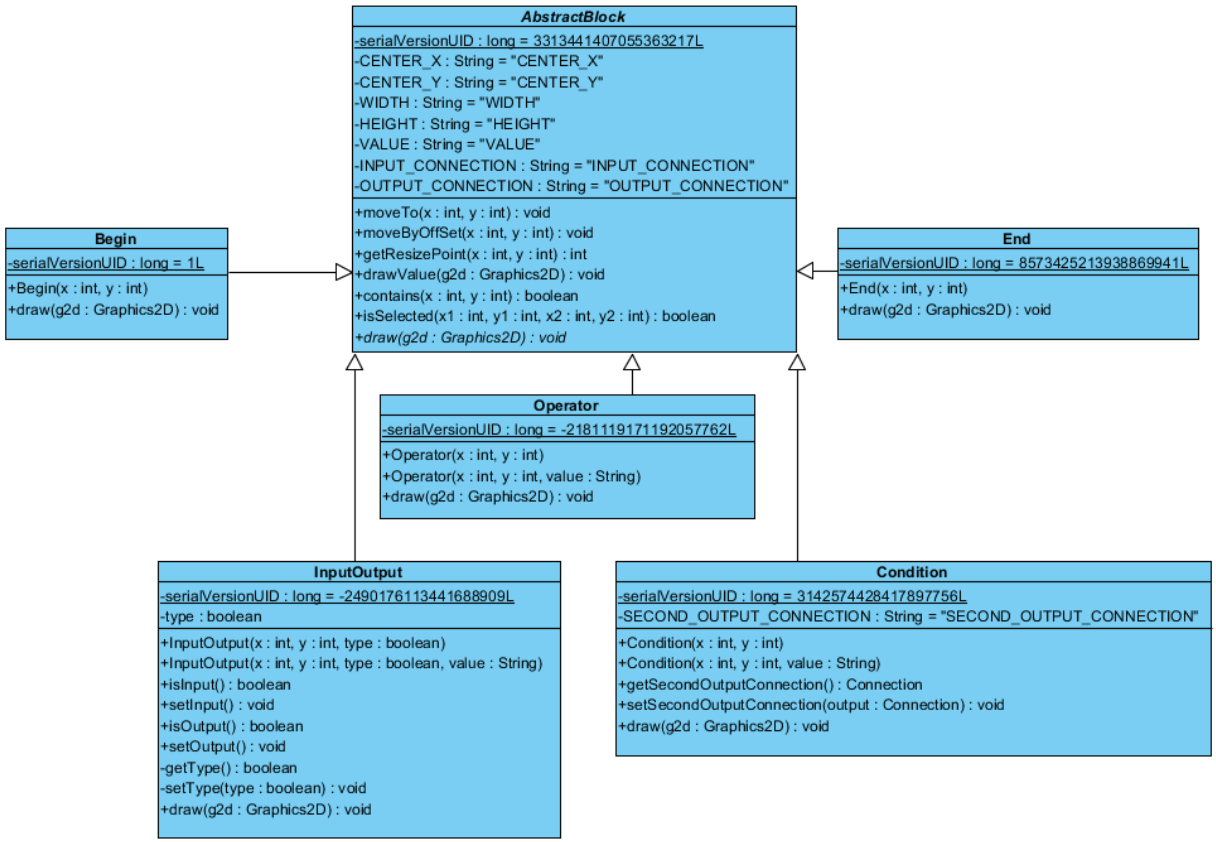



*Рис. 2. Диаграмма классов, представляющих узлы ГСА*

### Подсистема проверки корректности построения блок-схемы

Блок-схема представляет собой дерево, начинающееся от терминальной вершины «Начало» и заканчивающееся терминальной вершиной «Конец».

При составлении графической схемы алгоритма могут иметь место следующие структурные ошибки:

- отсутствие терминальных вершин;
- множественность терминальных вершин одного класса;

- наличие бесконечных циклов (частично, в той мере, в какой оно определяется структурой блок-схемы);
- недостижимость вершины ГСА.

Проверка корректности структуры ГСА выполняется алгоритмом, описываемым с помощью диаграммы активности, приведенной на рис. 3.

Проверка наличия терминальных вершин, а также поиск «висячих вершин», не соединенных ни с одной из терминальных, осуществляется простой проверкой значений соответствующих переменных (задающих начальную и конечную вершину ГСА в первом случае и предыдущей и следующей вершины на последнем этапе). Диаграмма активности поиска вершин, недостижимых из текущей, приведена на рис. 4.

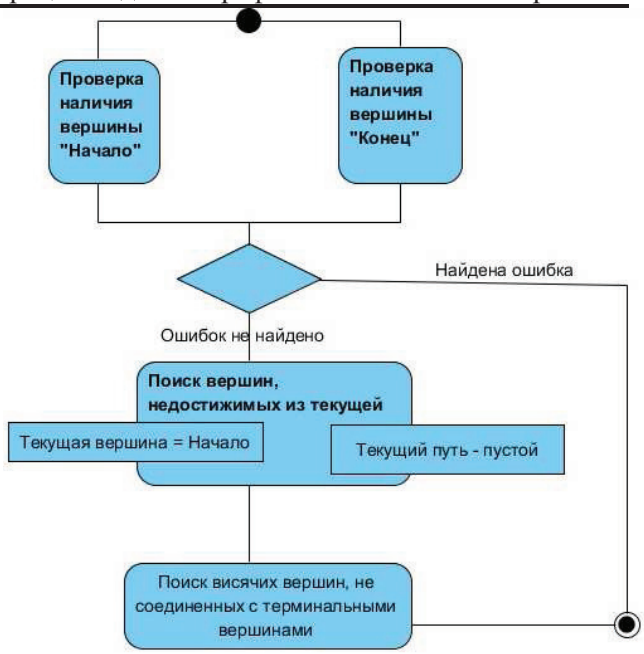



***Рис. 3. Диаграмма активности при проверке корректности структуры ГСА***

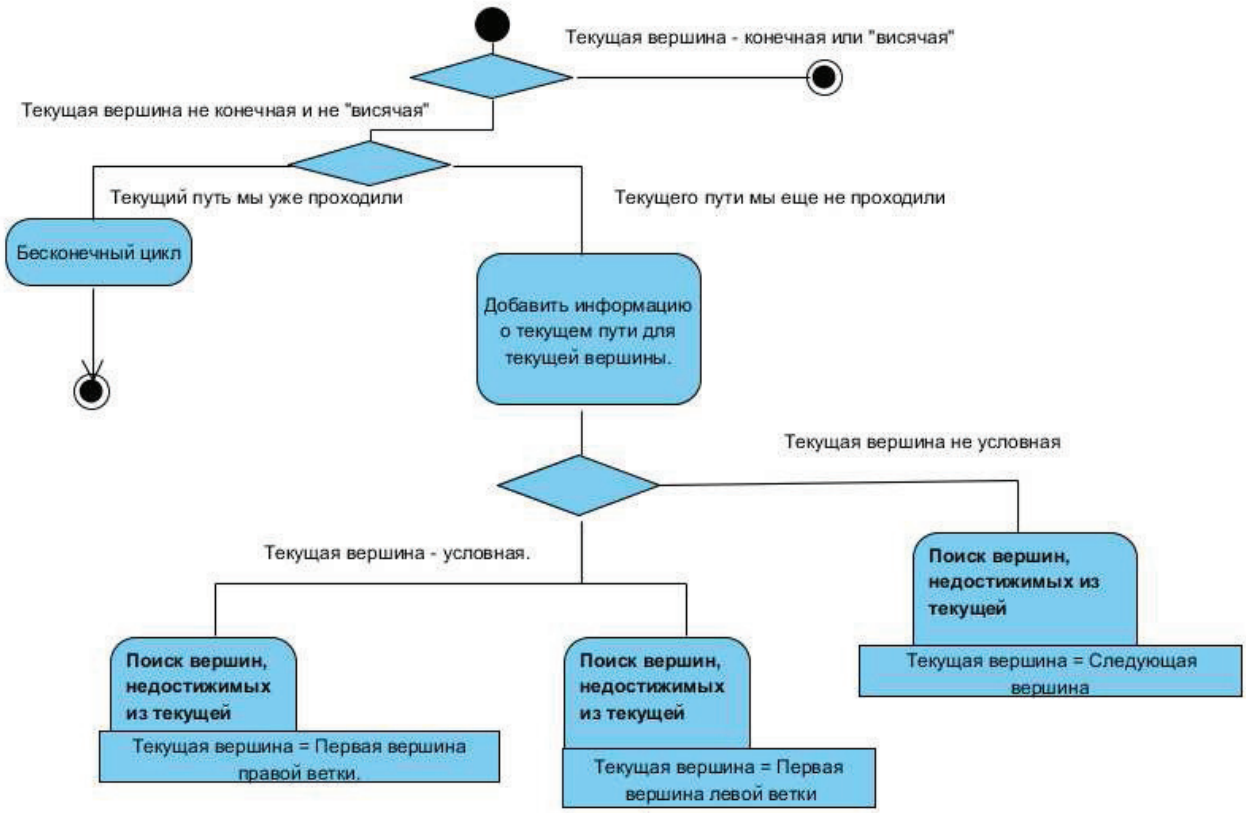



***Рис. 4. Диаграмма активности поиска вершин, недостижимых из текущей***

В соответствии с диаграммами активности проверка корректности ГСА сводится к алгоритму с возвратом поиска путей из каждой вершины в конечную. При отсутствии такого пути определяется количество бесконечных циклов и «висячие вершины».

## Подсистема синтаксического и лексического анализа. Принципы программной реализации автоматов синтаксического и лексического разбора

Лексический и синтаксический анализ оператора реализован с помощью нескольких детерминированных конечных автоматов разного уровня. Автоматы задаются с

помощью четырех таблиц, специфицированных в программе с помощью интерфейса Initiator:

- таблица классификации входных символов;
- таблица переходов ДКА;
- таблица выполняемых действий;
- таблица классификации возможных ошибок.

В таблице классификации входных символов каждому входному ASCII символу сопоставляется номер класса. Символы входного алфавита разделяются на классы, при этом символы одного класса одинаково обрабатываются.

Таблица переходов ДКА представляет собой массив целых чисел, задающий функцию переходов автомата. Двум аргументам (текущему состоянию и классу входного символа) сопоставляется следующее состояние и выполняемое действие. При этом если номер следующего состояния равен длине таблицы, это означает корректное завершение работы автомата, а в случае, когда он превышает длину таблицы, разность между ними определяет номер ошибки в таблице исключительных ситуаций.

Таблица выполняемых действий представляет собой массив экземпляров объектов, которые выполняют преобразования над элементами дерева синтаксического разбора. Также действием может быть предусмотрен переход в подавтомат.

Таблица классификации возможных исключительных ситуаций содержит экземпляры класса Exception. В случае обнаружения ошибки во входной последовательности автомат при помощи механизма исключительных ситуаций «выбрасывает» соответствующую ошибку, которая затем может быть обработана на уровне семантического анализа для информирования пользователя и попытки ее исправить.

В системе генерации кодов предусмотрено четыре конечных автомата разбора: автомат анализа операторов, автомат анализа имен переменных и функций, ДКА разбора числовых значений и автомат разбора текстовых констант.

Программная последовательность для реализации работы ДКА, заданного при помощи вышеуказанных четырех таблиц, определена в классе Thinker. Метод parse этого класса возвращает в качестве результата вершину дерева синтаксического разбора оператора. Этот класс имеет ссылку на экземпляр модели алгоритма, что позволяет обеспечить уникальность вводимых в дерево синтаксического разбора переменных.

Также введен класс StatementParser, приспособленный для поэтапного анализа значений всех операторных и условных блоков в ГСА. Он содержит дополнительные методы, связанные с корректным добавлением новых узлов в дерево синтаксического разбора.

### Целевая структура данных синтаксического и лексического анализа

Абстракцией узла дерева синтаксического и лексического разбора является класс Operand. Операндами являются целое число, действительное число, строка, переменная, функция и оператор. Операнд имеет строчное значение, которое сформировано в результате работы автомата анализа. В дальнейшем, при генерации кодов, разнообразие видов узлов облегчит трансляцию на соответствующий язык программирования.

Экземпляры класса Function в поле значения хранят имя функции и ссылку на аргумент, также являющийся узлом дерева разбора.

В поле значения экземпляров класса Statement содержится название оператора. Также этот объект имеет связи с левой и правой частями оператора, в свою очередь являющихся узлами дерева лексико-синтаксического разбора.

Связь между элементами дерева разбора реализована посредством непосредственных ссылок на операнды, хранимых в узлах дерева.

### Автомат разбора оператора

Детерминированный конечный автомат разбора оператора предназначен для корректного добавления новых операций в дерево разбора по следующим правилам:

- операция присваивания устанавливается в качестве начальной вершины дерева разбора. В дереве может быть лишь один оператор присваивания;
- остальные операции разделяются на уровни. Операция i-го, более высокого уровня, чем текущая, занимает в дереве разбора

место в правой ветке операции (i–1)-го уровня, при этом оператор, ранее стоявший в этом месте, становится левой веткой операции i-го уровня. В случае добавлении операции более низкого уровня, чем текущая, эта операция добавляется в правую ветку текущей.

Разделение операций по уровням показано в таблице 1.

***Табл. 1. Разделение математических и логических операций по уровням***

| Уровень операции | Операция |
|---|---|
| 0 | = (присваивание) |
| 1 | \| (логическое ИЛИ) |
| 2 | & (логическое И) |
| 3 | < (меньше) > (больше) |
| 4 | + (сложение) – (вычитание) |
| 5 | / (деление) * (умножение) % (деление по модулю) |

Пример дерева лексико-синтаксического анализа для оператора b = a+c+e > f+d | k&p представлен на рис. 5.

Автомат разбора операторов программно реализуется при помощи класса StatementParser, для инициализации которого используется класс ParseStatement. Граф автомата разбора операторов представлен на рис. 6.

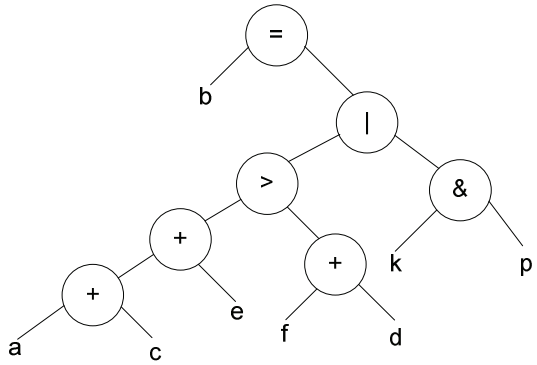



***Рис. 5. Пример дерева лексико-синтаксического разбора***

### Автомат разбора имен переменных и функций

Данный автомат предназначен для добавления в дерево разбора переменных и функций. Имя переменной или функции начинается с буквы и может содержать буквы, цифры, символы подчеркивания «_», точку и квадратные скобки. Функция отличается от переменной тем, что содержит круглые скобки, содержащие оператор, который называется аргументом функции.

Когда автомат встречает другой символ во входной последовательности, он завершает свою работу и передает управление автомату более высокого уровня.

Граф ДКА анализа имен переменных и функций представлен на рис. 7.

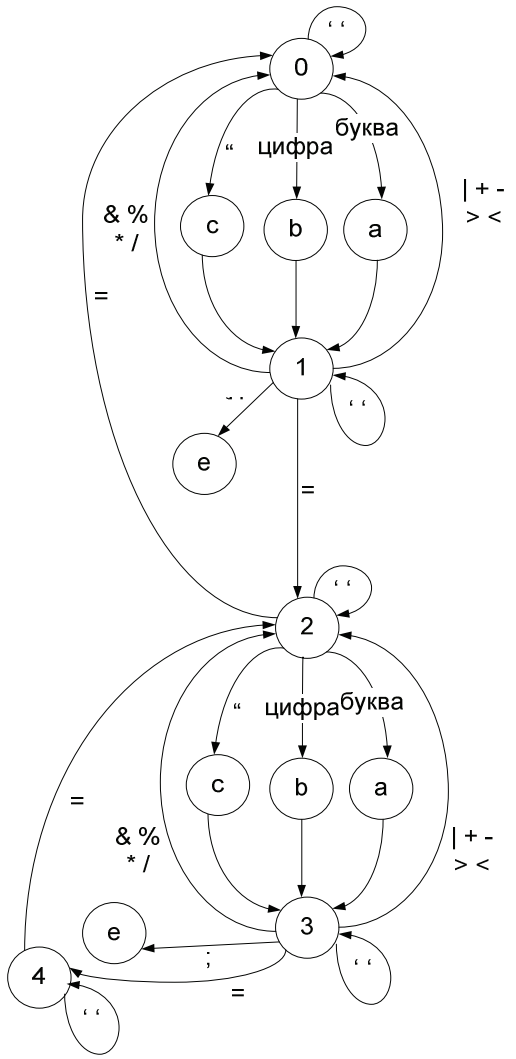



***Рис. 6. Граф автомата разбора операторов a – переход к подавтомату разбора имен переменных и функций, b – переход с подавтомату разбора численных констант, c – переход к подавтомату разбора строчных констант, e – корректное завершение работы автомата.***

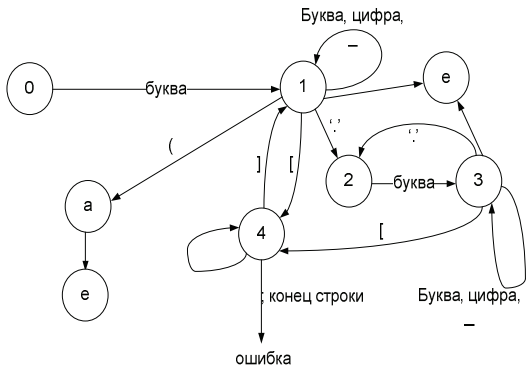



***Рис. 7. Граф автомата разбора переменных и функций. а – переход к анализу аргументов функции, е – корректное завершение работы автомата.***

### Автомат разбора числовых значений

Числа – целые или действительные - начинаются с цифры. Разделителем между целой и дробной частью в действительном числе служит точка. Действительное число может начинаться с точки, в таком случае его целая часть равна нулю.

В случае, если автомат встречает другой символ (не цифру и не точку) он завершает свою работу и передает управление автомату более высокого уровня.

Числовые значения хранятся в соответствующих узлах дерева лексико-синтаксического разбора в виде текста, однако работа данного подавтомата гарантирует возможность корректного преобразования его в целый либо вещественный числовой тип при помощи стандартных средств языка Java.

Граф автомата-анализатора числовых констант представлен на рис. 8.

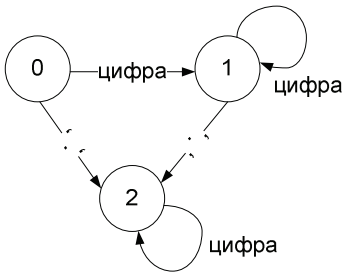



***Рис.8. Граф автомата разбора числовых констант.***

### Автомат разбора строчных констант

Строчная константа начинается и заканчивается двойными кавычками. Внутри строчной константы разрешено использовать любые символы, за исключением комбинаций с символом «обратный слеш» («\»), не являющихся служебными символами. Служебными являются комбинации «\n» (переход на новую строку), «\t» (табуляция), «\"» (двойные кавычки) «\\»(обратный слеш).

Граф переходов детерминированного конечного автомата анализа строчных констант представлен на рис. 9.

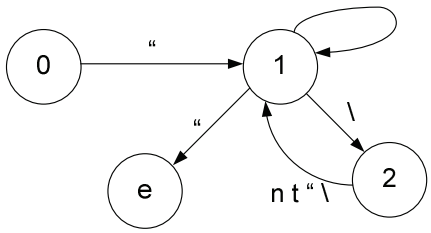



***Рис. 9. Граф автомата разбора строчных констант.***

### Подсистема генерации кодов

Этапу генерации кода предшествуют этапы построения блок-схемы (расположение блоков, их связывание и указание значений) и верификации (проверка корректности структуры блок-схемы, лексический и синтаксический анализ значений операторов). В случае корректного прохождения этапа проверки в каждом операторном и условном блоке формируется дерево лексико-синтаксического разбора, а модель алгоритма содержит список всех используемых переменных. С этого момента начинается этап генерации кода. Вначале пользователю предлагается ввести типы используемых данных в специальном окне. После этого происходит формирование части программного кода с указанием используемых переменных и их типов. Затем осуществляется формирование тела программы путем обхода дерева блок-схемы. На этом этапе важно выделить в блок-схеме три различные структуры: цикл с предусловием (рис.10), цикл с постусловием (рис.11) и обычное ветвление (рис.12).

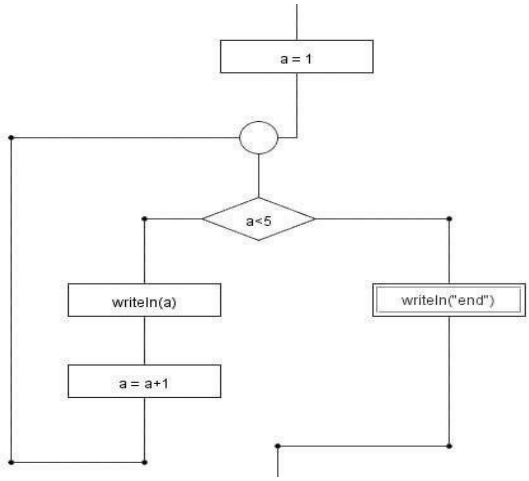



***Рис. 10. Пример цикла с предусловием***

Аналізуючи результати дослідження можна прослідкувати деяку схожість з теорети-

чними даними. Завдяки проведеним розрахункам і побудові графіка, що зображений вище, ми довели, що час роботи програми залежить не лише від кількості вузлів у топології, але й від кількості ребер, що з'єднують ці вузли.

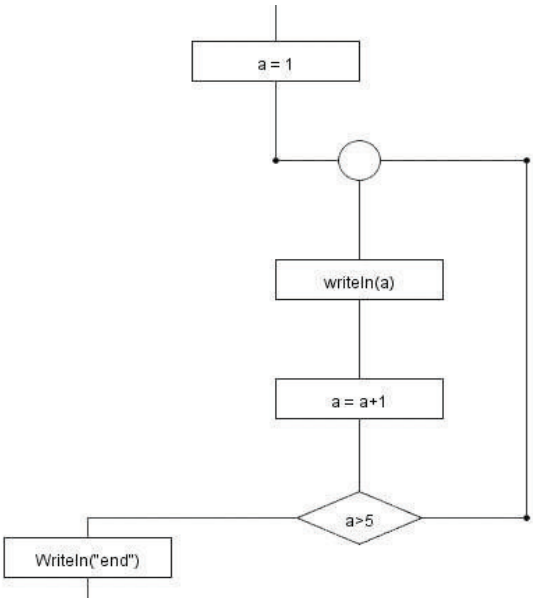



***Рис.11. Пример цикла с постусловием***

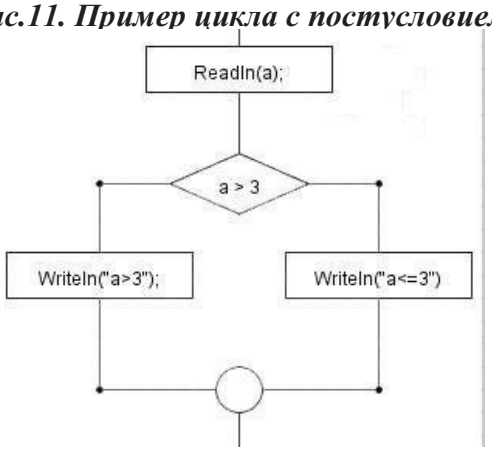



***Рис 12. Пример нециклического ветвления***

В случае, если текущая вершина – операторная, в код программы добавляется ее значение, оттранслированное на заданный язык программирования.

В том случае, если текущая вершина – предикатная (рис. 12), используется метод, возвращающий узел слияния, в котором сходятся обе ветки данной условной вершины. В текст программы добавляется оператор нециклического ветвления. Затем дважды рекурсивно запускается метод «step» для левой и правой ветки условия, при этом в качестве конечной вершины обхода задается найденный узел слияния. Таким образом обеспечивается рассмотрение условия как целостной структуры. В дальнейшем обход продолжается с выходного соединения найденного узла слияния.

Если текущий блок является узлом слияния, это является признаком цикла. В случае, когда концом исходящего соединения узла слияния является предикат (см. рис. 10), данный цикл является циклом с предусловием. Если же предикат является началом еще не пройденного входного соединения узла слияния (рис. 11), определяется цикл с постусловием. В обоих случаях для обработки тела цикла рекурсивно запускается метод «step», и дальнейшая обработка начинается со следующей за циклом вершины, что обеспечивает формирование цикла как единой структуры.

Разработанная система может быть использована как инструмент обучения программистов начального уровня, а также, при некоторой модификации, как дополнение к существующим CASE-системам для генерации внутреннего последовательного кода методов.

## Список литературы

1. Р. Лингер, Х. Миллс, Б. Уитт «Теория и практика структурного программирования», М.: «Мир», 1982 – с. 99.
2. http://office.microsoft.com/
3. http://www.koffice.org/kivio/
4. http://tinyelectronic.org.ua/sites/default/files/programs/block-schem.exe
5. http://vicking.narod.ru/flowchart/index.html
6. http://alglib.manual.ru/blseditor.zip
7. Р. У. Себеста «Основные концепции языков программирования» - М.: "Вильямс", 2001 – с. 179.

Поступила в редакцию 17.12.2009

УДК 004.89

*ЗАЙЧЕНКО Ю.П.,*
*БАСАРАБ А.В.*

# ПРИМЕНЕНИЕ МЕТОДОВ КОМПЛЕКСИРОВАНИЯ АНАЛОГОВ И НЕЧЕТКОЙ ЛОГИКИ ДЛЯ ПРОГНОЗИРОВАНИЯ БИРЖЕВЫХ ИНДЕКСОВ

Предложено применение методов комплексирования аналогов, контроллера Мамдани для анализа и прогнозирования биржевых индексов и курсов акций. Разработано программное обеспечение, реализующее данные методы, а также метод экспоненциального сглаживания, используемый для сравнения производительности и качества прогноза. Проведенные эксперименты показали высокую эффективность методов. Приводятся рекомендации по применению методов для прогнозирования различных выборок.

Method of analog complexity and Mamdani Controller application to loan indexes forecasting was proposed. Application software which implements these methods was built. In advance, Exponential smooth is used in order to compare these methods. Experiments shows good quality and applicability of these methods. Recommendation concerning usage of these methods are added.

## 1. Постановка задачи

Первый фондовый индекс в США был рассчитан Чарльзом Доу в 1884 году.

Идеальный фондовый индекс должен отвечать определенным критериям, оправдывающим ожиданиям пользователей [1].

В целом, поскольку фондовые индексы акумулируют в себе стоимость акций компаний, входящих в него, то изменение индекса также может быть описано как случайный временной процесс.

Со времени появления математических методов оценки и прогнозирования временных процессов а также оценки стоимости акций существует разделение основных методов анализа на фундаментальный и технический [2].

На данный момент, продолжают развиваться как фундаментальные так и технические методы анализа. В техническом анализе используются как графические так и математические методы [1].

Недостатками графических методов с использованием только классических фигур является размытость критериев построения той или иной фигуры. Также, классические методы довольно хаотичны, субьективны и не прослеживается строгой системы [3]. Дополнительным недостатком является необходимость человека строить графики.

Большинство используемых математических методов технического анализа [1] используют довольно простой математический аппарат, как правило сводящийся к исследованию экспоненциального сглаживания или разницы между начальной и конечной ценами акций за определенный период, например, за день. В силу простоты эти методы обладают как преимуществами так и недостатками. Преимущества – простота. Недостатки – тоже простота прогноза и, как следствие, неточность рассчета.

В данной статье рассматриваються некоторые математические методы технического анализа.

## 2. Выделение не решенных ранее частей проблемы

В рассмотренной ранее литературе [1,2,3] не предоставляется рекомендаций по применению и сравнительных характеристик Методов комплексирования аналогов, Нечеткого контроллера Мамдани и Экспоненциального сглаживания для задач прогнозирования финансовых индексов и цен акций. Метод комплексирования аналогов в постановке, описанной в литературе [4] не применим для анализа и прогнозирования финансовых индексов.

Исследования методов на основе нечеткой логики для прогнозирования финансовых индексов не сравнивает методы на основе нечеткой логики совместно с Методом экспоненциального сглаживания и Методом комплексирования аналогов и сравнительный анализ эффективности данных методов на одной и той же выборке не проводился.

### 3. Цель данной статьи

Выбор наиболее оптимального метода из Метода на основе нечеткого контроллера Мамдани, Метода комплексирования аналогов и Экспоненциального сглаживания для прогнозирования биржевых индексов. Критический анализ этих методов для прогнозирования биржевых индексов. Рекомендации по применению Методов комплексирования аналогов и Методов на основе контроллера Мамдани.

### 4. Методы рассчета

Для прогнозирования биржевых индексов использовались методы: Комплексирования аналогов, Экспоненциального сглаживания, Метод на основе нечеткой логики - контроллер Мамдани.

#### 4.1 Метод комплексирования аналогов

Идея метода комплексирования аналогов заключается в поиске нескольких аналогов к прогнозируемому процессу в исторической выборке. После нахождения аналогов строится прогноз на их основе. Различные вариации метода заключаются в разных методах выбора аналогов, а также методах построения прогноза на их основе.

Метод используется для прогнозирования многомерных случайных процессов

$$x(t) = [xj(t)]_{j=1,m}, t = 0,1,...,T \qquad (1)$$

заданных в виде матрицы наблюдений

$$X = \| X_{ij} \| \qquad (2)$$

где $X_{ij}$ – признаки процесса .

Для решения задачи прогнозирования повторяющихся случайных событий необходимо кроме матрицы X задать также матрицу векторов выходных величин Y. Для этой задачи важно, чтобы были коррелированы столбцы выборки X и выборки Y. Корреляция между строками выборки обычно отсутствует.

Для правильного прогноза событий необходима полнота и представительность выборок X и Y [4].

Недостатком является и то что ширина паттерна не подлежит изменению и равняется одной строке исходной выборки. Прогноз строится на один шаг вперед.

В экспериментах в качестве обучающей и прогнозной выборок использовались биржевые индексы, которые представляют собой одномерный случайный временной процесс. В этом случае, применение Метода комплексирования аналогов не возможно в постановке, описанной в литературе [4].

Было предложена модификация метода комплексирования аналогов.

Основные отличия –

- в качестве паттерна рассматривался участок временного ряда, а не строка.
- ширина паттерна – произвольна.

В качестве меры близости использовалась эвклидова метрика

$$d(Ai, B) = \| x_{Ai} - xB \| \qquad (3)$$

После нахождения нескольких аналогов строился прогноз на их основе. При построении прогнозных величин учитывалось временное расстояние от прогнозной даты, то есть более «старые» данные имели меньшее влияние на прогноз чем более «новые» данные. Для этого, были введены весы аналогов, при этом, вес аналога был обратно пропорциональным расстоянию по времени между аналогом и прогнозируемой выборкой.

#### Преимущества предложенного метода комплексирования аналогов

Паттерн произвольной длины позволяет обучать систему и строить прогноз на основе повторяющихся событий, например, волн на бирже, что, аналогично классическим методам технического анализа. Тем не менее, в отличие от классических методов, система не использует стандартный набор фигур а самостоятельно подбирает их на этапе обучения. На большой исторической выборке система может обнаружить закономерности лучше чем эксперт, использующий классические методы. На этапе обучения находится несколько аналогов. Далее, прогноз строится на основе нескольких аналогов.

Можно строить прогноз не на 1 а на несколько шагов, что полностью отвечает требованиям, предьявляемым к современным системам прогнозирования, и расширяет сферу практического применения метода.

Не было необходимости строить корелляционную матрицу.

#### Метод экспоненциального сглаживания

Для анализа и сравнения методов использовался метод экспоненциального сглаживания. Сглаживание – это способ, обеспечивающий быстрое реагирование вашего прогно-

за на все события, происходящие в течение периода протяженности базовой линии.

Основная идея применения метода сглаживания состоит в том, что каждый новый прогноз получается посредством перемещения предыдущего прогноза в направлении, которое дало бы лучшие результаты по сравнению со старым прогнозом. Метод широко описан в литературе [1].

Метод экспоненциального сглаживания применяется в связи с его простотой и в тоже время, хорошими прогнозными показателями. Соотношение усилия/качество прогноза являются непременным преимуществом данного метода.

### 4.3 Методы на основе нечеткой логики

Другим методом сравнения являлся метод на основе нечеткой логики. Контроллер Мамдани – широко известный метод нечеткой логики.

Обьединение нечеткого контроллера и нейронной сети позволяет увеличить преимущества и избежать недостатков. Такой подход использует нейронную сеть для оптимизации конечных показателей обычного нейронного контроллера или для удаления правил.

Основной особенностью методов на основе нечеткой логики является их способность принимать неточные или зашумленные сигналы и строить точный прогноз на их основании.

Методы на основе нечеткой логики применяются для прогнозирования различных случайных процессов, например [5] или для задач извлечения данных Data Mining [6].

Было предложено применение данного метода для анализа и прогнозирования биржевых индексов.

## 5. Эксперименты

Была разработана система для анализа и прогнозирования биржевых индексов которая реализовывает алгоритмы экспоненциального сглаживания, комплексирования аналогов и контроллер с нечетким выводом Мамдани.

Для построения прогноза была выбрана выборка индекса Dow Jones за 322 рабочих дня.

Временной процесс можно представить в виде:

$$f(x) = f_1(x) + f_2(x) + ... + a(x) \qquad (4)$$

где $f_n(x)$ – функция, сумма которых влияет на индекс, $a(x)$ – случайный «шум».

***Таблица 1. Пример выборки Dow Jones за 322 рабочих дня***

| № | Значение Индекса |
|---|---|
| 1 | 235,776,756 |
| 2 | 235,879,175 |
| 3 | 235,366,594 |
| 4 | 235,360,467 |
| 5 | 234,971,401 |
| ..... | ..... |

Прогноз строился на 4, 6, 8 и 10 шагов. При построении прогноза, прогнозные значения сравнивались с реальными показателями индекса и точность прогноза сравнивалась по коэффициантам:

Среднепроцентная ошибка

Среднеквадратическая ошибка

Средняя ошибка

Коэффициент Тейла

Результаты прогноза на 4 шага представлен в таблице 2. В столбцах – коэффициенты, в строчках – метод прогнозирования.

***Таблица 2. Прогноз индекса Dow Jones на 4 шага***

| | **Средне процентная ошибка** | **Среднеквадратическая ошибка** | **Средняя ошибка** | **Коэффициент Тейла** |
|---|---|---|---|---|
| Экспоненциальное сглаживание | 0,90287 | 29,210 | 0,00902 | 0,009174 |
| Комплексирование аналогов | 1,03942 | 33,441 | 0,01039 | 0,009765 |
| Нечеткий контроллер | 0,54163 | 18,539 | 0,00541 | 0,006051 |

Прогноз на 6 шагов представлен в таблице 3. В столбцах – коэффициенты, в строчках – метод прогнозирования.

***Таблица 3. Прогноз индекса Dow Jones на 6 шагов***

| | **Средне процентная ошибка** | **Среднеквадратическая ошибка** | **Средняя ошибка** | **Коэффициент Тейла** |
|---|---|---|---|---|
| Экспоненциальное сглаживание | 0,71819 | 24,555 | 0,00718 | 0,00725 |
| Комплексирование аналогов | 1,25623 | 43,646 | 0,01256 | 0,0131 |
| Нечеткий контроллер | 0,77629 | 27,635 | 0,00776 | 0,00896 |

Прогноз на 8 шагов представлен в таблице 4. В столбцах – коэффициенты, в строчках – метод прогнозирования.

***Таблица 4. Прогноз индекса Dow Jones на 8 шагов***

| | **Средне процентная ошибка** | **Среднеквадратическая ошибка** | **Средняя ошибка** | **Коэффициент Тейла** |
|---|---|---|---|---|
| Экспоненциальное сглаживание | 0,8824 | 29,7671 | 0,00882 | 0,00897 |
| Комплексирование аналогов | 0,6435 | 23,7698 | 0,00643 | 0,00733 |
| Нечеткий контроллер | 0,33772 | 11,7573 | 0,00337 | 0,00384 |

Дополнительно проводились эксперименты по прогнозированию курсов валют и индекса РТС данными методами. Обучающая выборка индекса РСТ составляла 1500 значений. Результаты подтвержают полученные результаты при прогнозировании индекса Dow Jones.

## 6. Выводы

Как видно из таблиц 2-4, прогноз индекса Dow Jones наиболее точен при прогнозировании с помощью нечеткого контроллера Мамдани, что показывает перспективность применения метода для подобного рода задач. При применении данных методов для анализа и прогнозирования различных типов выборок: курсов валют, а также индекса РТС оказалось, что с увеличением размера выборки точность прогноза методом комплексирования аналогов возрастает. Это обьясняется тем, что Метод комплексирования аналогов зависим от полноты выборки. Большая обучающая выборка резко увеличивает качество обучения метода, и таким образом влияет на прогноз. В большинстве случаев, прогноз Методом экспоненциального сглаживания демонстрировал большие среднеквадратические ошибки по сравнению с другими методами.

При увеличении количества шагов, на которые строится прогноз, ошибка увеличивается незначительно, что свидетельствует о высоком качестве прогноза. При построении прогноза на 6 шагов, ошибка незначительно уменьшилась, по сравнению с прогнозом на 4 шага. В данном случае, это скорее, благо-

приятное совпадение чем закономерность. При большом количестве испытаний системы на нескольких выборках ошибка увеличивалась с увеличением количество шагов, на которые строился прогноз. Увеличение ошибки всегда было умеренным, среднепроцентная ошибка была меньше 2%, что свидетельствует о высоком качестве прогноза.

Низкое значение коэффициента Тейла свидетельствует о высокой корреляции прогноза и реальных значений индекса.

## 7. Заключение

Из приведенных экспериментов следует, что качество прогноза рассмотренных методов полностью соответствует требованиям, предьявляемым к системам, работающим на бирже. Программное обеспечение, которое реализует Методы комплексирования аналогов, Методы нечеткой логики на основе контроллера Мамдани рекомендуется применять для анализа и прогнозирования курсов акций и курсов валют. Эксперименты по сравнению данных методов с Методом экспоненциального сглаживания и с результатами игры биржевых игроков продемонстрировали и доказали высокую эффективность методов для анализа и прогнозирования курсов акций. При анализе большой исторической выборки, рекомендуется использовать Метод комплексирования аналогов, для анализа относительно небольших выборок рекомендуется использовать Методы на основе нечеткой логики, например, на основе Контроллера Мамдани.

## Список литературы


1. Технический анализ с и его приминение на примере данных полученных с Московской фондовой биржи. <http://referat.niv.ru/referat/031/03100016.htm>
2. Хаертфельдер, Лозовская, Хануш. Фундаментальный и технический анализ рынка ценных бумаг.: Питер, 2004. – 352с.
3. Аппель Дж. Эффективные инвестиции. Как зарабатывать на росте и падении акций, инфляции, скачках цен на нефть... и не только / Дж. Аппель; пер. В.А. Кукушкиной. - СПб.: Питер, 2009. - 416 с.
4. Зайченко Ю.П. Основи проектування інтелектуальних систем: Навч. посіб. для студ. вищ. навч. закл. – К.: Вид. Дім “Слово”, 2004. – 352 с.
5. Голубева Е.Н., Васенёв Н.Ф., Гусева Е.Е., Королева Н.В., Галаншина Е.А. Исследование модернизированного узла дискретизации на пневмомеханической прядильной машине < http://www.igta.ru/files/konferencia/progress/2002/progress2002.pdf>.
6. Чернов И.А. Автоматизированное извлечение знаний из баз данных. < http://masters.donntu.edu.ua/2006/fvti/ichernov/diss/index.htm >


Поступила в редакцию 14.12.2009

УДК.004.358

*КОРОТКИЙ Е.В.,*
*ЛЫСЕНКО А.Н.*

# МЕТОД МОДЕЛИРОВАНИЯ РЕКОНФИГУРИРУЕМЫХ СЕТЕЙ НА КРИСТАЛЛЕ

Проведено огляд існуючих підходів до моделювання мереж на кристалі. Запропоновано метод агентного моделювання реконфігуруємих мереж на кристалі при будь-якому рівні абстракції.

A review of existing approaches to modeling networks on chip is presented. The method for agent based modeling of reconfigurable networks on chip at any level of abstraction is proposed.

## Введение

В связи с интенсивным развитием технологий производства интегральных микросхем (ИМС) остро встает вопрос объединения компонентов в больших системах на кристалле. Наиболее распространенный подход, использующий принцип общей шины, показывает отсутствие масштабируемости и уменьшение пропускной способности с увеличением числа соединяемых элементов. Одним из методов устранения подобных недостатков может стать использование сетевых технологий для обмена данными между подсистемами ИМС. Впервые концепция сети на кристалле (СтнК) сформулирована в [1]. Такой подход к организации связи в ИМС обладает преимуществами масштабируемости (с увеличением размера сети растет ее пропускная способность) и параллелизма (данные в разных сегментах сети передаются одновременно). Основные тенденции развития СтнК отображены в [2,3]. Подходы к реализации сетевого взаимодействия внутри микросхем с реконфигурированной архитектурой (FPGA) описаны в [4,5].

Использование универсальной статической архитектуры подсистемы связи для решения всего спектра возможных задач приводит к избыточности и, как следствие, к увеличению энергопотребления и необходимых для реализации аппаратурных ресурсов. Для разрешения проблемы избыточности в [6] предложена сеть с реконфигурируемой архитектурой. Такой подход позволяет в реальном времени адаптировать структуру сети к требованиям пропускной способности и потребляемой мощности.

Важную роль в процессе проектирования и верификации СтнК играет моделирование. Основные результаты исследований, предпринятых в этом направлении, отражены в [7-13]. В [8] утверждается, что на момент публикации работы не существовало средств, позволяющих моделировать реконфигурируемые СтнК при любом уровне абстракции. Анализ более поздних источников [10] подтвердил эту информацию.

Выше изложенное предопределяет актуальность данной работы и диктует необходимость разработки метода моделирования реконфигурируемых СтнК на разных уровнях абстракции, что и является целью предлагаемой статьи.

## Общие сведения о СтнК

Обобщенная структура СтнК показана на рис.1. Для обозначения подсистем ИМС использован термин ГВМ, что расшифровывается как Готовый Вычислительный Модуль. Непосредственно СтнК состоит из маршрутизаторов М, соединенных согласно определенной топологии. Блоки М управляют потоком пакетов данных, проходящим через порты ввода/вывода М, осуществляя маршрутизацию на основании адреса назначения указанного в заголовке пакета. Сетевой интерфейс СИ преобразует данные из формата передачи характерного для ГВМ в поток дискретных пакетов и наоборот.

Использованием описанной структуры подсистемы связи достигаются следующие преимущества:

- масштабируемость – для увеличения пропускной способности сети достаточно добавить новые маршрутизаторы и подключить к ним необходимые ГВМ;
- параллелизм – данные в разных сегментах сети передаются одновременно, что приводит к увеличению пропускной способности подсистемы связи.

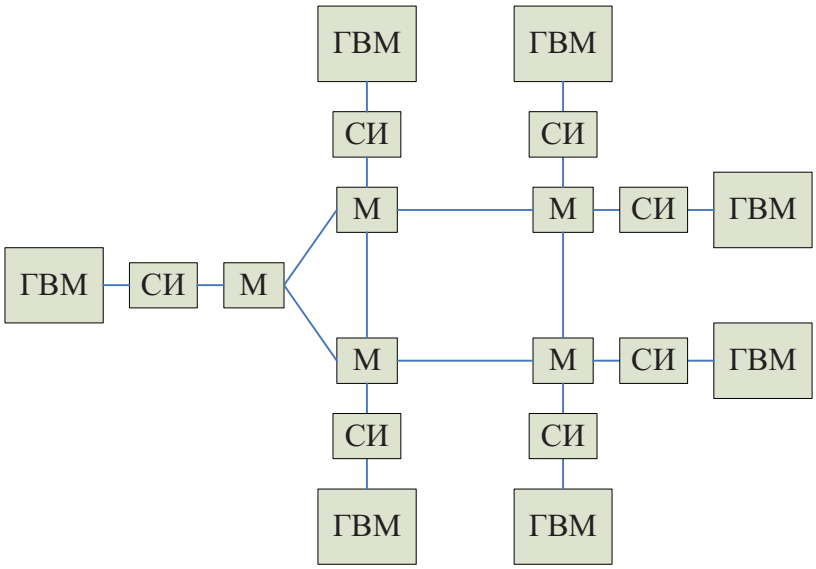



*Рис. 1. Структура сети на кристалле*

### Структура СтнК с реконфигурируемой архитектурой

Кроме достоинств, описанных в предыдущем разделе, СтнК присущи следующие недостатки [1,2]:

- сложность разработки;
- большое количество аппаратурных ресурсов необходимых для реализации;
- большая потребляемая мощность.

Первый из недостатков является общим свойством систем, предназначенных для эффективного решения сложных задач. Путем решения проблемы является создание специализированных САПР, облегчающих работу инженера.

Второй и третий недостатки проистекают из статичности архитектуры СтнК. В [6] показано, что использование универсальной статической архитектуры подсистемы связи для решения всего спектра возможных задач приводит к избыточности и, как следствие, к увеличению энергопотребления и необходимых для реализации аппаратурных ресурсов. Для решения проблемы в [6] предлагается использовать динамически реконфигури-руемую архитектуру СтнК, которая позволяет в реальном времени адаптировать ее структуру к требованиям пропускной способности и потребляемой мощности (рис. 2).

Как видно из рис.2, реконфигурируемая архитектура СтнК состоит из самой сети, подсистемы мониторинга контролируемых параметров и блока модификации структуры. На основании ошибки управления *e*, характеризующей отклонение требуемых параметров сети от текущих, блок модификации осуществляет минимизацию *e* путем изменения структуры СтнК. В качестве контролируемых параметров могут выступать пропускная способность, необходимые для реализации ресурсы ИМС и потребляемая мощность.

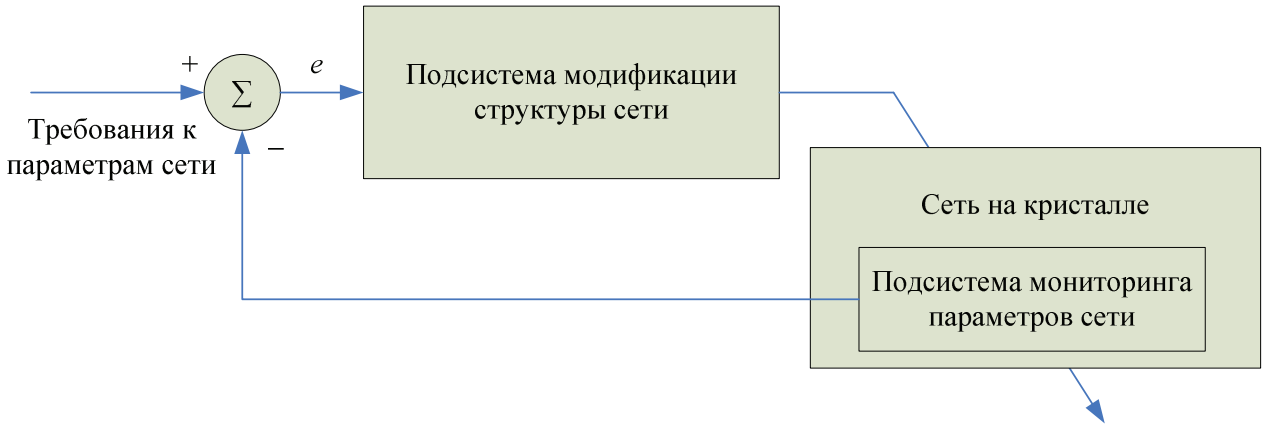



*Рис. 2. Обобщенная структура СтнК с реконфигурируемой архитектурой*

Действия блока модификации зависят от выбора параметров, подлежащих оптимизации. Например, для управления энергопотреблением и необходимыми для реализации ресурсами можно динамически изменять размер очередей портов ввода-вывода маршрутизаторов [14]. Управление пропускной способностью осуществляется путем модификации топологии сети и таблиц маршрутизации блоков М [6].

### Обоснование выбора метода моделирования реконфигурируемой СтнК

В контексте моделирования СтнК будем различать следующие уровни абстракции [7]:

- системный (высокий) – моделирование динамического изменения структуры сети –модификация количества маршрутизаторов и топологии;
- поведенческий (средний) – моделирование обмена данными на пакетном уровне –пропускной способности, задержек, алгоритмов маршрутизации, потерь пакетов;
- физический (низкий) – моделирование на регистровом/транзисторном уровнях – исследование потребляемой мощности и требуемых для реализации ресурсов ИМС.

В [7-9] предлагаются средства моделирования СтнК лишь на поведенческом (среднем) уровне абстракции. Используемый метод – дискретно-событийная имитация. Для исследования особенностей работы сети на физическом уровне в [10-13] разработаны точные RTL (Register Transfer Level) модели с использованием языков описания цифровых схем – Verilog и VHDL. Большим недостатком такого подхода является значительные временные затраты при моделировании на компьютере либо необходимость в наличии дорогостоящих устройств для аппаратурной имитации. Таким образом, ни один из предложенных методов не позволяет проводить моделирование СтнК при любом уровне абстракции, а преобразование модели от одного уровня к другому требует значительных временных затрат.

Проведенный обзор литературы показал, что на сегодняшний день отсутствуют средства для моделирования реконфигурируемых СтнК. Такая ситуация обусловлена тем, что существующие подходы к поведенческому моделированию СтнК используют дискретно-событийную имитацию (средний уровень абстракции), в то время как реконфигурируемая архитектура требует принятия решений о модификации подсистем и связей между ними (системный уровень), что затруднительно при использовании дискретно-событийного подхода.

В [15] выполнен сравнительный анализ методов, применяемых в имитационном моделировании. Результаты приведены на рис.3.

Из рис.3 видно, что наиболее универсальным является агентный подход к имитационному моделированию. Суть его состоит в том, что предметная область представляется в виде множества агентов, взаимодействующих между собой. Разработчиком модели описываются правила создания, уничтожения и изменения агентов. При этом под термином “агент” понимается некоторый объект (в терминах ООП), обладающий памятью и способностью принимать решения, а значит и собственным поведением разного уровня сложности. Внутренняя структура агента может быть описана разными способами – от формальной логики до нейронных сетей. В момент запуска процесса моделирования, каждый агент начинает функционировать согласно индивидуальному алгоритму работы, и глобальное поведение системы возникает, как результат взаимодействия всего множества агентов. Поэтому агентное моделирование (АМ) называют еще моделированием снизу вверх. Очевидно, что АМ существенно децентрализированно. В отличие от системной динамики или дискретно-событийных моделей, здесь нет такого места, где централизованно определялось бы поведение (динамика) системы в целом. Из этого следуют два важных следствия: во-первых, отсутствие координационного центра создает предпосылки для распараллеливания (ускорения) процесса моделирования, во-вторых, появляется возможность постепенно вносить коррективы в алгоритм работы агента, тем самым, детализируя модель. Таким образом, наличие нескольких сценариев функционирования агента разного уровня сложности дает возможность моделировать работу системы на различных уровнях абстракции.


***Рис. 3. Подходы в имитационном моделировании на шкале уровня абстракции***

К недостаткам вычислительных систем с агентами следует отнести значительно большую опасность взаимоблокировок, чем в других системах моделирования. Решением проблемы может стать использование готовых САПР АМ (например, AnyLogic) [15], в которых проблема взаимодействия агентов решена разработчиками.

По мнению авторов работы, использование агентного подхода для имитационного моделирования СтнК позволит достичь следующих преимуществ:

- моделирование работы СтнК на любом уровне абстракции; согласно [15] ни системная динамика, ни дискретно-событийный подход не предоставляют такой возможности;
- моделирование реконфигурируемых СтнК. Реконфигурируемая архитектура предполагает мониторинг параметров СтнК и принятие решений о модификации ее подсистем в соответствии с требованиями, предъявляемыми к качеству обслуживания в сети; дискретно-событийный подход не дает возможности принятия решений на системном уровне, а системная динамика не позволяет эффективно моделировать информационные системы с пакетной передачей данных [15]; таким образом, для моделирования адаптивной структуры реконфигурируемой СтнК единственным приемлемым решением является использование агентного подхода;
- одновременное функционирование всех агентов в мультиагентной системе приводит к уменьшению времени моделирования по сравнению с дискретно-событийным подходом, в котором алгоритм имитации реализуется, как правило, в виде последовательной программы, обрабатывающей заявки.

## Метод моделирования реконфигурируемой СтнК при любом уровне абстракции

Для реализации представленного в предыдущем разделе метода моделирования реконфигурируемых СтнК авторами предложен подход, содержащий следующие положения:

- все ресурсы реконфигурируемой СтнК (ГВМ, сетевые интерфейсы, маршрутизаторы, подсистемы мониторинга и оптимизации) представляются в виде агентов, функционирующих одновременно;
- индивидуальный алгоритм работы каждого агента описывается при помощи конечного автомата;
- в качестве единого языка описания структуры и связей между агентами используются UML-диаграммы классов;

Рассмотрим предложенную концепцию более подробно.

В зависимости от уровня абстракции модели количество используемых типов агентов может меняться. Например, агентное представление пакетов является приемлемым при моделировании пропускной способности и задержек в СтнК, когда исследователя интересует поведение отдельно взятых пакетов и их групп. Такой подход позволяет придать пакетам индивидуальные свойства, например возможность случайного изменения содержимого, что весьма полезно при моделировании помех на линиях связи. С другой стороны, при низкоуровневой имитации необходимость в агентном представлении пакетов исчезает, поскольку при данной степени абстракции важны другие характеристики системы, такие как потребляемая мощность и взаимное расположение компонентов СтнК на кристалле.

Индивидуальный алгоритм работы каждого агента предлагается задавать в виде конечного автомата. Высокий уровень формализации теории конечных автоматов обуславливает ее применение для описания дискретных цифровых систем и облегчает верификацию созданной модели. С понижением уровня абстракции к алгоритму работы агента добавляются все новые детали, точность модели увеличивается, а сложность соответствующего конечного автомата возрастает.

Предлагаемый метод не ограничивает подходы к описанию функционирования агента одной лишь теорией конечных автоматов. Поскольку концепция агента предполагает инкапсуляцию его внутренней структуры, алгоритм работы отдельно взятого компонента СтнК может быть описан любым удобным для разработчика образом. При условии соблюдения неизменности интерфейса, такая замена внутреннего содержимого агента не повлияет на корректность функционирования остальных подсистем СтнК.

Одним из качеств, обуславливающих эффективное моделирования сложных систем, является использование унифицированного языка представления модели. В предлагаемом методе в качестве такого языка используется UML 2.0. Сделанный выбор обусловлен тем, что UML является наиболее распространенной и удобной нотацией для описания статических и динамических объектно-ориентированных систем, к которым, безусловно, можно отнести и мульти-агентные модели [15]. Для представления структуры СтнК в предложенном методе используется UML-диаграмма классов. На рис. 4 показана СтнК на системном уровне абстракции в терминах UML.

Отношения между объектами задаются при помощи технологий агрегирования и осведомленности. Агрегирование подразумевает, что один объект (агент) владеет другим и несет за него ответственность (в терминах создания, удаления и изменения). Например, из рис. 4 видно, что объект СтнК содержит объекты маршрутизаторов, сетевых интерфейсов, ГВМ, подсистем мониторинга и модификации структуры сети. Таким же образом можно описывать буферы ввода-вывода как составную часть, например, маршрутизатора на более низком уровне абстракции.

Различие между агентом и объектом при таком подходе состоит в том, что агент имеет более сложное индивидуальное поведение и способен принимать решения.

Говоря же об осведомленности, имеется в виду, что объекту (агенту) известно о другом объекте (агенте). Осведомленные агенты могут запрашивать друг у друга операции (делегация), но они не несут никакой ответственности друг за друга (создание, удаление, модификация). Осведомленность – это более слабое отношение по сравнению с агрегированием, поскольку оно предполагает гораздо менее тесную связь между объектами (агентами). В предлагаемом методе осведомленность реализуется в виде ссылки на объект (агент) и используется для задания и динамической модификации топологии СтнК. Каждый маршрутизатор осведомлен о других маршрутизаторах и сетевых интерфейсах, подсоединенных к нему. Такой подход позволяет легко моделировать реконфигури-

руемую топологию сети, поскольку для ее изменения достаточно обновить соответствующие ссылки.

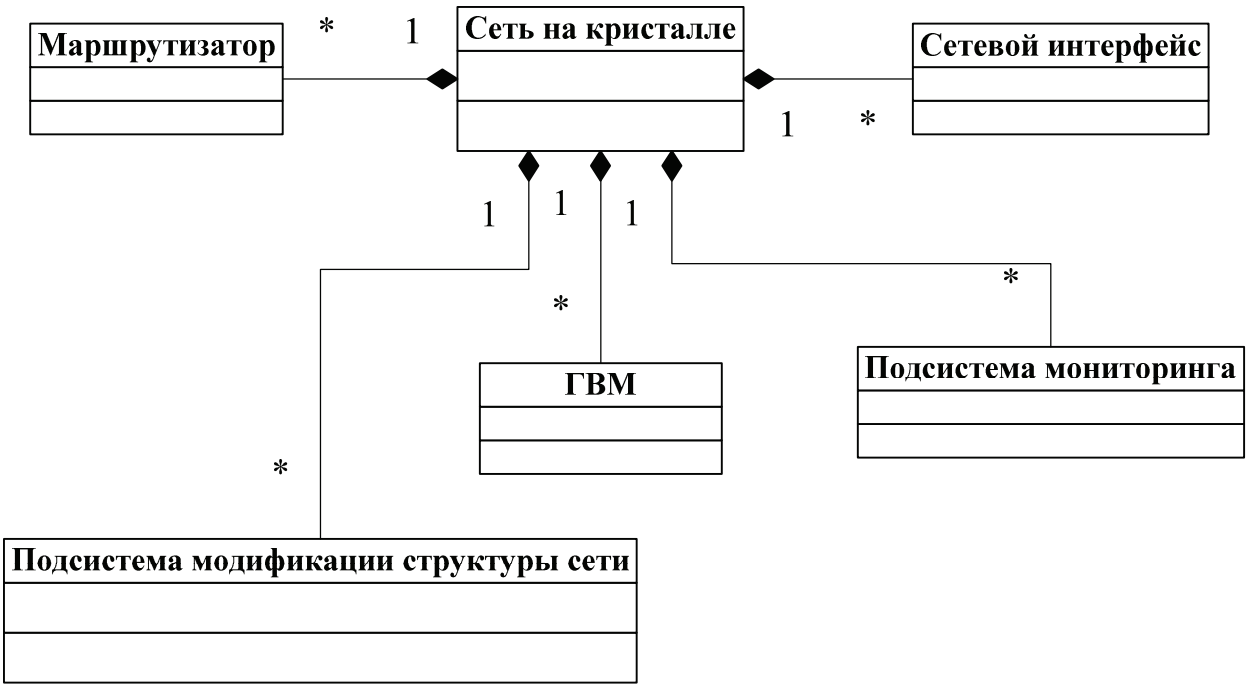



***Рис. 4. Описание структуры СтнК при помощи UML-диаграммы классов***

## Выводы

По результатам проведенных исследований можно сформулировать следующие выводы:

- обосновано применение мульти-агентного подхода к моделированию СтнК. Такое решение позволяет имитировать работу СтнК с реконфигурируемой архитектурой за счет возможности внесения изменений на любом уровне абстракции от преобразования топологии сети на системном уровне до модификации таблиц маршрутизации (средний уровень) и размеров очередей портов ввода-вывода маршрутизаторов на низком уровне;
- предложен метод мульти-агентной имитации реконфигурируемых СтнК, что создает предпосылки для сокращения времени моделирования за счет одновременного функционирования всех агентов в системе;
- использование языка UML 2.0 для описания структуры модели и связей между агентами упрощает освоение разработчиками предложенного метода, поскольку UML является широко распространенным унифицированным языком описания статических и динамических объектно-ориентированных систем, к которым, безусловно, можно отнести и мульти-агентные модели;
- полученные результаты являются гипотезой, поэтому следующим шагом исследований в данном направлении будет создание рабочего макета системы и получение количественных сравнительных характеристик относительно других средств моделирования СтнК.

## Список литературы

1. *Dally W., Towles B.* Route packets, not wires: on-chip interconnection networks // Proceedings of the 38th annual Design Automation Conference (June 2001). – Las Vegas, USA. – P.684-689.
2. *Gebali F., Elmiligi H., Watheq El-Kharashi M.* Networks-on-Chip: Theory and Practice.– Boca Raton (USA): CRC Press/Taylor and Francis Group LLC, 2009. – 307p.
3. *Bjerregaard T., Mahadevan S.* A survey of research and practices of Network-on-chip // ACM Computing Surveys. – 2006. –Vol.38, №1. – P.1-51.

4. *Brebner G., Levi D.* Networking on Chip with Platform FPGAs // Proceedings on Field-Programmable Technology (FPT) IEEE International Conference (December 2003). – P.13-20.
5. *Janarthanan A.* Networks-on-Chip based high performance communication architectures for FPGA's. – University of Cincinnati. Division of Research and Advanced Studies.– Cincinnati, USA, 2008. – 143p.
6. *Vincenzo R., Atienza D.* A Reconfigurable Network-on-Chip Architecture for Optimal Multi Processor SoC Communication // 16th IFIP/IEEE International Conference on Very Large Scale Integration (October 2008). – Rhodes, Greece. – P. 321-326.
7. *Rikard T.* A Network on Chip Simulator: Master of Science Thesis. – Royal Institute of Technology (KTH). Department of Microelectronics and Information Technology. – Kista, Sweden, 2002.– 55p.
8. *Ali M., Welzl M.* Using the NS-2 Network Simulator for Evaluating Network on Chips // Proceedings on IEEE International Conference of Emerging Technologies ICET '06 (November 2006). – P.506-512.
9. *Hang-Sheng W., Xinping Z.* Orion: a power-performance simulator for interconnection networks // Proceedings of the 35th annual ACM/IEEE international symposium on Microarchitecture (November 2002). – Istanbul, Turkey. – P. 294-305.
10. *Genko N., Atienza D., Benini L.* Noc Emulation: A Tool and Design Flow for MPSoC // IEEE Circuits and Systems Magazine. – 2007. – Vol.7, №4. – P. 42-51.
11. *Bertozzi D., Jalabert A., Benini L.* NoC Synthesis Flow for Customized Domain Specific Multiprocessor Systems-on-Chip // IEEE Transactions on Parallel and Distributed Systems.– 2005. – Vol.16, №2. – P.113-129.
12. *Goossens K., Dielissen J.* The Aethereal Network on Chip: Concepts, Architectures and Implementations // IEEE Design and Test of Computers. – 2005. – Vol.22, №5. – P.414-421.
13. *Siguenza-Tortosa D., Nurmi J.* VHDL-based simulation environment for Proteo NoC // Proceedings of the Seventh IEEE International High-Level Design Validation and Test Workshop (October 2002). – P.1-6.
14. *Matos D., Carro L.* The Need for Reconfigurable Routers in Networks-on-Chip // Proceedings of the 5th International Workshop on Reconfigurable Computing: Architectures, Tools and Applications (June 2009). – Karlsruhe, Germany. – P. 275-280.
15. *Борщев А.В.* Практическое агентное моделирование и его место в арсенале аналитика // Exponenta Pro. Математика в приложениях. – 2004. – № 3-4. – С. 38-47.

Поступила в редакцию 17.12.2009

УДК 044.77

*БИЛЫК И. И.*

# ВЛИЯНИЕ УСТРОЙСТВ ПРЕОБРАЗОВАНИЯ СЕТЕВОГО АДРЕСА НА ПРОИЗВОДИТЕЛЬНОСТЬ P2P СИСТЕМ

На сегодняшний день BitTorrent является одним из наиболее популярных peer-to-peer приложений, трафик которого составляет значительную часть всего Internet-трафика. С другой стороны, при построении сети в большинстве случаев используются устройства преобразования сетевого адреса (NAT). Не смотря на существование множества средств для прохождения NAT, вполне вероятно, что приложения, особенно P2P, не могут принимать запросы на входящее подключение, если они находятся за NAT. Хотя это явление широко наблюдалось в измерительных работах, до сих пор в литературе отсутствуют количественные исследования влияния NAT на приложения P2P. В этой статье рассмотрена аналитическая модель для измерения производительности P2P систем, подобных BitTorrent, при наличии однородных и неоднородных NAT-пиров.

BitTorrent nowadays is one of the most popular peer-to-peer applications on the Internet, contributing to a significant portion of the total Internet traffic. On the other hand, NAT devices have become widespread in almost all networking scenarios. Despite of the effort of NAT traversal, it is still quite possible that applications, especially P2P ones, cannot receive incoming connection requests, if they are behind NAT. Though this effect has been widely observed in measurement work, so far there is no quantitative study examining the impact of NAT on P2P applications. This article describes analytical model to capture the performance of BitTorrent-like P2P systems in the presence of NAT peers.

## Введение

В этой статье рассмотрено аналитическое моделирование для оценки воздействия закрытых NAT-пиров на показатели torrent-систем. С помощью модели было установлено негативное влияние присутствия NAT-пиров на производительность системы во время загрузки файла. Также в конце предложены некоторые незначительные изменения протокола, которые позволят существенно улучшить эффективность системы.

## P2P и BitTorrent

Парадигма peer-to-peer (P2P) является многообещающей архитектурой для передачи данных большого объема. Согласно парадигме, в P2P используются каналы участвующих в раздаче пользователей, благодаря чему рабочая нагрузка переносится с серверов на пользователей. Применение внеполосного механизма для распространения метаданных файлов и одного или несколько трекеров (tracker server) для слежения за пирами-участниками позволяет пирам обмениваться блоками файлов напрямую, что значительно снижает нагрузку, связанную с одноточечной моделью распространения данных в клиент-серверной парадигме. Пиры получают недостающие блоки файлов от соседних пиров и одновременно отдают им ранее загруженные. В зависимости от завершенности загрузки пиры делятся на два типа: сидеры – пиры, которые уже имеют полный файл, и личеры – имеют только часть файла, либо файл отсутствует. Для поощрения пользователей за содействие в BitTorrent-подобных системах реализована стратегия tit-for-tat («расплата», «услуга за услугу»). При отсутствии NAT-пиров такая стратегия вполне подходит для построения системы с соблюдением показателей справедливости. Совместно с ней используется стратегия optimistic unchoke. Использование этих стратегий дает пирам возможность находить соседей с большей скоростью отдачи (для личеров) либо загрузки (для сидеров).

## Негативное влияние NAT

В torrent-сообществах очень часто возникают жалобы по поводу NAT. Это ключевая технология в случае недостатка IP-адресов, поскольку NAT позволяет нескольким компьютерам подключаться к интернету используя один внешний IP. С другой стороны, NAT ограничивает направление подключения, так как входящие запросы на подключение отбрасываются, если в таблице преобразования отсутствует информация о инициаторе. Таким образом, во-первых, P2P-при-

ложение, запущенное за NAT, не может принимать входящие подключения от других пиров, из-за чего NAT-пиры будут иметь меньше соседей по сравнению с открытыми. Во-вторых, у пользователей NAT (как правило, за домашними роутерами, подключенные к Internet через DSL или кабельные модемы) скорость отдачи ниже, чем у открытых пиров (с высокоскоростными выделенными каналами). Согласно стратегии «tit-for-tat» в BitTorrent, более вероятно, что NAT-пиры будут подавлены открытыми пирами, поскольку они не могут отдавать с такой же скоростью. В результате, открытые пиры в основном работают между собой, объединяясь в кластеры [1].

## Модель

Взаимодействия в системе показаны на Рисунке 1. Как уже было отмечено ранее, в ней присутствуют два типа пиров: закрытые NAT-пиры, могут подключаться только к открытым соседним пирам; открытые пиры, могут иметь закрытых и открытых соседей.

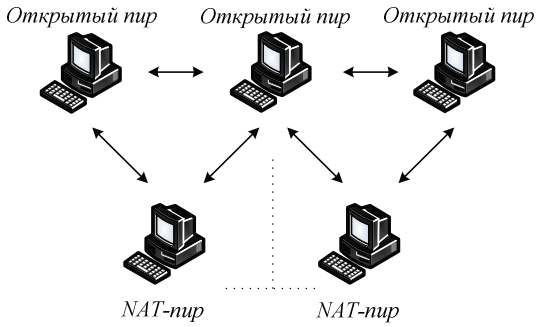



***Рис. 1. Общий вид P2P системы с NAT-пирами.***

В данном случае рассматривается производительность системы в устойчивом состоянии, то есть количество активных пиров постоянно. Модель построена на основе исходного протокола BitTorrent с акцентированием на среднем времени загрузки открытых и закрытых пиров. Также рассмотрены случаи для однородных (открытые и закрытые пиры имеют одинаковую скорость отдачи) и неоднородных систем (скорость отдачи открытых пиров больше, распространенный случай в реальных системах). Используемые обозначения приведены ниже.

$X, Y$ – тип пиров: $S$ (сид), $P$ (открытый), $N$ (NAT-личер).

$F$ – размер файла для загрузки.

$K$ – количество пиров в списке, который возвращается трекером.

$K_X$ – количество $X$ов в списке, который возвращается трекером.

$M$ – максимальное число одновременных отдач для пира.

$N$ – количество пиров в системе.

$N_X$ – количество $X$ов в системе.

$\alpha$ – процент NAT-пиров из общего числа личеров.

$C_X$ – производительность канала отдачи $X$.

$U_X$ – скорость отдачи данных $X$.

$U_{XY}$ – скорость отдачи данных на соединение $X$-$Y$.

$D_X$ – скорость приема данных $X$.

$n^u_{XY}$ – количество $Y$ов, которым отдает $X$.

$n^d_{XY}$ – количество $Y$ов, у которых загружает $X$.

$L_X$ – среднее количество соседних пиров для $X$.

$L_{XY}$ – среднее количество соседних пиров $Y$ для $X$.

$T_X$ – среднее время загрузки для $X$.

Охватить все функции BitTorrent-системы в математической модели практически невозможно, поэтому в данной модели выделены основные принципы работы и приняты следующие допущения:

- Отсутствие узкого места в канале загрузки пира. Как показали результаты измерений в [3], скорость загрузки 90% пиров в torrent-системах не превышает 520 Кб/с. Это медленнее большинства интернет-подключений.
- Производительность канала отдачи равномерно разделяется между соединениями.
- Отсутствие ограничения на максимальное и минимальное количество соседних пиров.
- Участвующие пиры полностью используют производительность их канала отдачи, что было теоретически доказано в [4].
- Личеры покидают раздачу после полной загрузки файла.
- Количество сидеров в системе значительно меньше количества личеров.

Исходя из этих допущений и Рисунка 2 ниже описываются основные идеи моделирования.

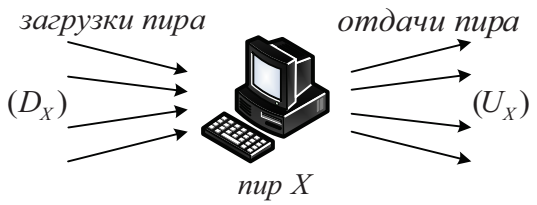



***Рис. 2. Простая аналитическая модель.***

На первом месте стоит задача измерения среднего времени загрузки файла $T_X$ открытым и закрытым пиром в устойчивом состоянии. Для этого необходимо получить среднюю скорость загрузки $D_X$ пира $X$. Очевидно, что $D_X$ определяется средним числом соседних пиров, которые отдают пиру $X$ ($n_{XY}^d$) и скоростью загрузки каждого соединения. С точки зрения всей системы, $n_{XY}^d$ зависит от количества пиров $n_{XY}^u$, которые загружают у пира $X$. В то время как $n_{XY}^u$ зависит от числа соседних пиров $L_X$, числа соседних $Y$ пиров $L_{XY}$, а также стратегий tit-for-tat и optimistic unchoke. Таким образом, начинать следует с выражений для $L_X$ и $L_{XY}$.

Когда новый пир, присоединяясь к раздаче, связывается с трекером, трекер случайно выбирает $K$ пиров и возвращает информацию о них. Чтобы уменьшить нагрузку на трекер, в нем не используются технологии прохождения NAT для определения доступности пиров. Из-за этого список может содержать сидеров, открытых личеров и NAT-личеров. Если учесть, что $N_S << (N_P + N_N)$, количество открытых и закрытых пиров в списке будет $K_P \approx (1-\alpha)K$ и $K_N \approx \alpha K$ соответственно.

Сидеры в torrent-раздаче работают как серверы, поэтому любые соединения между сидерами и личерами инициируются личерами. Таким образом, среднее количество открытых и закрытых соседей для сидера равно:

$$L_{SP} = \frac{N_P}{N} K \approx (1-\alpha)K,$$

$$L_{SN} = \frac{N_N}{N} K \approx \alpha K,$$

откуда среднее количество соседних пиров для сидера равно:

$$L_S = L_{SP} + L_{SN} = K .$$

Связь между открытым и закрытым пирами может устанавливать только закрытый пир. Следовательно, $L_{PN}$ определяется как:

$$L_{PN} = K_P N_N / N_P \approx (1-\alpha)KN_N / N_P = \alpha K .$$

Общее количество запросов соединения среди открытых пиров - $K_P N_P$. Тем не менее, запрос от $A$ к $B$ или от $B$ к $A$ устанавливает одно соединение между $A$ и $B$. По этой причине, количество соединений между открытыми пирами имеет следующий вид:

$$Z = N_P K_P - \frac{N_P}{2}\left(\frac{K_P}{N_P - 1}\right)^2 = N_P K_P - \frac{N_P K_P^2}{2(N_P - 1)}$$

.

Так как в соединении участвуют два пира, $L_{PP}$ записывается как:

$$L_{PP} = \frac{2Z}{N_P} \approx \left(2 - \frac{K}{N_P + N_N}\right)K_P .$$

В широкомасштабной torrent-системе, особенно в период пиковой нагрузки, обычно $K << (N_P + N_N)$. Тогда можно выполнить приближение:

$$L_{PP} \approx 2K_P \approx 2(1-\alpha)K .$$

Исходя из описанного выше, общее количество соседей для открытых пиров:

$$L_P = L_{PP} + L_{PN} = (2-\alpha)K .$$

Для закрытых NAT-пиров соседними могут быть только открытые пиры. Тогда $L_N$ получим как:

$$L_N = K_P \approx (1-\alpha)K .$$

Поскольку $n_{XY}^d$ связано с количеством отдач пира, сначала необходимо получить $n_{XY}^u$. В период пиковой нагрузки у пира, как правило, более $M$ соседей. Поэтому будет справедливым утверждение, что пир может принимать запросы на загрузку как минимум от $M$ соседних пиров. Закрытые пиры имеют возможность отдавать открытым пирам, тогда:

$$n_{XY}^u = M .$$

Выражения для $n_{SN}^u$, $n_{SP}^u$ и $n_{PP}^u$ имеют следующий вид:

$$n_{SN}^u = \frac{L_{SN}}{L_S} M = \alpha M ,$$

$$n_{SP}^u = \frac{L_{SP}}{L_S} M = (1-\alpha)M ,$$

$$n_{PP}^u = M - n_{PN}^u .$$

В устойчивом состоянии открытый пир отдает $G$ закрытым пирам и $H$ открытым пирам по стратегии tit-for-tat и $g$ закрытым и $h$ открытым пирам по стратегии optimistic unchoke. Поэтому, выражение для $n_{PN}^{u}$:

$$n_{PN}^{u} = G + g,$$

а $G$ и $g$ находятся из следующих равенств:

$$G + H = M - 1, \quad (1)$$

$$g + h = 1, \quad (2)$$

$$\frac{H + h}{M\alpha/(1-\alpha)} = \frac{H}{G}, \quad (3)$$

$$\frac{L_{PP} - H}{L_{PN} - G} = \frac{h}{g}, \quad (4)$$

Равенства (1) и (2) получены исходя из соответствующих стратегий. В однородном системе количество соседей, подключенных открытым пиром по стратегии tit-for-tat, должно быть пропорционально количеству соседей, подключаемых этот пир. Исходя из этого записано равенство (3). Числитель в левой части – среднее число открытых пиров, отдающих открытому пиру, а знаменатель – среднее число закрытых NAT-пиров, отдающих открытому пиру. Уравнение (4) записано, поскольку количество соседей, подключенных открытым пиром по стратегии optimistic unchoke пропорционально количеству оставшихся соседей, которые не были подключены этим открытым пиром по стратегии tit-for-tat.

В неоднородной системе открытый пир будет обслуживать $(M-1)$ открытых пиров по стратегии tit-for-tat и, возможно, один закрытый пир по стратегии optimistic unchoke. В результате, $n_{PN}^{u}$ вычисляется как:

$$n_{PN}^{u} = \frac{L_{PN}}{L_P - M + 1}.$$

После получения $n_{XY}^{u}$ необходимо определить $n_{XY}^{d}$:

$$n_{PP}^{d} = n_{PP}^{u} = M - n_{PN}^{U}.$$

$$n_{PN}^{d} = \alpha n_{PN}^{u}/(1-\alpha).$$

$$n_{PS}^{d} = n_{SP}^{u} N_S / N_P.$$

$$n_{NP}^{d} = (1-\alpha) n_{PN}^{u} / \alpha.$$

$$n_{NS}^{d} = n_{SN}^{u} N_S / N_N.$$

Так как пиры могут полностью использовать свой канал отдачи и, как правило, имеют более $M$ соседей, средняя скорость отдачи для каждого соединения определятся следующим образом:

$$U_{SP} = U_{SN} = \frac{U_S}{M}, \; U_{PP} = U_{PN} = \frac{U_P}{M},$$

$$U_{NP} = \frac{U_N}{M}.$$

Тогда средняя скорость загрузки открытых и закрытых пиров примет такой вид:

$$D_P = n_{PP}^{d} U_{PP} + n_{PN}^{d} U_{NP} + n_{PS}^{d} U_{SP},$$

$$D_N = n_{NP}^{d} U_{PN} + n_{NS}^{d} U_{SN}.$$

Среднее время загрузки для открытого пира:

$$T_P = F / D_P.$$

После загрузки файла открытые пиры покидают раздачу, а закрытые пиры скачали только $F_d = T_P D_N$. С этого момента каждый сидер обслуживает $M$ NAT-пиров, пока они не загрузят файл полностью, после чего к сидеру подключаются другие $M$ пиров. Среднее время загрузки для каждой такой группы:

$$T_R = (N_N + M)(F - F_d)/(2N_S U_S). \quad (5)$$

Из равенства (5) следует, что для минимизации времени каждому сидеру нужно отдавать только одному NAT-пиру с полной скоростью, пока он не загрузит файл. Тогда закрытым пиры будут ждать в среднем $T_N^{'} = (N_N + 1)(F - F_d)/(2N_S U_S)$, то есть $T_N$ выражается как:

$$T_N = T_P + T_N^{'}.$$

Среднее время загрузки для всех пиров:

$$T = (1-\alpha)T_P + \alpha T_N.$$

### Оценка модели

Для подтверждения правильности модели используется симулятор, созданный в [2]. В нем реализованы основные технологии BitTorrent, включая стратегии tit-for-tat и optimistic unchoke. При проведении моделирования рассматривался случай пиковой нагрузки, за 5 секунд к раздаче подключаются 500 пиров. Личер находится за NAT, то есть закрытый, с вероятностью $\alpha$. Раздача файла размером 100Мб производится одним сидером. Канал отдачи сидера составляет 1024Кб/с. Личеры покидают раздачу после загрузки всех блоков файла (64Кб). Каждый пир может отдавать $M = 5$ личерам, трекер возвращает информацию о $K = 50$ пирах. Скорость канала отдачи пиров в однородной

системе составляет 384Кб/с, в неоднородной системе открытые пиры имеют 1024Кб/с, закрытые – 384Кб/с. Скорость канала загрузки для всех пиров – 3000Кб/с.

Рисунок 3 отображает среднее количество соседних пиров для сидеров и личеров полученное на симуляторе (С) и с помощью аналитической модели (А). По графику видно, что открытые личеры имеют намного больше соседей, чем NAT-личеры, это является причиной огромной разницы в качестве обслуживания открытых и закрытых пиров. Также с помощью модели можно точно рассчитывать $L_{PP}$, $L_{PN}$, $L_{SN}$ и $L_{SP}$.

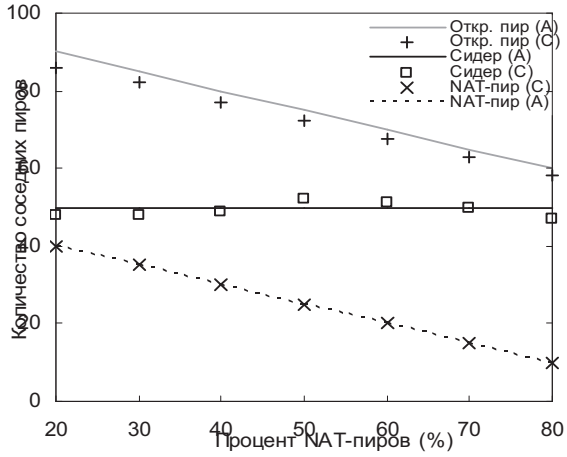



***Рис. 3. Среднее количество соседей для пира.***

На Рисунке 4 отображена производительность torrent-системы, пиры которой имеют одинаковую скорость канала отдачи. На Рис. 4 (также на Рис. 5) ось ординат имеет логарифмическую шкалу и отвечает за время загрузки. Чтобы лучше продемонстрировать влияние NAT-пиров, был добавлен вариант, когда в системе отсутствуют закрытые пиры. Это частный случай при $\alpha = 0$, при этом сидер отдаёт только открытым пирам, а также открытые пиры раздают между собой. Рис. 4 четко показывает, что средняя производительность всех пиров ухудшается с увеличением процента NAT-пиров.

Также следует отметить парадоксальный факт: чем больше в системе закрытых пиров, тем меньше времени на загрузку файла тратят открытые пиры, даже по сравнению с вариантом, когда в системе присутствуют только открытые пиры. Это объясняется тем, что закрытые пиры могут отдавать только открытым, в то время как канал открытых пиров распределяется между открытыми и закрытыми личерами. Поэтому, с увеличением количества закрытых пиров, суммарный объем канала отдачи, предоставляемого открытым пирам, значительно увеличивается.

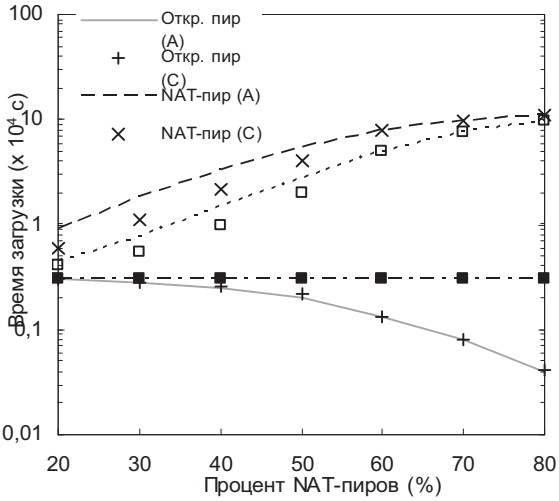



***Рис. 4. Среднее время загрузки для пиров в однородной системе.***

На Рисунке 5 рассмотрен случай неоднородной системы, более приближенный к реальности. Для полноты картины был добавлен дополнительный случай, когда все пиры открытые и разделены на категории: высокоскоростные (ВС) и низкоскоростные (НС). Рис. 5 повторяет результат на Рис. 4. А именно, производительность всей системы и NAT-пиров ухудшается с увеличением процента закрытых пиров. С другой стороны, открытые пиры получают преимущество, когда большинство пиров находятся за NAT. Судя по Рисунку 5, если принимать во внимание присутствие закрытых пиров, общая производительность системы, а также производительность НС-пиров (обычно NAT-пиры) будет значительно хуже, нежели в случае игнорирования присутствия NAT-пиров.

Таким образом, исходя из Рисунков 4, 5 можно сделать вывод, что присутствие NAT-пиров приводит к деградации производительности системы, в особенности – закрытых пиров. Тем не менее, увеличение количества закрытых пиров позволяет улучшить производительность открытых пиров в однородных и неоднородных системах.

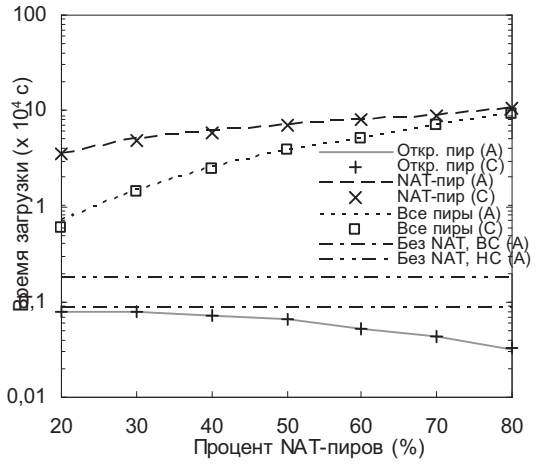



***Рис. 5. Среднее время загрузки для пиров в неоднородной системе.***

## Возможность улучшения эффективности

Для повышения производительности и справедливости torrent-системы можно ввести динамически регулируемый параметр, благодаря которому появится возможность управлять вероятностью выбора NAT-пира по стратегии optimistic unchoke. Необходимо дать системе способность динамически регулировать параметр в зависимости от текущего состояния. Это позволит значительно улучшить качество обслуживания NAT-пиров и справедливость всей системы в целом, в то время как производительность открытых пиров будет несколько занижена в допустимых пределах.

## Заключение

В этой статье было рассмотрено влияние NAT-устройств на P2P BitTorrent-системы. Для этого использовалась аналитическая модель, которая позволяет измерять производительность загрузки системы при наличии однородных и неоднородных NAT-пиров. Путем моделирования было определено, что присутствие в системе закрытых пиров существенно снижает эффективность загрузки NAT-пиров и производительность всей системы. С другой стороны, наличие закрытых пиров дает возможность открытым пирам загружать быстрее, нежели в системе, состоящей только из открытых пиров. Это нелогичный, но вполне объяснимый факт, связанный с недоступностью NAT-пиров и высокой эффективностью распространения данных в BitTorrent-системах. Для повышения производительности системы был предложен метод, суть которого в параметризации вероятности «оптимистичного выбора» открытым пиром закрытого.

## Список литературы

1. Legout, N. Liogkas, E. Kohler, and L. Zhang, “Clustering and sharing incentives in BitTorrent systems,” in Proc. of ACM SIGMETRICS, 2007.
2. Bharambe, C. Herley, and V. Padmanabhan, “Analyzing and improving Bit-Torrent performance,” in Proc. of IEEE INFOCOM, 2006.
3. J. Pouwelse, P. Garbacki, D. Epema, and H. Sips, “The BitTorrent P2P filesharing system: Measurement and analysis,” In Proc. of 4th International Workshop on Peer-to-Peer Systems (IPTPS), 2005.
4. D. Qiu and R. Srikant, “Modeling and performance analysis of BitTorrent-like peer-to-peer networks,” in Proc. ACM SIGCOMM, 2004.

Поступила в редакцию 13.02.2009

УДК 004

*EHSAN EBRAHIM SHIRAZI,*
*STIRENKO S.G.*

# EFFECTS OF INFORMATION TECHNOLOGY ON THE BUSINESS

Information Technology develops extremely fast and never stops moving because we need this important and strategic technology. The article is about effects of Information Technology on Business and help of IT to prevent products smuggling, and in this scientific article we tried to introduce big achievements which are ideal ***prevention and security solutions*** that IT delivered recently and it had a very big effect of Business World.

В современном мире, информатизация, как процесс, затрагивает все сферы деятельности государства как единой системы. Особо важно иметь весь срез данных при стратегическом планировании любого уровня, которое является одной из составляющих управленческой деятельности. В сфере большого, среднего и малого бизнеса информационные технологии переводят деятельность предприятий на качественно иной уровень. В статье рассматривается вопросы, связанные с применением информационных технологий для предотвращения проникновения на рынок контрафактной продукции.

## Introduction

The National System of Product and Services Classification and Identification Services could be designed and to be developed identify, classify and codify their products and services.

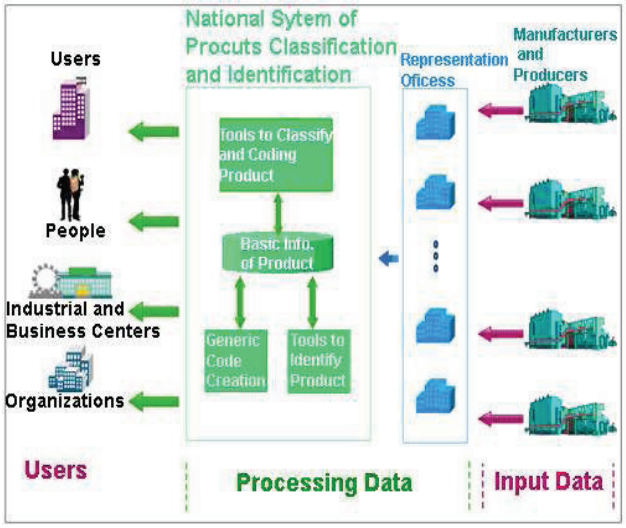



***Figure 1. National System of Products Classification and Identification***

The information of the supplier resources, the base information and technical specifications and price of products, the correspondence with the international coding system, and the base information of items would be generally placed in this national infrastructure and database.

List of Products Can Be Used By National Coding System:

- Food products;
- Living creatures, raw products, animal products;
- Yarn, fibers, wood, paper, cardboard, skin, leather and related products;
- Chemical raw materials, minerals, fuels and oils;
- Road making machinery, construction materials, prefabricated spaces, installations, tapware and piping necessities;
- Furniture, stationery, textile, clothing and services necessities;
- Entertainment, game equipment;
- Cable and rail transportation equipment;
- Air transportation and navigation and airport equipment;
- Marine transporting vehicles and navigation and docks and platforms and pools equipment;
- Medicine, medical items, dental, veterinary, hygiene;
- Telecommunication, electric, electronic and mass media systems equipment parts;
- Road transportation and maintenance equipment;
- Publication products, educational and extra educational equipment;
- Tools;
- Fire fighting equipment;
- Measuring and monitoring and laboratory equipment;
- Agriculture, animal husbandry, horticulture and forestry industries necessities;
- Oil, gas, petrochemical and minerals industries necessities;
- Special Industries necessities;
- Industries general necessities;
- Vehicle Parts.

## Objective

The main objective of this system is to create the common language of product at the national level and developing the comprehensive informational base of companies and products,

hence, distributing information to different users at the national level.

Code is composed of 16 digits from right to left as follows is divided:

- The first four digits indicate the code counter product specifications technical fine products.
- Five-digit code indicates the membership provider is the product that the whole country is unique.
- Seven digits indicate the end product is the technical structure


*Figure 2. International Coding*

- The product platform in the National System of Product Classification (National Structure of Products);
- National Product Code;
- Information of Companies;
- Base information of Products;
- Technical specification information of Products;
- Images of the items;
- Product banding management services;
- Exclusive web pages for each member company;
- Electronic Catalogue of Product;
- Accessing to Products and Manufactures via different routes;
- Reporting based on various information of Products and Companies;
- Accessing to Products and Manufactures through other International Classification Systems;
- Placing the Products in the information basket of user (Product Consumer);
- Placing the Products and Companies in Electronic publications;
- Comparison of Products technical specifications and accessing them.

## Mechanism

Placing the country's manufactures and distributors in this centralized information base will secure many benefits for different elements such as cooperation, government, commercial and industrial centers, and consumers. Utilizing each of these required elements of information can provide immense advantages and benefits for the country. This system has been developed at the Ministry of Commerce and plans to provide comprehensive infrastructure for various users by including the entire manufactured and distributed products in the country and recognizing their founders and proctors.[2,5]

This expert system using the classification and coding of goods, common language and common concepts established at the national level and all credit information such as references pen goods supplier, technical standards and national and international record and ... and will store.

Presenting the base information of a company and its products, hence receiving the National Code for each product, and placing in the stated information source, will collectively position the member company among others whom are in the virtual display of public users.

When an institution becomes a member of the system, in addition to the allocation of an accredited company membership code, the National Code, and the identification for each product, a dedicated web page would be set up for each institute as their Kiosk in the virtual exhibit.

The supplier will be able to present his complete data and product identification and catalogue in this Kiosk.

The project aimed at helping shape the market transparent, smooth and efficient, creating the possibility of direct supply of goods, providing maximum realization of the needs of domestic resources, to create transparency in economic, health and economic problem solving in smuggling runs.

The National Center of Products and Service Numbering as the administrator and proctor of the project implementation is honored to have made the initiative to facilitate the membership of manufactures and the distribution companies at the national level and to engage its capacities and resources in order to apply and accomplish this important task.[1]

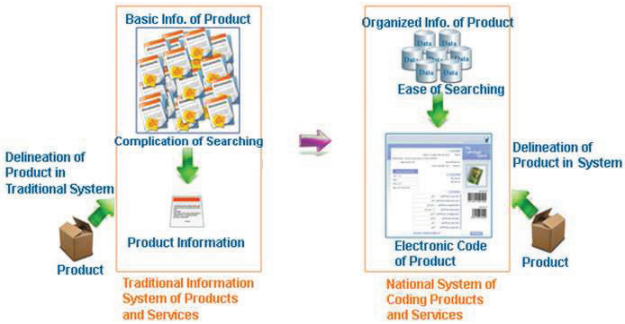



*Figure 3. National System of Coding Products*

## Benefits

Benefits of having a national code:

- Create a common language to facilitate communication and product identification in enterprises and market;
- Reproduction costs, purchasing, transportation, warehousing, marketing, sales;
- Preparing the grounds for the creation of social memory and context information in order to consolidate elements of supply chain management of goods and services;
- The ground presence in the field of e-commerce;
- Introduction and identify sources of manufacturing and distributing goods;
- Creating feature with International Coding System;
- Wide distribution of the ground for producers and distributors;
- Reduce the cost of distribution, sales and operations;
- The ground for introducing and promoting the extensive regional and global goods Organizations to increase productivity;
- To prevent entry without quality goods and deal with smuggling, importers after receiving permission from the New Standards Organization;
- How to monitor implementation of the contract to produce national code;
- Policy, support, scientific and practical implementation of approved national code at the time specified.

Having discussed secure storage for sensitive data, one type of sensitive data deserves special mention. Internet users are paranoid about their credit card numbers. If you are going to store them, you need to be very careful. You also need to ask yourself why you are doing it, and if it is really necessary.

What are you going to do with a card number? If you have a one-off transaction to process and real-time card processing, you will be better off accepting the card number from your customer and sending it straight to your transaction processing gateway without storing it at all. If you have periodic charges to make, such as the authority to charge a monthly fee to the same card for an ongoing subscription, this might not be an option. In this case, you should think about storing the numbers somewhere other than the Web server. If you are going to store large numbers of your customers' card details, make sure that you have a skilled and somewhat paranoid system administrator who has enough time to check up to- date sources of security information for the operating system and other products you use.[3,4]

## Conclusion

Information Technology has the potential to deliver widespread automatic identification of products and suppliers according to the information and technical specifications stored in the national center of products and service numbering database connected to our national coding identification network system. However, when considering whether the national system of product and services classification and identification services could add significant benefit over alternative strategies, the performance of the technology in the field, social and financial factors would need to be examined.

## References

1. Online Banking: What is online banking? 29 March 2004. www.bankrate.com/brm/green/ob/obl.asp.
2. W. F. Tichy, P. Lukowicz, L. Prechelt, & E. A. Heinz."Experimental evaluation in computer science: A quantitative study." Journal of Systems Software, Vol.28, No. 1, 2005, pp. 9-18.
3. Louwers, T. & VanDenburgh, W. (2003). Data confidentiality in an electronic environment. The CPA Journal,73 (3), 24-27.
4. Coppit, D. (2006). Implementing large projects in software engineering courses. Computer Science Education, 16(1), 53-73.
5. Louwers, T. & VanDenburgh, W. (2003). Data confidentiality in an electronic environment. *The CPA Journal, 73* (3), 24-27.

Поступила в редакцию 10.11.2009

УДК 75.004

*ГЕМБА Н.В.,*
*ГЕРАЩЕНКО Д.В.*

# ПРИМЕНЕНИЕ НЕЧЕТКИХ НЕЙРОННЫХ СЕТЕЙ ДЛЯ ПРОГНОЗИРОВАНИЯ ЛИНИЙ ТРЕНДА ФОНДОВОГО РЫНКА

Статья посвящена применению аппарата нечетких нейронных сетей к решению задачи прогнозирования показателей фондового рынка. Приведена структура исследуемой сети и выполнен сравнительный анализ результатов, полученных при использовании различных алгоритмов нечеткого вывода и функций принадлежности. В качестве входов сети используются значения индикаторов технического анализа.

The article focuses on the application apparatus of fuzzy neural networks to solve the problem of forecasting the stock market indices. The comparative analysis performed for various fuzzy logic conclusions and membership functions results. Technical analysis indicators are used as network input values.

## Введение

The article focuses on the application apparatus of fuzzy neural networks to solve the problem of forecasting the stock market indices. The Построение прогноза поведения сложных динамических систем, особенно в экономике и социальной сфере является слабо формализуемой задачей. Традиционные методы анализа рисков фондового рынка, основанные на методах теории вероятности, не являются валидными, поскольку объекты выборки не обладают свойством статистической однородности, а случайные процессы не имеют постоянных параметров. Использование нечетких нейронных сетей для анализа финансовой информации представляется эффективным дополнением традиционных методов исследования, таких как экспертная оценка индикаторов технического анализа.

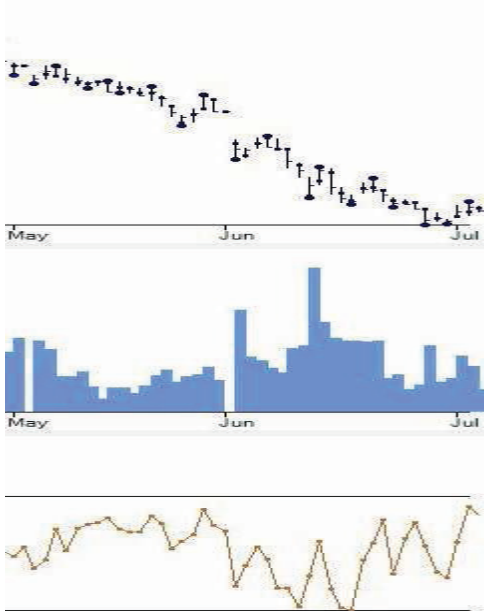
## Подготовка входных данных

Нейросетевое моделирование основано только на данных и не учитывает никаких априорных (экспертных) соображений. При таком подходе имеющихся данных может не хватить для обучения либо потребуется слишком большой число входов сети. Для преодоления данной проблемы предлагается в качестве входов сети использовать не первичные данные, а рассчитанные на их основе индикаторы технического анализа.

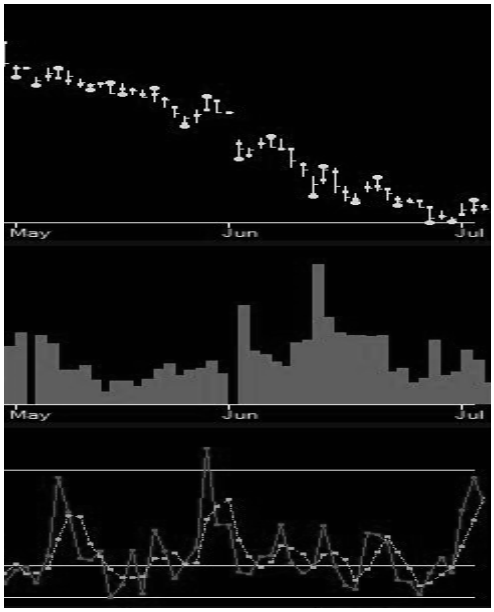
Предметом прогнозирования является направление линии тренда (наличие/отсутствие ее разворотов). Первичными данными для построения прогноза являются значения за последний год, включающие для каждой даты торгов максимальную цену, минимальную цену, цену открытия, цену закрытия, а также объем продаж. В качестве входов сети предложено использовать:

- наклон линии тренда цен,
- индикатор скорости изменения (цены закрытия) (Rate of Change – ROC),
- стохастический осциллятор (Stochastic Oscillator).

Выбор данных индикаторов технического анализа из всего множества существующих [1,2] обусловлен их предсказывающим характером (в отличие от запаздывающего характера большинства индикаторов). Вторым фактором, определяющим выбор множества индикаторов, является минимизация статистической корреляции входных величин. Вид кривых выбранных индикаторов показан на рис.1.




***(а)***




*(б)*

***Рис.1. Индикаторы технического анализа RoC (а) и Stochastic(б)***

### Структура нейронной сети с нечетким логическим выводом

Исследование проводилось на нечетких нейронных сетях ANFIS с логическим выводом по Сугено и Цукамото [3,4]. Количество слоев нейронов и назначение каждого слоя для каждого типа сети является фиксированным, настройка сети происходит за счет изменения параметров отдельных слоев, следовательно, для каждого примера расчета необходимо выбрать тип логического вывода и указать формы функций принадлежности (Гаусса, сигмоид) для каждого из типов логического вывода.

Структура сети приведена на рис.2.

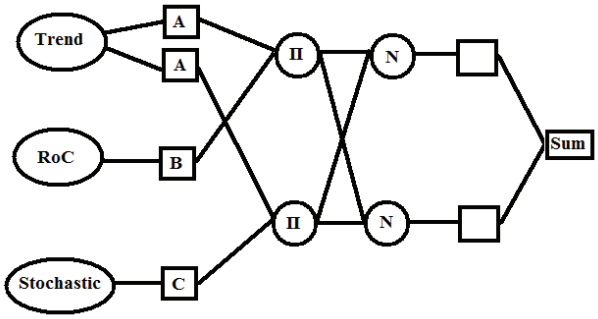



***Рис. 2. Структура нейронной сети с нечетким логическим выводом***

В первом слое (фаззификации) каждый узел является адаптивным узлом с функцией принадлежности заданной формы, осуществляющий преобразование каждого из четких входных значений в степень истинности соответствующей посылки для каждого правила [5]. Данная сеть использует по умолчанию четыре входа: наклон линии тренда и перечисленные выше индикаторы технического анализа (RoC, Stochastic).

Во втором слое (нечетких правил) каждый узел является фиксированным узлом, перемножающим входные сигналы, причем выходное значение узла представляет собой вес некоторого правила. Слой формирует по степени истинности посылок нечеткие подмножества заключений для каждого из правил, т.е. выполняет нечеткий логический вывод.

База правил сети включает в себя правила, прогнозирующие направление линии тренда. Каждое из правил определяет разворот линии тренда вверх либо вниз, либо отсутствие разворота (сохранение направления линии тренда) относительно текущего состояния.

Вербально правила формулируются следующим образом.

Для RoC:

1. Если тренд идет вверх – покупайте всякий раз, когда RoC опускается ниже своей средней линии и поворачивает вверх. Если тренд идет вниз – открывайтесь на короткие продажи всякий раз, когда RoC поднимается выше своей средней линии и поворачивает вниз.

2. Пересечение своей линии тренда моментом или RoC часто опережает перелом тренда рыночных цен на два-три дня.

И значение индикатора вычисляется по следующей формуле:

$$ROC = 100 \times P_0 / P_{-n}$$

где: P0 – цена закрытия текущего периода;
Pn – цена закрытия N периодов назад.

Для стохастического осциллятора в общем случае правило выглядит следующим образом: сигнал на покупку поступает, когда Stochastic Oscillator падает ниже линии 20, а затем проходит эту линию снизу вверх. Сигнал на продажу поступает, когда Stochastic Oscillator поднимается выше линии 80, а затем пробивает эту линию вниз. Но в данной формулировке от индикатора поступает слишком много ложных сигналов, поэтому используется фильтр, при котором формулировка правила имеет следующий вид:

Сигнал на покупку поступает, когда Stochastic Oscillator падает ниже линии 20, а затем проходит эту линию снизу вверх, при

этом %K пересекает %D снизу вверх только после того как %D уже движется вверх. Сигнал на продажу поступает, когда Stochastic Oscillator поднимается выше линии 80, а затем пробивает эту линию вниз, при этом %K пересекает %D сверху вниз только после того как %D уже движется вниз.

Значение индикатора вычисляется по следующей формуле:

$$\%K = 100 \times (C_0 - \min(L_n)) / (\max(H_n) - \min(L_n))$$

где $\max(H_n)$ – максимальный High за n - периодов,

$\min(L_0)$ – минимальный Low за n - периодов,

$C_0$ – цена закрытия текущего периода.

$$\%D = \text{Moving\_Average}_n(\%K)$$

т.е. скользящее среднее с периодом m от %K.

Посылки правил определяются лингвистическими переменными, например:

Up1 – направление тренда цен полого вверх;

Up2 – направление тренда цен средне вверх;

Up3 – направление тренда цен круто вверх;

Аналогично определяются переменные Downx для направление тренда цен вниз;

HiRoc1 – локальный максимум RoC между уровнями 100 и 110;

HiRoc2 – локальный максимум RoC между уровнями 110 и 150;

HiRoc3 – локальный максимум RoC выше уровня 150;

Аналогично для LowRoCx – локальный минимум RoC между уровнями 90 и 100, 50 и 90, ниже уровня 50;

HiStoch – разворот стохастического осциллятора вниз на верхних уровнях;

LoStoch – разворот стохастического осциллятора вверх на нижних уровнях.

Выходная переменная (прогноз разворота) может быть сформулирована как:

ProDown1 = «низкая вероятность разворота рынка вниз»;

ProDown2 = «средняя вероятность разворота рынка вниз»;

ProDown3 = «высокая вероятность разворота рынка вниз».

Аналогично для вероятности разворота тренда вверх (ProUpx).

Тогда правила могут быть сформулированы следующим образом:

1. IF Up[i] AND HiRoc[i] THEN ProDown[i], где i = 1,2,3;
2. IF Down[i] AND LowRoc[i] THEN ProUp[i], где i = 1,2,3;
3. IF Up[i] AND HiStoch THEN ProDown[i], где i = 1,2,3;
4. IF Down[i] AND LowStoch THEN ProUp[i], где i = 1,2,3.

Каждый узел третьего слоя (агрегирования нечетких правил) определяет отношение веса одного правила к сумме весов всех правил.

Узлы четвертого слоя определяются функциями принадлежности выходных переменных (линейных для модели Сугено).

Единственный узел пятого слоя (дефазификации) является фиксированным узлом, в котором вычисляется четкое выходное значение сети из нечетких подмножеств.

## Обучение нейронной сети

Для обучения сети применяется гибридный алгоритм, включающий прямой и обратный проходы [5,6]. При прямом проходе выполняется расчет выхода сети по заданным входам. На основе известного значения выхода сети выполняется расчет коэффициентов четвертого слоя методом наименьших квадратов. При обратном проходе методом градиентного спуска определяются параметры первого слоя.

## Результаты эксперимента

Для сравнения эффективности прогнозирования были проведены вычисления на данных для семи компаний [7]. Полученные результаты (процент правильных прогнозов) приведены в таблице 1.

## Выводы

Проведенные исследования показали, что применение нечетких нейронных сетей позволяет построить более точный прогноз по сравнению со статистическими методами, но, по-прежнему остается достаточно высокий процент ложных прогнозов (как не прогнозированных разворотов, так и отсутствие разворотов при их прогнозе). В качестве направлений дальнейших исследований планируется доработка структуры сети с использованием в первом слое различных функций принадлежности, увеличение входов сети за счет учета объемов продаж, а также повышение валидности исходных данных на основе автоматизированного анализа новостных блоков.

***Таблица 1. Результаты (процент правильных прогнозов)***

| Данные | Нейронная сеть с нечетким выводом по Цукамото | | Нейронная сеть с нечетким выводом по Сугено | |
|---|---|---|---|---|
| | ФП Гаусса | ФП сигмоид | ФП Гаусса | ФП сигмоид |
| Sun Microsystems, Inc. (Public, NASDAQ:JAVA) | 85.3 | 87.1 | 80 | 83.2 |
| Oracle Corporation (Public, NASDAQ:ORCL) | 86.1 | 88 | 81.9 | 85.7 |
| International Business Machines Corp. (Public, NYSE:IBM) | 85.6 | 87.4 | 81.4 | 85.5 |
| Chevron Corporation (Public, NYSE:CVX) | 83.2 | 86.7 | 79.9 | 81.1 |
| BP plc (ADR) (Public, NYSE:BP) | 84.5 | 86 | 79.1 | 82.4 |
| Royal Dutch Shell plc (ADR) (Public, NYSE:RDS.A) | 84.9 | 87.2 | 80.2 | 82.7 |
| Gazprom OAO (ADR) (Public, OTC:OGZPY) | 82 | 84.4 | 78.9 | 81.3 |

## Список литературы


1. Колби Р. Энциклопедия технических индикаторов рынка [Текст] / Р. Колби – М.: Финансы и статистика, 2007. – 838 с.
2. Курс технического анализа [Электронный ресурс]. – Режим доступа: http://www.parusinvestora.ru/systems/book_meladze
3. Сравнительный анализ нечетких нейронных сетей с различными алгоритмами вывода в задачах прогнозирования курса акций [Электронный ресурс]. – Режим доступа: http://efunds.com.ua/blog/isc/122.html
4. Моделирование технологического процесса с использованием адаптивных нечетких нейронных сетей. [Электронный ресурс]. – Режим доступа: http://www.nbuv.gov.ua/portal/natural/Tp/2008_2/G7.htm
5. Рутковская Д., Пилиньский М., Рутковский Л. Нейронные сети, генетические алгоритмы и нечеткие системы. [Текст] / Д. Рутковская, М. Пилиньский , Л. Рутковский – М.: Финансы и статистика, 2004. – 383 с.
6. Самоорганизация (самообучение) нейронных сетей [Электронный ресурс]. – Режим доступа: http://www.intuit.ru/department/expert/neuro/13/3.html#sect3
7. Котировки акций по NYSE [Электронный ресурс]. – Режим доступа: http://www.google.com/finance


Поступила в редакцию 15.12.2009