УДК 519.854.2

ПАВЛОВ О.А.,
МІСЮРА О.Б.,
МЕЛЬНИКОВ О.В.

ДОСЛІДЖЕННЯ ВЛАСТИВОСТЕЙ ТА РОЗВ'ЯЗАННЯ ЗАДАЧІ «МІНІМІЗАЦІЯ СУМАРНОГО ШТРАФУ ЯК ЗА ВИПЕРЕДЖЕННЯ, ТАК І ЗА ЗАПІЗНЕННЯ ВІДНОСНО ДИРЕКТИВНИХ СТРОКІВ ПРИ ВИКОНАННІ НЕЗАЛЕЖНИХ ЗАВДАНЬ ОДНИМ ПРИЛАДОМ»

Розглядається задача складання розкладів за критерієм мінімізації сумарного випередження і запізнення при виконанні завдань одним приладом (МВЗ). Ця задача NP-трудная, тому підходи локального пошуку дуже корисні для знаходження хороших близьких до оптимальних розкладів. Представлено деякі нові властивості задачі, які дозволяють нам побудувати ефективну процедуру пошуку розв'язків, близьких до оптимальних. Наведено приклад розв'язання задачі.

One machine scheduling problem to minimize total earliness and tardiness (E/T) is considered. This problem is NP-hard, therefore local search approaches are very useful to find good schedules that are close to optimum. Some new properties of a problem are presented which allow us to construct an effective procedure of searching the solutions that are close to optimum. An example of the problem solution is shown.

Вступ

Складання розкладів відносно директивних строків одержало значну увагу в літературі. Одна з причин цього явища – зростаючий тиск конкуренції на міжнародних ринках: фірми повинні запропонувати велике розмаїття різних і індивідуальних виробів, у той час як клієнти очікують, що замовлені товари будуть поставлені вчасно. Щоб задовольнити ці види вимог, були розроблені такі принципи і філософії, як управління відхиленням, одночасна розробка, виробництво «точно в строк» і т.д. Наприклад, принцип «точно в строк» встановлює, що необхідна кількість товарів повинна бути вироблена та поставлена точно в необхідний час. Таким чином, очікується, що роботи будуть своєчасними, тому що пізні постачання, так само, як і ранні, розглядаються як небажані [1, 2, 3, 4].

У той час як випередження робіт зв'язано з витратами на складування, запізнення робіт зв'язано зі штрафами, такими, як утрата доброзичливості клієнта і втрата замовлень. Тому мінімізація випередження і запізнення (В/З) дуже важлива в ринковій економіці.

Концепція визначення штрафів як за випередження, так і за запізнення, нова і має потребу в додатковому дослідженні. Більшість досліджень по задачах В/З зв'язані з моделями для одного приладу, хоча деякі результати для одного приладу були розширені на задачі для паралельних приладів.

Загальна модель задачі В/З:

1) Задачі В/З мають справу зі статичним складанням розкладів (множина робіт, що будуть призначені, відома заздалегідь і доступна в момент часу нуль).
2) Кількісний критерій звичайно інтерпретується як мінімізація сумарного штрафу за випередження і запізнення.
3) Штрафи можуть бути задані різними способами.
4) Маємо n завдань, що повинні бути призначені: $j = 1, 2, \ldots, n$.
5) Випередження і запізнення визначаються як, відповідно:

$$E_j = \max(0, d_j - C_j) = (d_j - C_j)^+ \qquad (1)$$
$$T_j = \max(0, C_j - d_j) = (C_j - d_j)^+, \qquad (2)$$

де C_j – момент завершення завдання j, d_j – його директивний строк.
6) Основна цільова функція В/З для розкладу S може бути записана як:

$$f(s) = \sum_{i=1}^{n} [\, \alpha_j (d_j - C_j)^+ + \beta_j (C_j - d_j)^+ \,] \qquad (3)$$

або

$$f(s) = \sum_{i=1}^{n} (\, \alpha_j E_j + \beta_j T_j \,), \qquad (4)$$

де $\alpha_j > 0$ і $\beta_j > 0$ – штрафи за випередження і запізнення завдання j.
7) Деякі з найпростіших результатів для задач В/З були отримані для моделей, у яких усі завдання мають загальний директивний строк ($d_j = d$).

У статті ми розглядаємо задачу складання розкладів за критерієм мінімізації сумарного

випередження і запізнення при виконанні завдань одним приладом (МВЗ), яка може бути сформульована таким чином.

Постановка задачі

Множина з n незалежних завдань $J=\{j_1, j_2,..., j_n\}$ повинна бути призначена без переривань на одному приладі, що може працювати не більш, ніж з одним завданням одночасно. Прилад і завдання передбачаються безупинно доступними з моменту часу нуль, а простої приладу не допускаються. Завдання j, де j = 1, 2, ..., n, вимагає часу обробки p_j і повинне в ідеалі бути закінчене у свій директивний строк d_j. Для будь-якого заданого розкладу випередження і запізнення завдання j можуть бути визначені виразами (1) і (2). Ціль полягає в тому, щоб знайти розклад, що мінімізує сумарне випередження і запізнення всіх завдань:

$$\sum_{j=1}^{n}\left(E_j + T_j\right). \qquad (5)$$

Використовуючи систему позначень Лоулера та ін. [5], статична задача складання розкладів МВЗ може бути описана як $1||\Sigma(E_j + T_j)$. Бейкер і Скуддер [6] приводять огляд відкритої літератури по задачі складання розкладів В/З, описуючи різні моделі В/З и різні штрафи за В/З завдань. Коли директивні строки завдань визначені, але різні, задачі В/З є NP-повними [7]. Яно та Кім [8] розробили евристику для часткового випадку зважених штрафів, у якому вага пропорційна тривалостям завдань. Вони використовують процедуру попарної перестановки, щоб покращити початкові евристичні розв'язки.

Оу і Мортон [9] досліджують функцію на основі пріоритету і різновид фільтрованого променевого пошуку для задач В/З, укладаючи висновок, що якість розв'язку значно погіршується, якщо витрати на випередження завдань ігноруються на користь розгляду тільки вартості запізнення. Планувальник Дэвиса і Кейнта [10] може бути вбудований в існуючий евристичний розв'язок для покращення якості розкладу МВЗ при низьких обчислювальних витратах. Шварц і Мухопад'яй [11] розробили підхід угруповання (кластерізації) для визначення оптимального часу запуску для завдань на прикладі задачі МВЗ за припущенням, що послідовність завдань відома апріорно. Підхід угруповання здійснює оптимальний вибір часу для 500 завдань за кілька секунд обчислювального часу. Срідхаран і Жоу [12] використовують евристику диспетчування для задач МВЗ, що допускає вставлений резерв приладу, якщо він адекватний.

Наш підхід відрізняється від приведеного у роботі [11] тим, що послідовність завдань не повинна бути визначена апріорно. Експериментальні результати показали отримання ефективних розкладів за короткий час обчислень.

Алгоритм розв'язання задачі «Мінімізація сумарного випередження/запізнення відносно директивних строків при виконанні незалежних завдань одним приладом» (МВЗ)

Алгоритм базується на наступних твердженнях.

Позначимо $r_j = \max(0, d_j - C_j)$ – резерв завдання j. Очевидно, $r_j = E_j$.

***Твердження* 1.** Якщо в послідовності σ завдання упорядковані за незростанням значень тривалості, і для всіх завдань $j\in\sigma$ виконується $r_j \geq 0$, то ця послідовність оптимальна згідно функціоналу В/З.

Позначимо $r_{\min} = \min\{r_j, p_j\}$; N_r – число завдань з резервами ($r_j > 0$); $N_з$ – число запізнілих завдань (у число запізнілих включаються завдання з нульовим резервом).

***Твердження* 2.** Якщо в послідовності σ $N_r>N_з$, то при збільшенні початку виконання завдань на величину, що дорівнює $r_{\min}$, значення функціоналу В/З зменшується також на $r_{\min}$.

Алгоритм полягає в послідовному збільшенні моментів початку виконання завдань початкової послідовності $\sigma^{уп}$ (упорядкованої за неспаданням тривалостей виконання завдань), а також усіх її модифікацій з різними моментами початку виконання завдань, і наступної оптимізації за критерієм В/З. З цією метою послідовність $\sigma^{уп}$ оптимізується за критерієм мінімізації сумарного запізнювання (МСЗ). Якщо в отриманій оптимальній послідовності $N_r > N_з$, момент початку виконання завдань послідовності збільшується. Така процедура виконується доти, поки виконується умова $N_r > N_з$, і якщо в отриманій послідовності σ^R є резерви, то до неї застосовується алгоритм оптимізації за критерієм В/З.

***Твердження* 3.** Якщо в послідовності σ^R резерви відсутні, то вона оптимальна за критерієм В/З.

Таким чином, алгоритм розв'язання задачі складається з наступних блоків:

Блок 1:

- Оптимізація поточної послідовності σ^k по алгоритму розв'язання задачі МСЗ, наведеному в [13] (отримуємо послідовність σ^0). В результаті виконання алгоритму зменшення запізнення завдань здійснюється за рахунок резервів незапізнених завдань.
- Збільшення моменту початку виконання всіх завдань при умові, якщо Nr > Nз, де Nr – число завдань з резервами, Nз – число завдань, що запізнюються або виконуються точно в строк. В результаті виконання цієї процедури зменшується значення функціоналу (В/З).
- Блок 1 виконується доти, поки Nr > Nз. В результаті отримуємо послідовність σ^R.

Блок 2 – оптимізація послідовності σ^R по критерію В/З.

Опис алгоритму

Перед початком алгоритму оптимізуємо послідовність σ за алгоритмом розв'язання задачі МСЗ, наведеному в [13] (отримуємо послідовність σ^*_0).

Алгоритм складається з k однотипних ітерацій, на кожній ітерації збільшується початок виконання завдань послідовності, що розглядається, якщо це призводить до зменшення значення функціоналу. Нехай вже виконана k–1 ітерація. Розглядаємо послідовність σ^{k-1}.

Блок 1.

1) Аналізуємо послідовність σ^*_{k-1}. Визначаємо N_r – число завдань з резервами, та $N_з$ – число завдань, що запізнюються або виконуються точно в строк. Перевіряємо умову $N_r > N_з$.

2) Якщо $N_r < N_з$, то $\sigma^R = \sigma^*_{k-1}$, переходимо на шаг 6.

3) Якщо $N_r > N_з$, вибираємо завдання з мінімальним резервом $r_{\min} > 0$ із послідовності σ^*_{k-1}. На величину $r_{\min}$ збільшуємо початок виконання всіх завдань послідовності σ^*_{k-1}: $R(\sigma^k) = R(\sigma_{опт}^{k-1}) + r_{\min}$. $k = k + 1$. Переходимо на шаг 1, початок наступної ітерації.

4) Якщо в послідовності σ^*_{k-1} $N_r = N_з$, вибираємо завдання з мінімальним резервом $r_{\min} > 0$ із послідовності σ^*_{k-1}. На величину $r_{\min}$ збільшуємо початок виконання всіх завдань послідовності σ^*_{k-1}: $R(\sigma^k) = R(\sigma_{опт}^{k-1}) + r_{\min}$. Оптимізуємо послідовність по алгоритму розв'язання задачі МСЗ. Якщо в послідовності σ^k $N_r = N_з$, то $\sigma^R = \sigma^k$, переходимо на шаг 6. Якщо ж $N_r < N_з$, то порівнюємо значення функціоналу послідовностей σ^k та σ^*_{k-1}, вибираємо в якості σ^R послідовність з меншим значенням функціоналу та переходимо на шаг 6.

5) Оптимізація послідовності σ^R по алгоритму блоку 2.

Блок 2.

Алгоритм складається з k однотипних ітерацій, на кожній ітерації перевіряється можливість зменшення значення функціонала за рахунок вставки завдань з більшою тривалістю на більш ранню позицію відповідно до твердження 1. Нехай уже виконана k–1 ітерація і цим алгоритмом вже упорядкована підпослідовність на інтервалі [1, k-1]. Розглянемо можливість завдання $j_{[k]}$ на більш ранню позицію.

1) Розглядаємо послідовність σ^{k-1} з початку. Знаходимо найбільш ранню позицію p, для якої виконуються умови: $l_{j_{[p-1]}} \geq l_{j_{[k]}}$, а $l_{j_{[p+1]}} < l_{j_{[k]}}$.

2) Якщо на інтервалі [p, k] є завдання $j_{[i]}$, для якого $l_{j_{[i]}} \geq l_{j_{[k]}}$ і $d_{j_{[i]}} \geq d_{j_{[k]}}$, то інтервал вставки визначається позиціями [i+1, k], інакше позиціями [p, k].

3) Перевіряємо, чи є резерви на інтервалі вставки. Якщо ні, то переходимо на шаг 7, інакше на шаг 4.

4) Завдання $j_{[k]}$ переносимо на позицію, визначену інтервалом вставки (i+1 або p).

5) Виконуємо вільні перестановки (див. [13]), що приводять до зменшення значення функціонала.

6) На інтервалі вставки знаходимо позицію, на якій значення функціонала мінімальне:

$$\left(\left|D_{j_{[k]}} - C_{j_{[k]}}\right| + \sum_{i=k}^{p} l_{j_{[i]}}\right) + \sum_{i=k}^{p}\left(\left|D_{j_{[i]}} - C_{j_{[i]}}\right| - l_{j_{[k]}}\right) \to \min \qquad (7)$$

і переносимо завдання з позиції p на цю позицію.

7) $k = k + 1$. Якщо $k > n$, кінець роботи алгоритму. Інакше перехід на шаг 1

Приведемо приклад роботи алгоритму.

Таблиця 1 – Вхідні дані ($\sigma^{уп}$) для прикладу розв'язання задачі МВЗ

j	l_j	d_j	C_j	$\lvert d_j – C_j \rvert$
1	121	528	121	407
2	147	537	268	269
3	102	668	370	298
4	79	534	449	85
5	130	605	579	26
6	83	604	662	58
7	96	951	758	193
8	88	987	846	141

Таблиця 2 – Послідовність $\sigma^*_6 = \sigma^R$, що одержана в результаті виконання блоків 1,2

j	l_j	d_j	C_j	$\lvert d_j – C_j \rvert$
4	79	534	253	281
1	121	528	374	154
2	147	537	521	16
6	83	604	604	0
3	102	668	706	38
5	130	605	836	231
7	96	951	932	19
8	88	987	1020	33

Таблиця 3 – Оптимальна послідовність $\sigma^{опт}$

j	l_j	d_j	C_j	$\lvert d_j – C_j \rvert$
2	147	537	321	216
1	121	528	442	86
4	79	534	521	13
6	83	604	604	0
3	102	668	706	38
5	130	605	836	231
7	96	951	932	19
8	88	987	1020	33

Заключення

Ми застосували наступні еволюційні стратегії для розв'язання задачі МВЗ: мінімізація сумарного запізнення за рахунок резервів незапізнених завдань; визначення оптимального моменту запуску на виконання; оптимізація за рахунок вставки завдань з більшою тривалістю на більш ранні позиції. Показано, що застосування цих стратегій є ефективним для знаходження розкладів, близьких до оптимальних.

Перелік посилань

1. Feldmann M., Biskup D. Single-machine scheduling for minimizing earliness and tardiness penalties by meta-heuristic approaches // Computers & Industrial Engineering.– 2003.– №44.– P.307-323
2. Tötüncüoglu R.A. Sequencing with earliness and tardiness penalties.– Department of Industrial Engineering, Bilkent University, 1999.– 14 p.
3. Valente J. M. S., Alves R. A. F. S. Improved heuristics for the early/tardy scheduling problem with no idle time // Computers & Operations Research.– 2005.– №32.– P.557-569
4. Jin S., Mason S.J. Minimizing earliness and tardiness costs on a single machine with uncommon job due dates // Department of Industrial Engineering, Bell Engineering Center, University of Arkansas Fayetteville, 2004.– 23 p.
5. Lawler, E.L. A fully polynomial approximation scheme for the total tardiness problem // Operations Research Letters.– 1982.– №1.– P.207-208
6. Baker K.R., Scudder G.D. Sequencing with earliness and tardiness penalties: a review // Operations Research.– 1990.– Vol.1.– №38.– P.22-36
7. Garey, M.R., Tarjan, R.E., Wilfong, G.T. One-processor scheduling with symmetric earliness and tardiness penalties // Mathematics of Operations Research.– 1988.– №13.– P.330-348
8. Yano C.A., Kim Y.D. Algorithms for a class of single-machine weighted tardiness and earliness problems // European Journal of Operations Research.– 1991.– №52.– P.167-178
9. Ow P.S., Morton T.E. The single machine early/tardy problem // Management Science.– 1989.– Vol.2.– №35.– P.177-191
10. Davis J.S., Kanet J.J. Single-machine scheduling with early and tardy completion costs // Naval Research Logistics.– 1996.– №40.– P.85-101
11. Szwarc W., Mukhopadhyay S.K. Optimal timing scheduling in earliness-tardiness single machine sequencing // Naval Research Logistics.– 1995.– №42.– P.1109-1114
12. Sridharan V., Zhou, Z. A decision theory based scheduling procedure for single-machine weighted earliness and tardiness problem // European Journal of Operations Research.– 1996.– №94.– P.292-301
13. Павлов А.А., Мисюра Е.Б. Эффективный точный ПДС-алгоритм решения задачи о суммарном запаздывании для одного прибора // Системні дослідження та інформаційні технології, 2004, №4, С.30-59

УДК 004.94(075.8)

ПАРАМОНОВ А.М.,
ТОМАШЕВСКИЙ В.Н.

ФОРМАЛИЗАЦИЯ АЛГОРИТМА МОДЕЛИРОВАНИЯ ДВИЖЕНИЯ АВТОМОБИЛЬНОГО ДОРОЖНОГО ТРАНСПОРТА

Запропонована формалізація алгоритму моделювання руху автомобільного дорожнього транспорту, зокрема, увага приділяється формалізації алгоритмів обгону та перестроєння із використанням мереж Петрі. Наводиться мережа Петрі, яка описує стани автомобільного засобу, переходи між станами, та описує умови, які повинні виконуватися для спрацьовування переходів.

The formalization of simulation algorithm for traffic control of vehicles is offered, specifically, the formalization of overtaking and re-formation algorithms are given consideration using Petri's network. The transition and nodes of Petri net for vehicle and conditions by fulfillment which it moves from one state to another are described.

Постановка проблемы в общем виде и ее связь с важными научными или практическими задачами

В Украине с увеличением количества машин на дорогах во многих городах возникла транспортная проблема. Для принятия решений, улучшающих сложившуюся транспортную ситуацию, необходимо разработать систему управления транспортными потоками, составной частью которой должна быть имитационная модель для моделирования поведения транспорта. С научной точки зрения важность разработки данной системы состоит в разработке оптимальных алгоритмов моделирования, с практической – улучшение ситуации на дорогах и решение транспортной проблемы.

Одним из этапов создания ядра системы моделирования автомобильного дорожного движения, является разработка алгоритма передвижения автомобиля по дороге с учетом дорожной ситуации. Данная статья предлагает формализацию данного алгоритма при помощи сетей Петри. Сети Петри являются одной из наиболее удобных математических конструкций представления сложных моделей событий. С помощью сетей Петри достаточно легко задать модель практически любой реально существующей системы, что способствует ее широкому распространению. Высокая формализация сетей Петри позволяет легко строить программные алгоритмы для их обработки. Сети Петри – это удобный инструмент для анализа и описания асинхронных, параллельных и распределенных систем. Как математический инструмент их можно использовать для описания уравнений состояний, алгебраических уравнений и других математических моделей, которые управляют поведением системы. Кроме того, сети Петри удобно использовать для анализа и автоматического построения программ имитационных моделей.

Формально сеть определяется тройкой [1]:

$$G = \langle P, T, F \rangle, \qquad (1)$$

где:

- P – не пустое множество элементов сети, называемых состояниями;
- T – не пустое множество элементов сети, называемых переходами;
- F – функция инцидентности, которая задает связь между элементами P и T.
- Для сети G должны выполняться условия:
- $P \cap T = \varnothing$, то есть множества состояний и переходов не пересекаются;
- $(F \neq \varnothing) \wedge (\forall x \in P \cup T, \exists y \in P \cup T : xFy \vee yFx)$, то есть любой элемент сети инцидентный по крайней мере одному элементу другого типа;
- если для любого элемента сети $x \in X$ (X – множество всех элементов сети) обозначить через $*x$ множество входящих в него элементов, а через $x*$ – множество выходящих из него элементов, то:

$$\forall p1, p2 \in P : (*p1 = *p2) \wedge (p1* = p2*) \Rightarrow (p1 = p2), \qquad (2)$$

то есть сеть не содержит таких пар состояний, которые инцидентны тому же самому множеству переходов.

На основании понятия сети, которая описывает только статическую топологию моделируемого процесса или системы, вводятся динамические сетевые структуры, в которых событиям приписывают специальные размет-

ки для моделирования выполнения условий. Функционирование сети – это смена разметок (условий) в результате переключения переходов. К таким динамическим сетям относятся сети Петри. Сеть Петри не является динамической моделью, в которой состояние модели изменяется во времени, а оно изменяется только при срабатывании переходов и смене разметки.

Сеть Петри – это набор [1]:

$$G^p = < G, M^0 >, \qquad (3)$$

где G – конечная сеть (множества P и T конечны); M^0 – функция начальной разметки сети, которая ставит в соответствие любому состоянию pi ∈ P некоторое число $M^0(p) = n$.

Функционирование сети Петри описывается формально с помощью множества последовательностей срабатываний и множества досягаемых в сети разметок. Эти понятия определяются через правила переключения переходов в сети.

Разметка сети G_p является функцией M: $P \Rightarrow N$ (N – множество натуральных чисел). Если допустить, что все состояния сети строго упорядочены некоторым образом, то есть $P=(p_1, p_2, ...p_n)$, то разметку M сети (в том числе и начальную разметку) можно задать как вектор чисел $M=(m_1, m_2, ..., m_n)$ такой, что для любого i, $(1 \le i \le n)$:

$$m_i = M(p_i), \qquad (4)$$

Анализ последних исследований и публикаций, в которых отражено начало процесса решения данной проблемы

В связи с интенсивным развитием автомобильного движения в нашей стране возникает задача увеличения пропускной способности дорог. Такое увеличение может быть достигнуто благодаря увеличению полос для движения (расширение дороги) или за счет увеличения средней скорости движения на участке дороги. Второй способ связан с анализом дорожного движения на некотором участке дороги и постепенным устранением «узких мест», которые влияют на снижение средней скорости движения. Основой системы моделирования автомобильного дорожного движения (СМАДД) является математическая модель для прогнозирования интенсивностей транспортных потоков в городских сетях с использованием систем массового обслуживания. СМАДД предназначена для определения оптимальных маршрутов движения, нахождения объездных путей в случае возникновения аварийных ситуаций на маршруте следования или наличии определенных ограничений для проезда транспорта, расчета пропускной способности дорог с целью обнаружения и устранения узких мест [2].

Выделение нерешенных ранее частей общей проблемы, которой посвящена статья

Существует множество алгоритмов моделирования движения автомобильного дорожного транспорта. Практически все они описаны в виде математических моделей, в которых присутствуют множество параметров, систем уравнений. Однако, главным преимуществом в формализации алгоритма движения при помощи сетей Петри является удобство их использования для анализа и автоматического построения программ имитационных моделей. Также, сети Петри наглядно представляют систему моделирования, что в свою очередь облегчает восприятие всех процессов, протекающих внутри моделируемой системы.

Постановка задачи

Формализация алгоритма моделирования движения автомобильного дорожного транспорта при помощи сетей Петри. Алгоритм движения автомобилей включает в себя:

- алгоритм обычного движения;
- алгоритм выполнения маневра перестроения;
- алгоритм выполнения маневра обгона.

Изложение основного материала исследования с полным обоснованием полученных научных результатов

Проведем формализацию алгоритма передвижения автомобильного дорожного транспорта с применением сетей Петри. Следует начать с определения ситуаций, которые происходят с автомобилем на дороге. Автомобиль может совершать разные маневры на дороге:

- изменение полосы следования автомобиля (перестроение);
- совершение маневра обгона другого транспортного средства;
- обычное движение по своей полосе.

На рисунке 1 представлена сеть Петри для этих вариантов:

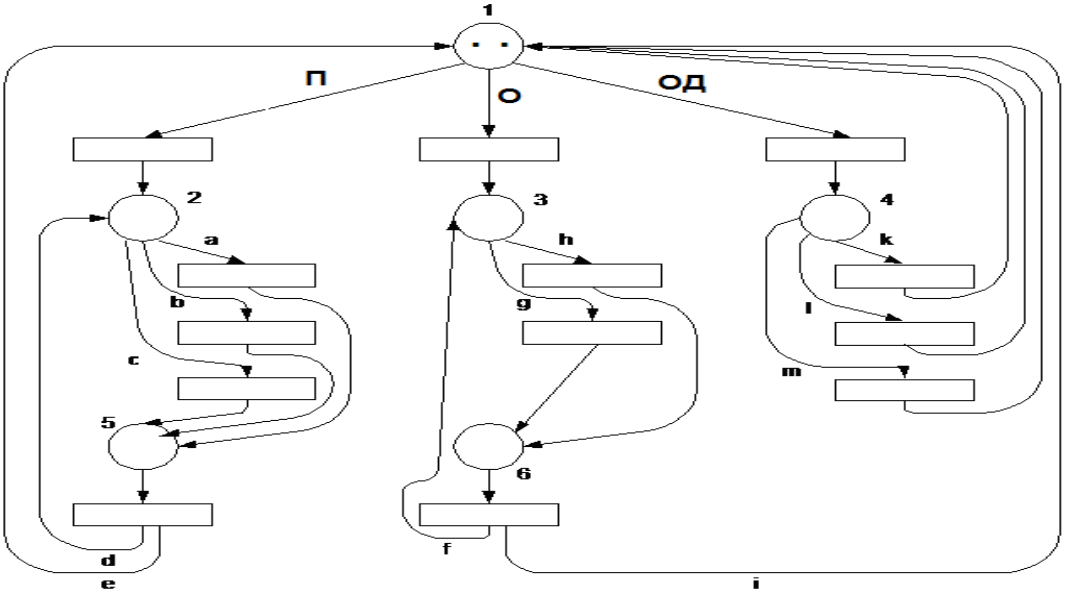


Рис. 1. Представление возможных состояний автомобиля

Опишем согласно формуле (1) множество состояний P={1,2,3,4,5,6} нашей сети (см. рис. 1):

1 – это начальное состояние – в этом состоянии сеть находится, когда автомобиль начинает обычное движение;

2 – это начало выполнения маневра перестроения с одной полосы на другую;

3 – это начало выполнения маневра обгона;

4 – это продолжение движения без выполнения каких-либо маневров;

5 – это завершение выполнения маневра перестроения с одной полосы на другую;

6 – это завершение выполнения маневра обгона.

Множество переходов T={П,О,ОД,a,b,c,d,e, f,g,h,i,k,l,m}, где *П* – перестроение, *О* – обгон, *ОД* – обычное движение. Остальные переходы опишем при рассмотрении конкретно ситуаций перестроения, обгона и обычного движения.

Начальная разметка для данной сети M^0=[2,0,0,0,0,0]. На этом этапе автомобиль может выбрать одно из трех состояний: совершение маневра перестроения, обгона или продолжать обычное движение. Рассмотрим все возможные переходы системы:

1) Активизация перехода *П*. В этом случае мы имеем новую разметку M^1=[1,1,0,0,0,0], которая представлена на рис. 2:

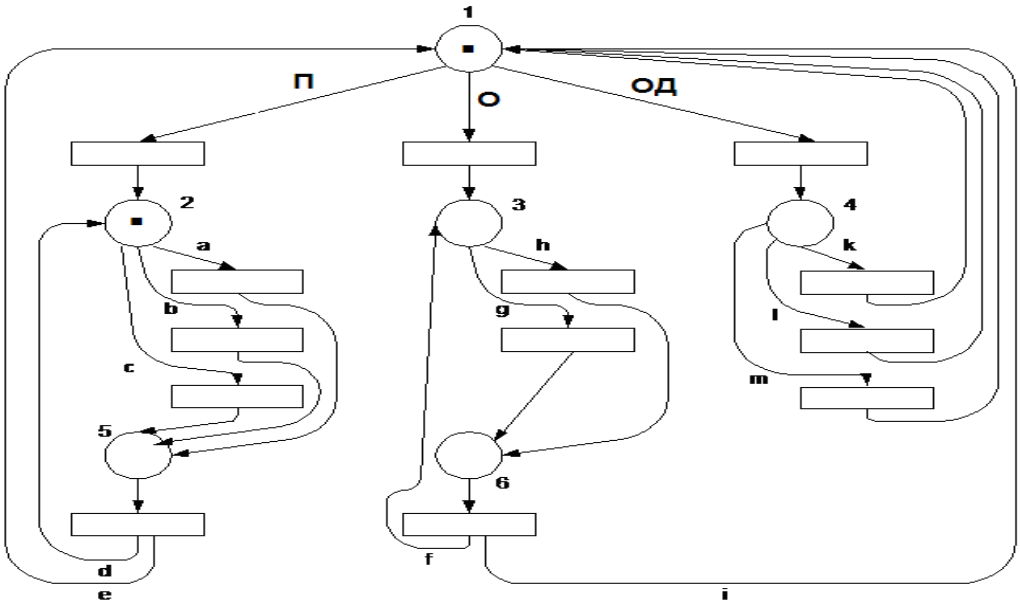


Рис. 2. Разметка M^1

Теперь могут быть активизированы переходы a,b,c, которые связаны с ускорением автомобиля. Переход a активизирует процесс разгона, b – процесс торможения, c – движе-

ние с постоянной скоростью (без ускорения). После срабатывания любого из переходов a,b,c имеем разметку M^2=[1,0,0,0,1,0], которая представлена на рис. 3:

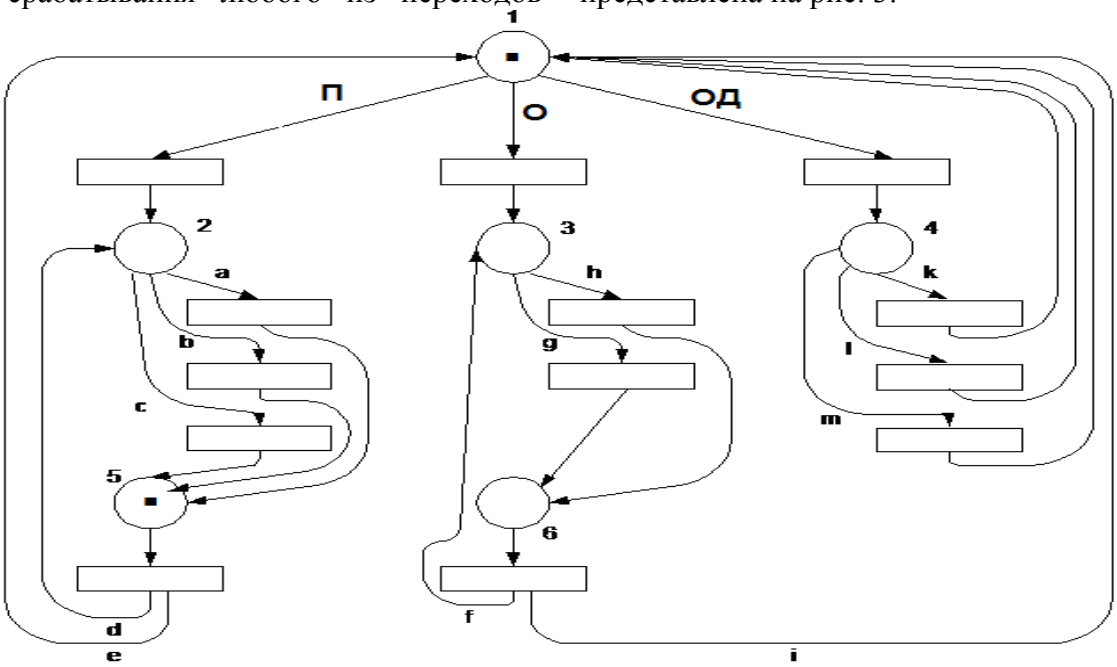


Рис. 3. Разметка M^2

Теперь активными являются переходы d – продолжение процесса перестроения, и e – переход в начальное состояние. При этом мы получаем в результате срабатывания перехода d разметку M^1=[1,1,0,0,0,0] (см. рис. 2), а в результате срабатывания перехода e – разметку M^0 = [2,0,0,0,0,0] (см. рис. 1).

2) Активизация перехода *О*. В этом случае мы имеем новую разметку M^3=[1,0,1,0,0,0], которая представлена на рис. 4:

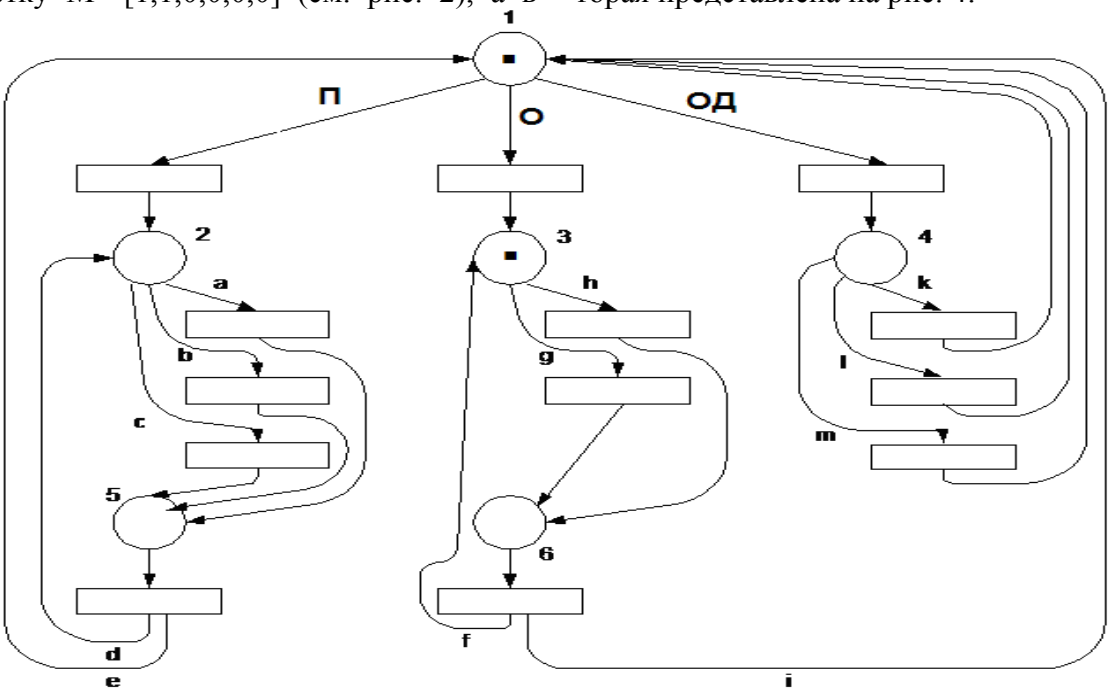


Рис. 4. Разметка M^3

Из полученного состояния быть активизированы переходы g,h, которые связаны с ускорением автомобиля. Переход g активизирует процесс разгона, h – движение с постоянной скоростью (без ускорения). При совершении обгона, разумеется, не возможно состояние, соответствующее процессу торможения автомобиля. После срабатывания любого из переходов g,h имеем разметку M^4=[1,0,0,0,0,1], которая представлена на рис. 5:

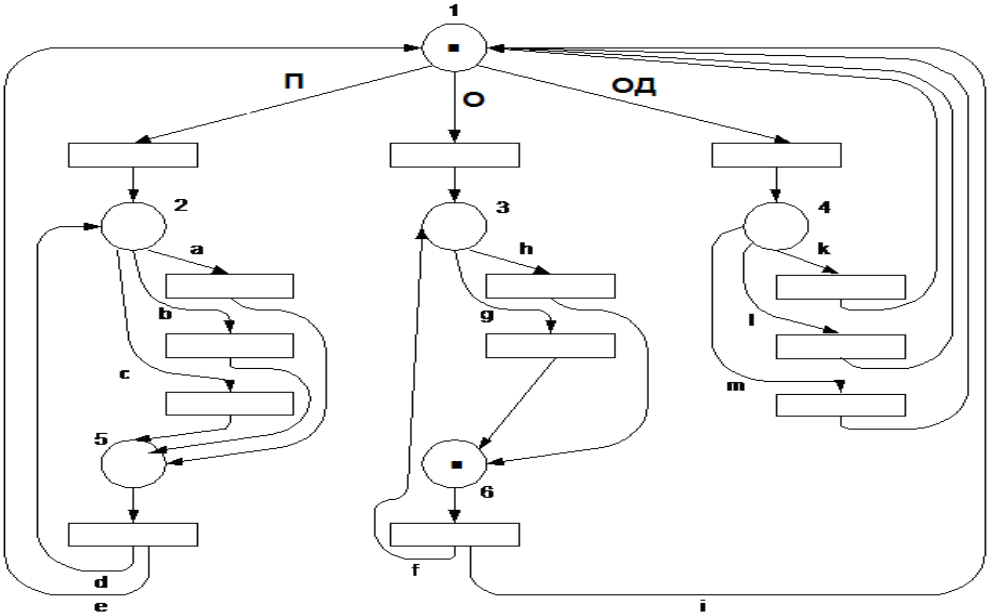


Рис. 5. Разметка M^4

Теперь активными являются переходы f – продолжение процесса обгона, и i – переход в начальное состояние. При этом мы получаем в результате срабатывания перехода f разметку M^3=[1,0,1,0,0,0] (см. рис. 4), а в результате срабатывания перехода i – разметку M^0=[2,0,0,0,0,0] (см. рис. 1).

3) Активизация перехода *ОД*. В этом случае мы имеем новую разметку M^5 = [1,0,0,1,0,0], которая представлена на рис. 6:

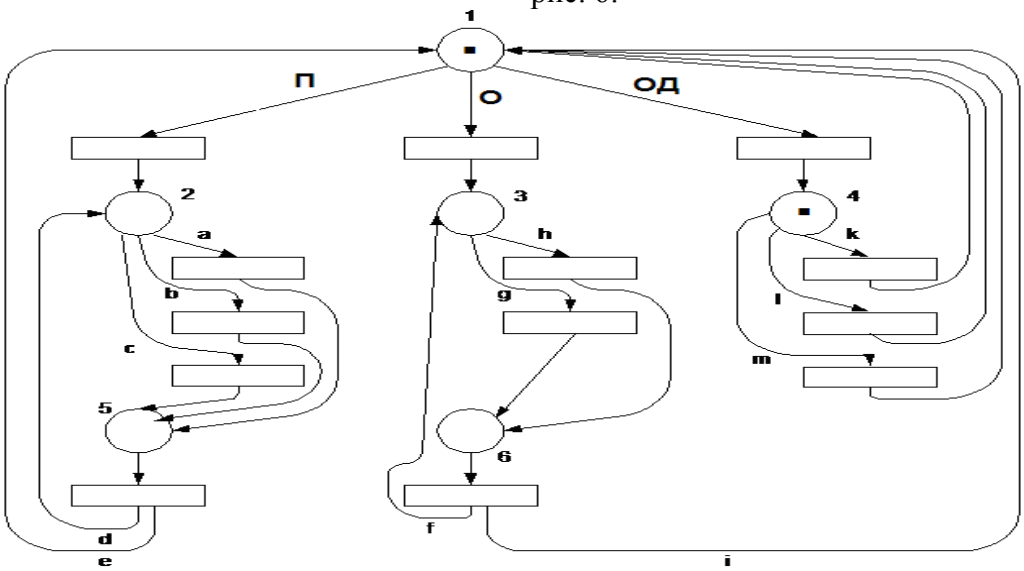


Рис. 6. Разметка M^5

В этом состоянии могут быть активизированы переходы k,l,m, которые связаны с ускорением автомобиля. Переход k активизирует процесс разгона, l – процесс торможения, m – движение с постоянной скоростью (без ускорения). После срабатывания любого из переходов k,l,m имеем разметку M^0=[2,0,0,0,0,0] (см. рис. 1).

Следует отметить, данная сеть описывает только возможные состояния автомобиля, и не учитывает конфигурацию дороги, а также взаимодействие с другими транспортными средствами. Алгоритмы выполнения маневра перестроения и обгона для системы, состоящей из более чем одного транспортного средства, приведены далее:

Алгоритм перестроения

1) Проверка возможности перестроения на желаемую полосу дороги. Для этого необходимо выполнение всех следующих условий:

- выполнение маневра не мешает движению остального транспорта;
- отсутствует знак о запрете перестроения;

Если возможность есть, то переходим на пункт 2, иначе – конец алгоритма.

2) Проверка того, что автомобиль, совершающий маневр перестроения, окажется на свободном месте полосы дороги, на которую перестраивается. Если результат положительный, то переход на пункт 3, иначе – конец алгоритма.

3) Проверка того, что после завершения автомобилем маневра перестроения расстояние между ним и автомобилями, которые едут за ним и перед ним, будет соответствовать нормам, описанным в правилах дорожного движения. Если результат положительный, то переход на пункт 4, иначе – конец алгоритма.

4) Проверка того, что ни один из автомобилей, едущих за данным и перед ним, не совершают каких-либо маневров. Если результат положительный, то переход на пункт 5, иначе – конец алгоритма.

5) Выполнение перестроения на желаемую полосу. Переход на пункт 6.

6) Конец алгоритма.

Алгоритм обгона

1) Проверка возможности обгона. Для этого необходимо выполнение всех следующих условий:

а) Перед началом обгона водитель должен убедиться в том, что:

- ни один из водителей транспортных средств, которые движутся за текущим и которым можно создать помеху, не начал обгон;
- водитель автомобиля, который движется перед текущим по той же самой полосе, не подал сигнала поворота влево.

б) Обгон запрещен:

- на перекрестке, кроме случаев, когда обгоняются двухколесные транспортные средства без бокового прицепа;
- на железнодорожных переездах и ближе 100 м до них;
- ближе чем за 50 м перед пешеходным переходом в населенном пункте и 100 м – вне населеного пункта
- в конце подъема, в туннелях, на мостах, эстакадах, крутых поворотах и других участках дорог с ограниченной видимостью;
- транспортного средства, совершающего обгон или объезд;
- на дорогах, которые имеют 2 и больше полос для следования транспорта в одном направлении;

в) Отсутствует знак о запрете обгона.

2) Вычисление времени совершения обгона.

3) Проверка условия, что обгон завершится удачно. Это означает, что должны выполниться условия:

- автомобиль не столкнется с автомобилем, едущим ему на встречу;
- когда автомобиль закончит обгон, расстояние между ним и автомобилем, идущим за данным, будет отвечать нормам, установленным в правилах дорожного движения.

Если результат положительный, то перейти на пункт 4, иначе – конец алгоритма.

4) Выполнение обгона. Переход на пункт 5.

5) Конец алгоритма.

Приведенные алгоритмы могут использоваться как булевы функции, которые, в случае положительного результата, активируют переходы $П$ и $О$.

Выводы по данному исследованию и перспективы дальнейшего развития в данном направлении

Формализация алгоритма моделирования движения автомобильного дорожного транспорта с помощью сетей Петри позволяет легко построить программный алгоритм для его обработки.

Предложенный формальный алгоритм позволяет наглядно представить возможные состояния транспортного средства с помощью сети Петри и разметок M^0 – M^5. Использование алгоритмов перестроения и обгона в качестве активаторов соответствующих переходов, позволяет полностью определить влияние на транспортное средство конфигурации дороги (с учетом дорожных знаков и разметки) и других участников движения.

Список литературы

1. Томашевський В.М. Моделювання систем. – К.: Видавнича група BHV, 2005. – 352 с.
2. Томашевский В.Н., Парамонов А.М. Система моделирования автомобильного дорожного движения. // Математическое и имитационное моделирование систем МОДС '2008. Тезисы докладов. К. – 2008. С. 183-187.

УДК 004.052.042

МАРКОВСЬКИЙ О.П.,
МАЗУР Р.Ф.,
САЇД РЕЗА МАХМАЛІ

ПІДВИЩЕННЯ ЕФЕКТИВНОСТІ ІДЕНТИФІКАЦІЇ АБОНЕНТІВ З ВИКОРИСТАННЯМ МОДЕЛІ ПЕРСЕПТРОНУ

У статті пропонується вдосконалення використання нейромережних технологій для ідентифікації віддалених абонентів багатокористувацьких систем. Застосування моделі персептрону в якості перетворювача з неоднозначною зворотною функцією дозволяє прискорити ідентифікацію на основі концепції “нульових знань”. Виявлено нові можливості прискорення обчислювальної реалізації моделі персептрону при її застосуванні для ідентифікації абонентів. Запропоновано нову схему обчислень, проведено аналітичне та експериментальне дослідження її ефективності для підвищення продуктивності ідентифікації.

The paper deals with offers how to improve the using of neuronet technologies for remote multi-user systems subscribers identification. Usage of the perceptron model as the ambiguous reversible functional transformation allows to accelerate the identification process based on “zero knowledge” concepts. New ways of speeding up the perceptron model are exposed. What the article brings out is the new calculation scheme. Its efficiency is discovered both analytically and experimentally in order to grade up the identification performance.

Вступ

Динамічний розвиток та поглиблення інформаційної інтеграції викликає зростання числа багатокористувацьких комп’ютерних систем. Такі системи, що пов’язані з абонентами через локальні та глобальні комп’ютерні мережі, дозволяють реалізувати колективний доступ до інтегрованих інформаційних ресурсів. При цьому важливим фактором ефективності багатокористувацьких систем є унеможливлення несанкціонованого доступу до їх інформаційних ресурсів, а також контроль за дотримання прав доступу з боку легальних абонентів. Окрім того, тенденцією останніх років стало не тільки зростання числа таких систем і розширення сфери їх застосування, але й збільшення кількості їх абонентів.

Указані фактори визначають необхідність удосконалення засобів ідентифікації абонентів багатокористувацьких систем, особливо коли їх застосування розширюється в таких сферах людської діяльності, неконтрольоване втручання в які може мати тяжкі наслідки. Це стосується, в першу чергу, систем управління повітряним транспортом, залізницею – тобто систем, які відповідають за забезпечення безпеки людської життєдіяльності. Розширення використання багатокористувацьких систем у галузях економіки стимулює зростання економічних злочинів, що реалізуються шляхом доступу до конфіденційної комерційної інформації. Таким чином, розширення сфер використання багатокористуваньких систем вимагає підвищення ефективності систем ідентифікації їх абонентів шляхом зменшення ризиків неконтрольованого доступу до їх інформаційних ресурсів.

Характерне для сьогоднішнього етапу розвитку багатокористувацьких систем зростання кількості їх абонентів вимагає відповідного підвищення продуктивності ідентифікації.

Оскільки підвищення надійності захисту від неконтрольованого доступу вимагає, як правило, збільшення об’єму часових ресурсів для його реалізації, то комплексне вирішення проблеми зменшення ризиків неконтрольованого доступу та підвищення продуктивності контролю за доступом може бути досягнуто лише за рахунок розробки принципово нових методів та засобів ідентифікації абонентів.

Отже, проблема вдосконалення методів та засобів ідентифікації абонентів багатокористувацьких комп’ютерних систем є актуальною і важливою для сучасного етапу розвитку інформаційних технологій.

Аналіз ефективності існуючих технологій ідентифікації абонентів

Сучасні технології ідентифікації поділяються на два класи: з використанням паролів (“слабка” ідентифікація) та на основі концепції “нульових знань” (“строга” ідентифікація). Остання, з точки зору рівня захищеності від

неконтрольованого доступу, значно переважає ідентифікацію з використанням паролів [2].

Сутність концепції “нульових знань” полягає в тому, що для доведення своєї автентичності абонент має неявним чином виявити знання певної інформації, якою система не володіє, але може перевірити її наявність у абонента. У якості механізму перевірки наявності у абонента знань виступає перетворення, зворотне до якого не може бути отримано аналітичним шляхом і є неоднозначним. У більшості протоколів ідентифікації, що реалізують цю концепцію (схеми Fiat-Shamir, Schnorr [2]) в якості вказаних перетворень виступають незворотні задачі теорії чисел. Базовими для таких перетворень є мультиплікативні модульні операції над числами, довжина яких на порядки перевищує розрядність процесора. Значна обчислювальна складність цих операцій має наслідком низьку продуктивність ідентифікації.

Тому доцільно шукати альтернативні способи реалізації базової концепції “нульових знань”. Одним із напрямків такого пошуку може бути застосування перетворень, які володіють властивостями неоднозначності, але не забезпечують умову строгої математичної незворотності. Відповідно до базового принципу захисту інформації, який полягає втому, що рівень захисту вибирається таким чином, щоб потенційний зиск її порушення (отримання несанкціонованого доступу) був завжди меншим вартості ресурсів, що витрачаються на подолання захисту, існує досить широке коло багатокористувацьких систем, для яких застосування концепції “нульових знань” на основі математично строгих незворотних перетворень є надлишковим. Для таких систем, потенційна користь несанкціонованого доступу до ресурсів яких відносно невелика, при реалізації концепції “нульових знань” можуть бути використані більш прості неоднозначні перетворення, що не володіють властивістю строгої математичної незворотності. В останні роки запропоновано ряд подібних перетворень для застосування в області концепції “нульових знань”. Вони є більш ефективними в обчислювальному плані, ніж мультиплікативні операції модульної арифметики, а їх незворотність основана на NP-повних задачах. У числі таких альтернативних варіантів реалізації концепції “нульових знань” можна назвати схеми ідентифікації об’єктів на основі задачі перестановки ядер (PKP – Permuted Kernels Problem), аналізу симптомів SD (Sendrom Decoding), з використанням обмежених лінійних рівнянь CLE (Constrained Linear Equations) [3].

Одним із можливих альтернативних варіантів реалізації спрощеної версії концепції “нульових знань” є використання нейромережних технологій, основаних на моделі персептрону [1,3].

Таким чином, метою досліджень є вдосконалення технології ідентифікації абонентів на основі концепції “нульових знань” з використанням для її реалізації моделі персептрону.

Протоколи ідентифікації абонентів з використанням моделі персептрону

До теперішнього часу запропоновано ряд математичних моделей базового компонента нейронної мережі, персептрону [1]. Усі ці моделі володіють властивістю неоднозначності. Це означає, що той самий вихідний сигнал персептрону може бути викликаний великим числом варіантів значень вхідних сигналів. Саме ця властивість персептрону відкриває можливості його використання в реалізації спрощеної версії концепції “нульових знань”. Для пересептрону завдання знаходження зворотного перетворення, тобто одержання набору значень вхідних сигналів, які викликають заданий вихідний сигнал, є задачею NP-складності [3].

З огляду на обмежено лінійний характер перетворень, реалізованих у моделі персептрону, завдання знаходження зворотного перетворення не є, у строгому математичному формулюванні, аналітично нерозв'язною. Однак при великій кількості вхідних сигналів, її рішення вимагає значних обчислювальних ресурсів, вартість яких для широкого кола багатокористувацьких систем перевищує вартісне вираження користі від несанкціонованого доступу до її ресурсів.

Найпростіший персептрон описується матрицею W, що складається з m рядків та n стовпчиків, при цьому елементи матриці належать множині $\{-1;1\}$. Нехай існує множина R, яка містить m невід’ємних цілих чисел, кожне з яких не перевищує n. Задача знаходження n-компонентного вектора Y, кожен з елементів якого належить множині $\{-1;1\}$, такого, щоб його добуток з матрицею W був рівний векто-

ру $S = \{s_1, s_2, ..., s_m\}$, $\forall j \in \{1,2,...,m\}: s_j \geq 0 : W \cdot Y = S$, і є задачею NP-складності.

У якості однакового для всіх абонентів відкритого ключа системи використовуються матриця *W* та вектор *S*. Обидва ці компоненти генеруються випадковим чином у системі та розсилаються абонентам при їх реєстрації.

Абонент А після одержання від системи матриці *W* і вектора *S*, виконує наступну послідовність дій.

1. Спочатку випадковим чином генерується *n*-компонентний вектор $Y=\{y_1,y_2,\ldots,y_n\}$, $\forall j\in\{1,\ldots,n\}: y_j\in\{-1,1\}$.
2. Виконується множення зазначеного вектора на матрицю *W*. Результатом цього множення є вектор $R=\{r_1,r_2,\ldots,r_m\}:R=W\cdot Y$, кожна з *m* компонент $r_1,r_2,\ldots,r_m$ якого є додатним або від'ємним цілим числом.

Отримана матриця *W* перетворюється в модифіковану матрицю *W′*. Для цього послідовно аналізуються компоненти $r_1,r_2,\ldots,r_m$ вектора R: якщо *i*-та ($i\in\{1,\ldots,m\}$) компонента r_i вектора *R* від'ємна ($r_i < 0$), то *i*-тий рядок модифікованої матриці *W′* утворюється як інверсія однойменного рядка матриці *W*: $\forall j\in\{1,\ldots,n\}$: $w_{ij}=-1\cdot w_{ij}$. В іншому випадку, тобто за умови, що *i*-та ($i\in\{1,\ldots,m\}$) компонента r_i вектора *R* невід'ємна ($r_i \geq 0$), то *i*-тий рядок модифікованої матриці *W′* збігається з однойменним рядком матриці *W*: $\forall j\in\{1,\ldots,n\}$: $w_{ij}=w_{ij}$.

3. Обчислюється *m*-компонентний вектор S_A добутку модифікованої матриці *W′* на вектор *Y*: $S_A = W'\cdot Y$.
4. Код *m*-компонентного вектора S_A відсилається системі як відкритий ключ користувача А. Ці коди, на відміну від *W*, різні для кожного з легальних абонентів системи.

Абонент А формує випадковим чином наступні компоненти, що становлять його секретний ключ.

- Вектор P перестановок рядків матриці W, що містить m неповторюваних компонентів P={p1,p2,…,pm}, при цьому, кожна j-та компонента вектора P не перевищує m: ∀j=1,…,m: pj≤m та показує позицію i-того рядка в перетвореній матриці.
- Вектор Q ={q1,q2,…,qn}, що описує перестановки стовпчиків матриці W зі зміною знаків їхніх компонент. Кожна i-та компонента вектора Q лежить в інтервалі від –n до n: ∀i=1,…,n: -n≤pi≤n і показує позицію i-того стовпчика в перетвореній матриці. Якщо qi <0, то додатково до перестановки змінюються на протилежні знаки всіх компонентів i-того стовпчика матриці W.
- Випадковий вектор Z.

Для подальшої роботи формується вектор $Q'=\{q_1',q_2',\ldots,q_n'\}$, зворотний до вектора перестановки стовпців $Q = \{q_1,q_2,\ldots,q_n\}$. Вектор *Q′* здійснює зворотну перестановку стовпців матриці зі зміною знака: $\forall i\in\{1,\ldots,n\}$: $q'_{q_i} = i \cdot sign(q_i)$.

При виконанні циклу звернення до системи користувач А виконує наступну послідовність дій.

1. Здійснюється перестановка рядків і стовпчиків матриці *W* відповідно з обраними векторами перестановки рядків *P* та стовпчиків *Q*. У результаті такої перестановки рядків і стовпчиків формується матриця *W′*: $W'=Q(P(W))$.
2. Модифікується вектор *Y* шляхом зворотної перестановки його компонентів відповідно до вектора Q^{-1}. У результаті такої зворотної перестановки буде утворено модифікований вектор Y' : $Y'=Q'(Y)$.
3. Вираховується *n*-компонентний вектор *D* як сума вектора *V* і модифікованого вектора Y' : $D= Z + Y'$.
4. Обчислюються хеш-згортки наступних кодів: конкатенації векторів перестановок *P* та *Q*: $h_0 = H(P|Q)$, хеш-згортки векторів *Z* та *D* окремо: $h_1 = H(Z)$ і $h_2 = H(D)$, а також хеш-згортки добутків зазначених *n-компонентних* векторів на модифіковану матрицю W' : $h_3 = H(W'\cdot Z)$ і $h_4 = H(W'\cdot D)$.
5. Коди h_1, h_2, h_3 та h_4 відсилаються до системи.

Система випадковим чином генерує ціле число $c\in\{0,1,2,3\}$ і відправляє це число користувачеві А.

Якщо *c*=0, то користувач А відправляє системі три коди: *P*, *Q* та *Z*. Система перевіряє правильність передачі кодів *P* й *Q* шляхом порівняння $H(P|Q)$ з раніше отриманим кодом хеш-згортки h_0; обчислює хеш-згортку $H(Z)$ отриманого від користувача вектора *Z* і порівнює її з отриманим раніше h_1; система виконує перестановку рядків матриці *W*: $P(W)$, здійснює перестановку компонентів отриманого від користувача вектора *Z*: $Q(Z)$ і формує хеш-згортку добутку $P(W)\cdot Q(Z)$. Якщо вона збігається з раніше отриманим кодом h_3, тобто

$H(P(W)\cdot Q(Z)) = h_3$, то користувачеві А надаються права доступу.

Якщо c=1, то користувач А також відправляє системі коди трьох векторів: P, Q та D. Система виконує аналогічні обчислення й перевірки з тією різницею, що те, що виконувалося для вектора Z виконується тепер для вектора D: перевіряється правильність передачі кодів P й Q шляхом порівняння $H(P|Q)$ з раніше отриманим кодом хеш-згортки h_0; обчислюється хеш-згортка $H(D)$ отриманого від користувача n-компонентного вектора D і порівнюється з отриманим раніше h_2; виконується перестановка рядків матриці W: $P(W)$, здійснюється перестановка компонентів отриманого від користувача вектора D: $Q(D)$, і формується хеш-згортка добутку $P(W)\cdot Q(D)$. Якщо хеш-згортка вказаного добутку збігається з раніше отриманим кодом h_4, тобто $H(P(W)\cdot Q(D)) = h_4$, то користувачеві А надаються права доступу.

Якщо c=2, абонент А множить модифіковану матрицю W' на вектор Z і відсилає системі отриманий добуток – $W'\cdot Z$, після чого, аналогічним способом множить модифіковану матрицю W' на вектор Y_A', а отриманий добуток $W'\cdot Y_A'$ відсилає системі. Система, при отриманні вказаних кодів, обчислює хеш-згортку добутку $W'\cdot Z$ і порівнює отриманий результат $H(W'\cdot Z)$ з раніше отриманим кодом h_3, потім система обчислює суму отриманих від абонента добутків $W'\cdot Z$ й $W'\cdot Y_A'$, після чого формується хеш-згортка $H(W'\cdot Z+W'\cdot Y_A')$ цієї суми, що порівнюється з отриманим від абонента кодом h_4. Нарешті, система перевіряє, щоб усі компоненти добутку $W'\cdot Y_A'$ містилися в множині S_A. Якщо всі вказані вище умови виконуються, то система надає абонентові А права доступу до своїх ресурсів.

Якщо c=3, то абонент А відсилає системі сформовані вектори Z й Y_A'. Система, при одержанні цих векторів обчислює хеш-згортку $H(Z)$ вектора Z і порівнює її з кодом h_1, обчислює хеш-згортку суми $H(Z+Y_A')$ отриманих від абонента векторів Z й Y_A', порівнює її з отриманим кодом h_4, якщо у всіх випадках виявляється збіг зазначених компонентів, то абонентові А надаються права на доступ до ресурсів системи.

Обчислювальна складність наведеної процедури ідентифікації може бути зменшена за рахунок урахування особливостей виконання операцій при ідентифікації.

Спосіб зменшення обчислювальної складності реалізації моделі персептрону

Найбільш масовою й трудомісткою операцією, що використовується в наведеній вище схемі ідентифікації на основі одношарового персептрону є скалярне множення. Кожен з компонентів результату при такому множенні є сумою попарних добутків елементів векторів або матриць, які належать множині {-1,1}. Очевидно, що згадані попарні добутки також можуть приймати значення із цієї ж множини {-1,1}.

Лема: Якщо значення чисел a та b належать множині {-1,1}: $a,b\in\{-1,1\}$, а самі числа представлені в доповнюючому коді, то й їхній добуток може бути представленим у вигляді: $a\cdot b = a \oplus b \vee 1$.

Доведення: Як відзначалося вище, якщо значення множників a і b належать бінарній множині {-1,1}, то значення добутку також належить цій множині. При цьому, якщо $a=b$, то $a\cdot b$ =1, а якщо $a\neq b$, то $a\cdot b = -1$. У першому випадку $a\oplus b=0$ і тому $a\oplus b\vee 1=1$. У другому випадку можливо два варіанти. Для визначеності, нехай a=1, b=–1, тоді $a\oplus b = 1\oplus(-1) = -2$. Отже, $a\oplus b\vee 1 = -2\vee 1 = -1$. Для зворотніх значень a та b результат буде таким же.

Доведена лема дозволяє реалізувати скалярне множення істотно швидше, за рахунок відмови від операції множення компонент векторів. Це має велике практичне значення для прискорення процесу ідентифікації, оскільки, для того, щоб забезпечувати прийнятний рівень захисту від підбору, реальна розмірність матриць, які використовуються, повинна лежати в діапазоні 200-300 [3].

При прямому скалярному множенні n-компонентних векторів та умові, що компоненти векторів зберігаються в неупакованому форматі, множення пари компонент вимагає однієї операції множення, тривалість якої становить t_m. Для одержання результату – скалярного добутку двох векторів затрачається час, рівний $n\cdot t_m+(n-1)\cdot t_a$, де t_a – час виконання операції додавання. Якщо прийняти, що тривалість t_m операції множення в α раз перевищує тривалість t_a операції додавання ($\alpha=t_m/t_a$), то час T_0 скалярного множення матриці, що складається з m рядків й n стовпчиків на n-компонентний вектор визначається формулою:

$$T_0 = m\cdot(n\cdot t_m+(n-1)\cdot t_a) \approx m\cdot n\cdot(\alpha+1)\cdot t_a \qquad (1)$$

При використанні запропонованого способу реалізації скалярного добутку доцільно зберігати матрицю й вектор, що множаться, в ”упакованому” форматі. У такому форматі використовуються для зберігання кожного компонента 2 розряди, відповідно для зберігання n компонент вектора буде потрібно $2 \cdot n/l$ машинних слів (кодів, рівних розрядності процесора – l). При використанні 2-х розрядів для зберігання доповнюючого коду кожного компонента a векторів, що множаться, значення 1 відповідає коду пари розрядів 01, а значення a = -1 відповідає коду 11. При використанні описаного формату одночасно можна виконувати операцію множення над $l/2$ парами компонентів. Сама операція множення при цьому зводиться до послідовного виконання двох логічних операцій – порозрядної суми по модулю 2 й операції АБО з константою. Для додавання отриманих у результаті цих операцій 2-бітових полів можна використати або табличний суматор, або операції зсуву й додавання. З урахуванням цього, і беручи до уваги те, що тривалість логічних операцій не перевищує тривалості операції додавання, час одержання скалярного добутку двох n-компонентних векторів складе $6 \cdot n \cdot t_a/l$. Тоді час T_1 скалярного множення матриці, що складається з m рядків й n стовпчиків на n-компонентний вектор при використанні запропонованої організації виконання операцій визначається формулою:

$$T_1 = \frac{6 \cdot m \cdot n \cdot t_a}{l} \qquad (2)$$

Порівняння часу скалярного множення матриці на вектор для двох описаних варіантів показує, що запропонований спосіб виконання цієї операції дозволяє зменшити час її виконання в β разів, при цьому чисельне значення β визначається наступним виразом:

$$\beta = \frac{T_0}{T_1} = \frac{l \cdot (\alpha + 1)}{6} \qquad (3)$$

Зокрема, для 32-розрядного процесора (l=32) і значення α =3, час виконання базової операції схеми ідентифікації на основі моделі персептрону, скалярного множення матриці на вектор, при використанні запропонованого способу її виконання скорочується в 21 раз (β=21.3) у порівнянні з традиційним способом її реалізації. Ще однією перевагою запропонованого способу виконання базової операції є скорочення необхідного для зберігання операндів об'єму пам'яті приблизно в $l/2$ разів. У таке ж число раз скорочується час виконання підсумовування векторів і виконання іншої операції – обчислення хеш-згортки.

На жаль, у самому протоколі ідентифікації неможливо отримати вказане прискорення через наявність операцій транспонування матриць у процедурах верифікації. Транспонування матриць з даними, що представлені в упакованому форматі, є досить складною операцією, а тому реальне прискорення в процесах ідентифікації, що було виявлено в ході поставлених експериментів, суттєво відрізняється від номінального в операції знаходження скалярного добутку та складає близько 30% – 50%.

Експериментальні дослідження запропонованої схеми обчислень показали результати, зображені на рисунках 1 та 2.

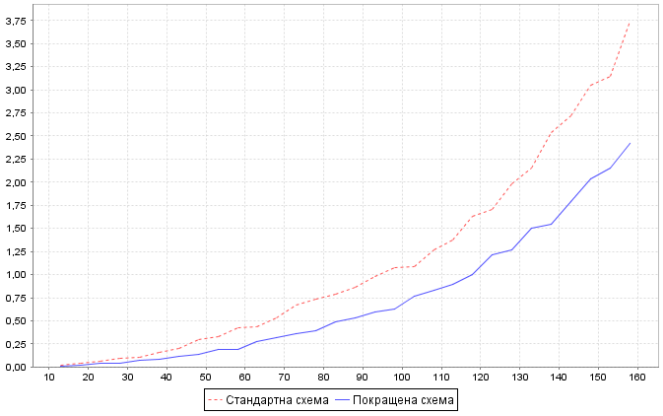


Рис. 1. Швидкість реєстрації

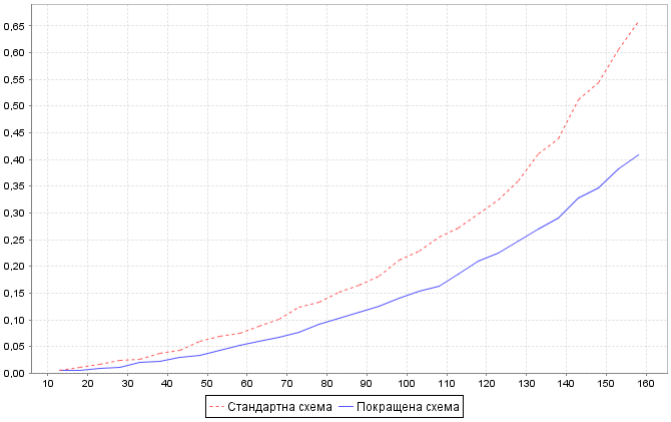


Рис. 2. Швидкість ідентифікації

Графіки показують залежність часу, який втрачається на реєстрацію та ідентифікацію абонентів у залежності від розмірностей матриць та векторів, що використовуються у протоколах відповідних процесів. Початкова розмірність була задана параметрами m=11, n=13.

Для процедури реєстрації було проведено тести при 30 різних розмірностях, нарощуючи її на кожному кроці на 5. Результати для кожної з розмірностей є середнім значенням 10 тис. ітерацій. Дані тестування швидкості ідентифікації були визначені також для 30 розмірностей матриць та векторів з кроком рівним 5. Для кожної з розмірностей було виконано 100 тис. ітерацій.

З наведених графіків випливає, що використання запропонованої організації обчислень дозволяє на 25-35% прискорити ідентифікацію.

Проведені дослідження залежності від розрядності параметрів m та n стійкості такої схеми ідентифікації абонентам до спроб підбору кодів h, V_1 й V_2 для несанкціонованого доступу до ресурсів системи, показали, що для практичного використання значення розрядностей указаних вище параметрів повинно бути не менше 200.

Основні переваги використання моделі персептрону для реалізації ідентифікації абонентів на основі спрощеної концепції “нульових знань” наведено нижче.

1. Ідентифікація використовує відкритий ключ системи – матрицю W великого об'єму однакову для всіх легальних абонентів. Відкриті індивідуальні ключі абонентів – вектори S мають малий об'єм. Це забезпечує істотно менший обсяг пам'яті, необхідний для зберігання в системі інформації, яка використовується для ідентифікації абонентів.

2. Генерація відкритого ключа абонента здійснюється набагато простіше й фактично не вимагає великих обчислювальних ресурсів.

3. Число сеансів звертання практично не обмежено.

Для практичного використання моделі персептрону при реалізації ідентифікації абонентів на основі спрощеної концепції “нульових знань” розроблено програмний продукт.

Висновки

В результаті проведених досліджень показано, що для спрощеної ідентифікації абонентів на основі теоретично строгої концепції нульових знань може бути застосована модель персептрону, яка має властивості неоднозначності зворотного перетворення. Вдосконалено спосіб використання моделі персептрону для ідентифікації абонентів за рахунок урахування особливостей моделі для вказанного застосування. Виявлені особливості дозволили спростити обчислення, повязані з реалізацією моделі персептрону і тим самим підвищити на 25-35% продуктивність ідентифікації. Доведено, що використання моделі персептрону дозволяє спростити генерацію ключів та зменшити об’єм інформації, що використовується в системі для ідентифікації.

Отримані результати можуть бути використані для підвищення ефективності багатокористувацьких систем зберігання інформації.

Перелік посилань

1. Ияд Мохд Маджид Ахмад Шахрури, Магрело В.Д. Потапенко М.М. Ідентифікація віддалених абонентів на основі технології нейронних мереж // Матеріали X Міжнародної науково-технічної конференції ”Системний аналіз та інформаційні технології”.– К.:НТУУ ”КПІ”.–2008.– С.357.
2. Fiat A., Shamir A. How to prove yourself: practical solutions to identification and signature problems // Proceeding of Workshop ”Crypto’86”.– Santa Barbara, USA, August 11-15, 1986.– Berlin: Springer-Verlag.– LNCS-263 – 1986.– P.186-194.
3. Pointcheval D. A new identification schemes based on perceptrons problem// Proceeding of International Conference “Eurocrypt’95”.– Saint-Malo, France, May 21-25, 1995.– Berlin: Springer-Verlag.– LNCS-921 – 1995.– P.319-328.

УДК 004.94(075.8)

БОГУШЕВСЬКА Н.В.,
ТОМАШЕВСЬКИЙ В.М.

ФОРМАЛЬНА МОДЕЛЬ КІЛЬЦЕВОЇ ТРАНСПОРТНОЇ МЕРЕЖІ. РЕАЛІЗАЦІЯ ПЕРЕТВОРЕННЯ ДІАЛОГОВОЇ НОТАЦІЇ ДО ПРОГРАМНОГО КОДУ

Предложены формальные модели, позволяющие задать характеристики транспортного средства и объекта перевозки для кольцевой транспортной сети. Созданы диалоговые нотации для задания параметров кольцевой сети. Определены функциональные зависимости, позволяющие использовать для моделирования транспортной сети программный код, в котором конфигурация транспортной сети зависит только от параметрической настройки зависимостей. Рассмотрен пример задания транспортной сети при помощи формальных моделей и функциональных зависимостей.

Formal models for defining description of transport facilities and transportation objects in circular traffic net are proposed. Dialog notations for defining ring net attributes are made. Functional dependencies that allow use for modeling the program segment, in which the configuration of ring net depends only on parametric tuning of those dependencies, are described. An example of defining circular traffic net by means of proposed formal models and functional dependencies is viewed.

Постановка задачі у загальному вигляді

Транспортна інфраструктура є однією з найважливіших структур, яка забезпечує життя міст та регіонів. В останній час в багатьох великих містах вичерпані (або близькі до цього) можливості екстенсивного розвитку транспортних мереж. Тому особливо важливим стає питання оптимального планування розвитку мереж, поліпшення організації руху, оптимізація маршрутів перевезення вантажів і громадського транспорту. Вирішення цих питань неможливе без застосування математичного моделювання транспортних мереж.

При моделюванні транспортних мереж особу зацікавленість являють собою задачі, пов’язані із завданням кільцевого маршруту (кільцевої мережі). Така мережа має ряд вузлів (зупинки для входу чи виходу пасажирів, пункти завантаження чи вивантаження товарів) та динамічні транспортні об’єкти (автобуси, тролейбуси, вантажівки), що рухаються за кільцевим маршрутом.

До вузлів кільцевої мережі надходять інші динамічні об’єкти (пасажири чи вантажі) – об’єкти перевезень, які мають бути перевезені об’єктами руху з вузла завантаження до вузла вивантаження. У вузлах мережі можуть утворюватися черги із об’єктів перевезень (рисунок 1).

Прикладом кільцевої мережі може служити кільцевий маршрут громадського або вантажного транспорту. Моделювання пасажирських перевезень є важливим не лише для стратегічного розвитку громадського транспорту, але й для розв’язку задачі оперативного контролю пасажирообігу. Саме для перевезення пасажирів найбільш важливим стає наявність у системі моделювання можливості побудови транспортної мережі із кільцевим маршрутом.

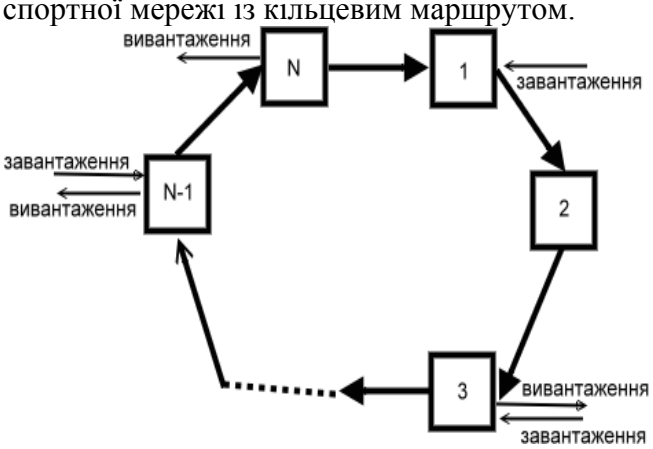


Рис. 1. Приклад кільцевої мережі

Також за допомогою кільцевої мережі можна моделювати дискретне складальне виробництво, документообіг, системи логістики, різного вигляду обслуговуючі системи та ін.

Аналіз останніх досліджень та публікацій по темі

У роботі [1] була запропонована формальна модель для визначення кільцевої транспортної мережі у вигляді:

$$M_t = < H, C, K, F >, \qquad (1)$$

де

H – граф, що описує структуру кільцевої мережі;

C – множина об'єктів перевезень (пасажири, вантажі);

K – множина типів транспортних засобів,

F – множина функцій, що описують взаємодію елементів із C і K.

У роботі [2] визначені основні дії, які можуть виконувати транспортні засоби та об'єкти перевезення. Робота [3] описує програмний сегмент мовою GPSS, який моделює кільцеву транспортну мережу із використанням технології непрямої адресації.

Реалізація механізму перетворення діалогової нотації кільцевої мережі до програмного коду базується на використанні *лінгвістичного процесора*, означення якого, а також приклади застосування наведені у роботах [4] та [5].

Виділення невирішених частин загальної проблеми

Після визначення формальної моделі кільцевої мережі постає питання створення формальної нотації для повного опису структури транспортного засобу та об'єкту перевезення, а також створення функціональних залежностей для реалізації механізму перетворення діалогової нотації до програмного коду.

Постановка задачі

Таким чином, необхідно вирішити наступні задачі

Створення діалогових нотацій для визначення структури кільцевої мережі (згідно із (1)).

Розробка формальних моделей для опису транспортного засобу та об'єкту перевезення, створення відповідних діалогових нотації.

Визначення формальних залежностей, для перетворення діалогових нотацій до програмного коду.

Дослідження

Для повного визначення елементу із множини K (1) використовуватимемо структуру наступного вигляду:

$$k_i =< k^{name}, k^c, k^m, k^{ch}, k^h >, \forall k_i \in K, i = \overline{1, K_N}, \quad (2)$$

де

k^{name} – символьне найменування типу транспортного засобу;

k^c – кількість транспортних засобів даного типу у кільцевій мережі;

k^m – максимальна пасажиромісткість транспортного засобу (тобто максимальна допустима кількість об'єктів перевезення у транспортному засобі);

k^{ch} – кількість каналів для одночасного завантаження чи вивантаження об'єктів перевезення до транспортного засобу (наприклад, кількість дверцят у тролейбусі чи автобусі);

k^h – початковий вузол транспортного засобу у кільцевій мережі;

K_N – потужність множини K, $K_N = |K|$.

Елементи множини об'єктів перевезень C (1) будемо задавати структурою

$$c_j =< c^{name}, c^{in}, c^{tin}, c^{out}, c^{tout} >, \forall c_j \in C, j = \overline{1, C_N}, \quad (3)$$

де

c^{name} – символьне найменування об'єкту перевезення;

c^{in} – вузол мережі, у якому відбувається завантаження об'єкту перевезення;

c^{tin} – час, необхідний об'єкту перевезення для завантаження до транспортного засобу;

c^{out} – вузол мережі, у якому відбувається вивантаження об'єкту перевезення;

c^{tout} – час, необхідний об'єкту перевезення для вивантаження із транспортного засобу.

C_N – потужність множини C, $C_N = |C|$.

Після визначення формальної структури множин K та C можна перейти до питання створення діалогових нотацій (відповідно формули (1), (2) та (3)):

Крок 1. Визначення структури кільцевої мережі (формула (1))

Нехай графічна нотація мережі, у якій присутня кільцева мережа, в системі ISS 2000 має наступний вигляд (рисунок 2):

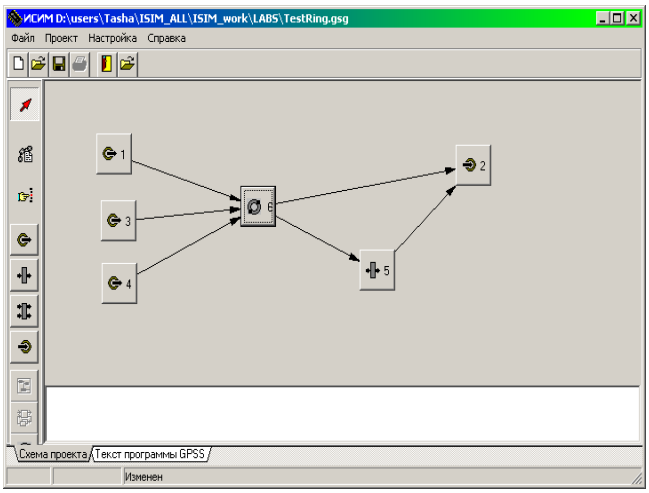
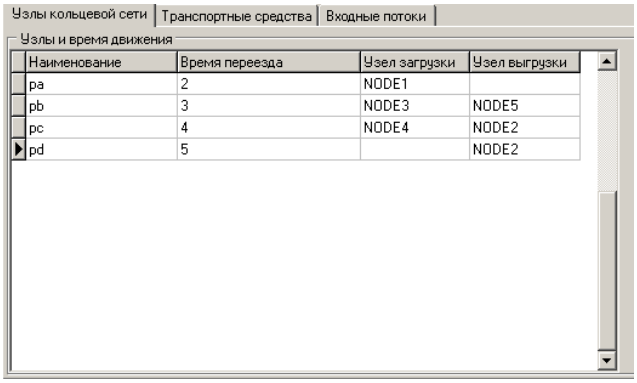


Рис. 2. Графічна нотація кільцевої мережі

Діалогову нотацію для визначення структури кільцевої мережі подамо наступним чином (рис. 3):



Наименование	Время переезда	Узел загрузки	Узел выгрузки
pa	2	NODE1	
pb	3	NODE3	NODE5
pc	4	NODE4	NODE2
pd	5		NODE2

Рис. 3. Діалогова нотація для визначення структури кільцевої мережі

Задану структуру можна представити у вигляді орієнтованого графу, як показано на рисунку 4:

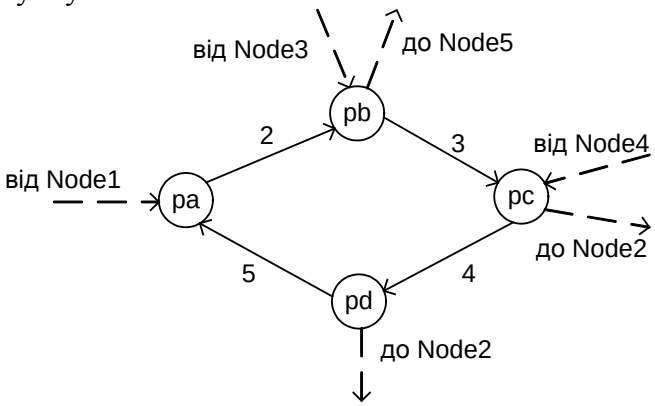


Рис. 4. Представлення структури кільцевої мережі у вигляді орієнтованого графу

У роботі [1] наведена структура графу *H* у наступному вигляді:

$$H =< R, P, W >, \qquad (4)$$

де

R – кінцева множина вершин (у кільцевій транспортній мережі можуть бути присутні вершини чотирьох типів: вершини завантаження, вершини вивантаження, вершини завантаження та вивантаження, вершини, в яких завантаження та вивантаження не здійснюються).

P – кінцева множина дуг – зв'язків між вершинами (для кожної дуги визначається її вага – час пересування конкретного транспортного засобу по даній дузі, причому час може бути випадковою величиною).

W – функція, що ставить у відповідність кожній дузі з *P* впорядковану пару вершин з *R*. Дана функція визначає маршрут транспортного засобу; для кільцевого маршруту з кожної вершини мережі може виходити лише одна дуга, та виходити лише одна дуга.

Така модель повністю описує структуру кільцевої мережі, проте не містить жодної інформації про зв’язок мережі із іншими об’єктами моделі, а саме із джерелами об’єктів перевезення. Тому більш доцільно використовувати структуру вигляду

$$H =< R, P, W, G_M >, \qquad (5)$$

де G_M – матриця, яка визначає зв’язок кільцевої мережі із оточенням, а саме куди мають прямувати об’єкти перевезень із вершин вивантаження, та звідки вони потрапляють до вершин завантаження. Кожен рядок матриці має таку структуру:

$$g_M^i =< v_i, g_{in}^i, g_{out}^i >, \qquad (6)$$

де g_{in}^i, g_{out}^i, – вузли загальної моделі

Тоді, із урахуванням (5) та (6), граф, поданий на рисунку 3, формально буде записаний наступним чином:

$$R = \{pa, pb, pc, pd\},$$

$$P = \begin{pmatrix} 0 & 2 & 0 & 0 \\ 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & 4 \\ 5 & 0 & 0 & 0 \end{pmatrix},$$

$$G_M = \begin{pmatrix} pa & Node1 & \\ pb & Node3 & Node5 \\ pc & Node4 & Node2 \\ pd & & Node2 \end{pmatrix}$$

Крок 2. Визначення елементів множини типів транспортних засобів *K* (формула (2))

Діалогову нотацію для визначення елементів із множини *K* представимо наступним чином (рисунок 5):

Тоді формальний опис матиме вигляд:

$$K = \{k_1, k_2\},$$

$$k_1 =< tra, 5, 40, 4, pa >,$$

$$k_2 =< trb, 10, 80, 4, pa >$$

Крок 3. Визначення елементів множини об'єктів перевезень C (формула (3))

Діалогову нотацію для визначення елементів із множини *C* представимо наступним чином (рис. 6).

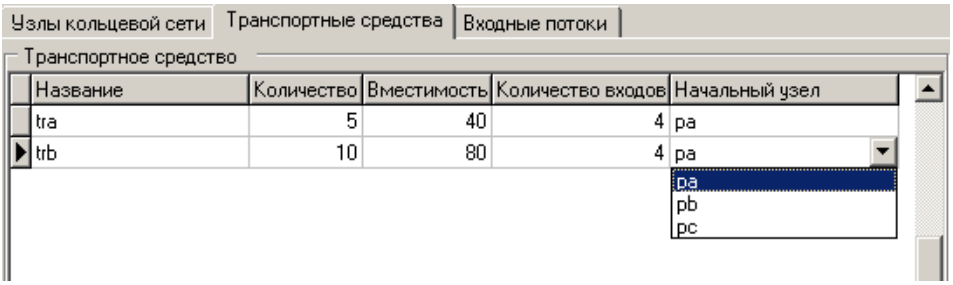


Рис. 5. Діалогова нотація для визначення множини транспортних засобів

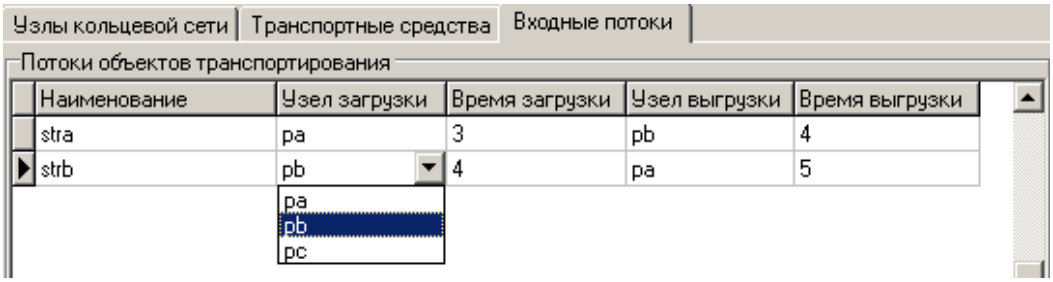


Рис.6. Діалогова нотація для визначення множини об'єктів перевезення

Формальний опис нотації

$$C = \{c_1, c_2\},$$

$$c_1 =< stra, pa, 3, pb, 4 >,$$

$$c_2 =< strb, pb, 4, pc, 5 >$$

Тепер можна перейти до питання визначення функціональних залежностей, які дозволять виконати параметричне налаштування моделі. Введемо наступне означення: нехай ω – характеристика функції користувача:

$$\omega = <\iota, \upsilon, \sigma, \theta>, \qquad (7)$$

де ι – назва (ідентифікатор) функції користувача;

υ – аргумент функції, $\{\upsilon\}|\upsilon \in \Phi \cup \{\wp\}$, де Φ – множина змінних, а $\{\wp\}$ – множина аргументів, що не є змінними (наприклад, стандартні числові атрибути);

σ – вид функції, $\{\sigma\}|\sigma \in \{\sigma\}$: $l \Rightarrow \Im$, де $l \in L$, а $\Im$ =<"*дискретна*", "*неперервна*">;

θ – тіло функції.

Для реалізації моделі (1), а також структур (2), (3) та (5) введемо такі функціональні залежності, використовуючи (7):

1) Визначення еквівалентного найменування об'єкту перевезення.

$$\omega_1 = < FunOType, P\$OTYPE, дискретна, c_1^{name}, 1/ c_2^{name}, 2/ ... / c_{C_N}^{name}, C_N >, \qquad (8)$$

де *P$OTYPE* – параметр транзакту, який містить найменування об'єкту перевезення.

2) Визначення вузла мережі, у якому відбувається завантаження об'єкту перевезення.

$$\Omega_2 = < FunLP, P\$OP, дискретна, 1, c_1^{in} / 2, c_2^{in} / ... / C_N, c_{C_N}^{in} >, \qquad (9)$$

де *P$OP* – параметр транзакту, у якому зберігається результат виконання функціональної залежності ω_1 (формула (8)).

3) Визначення часу, необхідного об'єкту перевезення для завантаження до транспортного засобу.

$$\Omega_3 = < FunLT, P\$OP, дискретна, 1, c_1^{tin} / 2, c_2^{tin} / ... / C_N, c_{C_N}^{tin} >, \qquad (10)$$

4) Визначення вузлу мережі, у якому відбувається вивантаження об'єкту перевезення.

$$\omega_4 = < FunULP, P\$OP, дискретна, 1, c_1^{out} / 2, c_2^{out} / ... / C_N, c_{C_N}^{out} >, \qquad (11)$$

5) Визначення часу, необхідного об'єкту перевезення для вивантаження із транспортного засобу.

$$\omega_5 = < FunULT, P\$OP, дискретна, 1, c_1^{tout} / 2, c_2^{tout} / ... / C_N, c_{C_N}^{tout} >, \qquad (12)$$

6) Визначення часу переміщення транспортного засобу до наступного вузлу кільцевої мережі.

$$\omega_6 = < FunTT, P\$CP, дискретна, r_1, p_{1l_1} / r_2, p_{2l_2} / ... / r_R, p_{|R|l_R} >, \qquad (13)$$

де $r_j \in R, j = \overline{1,|R|};\ p_{jl_j} \in P$, причому $W(p_{jl_j}) =< r_j, r_{l_j} >$;

P$CP – параметр транзакту, який зберігає інформацію про вузол мережі, в якому знаходиться транспортний засіб у поточний момент часу.

7) Визначення еквівалентного найменування транспортного засобу.

$$\Omega_7 = < FunTType,\ P\$TTYPE,\ дискретна,\ k_1^{name}, 1/k_2^{name}, 2/.../k_{K_N}^{name}, K_N >, \quad (14)$$

де *P$TTYPE* – параметр транзакту, який містить найменування транспортного засобу .

8) Визначення пасажиромісткості транспортного засобу.

$$\omega_8 = < FunAP,\ P\$TZ,\ дискретна,\ 1, k_1^m/2, k_2^m/.../K_N, k_{K_N}^m >, \quad (15)$$

де *P$TZ* – параметр транзакту, у якому зберігається результат виконання функціональної залежності ω_7 (формула (14)).

9) Визначення початкового вузлу мережі для транспортного засобу.

$$\omega_9 = < FunSP,\ P\$TZ,\ дискретна,\ 1, k_1^h/2, k_2^h/.../K_N, k_{K_N}^h >, \quad (16)$$

10) Визначення еквівалентного найменування списку об’єктів перевезення поточного вузлу мережі.

$$\omega_{10} = < FunPL,\ P\$CP,\ дискретна,\ r_1, 10/r_2, (10+1)/.../r_{|r|}, (10+|R|-1) >, \quad (17)$$

11) Визначення еквівалентного найменування списку об’єктів перевезення, які переміщуються у транспортному засобі *P$TZ*.

$$\omega_{11} = < FunTL,\ P\$TZ,\ дискретна,\ 1, (10+|R|)/2, (10+|R|+1)/.../K_N, (10+|R|+K_N-1) >, \quad (18)$$

12) Визначення еквівалентного найменування логічного ключу, який відповідає транспортному засобу.

$$\omega_{12} = < FunTK,\ P\$TZ,\ дискретна,\ 1, (10+|R|+K_N)/2,\ (10+|R|+K_N+1)/.../K_N,\ (10+|R|+2\cdot K_N-1), > \quad (19)$$

13) Визначення еквівалентного найменування багатоканального пристрою, який відповідає транспортному засобу.

$$\omega_{13} = < FunTC,\ P\$TZ,\ дискретна,\ 1, (10+|R|+2\cdot K_N)/2,\ (10+|R|+2\cdot K_N+1)/.../K_N,\ (10+|R|+3\cdot K_N-1), > \quad (20)$$

14) Визначення зв’язку вузлу вивантаження із вузлом зовнішньої мережі.

$$\Omega_{14} = < FunPOut,\ P\$UnPoint,\ дискретна,\ c_1^{out}, g_{out}^1/c_2^{out}, g_{out}^2/.../c_{C_N}^{out}, g_{out}^{C_N} >, \quad (21)$$

де *P$UnPoint* – параметр транзакту, у якому зберігається результат виконання функціональної залежності ω_4 (формула (11)).

Для прикладу кільцевої мережі, наведеної на рисунках 2–6, функціональні залежності (8) – (21) матимуть вигляд:

1) ω1 = < *FunOType*, P$OTYPE, *дискретна*, $stra, 1/strb, 2 >$,

FunOType FUNCTION P$OTYPE,D2
stra,1/strb,2

2) ω_2 = < *FunLP*, P$OP, *дискретна*, $1, pa/2, pb >$,

FunOType FUNCTION P$OP,D2
1,pa/2,pb

3) ω_3 = < *FunLT*, *P$OP*, *дискретна*, $1,3/2,4 >$,

FunLT FUNCTION P$OP,D2
1,3/2,4

4) ω_4 = < *FunULP*, *P$OP*, *дискретна*, $1, pb/2, pa >$,

FunULP FUNCTION P$OP,D2
1,pb/2,pa

5) ω_5 = < *FunULT*, *P$OP*, *дискретна*, $1,4/2,5 >$,

FunULP FUNCTION P$OP,D2
1,4/2,5

6) ω_6 = < *FunTT*, *P$CP*, *дискретна*, $pa, 2/pb, 3/pc, 4/pd, 5 >$,

FunTT FUNCTION P$CP,D4
pa,2/pb,3/pc,4/pd,5

7) ω_7 = < *FunTType*, *P$TTYPE*, *дискретна*, $tra, 1/trb, 2 >$,

FunTType FUNCTION P$TTYPE,D2
tra,1/trb,2

8) ω_8 = < *FunAP*, *P$TZ*, *дискретна*, $1,40/2,80 >$,

FunAP FUNCTION P$TZ,D2
1,40/2,80

9) ω_9 = < *FunSP*, *P$TZ*, *дискретна*, $1, pa/2, pa >$,

FunSP FUNCTION P$TZ,D2
1,pa/2,pa

10) ω_{10} = < *FunPL*, *P$CP*, *дискретна*, $pa, 10/pb, 11/pc, 12/pd, 13 >$,

FunPL FUNCTION P$CP,D4
pa,10/pb,11/pc,12/pd,13

11) ω_{11} = < *FunTL*, *P$TZ*, *дискретна*, $1,14/2,15 >$,

FunTL FUNCTION P$TZ,D2
1,14/2,15

12) ω_{12} = < *FunTK*, *P$TZ*, *дискретна*, $1,16/2,17 >$,

FunTK FUNCTION P$TZ,D2

1,16/2,17

13) ω_{13} = < *FunTC*, *P$TZ*, *дискретна*, 1,18/2,19 >,

FunTC FUNCTION P$TZ,D2

1,18/2,19

14) ω_{14} = < *FunPOut*, *P$UnPoint*, *дискретна*, *pb*, *Node*5/ *pc*, *Node*2 >,

FunPOut FUNCTION P$UnPoint,D2

pb,Node5/pc,Node2

Таким чином, функціональні залежності ω_1 – ω_{14} дають змогу провести повне визначення кільцевої мережі відповідно до (5).

Висновки

Визначення структури кільцевої мережі за допомогою моделей (1) – (3) дозволяє формалізувати створення програмного коду при використанні запропонованої діалогової нотації. Функціональні залежності (8) – (21) дозволяють використовувати для моделювання кільцевої транспортної мережі програмний код, який повністю базується на налаштуванні функціональних залежностей, що дозволяє змінювати конфігурацію модельованої транспортної мережі без зміни коду моделі.

Перелік посилань

1. Томашевський В.М., Богушевська Н.В. Педагогічний програмний засіб для навчання імітаційному моделюванню. // Математичне та імітаційне моделювання систем МОДС ‘2007. Тези доповідей. К. – 2007. С. 143-147.
2. Томашевский В. Н., Богушевская Н. В. Расширение возможностей системы моделирования ISS 2000. / Вторая всероссийская научно-практическая конференция по имитационному моделированию и его применению в науке и промышленности. Сб. докладов. – Санкт-Петербург, 2005. С. 103-108.
3. Богушевская Н.В. Особенности реализации кольцевой сети в системе ИСИМ / Міжнародна наукова конференція “Інтелектуальні системи прийняття рішень та прикладні аспекти інформаційних технологій”, м. Євпаторія. – 2005. – т.3
4. Томашевский В. Н., Богушевская Н. В. Расширение возможностей системы моделирования ISS 2000. / Вторая всероссийская научно-практическая конференция по имитационному моделированию и его применению в науке и промышленности. Сб. докладов. – Санкт-Петербург, 2003 С. 190-194.
5. Tomashevskiy V. Automatic generating of GPSS/PC programs. Proceedings 15th European Simulation Multiconference, Prague, 2001, pp. 805-809.

УДК 681.3.06

СТІРЕНКО С.Г.

ФОРМАЛЬНИЙ ОПИС СТРУКТУРНО-ПАРАМЕТРИЧНИХ ХАРАКТЕРИСТИК ТЕХНІЧНОГО ТЕКСТУ

Запропоновані структурно-параметричні характеристики текстових документів, наведені їх обґрунтування і відповідність задачам формальної оцінки блоків технічного документу згідно з рівнем ієрархії. Показано, що при умові виконання вимог вузькоспеціалізованого застосування, дані характеристики дозволяють отримати оцінки семантичної близькості, порівнянності і міру семантичної відповідності для структурних одиниць технічної природної мови.

Proposed structural and parametric characteristics of text documents are the reasons for them, and compliance with an assessment of the challenges of formal blocks of a technical document in accordance with the level of the hierarchy. We show that, subject to compliance with the requirements of specialized applications, these characteristics will provide a semantic affinity evaluation, comparability and consistency to measure the semantic units of natural language technology.

Вступ

Достатньо актуальним завданням є опис структурних компонентів технічних текстів (наприклад, специфікації на програмні продукти). Ринок програмних продуктів розвивається так динамічно, оновлення версій відбувається так часто, що фірми-розробники просто не в змозі відволікати інтелектуальні ресурси, необхідні для складання задовільної документації супроводу. Оскільки витрати часу на опис можна порівняти з витратами на розробку продукту, але вимагають більшого об'єму висококваліфікованої ручної праці, деякі фірми прагнуть заощадити засоби на розробці документації і програють при необхідності розвитку програмного забезпечення. У такій ситуації розвиток засобів автоматизації обробки текстових блоків знижують інтелектуальне навантаження на розробників технічної документації.

При формальному розгляді технічного тексту як структурованого набору символьної інформації стає зрозуміло, що основні елементи його багаторівневої структури – загальновживані слова, а головне – односкладові та багатоскладові поняття проблемної області за своїми взаємозв’язками – можуть бути наведені на основі статистичного аналізу. Проведення структурного статистичного аналізу на базі великої кількості текстів дозволяє автоматично сформулювати декларативне представлення граматики мови, а на множині текстів, які описують проблемну область, автоматично побудувати описання її семантики у вигляді мережі понять та їх зв’язків [1-4]. Відображення нового тексту, який аналізується, на мережу дозволяє виділити в ньому концептуальні поняття та їх зв’язки, розкласифікувати ділянки (частини) тексту за темами на основі віднесення до відповідних понять семантичної мережі і, таким чином, визначити структуру змісту тексту.

Щоб отримати лексико-семантичну інформацію про технічну документацію (ТД) будемо використовувати просторово-векторну модель (ПВМ) [5]. Кожен технічний документ (ТДТ) D_n відображається вектором (w_{n1}, w_{n2},…, w_{nM}), де w_{nk} – вага (або важливість) ключового слова або словосполучення (КСС) t_k з розділу Q_t бази знань (БЗ) G_i для ТДТ D_n, а M – кількість елементів розділу Q_t. Вектор (w_{n1}, w_{n2},…, w_{nM}) будемо називати *лексико-семантичним профілем документу*[1] (ЛСПД).

Частіше за все вагу документу визначають як частоту його появи у тексті [6], але для наших цілей доцільно використовувати логарифмічну шкалу: $\log(N/df_i)$, де N – число документів, а df_i частота використання терміну t_i.

У відповідності до ПВМ, при кількості документів N отримаємо частотну матрицю зустрічальності $F=N\times M$, в якій кожен рядок є ЛСПД:

$$F=\begin{pmatrix} d_1 \\ d_2 \\ \vdots \\ d_N \end{pmatrix}=\begin{pmatrix} w_{11} & w_{12} & \cdots & w_{1M} \\ w_{21} & w_{22} & \cdots & w_{2M} \\ \vdots & \vdots & \ddots & \vdots \\ w_{N1} & w_{N2} & \cdots & w_{NM} \end{pmatrix}. \quad (1)$$

[1] Оскільки різниця між синтаксисом та семантикою доволі розмита), іншими словами складно визначити, де закінчується синтаксис і починається семантика. Більше того, на наш погляд семантика починається вже в морфології.

Оцінка міри семантичної відповідності документів

Для контролю змістової відповідності [7] (семантичної близькості) *Sim* документів D_n будемо використовувати запити *qr*, кожен з яких представлений вектором (qr_1, qr_2,…, qr_M), де qr_i – вага КСС *i* в запиті *qr*.

Оцінювання виконується шляхом розрахунку міри змістової відповідності між документом D_n і запитом *qr* в призначеному змістовому просторі, вираженому вектором. Звичайно мірою близькості вважають косинус кута між двома векторами:

$$Sim_{\cos}(D_n, qr) = \frac{D_n \cdot qr}{\|D_n\| \, \|qr\|} = \frac{\sum_{i=1}^{M} w_{ni} qr_i}{\sqrt{\sum_{i=1}^{M} w_{ni}^2 \sum_{i=1}^{M} q_i^2}} . \quad (2)$$

Також можуть бути використані декілька інших варіантів розрахунку міри змістової відповідності (відстань χ^2, дивергенція Куллбека-Леблера). Їх опис представлений в роботах [8, 9].

Концепція розподіленої семантики

Основна ідея концепції розподіленої семантики (РС) полягає у твердженні лінгвістів, що існує сильна кореляція між розподіленою характеристикою слова (словосполучення) W_{Hd}, яка є об'єктом спостереження, та її значенням W_m. Іншими словами семантика слів Sem_w – це набір контекстів S_c, в яких вони використовуються. Розглянемо три вирази з однієї предметної області G_i:

1. *Вузол X знаходиться в гнізді системної плати.*
2. *Вузол X потребує хорошого охолодження.*
3. *Адміністратор апаратно збільшив тактову частоту вузла X.*

Множина слів W_{si} = {*вузол, знаходиться, гніздо, системна плата, потребує, охолодження, адміністратор, апаратно, тактова частота*} створює спрощене уявлення про сукупний контекст у даному дискурсі поняття *X*. Отримано загальне представлення про ідентифікацію вузла X (мікропроцесор).

Таким чином, резюмуючий тезис можна сформулювати наступним чином: два КСС є семантично близькими, якщо їх розширений контекст також є близьким. Звідси можна зробити висновок, що вищенаведене твердження можна застосувати и до цілих документів: два документи є семантично близькими, якщо складові їх КСС також семантично близькі.

Оцінка семантичних характеристик документу

Поняття *контекст слова* визначає модель зустрічальності. *Спільна зустрічальність*[2] [10, 11] між двома словами визначається як зустрічальність двох слів у даному текстовому модулі у вибраному змістовому просторі. Частоти парної зустрічальності граматичних слів розглядаються різними авторами як істотні характеристики формальної структури тексту.

Для обробки семантичних конструкцій пропонується ієрархія понять *G*. [68] В якості найбільш загального поняття або категорії береться те, яке має найбільший об'єм G_g і, відповідно, найменший зміст. Це найвищий рівень абстракції для даної ієрархії. Потім дане загальне поняття конкретизується, тим самим зменшується його об'єм і збільшується зміст G_{gi} (де *i* – номер рівня ієрархії). З'являється менш загальне поняття, яке на схемі ієрархії буде розташоване на рівень нижче обраного поняття. Цей процес конкретизації понять може продовжуватись доти, доки на самому нижньому рівні не буде отримане поняття, подальша конкретизація якого в даному контексті або неможлива, або недоцільна.

Отже, у якості текстового модуля використовуємо вікно з *k* слів, речень, параграфів, розділів або цілих документів, згідно з восьмирівневою моделлю [12]. Для конкретного КСС *профіль спільної зустрічальності* (ПСЗ) визначається як вектор спільної зустрічальності між цим КСС та кожним елементом, який апріорно належить до заданої множини КСС, створеної як множина *семантичних індексних характеристик* (СІХ).

Для множини *M* КСС будується сочастотна матриця $C = M \times P$, кожний рядок якої представляє собою ПСВ КСС t_i:

$$C = \begin{pmatrix} c_1 \\ c_2 \\ \vdots \\ c_M \end{pmatrix} = \begin{pmatrix} c_{11} & c_{12} & \cdots & c_{1M} \\ c_{21} & c_{22} & \cdots & c_{2M} \\ \vdots & \vdots & \ddots & \vdots \\ c_{M1} & c_{M2} & \cdots & c_{MP} \end{pmatrix} . \quad (3)$$

Весь документ D_n оцінюється за допомогою середньозваженого ПСВ для КСС і має вигляд:

$$D_n = \sum_{i=1}^{M} w_{ni} c_i . \quad (4)$$

Вага w_{ni} задана для кожного ПСВ c_i і має той же КСС t_i як було показано вище. Отже, ком-

[2] або *созустрічальність, сочастотність.*

плект документів представляється за допомогою добутку матриць $D = FC$.

Потрібно звернути увагу, що розмір вектора представлення Vr_i документу D_n – це розмір масиву СІХ, який може бути менше, ніж розмір масиву КСС. Проте ЗС має такий же розмір масиву КСС як і ПВМ, тому що всі КСС у масиві представлені середньозваженою величиною за допомогою їх ПСЗ в СІХ.

Модель ЗС має деяку перевагу, з точки зору оптимізації (скорочення) розмірності шляхом отримання об'єднаного вектора представлення Vc_i документу D_n через представлення КСС двох: спільної зустрічальності та СІХ.

Проблема вибору СІХ в ПВМ вирішується шляхом індексування масиву КСС на основі частоти їх зустрічальності в документі D_n. Для того, щоб не втратити, або не зашумлювати інформаційність КСС, що знаходиться в документі, будемо комбінувати два підходи РС і ПВМ.

Якщо F^* реструктурована середньозважена матриця тільки для КСС, які індексуються, а α скалярний комбінований параметр у діапазоні від 0 до 1, то комбіноване представлення документу D_n буде мати наступний вигляд:

$$D_n = (1-\alpha)\, F^* + \alpha FC. \qquad (5)$$

Уточнення розподіленої семантичної моделі

Зробимо деякі уточнення, які стосуються обчислення матриць КСС, які спільно зустрічаються, оскільки проста модель спільної зустрічальності (в основі якої лежить спільно зустрічні КСС без визначеного (будь-якого) текстового модуля з додатковими обмеженнями) може мати незадовільні результати обчислень. Розглянемо текстовий фрагмент: «*Потужний мікропроцесор швидко обробляє запити з робочих станцій»*

Візьмемо в якості текстового модуля ціле речення. Тоді КСС: *робоча станція*, *потужний мікропроцесор* з точки зору спільної зустрічальності є релевантними, тоді як *мікропроцесор запитів*, або *станція запитів* є ілюзорною (або найменш передбачуваною).

Ця проблема вирішується шляхом введення додаткової синтаксичної інформації в процес обчислень. Інакше кажучи, спільну зустрічальність потрібно вираховувати тільки:

1. Між КСС в однакових синтаксичних групах;
2. Між головними СКК в різних групах.

Зробимо уточнення для попереднього прикладу, в якому розглядалась спільна зустрічальність тільки між іменниками, дієсловами та прикметниками. У прикладі на малюнку нижче жирним курсивом виділені головні КСС у даній синтаксичній групі.

Підхід до верифікації фрагментів технічних документів у цільовій предметній області

У зв'язку з вищесказаним розглянемо методику визначення приналежності ТДТ (або його контексту) до конкретної проблемної області:

1. Для кожного з наявних документів отримуємо матрицю А частот парної зустрічальності.
2. Аналізуючи кожну матрицю, виділяємо для кожного тексту сукупність зв'язків з високими (тобто такими, що перевищують деяке порогове значення) частотами.
3. При порівнянні отриманої сукупності "істотні" зв'язків тексту, що досліджується, з іншими визначається відповідність, який текст характеризується найбільш близькою по деякому критерію сукупністю „істотних" семантико-синтаксичних зв'язків (ССЗ).

У тому випадку, коли передбачається введення в аналіз „загальну частину", або „ядро", в методику вбудовуються два додаткових пункти:

а) Порівнюючи отримані сукупності ССЗ, виділяємо загальномовне ядро, тобто набір таких зв'язків, які містяться в усіх (або майже усіх) текстах.

б) Сформоване "загальномовне ядро" видаляється з кожної сукупності відібраних ССЗ з високими частотами; "істотні" ССЗ кожної сукупності, що залишилися після цього уже більше характеризують приналежність до контексту.

Загальномовне ядро має різні рівні аналізу. При дослідженні приналежності одного документу – це найбільш характерні особливості його стилю. При дослідженні документів певної проблемної області, але різних типів документів – це риси, властиві перш за все проблемній області.

Висновки

В роботі були визначені основні підходи до логічної організації моделей текстових докуме-

нтів, реалізація яких в СПРТД створює передумови до зменшення часу розробки типових технічних документів, а також надає потенційну можливість підвищення рівня інтелектуалізації систем даного класу. Отримані результати можна виразити наступним чином:

1. Виділені компоненти моделі предметної області, які необхідно розробити, і визначені базові формалізми дослідження.

2. Запропоновані структурно-параметричні характеристики текстових документів, наведені їх обґрунтування і відповідність задачам формальної оцінки блоків ТДТ згідно з рівнем ієрархії.

3. Показано, що при умові виконання вимог вузькоспеціалізованого застосування, дані характеристики дозволяють отримати оцінки семантичної близькості, порівнянності і міру семантичної відповідності для структурних одиниць технічної природної мови.

Список литературы

1. Beeferman D., Berger A., Lafferty J. Statistical Models for Text Segmentation // Machine learning. – 1999. – Vol. 34. – P. 1-34.
2. Berry M.W., Dumais S.T., O'Brein G.W. Using Linear Algebra for Intelligent Information Retrieval // SIAM Review. – 1995. – Vol. 37, № 4, P. 573-595.
3. Rijsbergen C.J. van. Information Retrieval, Butterworths // London. – 1979. – 358 p.
4. Yang Y. An Evaluation of Statistical Approaches to Text Catigorization // 1999. –Vol. 1, № 1. – P. 69-90.
5. Salton G., Buckley C. Term weighting approaches in automatic text retrieval // Information Processing and Management. – 1988. – № 24. – P. 513-523.
6. Salton G., Yang C., and Yu C. A theory of term importance in automatic text analysis // Journal of the American Society for Information Science. – 1975. – №12. – P. 248-251.
7. Стиренко С.Г., Пикуза О.В. Оценка стилистических характеристик текстовой информации // Вісник НТУУ «КПІ». Інформатика, управління та обчислювальна техніка: Зб. наук. пр. – К.: Век+. – 2000. – № 35. – С. 127-131.
8. Lee L. Similarity–Based Approaches to Natural Language Processing. PhD thesis, Harvard University. – 1997.
9. Rajman M., L. Lebart. Similarités pour données textuelles // In 4th International Conference on Statistical Analysis of Textual Data (JADT'98), Nice. 1998. – P. 336-342.
10. Фукс В. По всем правилам искусства (точные методы в исследованиях литературы, музыки и изобразительного искусства. Искусство и ЭВМ. М.: Мир, 1975. – 320 с.
11. Lebanon G. Metric Learning for Text Documents // IEEE Transaction on Pattern and Machine Intelligence. – 2006. – Vol. 28, № 4. – P. 497-508.
12. Стиренко С.Г. Автоматизована розробка технічної документації з використанням засобів інтелектуальної підтримки // Вісник НТУУ «КПІ». Інформатика, управління та обчислювальна техніка: Зб. наук. пр. - К.: Век+, – 2006. – № 45. – С. 173-179.

УДК 519.5:681:513

ПАВЛОВ А.А.,
КУТ В.И.,
ШТАНЬКЕВИЧ А.С.

НАХОЖДЕНИЕ ВЕСОВ ПО МАТРИЦЕ ПАРНЫХ СРАВНЕНИЙ С ОДНОСТОРОННИМИ ОГРАНИЧЕНИЯМИ

Предлагаются и обосновываются модели оптимизации для нахождения весов объектов в методе анализа иерархий по матрице парных сравнений с односторонними ограничениями.

The optimization models for objects' priorities searching in the Analytic Hierarchy Process (AHP) using pair-wise comparision matrix with "one-side" boundaries are proposed and substantiated.

Ключевой проблемой при решении задачи многокритериального выбора с помощью метода анализа иерархий является нахождение весов $w_1,...w_n$ объектов $A_1,...A_n$ (альтернатив, критериев) по отношению к некоторому свойству, цели (критерию). Веса определяются по эмпирической матрице парных сравнений $(\gamma_{ij})_1^n$ задаваемой экспертом (экспертами). Число γ_{ij} задается экспертом и показывает, по мнению эксперта, во сколько раз вес объекта A_i больше веса объекта A_j по отношению к заданной цели (критерию).

В [2, 3] были рассмотрены более общие случаи матрицы парных сравнений, предложены и обоснованы методы нахождения по ним весов объектов. В данной статье исследуется случай, когда произвольный элемент матрицы парных сравнений имеет вид:

$$\frac{w_i}{w_j} \ge \gamma_{ij} \text{ или } \frac{w_i}{w_j} \le \gamma_{ij}, \qquad (1)$$

где $w_i, i = \overline{1,n}$ неизвестные веса объектов.

Такая постановка задачи порождается стандартной, когда матрицы парных сравнений задают интервальные оценки

$$\underline{\gamma_{ij}} \le \frac{w_i}{w_j} \le \overline{\gamma_{ij}} \qquad (2)$$

Действительно, если у эксперта (экспертов) существует дополнительная информация о том, что при фиксированных i, j $\frac{w_i}{w_j}$ скорее всего принимает значение ближе к $\underline{\gamma_{ij}}$ либо к $\overline{\gamma_{ij}}$, то эмпирическую матрицу парных сравнений (2) можно заменить эмпирической матрицей парных сравнений (1).

Исходя из логики обоснования односторонних ограничений можно сформулировать следующие функционалы качества, оптимальному значению которых должны соответствовать значения весов объектов: искомые веса объектов w_i должны быть такими, чтобы односторонняя мера (интегральная мера) отклонения $\frac{w_i}{w_j}$ от γ_{ij} была минимальной.

В [1] показано, что в качестве меры отклонения $\frac{w_i}{w_j}$ от эмпирического γ_{ij} логично выбрать

$$\frac{1}{\gamma_{ij}}\left|\frac{w_i}{w_j} - \gamma_{ij}\right| \qquad (3)$$

В [1, 2,] показано, что непосредственное использование этой меры может приводить к задачам нелинейного невыпуклого программирования и в связи с этим вводились меры формально отличные от (3), приводящие к задачам линейного либо выпуклого программирования. В данной статье будут рассматриваться модели оптимизации, основанные только на мере (3), построение которых логично вытекает из моделей (6)–(7), (8)–(9) [3].

Рассмотрим два частных случая.

1) $\frac{w_i}{w_j} \ge \gamma_{ij}$ для $\forall (ij) \in |A|$,

где A множество всех элементов матрицы парных сравнений удовлетворяющих условиям:
$i \ne j, \gamma_{ij} \ge 1,$
$i, j = \overline{1,n}$;
$|A|$ – соответствующее A множество пар индексов (ij)

Для нахождения весов объектов по матрице парных сравнений построим следующие модели оптимизации:

Модель 1.

$$\min_{\substack{y_{ij}\\ \forall(i,j)\in|A|}} \sum_{(ij)\in|A|}\sum \ell n(1+y_{ij}) \qquad (4)$$

$$1\le w_i\le b,\quad i=\overline{1,n}$$

$$\frac{w_i}{w_j}=\gamma_{ij}\left(1+y_{ij}\right) \qquad (5)$$

$$0\le y_{ij}\le b-1,\quad \forall\left(i,j\right)\in|A|$$

Переменными задачи являются $\forall(ij)\in|A|\ \ y_{ij},\ w_i,\ \ i=\overline{1,n}.$

Модель (4)–(5) является аппроксимационной для оптимизационной задачи:

$$\min_{\substack{y_{ij}\\ \forall(i,j)\in|A|}} \sum_{(ij)\in|A|}\sum y_{ij} \qquad (6)$$

$$\frac{w_i}{w_j}=\gamma_{ij}\left(1+y_{ij}\right) \qquad (7)$$

$$\forall\left(i,j\right)\in|A|\quad y_{ij}\ge 0.$$

$$1\le w_{ij}\le b,\quad i=\overline{1,n}$$

Переменные задач (4)–(5) и (6)–(7) совпадают.

Модель (6)–(7) является задачей нелинейного невыпуклого программирования, а модель 1 эквивалентна следующей задаче линейного программирования:

Модель 2.

$$\min \sum_{(ij)\in|A|}\sum \Delta_{ij}^1 \qquad (8)$$

$$W_i-W_j=\ell n\gamma_{ij}+\Delta_{ij}^1\quad \forall_{ij}\in|A|$$

$$0\le\Delta_{ij}^1\le\ell nb\quad \forall_{ij}\in|A| \qquad (9)$$

$$0\le W_i\le\ell nb\quad i=\overline{1,n}$$

Переменные задачи $W_i,\quad i=\overline{1,n},\quad \Delta_{ij}^1,\quad \forall\left(ij\right)\in|A|$

Оптимальные решения задач (4)–(5) и (8)–(9) совпадают.

Переменные w_i задачи (4)–(5) и W_i задачи (8)–(9) связаны соотношением $w_i=e^{W_i},\quad i=\overline{1,n}$.

Рассмотрим следующую задачу оптимизации:

Модель 3.

$$\min_y \ell n\left(1+y\right) \qquad (10)$$

$$\gamma_{ij}\le\frac{w_i}{w_j}\le\gamma_{ij}\left(1+y\right)$$

$$\forall\left(ij\right)\in|A|,\quad y\ge 0 \qquad (11)$$

$$1\le w_i\le b,\quad i=\overline{1,n}$$

Модель 3 эквивалентна следующей задаче линейного программирования:

Модель 4.

$$\min_{\Delta}\Delta \qquad (12)$$

$$\ell n\,\gamma_{ij}\le W_i-W_j\le\ell n\,\gamma_{ij}+\Delta\quad \forall\left(ij\right)\in|A| \qquad (13)$$

$$0\le\Delta\le\ell n\,b,$$

$$0\le W_i\le\ell n\,b,\quad i=\overline{1,n}$$

Переменным задачи является $\Delta,\ W_i,\ \ i=\overline{1,n}$.

Переменные задач (10)–(11) и (12)–(13) связаны соотношениями:

$w_i=e^{W_i},\quad i=\overline{1,n},\quad y=e^{\Delta}-1.$

В отличие от задачи (4)–(5), оптимальному решению задачи (10)–(11) соответствуют такие веса объектов, у которых достигается минимум выражения:

$$\max_{(ij)\in|A|}\frac{\left|\frac{w_i}{w_j}-\gamma_{ij}\right|}{\gamma_{ij}}$$

Тогда вместо решения задачи линейного программирования (8)–(9) веса объектов можно находить как оптимальное решение следующей задачи линейного программирования:

Модель 5.

$$\min \sum_{(ij)\in|A|}\sum \Delta_{ij}^1 \qquad (14)$$

$$W_i-W_j=\ell n\,\gamma_{ij}+\Delta_{ij}^1$$

$$0\le\Delta_{ij}^1\le\Delta^* \qquad (15)$$

$$0\le W_i\le\ell n\,b,\quad i=\overline{1,n},$$

Δ^* – оптимальное решение задачи (12)–(13)

2) $\frac{w_i}{w_j}\le\gamma_{ij}\quad \forall\left(ij\right)\in|A|$

Веса находятся, как оптимальное решение следующей оптимизационной задачи:

Модель 6.

$$\min_{\substack{z_{ij}\\ \forall(ij)\in|A|}} \sum_{(ij)\in|A|}\sum \ell n\frac{1}{1-z_{ij}} \qquad (16)$$

$$1\le w_i\le b,\quad i=\overline{1,n}$$

$$\frac{w_i}{w_j}=\gamma_{ij}\left(1-z_{ij}\right),\quad 0\le z_{ij}\le 1-\frac{1}{b^2} \qquad (17)$$

$$\forall\left(ij\right)\in|A|$$

Переменные задачи $z_{ij},\ \ ij\in|A|,\ \ w_i,\ \ i=\overline{1,n}$.

Модель 6 эквивалентна следующей задаче линейного программирования:

Модель 7.

$$\max_{\substack{\Delta_{ij}^2\\ (ij)\in|A|}} \sum_{(ij)\in|A|}\sum \Delta_{ij}^2 \qquad (18)$$

$$W_i - W_j = \ell n\, \gamma_{ij} + \Delta_{ij}^2 \quad (ij) \in |A| \tag{19}$$

$$\ell n\left(\frac{1}{b^2}\right) \le \Delta_{ij}^2 \le 0 \quad 0 \le W_i \le \ell n\, b.$$

$$(ij) \in |A| \qquad i = \overline{1,n}.$$

Переменными задачи являются Δ_{ij}^2, $(ij) \in |A|$ W_i, $i = \overline{1,n}$.

Связь между переменными задач (14)–(15) и (16)–(17) является следующей:

$w_i = e^{W_i}$, $i = \overline{1,n}$.

Модель 7 является аппроксимационной для следующей нелинейной оптимизационной задачи:

$$\min \sum_{(ij) \in |A|} \sum z_{ij}$$

$$1 \le w_i \le b, \quad i = \overline{1,n}$$

$$\frac{w_i}{w_j} = \gamma_{ij}\left(1 - z_{ij}\right), \quad 0 \le z_{ij} \le 1 - \frac{1}{b^2}$$

Переменными являются z_{ij}, $(ij) \in |A|$, w_i, $i = \overline{1,n}$

Для случая $\frac{w_i}{w_j} \le \gamma_{ij}$, $(ij) \in |A|$ аналогом модели (10) – (11) является следующая:

Модель 8.

$$\min_z \ \ell n \frac{1}{1-z} \tag{20}$$

$$1 \le w_i \le b, \quad i = \overline{1,n}$$

$$\gamma_{ij}(1-z) \le \frac{w_i}{w_j} \le \gamma_{ij} \quad \forall (ij) \in |A| \tag{21}$$

$$0 \le z \le 1 - \frac{1}{b^2}$$

Переменными являются z, w_i, $i = \overline{1,n}$.

Задача (20) – (21) эквивалентна следующей задаче линейного программирования:

Модель 9.

$$\max \Delta \tag{22}$$

$$\ell n\, \gamma_{ij} + \Delta \le W_i - W_j \le \ell n\, \gamma_{ij}, \ \forall (ij) \in |A| \tag{23}$$

$$\ell n \frac{1}{b^2} \le \Delta \le 0, \ 0 \le W_i \le \ell n\, b \quad i = \overline{1,n}$$

Аналогом задачи оптимизации (14)–(15) для данного случая является следующая задача линейного программирования:

Модель 10.

$$\max_{\Delta_{ij}^2} \sum_{(ij) \in |A|} \sum \Delta_{ij}^2 \tag{24}$$

$$W_i - W_j = \ell n\, \gamma_{ij} + \Delta_{ij}^2, \ (ij) \in |A| \tag{25}$$

$$\Delta^* \le \Delta_{ij}^2 \le 0, \quad 0 \le W_i \le \ell n\, b, \quad i = \overline{1,n},$$

где Δ^* – оптимальное решение задачи (22) – (23).

3) Общий случай.

Множество $|A|$ разбивается на два непересекающихся подмножества $|A| = |A_1| \cup |A_2|$ по следующему правилу:

$\forall (ij) \in |A_1|$ выполняется $\frac{w_i}{w_j} \ge \gamma_{ij}$;

$\forall (ij) \in |A_2|$ выполняется $\frac{w_i}{w_j} \le \gamma_{ij}$.

В этом случае в качестве модели оптимизации используем модификацию модели (6)–(7) [3]:

Модель 11.

$$\min_{\substack{y_{ij} \\ (ij) \in |A_1|}} \ \min_{\substack{z_{ij} \\ \forall (ij) \in |A_2|}} \left(\sum_{(ij) \in |A_1|} \sum \ell n\left(1 + y_{ij}\right) + C_{\ell m} \sum_{(ij) \in |A_2|} \sum \ell n \frac{1}{1 - z_{ij}} \right) \tag{26}$$

$$1 \le w_i \le b, \quad i = \overline{1,n}$$

$$\frac{w_i}{w_j} = \gamma_{ij}\left(1 + y_{ij}\right) \quad \forall (ij) \in |A_1| \tag{27}$$

$$0 \le y_{ij} \le \ell \ \Delta_1(\ell) \le b - 1;$$

$$\frac{w_i}{w_j} = \gamma_{ij}\left(1 - z_{ij}\right) \quad \forall (ij) \in |A_2|$$

$$0 \le z_{ij} \le m \ \Delta_2(m) \le 1 - \frac{1}{b^2}$$

ℓ, m – последовательно возрастающие числа, $\Delta_1(\ell) > 0$, $\Delta_2(m) > 0$ – заданы.

Коэффициенты $C_{\ell m}$ находятся из равенства:

$$\ell n\left(1 + \ell \ \Delta_1(\ell)\right) = C_{\ell m} \ \ell n \frac{1}{1 - m \ \Delta_2(m)}.$$

Последовательность задач оптимизации (26) – (27) эквивалентна следующей последовательности задач линейного программирования:

Модель 12.

$$\min_{\substack{\Delta_{ij}^1 \\ \forall (ij) \in |A_1|}}, \ \min_{\substack{\Delta_{ij}^2 \\ \forall (ij) \in |A_2|}} \left(\sum_{(ij) \in |A_1|} \sum \Delta_{ij}^1 - C_{\ell m} \sum_{(ij) \in |A_2|} \Delta_{ij}^2 \right) \tag{28}$$

$$
\begin{gathered}
W_i - W_j = \ell n\, \gamma_{ij} + \Delta_{ij}^1, \ \forall (ij) \in |A_1| \\
W_i - W_j = \ell n\, \gamma_{ij} + \Delta_{ij}^2, \ \forall (ij) \in |A_2| \\
0 \le \Delta_{ij}^1 \le \ell n\left(1 + \ell\ \Delta_1(\ell)\right); \quad \forall (ij) \in |A_1| \\
\ell n\left(1 - m\ \Delta_2(m)\right) \le \Delta_{ij} \le 0, \quad \forall (ij) \in |A_2| \\
0 \le W_i \le \ell n\, b, \quad i = \overline{1,n} \\
0 \le \ell\ \Delta_1(\ell) \le b - 1; \quad 0 \le m\ \Delta_2(m) \le 1 - \frac{1}{b^2}
\end{gathered}
\tag{29}
$$

ℓ, m – последовательно возрастающие натуральные числа.

Связь между переменными w_i и W_i очевидна $w_i = e^{W_i}, \quad i = \overline{1,n}.$

Заключение

Предложены и обоснованы модели оптимизации для нахождения весов объектов по матрицам парных сравнений, элементы которой являются односторонними неравенствами.

Список литературы

1. А.А. Павлов, Е.И. Лищук, В.И. Кут. Математические модели оптимизации для обоснования и нахождения весов в методе парных сравнений. Системні дослідження та інформаційні технології. №2, 2007р.
2. А.А. Павлов, В.И. Кут. Математические модели оптимизации для обоснования и нахождения весов объектов по неоднородным матрицам парных сравнений. Системні дослідження та інформаційні технології. №3, 2007р.
3. А.А. Павлов, Е.И. Лищук, В.И. Кут. Многокритериальный выбор в задаче обработки данных матрицы парных сравнений. Вісник НТУУ „КПІ”. Інформатика, управління та обчислювальна техніка. №46, 2008р.

УДК 62-50

ТЕЛЕНИК С.Ф.,
БІДЮК П.І.,
АМОНС О.А.,
КРИЖОВА К.О.

МЕТОД РОЗПОДІЛУ РЕСУРСІВ МІЖ ПРОЕКТАМИ

Робота присвячена розробці методу розв'язання задачі розподілу ресурсів між проектами, що ґрунтується на використання результатів теорії нечітких множин та цілочислового математичного програмування. Такий підхід відрізняється від відомих тим, що дає можливість врахувати суб'єктивні оцінки експертів та об'єктивні (фактичні) обмеження на фонди, що спрямовуються на реалізацію проектів.

This work is devoted to develop method of decision task of allocation resources between projects. This method based on using results of fuzzy set theory and integer mathematical programming. Such approach is differed from other famous theory by giving opportunity to take into account expert subjects marks objects (actual) restricts of funds, directed on realization of projects.

Задача розподілу ресурсів є предметом наукових досліджень на протязі останніх декількох десятиліть. Для розв'язку таких задач застосовують, в основному, методи дослідження операцій з обмеженнями різних типів [1]. Однак, застосування виключно «чітких» методів до подібних задач, як правило, не дає можливості враховувати більшість факторів, що впливають на розподіл ресурсів. Їх можна розглядати як складні багатокритеріальні задачі ієрархічного типу, для яких є характерною наявність ситуаційних та конфліктних невизначеностей [2].

Особливого значення ця проблема набуває в системах управління бізнес-процесами в організаційно-технологічних об'єктах управління, які функціонують на основі ресурсного підходу [3]. Також, розроблено декілька детермінованих та стохастичних моделей управління розподілом ресурсів в інформаційно-телекомунікаційних мережах [4,5,6], використання яких стримується відсутністю моделей оцінки важливості бізнес-процесів.

Задачу вибору проекта для фінансування в умовах обмеженості фінансових засобів будемо розглядати як задачу багатокритеріального прийняття рішень (БКПР). При постановці та розв'язку задачі вводиться бюджетне обмеження (ЕС0) та чотири додаткових обмеження – відносно пріоритетності проекту (ЕС1), типів проектів (ЕС2), розмірів проектів (ЕС3) та виконавців проектів (ЕС4). Кожна з умов досягнення рівноваги між розподілом бюджетних засобів {ЕС0, ЕС1, ЕС3, ЕС4} призначена для врахування (збалансовування) конкретних вимог і використовується для розподілу множини можливих проектів на кілька підмножин (підгруп). При цьому передбачається, що з кожної підгрупи буде відібрана для реалізації деяке фіксоване число проектів.

Процес вибору проектів ускладнюється також тим, що він має деякі нечіткі характеристики. Оскільки прийняття рішення щодо фінансування проектів, як правило, виконується членами експертних груп, то для кожного з них характерною є наявність суб'єктивності, неточність та невизначеність суджень, невизначеність та необгрунтованість у присвоєнні відносних вагових коефіцієнтів критеріям вибору. Дуже часто особа, що приймає рішення (ОПР), присвоює вагові коефіцієнти критеріям у вигляді лінгвістичних змінних, таких як "хороший", "дуже низький", "слабкий" і тому подібне. А тому проблема вибору проектів для фінансування складна і характеризується наявністю невизначеностей вже за своєю природою. Існують детерміновані підходи до вибору проектів на основі точних характеристик, але вони мають занадто загальний характер і не враховують особливості ситуації та вибору в умовах суттєвої обмеженості бюджетних ресурсів, а також тих невизначеностей, які зустрічаються при розв'язку реальних задач.

Багатокритеріальне нечітке оцінювання якості проектів. Для того щоб оцінити якість кожного проекту за допомогою вибраних критеріїв вибору, розглянемо наступні аспекти, що пов'язані із вибором: взаємозв'язки між різними критеріями; ранжування проектів за різними критеріями вибору; вагові коефіцієн-

ти для кожної альтернативи; агрегування рангів кожного кожного проекту з ваговими коефіцієнтами критерію. Використовуючи отриманий таким чином агрегований показник, далі можна обчислити індекс значимості проекту (ІЗП), який є інтегральним показником корисності його реалізації.

Взаємозв'язки між різними критеріями вибору можна найкраще показати на прикладі ієрархічної структури. На першому рівні визначається нечітка оцінка важливості суб'єктивних критеріїв (наприклад, необхідність та терміновість виконання проекту, принципова можливість реалізації проекту, корисність проекту та можливості отримання достатнього і реального фінансування). На другому рівні проекту присвоюється рейтинг у відповідності до кожного суб'єктивного критерію. Такий рейтинг можна присвоювати проектам за семизначною шкалою лінгвістичних змінних, наприклад, {надзвичайно хороший, дуже хороший, середній, нижче середнього, поганий, дуже поганий}.

Для визначення ступеня нечіткості індивідуальних суджень кожної ОПР щодо відносної важливості критеріїв скористаємось 5-значною шкалою лінгвістичної змінної – "Важливість": Важливість = {дуже важливий, важливий, середньої важливості, не важливий, дуже низької важливості}. Використовуючи таку модель, можна прийняти до уваги нечіткість ОПР при визначенні рейтингу, а також визначити альтернативу між різними критеріями оцінки в процесі агрегування. Все це сприяє структуризації процесу прийняття рішень та покращенню якості рішень. Введемо деякі визначення та поняття, які будуть використані при побудові алгоритму розподілу ресурсів між проектами.

Означення 1. Нехай X – область визначення деякої змінної. Нечітка підмножина $\tilde{A}$, що належить X, визначається функцією належності (ФН) $f_{\tilde{A}} : X \to [0,1]$. Значення функції $f_{\tilde{A}}(x)$ представляє собою значення належності величини x множині $\tilde{A}$.

Означення 2. Нечітке число $\tilde{A}$ є нечіткою підмножиною дійсної осі R, функція належності якої $f_{\tilde{A}}(x)$ має наступні характеристики:

$$f_{\tilde{A}}(x) = \begin{cases} f_{\tilde{A}}^{L}(x), & \alpha \le x \le \beta, \\ 1, & \beta \le x \le \gamma, \\ f_{\tilde{A}}^{R}(x), & \gamma \le x \le \delta, \\ 0, & \textit{інакше}, \end{cases} \qquad (1)$$

$-\infty \le \alpha \le \beta \le \gamma \le \delta \le \infty$;

де $f_{\tilde{A}}^{L}(x) : [\alpha, \beta] \to [0,1]$ – неперервна та строго зростаюча функція на $[\alpha, \beta]$, а $f_{\tilde{A}}^{R}(x) : [\gamma, \delta] \to [0,1]$ – неперервна та строго зростаюча функція на $[\gamma, \delta]$.

Позначимо через $\Im(R)$ множину всіх нечітких чисел. Тепер можна визначити (побудувати) багато різних функцій належності для нечітких чисел, які мають наведені вище характеристики. Окремим випадком є трикутна ФН.

Означення 3. Нечітке число $\tilde{a}$ з трикутною функцією належності є нечітким числом на осі **R** з ФН виду: $f_{\tilde{A}} : \mathbf{R} \to [0,1]$. Воно параметризується трійкою чисел: $\tilde{a} = (a_l, a, a_r)$, тобто

$$f_{\tilde{a}}(x) = \begin{cases} (x-a)/(a-a_1), & a_l \le x \le a, \\ (x-a_r)/(a-a_r), & a \le x \le a_r, \\ 0, & \textit{інакше}, \end{cases}$$

де $a_l \le a \le a_r$ – дійсні числа.

Призначення параметрів трійки $\tilde{a} = (a_l, a, a_r)$ наступні: параметр a визначає максимально можливий ступінь належності $f_{\tilde{a}}(x)$, тобто $f_{\tilde{a}}(a) = 1$; це найбільш можливе значення даних, що оцінюються. Параметри a_l, a_r представляють собою ліву і праву границю діапазону, в який може попасти величина, значення якої оцінюється. Так, для точного значення числа a можна записати, що $\tilde{a} = (a, a, a)$. Надалі будемо використовувати нечіткі числа з трикутною функцією належності для описування суб'єктивних оцінок, які дають ОПР. Причиною використання нечітких чисел з трикутною ФН є те, що ОПР їх інтуїтивно використовують на практиці [7, 8].

Арифметичні операції над нечіткими числами. Користуючись принципом розширення, можна записати, що арифметична операція $\tilde{C} = \tilde{A} * \tilde{B}$ над двома нечіткими числами $\tilde{A}$ і $\tilde{B}$ представляє собою нечітке число, функція належності якого $f_{\tilde{C}}(x)$ визначається як

$$f_{\tilde{C}}(z) = f_{\tilde{A}*\tilde{B}}(z) = \sup_{z=x*y} \min\{f_{\tilde{A}}(x),\ f_{\tilde{B}}(y)\},$$

де символ "*" визначає арифметичну операцію $+, -, \times$ або $\div$.

Нехай $\tilde{a} = (a_l, a, a_r)$ і $\tilde{b} = (b_l, b, b_r)$ – два нечітких числа. Відомо, що результатом нечіткого додавання є $\tilde{a} \oplus \tilde{b} = (a_l + b_l,\ a + b,\ a_r + b_r)$, нечіткого віднімання є $\tilde{a} - \tilde{b} = (a_l - b_l,\ a - b,\ a_r - b_r)$, а результатом множення коефіцієнта k на нечітке число $\tilde{a}$ є $k \otimes \tilde{a} = (ka_l, ka, ka_r)$ для $\forall k \in R^+$. Всі отримані результати арифметичних операцій також будуть нечіткими числами з трикутними ФН. Однак, у випадку коли $a_l > 0$ і $b_l > 0$, маємо: $\tilde{a} \otimes \tilde{b} \approx (a_l b_l,\ ab,\ a_r b_r)$ і $\tilde{a} \div \tilde{b} \approx (a_l / b_r,\ a/b,\ a_r / b_l)$, тобто, результатами множення і ділення двох нечітких чисел з трикутною функцією належності є наближені нечіткі числа з трикутною функцією належності.

Лінгвістичні змінні. Нечіткими лінгвістичними змінними називають такі, значеннями яких є слова або фрази повсякденної або синтезованої мови. Наприклад, характеристику (якість) проекту можна визначити за допомогою змінних "*Якість*" та "*Важливість*", які є лінгвістичними, а не числовими. Лінгвістичні змінні досить часто характеризують саме трикутними функціями належності. Якщо ввести позначення *Якість*=**S**, то у відповідності до наведеного вище переліку значень змінної *Якість* можна записати, що S={*EG, VG, G, M, P, VP, EP*}, де *EG=надзвичайно хороший; VG=дуже хороший; G=хороший; M=середній; P=нижче середнього; VP=поганий; EP=дуже поганий.* Якщо взяти за основу десятибальну шкалу, то лінгвістичну змінну **S** можна характеризувати в діапазоні [0,10] нечіткими числами з трикутною функцією функцією належності наступним чином: *EG*=(9.5;10;10), *VG*=(7;8.5;10), *G*=(5.5;7;8.5), *M*=(3.5;5;6.5), *P*=(1.5;3;4.5), *VP*=(0;1.5;3), *EP*=(0;0;0.5). Функції належності можна записати у вигляді:

EG=(9.5;10;10):

$$f_{EG}(x) = \begin{cases} 2x - 19, & 9{,}5 \le x \le 10, \\ 0, & 0 \le x \le 9{,}5; \end{cases}$$

VG=(7;8.5;10):

$$f_{VG}(x) = \begin{cases} \frac{1}{3}(2x - 14), & 7 \le x \le 8{,}5, \\ \frac{2}{3}(10 - x), & 8{,}5 \le x \le 10, \\ 0, & \textit{інакше}, \end{cases}$$

G=(5.5;7;8.5):

$$f_{G}(x) = \begin{cases} \frac{1}{3}(2x - 11), & 5{,}5 \le x \le 7, \\ \frac{1}{3}(17 - 2x), & 7 < x \le 8{,}5, \\ 0, & \textit{інакше}, \end{cases}$$

M=(3.5;5;6.5):

$$f_{M}(x) = \begin{cases} \frac{1}{3}(2x - 7), & 3{,}5 \le x \le 5, \\ \frac{1}{3}(13 - 2x), & 5 < x \le 6{,}5, \\ 0, & \textit{інакше}, \end{cases}$$

P=(1.5;3;4.5):

$$f_{P}(x) = \begin{cases} \frac{1}{3}(2x - 6), & 1{,}5 \le x \le 3, \\ \frac{1}{3}(9 - 2x), & 3 < x \le 4{,}5, \\ 0, & \textit{інакше}, \end{cases}$$

VP=(0;1.5;3):

$$f_{VP}(x) = \begin{cases} \frac{2}{3}x, & 0 \le x \le 1{,}5, \\ \frac{1}{3}(6 - 2x), & 1{,}5 < x \le 3, \\ 0, & \textit{інакше}, \end{cases}$$

EP=(0;0;0.5):

$$f_{EP}(x) = \begin{cases} 1 - 20x, & 0 \le x \le 0{,}5, \\ 0, & 0 < x \le 10. \end{cases}$$

Шкалу лінгвістичної змінної "*Важливість проекту*" розділимо на 5 значень: **W**={*VI, I, F, UI, VUI*}, де *VI* =(*дуже важливий*), *I* = (*важливий*), *F* = (*середньої важливості*), *UI* = (*не важливий*), *UI* = (*дуже низької важливості*). Значення лінгвістичної змінної **W** також будемо характеризувати функціями належності трикутного виду на інтервалі [0,10], тобто: *VI*=(8;10;10), *I*=(5;7;9), *F*=(3;5;7), *UI*=(1;3;5), *VUI*=(0;0;2). Значення цієї змінної будуть використані для визначення величин вагових коефіцієнтів (якісних) суб'єктивних критеріїв вибору, які використо-

вуються особами, що приймають рішення. Самі функції належності визначаються як
VI=(8;10;10):

$$f_{VI}(x)=\begin{cases}\frac{1}{2}(x-8), & 8\le x\le 10,\\ 0, & 0<x<8,\end{cases}$$

I=(5;7;9):

$$f_{I}(x)=\begin{cases}\frac{1}{2}(x-5), & 5\le x\le 7,\\ \frac{1}{2}(9-x), & 7<x\le 9,\\ 0, & \textit{інакше},\end{cases}$$

F=(3;5;7):

$$f_{F}(x)=\begin{cases}\frac{1}{2}(x-3), & 3\le x\le 5,\\ \frac{1}{2}(7-x), & 5<x\le 7,\\ 0, & \textit{інакше},\end{cases}$$

UI=(1;3;5):

$$f_{UI}(x)=\begin{cases}\frac{1}{2}(x-1), & 1\le x\le 3,\\ \frac{1}{2}(5-x), & 3<x\le 5,\\ 0, & \textit{інакше},\end{cases}$$

VUI=(0;0;2):

$$f_{VUI}(x)=\begin{cases}\frac{1}{2}(2-x), & 0\le x\le 2,\\ 0, & 2<x<10\end{cases}.$$

Лінгвістичне значення вагового коефіцієнта можна отримати шляхом опитування ОПР з використанням шкали ранжування. Оцінку якості кожного проекта за допомогою ОПР відносно різних критеріїв можна отримати шляхом простого ранжування.

Агрегування рішень (оцінок) осіб, що приймають рішення. Припустимо, що вже є нечіткі оцінки від ОПР щодо якості та важливості проектів по відношенню до вибраних критеріїв і ці оцінки необхідно об'єднати в загальний агрегований нечіткий індекс значимості проекту, який буде служити узагальненою мірою необхідності виконання та важливості кожного альтернативного проекту. Одним із простих (але оправданих) підходів до агрегування є усереднення набору оцінок. Для реалізації цього методу наобхідно скористатися нечітким оператором знаходження середнього.

Розглянемо задачу вибору проекту із *n* альтернативних варіантів, в розв'язку якої приймають участь *k* ОПР за допомогою *m* критеріїв. Нехай $S_{ijt}=(s_{ijt}^{(l)},\, s_{ijt},\, s_{ijt}^{(r)})\in \mathbf{S}$, $i=1,2,...,n;\quad j=1,2,...,m;\quad t=1,2,...,k$ – це лінгвістичний рейтинг, який присвоєний особою, що приймає рішення, D_t проекту P_i за критерієм C_j. Також покладемо, що $W_{jt}=(w_{jt}^{(l)},\, w_{jt},\, w_{jt}^{(r)}),\quad j=1,2,...,m;\quad t=1,2,...,k$ – є лінгвістичним ваговим коефіцієнтом, який присвоєний ОПР D_t за критерієм C_j. Введемо також агреговані змінні

$$\overline{S}_{ij}=\frac{1}{k}\otimes(S_{ij1}\oplus S_{ij2}\oplus ...\oplus S_{ijk}),$$

$$\overline{W}_{j}=\frac{1}{k}\otimes(W_{j1}\oplus W_{j2}\oplus ...\oplus W_{jk}),$$

де символами $\otimes$ і $\oplus$ – позначені операції нечіткого множення на коефіцієнт (шкалювання) та додавання, відповідно. При вибраних позначеннях змінна $\overline{S}_{ij}$ представляє собою усереднений нечіткий рейтинг (оцінку) проекту(P_i за суб'єктивним критерієм C_j, а $\overline{W}_j$ – це усереднене значення нечіткого вагового коефіцієнта важливості суб'єктивного критерія C_j. Змінні $\overline{S}_{ij}$ і $\overline{W}_j$ також є нечіткими числами з трикутною функцією належності наступного виду [2]:

$$\overline{S}_{ij}=(\overline{s}_{ij}^{(l)},\, \overline{s}_{ij},\, \overline{s}_{ij}^{(r)}),\quad \overline{W}_{j}=(\overline{w}_{j}^{(l)},\, \overline{w}_{j},\, \overline{w}_{j}^{(r)}),$$

де

$$\overline{s}_{ij}^{(l)}=\frac{1}{k}\sum_{t=1}^{k}s_{ijt}^{(l)},\quad \overline{s}_{ij}=\frac{1}{k}\sum_{t=1}^{k}s_{ijt},\quad \overline{s}_{ij}^{(r)}=\frac{1}{k}\sum_{t=1}^{k}s_{ijt}^{(r)},$$

$$\overline{w}_{j}^{(l)}=\frac{1}{k}\sum_{t=1}^{k}w_{jt}^{(l)},\quad \overline{w}_{j}=\frac{1}{k}\sum_{t=1}^{k}w_{jt},\quad \overline{w}_{j}^{(r)}=\frac{1}{k}\sum_{t=1}^{k}w_{jt}^{(r)}\quad .$$

В загальному випадку вагові коефіцієнти критеріїв повинні бути нормованими з використанням операцій нечіткого додавання та нечіткого ділення. Нормовані значення знайдемо як

$$\overline{W}_{jN}=\overline{W}_{j}\otimes\left(\sum_{k=1}^{m}\overline{W}_{k}\right)\approx$$

$$\approx\left(\left(\frac{\overline{w}_{j}^{(l)}}{\sum_{k=1}^{m}\overline{w}_{j}^{(r)}}\right),\left(\frac{\overline{w}_{j}}{\sum_{k=1}^{m}\overline{w}_{j}}\right),\left(\frac{\overline{w}_{j}^{(r)}}{\sum_{k=1}^{m}\overline{w}_{j}^{(l)}}\right)\right)= \qquad (2)$$

$$=(\overline{w}_{j}^{(l)},\ \overline{w}_{j},\ \overline{w}_{j}^{(r)})$$

Нечіткий індекс значимості проекту $ІЗП_i$ для проекту P_i можна отримати шляхом усереднення добутків рангів критеріїв та відповідних вагових коефіцієнтів, тобто:

$$ІЗП_i = \frac{1}{m} \otimes \left[\left(\overline{W}_1 \otimes \overline{S}_{i1} \right) \oplus \left(\overline{W}_2 \otimes \overline{S}_{i2} \right) \oplus \ \ldots \ \oplus \left(\overline{W}_m \otimes \overline{S}_{im} \right) \right]$$

У відповідності до принципу розширення $ІЗП_i$ не буде нечітким числом з трикутною функцією належності. Але для простоти на практиці $ІЗП_i$ приблизно розглядають як нечітке число з трикутною функцією належності виду:

$$ІЗП_i \approx (c_i^l, c_i, c_i^r) =$$

$$= \left(\frac{1}{m} \sum_{j=1}^{m} \overline{w}_j^{(l)} \overline{s}_{ij}^{(l)}, \ \frac{1}{m} \sum_{j=1}^{m} \overline{w}_j \overline{s}_{ij}, \ \frac{1}{m} \sum_{j=1}^{m} \overline{w}_j^{(r)} \overline{s}_{ij}^{(r)} \right).$$

Застосування нечіткого булевого програмування до розв'язку задачі вибору проекту

Наступним кроком після обчислення індексу значимості проекту є розробка процедури вибору проекту в умовах рівноваги (ВПУР). Як було сказано вище, умови рівноваги повинні складатися із одного бюджетного обмеження (EC_0) та ще чотирьох наступних обмежень: тобто, обмеження відносно пріоритетності проекту (EC_1), типів проектів (EC_2), розмірів проектів (EC_3) та виконавців проектів (EC_4). Сформулюємо цю задачу як задачу нечіткого булевого програмування (НБП) та розглянемо можливості її розв'язку за домогою пакету прикладних програм.

Формулювання задачі вибору проекту. Нехай $\mathbf{P}=\{P_1, P_2,\ldots,P_n\}$ – множина запропонованих проектів, вартість яких складає ($b_1, b_2,\ldots, b_n$) – нечіткий індекс значимості проекту P_i = 1,2,…n; b – загальний бюджет, запланований на виконання проектів. Присвоїмо кожному проекту змінну x_i, яка прийматиме значення 0 або 1 в залежності від того, приймається проект чи ні, тобто

$$x_i = \begin{cases} 1, & \textit{якщо проект вибирається для фінансування}; \\ 0, & \textit{якщо проект не вибирається}. \end{cases}$$

Таким чином, задачею ВПУР є вибір такого проекту, який максимізує загальний внесок, тобто $\sum_{i=1}^{n} ІЗП_i \ x_i$. Очевидно, що умову рівноваги EC_0 (бюджетне обмеження) можна просто представити у вигляді $\sum_{i=1}^{n} e_i \ x_i \le b$. Всі інші умови $\{EC_1, EC_2, EC_3, EC_4\}$ задовольняють конкретним вимогам при розподілі фінансових ресурсів. Вони розділяють всю множину проектів, що виносяться на розгляд, на кілька підмножин з метою формування підгруп із своїми пріоритетами. При цьому з кожної підмножини можна вибрати для реалізації обмежене число проектів або тільки один проект. Наприклад, при розгляданні пріоритетності проекту EC_1 загальну кількість варіантів можна розбити на кілька груп (наприклад, 6). Ця умова приводить до того, що всі проекти $\mathbf{P}$ діляться на шість груп $\{\mathbf{P}_{11}, \mathbf{P}_{12}, \mathbf{P}_{13},\ldots, \mathbf{P}_{16}\}$ у відповідності до числа пріоритетів. При цьому до групи $\mathbf{P}_{1k}$ відносяться всі проекти *k–го* напряму (*k=1,2,…,6*). Тепер можна записати, що $\mathbf{P} = \cup_{k=1}^{6} \mathbf{P}_{1k}$. Якщо для *k–го* напряму було вибрано фіксоване число проектів b_{1k}, то можна записати обмеження у вигляді:

$$\sum_{j \in P_{1k}} x_j = b_{1k}, \quad k = 1, 2, \ldots, 6 .$$

В загальному випадку умова рівноваги EC_t приводить до необхідності створення підгрупи $\{\mathbf{P}_{t1}, \mathbf{P}_{t2}, \ldots, \mathbf{P}_{tq_t}\}$, яка входить в $\mathbf{P}$ разом із заданим числом проектів $\{b_{t1}, b_{t2}, \ldots, b_{tq_t}\}$, які будуть вибрані із кожної підгрупи. Відповідно, для кожної умови можна записати обмеження

$$\sum_{j \in P_{tk}} x_j = b_{tk}, \quad k = 1, 2, \ldots, q_t; \quad t = 1, 2, 3, 4 .$$

Таким чином, задачу вибору проектів в умовах зберігання рівноваги можна сформулювати так:

Максимізувати $\sum_{i=1}^{n} ІЗП_i \ x_i$ при обмеженнях вигляду [8]:

$$\begin{cases} \sum_{i=1}^{n} e_i \ x_i \le b, & (EC_0), \\ \sum_{i \in P_{1k}} x_i = b_{1k}, \quad k = 1, 2, \ldots, q_1, & (EC_1), \\ \sum_{i \in P_{2k}} x_i = b_{2k}, \quad k = 1, 2, \ldots, q_2, & (EC_2), \\ \sum_{i \in P_{3k}} x_i = b_{3k}, \quad k = 1, 2, \ldots, q_3, & (EC_3), \\ \sum_{i \in P_{4k}} x_i = b_{4k}, \quad k = 1, 2, \ldots, q_4, & (EC_4), \end{cases} \tag{3}$$

$$x_i \in \{0, 1\}, \quad i = 1, 2, \ldots, n .$$

Тобто, система (3) представляє собою задачу нечіткого булевого програмування, оскільки $ІЗП_i$ – нечіткі числа з трикутними ФН.

Розв'язок задачі НБП. При розв'язку задачі нечіткого булевого програмування (3) важливу роль відіграє ранжування нечітких чисел. В даному випадку скористаємось методом ранжування нечітких чисел, який дозволяє ідентифікувати узагальнене очікуване значення. У порівнянні з іншими цей метод є відносно простим з обчислювальної точки зору.

Означення 4. Якщо $\tilde{A}$ – це нечітке число з функцією належності (1), то узагальнене очікуване значення з індексом оптимізму μ визначається так [2]:

$$E_\mu(\tilde{A}) = \mu E_R(\tilde{A}) + (1-\mu) E_L(\tilde{A}),$$

де $E_R(\tilde{A})$ і $E_L(\tilde{A})$ – праве та ліве очікувані значення числа $\tilde{A}$, відповідно; $\mu \in [0,1]$, а $E_R(\tilde{A})$ і $E_L(\tilde{A})$ визначаються так:

$$E_R(\tilde{A}) = \int_\alpha^\beta x\, f_{\tilde{A}}^R(x)\,dx, \qquad E_L(\tilde{A}) = \int_\gamma^\delta x\, f_{\tilde{A}}^L(x)\,dx \qquad (4)$$

Еквівалентним означенням для $E_R(\tilde{A})$ і $E_L(\tilde{A})$ є також наступне:

$$E_R(\tilde{A}) = \int_0^1 g_{\tilde{A}}^R(y)\,dy, \qquad E_L(\tilde{A}) = \int_0^1 g_{\tilde{A}}^L(y)\,dy \qquad (5)$$

де $g_{\tilde{A}}^R(y)$ і $g_{\tilde{A}}^L(y)$ – це функції, які є оберненими по відношенню до $f_{\tilde{A}}^R(x)$ і $f_{\tilde{A}}^L(x)$, відповідно.

Параметр $\mu \in [0,1]$ характеризує ступінь оптимізму ОПР, який приймає значення в інтервалі від нуля до одиниці. При цьому більше значення μ відповідає більшому значенню оптимізму. Так, при $\mu=0$ маємо: $E_0(\tilde{A}) = E_L(\tilde{A})$, тобто ОПР має песимістичну точку зору. Для високого рівня оптимізму $\mu = 1$ і маємо: $E_1(\tilde{A}) = E_R(\tilde{A})$. Для помірного рівня оптимізму μ=0,5 і маємо: $E_{0,5}(\tilde{A}) = 0{,}5[E_L(\tilde{A}) + E_R(\tilde{A})]$.

Для нечіткого числа $\tilde{A} = (a_l, a, a_r)$ та рівня оптимізму $\mu \in [0,1]$ легко визначити, що $E_L(\tilde{A}) = 0{,}5(a_l + a)$, $E_R(\tilde{A}) = 0{,}5(a + a_r)$, а $E_\mu(\tilde{A}) = \mu E_R(\tilde{A}) + (1-\mu) E_L(\tilde{A})$.

При деякому рівні оптимізму μ нечіткі числа можна упорядкувати шляхом порівняння їх узагальнених очікуваних значень при конкретних величинах μ. Тобто, для двох нечітких чисел $\tilde{A}, \tilde{B}$ співвідношення $E_\mu(\tilde{A}) < E_\mu(\tilde{B})$, $E_\mu(\tilde{A}) > E_\mu(\tilde{B})$, $E_\mu(\tilde{A}) = E_\mu(\tilde{B})$ означають, що $\tilde{A} < \tilde{B}$, $\tilde{A} > \tilde{B}$ та $\tilde{A} = \tilde{B}$, відповідно.

Використовуючи узагальнені очікувані значення нечітких чисел, розглянемо методику розв'язку задачі нечіткого булевого програмування (3). Нехай X – скінченні множина можливих розв'язків задачі (3), а $g: \mathbf{P} \to \Im(R)$ – функція відображення множини проектів в множину нечітких чисел, що визначаються як

$$g(x) = \sum_{t \in S} ІЗП_i, \quad S = \{i \in N : x_i = 1\}.$$

У відповідності до принципу розширення маємо: якщо $ІЗП_i = (c_i^l, c_i, c_i^r), (i \in S)$ – нечіткі числа з трикутними функціями належності, то $g(x) = (\sum_{i \in S} c_i^l, \sum_{i \in S} c_i, \sum_{i \in S} c_i^r)$ також є нечітким числом із трикутною функцією належності.

Означення 5. При заданому рівні оптимізму $\mu \in [0, 1]$ розв'язок $x^* \in X$ називають оптимальним розв'язком задачі (3), якщо $E_\mu(g(x^*)) \geq E_\mu(g(x)), \ \forall x \in X$.

<u>Теорема 1</u>. Для заданого рівня оптимізму μ, $x^* \in X$ є оптимальним розв'язком задачі (3), якщо x^* є оптимальним розв'язком наступної класичної задачі булевого програмування:

Максимізувати $\sum_{i=1}^n [c_i + \alpha c_r + (1-\alpha) c_l] x_i$

при обмеженнях

$$\begin{cases} \sum_{i=1}^n e_i x_i \leq b, & (EC_0), \\ \sum_{i \in P_{1k}} x_i = b_{1k}, \quad k = 1,2,...,q_1, & (EC_1), \\ \sum_{i \in P_{2k}} x_i = b_{2k}, \quad k = 1,2,...,q_2, & (EC_2), \\ \sum_{i \in P_{3k}} x_i = b_{3k}, \quad k = 1,2,...,q_3, & (EC_3), \\ \sum_{i \in P_{4k}} x_i = b_{4k}, \quad k = 1,2,...,q_4, & (EC_4). \end{cases} \qquad (6)$$

$x_i \in \{0, 1\}, \quad i = 1,2,...,n$.

Модель (4) – класична задачу цілочислового програмування. Якщо її вимірність не занадто велика, то її можна розв'язати за допомогою за допомогою оптимізаційного інструментарію системи MATLAB або іншого пакету прикладних програм. Наприклад, зручними є системи підтримки прийняття рішень, які містять спеціальні інструменти для реалізації обчислень та критерії для вибору кращих розв'язків [9].

Модельний приклад. Розглянемо застосування запропонованого вище методу до розв'язку задачі вибору проектів. Для цього скористаємось модельними даними, але це не означає, що даний приклад є спрощеним. В плануванні та прийманні рішень щодо проектів приймають участь представники фінансуючої організації (міністерства), та міністерств чи відомств, які ми розділимо на три групи і розглядатимемо як ОПР. Позначимо ці групи як D_1, D_2, D_3. Проілюструємо застосування методу на основі даних, що складають 52 пропозиції при бюджеті 4 000 000 грн. З метою збалансування інтересів між учасниками групи планування запишемо чотири умови рівноваги:

EC_1: розділимо всі 52 проекти по відношенню до пріоритетності (А) на шість груп $\{A_1, A_2,\ldots,A_6\}$. Нехай з кожної групи A_t, t=1,2,…,6 необхідно вибрати по два проекти.

EC_2: за типами проектів (T) розділимо всі 52 проекти також на 6 груп $\{T_1, T_2,\ldots,T_6\}$. При цьому з кожної групи T_t, t=1,2,…,6 необхідно вибрати по два проекти.

EC_3: за обсягами проектів (S) розділимо 52 проекти на три групи $\{S_1, S_2, S_3\}$. При цьому із кожної групи S_t, t=1,2,3 виберемо по чотири проекти.

EC_4: за виконавцями проектів розділимо 52 проекти на 5 груп $\{M_1, M_2,\ldots,M_5\}$. При цьому кожна група виконавців $M_1, M_2,\ldots,M_5$ повинна виконати певну кількість проектів. Нехай це буде 6, 2, 2, 1, 1, відповідно.

Тепер можна скласти таблицю розподілу проектів у відповідності до умов рівноваги $EC_1,\ldots,EC_4$. Таке групування проектів наведено в таблиці 1.

Особи, які приймають рішення, оцінюють ваговий коефіцієнт важливості для кожного критерія вибору, використовуючи при цьому лінгвістичні значення за шкалою:

W={*VI, I, F, UI, VUI*}. В таблиці 2 наведено лінгвістичні значення вагових коефіцієнтів важливості, а також нормовані середні значення вагових коефіцієнтів важливості (нечіткі числа для трикутних функцій належності).

Wн1=(0,11; 0,20; 0.39);
Wн2 = (0,18; 0,29; 0,47);
Wн3=(0,21; 0,33; 0.49);
Wн4 = (0,02; 0,18; 0,356).

Табл. 1. Умови рівноваги та групування проектів

№ пр.	1	2	3	4	5	6	7	8	9
*EC*1	*A*2	*A*2	*A*3	*A*6	*A*2	*A*5	*A*1	*A*5	*A*2
*EC*2	*T*5	*T*5	*T*5	*T*5	*T*5	*T*5	*T*5	*T*5	*T*5
*EC*3	*S*2	*S*2	*S*2	*S*2	*S*2	*S*2	*S*2	*S*2	*S*2
*EC*4	*M*1	*M*1	*M*1	*M*1	*M*1	*M*1	*M*1	*M*1	*M*1

№ пр.	10	11	12	13	14	15	16	17
*EC*1	*A*3	*A*4	*A*2	*A*1	*A*4	*A*2	*A*5	*A*3
*EC*2	*T*5	*T*5	*T*5	*T*5	*T*5	*T*5	*T*4	*T*4
*EC*3	*S*1	*S*1	*S*1	*S*1	*S*1	*S*1	*S*2	*S*3
*EC*4	*M*1	*M*1	*M*1	*M*1	*M*1	*M*1	*M*1	*M*1

№ пр	18	19	20	21	22	23	24	25
*EC*1	*A*4	*A*4	*A*4	*A*2	*A*2	*A*5	*A*5	*A*5
*EC*2	*T*4	*T*4	*T*6	*T*6	*T*2	*T*2	*T*2	*T*2
*EC*3	*S*2	*S*2	*S*2	*S*1	*S*2	*S*2	*S*3	*S*2
*EC*4	*M*1	*M*1	*M*1	*M*1	*M*1	*M*1	*M*2	*M*2

№ пр.	26	27	28	29	30	31	32	33
*EC*1	*A*5	*A*5	*A*6	*A*6	*A*6	*A*2	*A*6	*A*6
*EC*2	*T*2	*T*2	*T*2	*T*2	*T*2	*T*2	*T*2	*T*2
*EC*3	*S*2	*S*1	*S*1	*S*2	*S*3	*S*2	*S*1	*S*1
*EC*4	*M*2	*M*2	*M*2	*M*3	*M*4	*M*1	*M*1	*M*1

№ пр	34	35	36	37	38	39	40	41
*EC*1	*A*2	*A*2	*A*2	*A*4	*A*3	*A*3	*A*4	*A*4
*EC*2	*T*2	*T*2	*T*2	*T*2	*T*2	*T*2	*T*2	*T*2
*EC*3	*S*3	*S*1	*S*3	*S*1	*S*1	*S*3	*S*3	*S*3
*EC*4	*M*2	*M*3	*M*1	*M*1	*M*1	*M*1	*M*1	*M*2

№ пр	42	43	44	45	46	47	48	49
*EC*1	*A*3	*A*6	*A*4	*A*1	*A*4	*A*4	*A*1	*A*3
*EC*2	*T*2	*T*2	*T*2	*T*2	*T*2	*T*2	*T*1	*T*3
*EC*3	*S*3	*S*2	*S*3	*S*3	*S*3	*S*2	*S*1	*S*2
*EC*4	*M*2	*M*2	*M*2	*M*2	*M*2	*M*1	*M*3	*M*3

№ пр.	50	51	52
*EC*1	*A*5	*A*3	*A*6
*EC*2	*T*5	*T*3	*T*6
*EC*3	*S*1	*S*3	*S*2
*EC*4	*M*3	*M*5	*M*4

Табл. 2. Вагові коефіцієнти важливості для вибраних критеріїв

Критерії/ ОПР	C1	C2	C3	C4
*D*1	F	VI	I	UI
*D*2	I	I	VI	F
*D*3	F	I	VI	I
Нормовані вагові коефіцієнти	Wн1	Wн2	Wн3	Wн4

Особи, що приймають рішення, оцінюють якість кожного проекту у порівнянні із кожним критерієм вибору за допомогою лінгвістичних рейтингових змінних, наприклад, у відповідності із наступною шкалою: S={*EG*,

VG, G, M, P, VP, EP}. Характеристики розглянутого підходу до вибору проектів для фінансування в умовах обмеженого бюджету можна покращити за рахунок більш строгого підходу до вибору функцій належності, а також врахування інших факторів, що впливають на вибір, наприклад, соціально-політичних.

Висновки

Запропонований метод розв'язку задачі розподілу ресурсів між проектами або вибору проектів для фінансування грунтується на використання результатів теорії нечітких множин та цілочислового математичного програмування. Такий підхід відрізняється від відомих тим, що дає можливість врахувати суб'єктивні оцінки експертів та об'єктивні (фактичні) обмеження на фонди, що спрямовуються на реалізацію проектів. Застосування нечітких змінних дозволяє визначити індекси значимості проектів, які використані для оптимізації процесу розподілу ресурсів – максимізації ІЗП. Лінгвістичні змінні визначені за допомогою трикутних функцій належності, які є досить простими з точки зору формування та подальшого використання. Однак, в подальшому необхідно дослідити можливості застосування функцій іншого типу, наприклад, гаусових функцій, а також формалізувати процес вибору параметрів ФП.

Перелік посилань

1. Дж. Моудер Исследование операций. – Методологические основы и математические методы / Под ред. Дж. Моудера, С. Элмаграби. – Москва: Мир, 1981, том 1. – 712 с.
2. Згуровский М.З Системный анализ – проблемы, методология, приложения / Згуровский М.З., Панкратова Н.Д. – Киев: Наукова думка, 2005. – 744 с.
3. Крыжова К.А. Средства анализа обстановки и принятия решений в системе бизнес-процессов / Крыжова К.А. // Матеріали шостої міжнародної науково-практичної конференції з програмування УкрПРОГ'2008. №2-3, Спеціальний випуск.
4. Теленик С.Ф. Управління доступом до обмежених ресурсів інформаційно-телекомунікаційної мережі АСУ військового призначення / Теленик С.Ф., Ролік О.І., Букасов М.М., Терещенко П.І. // Зб. наук. праць ЦНДІ Збройних Сил України. – 2006. – №3. – С.33-43.
5. Теленик С.Ф. Моделі управління розподілом обмежених ресурсів в інформаційно-телекомунікаційній мережі АСУ / Теленик С.Ф., Ролік О.І., Букасов М.М. // Вісник НТУУ «КПІ». Інформатика, управління та обчислювальна техніка. – К.: Екотех. – 2006. – №44. – С.234-239.
6. Теленик С.Ф. Система управління інформаційно-телекомунікаційною системою корпоративної АСУ / Теленик С.Ф., Ролік О.І., Букасов М.М., Соколовський Р.Л. // Вісник НТУУ «КПІ». Інформатика, управління та обчислювальна техніка. – К. – 2006. – №45. – С.112-126.
7. Kaufmann A. Introduction to theory of fuzzy subsets / Kaufmann A. – New York: Academic Press, 1975. – 643 p.
8. Аверкин А.Н. Нечеткие множества в моделях управления и искусственного интеллекта / Аверкин А.Н., Батыршин И.З., Блишун А.Ф., Силов В.Б., Тарасов В.Б. – Москва: Наука, 1986. – 312 с.
9. Бідюк П.І. Системи підтримки прийняття рішень – проектування та реалізація / Бідюк П.І., Щербань Ю.Ю., Щербань В.Ю., Демківський Є.О. – Київ: КНУТД, 2004. – 112 с.

УДК 004.94(075.8)

ІВАНОВ В.О.

РОЗПОДІЛЕНА СИСТЕМА ІМІТАЦІЙНОГО МОДЕЛЮВАННЯ ДОРОЖНЬОГО РУХУ

Статтю присвячено огляду і аналізу систем імітаційного моделювання дорожнього руху. Приведені шляхи подолання недоліків існуючих систем. Доведено необхідність розробки системи моделювання дорожнього руху як розподіленої системи.

The article deals with the description and analysis of different traffic simulation systems. The ways of existing systems drawbacks overcoming are proposed. The necessity of distributed traffic simulation system development is proved in the article.

Вступ

У зв'язку з розвитком економіки, бізнес-структур і зростанням міст збільшується кількість транспортних засобів, що призводить до значних затримок при пересуванні транспорту і як наслідок – до багатомільйонних фінансових втрат і забруднення навколишнього середовища. Транспортні проблеми тісно пов'язані з проектуванням транспортних розв'язок у великих містах, з інфраструктурою міст і з організацією дорожнього руху. Інтегроване розв'язання проблеми можливе лише при застосуванні засобів імітаційного моделювання дорожнього руху і впровадженні комплексу математичних моделей, які орієнтовані на вирішення конкретних задач прогнозування і управління дорожнім рухом. Системи імітаційного моделювання дорожнього руху дозволяють знизити фінансові витрати на вирішення транспортних проблем.

Огляд і аналіз систем імітаційного моделювання транспортних мереж

Існує багато різних систем імітаційного моделювання транспортних мереж. Серед них такі системи, як AIMSUN, PA-RAMICS, AUTOBAHN, IHSDM, INTEGRATION, PLANSIM-T, FLEXSYT-II, TRANSIMS, SimTraffic 6, VISSIM, MITSIM та ін.

Розглянемо систему імітаційного моделювання транспортного руху AIMSUN.

AIMSUN – це система моделювання транспортного руху на мікрорівні. Система призначена для моделювання трафіка міських транспортних мереж, автострад і автомагістралей, кільцевих доріг і дорожніх розгалужень. AIMSUN було спроектовано і розроблено як засіб аналізу транспортного руху. В AIMSUN реалізовано детальне моделювання транспортної мережі. Це означає, що поведінка кожного окремого транспортного засобу в мережі моделюється на кожному кроці протягом всього часу моделювання. Система забезпечує високий рівень деталізації моделі транспортної мережі, розрізняє декілька типів транспортних засобів і водіїв, забезпечує широкий діапазон геометрії мережі для взаємодії. В системі AIMSUN також можуть бути промодельовані ДТП, конфліктні маневри і т.п. Для управління транспортним рухом в AIMSUN використовуються знаки дорожнього руху, адаптивне управління і управління транспортом, що має пріоритет руху (швидка допомога, пожежна служба, міліція і т.п.).

В ядрі моделювання реалізована можливість використання кроку моделювання від 0.1 секунди.

В системі існує можливість моделювання вибору маршруту водієм до початку і під час поїздки. AIMSUN включає чотири різних алгоритми моделювання динамічного вибору маршруту, редактор функції підрахунку вартості проїзду і опцію для визначення вартості проїзду, виходячи з інформації про попередні маршрути. Різні водії будуть використовувати різні критерії вибору маршруту: від постійного вибору того ж самого маршруту до зміни маршруту відповідно до умов руху.

В [2] відмічено, що моделі поведінки транспортних засобів в системі AIMSUN визначаються функціями від декількох параметрів, що дозволяє моделювати рух різних типів транспортних засобів (автомобілів, автобусів, вантажівок і т.д.), які потім можуть бути об'єднані в класи.

Переміщення транспортних засобів реалізовано двома способами:або за заданими маршрутами і відсотковим розподіленням потоку або згідно заданої матриці кореспонденцій. В останньому випадку маршрут визначається

трьома способами: раніше сформованою інформацією, перерахунком маршруту через визначені інтервали часу згідно матриці вартостей і ситуації, що склалась на дорозі, або шляхом динамічного перерахунку маршруту в процесі моделювання.

Поведінка транспортного засобу перевизначається кожну одиницю модельного часу. Для опису моделі поведінки автомобіля використовується модель Гіппса.

Слід зазначити, що в пакеті відсутні обмеження на розмір мережі, але швидкість роботи пакету обмежується обсягом оперативної пам'яті та потужністю процесора.

Розглянемо систему імітаційного моделювання транспортного руху MITSIM. MITSIMLab – це система, що базується на імітаційному моделюванні. Ця система була розроблена для оцінювання впливу альтернатив проектування систем управління рухом на оперативному рівні.

MITSIMLab є синтезом різних моделей і має такі характеристики: надає широкий вибір різних варіантів проектування систем управління рухом, моделює реакцію водіїв на поточну інформацію щодо руху і управляючих сигналів, об'єднує динамічну взаємодію між системою управління рухом і водіями в мережі.

Різні компоненти MITSIMLab об'єднані в такі три модулі:

1) модуль імітаційного моделювання руху на мікрорівні (Microscopic Traffic Simulator (MITSIM));

2) модуль імітаційного моделювання управління рухом (Traffic Management Simulator (TMS));

3) графічний інтерфейс користувача (GUI).

Розглянемо модуль імітаційного моделювання потоку руху (MITSIM). Елементи руху і мережі представляються детально для того, щоб визначити чутливість потоків руху до стратегій управління і маршрутизації. Основними елементами MITSIM є:

1) компоненти мережі: мережа доріг з обладнанням для спостереження і управління рухом представлена на мікрорівні. Мережа доріг складається з вузлів, зв'язків, сегментів (зв'язки розділені на сегменти з однаковими геометричними характеристиками) і рядів;

2) визначення пункту призначення і вибір маршруту руху: модуль імітаційного моделювання руху в якості вхідних даних отримує таблиці пунктів відправлення і призначення (OD), які залежать від часу. Ці таблиці або являють собою очікувані умови руху, або визначені як частина сценарію для оцінювання. Для визначення вибору маршруту водієм використовується імовірнісна модель вибору маршруту;

3) поведінка водіїв: виходячи з потоків відправлення/призначення визначається намір окремого транспортного засобу увійти в мережу у визначений момент часу. Параметри поведінки (бажана швидкість, ініціативність і т.д.) і характеристики транспортного засобу призначаються кожній комбінації «транспортний засіб/водій».

MITSIM здійснює переміщення транспортних засобів у відповідності до моделей слідування за автомобілем та зміни ряду. Модель слідування за автомобілем визначає реакцію водія на ситуацію на дорозі безпосередньо попереду транспортного засобу, виходячи з відносної швидкості, інтервалу руху та інших обмежень руху. Модель зміни ряду визначає різницю між обов'язковою і необов'язковою зміною ряду. Також моделюються злиття доріг, реакція водіїв на управляючі сигнали руху, обмеження швидкості і аварії.

Розглянемо модуль імітаційного моделювання управління рухом (TMS).

Модуль імітаційного моделювання управління рухом імітує систему управління рухом, що оцінюється. Для оцінки представлено широкий спектр систем управління рухом і маршрутизації, таких як:

1) «лежачий поліцейський» (штучна нерівність для обмеження швидкості транспорту);

2) засоби управління рухом на швидкісній автостраді:

a) знаки для управління зміною ряду;

b) знаки для обмеження швидкості;

c) знаки для позначення в'їзду в тунель;

3) засоби управління рухом на перехрестях;

4) знаки дорожнього руху.

TMS має загальну структуру, яка дозволяє представляти різні проекти систем з логікою різного рівня складності (від визначеної до гнучкої).

Система імітаційного моделювання має інтерфейс, який служить для відображення переміщень транспортних засобів в мережі та динамічних змін швидкості та щільності руху. Графічний інтерфейс користувача (GUI) може використовуватись як для відладки так і для

демонстрації взаємодії учасників руху за допомогою анімації транспортних засобів[5].

Кі Йенг (Qi Yang), один із розробників системи, в [3] вказує, що MITSIM представляє дороги зв'язками, вузлами мережі, сегментами і маршрутами. Дані, які описують мережу, що імітується, зчитуються з мережевої бази даних [4]. Мережева база даних включає опис всіх мережевих об'єктів, з'єднання маршрутів, привілеї використання траси, регулювання розворотів і переміщень на перехрестях (наприклад, заборона лівих поворотів і розворотів), датчиків руху, управляючих пристроїв і пунктів оплати.

MITSIM не може бути використаний для моделювання великих мереж, таких як мережа доріг міста, особливо, якщо обчислювальні ресурси обмежені. Існує багато параметрів, які можуть бути задані користувачем для відображення поведінки водія в різних географічних умовах. Однак, збір даних, їх оцінка і калібровка є трудомісткою задачею. В системі імітаційного моделювання MITSIM мережа, що моделюється, завжди є підмережею. При виникненні затору водій, насправді, може вибрати маршрут, який не моделюється в системі. [6]

Розглянемо систему імітаційного моделювання транспортного руху PARAMICS.

PARAMICS (PARAllel MICroscopic Simulation) (Quandstone Ltd., United Kingdom) – це набір програмних інструментів для моделювання трафіка на мікрорівні. Ця система широко використовується у Великобританії та США. Пакет призначено для моделювання транспортних вузлів в містах (перехрестя, що регулюються правилами пріоритету та світлофорами, транспортні розв'язки і т. д), моделювання перевантажених автострад, оптимізації роботи громадського транспорту, з'їздів з автомагістралей, регулювання маршрутів, світлофорів [2].

Моделюючий модуль системи PARAMICS забезпечує три фундаментальні операції: побудову моделі, імітаційне моделювання трафіка (з 3D візуалізацією) і вивід статистичної інформації з використанням дружнього графічного інтерфейсу користувача. За допомогою моделюючого модуля можуть бути проаналізовані такі елементи транспортної мережі:

1) міська мережа і мережа автострад;
2) правосторонній і лівосторонній рух;
3) управління рухом за допомогою знаків дорожнього руху;
4) ділянка дороги з круговим рухом;
5) громадський транспорт;
6) паркування машин;
7) ДТП;
8) зміна полоси руху;
9) затори на дорогах.

Поведінка кожного транспортного засобу в транспортній мережі моделюється на кожному кроці протягом всього часу моделювання [7].

В [2] відмічено, що PARAMICS підтримує можливість індивідуального переміщення близько 200000 автомобілів за одиницю часу. Стандартно задано 7 класів транспортних засобів, але користувач може створювати власні транспортні засоби. Вибір маршруту автомобілем визначається заданою таблицею вартостей. У кожного транспортного засобу є заданий інтервал часу (в середньому 1 сек.), через який визначається положення транспортного засобу та його поведінка. Зміна ряду в PARAMICS виконується на основі правил знаходження відповідного інтервалу між автомобілями і з використанням інформації про доступні відповідні інтервали. В пакеті реалізовано алгоритм, який задає рух автомобіля на заданій траєкторії маршруту. Рух регулюється фізичними атрибутами автомобіля і його поточною швидкістю. Також підтримується можливість визначити маршрут згідно з матрицею кореспонденцій.

В PARAMICS реалізовані можливості збору даних статистики і формування звітів щодо аналізу транспортної мережі.

Слід відзначити, що основним обмеженням на розмір мережі і швидкість роботи в пакеті є об'єм пам'яті і потужність комп'ютера.

Розглянемо систему імітаційного моделювання транспортного руху VISSIM.

VISSIM – пакет для моделювання трафіка на мікрорівні. Пакет призначено для аналізу, реінженірінгу і оптимізації міських і міжміських транспортних сполучень. Дозволяє моделювати міські перехрестя будь-якого типу регулювання, аналізувати пропускну спроможність транспортних систем і тестувати схеми транзитних пріоритетів. Дозволяє деталізувати схему транспортної мережі, в тому числі переходи, стоянки, зупинки громадського транспорту. В потоці беруть участь всі види транспортних засобів, а також пішоходи [2].

Модель потоку трафіка, що реалізована в Vissim, є дискретною стохастичною моделлю. Модель базується на кроці моделювання, рівному 1 секунді. Модель містить алгоритми зміни ряду, засновані на правилах дорожнього руху. Слідування транспортних засобів один за одним є коливальним процесом. За умови руху в один ряд транспортний засіб, що рухається з більшою швидкістю, наближуючись до транспортного засобу, що рухається з меншою швидкістю, повинен зменшити швидкість. При цьому час реакції водія залежить від різниці між швидкостями транспортних засобів, відстані між транспортними засобами та від моделі поведінки водія. За умови руху по дорозі в декілька рядів транспортний засіб, який наблизився до транспортного засобу, що рухається з меншою швидкістю, перевіряє, чи покращає ситуація після зміни ряду. Якщо так, то транспортний засіб буде шукати підходящі інтервали між автомобілями в сусідніх рядах. Слідування за транспортними засобами і зміна ряду формують модель потоку трафіка, яка є ядром Vissim [6].

В [2] відмічено, що в Vissim реалізовані стандартні типи транспортних засобів (автомобілі, вантажівки, автобуси, трамваї, потяги, мотоцикли, велосипеди). Всі ці види транспортних засобів можна параметризувати (габарити, потужність двигуна, розподілення прискорення та гальмування, вага і т.д.). Користувач може задати свій тип транспортного засобу. Задаються параметри інтенсивності потоку транспортних засобів, його пропорційного складу, графіки роботи світлофорів, імовірності вибору маршруту руху. Підтримується можливість підключення матриці призначень для опису розподілення трафіка.

В пакеті реалізовано модель Відерманна, яка описує поведінку водія за кермом. В ній враховуються психофізичні можливості людини (час реакції, час, необхідний для прийняття рішення в умовах оточуючого середовища і т.д.).

VISSIM надає можливість збору статистичних даних на будь-якій дільниці транспортної мережі і формування звітів, створення презентацій і відеороликів.

В VISSIM реалізовано можливість вибору між різними типами маршрутизації:

1) статична маршрутизація, яка базується на зборі даних про повороти та перехрестя;

2) вільний вибір маршруту з використанням динамічного призначення: модуль динамічного призначення пропонує різні параметри, використовуючи матриці відправлення-призначення, що засновані на меті поїздки та залежать від часу [8].

Розмір мережі не обмежений програмним забезпеченням, однак, Vissim моделює обмежену кількість перехресть, залежно від потужності комп'ютера. Час розрахунку залежить від кількості транспортних засобів, які одночасно знаходяться в транспортній мережі. При зростанні кількості транспортних засобів модельний час може досягти реального часу. Значення розподілень прискорень, максимальної швидкості і бажаної швидкості задані, але можуть змінюватись користувачем.

Модель потоку руху не може бути використана для автоматичного управління рухом, оскільки вона базується на кроці моделювання, що дорівнює 1 секунді. Vissim не включає ніяких алгоритмів вибору пункту призначення; маршрути транспортних засобів є вхідними даними [6].

Постановка проблеми і шляхи її вирішення

Необхідно створити систему імітаційного моделювання дорожнього руху, яка має низку переваг у порівнянні з існуючими системами моделювання транспортних мереж.

Існує багато систем імітаційного моделювання транспортних мереж. Всі вони мають один основний недолік: швидкість роботи і/або розмір транспортної мережі обмежені розміром оперативної пам'яті і потужністю процесора.

Одним із варіантів рішення цієї проблеми є зниження рівня деталізації моделювання, в тому числі використання збільшення розміру кроку моделювання. Це призводить до зниження якості моделі транспортної мережі.

Іншим варіантом рішення цієї проблеми є моделювання лише підмережі транспортної мережі. Однак, це призводить до того, що при виникненні затору водій, насправді, може обрати маршрут, який не моделюється в системі.

Проблеми підвищення швидкості роботи системи, обмеження розміру транспортної мережі і обмеження кількості транспортних засобів можуть бути вирішені шляхом реалізації системи моделювання дорожнього руху як розподіленої системи. Таким чином, вказані

вище проблеми можуть бути вирішені без зниження рівня деталізації моделювання.

Як вказано в [1], швидкість роботи системи залежить від кількості комп'ютерів, які використовуються для моделювання. Розподілені системи мають такі переваги:

1) вартість N комп'ютерів значно нижча за вартість N-процесорного комп'ютера;

2) для збільшення потужності обчислювальної бази, яка використовується при моделюванні, необхідно придбати потрібну кількість комп'ютерів замість заміни багатопроцесорного комп'ютера на більш продуктивний;

3) при виході з ладу одного з комп'ютерів, безперебійна робота розподіленої системи може бути продовжена.

Висновки

В даній статті авторами проаналізовані системи імітаційного моделювання транспортного руху, і доведена необхідність розробки системи моделювання дорожнього руху як розподіленої системи. Така система моделювання дозволить моделювати транспортні мережі великих міст з великою кількістю автомобілів, а також скоротити фінансові витрати на забезпечення роботи системи.

Перелік посилань

1. Томашевський В.М., Іванов В.О. Архітектура системи моделювання дорожнього руху. // Інтелектуальні системи прийняття рішень та проблеми обчислювального інтелекту (ISDMCI'2008): Матеріали Міжнародної наукової конференції (19-23 травня 2008г, г. Євпаторія), Том 2, Частина 2, 128 с. – С 82-84.
2. Яцкив И.В., Юршевич Е.А, Колмакова Н.В. Использование возможностей имитационного моделирования для анализа транспортных узлов [Електронний ресурс] // Имитационное моделирование. Теория и практика (ИММОД 2005): Материалы второй Всероссийской научно-практической конференции по вопросам применения имитационного моделирования в промышленности. (19-21 октября 2005 г, Россия, Санкт-Петербург) (http://www.gpss.ru/immod05/s3/yackiv/).
3. Qi Yang, A Microscopic Traffic Simulator for Evaluation of Dynamic Traffic Management Systems, [Електронний ресурс] http://web.mit.edu/qiyang/www/thesis/main.html
4. Sturdy J.L., Morin T.M. Binary Data Channel: User's Manual. MIT Lincoln Laboratory, Lexington, Massachusetts, 1994
5. http://mit.edu/its/mitsimlab.html
6. http://www.its.leeds.ac.uk/projects/smartest/append3d.html (Review of Micro-Simulation Models)
7. http://www.paramics-online.com
8. http://www.ptvamerica.com/vissim.html

УДК 519.854.2

ПАВЛОВ А.А.,
ИВАНОВА А.А.

АЛГОРИТМ ИДЕНТИФИКАЦИИ ФУНКЦИИ ЦЕЛИ В МЕТОДЕ АНАЛИЗА ИЕРАРХИЙ

Задача принятия решений на основе метода анализа иерархий обобщается для случая, позволяющего не только находить аппроксимации неизвестных значений глобальной цели на заданном множестве альтернатив, но и сводить построение аппроксимирующей функции, которая является формализацией глобальной цели, к задаче линейного программирования.

Decision making problem on the base of the Analytical Hierarchy Process is generalized for the case permitting to find approximations of unknown values of global goal on the assigned set of alternatives. It also permits to reduce tracing of approximation function which is the formalization of global goal, to the linear programming problem.

Введение

Метод анализа иерархий (МАИ) Саати [1-4] является наиболее применимым методом нахождения наилучшей альтернативы из конечного множества альтернатив в случае, когда глобальная цель не формализована. Фактически МАИ находит аппроксимацию значений не формализованной глобальной цели на заданном множестве альтернатив.

В [5] была сформулирована следующая задача.

1) Имеется множество альтернатив $A_{s_1},...A_{s_p}$ из множества заданных альтернатив $A_1,...A_m$, для которых с помощью МАИ построены оценки $f\left(A_{s_j}\right)=f\left(\overline{x}^{s_j}\right),\ j=\overline{1,p}$, где $n-$ мерный вектор $\overline{x}^{s_j}$ представляет альтернативу A_{s_j}, $f\left(\overline{x}\right)-$ неизвестная числовая скалярная функция векторного аргумента $\overline{x}$, формализующая глобальную цель, (формулировка которой в общем случае носит качественный характер).

2) Имеется множество парных предпочтений $G\left(A_i \text{ не хуже } A_j\right),\ i,j\in I,\ i,j\in\left\{\overline{1,m}\right\}$.

Каждую пару эксперт (эксперты) задают тогда, когда он уверен, что с точки зрения глобальной цели альтернатива A_i не хуже альтернативы A_j. Очевидно, что парные предпочтения не должны нарушать условия транзитивности, иначе $f\left(\overline{x}\right)$ не существует. Необходимо по этим данным построить наилучшую (по введенному и обоснованному критерию) аппроксимирующую функцию $f\left(\overline{x}\right)$, формализующую глобальную цель.

Для упрощения процесса формализации задачи будем считать, что для любой пары индексов $(i,j)\in I$ выполняется $i,j\in\left\{s_1,s_2,...,s_p\right\}$

Постановка задачи

В данной статье рассмотрена наиболее простая постановка сформулированной выше задачи, решение которой естественным образом используется в более общем случае.

Пусть имеют место следующие посылки, основанные на экспертном анализе структуры искомой аппроксимирующей функции $f\left(\overline{x}\right)$:

1. $f\left(\overline{x}\right)$ является взвешенной суммой известных составляющих, каждая из которых выражает вклад вектора $\overline{x}$, либо множества компонент (с учетом их взаимного влияния) в глобальную цель.

Веса каждой из составляющих являются неизвестными. Таким образом, построение функции $f\left(\overline{x}\right)$ сводится к нахождению весовых коэффициентов.

2. Известные компоненты принадлежат одному из следующих классов:

1) $$\sum_{\forall(i_1...i_k)\in K_0} a_{i_1...i_k}\cdot 1\left(x_{i_1}\cdot x_{i_2}...x_{i_k}\right)$$

$$1\left(x_{i_1}\cdot x_{i_2}...x_{i_k}\right)=\begin{cases}1, \forall x_{i_j}>0, & j=\overline{1,k}\\ 0, & \exists x_{ij}=0\end{cases}.$$

Первая компонента отражает постоянный вклад в глобальную цель, который определяется фактором неравенства одновременно нулю определенных совокупностей компонент вектора $\overline{x}$.

2) $$\sum_{\forall(i_1 i_2 \dots i_k)\in K_1} \; \sum_{\forall(j_1 j_2 \dots j_k)\in K_1(i_1 i_2 \dots i_k)} \pm\, b_{i_1\, i_2\, \dots\, i_k}^{p_{i_1}^{j_1} p_{i_2}^{j_2} \dots p_{i_k}^{j_k}} \left(x_{i_1}\right)^{p_{i_1}^{j_1}} \left(x_{i_2}\right)^{p_{i_2}^{j_2}} \dots \left(x_{i_k}\right)^{p_{i_k}^{j_k}}$$

$\forall b_{i_1 i_2 \dots i_k}^{p_{i_1}^{j_1} \dots p_{i_k}^{j_k}} \ge 0 \quad i_l, j_l, \quad l=\overline{1,n} \in \left\{\overline{1,n}\right\}$,

$p_{i_l}^{j_l}, \quad l=\overline{1,k} \le n$ – натуральные числа, $\pm$ перед каждым членом суммы обозначает, что эксперт точно знает знак соответствующей взвешенной составляющей.

3) $$\sum_{\forall(i_1 \dots i_k)\in K_2} \; \sum_{\forall(j_1 \dots j_k)\in K_2(i_1, \dots i_k)} c_{i_1\, \dots\, i_k}^{t_{i_1}^{j_1} t_{i_2}^{j_2} \dots t_{i_k}^{j_k}} \cdot \left(x_{i_1}\right)^{t_{i_1}^{j_1}} \cdot \left(x_{i_2}\right)^{t_{i_2}^{j_2}} \dots \left(x_{i_k}\right)^{t_{i_k}^{j_k}}$$

$c_{i_1\, \dots\, i_k}^{t_{i_1}^{j_1} t_{i_2}^{j_2} \dots t_{i_k}^{j_k}}$ – неизвестные коэффициенты,

$t_{i_l}^{j_l}, \quad l=\overline{1,k} \le n$ – натуральные числа.

Будем рассматривать случай, когда эксперты способны дать заключение, что с точностью до неизвестных численных значений коэффициентов $a_{i_1 \dots i_k}$; $b_{i_1\, \dots\, i_k}^{p_{i_1}^{j_1} \dots p_{i_k}^{j_k}}$; $c_{i_1\, \dots\, i_k}^{t_{i_1}^{j_1} \dots t_{i_k}^{j_k}}$ функция $f\left(\overline{x}\right)$ имеет вид:

$$f\left(\overline{x}\right) = \sum_{\forall(i_1 \dots i_k)\in K_0} a_{i_1 \dots i_k} \cdot 1\left(x_{i_1} \dots x_{i_k}\right) + \sum_{\forall(i_1 \dots i_k)\in K_1} \; \sum_{\forall(j_1 \dots j_k)\in K_1(i_1 \dots i_k)} \pm b_{i_1\, \dots\, i_k}^{p_{i_1}^{j_1} \dots p_{i_k}^{j_k}} \cdot \left(x_{i_1}\right)^{p_{i_1}^{j_1}} \cdot \left(x_{i_2}\right)^{p_{i_2}^{j_2}} \dots \left(x_{ik}\right)^{p_{i_k}^{j_k}} +$$

$$+ \sum_{\forall(i_1 \dots i_k)\in K_2} \; \sum_{\forall(j_1 \dots j_k)\in K_2(i_1 \dots i_k)} c_{i_1\, \dots\, i_k}^{t_{i_1}^{j_1} \dots t_{i_k}^{j_k}} \cdot \left(x_{i_1}\right)^{t_{i_1}^{j_1}} \cdot \left(x_{i_2}\right)^{t_{i_2}^{j_2}} \dots \left(x_{ik}\right)^{t_{i_k}^{j_k}}$$

При этом эксперты не гарантируют, что все весовые коэффициенты не равны нулю. Необходимо найти истинную структуру функции $f\left(\overline{x}\right)$ и значения неизвестных коэффициентов, соответствующих этой структуре.

Решение поставленной задачи

Оценка неизвестных коэффициентов проводится по одной из следующих задач линейного программирования:

$$\min y$$

$$-y \le f\left(\overline{x}^{s_l}\right) - \sum_{\forall(i_1 \dots i_k)\in K_0} a_{i_1 \dots i_k} \cdot 1\left(x_{i_1}^{s_l} \dots x_{i_k}^{s_l}\right) + \sum_{\forall(i_1 \dots i_k)\in K_1} \; \sum_{\forall(j_1 \dots j_k)\in K_1(i_1 \dots i_k)} \pm b_{i_1\, \dots\, i_k}^{p_{i_1}^{j_1} \dots p_{i_k}^{j_k}} \cdot \left(x_{i_1}^{s_l}\right)^{p_{i_1}^{j_1}} \cdot \left(x_{i_2}^{s_l}\right)^{p_{i_2}^{j_2}} \dots \left(x_{i_k}^{s_l}\right)^{p_{i_k}^{j_k}} +$$

$$+ \sum_{\forall(i_1 \dots i_k)\in K_2} \; \sum_{\forall(j_1 \dots j_k)\in K_2(i_1 \dots i_k)} c_{i_1\, \dots\, i_k}^{t_{i_1}^{j_1} \dots t_{i_k}^{j_k}} \cdot \left(x_{i_1}^{s_l}\right)^{t_{i_1}^{j_1}} \cdot \left(x_{i_2}^{s_l}\right)^{t_{i_2}^{j_2}} \dots \left(x_{i_k}^{s_l}\right)^{t_{i_k}^{j_k}} \le y \qquad y \ge 0, \quad l=\overline{1,p};$$

$$\sum_{\forall(i_1 \dots i_k)\in K_0} a_{i_1 \dots i_k} \cdot 1\left(x_{i_1}^{i} \dots x_{i_k}^{i}\right) + \sum_{\forall(i_1 \dots i_k)\in K_1} \; \sum_{\forall(j_1 \dots j_k)\in K_1(i_1 \dots i_k)} \pm b_{i_1\, \dots\, i_k}^{p_{i_1}^{j_1} \dots p_{i_k}^{j_k}} \left(x_{i_1}^{i}\right)^{p_{i_1}^{j_1}} \cdot \left(x_{i_2}^{i}\right)^{p_{i_2}^{j_2}} \dots \left(x_{i_k}^{i}\right)^{p_{i_k}^{j_k}} +$$

$$+ \sum_{\forall(i_1 \dots i_k)\in K_2} \; \sum_{\forall(j_1 \dots j_k)\in K_2(i_1 \dots i_k)} c_{i_1\, \dots\, i_k}^{t_{i_1}^{j_1} \dots t_{i_k}^{j_k}} \cdot \left(x_{i_1}^{i}\right)^{t_{i_1}^{j_1}} \cdot \left(x_{i_2}^{i}\right)^{t_{i_2}^{j_2}} \dots \left(x_{i_k}^{i}\right)^{t_{i_k}^{j_k}} \ge \sum_{\forall(i_1 \dots i_k)\in K_0} a_{i_1 \dots i_k} \cdot 1\left(x_{i_1}^{j} \dots x_{i_k}^{j}\right) +$$

$$+ \sum_{\forall(i_1 \dots i_k)\in K_1} \; \sum_{\forall(j_1 \dots j_k)\in K_1(i_1 \dots i_k)} \pm b_{i_1\, \dots\, i_k}^{p_{i_1}^{j_1} \dots p_{i_k}^{j_k}} \left(x_{i_1}^{j}\right)^{p_{i_1}^{j_1}} \cdot \left(x_{i_2}^{j}\right)^{p_{i_2}^{j_2}} \dots \left(x_{i_k}^{j}\right)^{p_{i_k}^{j_k}} +$$

$$+ \sum_{\forall(i_1 \dots i_k)\in K_2} \; \sum_{\forall(j_1 \dots j_k)\in K_2(i_1 \dots i_k)} c_{i_1\, \dots\, i_k}^{t_{i_1}^{j_1} \dots t_{i_k}^{j_k}} \cdot \left(x_{i_1}^{j}\right)^{t_{i_1}^{j_1}} \cdot \left(x_{i_2}^{j}\right)^{t_{i_2}^{j_2}} \dots \left(x_{i_k}^{j}\right)^{t_{i_k}^{j_k}}$$

$$\forall\left(i, j\right) \in I$$

Переменными задачи линейного программирования являются: y; $a_{i_1 \dots i_k}$; $b_{i_1\, \dots\, i_k}^{p_{i_1}^{j_1} \dots p_{i_k}^{j_k}}$; $c_{i_1\, \dots\, i_k}^{t_{i_1}^{j_1} \dots t_{i_k}^{j_k}}$.

$$\min \sum_{l=1}^{p} y_l$$

$$-y_l \le f\left(\overline{x}^{s_l}\right) - \sum_{\forall(i_1\ldots i_k)\in K_0} a_{i_1\ldots i_k}\cdot 1\left(x_{i_1}^{s_l}\ldots x_{i_k}^{s_l}\right) + \sum_{\forall(i_1\ldots i_k)\in K_1}\ \sum_{\forall(j_1\ldots j_k)\in K_1(i_1\ldots i_k)} \pm b_{i_1\ \ldots\ i_k}^{p_{i_1}^{j_1}\ldots p_{i_k}^{j_k}}\cdot\left(x_{i_1}^{s_l}\right)^{p_{i_1}^{j_1}}\cdot\left(x_{i_2}^{s_l}\right)^{p_{i_2}^{j_2}}\ldots\left(x_{i_k}^{s_l}\right)^{p_{i_k}^{j_k}} +$$

$$+\sum_{\forall(i_1\ldots i_k)\in K_2}\ \sum_{\forall(j_1\ldots j_k)\in K_2(i_1\ldots i_k)} c_{i_1\ \ldots\ i_k}^{t_{i_1}^{j_1}\ldots t_{i_k}^{j_k}}\cdot\left(x_{i_1}^{s_l}\right)^{t_{i_1}^{j_1}}\cdot\left(x_{i_2}^{s_l}\right)^{t_{i_2}^{j_2}}\ldots\left(x_{i_k}^{s_l}\right)^{t_{i_k}^{j_k}} \le y_l \qquad y_l \ge 0,\quad l=\overline{1,p};$$

$$\sum_{\forall(i_1\ldots i_k)\in K_0} a_{i_1\ldots i_k}\cdot 1\left(x_{i_1}^{i}\ldots x_{i_k}^{i}\right) + \sum_{\forall(i_1\ldots i_k)\in K_1}\ \sum_{\forall(j_1\ldots j_k)\in K_1(i_1\ldots i_k)} \pm b_{i_1\ \ldots\ i_k}^{p_{i_1}^{j_1}\ldots p_{i_k}^{j_k}}\left(x_{i_1}^{i}\right)^{p_{i_1}^{j_1}}\cdot\left(x_{i_2}^{i}\right)^{p_{i_2}^{j_2}}\ldots\left(x_{i_k}^{i}\right)^{p_{i_k}^{j_k}} +$$

$$+\sum_{\forall(i_1\ldots i_k)\in K_2}\ \sum_{\forall(j_1\ldots j_k)\in K_2(i_1\ldots i_k)} c_{i_1\ \ldots\ i_k}^{t_{i_1}^{j_1}\ldots t_{i_k}^{j_k}}\cdot\left(x_{i_1}^{i}\right)^{t_{i_1}^{j_1}}\cdot\left(x_{i_2}^{i}\right)^{t_{i_2}^{j_2}}\ldots\left(x_{i_k}^{i}\right)^{t_{i_k}^{j_k}} \ge \sum_{\forall(i_1\ldots i_k)\in K_0} a_{i_1\ldots i_k}\cdot 1\left(x_{i_1}^{j}\ldots x_{i_k}^{j}\right) +$$

$$+\sum_{\forall(i_1\ldots i_k)\in K_1}\ \sum_{\forall(j_1\ldots j_k)\in K_1(i_1\ldots i_k)} \pm b_{i_1\ \ldots\ i_k}^{p_{i_1}^{j_1}\ldots p_{i_k}^{j_k}}\left(x_{i_1}^{j}\right)^{p_{i_1}^{j_1}}\cdot\left(x_{i_2}^{j}\right)^{p_{i_2}^{j_2}}\ldots\left(x_{i_k}^{j}\right)^{p_{i_k}^{j_k}} +$$

$$+\sum_{\forall(i_1\ldots i_k)\in K_2}\ \sum_{\forall(j_1\ldots j_k)\in K_2(i_1\ldots i_k)} c_{i_1\ \ldots\ i_k}^{t_{i_1}^{j_1}\ldots t_{i_k}^{j_k}}\cdot\left(x_{i_1}^{j}\right)^{t_{i_1}^{j_1}}\cdot\left(x_{i_2}^{j}\right)^{t_{i_2}^{j_2}}\ldots\left(x_{i_k}^{j}\right)^{t_{i_k}^{j_k}}$$

$$\forall(i,j)\in I$$

Переменными задачи линейного программирования являются:

$$y_l;\ a_{i_1\ldots i_k};\ b_{i_1\ \ldots\ i_k}^{p_{i_1}^{j_1}\ldots p_{i_k}^{j_k}};\ c_{i_1\ \ldots\ i_k}^{t_{i_1}^{j_1}\ldots t_{i_k}^{j_k}}.$$

При анализе альтернатив $A_j \quad j=\overline{1,m}$, задаваемых вектором $\overline{x}$, $x=\left(x_1\ldots x_n\right)^{\mathrm{T}}$, может оказаться, что более естественным является построение функции глобальной цели $f\left(\overline{x}\right)$ как функции вида $f\left(A\overline{x}\right)$, где A – прямоугольная матрица размером $\left(l\times n\right)$ $l\le n$, где вектор $\overline{y}=A\overline{x}$ это вектор агрегированных характеристик альтернативы (проекта), которые более естественно, чем компоненты вектора $\overline{x}$ могут быть использованы для формализации неизвестной функции глобальной цели. В этом случае формально ранее приведенные модели остаются неизменными с учетом замены в них векторов $\overline{x}^{s_l}$, $\overline{x}^{i}$, $\overline{x}^{j}$ на вектора $\overline{y}^{s_l}=A\overline{x}^{s_l}$, $\overline{y}^{i}=A\overline{x}^{i}$, $\overline{y}^{j}=A\overline{x}^{j}$.

В этом случае эксперты строят множества K_0, K_1, $K_1\left(y_{i_1}\ldots y_{i_k}\right)$, K_2, $K_2\left(y_{i_1}\ldots y_{i_k}\right)$, анализируя влияние на функцию вектора $\overline{y}=A\overline{x}$.

Выводы

В данной статье формализация качественно заданной функции цели с использованием метода иерархий Саати сводится к задаче линейного программирования. Формализация качественно заданной функции цели позволяет автоматизировать процесс выбора наилучшей альтернативы из произвольного набора альтернатив без привлечения экспертов, т.е. без дополнительных временных и финансовых затрат.

Список литературы

1. Saaty T.L. Multicriteria Decision Making. The Analytic Hierarchy Process,. – New York: McGraw Hill International, 1990.– p.437.
2. 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.
3. 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.
4. Тоценко В.Г. Методы и системы поддержки принятия решений. Алгоритмический аспект. – Киев: Наукова думка. – 2002. – 381 с.
5. Павлов А.А., Иванова А.А., Зигура Р.А. Метод группового учёта аргументов и анализа иерархий (МГУАИАИ) в задачах принятия решений. Вісник НТУУ „КПІ” Інформатика, управління та обчислювальна техніка, Київ 2008р. №47. –350 с. – С.205-214.

УДК 681.3.06

БУХАНЦОВ С.Н.,
ПУСТОВАРОВ В.І.,
СТІРЕНКО С.Г.
ЖЕЛУДКОВА Т.В.

ДОКАЗОВЕ УПРАВЛІННЯ ГЕНЕРАЦІЄЮ ТА МОДИФІКАЦІЄЮ ПРОГРАМНИХ МОДУЛІВ ДЛЯ ІИНФОРМАЦІЙНО-АНАЛІТИЧНИХ СИСТЕМ

Розглянуті типові послідовності розв'язання задач інформаційно-аналітичних систем (ІАС) та модульна структура їх спеціальних програм. Запропонована методика визначення стану процесів для досягнення цілей оперативних регламентних робіт з підготовки та оцінки рішень з проектів, супроводжуваних в ІАС. Показано, що застосування цієї методики дозволяє обрати структуру програм для їх ефективного використання в критичних ситуаціях, що виникають в ІАС, інтегрованих з підсистемами обліку та розв'язання регламентних задач.

Typical sequences for information-analytic system (IAS) task decisions and modular structure of their special programs are considered. The techniques of task state definition for meeting operative regulated work purposes decision on preparation and estimation are offered for projects supported in IAS. It is shown that application of this technique allows program structure optimizing for their effective usage in the critical situations arising in IAS, integrated with accounting and regulated tasks decision subsystems.

Вступ

Корпоративні системи ІАС, їх структура та проблеми в розв'язанні задач у великих загальних обсягах обробки в корпоративній комп'ютерній системі та негарантованій регулярності задач. Для практичної реалізації спеціального програмного забезпечення (СПЗ) інформаційно-аналітичних системах ІАС з настроюванням на основі цих технологій важливо створити механізми оперативного настроювання цих технологій для задач прийняття рішень. Перспективним підходом до розв'язання задач автоматизації ППР є включення таблиць супроводження реалізації рішень безпосередньо в робочу середовище реляційних СУБД.

Головні дії функцій ІАС забезпечують підготовку рішень на основі аналізу ситуацій і стану системи [1, 2]. Залежно від області застосування (будь-то комерційне підприємство або фінансова установа) процес підготовки і вимоги до ефективності рішень мають свої особливості. На різних підприємствах підготовка і реалізація оперативних рішень стосуються задач документообігу, обліку наявних ресурсів і продукції, аналізу обсягів продаж готової продукції і закупівель ресурсів та аналіз доцільності випуску рекламної інформації про комерційні проекти, тощо. Для забезпечення ефективності фінансових структур видобувається інформація про клієнтів, доходи/витрати за різні періоди, прибутки, збитки, обсяги кредитів, обсяги депозитів клієнтів, інформація про конкурентів і про ситуації на відповідних ринках.

Вибір загальної структури та правил роботи СПЗ ІАС

З одного боку в ІАС треба забезпечити оперативну обробку запитів у реальному часі, своєчасне розв'язання задач регламенту і підготовки звітності та розумні витрати часу на підготовку і прийняття рішень за новими проектами контрольованого об'єкта. З іншого боку при створенні виконавчих функцій ефективного СПЗ важливо уникнути повторної обробки запитів або запитів, що частково перекриваються.

Структура задач і відповідних процедурних модулів СПЗ спирається на ієрархію підпрограм різних рівнів, яка включає:

- задачі підготовки, настроювання та навчання шляхом реконфігурації та зміни функціональних атрибутів управління в таблицях рішень, коефіцієнтів і базовій послідовності обробки, які формуються на основі прецедентів реалізації і моделювання рішень та їх наслідків; корекція правил логічного доведення для покращення наслідків рішень;
- архівування: запам'ятовування фактів і прецедентів реалізації рішень;
- задачі попередньої обробки даних: відбраківка та фільтрація даних за критеріями та статистиками ефективності, а також накопи-

чення попередніх підсумків та формування проміжних даних.

- задачі функціонального рівня оперативної обробки в ІАС: забезпечення оперативних трансакцій; аналіз поточних балансів і прогнозованих ситуацій; формування звітності за регламентом; супроводження реалізації рішень через контроль планових подій; розпізнавання подій, здатних вплинути на ситуацію в проекті або установі; аналітичних висновків і прийняття рішень за запитами та неплановими і критичними подіями;
- підрахунок статистик та підсумків семантичних атрибутів для оцінки станів і ситуацій на контрольованих об'єктах: як узагальнення процедур підбиття підсумків;
- контроль необхідного моменту запуску чергової задачі для розв'язання на поточних наявних ресурсах ІАС;
- визначення характеристик переваг для вибору першочергової задачі для розв'язання на поточних наявних ресурсах ІАС.

Щоб забезпечити своєчасне розв'язання кожної з задач, доцільно виділити спеціальну групу задач підрахунків підсумків, що дозволяють підготувати, як часткові, так повні підсумки залежно від наявності даних на момент виконання обробки регламентних задач. Крім того, треба забезпечити управління порядком виконання окремих процедур. До складу елементів базових послідовностей розв'язання всіх перелічених класів задач входять наступні процедури розв'язання часткових задач та задач проблемного рівня ІАС:

- часткове підбиття підсумків для задач регламенту і підготовки та аналізу рішень: для цього зручно використовувати технології накопичення та вибірки підсумків OLAP;
- управління за подіями, що керує оперативними трансакціями, аналізує поточні та прогнозовані ситуації, супроводжує реалізації рішень за плановими подіями, розпізнає події, здатні вплинути на стан ресурсів і ситуацію в ІАС, приймає рішення за неплановими і критичними подіями;
- формування або підготовка рішень проектів за таблицями створення і корекції планів реалізації рішень при зміні стану і ситуацій в контрольованому об'єкті та ІАС;
- логічне доведення та пояснення доцільності прийняття і виконання рішень: за результатами дослідження ситуацій в ІАС в цілому і стану ресурсів при реалізації окремих проектів, а також за результатами аналізу наслідків реалізації прецедентів проектів.

Таким чином, показані 3 рівні задач утворюють ієрархію, елементи якої одержують інформативні і управляючі дані від задач нижчих рівнів.

Структури вхідних, вихідних та внутрішніх даних корпоративних ІАС

На сьогоднішній день аналітичні задачі, що розв'язуються в багатьох відомих ІАС, забезпечують виконання регламентних задач та задач підтримки прийняття рішень [1, 3]. Використання технологій OLAP (On-Line Analytic Processing) на основі багатомірних сховищ даних (Warehouses) [2] та таблиць рішень дозволяють накопичити вхідні та проміжні дані для аналізу ситуацій і рішень, що формуються в ІАС. Органічні зв'язки цієї технології з технологіями реляційних СУБД роблять перспективною інтеграцію різноманітних підсистем підтримки прийняття рішень в ІАС на базі функцій підсистеми обліку.

Основу математичної моделі будь-якого об'єкта інформатизації можна представити ***вузлами графу***, що відображують інформаційні об'єкти, і ***дугами***, що відображають зв'язки між полями окремих таблиць. В ІАС, побудованих на основі реляційних СУБД з використанням технологій OLAP, основними сховищами даних є ***координатні таблиці*** $\boldsymbol{r}_{ck}(x_{k1},\ldots, x_{ki\max}, \xi_{k1},\ldots, \xi_{kj\max}, a_{k1},\ldots, a_{kn\max}))$ та ***куби*** $\prod_k \boldsymbol{r}_{fk}(\sigma_{k1}(\xi_{k1}),\ldots,\sigma_{kj}(\xi_{kj\max}))|\ x_{k1},\ldots, x_{ki\max}, \xi_{k1},\ldots, \xi_{kj\max}$, де k – номери координат, $x_{k1},\ldots, x_{ki\max}$ – поля ідентифікації, $\xi_{k1},\ldots, \xi_{kj\max}$ – поля семантично істотних даних, $a_{k1},\ldots, a_{kn\max}$ – поля додаткових даних обліку. Вони розглядаються як об'єктно-орієнтовані ***елементи*** моделі даних ІАС, з пов'язаними з ними ***процедурами*** і ***функціями*** підбиття підсумків σ_{kj}.

Протоколи інкапсулюють об'яви предиката, відділяючи інтерфейс від реалізації, а категорії інкапсулюють і об'яви, і визначення предикатів, які можуть бути імпортовані будь-яким об'єктом.

Куби $\prod_k \boldsymbol{r}_{fk}$ використовуються для підготовки і прийняття внутрішніх рішень при виникненні особливих ситуацій у стані організації і реалізації планів, при укладанні контрактів, а також для оцінки достовірності роботи підсистеми збору інформації з комп'ютерних мереж з відкритим доступом і з засобів масової інформації. Основні координатні таблиці $\boldsymbol{r}_{ck}$ зірко-

подібних сховищ та сховищ за схемою сніжинки визначають напрямки для формування підсумків семантичних даних, за якими приймають рішення. Комплекси таких таблиць зберігають дані ІАС, зв'язані з роботою окремих структурних підрозділів, співробітників та попитом на продукцію.

Особливості специфікації задач аналізу ресурсів ІАС. Традиційна структура формальних специфікацій процедур за принципами Z-специфікацій та інших систем специфікацій [3, 4, 5] в найбільш загальній формі визначає цілі комплексу задач ІАС, тобто цільові змінні та критерії розв'язання задач у формі:

Назва_специфікації
Ξ *Ім'я_функції*
Список_аргументів ::=
{*Ім'я_вхідного _аргументу*?: *Тип_аргументу*
…}
{*Ім'я_вихідного _аргументу*!: *Тип_аргументу*
…}
Список_зв'язків ::=
{*Вираз_множинного_відношення*|
Вираз_числового_відношення
…}

Тут входи і виходи визначають гіперкуби $\Pi_k \boldsymbol{r}_{fk}$, таблиці $\boldsymbol{r}_{ck}$ та прості або підсумкові семантичні змінні ξ_j та $\sigma_{kj}(\xi_{kj\max})$, які є або фактичними аргументами підпрограм, або глобальними даними моделі проблемної області. Модель проблемної області будується як супероб'єкт специфікації, який має опис, що поєднує ідентифікатори та заголовки специфікацій всіх функцій.

Змістовні цілі для перелічених задач і проміжних функцій ІАС відносно вхідних даних та таблиць об'єктів, що відображують множини ресурсів, проектів та трансакцій, можна представити засобами Z-специфікацій. Для специфікації таких задач доцільно використовувати обмежену комп'ютерну форму, яка включає зв'язки алгебри множин. Коди скриптів та прикладних програм звичайно будуються в рамках мови SQL або сполучення операторів мови програмування та Internet-скриптів з генерацією SQL-скриптів. До складу специфікацій SQL-запитів на вибірки та підбиття підсумків важливо включити додаткові умови фільтрації та адитивні функції підбиття часткових і повних підсумків. Доказове обґрунтування досягнення цілей вручну виконується за методикою формальних перетворень описів на фрагменти програм, описаною в [4].

Для досягнення можливостей підвищення ефективності роботи модуля в різних ситуаціях та вибору режиму і порядку його часткового виконання до розширеної Z-специфікації слід включити критерії часу виконання, які є вторинними щодо змісту обробки. Комп'ютерна реалізація цієї методики використовує загальний набір правил контролю відповідностей змістовних специфікацій фрагментам або процедурам комп'ютерних програм у формі Бекуса-Наура, який потрібно використовувати для формальних перетворень з контролю або синтезу виконавчих кодів.

Правило_перетворення ::=
Фраза_змістовної_специфікації → *Фраза_команд_комп'ютерної_мови*

Звичайно фрази змістовної специфікації об'єктів включають термінальні позначення дій і даних у формі операторів, операцій або їх імен. В цих правилах перетворення повинні бути обґрунтовані коректні послідовності дій, що відповідають елементам специфікації. Крім того, для умов і зв'язків специфікацій повинен бути визначений порядок аналізу цих елементів у фразах програмних кодів. Це відкриє шлях до автоматизованого доведення коректності за специфікаціями в системах програмування та проектування інтегрованого СПЗ.

Доказове обґрунтування досягнення змістовних цілей, заданих специфікаціями

Рекурсивне застосування правил перетворень при формальній верифікації можна використовувати через підстановки, що починаються як з боку специфікацій, так і з боку текстів. При цьому як ключі пошуку неперетворених нетермінальних позначень повинні обиратися з відповідної частини операції перетворення "→" в правилах. Фрази з послідовностями операторів повинні бути попередньо обґрунтовані вручну або, починаючи зі сторони мов програмування, через використання звичайних правил підстановок для числової та

множинної алгебри. Варіанти алгоритмів можуть подаватися за правилами декомпозицій задач, підзадач та їх елементів: послідовної, паралельної, умовної та циклічної декомпозицій. Правила підстановок для підготовки оперативних рішень з підтримки балансів та супроводження проектів можна обмежити використанням правил паралельної та послідовної декомпозиції наборів викликів процедур:

фраза_команд_алгоритмічної_мови ::=

$$P_{p1}, P_{p2},\ldots, P_{pn},\ldots \quad (1)$$

фраза_команд_алгоритмічної_мови ::=

$$P_{s1}; P_{s2};\ldots; P_{sn};\ldots \quad (2)$$

Умовами запуску на виконання перелічених процедур P_{pn} і P_{sn} є наявність потрібних вхідних даних, сформованих підсистемою відбору та формування вхідних даних ІАС. У випадку підбиття проміжних підсумків, коли вхідні дані надходять з критичною затримкою, ці задачі варто запускати частково за наявності даних в розраховані моменти часу.

Загальне доказове обґрунтування досягнення основних семантичних цілей виконується через застосування основних правил обробки даних в ІАС і використовує ланцюжки доказових перетворень, запропоновані в [4] і [5].

Аналітичне супроводження доказових перетворень при генерації та модифікації управляючих даних програм

Доказові перетворення використовують ланцюжки перетворень з відповідними операціями імплікативних '⇒' та еквівалентних '≡' перетворень для виразів процедур P_{sk}. Потрібні дії можуть виконуватися як вручну, так і в автоматизованому режимі з протоколюванням доведень для подальшої перевірки та наступних доведень.

Створені ланцюжки повинні забезпечувати перетворення виразів формул та операторів до варіантів, тотожних за правилами з базі знань перетворень. В протоколах перетворень зберігаються пояснення адекватності через посилання на використані правила перетворень у формах:

Вирази_специфікацій ⇒
вирази_комп'ютерної_мови|*заголовок_правила*

або

Вирази_специфікацій ≡
вирази_комп'ютерної_мови|*заголовок_правила*

Доказове обґрунтування досягнення вторинних цілей, заданих специфікаціями

Для специфікацій вторинних критеріїв для вторинних характеристик працездатних програмних модулів слід використовувати відношення над вторинними характеристиками витрат при виконанні програмних модулів.

Специфікація_вторинних_характеристик ::=
вираз_відношення_обмежень| *вираз_критерію*

Ці відношення не суперечать загальним принципам побудови специфікацій і використовуються для розширення існуючих систем специфікацій задач на комп'ютерах. Інформацію про вторинні характеристики слід зберігати в паспортах підпрограм, що є розширенням їх традиційних специфікацій у формі файлів заголовків. Для одержання такої інформації достатньо визначити спільні функціонали, які визначаються в заголовках за формулами (3) і (4) та використовуються в процесах доведень ефективності та своєчасності рішень.

Умови коректного і своєчасного розв'язання задач ІАС

Традиційні задачі виконання регламентних робіт та задачі підбиття та аналізу підсумків забезпечують повний комплекс обчислень в ІАС. Ці задачі повинні бути виконані до настання заданого моменту в повному або обмеженому обсязі. Умови контролю ефективності або своєчасності розв'язання складають вторинний рівень специфікацій задач. Для залишкових часток P_{pn} і P_{sn} з правил (1) і (2) повного комплексу задач ІАС $\cup_{n\in 1..n_{max}} P_{sn}$ найбільшу важливість мають оцінки часової складності $O(P_{sn}, N_r)$ відносно обсягів залишкових даних N_r у оброблюваних відношеннях

$$O_s(N_r) = \max_{n \in N_r} O(P_{sn}, N_r), \quad \text{де} \quad N_r = 1..n_{max}, \quad (3)$$

що дозволяє визначити потенційну можливість реалізації залишкової обробки відносно обсягу залишкових даних. Тут N_r – кількість блоків обробки в множині елементів залишку обробки з обсягом вхідних даних залишку та часові характеристики обробки $T(P_{sn})$ конкретних реалізацій процедур P_{sn}.

$$T_s(N_r) = \sum_{n \in N_r} T(P_{sn}) < \Delta t, \text{ де } \Delta t = t_r - t_c. \quad (4)$$

Розрахунок залишку задач і порівняння залишку часу від поточного часу t_c до граничного часу завершення регламентних робіт t_r. Для більшості обчислень регламенту витрати часу на них (4) достатньо точно можна оцінити, враховуючи специфіку підбиття підсумків при

їх розрахунку і розв'язанні задач настроювання та навчання.

Вибір загальних правил проектування та настроювання роботи СПЗ ІАС

З одного боку в ІАС треба забезпечити оперативну обробку запитів у реальному часі, своєчасне розв'язання задач регламенту і підготовки звітності та розумні витрати часу на підготовку і прийняття рішень за новими проектами контрольованого об'єкта. З іншого боку при створенні виконавчих функцій ефективного СПЗ важливо уникнути повторної обробки запитів або запитів, що частково перекриваються. При проектуванні і настроюванні порядку виконання особливу роль грають задачі часткового підбиття підсумків з наступних збереженням їх результатів в елементах кубічних структур, які доцільно виконувати в фоновому режимі під час підвищеної активності трансакцій.

Вибір порядку виконання задач та підзадач за модулями СПЗ

При розв'язанні задач з регламентованим та критичним часом виконання, тобтопобудови регламентних звітів та формування параметрів рішень проектів необхідно використовувати управління послідовністю виконання в наступному вигляді. Правила пріоритетності для трьох основних класів задач ІАС визначаються наступним чином:

- задачі виконання базових трансакцій ІАС запускаються за подіями запитів і виконуються у режимі безумовно вищої пріоритетності;
- задачі виконання регламентних робіт запускаються за регламентним часом і подіями наявності потрібних блоків даних та часткових підсумків і виконуються з пріоритетністю, що визначається залишками часу на їх виконання, в період мінімального обсягу робіт за трансакціями;
- задачі формування параметрів та оцінки проектів рішень, які виконуються в низько фоновому режимі співставлення з прецедентами, архівами та оперативними даними про стан об'єктів уваги.

Таким чином, для доказової розробки і оперативного настроювання СПЗ необхідно виконати наступну послідовність дій.

1. Визначити базову структуру окремих програмних модулів та послідовність функціональних фрагментів або викликів підпрограм.
2. Побудувати окремі модулі за схемою, що включає аналіз змістовних шаблонів обробки, блоків перевірки таблиць і критеріїв прийняття рішень для визначення їх параметрів.
3. Обґрунтувати відповідність базових функцій специфікаціям зв'язків семантичних змінних.
4. Визначити залишок часу виконання критичних підзадач та обмежити час запуску у випадку ненадходження відповідної інформації.

Процес доказової підготовки та настроювання функціональних модулів СПЗ повинен включати аналітичний контроль семантичних характеристик і характеристик продуктивності модулів, критичних за часом виконання [1, 2].

Висновки

Таким чином, визначені в процесі проектування або експлуатації основні і проміжні цілі моделей ІАС, а також проміжні підсумки можуть використовуватися для ефективної організації процесів обчислень. Автоматизація процесів доказових перетворень на етапах проектування, настроювання та оперативної обробки дозволить покращити характеристики цільових систем управління і характеристики СПЗ при використанні критеріїв ефективності, що відповідають поточній ситуації в ІАС. Накопичення протоколів доведення та обґрунтування дозволить спростити процеси доведення коректності і своєчасності.одержання рішень при зміні параметрів стану і ситуацій.

Список посилань

1. Буханцов С.Н., Пустоваров В.И. Интеллектуальный сбор данных для управления информационно-аналитическими системами – Вісник НТУУ "КПІ". Інформатика, управління та обчислювальна техніка – К.: «Век+». – 2006.– 45.– с. 228 – 236.
2. Бэлсон Д., Гокмен М. Ингрем Дж. Внутренний мир Oracle8. Проектирование и настройка: – К.: Изд-во "ДиаСофт". – 2000. – 800 с.
3. Лисков Б., Гатэг Дж. Использование абстракций и спецификаций при разработке программ. Пер. с англ. М.: Мир, 1989. 424 с.
4. Hehner E.C.R. Practical theory of programming. Springer-Verlag, New York, 1993 – 243 p.
5. Woodcock J., Davies J. Using Z. Specification, Refinement, and Proof. C.A.R. Hoare Series editor, 1995 – 390 p.

УДК 004.3

БЛІНОВА Т. О.

ДОДАВАННЯ ТА МНОЖЕННЯ В КОДАХ ЗОЛОТОЇ ПРОПОРЦІЇ В ПОРОЗРЯДНОМУ РЕЖИМІ, ПОЧИНАЮЧИ ЗІ СТАРШИХ РОЗРЯДІВ

Автором пропонується нестандартна мінімізована форма (MN-форма) представлення чисел у коді класичної золотої пропорції. У цій формі може бути представлене будь-яке дійсне число, причому в коді будуть відсутні послідовності цифр "111". Використання MN-форми дозволяє здійснювати та контролювати порозрядні обчислення, виконуючи арифметичні операції в MSDF-режимі (починаючи зі старшого розряду). Розроблено алгоритми виконання деяких арифметичних операцій, зокрема, алгебраїчного додавання та множення, для чисел, представлених у MN-формі.

Author proposes to write the golden ratio base numbers in the nonstandard minimized form (MN-form). Any real number can be presented in this form avoiding the digit sequence "111”. The MN-form use allows to carry out and control on-line computations, executing arithmetic operations in the Most Significant Digit First (MSDF)-mode. The algorithms of execution of some arithmetic operations, including the addition, subtraction and multiplication, are developed for MN-form numbers.

Проблеми побудови відмовостійких та контролездатних обчислювальних засобів є завжди важливими. У [1] показана ефективність використання системи АЦП – оптичний канал – ЦАП, яка працює цілком у класичному коді Фібоначчі або золотої пропорції, у середовищах із великими індустріальними перешкодами. Система містить АЦП та ЦАП, що самокоректуються, і цифровий оптичний канал, який нечутливий до електромагнітних перешкод, правильність передачі інформації по котрому може контролюватися за рахунок надмірності кодів.

При обробці інформації, яка передається такою системою, доцільно використовувати операційні пристрої, котрі працюють у коді класичної золотої пропорції (коді золотої 1-пропорції), щоб уникнути зайвих операцій по перекодуванню сигналів у(з) двійковий код та використати додаткові можливості по контролю інформації. Отже, завданням таких пристроїв буде обробка інформації, що порозрядно надходить, напрклад, з оптичного каналу та представлена в коді золотої 1-пропорції у вигляді мінімізованої форми (МН-форми), код якої може містити послідовності не більш ніж із двох одиниць [1].

Питанням виконання арифметичних операцій у коді золотої пропорції (золотого перетину) присвячені роботи, наведені в [2], зокрема, це роботи [3-6]. Виконанню операцій у порозрядному, починаючи з найбільш значущих, режимі (MSDF-режимі), присвячена робота [7]; пристрої, що виконують у MSDF-режимі обчислення в класичному коді Фібоначчі або золотої пропорції були створені автором статті, зокрема, це роботи [8-11].

Метою даної статті є опис розроблених автором алгоритмів виконання операцій алгебраїчного додавання та множення над числами, що поступають порозрядно, починаючи зі старших розрядів, і представлені в коді золотої 1-пропорції у вигляді мінімізованої форми.

Операції, що здійснюються в MSDF-режимі над числами, представленими в МН-формі, можна розділити на дві групи. До першої належатимуть базові операції, а саме: неповна згортка, розгортка, порівняння, перевід чисел із прямого коду в доповняльний, зсув та підсумовування. До другої групи – усі інші арифметичні операції: алгебраїчне додавання, множення, ділення тощо. Виконання операцій із другої групи зводяться до здійснення операцій із першої групи. Так, наприклад, виконання операції множення зводиться до послідовного здійснення операцій підсумовування, причому воно супроводжується виконанням операції неповної згортки над сумами часткових добутків. Чергове підсумовування починається після отримання перших цифр чергового проміжного результату, без очікування закінчення попередніх перетворень. Контрольні операції виконуються на кожному ітераційному кроці паралельно з обчислювальними операціями.

При обробці МН-форм на кожному кроці формується, як правило, один розряд результату, значення котрого на наступних ітераційних кроках обчислень не змінюється. Отже,

кількість виконуваних обчислень не змінюється, вона залежить від розрядності результату.

Операція неповної згортки використовується для приведення кодових слів до МН-форми представлення. Вона полягає у виконання всіх можливих згорток при виконанні одного перегляду коду числа, починаючи зі старших розрядів.

Коди золотої 1-пропорції допускають представлення чисел як у формі з фіксованою точкою, так й у формі з плаваючою точкою. Далі будемо детальніше розглядати операції над числами з фіксованою точкою ($|D|<1$):

$$D = \sum_{i=1}^{m} d_i a_1^{-i}, |D| = \sum_{i=1}^{m} d_i a_1^{-i},$$

де $d_i=\{0,1\}$ – i-й розряд кодового слова;
$\alpha_1 = (1+\sqrt{5})/2$ – основа системи числення.

Для представлення від'ємних чисел використовується доповнення числа D до α_1^2, що сприяє розробці простих алгоритмів обчислень у MSDF-режимі і дозволяє всі розряди (зі знаковими включно) обробляти за однаковими правилами. Формування доповняльного коду від'ємного числа зводиться до інвертування розрядів кодового слова; запису одиниць у розряди з вагою $\alpha_1^{-(m+1)}$ та $\alpha_1^{-(m+2)}$; виконанню неповної згортки; запису одиниці у розряд із вагою $\alpha_1^{-(m+1)}$ та виконанню неповної згортки. При цьому формуються два знакових розряди: 00 для кодування знака "+" та 10 для "–" [12].

Розряди кінцевого результату $Y = \sum_{i=-1}^{m+1} y_i a_1^{-i}$, формуються, починаючи із третього ітераційного кроку (при i=1). Зміна знака числа при перетворенні числа із прямого в доповняльний код або з доповняльного в прямий код свідчить про переповнення розрядної сітки. На кожному кроці перевіряється правильність виконання операції – якщо неповну згортку поточних розрядів неможливо провести (тобто в розряді, у який потрібно записати одиницю, вона вже є), реєструється помилка типу переходу 0 в 1.

Виконання зсуву на один розряд доповняльного коду від'ємного числа наліво або направо (тобто множення числа на α_1 або α_1^{-1}) виконується наступним чином: виконується зсув кодового слова праворуч (ліворуч) на один розряд ; до отриманого кодового слова приписується ліворуч (праворуч) 0; підсумувати 1 до розряду з вагою α_1^0.

Приклад зсуву на один розряд ліворуч доповняльного коду (10.1000001011) числа (–0.0010011000):

```
      зсув 01.0000010110;
        +   1.          ;
результат 10.0100010110.
```

Зазначимо, що в коді золотої 1-пропорції при підсумовуванні двох одиниць із вагою α_1^{-i} утворюються перенесення e_{i-1} та e_{i+2} у розряди з вагами $\alpha_1^{-(i-1)}$ та $\alpha_1^{-(m+2)}$ відповідно. Отже, при виконанні підсумовування двох операндів, представлених у МН-формі, необхідно враховувати чотири змінні.

При виконанні операції алгебраїчного додавання виконується операція підсумовування доповняльних кодів операндів, причому над усіма розрядами, включаючи знакові, здійснюються однакові перетворення. Перенесення у розряд із вагою α_1^{+2} не береться до уваги. Операція завершується виконанням неповної згортки.

Нехай виконується алгебраїчне додавання двох чисел – A та O. Позначимо розряди перенесення як e, проміжних сум як u, і результату як y. Тоді алгоритм цієї операції можна записати наступним чином.

1: $e_{-1} = e_0=0$;
$u_{-2}^{-2} = u_{-3}^{-2}= u_{-4}^{`} = u_{-5}^{``}=0$;
$i = -1$;
2: $e_{i+2} = E(a_i,o_i,e_i,u_{i-1}^{i-1})=o_ie_i\bar{u}_{i-1}^{i-1} \vee a_ie_i\bar{u}_{i-1}^{i-1} \vee a_io_i\bar{u}_{i-1}^{i-1} \vee a_io_ie_i$;
$u_i^i = U_0(a_i,o_i,e_i,u_{i-1}^{i-1})= \bar{a}_i\bar{o}_ie_i\bar{u}_{i-1}^{i-1} \vee a_i\bar{o}_i\bar{e}_i\bar{u}_{i-1}^{i-1} \vee \bar{a}_io_i\bar{e}_i\bar{u}_{i-1}^{i-1} \vee \bar{a}_io_i\bar{e}_iu_{i-1}^{i-1} \vee$
$\vee a_i\bar{o}_ie_iu_{i-1}^{i-1} \vee a_io_i\bar{e}_iu_{i-1}^{i-1}$;
$u_{i-1}^i = U_1(a_i,o_i,e_i,u_{i-1}^{i-1})=\bar{a}_io_ie_i\bar{u}_{i-1}^{i-1} \vee a_i\bar{o}_ie_i\bar{u}_{i-1}^{i-1} \vee a_io_i\bar{e}_i\bar{u}_{i-1}^{i-1} \vee a_io_ie_iu_{i-1}^{i-1}$;
$u_{i-2}^i = U_2(a_i,o_i,e_i,u_{i-1}^{i-1}, u_{i-2}^{i-1})=u_{i-2}^{i-1} \vee a_io_ie \vee \bar{a}_ie_iu_{i-1}^{i-1} \vee a_i\bar{o}_iu_{i-1}^{i-1} \vee o_ie_iu_{i-1}^{i-1}$;
3: $u_{i-2}^{`} = u_{i-2}\bar{u}_{i-3}^{`}$;
$u_{i-3}^{``} = \bar{u}_{i-2}^{`} u_{i-3}^{`}$;
$y_{i-4} = u_{i-4}^{``} \vee u_{i-2}\bar{u}_{i-3}^{`}$;
4: Якщо $i < m+4$, тоді $i = i+1$ і на 2;
5: Кінець.

Кінцеві значення розрядів результату починають формуватися при i = 3 (із затримкою у чотири ітераційних кроки). Для контролю правильності виконання операції на кожному кроці перевіряється, чи не виникли заборонені ситуації: $a_io_ie_iu_{i-2}^{i-1}=1$ або $u_{i-1}^{i-1}u_{i-2}^{i-1}=1$. При складанні чисел з однаковими знаками може

виникнути переповнення, ознакою чого є розбіжність знаків операндів і результату.

При виконанні операції множення кодів золотої 1-пропорції формування часткових добутків виконується за правилами, прийнятими в класичній двійковій системі числення. Для прискорення можна використовувати, наприклад, множення на два розряди множника, використовуючи можливість заміни групи розрядів 011 на групу 100.

Нехай виконується прискорене за вищезгаданим способом множення двох чисел – A та O. Позначимо розряди перенесення як e, проміжних сум як u, і результату як y. Тоді алгоритм цієї операції має наступний вигляд.

1: $i = k=1$;

2: $d_{i,k} = a_{i-2k+1}o_{2k-1}\, o_{2k} \vee a_{i-2k}o_{2k-1}\, \bar{o}_{2k} \vee a_{i-2k-1}\bar{o}_{2k-1}\, o_{2k}$;

$e_{i+2k} = E(y_{i,k-1},d_{i,k},e_{i,k},u_{i-1,k}^{\,i-1})$;

$u_i^{\,i} = U_0(y_{i,k-1},d_{i,k},e_{i,k},u_{i-1,k}^{\,i-1})$;

$u^{\,i}_{i-1} = U_1(y_{i,k-1},d_{i,k},e_{i,k},u_{i-1,k}^{\,i-1})$;

$u^{\,i}_{i-2} = U_2(y_{i,k-1},d_{i,k},e_{i,k},u_{i-1,k}^{\,i-1}, u_{i-2,k}^{\,i-1})$;

3: $u`_{i-2,k} = u_{i-2,k}\bar{u}`_{i-3,k-1}$;

$u``_{i-3,k} = \bar{u}`_{i-2,k}\, u`_{i-3,k-1}$;

$y_{i-4,k} = u``_{i-4,k-1} \vee u_{i-2,k-1}\bar{u}`_{i-3,k}$;

4: Якщо $i–2k < m+7$, тоді $i = i+1$ і на 2;

5: Якщо $k < m/2$, тоді $k = k+1$ і на 2;

6: Кінець.

При множенні чисел, представлених у прямому коді, знакові розряди обробляються окремо.

У загальному випадку добуток $[A]_\partial \times [O]_\partial$, отриманий при множенні доповняльних кодів $[A]_\partial$ та $[O]_\partial$ чисел A та O не дорівнює $[A\times O]_\partial$ і потребує корекції, якщо хоча б один із множників від'ємний.

Нехай $A<0$, $O<0$, тоді $[A]_\partial[O]_\partial=\alpha_1^4 - \alpha_1^2\,(|A| + |O|) + [A\times O]_\partial$. Отже для корекції необхідно до псевдодобутку додати зсунуті на два розряди ліворуч модулі обох операндів та відкинути два старших розряди (із вагами α_1^4 та α_1^3) отриманої суми.

Нехай $A>0$, $O<0$, тоді $[A]_\partial[O]_\partial= \alpha_1^2|A| - \alpha_1^2 + [A\times O]_\partial$. Отже для корекції необхідно до псевдодобутку додати зсунутий на два розряди ліворуч доповняльний код $[–|A|]_\partial$ та відкинути два старших розряди (із вагами α_1^4 та α_1^3) отриманої суми.

Приклад виконання корекції при множенні чисел A = ±0,1010 та O = ±0,1000 (для порівняння наводимо результати $[|A| \times |O|]_\partial$ = 00,01010000,

$[–|A|\times|O|]_\partial$=10,01001000):

($[–|A|]_\partial \times [|O|]_\partial$ = 10.0001× 00,1000 = 0001,00001000)

+ ($\alpha_1^2[–|O|]_\partial$ = 1001,00000000)

= 1010,01001000,

отже результат $[–|A|\times|O|]_\partial$= 10,01001000;

($[–|A|]_\partial \times [–|O|]_\partial$ = 10.0001× 10,0100 = 0100,10100100)

+ (α_1^2 ($|A|$ + $|O|$) = 0010,10+0010,00 = 0101,00100000)

= 1100,01010000,

отже результат $[|A|\times|O|]_\partial$= 00,01010000.

Таким чином, на прикладі деяких арифметичних операцій, зокрема, алгебраїчного додавання та множення, показано, як, використовуючи MN-форму представлення чисел у класичній золотій пропорції, можна здійснювати та контролювати порозрядні обчислення в MSDF-режимі.

Перелік посилань

1. Блінова Т.О. Використання кодів Фібоначчі та золотої пропорції в цифрових оптичних каналах передачі інформації //Вісник НТУУ "КПІ". Сер. Інформатика, управління та обчислювальна техніка. № 47. – 2007. С.255-261.
2. Институт золотого сечения. http://www.trinitas.ru/
3. Стахов А.П., Лужецкий В.А. Машинная арифметика ЦВМ в кодах Фибоначчи и золотой пропорции.– М.: Научный совет АН СССР по комплексной проблеме "Кибернетика", 1981. – 64 с.
4. Стахов А. П. Коды золотой пропорции. – М.: Радио и связь, 1984. –151с.
5. Stakhov A.P. Computer Arithmetic based on Fibonacci Numbers and Golden Section: New Information and Arithmetic Computer Foundations", Toronto, "SKILLSET-Training", 1997.
6. Компьютеры Фибоначчи и новая теория кодирования: история, теория, перспективы. Электронный журнал Таганрогского радиотехнического университета "Перспективные информационные технологии и интеллектуальные системы", № 2 (18), 2004. http://pitis.tsure.ru/files18/p5.pdf.
7. Ch. Frougny. On-line digit set conversion in real base. Theoret. Comp. Sci. – 2003, pp. 221-235. http://citeseer.ist.psu.edu/frougny00line.html.

8. Авт. св. СССР N 1137460 А, кл. G06F7/49, БИ 4, 1985. Конвейерный сумматор/ Луцкий Г.М., Блинова Т.А., Корочкин А.В.
9. Авт. св. СССР N 1254470 А1, кл. G06F7/49. БИ 32, 1986. Конвейерное операционное устройство/ Луцкий Г.М., Блинова Т.А., Дрофа С.В.
10. Авт. св. СССР N 1693601 А1, кл. G06F7/49. БИ. 47, 1991. Конвейерное вычислительное устройство/Блинова Т.А.
11. Авт. св. СССР N 1725401 А2, кл. Н03М13/12. БИ. 13, 1992. Устройство для преобразования кода /Луцкий Г.М., Блинова Т.А., Дрофа С.В., Порев В.Н.
12. Блинова Т.А. Представление отрицательных чисел в кодах золотой пропорции //Вестник КПИ. Сер. автоматика и электроприборостроение. – N24. – 1987. – С. 125-127.

УДК 681.3

САДЕГ РАСТГУ

МОДЕЛИ ВЫЧИСЛИТЕЛЬНЫХ ПРОЦЕССОВ В КЛАСТЕРНЫХ СИСТЕМАХ С МНОГОЯДЕРНОЮ АРХИТЕКТУРОЮ

В работе рассматриваются подходы к разработке математических моделей, описывающих поведение кластерной системы с многоядерною архитектурою. Проанализированы особенности организации вычислений в таких системах, связанных с двухуровневой параллельной обработкой. Представлена математическая модель кластерной системы, основанная на теории последовательных взаимодействующих процессов.

In paper a problem of creation a model of cluster systems with multicore architecture is considered. An analyze of computing in such systems related with two level of parallelisms is presented. Mathematical model of cluster system based on theory of sequential communicating processes is presented.

1. Введение

Развитие высокопроизводительных компьютерных систем в настоящее время связано с распределенными (кластерными) системам. Последняя редакция рейтинга суперкомпьютеров TOP500 (31-я редакция, июнь 2008 года) включает 61 кластерную систем в первых ста позициях. Количество процессоров в этих кластерных системах варьируется в диапазоне от 1992 до 122400, максимальная производительность – в диапазоне от 18,81 до 1026 Tflop/s, пиковая производительность – в диапазоне от 24,58 до 1375,78 Tflop/s. На сегодня наиболее мощной является кластерная система Roadrunner компании IBM, установленная в Лос-Аламосской национальной лаборатории (США).

Современная кластерная система (КС) включает набор вычислительных узлов, соединенных специальной быстродействующей системой связей, основанной на сетевых технологиях. Узел КС обеспечивает обработку и хранение данных, система связей – передачу данных между узлами КС.

Система связи узлов (ССУ) является важнейшей составляющей КС, так как эффективность КС в значительной мере определяется скоростью обмена данными между узлами системы. Основу всех реализаций систем связей составляют принципы, заимствованные их сетевых технологий, поэтому в КС широко используются технологии типа Ethernet (GigabitEthernet), которые на сегодня обеспечивают пропускную способность сети до 10Gbit/sec. Наряду с Ethernet технологией в современных КС используются специализированные системы связей типа Myrinet, Infiniband, cLan, SCI, Quadrics. Анализ кластерных систем, представленных в первых 100 позициях 31-го рейтинга TOP500 показал, что наиболее используемой коммуникационной технологией является InfiniBand (42 позиция). За ней следует технология GigEthernet (10). В списке представлены также технологии Myrinet (7) и QsNet (2).

Рост производительности КС традиционно связывался с увеличением количества узлов системы. Однако наращивание числа узлов ограничено из-за повышенного требования к ССУ при увеличении узлов и нагрузке на систему связей. Имеется второй путь повышения производительности КС, основанный на увеличении вычислительной мощности непосредственно самого узла. Такой подход обеспечивается использованием в узлах мощных процессоров, также реализацией в узлах КС принципов параллельной обработки за счет применения многопроцессорных систем. Использование в узлах многопроцессорных систем позволяет перейти в КС к двухуровневой параллельной обработке. Первый уровень обеспечивается параллельной обработкой на множестве узлов КС, второй – на множестве процессоров внутри узла.

Появление многоядерных процессоров открывает перспективы увеличения мощности узлов за счет более эффективной реализации второго уровня параллельной обработки в КС. Простая замена в узле обычного процессора на двухядерный процессор позволяет удвоить вычислительную мощность КС без изменения ее структуры. Появление на рынке четырехядерных процессоров и анонсированное в ближайшем будущем появления процессоров с восьмью и более ядрами, открывает возможности резкого увеличения производительности КС при построении систем на основе мно-

гоядерных процессоров. В этой связи следует отметить, что 31-я редакция TOP500 характеризуется широким использованием в компьютерных системах четырехядерных процессоров нового поколения (258 систем).

Целью настоящей работы является анализ особенностей организации вычислительных процессов в кластерных системах с многоядерною архитектурою (КСМА) и построение математических моделей, позволяющих формализовать эти особенности, которые позволят в полной мере реализовать все преимущества структурной организации КСМА.

2. Модели вычислительных процессов в КСМА

Рассмотрим КСМА как систему, поведение которой можно описать с помощью алгебры процессов, предложенной в теории взаимодействующих последовательных процессов (CSP) Ч.Хоара [1]. Выбор теории Ч.Хоара объясняется тем, что она позволяет описать любой объект с помощью процессов. Модель, основанная на понятии процесс, в свою очередь может быть эффективно реализована с помощью современных языков параллельного программирования, базирующихся на механизме потоков (легких процессов).

При разработке модели будем исходить из того, что КСМА является сложной системой и имеет два уровня. Первый уровень – система узлов, второй уровень – система ядер внутри узла. Тогда модель КСМА имеет вид иерархической структуры, состоящей из моделей двух уровней (КМСА1 и КСМА2). Модель первого уровня описывает поведение системы, связанные с узлами, а модель второго уровня – с поведением системы внутри узла.

2.1 Модель КСМА1. Для модели первого уровня КСМА1 используем модель клиент-сервер, которая включает объект – сервер и множество объектов – клиентов.

В теории Т.Хоара процесс определяет поведение объекта и может быть описан с помощью ограниченного набора событий, выбранного в качестве его алфавита.

Набор событий, связанных с объектом сервер (алфавит сервера) C можно представить в виде

$$\alpha C=\{\ вводC, выводC,\ принятьСК_i,\ передатьСК_i\ \ счетC\ \}$$

где события: *вводС* – ввод данных на сервере, *выводС* – вывод результат на сервере, *принятьСК*$_i$ – прием сервером данных от клиента, *передатьСК*$_i$ – передача данных от сервера клиенту K_i , *счетС* – счет на сервере.

Набор событий, связанных с i-м объектом клиентом (K_i) можно представить в виде

$$\alpha K_i=\{принятьK_iC,\ передатьK_iC,\ счетK_i,\ принятьK_iK_j,\ передатьK_iK_j\ \}$$

где события: *принятьК*$_i$*С* – прием данных от сервера, *передатьК*$_i$*С* – передача данных серверу, *счетК*$_i$ – счет на клиенте, *принятьК*$_i$*К*$_j$ – прием данных от клиента K_j , *передатьК*$_i$*К*$_j$ – передача данных клиенту K_j.

Поведение объекта сервера используя алфавит αC, можно описать как процесс

$$C=(вводC\rightarrow передатьСК_i\rightarrow счетC\rightarrow принятьСК_i\ \rightarrow выводC\ )$$

Для множества объектов клиентов K_i их поведение (клиентский процесс) описывается следующим образом

$$K_i =(\ принятьK_iC\ \rightarrow\ передатьK_iK_j\ \rightarrow\ счетK_i\ \rightarrow принятьK_iK_j\ \rightarrow\ передатьK_iC\ )$$

Поведение КСМА1 можно описать как параллельную комбинацию поведения процессов C и K_i :

$$КЛИЕНТЫ=(K1||K2||\ .\ .\ .||KP),\ \ СЕРВЕР=(C)$$

$$КСМА1\ =\ (СЕВЕР\ ||\ КЛИЕНТЫ\ )$$

Взаимодействие процессов в модели КСМА1

Взаимодействие узлов в КСМА1 осуществляется с помощью посылки сообщений. Будем рассматривать взаимодействие как событие. Взаимодействие состоит в передаче сообщения и является событием, требующим участия двух процессов.

Для описания взаимодействия процессов введем понятие объекта коммуникации (*ОК).* Алфавит *ОК* включает следующие события:

$$\alpha ОК\ =\ \{\ записьОК,\ \ чтениеОК\ \}$$

Процесс, описывающий поведение объекта *ОК*:

$$ОК\ =((\ записьОК\ \rightarrow\ чтениеОК\ |\ чтениеОК\ \rightarrow\ записьОК\ )\ \rightarrow\ ОК\ )$$

Составляющей *ОК* является спецификация *С*, который определяет тип и объем передаваемых данных (*ОК.С*) .

В модели КСМА1 взаимодействие объектов связано с событиями *принять* и *передать*. При этом взаимодействие должно быть связано с *ОК*, куда один объект передаст данные, а другой объект произведет считывание данных. Дополним события принять и передать параметром, определяющим используемый

объект коммуникации: *передатьСК*(*ОК*), *принятьСК*(*ОК*). Процесс *ОК* можно рассматривать как сопроцесс [1].

2.2 Модель КСМА2. Узел будем рассматривать как систему, включающую набор объектов, связанных с действиями, выполняемыми в многоядерном процессоре. При организации процессов внутри узла используем модель взаимодействия хост-рабочий [2]. Хост объект отвечает за связь с остальными узлами КСМА, и, в частности, – с серверным узлом КСМА. Хост объект (*X*) принимает исходные данные от сервера КСМА, распределяет их в узле по объектам – рабочим, собирает результаты счета в узле и отсылает результат в серверный узел КСМА. Объект рабочий (*P*) принимает данные от хост объекта, выполняет счет и возвращает результат хосту.

Набор событий, связанных с объектом хост *i*-го узла (*Xi*), можно представить в виде

$$\alpha X_i = \{ \textit{принятьX}_i,\ \textit{передать}\ X_iP_{ij},\ \textit{счетX}_i,\ \textit{принятьX}_iP_{ij},\ \textit{передать}\ X_i \}$$

где события: $\textit{принятьX}_i$ – прием данных от другого узла, $\textit{передатьX}_iP_{ij}$, – передать данные в узле объекта рабочий P_iP_{ij}, $\textit{счетX}_i$, – счет, $\textit{принятьX}_iP_{ij}$ – принять данные в узле от объекта рабочий P_{ij}, $\textit{передатьX}_i$ – передать данные в другой узел.

Набор событий, связанных с объектом *j*-ий рабочий *i*-го узла (P_{ij}) можно представить в виде

$$\alpha P_{ij} = \{\textit{принятьP}_{ij}X_i,\ \textit{передатьP}_{ij}X_i, \textit{счетP}_{ij},\ \textit{принятьP}_iP_{iк},\ \textit{передатьP}_{ij}P_{iк}\}$$

где события: $\textit{принятьP}_{ij}X_i$ – прием данных от хост объекта, $\textit{передатьP}_{ij}X_i$ – передача данных хост объекту, $\textit{счетP}_{ij}$ – счет в хост объекте, $\textit{принятьP}_iP_{iк}$ – прием данных от объекта рабочего $P_{iк}$, $\textit{передатьP}_{ij}P_{iк}$ – передача данных объекту рабочему Рік..

Описание поведения объектов хост и рабочий будет зависеть от системы связей ядер в многоядерном процессоре, которая будет определять форму взаимодействия объектов хост и рабочий. Увеличение количества ядер ставит проблему организации связей между ядрами. Используемая в 2-4 ядрах шинная архитектура станет тормозом для эффективной связи ядер и неизбежным будет переход на распределенную архитектуру. Так, например, связывание ядер с помощью топологии типа решетка позволит отказаться централизованных ресурсов, а также решить проблему надежности при выходе из строя ядер или связей.

Если связь ядер в многоядерном процессоре реализована через распределенную архитектуру с помощью, например, топологии звезда, где все ядра связаны через хост ядро, то поведение объектов внутри узла можно описать следующим образом.

Поведение хост объекта X_i (процесс хоста) используя алфавит αX_i, можно описать как

$$X_i = \{ \textit{принятьX}_i \rightarrow \textit{передатьX}_iP_{ij} \rightarrow \textit{счетX}_i \rightarrow \textit{принятьX}_iP_{ij} \rightarrow \textit{передатьX}_i \}$$

Поведение объекта рабочий P_{ij}:

$$P_{ij} = (\textit{принятьP}_{ij}X_i \rightarrow \textit{счетP}_{ij}, \rightarrow \textit{передатьP}_{ij}X_i)$$

Модель второго уровня для*i*-го клиентского узла можно описать как параллельную комбинацию поведения процессов X_i и P_{ij}:

$$\textit{ХОСТ}_i = (X_i),\ \textit{РАБОЧИЕ}_i = (P_i)$$

$$\textit{УЗЕЛ}_i = (X_i \parallel \textit{РАБОЧИЕ}_i)$$

Взаимодсействие процессов в модели КСМА2.

Организация взаимодействия процессов внутри узла зависит от выбранной системы связей ядер. При прямой связи ядер (распределенная архитектура) используется модель, основанная на посылке сообщений, для связи через память (шинная архитектура) – модель, основанная на общих переменных.

При использовании посылки сообщений справедлива схема взаимодействия через объекты коммуникации (*ОК*), рассмотренная для модели первого уровня КСМА1.

Рассмотрим подробнее организацию взаимодействия процессов при использовании модели, основанной на общих переменных. Данная модель связана с решением двух задач: задачи взаимного исключения и задачи синхронизации [4]. Введем понятие объекта синхронизации (*ОС*), который будет использоваться процессами при взаимодействии через общую память. Выделим три вида объектов синхронизации. Первый вид *ОС*(*A*) связан с неделимыми (атомик) операциями на переменными. Второй вид *ОС*(*S*) реализует концепцию низкоуровневых средств синхронизации, используемых в примитивах типа семафор, мютекс, событие. Третий вид *ОС*(*M*) реализует концепцию мониторов.

Объект синхронизации типа А

$$\alpha A = \{ \textit{чтениеA},\ \textit{записьA},\ \textit{счетA} \}$$

где *чтениеA* – чтение неделимой переменной, *записьA* – запись в неделимую переменную,

счетA – счет с использованием неделимой переменной.

Объект синхронизации типа S

αS ={ *установитьS, проверитьS, сигналS* }

где *установитьS* – установка начального несигнального состояния *S*, *проверитьS* – проверка состояния *S*; блокирование процесса при занятом (несигнальном) состоянии – *S*, *сигналS* – установка *S* в сигнальное состояние (при создании *ОС* принимает сигнальное состояние).

Объект синхронизации типа M

Набор событий, связанных с объектом-монитором *M* можно представить в виде

αM ={ *записьM, чтениеM, счетM, ждатьM, сигналM* }

где события: *записьM* – запись данных в монитор, *чтениеM* – чтение (копирование) данных из монитора, *счетM* – счет в мониторе, *ждатьM* – ожидание сигнала, *сигналM* – сигнал.

Использование объектов синхронизации в задачах взаимного исключения и синхронизации.

Объект A. Обращение процессов к общему ресурсу связано со следующими событиями в объектах *X* и *P*:

X: (*записьA*), *P*: (*чтениеA*)

Объект S. Обращение процессов к общему ресурсу связано со следующими событиями в объектах *X* и *P*:

X: (*проверитьS* → *счетX* → *сигналS*)

P: (*проверитьS* → *счетP* → *сигналS*)

Синхронизация связано со следующими событиями (хост посылает сигнал рабочему):

X: (*сигналS*), *P*: (*проверитьS*)

Взаимное исключение. **Объект M.** Обращение процессов к общему ресурсу связано со следующими событиями в описании объектов *X* и *P*:

X:(*записьM,чтениеM,счетM, вводM, выводM*)

P: (*чтениеM, записьM, счетM*)

Синхронизация связано со следующими событиями в описании объектов *X* и *P*:

X:(*ждатьM,сигналM*),*P*:(*сигналM, ждатьM*)

3. Применение моделей

Предложенные модели позволяют применить математический аппарат теории алгебры процессов Ч.Хоара для анализа вычислительных процессов в КСМА.

Рассмотрим анализ взаимодействия процессов на возможность возникновения тупиковой ситуации при синхронизации двух процессов через объект синхронизации типа *ОС*. Пусть *P* и *Q* – процессы, *D* – объект синхронизации типа *ОС*(*S*). Процессы синхронизируются по событию (ввод данных) в процессе *P* (процесс *Q* ждет сигнала от процесса *P*), после чего процессы начинают счет.

Алфавиты процессов будут иметь следующий вид:

αP = { *ввод, счет, установитьD, сигналD*}

αQ = { *счет, проверитьD, установитьD* }

αD = { *проверитьD, установитьD, сигналD* }

Поведение процессов:

P = (*установитьD*→*ввод*→*сигналD* → *счет*)

Q = (*проверитьD* → *счет*)

Процесс *D* следует рассматривать как подчиненный процесс по отношению к процессам *P* и *Q*. При этом *αD* ⊆ (*αP* U *αQ*). В комбинации (*P* || *Q* || *D*) каждое действие *D* может произойти, если это позволяют процессы *P* и *Q*.

Тупиковая ситуация при синхронизации процессов *P* и *Q* может возникнуть в двух случаях: а)объект *D* не установлен изначально в несигнальное состояние; б)процесс *P* не посылает сигнал о событии, которое ожидает процесс *Q*. Эти действия в процессах связаны с событиями *проверитьD, сигналD, установитьD*. Для борьбы с тупиком необходимо выполнить анализ алфавитов процессов и поведения процессов. Необходимое условие отсутствия тупика – наличие этих событий в алфавитах процессов:

{*проверитьD, сигналD, установитьD* } ⊆ (*αP* U *αQ*)

Достаточные условия – наличие события *сигналD* в алфавите процесса *P* и события *проверитьD* в алфавите процесса *Q*:

{ *сигналD*} ⊆ (*αP*), { *проверитьD*} ⊆ (*αQ*)

Для процессов – необходимо присутствие события *сигналD*, события *проверитьD* и *установитьD* при описании поведения процессов *P*, *Q* и *D* соответственно:

(*проверитьD*→*Q*) || (*сигналD* → *P*) || (*установитьD* → *D*)

Выводы

Предложены математические модели, основанные на теории взаимодействующих процессов Ч.Хоара, позволяющие описать орга-

низацию вычислений в кластерных системах с многоядерною архитектурою. Модели отображают двухуровневую организацию параллельных процессов в КСМА, а также систему взаимодействия процессов на основе посылки сообщений или общих переменных. Модель может быть использована для анализа параллельных процессов в КСМА, оптимизации взаимодействия процессов, связанной с передачей данных, а также выявления тупиковых ситуаций, возникающих при взаимодействии процессов в КСМА.

Список литературы

1. Хоар Ч. Взаимодействующие последовательные процессы: Пер с англ. – М.: Мир, 1989.
2. El - Rewini H, Lewis T., Distributed and Parallel Computing. – Manning Pub. Co., NY, 1998.
3. Burns A., Welling A. Real-Time Systems and Programming Languages, Third Edition, Addison-Wesley, NY, 2001.
4. Жуков І., Корочкін О. Паралельні та розподілені обчислення. – Корнійчук, К., 2006.

ПАВЛОВ А.А.,
МИСЮРА Е.Б.,
КУТ В.И.,
ЩЕРБАТЕНКО О.В.,
МИХАЙЛОВ В.В.

МАТЕМАТИЧЕСКИЕ МОДЕЛИ ИЕРАРХИЧЕСКОГО ПЛАНИРОВАНИЯ И ПРИНЯТИЯ РЕШЕНИЙ

Показано, что стандартная постановка задачи нахождения оптимального расписания для реальных организационно-технических объектов не полностью адекватна реальной ситуации. Для адекватного решения задачи предлагается метод анализа иерархий.

It is shown, that standard definition of optimal scheduling problem for real technical objects is not quite appropriate to the real situation. For appropriate decision of the problem the Analytical Hierarchy Process (AHP) is proposed.

В [1-9] описаны комплексы последовательных взаимосвязанных математических моделей, что совместимые с иерархией решений, которые должны быть приняты при иерархическом планировании и управлении сложными организационно-производственными системами, а так же создание на основе разработанной авторами конструктивной теории решения комбинаторных задач, эффективных методов решения одноэтапных задач теории расписания, являющихся основой алгоритмического обеспечения иерархической системы планирования и управления.

В результате разработки создана методология построения иерархических моделей планирования дискретного производства, которая учитывает сетевое представление технологий изготовления изделий, ограниченные ресурсы и направленные на максимизацию дохода предприятия. Разработаны комплексы последовательных взаимосвязанных математических моделей иерархического планирования и управления рабочим цехом, планирование производства «на заказ», планирование производств мелкосерийного типа; иерархического планирования производств по изготовлению партий, планирование и управление проектами.

В результате получена система новых высокоэффективных взаимосвязанных алгоритмов, которые впервые разрешили решать задачу планирования и управления дискретным производством общего вида с ограниченными ресурсами в комплексе, по критериям:

а) минимизация общего взвешенного момента окончания выполнения заданий при отношении порядка, заданного ориентированным ациклическим графом;

б) минимизация общего взвешенного момента окончания выполнения заданий при отношении порядка, заданного ориентированным ациклическим графом, при условии, что известны моменты поступления всех заданий на выполнение;

в) минимизация общего опережения и задержки выполнения заданий относительно директивных сроков (планирование точно в срок);

г) минимизация общего штрафа производства за опережение или задержку выполнения заданий относительно директивных сроков;

д) минимизация общей задержки выполнения заданий относительно директивных сроков;

е) минимизация общей взвешенной задержки выполнения заданий относительно директивных сроков;

ж) минимизация общего времени выполнения всех заданий проекта и их комбинаций при различных ограничениях.

Однако при оценке качества полученных расписаний и прогнозировании возможных результатов ЛПР (лицо принимающее решения), обычно сталкивается с сложной системой взаимозависимых критериев, которые нужно проанализировать [10-21]. Приведем некоторые критерии качества оценки расписаний:

1. Величина максимальной задержки относительно директивного срока; [10,11]
2. Величина максимального опережения относительно директивного срока [12,16];

3. Величина максимального отклонения относительно директивного срока [12];
4. Величина числа работ, которые задерживаются [10];
5. Величина суммарной задержки относительно директивного срока [12,14];
6. Величина суммарного опережения относительно директивного срока;
7. Величина суммарного опережения и задержки относительно директивного срока [12];
8. Величина максимальной взвешенной задержки относительно директивного срока;
9. Величина максимального взвешенного опережения относительно директивного срока;
10. Величина максимального взвешенного отклонения относительно директивного срока;
11. Величина взвешенного числа работ, которые задерживаются [11,12];
12. Величина суммарной взвешенной задержки относительно директивного срока [11,12];
13. Величина суммарного взвешенного опережения относительно директивного срока [16];
14. Величина суммарного взвешенного опережения и задержки относительно директивного срока [11,12];
15. Величина значения общего директивного срока [12];
16. Величина соответствия требованиям выполнения всех заданий до их крайних сроков [12];
17. Величина общего времени завершения работ и максимальной стоимости [20];
18. Величина суммарного взвешенного времени окончания выполнения работ [11];
19. Величина взвешенной задержки и сверхнормативных затрат одного ресурса [18];
20. Величина взвешенной задержки, сверхнормативных затрат и затрат на задержку поставок;
21. Числа перенастроек оборудования [19];
22. Величина максимальной длительности прохождения заданий в системе [20];
23. Величина суммарного взвешенного позднего времени выполнения $V_i(\sigma)$ (позднее время – количество времени, когда работа выполняется после директивного срока: $\omega_i V_i = \omega_i \min\{T_i, p_i\}$) [22].
24. Величина средней продолжительности прохождения заданий;
25. Величина среднего времени окончательного освобождения оборудования;
26. Величина максимального времени окончательного освобождения оборудования;
27. Величина продолжительности ожидания заданий в системе;
28. Величина среднего времени нахождения работ в системе;
29. Соответствия свойствам не задерживающих расписаний;
30. Соответствие свойствам компактных расписаний;
31. Величина максимальной стоимости выполнения множества заданий;
32. Величина суммарной стоимости (затрат на производство);
33. Величина дохода;
34. Оценка загрузки оборудования;
35. Величина простаивания оборудования;
36. Величина сокращения производственного цикла;
37. Эффективное использование оборудования (экономия энергоресурсов);
38. Величина выполнения заданий точно в срок [12];
39. Отношение длительности нахождения задания в системе к продолжительности его критического пути;
40. Отношение суммарного времени выполнения всех заданий в системе к критичному пути выполнения всех заданий.

Как правило на практике реальное расписание должно оцениваться не по одному критерию, а по всему множеству критериев и таким образом, окончательное определение расписания на практике является задачей многокритериального выбора: каждому из семи расписаний, реализующему задачу оптимального планирования и управления дискретным производством по одному из приведенных выше семи критериев оптимизации, на самом деле соответствует значение неизвестной функции качества расписания аргументами которой являются десятки (нами приведены 47) критериев. В такой постановке для выбора «наилучшего» расписания естественно применить метод иерархий Саати [23]. В простейшем виде его реализация приводит к следующей процедуре. Эксперту (экспертам) предлагается сравнить попарно получение расписания. Результаты сравнения заносятся в так называемую эмпирическую матрицу парных сравнений.

По элементам матрицы парных сравнений находятся веса расписаний, выбирается то

расписание, которому соответствует наибольший вес. Существует достаточно большое количество методов нахождения весов по эмпирической матрице парных сравнений [24]. С точки зрения авторов для нахождения точечных оценок весов наиболее перспективным является подход изложенный в [25–27], суть которого заключается в следующем: формулируются и обосновываются модели оптимизации. Оптимальному решению каждой из них соответствует вариант оценок весов. Далее предлагается многокритериальная процедура выбора наилучших оценок весов [27].

Выводы

В статье показано, что стандартная постановка задачи планирования и управления дискретным производством – нахождения расписания оптимального по некоторому критерию при всей своей сложности реализации в целом не полностью соответствует решению реальной задачи планирования и управления. Подход к решению задачи планирования и управления дискретным производством органически объединить с моделями принятия решений на основе модифицированного метода анализа иерархий Саати.

Список литературы

1. Створення математичних моделей та методів автоматизованого ієрархічного планування в складних організаційно – виробничих системах // Звіт про НДР №2977 – ф (заключний) – УДК 519.854, КВНТД 1.1 01.01.10.№держреєстрації 0106U002391. – Київ: КПІ. – 2007. – 253с.
2. Требования к созданию систем производственного планирования и управления сложными объектами, имеющими сетевое представление технологических процессов и ограниченные ресурсы / А.А. Павлов, Е.Б. Мисюра, О.В. Мельников, О.В. Щербатенко, В.В. Михайлов // Вісник НТУУ «КПІ». Інформатика управління та обчислювальна техніка. К:. «ВЕК+», 2007. – №46. – С. 3-12
3. Павлов О.А., Місюра О.Б., Халус О.А. Підвищення ефективності алгоритмічного забезпечення ієрархічної автоматизованої системи планування та управління проектами за рахунок точного розв’язання задачі мінімізації сумарного запізнення виконання завдань / Автоматизированные системы управления и приборы автоматики: Всеукраинский межведомств. науч.-техн. сбор.– Харьков: ХНУР, 2007.– Вып.138.– С.15-17
4. Общая модель и методы иерархического планирования функционирования сложных организационно-производственных систем с ограниченными ресурсами / А.А. Павлов, Е.Б. Мисюра, О.В.Мельников, Сераржи Али Рухани // Системні дослідження та інформаційні технології.– 2005.– №4.– С.7-24
5. Павлов А.А., Теленик С.Ф. Информационные технологии и алгоритмизация в управлении.– К.: Техника.– 2002.– 344 с., ISB№966-575-045-3.
6. Павлов О.А., Аксьонова Л.О. "Мінімізація сумарного зваженого моменту закінчення робіт" як перший рівень моделі дрібносерійного виробництва та способи її розв'язання // Системні дослідження та інформаційні технології.– 2002.– №1.– С.119-130
7. Павлов О.А., Місюра О.Б., Мельников О.В. Загальна схема розв’язання задач в багаторівневій системі планування дрібносерійного виробництва в умовах ринку / Вісник НТУУ “КПІ”. Інформатика, управління та обчислювальна техніка. К.: “ВЕК+”, 2000.– №33.– С.27-33
8. Павлов О.А., Місюра О.Б., Мельников О.В., Вакуленко О.С. Багаторівнева система планування дрібносерійного виробництва в умовах ринку.– «Автоматика–98».– п’ята українська конференція з автоматичного управління, ч.II, К.: НТУУ «КПІ», 1998.– ст. 182-186
9. Организационная модель планирования мелкосерийного производства в условиях рынка. / А.А.Павлов, Е.Б.Мисюра, О.В.Мельников, Алауи Исмаили Юнес // Проблемы информатизации и управления: Сб.науч.тр.– К.: КМУГА, 1997.– С.3-5
10. Chen C-L; Bulfin R.L. Scheduling a single machine to minimize two criteria: maximum tardiness and number of tardy jobs / IIE Transactions – Sept. 1994
11. Handbook of Combinatorial Optimization, D.–Z. Du and P.M. Pardalos (Eds.), Kluwer Academic Publishers, 1998
12. Finke G., Gordon V., Proth J.-M. Scheduling with due dates (Annotated Bibliography of complexity and algorithms). Les cahiers du laboratoire Leibniz #42, 2002
13. Joseph Y-T. Handbook of Scheduling: Algorithms, Models, and Performance Analysis. *http://books. google. com. ua/books?id=ymJTEjPg6CcC*
14. Koulamas C., [1994], The total tardiness problem: review and extensions, Operations research, 42, 6, 1025-1041
15. N.G. Hall (1986) Scheduling problems with generalized due dates, IIE Transactions 18, 220-222

16. Pathumnakula S., Egbelu P.J. Algorithm for minimizing weighted earliness penalty in single-machine problem. European Journal of Operational Research. Vol. 161 (3), 2005, P.780-796
17. Azizoglu M., Kondakci S., Köksalan M. Single machine scheduling with maximum earliness and number tardy. Computers & Industrial Engineering 45 (2), 2003, P.257-268
18. Yang B., Geunes J., O'Brien W.J. A heuristic approach for minimizing weighted tardiness and overtime costs in single resource scheduling. Computers & Operations Research 31 (8),2004,P.1273-1301
19. Tovey C.A. Non-approximability of precedence-constrained sequencing to minimize setups. Discrete Applied Mathematics 134 (1-3), 2004, P.351-360
20. Jain A.S., Meeran S. Deterministic job-shop scheduling: Past, present and future. European Journal of Operational Research 113 (1999), P.390-434
21. Jiao J., Zhanga Y., Wang Y. A heuristic genetic algorithm for product portfolio planning. Computers & Operations Research 34 (6), 2007, P. 1777-1799
22. Hariri A., Potts C.N., Van Wassenhove L.N. Single Machine Scheduling to Minimize Total Weighted Late Work / ORSA Journal of Computing 7 (2), 1995, P.232-242
23. 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.
24. Тоценко В.Г. Методы и системы поддержки принятия решений. Алгоритмический аспект. – Киев: Наукова думка. – 2002. – 381 с.
25. А.А. Павлов, Е.И. Лищук, В.И. Кут „Математические модели оптимизации для обоснования и нахождения весов в методе парных сравнений” Системні дослідження та інформаційні технології. №2, 2007р.
26. А.А. Павлов, В.И. Кут „Математические модели оптимизации для обоснования и нахождения весов объектов по неоднородным матрицам парных сравнений” Системні дослідження та інформаційні технології. №3, 2007р.
27. А.А. Павлов, Е.И. Лищук, В.И. Кут „Многокритериальный выбор в задаче обработки данных матрицы парных сравнений”. Вісник НТУУ „КПІ”. Інформатика, управління та обчислювальна техніка. №46, 2008р.

УДК 681.3.06

КУЗНЕЦОВ А.В

ПРЕДСТАВЛЕНИЕ ЗНАНИЙ В КОНЦЕПЦИИ ОБЪЕКТНО-ОРИЕНТИРОВАННОГО ЛОГИЧЕСКОГО ПРОГРАММИРОВАНИЯ

Предложена модель представления знаний на основе классического объектно-ориентированного формализма, охваченного семантической сетью, которая названа ST-сетью. Рассмотрены некоторые особенности реализации модели в языке объектно-ориентированного логического программирования Logtalk.

The model of knowledge representation is offered on the basis of the classic object-oriented formalism overcame a semantic network that is adopted a ST-network. Some of model realization features are considered in Logtalk object-oriented logical programming language.

Неструктурированность и процедурная избыточность программ в Прологе

Интерпретация семантических сетей (СС) [1] на Прологе сопровождается рядом трудностей и недостатков, связанных с тем, что решение задачи проблемной области (ПО) реализуется в виде *неструктурированной* совокупности отношений, представляющих множества фактов и правил вывода. Это приводит к громоздкости и запутанности программ, особенно в случаях, когда требуется спецификация понятий СС, которые включают динамически изменяющиеся свойства. Кроме того, использование исключительно классической логики для записи исходных формул СС, осложняет программирование выбора *возможных* и *требуемых* правил вывода, что ведет к порождению лишних неконструктивных решений и существенно снижает эффективность программ. С целью преодоления этих трудностей и улучшения возможностей инструментальной среды исследуются и разрабатываются средства объектно-ориентированного логического программирования (ООЛП), одним из представителей которых является система Logtalk.

Рассмотрим особенности и преимущества Logtalk с точки зрения реализации декларативно-процедурной семантики, т.е. формулировки семантики понятий СС при помощи определений и программ на основе концепции ООЛП.

Концептуальная основа Logtalk

Система Logtalk представляет собой расширение стандартного Пролога и взаимодействует со многими современными компиляторами. Основные идеи объектного подхода к программированию, заложенные в Logtalk можно кратко свести к следующим положениям [2]:

1. Система Logtalk объединяет основные преимущества двух парадигм программирования. С одной стороны, объектная ориентация позволяет работать с одним и тем же набором логически целостных элементов в последовательных фазах разработки приложения, предоставляя возможность организации и инкапсуляции знаний о каждом объекте в пределах заданной ПО. С другой стороны, логическое программирование предусматривает представление знаний о каждом объекте декларативным способом. Соединенные вместе, эти два преимущества существенно приближают разработанную программную модель к ПО, включая упрощение написания и сопровождения программ, а также повышение производительности разработчиков.
2. Обеспечена интеграция событийно-управляемого и объектно-ориентированного программирования.
3. Практически все доступные сегодня объектно-ориентированные языки программирования базируются, либо на классах, либо на прототипах, с преимущественным распространением языков базируемых на классах. В Logtalk предусмотрена поддержка обеих типовых иерархий. Это значит, что в одном и том же приложении допускается как иерархия прототипов, так и иерархия классов. Прототипы решают проблему систем базируемых на классах в случаях, когда необходимо определить класс с единственным экземпляром, который уместен при повторном использовании части кода. Классы решают проблему дуализма, характерную для систем, базируемых на прототипах, где невозможно инкапсулировать некоторый код для повторного использования другими объектами, а не только объектом, выделенным в самостоятельный элемент. Автономными считаются объекты, которые не принадлежат какой либо иерархии и дают удобные решения для инкапсуляции кода при по-

вторном использовании несколькими независимыми объектами.

4. В языках подобных Smalltalk-80, Objective-C и Java описывается единственная иерархия с корневым классом, который обычно называется Object. Это гарантирует, что все объекты разделяют общее поведение, но имеет тенденцию к созданию длинных иерархических описаний, где достаточно сложно описывать объекты с исключительным поведением. В Logtalk предусмотрены составные, независимые иерархии объектов. Некоторая часть таких иерархий может быть основана на прототипах, тогда как другие основаны на классах. Кроме того, автономные объекты обеспечивают простой способ инкапсуляции полезных предикатов, которые не требуют включения в иерархии объектов.

5. Logtalk предусматривает гибкую поддержку в разделении интерфейса и реализации так, что протокольные инструкции могут быть помещены в объект, категорию или протокол (первичные сущности Logtalk) или могут быть разбросаны (рассредоточены) по объектам, категориям и протоколам.

6. Подобно C++, Lagtalk поддерживает директивы наследования private, protected и public, предоставляя возможность ограничивать диапазон унаследованных, импортированных или включенных предикатов.

7. Еще одной характерной особенностью Logtalk является то, что имена объектов могут быть составными термами (вместо атомов), обеспечивая способ параметризации объектных предикатов. Но, доступ к значениям параметров осуществляется посредством встроенного метода вместо создания глобального диапазона параметров всего объекта.

В остальном система Logtalk спроектирована из расчета совместимости с большинством компиляторов Пролога.

Семантические сети в объектно-ориентированном проектировании

Прежде всего, остановимся на некоторых аспектах, связанных с введением соглашений, относящихся к использованию СС в объектно-ориентированном проектировании.

Как средство представления знаний СС исторически развивались, постепенно концентрируя в своей концепции такие понятия, как класс, объект, обобщение, иерархия, наследование свойств и т.п., которые характерны для существующего на сегодняшний день подхода к построению объектных моделей. Такая тенденция дает основание для рассмотрения выразительных возможностей СС с точки зрения их применения в классическом объектно-ориентированном проектировании, исходя из современных взглядов на процесс разработки. Отметим также, что семантические сети ориентированы на решение задач, относящихся к машинному анализу смысла фраз естественного языка, тогда как в объектно-ориентированном проектировании внимание разработчика акцентируется на структурно-функциональном аспекте ПО. Это значит, что объединение в ходе разработки двух формализмов решает задачу сближения языка проблемной области с объектной моделью. Другими словами, будем говорить о дополнении канонической формы сложной декомпозиции [3] семантической сетью, как это показано на рис. 1.

Представление иерархий и связей между объектной моделью и СС, требует рассмотрения некоторых терминологических и эксплуатационных отличий принятых в Logtalk по сравнению с другими языками объектно-ориентированного программирования (ООП).

Во-первых: Logtalk оперирует только с тремя видами сущностей: *объекты*, *протоколы*, и *категории*. Термины *объект*, *прототип*, *родитель*, *класс*, *подкласс*, *суперкласс*, *метакласс*, *экземпляр* и *предок* всегда обозначают объект. Различные имена используются скорее для того, чтобы сделать акцент на роли, которую играет объект в специфическом контексте. Роль объекта зависит, в свою очередь, от его отношения с другими объектами. Термин, отличный от термина *объект*, используется только в случаях, когда необходимо обратить внимание на явное отношение с другими объектами.

Во-вторых: также как и в других языках ООП, в Logtalk иерархии объектов предоставляют возможность общего использования интерфейса и реализации посредством наследования. В контексте ООЛП, наследование может быть интерпретировано как форма теории расширения: объект фактически содержит, в добавок к его собственным предикатам, все предикаты унаследованные от других объектов. Протоколы инкапсулируют объявления предиката, отделяя интерфейс от реализации, а категории инкапсулируют и объявления и определения предикатов, которые могут быть импортированы любым объектом. Таким образом, подобно тому, как протоколы поддерживают многократное использование интерфейса, категории поддерживают много-

кратное использование реализации, обеспечивая альтернативу решения проблемы множественного наследования, характерной для других языков программирования. Тем не менее Logtalk поддерживает и множественное наследование для тех случаев, где его композиционные механизмы не могут обеспечить подходящее решение.

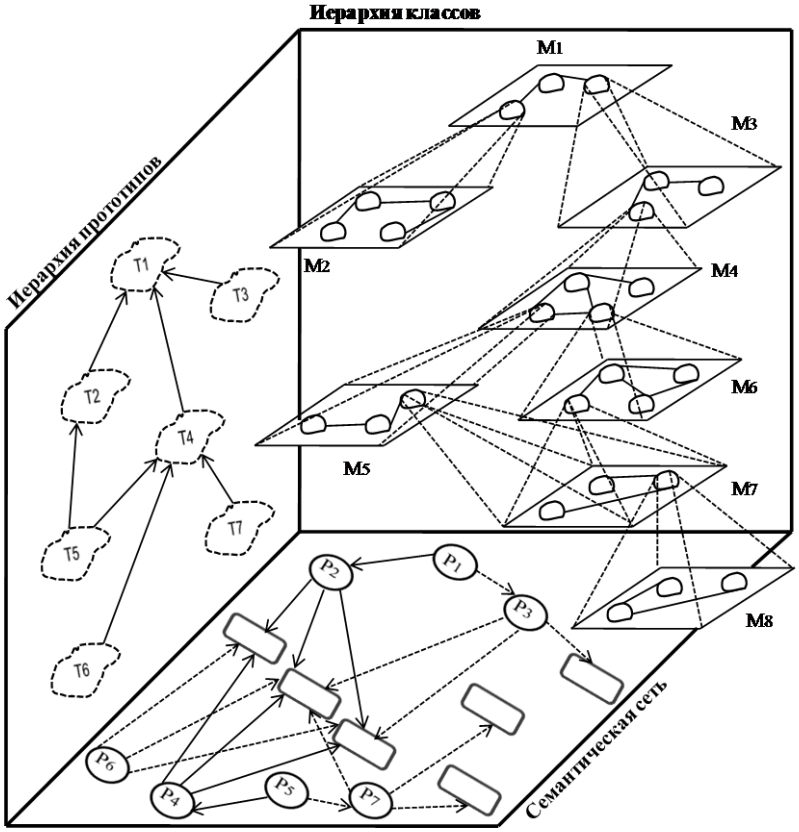


Рис. 1 Каноническая форма сложной декомпозиции, охваченной семантической сетью

В-третьих: для представления иерархий прототипов и классов в Logtalk используется три вида отношений. Иерархии прототипов конструируют определение отношений *расширения* (extension), тогда как иерархии классов, определяют отношения *экземпляра* (instantiation) и *специализации* (specialization) между объектами. На рис. 1 иллюстрируется общий случай построения комбинированной модели, основанной, как на иерархи прототипов, так и на иерархии классов. Такая форма сложной декомпозиции соответствует модели абстрактной рефлективной системы с динамически порождаемыми объектами, которая содержит метаклассы (классы интерпретируются как объекты более общих классов) и суперклассы (классы с которыми другие классы прямо или косвенно связаны отношением специализации). Так, метаклассы M_5 и M_6, представленные на рисунке, одновременно являются суперклассами метакласса M_7.

Механизм связи объектной модели с семантической сетью в Logtalk

Связь объектной модели с семантической сетью наиболее выразительно иллюстрируется возможностью Logtalk задавать имена объектов в виде составных термов, содержащих свободные переменные. Эти переменные играют роль объектных параметров, а объектные предикаты кодируются так, чтобы зависеть от их значений. Иначе говоря, параметрический объект может рассматриваться как родовой объект, из которого порождаются специфические экземпляры путем конкретизации параметров. Схему передачи параметров объектным предикатам рассмотрим на следующем примере:

```
:- object(foo1_obj).
  :- public(pred_pub/1).
  pred_pub1(something) .
...
:- end_object.
:- object(foo2_obj(_par_one, _par_two),
   specializes(protected::foo1_obj)).
```

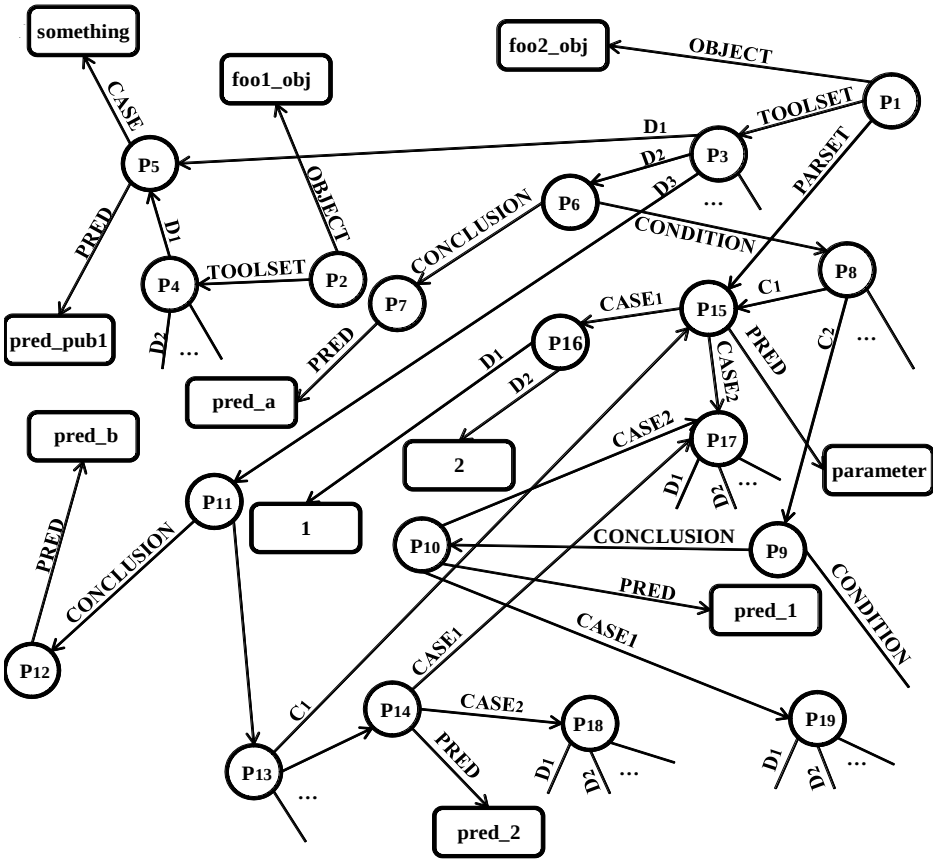


Рис. 2 Параметрический объект foo2_obj в семантической сети

CASE1 указывает на дизъюнктную вершину P16, которая объединяет возможные значения номеров позиций параметров, а связь CASE2 указывает на недоопределенную дизъюнктную вершину P17, которая объединяет возможные значения самих параметров. Подчеркнем еще раз, что в семантических падежах параметрической вершины могут быть указаны лишь возможные номера позиций параметров и их значений. Требуемые значения определяются только декларативно-процедурной семантикой объектных предикатов.

Оставшиеся недоопределенными две дизъюнктные вершины P18 и P19, объединяют возможные значения переменных Var_a и Var_b предикатов pred_1 и pred_2 соответственно.

Анализируя структуру сети на рис. 2 можно сказать, что она отображает семантику объектных моделей в ООЛП, так как охватывает объекты и связи с их инструментальными предикатами. Такие сети далее будем называть *ST-сетями*. В то же время на ST-сети никак не отражены иерархические межуровневые связи, которые формируют объектную модель ПО в целом. Требуемые связи создаются на этапе сложной декомпозиции ПО, как это показано на рис. 1, и реализуются программой Logtalk на основе декларативно-процедурной семантики. Например, в приведенном выше фрагменте программы объекты foo1_obj и foo2_obj компилируются как классы, так как они связаны отношением специализации. Тогда, объявленный в суперклассе foo1_obj public-предикат pred_pub1 становится protected-предикатом в подклассе foo2_obj, где наследование уточняется директивой protected.

В заключение отметим, что помимо возможности управлять вызовом правил вывода на основе инкапсуляции и наследования, определяемого директивами public, protected и private, в Logtalk предусмотрен гибкий механизм переопределения объектных предикатов с использованием оператора ^^/1 для организации супервызова [2]. С помощью супервызова можно управлять вызовом унаследованных предикатов, организуя *специализированное (specialization)*, *объединенное (union)* и *выборочное (selective)* наследования.

Специализированное наследование предусматривает добавление унаследованного определения к новому коду.

При объединенном наследовании, соответственно, объединяются новое и унаследованное определения предикатов. Этот механизм реализуется путем вызова единственного унаследованного определения с оператором ^^/1 в новом определении.

Выборочное наследование позволяет организовать сокрытие некоторых унаследованных определений. Эта форма наследования обычно используется для представления исключений в общих определениях.

Выводы

1. Смешанное представление знаний на основе объектных моделей и СС позволяет использовать преимущества объектно-ориентированного проектирования при создании программных систем, требующих семантической обработки понятий ПО.

2. В соответствии с концепцией ООЛП, принятой в Logtalk, интерпретация СС на Прологе упрощается за счет введения структурных отношений на объектах ПО согласно принципам объектно-ориентированного проектирования.

3. ST-сеть представляет собой разновидность семантической сети, в которой отображены связи объектов ПО с инструментальными предикатами этих объектов.

4. Процедурная семантика объектов графически не отражается в ST-сети, и в Logtalk реализуется только механизмами ООЛП.

Список литературы

1. Вагин В.Н. Дедукция и обобщение в системах принятия решений.–М.: Наука. гл. ред. физ. мат. лит. 1988.– 384 с.– (Пробл. искусств. интеллекта).
2. Lopes de Moura PJ (2003) Logtalk: Design of an object-oriented logic programming language. [Electronic resource] // PhD Thesis, Departamento de Informatica, Universidade da Beira Interior, Portugal. – Mode of access: logtalk.org/papers/thesis.pdf. – Last access: 2008. – Title from the screen.
3. Буч. Г. Объектно-ориентированное проектирование с примерами применения: Пер. с англ. – М.: Конкорд, 1992. – 519с.

УДК 519.718

РОМАНКЕВИЧ В.А.

НЕКОТОРЫЕ КОЛИЧЕСТВЕННЫЕ ОЦЕНКИ В *GL*-МОДЕЛЯХ

Работа посвящена определению некоторых граничных оценок, в частности, числа попарных рёберных циклов (ПРЦ), которые могут возникнуть при преобразованиях графо-логических моделей, отражающих реакцию отказоустойчивых многопроцессорных систем на появление отказов.

The work is dedicated for determination of some boundary estimation in particularly the numbers of pair edge cycles which may appear during transformation of graph-logical models reflecting the reaction of fault tolerant multiprocessor systems under appearance of faults.

1. Введение

Многопроцессорные системы с каждым годом находят все более широкое применение. В настоящей работе рассматриваются отказоустойчивые многопроцессорные системы (ОМС), создаваемые для повышения надёжности управления какими-либо ответственными объектами или процессами. Расчет надёжности, как одного из важнейших параметров таких систем, представляет собой определенные трудности именно для ОМС, и для решения этой задачи применяются разные методы [1,2,3], среди которых – использование графо-логических моделей (*GL*-моделей) [4]. Отличительной особенностью *GL*-моделей является сочетание свойств графов и булевых функций: ребрам графа, чья связность моделирует отказ системы, приписываются булевы функции. Базовые ОМС $K(m,n)$ и их модели (n – число процессоров системы, m – число отказов, к которым ОМС нечувствительна) отражают поведение сравнительно несложных систем в потоке отказов. Возможны ситуации, когда система становится неработоспособной при возникновении некоторых p отказов, в то же время оставаясь работоспособной при появлении других q отказов ($q>p$). Возникает необходимость отобразить в *GL*-модели тот факт, что на различные комбинации отказов одинаковой кратности система может реагировать по-разному. Иными словами, при построении моделей реальных ОМС иногда приходится модифицировать базовые модели, построенные на основе циклических графов, в частности, путем проведения дополнительных ребер со своими функциями [5].

Однако при этом могут возникнуть так называемые попарные реберные циклы (ПРЦ) [6], которые усложняют и модель, и процесс ее преобразования.

Суть попарных реберных циклов заключается в следующем. Пусть имеются 3 вектора состояния, появление каждого из которых приводит к выпадению в модели двух ребер. Допустим, что при появлении первого вектора состояния из модели удаляются ребра i и j, при появлении второго вектора состояния – ребра j и k, при появлении третьего вектора состояния – ребра i и k. Ребра i, j, k составляют, таким образом, своеобразный цикл, который и будем называть попарным реберным циклом (ПРЦ). В описанном примере ПРЦ состоит из трёх ребер, хотя в общем случае ПРЦ может содержать p ребер ($2 \le p \le n$).

2. Оценка числа векторов

Из [6] следует, что тройка векторов состояния ОМС, вызывающих появление в *GL*-модели определенного ПРЦ, не всегда является единственной. Попробуем определить количество различных комбинаций трех векторов из множества W, вызывающих один и тот же ПРЦ, для рассматриваемого в [6] случая $K(3,n)$. Это может оказаться полезным при дальнейших преобразованиях *GL*-модели. Напомним, что W – это множество векторов, имеющих 4 нулевых компоненты.

Вначале определим число векторов, которые приводят к пропаданию пары ребер, и используем для этого дерево иерархии. Для удобства читателя приведем пример реберных функций *GL*-модели $K(3,31)$ и соответствующего дерева иерархии, взятых из [6]. Дерево показано на рис.1, а реберные функции модели (каждая из которых соответствует какой-то вершине дерева), полученные по [7], следующие:

$$f_1 = x_1 \vee x_2 \vee x_3$$

$$f_2 = (x_1 \vee x_2)(x_1 x_2 \vee x_3) \vee x_4 x_5 x_6 x_7$$

$$f_3 = x_1 x_2 x_3 \vee (x_4 \vee x_5)(x_6 \vee x_7)(x_4 x_5 \vee x_6 x_7)$$

$$f_4 = x_4 x_5 \vee x_6 \vee x_7$$

$$f_5 = x_4 \vee x_5 \vee x_6 x_7$$

$$f_6 = x_8 x_9 \vee x_{10} \vee x_{11}$$

$$f_7 = x_8 \vee x_9 \vee x_{10} x_{11}$$

$$f_8 = (x_8 \vee x_9)(x_8 x_9 \vee x_{10} x_{11})(x_{10} \vee x_{11}) \vee x_{12} x_{13} x_{14} x_{15}$$

$$f_9 = x_8 x_9 x_{10} x_{11} \vee (x_{12} \vee x_{13})(x_{12} x_{13} \vee x_{14} x_{15})(x_{14} \vee x_{15})$$

$$f_{10} = x_{12} x_{13} \vee x_{14} \vee x_{15}$$

$$f_{11} = x_{12} \vee x_{13} \vee x_{14} x_{15}$$

$$f_{12} = \overbrace{(x_1 \vee x_2)(x_1 x_2 \vee x_3)(x_1 x_2 x_3 \vee x_4 x_5 x_6 x_7)(x_4 \vee x_5)(x_6 \vee x_7)(x_4 x_5 \vee x_6 x_7)}^{\sigma_1} \vee$$
$$\vee\, x_8 x_9 x_{10} x_{11} x_{12} x_{13} x_{14} x_{15}$$

$$f_{13} = \overbrace{(x_8 \vee x_9)(x_{10} \vee x_{11})(x_8 x_9 \vee x_{10} x_{11})(x_{12} \vee x_{13})(x_{14} \vee x_{15})(x_{14} x_{15} \vee x_{12} x_{13})(x_8 x_9 x_{10} x_{11} \vee x_{12} x_{13} x_{14} x_{15})}^{\sigma_2} \vee$$
$$\vee\, x_1 x_2 x_3 x_4 x_5 x_6 x_7$$

$$f_{14} = [\sigma_1 \cdot \sigma_2 \cdot (x_1 x_2 x_3 x_4 x_5 x_6 x_7 \vee x_8 x_9 x_{10} x_{11} x_{12} x_{13} x_{14} x_{15})] \vee x_{16} x_{17} x_{18} \ldots x_{31}$$

$$f_{15} = \overbrace{(x_{16} \vee x_{17})(x_{18} \vee x_{19})(x_{16} x_{17} \vee x_{18} x_{19})}^{\sigma_3} \vee x_{20} x_{21} x_{22} x_{23}$$

$$f_{16} = \overbrace{(x_{20} \vee x_{21})(x_{22} \vee x_{23})(x_{20} x_{21} \vee x_{22} x_{23})}^{\sigma_4} \vee x_{16} x_{17} x_{18} x_{19}$$

$$f_{17} = x_{20} x_{21} \vee x_{22} \vee x_{23}$$

$$f_{18} = x_{20} \vee x_{21} \vee x_{22} x_{23}$$

$$f_{19} = x_{16} \vee x_{17} \vee x_{18} x_{19}$$

$$f_{20} = x_{16} x_{17} \vee x_{18} \vee x_{19}$$

$$f_{21} = \overbrace{(x_{24} \vee x_{25})(x_{26} \vee x_{27})(x_{24} x_{25} \vee x_{26} x_{27})}^{\sigma_5} \vee x_{28} x_{29} x_{30} x_{31}$$

$$f_{22} = \overbrace{(x_{28} \vee x_{29})(x_{30} \vee x_{31})(x_{28} x_{29} \vee x_{30} x_{31})}^{\sigma_6} \vee x_{24} x_{25} x_{26} x_{27}$$

$$f_{23} = x_{28} \vee x_{29} \vee x_{30} x_{31}$$

$$f_{24} = x_{28} x_{29} \vee x_{30} \vee x_{31}$$

$$f_{25} = x_{24} \vee x_{25} \vee x_{26} x_{27}$$

$$f_{26} = x_{24} x_{25} \vee x_{26} \vee x_{27}$$

$$f_{27} = \overbrace{\sigma_3 \cdot \sigma_4 \cdot \left(x_{16}x_{17}x_{18}x_{19} \vee x_{20}x_{21}x_{22}x_{23}\right)}^{\delta_1} \vee x_{24}x_{25}\ldots x_{31}$$

$$f_{28} = \overbrace{\sigma_5 \cdot \sigma_6 \cdot \left(x_{24}x_{25}x_{26}x_{27} \vee x_{28}x_{29}x_{30}x_{31}\right)}^{\delta_2} \vee x_{16}x_{17}\ldots x_{23}$$

$$f_{29} = \delta_1 \cdot \delta_2 \cdot \left(x_{16}x_{17}\ldots x_{23} \vee x_{24}x_{25}\ldots x_{31}\right) \vee x_1x_2x_3\ldots x_{15}$$

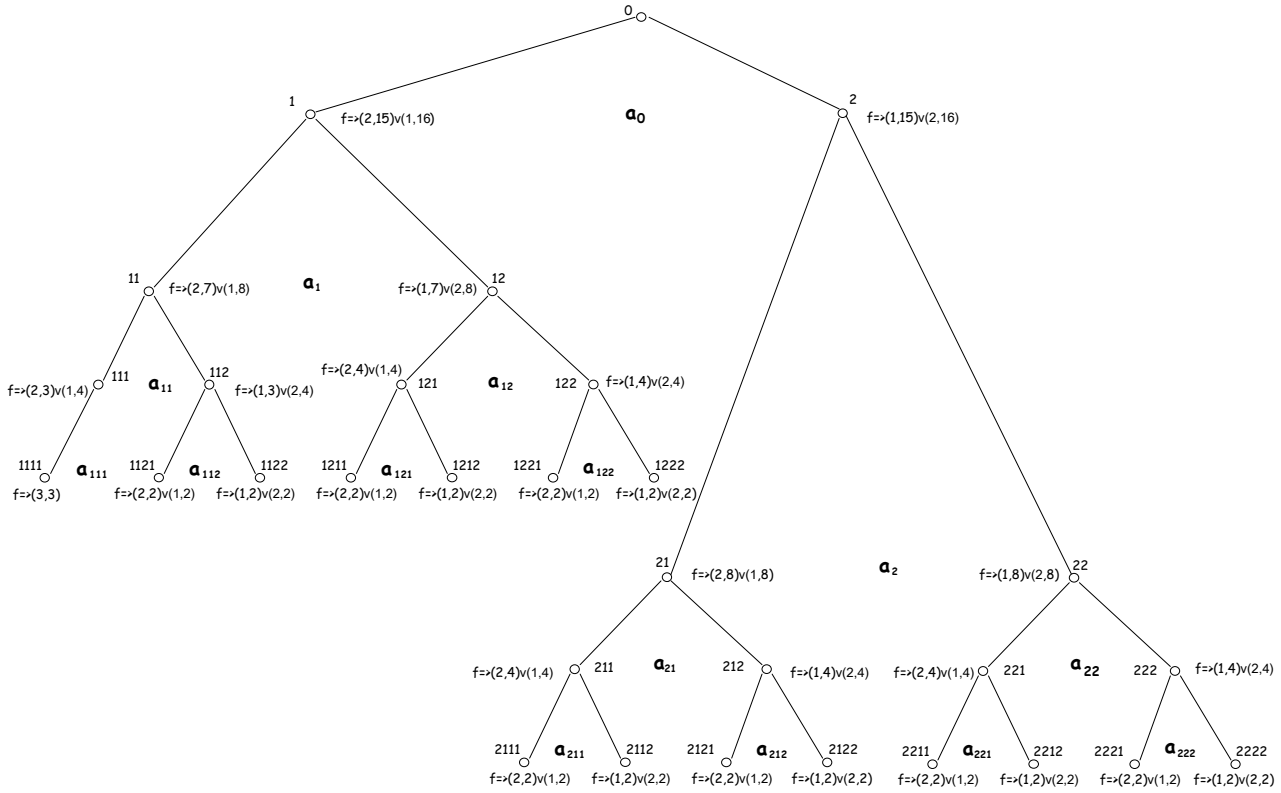


Рис. 1.

Обратимся к *утверждению* 2 из [6] и рассмотрим условие *а*) то есть, случай с двумя вершинами одного уровня. Если вспомнить, что функции, приписываемые ребрам, соответствующим этим вершинам, представляются выражениями

$f_{i1} = \varphi_1(2, n_1) \vee \varphi_2(1, n_2)$,

$f_{i2} = \psi_1(1, n_1) \vee \psi_2(2, n_2)$,

то станет понятным, что число векторов из множества *W*, обнуляющих эту пару функций

$S_{ii} = C_{n1}^2 \times C_{n2}^2$,

поскольку каждая из них должна содержать по две нулевых компоненты в каждом из подмножеств n_1 и n_2 переменных. Для простоты в выражении для S_{ii} (и далее также для S_{ij}) принимаем одно и то же обозначение – n_i – для множества n_i и его мощности. Например, ребра, соответствующие вершинам дерева 11 и 12, выпадают одновременно при появлении любого из $C_7^2 \times C_8^2 = 588$ векторов. В то же время пропаданию ребер, соответствующих вершинам 1121 и 1122, соответствует единственный вектор

x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	…	x_{31}
1	1	1	0	0	0	0	1	…	1

С другой стороны, пара ребер, соответствующих вершинам, расположенным на одной ветви дерева, т.е. условию а) из *утверждения* 2 из [6], с функциями f_i (ближе к корню) и f_j

$f_i = \varphi_1(2, n_{i1}) \vee \varphi_2(1, n_{i2})$,

$f_j = \psi_1(2, n_{j1}) \vee \psi_2(1, n_{j2})$

выпадает в *GL*-модели при появлении любого из $S_{ij} = C_{ni2}^1 \times C_{nj1}^2 \times C_{nj2}^1$ векторов. Действительно, напомним, что, в соответствии с принципом построения дерева, подмножества переменных n_{j1} и n_{j2} принадлежат множеству n_{i1}, и нулевые компоненты вектора в подмножестве n_{i1} расположены в строгом соответствии: две – в n_{j1} и одна – в n_{j2}. Четвертая нулевая компонента расположена в подмножестве n_{i2}, что в совокупности объясняет выражение для S_{ij}.

Перейдем к ПРЦ – они образуются тремя парами ребер. Докажем *Лемму*, которая в значительной степени упростит решение задачи.

Пусть множество векторов S_1 обнуляет одну и ту же пару функций f_1 и f_2 в *GL*-модели, а множество S_2 – пару функций g_1 и g_2.

Лемма

$S_1 \cap S_2 = \varnothing$.

Действительно, в противном случае нашёлся бы вектор $w_i \in W$, появление которого вызвало бы пропадание по крайней мере 3-х ребер в *GL*-модели, построенной по алгоритму, описанному в [7], что невозможно. Справедливость леммы не нарушается и в том случае, когда рассмотренные пары имеют общее ребро.

Лемма позволяет определить число комбинаций векторов, вызывающих появление конкретного ПРЦ, состоящего из трех ребер *a, b* и *c*:

$S_{abc} = S_{ab} \times S_{ac} \times S_{bc}$.

Очевидно, что количество этих комбинаций растет по мере приближения к корню дерева. Поскольку ребро, соответствующее вершине 0 отсутствует, максимум достигается во втором ярусе. Для нашего примера *K*(3,31) это вершины 2, 21, и 22. Минимум достигается на нижнем ярусе дерева, и существуют ПРЦ, которые образуются сравнительно небольшим числом комбинаций из трех векторов. В нашем примере ПРЦ, включающий вершины 112, 1121 и 1122, образуется 36-ю комбинациями из 3-х векторов $w_i \in W$.

3. Нижняя граница числа комбинаций из трех векторов $w_i \in W$, образующих ПРЦ

Если в общем случае максимальное число $P_{\max}$ комбинаций трех векторов $w_i \in W$, образующих один и тот же ПРЦ, зависит от величины *n* и не может иметь границу в виде конкретного числа, то величину $P_{\min}$ (ее теоретическое значение) можно определить. Понятно, что системы (и модели) *K*(3,1) и *K*(3,2) не существуют. Модель *K*(3,3) содержит одно, а модель *K*(3,4) – два ребра, и на наличие ПРЦ их нет смысла исследовать. Модель *K*(3,5) содержит три ребра, и ее дерево иерархии имеет следующий вид (см. рис. 2):

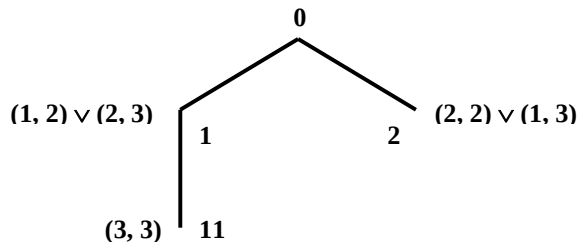


Рис. 2

Ребра, соответствующие вершинам 1, 2 и 11, не могут образовать ПРЦ, так как не соответствуют ни одному из условий *утверждения* 3 из [6].

То же самое можно сказать о модели *K*(3,6), где четыре ребра (см. рис. 3). Напомним, что нулевая вершина не имеет соответствующего ребра в *GL*-модели.

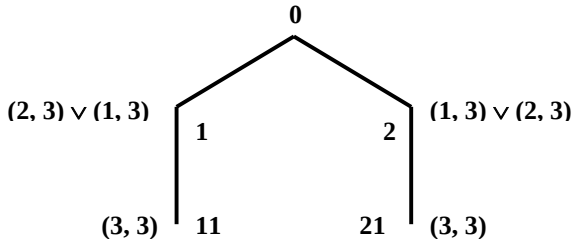


Рис. 3

Модель *K*(3,7) и ее дерево иерархии (см. рис. 4) свидетельствуют о возможности возникновения ПРЦ, как было сказано, в 36-ти случаях из трех векторов $w_i \in W$. Это и есть реальная граница $P_{\min}$ для *GL*-моделей, построенных по алгоритму [7].

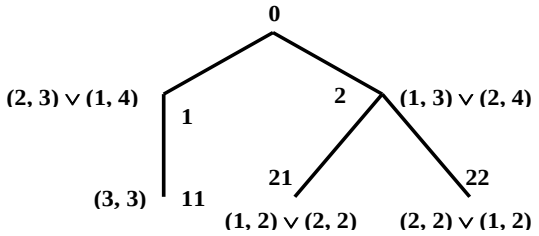


Рис. 4

Однако, если отклониться от требования разбиения множества переменных на две равные части, то ПРЦ оказывается возможным при $n<7$. Действительно, для построения модели *K*(3,6) можно разбить переменные на подмножества $\{x_1, x_2\}$ и $\{x_3, x_4, x_5, x_6\}$. Модель будет иметь 4 ребра, а ее дерево иерархии (см. рис. 5)

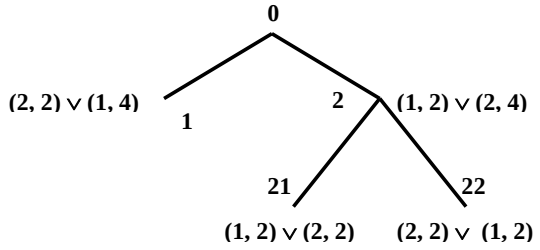


Рис. 5

позволяет определить число комбинаций из трех векторов (P=16) для единственного ПРЦ, который может быть образован ребрами, соответствующими вершинам 2, 21 и 22. Однако и это еще не предел, ибо модель K(3,5) при разбиении переменных на подмножества $\{x_1\}$ и $\{x_2, x_3, x_4, x_5\}$ приведет к модели, содержащей всего 3 ребра, и дереву, не имеющему нулевой вершины (см. рис. 6).

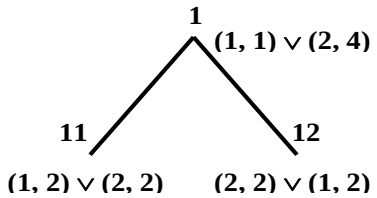


Рис. 6

Здесь единственный ПРЦ может быть образован всего четырьмя комбинациями из 3-х векторов $w_i \in W$. Случай $K(3,4)$ и $K(3,3)$ не рассматриваем, так как соответствующие модели имеют меньше 3-х ребер. Таким образом, теоретическая граница $P_{\min}$=4, и комбинации векторов следующие

$x1$	$x2$	$x3$	$x4$	$x5$
0	0	1	0	0
0	0	0	0	1
1	0	0	0	0
0	1	0	0	0
0	0	0	0	1
1	0	0	0	0
0	1	0	0	0
0	0	0	1	0
1	0	0	0	0
0	0	1	0	0
0	0	0	1	0
1	0	0	0	0

Однако при этом имеет место нарушение упомянутого выше алгоритма формирования модели.

4. Число различных ПРЦ

Мы рассматривали варианты множеств из 3-х векторов $w_i \in W$, которые образуют один и тот же ПРЦ. Иными словами для блокирования потери связности графом *GL*-модели при появлении любого из этих векторов понадобится в общей сложности 2 дополнительных ребра.

Интерес также представляет оценка числа E разных ПРЦ, которые могут быть образованы в модели. Приведем соотношение, позволяющее определить это число

$$E = r \cdot C_{q-1}^{2} + \sum_{i=1}^{q-3} 2^{q-i} C_{q-i-1}^{2} + s \cdot C_{q-1}^{1} + \sum_{i=2}^{q-1} 2^{q-i} C_{q-i}^{1}$$

Здесь r – общее число вершин последнего каскада дерева иерархии;

s – число парных вершин этого каскада;

q – число каскадов дерева: q=[log$n$]-1, [x] – ближайшее большее целое.

Доказательство этого соотношения достаточно громоздко, но несложно, поэтому мы его опускаем. Для приведенного выше дерева (n=31) число различных ПРЦ составляет (r=15, s=7, q=4)

E=15$\times$3+8$\times$1+7$\times$3+4$\times$2+2$\times$1=85

В заключение приведем некоторые количественные характеристики, полученные в результате эксперимента с выбранной выше в качестве примера *GL*-моделью $K(3,31)$ на предмет возникновения ПРЦ (см. рис.7), представленные в виде графика. Здесь по горизонтали отмечается число t векторов $w_i \in W$, которые предстоит блокировать; по вертикали – число v ПРЦ обоих видов, рассмотренных выше. Кривые отмечают число $v_{\text{ср}}$ ПРЦ, которые возникают при появлении t векторов: $v_{\text{ср}}$ – среднее значение числа ПРЦ при выполнении 5000 экспериментов со случайно выбранными множествами из t векторов. График иллюстрирует явную нелинейность зависимости. В то же время понятно, что при достижении некоторого значения t_n, зависящего от величины n, кривые будут более пологими.

Отметим также, что если два ПРЦ образованы разными ребрами, то вовсе не обязательно для их блокирования увеличивать вдвое число дополнительных рёбер. В общем виде задача оптимального определения числа дополнительных ребер и места их проведения решена в [5].

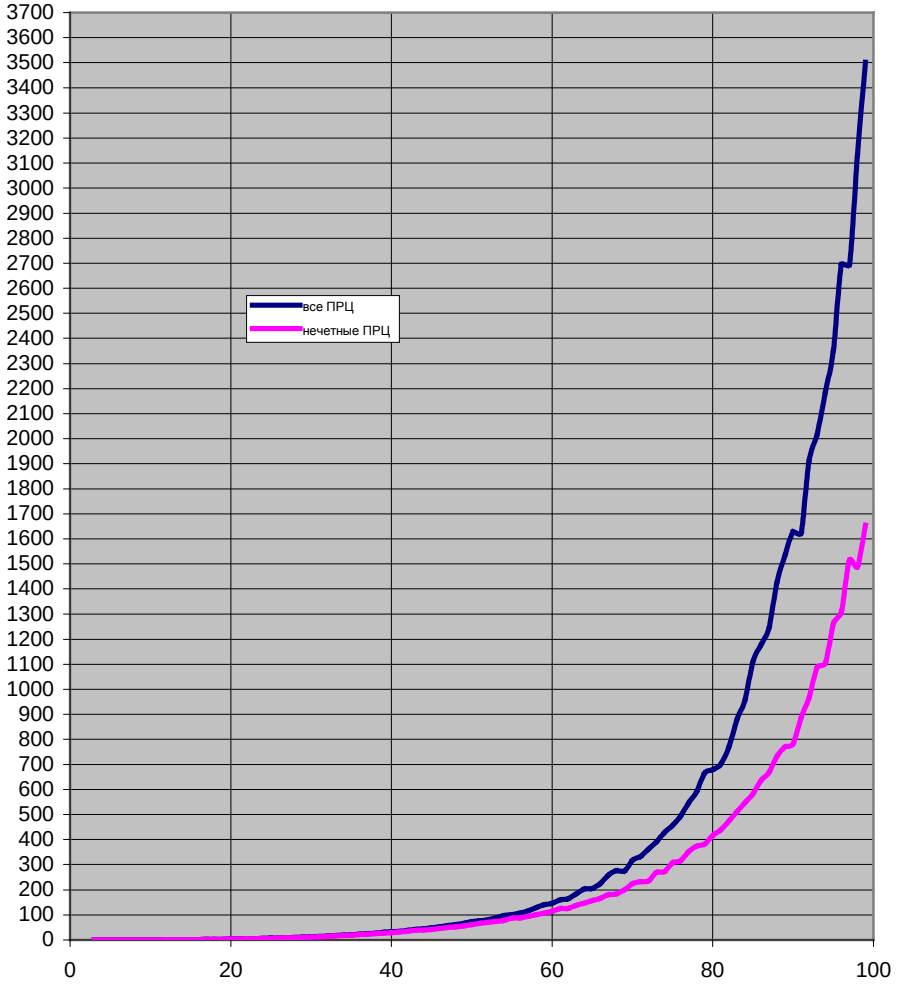


Рис. 7

Список литературы

1. Многоверсионные системы, технологии, проекты / В.С.Харченко, В.Я.Жихарев, В.М.Илюшко, Н.В.Нечипорук. – Харьков: Нац. Аэрокосм. Ун-т «Харьк. авиац. ин-т», 2003. – 486 с.
2. H. Yamamoto, V.J. Zuo, T. Akiba, Z. Tian Recursive Formulas for the Reliability of Multi-State Consecutive-*k*-out-of-*n*:G Systems // IEEE Transactions On Reliability.– №1, Vol. 55.– March 2006.– p.98-105.
3. M.J. Zuo, D. Lin, Y. Wu Reliability evaluation of combined *k*-out-of-*n*:F, consecutive-*k*-out-of-*n*:F and linear connected-(r,s)-out-of-(m,n):F system strucyures // IEEE Transactions On Reliability.– №1, Vol. 49.– March 2000.– p.99-104.
4. Романкевич А.М., Карачун Л.Ф., Романкевич В.А. Графо-логические модели для анализа сложных отказоустойчивых вычислительных систем // Электронное моделирование.– 2001.-т.23, №1.-С.102-111.
5. Романкевич А.М., Иванов В.В., Романкевич В.А. Анализ отказоустойчивых многомодульных систем со сложным распределением отказов на основе циклических GL-моделей // Электронное моделирование.–№5, т.26.– 2004.– с.67-81.
6. Романкевич В.А., Кононова А.А., Бахтари Хедаятоллах Условия существования попарных рёберных циклов в *GL*-моделях *K*(3,*n*) // Вісник НТУУ “КПІ”.– Інформатика, управління та ОТ.–2007.– №46.–С.54-61.
7. Романкевич В.А., Потапова Е.Р., Бахтари Хедаятоллах, Назаренко В.В. GL-модель поведения отказоустойчивых многопроцессорных систем с минимальным числом теряемых рёбер // Вісник НТУУ “КПІ”.– Інформатика, управління та ОТ.–2006.– №45.–С.93-100.

УДК 519.5:681:513

ПАВЛОВ А.А.,
ЛИЩУК Е.И.

ОПЕРАТИВНЫЕ АЛГОРИТМЫ ПРИНЯТИЯ РЕШЕНИЙ В ИЕРАРХИЧЕСКОЙ СИСТЕМЕ СААТИ, ОСНОВАННЫЕ НА ЗАМЕЩЕНИИ КРИТЕРИЕВ

В статье рассматривается оптимизационная задача управления проектами, когда достижение глобальной цели может быть представлено иерархическим деревом Саати. Предложены и обоснованы оперативные алгоритмы принятия решений, основанные на возможности замещения критериев.

In article the optimization task of projects management when achievement of the global purpose may be submitted by T.Saaty hierarchical tree is considered. Operative algorithms of the decisions making, which is based on an opportunity of the replacements of criteria are proposed and substantiated.

Введение

В настоящее время одной из наиболее актуальных задач в области стратегического планирования является задача управления проектами. Под управлением проектами понимается деятельность, в ходе которой определяются и достигаются определённые цели с максимально возможной эффективностью, а также оптимизируется использование ресурсов (например, время, денежные средства и др.) при заданных ресурсных ограничениях и качеством конечных результатов проекта (например, документированных в техническом задании). Следовательно, задачу управления проектами можно рассматривать как задачу многокритериального выбора.

В настоящее время для решения задач многокритериального выбора наиболее часто используемым является метод анализа иерархий (МАИ), предложенный Т.Саати. В качестве задач многокритериального выбора могут выступать как задачи выбора одного варианта из нескольких предлагаемых, так и задачи управления проектами, в том случае, когда задачу управления проектами (прогнозирования) можно представить в виде иерархического дерева Саати. Метод анализа иерархий предполагает декомпозицию проблемы на более простые составляющие части и обработку суждений лица, принимающего решение (ЛПР). В МАИ основным методом сравнения ЛПР объектов (альтернатив, критериев) является метод парных сравнений.

Общая постановка задачи

Имеется m участников конкурса на выполнение проекта (альтернатив), который характеризуется значениями n критериев $x^j,\ j=\overline{1,m}$ $x^j=\begin{pmatrix} x_1^j \\ \vdots \\ x_n^j \end{pmatrix}$. Ограничения на реализацию проекта у каждого из участников являются линейными.

Применим последовательно по этапам МАИ для поставленной задачи [1,2]. В результате получим задачу принятия решений, представленную в иерархической форме (Рис. 1).

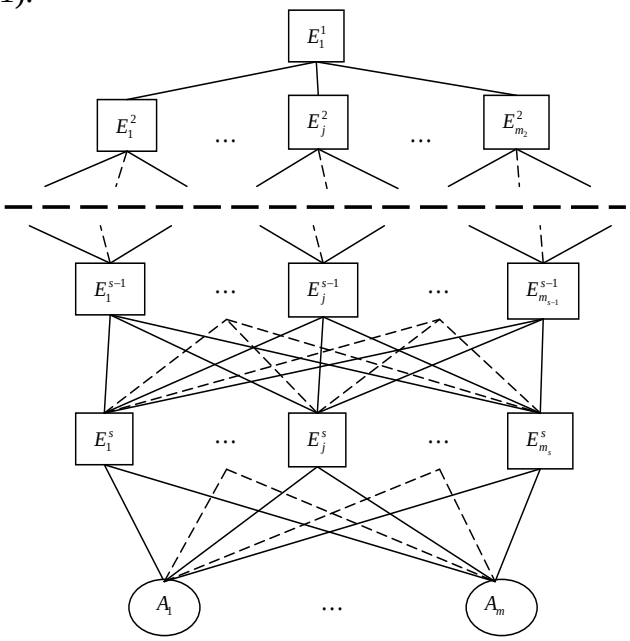


Рис. 1 Иерархическое представление решаемой задачи принятия решений

В полученной задаче имеем m альтернатив $A_{1\ldots}A_m$ и s уровней критериев $E_j^i,\ i=\overline{1,s},\quad j=\overline{1,m_i}$. Веса $w_j,\ j=\overline{1,m}$ альтернатив в глобальном критерии можно определить с помощью следующего функционала (1) [3]:

$$w_j = \sum_{t=1}^{m_s} \alpha_t \sum_{p=1}^{n_t} \alpha_p^{E_t^s} x_{t_p}^j, \quad j = \overline{1,m}, \ (1)$$

где α_t – соответствующие коэффициенты пропорциональности,

$\alpha_p^{E_t^s}$ – веса в $E_t^s, t = \overline{1,m_s}$ критерии, задаваемые экспертом, с учетом непосредственной связи между критериями нижнего (s -того) уровня иерархии и альтернативами $A_1 \ldots A_m$,

$x_{t_p}^j$ – значения критериев, характеризующих каждую альтернативу.

Задачи оперативной корректировки алгоритма принятия решения по глобальной цели на основе возможности замещения критериев с учетом ограничений на замещение обосновывается следующим образом:

критерий x^j как решение следующей задачи линейного программирования (2)-(3):

$$\max_{X^j} w_j = \max_{X^j} \sum_{t=1}^{m_s} \alpha_t \sum_{p=1}^{n_j} \alpha_p^{E_t^s} x_{t_p}^j, \ (2)$$

$$A_j x^j \le b_j. \ (3)$$

это стратегическая схема построения проекта A_j, характеризующаяся вектором x^j. При этом ограничения (3) являются ресурсными ограничениями, и в большинстве практических случаев они задаются линейными ограничениями.

Оперативная корректировка проекта x^j (без принятия стратегически новых решений) заключается в следующем: в заданных пределах изменять значения критериев, без изменения общей структуры реализации альтернативы A_j, которая характеризуется вектором x^j.

Рассмотрим альтернативу x^j. Путем замещения компонент альтернативы x^j пытаемся выиграть конкурс у альтернатив $x^i, i = \overline{1,m}, i \ne j$. По альтернативе x^j имеем следующую информацию: для $\forall l \in L_j^1$ (множество индексов) известно $x_l^j > x_l^i, \mathrm{i} = \overline{1,m}, \mathrm{j} \ne \mathrm{i}$.

Компоненты вектора x^j $x_l^j, l \in L_j^1$ могут участвовать в замещении компонент вектора x^j с индексами из множества $L_j^2(l)$ следующим образом:

$$\Delta x_l^{jl_m} = a_l^{jl_m} \Delta x_{l_m}^j, \ \begin{matrix} l_m \in L_j^2(l) \\ l \in L_j^1 \end{matrix}, \text{ где}$$

$\Delta x_l^{jl_m}$ – возможное уменьшение x_l^j компоненты с целью увеличения значения $x_{l_m}^j$ компоненты,

$a_l^{jl_m}$ – заданные коэффициенты пропорциональности, зависящие от сущности l и l_m компонент j -ой альтернативы,

$$0 \le \sum_{l_m \in L_j^2(l)} \Delta x_l^{jl_m} \le b_{x_l^j},$$

$b_{x_l^j}$ – заданные коэффициенты, показывающие возможную границу уменьшения компоненты $x_l^j, l \in L_j^1$.

В рамках такой постановки возможно решение следующих задач оперативной коррекции процесса принятия решений.

Задача 1 оперативной коррекции процесса принятия решений

Выяснилось, что в функционале (1) значения коэффициентов α_t и $\alpha_p^{E_t^s}$ неизвестны. Известно только то, что они не отрицательны. При этом необходимо отметить, что x^j альтернативе, у которой $\forall i = \overline{1,n}$ $x_i^j \ge x_i^l$ $\forall l \ne j$ соответствует $\max_i w_i$.

В данной задаче оперативной коррекции процесса принятия решений под множеством индексов $\forall l \in L_j^1$ подразумевается множество, состоящее из индексов компонент альтернатив, для которых выполняется $x_l^j > x_l^i, \mathrm{i} = \overline{1,m}, \mathrm{j} \ne \mathrm{i}$.

Тогда поставленная задачи сводится к решению следующей задачи линейного программирования:

$$\max\{\sum_{l \in L_j^1} \alpha_l + \sum_{\forall r \in \bar{L}_j^1} \alpha_r\}$$

Примечание. В ограничениях не участвуют k компоненты альтернатив x^j, для которых $x_k^j = const, \ j = \overline{1,m}$.

при следующих ограничениях:

$$\forall \Delta x_l^{jl_m} \ge 0$$

$$\alpha_l \ge 0, l \in L_j^1, \alpha_r \ge 0, l \in L_j^1, r \bar{\in} L_j^1$$

$$x_l^j - \sum_{l_m \in L_j^2(l)} \Delta x_l^{jl_m} \ge \max_i x_l^i + \alpha_l, \forall i \in \overline{1,n},\ j \ne i, l \in L_j^1$$

$$x_r^j + \sum_{l \in L_j^1} \sum_{l_m \in L_j^2(l)} \frac{1}{a_l^{jl_m}} \Delta x_l^{jl_m} \ge \max_i x_r^i + \alpha_r, i = \overline{1,n}, j \ne i,\ \forall r \bar{\in} L_j^1,\ \forall l_m = r$$

$$\forall r \bar{\in} L_j^1, \exists l, r \in L_j^2(l)$$

$$\sum_{l_m \in L_j^2(l)} \Delta x_l^{jl_m} \le b_{x_l^j},\ j \ne i,\ l \in L_j^1,\ \forall l_m = r.$$

В выше приведенной задаче линейного программирования переменными являются $\Delta x_l^{jl_m}, \alpha_l, \alpha_r$.

Задача 2 оперативной коррекции процесса принятия решений

Коэффициенты в (1) оперативно изменяются и известны. Тогда оперативным замещением решается m задач ЛП по max нового функционала (1). Побеждает та альтернатива, у которой достигается максимум функционала (1).

При этом добавляются ограничения на оперативно изменяющиеся компоненты альтернатив $x_l^j \ge c_l$, $j = \overline{1,m}$, что является гарантией «технологичности» ограничения.

В приведенной ниже задаче принятия решений под множеством индексов L_j^1 подразумевается множество индексов компонент, для которых выполняются веденные «технологические» ограничения $x_l^j > c_l$, $j = \overline{1,m}$, т.е. это те компоненты альтернатив, которые могут уменьшать свои значения.

Множество индексов L_j^2 – это множество индексов компонент, которые могут увеличивать свои значения (куда можно добавлять), т.е. те компоненты для которых $x_l^j < c_l$, $j = \overline{1,m}$ (перечень компонент, за счет которых можно увеличивать значения, задается множеством индексов L_j^1).

$$\max\left\{ \sum_{l \in L_j^1} \beta_l (x_l^j - \sum_{l_m \in L_j^2} \Delta x_l^{jl_m}) + \sum_{l_m \in L_j^2} \beta_{l_m} (x_{l_m}^j + \sum_{l \in L_j^1} \frac{1}{a_l^{jl_m}} \Delta x_l^{jl_m}) \right\}$$

$$\Delta x_l^{jl_m} \ge 0,\ \ l \in L_j^1, l_m \in L_j^2 \ \sum_{l_m \in L_j^2} \Delta x_l^{jl_m} \le b_{x_l^j},\ \ l \in L_j^1$$

$$x_l^j - \sum_{l_m \in L_j^2} \Delta x_l^{jl_m} \ge c_l, \quad l \in L_j^1\ x_t^j + \sum_{l \in L_j^1} \sum_{l_m \in L_j^2} \frac{1}{a_l^{jl_m}} \Delta x_l^{jl_m} \ge c_t,\ l \in L_j^1,\ \ l_m \in L_j^2,$$

$$\forall l_m = t,\ t \in L_j^2$$

В приведенной задаче линейного программирования переменными являются $\Delta x_l^{jl_m}, \beta_l, \beta_{l_m}$.

Задача 3 оперативной коррекции процесса принятия решений

Постановка для третей задачи оперативной коррекции принятия решений не изменяется. Т.е. коэффициенты в (1) оперативно изменяются и известны. Тогда оперативным замещением решается m задач ЛП по max нового функционала (1). Побеждает та альтернатива, у которой достигается максимум функционала (1).

Добавляются ограничения на оперативно изменяющиеся компоненты альтернатив $x_l^j \ge c_l$, $j = \overline{1,m}$, что является гарантией «технологичности» ограничения.

Отличие данной задачи от второй задачи принятия решений состоит в определении множеств индексов L_j^1 и L_j^2. В данной задаче

принятия решений под множеством индексов L_j^1 подразумевается множество индексов компонент, у которых в процессе оперативной коррекции новые коэффициенты в (1) уменьшились.

Наилучший вариант получается в том случае, существует некое подмножество компонент $L_j^{''} \subset L_j^1$ такое, что $x_l^j > c_l \quad \forall\{a_l^{jl_m}\} = J_l, a_l^j \geq 1, \bigcup_l J_l = L_j^2$, где $a_l^{jl_m} \geq 1$, коэффициенты в функционале (1) уменьшились и нижеприведенная задача разрешима.

$$\max\left\{\sum_{l\in L_j^{''}} \beta_l (x_l^j - \sum_{l_m \in J_l} \Delta x_l^{jl_m}) + \sum_{l_m \in J_l} \beta_{l_m} (x_{l_m}^j + \sum_{l\in L_j^{''}} \frac{1}{a_l^{jl_m}} \Delta x_l^{jl_m})\right\}$$

$$\Delta x_l^{jl_m} \geq 0, \quad l \in L_j^{''}, l_m \in J_l$$

$$\sum_{l_m \in J_l} \Delta x_l^{jl_m} \leq b_{x_l^j}, \quad l \in L_j^{''}$$

$$x_l^j - \sum_{l_m \in J_l} \Delta x_l^{jl_m} \geq c_l, \quad l \in L_j^{''}$$

$$x_t^j + \sum_{l\in L_j^{''}} \sum_{l_m \in J_l} \frac{1}{a_l^{jl_m}} \Delta x_l^{jl_m} \geq c_t, \; l \in L_j^{''}, \; l_m \in J_l,$$

$$\forall l_m = t, t \in J_l$$

В приведенной задаче линейного программирования переменными являются $\Delta x_l^{jl_m}, \beta_l, \beta_{l_m}$.

Данная задача является наилучшим случаем оперативной коррекции процесса принятия решений. Однако данную задачу решать не требуется, т.к. она автоматически получается из Задачи 2 оперативной коррекции процесса принятия решений.

Заключение

В статье рассмотрена оптимизационная задача управления проектами, когда достижение глобальной цели представлено иерархическим деревом Саати. Предложены и обоснованы возможные методы оперативной коррекции процесса принятия решений, при котором в заданных пределах выполняется изменение значений критериев одного из проектов.

Предложенные методы могут использоваться не только для задач оперативной коррекции значений критериев одного из проектов на этапе принятия решения, но и для задач прогнозирования.

Список литературы

1. Саати Т. Принятие решений. Метод анализа иерархий / Пер. с англ. Р.Г. Вачнадзе.– М.: Радио и связь, 1993.– 315 с.
2. Саати Т., Кернс К. Аналитическое планирование. Организация систем / Пер. с англ. Р.Г. Вачнадзе.– М.: Радио и связь, 1991.– 223 с.
3. Павлов А.А., Лищук Е.И. Принятие решений на основе метода анализа иерархий. // Вестник НТУ «ХПИ». Системный анализ, управление и информационные технологии. – 2007. – №41. – С. 69 – 76.
4. Павлов А.А., Лищук Е.И. Оперативное корректирование в задаче многокритериального выбора. / Автоматизированные системы управления и приборы автоматики. – 2007. – В печати.

УДК 681.322

МЄЛКОВСЬКА В.М.
СЕРГІЄНКО А.М.
СТІРЕНКО С.Г.

СПОСІБ ДЕМОДУЛЯЦІЇ СИГНАЛІВ З БАГАТОЧАСТОТНОЮ МОДУЛЯЦІЄЮ

Запропоновано виконувати демодуляцію сигналів з багаточастотною модуляцією за допомогою ковзного рекурсивного дискретного перетворення Фур′є (ДПФ) з використанням арифметики раціональних дробів, завдяки чому мінімізуються похибки обчислень. Висока ефективність таких обчислень перевірена в процесорі ДПФ, який реалізовано в програмованій логічній інтегральній схемі (ПЛІС).

An algorithm of the sliding recursive discrete Fourier transform (DFT) for the DTMF signal decoding. The algorithm uses the rational fraction numbers for the computational error minimization. The good parameters of the DFT processor configured in FPGA are achieved due to this algorithm implementation.

При багаточастотній модуляції (БЧМ) сигнал представляють як суміш кількох синусоід з різними частотами, фазами і амплітудами. БЧМ широко використовується в техніці зв′язку, наприклад, як двохтональна багаточастотна модуляція (DTMF), багатоканальна модуляція з ортогональними сигналами (OFDM).

Сигнали з БЧМ детектують за допомогою гребінки вузькосмугових фільтрів. Якщо в спектрі сигнала частоти кратні, то для реалізації гребінки фільтрів використовують швидке перетворення Фур′є (ШПФ), як наприклад, у випадку OFDM-модуляції [1].

Для детектування сигналів з DTMF, коли частоти є невзаємнократними, набув широкого поширення алгоритм Герцеля, який обчислює дискретне перетворення Фур′є (ДПФ) окремо для кожного частотного каналу [2]. Недолік алгоритму Герцеля полягає в його нестабільності, що примушує виконувати обчислення з плаваючою комою, якщо необхідні стійкі оцінки амплітуд синусоідальних сигналів.

Ковзне рекурсивне дискретне перетворення Фур′є (ДПФ) широко відоме в літературі, присв′яченій цифровій обробці сигналів, як простий алгоритм ДПФ, який має майже мінімальну кількість обчислень [1]. Передаточна функція одного k-го каналу N –точкового ДПФ виглядає як

$$H_k(z) = \frac{1 - z^N}{1 - z^{-1} W^k}, \qquad (1)$$

де $W^k = e^{j2\pi k/N} = \cos\theta + j\sin\theta$ – комплексний обертаючий коефіцієнт. В порівнянні з поширеним ШПФ, це перетворення дає змогу виконувати розрахунки для будь-якого значення N і має переваги за апаратними витратами, якщо кількість каналів обробки істотно менше N. Крім того, при DTMF параметр N необхідно підбирати таким чином, щоб на N відліків сигналу припадало ціле число періодів вхідного синусоідального сигналу з частотою, яка відповідає θ, що протирічить вибору розмірності ШПФ.

Передаточна функція (1) відповідає моделі ідеального резонатора з центральною частотою $\theta = 2\pi k/N$ від частоти дискретизації. Тому обчислення рекурсивного ДПФ нестійкі через похибку обчислень і через неточність представлення обертаючого коефіцієнта, модуль якого повинен бути строго рівним одиниці. Для реалізації рекурсивного ДПФ в пристроях обробки сигналів, наприклад, в фільтрах з частотною виборкою, складові z домножують на загрубляючий коефіцієнт, який трохи зсуває полюси функції (1) до центра одиничного кола [1].

В статті пропонується для виконання рекурсивного ДПФ використати обчислення з представленням чисел у вигляді раціональних дробів. Рацінальний дріб a/b – це числовий об′єкт, який складається з цілочисельних чисельника a і знаменника b. Цей дріб має ту властивість, що він може з досить високою точністю наближатись до заданого ірраціонального або трансцендентного числа $x \approx a/b$. Раціональні дроби мають нескладний апарат арифметичних дій. Так, множення a/b на c/d дорівнює $ac/(bd)$, а додавання цих дробів – $(ad+bc)/(bd)$. При цьому власне ділення цілих чисел не виконується.

При порівнянні складності операцій слід прийняти до уваги, що розрядність чисельни-

ків і знаменників приблизно вдвічі менша, ніж така, що необхідна для операцій з цілими числами при такій самій точності представлення чисел. Тому складність апаратних суматорів дробів наближається до складності блоків множення цілих чисел, а складність блоків множення дробів приблизно вдвічі менше, ніж блоків множення цілих чисел. Ефективність арифметики дробів перевірена при вирішенні задач лінійної алгебри на спеціалізованих обчислювачах [3]. Ця арифметика підтримується також тим, що в сучасних програмованих логічних інтегральних схемах (ПЛІС), які часто використовуються для цифрового декодування сигалів, кількість вбудованих блоків множення досягає кількох сотен і в багатьох випадках вони не всі використовуються.

Обчислення згідно зі знаменником дробу (1) виконуються у відповідності з наступними різницевими рівняннями

$$YR(i+1) = YRi \cdot \cos\theta - YIi \cdot \sin\theta + XRi;$$
$$Y_{I(i+1)} = Y_{Ii} \cdot \cos\theta + Y_{Ri} \cdot \sin\theta + X_{Ii}, \quad (2)$$

де $\boldsymbol{X}_i = X_{Ri} + jX_{Ii}$ – відліки, що дорівнюють різниці відліків вхідного сигналу і таких самих відліків, затриманих на N тактів, $\boldsymbol{Y}_i = Y_{Ri} + jY_{Ii}$ – відліки ковзного спектру на частоті θ. При виконанні обчислень вектор $\boldsymbol{Y}_{i+1}$, який повернуто на кут θ подається на вхід формул (2) як $\boldsymbol{Y}_i$. При умові точних обчислень, якщо не справджується рівність Піфагора: $\sin^2\theta + \cos^2\theta = 1$, то похибка обчислення вектора $\boldsymbol{Y}_{i+1}$ призведе до швидкого згасання, або збудження синусоїдальної складової з частотою θ, яка знаходиться в сигналі $\boldsymbol{X}_i$.

Задача Піфагора полягає в знаходженні всіх прямокутних трикутників з цілочисельними сторонами, тобто у вирішенні діофантового рівняння:

$$x^2 + y^2 = z^2.$$

З цього рівняння і рівності Піфагора виходить, що для деяких кутів θ синуси і косинуси дорівнюють раціональним числам: $\sin\theta = x/z$, $\cos\theta = y/z$. Рішення задачі Піфагора знаходяться з відношень:

$$x = 2mn;\ y = m^2 - n^2;\ z = m^2 + n^2,$$

де m і n – взємно прості числа і m > n [4]. Таким чином, підбираючи m і n, можна знайти точні значення синуса і косинуса для деякого кута $\theta' \approx \theta$. Так, наприклад, $\cos(2\pi/128) = 1304/26585$ і $\sin(2\pi/128) = 26533/26585$ з похибкою кута 0,001˚. Для порівняння, якщо взяти аппроксимацію цих коефіцієнтів цілими 16-розрядними числами, то похибка довжини вектора складе 14.5·10-6, а похибка кута – 0,0002˚. Представлення коефіцієнтів у рівняннях (2) раціональними дробами, які є розв′язками задачі Піфагора, дає змогу реалізувати процесор ковзного перетворення Фур′є зі стабільними обчисленнями.

Для демодуляції DTMF-сигналів, які широко використовуються в телефонії, були одержані набори коеффіцієнтів, які представлені в таблиці. Ці коеффіцієнти підібрані для частоти дискретизації 8 кГц і 16-розрядного представлення даних. Параметр ε дорівнює відхиленню центральної частоти канала від номінальної.

Частота, Гц	N	x	y	z	ε %
697	241	16416	26937	31545	-0.019
770	187	7412	10725	13037	0.005
852	169	18696	23647	30145	-0.024
941	170	15620	17139	23189	0.0004
1209	225	20732	14835	25493	0.016
1336	102	22680	13039	26161	-0.025
1477	130	28712	12495	31313	0.026
1633	147	8400	2489	8761	0.014

Була розроблена бібліотека обчислювальних модулів для реалізації арифметики раціональних дробів, які описані на мові VHDL і призначені для конфігурування в ПЛІС. Для досягнення високої пропускної спроможності, моделі блоків множення і додавання виконані з двома і чотирма конвейерними ступенями, відповідно.

В сигнальному графі алгоритму ковзного рекурсивного ДПФ до дуг додано вузлів затримок (зафарбовані прямокутники), кількість яких відповідає конвейерним затримкам блоків множення і додавання (рис.1).

Загальна кількість затримок, які включені в усі циклічні маршрути графа на рис.1, означає, що цей сигнальний граф є результатом згортки 10 однакових сигнальних графів обчислення окремих спектральних каналів ДПФ [5]. Лінія затримки ЛЗ виконана з відводами для регулювання затримки на N відліків в залежності від номеру спектрального каналу (див. табл.). Таким чином, структура обчислювача, яка побудована за цим графом, виконує паралельні обчислення десяти каналів ДПФ в конвейерному режимі.

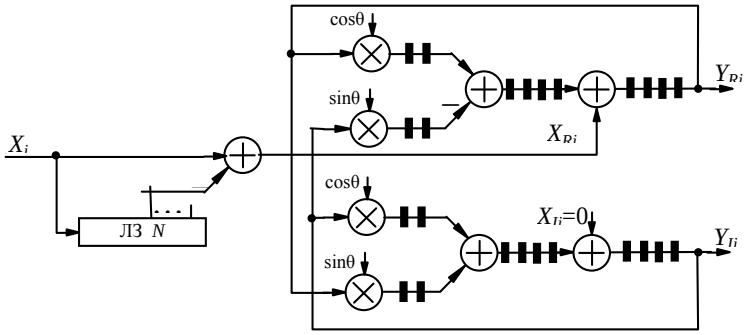


Рис.1. Сигнальний граф, що відповідає рекурсивному ДПФ

За сигнальним графом на рис.1 у відповідності з методикою, приведеною в [5], був розроблений проект процесора ковзного ДПФ, зконфігурованого в ПЛІС Xilinx xc4vsx35-12. Процесор виконує операції з 18-розрядними чисельниками і знаменниками. Він має апаратні витрати 627 еквівалентних конфігурованих логічних блоків (ЕКЛБ, CLB slices), 14 блоків множення DSP48 і тактову частоту до 200 МГц. Це означає, що процесор обробляє вхідний сигнал з максимальною частотою дискретизації 20 МГц.

При іспитах моделі процесора, що виконує ковзний рекурсивний ДПФ для N = 128 і $\theta = 2\pi/N$ з плаваючою комою з подвійною точністю оцінка амплітуди гармоніки вхідного сигналу мала похибку до 4-5%. При тих самих умовах процесор з арифметикою дробів виконував обчислення з похибкою 10-26%.

Якщо на вхід процесора подавати синусоідальний сигнал з частотою, що точно дорівнює θ, то через N тактів на виході суматора, який віднімає від вхідного сигнала затриманий сигнал, отдержується нульовий сигнал. При цьому схема повинна генерувати стабільний синусоідальний сигнал з частотою θ і амплітудою, що дорівнює амплітуді вхідного сигналу. При цих умовах процесор генерував такий сигнал протягом кількох тисяч тактів дискретизації, а в аналогічному процесорі з цілочисельною арифметикою вихідний сигнал згасав до нуля. Це свідчить про досить високу стабільність процесора, в якому демодуляція БЧМ виконується запропонованим способом.

Було розроблено фільтр з частотною виборкою [1], який складається з рекурсивного процесора ДПФ, до виходу якого приєднано блок зваженого додавання і перетворення дробу в цілий результат. Сумарні апаратні витрати складають 1230 ЕКЛБ і 16 DSP48. В порівнянні з аналогічним фільтром на базі процесора ШПФ, даний фільтр при приблизно однакових апаратних витратах має втричі більшу частоту дискретизації вхідного сигналу.

Таким чином, впровадження в декодер БЧМ арифметики раціональних дробів дало змогу виконувати ковзне рекурсивне ДПФ високостабільним з точністю, що наближається до точності арифметики з плаваючою комою. Це також дало змогу одержати малі апаратурні витрати і високу тактову частоту процесора. На відміну від алгоритму Герцеля, ковзний ДПФ дає змогу безперервно слідкувати за зміною гармонік вхідного сигналу і одержати більш точну оцінку їхніх амплітуд і фаз.

Запропонований алгоритм рекурсивного ДПФ також може бути впроваджений в сигнальних мікропроцесорах з 16-розрядним представленням даних, завдяки чому оцінка амплітуд складових сигналу стає стабільною і для цього не потрібне використання плаваючої коми або чисел з подвійною розрядністю.

Перелік посилань

1. Оппенгейм А.В., Шафер Р.В. Цифровая обработка сигналов. – М.: Связь. –1979. – 416 с.
2. Chen *C. J.* Modified Goertzel Algorithm in DTMF Detection Using the TMS320C80 //Texas Instruments Inc. Application Report. SPRA066. –USA. –June, 1996. –13 p.
3. Клименко О.М., Сергієнко А.М., Шевченко Ю.В., Овраменко С.Г. Конфігурована обчислювальна система для вирішення задач лінійної алгебри // Электронное моделирование.–2005.–Т.27.-№1.–С.109-114.
4. Оре О. Приглашение в теорию чисел. – М.: Наука. –Биб-ка Квант, вып.3. – 1980. –126с.
5. Сергиенко А.М. VHDL для пректирования вычислительных устройств. – Киев: ДиаСофт. – 2003. – 203 с.

УДК 519.854.2

ПАВЛОВ О.А.,
МІСЮРА О.Б.,
ХАЛУС О.А.,
КОСТИК Д.Ю.,
ЛИСЕЦЬКИЙ Т.М.

ОПТИМІЗАЦІЯ МОДИФІКОВАНОГО ПДС-АЛГОРИТМУ ЗАДАЧІ МІНІМІЗАЦІЇ СУМАРНОГО ЗАПІЗНЕННЯ ВИКОНАННЯ ЗАВДАНЬ

Наводиться блок-схема алгоритму розв'язання задачі мінімізації сумарного запізнення, нові правила відсікань та декомпозиції алгоритму для зменшення сумарного часу обчислення, статистика розв'язання задач.

The logic diagram for algorithm of the total tardiness minimization, new truncation and decomposition rules for the algorithm to decrease the total calculation time, and the statistics of the tasks solution are given.

Вступ

У теорії розкладів (ТР) особливе значення мають задачі з одним приладом. Результати, отримані при дослідженні таких задач, можуть бути використані для побудови алгоритмів вирішення складніших задач з багатьма приладами і багатостадійних задач, що виникають на практиці. Класичні задачі ТР з одним приладом є схематичними теоретичними моделями багатьох задач, що зустрічаються на практиці. Дослідження таких задач допомагає вивчити фундаментальні властивості і структуру практичних задач, що сприяє побудові ефективних алгоритмів їх рішення.

«Мінімізація сумарного запізнення виконання незалежних завдань з директивними строками одним приладом» (МСЗ)

Припустимо, що задано множину незалежних завдань $J = \{j_1, j_2, \ldots, j_n\}$, кожне з яких складається з однієї операції. Для кожного завдання відома тривалість виконання l_j і директивний строк виконання D_j. Завдання надходять у систему одночасно в момент $d_j = 0, j = \overline{1, n}$. Переривання не допускаються. Необхідно побудувати розклад виконання завдань для одного приладу, що мінімізує сумарне запізнення при виконанні завдань:

$$f = \sum_{j=1}^{n} \max\left(0, C_j - D_j\right),$$

Де C_j – момент завершення виконання завдання j.

Огляд відомих методів розв'язання цієї задачі приведений у [1]. Згідно Ду і Люнгу, задача є NP-складною.

На кафедрі АСОІУ було запропоновано ефективний точний ПДС-алгоритм (алгоритм із поліноміальною й експоненційною складовими) розв'язання задачі, заснований на новому підході до розв'язання задач з директивними строками, що полягає в оптимальному використанні резервів часу незапізнених завдань, що дозволяє розв'язувати задачі з числом завдань, істотно більшим, ніж 500.

Блок-схема алгоритму

Даний алгоритм складається з двох етапів – попереднього етапу і етапу оптимізації. Спочатку виконується попередній етап, який включає сортування всіх завдань за тривалістю виконання та так звані вільні перестановки. Ціль вільних перестановок перемістити кожне завдання на максимально можливу позицію, на якій воно не буде запізнюватися, якщо на інтервалі перестановки є завдання, що запізнюються. Вихідна послідовність є вхідною для наступного оптимізаційного етапу алгоритму. На цьому етапі спочатку визначається множина конкуруючих завдань (блок 1). Далі для кожного наступного завдання із цієї множини виконується оптимізація підпослідовності з початку до поточного завдання. Оптимізація починається із пошуку позиції p вбудовування завдання, на якій воно не буде запізнюватися (блок 3). Якщо на позиції p знаходиться завдання, яке не запізнюється (блок 10), то виконується вбудовування поточного завдання на позицію p (блок 11) і перехід на блок 20. В іншому випадку аналізуються резерви часу завдань, що передують в поточній послідовності завданню, що

розглядається (блок 12). У випадку, якщо резерви є, завдання вбудовується на нову позицію p^H (блок 16) і виконується оптимізація за рахунок резервів на інтервалі [1, p^H]. Якщо завдання зайняло більш ранню позицію, тоді відбувається перехід на блок 20 (перехід А). Якщо резерви відсутні, то корегується позиція вбудовування (блок 15), і якщо позиція вбудовування менше g, то відбуваєть відбувається перехід до блока 20. Інакше на блок 22 (перехід В). У блоці 20 завдання впорядковуються за неспаданням їх тривалостей. Завдання $j^H_{[p]}$ помічається *.

Якщо воно не запізнюється, то відбувається перехід до наступного конкуруючого завдання (блок 2). Інакше знаходиться завдання, помічене * або ** із більшою тривалістю (блок 22) і запам'ятовується поточна послідовність (блок 24). Далі знайдене завдання (* або **) вбудовується після нової позиції p^H (блок 25) і виконується оптимізація підпослідовності за вище наведеними правилами (блок 26).

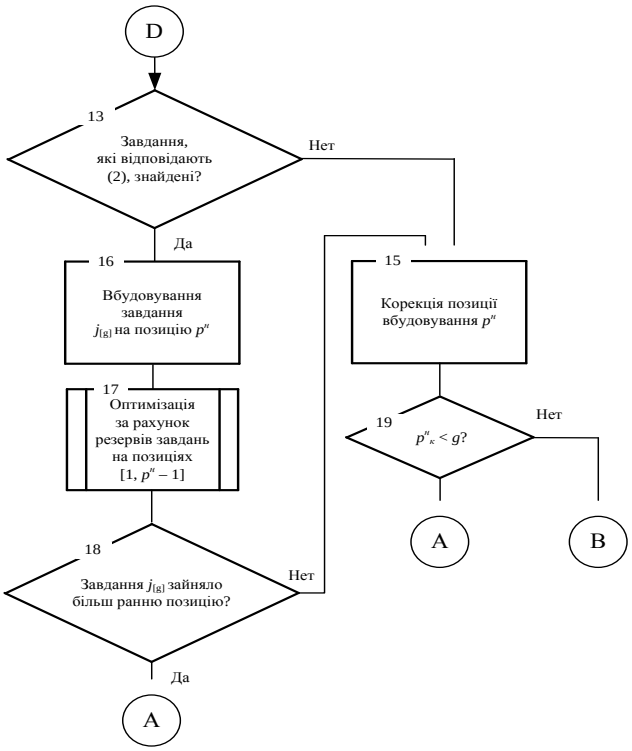
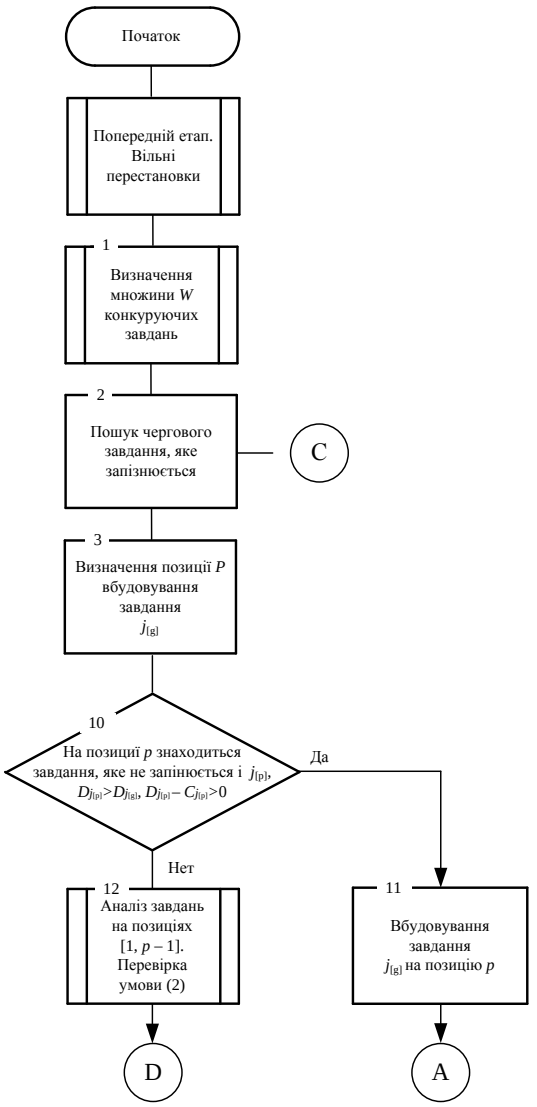


Рис. 1 – Блок-схема алгоритму (ч. 1)



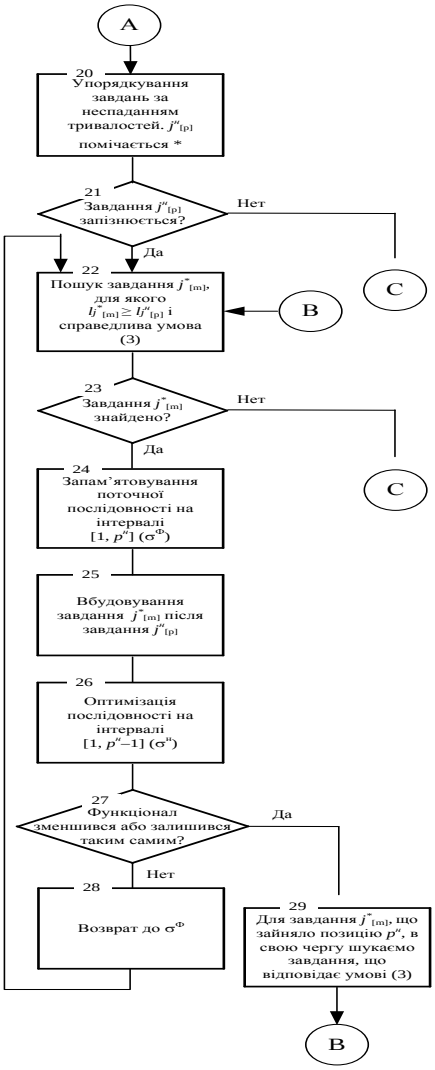


Рис. 2 – Блок-схема алгоритму (ч. 2)

Якщо функціонал збільшився (блок 27), то повертається попередня послідовність (блок 28), перехід до наступної * або **. Якщо таких завдань не знайдено, то перехід до наступного конкуруючого завдання. Якщо ж функціонал зменшився (у блоці 27), то послідовність зберігається, і відбувається пошук * або ** для завдання яке було вбудоване (блок 29).

Нові відсікання

Для покращення загальної швидкодії алгоритму [2] було проведено ряд досліджень за методикою [4]. За результатами цих ДОСліджень було виявлено ряд нових закономірностей та розроблені нові відсікання для існуючого алгоритму, які базуються на наступних твердженнях.

Твердження 1. Якщо конкуруюче завдання $j_{[g]}$ після виконання для нього ітерацій оптимізації зайняло ту ж саму позицію g в поточній послідовності σ^k, то завдання $j_{[i]}$, що слідують за завданням $j_{[g]}$, для яких виконується: $Dj_{[i]} \geq Dj_{[g]}$, $Lj_{[i]} \geq Lj_{[g]}$, в оптимальному розкладі не можуть займати позицію меншу, ніж g. В цьому випадку здійснюється декомпозиція послідовності σ^k на підпослідовності меншого розміру, і якщо позиція вбудовування такого завдання $j_{[i]}$ дорівнює p+1, то для цього завдання не виконуються блоки 12, 13, 17, 18.

Твердження 2. Нехай в послідовності σ, що отримана після виконання вільних перестановок, для всіх завдань на інтервалі [k, n] виконується: $Cj_{[i]} - Dj_{[i]} \geq 0$, i = $\overline{1, n}$. Позначимо цю послідовність $\sigma^{ЗАП}$. Якщо після виконання ітерації оптимізації для завдання $j_{[k]} \in \sigma^{ЗАП}$ це завдання зайняло ту ж саму позицію k, то завдання $j_{[i]}$, $i > k$, для яких виконується $Dj_{[i]} \geq Dj_{[k]}$, вилучаються з множини конкуруючих за резерви попередніх завдань і займають в оптимальному розкладі позиції $r \geq i$.

На даному етапі додано ще два алгоритми для оптимізації алгоритму. Перше відсікання полягає у декомпозиції поточній послідовності. При виконанні умов твердження 1 ряд блоків алгоритму не виконуються, що суттєво скорочує час розв'язання задачі. Друге відсікання полягає у маркуванні завдань, які задовольняють умовам твердження 2, позначками «прапорець» та виключенням їх з ряду конкуруючих завдань. При загальній схемі розв'язання алгоритму, наведеному у [3, 4], всі завдання, які промарковані «прапорцем», не розглядаються як конкуруючи, та вважаються на «своїй» позиції. Вони використовуються лише для спускання «*» або «**». Таким чином, ми отримуємо відсікання дуже великої кількості конкуруючих завдань, що сприяє покращенню швидкодії.

Визначення ефективності нових відсікань

За допомогою системи моделювання для дослідження ефективності ПДС-алгоритму (алгоритму із поліноміальною й експоненційною складовими) [2] було проведено ряд тестів, які показали, що додавання нових відсікань сприяє зменшенню сумарного часу роботи алгоритму у кілька разів, а у деяких класів задач у 70-80 разів. У таблиці 1 приведені деякі результати попереднього тестування версій програмного забезпечення до та після додавання відсікань.

Таблиця 1. Середній час розв'язання задач (сек.)

Розмірність	Середній час розв'язання без додаткових відсікань	Середній час розв'язання з додатковими відсіканнями
50	0,12	0,07
100	2,1	0,5
150	9,4	2,7
300	42,2	12,1
400	114,7	32,8
500	311,7	89,0
1000	1009,8	348,2

Таблиця 2. Порівняння часу розв'язання з відомими методами

№ прикладу	Сумарне запізнення	Час розв'язання відомими методами (мс)	Час розв'язання ПДС-алгоритмом (мс)
46	83	250	52
50	56	150	50
131	178	267	20
134	119	250	58
135	1169	1833	313
136	299	133	80
138	204	250	49
139	69	150	62

Загальна ефективність алгоритму

У таблиці 2 приводиться час розв'язання відомих прикладів [6] розмірністю 500 завдань. Всі випробування проводились на комп'ютері з наступними параметрами: CPU Pentium CORE 2 Duo 1.9ГГц; RAM 2048Мб; OS Microsoft Windows Vista.

Висновок

Розроблено нові правила відсікань до алгоритму, зроблено оптимізацію програмної реалізації, перебудова деяких блоків алгоритму для зменшення сумарного часу обчислення. Створена програмна реалізація функції декомпозиції на задачах великої розмірності, що дає змогу у майбутньому використовувати її для розгалужування алгоритму на багатопроцесорні системи. Планується зробити ще ряд відсікань та розбиття вхідних задач на підкласи, що дозволить запускати ту чи іншу гілку алгоритму, більш пристосовану для даної задачі. Для підвищення ефективності розв'язання задачі на багатопроцесорних системах у майбутньому планується розпаралелювати декомпозиційну складову алгоритму.

Перелік посилань

1. Павлов А.А., Теленик С.Ф. Информационные технологии и алгоритмизация в управлении.– К.: Техника.– 2002.– 344 с., ISBN 966-575-045-3.
2. Система моделювання для дослідження ефективності ПДС-алгоритму задачі мінімізації сумарного запізнення виконання завдань // Павлов А.А., Місюра О.Б., Халус О.А, Беньковський С.Б., Лисецький Т.М., Костик Д.Ю. / Вісник НТУУ "КПІ". Інформатика, управління та обчислювальна техніка. К.: "ВЕК+", 2007.– №47.– С.215-220
3. Павлов О.А., Місюра О.Б., Халус О.А. Модифікований ефективний ПДС-алгоритм рішення задачі мінімізації сумарного запізнення виконання незалежних завдань одним приладом / Радиоэлектроника и информатика.– Харьков: ХНУР, 2007.– №1(36).– С.21-23
4. Павлов А.А., Мисюра Е.Б. Эффективный точный ПДС-алгоритм решения задачи о суммарном запаздывании для одного прибора // Системні дослідження та інформаційні технології. – 2004.– №4.– С.30-59
5. Шейко В., Кушнаренко Н. Організація та методика науково-дослідницької діяльності: Підручник. – К.: Знання-Прес, 2002. – 295 с.
6. Fisher, M.L. (1976) A dual algorithm for the one machine scheduling problem. Mathematical Programming, 11, 229-251. Тестові задачі доступні в Інтернет на сторінці http://www.bilkent.edu.tr/~bkara.

УДК 004.8.

ЗАЙЧЕНКО Ю.П.,
МЕДІН М.Ю.

МЕТОД ВИРІШЕННЯ ЗАДАЧІ АНАЛІЗУ СИТУАЦІЙ СИСТЕМОЮ ПІДТРИМКИ ПРИЙНЯТТЯ РІШЕННЯ ДЛЯ ПРОГНОЗУВАННЯ

Зазначені основні проблеми пов'язані зі створенням системи ситуаційного аналізу системи підтримки прийняття рішення для прогнозування. Розроблено метод вирішення задачі ситуаційного аналізу із застосуванням нечіткої системи класифікації NefClass. Такий метод враховує не тільки досвід співтовариства систем підтримки прийняття рішень, але й рівень довіри до її учасників. Наведено експериментальні результати роботи при вирішенні задачі автоматичної видачі порад особі що приймає рішення щодо того, яким методом прогнозування краще скористатись.

The basic issues were singled out when creating a situation analysis system for the decision support system for forecasting. The method for solving situation analysis task was developed using neuro fuzzy classifier NefClass. Such a method takes into account both collective knowledge and levels of confidence. Experiment results were shown providing computer aided support for the person who makes decisions.

1. Вступ

Дослідження соціально економічних процесів являє собою актуальне завдання і дозволяє одержувати нову інформацію про процеси, що відбуваються в самій економіці та досліджувати соціально економічні зв'язки.

Надійність роботи систем підтримки прийняття рішення (СППР) досягається за рахунок використання декількох методів. Але дуже часто ефективність роботи цих методів залежить від вхідних даних і рішення про те, який алгоритм кращий, приймає людина. Тому система підтримки прийняття рішення повинна включати підсистему, яка за інформацією про процес може дати пораду особі що приймає рішення (ОПР), яким методом найкраще скористатись. При цьому важливо, щоб прийняття рішення відбувалося за зовнішнім критерієм і з використанням досвіду експертів. Для вирішення цієї задачі можна скористатись системою класифікації NefClass. Так, у роботі [1] запропонований підхід щодо створення такої СППР для прогнозування.

Специфікою інтелектуальних систем підтримки прийняття рішення (СППР) здатних до навчання є необхідність створення навчальної вибірки. Тому, на даному етапі впровадження такої системи матиме певні труднощі. Обмеженість апріорних знань про сферу дослідження не дозволяє швидко отримати бажаний рівень якості її навчання. Носієм більш повного знання іноді може бути інформаційне суспільство інтелектуальних систем, здатних кооперуватися з метою досягнення певної мети. Ось чому, сучасне покоління систем підтримки прийняття рішень повинно бути «соціально орієнтованим», тобто взаємодіяти з оточуючими об'єктами. На наш погляд, реалізація механізму врахування досвіду співтовариства систем підтримки прийняття рішень є ключовим питанням при підвищенні їх продуктивності.

Проблема полягає в тому, що в процесі роботи співтовариство СППР може видавати суперечливі результати. Тому необхідно створити такий механізм прийняття рішення, який би враховував рівень довіри до кожного із учасників співтовариства.

Метою даної роботи є підвищення продуктивності системи підтримки прийняття рішення для прогнозування при вирішенні задачі ситуаційного аналізу. Для цього пропонується метод, що використовує нечітку нейрону мережу NefClass. В результаті така система повина враховувати не тільки досвід співтовариства СППР, але й рівень довіри до її учасників.

2. Загальна схема метода

Загальна схема метода вирішення задачі аналізу ситуацій системою підтримки прийняття рішень показана на рисунку 1. На вхід модуля СППР видачі порад, який реалізує запропонований метод, подається спостережуваний відрізок часового ряду. Результатом його роботи є рішення системи класифікації про те, яким методом прогнозування краще скористатись.

Особливістю запропонованого методу є врахування рішень інших систем підтримки

прийняття рішення, що аналізують ситуацію виходячи із власного досвіду. СППР, які вирішують подібні задачі та реалізують функцію обміну результатами роботи в автоматичному режимі, можуть бути об'єднані у співтовариства.

Так, в результаті застосування фільтрів дискретного вейвлет перетворення (ДВП) до відрізку часового ряду, формується характеристичний вектор. Принцип роботи ДВП фільрів та процес формування характеристичного вектора докладно описано у роботі [1]. Нагадаємо, що у якості компонентів вектора використовують частки енергій, які зосереджені у вейвлет коефіцієнтах певного масштабу.

Сформований характеристичний вектор подається на вхід систем класифікації інших СППР. Результати співтовариства СППР обробляються системою врахування колективного досвіду при прийнятті рішення.

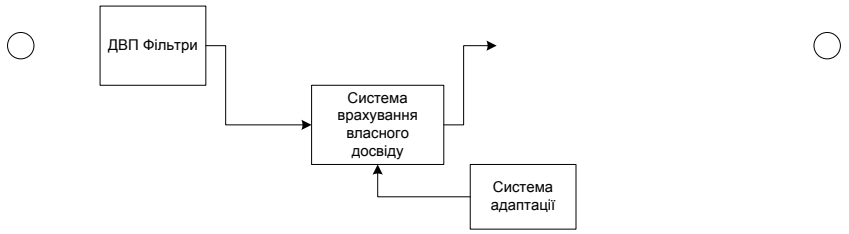


Рис.1. Загальна схема метода

Одночано система продукує рішення, що базується на власному досвіді роботи.

Далі обирається остаточне рішення, що є зваженою сумою власного та колективного рішень.

Проблема зазначеного підходу полягає в тому, що системи для забезпечення розрахунків прогнозування можуть характеризуються високою індивідуалізацією. І кожна із СППР керується власним досвідом при виборі методу прогнозування. Прийняті ними рішення можуть бути суперечливими. Тому необхідно, щоб система врахування колективного досвіду могла обрати рішення із врахуванням рівня довіри до учасників співтовариства.

Для вирішення цієї задачі пропонується використати систему класифікації NefClass, адже вона має наступні властивості:

- при навчанні системи Використовується метод спроб та коригування помилки;
- система може бути представлена у вигляді нейронної мережі прямого поширення, однак це не накладає ніякого додаткового змісту на те, як відбувається процес навчання, а служить лише зручною формою подання архітектури;
- нечітка система може бути легко інтерпретована у вигляді набору предикативних правил;
- нечітка система враховує семантику нечіткого лінгвістичного значення змінної;
- нечітка система виконує апроксимуючу функцію, не тільки функції приналежності нечіткого числа (у вузькому змісті), але й з погляду узагальнення логічних правил.

Зазначені властивості системи NefClass дають також змогу реалізувати механізм адаптації власної бази правил, а також врахувати рівень довіри до кожного з учасників співтовариства СППР.

Нехай кількість альтернатив методів прогнозування буде n, а кількість СППР учасників співтовариства – m. Результати рішень співтовариства СППР можуть бути представлені у вигляді набору векторів $\{a_{11},\ldots,a_{1n}\},\ldots,\{a_{m1},\ldots,a_{mn}\}$, де a_{ij} – рішення i-го учасника відносно j-ої альтернативи. Аналізуючи ці вхідні дані, система врахування колективного досвіду обирає рішення використовуючи відповідний рівень довіри до учасників співтовариства. Звичайно цей вибір залежить від попереднього досвіду, набутого в результаті співпраці, або інакше навчання.

3. Сценарій навчання

Запропонована схема роботи СППР може бути розбита на три етапи.

Сценарій роботи

Етап 1 Навчання системи врахування колективного досвіду

Метою даного етапу є налаштування системи NefClass врахування колективної думки учасників коаліції. Схема роботи систем показана на рисунку 2.

1. Відрізок часового ряду подається на вхід фільтра дискретного вейвлет перетворення.
2. Характеристичні ознаки передаються іншим системам класифікації.
3. На вхід системи вибору рішення подаються результати роботи віддалених СППР. Відбувається формування бази правил та настройка параметрів системи NefClass щодо компетенції учасників коаліції.
4. Оскільки на даному етапі довірчий рівень не сформовано, результатом роботи вважається думка більшості
5. Для кожного вхідного відрізку часового ряду повторити пункти 1-4 доки не буде сформована система NefClass.

Рис. 2. Етап 1

Етап 2 Навчання системи врахування власного досвіду

Метою даного етапу роботи є формування бази нечітких правил NefClass системи врахування власного досвіду. При цьому учасники співтовариства будуть виступати в ролі експертів. Схема роботи даного етапу зображено на рисунку 3. Етапи 1 та 2 можуть виконуватися паралельно.

1. Відрізок часового ряду подається на вхід фільтра дискретного вейвлет перетворення.
2. Виділені характеристичні ознаки подаються на вхід NefClass системи врахування власного досвіду. Виконуться етап формування бази правил NefClass.
3. Характеристичні ознаки передаються іншим системам класифікації.
4. На вхід системи вибору рішення подаються результати роботи віддалених СППР. Оскільки на даному етапі довірчий рівень не сформовано, експертною думкою вважається рішення більшості.
5. Для кожного вхідного відрізку часового ряду повторити пункти 1-4 доки не буде сформована система NefClass.

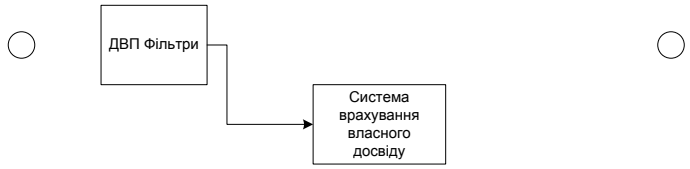


Рис. 3. Етап 2

Етап 3 Робота та адаптація

Метою даного етапу є використання систем врахування колективного досвіду та системи врахування власного досвіду, а також адаптування параметрів цих систем.

1. Відрізок часового ряду подається на вхід фільтра дискретного вейвлет перетворення.
2. Виділені характеристичні ознаки подаються на вхід нечіткої системи NefClass вибору метода прогнозування.
3. Характеристичні ознаки передаються іншим системам класифікації.
4. На вхід системи вибору рішення подаються результати роботи віддалених СППР. Її результатом буде зважена сума рішень систем врахування власного та колективного досвіду.
5. Система аналіза результатів визначає помилку класифікаціх і формує керуючі сигнали для систем врахування колективного та власного досвіду. Адапатація відбувається по параметрам, а у разі необхідності – по правилам.

Процес адаптації параметрів системи ситуаційного аналізу як і процес її навчання представляє двокроковий процес. Перший етап включає настроювання параметрів лінгвістичних змінних нечітких правил. На другому етапі відбувається переформування структури нечіткого класифікатора.

4. Експерименти

Метою експерименту була перевірка ефективності запропонованого методу вирішення задачі ситуаційного аналізу системою підтримки прийняття рішення для прогнозування. Необхідно було порівняти ефективність роботи СППР у автономному режимі та у розширеному варіанті із застосуванням запропонованого методу. Крім того, необхідно було показати стійкість роботи методу до суперечливих результатів співтовариства інтелектуальних систем за рахунок врахування рівня довіри.

Для експерименту у якості прогнозуємого процесу використовувалися дані по процентним ставкам федеральних фондів США (Федеральної Резервної Системи) за період 1950-2000 рр.

До арсеналу засобів прогнозування кожної СППР входили нейро мережеві алгоритми з прямим розповсюдженням сигналу. Необхідно зауважити, що метою даного експерименту не було дослідження методів прогнозування, їх порівняння та ін.

Співтовариство СППР для прогнозування складалося з трьох учасників.

Спостереження часового ряду були розбиті на відрізки однакової довжини і змішані таким чином, що частина з них була використана для настроювання алгоритмів прогнозування і для настроювання систем класифікації віддалених СППР. Інша частина відрізків часового ряду використовувалася для тестування роботи СППР у автономному режимі та розширеному варіанті.

Було проведено 5 тестів СППР на вихідних даних. Кожного разу вибірки для навчання та тестування змішувалися.

На рисунку 4 показані середні значення абсолютних похибок класифікації системи ситуаційного аналізу за участю коаліції інтелектуальних систем (графік 1) та у автономному режимі (графік 2). Видно, що у автономному режимі СППР знадобилося зробити більше кроків для формування системи врахування власного досвіду (точці В відповідає 18 кроків). У розширеному варіанті роботі система працювала з першого кроку, використовуючи коалційний досвід (точка А, 15 кроків).

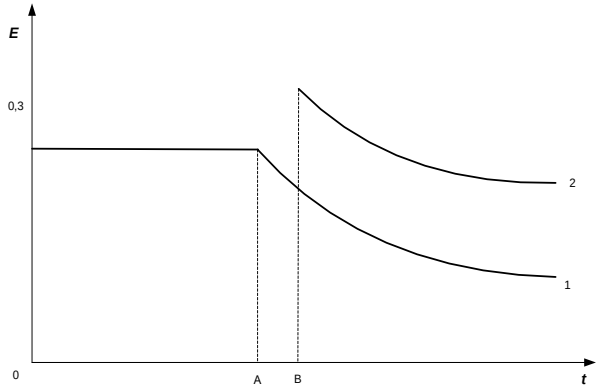


Рис. 4. Середні значення абсолютних похибок класифікації систем ситуаційного аналізу

Таблиця 1. Середні значення абсолютних похибок класифікації

Автономна робота		Робота за участю коаліції	
Навчання	Тест	Навчання	Тест
–	0,28	0,31	0,16
–	0,21	0,25	0,14
–	0,23	0,26	0,15
–	0,18	0,25	0,15
–	0,31	0,24	0,16

Результати 5-х повторних тестів, наведених у таблиці №1, дають підстави виділити наступні переваги запропонованого підходу:

1. Можливість використання СППР на ранньому етапі без попереднього її навчання.

2. Ефективність роботи системи ситуаційного аналізу вище, адже в середньому точність класифікації вища.

З результатів, наведених у таблиці №1, видно, що запропонований метод дає порівняно стабільну точність роботи системи класифікації. Як зазначалося вище, вибірки що для навчання та тестування змішувалися для кожного нового прогону СППР. Це означає, що якість навчання учасників співтовариства СППР змінювалася. Звідси можна говорити, що запропонований метод є стійким відносно суперечливих результатів співтовариства інтелектуальних систем саме за рахунок врахування рівня довіри.

5. Закінчення

В роботі було запропоновано метод вирішення задачі ситуаційного аналізу системою підтримки прийняття рішення для прогнозу-

вання із застосуванням нечіткої системи класифікації NefClass. В результаті така система враховує не тільки досвід співтовариства інтелектуальних систем, але й рівень довіри до її учасників.

Отримані експериментальні результати дослідження ефективності запропонованого методу дали змогу виділити переваги запропонованого підходу:

1. Можливість використання СППР на ранньому етапі без попереднього її навчання.
2. Ефективність роботи системи ситуаційного аналізу вище за умови колективної роботи інтелектуальних систем.
3. Запропонований метод є стійким відносно суперечливих результатів співтовариства інтелектуальних систем саме за рахунок врахування рівня довіри.

Це дає підстави зробити висновок, що представлений у роботі підхід може підвищити продуктивність роботи СППР для прогнозування вцілому.

Перелік посилань

1. Зайченко Ю.П., Медін М.Ю. Дослідження можливості застосування вейвлет перетворення для виділення інформативних характеристик часових рядів економічних процесів // Вісник Національного технічного університету України «КПІ» Інформатика, управління та обчислювальна техніка. К.: ТОО «ВЕК+». – №46. – 2007. – С.141-150.
2. Marc Holmes, Simon Cox. Delivering End-to-End High Productivity Computing // The Architecture Journal. – №11. – 2007. – p.9-17.
3. Berman, Fran; Anthony J. G. Hey, Geoffrey C. Fox. Grid Computing: Making The Global Infrastructure a Reality. USA – Wiley, 2003.
4. Foster, Ian; Carl Kesselman. The Grid: Blueprint for a New Computing Infrastructure. UK – Morgan Kaufmann Publishers, 2004.

УДК 62-82:658.512.011.56

БЕЗНОСИК А.Ю.

ПОСТРОЕНИЕ ЭКВИВАЛЕНТНЫХ МОДЕЛЕЙ БАЛОЧНЫХ СТРУКТУР

Предлагается использование конечно-элементной математической модели для получения эквивалентной электрической схемы замещения механических компонентов. Для уменьшения размерности полученных схем предложено использование RLC сокращения. Рассмотрен пример получения эквивалентной электрической схемы балочной структуры. Данный подход распространен на получение эквивалентных схем для МЭМС компонентов.

The usage of a finite-element mathematical model to obtain an equivalent electrical circuit for mechanical components is suggested. Using of RLC reduction algorithm to reduce a dimension of the circuits obtained is proposed. An example of getting equivalent electrical circuit for beam structure is considered. The approach in question is applied to obtaining equivalent circuits for MEMS components.

Введение

При проектировании современных МЭМС-устройств важную роль играет возможность использования единого инструментария для моделирования объектов, в которых протекают различные физические процессы: электрические, механические, оптические, тепловые и т.д. Это требует представления различных подсистем исходного объекта в виде эквивалентных моделей одной и той же физической природы, что позволит объединить их для решения в едином вычислительном процессе. Однако наиболее распространенные САПР, в основном, используют одноаспектное моделирование, например, Spice для анализа электронных схем или Adams для виртуального моделирования сложных машин и механизмов. Для расчета «нетрадиционных» для данных систем физических процессов используют несколько подходов: учет физических особенностей в виде дополнительных параметров базовых моделей; включение возможности расчета в дополнительных базисах, характерных для иных физических процессов и т.д.

На этапе схемотехнического моделирования для формирования математической модели объекта сложной физической природы наиболее часто применяется метод электромеханических аналогий. Это, в свою очередь, требует наличия схемных реализаций моделей неэлектрических блоков, обладающих необходимой степенью точности. Большинство современных средств САПР, применяемых при проектировании механических систем, для построения математической модели используют метод конечных элементов. Основной проблемой использования таких моделей является размерность получаемых систем, которые могут описываться десятками тысяч уравнений. Поэтому основные усилия разработчиков направлены на получение сокращенных моделей, пригодных для дальнейшего использования.

Таким образом, актуальной является разработка алгоритма получения моделей неэлектрических объектов в виде эквивалентных электрических схем на основе готового инструментария построения математической модели для применения в пакетах схемотехнического проектирования.

Описание примера

Рассмотрим метод построения схемных моделей механических компонентов и их дальнейшее использование на примере задачи нахождения собственных частот балки (рис. 1).

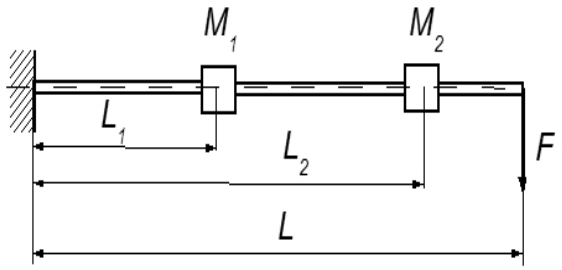


Рис. 1. Схема закрепленной неоднородной балки с тремя степенями свободы

Рассматриваемая конструкция представляет собой балку, работающую на изгиб. Левый конец балки закреплен неподвижно, правый — свободен. К правому концу приложена сила F, перпендикулярная оси балки. В двух точках

к балке прикреплены сосредоточенные массы M_1 и M_2, находящиеся на расстояниях L_1 и L_2 от точки закрепления.

Для построения математической модели был использован пакет ANSYS Multiphysics v.10.0, разработанный компанией ANSYS Inc. [1]. Моделирование в ANSYS выполнялось при следующих параметрах: L = 25 мм; L_1 = 10 мм; L_2 = 20 мм; поперечное сечение балки — прямоугольное: высота 3 мм, ширина 2 мм. Характеристики материала балки: модуль упругости E=2·10^{11} Па; коэффициент Пуассона μ=0,3; плотность материала ρ=6·10^3 кг/м^3. Массы грузов: m_1=m_2=36 г.

Для дискретизации балки были использованы конечные элементы типа BEAM3 из библиотеки ANSYS. Каждый балочный элемент имеет два узла (левый/правый) с 3 степенями свободы в каждом: координаты x, y в плоскости движения и угол θ поворота соответствующего поперечного сечения балки. Все элементы имеют одинаковую длину по 0,5 мм. Соответственно, расчетная схема содержит 50 элементов и 51 узел.

Эквивалентная схема замещения

Для данного механического компонента, из файлов проекта ANSYS (FULL и EMAT) с помощью программы mor4ansys [2] были получены матрицы, описывающие состояние системы, в формате MarketMatrix. Матрицы представляют собой коэффициенты уравнения

$$M\ddot{x} + D\dot{x} + Kx = BF,\ y = Cx$$

где M, D и K – матрицы массы, демпфирования и жесткости соответственно; B – матрица входов; C – матрица выходов; x – вектор неизвестных, включающий все степени свободы; F – внешний вектор силы и y – вектор выходных переменных.

Переходя к электромеханическим аналогиям [3], получим:

$$\frac{d}{dt}(Mv) + Dv + \int Kvdt = F(t)$$

или

$$\tilde{C}\dot{v} + \tilde{G}v + \tilde{L}v = F(t),$$

где $\tilde{C} = M$, $\tilde{G} = D$, $\tilde{L} = K$ – равноценные матрицы емкостей, проводимостей и индуктивностей. Элементы матриц C, G, L формируется из элементов матриц M, D и K по формулам:

$$C_{ij} = -m_{ij},\quad i, j = 1(1)N,\quad i \neq j$$

$$C_{ii} = \sum_{j=1}^{N} m_{ij},\quad i = 1(1)N.$$

$$L_{ij} = -1/k_{ij},\quad i, j = 1(1)N,\quad i \neq j$$

$$L_{ii} = 1/\sum_{j=1}^{N} k_{ij},\quad i = 1(1)N.$$

$$G_{ij} = -d_{ij},\quad i, j = 1(1)N,\quad i \neq j$$

$$G_{ii} = \sum_{j=1}^{N} d_{ij},\quad i = 1(1)N.$$

где N – количество уравнений для данной системы. Используя полученные матрицы C, G, L, можно перейти к эквивалентной схеме замещения. В отдельных случаях, некоторые из этих матриц могут отсутствовать. Это будет означать, что в схеме будут отсутствовать соответствующие элементы. Для рассматриваемого примера, эквивалентная схема замещения представлена на рис. 2.

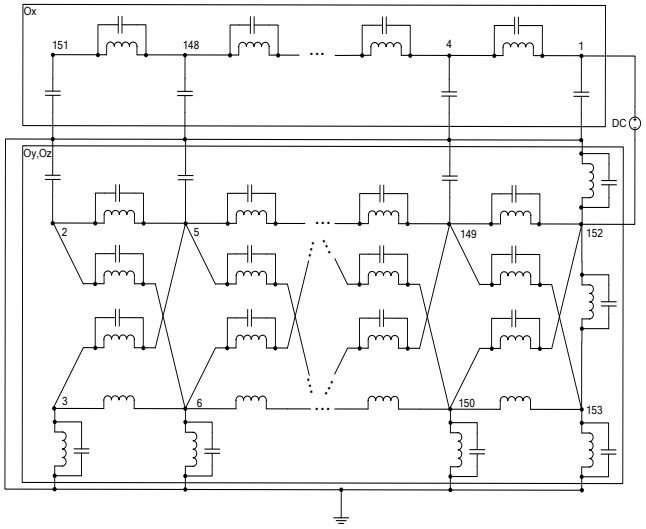


Рис.2. Эквивалентная схема замещения балки

В данной схеме подсхемы «Ox» и «Oy,Oz» связаны только по нулевому узлу схемы. Очевидно, что при подключении одного источника к узлу приложения силы для механического эквивалента, будут наблюдаться только частоты одной из подсхем. Так, для представленного примера при подключении источника между узлами 151, 0 будут наблюдаться частоты, которые в механической модели отвечают собственным частотам на растяжение. В случае источника, подключенного между узлами 152, 0 выходная характеристика будет содержать только частоты, соответствующие собственным частотам изгиба. С точки зрения ме-

ханики, такая ситуация представляет собой два независимых направления, по которым раскладывается начальный вектор силы. Для эквивалентной схемы это означает включения источника между обеими подсхемами, либо включение двух источников для возбуждения соответствующих подсхем.

Сокращение размерности схемы

Для объектов со сложной геометрией и большим количеством степеней свободы конечно-элементная модель может содержать сотни тысяч уравнений. В этом случае эквивалентная схема замещения может содержать миллионы и более элементов. Это обстоятельство делает невозможными использование данной схемы в качестве модели объекта, поэтому целесообразно воспользоваться алгоритмами сокращения размерности схемы на основе Y-Δ преобразования. Суть методов, основанных на Y-Δ преобразовании, изложена в [4,5,6].

Для каждого узла в цепи определены две временные постоянные: $\tau_{RCi} = C_i / G_i$ и $\tau_{LCi} = \sqrt{C_i / B_i}$, где $C_i = \sum_{j=1}^{k} c_j$ – сумма всех емкостей, $B_i = \sum_{j=1}^{k} b_j$ – сумма всех реактивных проводимостей (величина, обратная индуктивности), $G_i = \sum_{j=1}^{k} g_j$ – сумма всех проводимостей, подключенных к узлу i.

Узловая постоянная времени узла *i* определяется как:

$\tau_i = \max(\tau_{RCi}, \tau_{LCi})$.

i -ый узел считается быстрым, если:

$\tau_i < \tau_{\min} = 2\pi / \omega_{\max}$,

где $\tau_{\min}$ - временная постоянная, определенная пользователем, которая зависит от максимальной частоты цепи - $\omega_{\max}$. Отметим, что $\tau_{\min}$ пропорционально $1/\omega_{\max}$.

На практике, интерес обычно представляют нижние собственные частоты, которые являются рабочими для механических систем. Поэтому вариацией $\tau_{\min}$ достигается компромисс между точностью и размерностью получаемых моделей.

Результаты моделирования

Расчетом определены 6 первых собственных частот конструкции. Результаты частотного анализа, полученные с использованием пакета ANSYS Multiphysics, а также в результате моделирования в пакете NetALLTED [7,8] эквивалентной и сокращенных при различных значениях $\tau_{\min}$ схем представлены в таблице 1.

Результаты моделирования в пакетах ANSYS и NetALLTED совпадают, что свидетельствует о том, что переход от модели, полученной с помощью метода конечных элементов, к эквивалентной схеме замещения не вносит дополнительные погрешности. Как видно из таблицы 1, погрешность моделирования схемы при $\tau_{\min} = 10^{-5}$, по сравнению с результатами ANSYS, составляет менее 1%, что свидетельствует о высокой точности реализованной модели. При этом размерность сокращенной модели по сравнению с несокращенной уменьшилась более чем в 5 раз.

Выходные характеристики эквивалентной и сокращенных при различных значениях $\tau_{\min}$ схем представлены на рис. 4-7.

Выводы

Изложенный в работе метод построения моделей механических компонентов позволяет использовать пакеты схемотехнического проектирования в качестве единого инструментария для моделирования электромеханических объектов. Использование сокращения на основе Y-Δ преобразования позволило получить модели с необходимой степенью точности, не уступающей на этапе получения эквивалентной схемы замещения точности математической модели, полученной с помощью метода конечных элементов. Предложенный метод может быть использован для расчета МЭМС, а в общем случае расширен и на объекты немеханической природы при условии возможности получения их математической модели в виде конечно-элементных матриц.

Таблица 1

	ANSYS	NetALLTED			
		Экви-вал. схема	Сокращенная схема		
$\tau_{\min}$	-	-	$3*10^{-5}$	$1.5*10^{-5}$	10^{-5}
Количество узлов	-	151	16	23	28
Количество элементов	-	776	94	119	145
Сокращение по узлам, %	-	-	89.404	84.7682	81.457
Сокращение по элементам, %	-	-	87.8866	84.6649	81.3144
1 пик, Гц	13.993	13.988	13.768	13.390	14.258
2 пик, Гц	91.795	91.794	90.548	91.589	91.477
3 пик, Гц	176.29	176.29	176.25	176.28	176.29
4 пик, Гц	477.44	477.44	476.60	477.21	477.27
5 пик, Гц	495.69	495.69	611.01	529.98	493.82
6 пик, Гц	861.83	861.82	1043.8	920.16	861.35
Максимальная погрешность, %	-	-	23.31	6.96	0.34

Результаты моделирования

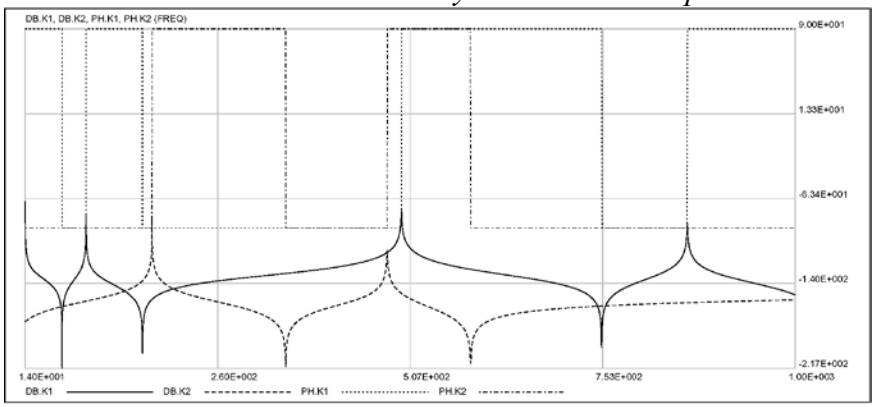


Рис. 4. АЧХ, ФЧХ эквивалентной схемы замещения для закрепленной неоднородной балки с тремя степенями свободы (несокращенный вариант)

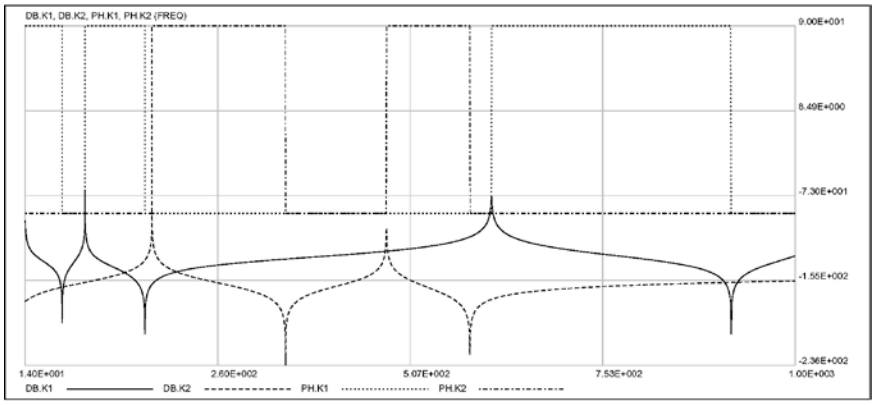


Рис. 5. АЧХ, ФЧХ эквивалентной схемы замещения для закрепленной неоднородной балки с тремя степенями свободы (сокращенный вариант, $\tau_{\min} = 3*10^{-5}$)

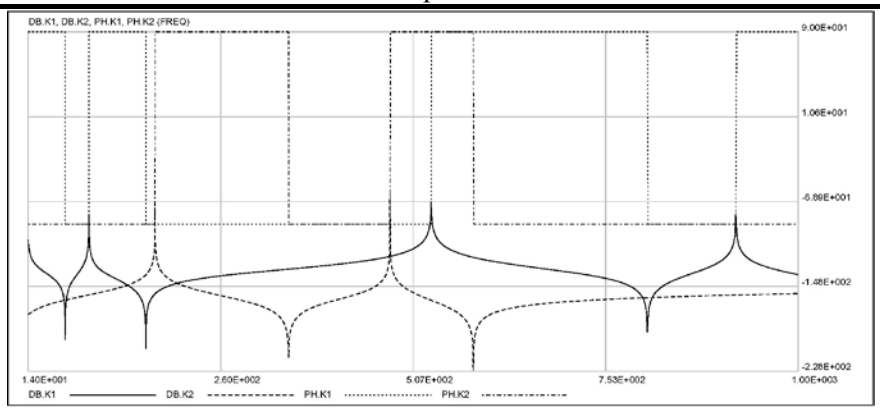


Рис. 6. АЧХ, ФЧХ эквивалентной схемы замещения для закрепленной неоднородной балки с тремя степенями свободы (сокращенный вариант, $\tau_{\min} = 1.5*10^{-5}$)

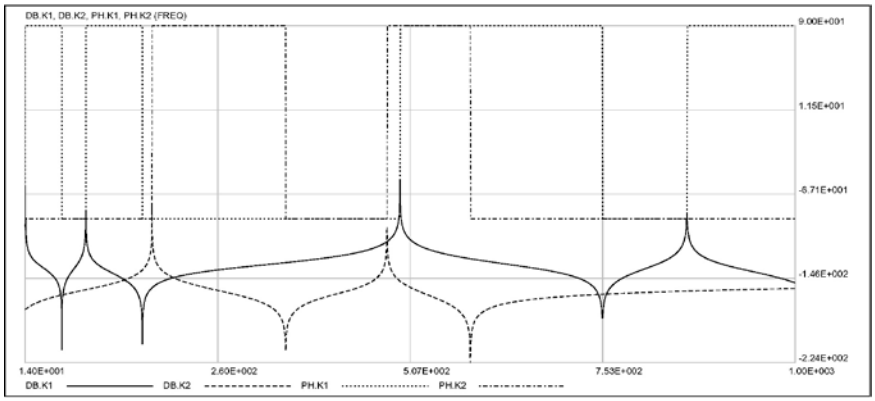


Рис. 7. АЧХ, ФЧХ эквивалентной схемы замещения для закрепленной неоднородной балки с тремя степенями свободы (сокращенный вариант, $\tau_{\min}$ = 10-5).

Список литературы

5. Ansys Inc.: http://www.ansys.com/.
6. Rudnyi E., Lienemann J., Greiner A., Korvink J. G. mor4ansys: Generating Compact Models Directly from ANSYS Models // Technical Proceedings of the 2004 Nanotechnology Conference and Trade Show (Nanotech 2004, March 7-11, 2004). – Boston, Massachusetts, USA, vol. 2. – P. 279-282.
7. Hsu J. T., Vu-Quoc L. A rational formulation of thermal circuit models for electrothermal simulation. – part 1: Finite element method // IEEE Transactions on Circuits and Systems – I: Fundamental Theory and Applications, vol. 43, 1996. – P. 721-732.
8. Руденко Ю.А., Ладогубец В.В., Ладогубец А.В. Алгоритм уменьшения размерности RLC цепей // Электроника и связь. – 2004. - №21. – с. 72-74.
9. Sheehan B.N. TICER: Realizable Reduction of Extracted RC Circuits // Digest of Technical Papers IEEE/ACM Proc. of ICCAD. – 1999. – p. 200-203.
10. Chirayu S.A., Masud H.C., Yehea I.I. Realizable RLCK Circuit Crunching // Proceedings of the 40th conference on design automation 2003. - Anaheim, CA, USA. - 2003. – p. 226 – 231.
11. Petrenko A., Ladogubets V., Tchkalov V., Pudlowski Z. ALLTED – a computer-aided engineering system for electronic circuit design. – Melbourne: UICEE, 1997, 205 p.
12. Петренко А.І. Мережний пакет для комп’ютерного проектування мікро-електромеханічних систем (МЕМС) // Розвиток інформаційно-комунікаційних технологій та розбудова інформаційного суспільства в Україні. – К.: Зв'язок, 2007. – С. 143-156.

УДК 62-82:658.512.011.56

ЛАДОГУБЕЦ В.В.,
КРАМАР А.В.,
ФИНОГЕНОВ А.Д.

АДАПТАЦИЯ ПАРАЛЛЕЛЬНОГО АЛГОРИТМА СПУИП ДЛЯ КЛАСТЕРА НТУУ «КПИ»

Описаны проблемы адаптации параллельного алгоритма СПУИП, реализованного в составе пакета Allted, для работы на кластере НТУУ «КПИ» и пути их решения. Выполнена модификация алгоритма баланса загрузки для использования возможностей аппаратной части. Приведены результаты решения тестовой задачи и сделаны выводы об эффективности предложенной реализации.

The problems of adaptation of SPUIP parallel algorithm implemented in ALLTED software to work on NTUU KPI's cluster and the ways to solve them are described. To make use of hardware possibilities, a modification of the load balance algorithm has been done. The results of a test task solution are given, and the conclusions regarding efficiency of the suggested realization are made.

С развитием многопроцессорных вычислительных систем роль параллельных вычислений неуклонно растет. Практически, параллельные вычисления стали единственным средством, позволяющим использовать современную компьютерную технику. Однако с усовершенствованием техники появилась необходимость адаптации разработанных параллельных алгоритмов к текущим реалиям.

Так в [1] был описан параллельный алгоритм оптимизации метода случайного поиска с уменьшением интервала поиска (СПУИП) [2], реализованного в составе ППП Allted [3]. Целью данной статьи является описание проблем и их решение при адаптации пакета и алгоритма для работы в составе кластера НТУУ «КПИ» [4].

В первую очередь рассмотрим отличия мультипроцессорной вычислительной среды от используемой ранее (рис.1).

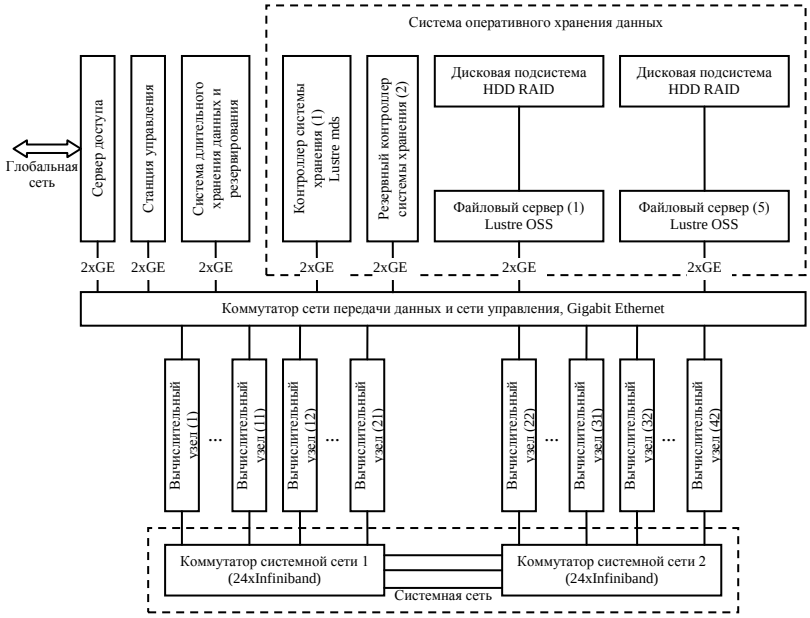


Рис.1. Структура кластера НТУУ «КПИ»

Каждый вычислительный узел кластера представляет собой два 2-х ядерных процессора Intel Xeon 5160 2x3.0 ГГц. Системная сеть построена на технологии InfiniBand/Gigabit Ethernet. В качестве операционной системы используется одна из версий ОС UNIX. Необходимо также отметить, что процессоры кластера имеют 64-х разрядную архитектуру.

Таким образом, можно выделить следующие этапы, необходимые для реализации алгоритм СПУИП в составе Allted на кластере НТУУ «КПИ»:

1)Адаптировать 32-х разрядный пакет для работы на 64-х разрядной платформе

2)Установить библиотеку PVM [5] и адаптировать ее к правилам безопасности, принятым на кластере.

3) Проанализировать конфигурацию вычислительной сети и адаптировать баланс загрузки.

Адаптация пакета Allted

Проблемы адаптации пакета в первую очередь сводятся к выбору одного или набора компиляторов, которые дают возможность произвести корректную сборку исходных файлов пакета. Исходные файлы пакета написаны на языке FORTRAN, однако многие дополнения к пакету описывались в более позднее время и написаны на языке Си (в том числе и реализация параллельного алгоритма СПУИП). Компиляция производилась в два этапа: в первом случае с помощью компилятора f2c[6] из исходных кодов на FORTRAN получались объектные файлы, на втором из файлов Си получались объектные файлы с помощью компилятора gcc 3.4.6. Компиляция исполняемых файлов также производилась с помощью компилятора gcc. Для эмуляции 32-х разрядной среды в команды компиляции была включена опция «–m32».

Установка библиотеки PVM.

При установке PVM также необходимо установить опцию эмуляции «-m32» для совместимости с пакетом. Кроме этого необходимо учитывать следующую особенность PVM. Передача сообщений происходит с использованием утилиты rsh (Remote shell). Ее недостатком является незащищенность, поэтому на большинстве систем в целях обеспечения безопасности ее использование ограничено. Разработчиками была предусмотрена такая возможность, и для передачи можно воспользоваться стандартным протоколом – ssh (secure shell), который обеспечивает шифрование передаваемых данных. Для его использования необходимо указать путь к файлу ssh – в файлах конфигурации PVM. Недостатком использования ssh является увеличение времени обмена между процессами.

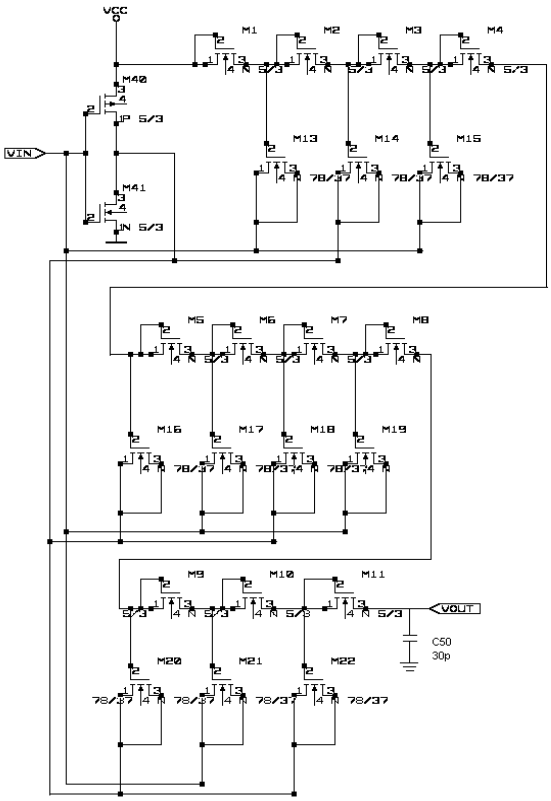
Адаптация алгоритма баланса загрузки

Описанный в [7] алгоритм баланса загрузки не предусматривал наличие в составе кластера многопроцессорных (а тем более многоядерных) вычислительных систем. Т.к. каждый узел кластера суммарно содержит 4-е вычислительных ядра, которые одинаковы по своим параметрам, то очевидно, что возможен запуск до 4-х slave процессов на одном узле одновременно. Т.к. относительные мощности всех ядер равны, то каждый slave процесс будет выполнять:

$$n_i = \frac{N}{i};$$

где N – количество вычислений целевой функции выполняемых узлом, i – количество запускаемых slave-процессов на одном узле.

В качестве теста использовался пример, который аналогичен рассмотренному в [1]: задача оптимизации схемы подкачки рис.2. с увеличенным диапазоном изменения варьируемых параметров (время решения такой задачи на «машина1», которая использовались в экспериментах из [1] составляет более 1 часа, что делало весьма затруднительным проведения «чистого» эксперимента в виду загруженности техники). Напомним, что для этой схемы необходимо так поднять выходное напряжение (VC80) что бы в момент времени 2 msec напряжение ровнялось бы 15 V. В качестве варьируемых параметров можно синхронно изменять W, L, и толщину окисла (TOX) транзисторов M13-M22. Время моделирования – 3ms. В качестве моделей транзисторов используется BSIM1 модель (аналогичная Level13 в HSPICE).



Параллельная виртуальная машина конфигурировалась с различным количеством узлов (от 1 до 5) и различным количеством запускаемых задач на каждом узле (от 1 до 4). Результаты времени расчетов (приведены в секундах), а также соответствующие им коэффициенты ускорения (K_y) представлены в табл. 1.

Отметим, что в качестве времени решения указывается полное время работы пакета, а не время работы алгоритма. Т.е. полученные временные значения включают в себя не только затраты на обмен данными, но и время трансляции и анализа файла задания, формирование математической модели и запись файла результатов. Во всех экспериментах относительные мощности всех узлов были равны.

Рис.2. Схема подкачки

Таблица 1. Решение задачи оптимизации при различных конфигурациях PVM*

Количество запускаемых процессов	Количество узлов в составе PVM				
	N=1 (K_y)	N=2 (K_y)	N=3 (K_y)	N=4 (K_y)	N=5 (K_y)
1	1777 (1)	916 (1.94)	639 (2.78)	493 (3.60)	405 (4.39)
2	912 (1.95)	486 (3.66)	345 (5.15)	275 (6.46)	231 (7.69)
3	643 (2.76)	363 (4.90)	254 (7.00)	208 (8.54)	180 (9.87)
4	484 (3.67)	273 (6.51)	201 (8.84)	186 (9.55)	145 (12.26)

* - приведенное время является лучшим из 10 испытаний при каждой конфигурации параллельной виртуальной машины.

Приведенные результаты свидетельствуют о высокой эффективности реализованного алгоритма. Коэффициент ускорения для параллельной части алгоритма будет равен:

$$K_y = \frac{T_{послед.} - T_{форм.}}{T_{парал.} - T_{форм.}} \qquad (1)$$

где $T_{послед.}$ – время решения задачи пакетом, с использованием последовательного алгоритма; $T_{парал.}$ – время решения задачи пакетом, с использованием параллельного алгоритма; $T_{форм.}$ – время, включающее трансляцию файла задания, установки начальных значений, формирования математической модели и выходного файла результатов.

Время, затрачиваемое на последовательную часть работы пакета, составляет для данной задачи $T_{форм} \approx 40$ секунд. Например, для конфигурации параллельной виртуальной машины с пятью узлами и запуском 4-х slave-процессов на каждом, в соответствии с (1), получим:

$$K_y = \frac{1777 - 40}{145 - 40} = \frac{1737}{105} \approx 16.54$$

что составляет 82.7% от теоретически возможного ускорения.

Рост коэффициента ускорения и времени решения для некоторых из конфигураций может носить нелинейный характер. Это связано с тем обстоятельством, что в ряде случаев невозможно получить равномерную загрузку для всех slave-процессов. Так в тестовой задаче размер выборки для алгоритма СПУИП был равен N=200. Соответственно, что в ряде случаев, например для эксперимента, с 3 узлами и 3 запускаемыми slave-процессами, время решения будет определяться временем работы процесса с наибольшей загрузкой. Суммарное количество рассчитанных целевых функций равнялось 790.

При анализе времени решения задачи необходимо учитывать, что эксперименты проводились на узлах кластера, которые могли быть задействованы для решения задач других пользователей. Часто, получаемые результаты отличались от лучших на 30-70%.

Необходимо отметить еще одну возможность сокращения времени решения. Предложенный алгоритм разрабатывался для случая систем с разделенной оперативной памятью, поэтому запуск slave-процессов и обмен данными с ними по аналогии с [1] производился для каждого slave-процесса в отдельности (рис.3).

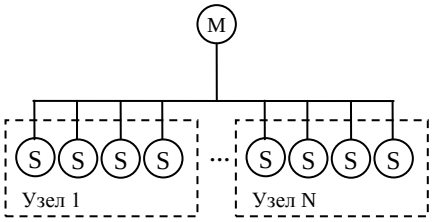


Рис.3 Организация обмена «master – каждый slave»

Т.к. на кластере НТУУ «КПИ» каждый узел имеет общую оперативную память, то, возможна организация взаимодействия по типу дерева, когда master-процесс производит запуск только одного slave-процесса на каждом узле (slave-процесс первого уровня), а тот в свою очередь запускает slave-процессы второго уровня.

Такая организация обмена предпочтительней для тех конфигураций мультипроцессорных вычислительных систем, где временные затраты на передачу данных значительны. Это характерно для систем, где используются низкоскоростные сети или сама сеть загружена. Уровней дерева может быть и больше, однако в этом случае необходимо учитывать необходимость передачи данных для уровней без общей памяти, что отразится на времени решения.

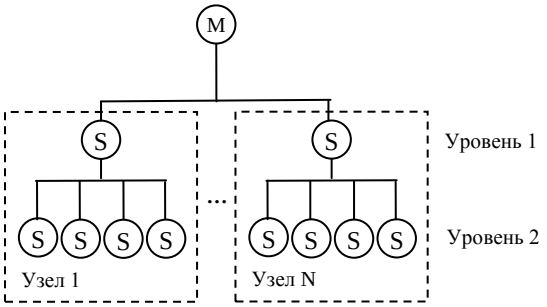


Рис.4 Организация обмена в виде дерева

Анализ времени передачи и приема данных на кластере показал, что основные затраты времени связаны не с передаваемым объемом данных, а временем согласования процессов. Напомним, что в соответствии с [1] при запуске slave-процессов осуществляется передача начальных данных (для данной задачи объем данных составлял $\approx 300Кб$). В дальнейшем передаются только данные, необходимые для расчетов значений целевых функций ($< 5Кб$ для задачи с двумя slave-процессами). В этом случае организация передачи между master- и slave-процессами в виде дерева может негативно сказаться на общем времени решения из-за необходимости обеспечения двух уровней связывания процессов.

Преимуществом организации обмена в виде дерева может стать возможность использование, отличных от других уровней, средств поддержки параллельных вычислений.

Выводы

Параллельный алгоритм СПУИП, реализованный в составе пакета Allted, показал высокую эффективность при реализации на кластере НТУУ «КПИ». Результаты тестирования показали постепенное замедление роста коэффициента ускорения, что говорит о достижении предела эффективного увеличения количества процессов для тестовой задачи. Конфигурация взаимодействия master- и slave-процессов должна выбираться в соответствии, как с анализом технических характеристик мультипроцессорной вычислительной системы, так и анализом особенностей реализуемого алгоритма.

Список литературы:

1. *Ладогубец В.В.*, Финогенов А.Д.: «Особенности реализации параллельных алгоритмов для одно-процессорных пакетов», Киев, Электроника и связь, №25, 2005, с.95-98.
2. *А.И. Петренко*, В.В. Ладогубец, В.В. Чкалов Оптимальное схемотехническое проектирование в машиностроении. - Киев,1989. – 164 стр.
3. *Petrenko A.*, Ladogubets V., Tchkalov V., Pudlowski Z.ALLTED - a computer-aided engineering system for electronic circuit design.(монографія) Melbourne: UICEE, 1997
4. Официальный сайт кластера НТУУ «КПИ»: http://hpcc.org.ua
5. Официальный сайт PVM: http://netlib.org/pvm
6. *S. I. Feldman*, David M. Gay, Mark W. Maimone, and N. L. Schryer, "A Fortran to C Converter," AT&T Bell Laboratories technical report, 1990
7. *Ладогубець В.В.* Алгоритм оптимального балансу загрузки Электроника и связь, №6, 1999

УДК 004.052.42

МАРКОВСЬКИЙ О.П.,
ФЕДОРЕЧКО О.І.,
ТУРЧЕНКО Ю.О.

КОРЕКЦІЯ ПОМИЛОК ПЕРЕДАЧІ ДАНИХ В КАНАЛАХ ЗІ СПЕКТРАЛЬНОЮ МОДУЛЯЦІЄЮ З ВИКОРИСТАННЯМ ЗВАЖЕНИХ КОНТРОЛЬНИХ СУМ

Стаття присвячена вирішенню проблеми підвищення ефективності виявлення, локалізації та виправлення помилок передачі цифрових даних в каналах зі спектральною модуляцією за рахунок врахування особливостей їх виникнення. Для виправлення бітових спотворень, викликаних одно та двократних помилок передачі канальних сигналів запропоновано зважену суму зі спеціальними ваговими коефіцієнтами. Розроблено алгоритм виявлення, локалізації та корекції таких бітових спотворень. Доведено, що запропонована технологія корекції помилок потребує менше контрольних розрядів та значно простіша у порівнянні з корегуючими кодами.

Paper is dedicated to solving the problem of increasing the efficiency of data transmission error detecting, localization and correcting in spectrum modulation channel by properties such errrors appearance accounted. For correction of the bit distortions coused by one or two errors of channel error transmission the weighed check sum with special wigh coefficients has been proposed. Algorithm for error detection and correction has been worked out. It has been shown that proposed error correction techniques is more simple and demanded control bits less than error correction codes.

Вступ

Розвиток процесів інформаційної інтеграції в усіх сферах людської діяльності вимагає невпинного вдосконалення засобів передачі цифрових даних. Важливою складовою такого вдосконалення виступає забезпечення надійності передачі даних. Труднощі забезпечення високої достовірності передачі даних зумовлені складною природою фізичних процесів проходження сигналів в довгих лініях, їх взаємним впливом (міжсигнальна інтерференція) та впливом зовнішніх перешкод. Нині проблема забезпечення ефективного контролю та корекції помилок передачі даних стає ще більш важливою. Протягом останнього десятиліття значно зросли об’єми та швидкість передачі даних. Ці процеси супроводжуються рядом чинників, що негативно впливають на надійність передачі цифрових даних. Зокрема, зменшення часових інтервалів між сигналами мають наслідком зростання числа помилок, які викликані явищами міжсигнальної інтерференції. Збільшення інтенсивності зовнішніх електромагнітних полів, зумовлене динамічним розширенням використання ефірних, бездротових технологій передачі має наслідком зростання помилок, викликаних зовнішніми перешкодами. Багатократне зростання об’ємів інформації, що передається в комп’ютерних мережах, вимагає перегляду та переоцінки імовірнісних методів контролю помилок передачі, зміщення акценту на користь методів їх гарантованого виявлення. Наведені фактори диктують необхідність вдосконалення засобів контролю та корекції помилок, адекватного прогресу технології передачі інформації.

Для підвищення швидкості передачі даних за рахунок більш повного використання спектральних характеристик каналів широке застосування знаходить спектральна модуляція цифрових даних. Найбільш розповсюдженим видом такої модуляції є амплітудно-фазова модуляція, або квадратурно-амплітудна модуляція (Quadrature Amplitude Modulation – QAM) при використанні якої група бітів модулюється стрибкоподібною зміною фази і амплітуди синусоїдального сигналу. Така заміна фази і амплітуди називається канальним сигналом. Помилка передачі канального сигналу призводить до спотворення цілої групи бітів. Ця особливість виникнення бітових спотворень в каналах зі спектральною модуляцією вимагає вдосконалення технології їх виявлення та виправлення.

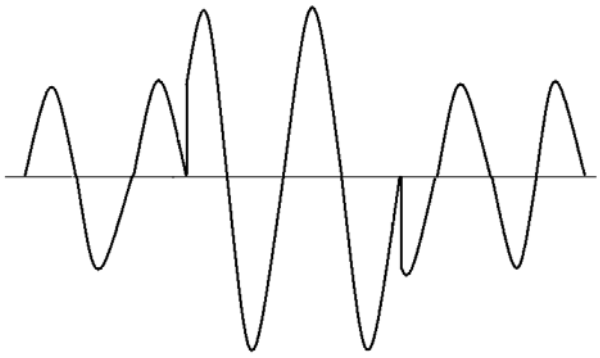
Наведене визначає необхідність збільшення надійності передачі даних з каналах зі спектральною модуляцією за рахунок розробки спеціальних методів виявлення і виправлення помилок з урахуванням особливостей їх виникнення в таких каналах. Досягнутий в останні роки прогрес в галузі інтегральної технології відкриває нові можливості для вдоско-

налення засобів контролю та корекції помилок.

Таким чином, задача підвищення ефективності виправлення помилок в каналах зі спектральною модуляцією з огляду на особливості сучасного етапу розвитку інформаційних технологій є актуальною та важливою для практики.

Аналіз ефективності існуючих методів корецїї помилок в каналах зі спектральною модуляцією

В сучасних комп'ютерних мережах та системах телекомунікацій дедалі ширше застосування знаходить спектральна модуляція цифрових даних [1]. Найпоширенішим видом такої модуляції є QAM. При використанні цього виду спектральної модуляції кожні k бітів даних модулюються стрибкоподібною зміною фази та амплітуди несучого синусоїдального сигналу. Рисунок 1 ілюструє таку модуляцію для k=4 (QAM-16).



0101 **1110**

Рис.1. Приклад модуляції груп із 4-ох біт стрибкоподібними змінами фази та ампулітуди синусоїдального сигналу для QAM-16

При передачі блоку B, що складається із m бітів: $B=\{b_1,b_2,\ldots,b_m\}$,. $l\in\{1,\ldots,m\}$: $b_l\in\{0,1\}$ і використанні спектральної модуляції його можна розглядати як такий, що складається з $t = m/k$ канальних символів $B=\{X_1,X_2,\ldots,X_t\}$. Кожен j-тий ($j\in\{1,\ldots,t\}$) із цих символів X_j включає k суміжних бітів блоку B: $X_j = \{x_{j1},x_{j2},\ldots,x_{jk}\}=\{b_{(j-1)\cdot k+1},b_{(j-1)\cdot k+2},\ldots,b_{j\cdot k}\}$.

В результаті помилок передачі канальних сигналів, відповідні їм символи спотворюються, тобто при виникненні помилки передачі j-го канального сигналу, коди j-тих символів на передавачі та приймачі не однакові: $X_{sj} \neq X_{rj}$. Очевидно, що при цьому коди символів X_{sj} та X_{rj} можуть відрізнятися один від одного на від одного до k бітів.

Таким чином, при виникненні помилок в передачі u канальних сигналів потенційно може бути спотворено до $k\cdot u$ бітів блоку B. При цьому спотворені біти локалізовані у u групах, що відповідають помилково переданим канальним символам. Ця особливість має бути врахованою при створенні ефективних засобів виявлення та корекції бітових спотворень, викликаних помилками передачі канальних сигналів. На практиці, ймовірність виникнення помилок передачі канальних символів відчутно (на 3-4 порядки) зменшується з ростом кратності – u. Тому, можна вважати, що кратність помилок передачі канальних сигналів не перевищує 2-х (u=2), причому ймовірність помилкової передачі двох канальних сигналів на 3-4 порядки менше ніж однократної помилки. Виникнення помилок більшої кратності дуже малоймовірне [1,2].

Для корекції бітових спотворень блоку B даних викликаних помилкою передачі канальних сигналів найбільш часто використовуються дві технології:

1. Корегуючи коди (коди Хемінга, Голея, лінійні коди та ін.), які здатні виправляти $u\cdot k$ бітових помилок в m-розрядному блоці даних, причому ці спотворені біти можуть бути локалізованими будь-яким чином. Кількість q контрольних розрядів при цьому визначається формулою [1]:

$$\sum_{j=0}^{u\cdot k}\binom{m}{j} \leq 2^q \qquad (1)$$

Наприклад, при контролі передачі блока об'ємом 1 Кбайт ($m=2^{13}$) і використанні модуляції QAM-16 (k=4) за умови корекції всіх бітових спотворень, зумовлених помилковою передачею одного канального сигналу значення q складає 48, при QAM-64 (k=6) q = 71. За умови корекції $2\cdot k$ бітових помилок (виправлення всіх бітових спотворень, викликаних помилками передачі двох канальних сигналів) за формулою (1) при QAM-16 q=89, а при QAM-64 (k=6) q=127. Таким чином, основним недоліком застосування корегуючи кодів для виправлення помилок передачі даних в каналах зі спектральною модуляцією є велика кількість контрольних розрядів. Це зумовлено тим, що корегуючи коди не враховують згадану вище особливість бітових спотворень, що виникають в таких каналах.

2. Така особливість частково врахована в рамках другої технології корекції помилок. Ця

технологія являє собою комбінування корегуючих кодів та спеціального групування бітів блоку, яке має на меті россіяння бітових спотворень [1]. Технологія передбачає формування кодових груп, що складаються з однойменних бітів канальних символів. Тобто формується k контрольних груп, кожна з яких містить m/k біт. До першої з яких входять перші біти всіх символів блоку, до другої – другі біти всіх символів і так далі. Завдяки такому групуванню бітів блоку, при виникненні u помилок передачі канальних сигналів в кожній із груп буде не більше, ніж u спотворених бітів. Для кожної контрольної групи використовується свій корегуючий код, здатний вип.-равляти u-кратну бітову помилку, а також відповідні контрольні розряди. При цьому загальна кількість контрольних розрядів визначається, добутком числа k груп на кількість g контрольних розрядів, здатних корегувати u-кратну бітову помилку в (m/k)-бітовій групі, причому значення g визначається формулою:

$$\sum_{j=0}^{u}\binom{m/k}{j} \le 2^{g} \qquad (2)$$

При $m=2^{13}$, $u=1$ і $k=4$ потрібне число g контрольних розрядів для корекції однократної бітової помилки в групі, що складається з $m/k=2^{11}$ бітів дорівнює 12, відповідно, загальна кількість контрольних розрядів дорівнює $4\cdot 12 = 48$. При корекції бітових спотворень, викликаних двома помилками передачі канальних сигналів за другою технологією, кількість контрольних розрядів в кожній із контрольних груп має забезпечувати виправлення 2-х бітових помилок, відповідно, кількість g контрольних розрядів в групі у відповідності з формулою (2) становить 21, а загальна кількість контрольних розрядів визначається як $4\cdot 21=84$. При корекції помилок в каналах з QAM-64 (k=6) для корекції бітових спотворень, викликаних однією помилкою передачі канального сигналу потрібно $6\cdot 8 = 48$ розрядів, а для виправлення двох – $6\cdot 14 = 84$ розрядів.

Таким чином, хоча використання другої технології корекції помилок в каналах зі спектральною модуляцією потребує меншої кількості контрольних розрядів, вона залишається досить значною.

Загальним недоліком відомих методів корекції помилок в каналах зі спектральною модуляцією є значна обчислювальна складність процесу виправлення помилок, яка квадратично залежить від розміру блоку, що контролюється. Для блоків значної довжини це може викликати помітні затримки, що є критичним для широкого кола застосувань.

Ціллю роботи є підвищення ефективності виправлення помилок передачі даних в каналах зі спектральною модуляцією за рахунок зменшення кількості потрібних контрольних розрядів та спрощення процедури корекції бітових спотворень.

Використання зважених контрольних сум для корекції помилок передачі канальних символів

Однім з найпростіших методів виявлення помилок передачі даних є контрольні суми та їх різновид – зважені контрольні суми (Weighed Check Sum – WCS) [3]. Для виявлення помилок в каналах зі спектральною модуляцією запропоновано спеціальну модифікацію зважених контрольних сум [4].

Аналіз інформаційних аспектів зважених контрольних сум показав, що вони можуть бути ефективно використані не тільки для виявлення, але й для корекції помилок в каналах зі спектральною модуляцією.

Для вирішення задачі виправлення бітових спотворень, зумовлених помилками передачі канальних сигналів, кратність яких не перевищує 2, розроблено відповідний метод, що має за основу спеціальну модифікацію зваженої контрольної суми.

Запропонований метод базується на використанні інформації, що міститься у різниці зважених контрольних сум приймача та передавача. Метод передбачає використання спеціальних вагових коефіцієнтів, які дозволяють відрізняти помилки передачі канальних сигналів парної та непарної кратності. Ваговий коефіцієнт W_j j-того символу X_j ($j\in\{1,\ldots,t\}$) пропонується визначати в вигляді: $W_j = 2\cdot j+1$. Розрядність r_w такого вагового коефіцієнта визначається як $\lfloor \log_2 t \rfloor + 1$. Таким чином, молодший розряд вагового коефіцієнту завжди дорівнює одиниці, а старші – містять інформацію про місце знаходження символу в інформаційному блоці.

Запропонована модифікація зваженої контрольної суми WCS містить в собі k r_w-бітових полів: WCS=$\{s_1,s_2,\ldots,s_k\}$. Кожне i-те поле s_i ($i\in\{1,\ldots,k\}$) пропонується формувати як суму за модулем 2 по всіх символах блоку добутків

i-го біту символу на відповідний ваговий коефіцієнт:

$$s_i = \bigoplus_{j=1}^{t} x_{ji} \cdot W_j \qquad (3)$$

Якщо позначити через WCS_S і WCS_R зважені контрольні суми, що обчислюються відповідно на передавачеві та приймачеві, останній формує різницю Δ контрольних сум як їх суму за модулем 2: $\Delta = WCS_S \oplus WCS_R$. Зрозуміло, що як і зважені контрольні суми, код Δ містить k r_w-розрядних полів: $\Delta = \{\Delta_1,\ldots,\Delta_k\}$. Якщо код Δ містить всі нульові елементи, то це говорить про те, що передача інформаційного блоку пройшла успішно і безпомилково. Наявність ненульових компонентів коду Δ свідчить про наявність помилки в одержаному інформаційному блоці.

При виникненні однократної помилки передачі канального сигналу, наприклад, для визначеності l-того, $l\in\{1,\ldots,t\}$, може бути спотворена довільна множина Ω з k бітів символу X_l. При цьому ця підмножина Ω включає в якості елементів номери тих бітів символу X_l котрі були спотворені: $\Omega = \{\phi_1, \phi_2,\ldots, \phi_q\}$, $q\le k$, $\phi\in\{1,\ldots,k\}$, $x^S_{l\phi} \neq x^R_{l\phi}$; очевидно, що множина $\Omega\neq\varnothing$. Очевидно, що поля коду Δ, номера яких не належать множині Ω дорівнюють нулю: $\forall e\notin\Omega$: $\Delta_e=0$. Поля коду Δ, номера яких належать множині Ω співпадають з ваговим коефіцієнтом W_l спотвореного символу: $\forall u\in\Omega$: $\Delta_u = W_l$.

Таким чином, процедура корекції однократної помилки передачі канального сигналу полягає в виконанні наступної послідовності дій:

1. Перевіряється виконання умов: всі ненульові компоненти різниці Δ зважених контрольних сум передавача та приймача рівні між собою і їх молодший розряд дорівнює одиниці: $\forall\ \Delta_i \neq 0$, $i\in\{1,\ldots,k\}$: $\Delta_i \bmod 2 = 1$, $\forall\Delta_u\neq0$: $\Delta_i = \Delta_u$.

2. Визначається номер l спотвореного символу блока шляхом ділення на 2 коду будь-якого не рівного нулю поля Δ_i : $l = (\Delta_i - 1)/2$, $\Delta_i\neq0$.

3. В l-тому символі прийнятого блоку інвертуються ті біти, номери яких відповідають ненульовим полям коду Δ: $\forall\Delta_i \neq0$: $x_{li} = x_{li}\oplus1$.

Використання запропонованої процедури ілюструється наступним прикладом. Нехай при передачі блоку довжиною 64 байти ($m=2^9$) використовується QAM-16 (k=4). Тоді кількість t символів дорівнює 128, а їх вагові коефіцієнти $W_1,\ldots,W_{128}$ розрядністю $r_w = \log_2 t+1=8$, утворюють ряд: 1, 3, 5, ...,255. Якщо припустити, що при передачі 42-го канального сигналу (l=42) трапилася помилка, яка викликала спотворення 1-го, 3-го та 4-го бітів відповідного символу X_{42}, ваговий коефіцієнт якого W_{42} = 85, то різниця Δ зважених контрольних сум передавача та приймача такі поля: $\Delta=\{\Delta_1=85, \Delta_2=0, \Delta_3=85, \Delta_4=85\}$. Очевидно, що $\Omega=\{1,3,4\}$. Всі три ненульові поля Δ_1, Δ_3, Δ_4 рівні між собою і містять одиницю в молодшому розряді, що розпізнається, як спотворення одного символу блоку. Згідно з п.2 визначається номер помилково переданого символу: $l = (\Delta_1 - 1)/2 = 84/2 = 42$. Оскільки не дорівнюють нулю поля Δ_1, Δ_3 та Δ_4 – виконується інвертування 1-го, 3-го і 4-го бітів символу X_{42} прийнятого блоку.

При виникненні двократної помилки передачі канального сигналу, наприклад, з номерами u та l, u,l$\in\{1,\ldots,t\}$, $u\neq$l отримають зміни два символи, блоку що контролюється: X_u та X_l, при цьому загальне число спотворених бітів лежить в інтервалі від 2-х до 2· k. В даній ситуації для кожного i-го із k однойменних бітів X_u та X_l є можливі чотири варіанти спотворень:

1. i-ті біти обох символів X_u та X_l не спотворені. Відповідно i-та компонента зваженої контрольної суми приймача не зміниться при передачі, так, що Δ_i=0.
2. i-тий біт символу X_u спотворюється (інвертується) при передачі, в той час як i-тий біт символу X_g не змінюється; в цій ситуації зміниться відповідна компонента зваженої контрольної суми передавача, так, що $\Delta_i = 2\cdot u+1$, тобто молодший розряд Δ_i дорівнює одиниці, а старші вказують на номер u помилково переданого символу ;
3. i-тий біт символу X_u не змінюється, а однойменний біт символу X_l спотворюється: в цьому випадку зміниться відповідна компонента Δ_i різниці зважених контрольних сум. Молодший розряд Δ_i дорівнюватиме одиниці, а старші – вказують на номер l символу X_l : $\Delta_i = 2\cdot l + 1$.
4. i-тий біт символу X_u та i-тий біт символу X_l спотворюються при помилці передачі відповідних канальних сигналів. В цьому випадку i-

та компонента Δ дорівнюватиме сумі за модулем два вагових коефіцієнтів W_u та W_l : $\Delta_i = W_u \oplus W_l = 2\cdot(u\oplus l)$. Тобто, молодший розряд Δ_i дорівнює нулю, а старші дорівнюють сумі за модулем 2 номерів u та l помилково переданих символів.

Для реалізації виправлення двократних помилок передачі канальних сигналів пропонується розпізнавати 4 типи таких помилок:

1. Має місце помилка передачі u-го та l-го канальних сигналів, при цьому в символах X_u та X_l спотворюються різні розряди. Ця помилка в подальшому іменується помилкою першого роду і вона виявляється коли виконується умова: серед кодів $\Delta_1,...,\Delta_k$ обов'язково є непарні відмінні між собою коди і відсутні ненульові парні. Іншими словами, існують непусті множини Θ та Ξ номерів, що належать інтервалу $1..k$, причому $\Theta\cap\Xi=\varnothing$, $\forall e\in\Theta$: $\Delta_e \bmod 2=1$, $\forall s\in\Xi$: $\Delta_s \bmod 2=1$. Виправлення помилок першого типу виконується шляхом інвертування в символі, номер u визначається як $u = (\Delta_e-1)/2$, бітів, номери котрих належать множині Θ. Крім того, в символі з номером $l = (\Delta_s-1)/2$ інвертуються біти, номери яких належать множині Ξ.

Наприклад, якщо з використанням QAM-16 (k=4) передається 15 4-розрядних символів (t=15) і помилково передані 7-й та 12 символи (u=7, l=12), причому на передавачеві $X_7^S = 1\ \ 0, X_{12}^S = 1\ \ 0\ \ 0$, а на приймачі $X_7^R = 1\ \ 1, X_{12}^R = 0\ \ 0\ \ 0$, то компоненти різниці зважених контрольних сум приймача та передавача прийматимуть значення: $\Delta_1=0$, $\Delta_2=15$, $\Delta_3=25$, $\Delta_4=15$. $\Theta=\{2,4\}$, $\Xi=\{3\}$. Згідно з наведеним вище, інвертуються 2-й та 4-й біти X_7^R та 3-й біт символу X_{12}^R.

2. В символах X_u та X_l спотворюються біти зі співпадаючими номерами, але, разом з тим, в кожному з символів X_u і X_l є біти, що спотворені лише в одному з символів. Ця помилка іменується помилкою другого роду і вона виявляється коли виконується умова: серед кодів $\Delta_1,...,\Delta_k$ обов'язково є непарні, відмінні між собою коди, а також не рівний нулю парний код, який є сумою за модулем 2 згаданих вище непарних. Іншими словами, існують непусті множини Θ, Ξ та Ψ номерів, що належать інтервалу $1..k$, причому $\Theta\cap\Xi=\varnothing$, $\Theta\cap\Psi=\varnothing$ і $\Xi\cap\Psi=\varnothing$. $\forall e\in\Theta$: $\Delta_e \bmod 2=1$, $\forall s\in\Xi$: $\Delta_s \bmod 2=1$, $\forall z\in\Psi$: $\Delta_z \bmod 2=0$. $\Delta_z = \Delta_e \oplus \Delta_s$. Виправлення помилок другого типу виконується шляхом інвертування в символі, номер u визначається як $u = (\Delta_e-1)/2$, бітів, номери котрих належать множинам Θ та Ψ. Крім того, в символі з номером $l = (\Delta_s-1)/2$ інвертуються біти, номери яких належать множинам Ξ та Ψ.

Наприклад, якщо в рамках наведеного вище прикладу 7-й та 12-й символи на передавачі рівні $X_7^S = 1\ \ 0, X_{12}^S = 1\ \ 0\ \ 0$, а на приймачі $X_7^R = 1\ \ 1, X_{12}^R = 0\ \ 0\ \ 1$, то компоненти різниці зважених контрольних сум приймача та передавача рівні: $\Delta_1=0$, $\Delta_2=22$, $\Delta_3=25$ і $\Delta_4=15$, $\Theta=\{3\}$, $\Xi=\{4\}$, $\Psi=\{2\}$, причому $\Delta_2=\Delta_3\oplus\Delta_4 = =01111_2\oplus11001_2=10110_2$. Згідно з наведеним вище, інвертуються 2-й та 4-й біти X_7^R, а також 2-й і 3-й біти символу X_{12}^R.

3. В символах X_u та X_l спотворюються біти зі співпадаючими номерами, але, разом з тим, в символі X_u є спотворені біти, які передалися вірно в X_l. Ця помилка іменується подвійною помилкою третього типу і вона виявляється коли виконується умова: серед кодів $\Delta_1,...,\Delta_k$ обов'язково є непарні, рівні між собою коди, а також не рівний нулю парний код. Іншими словами, існують непусті множини Θ та Ψ номерів, що належать інтервалу $1..k$, причому $\Theta\cap\Psi=\varnothing$. $\forall e\in\Theta$: $\Delta_e \bmod 2=1$, $\forall z\in\Psi$: $\Delta_z \bmod 2=0$. Виправлення помилок другого типу виконується шляхом інвертування в символі, номер u визначається як $u = (\Delta_e-1)/2$, бітів, номери котрих належать множинам Θ та Ψ. Крім того, в символі з номером $l = ((\Delta_e\oplus\Delta_z)-1)/2$ інвертуються біти, номери яких належать множині Ψ.

Наприклад, якщо в рамках наведеного вище прикладу 7-й і 12-й символи на передавачі: $X_7^S = 1\ \ 0, X_{12}^S = 1\ \ 0\ \ 0$, а на приймачі $X_7^S = 1\ \ 0, X_{12}^R = 1\ \ 0\ \ 0$, то компоненти різниці зважених контрольних сум приймача та передавача рівні : $\Delta_1=0$, $\Delta_2=22$, $\Delta_3=0$, $\Delta_4=15$. Для корекції інвертуються 2-й та 4-й 7-го символу X_7^R, визначається номер l другого спотвореного символу $l = ((\Delta_e\oplus\Delta_z)-1)/2 = (01111_2\oplus10110_2)-1)/2 = (11001_2-1)/2=12$. Інвертується 2-ий біт визначеного символу X_{12}^R.

4. В символах X_u та X_l спотворюються біти зі співпадаючими номерами, і не існує бітів, спотворених тільки в одному з символів. Ця помилка іменується подвійною помилкою че-

твертого типу і вона виявляється коли виконується умова: коди $\Delta_1,...,\Delta_k$ або рівні нулю, або дорівнюють не рівному нулю парному коду. Іншими словами, існує непуста множина Ψ номерів, що належать інтервалу $1..k$, причому $\forall z \in \Psi$: $\Delta_z \bmod 2=0$. Така помилка не може бути виправлена без повторної передачі, проте об'єм інформації, що передається може бути суттєво зменшений. Для виправлення помилок цього типу пропонується наступна процедура: символи блоку послідовно передаються повторно і кожен з них порівнюється з раніше прийнятим. При виявленні першої пари неспівпадаючих символів, наприклад з номером u, визначаються позиції неспівпадаючих розрядів та обчислюється номер l другого символу, при передачі котрого мали місце помилки $l = (2 \cdot u \oplus \Delta_z)/2$. В цьому, l-му символі коректуються біти, що належать множині Ψ. Цілком очевидно, що для корекції подвійної помилки 4-го типу необхідна повторна передача, в середньому $t/3$ символів.

Наприклад, якщо в рамках наведеного вище прикладу 7-й і 12-й символи на передавачі рівні $X_7^S = 1\ \ 0, X_{12}^S = 10\ \ 0$, а на приймачі – $X_7^R = 1\ \ 1, X_{12}^R = 00\ \ 1$, то компоненти різниці Δ зважених контрольних сум приймача та передавача рівні : $\Delta_1=0$, $\Delta_2=22$, $\Delta_3=0$, $\Delta_4=22$. Для корекції проводиться повторна передача символів із 1-го по 7-й. При повторній передачі 7-го символу виявляється неспівпадіння початково прийнятого та повторно прийнятого символів. Визначається, що при першій передачі спотворені 2-й та 4-й біти. Обчислюється номер l другого спотвореного символу: $l=(2 \cdot u \oplus \Delta_z)/2 = (01110_2 \oplus 10110_2)/2 = 12$. В 12-му символі коректуються 2-й та 4-й біти.

Очевидно, що за умови рівної ймовірності появи одного із 2^k варіантів спотворення k-бітових символів в результаті помилки передачі канального сигналу , що їх моделює ймобути помилково ідентифікована як двократна і, відповідно, невірно виправлена. Для того, щоб це було можливо, бітові спотворення при 4-кратних помилках мають мати певну локалізацію. Неважко продемонструвати, що ймовірність таких локалізацій при виникненні 4-кратних помилок передачі канальних сигналів приблизно складає $2^{-2 \cdot k}$. По аналогії з аналізом ймовірності ризику для невірної ідентифікації однократних помилок, можна показати, що вірність двократної помилки четвертого типу складає 2^{-k}, так, що середнє число N_k символів, що повторно передаються для виправлення двократної помилки визначається наступною формулою:

$$N_k = \frac{t}{3 \cdot 2^k} \qquad (4)$$

При виправленні однократної помилки виникає ризик її неправильного розпізнавання з трьохкартною. Для цього потрібно, щоб у всіх трьох символах, що були неправильно передані внаслідок таких помилок спотворенню підлягали одні і ті ж біти. Очевидно, що ймовірність виникнення такої ситуації рівна $2^{-2 \cdot k}$. Слід сказати й те, що ймовірність виникнення трьохкратної помилки передачі суттєво нижча у порівнянні з однократною. Для більшості каналів залежність ймовірності виникнення помилок передачі канальних сигналів від їх кратності носить біноміальний характер [1]. Якщо позначити через p_c ймовірність помилкової передачі одного символу в модемних лініях зі спектральною модуляцією, то ймовірність виникнення однократної помилки в T_{13} разів більша у порівнянні з трьохкратною, причому числове значення T_{13} визначається за формулою:

$$T_1 \approx \frac{6 \cdot t \cdot p_c \cdot (1-p_c)^{t-1}}{t^3 \cdot p_c^{\ 3} \cdot (1-p_c)^{t-3}} \approx \frac{6}{t^2 \cdot p_c^2} \qquad (5)$$

Відповідно, ймовірність ризику p_{13} невірної ідентифікації однократної помилки в запропонованому способі корекції помилок визначається наступним виразом:

$$p_1 \approx \frac{1}{T_1 \cdot 2^{2 \cdot k}} \approx \frac{t^2 \cdot p}{6 \cdot 2^{2 \cdot k}} \qquad (6)$$

Аналогічно, існує ризик невірної ідентифікації двократної помилки при її виправленні у рамках запропонованого методу. Цей ризик полягає у тому, що 4-кратна помилка може

ймовірність p_{24} невірної ідентифікації двократних помилок визначається за формулою:

$$p_{24} \approx \frac{t^2 \cdot p_c^2}{12 \cdot 2^{2 \cdot k}} \qquad (7)$$

Для оцінки числових значень ймовірностей p_{13} і p_{24} ризиків невірної ідентифікації типу помилок можна розглядати типовий приклад передачі в лінії з модуляцією QAM-64 (k=6) значення ймовірностей ризиків невірної

ідентифікації складуть: $p_{13}=1.1\cdot10^{-7}$, а $p_{24}=0.5\cdot10^{-7}$.

Кількість r контрольних розрядів, що використовуються в запропонованому способі виправлення помилок визначається формулою:

$$r = k\cdot(\left\lfloor \log_2 \frac{m}{k} \right\rfloor + 1) \qquad (8)$$

Існуючі способи корекції бітових спотворрень, викликаних двома помками передачі канальних сигналів потребують більшої кількості контрольних розрядів. Так, наприклад, при довжині блоку в 1 Кбайт ($m=2^{13}$) і модуляції QAM-16 (k=4) технологія комбінування россіяння бітових спотворень та корегуючих кодів потребує 84 контрольніх бітів, в той час, як в запропонованому способі – всього 48, тобто в 1.75 раз менше.

Але головна перевага запропонованого способу виправлення двократних помилок в лініях зі спектральною модуляцією у порівнянні з відомими технологіями на основі корегуючих кодів полягає в суттєво меншій обчислювальній складності локалізації спотворених бітів.

Висновки

Запропоновано оснований за використанні зважених контрольних сум метод корекції бітових спотворень, викликаних помилками передачі сигналів в каналах зі спектральною модуляцією. Метод враховує особливості виникнення таких бітових спотворень і дозволяє їх корегувати за умови, що кількість помилок передачі сигналів не перевищує 2-х.

Основним достоїнством запропонованого методу в порівнянні з відомими є суттєво менша обчислювальна складність операцій, пов’язаних з локалізацією бітових спотворень. Це дозволяє підвищити ефективність контролю помилок в швидкісних каналах передачі даних. Крім того, метод потребує меншу кількість контрольних розрядів в порівнянні з відомими.

Запропонований метод може бути ефективно використано для контролю і виправлення помилок в швидкісних кабельних каналах передачі цифрових даних.

Перелік посилань

1. Скляр Б. Цифровая связь. Теоретические основы и практическое применение. М.: Издательский дом ”Вильямс”, 2004.– 1104 с.
2. Дансмор Б., Скандьер Т. Справочник по телекоммуникационным технологиям. М.: Издательский дом ”Вильямс”, 2004.– 640 с.
3. Klove T., Korzhik V. Error Detecting Codes: General Theory and Their Application in Feedback Communication Systems. Norwell, MA: Kluwer, 1995. – 433 p.
4. Самофалов К.Г., Хазем Мохд Саид Абдель Маджид Хатамлех, Антоненко А.А. Модификация контрольной суммы для эффективного контроля ошибок в каналах передачи данных компьютерных сетей. // Проблеми інформатизації та управління. Збірник наукових праць НАУ, – Київ: НАУ – 2006 – Випуск 1(16).– С.134-141.

УДК. 62-50

ПАВЛОВ А. А.,
ЧЕХОВСКИЙ А.В.

СВЕДЕНИЕ ЗАДАЧИ ПОСТРОЕНИЯ МНОГОМЕРНОЙ РЕГРЕССИИ К ПОСЛЕДОВАТЕЛЬНОСТИ ОДНОМЕРНЫХ ЗАДАЧ

Обосновывается возможность сведения задачи построения многомерной полиномиальной регрессии к последовательности одномерных регрессионных задач и решению соответствующих систем алгебраических линейных уравнений с постоянными коэффициентами.

The possibility of bringing of the multidimensional polynomial regression building problem to the sequence of one-dimensional regression problems is grounded in the article. Also the possibility of solving of corresponding systems of algebraic linear equations with constant coefficients is also described in the article.

Проблема эффективного восстановления многомерной полиномиальной регрессии по настоящее время является актуальной [1–9]. Использование нормированных ортогональных полиномов Форсайта [9,10] позволяет предложить эффективный метод построения многомерной полиномиальной регрессии. Он базируется на следующем анализе построения одномерной полиномиальной регрессии с помощью нормированных ортогональных полиномов Форсайта [9]: рассмотрим одномерную модель

$$Y(x) = \theta_0 + \theta_1 x + \ldots + \theta_r x^r + \mathrm{E}, \qquad (1)$$

где x – детерминированная переменная, значение которой в эксперименте исследователь может задавать произвольно. θ_i, $i = \overline{0,r}$ – неизвестные коэффициенты, r может быть избыточно большим. E – случайная величина с произвольным распределением. δ_E^2 неизвестна, либо существует ее верхняя оценка δ^2. По результатам экспериментов (x_i, ny_i, $i = \overline{1,\ }$) нужно найти истинное значение r и оценить значение коэффициентов θ_i, $i = \overline{0,r}$.

Были построены таблицы для оценки дисперсий θ_j, $j = \overline{0,r}$ для случая, когда значения детерминированного аргумента x_i, $i = \overline{1,n}$ равномерно распределены внутри отрезка с концами (–a, a), a >0. Для a >1 анализ таблиц позволил сделать следующие выводы:

Для известной δ^2, можно определить необходимое число испытаний n, чтобы с заданной вероятностью оценка коэффициентов определялась с заданной погрешностью. Например для $a = 50,\ j \geq 2,\ n \geq 10,\quad D\hat{\theta}_j \leq \delta^2 \cdot 7{,}55 \cdot 10^{-6}$. При этом чем больше j тем меньшее число экспериментов гарантирует практически точную оценку неизвестных коэффициентов θ_j. Из этого следует возможность достоверного нахождения истинной степени полинома одномерной регрессии (1) по минимальному числу экспериментов.

Построение многомерной полиномиальной регрессии

Модель имеет вид:

$$\bar{y}(x) = \sum_{\forall(i_1,\ldots i_l)\in k} \sum_{\forall(j_1,\ldots j_l)\in k(i_1\ldots i_l)}^{n} b_{i_1\ldots i_l}^{j_1\ldots j_l} \left(x_{i_1}\right)^{j_1} \times \times \left(x_{i_2}\right)^{j_2} \ldots \times \left(x_{i_l}\right)^{j_l} + E, \qquad (2)$$

где $\bar{x} = (x_1 \ldots x_n)^T$ – детерминированный вектор входных переменных, $b_{i_1\ldots i_l}^{j_1\ldots j_l}$ – неизвестные коэффициенты l – натуральные числа, i_l – натуральные индексы из множества $\{1,\ldots,n\}$. Модель (2) является избыточной. Если возможен активный эксперимент, то задача восстановления многомерной регрессии сводится к последовательности построения одномерных регрессий и решению систем линейных равенств. Порядок вычислений заключается в следующем: последовательно для каждой переменной $x_j\,(j = \overline{1,n})$ фиксируется необходимое количество T_j различных значений переменных x_l, $\forall l(l \neq j)$. Тогда (2) превращается в T_j одномерных регрессий от переменной x_j, найденные коэффициенты которых составляют вектор – столбец правой части j-той $\left(j = \overline{1,n}\right)$системы линейных уравнений. Решение j-той системы линейных уравнений определяет все коэффициенты $b_{i_1\ldots i_l}^{j_1\ldots j_l}$ членов (2), которые содержат x_j и не были определены на предыдущих l этапах $l = \overline{1, j_{-1}}$.

Выводы

В статье предложен и обоснован эффективный метод построения многомерной полиномиальной регрессии. Метод предусматривает реализацию активного эксперимента и базируется на свойствах нормированных ортогональных полиномов Форсайта.

Список литературы

1. Адлер Ю.П., Маркова Е.В., Грановский Ю.В. Планирование эксперимента при поиске оптимальных условий. – 2-е изд., перераб. и доп. – М.: Наука, 1976. –280 С.
2. Айвазян С.А. Многомерный статистический анализ // Математическая энциклопедия / Гл. ред. И.М. Виноградов. – М., 1982. – Т.З. – Стб. 732-738.
3. Андерсон Т. Введение в многомерный статистический анализ / Пер. с англ.
4. Ю.Ф. Кичатова; Под ред. Б.В. Гнеденко. – М.: Физматтиз, 1963. – 500 С.
5. Ершов А. А. Стабильные методы оценки параметров: (Обзор) // Автоматика и телемеханика. – 1978. – № 8. – С. 66-100.
6. Колмогоров А.Н. К обоснованию метода наименьших квадратов // Успехи математических наук. – 1946. – Т. 1, Вып. 1. – С.57-70.
7. Форсайт Дж., Малькольм М., Моулер К. Машинные методы математических вычислений / Пер. с англ. Х.Д. Икрамова. – М.: Мир, 1980. – 280 С.
8. Радченко С.Г. Устойчивые методы оценивания статистических моделей: Монография – К.: ПП «Санспарель», 2005. – 504 С.
9. Норманн Р. Дрейпер, Гарри Смит. Прикладной регрессионный анализ, 3-е изд.: Пер. с англ. – М.: Издательский дом «Вильямс», 2007. – 912 С
10. Д. Худсон. Статистика для физиков. Москва, Мир, 1970.
11. Forsythe G, journ. Sos.Ind. Appl. Math 5,74 (1957).

УДК 007.51:681.518

РОЛИК А.И.
ГЛУШКО Е.В.

АНАЛИЗ КАЧЕСТВА ФУНКЦИОНИРОВАНИЯ ЭЛЕМЕНТОВ ИНФОРМАЦИОННО-ТЕЛЕКОММУНИКАЦИОННЫХ СИСТЕМ

В статье разработаны математические модели определения качества функционирования элементов и подсистем информационно-телекоммуникационной системы. Приведен пример использования предложенных моделей для оценки качества функционирования элемента системы.

The mathematical models of performance estimation for elements and subsystems of information and telecommunications systems have been developed in the paper. An example of applying these models to performance estimation of an element of the system is given.

Введение

Современные АСУ предприятий, учреждений и корпораций нацеливаются, прежде всего, на повышение эффективности бизнес-процессов или процессов деятельности. Для этого в них активно внедряются различные информационные и коммуникационные технологии, а информационно-телекоммуникационная система (ИТС) выступает в качестве базиса ИТ-инфраструктуры АСУ. Современные ИТС представляют собой сложные гетерогенные структуры, в среде которых взаимодействуют разнообразные программные и аппаратные средства.

Для непрерывного предоставления доступа к распределенным ресурсам ИТС, эффективного распределения и закрепления ресурсов между бизнес-процессами или процессами деятельности, повышения доступности сетевых ресурсов и сервисов пользователям и процессам, поддержания высокой работоспособности и готовности ИТС разрабатываются и внедряются системы управления функционированием (СУФ) ИТС [1]. Для решения этих и других задач, а также обеспечения надежного и качественного функционирования ИТС СУФ выполняет мониторинг, анализ функционирования, управление компонентами ИТС и др. функции.

В настоящее время автоматизация функций СУФ в основном ограничивается мониторингом и анализом функционирования ИТС, в то время, как при решении задач управления ИТС зачастую полагаются на опыт и квалификацию администраторов, деятельность которых регламентируется инструкциями и другими руководящими документами. При этом управление сводится к настройкам операционных систем и перезагрузке серверов и рабочих станций, изменению конфигурации активного сетевого оборудования и пр.

Существенного повышения эффективности управления можно достичь, разрабатывая и внедряя в СУФ модели и методы автоматического управления. Это дает возможность строить замкнутые системы управления ИТС с обратной связью, что не только повысит оперативность и качество управления, но и позволит минимизировать негативное влияние человеческого фактора.

В СУФ может быть организовано множество контуров управления. При этом решается широкий спектр задач управления, в частности, задачи управления, ориентированные на поддержание работоспособности ИТС, включающие в себя, например, поддержание качества функционирования элементов и подсистем ИТС, а также и ИТС в целом на заданном уровне, управление надежностью, управление устранением неисправностей и пр., управление оборудованием, управление доступом к ограниченным ресурсам [2–4] и множество других задач.

Для создания замкнутых контуров управления, необходимо получить формальные модели управления, проверить их адекватность и провести исследование, разработать алгоритмы и проверить их работоспособность, найти эффективные методы определения числовых показателей для оценки качества функционирования компонентов ИТС.

Компоненты ИТС можно представить в виде иерархической структуры, на верхнем уровне которой находятся функциональные и технологические подсистемы АСУ, на нижнем – аппаратные и программные элементы. Под-

системы включают в себя аппаратные и программные элементы, поэтому задача определения качества функционирования ИТС будет решаться в такой последовательности — определение качества функционирования элементов ИТС, подсистем, ИТС в целом. Элементы могут входить в состав различных подсистем. Например, на сервере могут выполняться различные серверные приложения, поддерживающие функционирование разных подсистем АСУ. В этом случае состояние аппаратного сервера будет учитываться при определении состояний всех функциональных или технологических подсистем АСУ, элементы распределенных приложений которых работают на данном сервере.

Элементы ИТС могут включать или не включать в себя другие элементы. В последнем случае состояние элемента определяется обработкой значений параметров, непосредственно полученных в результате мониторинга. Оценка качества функционирования подсистем или элементов, содержащих другие элементы, осуществляется путем анализа состояний включаемых элементов с учетом значений параметров функционирования, независимых от работы включаемых элементов, полученных в результате мониторинга рассматриваемых подсистем или элементов.

Решение задач управления, нацеленных на обеспечение функционирования компонентов ИТС и ИТС в целом в соответствии с заданным регламентом или связанные с поддержанием производительности ИТС на заданном уровне, требуют знания состояния, производительности и других характеристик ее структурных компонентов. Поэтому статья посвящена разработке моделей и методов определения качества функционирования компонентов ИТС.

Анализ существующих методов оценки качества функционирования элементов и подсистем ИТС

Для эффективного управления ИТС необходимо располагать информацией о состоянии элементов и подсистем. Эта информация может быть получена в результате мониторинга и анализа функционирования. Мониторинг параметров функционирования обеспечивает сбор первичной информации о состоянии различных программных и аппаратных элементов ИТС. Эти данные являются основой для дальнейшего анализа и определения значений целевых функций, показателей качества функционирования составных элементов и подсистем, надежности и пр. Мониторинг состояния может производиться с помощью встроенных в вычислительные системы средств диагностики, например, для операционной системы Windows это могут быть инструментарий управления Windows (Windows Management Instrumentation, WMI) или счетчики производительности (Performance Counters). Мониторинг также может осуществляться с помощью протоколов SNMP, CMIP и других средств, например, посредством использования агентских технологий [5].

Для оценки качества функционирования IP сетей разработаны стандартные метрики [6—7], стандартизованы и методы проведения испытаний для определения значений этих метрик. Работу IP сети можно оценить по малому количеству показателей, например, задержка передачи пакетов, частота ошибок в пакетах и пр., поэтому количество стандартных метрик сильно ограничено.

В [8] предложена система метрик для оценки работы провайдера сетевых услуг, организованная в форме дерева метрик. Там же рассмотрены вопросы корреляции метрик. Однако, рассмотрение ограничивается только метриками для различных сетевых технологий.

В масштабных ИТС используются не только коммуникационные технологии, но и широкий спектр информационных технологий, а на работу элементов оказывает влияние большое количество параметров, которые, несмотря на то, что их достаточно просто измерить, сложно обобщить в универсальной математической модели, пригодной для решения многочисленных задач управления. Поэтому целесообразно двигаться по пути создания множества простых моделей, отражающих отдельные аспекты функционирования элементов и подсистем ИТС и пригодных для решения отдельных или небольшого числа задач управления.

Целью данной статьи является рассмотрение методов анализа качества функционирования подсистем и структурных элементов ИТС и разработка математических моделей оценки качества функционирования элементов, подсистем и ИТС в целом.

Модели и алгоритмы анализа качества функционирования элементов и подсистем ИТС

Рассмотрим подход к определению качества функционирования элементов ИТС, который также является основой для определения качества функционирования подсистем ИТС и ИТС в целом.

Для оценки качества функционирования элемента необходимо проанализировать происходящие в нем процессы, определить параметры, характеризующие состояние и работу этих процессов, сопоставить значения множества параметров оценке качества функционирования элемента и формализовать характеристику, сводящую параметры, характеризующие важные аспекты функционирования элемента в единый показатель. В качестве такой характеристики предлагается использовать интегральный показатель качества функционирования (ИПКФ) элемента, принимающий значения в диапазоне [0, 1].

Вычисление ИПКФ сложных (составных) элементов и подсистем производится на основе ИПКФ компонентов, из которых они состоят. Естественно, что ИПКФ сложного элемента выражается не только через ИПКФ составляющих его элементов, но зависит и от собственных параметров качества функционирования.

Параметры можно разделить на измеряемые, вычисляемые и конфигурационные. Значения измеряемых и конфигурационных параметров можно получить с помощью встроенных диагностических средств операционных систем, например, счетчиков производительности и WMI, посредством протоколов управления – SNMP, CMIP и пр. или используя разработанные диагностические средства (тестовое ПО), возвращающие численные или логические значения параметров, характеризующих отдельный аспект поведения элемента. Например, приложение, генерирующее серию тестовых запросов к СУБД и возвращающее суммарное или среднее время их обработки.

Параметры могут быть численными и логическими. Пример численного параметра – объем свободного пространства на жестком диске, логического – параметр, фиксирующий наличие определенного приложения в памяти сервера. В целях унификации логические параметры приводятся к численным по следующему принципу – логическому значению «истина» присваивается числовое значение 1, а значению «ложь» – 0.

Конфигурационные параметры обычно остаются неизменными во время работы элемента. Новые значения они приобретают в случае переконфигурации элемента, например, после установки дополнительной планки ОЗУ увеличивается объем оперативной памяти. Конфигурационные параметры используются при определении значений вычисляемых параметров, а самостоятельного влияния на качество функционирования элемента не оказывают. Например, размер оперативной памяти сервера используется для определения доли свободной оперативной памяти сервера в процессе его функционирования.

Значения вычисляемых параметров определяются аналитически на основании значений одного или нескольких других параметров. Вычисляемые параметры являются простыми, когда их значение вычисляется на основании значений одного параметра и сложными, когда они вычисляются на основе значений нескольких параметров. Значения вычисляемых параметров могут быть получены в результате выполнения простых (нормирование, вычисление среднего) и достаточно сложных математических операций (например, корреляция значений нескольких параметров).

К простым вычисляемым параметрам относятся:

среднее значение параметра за период T_i. Период для каждого параметра устанавливается индивидуально в зависимости от его свойств во временной области и может изменяться администратором СУФ;

- нормированное значение параметра – приведение значений параметра к диапазону [0, 1];
- среднее нормированное значение параметра за период T_i.

Нормированные значения параметров удобны при применении библиотеки унифицированных функциональных зависимостей для определения оценочного коэффициента и коэффициента критичности параметра, о чем речь пойдет ниже.

Для большинства параметров использование мгновенного значения при анализе состояния не имеет смысла. Так, например, мгновенное значение параметра «загружен-

ность процессора» не несет достоверной информации о реальной загруженности процессора. Этот параметр в момент загрузки приложения в память принимает максимальное значение, а пики в загруженности процессора — нормальное явление, не являющееся сигналом о проблемах с производительностью вычислительной системы. В таких случаях для вычисления ИПКФ целесообразно использовать среднее за период значение.

Измерение значений для вычислений параметров производятся через фиксированный интервал времени, который задается индивидуально для каждого параметра и может изменяться администратором СУФ ИТС. Для упрощения работы планировщика СУФ интервалы опроса параметров могут группироваться и выбираться из фиксированного ряда. Кроме того, планировщик производит сбор значений параметров таким образом, чтобы равномерно распределить нагрузку на каналы связи.

Для каждого элемента ИТС необходимо определить множество $P = \{p_1, p_2, ..., p_n\}$ параметров, которые оказывают существенное влияние на качество его функционирования. Формирование множества P производится на основе анализа функционального назначения элемента, условий его функционирования, статистических данных о функционировании элементов и других факторов. Решение этой проблемы выходит за рамки данного рассмотрения.

Задачу определения E – численного значения ИПКФ можно свести к определению функциональной зависимости:

$$E = f(P). \quad (1)$$

Текущее значение ИПКФ предлагается определять следующим образом:

$$E(t) = \frac{[\sum_{i=1}^{n} k_i \cdot m_i(p_i(t),t)] \cdot \prod_{i=1}^{n} s_i(p_i(t),t)}{\sum_{i=1}^{n} k_i}, \quad (2)$$

где $p_i(t)$ – значение i-го параметра в момент времени t, k_i – весовой коэффициент i-го параметра, $m_i(p_i(t),t)$ – оценочная функция i-го параметра, $s_i(p_i(t),t)$ – функция критичности i-го параметра.

Значения ИПКФ, вычисленные по (2) принимают значения из промежутка [0, 1]. При этом максимальное значение, равное единице, соответствует безукоризненному функционированию элемента ИТС, а минимальное, равное нулю, может означать, что элемент не работоспособен.

Ключевыми понятиями в выражении (2) являются оценочная функция и функция критичности. Суть их состоит в следующем.

Из множества P выделяется подмножество критичных параметров, значения которых, принадлежащие некоторому диапазону, свидетельствуют о наличии серьезных проблем в элементе, которые могут существенно повлиять на его работу, вплоть до полного прекращения элементом выполнения своих функций. Такого рода параметры являются своеобразными показателями качества функционирования системы. Примером такого параметра может служить, количество свободного места на разделе жесткого диска рабочей станции, работающей под управлением Windows XP, на котором хранится файл подкачки. При недостаточном объеме памяти на этом разделе система может аварийно завершить работу, если она не сможет выделить необходимую память процессам по их запросу. В таких случаях СУФ выдает сигнал администраторам о необходимости устранения проблемы путем освобождения дискового пространства, наращивания оперативной памяти или увеличения объема дискового пространства раздела. Для таких параметров, которые влияют на значение показателя качества функционирования элемента в целом, функция критичности s_i принимает значения из диапазона [0, 1]. Для некритичных параметров s_i устанавливается равной единице.

Другой пример критичного параметра – наличие в памяти рабочей станции или сервера программных компонентов распределенного приложения. Естественно, что если приложение не запущено на сервере, то приложение полностью не работает, а если компоненты распределенного приложения не запущены на рабочей станции, то она не выполняет возложенных на нее приложением функций.

В функцию критичности $s_i(p_i(t),t)$ i-го параметра введена зависимость от времени, что позволяет компенсировать реакцию СУФ на негативное воздействие, например, проводимых в ИТС регламентных работ. На серверах

и рабочих станциях по расписанию могут запускаться антивирусные пакеты, производиться резервное копирование и другие регламентные работы. Эти работы изменяют режим функционирования элемента ИТС. Кроме того, некоторые элементы ИТС могут не работать на протяжении заведомо известных интервалов времени, например, рабочие станции ночью и в выходные дни. Учет этих событий в функции критичности позволяет блокировать в СУФ выдачу аварийных сигналов.

Параметры множества *P* являются некритичными, если при их изменении во всем диапазоне возможных значений не происходит значительного изменения качества функционирования элемента ИТС. В формуле определения ИПКФ некритичные параметры учитываются как взвешенная сумма значений их оценочных функций. Каждый из таких параметров определяет качество функционирования элемента в определенном аспекте. Оценочная функция $m_i \in [0,1]$ некритичного параметра зависит от текущего значения параметра и времени. Зависимость от времени введена из тех же соображений, что и для критичных параметров.

Значимость *i*-го параметра определяется весовым коэффициентом k_i. Для некоторых критичных параметров $k_i \equiv 0$, а расчет m_i не производится.

Оценочную функцию и функцию критичности *i*-го параметра можно задавать аналитически или в виде таблиц. Формы этих зависимостей для большинства параметров будут схожими, поэтому для использования библиотек стандартных функций с некоторым количеством настраиваемых атрибутов необходимо использовать нормированные значения параметров.

ИПКФ элемента используется для определения ИПКФ элементов вышерасположенного уровня иерархии, поэтому он также нормируется делением на сумму $\Sigma\ k_i$ для приведения его значений в диапазон [0, 1].

Для вычисления ИПКФ подсистем и ИТС в целом строится дерево зависимости элементов и параметров. Листьями самого нижнего уровня являются исключительно параметры, а верхних уровней – элементы и параметры или подсистемы и параметры. Каждый параметр измеряется или вычисляется через определенный интервал времени. Вычисления производятся независимо и используют текущие значения параметров и ИПКФ элементов нижнего уровня. Модель предполагает наличие событий, которые приводят к апериодическому пересчету значений параметров дерева. Такие события инициируются при достижении значения отдельного параметра или ИПКФ порогового уровня.

Пример дерева зависимостей параметров и элементов приведен на рис. 1. Количество штрихов в соответствующем обозначении указывает на уровень, на котором находится компонент в иерархии. Для упрощения обозначений количество компонентов условно обозначено через *n*, которое в общем случае имеет разное значение для различных уровней. На каждом уровне раскрыт только один компонент.

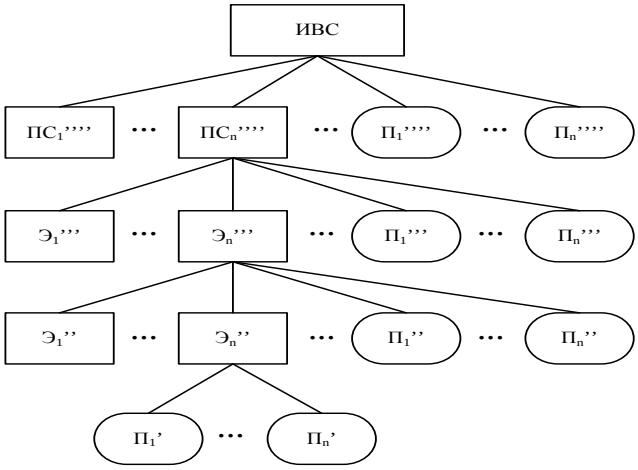


Рис. 1. Дерево зависимостей параметров и элементов. *ПС – подсистема, Э – элемент, П – параметр*

Главным достоинством такого подхода является его гибкость и возможность использования в ИТС различного назначения. При этом на каждом уровне входными данными для определения ИПКФ компонентов являются ИПКФ компонентов смежного нижнего уровня и значения оригинальных параметров. Причем параметры могут иметь различный смысл и тип, а также быть получены различными способами – это могут быть результаты тестирования или моделирования, данные, полученные в результате прогнозирования, статистического анализа, вычислений на основе нечетких моделей, моделей искусственного интеллекта и т. д. Это позволяет использовать предлагаемый подход для анализа качества функционирования сложных, иерархических, гетерогенных ИТС.

Обобщим выражение (2) для определения ИПКФ составного элемента или подсистемы с учетом того, что для определения ИПКФ в этом случае необходимо учитывать не только ИПКФ составных частей, но и значения оригинальных параметров.

Пусть, на функционирование l-го элемента оказывает влияние состояние k элементов (подэлементов) и значения m параметров. Тогда выражение для определения ИПКФ $E^{(l)}$ составного l-го элемента будет иметь вид:

$$E^{(l)} = [\frac{\sum_{i=1}^{k} K_i^{(l)} \cdot M_i^{(l)}(E_i^{(l)}(t),t) +}{\sum_{i=1}^{k} K_i^{(l)} + \sum_{i=1}^{m} k_i^{(l)}} + \frac{\sum_{i=1}^{m} k_i^{(l)} \cdot m_i^{(l)}(p_i^{(l)}(t),t)}{\sum_{i=1}^{k} K_i^{(l)} + \sum_{i=1}^{m} k_i^{(l)}}] \times \times \prod_{i=1}^{k} S_i^{(l)}(E_i^{(l)}(t),t) \cdot \prod_{i=1}^{m} s_i^{(l)}(p_i^{(l)}(t),t), \quad (3)$$

где $p_i^{(l)}(t)$ – значение i-го параметра l-го элемента в момент времени t, $k_i^{(l)}$ – весовой коэффициент для i-го параметра, $m_i^{(l)}(p_i(t),t)$ – оценочная функция i-го параметра, $s_i^{(l)}(p_i(t),t)$ – функция критичности i-го параметра, $E_i^{(l)}(t)$ – значение интегрального показателя качества функционирования i-го подэлемента в момент времени t, $K_i^{(l)}$ – весовой коэффициент i-го подэлемента, $M_i^{(l)}(p_i(t),t)$ – оценочная функция i-го подэлемента, $S_i^{(l)}(p_i(t),t)$ – функция критичности i-го подэлемента.

В качестве примера рассмотрим возможный набор параметров, функций и коэффициентов для определения качества функционирования такого распространенного элемента ИТС, как рабочая станция.

Здесь рабочая станция рассматривается как простой элемент и качество ее функционирования определяется только на основе значений параметров. Рабочая станция является совокупностью взаимодействующих аппаратных и программных элементов, а функционирование программных средств влияет на значения многих аппаратных параметров. Однако, для простоты изложения ограничимся рассмотрением только аппаратных параметров, анализ значений которых позволяет определить, насколько аппаратное обеспечение справляется с нагрузкой, создаваемой совокупностью выполняемых системных и прикладных программ.

В первую очередь, необходимо выбрать множество параметров и определить для каждого из них интервал времени Δt_i, через который проводятся измерения мгновенных значений, определить k_i, вид функциональной зависимости оценочной функции и функции критичности.

Интервалы и форма зависимостей определяются, исходя из сущности параметра и его влияния на функционирование системы.

Для определения ИПКФ необходимо отобрать наиболее важные параметры элемента, причем количество параметров должно быть не очень большим, чтобы не затруднять расчет ИПКФ и не малым, чтобы полностью учесть специфику работы элемента в ИТС.

В качестве набора параметров предлагается использовать следующие:

- средняя загруженность процессора за период, p_1 – доля времени, в течение которой процессор выполняет программный код системы или приложений;
- использование памяти, p_2 – относительное количество свободной физической памяти в системе;
- использование дискового пространства, p_3 – относительное количество свободного места на физическом диске;
- загруженность диска, p_4 – относительное количество времени, которое жесткий диск занят обработкой запросов ввода/вывода.

Для определения ИПКФ можно выбрать и другой набор параметров, исходя из специфической роли рабочей станции в ИТС. Выбор конкретного множества параметров является достаточно сложной задачей, и ее решение выходит за рамки данной статьи. Данный выбор параметров основан на наиболее существенных параметрах, характеризующих работу важнейших компонентов рабочей станции: процессора, памяти и жесткого диска, и является универсальным.

Выбираются значения Δt_i и T_i. В рассматриваемом случае эти значения одинаковы для всех параметров. В целях упрощения изложения, весовые коэффициенты k_i каждого параметра принимаются равными 1.

Средняя загруженность процессора за период T_1. Этот параметр является простым вычислимым параметром. Для его расчета используется параметр «мгновенное значение загруженности процессора» p_1', замеряемый на протяжении периода T_1 с интервалом Δt_i. Значение параметра рассчитывается по формуле:

$$p_1(t) = \frac{1}{N_1} \sum_{i=0}^{N_1-1} p_1'(t-i) \cdot t_1 , \qquad (4)$$

где $N_1 = \frac{T_1}{\Delta t_1}$.

Значения параметра лежат в диапазоне [0,1].

Данный параметр считается некритичным, поскольку рабочая станция может выполнять возложенные на нее функции даже при полной загруженности процессора в течение длительных временных интервалов, поэтому s_1=1, а форма функции критичности приведена на рис. 2б.

При выборе оценочной функции (рис. 2а) учитывается, что чем больше значение параметра, тем хуже функционирует рабочая станция.

Для того, чтобы не усложнять пример, в статье не учитываются временные зависимости оценочной функции и функции критичности, а используются простейшие кусочно-линейные функции.

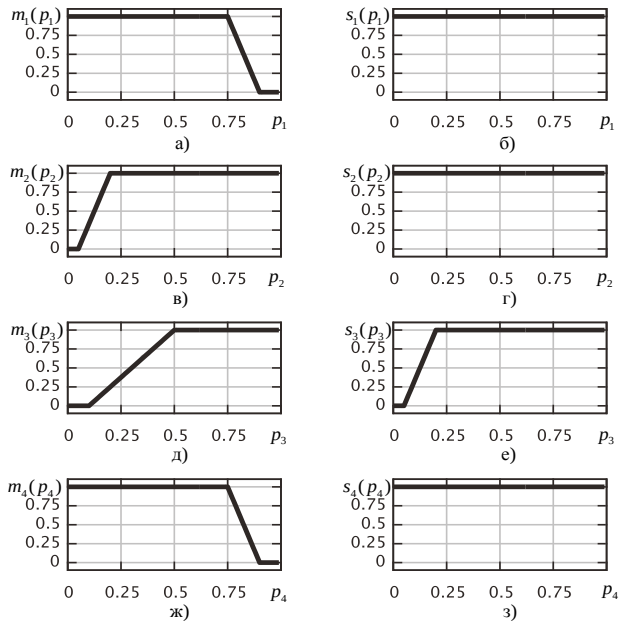


Рис. 2. Оценочные функции и функции критичности для параметров p_1–p_4

Использование памяти. Данный параметр является вычислимым и сложным, а его значение определяется по формуле:

$$p_2(t) = \frac{M(t)}{M}, \qquad (5)$$

где $M(t)$ – свободный объем физической памяти в текущий момент времени в байтах, M – объем физической памяти рабочей станции в байтах.

Значения параметра лежат в диапазоне [0, 1]. Данный параметр является некритичным: s_2=1, а вид зависимости приведен на рис. 2г.

На рис. 2в приведена форма оценочной функции.

Использование дискового пространства. Параметр является сложным и вычисляется по формуле:

$$p_3(t) = \frac{D(t)}{D}, \qquad (6)$$

где $D(t)$ – свободный объем дисковой памяти в текущий момент времени в байтах, D – объем физического диска в байтах.

Данный параметр является критичным. Форма функции критичности приведена на рис. 2д, а на рис. 2е представлена выбранная форма оценочной функции.

Загруженность диска за период T_4. Этот параметр является простым вычислимым параметром. Для его расчета используется параметр «мгновенное значение загруженности диска» p_4' (область значений параметра [0, 1]), замеряемый на протяжении периода T_4 с интервалом Δt_4. Значение параметра рассчитывается по формуле:

$$p_4(t) = \frac{1}{N_4} \sum_{i=0}^{N_4-1} p_4'(t-i) \cdot t_4 , \qquad (7)$$

где $N_4 = \frac{T_4}{\Delta t_4}$.

Значения параметра лежат в диапазоне [0, 1].

Данный параметр является некритичным: s_4=1. Форма функции критичности для этого параметра приведена на рис. 2з.

При выборе оценочной функции учитывается, что чем больше значение параметра, тем хуже функционирует рабочая станция, форма оценочной функции параметра приведена на рис. 2ж. Предложенная модель использована для исследования функционирования реальной рабочей станции. При моделировании значения весовых коэффициентов k_1, k_2, k_3, k_4 были взяты равными 1; продолжительность моделирования – 120 мин.; интервал Δt_i для всех параметров выбран одинаковый и равный

1 мин.; значения параметров p_1 и p_4 усреднялись по последним 10 значениям (T_i=10 мин.); в качестве оценочных функций и функций критичности использованы зависимости, приведенные на рисунке 2а – 2з; вычисление ИПКФ производилось каждую минуту.

Результаты моделирования приведены на рис. 3. На рис. 3а – 3г показано изменение значений параметров p_1, p_2, p_3, p_4 во времени, а на рис. 3д – изменение значения ИПКФ.

Снижение ИПКФ в диапазоне 80 – 100 мин. обусловлено увеличением загруженности процессора и дисковой подсистемы, вызванное запуском приложения, потребляющего значительные ресурсы.

Выводы

В статье предложено оценивать работу элементов и подсистем ИТС по интегральному показателю качества функционирования. Предложены выражения для определения ИПКФ простого и составного элементов. Получаемые в результате применения моделей численные оценки упрощают построение замкнутых систем управления ИТС и позволяют автоматизировать процесс управления сложными гетерогенными системами.

Дальнейшим направлением исследований является реализация средств анализа функционирования компонентов ИТС, использующих предложенные модели, разработка моделей определения числовых показателей, характеризующих надежность элементов и подсистем, а также разработка моделей целевых функций и исследования в области построения контуров управления сложными гетерогенными ИТС.

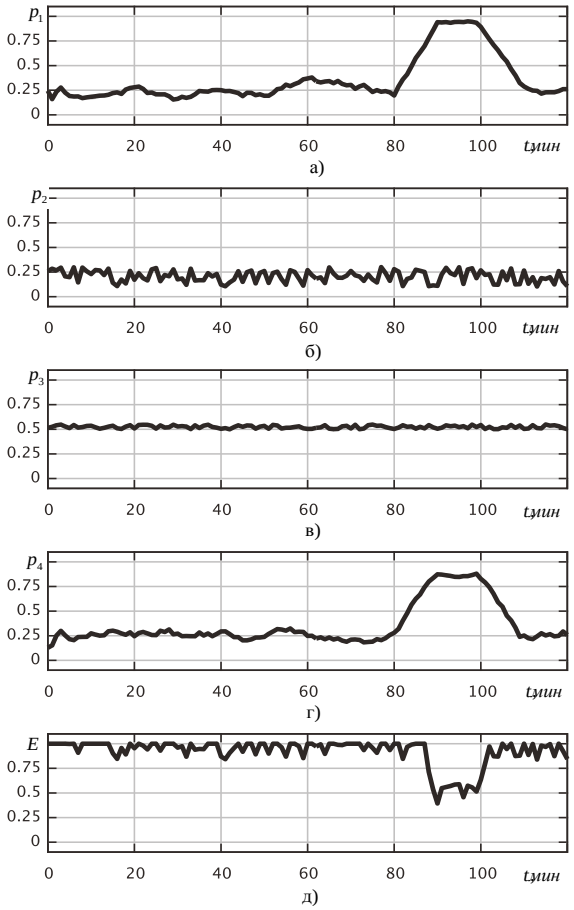


Рис. 3. Результаты моделирования рабочей станции

Список литературы

1. 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. –2006. – № 45. – С. 112–126.
2. Ролик А.И. Модель управления перераспределением ресурсов информационно-телекоммуникационной системы при изменении значимости бизнес-процессов// Автоматика. Автоматизация. Электротехнические комплексы и системы. ХГТУ, 2007. –№2 (20). – С. 73–82.
3. Теленик С.Ф., Ролік О.І., Букасов М.М. Моделі управління розподілом обмежених ресурсів в інформаційно-телекомунікаційній мережі АСУ// Вісник НТУУ «КПІ». Інформатика, управління та обчислювальна техніка. – 2006. – № 44. – С. 234–239.
4. Теленик С.Ф., Ролік О.І., Букасов М.М., Терещенко П.І. Управління доступом до обмежених ресурсів інформаційно-телекомунікаційної мережі АСУ військового призначення// Сб. наук. праць ЦНДІ Збройних Сил України. – 2006. – №3 (37). – С. 33–43.
5. Ролик А.И., Соколовский Р.Л. Распределение мобильных компонентов системы управления информационно-телекоммуникационной системой// Вісник НТУУ «КПІ». Інформатика, управління та обчислювальна техніка. –2007. – № 47. – С. 113–124.
6. Paxson V., Almes G., Mahdavi J., Mathis M. Framework for IP performance metrics. RFC2330. May, 1998.
7. Mahdavi J., Paxson V. IPPM Metrics for Measuring Connectivity. RFC2678. September, 1999.
8. Boutaba R., Xiao J., Aib I. CyberPlanner: A Comprehensive Toolkit For Network Service Providers. NOMS 2008 – 11th IEEE/IFIP Network Operations and Management Symposium, vol. 11, no. 1, pp. 379–386.

З М І С Т

Павлов О.А.,Місюра О.Б.,Мельников О.В. Дослідження властивостей та розв'язання задачі «Мінімізація сумарного штрафу як за випередження, так і за запізнення відносно директивних строків при виконанні незалежних завдань одним приладом»	3
Парамонов А.М., Томашевский В.Н. Формализация алгоритма моделирования движения автомобильного дорожного транспота	7
Марковський О.П., Мазур Р.Ф., Саїд Реза Мах Малі. Підвищення ефективності ідентифікації абонентів з використанням моделі персептрону	13
*Богушевська Н.В., Томашевський В.М.*Формальна модель кільцевої транспортної мережі. Реалізація перетворення діалогової нотації до програмного коду	19
Стіренко С.Г. Формальний опис структурно-параметричних характеристик технічного тексту	25
Павлов А.А., Кут В.И., Штанькевич А.С. Нахождение весов по матрице парных сравнений с односторонними ограничениями	29
*Теленик С.Ф., Бідюк П.І., Амонс О.А., Крижова К.О.*Метод розподілу ресурсів між проектами	33
Іванов В.О. Розподілена система імітаційного моделювання дорожнього руху	41
Павлов А.А., Иванова А.А. Алгоритм идентификации функции цели в методе анализа иерархий	46
Буханцов С.Н., Пустоваров В.І., Стіренко С.Г., Желудкова Т.В. Доказове управління генерацією та модифікацією програмних модулів для інформаційно-аналітичних систем	49
Блінова Т. О. Додавання та множення в кодах золотої пропорції в порозрядному режимі, починачи зі старших розрядів	54
Садег Растгу. Модели вычислительных процессов в кластерных системах с многоядерною архитектурою	58
Павлов А.А., Мисюра Е.Б., Кут В.И., Щербатенко О.В., Михайлов В.В. Математические модели иерархического планирования и принятия решений	63
Кузнецов А.В. Представление знаний в концепции объектно-ориентированного логического программирования	67
Романкевич В.А. Некоторые количественные оценки в *GL*-моделях	72
Павлов А.А., Лищук Е.И. Оперативные алгоритмы принятия решений в иерархической системе Саати, основанные на замещении критериев	78
Мєлковська В.М. Сергієнко А.М., Стіренко С.Г. Спосіб демодуляції сигналів з багаточастотною модуляцією	82
Павлов О.А., Місюра., Халус О.А., Костик Д.Ю., Лисецький Т.М. Оптимізація модифікованого ПДС-алгоритму задачі мінімізації сумарного запізнення виконання завдань	85
Зайченко Ю.П., Медін М.Ю. Метод вирішення задачі аналізу ситуацій системою підтримки прийняття рішення для програмування	89
Безносик А.Ю. Построение эквивалентных моделей балочных структур	94
Ладогубец В.В., Крамар А.В., Финогенов А.Д. Адаптация параллельного алгоритма СПУИП для кластера НТУУ «КПИ»	99
Марковський О.П.,Федоречко О.І., Турченко Ю.О. Корекція помилок передачі даних в каналах зі спектральною модуляцією з використанням зважених контрольних сум	104
*Павлов А. А.,Чеховський А.В.*Сведение задачи построения многомерной регрессии к последовательности одномерных задач	111
*Ролик А.И., Глушко Е.В.*Анализ качества функционирования элементов информационно-телекоммуникационных систем	113

Відомості про авторів

Амонс Олександр	к.т.н., старший викладач кафедри АУТС
Безносик А.Ю.	асистент, каф. СП УНК ИПСА НТУУ «КПИ»
Бідюк Петро Іванович	д.т.н., професор Інституту прикладного та системного аналізу
Блінова Т. О.	канд. техн. наук, кафедра обчислювальної техніки НТУУ ”КПІ”
Богушевська Наталія Володимирівна	старший викладач кафедри АСОІУ факультету ІОТ НТУУ “КПІ”. E-mail: tasha@asbase.kiev.ua
Буханцов С.Н.,	аспірант кафедри обчислювальної техніки НТУУ ”КПІ”.
Глушко Евгений Васильевич	асистент кафедри АУТС НТУУ «КПІ»
Желудкова Т.В.	студентка, кафедра обчислювальної техніки НТУУ ”КПІ”
Зайченко Юрий Петрович	д.т.н., профессор кафедры ММСА НТУУ «КПИ» УНК «ИПСА». E-mail: zaych@i.com.ua
Иванова Анна Анатолиевна	Аспирантка каф. E-mail: a.a.ivanova@gmail.com
Іванов В.О.	аспірант каф. АСОІУ НТУУ «КПІ» E-mail: v.o.ivanov@gmail.com
Костик Д.Ю.	
Крамар А.В.	снс, каф. СП УНК ИПСА НТУУ «КПИ», E-mail: kramar@cad.ntu-kpi.kiev.ua
Крижова Ксенія Олександрівна	асистент кафедри АУТС
Кузнецов А.В.	кафедра обчислювальної техніки НТУУ ”КПІ”
Кут В.И	кафедра АСОИУ НТУУ ”КПІ”
Ладогубец В.В.	доцент, каф. СП УНК ИПСА НТУУ «КПИ», E-mail: ladog@ntu-kpi.kiev.ua
Лисецький Т.М.	студент кафедры АСОИУ НТУУ ”КПИ”
Лищук Е.И.	ассистент, кафедра АСОИУ НТУУ «КПИ»
Мазур Роман Федорович	студент, кафедра обчислювальної техніки НТУУ ”КПІ”.
Марсковський Олександр Петрович	кандидат технічних наук, кафедра обчислювальної техніки НТУУ ”КПІ”. E-mail: markovskyy@mail.ru
Медін М.Ю.	аспирант, ММСА НТУУ «КПИ» УНК «ИПСА»-
Мельников О.В	канд. техн. наук, ст. наук. спів., кафедра АСОІУ НТУУ
Мєлковська В.М.	
Михайлов В.В.	канд. техн. наук, ст. наук. спів., кафедра АСОІУ НТУУ ”КПІ”
Місюра О.Б.	канд. техн. наук, ст. наук. спів., кафедра АСОІУ НТУУ ”КПІ”
Павлов Александр Анатолиевич	д.т.н., проф. каф. АСОИУ НТУУ «КПИ» E-mail: compscience@asu.ntu-kpi.kiev.ua
Парамонов Антон Михайлович	магистр кафедры АСОИУ факультета ИОТ НТУУ «КПИ”. E -mail: paramonov-86@mail.ru.
Пустоваров В.І.	канд. техн. наук, кафедра обчислювальної техніки НТУУ ”КПІ”
Ролик Александр Иванович	канд. техн. наук, доцент, кафедра АУТС НТУУ «КПІ»
Романкевич В.А.	

Садег Растгу.	аспирант кафедры вычислительной техники
Саїд Реза Мах Малі	аспірант, кафедра обчислювальної техніки НТУУ ”КПІ”.
Сергієнко А.М.	канд. техн. наук, ст. наук. спів., кафедра обчислювальної техніки НТУУ ”КПІ”
Стіренко Сергій Григорович	канд. техн. наук, кафедра обчислювальної техніки НТУУ ”КПІ”
Теленик Сергій Федорович	д.т.н., професор кафедри АУТС
Томашевський Валентин Миколайович	професор, доктор технічних наук, професор кафедри АСОІУ факультету ІОТ НТУУ “КПІ”. E-mail: simtom@svitonline.com
ТурченкоЮ.О.	
Федоречко Ольга Іванівна	студентка, кафедра обчислювальної техніки НТУУ ”КПІ”.
Финогенов А.Д.	ассистент, каф. СП УНК ИПСА НТУУ «КПИ». E-mail: fenya@cad.ntu-kpi.kiev.ua
Халус О.А.	кафедра АСОИУ НТУУ «КПІ»
Чеховський А.В.	студент, кафедра АСОІУ факультету ІОТ НТУУ “КПІ”.
Штанькевич А.С.	студент, кафедра АСОІУ факультету ІОТ НТУУ “КПІ”.
Щербатенко О.В.	канд. техн. наук, ст. наук. спів., кафедра АСОІУ НТУУ ”КПІ”

ЛЮБЧЕНКО К. М.

ЕКСПЕРТНІ СИСТЕМИ В ПРАКТИЧНІЙ МЕДИЦИНІ

Експертні системи є одним з найпоширеніших типів систем штучного інтелекту. Вони розроблялися як науково-дослідні інструментальні засоби з 1960-х років і розглядалися як штучний інтелект спеціального типу, призначеного для успішного вирішення складних задач у вузькій предметній галузі, такий як медична діагностика захворювань [2, 28]. Найбільш широке поширення експертних систем в різних областях людської діяльності почалося на початку 80-х років XX століття.

Оскільки одними з перших експертних систем, які мають попит і зараз, є медичні, то основною метою даної статті є огляд в історичній ретроспективі цього класу експертних систем і визначення перспектив його подальшого розвитку.

Сформульована мета передбачає розгляд наступних питань:

- аналіз поняття "Експертна система";
- приклади медичних діагностичних експертних систем, їх класифікація;
- вимоги сучасного підходу до оцінки стану здоров'я людини;
- постановка завдання створення експертної системи для біорезонансної діагностики.

Приведемо декілька класичних визначень експертних систем і проаналізуємо їх.

Розробник медичної експертної системи MYCIN Э. Фейгенбаум із Станфордського університету визначив поняття експертної системи як "... інтелектуальної комп'ютерної програми, в якій використовуються знання і процедури логічного виводу для вирішення завдань, досить важких для того, щоб вимагати для свого вирішення значного обсягу експертних знань людини" [2, 33].

П. Джексон в [3, 19] дає наступне формальне визначення: "Експертна система – це програма для комп'ютера, яка оперує із знаннями в певній предметній галузі з метою вироблення рекомендацій або вирішення проблем".

Експертна система – це комп'ютерна програма, яка в деякій галузі виявляє ступінь пізнань рівнозначну ступеню пізнання людини-експерта. Зазвичай ця галузь строго обмежена [4, 406].

Експертні системи – це складні програмні комплекси, що акумулюють знання фахівців в конкретних предметних галузях і тиражують цей емпіричний досвід для консультацій менш кваліфікованих користувачів [1, 39].

Автори [6, 32] відзначають: реалізація концепції про те, що ефективність комп'ютерної програми при вирішенні задач залежить від знань, якими вона володіє, а не лише від формалізмів і схем виводу, які вона використовує, привела до розвитку спеціалізованих програмних систем, кожна з яких є "експертом" в деякій вузькій предметній галузі – ці програми отримали назву експертних систем.

Основні етапи процесу розробки експертної системи представлені на рис. 1 [2, 41].

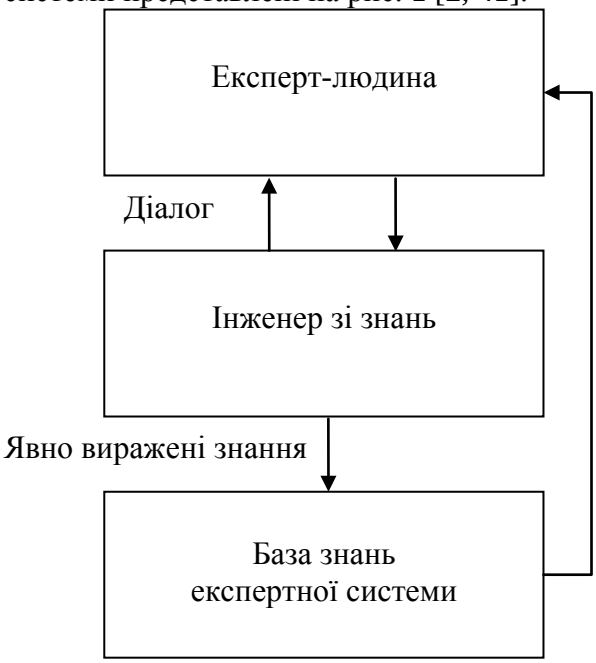


Рис. 1. Процес розробки експертної системи

Таким чином, можна зробити висновок про те, що експертні системи відрізняються від інших систем штучного інтелекту наступними основними ознаками:

1) орієнтованість на вузькоспеціалізовану предметну галузь;
2) вони розробляються для вирішення задач, які підходять для людини-експерта;
3) здатність пояснювати свої рішення зрозумілим користувачеві образом.

Ці ознаки в тій чи іншій мірі простежуються в медичних експертних системах, перші суттєві розробки яких відносяться до початку 1970-х років. Застосування експертних систем в медицині найефективніше при вирішенні задач діагностики, інтерпретації даних, прогнозуванні перебігу захворювань і ускладнень, моніторингу перебігу захворювання і

планування лікувально-діагностичного процесу. В наш час побудова ефективних експертних систем різного призначення, у тому числі, медичних систем, залишається завданням актуальним і перспективним. Пройдений лише перший відрізок довгого шляху, труднощі подолання якого обумовлені, з одного боку, надзвичайною складністю об'єкту, що моделюється з використанням експертних систем – мозку людини, а з іншого боку високою складністю завдань, вирішення яких покладається на медичні експертні системи [6, 8].

Одна з перших діагностичних експертних систем MYCIN призначена для роботи в галузі діагностики і лікування зараження крові і медичних інфекцій. На підставі симптомів система видає декілька діагнозів із зазначенням відповідних коефіцієнтів визначеності і пропонує курс лікування виявленої інфекції. База знань MYCIN містить близько 500 правил, розроблених за допомогою групи з інфекційних захворювань Станфордського університету.

На основі обчислених коефіцієнтів визначеності для кожного діагнозу користувачеві пред'являється список можливих результатів.

Успіх MYCIN спричинив створення на її основі оболонки EMYCIN і цілого ряду нових експертних систем. Наприклад, NEOMYCIN, в якій була спроба використовувати більш абстрактний підхід до вирішення медичних проблем, ніж у прототипі, допомагає лікарям діагностувати і лікувати менінгіт і схожі з ним захворювання. При розробці системи було надано значної уваги моделюванню того методу вирішення проблеми, якій властивий лікареві (когнітивне моделювання). Тому в даному програмному продукті знання представлені в такий спосіб, який полегшує пояснення і навчання. Ключова відмінність між MYCIN і NEOMYCIN – це явне відокремлення діагностичної процедури від знань про хвороби.

Експертна система INTERNIST, яка розроблена в Піттсбургськом університеті, діагностує декілька сотень різноманітних хвороб на рівні кваліфікованого лікаря. Складні діагнози внутрішніх захворювань, що виводяться системою, визначаються на підставі історії хвороби, симптомів і результатів лабораторних досліджень. Система спирається в своїх рішеннях на набір профілів захворювань (більше 500), що описуються більш, ніж 3500 проявами хвороби і які містять факти, що зустрічаються у зв'язку з кожним із захворювань. Перший варіант системи називається INTERNIST-I, другий – CADUCEUS.

На окрему увагу заслуговує експертна діагностична система по невідкладних станах у дітей ДИН, яка розроблена в Московському НДІ педіатрії і дитячої хірургії.

Як відмічено в [7], база знань ДИН містить описи 34 синдромів, які включають 84 стани. Для системи це список діагностичних припущень-гіпотез. База експертних знань лікаря-реаніматолога налічує більше 1000 діагностичних критеріїв і висновків про динаміку розвитку невідкладного стану.

База знань про синдроми в системі містить декларативну і процедурну інформацію. Декларативні знання описують синдроми (клінічна картина, додаткові синдроми), а процедурні вказують на те, як використовувати знання в процесі діагностики.

Механізм логічного виводу ДИН формує висновок на основі запрограмованої логіки лікаря-реаніматолога. Основою його роботи є змішана стратегія: використовується і прямий, і зворотний ланцюжок міркувань.

Відзначимо, що переважна більшість медичних експертних систем орієнтована на роботу з окремими органами або системами органів. Прокласифікуємо найбільш відомі діагностуючі експертні системи за цим критерієм:

1) порушення кислотно-лужної і водно-сольової рівноваги у пацієнтів, що визначаються на основі застосування знань про захворювання і симптоми, які викликаються ними (ABEL);
2) захворювання крові (MYCIN, HEME, порушення здатності згущуватися крові: AI/COAG, CLOT);
3) захворювання нирок (AI/MM, DIALYSIS THERAPY ADVISOR, EEG ANALYSIS SYSTEM);
4) ревматичні захворювання (AI/RHEUM, ARAMIS);
5) захворювання серцево-судинної системи (ANGY, ANNA, DIAGNOSER, DIGITALIS ADVISOR, GALEN, HEART IMAGE INTERPRETER, HT-ATTENDING, MECS-AI, MI, МОДИС);
6) захворювання щитовидної залози (MECS-AI, THYROID MODEL);

7) порушення в роботі нервової системи (BLUE BOX, HEADMED, NEUREX);
8) захворювання очей (CAS-NET/GLAUCOMA, MEDICO, OCULAR HERPES MODEL, PEC);
9) порушення функцій легенів (CENTAUR, PUFF, WHEEZE, TAPTA);
10) захворювання, пов'язані з болем в грудях (EMERGE, MED1);
11) захворювання печінки (MDX, PATREC, RADEX).

Таблица 1

сегменти	значення	інтерпретація
1, 2	0-20	4 стадія функц. нед.
	21-30	3 стадія функц. нед.
	31-40	2 стадія функц. нед.
	41-49	1 стадія функц. нед.
	50-65	норма
	66-90	функц. напр.
	91-100	істинна гіперфункція
3-30	0-20	4 стадія функц. нед.
	21-30	3 стадія функц. нед.
	31-50	2 стадія функц. нед.
	51-77	1 стадія функц. нед.
	78-86	норма
	87-90	функц. напр.
	91-100	істинна гіперфункція

Більшість медичних експертних систем застосовуються лише при нозологічній формі захворювання.

Сучасна медицина вимагає частішого застосування методів діагностики, що дозволяють оцінити стан організму на ранній стадії захворювання або навіть до його початку, коли розвитку недуги ще можна запобігти. Для ефективного підтримання здоров’я людини надзвичайно важливо вчасно виявити зміни в організмі людини, які потенційно можуть призвести до виникнення хвороб [5].

Одним із сучасних методів діагностики, що дозволяє оцінити стан здоров'я людини, є метод біорезонансної діагностики організму Р. Фолля, методологія якого побудована на інтеграції класичної китайської акупунктури і сучасних досягнень в області електрофізіології, гомеопатії, імунології і інших розділів медицини. Зокрема, апаратно-програмний комплекс ATM Express дозволяє отримати уявлення про об'єктивний стан організму, особливостях перебігу патології у даного пацієнта [8]. Для оцінки цього стану програма ATM Express Test реєструє частотні характеристики хвилевих процесів в організмі людини згідно сегментарним зонам шкіряної проекції органів і систем. У результаті цього формується відповідна діаграма з 30 показників, значення яких інтерпретуються у відповідності з таблицею 4.

Отримані дані характеризують функціональний стан організму пацієнта [8]:

- інтенсивність і збалансованість фізіологічних процесів в органах і системах;
- міра ендогенних і екзогенних інтоксикацій;
- наявність радіаційного навантаження;
- можливі пошкодження мікробіологічних циклів на рівні крові, лімфи або в кишечнику;
- дозволяють визначити локалізацію і стадію хронічних процесів;
- візуалізують спроможність таких енергетичних структур як аура і чакри.

Для традиційної оцінки цих показників і подальших досліджень по постановці діагнозу пацієнта необхідний висококваліфікований лікар. Проте, для успішної постановки діагнозу необхідні значні витрати часу і т.д. Тому актуальним є задача розробки експертної системи, яка б в автоматичному режимі проводила оцінку отриманих даних за наступними основними критеріями:

- наявність феномену падіння стрілки, які свідчить про скупчення в тканинах вільних радикалів, що виникають при хронічних процесах, локальних метаболічних порушеннях;
- наявність феномену падіння стрілки, які свідчить про скупчення в тканинах вільних радикалів, що виникають при хронічних процесах, локальних метаболічних порушеннях
- рівень діаграм, який порівнюється з отриманими даними відносно рівня нормальної функції;
- наявність бокової диференціальної різниці в симетричних відведеннях, що свідчить про однобічний процес;
- лабільність функціонального стану.

Експертна система, що розробляється, дозволяє провести як нозологічну, так і донозологічну діагностику стану здоров'я людини.

Перелік посилань

1. Базы знаний интеллектуальных систем / Т. А. Гаврилова, В. Ф. Хорошевский – СПб: Питер, 2000. – 384 с.
2. Джарратано Д., Райли Г. Экспертные системы: принципы разработки и программирование, 4-е издание.: Пер. с англ. – М.: ООО "И.Д. Вильямс", 2007. – 1152 с.
3. Джексон П. Введение в экспертные системы.: Пер. с англ.: Уч. пос. – М.: Издательский дом "Вильямс", 2001. – 624 с.
4. Ин Ц., Соломон Д. Использование Турбо-Пролога: Пер. с англ. – М.: Мир, 1993. – 608 с.
5. Кузьмук В. В., Супруненко О. О. Роль інформаційних технологій у донозологічній оцінці стану здоров’я людини // Тезисы и доклады международной конференции "Интегративная медицина" (24-25 мая 2008 р.). – К.: Алтимед, 2008. – С. 108-109.
6. Продеус А. Н., Захрабова Е. Н. Экспертные системы в медицине / А. Н. Продеус, Е. Н. Захрабова. – К.: ВЕК +, 1998. – 320 с.
7. Таперова Л. Н. Дин – экспертная диагностическая система по неотложным состояниям / Л. Н. Таперова, В. Веприцкая, Б. А. Кобринский // Программные продукты и системы. – 1995. – № 1. – С. 30-32.
8. Филюнова Е. Г. АТМ Express: Биорезонансная диагностика и терапия (ПБРТ): методические рекомендации / Филюнова Е. Г., Сиряковская Е. И., Демъянцева И. В. – К.: Алтимед, 2007. – 27 с.

УДК 004.93(015.7)

МАРКОВСКИЙ А.П.,
ЗЮЗЯ А.А.,
ШЕРСТЮК В.Д.

ПОЛУЧЕНИЕ БУЛЕВЫХ ПРЕОБРАЗОВАНИЙ СПЕЦИАЛЬНЫХ КЛАССОВ ДЛЯ ПОСТРОЕНИЯ ЭФФЕКТИВНЫХ АЛГОРИТМОВ ЗАЩИТЫ ИНФОРМАЦИИ

В статье представлены результаты исследований нелинейных булевых преобразований, обратные к которым обладают свойством неоднозначности и их использования в криптографических алгоритмах. Предложен новый метод получения булевых функциональных преобразований этого класса. Метод оперирует с процедурной формой представления булевых преобразований. Это позволяет строить преобразования от сотен булевых переменных. Булевы преобразования такого класса могут быть использованы для ускорения идентификации пользователей на основе концепции нулевых знаний. Установлены зависимости между временем построения преобразований и параметрами процедурной формы.

This paper presents investigation of nonlinear Boolean transformations inverse for which are ambiguous and its application in cryptographical algorithms. A new method for designing such class of Boolean transformations is suggested. The method deals with the procedure form representation of Boolean transformations. It allowed to buil Boolean transformatiom from hundreds Boolean variables. Boolean transformation on such class can be use for accelerate of user identification based on “zero-knoledge” conseption. The relationship between transformation building time and procedure form parameters is established.

Введение

Динамичное расширение и углубление процессов информационной интеграции требует адекватного совершенствования методов и средств защиты данных и контроля над правами доступа к ним.

К настоящему времени в основе большей часть систем защиты информации лежит использование криптографических механизмов. В свою очередь, такие механизмы базируются на математических преобразованиях, обладающих специфическими свойствами. Эти преобразования относятся к различным областям современной математики, но наиболее часто к теории чисел, конечных полей и булевой алгебре. Булевы преобразования обеспечивают существенно более высокую производительность реализации функций защиты информации в сравнении с другими преобразованиями.

Расширение использования средств защиты данных, в том числе в системах реального времени, требует радикального повышения скорости реализации вычислений, связанных с информационной безопасностью. Это определяет важность расширения возможностей использования для защиты информации именно булевых преобразований. Такое расширение может быть достигнуто путем получение преобразований со специальными свойствами, важными для защиты данных.

Таким образом, проблема расширения возможностей использования булевых функциональных преобразований в свете требований повышению скорости реализации функций защиты информации является актуальной.

Анализ проблемы получения необратимых преобразований, обратные к которым неоднозначны

Эффективность алгоритмов защиты информации определяется двумя факторами: уровнем защищенности и объемом вычислительных ресурсов, затрачиваемых на реализацию функций защиты.

В основе всех криптографических алгоритмов защиты информации лежит аналитически неразрешимая математическая задача [1]. В большинстве случаев практического использования, такие задачи имеют вид необратимого преобразования $Y=F(X)$, то есть преобразования, для которого определена алгоритмически функция $F(X)$ вычисления в прямом направлении и не существует аналитического способа получения функции $\Phi(Y)$ обратного преобразования $X=\Phi(Y)$ по известной функции $F(X)$. Единственным способом решения таких задач, то есть нахождения значения X для заданного значения Y при известной функции $F(X)$ является перебор значений X. Большая часть аналитически неразрешимых задач, лежащих в основе современных криптографических алгоритмов относится к теории чи-

сел, и булевой алгебре. В частности, к теории чисел относятся задачи дискретного логарифмирования, на основе которых построены большинство алгоритмов несимметричного шифрования (алгоритмов с открытым ключом), в том числе, широко используемые на практике RSA, El-Gamal, EEC, а также алгоритмы цифровой подписи, такие как DSS [1]. В основе достаточно широкого класса криптографических алгоритмов лежит аналитически неразрешимая задача булевой алгебры – отыскание корней систем нелинейных булевых уравнений. К этому классу алгоритмов относятся все алгоритмы блочного симметричного шифрования, такие как DES, IDEA, Rijndael, а также большая часть хеш-алгоритмов, в том числе, наиболее распространенные на практике RC-5, SHA и RIPEMD-160 [2].

Основным достоинством алгоритмов построенных на основе аналитически неразрешимых задач теории чисел является существование нескольких ключей. Так в алгоритмах шифрования RSA, El-Gamal, EEC, ключи, используемые для прямого и обратного преобразований различны. При наличии нескольких ключей, часть из них могут использоваться как открытые, а часть – как закрытые. Это позволяет строить на этой основе существенно более эффективные механизмы защиты информации и организации доступа к информационным ресурсам по сравнению с алгоритмами, имеющими единый ключ. Именно поэтому появление в 1978 г. алгоритма с “открытым” ключом – RSA, в основе которого лежит аналитически необратимое преобразование $F(X) = A^X \bmod M$, связывают с “открытием новой эры в технологии защиты информации” [2]. В теоретическом плане, существование нескольких ключей криптографического преобразования обусловлено тем, что необратимая задача является неоднозначной, то есть, существует, по крайней мере, два ключа X_1 и X_2 для которых выполняется $F(X_1) = F(X_2)$. Основным недостатком алгоритмов защиты информации, в основе которых лежат аналитически неразрешимые задачи теории чисел является низкая скорость их реализации, обусловленная высокой вычислительной сложностью операций модулярного экспоненцирования над числами, разрядность которых составляет тысячи.

Этого недостатка лишены алгоритмы защиты информации, в основе которых лежит аналитически неразрешимая задача булевой алгебры. Рекурсивное вычисление систем булевых функций может быть организовано достаточно эффективно как программными средствами на универсальных процессорах, так и специализированными аппаратными вычислителями. В оценочном плане, скорость криптографической обработки данных алгоритмами на основе булевых функций и на основе модулярной арифметики отличается на 3-4 порядка [3]. Однако, алгоритмы на основе булевых преобразований обладают существенно меньшими функциональными возможностями, которые не позволяют реализовать с их использованием эффективные протоколы защиты информации, подобно алгоритмам на основе неразрешимых задач теории чисел. В частности, использование булевых преобразований не позволяет реализовать базовые для современных технологий защиты информации концепции несимметричного шифрования данных, цифровой подписи сообщений, идентификации на основе схемы “нулевого знания”. Одним из факторов, обуславливающих ограниченные функциональные возможности алгоритмов на основе булевых преобразований является однозначность последних [3].

Одним из наиболее перспективных направлений совершенствования технологии защиты данных является расширение функциональных возможностей алгоритмов, основанных на булевой алгебре [2]. Это позволит строить на основе таких алгоритмов эффективные протоколы защиты данных в сетях, реализация которых требует существенно меньших вычислительных ресурсов и, соответственно, может выполняться на порядки быстрее, чем при использовании алгоритмов, основанных на модулярной арифметике.

В рамках реализации этого направления совершенствования технологии защиты информации важным представляется разработка методологии получения необратимых булевых преобразований, обладающих свойством неоднозначности [3]. Такие преобразования, в частности, могут быть эффективно использованы для создания эффективных протоколов идентификации удаленных абонентов многопользовательских систем на основе концепции “нулевых знаний” [4].

Целью исследований является разработка метода получения булевых функциональных преобразований, обладающих свойствами не-

обратимости и неоднозначности обратного преобразования.

Метод получения преобразований специальных классов в процедурной форме

Для достижения поставленной цели необходимо сформировать булево функциональное преобразование $F(X)$, определенное на множестве 2^n значений n-битового бинарного вектора $X=\{x_1,x_2,\ldots,x_n\}$, $\forall i\in\{1,\ldots,n\}$: $x_i\in\{0,1\}$ результатом которого является n-битовый выходной бинарный вектор $Y=F(X)$, $Y=\{y_1,y_2,\ldots,y_n\}$, $\forall i\in\{1,\ldots,n\}$: $y_i\in\{0,1\}$. Каждую i-тую бинарную компоненту y_i выходного вектора Y можно рассматривать, как значение булевой функции $f_i(X)$, определенной на множестве значений X. Исходя из этого, можно говорить, что преобразование F(X) состоит из n булевых функций $f_1(X),f_2(X),\ldots,f_n(X)$: $F(X)=\{f_1(X),f_2(X),\ldots,f_n(X),\}$.

Для того, чтобы синтезируемое преобразование $F(X)$ обладало свойством необратимости, необходимо, чтобы каждая из булевых функций $f_1(X),f_2(X),\ldots,f_n(X)$, составляющих преобразование $F(X)$ должна быть нелинейной и зависеть от каждой из n входных переменных $X=\{x_1,x_2,\ldots,x_n\}$.

Для того, чтобы преобразование $F(X)$ обладало свойством неоднозначности, необходимо, чтобы существовало множество Ω входных векторов, на которых преобразование $F(X)$ принимает одинаковое значение: $\forall Q,G\in\Omega,\ Q\neq G$: $F(Q)=F(G)=U$.

Как известно, булевы функциональные преобразования могут быть представлены в следующих трех формах:

- в виде таблиц истинности;
- в алгебраических нормальных формах
- в процедурном виде, то есть в виде алгоритма вычисления выходного вектора Y по значению входного вектора X.

Применительно к задачам защиты информации, преимущественно, используется третья из указанных форм представления булевых функциональных преобразований, поскольку число переменных, на которых определены такие преобразования составляет сотни. Исходя из этого, предлагается формировать преобразование в процедурном виде.

В качестве базовой процедурной схемы вычисления преобразования $F(X)$ предлагается структура, показанная на рис.1.

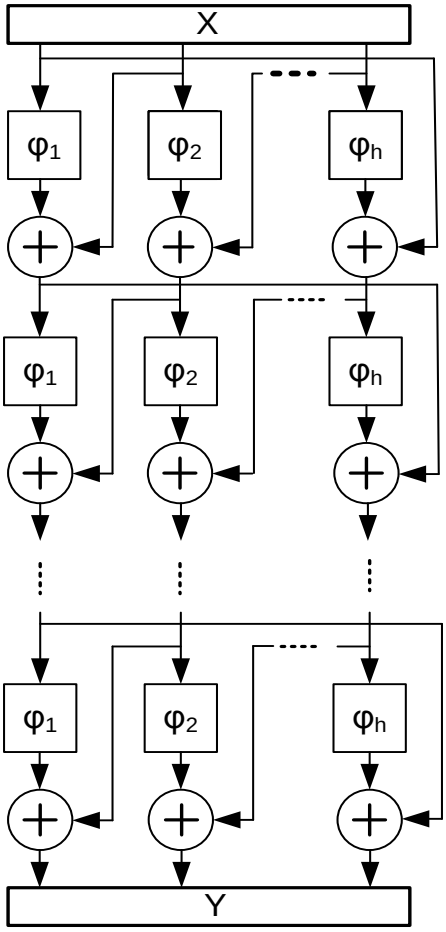


Рис.1. Структура процедурной формы вычисления булевых преобразований

Структура предполагает организацию вычисления $F(X)$ с разбиением данных на k-разрядные фрагменты. Если полагать, что n и k являются степенями 2, то число h фрагментов определяется как $h = n/k$. Подобно большинству структур, используемых для реализации булевых преобразований алгоритмов защиты информации, структура, показанная на рис.1, состоит из перемежающихся слоев нелинейного преобразования и перемешивания фрагментов. Нелинейное преобразование осуществляется с использованием h табличных булевых функциональных преобразований $\varphi_1,\varphi_2,\ldots,\varphi_h$. Каждое j-тое, $j\in\{1,\ldots,h\}$, фрагментарное преобразование $\varphi_j(V)$ определено на множестве 2^k всех возможных значений k-битового вектора $V=\{v_1,v_2,\ldots,v_k\}$, $\forall l\in\{1,\ldots,k\}$: $v_l\in\{0,1\}$. Для перемешивания результатов обработки отдельных фрагментов используется суммирование по модулю 2 результатов нелинейных преобразований смежных фрагментов. Для того, чтобы каждый бит выходного вектора Y зависел от каждого из битов входного

вектора X, число повторяющихся циклов в структуре преобразования должно быть не меньше h – числа фрагментов.

Если обозначить через Z_{qj} k-битовый вектор, являющийся j-тым фрагментом n–битового кода $W_q=\{Z_{q1},Z_{q2},\ldots,Z_{qh}\}$ формируемого на выходе q-того цикла, при этом $W_0 = X$, $W_h = Y$, то процедура вычисления функции, представленной на рис.1 аналитически может быть описана в виде:

$$\begin{aligned}&\forall j \in \{1,...,h\}: Z_{0,j} = \{x_{(j-1)\cdot k+1}, x_{(j-1)\cdot k+2},..., x_{j\cdot k}\};\\&\forall q \in \{1,...,h\}:\\&\forall g \in \{1,...,g-1\}: Z_{qg} = \varphi_j(Z_{q-1,g}) \oplus Z_{q-1,g+1}\\&Z_{qh} = \varphi_h(Z_{q-1,h}) \oplus Z_{q-1,1}\end{aligned} \quad (1)$$

Ниже через $W_q(Q)$ обозначено значение промежуточного кода на выходе q-го цикла для значения входного вектора $X=Q$. Соответственно, через $Z_{qj}(Q)$ обозначено код j-того фрагмента вектора $W_q(Q)$.

Задачей формирования неоднозначного и необратимого булевого преобразования $F(X)$ в рамках структуры, показанной на рис.1, является получение фрагментарных булевых функциональных преобразований – $\varphi_1,\varphi_2,\ldots\varphi_h$, которые для заранее выбранного множества Ω входных бинарных векторов $\Omega =\{X_1,X_2,\ldots,X_m\}$ обеспечивают единое значение результата, вычисляемого в соответствии с (2.10). Каждое j-тое, $j\in\{1,\ldots,h\}$, фрагментарное булево преобразование $\varphi_j(V)$ состоит из k булевых функций $\varphi_j =\{\phi_{j1}(V),\phi_{j2}(V),\ldots,\phi_{jk}(V)\}$ и может иметь результатом значения, принадлежащие интервалу от 0 до 2^k. Для решения этой задачи предлагается специальный метод, который состоит в последовательном определении значений фрагментарных функциональных преобразований в процессе преобразования входных векторов из множества Ω. Метод использует множество Θ, в которое включается для каждого из фрагментарных функциональных преобразований $\varphi_1,\varphi_2,\ldots\varphi_h$ список наборов их аргументов, на которых значение преобразования определяются случайным образом в процессе вычисления значения текущего входного вектора X_t.

Предлагаемый метод сводится к выполнению следующей последовательности действий:

1. Значения всех h фрагментарных преобразований $\varphi_1,\varphi_2,\ldots\varphi_h$ на всех 2^k возможных булевых наборах значений их аргументов принимаются равными –1, что соответствует неопределенному значению. Множество $\Omega=\varnothing$.

2. Выбирается произвольный n-битовый вектор X_1, который включается во множество Ω.

3. Вычисляется значение $U = F(X_1)$ в соответствием с (2.10). При этом, на каждом q–том из h слоев процедуры, для каждого j-го фрагмента (j=1,…h), если для предшествующего фрагмента $z_{q-1,j}(X_1)$ значение фрагментарного функционального преобразования не определено, то есть $\varphi_j(z_{q-1,j}(X_1))$=-1, то оно определяется как случайное число из интервала от 0 до 2^k-1.

4. Выбирается произвольно вектор X_t. Положить $W_0(X_t)=X_t$. Множество Θ полагается пустым: $\Theta=\varnothing$. Случайным образом генерируется номер N_c слоя ($N_c=\{1,\ldots,h\}$), на котором осуществляется уравнивание (“слияние”) промежуточных значений вектора X_t и одного из векторов множества Ω с порядковым номером меньшим t. Установить номер q текущего слоя в единицу: q=1. Если N_c=1, то переход на п.7.

5. Если q<N_c-1, то вычисление промежуточных значений для каждого j–го фрагмента осуществляется следующим образом: в случае $\varphi_j(z_{q-1,j})\neq$-1, то вычисление производится в соответствии с (1). Если значение фрагментарного функционального преобразования φ_j на наборе $z_{q-1,j}$ не определено: $\varphi_j(z_{q-1,j})$= -1, то случайным образом формируется принадлежащее интервалу от 0 до 2^k-1 значение φ_j на наборе $z_{q-1,j}$, после чего вычисление осуществляется согласно (1). При этом номер j доопределяемого преобразования φ_j вместе с набором $z_{q-1,j}$ заносятся в множество Θ: $\Theta=\Theta\cup <j,z_{q-1,j}>$. После вычисления таким образом всех h фрагментов промежуточных результатов q-слоя для кода X_t осуществляется переход к следующему слою, путем присваивания $q=q$+1 и возврата на повторное выполнение пп. 5.

6. Если $q=N_c$-1, и $\varphi_j(z_{q-1,j})$= -1, то значение преобразования φ_j на наборе $z_{q-1,j}$, выбирается таким образом, чтобы на наборе $z_{q,j} = \varphi_j(z_{q-1,j})\oplus z_{q-1,j+1}$ значение преобразования φ_j было не определено, то есть выполнялось: $\varphi_j(z_{q,j})$= -1. При этом номер j доопределяемого преобразования φ_j вместе с набором $z_{q-1,j}$ заносятся в множество Θ: $\Theta=\Theta\cup <j,z_{q-1,j}>$. Если $\varphi_j(z_{q-1,j}) \neq$ -1, то вычисление $z_{q,j}$ осуществляется в соответствии с (2.10). После вычисления таким

образом всех h фрагментов промежуточных результатов q-слоя для кода X_t осуществляется переход к следующему слою, путем присваивания $q=q+1$.

7. Для каждого j-го фрагмента проверяется выполнение условие: $\varphi_j(z_{q-1,j}) = -1$, если это условие выполняется, то случайным образом $d\in\{1,\ldots,t-1\}$, после чего осуществляется переход на пп.8. Для каждого j-го фрагмента проверяется выполнение условие: $\varphi_j(z_{q-1,j})\neq -1$, но при этом существует такое $d\in\{1,\ldots,t-1\}$, что $\varphi_j(z_{q-1,j})\oplus z_{q-1,j+1} = z_{qj}(X_d)$. Если это условие выполняется, то осуществляется переход на пп.8. Если означенные выше условия не выполняются, то проведенный подбор значений фрагментарных функциональных преобразований для кода X_t является конфликтным. Все определенные при преобразовании X_t значения фрагментарных булевых преобразований вновь считаются неопределенными: $\forall\langle j,z\rangle\in\Theta$: $\varphi_j(z)= -1$. Возврат на пп.4.

8. Для каждого j-го фрагмента выполняется следующее: если $\varphi_j(z_{q-1,j}) = -1$, то соответствующее значение фрагментарного преобразования на наборе определяется как: $\varphi_j(z_{q-1,j}(X_t)) = z_{q-1,j+1}(X_t) \oplus z_{q-1,j}(X_d)$, если $j<h$ и $\varphi_j(z_{q-1,j}(X_t)) = z_{q-1,1,}(X_t) \oplus z_{q-1,j+1}(X_d)$, если $j=h$.

9. Если $t < m$, то увеличивается на единицу номер t текущего вектора X из множества Ω: $t = t+1$. Возвращение на пп. 4.

10. Для всех h фрагментарных функциональных булевых преобразований $\varphi_1,\varphi_2,\ldots,\varphi_h$, на наборах, которые не определены раньше, установить значения, которые генерируется, как случайные целые числа, принадлежащие интервалу от 0 до 2^k–1: $\forall j\in\{1,\ldots,h\}$, $\forall z\in\{0,\ldots,2^k-1\}$: $\varphi_j(z)= -1$ определить : $\varphi_j(z)=$ Random$(0,2^k-1)$.

Результатом работы приведенной выше последовательности действий является h таблиц фрагментарных булевых преобразований $\varphi_1,\varphi_2,\ldots\varphi_h$, которые полностью задают, в совокупности со структурой, показанной на рис.1, процедуру вычисления преобразования $F(X)$.

Вероятность того, что результатом преобразования над случайным входным кодом X_e будет получено значение, совпадающее с U равно 2^{-n}. Учитывая, что численное значение n на практике составляет сотни, то вполне очевидно, что вероятность подбора идентифицирующего кода внешним по отношению к системе лицом, при использовании предложенного преобразования, практически равна нулю.

Использование разработанного метода получения нелинейных и неоднозначных булевых преобразований иллюстрируется следующим примером. Пусть n – разрядность кодов на входе и выходе преобразования $F(X)$ равна 16, то есть n=16. Пусть, далее, обрабатываемые преобразованием $F(X)$ коды разделяются на 4 фрагмента (h=4), каждый по 4 разряда ($k=n/h$=4). Это означает, что процедура преобразования состоит из 4-х слоев. Фрагментарные табличные булевы преобразования $\varphi_1,\varphi_2,\varphi_3,\varphi_4$ определены на множестве 16-ти возможных значений 4-х булевых переменных и каждое из низ содержит 4 булевых функции, так, что каждое из упомянутых преобразований может принимать значения от 0 до 15-ти. Пусть множество Ω состоит из 3-х 16-битовых ключей, которые могут быть представлены с использованием 16-ричной системы в следующем виде: X_1=5E96h, X_2 = B1C6h, X_3 = 7A48h. Динамика заполнения таблиц истинности фрагментарных булевых функциональных преобразований $\varphi_1,\varphi_2,\varphi_3,\varphi_4$ в процессе обработки ключей, принадлежащих множеству Ω отражена в таблице 1. Незаполненные клетки таблицы соответствуют неопределенному значению фрагментарных булевых преобразований на соответствующих наборах входных переменных.

Результатом работы предложенного метода являются таблицы истинности фрагментарных булевых преобразований $\varphi_1,\varphi_2,\varphi_3,\varphi_4$.

После обработки всех 3-х кодов X_1,X_2,X_3 в значения фрагментарных функциональных преобразований $\varphi_1,\varphi_2,\varphi_3,\varphi_4$ определены на 64% наборов значений их входных переменных. Соответственно, на 36% наборов в соответствии с пп.10 изложенного выше метода, значения функций определяются случайным образом.

Табл.1. Пример заполнения таблиц фрагментарных преобразований $\varphi_1,\varphi_2,\varphi_3,\varphi_4$

X	Таблицы значений фрагментарных булевых преобразований							
	После обработки X_1				В окончательном виде			
	φ_1	φ_2	φ_3	φ_4	φ_1	φ_2	φ_3	φ_4
0			11		12	4	11	8
1					3	14	9	2
2					0	0	6	7
3	4				4	8	4	11
4	8				8	12	6	5
5	10				10	7	3	8
6				2	3	10	7	2
7			12	15	13	15	12	15
8					12	7	15	6
9	3		6		3	9	6	8
10		15			5	15	15	10
11		13		14	15	13	10	14
12			12		8	9	12	11
13		6		0	5	6	10	0
14		2			8	2	12	9
15					0	3	15	3

Формирование фрагментарных булевых функциональных преобразований $\varphi_1,\varphi_2,\varphi_3,\varphi_4$ в рамках изложенного метода выполняется с использованием проб: если для при обработке очередного входного вектора $X\in\Omega$, в процессе которого используется случайное определение значений преобразований $\varphi_1,\varphi_2,\varphi_3,\varphi_4$ на определенных наборах, не удается избежать конфликта с ранее определенными значениями, то, согласно пп.7 осуществляется возврат на повторную обработку вектора X. Таким образом, основным фактором, от которого зависит время формирования нелинейного функционального преобразования является количество проб заполнения таблиц фрагментарных функциональных преобразований.

при обработке вектора окажется успешной определяется следующим выражением:

$$P_g = (1-\frac{h^2\cdot g^2}{2^{2\cdot(k+1)}})^h \qquad (2)$$

Соответственно, среднее число t_g проб, затрачиваемых для заполнения таблиц фрагментарных преобразований при обработке g–го входного вектора определяется как: $t_g = 1/P_g$. Тогда, среднее число T_m проб, затрачиваемых для заполнения таблиц фрагментарных преобразований при обработке всех m входных кодов, составляющих множество Ω, определяется как сумма среднего числа проб, требующихся для обработки каждого из m входных кодов:

$$T_m = \sum_{j=1}^{m}\frac{1}{(1-\frac{h^2\cdot j^2}{2^{2\cdot(k+1)}})^h} \qquad (3)$$

Общий объем V_T памяти таблиц фрагментарных нелинейных булевых преобразований определяется в виде:

$$V_T = h\cdot 2^{\frac{n}{h}} \qquad (4)$$

Исходя из сказанного, важным аспектом анализа предложенного метода получения неоднозначного и необратимого функционального булевого преобразования в процедурной форме является оценка зависимости числа проб от ее параметров.

Пусть, обрабатывается g-тый по очереди входной вектор X_g множества Ω. Поскольку при обработке вектора, в среднем, определяется значения фрагментарных преобразований на половине слоев, то среднее число значений одного преобразования, определяемых при обработке одного вектора X составляет $h/2$. Из этого следует, что к концу обработки g–го входного вектора X_g в каждом из h фрагментарных преобразований будет определено, в среднем, $h\cdot g/2$ значений. Из приведенного выше описания метода (пп.7) следует, что конфликтная ситуация возникает, если при обработке j-го фрагмента $z_{q,j}$ промежуточного значения на q-том слое значения соответствующего преобразования φ_j ранее определены как на наборе $z_{q,j}$, так и на наборе $\varphi_j(z_{q,j})\oplus z_{q-1,j+1}$ для $j<h$ или наборе $\varphi_j(z_{q,j})\oplus z_{q-1,1}$ для $j=h$. То есть, для того, чтобы конфликтная ситуация возникла, необходимо, чтобы для двух фиксированных наборов значения функционального преобразования φ_j были ранее определены. Учитывая, что общее число наборов, на которых определяется φ_j равно 2^k, из них, в среднем, на $h\cdot g/2$ значение преобразования φ_j уже определено ранее, то вероятность того, что на двух фиксированных наборах значение φ_j определено ранее составляет $(h\cdot g)^2/2^{2\cdot(k+1)}$. Принимая во внимание, что для повторной обработки вектора достаточно наличие конфликтной ситуации в одном фрагменте, вероятность P_g того, что проба случайного заполнения всех h фрагментарных преобразований $\varphi_1,\varphi_2,\ldots,\varphi_h$

Для проверки полученной теоретическим путем зависимости времени формирования функционального преобразования в зависимости от параметров процедурной формы и числа m возможных входных векторов были проведены статистические исследования с применением программной модели. Результаты экспериментальных исследований показали, что теоретическая оценка (3) упомянутых зависимостей, в целом, соответствует результатам, полученным при статистическом моделировании. Так, при длине кода доступа n=256 при параметрах процедурной формы h=16 и k=16, ее построение для числа кодов доступа m=4000 требует около 40000 проб, что занимает не более часа работы персонального компьютера. При этом множество Ω содержит около 4000 входных векторов, для которых результат преобразования $F(X)$ одинаков. При этом в процессе построения функционального преобразования заполнено около 47% таблиц фрагментарных преобразований. Остальные наборы таблиц в соответствии с пп.10 заполняются случайным образом.

Выводы

В результате проведенных исследований предложен метод получения процедурной формы нелинейных булевых функциональных преобразований, обратное к которым обладает свойством неоднозначности. Такие преобразования позволяют использовать их в некоторых применениях вместо модулярных операций над большими числами, что обеспечивает значительное (на 2-3 порядка) повышение производительности.

Установлены зависимости между параметрами процедурной формы, временем реализации алгоритма и характеристиками булевых перобразований.

Разработанный метод позволяет получать булевы функциональные преобразования, применение которых обеспечивает повышение скорости реализации алгоритмов защиты информации и средств идентификации удаленных абонентов.

Список литературы

1. Иванов М.А., Криптографические методы защиты информации в компьютерных системах и сетях. М.: ”Кудиз-образ”, 2001.– 368 с.
2. Самофалов К.Г., Марковский А.П., Гаваагийн Улзисайхан, Бардис Н., Метод получения булевых балансных SAC-функций для систем защиты информации. // Вісник Національного технічного університету України ”КПІ”. Інформатика, управління та обчислювальна техніка.–1998.– № 31.– С.131-140.
3. Seberry J., Zhang X., Zheng Y. Nonlinearity and propagation characteristics of balanced Boolean functions.//Information and Computation Academic Press. 1995.–Vol. 119, № 1 –P.1-13.

УДК 681.3.07

ЗАЙЧЕНКО Ю.П.,
МУРГА Н.А.

ПРИМЕНЕНИЕ СИСТЕМ НА НЕЧЁТКОЙ ЛОГИКЕ К ЗАДАЧЕ МЕДИЦИНСКОЙ ДИАГНОСТИКИ

В данной работе проводится исследование возможности применения нечётких нейронных сетей для задачи медицинской диагностики на примере применения сетей для определения раковых образований в гинекологии. Производится исследование возможностей нечёткой нейронной сети Такаги-Сугено-Канга для решения данной задачи. Так же в работе произведена оценка возможности уменьшения количества тестов, необходимых для процесса диагностирования, что и завершает данную статью.

Investigation of neural network use for medical diagnostic problem on cancer detection example in gynaecology is conducted in this work. Investigation of Takagi-Sugeno-Kang's fuzzy neural network opportunities for this problem solving is conducted. Furthermore, posibility of test's (which are necessary for diagnostics process) number decrease is conducted in this work.

Вступление

Существует очень много работ по применению нейронных сетей в задачах экономики, однако применение нейронных сетей к задаче медицинской диагностики авторам не встречалось. В то же время эта задача весьма актуальна. Применение математических моделей вообще к данной задаче предложено в [1]. Однако ещё авторы данной работы указывают на значительную ошибку, выдаваемую данной моделью и указывают на необходимость поиска более совершенной (модели). Целью данной работы будет анализ применения нечёткой нейронной сети Такаги-Сугено-Канга (TSK-сети) ([2]) к данным, полученным в результате проведения колпоскопического исследовани с целью обнаружения опухолей. Данные представляют из себя (для каждого пациента) 32 теста и 1 диагноз (диагноз – агрегированный диагноз нескольких специалистов). 32 теста рассматриваются как входы сети TSK, а диагноз – выход. Вся выборка данных разбивается на обучающую и проверочную при соотношениях 50:50, 60:40, 70:30, 80:20, 90:10 соответственно. На основании этого строится и обучается нечёткая сеть. Работу завершает анализ возможности уменьшения числа тестов, необходимых для эффективной работы системы.

Постановка задачи

Имеется выборка данных по результатам обследования пациентов. Первые 32 столбца выборки – результаты тестов. Последний столбец – диагноз, поставленный специалистами по результатам тестов. То есть, имеем для каждого случая (пациента) 32 входа и 1 выход. Выборка данных разбивается на обучающую и проверочную в соотношениях: 50:50, 60:40, 70:30, 80:20, 90:10.

В данной работе необходимо провести анализ возможности применения сети TSK для данных обучающих выборок и выполнять проверку эффективности функционирования сети на соответственных проверочных выборках при различном числе правил нечёткого вывода. По результатам экспериментов выделить закономерности и сделать выводы.

Далее, для найденного оптимального соотношения обучающей и проверочной выборки и числа правил произвести последовательное уменьшение тестов, участвующих в диагностике.

1. Применение нечёткой нейронной сети TSK

Ниже приводится описание сети, изложенное в [2].

Обобщённую схему вывода в модели TSK при использовании M правил и N переменных x_j можно представить в следующем виде:

$$R_1: \ if \quad x_1 \in A_1^{(1)}; x_2 \in A_2^{(1)}; ...; x_n \in A_n^{(1)} \quad then$$

$$y_1 = p_{10} + \sum_{j=1}^{N} p_{1j} x_j \ ;$$

$$R_M \ : \ if \quad x_1 \in A_1^{(M)}; x_2 \in A_2^{(M)}; ...; x_n \in A_n^{(M)}$$

$$then \ \ y_M = p_{M0} + \sum_{j=1}^{N} p_{Mj} x_j \ ,$$

где $A_i^{(k)}$ – значение лингвистической переменной x_i для правила R_k с функцией принадлежности $\mu_A^{(k)}(x_i), i = \overline{1,N}; k = \overline{1,M}$.

В нечёткой сети TSK пересечение условий правила R_k определяется функцией принадлежности в форме произведения, то есть:

$$\mu_A^{(k)}(x) = \prod_{j=1}^{N} \mu_A^{(k)}(x_j). \qquad (1)$$

При M правилах вывода композиция выходных результатов сети определяется по следующей формуле (аналогично выводу Сугено):

$$y(x) = \frac{\sum_{k=1}^{M} \omega_k y_k(x)}{\sum_{k=1}^{M} \omega_k}. \qquad (2)$$

где $y_k = p_{k0} + \sum_{j=1}^{N} p_{kj} x_j$. Присутствующие в этом выражении веса ω_k интерпретируются как степень выполнения условий правила: $\omega_k = \mu_A^{(k)}(x)$, которые задаются формулами (1).

Для того чтобы сеть (далее будет иногда заменяться название нечёткая нейронная сеть TSK на, просто, сеть) смогла выполнять функцию, которая от неё ожидается, необходимо её обучить.

Это обучение подразумевает, во-первых, задание количества правил, которые будут использованы сетью TSK (аналогично, сеть и сеть TSK в изложении будут равнозначны). Во-вторых, необходимо задать начальные функции принадлежности (ФП) для каждого входа и каждого правила. В-третьих, задаться видом «то»-части нечётких правил. После выполнения данных действий можно приступать, собственно, к обучению сети.

Возникает немаловажный вопрос: как задать количество правил и ФП входов?

Здесь на помощь приходит ещё один механизм, называемый кластеризацией.

Ниже приведено описание кластеризации, предложенное в [2].

Алгоритм самоорганизации относит вектор x к соответственному кластеру данных, которые представляются центром c_i , используя соревновательное обучение.

В данной работе будет интересен такой алгоритм самоорганизации, как алгоритм разностного группирования.

Для его описания используется всё та же работа [2].

Алгоритм разностного группирования – это модификация алгоритма пикового группирования, в котором векторы, подлежащие кластеризации x_j, рассматриваются как потенциальные центры кластеров. Пиковая функция $D(x_i)$ задаётся формулой:

$$D(x_i) = \sum_{j=1}^{N} \exp\left\{ -\frac{\left\| x_i - x_j \right\|^{2b}}{\left(\frac{r_a}{2} \right)^2} \right\}. \qquad (3)$$

где значение коэффициента r_a определяет сферу соседства. На значения $D(x_i)$ значительно влияют только x_j, которые находятся в пределах данной сферы.

При большой плотности точек вокруг x_i значение функции $D(x_i)$ большое. После расчёта значений пиковой функции для каждой точки x_i, выбирается вектор x, для которого мера плотности $D(x)$ окажется самой большой. Именно эта точка и становится первым центром c_1.

Выбор следующего центра c_2 возможен после исключения предыдущего центра и всех точек, лежащих в его окрестности.

Пиковая функция переопределяется следующим образом:

$$D_{new}(x_i) = D(x_i) - D(c_1) \cdot \exp\left\{ -\frac{\left\| x_i - c_1 \right\|^{2b}}{\left(\frac{r_b}{2} \right)^2} \right\}. \qquad (4)$$

При новом определении функции D коэффициенты r_b обозначают новые значения константы, которая задаёт сферу соседства очередного центра. Обычно придерживаются условия, что $r_b \geq r_a$.

После модификации значения пиковой функции ищется новая точка x, для которой

$D_{new}(x) \to \max$. Она становится новым центром.

Процесс поиска очередного центра возобновляется после исключения всех компонент, которые отвечают уже отобранным точкам. Инициализация завершается в момент фиксации всех центров, которые предусмотрены начальными условиями.

В соответствии с описанным алгоритмом происходит самоорганизация множества векторов x, которая состоит в нахождении оптимальных значений центров, которые представляют множество данных с минимальной погрешностью.

Если мы имеем дело с множеством учебных данных в виде пар векторов $(x_i \quad d_i)$, так как это имеет место при обучении с учителем, то для нахождения центров, которые отвечают множеству векторов d_i, достаточно сформировать расширенный вектор: $[x_i, d_i] \to x_i$.

Процесс группирования, который проводится с использованием расширенных векторов x_i, позволяет определить также и расширенные версии центров c_i.

С учётом того, что размерность каждого нового центра равна сумме размерностей векторов x и d, то в описании этого центра можно выделить часть p, соответствующую вектору x (первые N компонент) и остаток q, который отвечает вектору d. Таким образом, можно получить центры как входных переменных, так и ожидаемых выходных значений $c_i = [p_i, q_i], i = \overline{1, K}$.

Теперь необходимо на вопрос: как именно может помочь данный алгоритм для решения поставленной проблемы?

Ответ очевиден. Количество полученных кластеров после его применения и будет искомым количеством правил. Значения компонент центров кластеров будут определять центры ФП. Дисперсии ФП (берутся ФП гауссовского вида) задаются некоторыми начальными значениями, которые в дальнейшем будут корректироваться.

«Если»-часть правил построена.

Рассматривается «то»-часть.

А здесь возникает два сценария дальнейших действий: считать функцию в «то»-части либо линейной, либо константой.

Необходимо разобраться с каждым из этих случаев в отдельности.

Для начала, константа. Как задать данную константу для каждого правила? А данная задача уже решена. Выше описано применение алгоритма разностного группирования для обучения с учителем. Так вот, размерность вектора d для данного случая равна единице. Выходит, для каждого i-го правила q_i и будет искомой константой.

Сеть можно «доучить» (подкорректировать дисперсии), используя метод Back Propagation, с которым можно ознакомиться в [2].

Далее, линейная функция. Для её построения можно применить метод наименьших квадратов (МНК далее). Необходимо рассмотреть, как это можно сделать, однако, вместе с алгоритмом, который будет использоваться для обучения сети такого вида – гибридным алгоритмом обучения нечётких нейронных сетей.

Приводится выдержка из его описания, предложенного в работе [2].

В гибридном алгоритме параметры, которые подлежат адаптации, разделяются на две группы. Первая из них состоит из линейных параметров p_{kj} – «то»-части нечётких правил, а вторая – из параметров нелинейных ФП «если»-части. Уточнение параметров происходит в два этапа.

На первом этапе при фиксации отдельных значений параметров функций принадлежности (в первом цикле – это значения, которые получены путём инициализации), решая систему линейных уравнений, рассчитываются линейные параметры p_{kj} полинома TSK. При известных значениях ФП зависимость для выхода можно представить в виде линейной формы относительно параметров p_{kj}:

$$y = \sum_{k=1}^{M} \omega'_k \left(p_{k0} + \sum_{j=1}^{N} p_{kj} x_j \right). \qquad (5)$$

где

$$\omega'_k = \frac{\prod_{j=1}^{N} \mu_A^{(k)}(x_j)}{\sum_{r=1}^{M} \prod_{j=1}^{N} \mu_A^{(r)}(x_j)}, \; k = \overline{1, M}. \qquad (6)$$

При размерности обучающей выборки $L(x^{(l)}, d^{(l)})$, $(l = \overline{1, L})$ и замене выходного сигнала сети ожидаемым значением $d^{(l)}$, получаем систему из L линейных уравнений вида:

$$\begin{bmatrix} \omega'_{11} & \omega'_{11}\cdot x_1^{(1)} & \dots & \omega'_{11}\cdot x_N^{(1)} & \dots & \omega'_{1M} & \omega'_{1M}\cdot x_1^{(1)} & \dots & \omega'_{1M}\cdot x_N^{(1)} \\ \omega'_{21} & \omega'_{21}\cdot x_1^{(2)} & \dots & \omega'_{21}\cdot x_N^{(2)} & \dots & \omega'_{2M} & \omega'_{2M}\cdot x_1^{(2)} & \dots & \omega'_{2M}\cdot x_N^{(2)} \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ \omega'_{L1} & \omega'_{L1}\cdot x_1^{(L)} & \dots & \omega'_{L1}\cdot x_N^{(L)} & \dots & \omega'_{LM} & \omega'_{LM}\cdot x_1^{(L)} & \dots & \omega'_{LM}\cdot x_N^{(L)} \end{bmatrix} \bullet \begin{bmatrix} p_{10} \\ p_{11} \\ \dots \\ p_{1N} \\ \dots \\ p_{M0} \\ p_{M1} \\ \dots \\ p_{MN} \end{bmatrix} = \begin{bmatrix} d^{(1)} \\ d^{(2)} \\ \dots \\ d^{(L)} \end{bmatrix} \qquad (7)$$

где ω'_{lj} означает уровень активации (вес) условия i-го правила при предъявлении l-го входного вектора $x^{(l)}$. Это выражение можно записать в матричном виде:

$$Ap = d \qquad (7)$$

Размерность матрицы A равна $L\cdot(N+1)\cdot M$. При этом количество рядов L обычно бывает значительно больше количества столбцов $(N+1)\cdot M$. Для отыскания p применим метод наименьших квадратов.

На втором этапе после фиксации линейных параметров p_{kj} рассчитываются фактические выходные сигналы $y^{(l)}, l=\overline{1,L}$, для чего используется линейная зависимость:

$$y = Ap \qquad (8)$$

После этого рассчитывается вектор ошибки $\varepsilon = y - d$ и критерий:

$$E = \frac{1}{2}\sum_{l=1}^{L}\left(y\left(x^{(l)}\right) - d^{(l)}\right)^2 \qquad (9)$$

Сигналы ошибок распространяются через сеть в обратном направлении соответственно к методу Back Propagation, прямо к первому слою сети (ФП), где могут быть рассчитаны компоненты вектора градиента целевой функции относительно параметров ФП. После вычисления вектора градиента делается шаг спуска градиентным методом.

После уточнения нелинейных параметров снова запускается процесс адаптации линейных параметров функции TSK (первый этап) и не нелинейных (второй этап). Этот цикл повторяется до тех пор, пока стабилизируются все параметры процесса.

2. Уменьшение количества входных переменных для диагностики

Теперь необходимо ответить ещё на один вопрос: действительно для диагностики нужны все входные переменные или их число можно уменьшить.

Необходимо определиться с тем, по какому признаку будут исключаться тесты. Здесь ответ, с одной стороны, прост и очевиден, а с другой – очень сложен. Прост он тем, что порядок исключения будет определяться величиной корреляции результатов тестов. Сложность возникает в том, что если есть пара элементов, для которых коэффициент корреляции равен 0,98001 и пара элементов, для которых коэффициент корреляции равен 0,98, то сказать точно между какими элементами больше зависимость – нельзя, так как последующие эксперименты могут существенно изменить картину. Однако в данной работе считается, что чем больше корреляция, тем строго больше зависимость между элементами и рассмотрение данной проблемы оставлено для последующих работ.

Строится корреляционная матрица для всех тестов – A, где a_{ij} – коэффициент корреляции между тестами i и j.

Далее, допускается, что найден максимальный элемент в матрице A – a_{ij}.

Выбирается из i-го и j-го (произвольно) какой тест оставляется для рассмотрения, а второй – исключается.

Допускается, что исключается тест под номером i.

Из матрицы A вычёркивается строка и столбец с индексом i.

Далее, необходимо определиться с критерием, не выполнение которого будет свидетельствовать о том, что текущее количество тестов недостаточно для постановки диагноза.

Допустима ошибка диагностики 0,005. Следовательно, на проверочной выборке должно выполнятся условие

$\max_i \left| y_i - \overline{y_i} \right| < 0{,}005$, где y_i – значение, поставленное врачами; $\overline{y_i}$ – диагноз, выданный системой. Принимается данное выражение за вышеуказанный критерий и обозначается как K.

3. Результаты экспериментов

Теперь можно перейти к практическому рассмотрению указанных выше вопросов.

Прежде всего, производится, как было сказано ранее, кластеризация и, одновременно, строится сеть с константой в «то»-части.

Все данные, которые имеются (193 пациента) отображаются в единичный гиперкуб. $r_a = r_b$. И в таблице будет обозначено RI (range influence).

Количество полученных правил в таблице будет обозначено RQ (rules quantity).

Критерием эффективности работы алгоритма берётся $RMSE = \frac{1}{n}\sqrt{\sum_{i=1}^{n}\left(y_i - \overline{y_i}\right)^2}$.

Дисперсии для гаусовских функций принадлежности для каждого входа будут свои и одинаковые.

Результаты экспериментов заносятся в нижеприведённые таблицы. Отметим отдельно, что строка «Об:Пров» обозначает отношение обучающей выборки к проверочной.

Таблица 1

Об:Пров	**50:50**			**60:40**		
RI	**RQ**	**RMSE train**	**RMSE test**	**RQ**	**RMSE train**	**RMSE test**
0,1	64	$1{,}8315 \cdot 10^{-17}$	0,0332	64	$1{,}5072 \cdot 10^{-17}$	0,0414
0,2	64	$2{,}6832 \cdot 10^{-17}$	0,0312	64	$3{,}1898 \cdot 10^{-17}$	0,0389
0,3	64	$3{,}2746 \cdot 10^{-17}$	0,0297	64	$3{,}7296 \cdot 10^{-17}$	0,0371
0,4	64	$5{,}9620 \cdot 10^{-17}$	0,0282	64	$4{,}5869 \cdot 10^{-17}$	0,0351
0,5	62	$6{,}0560 \cdot 10^{-17}$	0,0277	61	$5{,}4587 \cdot 10^{-17}$	0,036
0,6	55	$2{,}3649 \cdot 10^{-16}$	0,0623	57	$3{,}6669 \cdot 10^{-16}$	0,0387
0,7	47	$2{,}1801 \cdot 10^{-16}$	0,0552	47	$1{,}6546 \cdot 10^{-16}$	0,047
0,8	32	$4{,}3250 \cdot 10^{-16}$	0,0376	33	$2{,}2483 \cdot 10^{-16}$	0,0637
0,9	21	$2{,}7109 \cdot 10^{-16}$	0,0504	27	$4{,}2174 \cdot 10^{-16}$	0,0776
1	19	$6{,}3821 \cdot 10^{-16}$	0,0549	20	$7{,}7099 \cdot 10^{-16}$	0,0912

Таблица 2

Об:Пров	**70:30**			**80:20**		
RI	**RQ**	**RMSE train**	**RMSE test**	**RQ**	**RMSE train**	**RMSE test**
0,1	70	$1{,}4754 \cdot 10^{-17}$	0,0435	89	$1{,}2581 \cdot 10^{-17}$	0,0331
0,2	70	$2{,}1681 \cdot 10^{-17}$	0,0418	89	$2{,}0689 \cdot 10^{-17}$	0,0302
0,3	70	$3{,}3746 \cdot 10^{-17}$	0,0388	89	$2{,}9218 \cdot 10^{-17}$	0,0293
0,4	69	$4{,}7760 \cdot 10^{-17}$	0,0365	88	$4{,}1680 \cdot 10^{-17}$	0,0286
0,5	68	$5{,}2207 \cdot 10^{-17}$	0,0358	85	$5{,}0317 \cdot 10^{-17}$	0,0269
0,6	64	$5{,}6841 \cdot 10^{-17}$	0,0386	74	$8{,}3585 \cdot 10^{-17}$	0,0290
0,7	51	$2{,}0565 \cdot 10^{-16}$	0,0534	57	$2{,}3107 \cdot 10^{-16}$	0,047
0,8	37	$4{,}0681 \cdot 10^{-16}$	0,0967	34	$6{,}5636 \cdot 10^{-16}$	0,0956
0,9	28	$4{,}5956 \cdot 10^{-15}$	0,1212	25	$9{,}5112 \cdot 10^{-16}$	0,0865
1	6	$1{,}2788 \cdot 10^{-15}$	0,1691	5	$2{,}3033 \cdot 10^{-15}$	0,1637

Учебная выборка – 90%, проверочная – 10%.

Таблица 3

RI	**RQ**	**RMSE train**	**RMSE test**
0,1	96	$1{,}1710 \cdot 10^{-17}$	$2{,}9216 \cdot 10^{-18}$
0,2	96	$2{,}1647 \cdot 10^{-17}$	$5{,}9195 \cdot 10^{-17}$
0,3	96	$2{,}6863 \cdot 10^{-17}$	$8{,}1233 \cdot 10^{-17}$
0,4	96	$3{,}3784 \cdot 10^{-17}$	$9{,}6381 \cdot 10^{-17}$
0,5	92	$4{,}4979 \cdot 10^{-17}$	$1{,}3867 \cdot 10^{-16}$
0,6	80	$1{,}6441 \cdot 10^{-16}$	$3{,}0583 \cdot 10^{-16}$
0,7	61	$2{,}2497 \cdot 10^{-16}$	$4{,}5655 \cdot 10^{-16}$
0,8	44	$5{,}5951 \cdot 10^{-16}$	$1{,}3725 \cdot 10^{-15}$
0,9	31	$1{,}0292 \cdot 10^{-15}$	$3{,}4575 \cdot 10^{-15}$
1	5	$1{,}7631 \cdot 10^{-15}$	$3{,}3432 \cdot 10^{-15}$

Таким образом, видно, что данный алгоритм позволяет построить очень хорошую модель диагностирования. Также видно, что качество диагностики возрастает ростом числа правил (падением индекса влияния). Однако, как видно из графиков (Рис. 2, Рис. 3), этот рост монотонным в общем случае назвать нельзя.

Если же характеризовать зависимость «рост качества диагностики – рост объёма обучающей выборки», то следует более внимательно присмотреться к графикам:

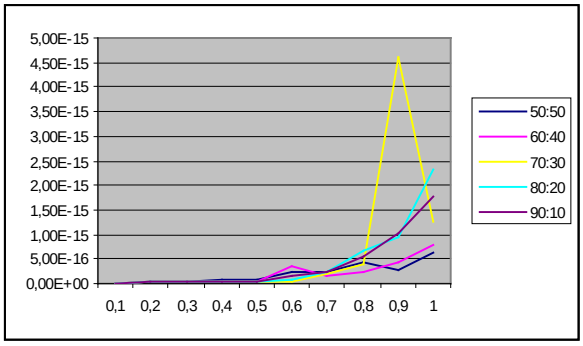


Рис. 1 Зависимость RI – RMSE train для таблиц 1 – 3

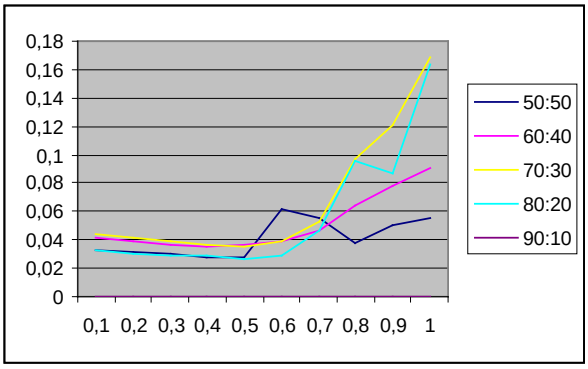


Рис. 2 Зависимость RI – RMSE test для таблиц 1 – 3

Как видно из графиков на рис. 1, для обучающей выборки при $RI = 0{,}1 \div 0{,}5$ будет точно наблюдаться прямая зависимость между качеством диагностики и числом правил нечёткого вывода, а для остальных *RI* четкой зависимости не наблюдается; а для проверочной (рис. 2) – возрастание ошибки до соотношения выборок 7:3 и дальнейшее стремительное падение оной.

Далее рассматривается случай, когда функция в «то»-части линейна.

Задаются следующие начальные условия. Правила и функции принадлежности берутся из предыдущих экспериментов (кластеризации). Обозначения в таблицах аналогичны (хотя RI уже не актуален). Обучение будет проводиться гибридным методом.

Таблица 4

Об:Пров	**50:50**			**60:40**		
RI	**RQ**	**RMSE train**	**RMSE test**	**RQ**	**RMSE train**	**RMSE test**
0,1	64	$2{,}3976 \cdot 10^{-8}$	0,033091	64	$1{,}7003 \cdot 10^{-8}$	0,041256
0,2	64	$2{,}3981 \cdot 10^{-8}$	0,031066	64	$1{,}7008 \cdot 10^{-8}$	0,038731
0,3	64	$2{,}411 \cdot 10^{-8}$	0,029487	64	$1{,}7095 \cdot 10^{-8}$	0,036763
0,4	64	$2{,}5551 \cdot 10^{-8}$	0,027863	64	$1{,}8073 \cdot 10^{-8}$	0,034738
0,5	62	$2{,}235 \cdot 10^{-6}$	0,027419	61	$1{,}6779 \cdot 10^{-6}$	0,035122
0,6	55	$5{,}51 \cdot 10^{-6}$	0,028531	57	$3{,}931 \cdot 10^{-6}$	0,03407
0,7	47	$8{,}112 \cdot 10^{-6}$	0,030635	47	$6{,}314 \cdot 10^{-6}$	0,043796
0,8	32	$8{,}5395 \cdot 10^{-6}$	0,03862	33	$7{,}1297 \cdot 10^{-6}$	0,041018
0,9	21	$1{,}9289 \cdot 10^{-5}$	0,038845	27	$1{,}0663 \cdot 10^{-5}$	0,046861
1	19	$3{,}6898 \cdot 10^{-5}$	0,045584	20	$2{,}8545 \cdot 10^{-5}$	0,047587

Таблица 5

Об:Пров	**70:30**			**80:20**		
RI	**RQ**	**RMSE train**	**RMSE test**	**RQ**	**RMSE train**	**RMSE test**
0,1	70	$1{,}4761 \cdot 10^{-8}$	0,043158	89	$1{,}4916 \cdot 10^{-8}$	0,033651
0,2	70	$1{,}4767 \cdot 10^{-8}$	0,041357	89	$1{,}4919 \cdot 10^{-8}$	0,030443
0,3	70	$1{,}4817 \cdot 10^{-8}$	0,038156	89	$1{,}4985 \cdot 10^{-8}$	0,0293
0,4	69	$5{,}7658 \cdot 10^{-8}$	0,035841	88	$5{,}5153 \cdot 10^{-8}$	0,0288
0,5	68	$1{,}3224 \cdot 10^{-6}$	0,035571	85	$6{,}0647 \cdot 10^{-8}$	0,0291
0,6	64	$3{,}4702 \cdot 10^{-6}$	0,043127	74	$3{,}7547 \cdot 10^{-6}$	0,031
0,7	51	$4{,}9418 \cdot 10^{-6}$	0,047605	57	$6{,}8552 \cdot 10^{-6}$	0,0359
0,8	37	$7{,}3412 \cdot 10^{-6}$	0,048571	34	$1{,}0559 \cdot 10^{-5}$	0,0148
0,9	28	$1{,}0643 \cdot 10^{-5}$	0,046314	25	$1{,}7685 \cdot 10^{-5}$	0,0376
1	6	0,00022524	0,095813	5	$6{,}9307 \cdot 10^{-4}$	0,1178

Обучающая выборка – 90%, проверочная – 10%.

Таблица 6

RI	RQ	RMSE train	RMSE test
0,1	96	$1{,}3748\cdot10^{-8}$	$7{,}3358\cdot10^{-8}$
0,2	96	$1{,}3750\cdot10^{-8}$	$7{,}3361\cdot10^{-8}$
0,3	96	$1{,}3806\cdot10^{-8}$	$7{,}3317\cdot10^{-8}$
0,4	96	$1{,}4470\cdot10^{-8}$	$7{,}3233\cdot10^{-8}$
0,5	92	$5{,}4135\cdot10^{-7}$	$1{,}3909\cdot10^{-6}$
0,6	80	$3{,}2039\cdot10^{-6}$	$9{,}8092\cdot10^{-6}$
0,7	61	$6{,}0106\cdot10^{-6}$	$2{,}4068\cdot10^{-5}$
0,8	44	$9{,}8820\cdot10^{-6}$	$3{,}0978\cdot10^{-5}$
0,9	31	$1{,}6437\cdot10^{-5}$	$6{,}3682\cdot10^{-5}$
1	5	$7{,}0728\cdot10^{-4}$	0,003

Как видно, тенденции здесь аналогичны экспериментам с алгоритмом разностного группирования. Наблюдается зависимость роста качества диагностики с ростом числа правил. Что касается зависимости между объёмом выборки и качеством диагностики, то можно сказать следующее.

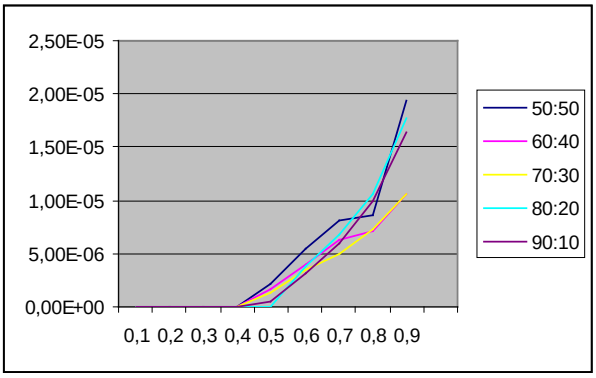


Рис. 3 Зависимость RI – RMSE train для таблиц 4 – 6

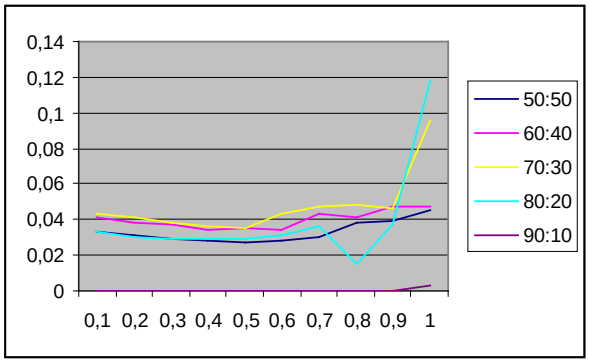


Рис. 4 Зависимость RI – RMSE test для таблиц 4 – 6

На обучающей выборке (рис. 3) эта зависимость не имеет постоянный характер и колеблется. Для проверочной выборки (рис. 4) наблюдается возрастание ошибки с ростом объёма обучающей выборки до соотношения 7:3 и дальнейшее стремительное падение ошибки.

4. Сравнительный анализ алгоритмов

Красноречивее всех описаний говорят результаты, зафиксированные в таблицах 1-3 и 4-6. При функциях-константах в «то»-части правил, полученных после кластеризации система выдавала гораздо лучшие результаты и быстрее шли вычисления, чем линейные функции, получаемые с помощью МНК (см. описание гибридного алгоритма). Необходимо отметить также, что применение для этой же задачи чёткой нейронной сети Back Propagation, рассмотренное в [3], показало, что она по параметру качество диагностики уступает нечёткой сети TSK в обоих случаях. А что касается параметра скорость обучения – уступает случаю функции-константы в «то»-части нечётких правил, но превосходит случай линейной функции.

Однако, как показывают эксперименты, у алгоритмов есть и общие свойства. Из опытов видно (таблицы 1-6), что с ростом числа правил растёт точность диагностики, как для алгоритма разностного группирования (функция-константа), так и гибридного алгоритма (линейная функция). Для обоих алгоритмов наблюдается зависимость падения точности диагностики на проверочной выборке до соотношения выборок 7:3 и дальнейший стремительный рост точности. Однако, что касается обучающей выборки, то по поводу точности диагностики можно сказать, что с ростом объёма растёт точность диагностики для случая функции-константы в «то»-части правил при $RI = 0{,}1 \div 0{,}5$, а про второй случай – чёткой закономерности не наблюдается.

5. Уменьшение числа тестов, необходимых для постановки диагноза

К диагностическим данным применяются инструкции (пункт 2), предложенные ранее

для решения вопроса о количестве тестов, необходимых для диагностики.

Берётся отношение обучающей выборки к проверочной 9:1. Для формирования системы диагностики применяется алгоритм разностного группирования. Обозначается через *n* – число тестов, которые используются для диагностики. RI берётся равным 0,1. *K* – критерий, приведённый во втором пункте.

Результаты занесены в таблицу 7.

Таблица 7

n	RQ	RMSE train	RMSE test	K
2	76	$9{,}1466\cdot10^{-15}$	$2{,}5951\cdot10^{-14}$	$3{,}6710\cdot10^{-13}$
3	88	$9{,}8243\cdot10^{-16}$	$3{,}5033\cdot10^{-15}$	$6{,}1173\cdot10^{-14}$
4	95	$2{,}9052\cdot10^{-17}$	$7{,}9815\cdot10^{-17}$	$7{,}7716\cdot10^{-16}$
5	95	$3{,}2477\cdot10^{-17}$	$7{,}6187\cdot10^{-17}$	$6{,}6613\cdot10^{-16}$
6	96	$2{,}1861\cdot10^{-17}$	$6{,}4230\cdot10^{-17}$	$6{,}6613\cdot10^{-16}$
7	96	$2{,}0654\cdot10^{-17}$	$6{,}5948\cdot10^{-17}$	$7{,}77168\cdot10^{-16}$
8	96	$3{,}5415\cdot10^{-17}$	$6{,}5741\cdot10^{-17}$	$6{,}6613\cdot10^{-16}$
9	96	$1{,}7497\cdot10^{-17}$	$3{,}9496\cdot10^{-17}$	$4{,}4409\cdot10^{-16}$
10	96	$1{,}7743\cdot10^{-17}$	$6{,}2081\cdot10^{-17}$	$8{,}8818\cdot10^{-16}$
11	96	$1{,}5131\cdot10^{-17}$	$3{,}8456\cdot10^{-17}$	$4{,}4409\cdot10^{-16}$
12	96	$2{,}0499\cdot10^{-17}$	$5{,}4912\cdot10^{-17}$	$6{,}6613\cdot10^{-16}$
13	96	$1{,}6495\cdot10^{-17}$	$6{,}60624\cdot10^{-17}$	$6{,}6613\cdot10^{-16}$
14	96	$1{,}8812\cdot10^{-17}$	$4{,}3088\cdot10^{-17}$	$4{,}4409\cdot10^{-16}$
15	96	$1{,}5806\cdot10^{-17}$	$3{,}1263\cdot10^{-17}$	$3{,}3307\cdot10^{-16}$
16	96	$1{,}4932\cdot10^{-17}$	$3{,}5029\cdot10^{-17}$	$4{,}4409\cdot10^{-16}$
17	96	$1{,}7960\cdot10^{-17}$	$3{,}9873\cdot10^{-17}$	$4{,}4409\cdot10^{-16}$
18	96	$1{,}4720\cdot10^{-17}$	$3{,}3978\cdot10^{-17}$	$3{,}3307\cdot10^{-16}$
19	96	$1{,}7069\cdot10^{-17}$	$3{,}5090\cdot10^{-17}$	$3{,}3307\cdot10^{-16}$
20	96	$1{,}2195\cdot10^{-17}$	$3{,}0116\cdot10^{-17}$	$3{,}3307\cdot10^{-16}$
21	96	$1{,}4985\cdot10^{-17}$	$3{,}0080\cdot10^{-17}$	$3{,}3307\cdot10^{-16}$
22	96	$1{,}4824\cdot10^{-17}$	$3{,}5931\cdot10^{-17}$	$4{,}4409\cdot10^{-16}$
23	96	$1{,}5059\cdot10^{-17}$	$2{,}8923\cdot10^{-17}$	$4{,}4409\cdot10^{-16}$
24	96	$1{,}3239\cdot10^{-17}$	$3{,}2038\cdot10^{-17}$	$4{,}4409\cdot10^{-16}$
25	96	$1{,}2587\cdot10^{-17}$	$2{,}3005\cdot10^{-17}$	$2{,}2204\cdot10^{-16}$
26	96	$1{,}5864\cdot10^{-17}$	$2{,}2263\cdot10^{-17}$	$2{,}2204\cdot10^{-16}$
27	96	$1{,}3964\cdot10^{-17}$	$2{,}9947\cdot10^{-17}$	$3{,}3307\cdot10^{-16}$
28	96	$1{,}6132\cdot10^{-17}$	$1{,}7530\cdot10^{-17}$	$2{,}2204\cdot10^{-16}$
29	96	$1{,}2920\cdot10^{-17}$	$1{,}4681\cdot10^{-17}$	$2{,}2204\cdot10^{-16}$
30	96	$1{,}2898\cdot10^{-17}$	$2{,}9216\cdot10^{-18}$	$5{,}5511\cdot10^{-17}$
31	96	$1{,}1404\cdot10^{-17}$	$2{,}9216\cdot10^{-18}$	$5{,}5511\cdot10^{-17}$

Таким образом, видно, что во всех опытах значение критерия *K* удовлетворяется и, следовательно, количество тестов можно значительно уменьшить.

Проводились ещё опыты для различных значений RI с целью добиться наименьшего числа правил при наименьшем числе тестов. Все опыты здесь приведены не будут, но вот несколько примеров:

9 тестов:

Таблица 8

RI	RQ	RMSE train	RMSE check	K
1	2	0,0233	0,0672	0,5364
0,6	11	$5{,}6436\cdot10^{-15}$	$1{,}0704\cdot10^{-14}$	$1{,}2401\cdot10^{-13}$

7 тестов:

Таблица 9

RI	RQ	RMSE train	RMSE check	K
1	3	0,0218	0,0621	0,6588
0,7	6	0,0164	0,0647	0,7306
0,5	32	$5{,}784\cdot10^{-15}$	$1{,}922\cdot10^{-14}$	$1{,}1446\cdot10^{-13}$

Эксперименты показали, что сокращать количество тестов можно. Однако не следует считать, что так же легко их можно уменьшить на практике. Во-первых, врачу необходимо обезопасить себя от поломки оборудования, которое производит тест. Во-вторых, есть заболевания, для которых значения тестов могут быть одинаковые и, поэтому, необходимо проводить дополнительные тесты для определения вида заболевания (в данном случае – действительно ли это рак, или фоновое заболевание). В-третьих, очевидно, что человек не может оперировать одновременно таким большим числом информации, которое, де-факто, требуют от него результаты экспериментов. Естественно в мозгу врача-диагноста происходят несколько отличные процессы от модели. И т.д.

Выводы

В данной работе был проведён анализ применения нечёткой нейронной сети TSK к задаче медицинской. Было проведено ряд вычислительных экспериментов, в результате которых были установлены некоторые факты. Первое, нечёткая нейронная сеть Такаги-Сугено-Канга применима для рассматриваемой задачи. Второе, для задачи, рассматриваемой в работе, применение функции-константы в «то»-части нечётких правил оказалось более эффективным, чем применение линейной функции. Третье, наблюдалась чёткая зависимость роста ошибки на проверочной выборке от соотношения выборок 50:50 до 70:30 в обоих случаях, в то же время на обучающей выборке никакие закономерности не просматривались за исключением случая функции-константы в «то»-части правила и только для $RI = \overline{0{,}1;0{,}5}$. Четвёртое, количество входов системы (тестов) можно значительно сократить. Однако на практике это не стоит делать по указанным в статье причинам. Полученные результаты подтверждают эффективность применения нечёткой нейронной сети TSK к задаче медицинской диагностики, однако ряд рассуждений показывают, что оптимальная система для решения данной задачи не найдена. Поиск данной системы и составит предмет последующих работ.

Список литературы

1. Steve Saggese, Trevor Johnson, Ivy Basco, Yalew Tamrat. Automatic Segmentation of Uterine Cervix for in vivo localization and Identification of Cervical Intraepithelial Neoplasia.– Apogen Technologies 7545 Metropolitan Dr San Diego, CA 92108.
2. Зайченко Ю.П. Основи проектування інтелектуальних систем. Навчальний посібник. – К.: Видавничий Дім «Слово», 2004. – 352 с.
3. Мурга Н.А. Применение нейросетей при колпоскопическом осмотре шейки матки с целью обнаружения кансероподобных образований. Системний аналіз та інформаційні технології: Матеріали X Міжнародної науково-технічної конференції (20-24 травня 2008 р., Київ).–К.:НТУУ «КПІ» 2008.– 235 сторінка.

УДК 004.052.42

АБУ УСБАХ А.Н.,
МЕЛЬНИК А.П.,
ПОНОМАРЧУК Д.С.

ОБ ОДНОМ ПОДХОДЕ К ОБНАРУЖЕНИЮ ОШИБОК ПЕРЕДАЧИ ДАННЫХ НА ОСНОВЕ СИСТЕМ ОСТАТОЧНЫХ КЛАССОВ

Статья посвящена решению проблемы повышения эффективности выявления ошибок передачи данных в каналах со спектральной модуляцией за счет учета природы их возникновения. С целью гарантированного обнаружения канальных ошибок одной и более кратности был предложен подход, использующий модульное представление на основе Китайской теоремы об остатках. Проведенные теоретические исследования предложенного подхода позволили доказать гарантированность обнаружения символьных ошибок кратностью менее или равной числу контрольных символов в рассматриваемом модульном представлении.

The paper is dedicated to solving the efficiency increasing problem for data transmission error detection in spectrum modulation channel by properties such errors appearance accounted. For the guaranteed errors detection in one and more channel symbols the approach based on Chinese Reminder Theorem has been proposed. In the course of the theoretical researches of proposed approach the guaranteed error detection has been proved for errors quantity less or equal to number of the check symbols in modular representation.

Введение

Развитие и интеграция средств передачи цифровых данных играет важную роль на современном этапе развития компьютерных и информационных технологий. Передача данных в цифровом виде находит применение в таких областях как мобильная связь (EDGE, CDMA2000, 1xEV-DO), цифровое телевидение (DVB-T/H/C/S/S2, ATSC), телекоммуникационных системах беспроводной передачи данных (Wi-Fi, Wi-MAX, Bluetooth), компьютерные сети и др [1]. Характерной особенностью современных средств передачи данных является использование различных видов спектральной модуляции (фазовой-PSK, квадратурно-амплитудной-QAM) для повышения пропускной способности канала. При этом одно изменение фазы/амплитуды может нести в себе более одного информационного бита – канальный символ. Например: QPSK – канальный символ 2 бита, 8-PSK – 3 бита, 16-,32-,64-,128- и 256-QAM – 4,5,6,7 и 8 бит соответственно.

Рост пропускной способности каналов передачи цифровых данных осложняется помехами при передаче и требует адекватного развития средств обнаружения ошибок. Для значительной части систем передачи цифровых данных исправление ошибок выполняется путем повторной передачи блока. Для таких систем, в отличие от корректирующих кодов, фазы обнаружение ошибок и их исправления разнесены. При использовании спектральной модуляции доминируют многократные битовые искажения данных, поскольку одиночная ошибка передачи канального сигнала может изменить произвольное число бит канального символа. Эту особенность битовых искажений необходимо учитывать при создании средств контроля ошибок в каналах со спектральной модуляцией.

Таким образом, проблема повышения эффективности обнаружения многократных битовых искажений в каналах со спектральной модуляцией является актуальной и имеющей практическую значимость.

Анализ существующих средств обнаружения многократных ошибок

В каналах передачи данных со спектральной модуляцией цифровая информация фактически передается символами, каждый из которых модулируется одним канальным сигналом.

Контролируемый блок B, содержащий m бит: $B=\{b_1,b_2,\ldots,b_m\}$, $b_l\in\{0,1\}$, $l=1,\ldots,m$ можно рассматривать состоящим из $t = m/k$ канальных символов: $B=\{X_1,X_2,\ldots,X_t\}$. Каждый j-тый из этих символов X_j, $j\in\{1,2,\ldots,t\}$ включает k смежных бит $X_j=\{x_1,x_2,...,x_k\}=\{b_{(j-1)\cdot k+1},b_{(j-1)\cdot k+2},...,b_{j\cdot k}\}$ контролируемого блока.

При передаче цифровых данных для обнаружения ошибок наиболее часто используют циклические коды и в частности CRC (Cyclic Redundancy Check – циклическая избыточная

проверка). Как и контрольные суммы, CRC относятся к средствам блокового контроля.

Сущность контроля с использованием CRC состоит в том, что блок $B=\{b_1,b_2,\ldots,b_m\}$ представляется полином $P(B)$ степени $m+k$: $P(B)=b_1\cdot x^k+b_2\cdot x^{k+1}+b_3\cdot x^{k+2}+\ldots+b_{m-1}\cdot x^{k+m-1}+b_m\cdot x^{k+m}$. Контрольный код $R(B)$ вычисляется как остаток от деления полинома $P(B)$на образующий полином $Q(X)$ степени k CRC.

По основному критерию эффективности – надежности обнаружения ошибок, циклические коды превосходят контрольные суммы. Ошибки при передаче блока данных не обнаруживается, если полином $E(X)$, соответствующий вектору ошибки делится на образующий полином $Q(X)$ CRC без остатка. Показано [3], что все полиномы $E(X)$, соответствующие указанным ниже ошибкам не делятся на специальным образом выбранный базовый полином $Q(X)$, а следовательно, они обнаруживаются гарантированно:

1. Все искажения битов $b_1,b_2,\ldots,b_m$ нечетной кратности, если базовый полином $Q(X)$ может быть представлен в виде произведения полиномов: $Q(X)=(x+1)\cdot S(X)$;
2. Все двукратные искажения битов $b_1,b_2,\ldots,b_m$ контролируемого блока, если базовый полином $Q(X)=q_0+q_1\cdot x+q_2\cdot x^2+\ldots+q_k\cdot x^k$ содержит не менее трех ненулевых компонент.
3. Группа ошибок, локализованные в рамках k разрядов.

Для остальных ошибок показано [3], что остаток $R(B)$ представляет собой результат хеширования блока B данных в пространство 2^k всех возможных контрольных кодов. Соответственно, вероятность P_{CRC} того, что эти ошибки не будут обнаружены с использованием CRC с образующим полиномом $Q(X)$ степени k определяется как $P_{CRC}=2^{-k}$.

В каналах со спектральной модуляцией, CRC позволяет гарантированно обнаружить только однократную ошибку передачи модулированного сигнала, поскольку искажаемые при этом биты локализованы в рамках группы из k битовых позиций, а значение k на практике меньше степени образующего полинома CRC. При большей кратности ошибок модулированного сигналов, CRC не гарантирует их обнаружения.

Наряду с высокой надежностью, CRC обладает рядом недостатков, наиболее важным из которых является принципиально последовательный характер вычисления контрольного кода, что обуславливает существование ограничений на скорость выполнения операций, связанных с контролем ошибок. Этот недостаток особенно актуален в современных условиях быстрого роста скоростей передачи данных.

Другим эффективным средством обнаружения ошибок в каналах со спектральной модуляцией являются взвешенные контрольные суммы (Weighed Check Sum – WCS) [4]. В отличие от CRC, использование технологии WCS учитывает особенности возникновения битовых искажений в каналах со спектральной модуляцией и позволяет гарантированно обнаруживать битовые искажения, возникающие в двух символах. Недостатком WCS является то, что она не позволяет гарантированно обнаруживать битовые искажения, вызванные ошибочной передачей более 2-х канальных сигналов. Существенным недостатком WCS является также значительно большее по сравнению с CRC число контрольных разрядов, равное $k\cdot(1+\log_2 t)$.

Таким образом, существующие средства обнаружения многократных ошибок в каналах со спектральной модуляцией не обеспечивают эффективное решение обнаружения битовых искажений, вызванных ошибочной передачей более 2-х канальных сигналов.

Целью работы является создание эффективного способа обнаружения битовых искажений, вызванных многократными ошибками передачи канальных сигналов.

Организация контроля ошибок при представлении передаваемых данных в системах остаточных классов

Для гарантированного обнаружения ошибок в одном и более канальном символе необходимо учитывать символьную структуру передаваемых данных. Для этого предлагается использовать метод, основанный на представлении целых чисел в системах остаточных классов – СОК.

Пусть существует система n взаимнопростых чисел $\{p\}^n=\{p_1,p_2,\ldots,p_n\}$ и число P, равное их произведению:

$$P=\prod_{j=1}^{n}p_j \qquad (1)$$

Тогда, согласно Китайской теореме об остатках [1] любое число M, принадлежащее интервалу [0…P-1] может быть однозначно представлено множеством остатков от деления на взаимно-простые числа, образующие систему $\{p\}^n$.

$$\forall M \in \{0,...,P-1\} : \{r_1, r_2,..., r_n\} \Leftrightarrow M, r_j = M \bmod p_j,\ j = 1,...,n \quad (2)$$

где $\Leftrightarrow$ символ, обозначающий взаимно-однозначное соответствие. представление числа M (2) называется представлением числа M в системе остаточных классов (СОК). Всегда можно выбрать такую систему взаимно-простых чисел, что ее элементы $p_1,\ldots,p_n$ будут удовлетворять условию:

$$\forall j \in \{1,...,n\} : p_j \geq 2^k \quad (3)$$

Пусть число элементов системы $\{p\}^n$ будет больше числа символов в блоке: $n>t$ $\{p\}^t = \{p_1,p_2,\ldots,p_t\} \subseteq \{p\}^n$. Тогда, согласно (2) и (3) последовательность символов $X_1,X_2,\ldots,X_t$, составляющих контролируемый блок B данных можно рассматривать как представление в системе остаточных классов некоторого числа A:

$$B = \{X_1,..., X_t, X_{t+1},..., X_n\} \Leftrightarrow A, \quad \forall i \in \{1,...,t\} : X_i = A \bmod p_i \quad (4)$$

То же число A может быть представлено в системе остаточных классов на основе $\{p\}^n$:

$$\{X_1,..., X_t, X_{t+1},...., X_n\} \Leftrightarrow A, \quad \forall j \in \{1,...,n\} : X_j = A \bmod p_j \quad (5)$$

Изложенные теоретические положения могут быть положены в основу метода контроля ошибок, возникающих в последовательных интерфейсах со спектральной модуляцией.

Сущность предлагаемого метода состоит в следующем:

1. Выбирается система $\{p\}^n$ взаимно-простых чисел, в зависимости от типа спектральной модуляции определяется число k бит, модулируемых одном сигналом, протоколом определяется число t пересылаемых символов в блоке.
2. Согласно (5) передаваемый блок B данных представляется в выбранной системе $\{p\}^n$ числом A. При этом символы $X_1,X_2,\ldots,X_t$ являются информационной частью передаваемого блока B, а символы $X_t,\ldots,X_n$ образуют контрольные символы. Обозначим через T передаваемый блок, представляющий собой конкатенацию информационного блока B и блока Z контрольных символов: $T = B \| Z$.
3. Информационная и контрольная части блока передаются приемнику.
4. Принятый блок представляет собой посимвольную сумму переданного блока T и n-символьного вектора ошибки $E=\{e_1,e_2,\ldots,e_n\}$, где e_j – k-битовый символ: $e_j = \{\xi_{1j}, \xi_{2j},\ldots,\xi_{kj}\}$, компоненты которого $\xi \in \{0,1\}$:

$$R = \{Y_1,..., Y_n\} = T + E \quad (6)$$

5. Проверка отсутствия ошибок в передаче блока состоит в восстановлении, согласно Китайской теореме об остатках, некоторых чисел C_1 и C_2 [2]:

$$\{Y_1, Y_2,..., Y_t\} \Leftrightarrow C_1, \{Y_1, Y_2,...., Y_n\} \Leftrightarrow C_2 \quad (7)$$

При этом, информация считается переданной без ошибок, если выполняется равенство указанных чисел C_1 и C_2: $C_1=C_2$.

Можно показать, что предложенный метод контроля гарантирует обнаружение ошибок в передаче до (n-t) символов, если выполняются следующие условия:

а). Система $\{p\}^n$ является монотонно возрастающей последовательностью.

б). Произведение модулей $p_{t+1},p_{t+2},\ldots,p_n$ контрольных символов не сравнимо с единицей по любому из модулей символов данных блока B:

$$\prod_{i=t+1}^{n} p_i \bmod p_j \neq 1, \forall j \in \{1,...,t\} \quad (8)$$

Действительно, ошибки в передаче канальных символов могут вызвать искажения битов символов информационного блока B, контрольного блока Z, информационного и контрольного блоков одновременно. Это требует доказательства для ряда частных случаев.

Частный случай А: Искажению подверглись символы только контрольного блока. В этом случае, поскольку, ошибки не затронули символов информационного блока B, то справедливо следующее:

$$\{X_1, X_2,..., X_t\} = \{Y_1, Y_{2,},..., Y_t\} : C_1 = A \quad (9)$$

В соответствии с Китайкой теоремой остатках, представление в системе остаточных классов однозначно, следовательно, двум различным множествам остатков соответствуют различные целые числа:

$$\{X_1,..., X_n\} \Leftrightarrow A, \{X_1,..., X_t, Y_{t+1},...., Y_n\} \Leftrightarrow C_2 \neq A \quad (10)$$

Исходя из (9) и (10) справедливо следующее: $C_1 = A \neq C_2$. Из этого следует, что $C_1 \neq C_2$,

то есть ошибка гарантированно обнаруживается, если число неверно переданных канальных сигналов не превышает *n-t*.

Частный случай B: ошибки имеют место как при передаче информационных символов, так и при передаче символов контрольного блока. В этом случае, не нарушая общности, можно рассматривать вариант локализации ошибок, при котором ошибки произошли при передаче *d* последних символов и *q* последних символов контрольного блока.

$$\begin{aligned} &A \Leftrightarrow \{X_1,...,X_t\} \\ &A \Leftrightarrow \{X_1,....X_n\} \\ &C_1 \Leftrightarrow \{X_1,...,X_{t-d}, X_{t-d+1}+e_{t-d+1},...,X_t+e_t\} \\ &C_2 \Leftrightarrow \{X_1,...,X_{t-d}, X_{t-d+1}+e_{t-d+1},....,X_t+e_t, \\ &X_{t+1},...,X_{n-q}, X_{n-q+1}+e_{n-q+1}, X_n+e_n\} \end{aligned} \tag{11}$$

Согласно Китайской теореме об остатках справедливо следующее:

$$\begin{aligned} &M=\sum_{i=1}^{n} c_i, g_i, r_i \bmod P, \{r_1.r_2,...,r_n\} \Leftrightarrow M, \\ &c_i = P/p_i; g_i \cdot c_i = 1 \bmod p_i \end{aligned} \tag{12}$$

Выражения (11) можно представить, с учетом (12), в следующем виде:

$$\begin{aligned} &A=(\sum_{i=1}^{t} c_i \cdot g_i \cdot X_i) \bmod P^t \\ &A=(\sum_{i=1}^{t} c_i' \cdot g_i' \cdot X_i) \bmod P \\ &C_1=(\sum_{i=1}^{t} c_i \cdot g_i \cdot X_i + \sum_{j=t-d+1}^{t} c_j \cdot g_j \cdot e_j) \bmod P^t \end{aligned} \tag{13}$$

где $P^t = \prod_{i=1}^{t} p_i$, $P = P^t \cdot \prod_{j=t+1}^{n} p_j = P^t \cdot P^{C_1} \cdot P^{C_2}$,

$P^{C_1} = \prod_{l=t+1}^{n-q} p_l$ – произведение модулей, соответствующих символам контрольного кода, которые переданы без ошибок.

$P^{C_2} = \prod_{h=n-q+1}^{n} p_h$ – произведение модулей, соответствующих символам контрольного кода, переданных с ошибками.

Если предположить, что ошибки не обнаруживаются, то тогда верно:

$$C_1 = C_2 \Rightarrow C_1 - A = C_2 - A \tag{14}$$

Подставив (13) в (14) получим следующее:

$$\begin{aligned} &(\sum_{j=t-d+1}^{t} c_j \cdot g_j \cdot e_j) \bmod P^t = \\ &=(\sum_{j=t-d+1}^{t} c_j' \cdot g_j' \cdot e_j + \sum_{l=n-q+1}^{n} c_l' \cdot g_l' \cdot e_l) \bmod P \end{aligned} \tag{15}$$

Согласно (13) и (14) справедливо:

$$\begin{aligned} &P = P^t \cdot P^{C_1} \cdot P^{C_2} \\ &c_j' = c_j \cdot P^{C_1} \cdot P^{C_2}, \forall j \in \{1,...,t\} \\ &c_l' = P^t \cdot P^{C_1} \cdot \frac{P^{C_2}}{p_l} = P^t \cdot P^{C_1} \cdot P^{C_2/l}, \forall l \in \{t+1,...,n\} \end{aligned} \tag{16}$$

где $P^{C_2/l}$ – произведение модулей, соответствующих символам контрольного блока, переданным с ошибками, за исключением *l*-того символа.

Пусть произведение модулей символов информационного блока, при передаче которых не произошло ошибок, равно P^{tr}, а произведение модулей символов информационного блока, при передаче которых произошли ошибки равно - P^{te}:

$$\begin{aligned} &P^t = P^{tr} \cdot P^{te} \\ &P = P^{tr} \cdot P^{te} \cdot P^{C_1} \cdot P^{C_2} \\ &c_j = P^{tr} \cdot \frac{P^{te}}{p_j} = P^{tr} \cdot P^{te/j} \end{aligned} \tag{17}$$

где $P^{te/j}$– произведение модулей, соответствующих символам информационного блока, при передаче которых произошли ошибки, кроме *j*–го.

С учетом ранее полученных выражений (15) – (17) справедливо следующее:

$$\begin{aligned} &(P^{tr} \cdot \sum_{j=t-d+1}^{t} P^{te/j} \cdot g_j \cdot e_j) \bmod P^{tr} \cdot P^{te} = \\ &=(P^{C_1} \cdot (P^{C_2} \cdot \sum_{j=t-d+1}^{t} P^{te/j} \cdot g_j' \cdot e_j + \\ &+P^{te} \cdot \sum_{l=n-q+1}^{n} P^{C_2/l} \cdot g_l' \cdot e_l)) \bmod P^{tr} P^{te\bullet} \cdot P^{C_1} \cdot P^{C_2} \end{aligned} \tag{18}$$

В выражении (18) левая и правая части могут быть сокращены на P^{tr}, поскольку оба модуля и подмодульных выражения делятся на P^{tr}:

$$(\sum P^{te/j} \cdot g_j \cdot e_j) \bmod P^{te} =$$

$$= (P^{C_1} \cdot (P^{C_2} \cdot \sum_{j=t-d+1}^{t} P^{te/j} \cdot g_j' \cdot e_j + \qquad (19)$$

$$+ P^{te} \sum P^{te/l} \cdot g_l' \cdot e_l)) \bmod P^{te} P^{C_1} P^{C_2}$$

Согласно свойству сравнений в выражении (19) правая часть делится на P^{C1}, поскольку и модуль и подмодульное выражение делятся на P^{C1}. Следовательно, и левая часть выражения (19) делится на P^{C1}.

Одним из возможных решений уравнения (19) является нулевое значение левой и правой части. Множители как левой, так и правой частей (19) не равны нулю, поскольку из (12) следует, что:

$$\begin{aligned} g_j &= 1..p_j - 1 \\ e_j &= 1..p_j - 1 \end{aligned} \qquad (20)$$

Следовательно, равенство нулю левой и правой частей уравнения (19) выполняется при справедливости следующей системы:

$$\begin{aligned} g_j \cdot e_j &= p_j, \forall j \in \{t-d+1,...,t\} \\ g_j' \cdot e_j &= p_j, \forall j \in \{t-d+1,...,t\} \\ g_l' \cdot e_l &= p_l, \forall l \in \{n-q+1,...,t\} \end{aligned} \qquad (21)$$

Из (21) следует: $g_j = g_j', \forall j \in \{t-d+1,...,t\}$. Из этого очевидным является следующее: $c_j \bmod p_j = c_j' \bmod p_j = c_j \cdot P^{C_1} \bmod p_j$. Из последнего следует, что $P^{C_1} \bmod p_j = 1$. Однако это противоречит условию (7). Следовательно, левая и правая части уравнения (19) не могут быть равными нулю. Левая часть выражения (19) не может превышать модуль, равный P^{te}. Достаточным условием того, чтобы выражение (19) не выполнялось является условие:

$$P^{te} < P^{C_1} \qquad (22)$$

По условию а) последовательность модулей монотонно возрастает, следовательно, неравенство (22) будет иметь место, если число модулей в произведении левой части (22) не превышает число модулей в произведении правой части (22). Таким образом, будут гарантированно обнаружены ошибки передачи канальных сигналов, кратность которых не соответствовать контрольным разрядам (16 бит контроля), следовательно, разрядам данных будет сопоставлен 51 модуль. Это позволяет контролировать блок данных длиной 51*4=204 бита. Следует отметить, что рост α приводит к экспоненциальному увеличению

превышает $u<n-t$ ошибок в контрольных символах и $n-t-u$ ошибок передачи символов информационного блока. Таким образом, количество h гарантированно обнаруживаемых ошибок передачи символов определяется следующей формулой:

$$h = n - t \qquad (23)$$

Важным аспектом эффективности предложенного метода является анализ количества контрольных разрядов, используемых для обнаружения битовых искажений, вызванных ошибками передачи канальных сигналов.

Следует учитывать, что в предложенном методе разрядность контролируемых символов и символов контроля может быть различной, при условии, что обе указанные величины кратны разрядности канального символа. Это условие обеспечивает локализацию ошибок – ошибка в одном канальном символе приведет к ошибке только в одном символе данных или контроля. В простейшем случае контролируемый символ совпадает с канальным, а разрядность контрольных символов кратна разрядности канальных символов.

Проведенный анализ показал, что предложенный метод имеет ряд ограничений накладываемых необходимостью иметь монотонно возрастающую последовательность взаимно простых модулей удовлетворяющих (8). Объективно, длина такой последовательности, а значит, и размер контролируемого блока данных непосредственно зависит от разрядности символов контроля, которая определяет верхнюю границу для модулей. Все члены последовательности должны быть $p_j < 2^{\alpha}$, $j=1..n$, где α – разрядность символов контроля. Также влияет на длину последовательности модулей разрядность контролируемых символов данных, поскольку минимальный член последовательности должен быть больше любого контролируемого символа, т.е. все члены последовательности должны быть $p_j \geq 2^k$, $j=1..n$. Значение h определяет какое число модулей будет использовано для функций контроля, следовательно уменьшает число модулей для данных. Таким образом, например, при h=2, k=4, α=8 последовательность модулей будет иметь 53 члена, из которых 2 старших будут

длинны контролируемых данных. Так, при увеличении α в два раза (32 бита контроля) можно контролировать блок данных длиной 26156 бит. Другим способом увеличения размера контролируемого блока является использование разрядности контролируемых символов кратной разрядности канальных символов. Однако при этом рост будет линейным.

Таким образом, предложенный метод обнаружения ошибок в модемных интерфейсах со спектральной модуляцией эффективен при относительно больших значениях α и k. Учитывая, что с развитием технологии передачи цифровых данных значение k увеличивается, перспективность предложенного метода обнаружения ошибок в перспективе увеличивается. Не взирая на то, что предложенный метод не позволяет контролировать блоки данных произвольной длины, он, в отличие от CRC, обеспечивает гарантированное обнаружение ошибок и может адаптироваться под качество канала передачи путем использования большего или меньшего числа контрольных разрядов.

Выводы

Предложен метод контроля ошибок в каналах со спектральной модуляцией, основанный на представлении информационных и проверочных символов в СОК.

Метод позволяет повысить эффективность обнаружения ошибок в каналах со спектральной модуляций благодаря тому, что учитывается символьная природа таких ошибок. Предложенный метод позволяет гарантированно обнаруживать пачки битовых ошибок любой длины, а также адаптировать процедуру контроля под качество линии передачи.

Метод может использоваться при модификации существующих и разработке новых протоколов передачи цифровых данных в различных отраслях – мобильной связи, цифровом телевещании, беспроводной передачи и др.

Наиболее перспективным направлением развития предложенного подхода является разработка методов коррекции ошибок с использованием представления в СОК.

Список литературы

1. Анісімов А.В. Алгоритмічна теорія великих чисел. –К.: Академперіодика.–2001.–153 с.
2. Бєйкер А. Введение в теорию чисел. Мн.: Вышєйш. шк., –1999.–340 с.
3. Скляр Б. Цифровая связь. Теоретические основы и практическое применение. М.: Издательский дом ”Вильямс”, 2004.– 1104 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

УДК 004.415.2.043

САЛАПАТОВ В.І.

СИНТАКСИЧНИЙ АНАЛІЗ ІЗ РОЗПОДІЛОМ ЛЕКСЕМ НА ГРУПИ

Традиційний розбір операторів мов програмування із розбиттям їх на окремі лексеми як рівноправні одиниці мови достатньо ускладнює синтаксичний аналіз. Природні мови спілкування мають при їхньому сприйнятті відокремлення підмету, присудку та інших частин мови для усвідомлення сенсу речення. Дещо подібне пропонується на етапі лексичного розбору операторів мов програмування. Всі лексеми при цьому розбиваються на три групи: лексеми-об'єкти, лексеми дії та інші. Особливість мов програмування полягає у обов'язковій наявності пари лексем: лексеми-об'єкту та лексеми дії. Причому з цієї пари починаються всі оператори і обов'язковим елементом у цій парі завжди є лексема дії. Тому, якщо при лексичному аналізі з'являється лексема-об'єкт, то доречним є одночасний пошук відповідної лексеми дії. Такий підхід дозволяє значно спростити синтаксичний аналіз операторів мови та прискорити його виконання.

The traditional analysis of program language operators if with laying out of them on separate lexemes as equal in rights units of language complicates enough the syntactic analysis. The natural languages of intercourse are had at their perception of separation to the subject, to the predicate and other parts of language for the awareness of sense of suggestion. Something similar is offered on the stage of lexical analysis of program language operators. All lexemes are here divided into three groups: lexemes-objects, lexemes of action at al. A feature as programming if consists in the obligatory presence of pair of lexemes: lexeme-object and lexeme of action. Thus from this pair all operators begin and in this pair it is always the lexeme of action is a mandatory member. Therefore, if a lexeme-object appear at the lexical analysis, the simultaneous search of the proper lexeme of action is appropriate. Such approach allows considerably simplify the syntactic analysis of operators of language and accelelerate his implementation.

Ефективність засобів компіляції

Потреба в нових мовах програмування та створенні для них систем компіляції ставить задачу щодо їхньої ефективної та швидкої розробки. Сучасні системи компіляції ще й досі породжують надто надмірні коди програм. Тому методи створення ефективних систем компіляції містять великий резерв щодо підвищення продуктивності роботи комп'ютерів і на даний час залишаються вельми актуальними. Важливим компонентом систем компіляції є синтаксичний аналізатор, який перетворює текст програми у внутрішнє уявлення. Математичною моделлю аналізатора є граматика мови програмування. Контекстно вільна граматика описує всі оператори мови і складається з низки правил, у правій частині яких можуть бути присутніми нетермінали. У мовах програмування такими нетерміналами є такі поняття, як вираз, оператор або інструкція та окремі фрази від часток операторів.

Створення сучасних синтаксичних аналізаторів пов'язано з двома різновидами аналізу – згори донизу та знизу догори, а точніше їхніми різновидами LL-аналізу та LR-аналізу [1, 2]. Останні два алгоритми передбачають посимвольний перегляд тексту програми з кроком уперед. При цьому всі елементи оператора мають оброблятися за єдиним алгоритмом. Це вносить певні складнощі, оскільки нетермінали, які мають бути присутніми у певних місцях правил граматики, описуються за своїми власними правилами. Ці правила мають вигляд $G = f(T, N, P, N_s)$, де N_s – стартовий символ (спеціальний нетермінал), який визначає тип оператора. T являє собою множину терміналів (в мові програмування це константи, ідентифікатори, ключові слова, символи пунктуації і т.і.). N – це нетермінали (в нашому випадку такі поняття, як вирази, оператори та окремі частки операторів). В зв'язку з цим доречно розглядати граматику кожного нетерміналу та його обробку окремо. Так, якщо згідно з граматикою у певному місці оператора має бути вираз, то обробку оператора мови в цій частині треба виконувати згідно граматики виразу, тобто окремою підпрограмою. При цьому ознакою кінця виразу є або ключове слово з конструкції оператора, або спеціальні знаки-розподільники.

Поєднання в одному аналізі розбору оператора, розбору виразу та інших частин оператора ускладнює та уповільнює процес компіляції в цілому. Тому розробка нових ефектив-

них засобів синтаксичного аналізу у мовах програмування на даний час є все ще актуальною.

Огляд методів синтаксичного аналізу

З моменту появи компіляторів і до даного часу алгоритми синтаксичного аналізу майже не змінились. При цьому всі лексеми мови розбираються однаково згідно одного з обраних алгоритмів [1, 2, 3]. Але одні лексеми вимагають виконання певних дій (оператори, знаки арифметичних операцій, фразові ключові слова), а інші являють собою дані, над якими виконуються дії. Синтаксичний розбір операторів за конкретним алгоритмом згідно граматики кожного разу вносить певні корективи у всі попередні алгоритми синтаксичного розбору. Тому кожного разу, при створенні нової мови програмування і, відповідно, нового компілятора приходиться відтворювати такий алгоритм заново. Хоча на даний час створено багато засобів по створенню систем компіляції (компілятори компіляторів) [2], але за своєю ефективністю вони значно поступаються тим, що заново створюються. На жаль, в цьому компоненті за останні роки майже нічого не змінилось. Тому виникла нагальна потреба створити такий алгоритм аналізатора, який складався б із загальної частини, яка присутня в усіх компіляторах та спеціальної частини, де б можна було врахувати особливості конкретного компілятора. До загальної частини можна віднести обробку ключових слів, констант, змінних, знаків пунктуації та арифметичних дій. Задача полягає у розробці такого універсального алгоритму синтаксичного аналізатора, який би мав можливість налаштовуватись на конкретний алгоритм згідно граматики мови та не вносив би надмірність у машинний код.

Розподіл лексем

Огляд існуючих мов програмування показав, що всі лексеми, які являють собою термінальні символи, можна розділити на дві великі групи: лексеми-об’єкти та лексеми дії. Перші описують данні, а другі – дії над цими даними так само, як це має місце у різних системах: алгебрі, геометрії, базах даних і т.і. Таким чином, $L \rightarrow L_d + L_a$, де L – множина всіх лексем, L_d - лексеми даних, L_a – лексеми дії. Припустимість тих чи інших дій задається правилами граматики. Лексеми, які визначаються за допомогою символів, можуть бути визначені через таблиці символів, ключові слова – за допомогою таблиці ключових слів. У правій частині правил граматики для операторів мов програмування присутні нетермінальні символи, які, у свою чергу, можуть бути описані окремими правилами. До таких нетерміналів відносяться вирази та оператори мови. Вирази описуються граматикою, яка співпадає з правилами арифметичний дій, тобто існує пріоритетність виконання операторів, урахування дужок, які змінюють пріоритетність дій, та деякі інші. Вирази є одним з основних компонентів мов програмування. В залежності від типу результату у виразах припускаються дані певного типу та відповідно дії, що припустимі для обробки цих даних. Те саме виникає і при обробці операторів. За ключовими словами операторів згідно граматик розміщуються вирази. При цьому ознаками кінця виразу можуть бути як ключові слова операторів, так і спеціальні символи.

Лексеми даних - L_d обробляються лексичним аналізатором під час їх появи. Тип результату визначається припустимими термінами дії - L_a. Ці лексеми визначають порядок їхньої обробки згідно пріоритетів. Це в значній мірі спрощує опис правил граматики. Ключові слова операторів обробляються в останню чергу, тому вони мають найнижчий пріоритет. Результатом їх обробки є породження гілок дерева поточного оператора. Треба зазначити, що даний алгоритм аналізу є модифікацією алгоритму “знизу догори”, тобто від кінцівок дерева до його кореня. Стартовий символ N_s, тобто тип оператора мови, розпізнається через першу або другу лексему, за один крок роботи лексичного аналізатора, про що йдеться далі.

Модифікований стековий алгоритм

У виразах усі дії, за деяким виключенням, є бінарними, тобто вони вимагають наявності двох операндів. Через це доцільно розпізнавати пари лексем у такому порядку: лексема-об’єкт, лексема дії. Лексема-об’єкт у нашому випадку являє собою дані (для виразів це константи, змінні, масиви), а лексема дії вказує на обробку даних. Опис лексем-об’єктів у мовах програмування в основному стандартний, відмінності пов’язані, головним чином, з ідентифікаторами. Лексеми дії також мають стандартний за семантикою набір, що присутній в

усіх мовах програмування, та спеціальний набір тільки для конкретної мови.

Унарними є такі дії, як унарний мінус, дужки та деякі інші. Для унарного мінуса можна створити фіктивний операнд – нуль, а для дужок – пустий операнд.

Пріоритетність дій у виразах є найбільш зрозумілою і визначає порядок виконання цих дій. Якщо пріоритет поточної лексеми дії не перевищує пріоритет попередньої, то виконується обробка попередньої лексеми дії. Інакше обробка відкладається у стек. Таким чином, маємо порядок обробки лексем дії, що представлений нижче у табл. 1.

Табл. 1

Порівняння пріоритету поточної операції з попередньою, перевірка стеку	**Дія**
не перевищує	обробка попередньої операції з поточною лексемою-об'єктом
перевищує	запис у стек попередньої пари: лексема-об'єкт, лексема дії
стек непустий, права дужка або кінець виразу	читання зі стеку пари лексем та їх обробка
стек пустий, кінець виразу	кінець обробки виразу

У цьому алгоритмі необхідно передбачити змінні під *попередню* лексему-об'єкт та лексему дії, а також під *поточну* лексему-об'єкт та лексему дії. В разі запису в стек чергової пари лексем змінні з *попередніх* лексеми-об'єкту та лексеми дії записуються в стек, на їхнє місце переписуються відповідні *поточні* лексеми. Таким чином вивільняються місця під чергові нові *поточні* лексеми з виразу при продовженні обробки виразу.

В разі запису в стек лівої дужки вона записується в стек з "пустим" операндом і продовжується обробка виразу за даним алгоритмом пріоритету. Коли зустрічається права дужка, здійснюється так би мовити зворотний хід у стеку, тобто обробка лексем у стеку з наступним їх виштовхуванням звідти. Ознакою кінця виразу є або спеціальний знак, або ключове слово відповідного оператора згідно його граматики. Кінцем обробки виразу є наявність пустого стеку та досягнення в обробці виразу кінця оператора. Обробка передбачає побудову гілок синтаксичного дерева розбору оператора.

Відсутність паритету дужок одразу ж виявляється через стек як помилка. В разі невідповідності типів операндів стандартний алгоритм має передбачати в разі необхідності перетворення типів (за допомогою спеціальних підпрограм). Тип результату виразу має відповідати опису оператора.

Якщо за основу взяти цей алгоритм, то опис граматики зводиться до опису ключових слів (стандартних та нестандартних) та їхніх семантик як стандартних, так і нестандартних.

Лексеми дії можуть являти собою як спеціальні символи, так і ключові слова, але в обох випадках їхній код складається з коду пріоритету та власно коду лексеми дії, що породжує певний тип обробки.

Ключові слова операторів мови та окремих фраз операторів повинні також визначати пріоритет обробки та відповідну семантику. Так, наприклад семантика ключового слова IF умовного оператора породжує команди порівняння та відповідного умовного переходу.

Всі ключові слова можна звести в одну таблицю та застосувати їх кодування, щоб відрізнити лексеми дії виразу від лексем дії операторів.

Синтаксичний аналіз операторів з таким розподілом лексем доречно розбити на дві частини: обробку виразів та обробку операторів. При цьому обробку виразів необхідно виділити в окрему підпрограму, що породжує дерево розбору саме виразу, а фразові ключові слова операторів мови обробляти окремо. Такий підхід дозволяє достатньо легко вносити зміни синтаксису у мови програмування. Причому окремо в обробку виразів і окремо в обробку операторів.

Синтаксис лексем-об'єктів може відрізнятись в різних мовах і охоплює ідентифікатори, константи (цілого, дійсного, символьного та інших типів) і визначається у кожному випадку окремо. Їхній опис не складає великих труднощів щодо конкретної мови.

Лексеми дії крім стандартних дій повинні передбачати ще й нестандартні згідно правил граматики. Визначення тієї чи іншої дії легко задається в таблиці ключових слів спеціальним кодом, що являє собою посилання і забезпечує перехід на відповідну частину програми синтаксичного аналізатора. Структура елементу таблиці ключових слів має наступний вигляд.

мнемоніка MNEM	тип нетерміналу TYP	код CODE	пріоритет PRIOR

Рис. 2

Такий термінал розпізнається шляхом пошуку у таблиці ключових слів за мнемонікою MNEM.Тип нетерміналу TYP визначає, до якого типу нетерміналу відноситься дана лексема – до виразу чи до оператора. Це значною мірою спрощує перевірку правильності синтаксису оператора і дозволяє швидко виявити синтаксичні помилки. Код CODE разом з пріоритетом PRIOR визначають послідовність обробки та власно перехід на відповідний алгоритм обробки згідно коду. Таким чином, визначаючи нову лексему дії як термінал, для її обробки необхідно додати відповідний програмний фрагмент.

До стандартних дій по обробці операторів таких, як умовний оператор (IF), оператори циклу (FOR, WHILE, REPEAT), оператори введення/виведення (READ, WRITE) [3, 4] можна додавати й нестандартні оператори (наприклад побітової обробки). Для цього необхідно визначити правило обробки та додати підпрограми, які забезпечують синтаксичний розбір таких нестандартних операторів. Звернення до цих операторів забезпечується через посилання, яке подано як код CODE у таблиці ключових слів.

Ключові слова в синтаксисі операторів визначають їхню послідовність в тій чи іншій конструкції оператора мови і їхня обробка виконується у самому кінці, тобто з найнижчим пріоритетом. Як бачимо, пріоритет використовується, в основному, при обробці виразів і досить зручно вписується у загальний алгоритм синтаксичного аналізу. Причому ключові слова операторів у нашому випадку використовуються також як ознаки кінця виразів у мовних конструкціях.

Запропонований алгоритм синтаксичного розбору покладає на лексичний аналізатор розпізнавання не однієї лексеми, а пари лексем: лексеми-об’єкту та лексеми дії і повертати їх синтаксичному аналізатору для подальшого розбору. При цьому на першому кроці аналізу оператора лексичний аналізатор розпізнає стартовий символ N_s – тип оператора мови та подальшій алгоритм аналізу згідно правил граматики.

За даним алгоритмом синтаксичного аналізу було створено кросс-компілятор мови С для мікроконтролера AVR, в якому реалізовані методи машинно-залежної оптимізації коду програм.

Висновки

Запропонований алгоритм синтаксичного розбору дозволяє легко вносити корективи у граматику і, відповідно, у алгоритми обробки виразів та як власно операторів, так і сполучених операторів. За допомогою коду лексеми забезпечується посилання на підпрограму обробки, а за допомогою її пріоритету – порядок обробки у синтаксичному у дереві розбору. Оскільки аналіз здійснюється через розбір пари лексем, то швидкість обробки операторів вища, ніж за класичними алгоритмами через направлений перебір варіантів. При цьому дещо ускладнюється алгоритм лексичного аналізатора, що суттєво не впливає на загальну швидкість синтаксичного аналізу.

Другою перевагою даного методу є простота опису правил синтаксичного розбору операторів та виразів.

Перелік посилань

1. Хопкрофт Джон Э., Мотвани Раджив, Ульман Джефри Д. Введение в теорию автоматов, языков и вычислений. – М.: Издательский дом “Вильямс”, 2002. – 528 с.
2. Ахо Альфред В., Сети Равви, Ульман Джефри Д. Компиляторы: принципы, технологии и инструменты. -М.: Издательский дом “Вильямс”, 2003. – 768 с.
3. Себеста Роберт У. Основные концепции языков программирования. - М.: Издательский дом “Вильямс”,2001.–672с.
4. Бен Ари М. Языки программирования. Практический сравнительный анализ. – М,; Мир, 2000. – 366 с.

УДК 004.074.32

МАРКОВСКИЙ А.П.,
ПОРХУН Е.В.
МНАЦАКАНОВ А.В.

ХЕШ-ПАМЯТЬ С ОГРАНИЧЕННЫМ ВРЕМЕНЕМ ПОИСКА ПО КЛЮЧУ

В статье предложена новая организация хеш-поиска, которая предполагает хранение ключа по одному из двух хеш-адресов. Это позволяет ограничить время поиска двумя обращениями к памяти. Предложена процедура рекурсивной записи ключей в память. Получены аналитические оценки вероятности коллизий. Проанализированы возможности использования предложенной организации хеш-поиска для статических и динамических массивов данных.

In article the new hash-searching organization such that keys may storing in one of dual hash-address has been proposed. It is allowed to limit the searching time by dual memory access. The procedure of recursion recording keys into the hash-memory has been developed. The analytical evaluation of collision probability has been obtained. The possibilities of proposed organization for hash-searching in static and dynamic arrays of data has been analyzed

Введение

Поиск по ключу традиционно является одной из базовых операций обработки информации. В последнее десятилетие многократно выросли объемы информационных массивов, в которых выполняется поиск. Расширение использования информационных технологий для управления процессами, протекающими в реальном времени имеет следствием ужесточение требований к времени выполения поиска по ключу.

Указанные факторы снижают эффективность использования в современных условиях традиционных технологий поиска, для которых время поиска зависит от объема поискового массива. В этих условиях создаются предпосылки расширения применения хеш-поиска — потенциально наиболее быстрой технологии поиска, для которой время доступа не зависит от объема данных, в которых выполняется поиск. До последнего времени широкое применение хеш-памяти сдерживалось присущими ей недостатками: наличием коллизий и избыточностью требуемой памяти. Успехи технологии изготовления интегральных микросхем привели к резкому удешевлению памяти и, тем самым, создали предпосылки к снижению значимости второго из указанных недостатков хеш-адресации.

Наличие коллизий, обусловленных совпадением хеш-адресов различных ключей, требует специальных механизмов для их разрешения. Это усложняет работу с хеш-памятью и обуславливает случайный характер времени поиска. Вместе с тем, существует достаточно широкий круг применений, прежде всего, связанных с обработкой информации в реальном времени, для которых важным является гарантированное время поиска.

Исходя из этого, на современном этапе развития информационных технологии, проблема ограничения времени поиска по ключу при использовании хеш-адресации является актуальной и значимой для практики.

Анализ известных технологий разрешения коллизий

Сущность хеш-адресации состоит в том, что адрес, по которому хранится информация в хеш-памяти, функционально зависит от этой информации. Для ключа X его хеш-адрес A определяется в результате выполнения над ключом хеш-преобразования $h(X)$: $A=h(X)$. В большинстве случаев, число ключей, хранимых в хеш-памяти меньше общего числа ее ячеек, так, что хеш-памяти присуща избыточность использования объема накопителя. Основной характеристикой избыточности является коэффициент загрузки – α, определяемый как отношение числа m хранящихся в памяти ключей к общему числу M ячеек хеш-памяти: $\alpha=m/M$ [1]. Поскольку разрядность ключей всегда больше разрядности хещ-адреса, то при их размещении в хеш-памяти возникает явление коллизии – ситуации, при которой несколько различных ключей имеют одинаковый хеш-адрес.

В современной технологии хеш-адресации целесообразно выделять два типа хеш-поиска: в динамическом и статическом массивах клю-

чей. В первом случае коллизии неизбежны и для их разрешения используются специальные средства. Для статических массивов ключей можно путем подбора хеш-функции добиться отсутствие коллизий. Такая хеш-адресация постоянных массивов ключей называется perfect или совершенной [2].

Эффективность динамической хеш-памяти характеризуется зависимостью скорости поиска от коэффициента α ее загрузки [1]. В свою очередь, скорость хеш-поиска характеризуется средним s_{ave} и максимальным s_{max} числом обращений к памяти, необходимыми для выполнения поиска. Необходимость разрешения коллизий увеличивает число обращений к памяти. Наиболее распространенным на практике способом разрешения коллизий является пробинг – то есть запись ключа X в первую свободную ячейку, адресуемую совокупностью кодов, получаемых в результате ряда хеш-преобразований: $h_1(X)$, $h_2(X)$, $h_3(X)$,… В простейших случаях функции ряда зависят от друг друга. Так, при линейном пробинге, каждый следующий адрес на единицу больше предыдущего: $h_i(X) = (h_1(X)+i) \bmod M$. Хотя такие функции просто вычисляются, при их использовании имеет место недостаток - вторичная группировка записей, которая существенно замедляет процесс поиска. При использовании специальных генераторов независимых хеш-функций [3] достигается теоретически наименьшее среднее число s_{ave} обращений к памяти для разрешения коллизий, которое определяется формулой:

$$s_{ave} = \frac{1}{1-\alpha} \qquad (1)$$

Основным недостатком известных технологий пробинга является то, что они не ограничивают максимальное число s_{max} обращений к памяти в процессе разрешения коллизий. Так, с вероятностью $p_s = \alpha^{s+1}$ при поиске может возникнуть необходимость в более, чем s обращениях к памяти.

Для ряда важных практических применений, связанных с обработкой информации в реальном времени, доминирующее значение имеет ограничение значения s_{max}.

Для совершенной хеш-адресации (то есть без коллизий) постоянных массивов ключей, основным критерием эффективности является зависимость времени подбора хеш-преобразования, не образующего коллизий от коэффициента α загрузки памяти.

Известно ряд технологий выполнения такого подбора [1,2,4], наиболее известными из которых являются методы Cichelli R.J. и Czech Z.J. Все они используют перебор функций хеш-преобразования. Для этих методов, среднее число проб T_α, требующееся для поиска perfect хеш-преобразования определяется следующей формулой [4]:

$$T_\alpha = e^m \cdot (1-\alpha)^{m\cdot(\frac{1-\alpha}{\alpha})+\frac{1}{2}} \qquad (2)$$

Как следует из формулы (2), основным недостатком известных методов получения совершенных хеш-преобразований для постоянных массивов ключей является большое число требуемых проб.

Целью исследований является ограничение числа требуемых обращений к хеш-памяти в процессе разрешения коллизий для динамических массивов ключей и сокращение времени получения хеш-преобразования с ограниченной длиной цепочки коллизий для постоянных массивов ключей.

Концепция хеш-памяти с ограничением коллизий

Для решения задачи ограничения времени хеш-поиска предлагается концепция хранения ключей только по одному из двух возможных хеш-адресов. Это позволяет ограничить длину цепочки возникающих коллизий и обеспечить поиск по ключу не более, чем за два обращения к накопителю.

Предлагаемая концепция предполагает организацию хеш-памяти в виде двух банков (или таблиц), обозначенных так T_1 (левый банк памяти) и T_2 (правый банк памяти), каждый из которых содержит $M/2$ ячеек памяти. Для каждого из ключей определяется один возможный хеш-адрес в банке T_1 и один возможный хеш-адрес в банке T_2. Для адресации ключа X в банках памяти используются две хеш-функции $h_1(X)$ для T_1 и $h_2(X)$ для T_2, такие, что пара значений $h_1(X)$ и $h_2(X)$ однозначно определяют ключ X.

Хранение ключа X в левом банке памяти является приоритетным. Соответственно, запись кода ключа X вначале выполняется в ячейку, адресуемую $h_1(X)$ левого банка T_1. Если она уже занята, то запись выполняется в ячейку с адресом $h_2(X)$ правого банка памяти T_2.

Поскольку коды хеш-адресов $h_1(X)$ и $h_2(X)$ ключа X однозначно определяют код самого ключа X, то в рамках предлагаемой концепции можно существенно сократить объем используемой памяти. Для этого, при размещении ключа X в ячейку $h_1(X)$ левого банка памяти T_1, в ней реально записывается код $h_2(X)$, который вместе с адресом $h_1(X)$ однозначно определяет значение ключа X. Аналогично, при размещении ключа X в правом банке T_2 в его ячейку с адресом $h_2(X)$ записывается код $h_1(X)$. Кроме указанных кодов, которые вместе с хеш-адресом однозначно определяют ключ, в ячейках хранится теговый бит занятости.

Поиск ключа в предлагаемой хеш-памяти не более, чем за два обращения к накопителю, достигается за счет того, что часть ключа, которая вместе с адресом её ячейки однозначно определяет ключ X, может храниться только в одной из двух "доступных для хранения" ячеек: по адресу $h_1(X)$ в банке T_1, либо по адресу $h_2(X)$ в банке T_2.

Алгоритм поиска по ключу X состоит в следующем:

1. Для заданного ключа X вычисляются значения хеш-функций $h_1(X)$ и $h_2(X)$

2. Считывание кода из ячейки, адресуемой $h_1(X)$ левого банка памяти. Если теговый бит считанного кода равен нулю (ячейка свободна), то поиск завершен с отрицательным результатом.

3. Сравнение считанного кода с $h_2(X)$. Если сравниваемые коды совпадают, то поиск завершен с положительным результатом.

4. Считывание кода из ячейки, адресуемой $h_2(X)$ правого банка памяти. Если теговый бит считанного кода равен нулю (ячейка свободна), то поиск завершен с отрицательным результатом.

5. Сравнение считанного кода с $h_1(X)$. Если сравниваемые коды совпадают, то поиск завершен с положительным результатом. В противном случае – поиск завершен с отрицательным результатом.

Выполнение условия хранения каждого из ключей в одной из двух ячеек хеш-памяти обеспечивает длину цепочки коллизий не больше двух, Соответственно, состояние хеш-памяти, для которого выполняется это условие, может быть названо состоянием ограниченных коллизий (СОТ).

Запись ключа X в хеш-памяти всегда осуществляется в одну из двух доступных для его хранения ячеек. Однако, может возникнуть ситуация, при которой обе эти ячейки хеш-памяти заняты другими, ранее записанными ключами.

Схематично такая ситуация показана на рис.1. Здесь через a ,b обозначены ячейки левого банка памяти T_1, а через v,z и w – ячейки правого банка T_2. До записи ключа X_4 в хеш-память записаны: ключ X_1, для которого $h_1(X_1)=a$, $h_2(X_1)=z$, записывается в ячейку a левого банка T_1. Ключ X_2, для которого допустимыми являются ячейки b и w ($h_1(X_2)=b$, $h_2(X_2)=w$) помещается в свободную ячейку b левого банка T_1. Для ключа X_3, ($h_1(X_3)=b$, $h_2(X_3)=v$), допустимыми являются ячейки b и v. Однако ячейка b левого банка T_1 уже занята ключом X_2, поэтому ключ X_3 помещается в ячейку v правого банка T_2. Ситуация после записи трех ключей X_1, X_2 и X_3 показана на рис.1.

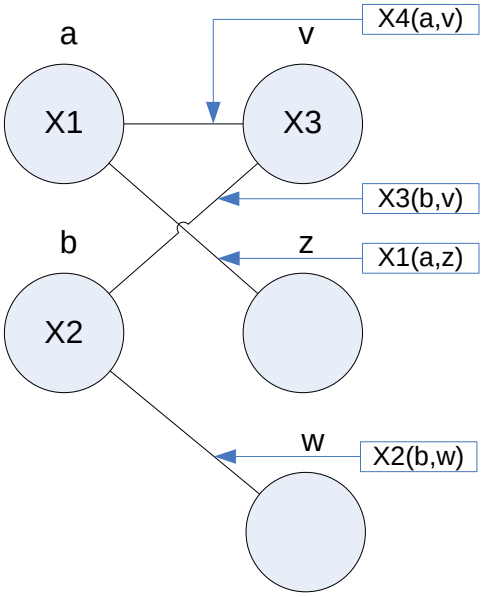


Рис. 1. Запись ключей X_1, X_2, X_3 в память

При записи ключа X_4, для которого $h_1(X_4)=a$ и $h_2(X_4)=v$, возникает ситуация, когда обе допустимых для него ячейки – a в банке T_1 и v в банке T_2 заняты.

Для разрешения такой ситуации предлагается использовать рекурсивное перемещение в памяти ранее записанных ключей с тем, чтобы освободить место для нового ключа.

В рамках рассматриваемого примера, при невозможности записи ключа X_4 производится анализ возможности перемещения ключа X_1, занимающего ячейку a, на которую претендует ключ X_4 в левом банке. Поскольку альтернативная для ключа X_1 ячейка z правого банка T_2

свободна, то ключ X_1 переписывается в нее (фактически из ячейки $a=h_1(X_1)$ считывается код $h_2(X_1)$, адресующий ячейку $z=h_2(X_1)$ правого банка памяти T_2, которую записывается код $a=h_1(X_1)$). После освобождения ячейки a, в нее помещается ключ X_4 (фактически в ячейку a записывается $h_2(X_4)$). Ситуация после записи ключа X_4 схематично показана на рис.2.

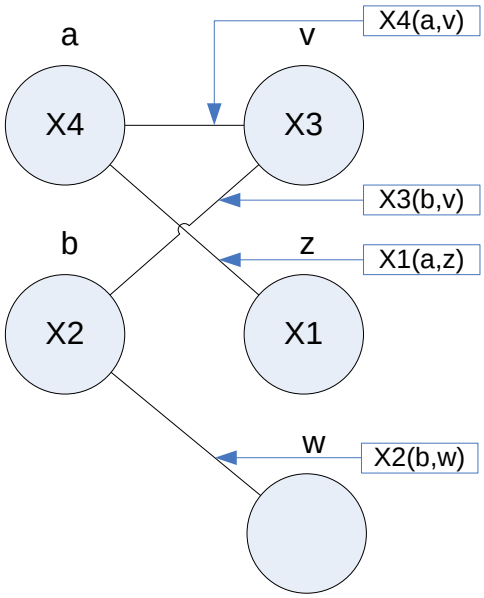


Рис. 2. Запись ключа X_4 в память

Для выполнения рекурсивной записи ключей в хеш-память предлагается формализованный алгоритм. Для его описания используется переменная t ($t \in \{1,2\}$), которая соответствует номеру банка памяти, с которым осуществляется работа в текущий момент времени. Для работы алгоритма необходимо организовать стек, в котором размещаются записываемые и перемещаемые в процессе рекурсии ключи. Каждая из ячеек стека содержит два поля L, R для записи адресов в банках хеш-памяти и однобитовое поле nb для сохранения номера банка памяти.

Алгоритм записи ключа X сводится к выполнению последовательности действий:

1. Вычисляются значения хеш-функций $h_1(X)$ и $h_2(X)$, устанавливается значение t=1.

2. Из ячейки по адресу $h_1(X)$ в левом банке памяти T_1 считывается код d ссылки и теговый бит β. Если теговый бит β равен нулю: β=0, то есть ячейка не занята, то в нее записывается код $h_2(X)$ и теговый бит β равный единице. Операция записи заканчивается. Иначе переход на пп.4.

3. Из ячейки по адресу $h_2(X)$ правого банка памяти T_1 считывается код g ссылки и теговый бит β. Если теговый бит β равен нулю: β=0, то есть ячейка не занята, то в нее записывается код $h_1(X)$ и теговый бит β равный единице. Операция записи заканчивается.

4. Коды $h_1(X)$, $h_2(X)$ и t помещаются в поля L,R, nb стека. Если код d совпадает с одним из кодов, размещенных в поле R стека, то выполнить запись невозможно.

5. Модифицируется код $h_2(X)$: $h_2(X)=d$. Значение переменной t изменяется: t = 3-t.

6. Из ячейки по адресу $h_2(X)$ правого банка памяти T_2 считывается код g ссылки и теговый бит β. Если теговый бит β равен нулю: β=0, то есть ячейка не занята, то в нее записывается код $h_1(X)$ и теговый бит β равный единице. Переход на пп.10.

7. Коды $h_1(X)$, $h_2(X)$ и t помещаются в поля L,R, nb стека. Если код g совпадает с одним из кодов, размещенных в поле L стека, то выполнить запись невозможно.

8. Модифицируется код $h_1(X)$: $h_1(X)=g$. Значение переменной t изменяется: t = 3-t.

9. Из ячейки по адресу $h_1(X)$ считывается код d. Переход на пп.5.

10. Из полей L,R и nb стека выталкиваются коды соответственно $h_1(X)$, $h_2(X)$ и t.

11. Если t=1, код $h_2(t)$ и единичный бит β записывается по адресу $h_1(t)$ в левый банк памяти T_1. Переход на пп. 13.

12. Если t=2, код $h_1(t)$ и единичный бит β записывается по адресу $h_2(t)$ в правый банк памяти T_2.

13. Если стек не пуст, значение переменной t изменяется: t = 3-t и возврат на повторное выполнение пп.8. Иначе конец операции записи.

Количество обращений к накопителю при выполнении описанной рекурсивной процедуры записи ключа является случайным числом, зависящим от коэффициента α загрузки памяти. Средне значение числа T_W обращений к памяти при записи определяется в виде:

$$T_W = 2\cdot(1-\alpha)+\sum_{j=1}^{\infty}\alpha^j\cdot(1-\alpha)\cdot(2\cdot j+1) = \frac{2}{1-\alpha}-1 \qquad (3)$$

Анализ формулы (3) показывает, что для используемых на практике значений α: 0.7≤ α

≤ 0.9 время записи по сравнению с обычным пробингом увеличивается практически вдвое.

Важным для исследования эффективности предлагаемой концепции представляется получение оценки вероятности возникновения коллизии, как ситуации, в которой запись нового ключа с сохранением СОК невозможна.

Оценка вероятности возникновения коллизий

Для каждого из ключей формируется два хеш-адреса. Возможна редкая ситуация, когда для ограниченное подмножество Ω из w ключей число формируемых хеш-адресов будет меньше w. В этой ситуации, размещение ключей подмножества Ω согласно предложенной концепции является принципиально невозможным. То есть, существует возможность возникновения коллизий. Пример такой ситуации схематически показан на рис.3. Кружочками на этом рисунке показаны ячейки левого и правого банков памяти, а дуги – соответствуют ключам подмножества $\Omega=\{X_1,X_2,\ldots,X_6\}$, $h_1(X_1)=a$, $h_2(X_1)=v$, $h_1(X_2)=a$, $h_2(X_1)=z$; $h_1(X_3)=b$, $h_2(X_3)=v$; $h_1(X_4)=b$, $h_2(X_4)=z$; $h_1(X_5)=c$, $h_2(X_5)=v$; $h_1(X_6)=c$, $h_2(X_6)=z$.

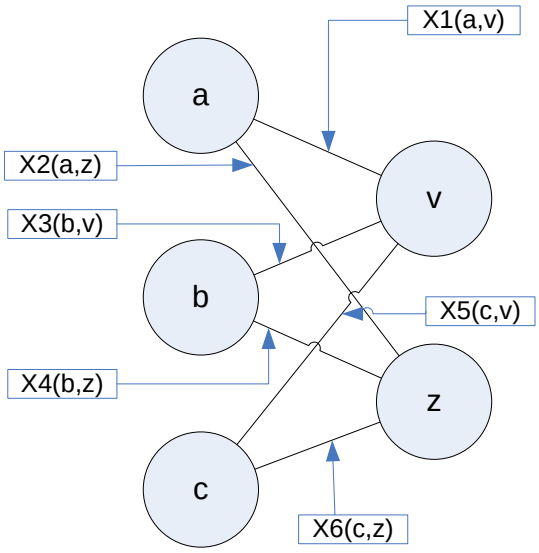


Рис. 3. Пример коллизии при размещении 6-ти ключей в 5-ти ячейках

В ситуации, показанной на рис.3, коллизия получается при размещении 6-ти ключей. При этом, в правом банке занятыми оказываются только две ячейки v и **z**, причем существуют ключи, адресующие каждую из упомянутых ячеек и каждую из 3-х ячеек a,b,c левого банка памяти. Общее число вариантов ключей при этом равно 6-ти. Очевидно, что для подмножества Ω, содержащего меньше 6-ти ключей коллизия невозможна. Действительно, если предположить, что в левом банке используется только одна ячейка, то общее число ключей, адресующих эту ячейку и u ячеек левого банка памяти не превысит u, в то время, как суммарное число ячеек составит u+1.

Вместе с тем, возможны и другие варианты коллизий, при которых множество Ω содержит более 6-ти ключей. Один из таких вариантов, при котором множество Ω включает 9 ключей, которые должны быть размещены в 6-ти ячейках, показан на рис.4.

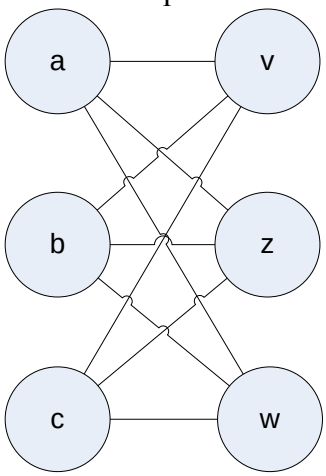


Рис. 4. Пример коллизии при размещении 9-ти ключей в 6-ти ячейках

Важным аспектом исследования эффективности предложенной организации хеш-доступа является получение оценки вероятности возникновения коллизий.

Пусть в хеш-памяти размещается m ключей, а общее количество ячеек в обоих банках равно M. Аналитическая оценка вероятности p_6 возникновения ситуации, показанной на рис.3, при которой для множества из 6-ти ключей формируется только 5 адресов, может быть выполнена следующим образом.

Пусть фиксируются две ячейки правого банка хеш-памяти, обозначенные как v и z. Если $\alpha>0.5$, то число адресных ссылок m ключей на правый банк памяти превышает количество $M/2$ его ячеек, то есть $m>M/2$. Это означает, что с вероятностью близкой к единице имеется ключ X, с адресной ссылкой v для правого банка памяти: $h_2(X)=v$. Адресная ссылка этого ключа X в левом банке адресует некоторую ячейку a, то есть $h_1(X)=a$. Аналогично, с вероятностью, близкой к единице существует ключ Y, такой, что $h_2(Y)=z$ и $h_1(Y)=b$.

Если выбрать фиксированную ячейку c левого банка памяти, то вероятность того, что

существует ключ G, для которого $h_1(G)=c$ и $h_2(G)=v$ составляет M^{-1}. Вероятность того, что фиксированный ключ имеет ссылки на два фиксированных в левом и правом банках хеш-памяти адреса составляет M^{-2}. Поэтому, если фиксировать три ключа Q_1,Q_2 и Q_3, то вероятность того, что все они имеют заранее определенные адресные ссылки (в частности, для первого Q_1 из них: $h_1(Q_1)=a$ и $h_2(Q_1)=z$; для второго Q_2 : $h_1(Q_2)=b$ и $h_2(Q_2)=v$; а для третьего Q_3 из них: $h_1(Q_3)=c$, $h_2(Q_3)=z$) составляет M^{-6}. Таким образом, вероятность того, что возникнет ситуация, показанная на рис.3 для трех фиксированных ключей Q_1,Q_2 и Q_3 и трех фиксированных ячеек: c, v, z составляет M^{-7}. Исходя из этого, ситуация, показанная на рис.3 не возникает, если она не имеет места для всех возможных вариантов выбора упомянутых трех ключей и всех возможных вариантов выбора ячеек c и v, z. Количество вариантов выбора трех ключей Q_1,Q_2 и Q_3 равно:

$$\frac{m!}{3!\cdot(m-3)!}=\frac{m\cdot(m-1)\cdot(m-2)}{6}$$

Для больших m, приближенно равно $m^3/6$.

Число вариантов выбора ячейки c равно M, а количество вариантов выбора пары ячеек v и z для больших M приближенно равно $M^2/2$. С учетом этого, вероятность q_6 того, что ситуация, показанная на рис.3 не возникает определяется выражением:

$$q_6=(1-M^{-7})^{\frac{m^3\cdot M^3}{12}}$$

Вероятность p_6 того, что при хеш-адресации множества из m ключей для 6-ти из них возникнет коллизийная ситуация, показанная на рис.3 определяется в виде:

$$p_6=1-q_6=1-(1-M^{-7})^{\frac{m^3\cdot M^3}{12}} \qquad (4)$$

Для имеющих место на практике больших значений m и M формула (4) может быть упрощена путем раскрытия скобок и учета только первых двух членов биномиального ряда, поскольку значения всех остальных составляют не более 5% второго члена ряда:

$$p_6\approx 1-(1-\frac{m^3\cdot M^3}{12}\cdot M^{-7})=\frac{m^3}{M^4\cdot 12}=\frac{\alpha^3}{12\cdot M} \qquad (5)$$

Проведенные экспериментальные исследования, в целом, подтвердили справедливость теоретической оценки (5) для вероятности p_6 коллизий для подмножества из 6-ти ключей.

Так, проведя аналогичные выкладки для ситуации показанной на рис.4, когда коллизия образуется при адресации 9-ти ключей, можно показать, что вероятность p_9 оценивается следующим выражением:

$$p_9\approx\frac{\alpha^3}{36\cdot M^3} \qquad (6)$$

Сравнение выражений (5) и (6) показывает, что для имеющих место на практике больших значений M: $p_6>>p_9$. Отсюда можно сделать вывод, что вероятность коллизии, возникающей при размещении в хеш-памяти подмножества ключей, быстро падает с увеличением числа ключей этого подмножества. Это означает, что с точностью, достаточных для оценочных расчетов, вполне можно полагать, что коллизии могут быть образованы только подмножеством из 6-ти ключей, вероятность которой оценивается полученной выше формулой (5).

Оценка эффективности применения для динамических и постоянных ключей

Предложенная концепция хеш-адресации ключей с ограничением адресов их размещения может быть использована для повышения эффективности как динамической, так и статической хеш-памяти.

Важнейшей характеристикой эффективности является время поиска, которое определяется средним s_{ave} и максимальным s_{max} числом обращений к накопителю.

Как уже отмечалось, использование предложенного подхода позволяет принципиально ограничить максимальное время поиска двумя обращениями к памяти. То есть, при использовании предложенного подхода $s'_{max}=2$. Как указывалось выше, при использовании пробинга, вероятность p_s того, что потребуется не менее s обращений к памяти определяется как $p_s=\alpha^{s+1}$. Например, при типичном на практике значении коэффициента загрузки $\alpha=0.75$ при использовании пробинга с вероятностью 0.01 потребуется для поиска 28 обращений к памяти. То есть, для такой ситуации, применение предложенного подхода позволяет уменьшить максимальное время поиска в 14 раз.

Среднее число s'_{ave} обращений к накопителю зависит от коэффициентов загружен-

ности каждого из банков хеш-памяти и определяется формулой:

$$s'_{ave} = \frac{\alpha_L + 2 \cdot \alpha_R}{\alpha_L + \alpha_R} \qquad (7)$$

где α_L – коэффициент загрузки левого банка памяти, а α_R – коэффициент загрузки правого банка.

Экспериментально установлено, что при малых значения α (α< 0.7) более загруженным оказывается левый банк хеш-памяти, соответственно, среднее число обращений к накопителю при поиске меньше 1.5. Так, при α=0.6 коэффициент α_L загрузки левого банка памяти составляет 0.7, в то время как для правого банка $\alpha_R = 0.5$. Соответственно, при этом, среднее число s'_{ave} обращений к накопителю при поиске, в соответствии с формулой (7) составляет 1.42.

При α > 0.7 левый и правый банки памяти заполняются практически одинаково и среднее число обращений к памяти при поиске практически равно 1.5.

Таким образом, сопоставление значений среднего числа обращений к памяти при поиске для предлагаемой концепции s'_{ave} и традиционного пробинга s_{ave} позволяет сделать следующий вывод. При α > 0.4 (то есть практически всегда) справедливо $s'_{ave}<s_{ave}$, это означает, что применение предложенной концепции позволяет уменьшить среднее число обращений к памяти при поиске, то есть ускорить поиск. Например, при наиболее типичном значении коэффициента загрузки α=0.75 использование предложенного подхода позволяет в 2.7 раз уменьшить время поиска по сравнению с пробингом.

Следует указать, что повышение скорости поиска достигается за счет замедления процесса записи ключей. Поэтому, реальная эффективность применения предложенного подхода может быть достигнута только для класса задач, в которых интенсивность операций поиска существенно выше интенсивности операций модификации поискового массива.

При реализации предложенной концепции для хеш-адресации динамических массивов ключей необходимо предусмотреть механизм разрешение коллизий – то есть ситуации, при которой существует подмножество Ω ключей, хеш-адреса которых образуют замкнутый класс и их число меньше, чем количество ключей, составляющих множество Ω.

В качестве такого механизма предлагается использовать дополнительную память переполнения малого объема. В такой памяти предполагается размещать ключи, которые не могут быть размещены в банках хеш-памяти по своим хеш-адресам. Для того, чтобы реализовать переход к дополнительной памяти переполнения необходимо в ячейках основной памяти, помимо тегового бита занятости, выделить второй теговый бит перехода к дополнительной памяти.

Применительно к хеш-поиску в постоянных массивах ключей использование предложенного подхода позволяет значительно сократить время получения хеш-преобразования, допускающего размещение ключа в одной из двух альтернативных ячейках хеш-памяти.

Для получения такого хеш-преобразований случайно выбираются две хеш-функции $h_1(X)$ и $h_2(X)$. Массив ключей в соответствии с описанной выше рекурсивной процедурой записывается в банки хеш-памяти. Если в процессе записи возникнет коллизия – то есть невозможность размещения ключей в соответствии с предложенной концепцией, то хеш-функции $h_1(X)$ и $h_2(X)$ меняются, после чего производится повторная попытка размещения заданного массива постоянных ключей. Среднее число T_p выбора пары хеш-функций $h_1(X)$ и $h_2(X)$ исходя из (5) составит:

$$T_p = \sum_{j=1}^{\infty} j \cdot \frac{\alpha^{3\cdot(j-1)}}{12^{j-1} \cdot M^{j-1}} \cdot \left(1 - \frac{\alpha^3}{12 \cdot M}\right) = \frac{1}{1 - \frac{\alpha^3}{12 \cdot M}} \qquad (8)$$

Сопоставление формул (2) и (8) показывает, что применение предложенного подхода к хеш-адресации постоянных массивов ключей позволяет на несколько порядков ускорить подбор подходящего хеш-преобразований. Однако, необходимо иметь ввиду, что указанный значительный выигрыш во времени формирования хеш-преобразования достигнут за счет того, что при использовании предложенного подхода, в отличие от совершенной хеш-адресации, допускается два (а не одно) обращений к накопителю в процессе поиска, то есть за счет снижения скорости поиска в полтора раза.

Принимая во внимание, что число обращений к памяти, требующее для записи одного ключа определяется формулой (3), общее число N_w обращений к памяти, требующееся для размещения постоянного массива ключей в хеш-памяти определяется следующей формулой:

$$N_w = \frac{m}{1-\frac{\alpha^3}{12\cdot M}}\cdot\left(\frac{2}{1-\alpha}-1\right) \qquad (9)$$

Анализ формулы (9) показывает, что использование предложенного подхода для хеш-адресации постоянных массивов ключей эффективнее существующих методов совершенной хеш-адресации при больших значениях коэффициента α загрузки памяти и при большом числе размещаемых в хеш-памяти ключей. Практическая значимость предлагаемого подхода для совершенной хеш-адресации состоит в том, что при его использовании может быть получено решение задачи, недостижимое при применении известных методов, а именно: нахождение хеш-преобразований, которые гарантируют размещение ключей большого постоянного массива ключ по одному из двух альтернативных адресов, с коэффициентом использования памяти, близким к единице.

Выводы

В результате проведенных исследований предложена концепция организации хеш-памяти с ограниченным временем поиска по ключу. Разработаны технологические аспекты этой концепции, проведен теоретический и экспериментальный анализ ее эффективности.

Основной фактор эффективности предложенной концепции состоит в ограничении числа обращений в памяти при поиске за счет усложнения процедуры записи ключей. Доказано, что это позволяет в 2-3 раза уменьшить среднее время поиска по ключу и на порядок – максимальное время в сравнении с использованием пробинга. Это достигается за счет соответствующего увеличения времени записи ключа. Поэтому использование предложенного подхода эффективно при условии когда интенсивность операций поиска по ключу превышает интенсивность операций модификации поискового массива.

Исследованиями показано, что предложенный подход может быть использован не только для повышения скорости поиска в динамических массивах ключей, но и для многократного ускорения в сравнении с известными методами, подбора эффективных хеш-преобразований для постоянных массивов ключей.

Список литературы

1. Кохонен Т. Ассоциативная память.-М.:Мир,1980.- 198 С.
2. Марковский А.П., Гаваагийн Улзисайхан, Бардис Николас. Об одном подходе к повышению эффективности и уровня защищенности систем хранения информации на основе хеш-памяти//Вісник НТУУ “КПІ”. Інформатика, управління та обчислювальна техніка.- 1998,- № 31,- С.14-23.
3. Салех Ибрагим Аль-Омар. Использование генераторов булевых функций для повышения эффективности хеш-памяти. // Вісник Національного технічного університету України ”КПІ”. Інформатика, управління та обчислювальна техніка.- 2003.- № 40.- С.131-140.
4. Czech Z.J., Havas G., Majevski B.S.. An Optimal algorithm for generating minimal perfect hash functions.//Information processing letters. – 1997. – Vol.43.- № 5. - P.257-264.

УДК 681.3.06

ПУСТОВАРОВ В.І.

ПОБУДОВА БАЗ ЗНАНЬ ДЛЯ СУМІСНОГО СТВОРЕННЯ ПРОГРАМ І ОБЛАДНАННЯ ТА ЇХ ФОРМАЛЬНОЇ ВЕРИФІКАЦІЇ

Розглянуто підхід до комплексного розв'язання задач аналітичної підготовки аналізу і синтезу програмного забезпечення. Ключові елементи такого підходу складають формати подання заголовків специфікацій і змісту ресурсів розв'язання задач (РРЗ) і база знань, що систематизує пошук аналогів, вибір близьких компонентів, використання правил перетворень та механізми декомпозиції РРЗ. Показано, що застосування таких баз знань дозволяє максимально використати попередні розробки і гранично наблизити значення цільових семантичних змінних до вимог обмежень і критеріїв щодо характеристик обробки.

The approach to the complex task decision for analytical support of software analysis and synthesis is considered. Key elements of such approach use representation formats for specification headers contents of task decision resources (TDR) and knowledge base for analogues search, a choice of best components, use of transformation rules and decomposition mechanisms for TDR. It is shown, that application of such knowledge bases allows using the most of the previous software, and making values of target semantic variables extremely close to restriction requirements and criteria of processing speed.

Вступ. Традиційна побудова систем автоматизованого проектування (САП) програмних і апаратних ресурсів розв'язання задач (РРЗ) спирається на табличну організацію [1] вбудованих структур даних. Більшість цих таблиць, що настроюються при реалізації мови та трансляції з вхідних мов зберігають актуальність і в сучасних засобах оптимізації, синтезу та верифікації [3, 6, 7]. Перспективним підходом до комп'ютерної автоматизації сумісного проектування програм і апаратури прискорення обробки є включення баз знань (БЗ) аналітичного супроводження рішень [2] з безпосереднім використанням робочого середовища табличних і кубічних сховищ реляційних СУБД до складу САП РРЗ.

Фактично всі правила і механізми обробки можна задати комплексами відношень, накопиченими в таблицях і кубах. Головне призначення таблиць і правил БЗ в САП РРЗ – забезпечити підготовку проектних рішень на базі аналізу наявних РРЗ і вимог до розробок [1, 3]. Інформаційно-аналітичну базу (ІАБ) САП РРЗ складають комплекси табличних і кубічних сховищ для: **граматичного аналізу** об'яв, описів і реалізацій проблемних галузей досліджень (ПГД) і РРЗ; **визначення базових характеристик** якості РРЗ; **збереження** попередньо розроблених **РРЗ** разом з характеристиками їх виконання; **збереження даних про верифікацію РРЗ, дії з верифікації та осіб**, відповідальних за її виконання; **розпізнавання** проблемних областей і РРЗ за їх специфікаціями, реалізаціями та іменуваннями; **вибір** РРЗ, що найкраще відповідають заданим критеріям.

Головна проблема використання реляційних СУБД полягає у складності обробки правил, декларацій виразів і операторів комп'ютерних мов. Одним з шляхів підвищення рівня автоматизації проектування в САП РРЗ є використання полів даних спеціального аналітичного типу [4]. Примірники таких даних визначають елементи програмних модулів і їх специфікацій з накопиченням бажаних і фактичних характеристик модулів програм і моделей апаратури. В різних САП РРЗ підготовка і реалізація проектних рішень стосуються задач аналізу і синтезу об'єктів ПГД, тощо.

Визначення сховищ реляційної БД для задач вхідного синтаксичного і лексичного аналізу. Лексичний аналіз (ЛА) та синтаксичний аналіз (СА) використовують ІАБ з сукупністю таблиць і списків лексем, шаблонів, правил та кубів підсумкових даних. Основні сховища даних для граматичної обробки вхідних мов включають ***інформаційні таблиці*** $\boldsymbol{r}_{sk}$, $\boldsymbol{r}_{tk}$, $\boldsymbol{r}_{rk}$, $\boldsymbol{r}_{kk}$, $\boldsymbol{r}_{dk}$, $\boldsymbol{r}_{lk}$, $\boldsymbol{r}_{mk}$, ***координатні таблиці*** $\boldsymbol{a}_{dk}$, $\boldsymbol{a}_{sk}$, $\boldsymbol{a}_{tk}$, $\boldsymbol{a}_{ek}$, $\boldsymbol{a}_{rk}$, $\boldsymbol{a}_{lk}$, $\boldsymbol{a}_{ck}$, $\boldsymbol{a}_{fk}(x_{k1},\ldots,x_{ki_{max}}, \xi_{k1},\ldots, \xi_{kj_{max}}, a_{k1},\ldots, a_{kn_{max}})$) та ***куби*** $\prod_k \boldsymbol{a}_{fk}(\sigma_{k1}(\xi_{k1}), \ldots, \sigma_{kj}(\xi_{kj_{max}}))|x_{k1}, \ldots, x_{ki_{max}}, \xi_{k1},\ldots, \xi_{kj_{max}}$, де k – номери координат; $x_{k1}, \ldots, x_{ki_{max}}$ – поля ідентифікації; $\xi_{k1},\ldots, \xi_{kj_{max}}$ – поля семантично істотних даних; $a_{k1},\ldots,a_{kn_{max}}$ – поля додаткових даних обліку. Ключова частина структур основних таблиць та куба підсумків граматичного аналізу показана на Рис. 1. Фактично вона охоплює всі використані таблиці граматичного аналізу і

при наявності таблиць $\boldsymbol{r}_{tk}$, $\boldsymbol{r}_{rk}$, $\boldsymbol{r}_{kk}$, $\boldsymbol{r}_{lk}$, $\boldsymbol{r}_{mk}$, $\boldsymbol{a}_{dk}$, $\boldsymbol{a}_{tk}$, $\boldsymbol{a}_{ek}$, $\boldsymbol{a}_{rk}$, $\boldsymbol{a}_{ck}$, $\boldsymbol{a}_{vk}$ для кожної мови, реалізованої в САП РРЗ, забезпечує обробку кодів, складених на різних мовах програмування і моделювання апаратури, перетворених в єдину форму внутрішнього подання.

Рис. 1. Узагальнена інформаційно-логічна структура БЗ перетворень парсера САП

Визначення структури сховищ БЗ внутрішніх даних семантичної обробки задач. Семантична обробка використовує ІАБ, показану на Рис. 2, з сукупністю таблиць $\boldsymbol{r}_{cm}$, $\boldsymbol{r}_{sm}$, $\boldsymbol{r}_{tm}$, $\boldsymbol{r}_{lpm}$, $\boldsymbol{r}_{dm}$, $\boldsymbol{r}_{lrm}$, $\boldsymbol{r}_{lm}$, $\boldsymbol{r}_{fm}$, що включають специфікації даних, цілей, шаблонів, правил і шляхів розв'язання користувача і стандарти вбудованої семантичної обробки мов САП та

РРЗ. Кубічне сховище використовується для накопичення оцінок ефективності і статистик виконання окремих РРЗ, спрямованих виключно на розв'язання внутрішніх задач ПГД. Координатні таблиці куба ПГД $\boldsymbol{a}_{tm}$, $\boldsymbol{a}_{pm}$, $\boldsymbol{a}_{sm}$, $\boldsymbol{a}_{lm}$, $\boldsymbol{a}_{cm}$, $\boldsymbol{a}_{vm}$, $\boldsymbol{a}_{dm}$, $\boldsymbol{a}_{dvm}$ визначають обробку задач підтримки прийняття рішень, якщо такі розв'язуються в рамках цієї ж ПГД.

Статистики і підсумки, накопичені в кубі ПГД, використовуються на етапі проектування для пошуку найкращої реалізації всіх РРЗ.

Рис. 2. Узагальнена інформаційно-логічна структура БЗ семантичних перетворень САП РРЗ

Визначення структури сховищ БЗ даних процесу проектування РРЗ. За аналогією з етапами аналізу моделей програм побудуємо схему ІАБ для процесу проектування. Проектування РРЗ використовує ІАБ, показану на Рис. 3, з сукупністю таблиць $\boldsymbol{r}_{en}$, $\boldsymbol{r}_{pn}$, $\boldsymbol{r}_{sn}$, $\boldsymbol{r}_{tn}$, $\boldsymbol{r}_{rn}$,

$\boldsymbol{r}_{an}$, $\boldsymbol{r}_{dn}$, $\boldsymbol{r}_{in}$, що зберігають дані про проекти РРЗ, плани і стан їх реалізації, шаблонів, правил та РРЗ спроектованих і в процесі проектування. Кубічне сховище використовується для накопичення оцінок стану створення проектів ПГД і окремих РРЗ. Координатні таблиці куба проектів $\boldsymbol{a}_{pn}$, $\boldsymbol{a}_{zn}$, $\boldsymbol{a}_{sn}$, $\boldsymbol{a}_{nn}$, $\boldsymbol{a}_{cn}$, $\boldsymbol{a}_{dn}$, $\boldsymbol{a}_{mn}$, $\boldsymbol{a}_{fn}$ задають напрямки підтримки прийняття рішень при реалізації проектів в одній ПГД.

Рис. 3. Узагальнена інформаційно-логічна структура БЗ процесів проектування САП РРЗ

Оскільки весь процес створення РРЗ пов'язаний зі способами подання специфікацій, програм, моделей і процесів доведення [7] важливо використати узагальнений механізм формуван-

ня аналітичних моделей функціональності обчислень і всіх можливих оцінок їх складності. Такий механізм створюється на базі узагальненого абстрактного типу аналітичного даних [4], в якому використовуються спеціальні канонічні форми зберігання [4, 6] для полегшення зіставлень варіантів рішень програм і доведень.

Вимоги до канонічних форм подання кодів програм та протоколів верифікації [5] спираються на відношення лінійного порядку $\preccurlyeq$ [6] на базі кодів типів різних термінальних і нетермінальних вузлів УСГ. Термінальні вузли включають поля образів ідентифікації змінних або внутрішню форму подання константних даних:

- ***впорядковуюча нумерація кодів типів*** термінальних та нетермінальних вузлів УСГ;
- ***впорядкування або нумерація змінних*** через запам'ятовування імен або ідентифікаторів, заданих за будь-якою системою довільних евристичних правил [6];
- ***впорядкування константних даних*** за їх типами і значеннями у внутрішній формі;
- ***можливість встановлення відношення канонічного лінійного порядку*** для комутативних операцій серед виразів та операторів;
- ***можливість побудови формул різних оцінок складності*** для визначення точності, надійності та ефективності варіантів РРЗ;
- ***можливість підрахунку значень різних оцінок складності*** для конкретизованих варіантів РРЗ;
- ***можливість порівняння аналітичних і числових*** оцінок складності для вибору кращого варіанта реалізації РРЗ**.**

Додатково розглянемо можливості створення РРЗ для конструктивних задач. Звичайно вони в технічних ПГД вони обмежуються розрахунковими задачами аналізу і синтезу з наступною перевіркою відповідності розрахунків гранично припустимим значенням. В задачах аналізу обробляються дані про навантаження конкретизованого технічного об'єкта на рівні його моделі. В задачах синтезу визначаються параметри елементів технічних об'єктів на основі граничних значень їх навантажень. При розв'язанні базових класів конструктивних задач використовують наступні процедури розв'язання часткових задач та задач проблемного рівня САП РРЗ:

- ***часткове підбиття підсумків в задачах аналізу поточного стану та рішень*** з використанням технологій накопичення та вибірки підсумків в гіперкубах [2];
- ***оцінка навантажень на елементи об'єктів***, що виконується шляхом прямих обчислень або розв'язання рівнянь операціями перетворень даних аналітичного типу;
- ***підготовка рішень та формування значень параметрів елементів об'єктів*** розрахунковими формулами, одержаними через зворотні перетворення аналітичних даних;
- ***оцінки обмежень навантажень і параметрів об'єктів*** для оцінки працездатності, надійності і ефективності проектних рішень.

Таким чином, показані 4 класи задач створюють основу реалізації комплексу РРЗ для будь-якої ПГД, в тому числі і для побудови САП. Для задач оцінок варіантів проектів визначають гіперкуби $\Pi_k \boldsymbol{r}_{fk}$, таблиці $\boldsymbol{r}_{ck}$ та прості або підсумкові змінні оцінок ξ_j та $\sigma_{kj}(\xi_{kj\max})$, які входять до критеріїв рішень ПГД.

Умови коректного розв'язання задач САП зі створення надійних і ефективних РРЗ. Важливо, щоб БЗ дозволила знайти найкращі РРЗ, що розв'язують близькі задачі, і автоматизованим шляхом побудувати необхідний залишок РРЗ, що проектується. Контроль точності, надійності, ефективності та своєчасності розв'язання забезпечується специфікаціями РРЗ вторинного рівня.

Доказові перетворення виразів специфікацій зв'язків виконуються вбудованими операціями аналітичного типу даних з контролем вторинних цілей і критеріїв, заданих специфікаціями ПГД і РРЗ [4]. Для перетворень важливо визначити шлях формування прирощення тексту коду P_{pn} при побудові модифікованого РРЗ і оцінки прирощень потрібних елементів критеріїв. Спочатку треба визначити різницю між специфікацією S_{pn} і P_{pn}, а потім трансформувати її в код додатку до базового коду РРЗ для формування результату $\Delta P_{pn} = f(y, S_{pn} - P_{pn})$. При відсутності базового коду перетворення синтезу формуються як послідовність прирощень, що формуються з первинної специфікації з врахуванням заданих критеріїв.

Визначення загальних правил проектування РРЗ та настроювання роботи САП. З одного боку в шлях просування при синтезі кодів визначається або вручну, або автоматизованою підказкою прирощення. Якщо при автоматизованому синтезі необхідно зробити, що за попереднім доведенням відповідає специфікації, то при ручному синтезі необхідно формально довести коректність перетворення.

З іншого боку треба автоматизувати генерацію різних варіантів повний перебір яких є NP-

складною задачею [6]. Однак переший ліпший варіант програми можна побудувати через перетворення, що послідовно наближують створення повного коду РРЗ, а потім удосконалювати його результатами адекватних перетворень, які покращують значення критеріїв.

Кількість можливих шляхів n_c перетворення визначається кількістю n_k відомих методів потенційно точного або наближеного розв'язання задач, занесених до БЗ САП РРЗ. Ці кількісні оцінки визначаються кількістю і розмірністю вхідних $n_i(N_{ik})$, вихідних $n_o(N_{ok})$ та робочих змінних $n_w(N_{wk})$, сформованих аналітичними методами підготовки розв'язання задач

$$n_k = n_k(n_i(N_{ik}), n_w(N_{wk}), n_o(N_{ok})). \quad (1)$$

Як правило, кількість ефективних варіантів розв'язання найбільш досліджених задач не перевищує десятків, а для одержання технічно ефективного результату можна обмежитися першою десяткою методів.

Вибір порядку створення задач та підзадач в САП РРЗ. Звичайно після первинного визначення типів даних (координат) будь-якого супероб'єкта ПГД доцільно створити набір типових комплексів задач для розв'язання в цій галузі. Розмірність комплексів задач визначається кількістю та розмірністю характеристик конкретизацій об'єктів, включених до моделі конкретної комплексної ситуації в ПГД. Кількість задач n_t обмежується кількістю об'єктів та агрегатів даних, які можуть бути вхідними та вихідними в підпрограмах. Кількість задач буде істотно менше кількості попарних комбінацій об'єктів, як вхідних та вихідних даних. Таким чином, n_t залежить від кількості та структурованості даних в об'єкти для їх використання в задачах як вхідних і вихідних даних.

Таким чином, для доказової розробки, верифікації і оперативного настроювання РРЗ вдосконалені САП повинні виконувати наступну послідовність дій, пов'язаних з вмістом вбудованої БЗ супроводження проектів та аналітичних перетворень.

1. Визначити базову структуру типів даних (координат) ПГД та комплексів об'єктів для розв'язання задач аналізу та синтезу.
2. Визначити цільові змінні та вторинні критерії в специфікаціях окремих задач ПГД.
3. Організувати адекватні перетворення за правилами перетворень аналітичних даних і методами розв'язання задач, занесеними до БЗ семантичних перетворень, показаної на рис.2.
4. Організувати пошук аналогів та супроводження реалізації проектів з використанням БЗ процесів проектування, показаної на рис. 3.

Процес доказової підготовки та настроювання РРЗ повинен супроводжуватись аналітичним контролем функціональності і продуктивності модулів, критичних за часом виконання [4].

Висновки

Таким чином, для підвищення рівня автоматизації САП РРЗ доцільно істотно розширити вбудовану БЗ до блоків управління обробкою, а також проміжні підсумки можуть використовуватися для ефективної комбінації фрагментів кодів. Ключовим елементом організації таких БЗ є використання абстрактного типу аналітичних даних. Кількість задач аналізу і синтезу в конкретному комплексі ПГД істотно обмежена кількістю істотних груп об'єктів та їх характеристик. Накопичення даних про ефективні реалізації РРЗ в БЗ є спеціальним варіантом задачі вибору ефективних рішень за допомогою БЗ, реалізованих в рамках реляційних СУБД [2].

Список посилань

1. Ахо А., Сети Р., Ульман Дж. Компиляторы: принципы, технологии, инструменты: Пер. с англ. – М.: Издательский дом Вильямс, 2001. – 768 с.
2. Бэлсон Д., Гокмен М. Ингрем Дж. Внутренний мир Oracle8. Проектирование и настройка: – К.: Изд-во "ДиаСофт". – 2000. – 800 с.
3. Лисков Б., Гатэг Дж. Использование абстракций и спецификаций при разработке программ. Пер. с англ. М.: Мир, 1989. 424 с.
4. Пустоваров В.І. Супроводження контролю відповідності програм і моделей формальним специфікаціям задач // Вісник НТУУ "КПІ". Інформатика, управління та обчислювальна техніка – К.: «Век+». 2007, 47, с. 269-279.
5. Hehner E.C.R. Practical theory of programming. Springer-Verlag, New York, 1993 – 243 p.
6. Metzger R.C., Zhaofang W. Automatic algorithm recognition and replacement: a new approach to program optimization / The MIT Press, Cambridge, 2000. 219 p.
7. Woodcock J., Davies J. Using Z. Specification, Refinement, and Proof. C.A.R. Hoare Series editor, 1995 – 390 p.

УДК 004.94

ТОМАШЕВСЬКИЙ В.М., д.т.н., професор
ДІДЕНКО Д.Г., аспірант, асистент кафедри

ПОРІВНЯННЯ РЕЗУЛЬТАТІВ РОБОТИ СИСТЕМ ІМІТАЦІЙНОГО МОДЕЛЮВАННЯ OPENGPSS ТА GPSS/PC

В статье рассмотрены различные аспекты проведения экспериментов в дискретно-событийных системах OpenGPSS и GPSS/PC, сравниваются их качественные и количественные результаты работы. Предложен подход разделения эксперимента на независимые части для параллельного выполнения на узлах кластера с последующей сборкой результатов, который приводит к повышению производительности системы или уменьшению времени моделирования и при этом не влияет на правильность конечных результатов.

COMPARISON OF JOB OF SIMULATION SYSTEMS OPENGPSS AND GPSS/PC

The report deals with questions of computing experiment in distributed discrete-event simulation systems OpenGPSS and GPSS/PC, their high-quality and quantitative job performances are compared. The problems of experiment distribution by independent part, deployment this part on cluster node, parallel execution and result assembles and here does not influence on the rightness of end-point also were laboured in the report.

Вступ

Імітаційне моделювання є одним з важливих методів аналізу різних складних систем. Разом зі збільшенням продуктивності та складності сучасних комп'ютерних систем постають нові вимоги до систем імітаційного моделювання – прискорення отримання результатів моделювання. Для цього можна використовувати розподілені системи імітаційного моделювання, які розглянуті в роботах [1, 2].

Існує багато реалізацій розподілених систем і систем тиражування експериментів, наприклад, SPEEDES [3], PARASOL [4] або Triad.Net [5]. Нажаль ці системи не охоплюють розробників, які працюють з поширеною мовою імітаційного моделювання GPSS [6]. Тому доцільно підвищення швидкості моделювання вже побудованих GPSS моделей. Останнім часом була розроблена нова розподілена дискретно-подійна система імітаційного моделювання OpenGPSS [7], яка доступна в режимі онлайн і використовує мову імітаційного моделювання GPSS. Але постає проблема довіри до результатів роботи нової системи моделювання.

Постановка задачі

Цілі дослідження: необхідно якісно та кількісно порівняти результати роботи систем імітаційного моделювання GPSS/PC та OpenGPSS. Еталоном при порівнянні може виступати широко відома прикладна програма GPSS/PC версії 2.0 компанії Minuteman Software (http://www.minitemansoftware.com/), яку багато років використовують розробники моделей і викладачі вузів.

Проведення обчислювального експерименту в послідовних системах моделювання

Розглянемо типовий обчислювальний експеримент мовою GPSS, який моделює роботу одноканальної системи масового обслуговування (СМО) з чергою (табл. 1).

Табл. 1. Текст тестової GPSS-програми

№	Команди GPSS
100	GENERATE 10,5
110	QUEUE QUE1
120	SEIZE PRIB1
130	DEPART QUE1
140	ADVANCE 15,5
150	RELEASE PRIB1
160	TERMINATE
170	GENERATE 5000
400	TERMINATE 1
500	START 1

Проведемо комп'ютерні прогони тестової GPSS-програми в двох системах моделювання. Для накопичення статистично-значимих результатів в системі GPSS/PC, кожний експеримент проводиться 3 рази, і в звіті відобража-

ються лише середні арифметичні показники в таблицях результатів.

Система OpenGPSS може працювати як в звичайному послідовному режимі (використовується лише один вузол моделювання), так і в розподіленому режимі, як описано в роботі [8] (використовується два або більше вузла моделювання). Спочатку будемо використовувати систему OpenGPSS всього з одним вузлом моделювання. У зведеній табл. 2 наведена порівняльна інформація про роботу обох систем.

Табл. 2. Порівняння результатів послідовного моделювання

Показник	**Значення**	**GPSS/PC 2.0**	**OpenGPSS (один вузол)**
Модельний час		5000	5000
Інформація про пристрій PRIB1			
ENTRIES	Кількість входів в пристрій	338	334
UTIL (%)	Коефіцієнт використання пристрою	0,997	0,997
AVE.TIME	Середній час зайняття пристрою	14,75	15,0
Інформація про чергу QUE1			
MAX.	Максимальна довжина черги	169	161
CONTENT	Поточна довжина черги	168	161
ENTRIES	Кількість входів в чергу	506	495
ENTR (0)	Кількість входів з нульовим часом чекання	2	3
AVE.CON	Середня довжина черги	80,85	81,0
AVE.TIME	Середній час перебування всіх транзактів в черзі	798,89	816,0
AVE.(0)	Середній час знаходження транзактів в черзі крім нульових входів	802,06	821,0

Результати моделювання дещо відрізняються, тому що в кожній системі моделювання використовуються різні датчики псевдовипадкових чисел (ДПЧ).

Розподілені обчислювальні експерименти

Система OpenGPSS використовує автоматичне розподілення GPSS-програм. Далі розглянемо чи впливає це на результати моделювання.

Для того, щоб система OpenGPSS почала працювати в розподіленому режимі, модифікуємо тестову GPSS-програму – додаємо команди керування імітаційним експериментом та отримаємо модифіковану GPSS-програму (табл. 3). Тобто замість того щоб запускати програму моделювання три рази як раніше, будемо виконувати комп'ютерні прогони за допомогою команд керування START, CLEAR та RMULT. Проміжні результати статистики для пристрою PRIB1 та черги QUE1 будемо зберігати у файлі RES.TXT командою RESULT.

Табл. 3. Текст модифікованої GPSS-програми

№	**Команди GPSS**
100	GENERATE 10,5
110	QUEUE QUE1
120	SEIZE PRIB1
130	DEPART QUE1
140	ADVANCE 15,5
150	RELEASE PRIB1
160	TERMINATE
170	GENERATE 5000
180	SAVEVALUE MSV01,FR$PRIB1
190	SAVEVALUE MSV02,FC$PRIB1
200	SAVEVALUE MSV03,FT$PRIB1
210	SAVEVALUE MSV04,Q$QUE1
220	SAVEVALUE MSV05,QA$QUE1
230	SAVEVALUE MSV06,QM$QUE1
240	SAVEVALUE MSV07,QC$QUE1
250	SAVEVALUE MSV08,QZ$QUE1
260	SAVEVALUE MSV09,QT$QUE1
270	SAVEVALUE MSV10,QX$QUE1
400	TERMINATE 1
410	CLEAR
420	RMULT 1,2,8,5,8,2,8
500	START 1
510	RESULT RES.TXT,MSV01,01;FR$PRIB1
520	RESULT RES.TXT,MSV02,02;FC$PRIB1
530	RESULT RES.TXT,MSV03,03;FT$PRIB1
540	RESULT RES.TXT,MSV04,04;Q$QUE1
550	RESULT RES.TXT,MSV05,05;QA$QUE1
560	RESULT RES.TXT,MSV06,06;QM$QUE1
570	RESULT RES.TXT,MSV07,07;QC$QUE1
580	RESULT RES.TXT,MSV08,08;QZ$QUE1
590	RESULT RES.TXT,MSV09,09;QT$QUE1
600	RESULT RES.TXT,MSV10,10;QX$QUE1
610	CLEAR
620	RMULT 5,1,6,6,7,1,1
630	START 1
640	RESULT RES.TXT,MSV01,01;FR$PRIB1

650	RESULT RES.TXT,MSV02,02;FC$PRIB1
660	RESULT RES.TXT,MSV03,03;FT$PRIB1
670	RESULT RES.TXT,MSV04,04;Q$QUE1
680	RESULT RES.TXT,MSV05,05;QA$QUE1
690	RESULT RES.TXT,MSV06,06;QM$QUE1
700	RESULT RES.TXT,MSV07,07;QC$QUE1
710	RESULT RES.TXT,MSV08,08;QZ$QUE1
720	RESULT RES.TXT,MSV09,09;QT$QUE1
730	RESULT RES.TXT,MSV10,10;QX$QUE1
740	CLEAR
750	RMULT 2,2,5,2,8,9,3
760	START 1
770	RESULT RES.TXT,MSV01,01;FR$PRIB1
780	RESULT RES.TXT,MSV02,02;FC$PRIB1
790	RESULT RES.TXT,MSV03,03;FT$PRIB1
800	RESULT RES.TXT,MSV04,04;Q$QUE1
810	RESULT RES.TXT,MSV05,05;QA$QUE1
820	RESULT RES.TXT,MSV06,06;QM$QUE1
830	RESULT RES.TXT,MSV07,07;QC$QUE1
840	RESULT RES.TXT,MSV08,08;QZ$QUE1
850	RESULT RES.TXT,MSV09,09;QT$QUE1
860	RESULT RES.TXT,MSV10,10;QX$QUE1

За допомогою команди RMULT встановлюються початкові значення для всіх датчиків псевдо випадкових чисел. Команда START запускає обчислювальний експеримент. У GPSS-програмі можливо декілька разів використовувати команду START, для отримання проміжних результатів моделювання. Команда CLEAR очищає всю статистику системи і всі видаляє транзакти з моделі. Після виконання команди CLEAR уся система, окрім ДПЧ, знаходиться у початковому стані. Команда RESULT заносить результати моделювання у тимчасовий буфер результатів, для збереження значень і наступного аналізу.

Введемо визначення, які необхідні для подальшого розгляду матеріалу статті.

Нехай *сегмент обчислювального експерименту* (СОЕ) – це послідовність команд керування, яка починається командою CLEAR або RMULT, містить не більше однієї команди CLEAR, RMULT, START і закінчується командою START або RESULT. Сегменти обчислювального експерименту розділяються на два типи: залежні і незалежні.

Будемо називати *залежним сегментом обчислювального експерименту* (ЗСОЕ) такі СОЕ, які не містять команду RMULT або не містять команду CLEAR. Дані в таких сегментах залежать від передісторії станів модельованої системи.

Незалежні сегменти обчислювального експерименту (НСОЕ) – це такі СОЕ, які обов'язково складаються з послідовності команд CLEAR, RMULT, START. Саме послідовність команд CLEAR, RMULT заставляє систему "забути" свої попередні стани, тобто сегмент не залежить від передісторії станів системи.

Будемо називати *кадром сегмента обчислювального експерименту* (КСОЕ) – послідовність СОЕ, яка починається з одного НСОЕ, та включає в себе всі ЗСОЕ, які залежать від цього початкового сегмента.

Припустимо, що S_{ij} – *j*-а імітаційна модель, яка виконується на *i*-му вузлі, $i=\overline{1,n}$, $j=\overline{1,n_i^j}$, де n_i^j – кількість моделей на *i*-му вузлі. У кожної моделі *i*-го вузла є свій модельний час, який позначимо $T_i=\{t_{i1},t_{i2},..,t_{in_i^f}\}$.

Наявність послідовності команд CLEAR, RMULT, START, дозволяє виділити незалежні СОЕ, множину яких позначимо $U_{ij}=\{u_{ij1},u_{ij2},..,u_{ijn_i^u}\}$. Усі інші СОЕ відносимо до залежних, множина – $C_{ij}=\{c_{ij1},c_{ij2},..,c_{ijn_i^c}\}$. Зауважимо, що перший СОЕ в моделі завжди незалежний, тому він належить множині $U_{ij}=\{u_{ij1},u_{ij2},..,u_{ijn_i^u}\}$.

Побудуємо кадр сегмента обчислювального експерименту як послідовність СОЕ, що починається з одного НСОЕ, і включає в себе всі ЗСОЕ, які залежать від цього початкового сегмента:

$$K_{ijm}=\{u_{ijm}\mid\forall m:u_{ijm}\in U_{ij}\}\cup\{c_{if}\mid c_{if}\in C_{ij},$$
$$c_{if}\,залежить_від_u_{im}\},i=\overline{1,n},\ j=\overline{1,n_i^j},m=\overline{1,n_i^u}.$$

З побудови КСОЕ випливає незалежність кадрів один від одного.

Нехай K_{ij} – множина КСОЕ на *i*-му вузлі для *j*-ої моделі:

$$K_i=\bigcup_{n_i^u}^{j=1}K_{ij},$$

а результати проведення всіх СОЕ з КСОЕ позначимо R_{ij}:

$$R_{ij}=\bigcup_{n_i^k}^{m=1}R_{ijm}.$$

Об'єднання результатів по всім КСОЕ *j*-ої моделі дає загальний результат:

$$R_i=\bigcup_{n_i^u}^{j=1}R_{ij}.$$

Тобто проведення одного КСОЕ дає один і тільки один результат з множини R_i.

Нехай A_i = { A_i^{AgSim}, $A_i^{Ag\,Re\,p}$, A_i^{AgSpl}, A_i^{AgSnc}, A_i^{AgTrf}, A_i^{AgPwr}, A_i^{AgUsr}, A_i^{AgGbr} } – множина активних агентів *i*-го сервера, $i=\overline{1,n}$, де *n* – кількість серверів імітаційного моделювання в кластері [9].

Введемо матрицю $\mathbf{B}_{iz}$ розмірністю $n \times n$, яка вказує чи є зв'язок між *i*-м та *z*-м сервером:

$$\mathbf{B}_{iz} = \begin{cases} 1, \textit{якщо існує зв'язок} \\ \qquad \textit{між } i-\textit{м та } z-\textit{м сервером;} \\ 0, \qquad \textit{в іншому випадку;} \end{cases},$$

$$i=\overline{1,n},\ z=\overline{1,n}.$$

Стан *i*-го вузла опишемо як $P_i(A_i,S_i,T_i,K_i,R_i)$, а стан кластера моделювання – $P=\bigcup_{n}^{i=1} P_i(A_i,S_i,T_i,K_i,R_i)$.

Незалежність КСОЕ дозволяє проводити тиражування – одночасне виконання КСОЕ на різних вузлах кластера імітаційного моделювання, з подальшим об'єднанням результатів.

Система OpenGPSS автоматично виділяє множину КСОЕ, які можуть одночасно виконуватись на різних вузлах кластера. Після запуску *j*-ої GPSS-моделі агент користувача A_i^{AgUsr}, який виконує функцію компілятора [10] і переводить GPSS-текст у внутрішній формат та виділяє кадри СОЕ:

$$A_i^{AgUsr}: P_i(S_i,K_i) \to P_i(S_i \cup \{s_{ij}\}, K_i \cup \{k_{ij}\}).$$

Агент реплікації AgRep [11] копіює GPSS-модель та всі КСОЕ на інші вузли:

$$A_i^{Ag\,Re\,p}: P_i(S_i,K_i) \to$$
$$\to P_z(S_z \cup S_i, K_z \cup K_i), \forall i,z: \mathbf{B}_{iz}=1,$$

тому кожний вузол повністю зберігає інформацію про модель та експеримент для запобігання втрат даних у разі вимкнення одного з вузлів.

У кластері відсутнє централізоване керування запуском КСОЕ – кожний вузол випадковим чином визначає КСОЕ із множини ще необроблених сегментів. Це дозволяє уникати конфліктів блокування: жоден вузол не чекає на результати обробки ОЕ іншими вузлами – вони діють за оптимістичним планом – продовжують виконувати моделювання над сегментами. Цей підхід відрізняється від підходу з динамічним розподілом на підмножини вузлів кластера і визначення локального вузла-координатора за допомогою одного з алгоритмів голосування, які наведені в [12] і мають меншу масштабованість (канал зв'язку координатора може бути перевантаженим) та надійність (при відмові координатора необхідно чекати переголосування).

Агент імітаційного моделювання AgSim виконує КСОЕ на *i*-му вузлі:

$$A_i^{AgSim}: P_i(\{t_{i1},t_{i2},..,t_{ij},..,t_{in_j^t}\}) \to$$
$$\to P_i(\{t_{i1},t_{i2},..,\grave{t}_{ij},..,t_{in_j^t}\}),$$
$$\text{де } t_{ij} <= \grave{t}_{ij}.$$

Агент реплікації AgRep, реалізований мовою PL/SQL [13, 14], періодично копіює таблиці результатів REPORT и REPORT_DETAIL на інші вузли:

$$A_i^{Ag\,Re\,p}: P_z(R_z) \to P_i(R_i \cup R_z), \forall i,z: \mathbf{B}_{iz}=1.$$

Потім AgRep виконує *збирання результатів* – об'єднання результатів всіх СОЕ для подальшого виконання та аналізу загальних результатів і завершує ОЕ.

Розглянемо приклад. Розділимо модифіковану GPSS-програму з табл. 3 на сегменти, отримаємо три СОЕ, які наведені в таблиці 4. СОЕ1 – незалежний, тому що це перший СОЕ та його виконання ні від чого не залежить; СОЕ2 та СОЕ3 – також незалежні, тому що виконуються команди CLEAR, RMULT, які встановлюють "початковий" стан системи.

Табл. 4. Уривок тексту GPSS-програми з сегментами

№	Команди GPSS	Сегменти	Кадри сегментів
	…	СОЕ1 (НСОЕ)	КСОЕ1
410	CLEAR		
420	RMULT 1,2,8,5,8,2,8		
500	START 1		
510	RESULT RES.TXT,MSV01,01;FR$PRIB1		
520	RESULT RES.TXT,MSV02,02;FC$PRIB1		
530	RESULT RES.TXT,MSV03,03;FT$PRIB1		
540	RESULT RES.TXT,MSV04,04;Q$QUE1		
550	RESULT RES.TXT,MSV05,05;QA$QUE1		

560	RESULT RES.TXT,MSV06,06;QM$QUE1		
570	RESULT RES.TXT,MSV07,07;QC$QUE1		
580	RESULT RES.TXT,MSV08,08;QZ$QUE1		
590	RESULT RES.TXT,MSV09,09;QT$QUE1		
600	RESULT RES.TXT,MSV10,10;QX$QUE1		
610	CLEAR	COE2 (HCOE)	KCOE2
620	RMULT 5,1,6,6,7,1,1		
630	START 1		
640	RESULT RES.TXT,MSV01,01;FR$PRIB1		
650	RESULT RES.TXT,MSV02,02;FC$PRIB1		
660	RESULT RES.TXT,MSV03,03;FT$PRIB1		
670	RESULT RES.TXT,MSV04,04;Q$QUE1		
680	RESULT RES.TXT,MSV05,05;QA$QUE1		
690	RESULT RES.TXT,MSV06,06;QM$QUE1		
700	RESULT RES.TXT,MSV07,07;QC$QUE1		
710	RESULT RES.TXT,MSV08,08;QZ$QUE1		
720	RESULT RES.TXT,MSV09,09;QT$QUE1		
730	RESULT RES.TXT,MSV10,10;QX$QUE1		
740	CLEAR	COE3 (HCOE)	
750	RMULT 2,2,5,2,8,9,3		
760	START 1		
770	RESULT RES.TXT,MSV01,01;FR$PRIB1		
780	RESULT RES.TXT,MSV02,02;FC$PRIB1		
790	RESULT RES.TXT,MSV03,03;FT$PRIB1		
800	RESULT RES.TXT,MSV04,04;Q$QUE1		
810	RESULT RES.TXT,MSV05,05;QA$QUE1		
820	RESULT RES.TXT,MSV06,06;QM$QUE1		
830	RESULT RES.TXT,MSV07,07;QC$QUE1		
840	RESULT RES.TXT,MSV08,08;QZ$QUE1		
850	RESULT RES.TXT,MSV09,09;QT$QUE1		
860	RESULT RES.TXT,MSV10,10;QX$QUE1		
		Неявне автоматичне збирання результатів	

З прикладу видно, що початкові значення всіх ДПЧ, які використовувались раніше обов'язково повинні бути перевизначені командою RMULT.

Проведемо з GPSS-програмою, яка наведена в табл. 4, обчислювальний експеримент для кількості вузлів кластера один, два та три вузла. Незалежні СОЕ можуть безпечно виконуватись на різних вузлах кластера імітаційного моделювання з подальшим збиранням результатів, що приводить до зменшення часу на проведення імітаційного експерименту. Якщо кластер складається з одного вузла, то всі СОЕ виконуються на ньому. При наявності двох вузлів: на одному вузлі виконуються два сегмента, а третій НСОЕ виконується на другому вузлі. Для трьох вузлів – кожен сегмент виконується на окремому вузлі. Середнє арифметичне результатів наведене в табл. 5. Система GPSS/PC не може працювати в розподіленому режимі, тому для неї наведена інформація при послідовному проведенні експерименту. Як видно із табл. 5 результати моделювання знову приблизно однакові.

Табл. 5. Порівняння результатів розподіленого моделювання

Показник	**Розшифровка**	**GPSS/PC 2.0**	**OpenGPSS (один вузол)**	**OpenGPSS (два вузла[1])**	**OpenGPSS (три вузла[2])**
Астрономічний час, с	Астрономічний час моделювання, секунди	4	8	9	6
Модельний час	Модельний час	5000	5000	5000	5000
FR$PRIB1	Коефіцієнт використання пристрою (в тисячних долях)	997	997	998	998
FC$PRIB1	Число входів в пристрій	338	331	341	338
FT$PRIB1	Середній час використання пристрою	14	15	14	14
Q$QUE1	Довжина черги	168	161	162	158
QA$QUE1	Середня довжина черги	80	79	82	77
QM$QUE1	Максимальна довжина черги	169	162	162	159
QC$QUE1	Загальне число входів в чергу	506	492	503	596
QZ$QUE1	Число нульових входів в чергу	2	1	1	1
QT$QUE1	Середній час знаходження транзакта в черзі (нульові входи включно)	798	811	819	776
QX$QUE1	Середній час знаходження транзакта в черзі (без нульових входів)	802	813	820	777

[1] Використовується середнє арифметичне по двом вузлам.
[2] Використовується середнє арифметичне по трьом вузлам.

Порівняння швидкості моделювання

Розглянемо, як кількість транзактів в моделі впливає на швидкість моделювання, для чого в нашій останній GPSS-програмі у рядку 170 замість блоку «GENERATE 5000» будемо використовувати «GENERATE 5000», «GENERATE 10000», «GENERATE 20000»... «START 90000». Усі експерименти будемо проводити для різної кількості вузлів кластера. Результати проведення експериментів відображені на рис. 1. По осі абсцис відкладені значення модельного часу, а по осі ординат астрономічний час в секундах.

На рис. 1 астрономічний час моделювання для системи з двома вузлами «приблизно» такий самий як і для одного вузла, тому що GPSS-програма містить три (непарне число) сегмента, із-за чого один з вузлів «чекає» другий вузол поки той послідовно моделює два сегменти. Швидкість збільшується але не в два рази. На тренд, що відображає час для трьох вузлів, на початку моделювання впливають накладні витрати, але для модельного часу 90 тис. од. система моделювання OpenGPSS випереджає GPSS/PC.

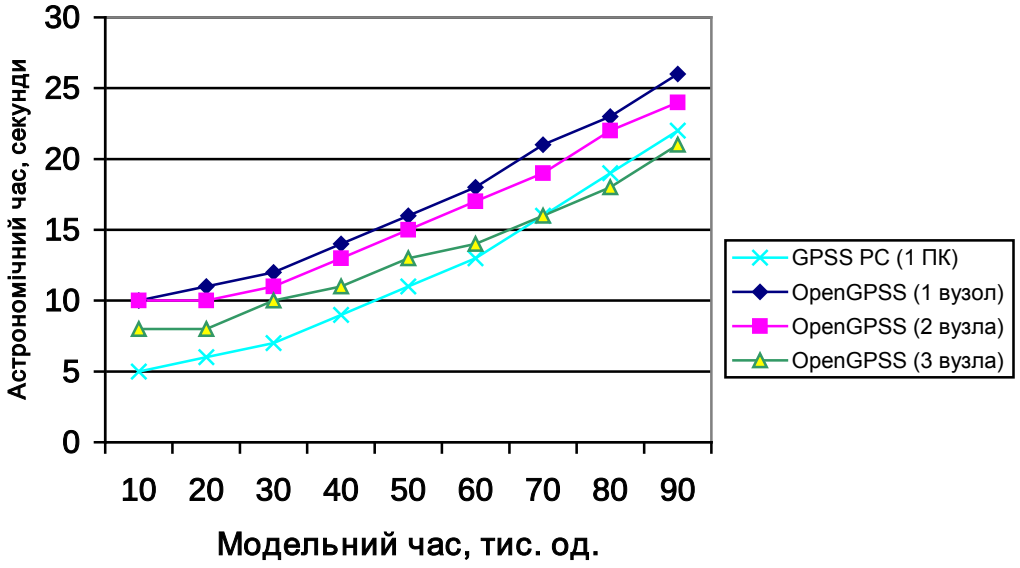


Рис.1. Залежність астрономічного часу моделювання від кількості вузлів кластера та модельного часу

Висновки

1. Системи GPSS та OpenGPSS на однакових GPSS-програмах видають приблизно однакові результати при послідовному і розподіленому режимі проведення експерименту. Для отримання більш точних результатів необхідно збільшити кількість прогонів моделі.

2. Запропонований підхід розділення експерименту на незалежні частини для подальшого паралельного виконання на вузлах кластера і збирання результатів призводить до підвищення продуктивності системи (зменшення часу моделювання), що помітно на кластері з трьома або більше вузлами.

3. Отримані результати підтверджують можливість прозорого автоматичного розділення експерименту на незалежні частини для прискорення швидкості моделювання.

Перспективи досліджень

Оптимізація режимів роботи кластера для підвищення швидкості моделювання.

Побудова та дослідження кластера з великою кількістю вузлів (16-32).

Побудова розподіленої дискретно-подійної системи імітаційної моделювання з оптимістичною синхронізацією модельного часу у вузлах кластера.

Список літератури

1. Richard M. Fujimoto. Parallel and Distributed Simulation Systems. Wiley, 2000.
2. Замятина Е.Б. Современные теории имитационного моделирования: Специальный курс. – Пермь: ПГУ, 2007. – 119 с.
3. SPEEDES. http://www.speedes.com.
4. Mascarenhas E., Knop F., Vernon R. ParaSol: A multithreaded system for parallel simulation based on mobile threads. Winter Simulation Conference, 1995.
5. А.И. Миков, Е.Б. Замятина, А.Н. Фирсов. Инструментальные средства удалённого параллельного моделирования. В книге Proceedings of XXII International Conference “Knowledge-Dialogue-Solution”.– FOI-COMMERCE, Sofia, 2006, pp. 280-287.
6. Шрайбер Томас Дж. Моделирование с использованием GPSS. – М.: Машиностроение, 1980. – 593 с.
7. Киевский центр имитационного моделирования. http://www.simulation.kiev.ua.
8. Діденко Д.Г. Реалізація тиражування обчислювального експерименту в розподіленій системі моделювання OpenGPSS. – Наукові вісті – К. 2007. – № 5. С. 49-53.
9. Томашевский В.Н., Диденко Д.Г. Агентная архитектура распределенной дискретно-событийной системы имитационного моделирования OpenGPSS. Системні дослідження та інформаційні технології. № 4, 2006. – К.: ВПК “Політехніка”, 2006. С.123–133.
10. Ахо Альфред, Сети Рави, Ульман Джеффри. Компиляторы: принципы, технологии, инструменты. Пер. с англ. – М.: Издательский дом "Вильямс", 2001. – 768 с.
11. Діденко Д.Г. Агент реплікації в розподіленій дискретно-подійній системі імітаційного моделювання OPENGPSS. Матеріали міжнародної наукової конференції “Інтелектуальні системи прийняття рішень та прикладні аспекти інформаційних технологій”, 2006. С. 264–266.
12. Таненбаум Э., Стеен ван М. Распределенные системы. Принципы и парадигмы. – СПб.: Питер, 2003. – 877 с.
13. Кайт Том. Oracle для профессионалов. Книга 2. Расширение возможностей и защита: Пер. с англ. – М.: ДиасофтЮП, 2003. – 848 с.
14. Кайт Том. Oracle для профессионалов. Книга 1. Архитектура и основные особенности: Пер. с англ. – М.: ДиасофтЮП, 2003. – 672 с.

УДК 004.93(015.7)

АМОНС О.А. (СТ.ВИКЛ., К.Т.Н., НТУУ «КПІ»),
ЯНОВ Ю.О. (АСПІРАНТ, НТУУ «КПІ»),
БЕЗПАЛИЙ І.О. (СТУДЕНТ, НТУУ «КПІ»)

КЛАСТЕРИЗАЦІЯ ДОКУМЕНТІВ НА ОСНОВІ СТАТИСТИЧНОЇ БЛИЗЬКОСТІ ТЕРМІВ

Анотація

У статті описано підхід до кластеризації колекції документів з невідомою наперед кількістю кластерів. В основу підходу покладено метод, оснований на статистиці появи ключових термів. Запропоновано модифікацію методу знаходження матриці подібності на основі схожості косинуса. Для аналізу якості й знаходження граничних значень алгоритму використана модифікація функції конкурентної подібності. Підхід реалізований у вигляді прикладного застосування сервера SmartBase. Наведені результати експериментальних досліджень запропонованого підходу до кластеризації інформації з використанням часто вживаного текстового корпусу підтверджують працездатність запропонованих рішень.

Abstract

In the given work the approach to clustering of documents collections with unknown quantity of clusters is described. A method of finding matrix of similarity is improved. The method is based on the statistics of key terms occurrence in documents. For quality analysis and finding of limiting values of algorithm, there was used a function of competitive similarity improving. The approach is realized as the application server SmartBase's application. Implementation details and results of the process are shown. Russian text set is used.

Вступ

Зі швидким ростом всесвітньої павутини, впровадженням у міністерствах, відомствах і великих організаціях різноманітних засобів підтримки інформаційно-аналітичної діяльності, насамперед інформаційних систем, систем електронного документообігу, інформаційні бази інформаційно-телекомунікаційних систем спеціального призначення (ІТС СП) набувають суттєвих розмірів. Вони стрімко розширюються переважно за рахунок неструктурованих даних, що породжує проблему швидкого орієнтування в них. Дійсно, відсутність можливості отримувати актуальну і повну інформацію з конкретної теми, яка цікавить користувача, перетворює в непотріб велику частину накопичених ресурсів і робить марними зусилля фахівців. Оскільки безпосередній пошук і аналіз інформації за заданою темою досліджень вимагає все більших трудовитрат, багато рішень приймаються на основі неповного бачення проблеми. Використання засобів для автоматичної систематизації і класифікації текстової інформації дозволяє скоротити час на пошук потрібної інформації, забезпечити її повноту і тим самим підвищити якість дослідження та швидкість реагування на зовнішні зміни. Одним із засобів для підвищення ефективності роботи фахівців в умовах функціонування ІТС СП є інструменти кластеризації текстів, які підлягають аналізу. Стаття присвячена аналізу існуючих методів і засобів пошуку текстової інформації, розробленню дієвого алгоритму кластеризації текстової інформації і його дослідженню на наявному текстовому корпусі.

1 Пошук документів і кластеризація

Пошук документів є однією з найуживаніших операцій оброблення інформації. Дійсно, перш ніж документи можуть бути опрацьованими різними процедурами, їх необхідно знайти серед величезного числа подібних. Якщо пошук документів буде не досить ефективним, то й загальні процедури оброблення інформації, представленої у вигляді документів, не будуть ефективними. Якщо історію пошуку документів можна

розпочинати від зародження бібліотечних інформаційних систем [1], то період його інтенсивних досліджень започаткований формуванням Глобальної мережі і її перетворенням у практичний інструмент підтримки різних видів діяльності людини. Саме тоді з'явилися відомі пошуковики, ефективність яких швидко зростала під впливом нагальних потреб [2]. Останні події зі світу пошуку документів свідчать, що для збільшення його ефективності застосовуються не лише моделі, традиційні для морфологічного і синтаксичного рівнів мовної системи, але й семантичні моделі [3].

Хоч сутність проблеми пошуку документів розуміють як користувачі, так і фахівці в галузі інформаційних технологій (ІТ), лише останні розуміють її складність як науково-практичної проблеми. І джерело цієї складності не стільки у необхідності застосування серйозних моделей і методів лінгвістики, скільки у необхідності такої їх реалізації засобами ІТ, яка задовольняє високий рівень вимог до пошуку, насамперед його адекватності і швидкості. Для кращого розуміння проблем пошуку наведемо необхідні визначення. Під адекватністю розуміють відповідність результатів пошуку інформаційним потребам користувача з погляду визначених цілей і створених на їх основі запитів. Швидкість розглядається у контексті ефективності алгоритму і можливого часу його виконання. Сам пошук інформації становить собою процес виявлення в деякій множині (колекції) документів (текстів) усіх таких з них, які присвячені певній темі, задовольняють заздалегідь визначеній умові пошуку (запиту) чи містять необхідні (що відповідають інформаційній потребі) факти, відомості, дані. Кластеризація – це процес створення кластерів. Згідно з розповсюдженним визначенням, "кластери – це неперервні області (якогось) простору з більш високою щільністю елементів, відділені від інших таких ж областей з відносно низькою щільністю елементів". Під кластеризацією документів розуміють процес поділу всієї колекції на групи, у середині яких знаходяться близькі за тематикою документи, а в різних групах, навпаки, далекі.

Хоч існують різноманітні підходи до побудови пошуковиків, для найефективніших з них загалом властиве застосування, крім зазначених моделей і методів лінгвістики, технологій кластеризації та індексування. І якщо індексування швидше пов'язане з технологічними аспектами, то кластеризація – це традиційна для ІТ наукова проблема. На сьогодні, як засіб для впорядкування текстової інформації, у пошукових системах застосовують різні методи кластерного аналізу. Це, насамперед, статистичні класифікатори на основі ймовірносних методів. Найбільш відомими з них є сім'я Байесових алгоритмів. Наступною групою є класифікатори, що використовують методи на основі штучних нейронних мереж, наприклад алгоритми ART, SOM [4]. І останню групу складають класифікатори, що базуються на функціях схожості, насамперед k-means, Forel, FRiS Cluster [5,6,11].

Одними із найуживаніших є методи, засновані на метриці близькості. У цьому випадку документи представляються у вигляді вектору ознак у просторі ознак, тобто набором ключових слів. Є декілька підходів до його формування. В найпростішому випадку кожна ознака відповідає присутності в тексті одної із словоформ, що зустрічається у текстовій колекції. Величину кожного елемента вектора можуть підраховувати по різному: наприклад, прирівнювати одиниці, якщо ознака зустрічається у даному тексті, чи нулю у іншому разі; вона може бути рівною кількості входжень його у документ, нормованою до кількості ознак; чи також враховувати частоту появи ознаки у всьому текстовій колекції. Таке представлення має суттєвий недолік - простір ознак має велику розмірність, більша частина його елементів є надмірними, навіть шкідливими. Для подолання цієї проблеми використовуються методи зменшення розмірності простору ознак. Це, насамперед, виділення лем слова, нормальних форм, видалення стоп-слів, використання синонімічних груп тощо. Але в цьому випадку є ймовірність не використати значущу інформацію.

На наступному кроці підраховується матриця близькості між векторами документів. І виконується власне кластеризація. Більш детальну інформацію з описом особливостей існуючих алгоритмів можна знайти в працях [7,8].

Крім загальних проблем, для всіх методів кластеризації текстів постає проблема відображення змісту кластеру, на основі якого до нього приєднується той чи інший текстовий документ. Це необхідно для зручного використання результатів людиною. Найбільш розповсюджений підхід до рішення цієї проблеми складається у використанні представлення кластеру у вигляді набору найбільш важливих слів.

2. Постановка проблеми

Проблема полягає у розробленні текстового кластеризатора і дослідженні його властивостей. Загальну модель текстового кластеризатора можна представити багатоосновною алгебраїчною системою такого вигляду:

$$R = < T, C, D, B, R, S > \quad (1)$$

де:

$T = \{T_1, T_2, ..., T_l, ..., T_n\}$ – множина текстів, що підлягають класифікації (колекція);

$T_l = \{t_1, t_2, ..., t_j, ...\}$ – множина термів, з яких складається l-й документ;

$C = \{C_1, C_2, ..., C_k\}$ – множина класів-рубрик (кластерів), де k – кількість кластерів;

$D = \{D_1, D_2, ..., D_m\}$ – множина описів, кожний з яких має певну внутрішню структуру, де m – кількість описів;

$B = \{b_1, b_2, ..., b_k\}$ – множина еталонних зразків (стовпів), $i = 1,2,\ldots,k$;

$R \subset C \times D$ – відношення між кластерами і описами, яке має таку властивість: $\forall C_i \in C \; \exists \; D_j \in D : (C_i, D_j) \in R$, причому кожному кластеру відповідає єдиний опис;

S – сигнатура, яка включає такі операції:

S_1: $T \rightarrow C$ – операція кластеризації, яка полягає у виконанні перетворень над текстами, після яких, або робиться висновок про належність документа T_l зі структурою D_l до класу C_i, або висновок про створення нового кластеру C_j, до якого можна буде віднести даний текстовий документ. Будемо вимагати, щоб жодний текст не міг відноситись до декількох кластерів одночасно;

S_2: $C \times C \rightarrow C$ – теоретико-множинна операція перетину кластерів;

S_3: $C \times C \rightarrow C$ – теоретико-множинна операція об'єднання кластерів.

3. Загальний опис підходу до розв'язання проблеми

Підхід, реалізований авторами, полягає у використанні векторної моделі текстових документів, згідно з якою кожний текстовий документ представляється у вигляді вектору зважених ознак і належність документів до кластерів визначається на основі міри близькості відповідних векторів.

Досить логічно кластеризація виконується у таких чотири основних етапи:

Етап 1. Це підготовчий етап, який полягає у переході від множини текстів T до множини T^* їх векторів.

Етап 2. На цьому етапі відбувається перехід до множини описів документів з урахуванням ваги ознак. Будемо використовувати статистичні міри ваги, які добре зарекомендували себе у пошуку документів, насамперед міри, що характеризують частку деякого терму документа у загальній кількості термів та їх появи в документах всього набору [8].

Етап 3. Будуємо матрицю близькості між документами на основі популярної функції схожості косинуса, фізичним змістом якої є косинус кута між векторами.

Етап 4. На цьому етапі виконується власне кластеризація.

Розглянемо наведені етапи більш детально.

1) Підготовчий етап. Для кожного документу T_l множини текстів T виділяємо множину значимих слів, які приводяться до нормальних форм. Будемо традиційно називати ці слова ключовими термами і позначати t_q. Як і в багатьох інших підходах, ми розглядаємо в якості ключових ті терми, частота яких у даному тексті істотно перевищує деяку середню частоту. Крім того ми будемо виключати стоп-слова, такі як прийменники, сполучники тощо.
2) Від колекції текстів переходимо до множини описів текстів. Знаходимо вагу кожного терму. Для цього кожному терму t_i документу T_l ставимо у відповідність статистичну міру w_{li}, що характеризує відношення кількості входжень цього терму у документ до загальної кількості термів та враховує час-

тоту появи терму в документах всієї колекції:

$$w_{li} = -\log(p(t_i)\cdot f_t(T_l,t_i))$$
$$f_t(T_l,t_i) = \frac{f_r(T_l,t_i)}{f_r(T_l,t_i)+1+\frac{d(T_l)}{350}}, \quad (2)$$
$$p(t_i) = 1 - e^{-1.5\cdot\frac{f_c(t_i)}{n}}$$

де

$f_r(T_l,t_i)$ – число входжень терма t_i у документ T_l;

$d(T_l)$ – кількість термів у документі T_l;

$f_c(t_i)$ – число входжень терма t_i в колекцію.

3) Будуємо матрицю близькості між документами на основі популярної функції схожості косинуса, фізичним змістом якої є косинус кута між векторами:

$$sim(T_1,T_2) = \frac{\sum_i w_{1i} w_{2i}}{\sqrt{\sum_i w_{1i}^2 \sum_i w_{2i}^2}}, \quad (3)$$

де w_{1i}, w_{2i} - вага терма t_i в документах T_1,T_2 відповідно.

У виразі (3) враховуються лише терми, що входять одночасно як в документ T_1, так і в документ T_2. Але цій оцінці притаманні певні недоліки. Продемонструємо їх на прикладі. Розглянемо два тексти:

a) “Президент обратился с представлением в Конституционный суд с просьбой растолковать конституционный механизм замены отдельных членов правительства.”

b) “Как Президент выражаю почет и благодарность всем ветеранам трагической афганской войны и украинским воинам-миротворцам за отвагу, доблесть, верность присяге и чести”

У цих текстах є лише одне спільне ключове слово “Президент” і воно вживається лише одного разу. І оскільки кількість ключових термів у кожному з документів приблизно рівна, то використання виразів (2) і (3) дає оцінку близькості sim(T_1,T_2) ~ 1, що не повною мірою характеризує дійсний зв’язок цих текстів.

Цього недоліку можна позбутися використовуючи для оцінки близькості такий вираз:

$$sim(T_1,T_2) = \frac{\sum_i w_{1i} w_{2i}}{\sqrt{\sum_i w_{1i}^2 \sum_i w_{2i}^2}} \cdot \sum_i \frac{w1_{1i} + w2_{2i}}{w1_{1i} + w2_{2i}}, \quad (4)$$

$w1_{1i}, w2_{1i}$ - вага термів, що одночасно присутні в обох документах.

4) Власне кластеризація. Будемо виконувати кластеризацію на основі FRiS-функцій [10]. Розглядаємо колекцію T із n документів, що розбита на k угрупувань. Кожне угрупування i описане одним еталонним об’єктом (стовпом, центроїдом) b_i. Для будь-якого документа $T_1 \in T$ можна знайти відстань $r(T_1,b_i)$ до найближчого стовпа угрупування i. Тоді $r1(T_1,b_i) = \min_i r(T_1,b_i)$ - близькість до найближчого стовпа i, а $r2(T_1,b_i) = \min_{i \neq i^*} r(T,b_i)$ - близькість до найближчого конкурента.

FRiS-функція [10], модифікована для використання матриці близькості між документами, має такий вигляд:

$$F(T,B) = \left(\frac{r1(T,b) - r2(T,b)}{r1(T,b) + r2(T,b)}\right) \quad (5)$$

Вона є мірою подібності об’єкту T зі стовпом b_i у конкуренції з іншими стовпами.

Знайшовши середнє значення FRiS-функції по усій вибірці, за допомогою виразу (6) отримаємо величину $F(B)$, що характеризує наскільки повно набір стовпів характеризує колекцію:

$$F(B) = (1/m)\sum_{T_i \in T} F(T_i,b) \quad (6)$$

У роботі також будемо використовувати редуковану FRiS-функцію, у якій стовп конкурента рівновіддалений від кожного об’єкта на відстань $r2^*$. Тоді вирази (5), (6) набудуть такого вигляду:

$$F^*(T,B) = \left(\frac{r1(T,b) - r2^*(T,b)}{r1(T,b) + r2^*(T,b)}\right) \quad (7)$$
$$F(B) = (1/m)\sum_{T_i \in T} F^*(T_i,b) \quad (8)$$

4 Опис алгоритму кластеризації

Деталізуючи наведений узагальнений опис, кластеризацію будемо виконувати за допомогою наступного алгоритму:

Крок 1. При k=1 створюємо новий кластер C_{first} і всі документи включаємо до нього.

Крок 2. У кластері, отриманому на попередньому етапі, призначаємо стовпом довільно вибраний документ t і для нього по формулі (8) обчислюємо середню редуковану FRiS-функцію $F^*(B)$.

Крок 3. Крок 1 повторюється при призначенні стовпами всіх m об'єктів кластеру почергово. Першим стовпом b_1 обирається документ T^*_1, для якого величина F^* виявляється максимальною.

Крок 4. Призначаємо другим стовпом довільно обраний документ і підраховуємо для нього середню редуковану FRiS-функцію.

Крок 5. Крок 4 повторюється при призначенні стовпами всіх m документів кластера почергово, крім існуючого стовпа b_1. Другим стовпом b_2 вибирається документ T^*_2, для якого величина F^* виявляється максимальною.

Крок 6. Замість одного кластеру C_{first} створюємо два нових з центроїдами b_1 та b_2.

Крок 7. Після того, як були знайдені два нових стовпи, уся колекція розподіляється за наступним правилом: документ відноситься до того кластеру, для якого близькість $r1$ до найближчого центроїду максимальна.

Крок 8. Якщо при k=1 у перший кластер C_{first} входили всі m документів, то тепер при k=2 документи розподіляються між двома новими кластерами. При цьому може вийти так, що для опису кластера C_1 найкращим виявиться не стовп b_1, а якийсь інший документ із цього кластера. Для покращення місця розташування стовпа b_1 виконаємо наступну процедуру. Почергово для кожного кластера, у нашому випадку для C_1,C_2, призначаємо внутрішні документи на роль стовпа, і за допомогою виразу (6) обчислюємо середню FRiS-функцію $F(T_{1i},b_i)$. Центроїдом вибирається той документ, що забезпечує максимальну величину FRiS-функції. Аналогічно у кластері C_2 визначається нове положення стовпа b_2 на основі максимуму функції $F(T_{2i},b_i)$. На цьому етапі замість редукованаї FRiS-функції використовуємо звичайну, що відображає процес конкуренції між реальними стовпами.

Крок 9. Для подальшого розбиття сукупності документів, необхідно вибрати кластер з мінімальним загальним відхиленням від середнього значення сумісної близькості документів. Підрахунок її виконуємо за допомогою виразу (9)

$$E_k = \sqrt{\sum_{i=1}^{nk}(sim(T_i,b_k) - middleSim_k)^2}, \quad middleSim_k = \frac{1}{n_k}\sum_{i=1}^{nk} sim(T_i,b_k), \qquad (9)$$

де n_k - кількість документів у кластері.

Кластер з мінімальною сумісною близькістю E_k вибираємо для подальшої роботи.

Крок 10. Процес продовжується, переходячи на крок 2, тобто у вибраному кластері шукаємо 2 стовпа за допомогою описаного вище алгоритму, створюємо 2 нових кластера, перерозподіляємо всі документи між всіма стовпами, і уточнюємо місце розташування центроїда. Процес повторюється доти, поки колекція документів не поділиться на k кластерів.

5. Програмна реалізація

Реалізація і дослідження розробленого алгоритму виконується у рамках розробки системи прийняття рішень на платформі SmartBase.

Для виконання досліджень була використана колекція документів з Інтернет-ресурсу аналітичної газети «Дзеркало тижня» за 3 періоди. Вона включає новини з 26 різних рубрик. Рубрики використовувались як еталони у подальшому дослідженні.

Для стемінгу, тобто виділення нормальної форми слова та виявлення належності його до тієї або іншої частини мови, було використано вдосконалений алгоритм SnowBall.

У виконаних експериментах використовувався список стоп-слів. Він зчитувався з файлу stopword.xml, створеного на основі списку Штейнфельдта, а також з переліку слів, вільно розповсюджуваного компанією "Яндекс" продукту Yandex.Server-FREE-020-3.8.3. Список включає 279 часто вживаних слів російської мови.

Для видалення HTML-тегів була розроблена власна бібліотека HTML Parser, що надає потрібну функціональність.

Програмний інтерфейс складається з web-застосування, яке надає можливість зчитувати новини з сайту http://www.zn.ua/ за певний період чи з локального файлу даних. Вхідними даними для алгоритму є максима-

льна кількість кластерів k, $k \in (1,...,n)$, де n – кількість документів у колекції. У алгоритмі кластеризації використовується лише тіло документу (основний текст), заголовок і додаткова інформація не враховуються.

Експерименти проводились на двопроцесорному сервері Intel(R) Xeon (TM) CPU 2,40 GHz 2,40 GHz 1GB RAM. Результати передавались клієнту по http протоколу. Характеритики клієнта наступні: Intel(R) Core(TM) 2 Duo CPU 2,33 GHz 2GB RAM.

6. Методика проведення експерименту

Кожний документ із колекції методом експертної оцінки було віднесено до певних рубрик (наприклад, людина, внутрішня політика, міжнародна політика, право, тощо). Максимальна кількість таких рубрик 26. Але для кожної конкретної колекції документів рубрик може бути менше. На основі використання цих даних були створені еталонні кластери.

Існує декілька характеристик оцінки якості роботи текстового класифікатора. Найбільше поширення одержали точність (P) і повнота (R). Вони також використовуються при оцінці якості пошуку за запитом, наприклад, у пошукових машинах мережі Інтернет.

Під правильною і неправильною рубрикацією будемо розуміти випадки, коли класифікатор приписує аналізований документ деякій рубриці, що розцінюється деяким експертом як вірне і невірне рішення відповідно. Під невірнім відсіюванням документу розуміємо випадок, коли класифікатор не приписує документ рубриці, що, на думку експерта, невірно.

Для аналізу отриманих результатів використовувалися графіки точності/повноти. Незважаючи на широку поширеність і популярність метрик точності і повноти у задачах інформаційного пошуку, стосовно задачі розбиття документів на кластери їх застосування вимагає деяких уточнень. Введемо наступні позначення:

C_{ei} – i-й нетривіальний (утримуючий більше одного документа) еталонний (складеними експертами) кластер;

C_j – j-й нетривіальний текстовий (побудованою системою) кластер;

$$M_{ij} = \begin{cases} \left|C_{ei} \cap C_j\right|, при \left|C_{ei} \cap C_j\right| > 1; \\ 0, у \quad іншому \quad випадку; \end{cases}$$

$$I_{ij} = \begin{cases} 1, при \left|C_{ei} \cap C_j\right| > 1; \\ 0, у \quad іншому \quad випадку; \end{cases}$$

$$U_j = \begin{cases} \sum_i I_{ij}, при \sum_i I_{ij} > 1; \\ 0, у \quad іншому \quad випадку; \end{cases}$$

$$K_i = \begin{cases} \sum_j I_{ij}, при \sum_j I_{ij} > 1; \\ 1, у \quad іншому \quad випадку. \end{cases}$$

Тоді точність є відношенням різниці потужності перетину документів в еталонних і в тестових кластерах і кількості повторно використаних (при побудові даного перетину) еталонних кластерів до ефективної кількості документів у тестових кластерах:

$$P = \frac{\sum_{ij} M_{ij} - \sum_j U_j}{\sum_i \left|C_{ei}\right| \cdot K_i}$$

Повнота виражається відношенням різниці потужності перетину документів в еталонних і в тестових кластерах і кількості повторно використаних (при побудові даного перетину) еталонних кластерів до кількості документів в еталонних кластерах:

$$R = \frac{\sum_{ij} M_{ij} - \sum_j U_j}{\sum_j \left|C_j\right|}$$

7. Результати експериментальних досліджень

Дослідження проводились на трьох колекціях документів, за різний проміжок часу:

- за період з 29.12.2007 по 26.04.2008, 1515 документів;
- за період з 26.04.2008 по 27.00.2008, 1543 документів;
- за період з 31.05.2008 по 5.06.2008, 418 документів;

У тестуванні брало участь 2 алгоритми: розроблений авторами алгоритм Cluster і алгоритм FRiS Cluster [11] (далі FRiSCluster). Останній обраний за результатами досліджень, наведених у праці [9], як кращий з існуючих алгоритмів цього класу. Обидва алгоритми оперують поняттям

центра кластеру. Після закінчення своєї роботи вони надають користувачу не тільки результуючий розподіл, але і еталонні зразки - стовпи.

На рисунках 1 - 3 наведені результати оцінки якості кластеризації за показником точності/повноти для різної кількості документів у колекції новин, від 100 до 1600. Кожний документ було представлено за допомогою двох десятків найбільш інформативних термів.

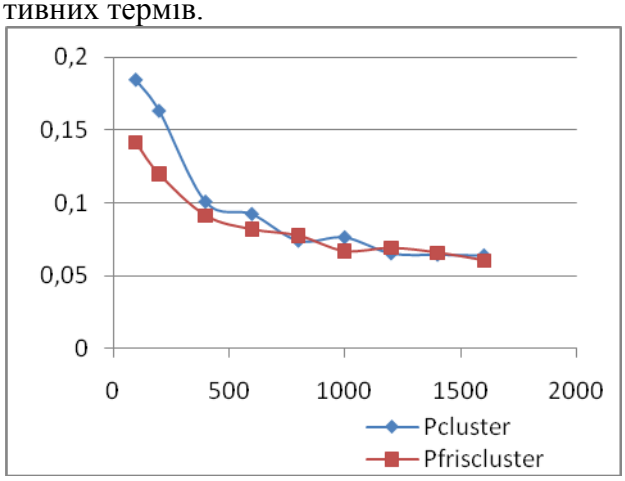


Рис. 1.Порівняння точності алгоритмів

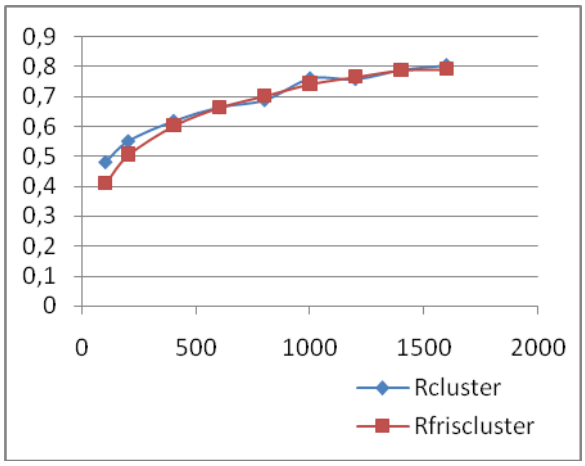


Рис. 2. Порівняння повноти кластеризації

Графіки точності і повноти, зображені на рисунках 1 і 2 відповідно, показують що розроблений алгоритм має співставні з конкурентом, який є одним із найкращих серед відомих алгоритмів, результати. Тільки при невеликій кількості документів його точність відчутно перевищує точність алгоритму FRiSCluster

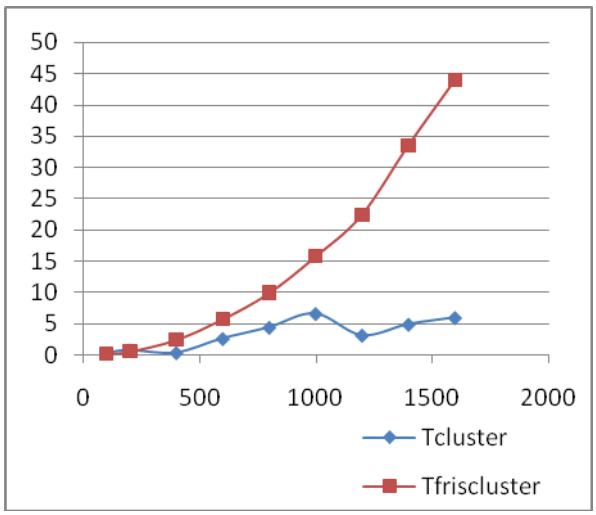


Рис. 3. Порівняння швидкості роботи алгоритмів

На рисунку 3 відображено залежність часу роботи алгоритмів у секундах (ордината) від кількості документів (абсциса). Судячи з графіків, алгоритм Cluster має суттєві переваги, час його роботи менш залежить від збільшення кількості документів у колекції.

Необхідно звернути увагу на невелику точність результатів. Це пояснюється насамперед недосконалістю алгоритму стемінгу, не врахуванням віднесення слова до тієї або іншої частини мови, не врахуванням синонімії, тобто браком засобів виявлення значущих слів та їх відношень.

8. Перспективи досліджень

Для поліпшення показників якості роботи прикладного застосування і розширення сфери його використання видається необхідним виконати такі роботи:

1. Розробка, реалізація і дослідження більш ефективного алгоритму стемінгу.
2. Розробка, реалізація і дослідження моделей представлення семантики і методу кластеризації колекції документів на основі статистичної і семантичної близькості, що дозволить врахувати синонімію та інші відношення термів.
3. Розробка, реалізація і дослідження алгоритму виявлення словосполучень.
4. Розширення кількість мов, з якими працює текстовий кластеризатор.

Висновки

5. Запропоновано підхід до кластеризації колекції документів з невідомою наперед кількістю кластерів на основі статистичних показників, що характеризують кількість

входжень ключових термінів у документах і колекції у цілому.

6. Удосконалено метод знаходження матриці подібності на основі схожості косинуса, для аналізу якості й складності якого використана модифікація функції конкурентної подібності.

7. Підхід реалізований у вигляді прикладного застосування сервера SmartBase і виконане його експериментальне дослідження з використанням наявного текстового корпусу. Результати досліджень підтверджують працездатність запропонованих рішень і їх відповідність за показниками точності і повноти відомим розробкам, а за показником швидкості перевищують їх.

Перелік літератури

1. Дж Солтон. Динамические библиотечно-информационные системы. М.: - Мир, 1979.- с.557.
2. Козлов Д.Д. Проблемы применения методов поиска тематических сообществ к задаче тематического информационного поиска в интернет // Труды Всероссийской научной конференции "Методы и средства обработки информации" -М.: Издательский отдел факультета ВМиК МГУ, 2003. - С. 211-215.
3. AllaZaboleeva, Yulia Orlova Computer-aided system of semantic text analysis of a technical specification//Information Technologies and Knowledge. – 2008. – Vol.2. – P.139-145.
4. Vassilis G. Kaburlasos Unified Analysis and Design of ART/SOM Neural Networks. Heidelberg: - Springer Berlin, Volume 4507, 2007.-p 80-93
5. MacQueen J. Some methods for classification and analysis of multivariate observations // Proceedings of the 5th Berkley Symposium on Mathematical Statistic and Probability, University of California Press, 1967, Vol.1, p. 281-297.
6. Peter Grabusts A Study of Clustering Algorithm Application In RBF Neural Networks. //Information technology and management science. - Riga, - 2001. - 5.serija., p.50-57.
7. Корунова Н. В., Кластеризация документов проектного репозитария на основе нейронной сети Кохонена // Труды конф. «Нечеткие системы и мягкие вычисления». – М. - 2008. – С. 77-86.
8. М. В. Киселев, В. С. Пивоваров, М. М. Шмулевич Метод кластеризации текстов, учитывающий совместную встречаемость ключевых терминов, и его применение к анализу тематической структуры новостного потока, а также ее динамики// Автоматическая обработка веб-данных. - М., 2005. - С. 412-435.
9. Красильников П.В. Воспроизведение лучших результатов ad hoc поиска семинара РОМИП. -М.: IMAT, 2007. – 220с.
10. Борисова И.А., Дюбанов В.В., Загоруйко Н.Г., Кутненко О.А. Использование FRiS-функции для построения решающего правила и выбора признаков // Материалы Всероссийской конференции с международным участие «Знания – Онтологии – Теории» (ЗОНТ–07), Новосибирск, 2007 г. – Т. 2. – С. 67-76.
11. Борисова И.А., Загоруйко Н.Г., Функции конкурентного сходства в задаче таксономии //Материалы Всероссийской конференции с международным участие «Знания – Онтологии – Теории» (ЗОНТ–07), Новосибирск, 2007 г. – Т. 2. – С. 77-86.

УДК 62-82:658.512.011.56

АЛЕНИН О.И.

СИСТЕМА УПРАВЛЕНИЯ ВИРТУАЛЬНЫМИ КЛАСТЕРАМИ

Рассмотрен способ повышения эффективности использования кластерных систем с помощью виртуализации. Сделан краткий обзор типов виртуализации, выбран наиболее подходящий. Предложена структура системы управления виртуальными кластерами в рамках физического кластера.

The way to increase cluster system efficiency using virtualization is considered. The short review of virtualization types is made, and the most suitable is chosen. The structure of management system for virtual clusters over a physical cluster is proposed.

В настоящее время в связи с расширением круга научно – исследовательских задач требуются значительные вычислительные мощности для расчетов, как правило, большие, чем вычислительные мощности соответствующих организаций. При этом вычислительные ресурсы других организаций, доступные удаленно, не удовлетворяют требованиям задач из-за отсутствия на них необходимого специального программного обеспечения [1]. Часто это программное обеспечение может работать только на определенной платформе.

В настоящее время для решения этой задачи используются различные системы виртуализации. В большинстве случаев они позволяют виртуализировать только вычислительные узлы [2]. Вторая проблема заключается в том, что при этом основное внимание уделяется виртуализации вычислительных ресурсов, а виртуализация дискового пространства и сети выполняется неэффективно или не выполняется вообще. С переходом на Грид технологии возникает новая проблема, связанная с тем, что хотя грид может предоставить доступ к большому числу разнообразных ресурсов, часто эти ресурсы не соответствуют требованиям конкретных приложений или сервисов[3]. В вычислительной среде, где программное обеспечение развивается быстро, это несоответствие может привести к недоиспользованию ресурсов, недовольству пользователей и необходимости затрачивать большие усилия на преодоление несоответствия между ресурсами и приложениями. Эти проблемы могут быть решены путем создания кластеров виртуальных машин, с соответствующим набором программного обеспечения, необходимого для запуска приложений. Анализ подобного показывает эффективность его использования в грид – приложениях. При использовании виртуальных машин для запуска приложений потери производительности из-за использования виртуализации не превышают 5%. В работе [4] показано также, что учитывая время на создание и развертывание виртуальной среды в планировщике, а не оставляя этот процесс пользователю, можно достичь значительно большей эффективности использования ресурсов и более точного соответствия требованиям ко времени выполнения заданий. Учет в планировщике времени на передачу и развертывание образов виртуальных машин имеет два преимущества: во-первых, передачу образа можно запланировать и выполнить заранее, во-вторых, можно кешировать часто используемые образы виртуальных машин.

В статье [5] описан способ выделения ресурсов с помощью виртуальных машин, в котором предлагается модифицировать и расширить функции существующих планировщиков, таких как PBS и Sun Grid Engine, что также приводит к повышению эффективности использования ресурсов. Современные кластерные системы используются для разных типов задач. Некоторые задачи требуют большого объема ресурсов, но не накладывают строгих ограничений на время выполнения, в то время как другие задачи должны выполняться с максимальным приоритетом в заданный момент времени [6]. Использование виртуализации предоставляет возможность приостанавливать и возобновлять выполнение задач, переносить задачи вместе с виртуальными машинами на другие физические узлы, а также предоставлять вычислительные ресурсы (виртуальные машины) с предустановленным программным обеспечением, необходимым для выполнения данной задачи.

Есть несколько факторов, оказывающих существенное влияние на эффективность применения виртуализации для систем с кластерной архитектурой:

- используемый тип виртуализации;
- выбор между виртуализацией отдельных ресурсов и построением виртуальных кластеров;
- эффективность работы системы управления ресурсами;
- класс задач, решаемых на кластере.

Для выполнения параллельных прикладных программ, изначально рассчитанных на работу в системах с кластерной архитектурой, целесообразно виртуализировать не отдельные ресурсы кластера, а создать на основе физических ресурсов виртуальный кластер, наиболее точно соответствующий требованиям задач. Для запуска на кластере множества не параллельных программ виртуализация отдельных ресурсов может оказаться предпочтительнее из-за отсутствия накладных расходов на систему управления виртуальными кластерами. Однако, большинство программ, выполняемых на кластерных системах, являются параллельными, и здесь будет рассматриваться только этот вариант.

Типы виртуализации

Существуют такие типы виртуализации: *эмуляция аппаратуры, полная виртуализация, паравиртуализация, виртуализация уровня операционной системы* и *виртуализация уровня приложений*.

При использовании *эмуляции аппаратуры* полностью эмулируется архитектура целевой машины, фактически происходит интерпретация команд гостевого процессора на хост-процессоре. Это самый медленный тип виртуализации, программы выполняются в сотни раз медленнее, чем на физической машине. Примеры систем – Bochs, QEMU.

Полная виртуализация – самый популярный способ виртуализации, предполагает использование программного обеспечения, получившего название «гипервизор», суть которого заключается в создании уровня абстракции между виртуальными серверами и базовым аппаратным обеспечением. Примерами коммерческих решений, в которых реализован данный подход, могут служить программные продукты VMware и Microsoft Virtual PC, а KVM (Kernel Virtual Machine) – это свободно распространяемое решение для ОС Linux. Гипервизор перехватывает команды центрального процессора и служит посредником для доступа к аппаратным контроллерам и периферии. В результате полная виртуализация позволяет установить на виртуальный сервер практически любую операционную систему без каких-либо изменений, причем сама ОС ничего не будет знать о том, что она работает в виртуализованной среде. Основной недостаток данного подхода связан с накладными расходами, которые несет процессор в связи с работой гипервизора. Эти накладные расходы невелики, но ощутимы [7]. В полностью виртуализованной среде гипервизор взаимодействует непосредственно с аппаратным обеспечением и серверами в качестве хостовой операционной системы. Операционные системы, работающие на виртуальных серверах, которыми управляет гипервизор, называют гостевыми.

Полная виртуализация предполагает серьезное использование ресурсов процессора, обусловленное наличием гипервизора, управляющего различными виртуальными серверами и обеспечивающего независимость этих серверов друг от друга. Уменьшить эту нагрузку можно, например, модифицировав каждую операционную систему таким образом, чтобы она «знала» о том, что она работает в виртуализованной среде, и могла взаимодействовать с гипервизором. Такой подход называют *паравиртуализацией*. Примером свободно распространяемой реализации технологии паравиртуализации может служить Xen. Прежде чем операционная система сможет работать в качестве виртуального сервера в гипервизоре Xen, в нее необходимо внести определенные изменения на уровне ядра.

Существует еще один способ виртуализации – встроенная поддержка виртуальных серверов на уровне операционной системы. Этот подход называется *виртуализацией на уровне операционной системы* и использован, например, в Solaris Containers. Существует также Virtuozzo/OpenVZ для ОС Linux. При виртуализации на уровне операционной системы не существует отдельного слоя гипервизора. Вместо этого сама хостовая операционная система отвечает за разделение аппаратных ресурсов между несколькими виртуальными серверами и поддержку их независимости друг от друга. Отличие этого подхода от

других проявляется, прежде всего, в том, что в этом случае все виртуальные серверы должны работать в одной и той же операционной системе (хотя каждый экземпляр имеет свои собственные приложения и регистрационные записи пользователей). Виртуализация на уровне операционной системы теряет в гибкости, но производительность близка к производительности физического сервера. Кроме того, системой, которая использует одну стандартную ОС для всех виртуальных серверов, намного проще управлять, чем более гетерогенной средой.

Виртуализация прикладных приложений включает в себя рабочую среду для локально выполняемого приложения, использующего локальные ресурсы. Виртуализируемое приложение запускается в небольшом виртуальном окружении, которое включает в себя ключи реестра, файлы и другие компоненты, необходимые для запуска и работы приложения. Такая виртуальная среда работает как прослойка между приложением и операционной системой, что позволяет избежать конфликтов между приложениями.

Каждый тип виртуализации обладает своими достоинствами и недостатками. При применении виртуализации для повышения эффективности использования кластерных систем к выбранному типу виртуализации предъявляются такие основные требования:

- невысокие накладные расходы на виртуализацию;
- возможность запуска в гостевой среде операционных систем с предустановленным программным обеспечением.

Следовательно, нецелесообразно использовать эмуляцию аппаратуры и невозможно использовать виртуализацию уровня приложений. Использование паравиртуализации целесообразно только в тех случаях, когда модификация гостевой ОС не представляет трудностей. Наиболее приемлемым является использование виртуализации уровня ОС, если гостевая ОС близка к базовой (например, различные версии Linux), и полной виртуализации для запуска других гостевых ОС (например, запуск Windows на Linux).

Система управления виртуальными кластерами

Как уже было сказано, одним из факторов, существенно влияющих на эффективность использования виртуализации в кластерных системах, является система эффективная работа системы управления виртуальными кластерами. Она управляет виртуальными узлами, сетями, дисковыми ресурсами, осуществляет создание и удаление виртуальных кластеров, мониторинг состояния узлов и т.д.

Для возможности одновременной работы нескольких виртуальных кластеров с разным набором прикладного ПО на одном физическом кластере, необходимо, чтобы с точки зрения гостевых ОС виртуальные кластера были изолированными друг от друга, использовали разные дисковые и сетевые ресурсы. Такая схема показана на рис. 1

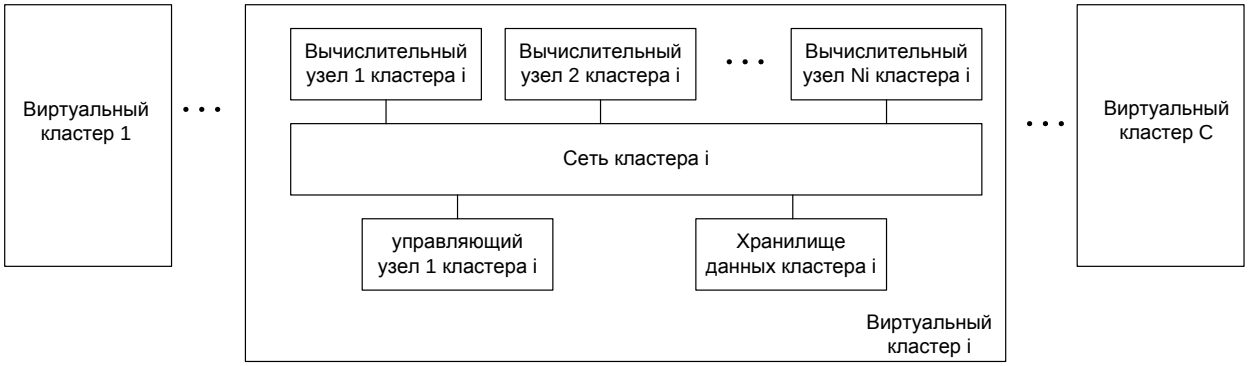


Рис.1 Виртуальные кластера

В соответствии с такой схемой, система управления виртуальными кластерами должна содержать блоки, обеспечивающие управление вычислительными, сетевыми и дисковыми ресурсами. На рис. 2 показана схема системы управления виртуальными кластерами и взаимодействие ее с другими системами, обеспечивающими работу физического кла-

стера, такими как система управления ресурсами физического кластера, интерфейс грузки заданий, система хранения.

Систему можно разделить на несколько блоков. Блок управления вычислительными ресурсами отвечает за создание, удаление и запуск виртуальных узлов на физических узлах. Блок управления системой хранения отвечает за распределение доступного дискового пространства, создание и удаление виртуальных дисковых ресурсов. Блок управления сетевыми службами осуществляет создание, удаление и настройку виртуальных фейсов, распределение адресов, управляет сетевыми сервисами, необходимыми для загрузки виртуальных узлов. Блок обработки очереди заданий связан непосредственно с интерфейсом загрузки заданий (или с мой управления заданиями физического кластера, если физические ресурсы виртуализируются не полностью, и есть возможность запуска заданий в базовой ОС физических узлов). База данных с информацией о ресурсах содержит все записи, необходимые для ты остальных блоков.

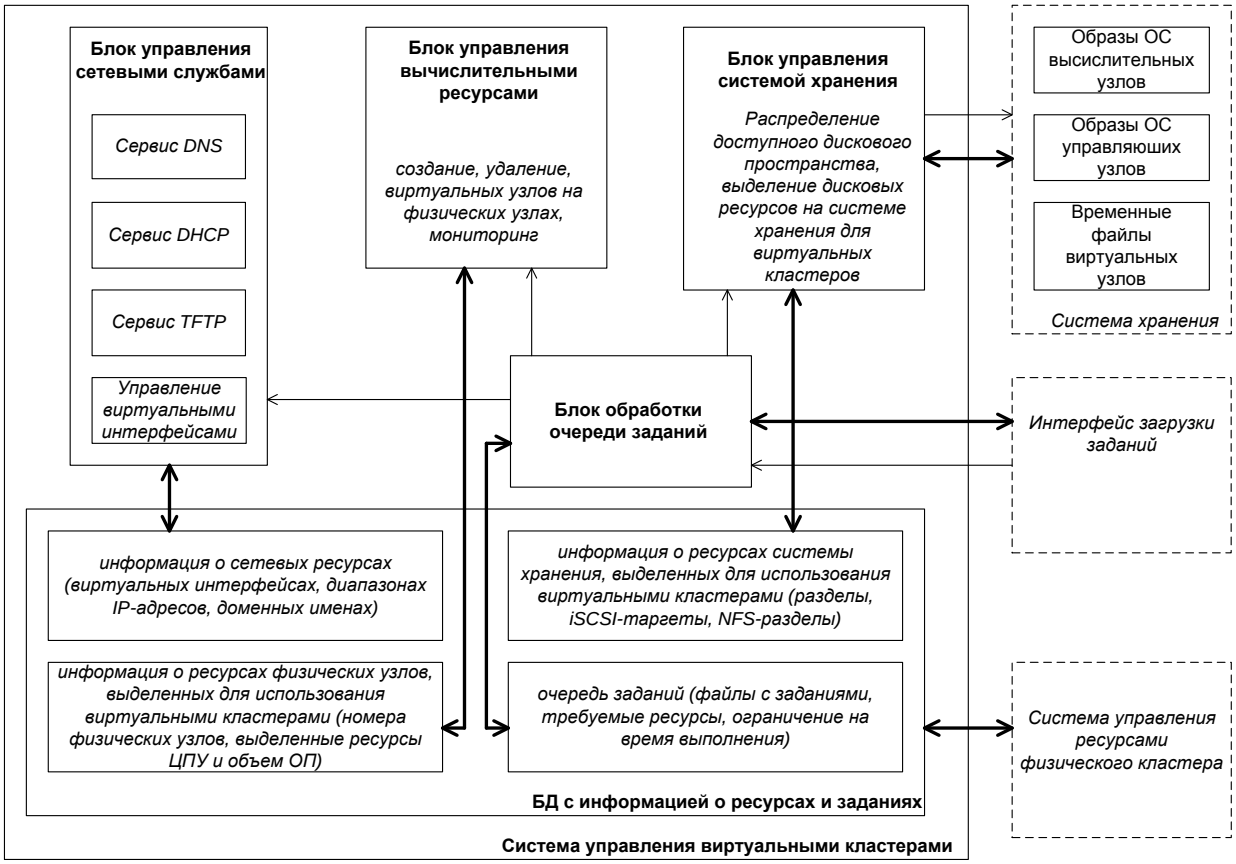


Рис.2 Система управления виртуальными кластерами

Создание/удаление виртуальных кластеров происходит таким образом (рис 3).

Через интерфейс пользователя в блок обработки очереди заданий поступает задание, состоящее из образа виртуального управляющего и вычислительного узлов с предустановленным прикладным программным обеспечением, которое необходимо для решения задачи пользователя, и дополнительных параметров. В дополнительных параметрах задания могут быть указаны такие как необходимый размер оперативной памяти, производительность процессора, количество узлов, требуемое дисковое пространство для хранения временных файлов, требования к сети обмена данными между узлами и т.д. Образы ОС виртуальных узлов передаются на систему хранения и сохраняются там. В базе данных сохраняются ссылки на месторасположение образов ОС узлов в системе хранения с привязкой к данному заданию. При наличии свободных ресурсов в соответствии с дисциплиной обслуживания выбирается задание из очереди. После этого выполняется выделение ресурсов, и загрузка виртуальных узлов на виртуальных машинах в физических узлах. Образ ОС поступает с системы хранения. После загрузки выполняется автоматическая настройка за-

груженных виртуальных узлов, с помощью протокола DHCP задаются настройки сети, выполняется монтирование каталогов или виртуальных дисков для записи временных файлов.

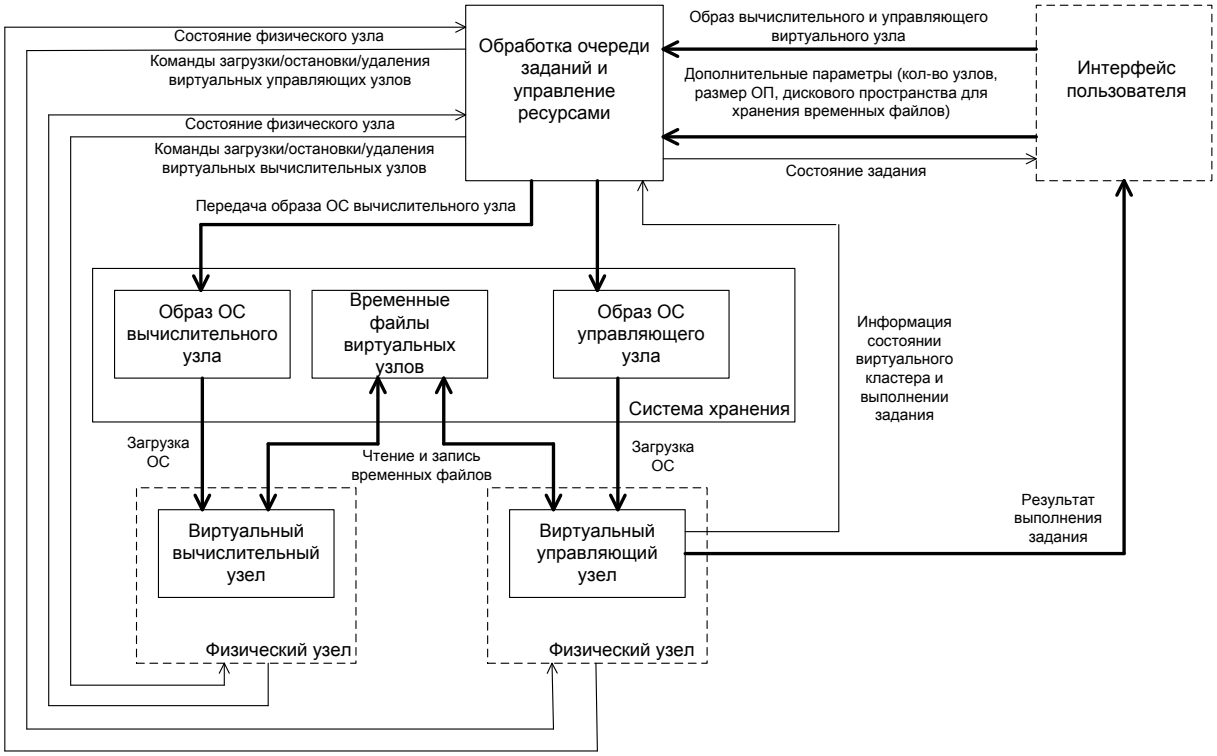


Рис.3 Схема обмена данными и управления при создании и удалении виртуальных кластеров

В процессе работы осуществляется мониторинг состояния физических узлов. При выходе из строя физического узла виртуальные узлы, работавшие на нем, переносятся на работоспособные физические узлы. Это осуществимо потому, что образы ОС и промежуточные результаты работы прикладных программ сохраняются не на локальных дисках физических узлов, а в системе хранения.

После завершения выполнения прикладных программ с помощью интерфейса пользователя подается команда удаления виртуального кластера. Это может выполняться как вручную, так и автоматически при передаче результатов выполнения прикладной программы. При удалении виртуального кластера выполняется размонтирование подключенных дисковых ресурсов, удаление виртуальных сетевых интерфейсов, остановка соответствующих виртуальных машин на физических узлах, и в базе данных о ресурсах занятые ресурсы помечаются как свободные.

Список литературы

1. Keahey, K., T. Freeman, J. Lauret, D. Olson. Virtual Workspaces for Scientific Applications, SciDAC 2007 Conference, Boston, MA. June 2007.
2. NAKADA, H., YOKOI, T., EBARA, T., TANIMURA, Y., OGAWA, H., AND SEKIGUCHI, S. The design and implementation of a virtual cluster management system. In Proceedings of the first IEEE/IFIP International Workshop on End-to-end Virtualization and Grid Management, 2007.
3. Foster, I., T. Freeman, K. Keahey, D. Scheftner, B. Sotomayor, X. Zhang. Virtual Clusters for Grid Communities, CCGRID 2006, Singapore. May 2006.
4. Overhead Matters: A Model for Virtual Resource Management, Sotomayor, B., K. Keahey, I. Foster. VTDC 2006, Tampa, FL. November 2006.
5. Enabling Cost-Effective Resource Leases with Virtual Machines, Sotomayor, B., K. Keahey, I. Foster, T. Freeman. HPDC 2007 Hot Topics session, Monterey Bay, CA. June 2007
6. Combining Batch Execution and Leasing Using Virtual Machines, Sotomayor, B., K. Keahey, I. Foster. HPDC 2008, Boston. June 2008.

7. Jones T. An overview of virtualization methods, architectures, and implementations, IBM, 2006, http://www.ibm.com/developerworks/linux/library/l-linuxvirt.

УДК 004.22

БАНЯ Е.Н.,
СЕЛИВАНОВ В.Л.

ОБ ОСОБЕННОСТЯХ ПОСТРОЕНИЯ СИСТЕМ СЧИСЛЕНИЯ РАЗНЫХ КЛАССОВ

Класс систем счисления целесообразно определять порядком вычисления символов. В системах счисления класса А, к которым относятся позиционные системы, порядок вычисления символов зависим и последовательный. В системах класса В, к которым относятся системы счисления остаточных классов, порядок вычисления символов параллелен и независим. Дана классификация систем счисления.

The class of scales of notation should be defined by the order of symbol's calculation. In A class scales of notation, to which positional scales of notation referred to, the order of symbol's calculation is dependent and successive. In scales of notation of B class, to which scales of notation of vestigial classes referred to, the order of symbol's calculation is parallel and dependent. The classification of scale of notation is given.

История развития вычислительной техники непрерывно связана с разработокой и внедрением все более новых принципов представления и кодирования числовой информации. Количество систем счисления, которые используются в цифровой вычислительной технике, непрерывно растет. На ряду с двоичной и двоично-десятичными системами счисления известны примеры применениия троичной, двоично-пятеричной, систем счисления, систем счисления с отрицательным, комплексным, иррациональными основаниями, системы счисления остаточных классов. В настоящее время нет единой классификации систем счисления, четкого разделения на классы. В научно-технической литературе встречаются противоречивые и неоднозначные трактовки базовых характеристик и параметров систем счисления, что значительно усложняет разработку теории проектирования преобразователей форм представления информации. Цель данной работы – устранение противоречий, имеющихся неточностей в определениях и терминологии, разработке классификация систем счисления.

Построение произвольной системы счисления следует начать с выбора базовых параметров, характеризующих данную систему, и разработки определенных правил, позволяющих представить любую величину в этой системе.

Под системой счисления будем понимать такой способ изображения множества чисел с помощью ограниченного набора символов, составляющих ее алфавит, при котором эти символы (элементы алфавита) располагаются в установленном порядке, занимая определенные места (позиции).

В качестве базовых параметров произвольной систем счисления выбирают:

1. Максимальную длину последовательности (общее количество позиций (разрядов)) - $(n+1)$
2. Номер позиции (разряда) – $i\ (i=\overline{0,n})$
3. Допустимое для данной позиции количество символов l_i
4. Возможные значения символа в i – позиции $a_i^j[\,j=\overline{1,l_i}\,]$
5. Набор символов в i-позиции - $A_i \in [a_i^1, a_i^2, ..., a_i^{l_i}]$
6. Полный набор символов, применяемых в системе счисления составляет алфавит систем счисления – А.
7. Значение (мера) i-й позиции (количественное значение каждой единицы i – позиции или количественный эквивалент i – позиции) - Q_i
8. Базис системы счисления (набор выбранных мер) - $B \in [Q_0, Q_1, ..., Q_n, Q_{n+1}]$
9. Диапазон представления чисел - Q
10. Основание, характеризующее i – позицию - p_i
11. Набор оснований $p \in [p_0, p_1, ..., p_n]$

В произвольной системе счисления целое положительное число N изображается последовательностью символов

$$[N] = a_n^j a_{n-1}^j ... a_2^j a_1^j a_0^j,$$

где

$[N]$ - представление числа в этой системе счисления.

Такая запись означает, что величина числа может быть определена по формуле:

$$[N] \equiv \sum_{i=0}^{n} a_i^j Q_i \bmod Q$$

Следовательно, в любой системе счисления величина числа зависит как от значений символов a_i^j, так и от количественных мер каждой позиции Q_i.

Порядок вычисления символов a_i^j определяется выбранным классом систем счисления. Символы могут вычисляться либо последовательно во времени, начиная со старшего a_n^j или с младшего символа a_0^j, либо одновременно(параллельно) и независимо. Следует заметить, что такие базовые параметры, как базис B, набор оснований $p \in [p_0, p_1, ..., p_n]$ связана между собой с помощью определенных математических соотношений, вид которых зависит от класса системы счисления. В каждой системе счисления должны выполняться также определенные ограничения, накладываемые на выбор мер Q_i, оснований p_i, количества l_i и набора символов A_i в каждой позиции, обусловленные требованиями обеспечения однозначного и непрерывного представления величин из заданного диапазона Q. Под непрерывным представлением понимают возможность представления всех чисел диапазона Q с фиксированной дискретностью (так, на пример, для целых чисел эта дискретность равна 1).

В системах счисления, применяемых в ЭВМ алфавит А представляет собой конечный целочисленный набор.

Построение систем счисления класса А, которые принято называть позиционными ППС, начинают с выбора допустимого для каждой позиции количества символов l_i, формирования алфавита А и вычисления количественных мер Q_i, называемых в этом случае весами разрядов, т.е. нахождения базиса В. Таким образом, для позиционных систем счисления первичными параметрами являются базис В (набор всех мер Q_i), допустимое для каждой позиции количество символов l_i и алфавит А.

В системах счисления, в которых $Q_0 = I$ и $a_i^1 = 0$, для удовлетворения требований обеспечения непрерывности представление величин веса последующих разрядов должны выбираться, исходя из следующего условия

$$Q_i \le \sum_{s=0}^{i-1} Q_s (l_s - 1) + 1 \quad \forall \mathrm{i} = \overline{0, \mathrm{n}}.$$

Выбрав веса, рассчитывают основания, характеризующие каждую позицию по формуле:

$$p_i = \frac{Q_{i+1}}{Q_i}$$

Если все количественные меры, входящие в базис, равны между собой, т.е. $|Q_i| = Q_0$, получаем непозиционную систему счисления МПС, выступающая как частный случай позиционной системы. В этих системах счисления все основания $|p_i| = 1$. В качестве примера рассмотрим непозиционную систему счисления, называемую унарной или единичной, в которой для записи числа применяется всего один символ. В этой системе все

$$Q_i = 1, p_i = 1, l_i = 1, j = 1, a_i = 1 \quad \mathrm{A} \in [1], \mathrm{Q} = \mathrm{n} + 1$$

К этому же классу относят и “римскую” систему счисления, в которой для обозначения чисел 1,5,10,50,100,500,1000 используются заглавные буквы I,V,X,..,C,D,M. В “римской” системе $Q_i = 1$ при $\mathrm{a}_\mathrm{i}^\mathrm{j} \ge \mathrm{a}_{\mathrm{i-1}}^\mathrm{j}$ и $Q_i = -1$ при $\mathrm{a}_\mathrm{i}^\mathrm{j} \lhd \mathrm{a}_{\mathrm{i-1}}^\mathrm{j}$ т.е. $p_i = \pm 1 \quad \mathrm{A}_\mathrm{i} \in [I, V, X, L, C, D, M]$

Особенностью позиционных систем состоит в том, что в них базис обязательно включает не все равные между собой количественные меры (веса). Систему счисления, в которой вес каждого следующего разряда не меньше, чем веса всех предыдущих разрядов, будем называть упорядоченной. Очевидно, что в упорядоченных системах счисления обязательно все основания $p_i \ge 1$.

Позиционная система счисления, в которой основания всех разрядов оказались одинаковыми т.е. $p_i = p$ для всех $i = \overline{o, n}$ называется однородной.

В однородной системе счисления, имеющей $Q_o = I$, вес i – разряда вычисляется по формуле

$$Q_i = p_i.$$

В связи с тем, что в однородной системе используется только одно основание, удобно такую систему называть по значению её основания. Например, систему счисления с осно-

ванием $p = 2$ - двоичной, с основанием $p = 3$ - троичной и т.д. Если в позиционной системе счисления веса выбраны таким образом, что не все основания оказались одинаковым, получают систему счисления которую принято называть неоднородной. В неоднородной системе счисления вес *i*-разряда связан с основаниями всех предыдущих разрядов следующим соотношением:

$$Q_i = Q_0 \prod_{k=0}^{i-1} p_k$$

К неоднородным системам счисления, имеющим целочисленные основания, можно отнести систему измерения времени , систему счисления, в которой в качестве основания выбран набор взаимно простых чисел.

В позиционных системах как однородных, так и неоднородной, могут использоваться не только целые, но дробные и иррациональные основания. Так, например, известны примеры использования однородных систем счисления с дробными основаниями, равными $p = 1 + 2^{1-\gamma}$, где γ -может выбраться, равным 2,3 и т.д., иррациональным основанием $p = \sqrt[\gamma]{2}$ для γ=2,3,…; с основанием р, которое равно числу “золотой” S-пропорции, определяемой выражением

$$p = \lim_{n \to \infty} \frac{\varphi_s(n)}{\varphi_s(n-1)}$$,где $\varphi_s(n)$ - n -ое S-число Фибоначчи.

В так называемой факториальной системе счисления, в которой веса $Q_i = (i+1)!$, все основания представляют собой также целые числа, так как

$$p_i = \frac{(i+2)!}{(i+1)!} = (i+2) \text{ для всех } i = \overline{0,n}$$

Можно привести большое число примеров неоднороднных систем счисления с иррациональным основанием. Например, система счисления, в которой веса представляют собой ряд последовательных натуральных чисел, т.е. $Q_i = i + 1$, имеет основания, равные

$$p_i = \frac{i+2}{i+1}.$$

Система счисления, в которой вес младшего разряда $Q_i = 1, p_0 = 2$, а веса остальных разрядов представляемых собой четные числа, имеет основания $p_i = 1 + \frac{1}{i}$ для $i = \overline{1,n}$.

Система счисления, в которой веса Q_i равны числам Фибоначчи, т.е

$$Q_i = \varphi_s(i) = \begin{cases} 0, i < 0, \\ 1, i = 0, \\ \varphi_s(i-1) + \varphi_s(i-s-1), i > 0 \end{cases}$$ где

S=1,2,3,…

Тогда $p_i = \dfrac{\varphi_s(i+1)}{\varphi_s(i)}$

Если выбрано S=1, веса разрядов такой неоднородной системы оказываются равными 1,2,3,5,8,13, и т.д., если выбрано S=2, веса разрядов соответственно равны 1,1,2,3,4,6,9,13 и т.д.

Если базис позиционной системы счисления найден, можно определить в каком количестве присутствует каждая из выбранных мер Q_i, начиная с наибольшего из весов Q_{n+1}. В результате первого деления получаем частное Q_n^j и остаток r_n:

$$N = a_n^j Q_{n+1} + r_n \text{, где } 0 \le r_n \le Q_{n+1}.$$

Затем делим остаток r_n на следующий вес Q_n

$$r_n = a_{n+1}^j Q_n + r_{n-1} \text{, где } 0 \le r_{n-1} \le Q_n$$

или $N = a_n^j Q_{n+1} + a_{n-1}^j Q_n + r_{n-1}$

Далее остаток r_{n-1} делим на Q_{n-1} и т.д. Процесс деления продолжаем до тех пор, пока не будет найден последний остаток $r_o < Q_1$. В результате такого деления получили представление числа N в виде последовательности символов a_i^j , начиная со старшего символа a_n^j. Нетрудно убедиться в том, что выполняя последовательное деление исходного числа N получающихся в результате деления частных на основание p_i , начиная с основания p_0 , можно определить символы a_i^j , начиная с младшего символа a_0^j.

Заметим, что в каждом разряде количество целочисленных значений l_i зависит от основания данного разряда. Если все p_i - целые числа и количество символов $l_i = p_i$, такая система счисления является однозначной (не избыточной), а при $a_i^1 = 0$ ее называют натуральной.

Если $a_i^1 \ne 0$, при построении системы счисления необходимо оговаривать, какие

конкретно символы выбраны для изображения чисел. Системы счисления могут иметь не только все положительные цифры, но и все отрицательные. Система счисления с нечетным натуральным основанием $p = 2l + 1$ и цифрами $a_i \in [-l, -l+1, ..., -1, 0, 1, .., l-1, l]$ называют симметричными. Такие системы счисления позволяют представить любое целое число как положительное, так и отрицательное. Примером симметричной системы счисления может служить троичная система с цифрами [-1,0,1]. Если p_i - целые, а количество символов l_i выбрано бо́льшим, чем p_i т.е. $l_i > p_i$, система счисления будет неоднозначной. В таких системах счисления одна и та же величина может быть представлена различными последовательностями символов. Например двоичная система счисления с набором символов (-1,0,1). Для дробных и иррациональных значений p_i допустимое количество символов l_i выбирается с округлением в большую сторону, т.е $l_i \geq]p_i[$ и так как при этом всегда оказывается что $l_i > p_i$, такие системы счисления будут всегда неоднозначными.

Если для двух однородных чисел счисления с основанием p_1 и p_2 справедливо соотношение $p_1 = p_2^m$, будем называть такие счисления родственными. Так, например, родственными будут двоичная система с четверичной, восьмеричной, шестнадцатеричной, четверичная с шестнадцатеричной, а система счисления с основаниями 2 и $\sqrt{2}$,основаниями $i\sqrt{2}$ и -2, с основаниями 2i и 4, с основаниями -2 и -8 и т.п.

Для неоднородных систем счисления также существует понятие родственных систем при следующем условии, если

$$p_{i,1} = \prod_{j=S1}^{s2} p_{j,2} \text{ для } i = \overline{0,n}$$

$$S_1 = \begin{cases} 0, i = 0 \\ \sum_{j=0}^{i-1} (n_s + 1), i = \overline{1,n} \end{cases}$$

$$S_2 = \sum_{j=0}^{i} (n_s + 1) = 1; j = \sum_{s=0}^{i-1} (n_s + 1) + k, i = \overline{1,n}, k = \overline{0,n}$$

Следует отметить, что $S_2 - S_1 + 1 = n_i + 1$, где $p_{i,1} - i$ - основание первой системы счисления

$p_{j,1} - j$ - основание второй системы счисления.

Если в какой-то системе счисления ее символы представляются с помощью цифр другой системы счисления, то такую систему называют системой с кодированным представлением ее цифр. Десятичная система счисления, в которой каждая десятичная цифра представляется тетрадой из двоичных цифр называется двоично-кодированной. Например, так называемая двоично-десятичная система счисления 8421 представляет собой неоднородную систему счисления, у которой используются основания и веса:

$p_{0,2} = 2; p_{1,2} = 2; p_{2,2} = 2; p_{3,2} = 1{,}25; p_{4,2} = 2; p_{5,2} = 2; p_{6,2} = 2; p_{7,2} = 1{,}25$ и т.д.

$Q_{0,2} = 1; Q_{1,2} = 2; Q_{2,2} = 4; Q_{3,2} = 8; Q_{4,2} = 10; Q_{5,2} = 20;$ и т.д.

Другая широко распространенная двоично-десятичная система счисления 2421 представляет собой неоднородную систему счисления с основаниями 2,2,0,5,5; 2,2,0,5,5; и.т.д. обе эти системы счисления являются родственными, так как, по принятому определению:

$$\prod_{j=0}^{3} p_{j,2} = p_1 = 10$$

Построение систем счисления класса В, куда входит система счисления остаточных классов СОК, начинают с выбора основания p_i, которые обязательно являеются целыми, причем, с целью обеспечения однозначности и непрерывности представления набор оснований должен включать p_i, представляющее собой взаимно простые числа.

Выбрав основания, можно рассчитывать величины Q_i:

$$Q_i = m_i \frac{p}{p_i} = m_i \hat{p}_i \text{ ,где } m_i = 1,2..p_i \text{ где}$$

$$|Q_i|_{p_i} \equiv 1, |Q_i|_{p_j} \equiv 0, \forall i \neq j$$

Q_i -ортогональные базисы

т.е. определить ее базис В.

Произвольное число X определяется набором остатков ($x_0, x_1, ..., x_n$) где $X \equiv x_1 \bmod p_1$ или $X \equiv |x_1|_{p1}$, $X \equiv x_2 \bmod p_2$, $X \equiv x_n \bmod p_n$. Процесс вычисления символов осуществляется независимо и параллельно по каждому основанию p_i.

Таким образом, особенность систем счисления остаточных классов состоит в том, что в них первичными параметрами являются целые, взаимно простые основания. При этом Китайская теорема об остатках гарантирует однозначное представление чисел X в диапазоне[0..P-1], где

$$P = \prod_{i=0}^{n} p_i \ .$$

Формула перевода из СОК в десятичную систему счисления имеет вид

$$X \equiv | x_1 Q_1 + x_2 Q_2 + ... + x_n Q_n |_P \ .$$

В литературе утверждается, что СОК является непозиционной символической невзвещенной системой счисления [Л.3]. Пример перевода чисел из СОК в десятичную систему счисления с помощью ортогональных базисов, полностью отрицает это утверждение. Эта система система счисления является представителем совсем другого класса В систем счисления, в которых, еще раз подчеркнем, процесс нахождения символов, в отличии от систем класса А, осуществляется паралельно и независимо.

Двоично-десятичная система счисления получившая название 8421+3, представляет собой смещенную на 3 единицы классическую двоично-десятичную систему 8421. Недопустимо относить ее к “незвешенным” системам счисления, в которых веса отдельных разрядов имееют переменные значения.

Если позиционная система счисления имеет p>>2 (например $p = 2^{16}$) то любой ее символ можно представить с помощью СОК (на пример для $p = 2^{16}$ можно выбрать модули $p_1 = 13, p_2 = 16, p_3 = 17, p_4 = 19$ тогда P=67184> 2^{16})для кодировки этих модулей достаточно иметь 18 двоичных разрядов, что не намного больше 16. Целесообразность такого комбинированного представления объясняется возможностью реализации сумматора, в котором перенос будет распространяться не более чем на 5 разрядов. В результате приведенных выше рассуждений, приведем следующую классификацию систем счисления (Рис. 1).

Выводы

1. Предложено системы счисления разделить на два класса. Класс систем счисления определяется порядком вычисления символов. В системах счисления класса А, к которым относятся позиционные системы, порядок вычисления символов зависим и последовательный. В системах класса В, к которым относятся системы счисления остаточных классов, порядок вычисления символов параллелен и независим.
2. Показано, что к системам класса А относятся непозиционные системы счисления, представляющие собой частный случай позиционных систем счисления.
3. Четко сформулировано понятие неоднозначности систем счисления. Показано, что системы счисления с дробными и иррациональными основаниями всегда неоднозначны.
4. Введено понятие родственных систем счисления, показано, что перевод из одной родственной системы счисления в другую осуществляется перекодировкой символов.

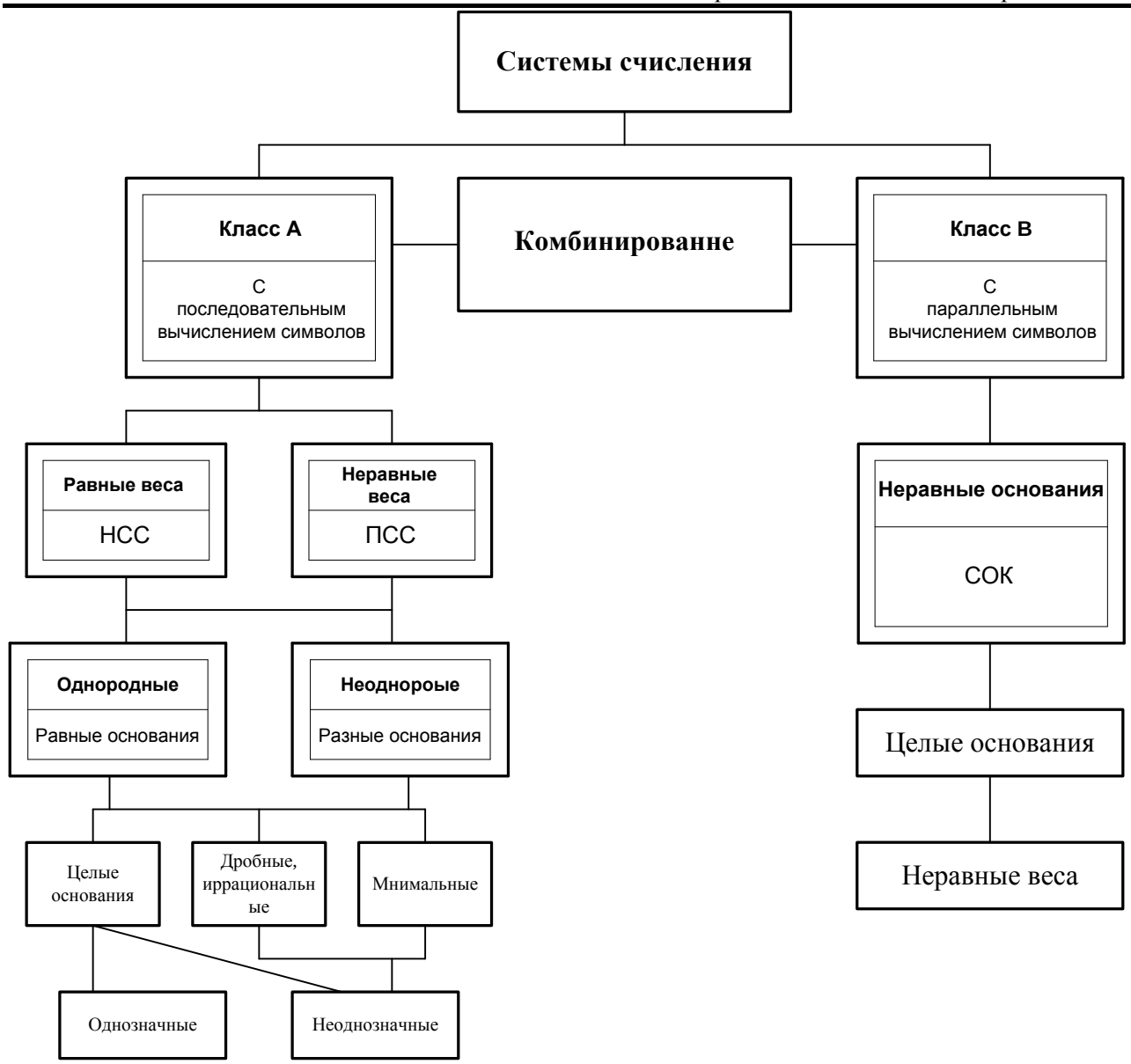


Рис.1.

Список литературы

1. В.Н. Касаткин “Новое о системах счисления” “Вища школа” К.82-96 стр.
2. Фомин С.В. “Система счисления” М.Наука 1987 стр.48
3. Энциклопедия кибернетики. К.Главная редакция УСЭ, 1974г.
4. Лысиков Б.Т. “Арифметические и логические основы цифровых автоматов”. Минск В.Ш. 1980г. 336стр.
5. Стахов А.Н. “Введение в алгоритмическую теорию измерений”. 1977г., 287 стр. “Сов.радио”.М.
6. Стахов А.П. “Коды золотой пропорции” М.,1984г
7. Поспелов Д.А “Арифметические основы вычислительной машины дискретного действия” “Высшая школа”. М.,1970г.,308 стр.
8. Шауман А.М. “Основы машинной арифметики”. Изд. Ленинград. Университет Л., 1978г
9. В.И.Корнейчук, В.П.Тарасенко.”Основы компьютерной арифметики. К.-2002. 176стр”
10. Е.Н.Баня, В.И.Селиванов “Кодирование знакопеременой информации в цифровых устройствах” Кибернетика №3 1987г. Стр 81-86.

УДК 681.322

СЕРГІЄНКО А.М.

ПРОЕКТУВАННЯ ОБЧИСЛЮВАЧІВ З РЕГІСТРОВИМИ ЗАТРИМКАМИ

Пропонується методика проектування конвейерних обчислювачів, зконфігурованих в ПЛІС, яка забезпечує мінімізацію апаратурних витрат за рахунок широкого застосування регістрових затримок. Показана дієвість методики на прикладі проектування блоку тасування даних.

A method of designing pipelined datapaths which are configured in FPGA is proposed. The method provides the hardware minimization due to the wide utilization of the shift register components. The method is proven at the example of the zigzag scan reordering buffer design.

Програмовані логічні інтегральні схеми (ПЛІС) представляють собою незамінну елементну базу в таких галузях, як цифрова обробка сигналів, телекоммунікації, завдяки можливості організації в них швидкістного виконання алгоритмів в конвейерному режимі. Таке виконання підтримується архітектурою ПЛІС, яка включає велику множину конвейерних блоків множення з додаванням, пам'яті. Але розробка обчислювачів на базі ПЛІС залишається трудомістською, висококваліфікованою роботою. В статті пропонується методика проектування конвейерних обчислювачів, які в повній мірі використовують можливості апаратури сучасних ПЛІС.

В ПЛІС Virtex фірми Xilinx серед бібліотечних компонентів є елемент SRL16, що представляє собою 16-розрядний регістр зсуву з одним входом і одним виходом, який підключається до одного з тригерів цього регістру через багатовходовий мультиплексор, що керується чотирьохрозрядною шиною адреси. Цей елемент є регістровим буфером з регульованою затримкою або просто регістровою затримкою (РЗ) і призначений саме для організації конвейерних обчислень складних потокових алгоритмів. Ресурси РЗ досить великі – кожну другу логічну таблицю ПЛІС можна зконфігурувати як РЗ SRL16. Причому в критерії ефективності складність одного елемента SRL16 оцінюється як складність одного – двох тригерів. Крім того, одна РЗ може замінити собою до 16 регістрів, а також зекономити відповідні програмовані лінії зв′язку. Цим самим можна не тільки зменшити апаратні витрати, але і збільшити швидкодію обчислювача, зменшивши кількість ліній з′єднання з великою затримкою. Тому має сенс розробити методику еффективного використання ресурсів РЗ.

Загальний підхід до розробки функціональної схеми на рівні регістрових передач полягає в наступному. Вибирається множина ресурсів (суматорів, блоків множення, пам′яті і т.і.), складається розклад виконання операцій алгоритму і виконується призначення операцій на ресурси. За цим знаходять множину необхідних регістрів і мережу комутації ресурсів. Такий підхід також застосувють для проектування схеми з РЗ. При цьому ланцюжки регістрів, побудованих в схемі, заміняються на відповідні РЗ. Але ланцюжки регістрів в такій схемі виникають випадково і тому їх виявляється значно менше, ніж можливо і таким чином, РЗ в схемі задіяні нефективно. Крім того, властивість затримки РЗ, яка змінюється динамічно, не використовується.

В роботах [1–3] пропонується метод проектування конвейерних обчислювачів шляхом відображення графа синхронних потоків даних (ГСПД), який представлено в просторі ресурси – час у вигляді конфігурації алгоритму (КА). Метод дає змогу одночасно як складати розклад, мінімізувати кількість процесорних елементів (ПЕ), так і шукати ефективну систему з′єднань між цими ПЕ. Тут під ПЕ розуміється елементарний обчислювач з пам′яттю або без неї, наприклад, суматор, мультиплексор з регістром, РЗ, тощо. Тому має сенс створити методику розробки пристроїв з РЗ на основі цього методу.

На першому етапі синтезу за вказаним методом вершини-оператори однорідного ГСПД разом з дугами розташовуються в трьохвимірному просторі як множини векторів $\boldsymbol{K}_i$ та $\boldsymbol{D}_j$ з урахуванням умов, приведених в [2,3]. При цьому координати вектора $\boldsymbol{K}_i = (s,q,t)^T$ означають номер s ПЕ, де виконується оператор, тип q ПЕ і часову складову t, яка дорівнює номеру такту в періоді виконання алгоритму. Вектори

$\boldsymbol{K}_i$ з однаковою часовою складовою формують один ярус і тому виконуються одночасно. Часова складова $R(\boldsymbol{D}_j)$ вектора $\boldsymbol{D}_j = \boldsymbol{K}_i - \boldsymbol{K}_l$ дорівнює затримці між виконаннями операторів, вершини $\boldsymbol{K}_i$, $\boldsymbol{K}_l$ яких є суміжними. Виконується мінімізація числа ПЕ шляхом виконання вимог $|\boldsymbol{K}_{s,q}| \to L$, тобто число вершин, що відображаються в s-й ПЕ, прямує до L, де L – період виконання алгоритму, тактів. Крім того, при формуванні КА бажано будувати досконалий кістяк ГСПД, як це пропонується в [4].

На другому етапі виконується урівноважування КА, яке полягає в додаванні в дуги графа вершин затримки, поки часові складові усіх векторів $\boldsymbol{D}_j$ не дорівнюватимуть 0 або 1. Після цього виконується оптимізація КА шляхом взаємних перестановок векторів-вершин з одного ярусу з метою мінімізації числа регістрів та числа входів мультиплексорів в результуючій структурі і/або з застосуванням інших стратегій, наприклад, ресинхронізації [1]. Також мінімізується число регістрів склеюванням вершин затримки з одного яруса, які зберігають один і той самий операнд.

На третьому етапі одержана оптимізована КА відображається в граф структури обчислювача шляхом склеювання векторів-вершин з однаковими координатами s,q. КА перетворюється в розклад виконання операторів, використовуючи ту властивість, що часова складова вектору $\boldsymbol{K}_i$ дорівнює моменту виконання оператора безвідносно номера періоду виконання. При цьому можна не будувати структуру і розклад, якщо зразу описати схему обчислювача на мові VHDL [3].

Розглянемо деякий підграф КА, який виконується в РЗ. Цей підграф виконує пересилку операнда x з джерела $\boldsymbol{K}_{i1}$ до споживачів $\boldsymbol{K}_{j1}$, $\boldsymbol{K}_{l1}$ через дуги $\boldsymbol{D}_{j1} = \boldsymbol{K}_{j1} - \boldsymbol{K}_{i1}$, $\boldsymbol{D}_{l1} = \boldsymbol{K}_{l1} - \boldsymbol{K}_{i1}$, а також операнда y з джерела $\boldsymbol{K}_{i2}$ до споживачів $\boldsymbol{K}_{j2}$, $\boldsymbol{K}_{l2}$ через дуги $\boldsymbol{D}_{j2} = \boldsymbol{K}_{j2} - \boldsymbol{K}_{i2}$, $\boldsymbol{D}_{l2} = \boldsymbol{K}_{l2} - \boldsymbol{K}_{i2}$, відповідно (рис.1,а). Для того, щоб підграф виконувався на РЗ, принаймні всі його вхідні вершини повинні мати однакові просторові координати p.

При виконанні алгоритму в РЗ операнди x і y в кожному такті пересилаються на наступний регістр РЗ. Це еквівалентно тому, що в урівноваженій КА ці операнди в кожному такті передаються в наступний ярус і наступний ряд вершин затримки, який відображається в регістр РЗ. Іншими словами, ланцюжки суміжних вершин затримки $\boldsymbol{K}_{Di}$ при рівномірно зростаючих координатах $R(\boldsymbol{K}_{Di})$ розміщуються вздовж паралельних прямих, розташованих під одним кутом до осі ot. Пройшовши ланцюжки з $R(\boldsymbol{D}_{j1})$, $R(\boldsymbol{D}_{l1})$, $R(\boldsymbol{D}_{j2})$, $R(\boldsymbol{D}_{l2})$ вершин затримки, операнди x і y видаються на відповідні вихідні вершини підграфу (рис.1,б). Результуюча схема РЗ показана на рис.1,в.

РЗ має єдиний вихід, тому на підграф КА накладається додаткове обмеження – з вершин затримки, які належать одному ярусу і які відображаються в РЗ, повинно виходити не більше ніж одна дуга, що веде у вихідну вершину. При цій умові вихідний мультиплексор РЗ одночасно підключатиметься тільки до одного регістра РЗ. В іншому разі РЗ повинен мати більше, ніж один вихід або вихідний мультиплексор, як показано пунктиром на рис.1,б,в. Такий підграф повинен бути реалізований на кількох РЗ, як на рис.1,г.

РЗ типу SRL16 має додатковий вхід дозволу синхросерії, керування яким дає змогу загальмувати просування операндів по регістрах РЗ. При використанні цього входу можна зекономити кількість РЗ, якщо величина $R(\boldsymbol{D}_j)$ більше кількості регістрів в РЗ. На рис.2 показано приклад перетворення КА на рис.1,б з метою додаткової затримки на такт операндів, які поступають в вершини $\boldsymbol{K}_{l1}$, $\boldsymbol{K}_{l2}$. Така затримка відповідає векторам $\boldsymbol{D}_j$, які розміщуються паралельно осі ot.

Якщо вершини-джерела операндів мають різні просторові координати s, то на вході РЗ одержується вхідний мультиплексор. Мінімізацію входів таких мультиплексорів можна виконувати згідно з методикою, приведеною в [5].

Таким чином, методика проектування конвейерних обчислювачів з РЗ виглядає як наступна. Начальні дані – КА, період виконання алгоритму L та інші параметри оптимізації. Методика виконується таким самим чином, як це описано в [1], за виключеннями, які описані нижче.

На першому етапі синтезу слід виділити підграфи КА, що відповідають пересилці операндів між ресурсами обчислювача з затримками у часі і/або тасуванням операндів, які передбачається відобразити в окремі РЗ.

На другому етапі треба урівноважити дуги залежностей за допомогою проміжних вершин затримки. Виконати зменшення кількості проміжних вершин затримки для всіх дуг, якщо можливо.

Розмістити вершини затримки на паралельних прямих, що знаходяться під кутом до осі часу або паралельно цій осі таким чином, щоб суміжні вершини затримки відрізнялись по часовій координаті на один такт. Виконати вимоги коректного розміщення вершин, включаючи вимогу реалізації РЗ з одним входом і виходом. У разі неможливості одержати один вхід у РЗ використовують евристику мінімізації числа входів додаткового мультиплексора на вході РЗ згідно з [5], а при неможливості одержати РЗ з одним виходом ланцюжки вершин затримки розщеплюють, щоб вони відобразжались в додаткові РЗ (див. рис.1,г).

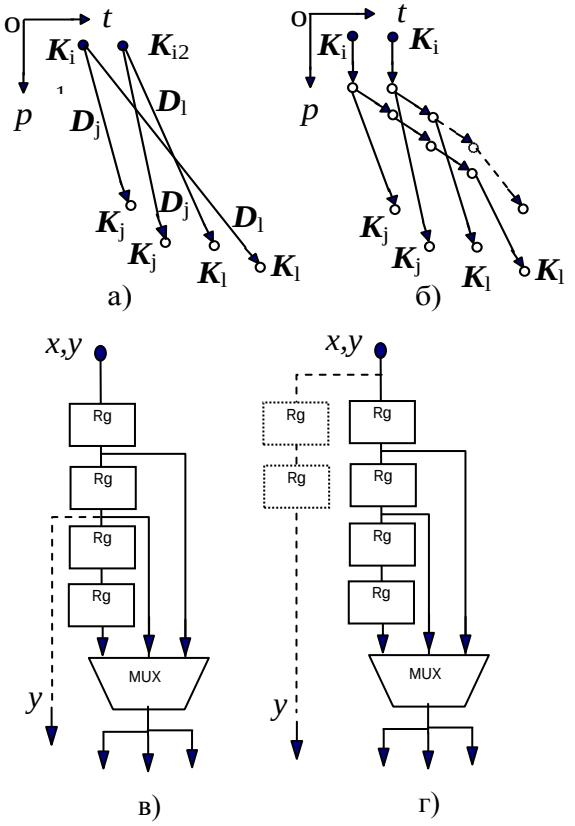


Рис.1. Відображення підграфу КА в РЗ

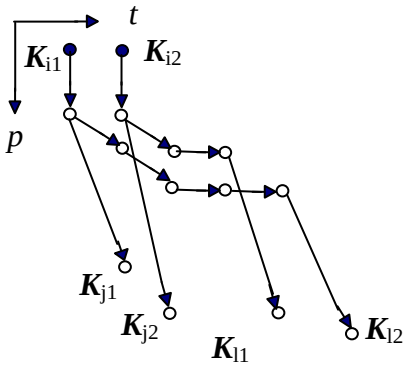


Рис.2. КА, що відповідає РЗ з входом дозволу

Дуги залежностей разом з відповідними вершинами затримки, що інцидентні вершинам-приймачам, відобразити в РЗ. При складанні алгоритму керування обчислювачем якщо в РЗ відображаються тільки дуги, які знаходяться під кутом до осі часу, то операнди в РЗ записуються в кожному такті, а якщо є дуги паралельні цій осі, то у відповідних їм тактах забороняється запис в РЗ.

На третьому етапі обчислювач, описаний на мові VHDL згідно з методикою, представленою в [1], компілюється в конфігурацію ПЛІС, яка вміщує РЗ типу SRL16, які відповідають виділеним підграфам КА.

Розглянемо приклад проектування буфера тасування даних згідно з правилом z-видного обходу, який використовується в кодерах зображення за стандартом H264 [6]. Якщо 16 вхідних даних приходять на вхід такого буфера в натуральному послідовному порядку, то вони виходять з нього в порядку згідно з послідовністю: 0,1,4,8,5,2,3,6,9,12,13,10,7,11,14,15. Як правило, такий буфер будують на основі двохпортової оперативної пам′яті, що призводить до нераціонального використання ресурсів, а також до великої латентної затримки між прийомом вхідних даних і видачею вихідних даних.

Урівноважена КА алгоритму роботи такого буфера, яка підготовлена згідно з одержаною методикою, показана на рис.3. Опис КА на мові VHDL приведено нижче.

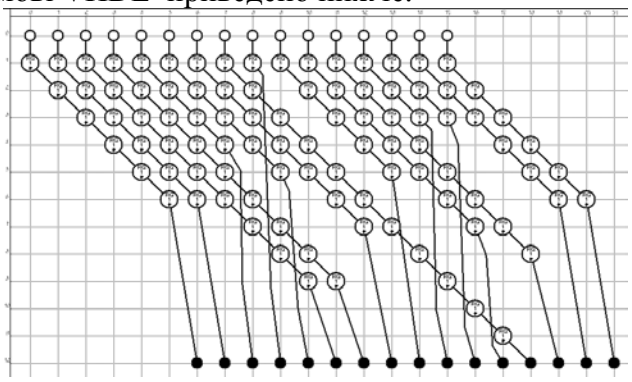


Рис.3. Урівноважена КА буфера тасування

Одержаний буфер при конфігуруванні в ПЛІС Virtex4 фірми Xilinx має мінімальні апаратні витрати – 4 тригери, 9 логічних таблиць і 12 елементів SRL16 – за кількістю розрядів даних. Цей буфер може бути використаний при рекордній тактовій частоті 900 МГц. Такі результати показують дієвість розробленої методики.

Було розроблено ряд конвейерних процесорів швидкого перетворення Фур′є з алгоритмом Винограда. Цей алгоритм характеризується складними перестановками даних і тому мінімальні апаратурні витрати процесорів досягнуті саме завдяки використанню запропонованої методики.

```
entity ZZ4x4 is
   port(CLK : in STD_LOGIC;     – синхросигнал
      START : in STD_LOGIC;     – запуск буфера
      DI : in STD_LOGIC_VECTOR(11 downto 0); – вхідні дані
      DO : out STD_LOGIC_VECTOR(11 downto 0)   ); – вихідні дані
end ZZ4x4;
architecture ZZ4x4 of ZZ4x4 is
   type TARR16 is array (0 to 15) of bit_vector(11 downto 0);
   type TA is array(0 to 15) of natural range 0 to 10;
   signal sr:TARR16;                                  – масив регістрів SRL16
   constant table :TA:=(5,5,3,0,4,8,8,6,4,2,2,6,10,7,5,5); – номери відводів SRL16
   signal fa, addr:natural range 0 to 15;
begin
   FSM:process(CLK,RST)  begin      – лічильник тактів  періоду алгоритму
      if CLK'event and CLK='1' then
         if START='1' then  addr<=0;  else addr<=(addr+1) mod 16; end if;
      end if;
   end process;
   fa<=table(addr);        – перекодування такту в номер відводу SRL16
   SRL16:process(CLK)  begin                  – опис SRL16
      if CLK'event and CLK='1' then
         sr<=DI & sr(0 to 14);              – власне зсув в РЗ
      end if;
   end process;
   DO<= sr(fa);              – видача результату з fa-го відводу  SRL16
end ZZ4x4;
```

Таким чином, запропонована методика проектування конвейерних обчислювачів, зконфігурованих в ПЛІС, яка забезпечує мінімізацію апаратурних витрат за рахунок широкого застосування регістрових затримок. Методика може бути використана для проектування спеціалізованих обчислювачів, в яких використовується буферна пам′ять типу FIFO.

Перелік посилань

1. Сергиенко А.М. VHDL для проектирования вычислительных устройств. – Киев: ДиаСофт. – 2003. – 203 с.
2. Сергієнко А.М. Синтез структур для виконання періодичних алгоритмів з операторами керування // Вістник НТУУ “КПІ”. Сер. Інформатика, управління та обчислювальна техніка. – 2007. – №47. – с.221–227.
3. 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.
4. Сергієнко А.М. Досконалий кістяк алгоритму.
5. Сергиенко А.М., Симоненко В.П. Отображение периодических алгоритмов в программируемые логические интегральные схемы //Электронное моделирование. –Т.29. –2007.–№2.–с.49–62.
6. Richardson I.E.G. H.264 and MPEG-4 Video Compression. Video Coding for Next-generation Multimedia. –Wiley. –2003. –281p.

УДК 519.854.2

ПАВЛОВ О.А.,
МІСЮРА О.Б.,
МЕЛЬНИКОВ О.В.,
ЩЕРБАТЕНКО О.В.,
МИХАЙЛОВ В.В.

ЗАГАЛЬНА СХЕМА ПЛАНУВАННЯ ТА УПРАВЛІННЯ СКЛАДНИМИ ОБ'ЄКТАМИ, ЩО МАЮТЬ МЕРЕЖНЕ ПРЕДСТАВЛЕННЯ ТЕХНОЛОГІЧНИХ ПРОЦЕСІВ Й ОБМЕЖЕНІ РЕСУРСИ

Представлена схема реалізації інформаційної технології планування та управління у системах з мережним представленням технологічних процесів та обмеженими ресурсами (МПТПОР), на основі якої створено комплекси послідовних взаємозв'язаних математичних моделей, сумісних з ієрархією рішень, що приймаються на кожному рівні планування, та системи нових високоефективних взаємозв'язаних алгоритмів розв'язання задач планування в сучасних умовах. Це вперше дозволило розв'язати задачу планування за різними критеріями оптимальності у комплексі.

The realization scheme of the informational technology for planning and management in systems with the network imagination of technological processes and the limited resources (NITPLR) is given. On the basis of the scheme the complexes of sequential interconnected mathematical models which are compatible with the hierarchy of decisions that made on every planning level, and the systems of new highly effective interconnected algorithms for the planning tasks solution in current conditions were created. This has let in the first time to solve the task of planning by different optimality criteria in complex.

Вступ

Постановкам задач планування та управління складними об'єктами та методам їх розв'язання в останні десятиліття приділяється істотна увага з боку багатьох дослідників. Найбільш добре вивчена область планування та управління складними об'єктами – виробниче планування та управління, підходи до якого та отримані результати також мають силу для інших областей – наприклад, в будівництві та в управлінні проектами. Найбільш широке поширення (до 80% усіх підприємств) одержали системи з мережним представленням технологічних процесів та обмеженими ресурсами (МПТПОР). У даній статті розглядається схема реалізації інформАційної технології планування та управління у таких системах.

Загальна постановка задачі планування

Нехай задана множина n комплексів взаємозв'язаних робіт $J = \{J_1, J_2, \ldots, J_n\}$ (комплекс робіт J_i, $i = \overline{1, n}$, надалі називається завданням). На кожній підмножині J_i заданий частковий порядок орієнтованим ациклічним графом. Часткова упорядкованість очевидним образом визначається технологією виконання комплексу робіт. Кожна наступна робота може початись тільки по завершенню попередніх робіт. Вершини графа відповідають роботам, зв'язки вказують на відносини передування. Кінцеві вершини відповідають завершенню виконання завдань. Для кожної вершини j графа відома l_j – *детермінована* тривалість виконання (інтегрований показник, що відображає виділені ресурси – матеріальні, людські, виробничі), для кожної роботи $j^i \in I$ (I – множина кінцевих вершин) задана вага $\omega_i = \omega_j^i$; для окремих завдань заданий директивний строк закінчення D_i. Величина ваги визначається потенційною складністю, важливістю і неоднозначністю (для робіт, зв'язаних з необхідністю одержання нового наукового розв'язку) виконання тих робіт, без яких у цілому завдання не може бути виконано. Для виконання робіт застосовується множина обмежених ресурсів. Сукупність ресурсів і виконавців розділена на окремі, досить автономні модулі – *мультиресурси* (мультиресурс – стійка група разом працюючих ресурсів – наприклад, бригада, група однотипного устаткування, однопрофільний підрозділ). Мультиресурси можуть знаходитися як в одній, так і в різних організаціях.

Необхідно побудувати погоджений план виконання комплексів робіт мультиресурсами

та розподіл виконання робіт по ресурсах, з урахуванням наступних критеріїв оптимальності і їх комбінацій:

а) мінімізація сумарного зваженого моменту закінчення виконання завдань (задача 1);

б) мінімізація сумарного часу виконання всіх завдань проекту (задача 2);

в) виконання завдань без порушення директивних строків D_i (планування «точно в строк», задача 3);

г) мінімізація сумарного зваженого моменту закінчення виконання завдань, якщо для деяких завдань $i \in \overline{1,n}$ не можуть бути порушені директивні строки D_i (задача 4);

д) мінімізація сумарного зваженого запізнення виконання завдань відносно директивних строків: $\omega_i \, max(0, C_i - D_i) \to min$, де C_i – момент закінчення виконання комплексу робіт J_i, $i = \overline{1,n}$ (задача 5);

е) мінімізація сумарного зваженого запізнення виконання завдань відносно директивних строків, якщо для деяких завдань $i \in \overline{1,n}$ директивні строки D_i не можуть бути порушені (задача 6); ж) мінімізація сумарного зваженого випередження та запізнення відносно директивних строків: $\omega_i \, |C_i - D_i| \to min$ (задача 7).

У задачах без директивних строків ω_i позначає вагу комплексу робіт, яка дорівнює вазі кінцевої вершини, а в задачах з директивними строками ω_i позначає штраф за відхилення від директивного строку на одиницю часу.

Назвемо *агрегованої роботою* сукупність робіт, виконуваних в одному мультиресурсі в рамках одного заходу в мультиресурс по одному завданню.

Задачі 1–7 розв'язуються при наступних обмеженнях:

тривалість виконання кожного завдання визначається його критичним шляхом;

спільні агреговані роботи різних завдань лежать на їх критичних шляхах і виконуються в одному мультиресурсі;

агрегована робота не передається в інші мультиресурси до її повного завершення.

Задачі 1–7 належать до класу *NP-складних* задач у сильному розумінні.

Трирівнева модель планування та управління складними об'єктами

В основу планування та управління складними об'єктами покладено математичне забезпечення трирівневої моделі планування та управління дрібносерійним виробництвом [1], адаптоване для планування інших класів об'єктів. У відповідності до цієї моделі, побудова розподілу робіт по ресурсах здійснюється в три етапи. Перший етап складається в побудові агрегованої моделі. Якщо які-небудь роботи виконуються в одному мультиресурсі в рамках одного заходу в мультиресурс по одному завданню, то вони агрегуються в одну агреговану роботу. Тривалість виконання агрегованої роботи визначається її критичним шляхом. Для кожного комплексу робіт визначається критичний шлях виконання агрегованих робіт. На основі агрегованої інформації будується граф на критичних шляхах завдань. Вершини отриманого графа – це агреговані роботи, дуги відображають зв'язки між мультиресурсами, що регламентують технологію виконання завдань. Деякі роботи, що належать різним завданням, вимагають виконання в спеціалізованих унікальних мультиресурсах, бажано в рамках одного заходу в мультиресурс. У цьому випадку при Виконанні деяких умов, описаних нижче, формується об'єднана агрегована робота, що на графі зв'язності відображено спільними вершинами. Для визначення пріоритетів завдань при побудові погодженого плану виконання завдань відповідно до критеріїв оптимальності, важливим є розв'язання на першому рівні задачі «Мінімізація сумарного зваженого моменту закінчення виконання завдань одним приладом при відношенні порядку, заданому орієнтованим ациклічним графом» (МЗМ) для випадку, коли вагові коефіцієнти усіх вершин графу зв'язності, крім кінцевих, дорівнюють нулю [2, 3]. У результаті розв'язання цієї задачі формується послідовність виконання агрегованих робіт, у якій підпослідовності агрегованих робіт упорядковані за пріоритетами, що визначають черговість запуску агрегованих робіт на виконання.

Другий етап полягає в побудові погодженого плану виконання завдань з урахуванням зазначених вище критеріїв оптимальності.

Отримані на першому рівні пріоритети агрегованих робіт служать додатковою інформа-

цією, що дозволяє значно підвищити ефективність отриманих розв'язків.

Процедури третього рівня дозволяють у відповідності з побудованим планом для агрегованих робіт побудувати розподіл робіт по ресурсах (так зване точне планування). На цьому рівні розв'язуються задачі оптимізації за критеріями:

а) Мінімізація сумарного зваженого моменту закінчення виконання завдань (Задача 1);

б) Мінімізація сумарного часу виконання всіх завдань (Задача 2);

в) Мінімізація сумарного зваженого запізнення виконання множини завдань із різними директивними строками (Задача 3);

г) Мінімізація сумарного запізнення виконання множини завдань із загальним директивним строком (Задача 4);

д) Мінімізація сумарного зваженого запізнення виконання множини завдань за умови, якщо для деяких завдань не можуть бути порушені директивні строки (Задача 5);

е) Мінімізація сумарного зваженого запізнення виконання множини завдань за умови, коли для частини завдань заборонені випередження й запізнення відносно директивних строків (Задача 6);

ж) Мінімізація сумарного зваженого запізнення виконання множини завдань за умови мінімізації числа переналагоджень в мультиресурсах (Задача 7);

з) Мінімізація сумарного запізнення виконання множини завдань відносно директивних строків (Задача 8);

и) Мінімізація сумарного зваженого випередження та запізнення відносно директивних строків (Задача 9).

Основні принципи реалізації розв'язання задач планування

Алгоритмічне забезпечення погодженого планування складається з трьох основних блоків, що відповідають трьом рівням загальної схеми планування. Нижче наведені базові принципи алгоритмів розв'язання задач у системі планування. Рівень І. Побудова моделі технологічної та конструкторської агрегації (рис. 1).

При побудові моделі технологічної агрегації виконується агрегація до рівня мультиресурсів і побудова агрегованих робіт. Використовуються наступні принципи агрегування інформації.

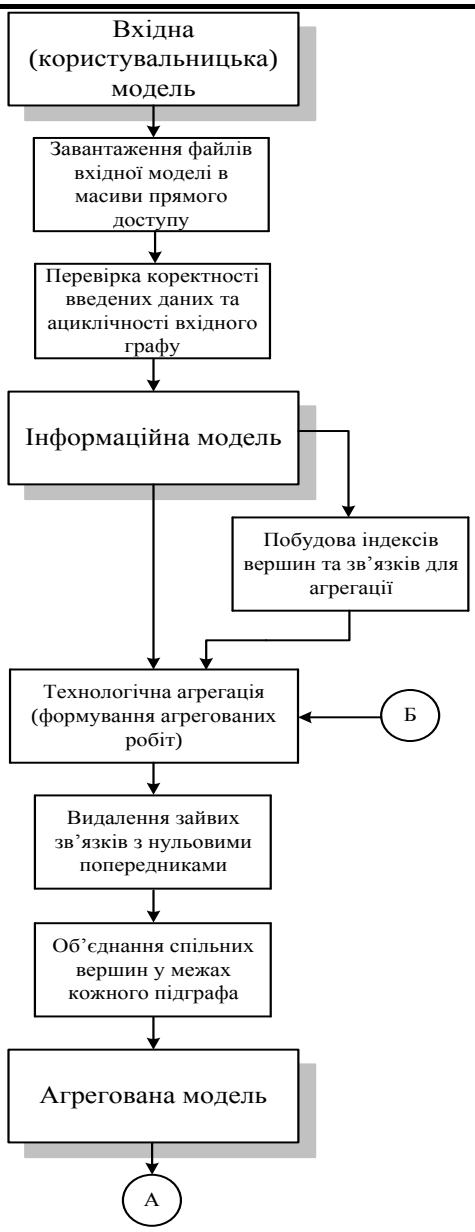


Рис. 1. Рівень І. Побудова агрегованої моделі

Вхідна модель інформації, як правило, представляє собою бази даних з інформацією про операції, взаємозв'язки між ними, групи мультиресурсів, інформацію про завдання та календар роботи (якщо інформація представлена інакше, її треба перетворити до необхідного виду). Вхідний ациклічний орієнтований граф операцій представляється у вигляді ланцюжків технологічних операцій (переліку робіт) і переліку взаємозв'язків проекту (рис. 2). Перетворення вхідного графа в граф агрегованих робіт здійснюється за допомогою процедури первинної агрегації, або агрегації агрегованих робіт. Така агрегація є агрегацією технологічного типу і робить об'єднання суміжних операцій (по технологічних ланцюжках), виконуваних в одному і тому же мультиресурсі по одному завданню, в один елемент, названий агрегованою роботою (рис. 4).

Тривалість виконання агрегованої роботи визначається критичним шляхом у графі виконання робіт, що входять до неї; час переналагодження для агрегованої роботи дорівнює максимальному з часів переналагодження її перших робіт (листів графа). При цьому тривалість роботи *в одному завданні* визначається технологією виконання роботи. Для агрегованої роботи тривалість дорівнює сумі тривалостей робіт усіх завдань у даному мультиресурсі по критичному шляху на одиницю ресурса, тобто

$$L_i = \sum_{j \in CP_i} \frac{Lроб_{ij} \cdot Nроб_i \cdot N_i}{KR_i},$$

де L_i – тривалість i-ї агрегованої роботи; $Nроб_i$ – кількість робіт у завданні (застосовність); $Lроб_{ij}$ – тривалість j-й роботи i-ї агрегованої роботи; N_i – кількість завдань, до яких входить i-а агрегована робота; KR_i – кількість ресурсів у всіх мультиресурсах групи (з розрахунком на паралельне виконання в однотипних мультиресурсах); CP_i – критичний шлях робіт агрегованої роботи;

$Lon_i = \max(\sum_{j \in CP_i} Lon_{ij}, 1)$ назвемо тривалістю *агрегованої операції*; $Non_i = \frac{Nроб_i \cdot N_i}{KR_i}$ – кількість агрегованих операцій в агрегованій роботі.

З вищенаведеної формули видно, що фактично тривалість агрегованої роботи виявляється кратною тривалості однієї агрегованої операції, що визначається як сума тривалостей робіт по критичному шляху агрегованої роботи. Агрегована операція – найменша частина агрегованої роботи, що не допускає дроблення на більш дрібні частини.

На рис. 2 штриховою лінією позначені роботи, що об’єднаються. У колі – код (назва) роботи, поруч з ним – код групи мультиресурсів (в дужках – номер заходу). Стрілки позначають входження, числа над стрілками – кількість робіт, що потребують виконання для передачі до попередника (застосовність). На рис. 4 числа зліва від кружків позначають номер агрегованої роботи після перенумерації.

Після одержання первинних агрегованих робіт методом технологічного об’єднання може здійснюватись конструкторське об’єднання агрегованих робіт, що мають однакові коди вершин і мультиресурсів в одному завданні, час запуску яких по критичному шляху завдання відрізняється не більше, ніж на величину максимального з часів їхнього переналагодження. При цьому агреговані роботи стають однією агрегованою роботою із сумарною тривалістю і часом переналагодження, рівному максимальному з цих часів переналагодження (рис. 5).

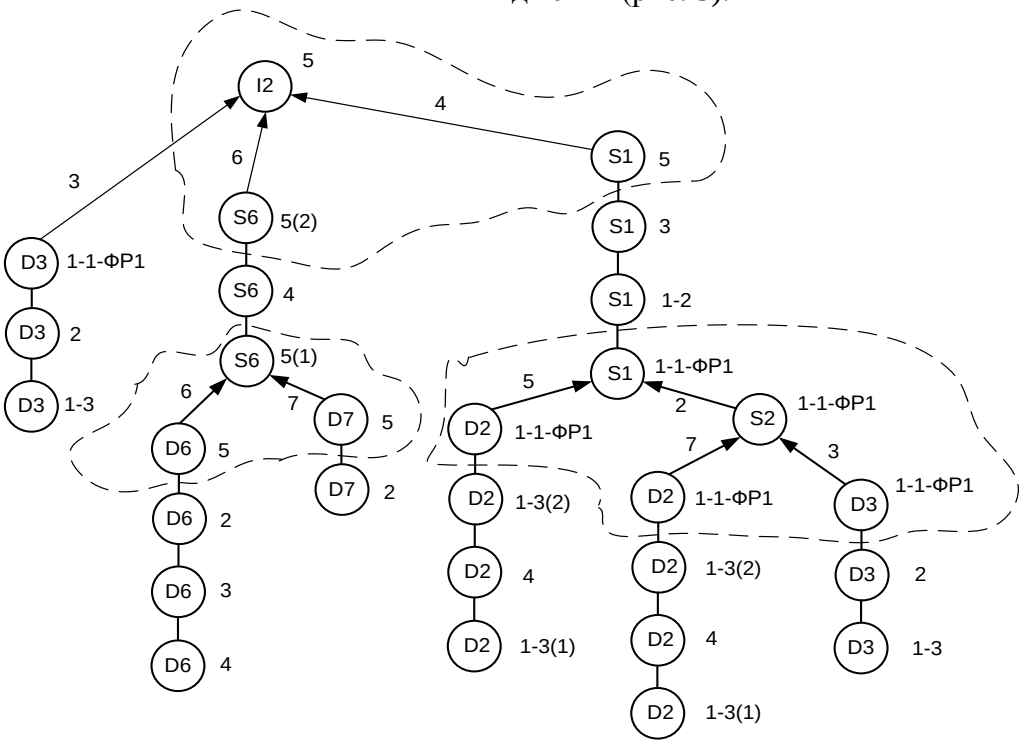


Рис. 2. Приклад вхідного графу виконання робіт та об’єднання в агреговані роботи

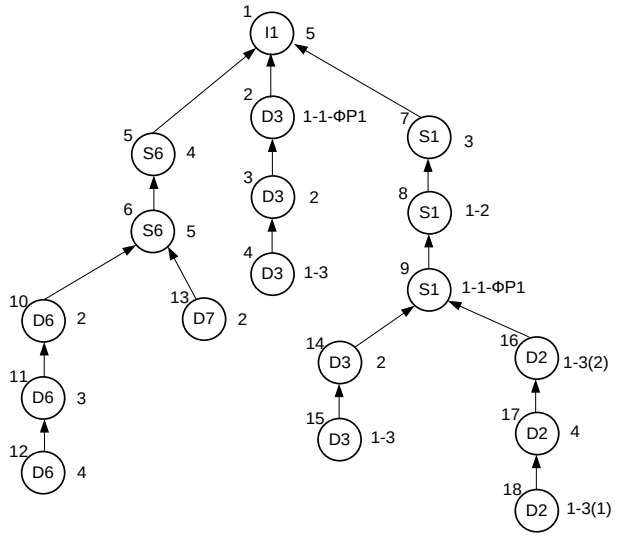


Рис. 4. Граф агрегованих робіт після технологічної агрегації

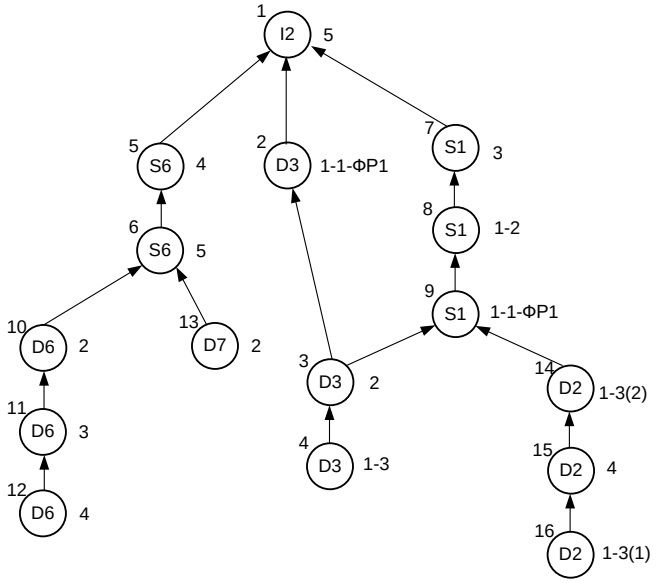


Рис. 5. Граф агрегованих робіт після конструкторської агрегації

Конструкторська агрегація здійснюється рідко, тому що вона звичайно приводить до великих довжин агрегованих робіт, що погано впливає на результат розподілу. Але об'єднання робіт у загальні вершини, що розглядається далі, також є методом конструкторської агрегації.

Рівень 2. Побудова плану виконання агрегованих робіт мультиресурсами (рис. 3, 7)

Спочатку здійснюється побудова спеціальних індексних масивів для швидкого доступу до інформації, необхідної для виконання наступних блоків. Побудована інформаційна модель називається алгоритмічною. При індексації визначаються значення окремих реквізитів моделі, що відповідають критерію, обраному користувачем.

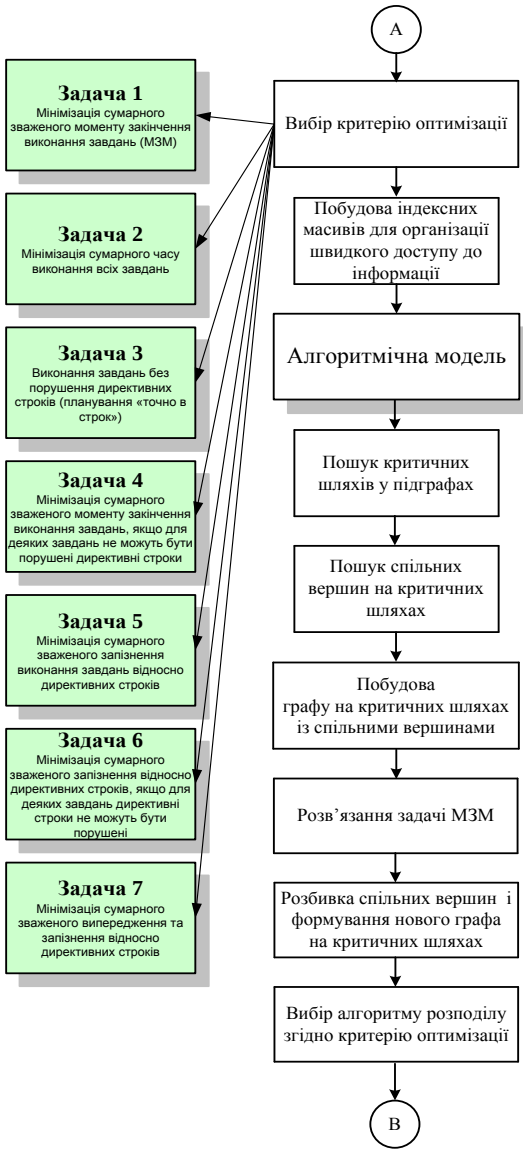


Рис. 3. Рівень II. Побудова плану виконання агрегованих робіт мультиресурсами (ч.1)

На сформованій у результаті перенумерації алгоритмічній моделі здійснюється пошук критичних шляхів завдань. Критичний шлях у графі, де кожна вершина навантажена тривалістю, – це шлях з максимальною тривалістю. Критичні шляхи шукаються за допомогою спрямованого повного перебору вершин графа агрегованих робіт. Для того, щоб був можливий пошук критичного шляху, перед його виконанням перевіряється ациклічність побудованого графа.

Після того, як знайдені критичні шляхи завдань, будується граф на критичних шляхах. Таким чином реалізується агрегування моделі до рівня «одного ресурсу». На графі на критичних шляхах розв'язується задача побудови

оптимального розкладу виконання агрегованих робіт, вважаючи, що це задача теорії розкладів для одного приладу.

Граф на критичних шляхах будується шляхом пошуку та об'єднання спільних вершин, що лежать на критичних шляхах. Ознакою спільної для двох критичних шляхів вершини є збіг кодів вершин і кодів мультиресурсів і факт відмінності тривалостей шляху від вершини до кінцевої вершини не більше, ніж на час налагодження в мультиресурсі. Побудований за такими правилами граф (рис. 6) є графом меншої розмірності, тому що він включає тільки вершини на критичних шляхах. На отриманому графі потім здійснюється побудова оптимальної послідовності.

Особливістю процедур нумерації є те, що при об'єднанні спільних вершин можуть з'явитися однакові ребра. Тому при нумерації передбачається процедура видалення однакових зв'язків. Крім того, потрібно виключити ситуації, коли поєднуються вершини, з яких не всі листові. Тоді потрібно видалити виникаючі при цьому «зайві» нульові зв'язки.

Якщо виконується планування виробничих проектів, то після побудови графа на критичних шляхах можуть бути розраховані розміри партій для кожної агрегованої роботи. Агреговані роботи розбиваються на однакове число партій (при цьому процедура розподілу найбільш проста і швидка, тому що не вимагає перегляду попередників по кожній призначуваній партії і кожному спадкоємцю).

Мінімальне число партій визначається максимальною довжиною партії, рівною довжині зміни, так як не повинна бути порушена умова безперервності виконання партії. Тому мінімальна кількість партій дорівнює цілій частині максимальної з тривалостей агрегованих робіт, поділеній на довжину зміни, плюс одиниця. Нарешті, кількість агрегованих робіт у партії визначається для агрегованої роботи як тривалість агрегованої операції, помножена на кількість мультиресурсів у групі і розділена на кількість партій.

По отриманих масивах графа на критичних шляхах будується оптимальна послідовність агрегованих робіт за допомогою розв'язання задачі МЗМ. Алгоритм розв'язання є оригінальною розробкою лабораторії комбінаторної оптимізації НТУУ «КПІ» під керівництвом д.т.н. проф. О.А.Павлова, що враховує останні досягнення в області проектування ПДС-алгоритмів для важкорозв'язуваних комбінаторних задач [2]. У результаті розв'язання задачі МЗМ отримуємо послідовність виконання агрегованих робіт, розбиту на підпослідовності максимального пріоритету (ПМП).

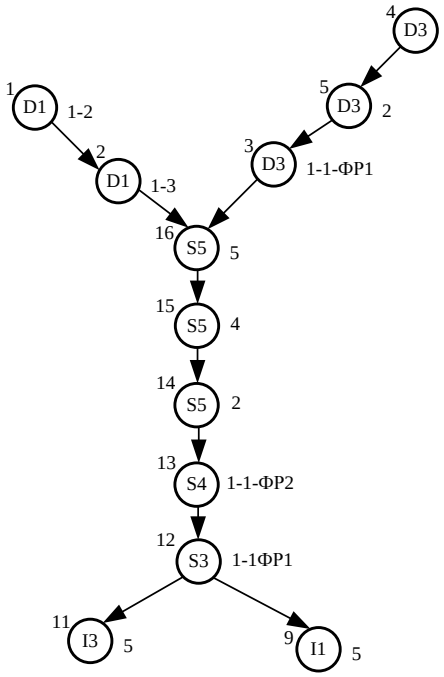


Рис. 6. Граф на критичних шляхах двох завдань (I1 та I3)

Після цього будується новий граф зв'язності, у якому кожна з спільних вершин перетворюється в ланцюжок вершин критичних шляхів завдань зі збереженням відношення передування графа на критичних шляхах. При цьому попередники кожної об'єднаної в графі на критичних шляхах спільної вершини стають попередниками першої (починаючи з листів графа) вершини в ланцюжку, а спадкоємці кожної об'єднаної спільної вершини стають спадкоємцями останньої вершини в ланцюжку.

Наприклад, якщо вершині *i* графа на критичних шляхах відповідає вершина P_i загального графа, а спільні вершини мають номера 3, 6, і їм відповідають по дві вершини загального графа (позначимо їх умовно P_{31}, P_{32}, P_{61}, P_{62}; вони здобувають свої власні номери в даній процедурі), то здійснюється перетворення графа на критичних шляхах, показане на Рис. 8.

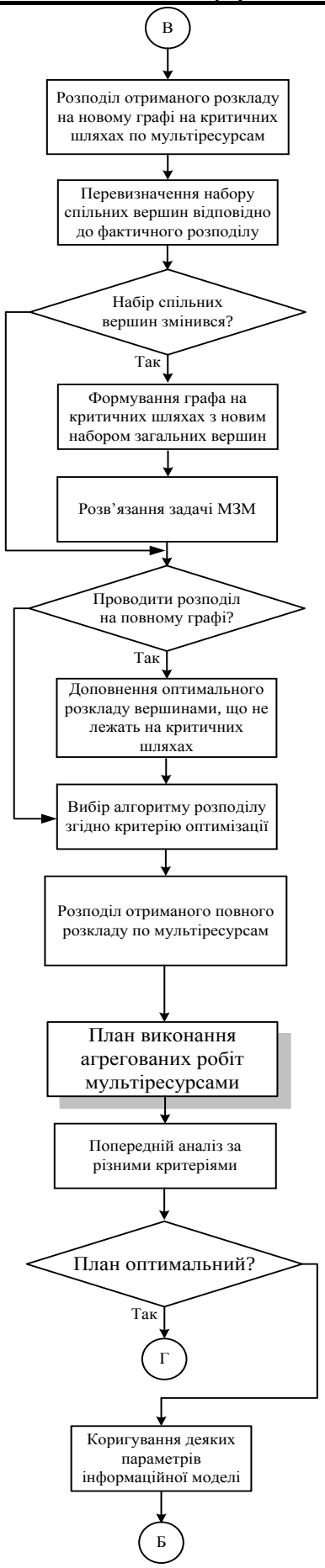


Рис. 7. Рівень II. Побудова плану виконання агрегованих робіт мультиресурсами (ч.2)

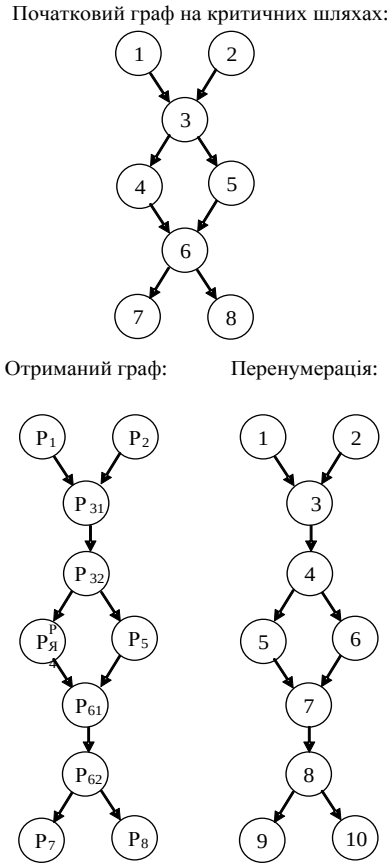


Рис. 8. Приклад перетворення графа в блоці 6

Далі виконується розподіл агрегованих робіт побудованого графу критичних шляхів з розбивкою спільних вершин для уточнення інформації про об'єднання спільних вершин на стадії розподілу. У користувача в діалозі запитуються дата початку планового періоду і дата закінчення планового періоду. За календарем робочих днів визначаються номери робочих днів у плановому періоді. По кожному мультиресурсу розраховується фонд часу мультиресурсу в днях, що дорівнює добутку кількості робочих днів у плановому періоді і коефіцієнта змінності.

Процедура розподілу працює для будь-якого вхідного графа (загальний граф зв'язності агрегованих робіт, граф на критичних шляхах чи граф на критичних шляхах з розбивкою спільних вершин). Для правильної роботи відповідно до обраної підзадачі відкриваються відповідні заданому графу масиви зв'язків і вершин. Це можливо завдяки уніфікації їхніх структур. На даному етапі алгоритму виконується розподіл по графу з розбивкою спільних вершин. При виробничому плануванні вводяться наступні припущення:

- Агреговані роботи не розбиваються на партії: при розбивці на партії неможливо ви-

значити фактичний набір спільних вершин, тому під «партією» в алгоритмі треба розуміти «агрегована робота»;

- У зв'язку з тим, що тривалість нерозбитих на партії агрегованих робіт може перевищувати тривалість робочого дня в деяких мультиресурсах, вважається, що всі мультиресурси працюють цілодобово.

Алгоритми розподілу побудовані з використанням алгоритмів Гіфлера-Томпсона [3]. Під завданнями розуміються або агреговані роботи, або партії робіт, залежно від обраної задачі.

Позначення:

P_t – частковий календарний план з t–1 призначеного завдання;

S_t^k – множина завдань, готових до призначення на k-ий мультиресурс на стадії t;

Sj_k – можливий найбільш ранній момент початку виконання завдання з S_t^k (Sj_k дорівнює моменту завершення попереднього завдання або нулю для завдання без попередників за графом зв'язності);

R_{ik} – i-й інтервал масиву R резервного часу мультиресурсу k, не зайнятий виконанням завдань, призначених на попередніх стадіях;

$R_{ik}^{н}$ – початок і-го інтервалу;

$R_{ik}^{к}$ – кінець і-го інтервалу;

$\sigma_{j_k} = \max(R_{ik}^{н}, Sj_k)$ – момент початку виконання завдання j_k, призначеного на виконання на інтервалі R_i мультиресурсу k;

Cj_k – момент завершення виконання завдання j_k, призначеного на виконання на інтервалі R_i мультиресурсу k: $Cj_k = \sigma_{j_k} + l_{j_k}$.

$\sigma_{j_k}^0$ – мінімально можливий момент початку виконання завдання $j_k \in S_t^k$: $\sigma_{j_k}^0 = \min_k \min_{j \in S_t^k} \sigma j_k \; \forall k, i$: $\sigma_{j_k}^0 + l_{j_k} \le R_{ik}^{к}$.

$C_{j_k}^0$ – мінімально можливий момент завершення виконання завдання $j_k \in S_t^k$: $C_{j_k}^0 = \min_k \min_{j \in S_t^k} Cj_k \; \forall k, i$: $R_{ik}^{к} - l_{j_k} \le R_{ik}^{н}$.

Розподіл агрегованих робіт заданого графа (оптимальної послідовності) по мультиресурсах здійснюється з виконанням наступних принципів. Виконується стільки ітерацій, скільки партій агрегованих робіт розраховано (без розбивки на партії – 1 ітерація). На кожному кроці призначається на мультиресурс партія агрегованих робіт з мінімальним часом звільнення мультиресурсу і мінімальним номером ПМП. Мінімальний час звільнення мультиресурсі партією агрегованих робіт визначається раннім часом початку виконання партії з урахуванням моменту початку виконання завдання плюс тривалість її виконання (з необхідним налагодженням устаткування). Ранній час початку виконання партії визначається максимальним з часів закінчення всіх його попередників по графу, визначених при призначенні попередника на виконання. При призначенні партії шукається перший вільний проміжок часу в рамках фонду часу мультиресурсу, початок якого не раніш, ніж ранній час початку призначуваної партії, а тривалість не менш тривалості виконання партії з переналагодженням. Резерв шукається в «реальному» часі мультиресурсу. Це означає, що враховується не тільки робочий час мультиресурсу, у якому власне йде виконання, але і часи початку і кінця робочих днів, доробків на початок планового періоду, і час початку роботи всіх мультиресурсів (наприклад, 8-ма година ранку).

Алгоритм 1 виконує mn стадій та будує *компактні* [3] розклади.

1. $t = 1$, $P_1 = \varnothing$, $\sigma_i = \sigma_1$, де σ_i – i-я ПМП, яка включає всіх попередників по графу G. Формуємо множини S_1^k завдань, у яких немає попередніх та потребуючих виконання в k-ому мультиресурсі, $k = \overline{1,m}$.

2. Вибираємо з множини S_1^k для призначення на виконання завдання j_k з мінімальним часом закінчення виконання.

3. Переходимо до наступної стадії. При цьому:

а) поміщаємо завдання j_k в P_t і, таким чином, створюємо P_{t+1};

б) формуємо S_{t+1}^k, $k = \overline{1,m}$, додавши до одного з S_t^k завдання, що безпосередньо слідує за завданням j_k, якщо тільки j_k не останнє в σ_1, а інші S_t^k залишивши без змін;

в) формуємо масив резервів мультиресурсу k;

г) покладемо $t = t + 1$.

4. Якщо залишилися непризначені завдання з ПМП №1, то переходимо к п.2. У противному випадку переходимо до наступної ПМП и к п.2. Якщо призначені на виконання всі завдання – кінець алгоритму.

Наступна модифікація алгоритму 1 дозволяє одержати незатримуючі розклади. ***Алгоритм 2*** заснований на наступній евристиці: у п. 2 алгоритму призначається на мультиресурс завдання з мінімальним номером ПМП та мінімальним часом початку виконання, який визначається як σ_{j_k} = max($R_{ik}^{н}$, Sj_k); $\sigma_{j_k}^0$ = $\min_k \min_{j \in S_t^k} \sigma j_k \, \forall k, i$: $\sigma_{j_k}^0 + l_{j_k} \le R_{ik}^{к}$.

Алгоритм 3 може застосовуватись для розподілу завдань при оптимізації за критерієм мінімізації запізнення, тобто без порушення директивних строків завдань. При цьому послідовність розглядається з початку, із завдань більш високого пріоритету. На першому кроці розглядається завдання зі списку кінцевих вершин, що входять до складу ПМП №1 і призначається таким чином, щоб момент закінчення його виконання відповідав директивному строку завдання. На кожному наступному кроці призначається на мультиресурс завдання з максимальним часом запуску в мультиресурсі та мінімальному номері ПМП, у яку воно входить. Максимальний час запуску завдання в мультиресурсі визначається найбільш пізнім часом закінчення виконання завдання мінус тривалість його виконання (з необхідним налагодженням). Найбільш пізній час закінчення виконання завдання визначається мінімальним із часів запуску всіх його спадкоємців по графу, визначених при призначенні попередників на виконання. При призначенні завдання шукається перший вільний проміжок часу в рамках фонду часу мультиресурсу, кінець якого не пізніше, ніж найбільш пізній час закінчення призначуваного завдання, а тривалість не менш тривалості виконання завдання з переналагодженням. Резерв шукається в «реальному» часі мультиресурсу. Якщо необхідний для призначення завдання резерв не знайдений у всіх однотипних мультиресурсах, то це означає, що виконання даного завдання не укладається в його директивний строк, про що користувачу видається повідомлення. Користувач при цьому повинен виключити дане завдання з числа комплексів робіт та перерозподілити план виконання робіт.

Після першого розподілу на графі критичних шляхів, проводиться перевизначення набору спільних вершин відповідно до фактичної інформації про розподіл. Уточнення набору спільних вершин графа на критичних шляхах здійснюється тому, що фактична інформація про розподіл, отриманий на попередньому кроці, визначає більш точну модель графа, чим отримана в порівнянні тривалостей виконання з початку критичному шляху.

Це здійснюється наступним алгоритмом: для кожній спільній вершини перевіряються ознаки об’єднання в графі на критичних шляхах та об’єднання при розподілі, і якщо вони не збігаються, то ознака об’єднання в графі на критичних шляхах встановлюється у значення об’єднання при розподілі, ознака необхідності перерозподілу в «Істина».

Якщо набір спільних вершин не змінився, то модель не має потреби в зміні і, отже, потрібно тільки перерозподілити агреговані роботи з розбивкою на партії (при виробничому плануванні) та доповненням до повного графу (див. наступний крок).

Інакше модель перебудовується і шукається нова оптимальна послідовність виконання агрегованих робіт. Для цього повторно виконуються процедури, починаючи з формування графа на критичних шляхах завдань. Для побудови графа використовується набір спільних вершин, отриманий на попередньому кроці.

У результаті блоку оптимізації отримана припустима оптимальна послідовність, що розбита на підпослідовності спадаючого пріоритету. Така послідовність дає мінімальне значення критерію. Однак вона включає тільки агреговані роботи, що лежать на критичних шляхах завдань, а для розподілу плану виконання робіт необхідно врахувати всі агреговані роботи, необхідні для виконання усіх комплексів робіт. Тому для подальшої роботи алгоритму необхідно доповнити отриману оптимальну послідовність агрегованими роботами завдань, що не лежать на критичних шляхах, з присвоєнням їм відповідного номеру ПМП. Весь надграф кожної з агрегованих робіт, що входить до оптимальній послідовності, буде мати той же номер ПМП, що й ця агрегована робота.

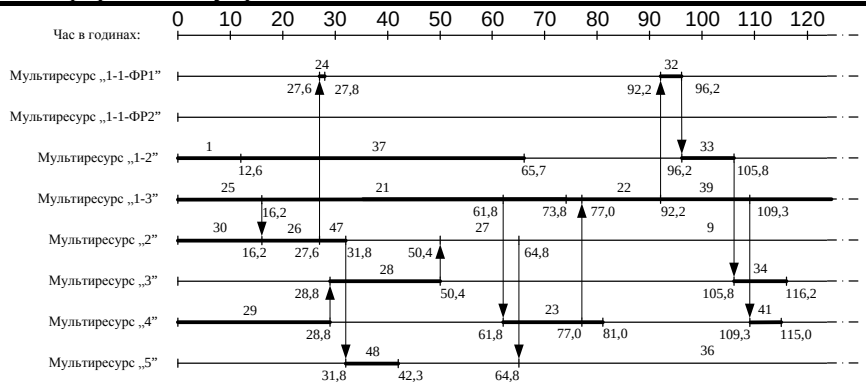


Рис. 9. Результат розподілу плану виконання робіт

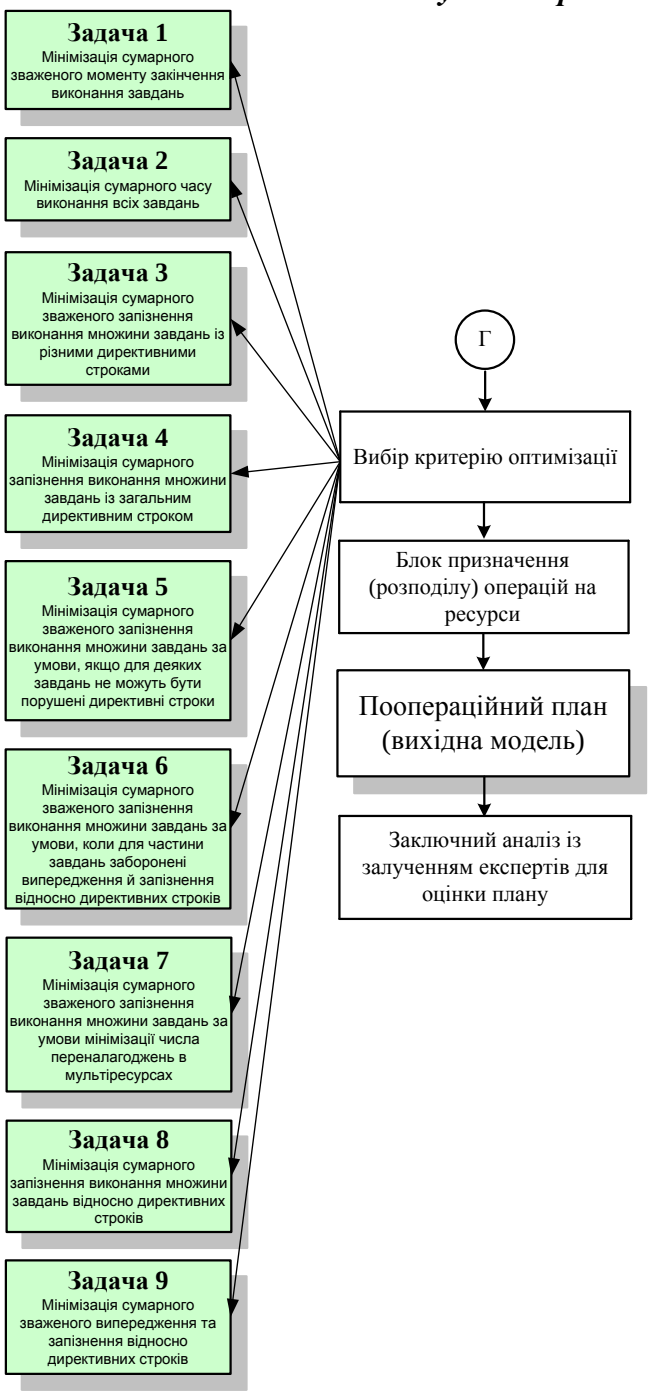


Рис. 10. Рівень III. Розподіл робіт по ресурсах

Розподіл отриманого розкладу по мультиресурсах з прив’язкою до планового періоду виконується за одним з алгоритмів розподілу, описаних вище, за такими виключеннями: алгоритм виконується по загальному графу зв’язності агрегованих робіт чи графу на критичних шляхах, у залежності обраної підзадачі; у виробничому плануванні виконується розбивка агрегованих робіт на партії (кількість ітерацій дорівнює кількості партій та виконується для партій замість однієї ітерації для агрегованих робіт повної тривалості); мультиресурси мають відповідну позмінну роботу, а не цілодобову; не Виконується переоб’єднання спільних вершин.

Рівень III. Побудова плану виконання агрегованих операцій мультиресурсами (рис. 10)

Як було сказано раніше, на третьому рівні трирівневої моделі планування здійснюється формування повного плану виконання всіх робіт з розбивкою по ресурсах всіх мультиресурсів і з урахуванням розподілу агрегованих робіт, отриманого на другому рівні. На цьому рівні можуть розв’язуватись оптимізаційні задачі.

Процедури розподілу агрегованих операцій по ресурсах аналогічні алгоритмам розподілу агрегованих робіт по мультиресурсах, описаним вище. Вибір алгоритму розподілу здійснюється залежно від необхідного критерію оптимізації розкладу: для одержання компактних розкладів вибирається Алгоритм 1-АО, для незатримуючих – Алгоритм 2-АО, розкладів без запізнення (для критеріїв, у яких завдання мають директивні строки, що не можна порушувати) – Алгоритм 3-АО. В алгоритмах третього рівня не здійснюється аналіз об’єднання спільних вершин.

Розглянемо на прикладі алгоритм розподілу для одержання компактних розкладів агрегованих операцій (***Алгоритм 1-АО***). Розподіл агрегованих операцій по ресурсах здійснюється за наступними принципами.

Розглядається кожна група однотипних мультиресурсів та здійснюється перебір усіх агрегованих робіт (партій), назначених на виконання у даній групі мультиресурсів. На кожному кроці призначаються на вільне робоче місце агреговані операції з мінімальним часом закінчення, який визначається як час початку виконання партії плюс тривалість виконання агрегованої операції. За побудовою плану виконання агрегованих робіт, усі ресурси будуть мати достатні резерви для виконання операцій, що лежать на критичному шляху операцій агрегованої роботи. За побудовою, кожна агрегована робота містіть ціле число агрегованих операцій рівної тривалості.

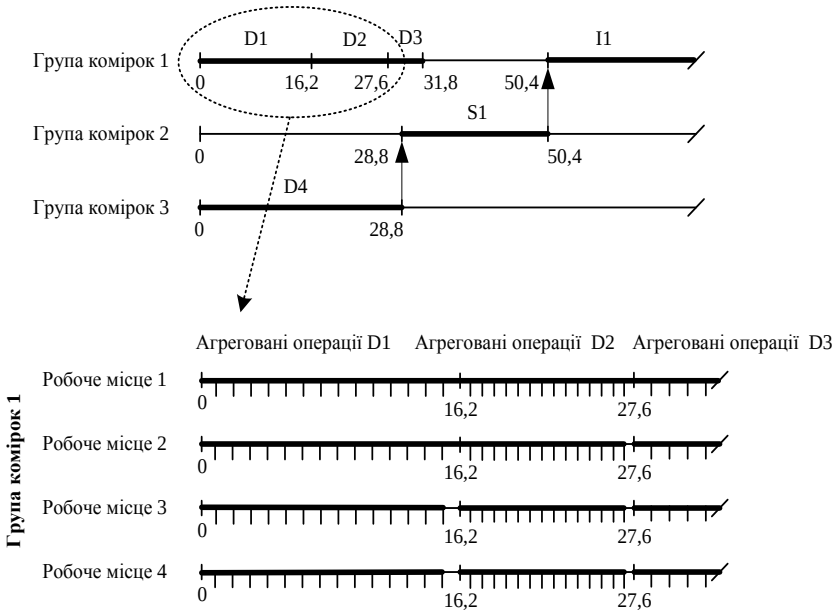


Рис. 11. Приклад розподілу агрегованих операцій по робочих місцях

Схематично процес розподілу агрегованих операцій показаний на рис. 11. Ці резерви можуть бути використані для виконання інших паралельних операцій без переналагодження ресурсу. Як можна бачити, у результаті розподілу, через округлення тривалості агрегованої операції при визначенні тривалості агрегованих робіт, у ресурсах можуть виникати резерви.

Висновки

Загальна схема розв’язання задач в багаторівневій системі планування дрібносерійного виробництва в умовах ринку [1] була адаптована для розв’язання задач планування та управління складними об’єктами з МПТПОР. До таких об’єктів можна віднести широкий клас систем як виробничого, так і невиробничого характеру. Зокрема, на основі створеної моделі авторами вже розроблені комплекси послідовних взаємозв’язаних математичних моделей ієрархічного планування та управління робочим цехом, планування виробництв «на замовлення», планування виробництв дрібносерійного типу; ієрархічного планування виробництв по виготовленню партій, планування та управління проектами. Планується реалізація моделі для будівничих проектів. У статті наведена розширена концептуальна схема інформаційної технології розв’язання задач планування у будь-якому з розглядаємих класів об’єктів управління.

Перелік посилань

1. Павлов О.А., Місюра О.Б., Мельников О.В. Загальна схема розв’язання задач в багаторівневій системі планування дрібносерійного виробництва в умовах ринку / Вісник НТУУ “КПІ”. Інформатика, управління та обчислювальна техніка. К.: “ВЕК+”, 2000.– №33.– С.27-33
2. Основы системного анализа и проектирования АСУ: Учеб. пособие /А.А.Павлов, С.Н.Гриша и др.; Под общ. Ред. А.А.Павлова.– К. : Выща шк.; 1991.– 367 с.
3. Конвей Р.В., Максвелл У.Л. Теория расписаний.– М.: Наука, 1975.– 359 с.

УДК 683.519

ТКАЧЕНКО С. В.

ПРАКТИЧЕСКОЕ ПРИМЕНЕНИЕ МППО НА ПРИМЕРЕ РАЗРАБОТКИ ТОРГОВОЙ СТРАТЕГИИ ДЛЯ РЫНКА FOREX

1. Введение

В настоящей статье рассмотрено практическое применение метода прогнозирования с показателем определенности на примере задачи разработки формальной модели торговой стратегии для рынка FOREX. Демонстрируется использование показателя определенности, описана формальная процедура определения шкалы значимости этой качественной оценки прогноза. На примере прогнозирования тренда курса пары валют приведены количественные расчеты эффективности использования показателя определенности.

2. Метод прогнозирования с показателем определенности (МППО)

Индуктивные методы прогнозирования (такие как нечеткий метод группового учета аргументов (НМГУА), нейронные сети (НС)) не приспособлены для обучения на больших выборках. Как правило для построения модели НМГУА берутся последние N точек имеющихся фактических данных, где N – экспериментально установленный размер окна, дающий в результате аппроксимации наиболее адекватную для прогнозирования модель. В случае превышения N, структура получаемых моделей резко усложняется и ошибка прогноза также быстро возрастает. Таким образом упомянутые методы не способны использовать для обучения всю предысторию временного ряда, и большая часть накопленных знаний не используется. Та же проблема существует и с нейронными сетями, в которых с определенного момента предыдущие знания начинают вытесняться новыми поступающими фактами (эффект забывания).

МППО предлагает способ снять перечисленные ограничения и использовать предысторию для улучшения прогнозирующих характеристик базового метода (НМГУА, НС и т.д.), а также дополнительно предоставляет качественную оценку рассчитанному прогнозу (показатель определенности). В основу метода положено 3 принципа:

1. скользящего окна
2. образной памяти
3. суперпозиции

Принцип скользящего окна позволяет базовому методу обучаться на выборках оптимальной для него длины. За $T-W$ итераций (где T − длина обучающей выборки; W − размер окна, $T>W$) обрабатывается вся выборка. В результате получается некоторое множество N построенных с помощью базового метода интервальных моделей, причем $0<N\le T-W$. Полученные модели можно рассматривать как шаблоны поведения временного ряда на отдельных его участках (образы). При прогнозировании метод пытается найти в памяти аналогии с поведением временного ряда на последнем временным отрезке (окне), непосредственно предшествующем прогнозируемой точке. Полученное множество моделей, способных аппроксимировать последнее окно в рамках заданной погрешности, составляют множество альтернатив A. Если $A=0$, то для такого окна строится новая интервальная модель средствами базового метода (единственная альтернатива, $A=1$). Прогноз представляет собой линейную комбинацию прогнозов по интервальным моделям, из числа A. Факторы, влияющие на прогнозное значение и показатель его определенности, включают величину A, накопленную статистику по рейтингам повторяемости и ошибкам прогнозирования интервальных моделей, ошибки аппроксимации моделями текущего окна, критерий несмещенности и др.. т.н. частные показатели. Общий вид формулы для расчета показателя определенности в прогнозируемой точке $i\in(W,T]$ следующий:

$$D_i=\frac{1}{\sum_v d_b}\sum_b d_b D_{bi},$$

где D_{bi} − частный показатель определенности из фиксированного числа B частных показателей определенности;

d_b − вес частного показателя определенности (внешний параметр метода, константа).

Каждый из частных показателей D_{bi}, в свою очередь, вычисляется на основании характеристик найденного множества моделей-альтернатив для окна, непосредственно предшествующего точке i.

$$D_{bi} = \frac{1}{\sum_m r_m} \sum_m r_m D_{mbi},$$

где D_{mbi} – частный показатель определенности, расчитанный для модели-альтернативы m;

r_m – рейтинг повторяемости модели m, отражающий то, как часто модель m попадала во множество альтернатив при расчетах прогнозных значений в предшествующих точках.

Прогнозное значение рассчитывается как взвешенная свертка прогнозов по найденным альтернативам:

$$P_i = \frac{1}{\sum_m q_{mi}} \sum_m q_{mi} P_{mi},$$

где P_{mi} – прогнозируемое значение в точке i, рассчитанное по модели m;

q_{mi} – вес прогнозного значения по модели m, рассчитываемый на основании ее частных показателей определенности:

$$q_{mi} = \frac{1}{\sum_b d_b} \sum_b d_b D_{bmi}.$$

Подробное описание метода можно найти в [1].

3. Постановка задачи торговой стратегии (ТС) в общем виде

Задано множество исходных данных (обучающая выборка): множество векторов входных переменных $\{X_1, X_2, ..., X_i, ..., X_M\}$ и соответствующих им значений выходной переменной $\{y_1, y_2, ..., y_i, ..., y_M\}$, где i – порядковый номер точки наблюдения; M – число точек наблюдения; $X_i = \{x_{i1}, x_{i2}, ..., x_{iN}\}$ – набор лаговых переменных, изменений курса пары валют (приращений) на N предыдущих точках наблюдения; y_i – фактическое изменение курса пары валют в точке i. Требуется на основании наблюдаемых данных построить модель зависимости выходной переменной от вектора входных переменных $Y_i = Y(X_i)$, а также модель управляющих решений $T_i = T(Y_i)$, где T_i – действие, рекомендуемое по отношению к паре валют, которое может принимать значения: -1 (продавать, курс пары валют будет снижаться), 1 (покупать, курс пары валют будет повышаться) , 0 (никаких действий). Фактическое значение t_i выражается как:

$$t_i = \begin{cases} -1, & \text{если } y_i < 0; \\ 1, & \text{если } y_i > 0; \\ 0, & \text{иначе.} \end{cases}.$$

4. Роль и место МППО в модели ТС

Используя исходные данные, МППО способен построить модель $Y_i = Y(X_i)$, адекватную наблюдаемым данным. Метод также предоставляет качественную оценку, характеризующую уверенность в сделанном прогнозе, – показатель определенности $D_i = D(X_i)$, условно нормализованный к диапазону $[0;1]$, $D_i \in [0;1]$, но не дается никаких указаний по использованию этой оценки. В зависимости от различных характеристик временного ряда (таких как полнота входных переменных, волатильность и др.) шкала значимости этой качественной оценки будет отличаться. Более того, со временем эта шкала требует пересмотра и корректировки. Для получения формальной модели ТС, нужно установить рабочий уровень показателя определенности (РУПО), – некоторое значение, по достижению которого прогноз будет считаться надежным. Поиск значения РУПО является отдельной подзадачей, рассматриваемой в разделах **6** и **7**. Пока, допустив наличие определенного РУПО, выразим T следующим образом:

$$T_i = T_i^{МППО} = \begin{cases} -1, & \text{если } Y_i < 0 \wedge D_i \geq D^w; \\ 1, & \text{если } Y_i > 0 \wedge D_i \geq D^w; \\ 0, & \text{иначе.} \end{cases} \quad (0.1)$$

где $D^w \in [0;1]$ – значение РУПО.

5. Оценка эффективности ТС: моделирование выигрышей/проигрышей

Будем называть точку, в которой $T_i \neq 0$, стратегической точкой. Рассчитаем эффективность Z_i для предлагаемого действия T_i в такой точке:

$$Z_i = T_i t_i |y_i|. \quad (0.2)$$

Таким образом, эффективность измеряется в курсовых единицах и отражает сумму выиг-

рышей/проигрышей по обоим трендам: восходящему и нисходящему. Эффективность ТС на участке $i = [1..M]$ оценивается как:

$$Z = \sum_{i}^{M} Z_i \qquad (0.3)$$

6. Определение РУПО

Зависимость Z от D^w не имеет аналитического выражения, поэтому предлагается разбить диапазон допустимых значений D^w $[0;1]$ на K равных подмножеств, и, принимая верхнюю границу каждого подмножества в качестве РУПО, рассчитать общую эффективность для каждого случая (Z_k) на некоторой обучающей выборке (обычно непосредственно предшествующей рабочему периоду). Так получаем табличную функцию искомой зависимости.

В простейшем случае для работы выбирается РУПО, показавший максимальную эффективность:

$$D^w \text{ соответствующий } Z_{\max}, \qquad Z_{\max} = \max_k Z_k \qquad (0.4)$$

7. Определение стратегических точек

Определение РУПО, само по себе, является частью более сложной задачи, а именно задачи определения стратегических точек (т.е. таких для которых $T_i \neq 0$), решение которой дает модель $T_i = T(Y_i)$.

В некоторых задачах доступны дополнительные качественные оценки, которые также важно учитывать наравне с РУПО. В случае прогнозирования курсов валют, такой оценкой является абсолютная величина прогнозируемого отклонения $|Y_i|$, поскольку она характеризует силу тренда. Прогноз определенного но незначительного отклонения ($|Y_i| < Y^w \wedge D_i > D^w$) при прочих равных условиях более рискован чем прогноз значительного отклонения ($|Y_i| > Y^w \wedge D_i > D^w$). Включим эту оценку в модель T. После адаптации (0.1), модель T принимает конечный вид:

$$T_i = \begin{cases} -1, \text{ если } Y_i < 0 \wedge D_i \geq D^w \wedge |Y_i| \geq Y^w; \\ 1, \text{ если } Y_i > 0 \wedge D_i \geq Y^w \wedge |Y_i| \geq Y^w; \\ 0, \text{ иначе.} \end{cases} \qquad (0.5)$$

Это изменение затрагивает расчет максимальной эффективности $Z_{\max}$ на обучающей выборке, которая теперь является ф-ей двух аргументов: D^w и Y^w. Вместо (0.4) теперь будем использовать:

$$\begin{aligned} &D^w \text{ соответствующий } Z_{\max}, && Z_{\max} = \max_{k,l} Z_{kl} \\ &Y^w \text{ соответствующий } Z_{\max}, && Z_{\max} = \max_{k,l} Z_{kl} \end{aligned} \qquad (0.6)$$

где k − индекс значения-кандидата D^w;
l − индекс значения-кандидата Y^w.

Как и в случае D^w, для Y^w существует проблема сбалансированности между рискованностью и доходностью, ведь прогнозов с определенностью в районе единицы исключительно мало, а величина прогнозируемого колебания может никогда не превысить заданного Y^w. Рассмотрим пример вида зависимости $Z(D_k^w, Y_l^w)$ построенной на основании обучающей выборки (взято из расчетов проведенного исследования для пары USD/CHF):

Как видно из графика (проекция 1), подтверждается гипотеза о том, что при низких значениях Y^w эффективность падает, особенно это заметно при низких значениях D^w. При $D^w = 0 \wedge Y^w = 0$ эффективность соответствует той, которая была бы при отсутствии качественных оценок, здесь она принимает отрицательное значение, т.е. сумма проигрышей перекрывает сумму выигрышей.

Проекция 2 того же графика подтверждает другую гипотезу, а именно, о балансе между надежностью и доходностью. По мере роста значений D_k^w Z_{kl} при любых Y_l^w демонстрирует одну и ту же тенденцию, − сперва резкий скачок, затем плавное снижение до 0, вызванное уменьшением общего кол-ва стратегических точек ($T_i \neq 0$). В данном случае, ТС имела бы эффективность близкую к нулевой уже при $D^w = 0,21$, что существенно меньше максимально возможного значения, 1, тогда как максимальные значения Z_{kl} приходятся на $D_k^w \in [0,02; 0,05]$, т.е. близких к нулю. Все это говорит о достаточно высокой волатильности прогнозируемости ряда, слабой корреляции выходной и управляющих переменных. Также стоит отметить здесь высокую чувствительность Z к малейшим изменениям D^w в диапазоне $[0; D_{\max}^w]$ и, наоборот, слабую для D^w в диапазоне $[D_{\max}^w; 1]$.

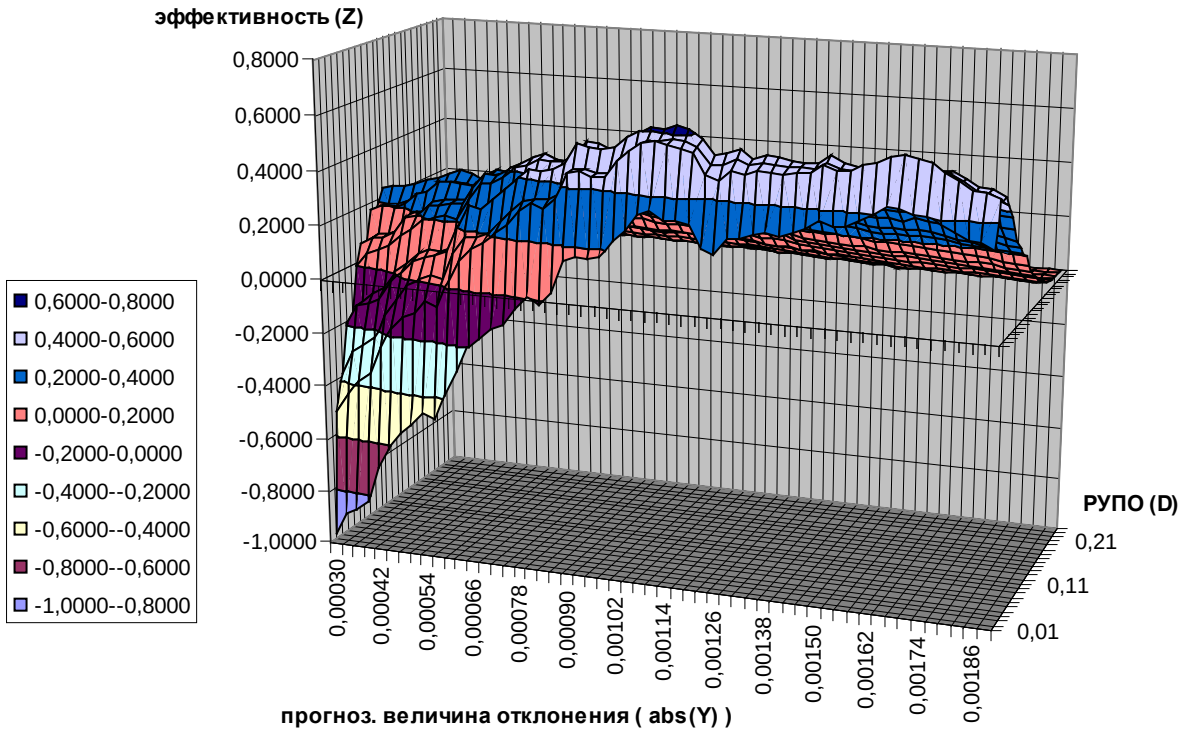


Рис. 1. Пример графика зависимости $Z(D_k^w, Y_l^w)$ ***(проекция 1)***

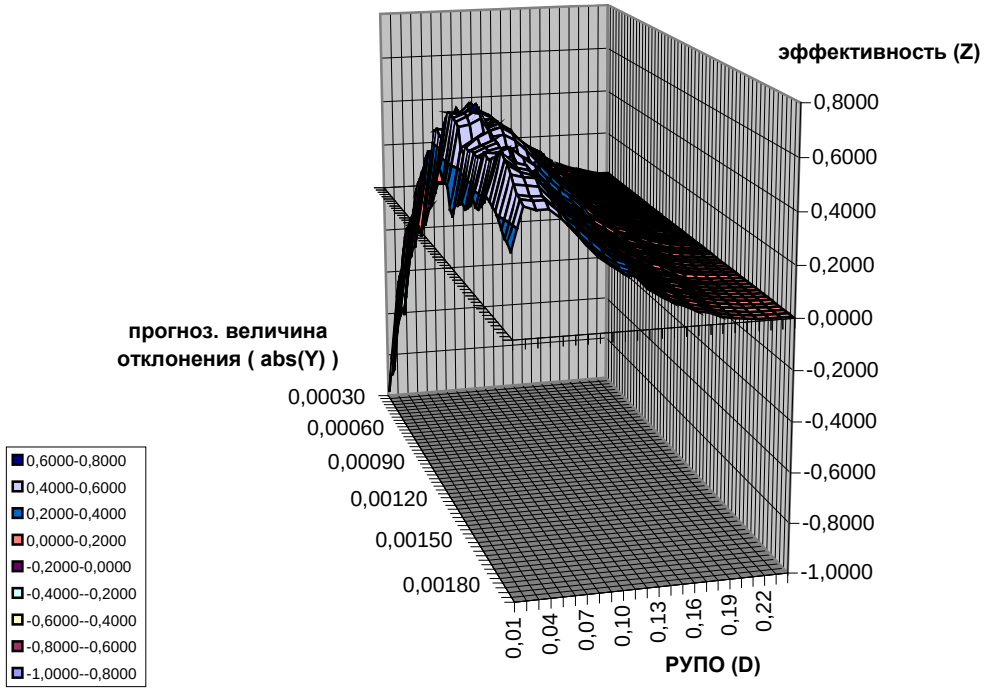


Рис. 2. Пример графика зависимости $Z(D_k^w, Y_l^w)$ ***(проекция 2)***

Процедура определения подходящих значений D^w и Y^w может отличаться от (0.6). Более сложный вариант может включать анализ смежных значений Z, для минимизации риска, т.к. зачастую $Z_{\max}$, полученные по (0.6), расположен недалеко от места, где график резко «проваливается».

Двумя внешними параметрами этой процедуры являются размер выборки, на которой подбираются оптимальные значяения D^w и Y^w и периодичность их корректировки. И если объективность полученных значений очевидно будет тем выше чем чаще они будут пересчитываться по мере поступления новых фактических данных, то вопрос о размере выборки для расчета остается открытым.

8. Результаты исследования

Исходные данные:

- Пара валют: USD/CHF
- Тайм фрейм: 4 часа
- Общая выборка: 6 полных лет (2001−2006 гг.)
- Рабочая выборка: 5 лет (данные за 1-ый год использовались для начальной настройки параметров D^w и Y^w

Параметризация:

- МППО использовался в связке с НМГУА для моделирования Y в пределах скользящего окна.
- Параметры НМГУА:
  - кол-во вх. переменных: $N = 6$ содержащих значение курса на пред. N точках.
  - размер обучающей выборки: 5 точек (см. тайм фрейм)
  - размер проверочной выборки: 6 точек (см. тайм фрейм)
- В качестве модели T использовалась (0.5).
- D^w и Y^w рассчитывались по (0.6) на выборке размером 1 год и пересчитывались в начале каждого года по результатам предыдущего.
- При построении табличной ф-ии $Z(D_k^w, Y_l^w)$ шаг приращения для D^w : $D_{k+1}^w - D_k^w = 0{,}01$, шаг приращения для Y^w : $Y_{l+1}^w - Y_l^w = 0{,}00003$.

Результаты:

Табл. 1. Результаты экспериментального исследования ТС

Год	N	Ошибок прогноза тренда (%)	$\overline{Z^{neg}}$ $\times 10^{-4}$	$\overline{Z^{pos}}$ $\times 10^{-4}$	$\frac{Z}{N}$	Z $\times 10^{-4}$	$Z_{\max}$ $\times 10^{-4}$	$\frac{Z}{Z_{\max}}$ (%)	Z^D $\times 10^{-4}$	Z^Y $\times 10^{-4}$	Z_{GMDH} $\times 10^{-4}$
2001							*1698*				*-2374*
2002	326	42	-16	21	2.82	919	1168	79	345	-712	-2534
2003	272	36	-14	21	5.74	1562	1562	100	1003	-1751	-5472
2004	226	37	-12	20	5.08	1148	1474	78	-780	-1660	-6312
2005	212	42	-14	20	2.76	585	866	68	-3040	-267	-5852
2006	162	52	-15	28	2.69	436	660	66	-4418	-229	-6503
2007	53	42	-12	21	4.36	231	287	80	-4244	-389	-5267
AVG	**209**	**43**	**-14**	**22**	**3.91**	**814**	**1003**	**78**	**-1856**	**-835**	**-5323**

где N − кол-во стратегических точек в году, т.е. таких для которых $T_i \neq 0$;

Z − эффективность ТС за год (рабочие значения D^w и Y^w определялись на основании $Z^{\max}$, рассчитанного на фактических данных предыдущего года);

$\overline{Z^{neg}}$ при $\frac{1}{M}\sum_{i=1}^{M} Z_i$, $_i < 0$ − ср. проигрыш по убыточным сделкам в году;

$\overline{Z^{pos}}$ при $\frac{1}{M}\sum_{i=1}^{M} Z_i$, $_i > 0$ − ср. выигрыш по прибыльным сделкам в году;

$\frac{Z}{N}$ − средний выигрыш по сделке в году;

$\frac{Z}{Z_{\max}}$ − отношение, показывающее эффективность априорных значений D^w, Y^w полученных на основании данных за предыдущий год по сравнению с их апостериорными значениями, полученными по результатам года;

Z^D − годовая эффективность для модели T, учитывающей только D^w (Y^w игнорируется);

Z^Y − годовая эффективность для модели T, учитывающей только Y^w (D^w игнорируется);

Z_{GMDH} − годовая эффективность для модели T, игнорирующей критерии D^w, Y^w (соответствует применению базового метода НМГУА в каждой точке, т.е. кол-во стратегических точек равно общему кол-ву точек). GMDH − Group Method of Data Handly.

Примечание: результаты Z и $Z_{\max}$ учитывают расход на комиссию за торговую сделку в размере 3 курсовых ед. для каждой стратегич. точки.

Интерпретация полученных результатов

Полученные результаты подтверждают ценность предоставляемого МППО показателя

определенности сделанного прогноза. Значение $\overline{Z^{pos}}$ по всем годам стабильно больше $\overline{Z^{neg}}$. Что касается процентного отношения ошибок прогноза тренда, то только в 2006 году этот показатель превышает 50%, тем не менее разница между $\overline{Z^{pos}}$ и $\overline{Z^{neg}}$ позволяет достичь за этот год положительной эффективности в 436 курсовых единиц. Z^D, Z^Y, Z_{GMDH} приведены для того, чтобы показать как добавление качественных оценок улучшает результаты.

Значения показателя $Z / Z_{\max}$ дают основания полагать, что есть запас увеличения эффективности за счет сокращения периода корректировки параметров D^w, Y^w, и, возможно, изменения размера выборки для поиска наилучших значений.

Среди прочего стоит отметить некоторую корреляцию между N и Z. Практически весь диапазон значений, принимаемых $\frac{Z}{N}$ (за исключением 2003г.) можно описать треугольным нечетким числом (3.9, 1.2). Поэтому несмотря на тенденцию уменьшения N год от года, что говорит о возрастающей неопределенности временного ряда, средняя прибыль по сделке остается более или менее постоянной.

9. Заключение

На примере задачи построения формальной модели ТС, было продемонстрировано применение показателя определенности, предоставляемого МППО, а также приведены расчеты эффективности полученной модели. Показано значительное преимущество МППО по сравнению с «чистым» НМГУА, который не дает качественных оценок сделанного прогноза.

МППО имеет значительное кол-во параметров, что присуще методам индуктивного моделирования. Процедура настройки и адаптации некоторых из параметров частично поддается формализации. Был приведен пример формальной процедуры определения РУПО. В дальнейшем имеет смысл рассмотреть методы автонастройки значений и других параметров, значения которых сейчас определяются неформальными методами (являются эспертными оценками).

Список литературы

1. *Ткаченко С.В.* Метод прогнозирования с показателем определенности //Системные исследования и прогрессивные технологии. – 2007. – № 2. – С. 131-141.
2. *Зайченко Ю.П., Кебкал О.Г., Крачковский В.Ф.* Нечеткий метод группового учета аргументов и его применение в задачах прогнозирования макроэкономических показателей //Научные вести НТУУ КПИ, №2, 2000г., с. 18-26.
3. *Зайченко Ю.П.* Основы проектирования интеллектуальных систем. – Киев: Слово, 2004. – 352 с.
4. *Лукашин Ю.П.* Адаптивные методы краткосрочного прогнозирования временных рядов. – Москва: Финансы и статистика, 2003. – 416 с.

З М І С Т

Любченко К. М. Експертні системи в практичній медицині	3
Марковский А.П., Зюзя А.А., *Шерстюк В.Д.* Получение булевых преобразований специальных классов для построения эффективных алгоритмов защиты информации	7
Зайченко Ю.П., Мурга Н.А. Применение систем на нечёткой логике к задаче медицинской диагностики	14
Абу Усбах А.Н., Мельник А.П., Пономарчук Д.С. Об одном подходе к обнаружению ошибок передачи данных на основе систем остаточных классов	23
Салапатов В.І. Синтаксичний аналіз із розподілом лексем на групи	29
Марковський А.П.,Порхун Е.В.,Мнацаканов А.В. Хеш-память с ограниченным временем поиска по ключу	33
ПУСТОВАРОВ В.І.ПОБУДОВА БАЗ ЗНАНЬ ДЛЯ СУМІСНОГО СТВОРЕННЯ ПРОГРАМ І ОБЛАДНАННЯ ТА ЇХ ФОРМАЛЬНОЇ ВЕРИФІКАЦІЇ	41
ТОМАШЕВСЬКИЙ В.М., ДІДЕНКО Д.Г. ПОРІВНЯННЯ РЕЗУЛЬТАТІВ РОБОТИ СИСТЕМ ІМІТАЦІЙНОГО МОДЕЛЮВАННЯ OPENGPSS ТА GPSS/PC	
АМОНС О.А., ЯНОВ Ю.О., БЕЗПАЛИЙ І.О. КЛАСТЕРИЗАЦІЯ ДОКУМЕНТІВ НА ОСНОВІ СТАТИСТИЧНОЇ БЛИЗЬКОСТІ ТЕРМІВ	
Аленин О.И. Система управления виртуальными кластерами	
*Баня Е.Н., Селиванов В.Л.*Об особенностях построения систем счисления разных классов	
Сергієнко А.М. Проектування обчислювачів з регістровими затримками	
Павлов О.А., Місюра О.Б., Мельников О.В., Щербатенко О.В., Михайлов В.В. Загальна схема планування та управління складними об'єктами, що мають мережне представлення технологічних процесів й обмежені ресурси	
Ткаченко С. В. Практическое применение МППО на примере разработки торговой стратеги для рынка FOREX	

Відомості про авторів

Абу Усбах Олексій Нідалійович	кандидат технических наук, доцент кафедри обчислювальної техніки НТУУ ”КПІ”. E-mail: aliksey@mail.ru
Аленин О.И.	
Амонс Олександр	к.т.н., старший викладач кафедри АУТС
Баня Евгений Николаевич	доцент, кандидат технических наук, доцент кафедры АСОИУ факультета ИОТ НТУУ «КПИ”. Телефон 80444529288.
БЕЗПАЛИЙ І.О.	*СТУДЕНТ, НТУУ «КПІ»*
Діденко Д.Г., аспірант, асистент кафедри	*Діденко Д.Г., аспірант, асистент кафедри*
Зайченко Юрий Петрович	Доктор технических наук, профессор кафедры ММСА НТУУ «КПИ» УНК «ИПСА». E-mail: zaych@i.com.ua
Зюзя Олександр Андрійович	аспірант кафедри обчислювальної техніки НТУУ ”КПІ”.
Любченко Костянтин Миколайович	старший викладач кафедри інтелектуальних і інформаційних систем Черкаського національного університету ім. Б. Хмельницького. E-mail: lkn@ukr.net
Марсковський Олександр Петрович	кандидат технічних наук, кафедра обчислювальної техніки НТУУ ”КПІ”. E-mail: markovskyy@mail.ru
Мельник Олександр Петрович	аспірант кафедри обчислювальної техніки НТУУ ”КПІ”.
Мельников О.В	канд. техн. наук, ст. наук. спів., кафедра АСОІУ НТУУ
Михайлов В.В.	канд. техн. наук, ст. наук. спів., кафедра АСОІУ НТУУ ”КПІ”
Місюра О.Б.	канд. техн. наук, ст. наук. спів., кафедра АСОІУ НТУУ ”КПІ”
МНАЦАКАНОВ А.В.	
Мурга Николай Алексеевич.	Магистрант НТУУ «КПИ» УНК «ИПСА» кафедры ММСА E-mail: murga.nicholas@gmail.com
Павлов Александр Анатолиевич	д.т.н., проф. каф. АСОИУ НТУУ «КПИ» E-mail: compscience@asu.ntu-kpi.kiev.ua
Пономарчук Дмитро Сергійович	студент кафедри обчислювальної техніки НТУУ ”КПІ”.
Порхун Е.В.	
Пустоваров В.І.	канд. техн. наук, кафедра обчислювальної техніки НТУУ ”КПІ”
Салапатов В.І.	канд. техн. наук, доцент, ***кафедра кібернетики, ЧНУ***
Селиванов Виктор Львович	доцент, кандидат технических наук, доцент кафедры ВТ факультета ИОТ НТУУ “КПИ”. Рабочий телефон 80444549291
Сергієнко А.М.	канд. техн. наук, ст. наук. спів., кафедра обчислювальної техніки НТУУ ”КПІ”
Ткаченко Сергей Викторович	Место работы: аспирантура Института Прикладного Системного Анализа (НТУУ «КПИ» УНК «ИПСА») E-mail: tkachenko_sv@ukr.net
Томашевський Валентин Миколайович	професор, доктор технічних наук, професор кафедри АСОІУ факультету ІОТ НТУУ “КПІ”. E-mail: simtom@svitonline.com
Шерстюк Валентина Дмитрівна	студентка кафедри обчислювальної техніки НТУУ ”КПІ”.
Щербатенко О.В.	канд. техн. наук, ст. наук. спів., кафедра АСОІУ НТУУ ”КПІ”