Міністерство освіти України
Національний технічний університет України “КПІ”

ВІСНИК

НАЦІОНАЛЬНОГО
ТЕХНІЧНОГО
УНІВЕРСИТЕТУ
УКРАЇНИ “КПІ”

Інформатика, управління та обчислювальна техніка

Заснований у 1964 р.
Випуск 47

Київ “ВЕК+”
2007

УДК 004

Рекомендований до друку Вченою радою факультету інформатики та обчислювальної техніки, протокол № 4 від 13.12.2007

Головний редактор: Самофалов К.Г., член кор. НАНУ, д.т.н., проф.

Заст. гол. ред.: Пустоваров В.І., к.т.н., доц.,

Стіренко С.Г., к.т.н.

Відповідальний секретар: Кушніренко Н.В.

Редакційна колегія: *Павлов О.А., д.т.н., проф.,*
Луцький Г.М., д.т.н., проф.,
Костюк В.І., д.т.н., проф.,
Теленик С.Ф., д.т.н., проф.,
Бузовський О.В., д.т.н., проф.,
Симоненко В.П., д.т.н., проф.,
Жабін В.І., д.т.н., проф.,
Кулаков Ю.О., д.т.н., проф.,
Марковський О.П., к.т.н., доц.,
Стенін Н.А., д.т.н., проф.,
Гріша С.Н., д.т.н., проф.,
Томашевський В.М., д.т.н., проф.

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

ISSN 0201-744X
ISSN 0135-1729

Збірник наукових праць
Українською і російською мовами

Підп. до друку 13.12.2007. Формат 60×84 1/16. Гарнітура Tames. Папір офсетний №1. Наклад 150 прим.
Надруковано в ЗАТ “ВІПОЛ”, 03151, г. Київ, вул. Волинська, 60.

© Національний технічний університет України “КПІ”, 2007

УДК 519.854.2

ПАВЛОВ А.А.,
АРАКЕЛЯН Г.А.,
КУТ В.И.

НАХОЖДЕНИЕ ВЕСОВ ОБЪЕКТОВ ПО МАТРИЦЕ ПАРНЫХ СРАВНЕНИЙ, НЕ СОДЕРЖАЩЕЙ ЦИФРОВОЙ ИНФОРМАЦИИ

По матрице парных сравнений, содержащей информацию вида $w_i \gtrless w_j$ (w_i, $i=\overline{1,n}$ вес *i*-того объекта), при определенных допущениях предлагаются и обосновываются процедуры оценки значений весов объектов.

In this paper the procedures of the objects weights estimation are offered and proved at the certain assumption by the matrix of pair comparisons containing the following information: $w_i \gtrless w_j$ ($w_i, i=\overline{1,n}$ weight of i object).

Эффективность метода анализа иерархий Саати существенно зависят от точности нахождений весов объектов по матрицам парных сравнений [1–7]. В этих работах предполагалось, что матрицы парных сравнений содержат пусть искаженную, но цифровую информацию о сравнении двух альтернатив.

В статье исследуется формально некорректная задача: $\forall\, ij$ элемент матриц парных сравнений последнего уровня иерархии [6] содержит только информацию вида $w_i > w_j$ $\forall i \neq j$ ($w_i, i=\overline{1,m}$ – неизвестные веса объектов, которые априори считаются неотрицательными) либо являются пустыми. В такой общей постановке, корректно оценить неизвестные значения весов невозможно.

Однако в некоторых случаях и такая постановка задачи может привести к конструктивным результатам.

Предварительно рассмотрим кратко метод анализа иерархий Саати. (МАИ).

Постановка задачи

Рассмотрим следующую задачу многокритериального выбора:

Есть глобальная цель и ряд альтернатив $A_1 \ldots A_m$. Необходимо выбрать ту альтернативу, которая является «наилучшей» с точки зрения глобальной цели. Если бы можно было, введя единицу измерения достижения альтернативой глобальной цели, построить функцию $f(A_i)$ $i=\overline{1,m}$, измеряющую в заданных единицах соответствие альтернативы $A_i, i=\overline{1,m}$ за-

данной цели, то задача имеет тривиальное решение: выбирается та альтернатива, для которой достигается максимум $\max_i f(A_i)$.

Примечание. Глобальная цель может принципиально не задаваться функцией $f(A)$ – числовой, скалярной, ограниченной, однозначной. Относительно глобальной цели может нарушаться условие транзитивности: существуют такие альтернативы A_i, A_j, A_e, для которых выполняется $A_i \succ A_j$, $A_j \succ A_e$, $A_e \succ A_i$.

Для большинства реальных задач проблема непосредственного построения функции $f(A_i)$ является неразрешимой в силу многих факторов:

1) слабо структурированные задачи;

2) сочетания количественных и качественных факторов, с помощью которых оцениваются задачи;

3) уверенность в том, что существуют факторы, задающие функцию $f(A_i)$, которые исследователю не известны либо непосредственно не могут быть измерены.

В этом случае и применяется метод анализа иерархий, который является аппроксимацией исходной задачи.

Вкратце основные этапы решения задачи многокритериального выбора с помощью метода анализа иерархий Т. Саати [1–4] следующие.

1. Очертить проблему и определить, что необходимо узнать.

2. Построить иерархию, начиная с вершины (цели – с точки зрения управления), через промежуточные уровни (критерии, от которых зависят следующие уровни) к самому нижнему уровню, который обычно является перечнем альтернатив.

3. Построить множество матриц парных сравнений для каждого из нижних уровней – по одной матрице для каждого элемента примыкающего сверху уровня.

4. По матрицам парных сравнений найти значения весов объектов.

5. Найти согласованность всей иерархии.

6. Найти наилучшую альтернативу.

Заложенный в МАИ принцип идентичности и декомпозиции предусматривает структурирование проблем в виде иерархии или сети. Обычно в наиболее элементарном виде иерархическая структура строится с вершины, через промежуточные уровни к самому низкому уровню. Поэтому первый шаг при решении задачи многокритериального выбора состоит в декомпозиции и представлении задачи в иерархической форме. Имеем следующую задачу принятия решений (Рис. 1), представленную в иерархической форме.

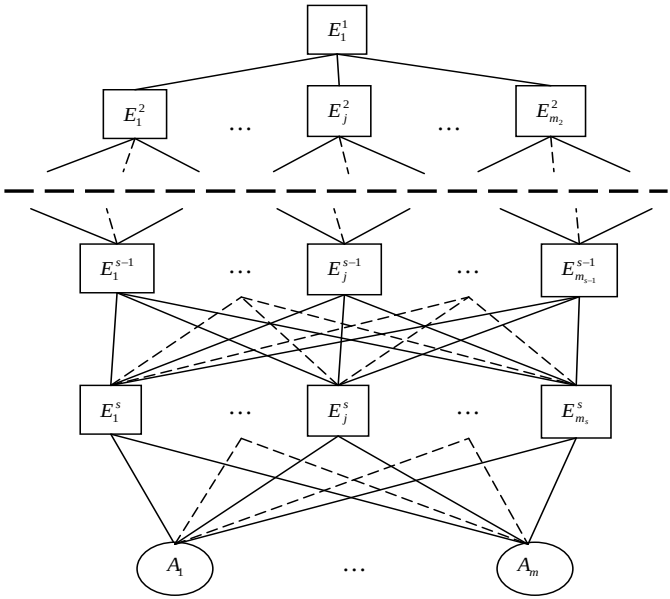


Рис. 1 Пример иерархического представления задачи принятие решений

В представленной на рисунке задаче имеем m альтернатив $A_1 ... A_m$ и s уровней критериев $E_j^i,\ i = \overline{1,s}\ \ j = \overline{1,m_i}$.

На последующих этапах МАИ элементы иерархии сравниваются попарно по отношению к их воздействию на общую характеристику (глобальную цель либо критерий верхнего уровня). Из группы матриц парных сравнений формируется набор локальных приоритетов, которые выражают относительное влияние множества элементов на элемент примыкающего сверху уровня, т.е. в нашем случае компоненты векторов будут иметь следующий вид [6]:

$$W^A_{E_1^{s-1}} = [W^A_{E_1^s} W^A_{E_2^s} \dots W^A_{E_{m_s}^s}]\, W^E_{E_1^{s-1}}$$

$$\dots$$

$$W^A_{E_{m_{s-1}}^{s-1}} = [W^A_{E_1^s} W^A_{E_2^s} \dots W^A_{E_{m_s}^s}]\, W^E_{E_{m_{s-1}}^{s-1}}$$

$$\dots$$

$$W^A_{E_1^{s-j}} = [W^A_{E_1^{s-j+1}} W^A_{E_2^{s-j+1}} \dots W^A_{E_{m_{s-j+1}}^{s-j+1}}]\, W^E_{E_1^{s-j}}$$

$$\dots$$

$$W^A_{E_{m_{s-j}}^{s-j}} = [W^A_{E_1^{s-j+1}} W^A_{E_2^{s-j+1}} \dots W^A_{E_{m_{s-j+1}}^{s-j+1}}]\, W^E_{E_{m_{s-j}}^{s-1}}$$

Последняя формула

$$W^A_{E^1_1} = [W^A_{E^2_1} W^A_{E^2_2} \ldots W^A_{E^2_{m2}}] W^E_{E^1_1}, \qquad (1)$$

$W^E_{E^1_1} = \begin{pmatrix} w^{E^2_1}_{E^1_1} \\ \vdots \\ w^{E^2_{m_2}}_{E^1_1} \end{pmatrix}$, где $w^{E^2_j}_{E^1_1}$ – вес критерия E^2_j в глобальную цель E^1_1, $j = \overline{1, m_2}$.

$W^A_{E^1_1} = \begin{pmatrix} w^1_{E^1_1} \\ \vdots \\ w^m_{E^1_1} \end{pmatrix}$, где $w^j_{E^1_1}$ – результирующий вес i-ой альтернативы в глобальную цель E^1_1, $j = \overline{1, m}$.

Таким образом, j-я компонента (1) – это $W^{\ j}_{E^1_1}$.

$W^A_{E^{s-j}_l} = \begin{pmatrix} w^1_{E^{s-j}_l} \\ w^2_{E^{s-j}_l} \\ \vdots \\ w^m_{E^{s-j}_l} \end{pmatrix}$, где $w^i_{E^{s-j}_l}$ – вес (вклад) j-ой альтернативы в критерий E^{s-j}_l, $i = \overline{1, m}$, $j = \overline{1, s-1}$, $l = \overline{1, m_{s-j}}$.

$W^E_{E^{s-j}_l} = \begin{pmatrix} w^{E^{s-j+1}_1}_{E^{s-j}_l} \\ w^{E^{s-j+1}_2}_{E^{s-j}_l} \\ \vdots \\ w^{E^{s-j+1}_{m_{s-j+1}}}_{E^{s-j}_l} \end{pmatrix}$, где $w^{E^{s-j+1}_p}_{E^{s-j}_l}$ - вес E^{s-j+1}_p критерия в критерий E^{s-j}_l, $j = \overline{1, s-1}$, $l = \overline{1, m_{s-j}}$, $p = \overline{1, m_{s-j+1}}$.

Принимается та альтернатива, на которой достигается максимум $\max_j w^j_{E^1_1}$.

Если в дереве иерархий не все связи имеют место, то соответствующие веса принимаются равными нулю. Поэтому в данной статье рассматривается только общий случай.

Рассмотрим, каким образом дерево иерархий с весами приписанными каждой из ветвей, аппроксимирует неизвестное выражение $f(A_i)$, $i = \overline{1, m}$. Для этого рассмотрим следующие выражения:

$E_j^s(A_i)$ – значение, которое принимает критерий E_j^s – на альтернативе A_i,

$E_j^s(A_i) = w^i_{E_j^s}$, $i = \overline{1,m}$, $j = \overline{1,m_s}$. При этом для фиксированного j $w^i_{E_j^s}$ измеряются в одних и тех же единицах и нормируются: $\sum_{i=1}^{m} w^i_{E_j^s} = 1$.

$E_j^{s-1}(A_i) = \sum_{l=1}^{m_s} w^{E_l^s}_{E_j^{s-1}} E_l^s(A_i)$, при этом веса $w^{E_l^s}_{E_j^{s-1}}$ при фиксированных s и j измеряются в одних и тех же единицах и нормируются $\sum_{j=1}^{m_s} w^{E_l^s}_{E_j^{s-1}} = 1$.

В общем случае $E_t^{s-j}(A_i) = \sum_{l=1}^{m_{s-j+1}} w^{E_l^{s-j+1}}_{E_t^{s-j}} E_l^{s-j+1}(A_i)$, при этом веса $w^{E_l^{s-j+1}}_{E_t^{s-j}}$ при фиксированных s j и t измеряются в одних и тех же единицах и нормируются.

$\sum_{l=1}^{m_{s-j+1}} w^{E_l^{s-j+1}}_{E_t^{s-j}} = 1$.

Примечание: в иерархической системе Саати все веса считаются неотрицательными.

И наконец:

$$E_1^1(A_i) = \sum_{l=1}^{m_2} w^{E_l^2}_{E_1^1} E_l^2(A_i), \quad i = \overline{1,m} \qquad (2)$$

Выражение (2) и есть аппроксимация неизвестного значения $f(A)$, $i = \overline{1,m}$.

$E_1^1(A_i)$, $i = \overline{1,m}$ можно также представить следующим образом:

$$E_1^1(A_i) = \sum_{t=1}^{m_s} \alpha_t w^i_{E_t^s}, \quad i = \overline{1,m},$$

где коэффициенты $\alpha_t \geq 0$ очевидным образом определяются из выражения (2).

Посылка 1. Веса объектов (альтернатив) объективно существуют ($f(A)$ существует).

В этом случае не может нарушаться условие транзитивности и анализ неравенств каждой матрицы парных сравнений последнего уровня иерархии (рис. 1) приводит к построению полного порядка $w_{i_1} > w_{i_2} \ldots > w_{i_m}$.

Рассмотрим последний и предпоследний ряды дерева иерархий МАИ (рис 1).

Пусть $W_1 \dots W_{m_s}$ матрицы парных сравнений альтернатив $A_1 \dots A_m$, относительно критериев E_j^s, $j = \overline{1, m_s}$ соответственно.

Матрицы $W_1 \dots W_{m_s}$ удовлетворяют посылке 1.

Веса альтернатив $A_1 \dots A_m$ в каждой матрице парных сравнений задают строгий порядок:

$$w^{j_1}_{E_j^s} > \dots > w^{j_m}_{E_j^s}, \quad j = \overline{1, m_s}, \quad (3)$$

j_l – номер альтернативы, у которой вес в W_j является *l*-тым по величине.

Обозначим через I множество номеров альтернатив, удовлетворяющих следующим условиям:

Условие 1.

Пусть $|I|$ =к, к – мощность множества I. Тогда из альтернатив множества I можно построить последовательности альтернатив $A_{j_1} \; A_{j_2} \dots A_{j_k}$ для всех $j = \overline{1, m_s}$, т.е. начальные подпоследовательности длины к последовательностей (3).

Условие 2.

Число к является минимально возможным натуральным числом, для которого выполняется условие 1.

Утверждение. Номер оптимальной альтернативы принадлежит множеству I.

Доказательство следует из анализа выражения (2), т.к. $\alpha_t \geq 0 \quad \forall t,$ а для $\forall j \in I$ и любого $l \notin I$ выполняется $w^j_{E_t^s} \geq w^l_{E_t^s}$, $t = \overline{1, m_s}$.

Следствие 1. Если к=1, то множество I задает оптимальную альтернативу.

Следствие 2. Вместо альтернатив $A_1 \dots A_m$ достаточно рассматривать только альтернативы с индексами из множества I.

Обоснуем выбор значений весов в случае, когда матрицы парных сравнений задают лишь полный порядок. Обоснование построим по принципу аналогии. Вспомним классическое определение вероятности: если ни одному элементарному событию нельзя заранее отдать предпочтение до испытания, все элементарные события считаются равновероятными и равными $\frac{1}{s}$, s – общее число элементарных событий.

Матрица парных сравнений не содержит цифровой информации, т.е эксперт (эксперты) не имеет основания считать, что величины

$$\left| w^{j_e}_{E^s_j} - w^{j_{e+1}}_{E^s_j} \right| \text{ и } \left| w^{j_t}_{E^s_j} - w^{j_{t+1}}_{E^s_j} \right|, \quad (4)$$

$e,t = \overline{1,m-1}$, $j = \overline{1,m_s}$, являются различными.

Таким образом, с учетом нормировки получим:

$$w^{j_m}_{E^s_j} = \left| w^{j_e}_{E^s_j} - w^{j_{e+1}}_{E^s_j} \right| = \frac{2}{m(m+1)}; e = \overline{1,m-1} \quad w^{j_1}_{E^s_j} = \frac{2m}{m(m+1)};$$

$$w^{j_e}_{E^s_j} = \frac{2(m-e+1)}{m(m+1)}, \quad e = \overline{2,m.}$$

Посылка 1 должна соответствовать всем матрицам парных сравнений последнего уровня иерархии (рис. 1).

Посылка 2. В матрицах парных сравнений последнего уровня иерархии (рис. 1) дерева иерархий нарушаются условия транзитивности и формально элементы матриц парных сравнений содержат условия вида $A_i \succ A_j$ или $A_j \succ A_i$, втесто неравенств $w_i > w_j$ или $w_j > w_i$.

Это обозначает (в случае достоверности информации, содержащейся в матрицах парных сравнений), что глобальная цель не выражается числовой скалярной ограниченной однозначной функцией от альтернатив. Тогда можно предложить следующую модификацию метода анализа иерархий, которую следует считать «грубой» формализацией процесса принятия решений для всех уровней иерархий. По матрицам парных сравнений последнего уровня иерархии (рис. 1)оценки весов находятся с помощью аналога алгоритма Копленда [8].

Выражения записываем для абстрактной матрицы парных сравнений (т.е. веса не привязаны к критериям предпоследнего уровня иерархии (рис.1)).

Для $j = \overline{1,m}$ рассматриваем *j*-ю строку матрицы парных сравнений: $w_j = 1 + k_j$, где κ_j – количество неравенств в j строке вида $w_j > w_p$ $p = \overline{1,m}$, $p \neq j$.

Примечание. Исходя из общих положений МАИ найденные значения весов должны быть пронормированы $\sum_{i=1}^{m} w_i = 1$.

Список используемой литературы

1. *Saaty T.L.* Multycriteric Decision Making. The Analytic Hierarchy Process, McGraw Hill International. – New York, 1980. Translated to Russian, Portuguese, and Chinese. Revised edition, Paperback. – Pittsburgh, PA: RWS Publications, 1990,1996.
2. *Саати Т., 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. *Ларичев О.И.* Теория и методы принятие решений. – М.:Логос, 2000.
6. *Андрейчиков А.В., Андрейчикова О.Н.* Анализ, синтез, планирование решений в экономике. – Москва: Финансы и статистика. – 2001.
7. *А.А. Павлов, Е.И. Лищук, В.Н. Кут.* Математические модели оптимизации для обоснования и нахождения весов в методе парных сравнений. // Системні дослідження та інформаційні технології 2007р. №2.
8. *Э. Мулен.* Кооперативное принятие решений аксиомы и модели М.: Мир.– 1991.– 451 с.

УДК 004.414.23

КОЛОМОЕЦ Г.П.

ИССЛЕДОВАНИЕ СТАТИСТИЧЕСКИХ ПАРАМЕТРОВ ПЕРЕДАЧИ ПАКЕТОВ С ДАННЫМИ HTTP

Выполнено исследование полученных в ходе эксперимента выборок межпакетных интервалов и длин пакетов с HTTP-запросами и HTTP-ответами с целью определения наиболее подходящих распределений плотности вероятности, которые могут быть использованы при моделировании процессов в компьютерных сетях.

Experimental research of samples of the interpacket intervals and lengths for HTTP-queries and HTTP-replies is carried out with the purpose of definition of the most approaching probability density distributions which can be used for processes simulation in computer networks.

В настоящее время достаточно распространенным является применение специализированных программных систем для моделирования процессов, происходящих при передаче данных в компьютерных сетях [1, 2]. При моделировании активно используются стандартные распределения вероятности, например, для задания времен между последовательно высылаемыми в сеть пакетами, размера пакетов и т.д. Вопросы, связанные с выбором определенного распределения вероятности в качестве входного при моделировании тех либо иных процессов, являются актуальными, поскольку непосредственно влияют на результаты моделирования.

Целью данной работы является определение наиболее подходящих распределений плотности вероятности для собранных в ходе эксперимента наборов межпакетных интервалов и длин пакетов при передаче данных в рамках HTTP-сеанса между клиентом и сервером.

В качестве клиента использовался компьютер с программой браузером, в качестве сервера – прокси-сервер, через который происходит передача всего внешнего по отношению к локальной сети HTTP-трафика. Клиент высылал запросы и получал ответы в течение примерно 124 секунд с 23-х HTTP-серверов Интернета. Следует отметить, что кроме указанного клиента в локальной сети находились и другие компьютеры, однако сетевую активность они не проявляли, за исключением относительно редкой широковещательной рассылки (доля служебных пакетов составила 6,78% от всех захваченных пакетов). В то же время исследуемый клиент проявлял активность, соответствующую обычной работе пользователя в Internet. Для захвата пакетов HTTP-клиентом и регистрации параметров сеанса использовалась свободно распространяемая программа анализатор сетевых протоколов Wireshark v. 0.99.6a [3].

Следует отметить, что из захваченных 1917 пакетов достаточно большое количество составляли служебные пакеты, устанавливающие и завершающие соединения на транспортном уровне. Поскольку в системах моделирования распределения входных параметров обычно задаются на прикладном уровне [1, 2], необходимо было отделить HTTP-запросы и ответы от служебных TCP-пакетов. Количественные характеристики захваченных пакетов приведены в табл. 1, а относительная частота HTTP-запросов и HTTP-ответов – на рис. 1. Количество ответов превышает количество запросов, поскольку если содержимое ответа превышает размер максимального переносимого кадром Ethernet блока в 1500 байт, высылаются несколько кадров с содержимым одного HTTP-ответа [4]. Количество служебных TCP-пакетов сравнимо с количеством HTTP-запросов и ответов.

Табл. 1. Количественные характеристики выборки

Общее время активности 2-х ПК, сек	123,94
Общее количество захваченных пакетов	1917
Количество пакетов только между клиентом и сервером	1787
Количество служебных пакетов	130
Количество HTTP запросов и ответов	887
Количество служебных TCP-пакетов	900
Количество HTTP-запросов	123
Количество HTTP-ответов	764

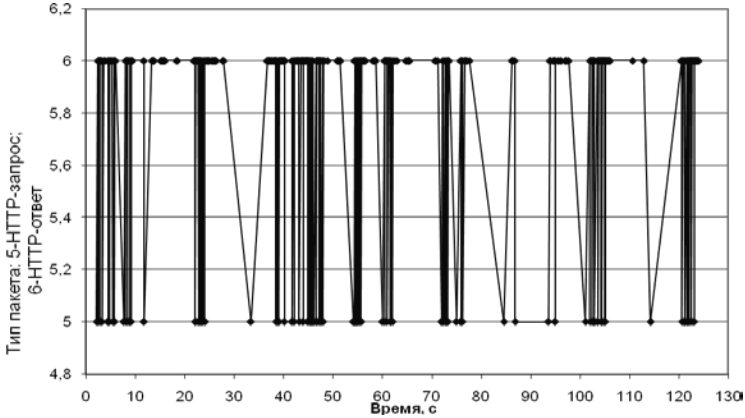


Рис. 1. Относительная частота HTTP-запросов и HTTP-ответов (без служебных TCP-пакетов)

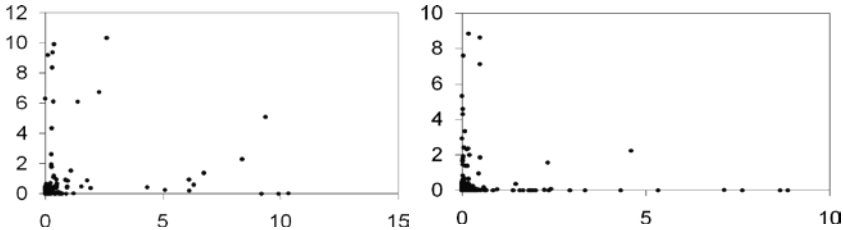
Значения статистических параметров для экспериментальных наборов времен между HTTP-запросами и времен между HTTP-ответами, а также для длин HTTP-запросов и HTTP-ответов приведены в табл. 2 (времена измеряются в секундах, длины пакетов – в байтах).

Табл. 2. Статистические параметры экспериментальных данных

Параметр	**Для времен между HTTP-запросами**	**Для времен между HTTP-ответами**	**Для длин HTTP-запросов**	**Для длин HTTP-ответов**
Количество испытаний	122	763	123	764
Минимальное значение	0,000048	0,000009	60	60
Максимальное значение	10,323789	8,849985	705	1514
Среднее значение	0,992236	0,159238	430,7724	1230,556
Медиана	0,265827	0,020460	459	1502
Дисперсия	4,703899	0,510741	11272,91	237494,709
Коэффициент вариации	2,185818	4,488008	0,24647	0,39603
Асимметрия	3,066372	8,812262	-1,03401	-1,4639203

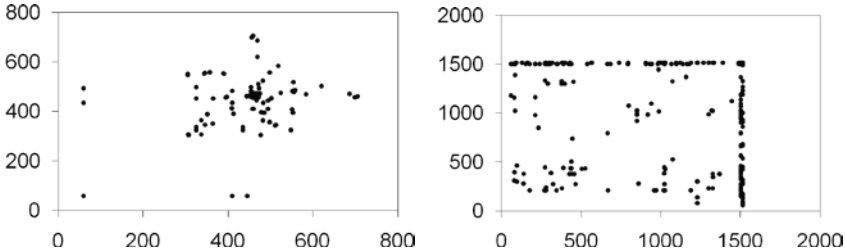
По значениям статистических параметров можно выдвинуть предположения относительно семейства наиболее подходящих распределений плотности вероятности. Так, близость значений среднего и медианы для длин HTTP-запросов и HTTP-ответов может указывать на симметричное распределение (например, нормальное). Учитывая, что для экспоненциального распределения коэффициент вариации равен 1 для любых значений масштабного параметра β, можно предположить, что вероятность ни одной из наблюдаемых величин не подчиняется этому распределению [5]. В то же время именно экспоненциальное распределение рекомендуется в качестве входного при моделировании времен между поступлениями запросов в систему, происходящими с постоянной интенсивностью [6]. Основываясь на полученных значениях асимметрии, можно предположить, что времена между HTTP-запросами и времена между HTTP-ответами имеют смещенную вправо, а их длины – смещенную влево форму распределения.

Поскольку формальное определение соответствия стандартных распределений плотности вероятности экспериментальным наборам данных предполагает выборочную независимость данных, необходимо было определить степень такой независимости [6]. Для этого были построены диаграммы разброса наблюдений для пар времен между HTTP-запросами и пар времен между HTTP-ответами (Рис. 2), а также для пар длин HTTP-запросов и пар длин HTTP-ответов (Рис. 3).



а) б)

Рис. 2. Диаграмма разброса наблюдений для пар времен между HTTP-запросами (а) и пар времен между HTTP-ответами (б)



а) б)

Рис. 3. Диаграмма разброса наблюдений для пар длин HTTP-запросов (а) и пар длин HTTP-ответов (б)

Характер разброса для межпакетных времен позволяет заключить, что выборочные данные обладают определенной степенью независимости, хотя, судя по графикам, типы распределений для времен между HTTP-запросами и времен между HTTP-ответами должны отличаться. Выборочные данные для длин HTTP-запросов также указывают на их относительную независимость, однако длины HTTP-ответов назвать таковыми нельзя, поскольку среди ответов преобладают пакеты максимального размера в 1514 байт (133 значения) и близкого к нему в 1502 байта (404 значения) (их суммарная доля составляет 70%). Этот факт отражает работу драйвера сетевого адаптера прокси-сервера, пытающегося передать в кадрах канального уровня пакет максимально возможного размера с целью уменьшения накладных расходов на передачу заголовков пакетов. Если же не учитывать длины пакетов в 1502 и 1514 байт, то диаграмма разброса для оставшихся длин HTTP-ответов демонстрирует (Рис. 4) достаточную степень независимости значений длин, очевидно связанную со случайным значением длины передаваемых с WWW-серверов файлов.

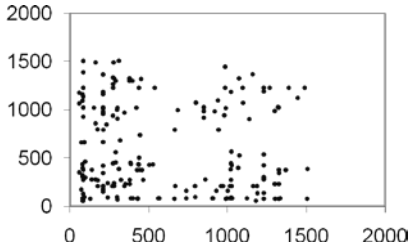


Рис. 4. Диаграмма разброса наблюдений для пар длин HTTP-ответов без учета длин пакетов в 1502 и 1514 байт

Анализ полученных выборок HTTP-запросов клиента к серверу, HTTP-ответов сервера клиенту, а также выборок длин пакетов с HTTP-запросами и длин пакетов с HTTP-ответами на предмет соответствия стандартным непрерывным распределениям плотности вероятности выполнялся в системе статистического анализа данных Statistica 6.0 разработки StatSoft Inc. [7].

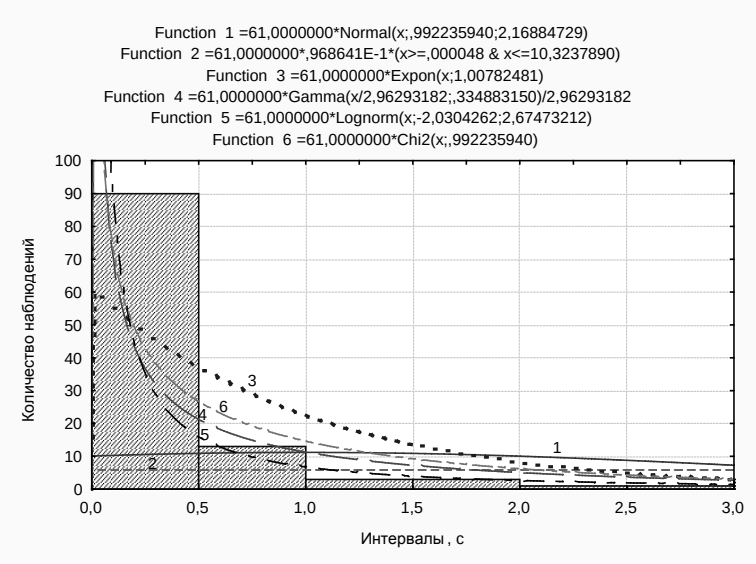


Рис. 5. Гистограмма интервалов времени между HTTP-запросами и стандартные плотности распределения вероятности

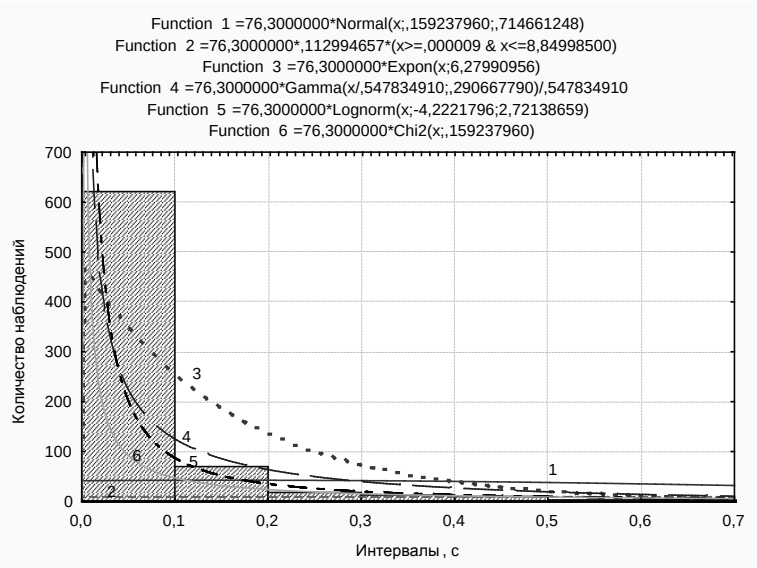


Рис. 6. Гистограмма интервалов времени между HTTP-ответами и стандартные плотности распределения вероятности

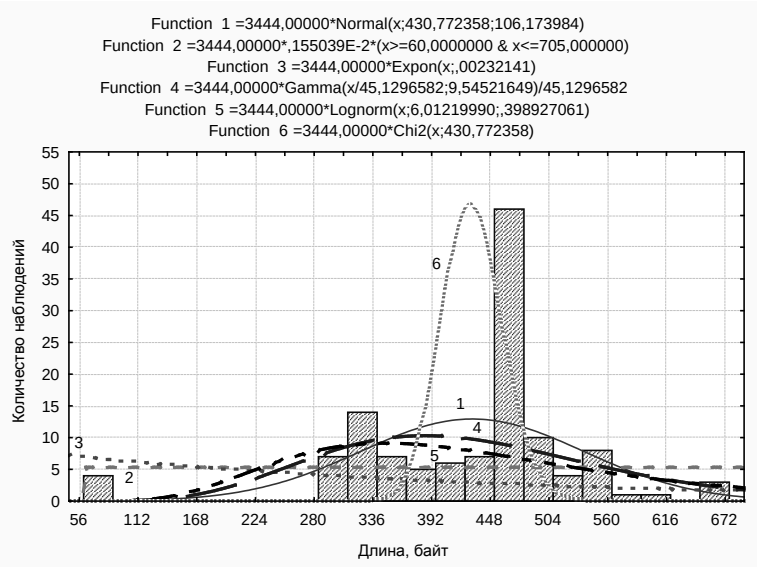


Рис. 7. Гистограмма длин HTTP-запросов и стандартные плотности распределения вероятности

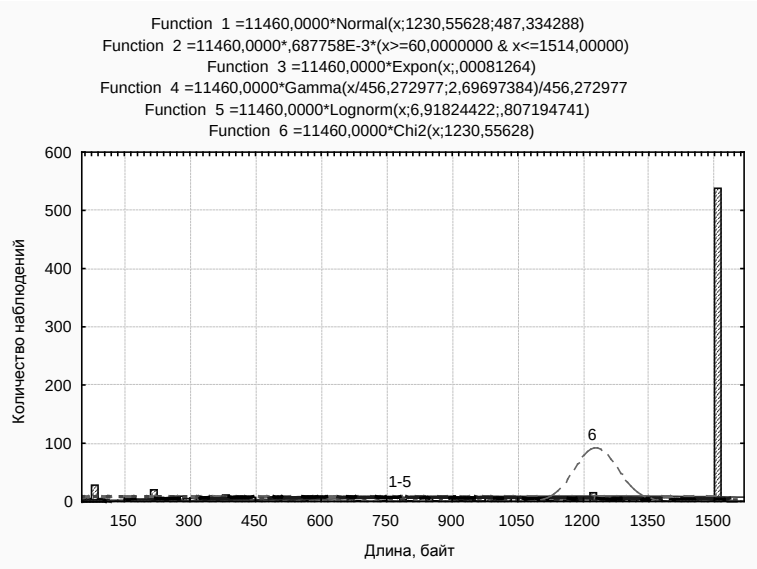


Рис. 8. Гистограмма длин HTTP-ответов и стандартные плотности распределения вероятности:
1 - нормальное, 2 -равномерное, 3 - экспоненциальное,
4 - гамма, 5 - логнормальное, 6 - Хи-квадратичное

Гистограммы времен между HTTP-запросами и времен между HTTP-ответами, длин HTTP-запросов и длин HTTP-ответов, а также кривые доступных в программе плотностей вероятности нормального, равномерного, экспоненциального, гамма, логнормального и Хи-квадратичного распределений приведены на рис. 5 – рис. 8. Согласие стандартной плотности вероятности с экспериментальными значениями оценивалось с помощью критерия "Хи-квадрат" и критерия Колмогорова-Смирнова [6]. Кроме численных значений критериев согласия, естественно, оценивалось совпадение форм плотностей вероятности и гистограмм. Учитывая, что проверка по критерию "Хи-квадрат" чувствительна к выбору количества и размеров интервалов, используемых при построении гистограмм, эти параметры выбирались в соответствии с рекомендациями, состоящими в выборе интервалов одинакового размера и их количества такого, чтобы произведение доли единичного интервала на количество данных в выборке было не меньше 5 [6]. Этим рекомендациям удовлетворяют 22 интервала для 122 значений времен между HTTP-запросами (122/22≈5,55) и 100 интервалов для 763 значений времен между HTTP-ответами (763/100=7,63). При построении гистограмм для длин пакетов по тем же соображениям были выбраны 22 интервала для 123 длин HTTP-запросов и 100 интервалов для 764 длин HTTP-ответов.

Табл. 3. Результаты тестов на соответствие стандартным распределениям

Распределение	**Тест Хи-квадрат**				**Тест Колмогорова-Смирнова**	
	χ^2	**df**	**p**	**χ^2(df, 0,99)**	**d**	**$(n^{1/2}+0,12+0,11/n^{1/2})\cdot d$**
1	**2**	**3**	**4**	**5**	**6**	**7**
Параметры для времен между HTTP-запросами (22 интервала)						
Нормальное	68,91287	6	0,00000	16,812	0,35518	3,97
Равномерное	1284,3637	17	0,00000	33,409	0,75247	8,41
Экспоненциальное	67,69580	4	0,00000	13,277	0,34647	3,88
Гамма	21,52803	4	0,00025	13,277	0,15209	1,70
Логнормальное	15,03783	4	0,00462	13,277	0,16294	1,82
Хи-квадратичное	32,00457	5	0,00001	15,086	0,21599	2,41
Параметры для времен между HTTP-ответами (100 интервалов)						
Нормальное	485,35127	15	0,00000	30,578	0,41184	11,43
Равномерное	44614,687	85	0,00000	≈120	0,90405	25,08
Экспоненциальное	420,18642	6	0,00000	16,812	0,54846	15,22
Гамма	142,73452	10	0,00000	23,209	0,24999	6,94
Логнормальное	42,51791	14	0,00010	29,141	0,31854	8,84
Хи-квадратичное	68,27119	13	0,00000	27,688	0,50593	14,04
Параметры для распределений длин HTTP-запросов (22 интервала)						
Нормальное	118,32683	9	0,00000	21,666	0,18899	2,121
Равномерное	386,86109	19	0,00000	36,191	0,34732	3,897
Экспоненциальное	533,26119	11	0,00000	24,725	0,47487	5,328
Гамма	192,89382	11	0,00000	24,725	0,21137	2,371
Логнормальное	256,61483	12	0,00000	26,217	0,21329	2,393
Хи-квадратичное	187,46842	3	0,00000	11,345	0,37837	4,246
Параметры для распределений длин HTTP-ответов (100 интервалов)						
Нормальное	36207,672	70	0,00000	≈100	0,41542	11,53
Равномерное	38862,774	94	0,00000	≈125	0,69594	19,32
Экспоненциальное	52503,057	72	0,00000	≈102	0,40913	11,36
Гамма	31873,700	80	0,00000	≈107	0,40949	11,37
Логнормальное	43231,961	75	0,00000	106,393	0,39246	10,90
Хи-квадратичное	43285,040	15	0,00000	30,578	0,70839	19,67

В табл. 3 приведены значения параметров теста Хи-квадрат и теста Колмогорова-Смирнова для сравниваемых теоретических распределений и экспериментальных данных. Здесь χ^2 – значение подобранного параметра теста, df – количество степеней свободы, p – относительная величина достоверности используемого распределения, χ^2(df, 0,99) – табличное значение χ^2 для указанного числа степеней свободы, не превышение которого позволяет заключить о достоверности используемого приближения на уровне 0,99 (минимально возможный уровень достоверности), d – наибольшее вертикальное расстояние между теоретической и экспериментальной функциями распределения, в последнем столбце приведена выверенная статистика критерия Колмогорова-Смирнова (минимальным значением этой статистики для выбора с достоверностью на уровне 0,99 является 1,628), n – количество испытаний случайной величины [6].

Как видно из таблицы 3, в соответствии с результатами обоих тестов ни одно из исследованных распределений не может быть выбрано для описания экспериментальных данных даже с минимально возможной достоверностью, однако наилучшими кандидатами для задания времен между HTTP-запросами могут быть гамма или логнормальное распределение. Для определения возможных кандидатов для моделирования межпакетных интервалов HTTP-ответов необходимо исследование других стандартных распределений плотности вероятности. Что касается длин пакетов, то для них также ни одно из исследованных распределений формально не может быть выбрано. Мы удалили детерминированную составляющую, соответствующую максимальным длинам HTTP-ответов в 1502 и 1514 байт и повторили анализ. Соответствующая гистограмма и графики исследованных стандартных распределений плотностей вероятности приведены на рис.9. В этом случае выборка составила 228 наблюдений, распределенных между 60 и 1507 байтами, и при построении гистограммы было выбрано 30 интервалов (228/30=7,6).

В табл. 4 приведен анализ соответствия экспериментального набора данных исследованным стандартным распределениям плотности вероятности. И хотя значения статистики Колмогорова-Смирнова в этом случае находятся ближе к граничному критерию (для экспоненциального и гамма распределений), позволяющему осуществить выбор стандартного распределения, все же формально мы этого сделать не можем. Учитывая полученные результаты, можно рекомендовать в качестве входного распределения плотности вероятности HTTP-ответов выбирать экспоненциальное или гамма распределение для длин 30% пакетов вместе с фиксированной длиной в 1502 (или 1514) байт для 70% пакетов.

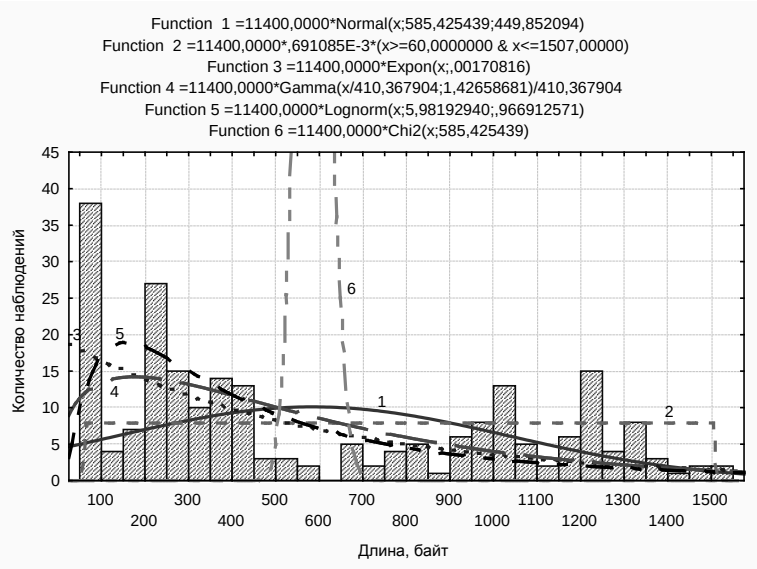


Рис. 9. Гистограмма длин HTTP-ответов, не включающих пакеты длиной 1502 и 1514 байт и стандартные плотности распределения вероятности: 1 - нормальное, 2 -равномерное, 3 - экспоненциальное, 4 - гамма, 5 - логнормальное, 6 - Хи-квадратичное

Табл. 4. Результаты тестов на соответствие стандартных распределений и экспериментальных наборов длин HTTP-ответов без учета HTTP-ответов длиной 1502 и 1514 байт

Распределение	**Тест Хи-квадрат**				**Тест Колмогорова-Смирнова**	
	χ^2	**df**	**p**	**χ^2(df, 0,99)**	**d**	**$(n^{1/2}+0,12+0,11/n^{1/2})\cdot d$**
Параметры для распределений длин HTTP-ответов без 1502 и 1514 байт (30 интервалов)						
Нормальное	145,71012	22	0,00000	40,289	0,18394	2,801
Равномерное	285,43559	26	0,00000	45,642	0,29581	4,504
Экспоненциальное	154,89075	20	0,00000	37,566	0,12115	1,845
Гамма	154,26182	19	0,00000	36,191	0,12819	1,952
Логнормальное	211,13851	17	0,00000	33,409	0,14033	2,137
Хи-квадратичное	1429,8538	2	0,00000	9,210	0,57448	8,748

Выводы

При моделировании передачи пакетов в сети следует учитывать достаточно большое количество служебных пакетов, которые добавляются к запросам и ответам пользовательских приложений, поэтому задание входных распределений статистических параметров передачи должно выполняться для разных уровней сетевого взаимодействия и в общем случае должно отличаться.

Наилучшими кандидатами из исследованных стандартных распределений для выбора в качестве входных распределений плотности вероятности при моделировании интервалов между HTTP-запросами являются гамма и логнормальное распределение. Для определения возможных кандидатов для моделирования временных интервалов между HTTP-ответами необходимо дополнительное исследование.

Ни одно из исследованных распределений не подходит для моделирования распределения длин HTTP-запросов и HTTP-ответов, что связано, очевидно, с достаточно большой детерминированной составляющей в них. Эта составляющая возникает вследствие определенного количества полей в заголовках пакетов с относительно мало варьирующейся длиной для HTTP-запросов и с стремлением драйвера сетевого адаптера передавать HTTP-ответы в кадрах максимально возможного размера с целью уменьшения доли накладных расходов.

Анализ выборки длин HTTP-ответов без учета кадров максимального размера показывает, что возможным решением для выбора входных распределений является использование экспоненциального или гамма распределения для длин 30% пакетов вместе с фиксированной длиной в 1502 (или 1514) байт для 70% пакетов.

Список использованной литературы

9. Система моделирования компьютерных сетей OPNET Modeler – OPNET Technologies,Inc. - http://www.opnet.com/solutions/network_rd/modeler.html.
10. Система имитационного моделирования компьютерных сетей OMNET++ – http://www.omnetpp.org/index.php.
11. Анализатор сетевых протоколов Wireshark – http://www.wireshark.org.
12. Таненбаум Э. Компьютерные сети. 4-е изд. – СПб.: Питер, 2003. - 992 с.
13. Гмурман В.Е. Теория вероятностей и математическая статистика. Учеб. пособие для вузов.-Изд. 7-е, стер.-М.: Высш. шк., 2001.-479с.
14. Лоу А. М., Кельтон В. Д. Имитационное моделирование. Классика CS. 3-е изд. - СПб.: Питер; Киев: Издательская группа BHV, 2004. – 847 с.
15. Система статистического анализа данных Statistica v.6.- StatSoft, Inc. – http://www.statsoft.com.

УДК 004.052.42

МАРКОВСКИЙ А.П.,
МУХАММАД МЕФЛЕХ АЛИСА АБАБНЕ.,
ЗЮЗЯ А.А.,
ГАРАЗД В.М.

К ПРОБЛЕМЕ ЗАЩИТЫ ОПЕРАНДОВ МОДУЛЯРНОГО ЭКСПОНЕНЦИРОВАНИЯ ОТ ИХ РЕКОНСТРУКЦИИ АНАЛИЗОМ ДИНАМИКИ ПОТРЕБЛЕНИЯ МОЩНОСТИ

Целью представленных в статье исследований является анализ опасности реконструкции операндов модулярного экспоненцирования анализом динамики потребления мощности и разработка способов противодействия. Показано, что степень модулярного экспоненцирования, являющаяся ключом алгоритмов RSA, El-Gamal, DSA может быть реконструирована временным анализом потребляемой мощности. Для противодействия разработан специальный алгоритм модулярного экспоненцирования. Алгоритм не имеет условных операторов и включает ложные операции, затрудняющие временной анализ мощности. Применение предложенного подхода увеличивает на 25% время выполнения модулярного экспоненцирования.

The goal of presented by article research is to point out the potential vulnerabilities of modular exponentiation operands reconstruction by power dynamic analysis and to elaborate countermeasures. It has been shown that exponent of modular exponentiation which is secret key of RSA, El-Gamal and DSA can be reconstruction by timing power analysis. For countermeasure the special algorithm for modular exponenttiation has been worked out. Proposed algorithm does not conditional operators use and include the false operators which inhibit to timing power analysis. It has been shown that implementation of proposed approach demand about 25% more time for modular exponentiation.

Введение

Динамическое углубление информационной интеграции на основе компьютерных сетей играет доминирующую роль на современном этапе развития большинства сфер человеческой деятельности. Важным условием расширения информационной интеграции является решение проблемы эффективной защиты данных и размежевание прав доступа к информации в компьютерных системах и сетях. Расширение использования компьютерных и сетевых технологий в сфере финансовой деятельности, для управления сложными техническими системами, связанными с техногенным риском требует обеспечения высокой надежности защиты информации в компьютерных системах управления и сетях. Таким образом, существует объективная необходимость повышения эффективности средств защиты информации с точки зрения расширения сфер использования

интегрированных систем. С другой стороны, объективно существует ряд факторов, снижающих надежность средств защиты данных. В последние годы, одной из наиболее потенциально опасных угроз информационной безопасности в системах компьютерного управления является несанкционированный доступ к данным посредством измерения и анализа динамики изменения параметров работы вычислительного устройства во время реализации им программ [1]. Особенно актуальной является проблема защиты данных от их реконструкции посредством измерения и анализа динамики изменения потребляемой мощности. В современных компьютерных сетях значительную часть терминальных устройств составляют микроконтроллеры, для которых измерение потребляемой мощности и сопоставление ее с выполняемыми командами осуществляется относительно просто [1]. Вместе с тем, такие терминальные устройства поддерживают сетевые протоколы защиты информации, при реализации которых используются секретные данные – ключи и пароли. Современные технологии позволяют выполнять статистическую реконструкцию этих секретных данных на основе анализа динамики потребляемой мощности [2]. Анализ таких технологий свидетельствует о том, что они наиболее эффективны для доступа к постоянным компонентам систем защиты информации – ключам и паролям. Это особенно опасно для механизмов защиты данных, ключи которых редко меняются. К таким механизмам, в частности, относятся алгоритмы с открытым ключом, наиболее распространенными из которых является RSA, El-Gamal, DSA. Изменение ключей при каждом сеансе работы этого алгоритма принципиально нарушает концепцию открытости ключей. Поэтому ключи упомянутых алгоритмов наиболее уязвимы для несанкционированного доступа к ним путем анализа динамики потребления мощности. Вместе с тем, эти алгоритмы лежат в основе сетевых протоколов информационной безопасности, поэтому потеря контроля над их ключами грозит серьезными нарушениями в работе систем защиты информации в сетях.

Таким образом, проблема защиты от реконструкции анализом динамики потребления мощности ключей алгоритмов, в основе которых лежит модулярное экспоненцирование является важной и актуальной на современном этапе развития технологии защиты данных в сетях.

Анализ проблемы защиты данных от реконструкции, осуществляемой анализом динамики потребляемой мощности

Базовой вычислительной операцией при программной реализации большей части алгоритмов защиты информации с открытым ключом, в том числе RSA, является модулярное экспоненцирование $X^E \bmod M$, причем X, $E < M$. Количество n двоичных разрядов чисел X, E и M существенно превышает разрядность k процессора. Каждое из чисел X, E и M, можно представить в виде s слов, состоящих из k двоичных разрядов

($s=n/k$), каждое из которых лежит в интервале от 0 до 2^k-1: $X=\{x_{s-1}, \ldots, x_1, x_0\}$, $E=\{e_{s-1},\ldots,e_1,e_0\}$ и $M=\{m_{s-1},\ldots,m_1,m_0\}$, $\forall j\in\{0,...,s-1\}$, $0\leq x_j,e_j,m_j\leq 2^k-1$. Для указания двоичных разрядов слов X, E и M используются обозначения с двумя индексами, например $e_0=\{e_{0,k-1},e_{0,k-2},\ldots,e_{0,1},e_{0,0}\}$; $\forall l\in\{0,...,k-1\}$: $x_{j,l}$, $e_{j,l}$, $m_{j,l}\in\{0,1\}$:

$$X=\sum_{j=0}^{s-1}x_j\cdot 2^{j\cdot k}=\sum_{j=0}^{s-1}\sum_{l=0}^{k-1}x_{j,l}\cdot 2^{j\cdot(k-1)+l}, E=\sum_{j=0}^{s-1}e_j\cdot 2^{j\cdot k}=\sum_{j=0}^{s-1}\sum_{l=0}^{k-1}e_{j,l}\cdot 2^{j\cdot(k-1)+l},$$

$$M=\sum_{j=0}^{s-1}m_j\cdot 2^{j\cdot k}=\sum_{j=0}^{s-1}\sum_{l=0}^{k-1}m_{j,l}\cdot 2^{j\cdot(k-1)+l}$$

Процесс модулярного экспоненцирования сводится к последовательному выполнению $\log_2 E=n$ циклов, в каждом из которых осуществляется операция возведения в квадрат полученного на предшествующем цикле результата и, дополнительно, в зависимости от текущего бита степени E, выполняется операция умножения. Исходя из порядка, в котором анализируются разряды степени E, существует две разновидности модулярного экспоненцирования: справа – налево и слева – направо.

Алгоритм двоичного экспоненцирования справа – налево.

1. $A = 1$; $S = X$;
2. for (j=0; j< s ; j++)
   2.1. for (l=0; l< k; l++)
   {
      2.1.1. if ($e_{jl} = 1$) $A = A\cdot S$ mod M;
      2.1.2. $S = S\cdot S$ mod M;
   }

Результат $A=X^E$ mod M.

Для вычисления весов используется возведение в квадрат, а для формирования результата – умножение. Среднее число операций умножения – $1.5\cdot n$.

На практике, большее распространение получили алгоритмы, которые предполагают анализ разрядов степени E начиная со старших разрядов (слева-направо): в этом случае множитель при выполнении операции умножения представляет собой постоянной число, равное X, что создает потенциальные предпосылки для повышения скорости умножения.

1. $A = 1$;
2. for (j= s-1; j >=0; j --)
   2.1. for (l=k-1; l>=0; l--)
   {
      2.1.1. $A = A \cdot A$;
      2.1.2. if ($e_{j,l} = 1$) $A = A\cdot X$;
   }

Результат $A=X^E$ mod M.

При использовании алгоритмов, основанных на модулярном экспоненцировании, в качестве секретного ключа выступает код экспоненты *E,* который практически не меняется. Соответственно, основной задачей нарушения защиты является получение кода *E*. В качестве побочной задачи выступает задача доступа к значению кода *X* или результату - $A=X^E \bmod M$ при неизвестном, но постоянном *E* [2] .

Как уже отмечалось, значительную часть терминальных устройств сетей составляют встроенные микроконтроллеры и микропроцессоры, используемые для сбора данных или управления технологическим оборудованием. К 2005 году доля таких терминальных устройств составляла до 60% [2], причем имеет место тенденция ее роста с развитием компьютерных и сетевых технологий, а также с расширением сфер их применения.

Этот класс терминальных устройств поддерживает протоколы защиты информации в сетях, в процессе реализации которых использует постоянные секретные параметры – пароли, ключи криптографических алгоритмов. Для микроконтроллеров и микропроцессоров относительно просто измерить во время выполнения программ динамику потребляемой мощности и сопоставить ее с выполняемыми командами. Все процессорные устройства вычислительной техники построены на логических вентилях, основной частью которых являются транзисторы. Токи, протекающие через транзисторы, зависят от их рабочих точек и меняются скачкообразно при переключении вентилей. При этом, величина потребляемого тока зависит от значения битов данных, которые обрабатываются в текущий момент времени [3]. Для измерения тока, потребляемого вычислительным устройством, в его цепь питания добавляется резистор, напряжение на котором измеряется высокоточным цифровым микровольтметром [3]. Исследовательские цифровые микровольтметры позволяют производить измерения с частотой дискретизации 1-2 ГГц. и точностью до 1мкВ. Промышленные общедоступные образцы обеспечивают частоту дискретизации 50 МГц и точность 5мВ.; Большинство микроконтроллеров, встроенных микропроцессоров и смарт-карт работает с тактовой частотой 20-50 МГц.

С помощью описанного оборудования можно выполнить привязку команд выполняемой программы (для вычислительных устройств рассматриваемого класса они практически не меняются) к диаграмме изменения потребляемой мощности. К настоящему времени известно две технологии реконструкции данных по результатам измерения динамики потребляемой мощности. Первая из этих технологий получила название SPA (Simple Power Analysis) - простого анализа мощности [4]. Сущность этой технологии состоит в непосредственной интерпретации измерений потребляемой мощности во время выполнения известной программы.

На рис.1. показана диаграмма измерений мощности при выполнении модулярного экспоненцирования, предусмотренного алгоритмом RSA на смарт-карте [2]. На диаграмме отчетливо видны циклы умножения и возведения в квадрат. Динамика потребляемой мощности зависит от выполняемой команды и данных, над которыми эта команда выполняется.

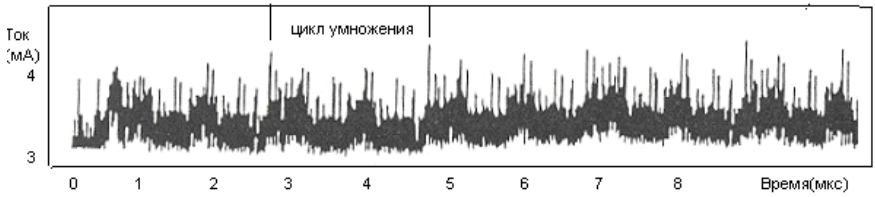


Рис.1. Фрагмент диаграммы потребляемой мощности при модулярном экспоненцировании на смарт-карте.

С использованием технологий SPA достаточно точно можно установить последовательность выполнения команд. Поскольку каждая из команд занимает определенное время, технология SPA в определенном смысле смыкается с временным анализом выполнения команд программы [1].

Другой технологией использования измерений динамики потребляемой мощности для реконструкции обрабатываемых данных является дифференциальный анализ мощности (DPA-Differential Power Analysis). Эта технология также описана впервые в 1999 году [5] и ориентирована на реконструкцию постоянно используемых в программах данных путем статистической обработки большого числа диаграмм потребления мощности при выполнении одной и той же программы. В частности, технология DPA может быть эффективной при анализе большого числа диаграмм потребления мощности при выполнении одной и той же программы, реализующей криптографический алгоритм, которая обрабатывает разные данные, но используемые ключи остаются неизменными. Эта ситуация является типичной при использовании микроконтроллеров и смарт-карт в качестве терминальных устройств компьютерных сетей.

Судя по результатам выполненного анализа публикаций [2], касающихся технологий реконструкции данных с использованием технологий SPA и DPA, в настоящее время не существует математической модели влияния битов обрабатываемой информации на потребляемую мощность. Но, это влияние имеет место и может быть выявлено статистическими методами, на которых построена идея DPA.

В первом приближении можно говорить о том, что технология SPA эффективна для реконструкции данных, от которых зависит порядок выполнения команд, в то время, как DPA позволяет реконструировать данные, непосредственно используемые в вычислениях.

Исходя из этого, для реконструкции экспоненты E наибольшую опасность представляет SPA, поскольку, как следует из приведенных выше алгоритмов, она прямо не участвует в вычислениях, но определяет порядок выполнения команд. Проанализируем потенциальную опасность применения SPA для получения значений разрядов экспоненты E.

Одним из узловых вопросов такого анализа является оценка возможности различения операций возведения в квадрат и умножения. Полученные в работе [2] результаты позволяют положительно оценивать такую возможность даже при условии, что для обеих операций используется один и тот же алгоритм умножения. Исходя из этого, значения всех разрядов E могут быть достаточно просто восстановлены путем отслеживания по диаграмме участков, соответствующих пп.2.1.1 (для экспоненцирования справа-налево) или пп.2.1.2. (для экспоненцирования слева-направо).

Вторая опасность применения SPA состоит в возможности отслеживания по диаграмме изменения последовательности команд в зависимости от условного перехода, выполняемого по результату анализа текущего бита экспоненты E. Упомянутая опасность не зависит от того, различимы ли по диаграмме операции возведения в квадрат и умножение. Пользуясь такой технологией, реконструкция E может быть полнена достаточно просто.

Технология DPA представляет опасность при решении задачи реконструкции кода X путем анализа диаграммы при неизвестном значении экспоненты E. Очевидно, что такая возможность открывается при использовании для модулярного экспоненцирования алгоритма слева-направо, поскольку при реализации пп. 2.1.2. участвует постоянный код X. Этот код потенциально может быть восстановлен путем статистического анализа диаграмм мощности при выполнении соответствующих операций. Следовательно, с точки зрения защиты операндов модулярного экспоненцирования от реконструкции с использованием DPA, предпочтительным является алгоритм справа-налево, в котором нет постоянных операндов.

Таким образом, проведенный анализ показал, что базовые алгоритмы модулярного экспоненцирования не обеспечивают защиты от реконструкции кода экспоненты E с использованием SPA.

Обзор способов противодействия реконструкции закрытых данных, осуществляемой анализом динамики потребляемой мощности

К настоящему времени для противодействия реконструкции данных посредством анализа динамики потребления мощности предложено ряд решений [1-6], которые укладываются в рамках следующих направлений:

- маскирование всех операндов, зависящих от постоянных ключей, случайными кодами, меняющимися при каждом выполнении алгоритма;

- случайные перестановки операций, не нарушающие результата работы алгоритма;
- вставка случайных команд, не влияющих на результат и затрудняющих ”привязку” диаграммы потребления мощности к операциям алгоритма.

Маскирование операндов является наиболее эффективным способом противодействия реконструкции постоянных ключей путем анализа динамики потребляемой мощности [6].

Применительно к маскированию ключа RSA -экспоненты E наиболее известный подход [1] состоит в том, что в качестве аддитивной маски E используется произведение случайного числа r на $\phi(M)=(p-1)\cdot(q-1)$, причем $p\cdot q=M$. Маскированная экспонента E' может быть, таким образом, представлена в виде: $E' = E + r\cdot\phi(M)$. Поскольку для любого X выполняется $X^{r\cdot\phi(M)} \bmod M = 1$, то $X^{E'} \bmod M = X^{E} \bmod M$. Это означает, что для снятия маски нет необходимости в специальных операциях. Недостатком изложенного способа маскирования является то, что наложение аддитивной маски $r\cdot\phi(M)$ существенно увеличивает разрядность n' экспоненты E'. При известном значении модуля M, код $\phi(M)$ достаточно просто определить. Если не использовать в комбинировании с маскированием специальных средств защиты от SPA, то значения n' и E', как показано в предыдущем разделе, могут быть реконструированы простым анализом диаграммы потребления мощности. Учитывая, что $r\cdot\phi(M) >> E$, приближенное значение r' можно определить, как частное $r' \approx E'/\phi(M)$. В силу того, что величины E и $\phi(M)$ являются величинами одного порядка (n-разрядными двоичными числами), то истинное значение $r \le r'$ и отличается от r' на единицы (при близких значениях E и $\phi(M)$, наиболее вероятно, что $r = r'-1$). Это означает, что можно, в принципе, восстановить значение r путем небольшого подбора, а значит и реконструировать значение E.

Следовательно, без применения специальных мер противодействия SPA, маскирование экспоненты E не позволяет исключить возможность реконструкции ключа RES.

Для противодействия SPA наиболее широко используется случайная перестановка операций с использованием комбинирования двух базовых алгоритмов модулярного экспоненцирования [2]. Для этого выбирается случайная, лежащая в интервале от 0 до s-1 стартовая точка $d\in\{0,\ldots,s-1\}$ и осуществляется модулярное экспоненцирование справа-налево от 0 до точки d ,с фиксацией результата в переменной A и степени в переменной S. Затем, имея в качестве начальных полученные значения A и S выполняется модулярное экспоненцирование слева-направо от s-1 до d+1. При использовании такого способа, по мнению его авторов [2] при проведении SPA сложно установить порядок следования битов экспоненты. Однако, следует иметь ввиду, что число вариантов стартовой точки d очень неве-

лико (не более 10^3) и все эти варианты можно перебрать, если знать X и результат X^E mod M. На практике, при постоянной экспоненте E, имея в распоряжении диаграммы потребления мощности при реализации нескольких операций модулярного экспоненцирования достаточно просто реконструировать E.

Другим способом противодействия SPA является перестановка операций возведения в квадрат и умножения. Этот способ предполагает в качестве алгоритмической основы экспоненцирование справа-налево. Сущность способа состоит в том, что вначале выполняются все возведения в квадрат с сохранением в памяти тех результатов (значений S), которые участвуют в операциях умножения (пп. 2.1.1. соответствующего алгоритма). Затем из памяти извлекаются все сохраненные квадраты, участвующие в умножении и подряд выполняются все операции умножения [1]. Ниже представлена соответствующая модификация алгоритма справа – налево.

1. $A = 1$; $S = X$; $h = 0$;
2. for (j=0; j< s ; j++)
   2.1. for (l=0; l< k; l++)
   { 2.1.1. if ($e_{jl} = 1$) { 2.1.1.1. $Z[h] = S$; 2.1.1.2. h ++ ; }
   2.1.2. $S = S \cdot S$ mod M; }
3. for (q =0; q<h; q++) $A = A \cdot Z[q]$ mod M;

Результат $A=X^E$ mod M.

Недостатком такого способа является то, что с использованием технологии SPA можно зафиксировать проверку условия, осуществляемого в пп.2.1.1. Фрагменты диаграммы потребления мощности при выполнении команд пп. 2.1.1.1. и 2.1.1.2. могут быть достаточно просто выделены.

Проведенный краткий обзор существующих способов противодействия SPA, показал, что они не обеспечивают надежной защиты от реконструкции экспоненты E анализом динамики потребления мощности.

Целью работы является создание способов защиты от восстановления кода экспоненты E по результатам анализа динамики мощности, потребляемой вычислительным устройством в процессе выполнения модулярного экспоненцирования.

Организация модулярного экспоненцирования, затрудняющая реконструкцию экспоненты временным анализом потребляемой мощности

Приведенный анализ известных способов противодействия реконструкции экспоненты, осуществляемой с использованием SPA, показал, что узловой проблемой является исключение возможности отслеживания средствами SPA условных операторов, выполняющих последовательное тестирование разрядов E. Для защиты от SPA необходимо модифицировать алгоритм модулярного экспоненцирования таким образом, чтобы

исключить из него условные операторы. В качестве основы модифицированного алгоритма выбрано модулярное экспоненцирование справа-налево, поскольку оно не использует постоянных элементов и, соответственно, более устойчиво к DPA. Ниже представлена предлагаемая модификация алгоритма модулярного экспоненцирования:

1. $A[0] = 1$; $A[1]=1$; $S = X$; $y = 1$;
2. for (j=0; $j< s$; j++)
    2.1. for (l=0; $l< k$; l++)
    {
        2.1.1. $q = e_j \& y$;
        2.1.2. $e_j = e_j >> 1$;
        2.1.3. $A[q] = S \cdot A[1] \bmod M$;
        2.1.4. $S = S \cdot S \bmod M$;
    }

Результат $A[1] = X^E \bmod M$.

В предлагаемой модификации отсутствуют условные операторы, на каждом цикле выполняется одна операция возведения в квадрат и умножения, что не позволяет средствами SPA определить значение обрабатываемого в этом цикле разряда экспоненты. Исключение условных операторов выполнено за счет введения паразитной операции умножения, результат которого фиксируется в $A[0]$, а накопление правильного результата осуществляется в $A[1]$. Вполне очевидно, что за счет введения паразитной операции умножения, вычислительная сложность модулярного экспоненцирования возрастает на 25%.

В качестве второго, эффективного способа противодействия SPA-реконструкции экспоненты E предлагается выполнение модулярного экспоненцирования с использованием аддитивных цепочек.

Аддитивной цепочкой V длиной L для положительного целого числа E называется последовательность $u_0,u_1,\ldots,u_s$ положительных чисел и связанная с ними последовательность $w_1,w_2,\ldots,w_s$ пар $w_i=<i_1,i_2>$, $0\le i_1,i_2<i$, которые обладают следующими свойствами:

1. $u_0 = 1$ и $u_L = E$;
2. $\forall\ u_i$, $1\le i\le L$, $u_i = u_{i_1} + u_{i_2}$

Алгоритм модулярного экспоненцирования $X^E \bmod M$ на основе аддитивных цепочек имеет следующий вид:

1. $G[0] = X$;
2. for(j=1; j<=L; j++) $G[j] = G[j_1]\cdot G[j_2] \bmod M$;

Результат: $G[L] = X^E \bmod M$.

Например, для E=12, можно сформировать аддитивную цепочку в виде множества чисел {1,2,3,6,12}, которым соответствуют пары: w_1=<0,0>, w_2=<0,1>, w_3=<2,2>, w_4=<3,3>. Для той же экспоненты E=12, можно сформировать другую аддитивную цепочку, которая задается множеством

чисел {1,2,4,5,10,12}, которым соответствуют пары: w_1=<0,0>, w_2=<1,1>, w_3=<0,2>, w_4=<3,3>, w_5=<1,4>. В соответствии с приведенным выше алгоритмом, вычисление X^{12} с использованием последней цепочки выполняется в следующей последовательности: $G[0]=X$;

$G[1]=G[0]\cdot G[0] \bmod M = X^2 \bmod M$;

$G[2]=G[1]\cdot G[1] \bmod M = X^4 \bmod M$;

$G[3]=G[0]\cdot G[2] \bmod M = X^5 \bmod M$;

$G[4]=G[3]\cdot G[3] \bmod M = X^{10} \bmod M$;

$G[5]=G[1]\cdot G[4] \bmod M = X^{12} \bmod M$.

Поскольку приведенный алгоритм не содержит условных операторов, то временной анализ динамики потребления мощности при его реализации практически не позволит восстановить значение E, даже, если удастся отличить операции умножения и возведения в квадрат. Поскольку для используемой на практике разрядности n экспоненты (порядка 1024) вариантов ее разложения на аддитивные цепочки существует очень много, то восстановить конкретно используемую цепочку не представляется возможным по результатам анализа динамики потребляемой мощности.

Эффективность использования аддитивных цепочек при модулярном экспоненцировании, как средства противодействия SPA, определяется тем, что на практике экспонента E, являясь закрытым ключом RSA абонента не меняется. Это позволяет сгенерировать, с использованием случайных чисел, аддитивную цепочку, в соответствии с которой скомпилировать программу, которая будет непосредственно выполняться.

Недостатком использования аддитивных цепочек является необходимость в дополнительном объеме памяти для массива G и хранения пар w, которые задают связи компонент цепочек. При этом требуемый объем памяти определяется длиной L цепочки. Если считать, что среднее количество единиц в n-разрядной экспоненте E равно $n/2$, то нижняя граница значения длины L цепочки составляет $n + \log_2 n - 2$. Для реализации приведенного выше алгоритма требуется хранение относительно небольшого числа элементов массива G: в принципе можно выполнить алгоритм при хранении всего 2-х чисел указанного массива. Для хранения пар w, требуется объем V_w памяти, определяемый формулой :

$$V_w = 2\cdot(n+\log_2 n-2)\cdot\lfloor\log_2(n+\log_2 n-2)\rfloor \qquad (1)$$

Например, для наиболее часто встречающегося в практике применения RSA значения n=1024, V_w = 2838 байт, что не превышает объем встроенной памяти большинства современных встроенных микроконтроллеров.

Выводы

Проведенные исследования, направленные на разработку способов противодействия восстановлению экспоненты, осуществляемого путем измерения и анализа динамики потребляемой вычислительной платформой

мощности при модулярном экспоненцировании позволяют сделать выводы:

1. Операция модулярного экспоненцирования лежат в основе ряда криптографических алгоритмов, таких как RSA, El-Gamal, DSA, которые широко используются в протоколах защиты информации в сетях. Экспонента указанной операции является частью закрытого ключа этих алгоритмов, который меняется весьма редко.

2. Наибольшую опасность для реконструкции экспоненты представляет временной анализ потребления мощности вычислительной платформой при выполнении на ней операции модулярного экспоненцирования.

3. Эффективным способом противодействия реконструкции экспоненты временным анализом потребления мощности является исключение условных операторов. Предложен алгоритм, удовлетворяющий этому требованию, включающий паразитные операции, которые затрудняют эффективный анализ динамики потребления мощности. Реализация предложенного алгоритма требует на 25% больше времени по сравнению с обычным модулярным экспоненцирования.

4. Предложен подход к решению задачи противодействия временному анализу потребления мощности для реконструкции секретного кода экспоненты. Подход основан на использовании технологии модулярного экспоненцирования на основе аддитивных цепочек.

Предложенные способы противодействия доступа к закрытым ключам алгоритмов защиты информации, в основе которых лежит операция модулярного экспоненцирования, могут быть эффективно использованы для повышения информационной безопасности компьютерных сетей.

Список использованной литературы

1. Cohher P. Timing Attack on Implementations of Diffie-Hellman, RSA, DSS and other systems.// Proceeding of Advances in Cryptology-“CRYPTO-96”. LNCS-1882. Springer-Verlag.- 1996, PP.104-113.
2. Messerges T.S., Dabbish E.A., Sloan R.H. Power Analysis Attacks of Modular Exponentiation in Smartcards // Proceeding of 1-th International Workshop “Cryptographic Hardware and Embedded Systems”(CHES-1999), LNCS-1717. Springer-Verlag.- 1999.- P. 145-157.
3. Akkar M.L., Bevan C., Dischamp P., Moyart D. Power analysis, what is now possible // Proceeding of International Workshop “Asiacrypt-2000”. LNCS-1976. Springer-Verlag.- 2000.- P. 489-502.
4. Mayer-Sommer R., Smartly analyzing the simplicity and power of simple power analysis on smartcards // Proceeding of 2-th International Workshop “Cryptographic Hardware and Embedded Systems”(CHES-2000), LNCS-1965. Springer-Verlag.- 2000.- P. 78-92.
5. Kocher P., Jaffe J., Jun B. Differential Power Analysis // Proceeding of CRYPTO’99 .-Springer-Verlag.-1999.- PP.388-404.
6. Марковский А.П., Абабне О.А., Ияд Мохд Маджид Ахмад Шахрури. Способ защиты ключей алгоритма ГОСТ 28.147-89 от реконструкции анализом дина-

мики потребляемой мощности //Вісник національного технічного університету України ”КПІ”. Інформатика, управління та обчислювальна техніка. – 2007.- № 46.- С.128-138.

УДК 004.421.5

ЗОХРЕ КАРИМ ЗАДЕ.,
МЕЛЬНИК А.П.,
САМКОВСКИЙ К.С.

МЕТОД ПОСТРОЕНИЯ НЕЛИНЕЙНОГО ГЕНЕРАТОРА ДВОИЧНЫХ ПОСЛЕДОВАТЕЛЬНОСТЕЙ НА СДВИГОВОМ РЕГИСТРЕ

Статья посвящена исследованию проблемы технологии проектирования сдвиговых регистров с нелинейными функциями обратной связи, которые гарантируют период повторения 2n для n-разрядного сдвигового регистра и обеспечивают улучшение характеристик двоичных последовательностей, которые важны для эффективности защиты информации. Разработан комбинаторный метод получения нелинейных булевых функций, обеспечивающий 2n для n-разрядного сдвигового регистра. Доказано, что предложенный метод позволяет на порядок увеличить количество нелинейных функций обратной связи по сравнению с известными методами.

Paper is dedicated to a problem of the design techniques of Nonlinear Feedback Shift Register (NFSR) with nonlinear feedback Boolean function, which ensure the repeat cycle 2n for n-bit register and improve the sequences quality characteristics which have an impact on data protection efficiency. The combinatorial method for obtaining nonlinear feedback functions which ensure the repeat cycle 2n for n-bit shift register has been worked out. It has been proved that proposed method allowed to increase the number of obtained feedback nonlinear function on one order in compare to known methods.

Введение

Псевдослучайные двоичные последовательности (ПСДП) активно применяются в информационных технологиях с середины 50-х годов. В последние годы технология получения таких последовательностей переживает этап динамичного развития в связи с расширением области их практического использования. Так, ПСДП широко используются в современных теле коммуникационных технологиях, в частности, в системах беспроводной передачи цифровых данных. Однако наиболее востребованным для ПСДП стало их использование в системах защиты информации. Они применяются в качестве базового элемента одного из трех типов алгоритмов защиты информации - потоковых алгоритмов. Кроме того, ПСДП широко используются для генерации ключей симметричных алгоритмов защиты данных и псевдослучайных двоичных строк протоколов аутентификации удаленных пользователей интегрированных систем [1].

В современных условиях роста производительности вычислительных систем и возможностей объединения значительного числа компьютеров в

сеть для нарушения защиты, актуальной становится проблема адекватного повышения надежности защиты информации, в том числе, за счет совершенствования средств получения и использования ПСДП.

Защитные свойства псевдослучайных последовательностей и функциональных преобразований, в теоретическом плане, определяются невозможностью аналитического решения систем нелинейных булевых уравнений [2]. Именно это свойство булевых преобразований лежит в основе использования ПСДП и булевых функций для защиты информации. Поэтому, уровень защищенности данных с использованием ПСДП прямо зависит от нелинейности булевых функций, которые используются для генерации ПСДП. Исходя из этого, важным резервом повышения эффективности широкого класса средств защиты информации, в которых применяются ПСДП, является усовершенствование средств формирования таких последовательностей в направлении увеличения нелинейности булевых преобразований, которые они используют.

Рост быстродействия вычислительных систем и скорости передачи цифровых данных выдвигает все более жесткие требования к производительности средств защиты информации, в том числе, к генераторам ПСДП: их реализация не должна заметно сказываться на основных процессах обработки или передачи информации. В этом плане, наиболее эффективными являются генераторы ПСДП на основе сдвиговых регистров. Именно поэтому, генераторы ПСДП на основе сдвиговых регистров с линейными функциями обратной связи получили наибольшее распространение в современных системах защиты данных [1]. Однако, использование сдвиговых регистров с линейной функцией обратной связи не обеспечивает необходимый уровень защиты: для этого в схему генераторов ПСДП вводятся дополнительные нелинейные преобразователи. Это усложняет средства получения ПСДП и снижает их быстродействие.

Таким образом, на современном этапе развития технологии защиты информации в системах и сетях вычислительной техники актуальной является задача повышения эффективности средств генерации ПСДП, в том, числе на основе сдвиговых регистров с нелинейными функциями обратной связи.

Генераторы ПСДП на сдвиговых регистрах

Как отмечалось выше, в подавляющем большинстве, генераторы ПСДП для систем защиты информации строятся на основе сдвиговых регистров с линейными функциями обратной связи (LFSR Linear Feedback Shift Register). Если линейная функция обратной связи соответствует простому полиному на полях Галуа, то период повторения кода на n-разрядном сдвиговом регистре равен 2^n-1 [1]. При аппаратной реализации LFSR обеспечивают высокий темп генерации ПСДП, которые хотя имеют хорошие статистические характеристики, но элементы которых линейно

зависимы. Соответственно, такие ПСДП могут быть легко предсказаны путем анализа выборки длиной $2 \cdot n$ бит. Поэтому реальные генераторы на основе LFSR имеют в своем составе нелинейные преобразователи. Это усложняет схему генератора и замедляет его работу.

Наиболее естественным решением проблемы повышение эффективности генерации ПСДП является использование для этой цели сдвигового регистра с нелинейной функцией обратной связи (NFSR-Nonlinear Feedback Shift Register). Имея простую структуру, NFSR могут быть эффективно реализованы аппаратно, обеспечивают высокий темп генерации ПСДП. Нелинейный характер функции обратной связи сводит задачу предсказания последовательности к решению системы нелинейных булевых уравнений, которое принципиально не может быть выполнено аналитически [2].

Основной трудностью практического применения NFSR является сложность получения функции $f(x_1,x_2,\ldots,x_n)$ обратной связи, определенной на компонентах вектора $X=\{x_1,x_2,\ldots,x_n\}$ значений разрядов сдвигового регистра и обеспечивающей период повторения 2^n. Общее количество N_f таких функций определяется формулой [2]:

$$N_f(n) = 2^{2^{n-1}-n} \qquad (1)$$

Хотя, определяемое приведенной формулой (1) число N_f для имеющих место на практике значений n достаточно велико и на несколько порядков превышает число подходящих функций для LSFR, получение самих функций, обеспечивающих период повторения 2^n представляет собой сложную задачу [2]. Так, уже при n=10 доля подходящих функций составляет $2^{-522} \approx 10^{-174}$ от общего числа функций.

Поэтому к настоящему времени предложено ряд методов получения функций рассматриваемого класса. Практически все из них используют те или иные специфические свойства функций, обеспечивающих в качестве обратной связи максимальный период повторения кода на сдвиговом регистре. Текущее состояние разрядов сдвигового регистра длиной n бит характеризуется двоичным вектором X_w значений его разрядов, который соответствует числу w:

$$X_w = \{x_1^w, x_2^w, \ldots, x_{n-1}^w, x_n^w\},$$

$$\forall j \in \{1,\ldots,n\} : x_j^w \in \{0,1\},$$

$$w = \sum_{j=1}^{n} x_j^w \cdot 2^{j-1}$$

При сдвиге регистра, его младшему разряду x_1 присваивается значение функции обратной связи $f(x_1,x_2,\ldots,x_n)$, соответственно, новое значение вектора битовых значений регистра становится равным X_v. Значение вектора X_v и соответствующего ему кода v определяется следующим образом:

$$X_v = \{ f(X_w), x_1^w, ..., x_{n-2}^w, x_{n-1}^w \},$$

$$v = (2 \cdot w) \bmod 2^n + f(X_W)$$

Важнейшим свойством функций $f(x_1,x_2,\ldots,x_n)$ обратной связи, обеспечивающих полный период повторения кода на сдвиговом регистре является условие единственности перехода в каждое из состояний регистра:

$$f(x_1, x_2, ..., x_n) = 1 \oplus f(x_1 \oplus 1, x_2, ..., x_n) \qquad (2)$$

Основными критериями оценки методов получения нелинейных функций $f(x_1,x_2,\ldots,x_n)$ обратной связи сдвигового регистра являются:

- объем вычислительных ресурсов, затрачиваемых на получение функций рассматриваемого класса;
- количество функций, которые могут быть получены с использованием метода;
- нелинейность формируемых функций.

Наиболее простым методом получения нелинейных функций обратной связи NFSR является модификация линейной функции LFSR [1]. Очевидными недостатками метода является малая нелинейность получаемых с его помощью функций и малое их количество.

Малое количество функций можно получить и с использованием итерационного метода построения NFSR [2], сущность которого состоит в специальном порядка заполнения таблицы истинности функции обратной связи. Такой же недостаток присущ и методам, основанным на модификации заданной нелинейной функции обратной связи [2].

Более эффективный подход к получению функций для NFSR основан на использовании концепции объединения колец – кодов, получаемых при циклическом сдвиге. Методы, основанные на этой концепции, позволяют получать функции с высокой нелинейностью и имеют вычислительную сложность $O(n^2)$ [3]. Основными недостатками является сложность процедуры генерации колец и весьма узкий класс формируемых функций по отношению к общему количеству (1).

Таким образом, существующие методы не позволяют в достаточной для практики степени решать задачу получения большого числа функций для NFSR. Это имеет следствием то, что при нарушении защиты имеется реальная возможность перебора всех функций, которые могут быть получены с использованием известных методов и, тем самым, снижается эффективность реализуемой с использованием ПСДП защиты данных.

Целью работы является разработка метода получения нелинейных функций обратной связи для NFSR, обеспечивающего увеличение числа формируемых функций указанного класса по сравнению с известными методами.

Метод получения нелинейной функции обратной связи для NFSR

В основе предлагаемого комбинаторного метода получения булевой функции обратной связи сдвигового регистра с максимальным периодом повторения лежат специфические свойства таких функций.

Если функция $f(x_1, x_2,\ldots,x_n)$ обратной связи удовлетворяет условию (2), то каждому коду v на р**Ошибка! Закладка не определена.**егистре предшествует только один код w.

Для каждого кода $w\in\{0,\ldots,2^n\text{-}1\}$ сдвигового регистра можно указать пару $< v_0^w, v_1^w >$ кодов, которые могут образовываться при сдвиге регистра после кода w : $v_0^w = (2\cdot w) \bmod 2^n$ и $v_1^w = (2\cdot w) \bmod 2^n + 1$.

Для каждого кода $w\in\{0,\ldots,2^n\text{-}1\}$ сдвигового регистра существует в точности один сопряженный ему код u такой, что: $v_0^W = v_0^u$ и $v_1^W = v_1^u$.

Векторы состояний сдвигового регистра X_w и X_u, которые соответствуют сопряженным кодам w и u отличаются значениями старшего бита: $\forall i \in \{1,\ldots,n-1\} : x_i^w = x_i^u, x_n^w \neq x_n^u$. Согласно свойству (2) функция обратной связи на сопряженных векторах принимает противоположные значения: $f(X_w)\neq f(X_u)$. Обозначим вектор X_u, сопряженный X_w как X_w'.

Текущее состояние характеризуется множеством ϑ фрагментов: $\vartheta=\{P_1,P_2,\ldots,P_r\}$. Текущее значение количества фрагментов равно r. Каждый j-тый фрагмент характеризуется парой кодов $P_j = <X_j^s,X_j^f>$.

Предлагаемый алгоритм получения функции обратной связи, обеспечивающей максимальный период повторения кодов на сдвиговом регистре сводится к следующей последовательности операций:

1. Установить, что множество ϑ состоит из двух (r=2) фрагментов: $P_1=<X_1^s,X_1^f>, P_2=<X_2^s,X_2^f>$, причем $X_1^s=\{1,0,\ldots,0\}$, $X_1^f=\{0,\ldots,0,1\}$, $f(X_1^s)=0$; $f(0,\ldots,0)=1$, $X_2^s=\{0,1,\ldots,1\}$, $X_2^f=\{1,\ldots,1,0\}$, $f(X_2^s)=1$, $f(1,\ldots1,1) = 0$.

2. Произвольно выбирается один из r фрагментов множества ϑ. Пусть его номер равен j, $j\in\{1,\ldots,r\}$. Конечный код j-то фрагмента принимается в качестве текущего: $X_t = X_j^f$. Определяется код - X_t', сопряженный текущему коду X_t.

3. Выполняется поиск фрагмента, содержащего сопряженный код X_t' в качестве конечного. Если в результате поиска найден i-тый фрагмент, такой, что $X_i^f = X_t'$, то сопряженный фрагмент равен i и выполняется переход на пп.5.

4. В противном случае строится новый, (r+1)-й фрагмент, начальный и конечный код которого совпадает и состоит только из одного кода X_t' : $P_{r+1} = <X_t', X_t'>$; в этом случае – i:=r+1. Счетчик числа фрагментов увеличивается на единицу: r:=r+1.

5. Определяется пара кодов $<X_{n0},X_{n1}>$, в которые возможен переход из текущего кода X_t и сопряженного ему кода X_t': $X_{n0} = X_t<<1= X_t'<<1$, а X_{n1}

$=(X_t<<1)\oplus 1 = (X_t'<<1)\oplus 1$, где через $<<1$ обозначен логический сдвиг влево на один разряд с заполнением нулем.

6. Если $X_{n0} = X_j^s$ или $X_{n1} = X_i^s$, переход на пп.9.
7. Если $X_{n1} = X_j^s$ или $X_{n0} = X_i^s$, переход на пп.10.
8. Выполняется произвольно переход либо на пп.9, либо на пп.10.
9. Устанавливается $f(X_t) = 1$, $f(X_t')=0$. Если X_{n1} не совпадает с начальным кодом ни одного из фрагментов, то установить $X_j^f = X_{n1}$. Если X_{n1} совпадает с началом q-го фрагмента: $X_{n1} = X_q^s$, то отметить как удаленный q-тый фрагмент, заменив конечный код j-то фрагмента конечным кодом q-того фрагмента: $X_j^f = X_q^f$. Если X_{n0} не совпадает с начальным кодом ни одного из фрагментов, то установить $X_i^f = X_{n0}$. Если X_{n0} совпадает с началом g-го фрагмента: $X_{n0} = X_g^s$, то отметить как удаленный g-тый фрагмент, заменив конечный код i-то фрагмента конечным кодом g-того фрагмента: $X_i^f = X_g^f$. Переход на пп.11.
10. Устанавливается $f(X_t) = 0$, $f(X_t')=1$. Если X_{n0} не совпадает с начальным кодом ни одного из фрагментов, то установить $X_j^f = X_{n0}$. Если X_{n0} совпадает с началом q-го фрагмента: $X_{n0} = X_q^s$, то отметить как удаленный q-тый фрагмент, заменив конечный код j-то фрагмента конечным кодом q-го фрагмента: $X_j^f = X_q^f$. Если X_{n1} не совпадает с начальным кодом ни одного из фрагментов, то установить $X_i^f = X_{n1}$. Если X_{n1} совпадает с началом g-го фрагмента: $X_{n1} = X_g^s$, то отметить как удаленный g-тый фрагмент, заменив конечный код i-то фрагмента конечным кодом g-того фрагмента: $X_i^f = X_g^f$.
11. Удалить из ϑ все отмеченные к удалению фрагменты, изменив и[нумерацию и скорректировав значение r. Если $r>0$, то возврат на пп.2.

Предложенный алгоритм синтеза булевой нелинейной функции обратной связи, обеспечивающей максимальный период повторения иллюстрируется следующим примером.

Пусть $n=4$. В соответствии с пп.1 алгоритма, множество ϑ вначале состоит из двух фрагментов: $P_1 = <\{1000\},\{0001\}>$, $P_2 = <\{0111\},\{1110\}>$ значения функции обратной связи определяются на 4-х наборах: $f(1000)=0$; $f(0000)=1$; $f(1110)=1$; $f(1111)=0$.

В соответствии с пп.2 произвольно выбирается второй фрагмент множества ϑ, то есть j = 2. В качестве текущего принимается код $X_t = X_2^f = \{1110\}$. Сопряженный ему код $X_t' = \{0110\}$.

Код X_t' не совпадает с конечным кодом ни одного из 2-х фрагментов. Поэтому, в соответствии в пп.4 строится 3-й фрагмент $P_3 = <\{0110\},\{0110\}>$, $i=3$. Согласно пп.5 определяется пара кодов $X_{n0} = X_t<<1= \{1100\}$, $X_{n1} =(X_t<<1)\oplus 1 = \{1101\}$. Коды X_{n0} , X_{n1} не совпадают с конечными кодами j-го и i-го фрагментов, поэтому, согласно пп. 8 осуществляется переход, например, на пп.9. Соответственно, устанавливаются значения функций для X_t и X_t': $f(1110) = 1$, $f(0110)=0$. Коды X_{n1} и X_{n0} не

совпадают с начальным кодом ни одного из фрагментов множества ϑ, поэтому конечный код фрагмента P_2 заменяется кодом X_{n1}: $X_2^f = X_{n1}$={1101}, а конечный код фрагмента P_3 заменяется кодом X_{n0}: $X_3^f = X_{n0}$= {1100}.

Поскольку $r = 3 > 0$, то выполняется возврат на пп.2. Из множества ϑ={<{1000},{0001}>,<{0111},{1101}>,<{0110},{1100}>} произвольно выбирается, например, 3-й фрагмент, то есть j=3. Соответственно, $X_t = X_3^f$ = {1100}, X_t' = {0100}. Код X_t' не совпадает с конечным кодом ни одного из фрагментов, поэтому, согласно пп. 4 множество дополняется новым фрагментом P_4 = <{0100},{0100}> и i=3. Вычисляются коды X_{n0} = {1000}, X_{n1} = {1001}. Поскольку они не совпадают с начальными кодами фрагментов P_3 и P_4, то, согласно пп.8 выбор значения функций образной связи на наборе X_t осуществляется произвольно: например, по пп.10 f(1100)=0, f(0100)=1. Код X_{n0} совпадает с начальным кодом 1-го фрагмента (q=1). Согласно пп.10 указанный фрагмент помечается к удалению, а конечный код 3-го фрагмента замещается конечным кодом 1-го фрагмента: $X_3^f = X_1^f$ = {0001}. Код X_{n1} не совпадает с начальным кодом ни одного из 4-х фрагментов множества ϑ, поэтому X_{n1} замещает конечный код фрагмента P_4: $X_4^f = X_{n1}$ = {1001}. Из 4-х фрагментов пп.11 удаляется ранее отмеченный фрагмент P_1. Оставшиеся перенумеровываются, так, что после выполнения пп.11 множество ϑ состоит из 3-х фрагментов: ϑ={P_1,P_2,P_3}, P_1=<{0111},{1101}>, P_2= <{0110},{0001}>, P_3=<{0100},{1001}>. Поскольку $r = 3>0$, то осуществляется возврат на пп.2.

На третьем цикле в качестве текущего выбирается код X_t = {1101} , конечный в первом фрагменте P_1: j=1. Сопряженный код X_t'={0101} не равен ни одному из конечных, поэтому множество ϑ дополняется новым фрагментом P_4= <{0101},{0101}>, i=4. Образуемые коды X_{n0} = {1010}, X_{n1} = {1011} не совпадают с начальными кодами фрагментов P_1 и P_3, что определяет произвольный выбор значения функции на наборе X_t : f(1101)=1, f(0101)=0. Так как коды X_{n0}, X_{n1} не совпадают с начальным кодом ни одного из 4-х фрагментов множества ϑ, то X_{n1} заменяет конечный код фрагмента 1, а X_{n0} заменяет конечный код фрагмента 4. После этого P_1=<{0111},{1011}>, P_2= <{0110},{0001}>, P_3=<{0100},{1001}>, P_4=<{0101},{1010}>. Поскольку r= 4>0, то осуществляется возврат на пп.2.

На четвертом цикле в качестве текущего выбирается код X_t = {0001} , конечный во втором фрагменте P_2: j=2. Сопряженный код X_t'={1001} совпадает с конечным кодом фрагмента P_3, поэтому i=4. Образуемые коды X_{n0} = {0010}, X_{n1} = {0011} не совпадают с начальными кодами фрагментов P_2 и P_3, что определяет произвольный выбор значения функции на наборе X_t: например в соответствии с пп.10: f(0001)=0, f(1001)=1. Так как коды X_{n0}, X_{n1} не совпадают с начальным кодом ни одного из 4-х фрагментов множества ϑ, то X_{n1} заменяет конечный код P_2, а X_{n0} заменяет

конечный код фрагмента P_3. В конце 4-го цикла ϑ состоит из: P_1=<{0111}, {1011}>, P_2= <{0110},{0010}>, P_3=<{0100},{0011}>, P_4=<{0101}, {1010}>. Так как r=4>0, то осуществляется возврат на пп.2.

На пятом цикле в качестве произвольно выбираемого текущего кода X_t выделяется конечный код P_4: X_t = {1010}, j=4. Сопряженный код X_t'={0010} совпадает с конечным кодом фрагмента P_2, поэтому i=2. Формируются коды X_{n0} = {0100} и X_{n1} = {0101}. Второй из них совпадает с начальным кодом фрагмента P_4, X_{n1}= X_i^s, то есть выполняется условие пп.7, соответственно осуществляется переход на пп.10, в рамках которого устанавливается f(1010)=0, f(0010)=1. Код X_{n0} совпадает с началом 3-го фрагмента (q=3): X_{n0} = X_3^s, поэтому фрагмент P_3 отмечается к удалению, конечный код P_4 заменяется конечным кодом 3-го фрагмента - {0011}: P_4=<{0101},{0011}>. Код X_{n1} совпадает с началом 4-го фрагмента (g=4) : X_{n1} = X_4^s ={0101}. Поэтому, в соответствием с пп.10, фрагмент P_4 отмечается к удалению, а конечный код 2-го фрагмента заменяется конечным кодом 4-го фрагмента, так, что: P_2= <{0110},{0011}>. При выполнении пп.11 удаляются отмеченные 3-й и 4-й фрагменты, после чего множество ϑ состоит только из 2 фрагментов: P_1=<{0111},{1011}> и P_2= <{0110}, {0011}>. Так как r = 2>0, то осуществляется возврат на пп.2.

На шестом цикле в качестве текущего выбирается код X_t = {1011}, конечный во втором фрагменте P_1: j=1. Сопряженный код X_t'={0011} совпадает с конечным кодом фрагмента P_2, поэтому i=2. Формируются коды X_{n0} = {0110} и X_{n1} = {0111}. Второй из них совпадает с начальным кодом фрагмента P_1, X_{n1}= X_i^s, то есть выполняется условие пп.7, соответственно осуществляется переход на пп.10, устанавливается f(1011)=0, f(0111)=1. Код X_{n0} совпадает с началом 2-го фрагмента (q=2): X_{n0} = X_2^s, поэтому фрагмент P_2 отмечается к удалению, конечный код P_1 заменяется конечным кодом 2-го фрагмента – {0011}: P_1=<{0111},{0011}>. Код X_{n1} совпадает с началом 1-го фрагмента (g=1): X_{n1} = X_1^s ={0111}. Поэтому, в соответствием с пп.10, фрагмент P_1 отмечается к удалению. При выполнении пп.11 удаляются отмеченные 2-й и 1-й фрагменты, после чего множество ϑ становится пустым. Так как r = 0, то работа алгоритма закончена. Построенная функция в результате функция обратной связи f(X) приведена в таблице.

Таблица истинности функции f(X) обратной связи, обеспечивающей период 2^4 повторения кода на сдвиговом регистре

$x_1x_2x_3x_4$	$f(X)$	$x_1x_2x_3x_4$	$f(X)$	$x_1x_2x_3x_4$	$f(X)$	$x_1x_2x_3x_4$	$f(X)$
0 0 0 0	1	0 1 0 0	1	1 0 0 0	0	1 1 0 0	0
0 0 0 1	0	0 1 0 1	0	1 0 0 1	1	1 1 0 1	1
0 0 1 0	1	0 1 1 0	0	1 0 1 0	0	1 1 1 0	1
0 0 1 1	1	0 1 1 1	1	1 0 1 1	0	1 1 1 1	0

Выводы

В результате проведенных исследований предложен метод построения генераторов ПСДП на NFSR, основой которого является комбинаторный подход к получению нелинейной функции обратной связи, обеспечивающий максимальный период повторения кода на сдвиговом регистре. Фактически разработанный подход представляет собой способ уменьшения перебора всех функций, указанного класса на основе их специфических свойств.

Проведенные экспериментальные исследования показали, что предложенный метод эффективен для относительно небольшой длины n сдвигового регистра: $n \leq 20$. Основным достоинством комбинаторного подхода, лежащего в основе разработанного метода является значительное расширение числа получаемых функций обратной связи, обеспечивающих максимальный период повторения ПСДП. Метод может быть использован при проектировании высокоскоростных средств защиты данных при их передаче по беспроводным линиям.

Список использованной литературы

1. Иванов М.А., Чугунков И.В. Теория, применение и оценка качества генераторов псевдослучайных последовательностей. М.: КУДИЦ-ОБРАЗ.- 2003 – 260 с.
2. Golomb S.W. Shift register sequences. Laguna Hill. California: Aegean Park Press.- 1982.-324 p.
3. Марковський О.П., Зохре Карім Заде Сейфолах, Гурін В.Є. Ефективний метод побудови нелінійних генераторів для телекомунікаційних систем //Электроника и связь. Тематический выпуск ”Проблемы электроники” ч.3. – ПЦ ”Аверс” - 2007.- С.87-89.

УДК 004.074.32

МНАЦАКАНОВ А.В.,
КУЗНЕЦОВ А.А.

К ПРОБЛЕМЕ ХЕШ-АДРЕСАЦИИ БЕЗ КОЛЛИЗИЙ ПОСТОЯННОГО МАССИВА КЛЮЧЕЙ

В статье исследуется проблема получения хеш-адресации без коллизий заданного постоянного массива ключей. Разработан алгоритм построения совершенной хеш-адресации в виде иерархической системы разделяющих булевых функций, позволяющий значительно сократить объем вычислительных ресурсов, необходимых для формирования хеш-преобразования, не порождающего коллизий при заданных ограничениях на сложность хеш-функции. Проведенный анализ показал преимущества предложенного алгоритма получения совершенной хеш-адресации, по сравнению с известными алгоритмами решения этой проблемы.

In article the problem of designing of the perfect hash-addressing for a constant set of keys is investigated. The algorithm of designing of the perfect hash-addressing as hierarchical system of Boolean functions is developed. The algorithm allows to significant reduce of volume of computing resources which is necessary for design of the perfect hash-transformation with the prescribed limit of hash-function complexity. The analysis has shown the advantages of proposed perfect hash-addressing designing algorithm in comparison to known algorithms of this problem decision.

Введение

Операции поиска по ключу и сортировки данных являются одними из базовых в информационных технологиях. Существует значительное число важных практических применений, в которых удельный вес этих операций составляет до 80% [2].

Прогрессирующие возможности вычислительных систем и информационная интеграция обуславливают динамичный рост объемов массивов ключей, на которых выполняется поиск [2]. Непрерывное расширение сферы использования информационных технологий имеет следствием ужесточение требований к оперативности поиска. В частности, большая часть систем распознавания зрительных и звуковых образов, в которых активно используется поиск по ключу, работают в режиме реального времени. В близком режиме работает значительная часть современных баз данных. Расширение информационной интеграции привело к росту числа информационных систем коллективного доступа, специфика которых предъявляет повышенные требования к скорости поиска.

Анализ показывает, что в современных условиях роста объемов поисковых массивов и ужесточения требований к оперативности доступа, эффективность использования двоичного поиска и B-деревьев сущест-

венно снижается в силу зависимости времени поиска от объема массива ключей.

Это требует расширения использования альтернативных технологий поиска по ключу, таких как хеш-адресация и генетические алгоритмы [5].

Самым быстрым методом поиска по ключу является ассоциативный поиск, который может быть реализован как аппаратно (ассоциативные запоминающие устройства - АЗУ), так и программно (хеш-память). Важным достоинством ассоциативного поиска является то, что время поиска не зависит от объема массива ключей. Ввиду значительной стоимости выпускаемых микросхем АЗУ их практическое использование ограничено. Поэтому в современных условиях для большинства применений более эффективным представляется использование хеш-адресации.

Существует класс практических применений поиска по ключу, для которых массив ключей является постоянным или квазипостоянным (интенсивность операций поиска по ключу на несколько порядков превосходит интенсивность операций изменения поискового массива). К их числу относится большинство систем распознавания образов, электронного перевода и аутентификации удаленных пользователей, заметная часть баз данных.

При хеш-поиске в постоянных массивах ключей возможно получение однозначного хеш-преобразования, исключающего возникновение коллизий. В литературе такой метод хеш-поиска получил название совершенной хеш-адресации (perfect hash-addressing)[1].

Основным достоинством совершенной хеш-адресации является отсутствие коллизий, то есть, время поиска по ключу определяется временем однократного обращения к памяти. Это позволяет достичь предельно-высокой скорости поиска, независимо от объема поискового массива [1].

Задача получения хеш-преобразования, не порождающего коллизий для заданного массива ключей допускает множество решений, однако, при имеющих место на практике ограничениях на объем вычислительных ресурсов, используемых при совершенной хеш-адресации, эта задача имеет экспоненциальную сложность [1].

В связи с этим, актуальной является задача построения алгоритма, позволяющего значительно сократить объем вычислительных ресурсов, необходимых для формирования хеш-преобразования, не порождающего коллизий и удовлетворяющего заданным ограничениям на сложность функциональных преобразований.

Основные определения и постановка задачи

Хеш-преобразованием, не порождающем коллизий для заданного постоянного набора ключей, называется однозначная функциональная зависимость $F(\Omega) = \Theta$ (где – Ω множество состоящее из m n-разрядных клю-

чей и Θ – множество r-разрядных адресов хеш-памяти), причем $F(k_i) \neq F(k_j)$ при любых $k_{i,}k_j \notin \Omega$ и $i \neq j$ [2].

При имеющих место на практике ограничениях на сложность функциональных преобразований и объем используемой памяти формирование хеш-преобразования без коллизий (ХБК) требует значительных затрат вычислительных ресурсов. В связи с этим, основным критерием эффективности ХБК является временная T_c сложность его получения при заданном постоянном или наборе Ω ключей.

Основным недостатком хеш-адресации является избыточность используемой памяти, что значительно ограничивает сферу использования хеш-памяти. Избыточность использования памяти характеризуется коэффициентом α загрузки, который определяется отношением количества m заданных ключей к числу адресов хеш-памяти:

$$\alpha = \frac{m}{2^r} \qquad (1)$$

К настоящему времени предложено ряд методов, позволяющих, для заданного набора ключей, получить хеш-преобразование не порождающее коллизий [4, 5, 6]. Большинство из них основаны на подборе хеш-преобразования и ориентированы на программную реализацию хеш-адресации.

Недостатками этих методов являются высокая вычислительная сложность и возможность возникновения тупиковых ситуаций. Большая вычислительная сложность существующих методов ограничивает область их использования сравнительно небольшими массивами ключей [6]. Поэтому актуальной является проблема разработки подходов, требующих меньших затрат вычислительных ресурсов.

Решение этой проблемы может быть найдено в рамках использования многоуровневого направленного формирования хеш-преобразования без коллизий (ХБК).

Методика построения хеш-преобразования без коллизий формированием системы разделяющих булевых функций

Решение задачи формирования ХБК, путем сокращения разрядности ключей, можно разделить на два этапа. При этом, первом этапе производится сокращение разрядности ключей за счет удаления малоинформативных разрядов ключей, которые не влияют на их различение, а на втором, производится выделение групп разрядов по которым осуществляется перекодирование с использованием табличных преобразователей.

Предположим, что вероятность появления нуля и единицы в коде ключа одинакова, тогда вероятность P_p того, что будет выполняться условие $k_i \neq k_j, i \neq j$ при любом $k_i, k_j \in \Omega$, будет равна:

$$P_p = \frac{1}{\exp(m)} \bullet \left(\frac{2^n}{2^n - m} \right)^{2^n - m} \bullet \sqrt{\frac{2^n}{2^n - m}} \quad (2)$$

Из выражения (2) можно определить минимальную разрядность n', которая может быть достигнута на первом этапе формирования ХБК, при заданном уровне вероятности P_p различимости ключей.

Предлагаемый алгоритм сокращения разрядности ключей, путем исключения малозначащих разрядов включает в себя две стадии:

На первой стадии производится ранжирования разрядов по невозрастанию их значимости для различимости ключей. При этом используется интегральная оценка $W(j)$ значимости j-го разряда ключа, которая вычисляется по формуле:

$$W(j) = h(j) + \frac{1}{m} \cdot \sum_{l=j+1}^{m} K(j,l) \quad (3)$$

где $h(j)$ – частота равенства единицы j-го разряда ключа и $K(j,l)$ – коэффициент корреляции j-го и l-го разрядов ключевого слова, которые в свою очередь могут выть определены по следующим формулам [6]:

$$h(j) = \frac{1}{m} \cdot \sum_{i=1}^{m} x_{ij},\ j \in \{1,...,n\} \quad (4)$$

$$K(j,l) = \frac{1}{m} \cdot \sum_{i=1}^{m} (1 \oplus x_{ij} \oplus x_{lj}),\ j,l \in \{1,...,n\},\ j \neq l \quad (5)$$

Вторая стадия состоит в непосредственном исключении малозначащих разрядов ключей:

1. Текущим разрядом t принимается разряд, имеющий минимальное значение интегральной оценки значимости: $t = n$.
2. Разряд t исключается из всех кодов ключей и выполняется проверка условия различимости ключей. Если условие выполняется и $t > 1$, то при $t = t - 1$ осуществляется переход на п. 5. В случае невыполнения условия различимости разрядов осуществляется переход на п. 3.
3. Из кодов всех ключей исключается разряд d, который имеет максимальную корреляцию с разрядом t: $d = \arg\max K(t, j)$.
4. Если выполняется условие различимости ключей, то разряд d окончательно исключается из отсортированного списка разрядов ключа и при $t = t - 2$ осуществляется переход на п. 5. В случае невыполнения условия различимости ключей, выполняется переход к п. 5, при $t = t - 1$.
5. Если t ≥ 1 то осуществляется возврат на п. 2, иначе конец.

После выполнения первого этапа формирования ХБК будет получено упорядоченное по $W(j)$ множество Ξ номеров разрядов ключей, которые не могут быть исключены без нарушения условия различимости.

На втором этапе формирования ХБК, осуществляется построение систем разделяющих булевых функций, которые выполняют перекодирование множества Ω' сокращенных ключей в множество Θ, содержащее ключи (хеш-адреса) меньшей разрядности. В большинстве случаев системы разделяющих булевых функций реализуется таблично, так, что ограничением на число разрядов, составляющих множество Θ является объем памяти – V, выделяемый под табличный преобразователь.

Исходя из полученного на первом этапе формирования ХБК множества Ξ состоящего из q разрядов (численное значение q может быть определено из (1)), формирование системы разделяющих булевых функций состоит в следующем:

1. Из множества Ξ выбирается разряд *d* такой, что h(*d*) наиболее близка к 0.5. Разряд *d,* в свою очередь, делит множество Ω ключей на два подмножества ϑ_0 (множество, состоящее из ключей в которых разряд d равен 0) и ϑ_1 (множество, состоящее из ключей в которых разряд d равен 1). Множество, содержащее наибольшее число ключей выбирается текущим G: $G = \max(\vartheta_0, \vartheta_1)$. Далее разряд d исключается из множества Ξ и добавляется к множеству Φ. При этом счетчик t подмножеств устанавливается равным 2-м.
2. На множестве Ξ выбирается разряд d, который делит G на два подмножества, с минимальной разностью количества ключей.
3. Проверяется условие:

$$\forall \vartheta_j,\ j = 1,...,2t,\ \vartheta_j \notin G : \left|\vartheta_j\right| \le \frac{n}{2 \cdot (t+1)}, \qquad \textbf{(6)}$$

4. где $\vartheta_1, \vartheta_2, ..., \vartheta_{2t} \not\subset G$ – подмножества на которые множество Ω ключей делится разрядами множества Φ и разрядом d. Если условие (6) выполняются, то разряд d исключается из множества Ξ и добавляется к множеству Φ. При этом $t = t \cdot 2$ и в качестве G выбирается одно из подмножеств $\vartheta_0, \vartheta_1, ..., \vartheta_t$ содержащее максимальное число элементов.
5. Если $t < V$, то осуществляется переход на п.2, иначе находится такое наибольшее число g, для которого количество ключей в каждом из подмножеств $\vartheta_0, \vartheta_1, ..., \vartheta_t$ меньше 2^g.
6. Осуществляется построение $\log_2 t$ -разрядной таблицы, состоящей из *t* строк, выбор которых осуществляется в соответствии со значениями разрядов множества Φ. Каждая строка полученной таблицы содержит g-разрядные коды замещения $\log_2 t$ разрядов ключа, являющиеся фактически старшими разрядами хеш-адреса. Таким образом, исполь-

зование этой таблицы позволяет осуществить перекодирование $\log_2 t$ разрядов ключа в g разрядов. Кроме того, каждая из строк таблицы содержит данные для проведения дальнейшего сжатия каждого из подмножеств $\vartheta_0, \vartheta_1, ..., \vartheta_t$.

Приведенную методику можно повторно применять к каждому из подмножеств $\vartheta_0, \vartheta_1, ..., \vartheta_t$ в отдельности, пока в каждом из разделяемых подмножеств не останется один ключ.

Для ускорения нахождения табличного представления разделяющих функций первые две операции предложенной методики можно модифицировать следующим образом:

1. Из множества Ξ выбирается любой q-тый разряд, при этом механизм выбора множества G не изменяется.
2. В качестве текущего d, выбирается разряд $d \in \Xi, d \neq q$ с наименьшим коэффициентом корреляции, то есть $K(d, q)$.

В остальном модифицированная методика построения разделяющих функций полностью повторяет вышеописанную.

Полученный набор таблиц перекодирования представляет собой иерархическое хеш-преобразование.

Построенная, в результате использования предложенного алгоритма, система разделяющих булевых функций может быть реализована средствами FPGA, что позволяющего значительно повысить быстродействие хеш-преобразователя в сравнении с программной реализацией.

Оценка основных характеристик и сравнительный анализ

Операция исключения разрядов требует многократного формирования подмножеств $\vartheta_0, \vartheta_1, ..., \vartheta_t$. Если максимально допустимая разрядность таблицы перекодирования равна u, то общее число N_o операций сравнения, требующихся для формирования системы разделяющих функций [6]:

$$N_o = \sum_{j=u}^{n} j^2 \qquad (7)$$

Таблица перекодирования будет выполнять разбиение множества Ω на 2^u подмножеств, каждое из которых содержит $n/2^u$ ключей. В связи с этим, основной является задача выбора множества Φ, включающего u из n разрядов и формирования на выбранном множестве булевой функции $\varphi(x_1,\ldots,x_u)$ от u переменных. В [3] показано, что существует C_n^u способов выбора подмножества Φ.

В общем случае, число функций от u переменных равно 2^{2^u}, однако ввиду того, что эти функции должны разделять множество m ключей на

2^u равновеликих подмножества, то общее число функций, удовлетворяющих этому условию составит $2^u!$. Таким образом, на первом уровне иерархии ХБК, верхняя граница общего числа вариантов выбора разделяющих функции составит $C_n^u \cdot 2^u!$.

Технологически формирование системы разделяющих функций в общем случае состоит в выборе множество из u разрядов такого, что число неразличимых ключей каждого вида было бы меньше $n/2^u$.

В данном случае важной является задача выбора разрядности функций u. С одной стороны, численное значение u определяется ограничением на объемом памяти, с другой – достаточно, с большой вероятностью гарантировать, что приемлемое множество может быть найдено.

Поэтому одним из основных критериев эффективности предложенной методики построения ХБК является оценка вероятности P_e того, что на множестве из u разрядов ключа может быть построена система булевых разделяющих функций. Вероятности P_c того, что совпадут h ключей из m, при использовании u – разрядного подмножества [3]:

$$P_c = C_n^h \cdot (p_e{}^2 + (1-p_e)^2)^{h \cdot u} (1 - p_e{}^2 - (1-p_t)^2)^{u \cdot (n-h)} \quad (8)$$

где p_e – вероятность того, что разряд ключа равен единице (на практике вместо p_e целесообразным представляется использование среднего значения частоты единиц в массиве ключей).

Тогда, среднее число ξ совпавших ключей на множестве из выделенных разрядов u составит:

$$\xi = n \cdot (p_e^2 + (1-p_e)^2)^u \quad (9)$$

Для достаточно быстрого поиска подходящего множества из u разрядов ключа, необходимо и достаточно чтобы выполнялось условие $\xi << m/2^u$. Согласно [5] это условие трансформируется к виду:

$$u > \frac{\ln(n)}{\ln\left(\dfrac{2}{p_e{}^2 + (1-p_e)^2}\right)}. \quad (10)$$

Например, при $p_e = 0.6$, для того, чтобы выполнялось условие разделимости массива разрядности $n = 1000$ по u разрядам на 2^u подмножества необходимо, чтобы $u > 5.1$.

Согласно [5], среднее число T_e проб, необходимых для разделения Ω на 2^u подмножеств так, чтобы не существовало более $n/2^u$ одинаковых ключей составляет:

$$T_e = \frac{1}{\Phi\left(\frac{n/2^u - n\cdot(p_e^2 + (1-p_e)^2)^u}{\sqrt{n\cdot(p_e^2+(1-p_e)^2)^u\cdot(1-(p_e^2+(1-p_e)^2)^u)}}\right)} \quad (11)$$

где Ф() – функция Лапласа. Анализ (11) показывает, что для практически значимых значений n и p_e, значение u всегда меньше границы, определяемой техническими возможностями.

Пусть $\{u_1, u_1, ..., u_k\}$ – разрядность таблиц перекодирования на каждом из k уровней иерархии хеш-преобразования тогда, при максимальном значении коэффициента α загрузки памяти, должно выполняться условие [5]: $\sum_{i=1}^{k} u_i = \log_2 n$. Учитывая, что общий объем памяти V необходимый для построения табличного преобразователя равен $V = \sum_{i=1}^{k}\prod_{j=1}^{i} 2^{u_j}$, получим, что, при использовании предложенной методики, максимально достижимое значение коэффициента α загрузки составит:

$$\alpha = \frac{n\cdot m}{V\cdot\log_2 n} \quad (12)$$

Проведенный анализ характеристик предложенной методики формирования ХБК показал, что при большом числе ключей m ($m > 300$), скорость построения ХБК существенно превосходит известные алгоритмы Чанга и Сичели [3], основанные на случайном формировании таблиц перекодирования. При увеличении числа m ключей преимущество предложенной методики в скорости становится более ощутимой. Вместе с тем, значительное повышение скорости достигается за счет увеличенного расхода памяти на хранение таблиц перекодирования.

В отличии от известных методов, которые не гарантируют нахождения ХБК для любого заданного массива ключей, предложенный алгоритм позволяет получить решение для любого заданного массива ключей за конечное число шагов.

Выводы

В результате проведенных исследований, направленных на повышение эффективности хеш-адресации без коллизий постоянных массивов ключей был предложен алгоритм формирования многоуровневого хеш-преобразования в виде иерархической системы разделяющих булевых функций. Алгоритм предполагает формирования хеш-преобразования в виде системы булевых функций и позволяет существенно сократить (на 50%) время их получения по сравнению с известными подходами [4,6].

В отличие от известных методов формирования хеш-преобразования без коллизий предложенный алгоритм позволяет получить решение для любого заданного массива ключей за конечное число шагов.

Использование предложенного алгоритма позволяет сократить сроки разработки специализированных систем обработки информации с большим удельным весом операций поиска в постоянных массивах ключей или время перенастройки для хеш-поиска в квазипостоянных массивах ключей.

Ключевые слова: perfect hash-addressing, perfect hash-memory, database management systems, Boolean functions

Список использованной 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 Національного технічного університету України „КПИ”. Інформатика, управління та обчислювальна техніка. К., ТОО „ВЕК+” 1998, - №31. С.94-108.
3. Berman F., Bock M.E., Dittert E., O`Donnel M.J., Plank D. Collections of function of perfect hashing // SIAM Journal Computers. – 1986, - Vol. 15, №2, - P. 604-618.
4. Jagannathan R. Optimal partial-match hashing design // ORSA Journal of Computing. – 1991, - Vol.3, №2, - P.86-91.
5. Ningping Sun, Ryozo Nakamura, Nonbing Zhu, Akiro Tada, Wenling Sun. An analysis of average search cost of external hashing with separate chain.// Processing of 7-th WSEAS International Conference on Circuits, Systems, Communications and Computers (CSCC-2003).- 2003,-P. 315-324.
6. Ramakrishna M.V., Bannai Y. Direct perfect hashing function of external files. // Journal of Database Administration. – 1991, - Vol.2, №1, - P.19-28.

УДК 004.052.42

МАРКОВСКИЙ А.П.,
ТУРЧЕНКО Ю.А.,
КОВАЛЬЧУК А.В.

ПОВЫШЕНИЕ ЭФФЕКТИВНОСТИ ОБНАРУЖЕНИЯ ОШИБОК ПЕРЕДАЧИ ДАННЫХ В ЛИНИЯХ С АМПЛИТУДНО-ИМПУЛЬСНОЙ МОДУЛЯЦИЕЙ

В статье предлагаются подходы к повышению эффективности обнаружения ошибок в линиях с амплитудно-импульсной модуляцией. Предложенные подходы основаны на учете специфических особенностей ошибок, которые возникают в таких линиях. Первый подход состоит в модификации CRC для линий с амплитудно-фазовой модуляцией. Второй подход основан на использовании взвешенных контрольных сумм. Разработан способ формирования компонент взвешенной контрольной суммы. Показано, что предложенная модификация контрольной суммы обеспечивает большую надежность контроля ошибок передачи данных в линиях с амплитудно-импульсной модуляцией по сравнению с CRC.

In paper an approaches for increasing the effectiveness of error detection in computer network channels with Pulse-Amplitude Modulation (PAM) has been proposed. Proposed approaches are based on consideration of special feature of errors which are appearances in line with PAM. The first approach consist of CRC modification specifically for line with PAM. The second approach is base of weighted checksums using. The way for checksum components coding has been developed. It has been shown that proposed checksum modification ensure superior digital data control transformation reliability for channels with PAM in compare to CRC.

Введение

Одним из определяющих факторов прогресса во всех областях человеческой деятельности является развитие информационной интеграции на основе компьютерных технологий и систем телекоммуникаций. Важной проблемой усовершенствования технологий информационного обмена является обеспечение высокой надежности передачи данных в шинах вычислительных систем и линиях компьютерных сетей. В современных условиях актуальность проблемы обеспечения высокой достоверности передачи цифровых данных возрастает. Это обусловлено рядом факторов, наиболее значимым из которых является повышение скорости передачи и, соответственно, увеличение негативного воздействия межсигнальной интерференции. На увеличение числа ошибок, вызванных внешними электромагнитными полями, оказывает влияние резкий рост их интенсивности, обусловленный динамичным расширением использования средств мобильной связи и беспроводных линий передачи данных компьютерных сетей. С другой стороны, продолжается процесс расширения исполь-

зования информационных технологий во всех сферах человеческой деятельности, включая те, которые связаны с техногенным риском, что ужесточает требования, предъявляемые к надежности всех компонент вычислительных систем и компьютерных сетей, включая средства передачи цифровых данных. Принимая во внимание рост объемов передаваемой в вычислительных системах и сетях информации, особенно важным является повышение надежности процессов передачи путем обеспечения гарантированного обнаружения наиболее часто встречаемых ошибок. Таким образом, поступательное развитие техники передачи данных требует адекватного совершенствования средств обеспечения высокой достоверности передачи данных в системах и сетях вычислительной техники.

Для повышения пропускной способности, в компьютерных сетях широко используются линии со спектральной и амплитудной модуляцией цифровых данных [1]. Основной особенностью такой модуляции является то, что передача группы битов контролируемого блока осуществляется одним канальным сигналом. В частности, амплитудно-импульсная модуляция цифровых данных применяется в телефонных и кабельных линиях компьютерных сетей [1,2]. Обнаружение ошибок в таких линиях имеет специфические особенности, которые не в полной мере учитываются существующими средствами обнаружения ошибок. Исходя из этого, важной и актуальной задачей представляется анализ особенностей возникновения ошибок в линиях с модуляцией, а также разработка способов повышение эффективности средств контроля за счет их адаптации к выявленным особенностям.

Непрерывный рост скоростей передачи цифровых данных в современных компьютерных системах с одной стороны имеет следствием увеличение числа ошибок, вызванных межсимвольной интерференцией, а с другой – диктует жесткие требования к производительности средств контроля ошибок: она должна быть соизмеримой с пропускной способностью канала [1]. Эти обстоятельства определяют необходимость радикального повышения надежности средств контроля ошибок, обладающих повышенным быстродействием и допускающих распараллеливание при реализации аппаратными средствами. К таким средствам относится контрольная сумма, надежность обнаружения ошибок с использованием которой уступает циклическим кодам, однако, в отличие от последних может вычисляться параллельно [2].

Таким образом, особенности современного этапа развития и использования компьютерных технологий обуславливают актуальность и практическую важность разработки новых и совершенствования известных средств обеспечения надежности передачи и хранения данных в компьютерных системах.

Анализ современного состояния проблемы обеспечения надежности передачи данных в линиях с амплитудно-импульсной модуляцией

Учитывая относительно небольшую интенсивность возникновения ошибок передачи данных в шинах вычислительных систем и проводных линиях компьютерных сетей, действующими протоколами регламентируется использование технологии ARQ (Automatic Repeat Request), которая предусматривает при обнаружении ошибок повторную передачу контролируемого блока данных. Наиболее распространенными средствами обнаружения ошибок передачи цифровой информации в компьютерных системах и сетях являются циклические избыточные коды (CRC-Cyclic Redundancy Check), и контрольные суммы (CS-Check Sum). CRC с образующим полиномом степени n позволяет обнаруживать все ошибки нечетной кратности, все ошибки кратности 2, а также пачки ошибок, длина которых не превышает n [3]. CS позволяет гарантированно обнаруживать только ошибки нечетной кратности, однако по сравнению с CRC контроль с использованием CS выполняется намного проще и быстрее.

В низкочастотных линиях передачи цифровых данных компьютерных сетей для повышения пропускной способности достаточно части используется амплитудно-импульсная модуляция (*PAM* – Pulse-Amplitude Modulation). При этом виде модуляции цифровых данных группа из k смежных бит, называемая кодовой группой $X=\{x_1,x_2,\ldots,x_k\}$, $\forall j\in\{1,\ldots,k\}:x_j\in\{0,1\}$ модулируется одним канальным сигналом, который может принимать 2^k уровней: от 0 до $(2^k-1)\cdot s$, где s-шаг дискретизации амплитуды канального сигнала.

При искажении канального сигнала *PAM* принципиально могут быть искажены k битов, образующих кодовую группу. Однако согласно [1] ошибки, возникающие в таких линиях имеют специфические особенности, состоящие в следующем:

- ошибки передачи канальных сигналов в сетях практически независимы и зависимость вероятности их появления от кратности описывается биномиальным законом;
- при возникновении ошибки передачи канального сигнала вероятности искажений отдельных битов k-разрядной кодовой группы, модулируемой одним канальным сигналом неодинаковы.

Действительно, если уровень $U(X)$ канального сигнала при амплитудно-импульсной модуляции определяется двоичным числом, соответствующим двоичному значению кодовой группы $X=\{x_1,x_2,\ldots,x_k\}$:

$$U(X)=s\cdot\sum_{j=1}^{k}x_j\cdot 2^{j-1} \qquad (1)$$

то, согласно [1] при возникновении ошибки передачи канального сигнала, обусловленного гауссовым шумом, наиболее вероятно искажение этого сигнала на величину одного шага s дискретизации амплитуды. Определим

эту вероятность как p_1 (через p_0 обозначим вероятность того, что канальный сигнал будет передан правильно). Существенно менее вероятным является изменение под воздействием гауссова шума канального сигнала на 2 шага дискретизации (обозначим эту вероятность через p_2). Аналогично, обозначим через p_i вероятность изменения, вследствие гауссова шума, канального сигнала на i-шагов дискретизации). Согласно данным [1] на практике выполняется неравенство:

$$p_0 >> p_1 >> p_2 >> p_3 >> .... >> p_{2^k-1}, \qquad (2)$$

причем отличие значений вероятностей, входящих в неравенство (2) составляет несколько порядков.

При возникновении ошибок передачи канальных сигналов, вызванных гауссовым шумом, меняются биты кодовой группы X, соответствующей искажаемому канальному сигналу. Важным моментом при этом является то, что вероятности искажения k битов, составляющих кодовую группу X неодинаковы. Так, при наиболее вероятном изменении канального сигнала на один шаг дискретизации младший бит x_1 меняется всегда, бит x_2 искажается с вероятностью 2^{-1}, бит x_3 изменяет свое значение с вероятностью 2^{-2} и так далее. При изменении под воздействием гауссова шума канального сигнала на два шага дискретизации безусловно меняется второй бит x_2 кодовой группы, в то время как третий бит x_3 меняется с вероятностью 2^{-1}. При изменении амплитуды канального сигнала на величину $3 \cdot s$, безусловно меняются младший $-x_1$ и следующий на ним – x_2 биты кодовой группы. Исходя из приведенных рассуждений ясно, что старший бит $-x_k$ искажается только при изменении в результате ошибки амплитуды канального сигнала на $2^{k-1} \cdot s$. С учетом вероятностей $p_1, p_2, \ldots, p_3$ изменения амплитуды канального сигнала под воздействием гауссова шума на соответствующее число шагов дискретизации, вероятности $q_1, q_2, \ldots q_k$ искажения битов кодовой группы при возникновении ошибки передачи канального сигнала могут записаны в следующем виде:

$$q_1 = p_1 + p_3 + p_5 + p_7 + ... \qquad (3)$$

$$q_2 = \frac{1}{2} \cdot p_1 + p_2 + p_3 + \frac{1}{2} \cdot p_5 + p_6 + p_7 + ...$$

$$q_3 = \frac{1}{4} \cdot p_1 + \frac{1}{2} \cdot (p_2 + p_3) + p_4 + p_5 + p_6 + p_7 + \frac{1}{4} \cdot p_9 + ...$$

$$q_4 = \frac{1}{8} \cdot p_1 + \frac{1}{4}(p_2 + p_3) + \frac{1}{2} \cdot (p_4 + p_5 + p_6 + p_7) + p_8 + ...$$

$$\ldots$$

$$q_k = \frac{1}{2^{k-1}} \cdot p_1 + \frac{1}{2^{k-2}} \cdot (p_2 + p_3) + \frac{1}{2^{k-3}} \cdot (p_4 + p_5 + p_6 + p_7) + ...$$

Анализ выражений (2) и (3) показывает, что вероятности искажений различных битов кодовой группы битов при возникновении ошибки передачи соответствующего канального сигнала существенно различаются.

Наиболее вероятно искажение младшего бита – x_1 кодовой группы, вероятность q_2 искажения второго бита x_2 практически вполовину меньше. При наиболее вероятной ошибке передачи канального сигнала – изменении его амплитуды на величину близкую к s, с вероятностью 0.5 исказится только один бит (младший– x_1) соответствующей кодовой группы. С вероятностью – 0.5 – большее число бит кодовой группы.

Существующие средства контроля ошибок, в частности, CRC не учитывают эту весьма существенную разницу в вероятности искажения битов кодового слова при возникновении ошибки передачи модулирующего канального сигнала. Вполне очевидно, что учет этой разности является потенциальным источником повышения эффективности обнаружения ошибок. Эта возможность на практике может быть реализована путем оптимизации кодирования контрольной информации.

Анализ возможных битовых искажений цифровых данных при их передаче по сетевым линиям с амплитудно-импульсной модуляцией необходимо проводить с учетом возможности многократных искажений канальных сигналов при передаче блока цифровых данных, содержащего m бит.

При однократной ошибке передачи канального сигнала, любые битовые искажения блока передаваемых данных гарантированно обнаруживаются при использовании CRC, если степень n его образующего полинома больше длины k кодовой группы: $n>k$. Это обусловлено свойствами CRC гарантированно обнаруживать ошибки, локализованные в области, длина которой не превышает степени образующего полинома [4].

Рассмотрим, в частности, достаточно вероятный на практике [3] случай двукратной ошибки передачи канального сигнала. Если обозначить через P_c – вероятность ошибочной передачи одного канального сигнала, то вероятность P_2 указанной ситуации при больших значениях m определяется выражением:

$$P_2 \approx \frac{m^2}{2 \cdot k^2} \cdot P_C^2 \qquad (4)$$

В рассматриваемом случае, потенциально могут быть искажены $2 \cdot k$ битов блока, не локализованные в одной области. В рамках проводимого анализа предположим наиболее вероятную ситуацию, что в при искажении обоих канальных сигналов, их амплитуда вследствие гауссова шума меняется на s. Такая двойная ошибка передачи канального сигнала гарантированно обнаруживается CRC, если при каждой из них искажению подвергся только один бит соответствующей кодовой группы, вероятность этого равна 0.25. С вероятностью, 0.75 искажено больше, чем два бита, причем не локализованных в одной обрасти. Вероятность необнаружения такой ошибки с использованием CRC составляет 2^{-n}. Следовательно, при применении CRC двойная ошибка передачи канального сигнала обнаруживается с вероятностью $3 \cdot 2^{-(n+2)}$.

Таким образом, использование существующих средств контроля ошибок, в частности CRC, не позволяет эффективно обнаруживать ошибки передачи канальных символов в линиях с амплитудной модуляцией. Фактически гарантированно может быть обнаружена только однократная ошибка передачи канальных символов.

Целью работы является повышение эффективности обнаружения ошибок передачи данных в линиях с амплитудной модуляцией за счет повышения кратности гарантировано обнаруживаемых ошибок передачи канальных символов.

Модификация CRC для повышения эффективности обнаружения ошибок в линиях с амплитудно-импульсной модуляцией

Предлагается способ повышения эффективности использования CRC для контроля ошибок в сетевых линиях с амплитудно-фазовой модуляцией цифровых данных за счет расширения класса гарантированно обнаруживаемых ошибок, а именно – обнаружения двойных ошибок передачи канальных сигналов, а также за счет ускорения вычислений, связанных с контролем ошибок. Способ ориентирован на использование в линиях передачи данных, для которых доминирующим типом искажений импульсов является изменение ε амплитуды импульса на величину, не превышающую $3 \cdot s/2$, где s- значение шага дискретизации импульса при амплитудно-импульсной модуляции.

Сущность предлагаемого способа состоит в том, что вычисление контрольного кода CRC производится только с учетом младшего бита кодовой группы – x_1 с пропуском остальных бит. Тогда, при возникновении однократной ошибки передачи канального сигнала, при условии, что искажение амплитуды составит не более s, такая ошибка всегда будет гарантированно обнаружена. При возникновении двукратной ошибки передачи канальных сигналов, амплитуда которых искажается не более, чем на шаг дискретизации – s, в контролируемой CRC последовательности битов искажаются только два бита. Такое искажение гарантированно обнаруживается при использовании стандартизированного полинома CRC. Для того, чтобы гарантированно обнаруживались однократные ошибки передачи канальных сигналов с изменением ε амплитуды до $7 \cdot s/2$, предлагается вычислять в качестве дополнительного контрольного бита сумму по модулю 2 вторых битов $-x_2$ всех кодовых групп контролируемого блока данных.

Таким образом, в отличие от обычной схемы применения CRC, расширяется класс гарантированно обнаруживаемых ошибок за счет двойных ошибок передачи канального символа. Кроме того, при использовании предлагаемого способа вычисления контрольного кода CRC, в силу того, что обрабатывается только каждый k–тый бит передаваемого блока

данных, соответственно возрастает в k раз скорость реализации вычислений, связанных с контролем ошибок.

Обнаружение многократных ошибок передачи канальных символов с использованием взвешенной контрольной суммы

Неравновероятный характер искажения битов кодовых групп при возникновении ошибки передачи канального сигнала может быть использован для повышения эффективности обнаружения ошибок с применением взвешенной контрольной суммы [5].

Ниже предлагается один из вариантов реализации такой возможности, учитывающей близкие к реальным распределения вероятностей появления ошибок различных классов [2].

Модифицированную взвешенную контрольную сумму блока $B=\{b_1,b_2,\ldots,b_m\}=\{X_1,X_2,\ldots,X_t\}$, где $t=m/k$ – количество кодовых групп в блоке, предлагается вычислять в виде суммы по модулю 2 контрольных кодов кодовых групп: $CS = V_1 \oplus V_2 \oplus \ldots \oplus V_t$. Контрольный код кодовой группы предлагается формировать состоящим из $\log_2 t+2$ битов, организованных в три битовых поля: $\log_2 t$-разрядное поле Z_1 и двух однобитовых полей z_2 z_3 : $V=Z_1 \mid z_2 \mid z_3$. Поле Z_1 контрольного кода V_i, $i=1,\ldots,t$, соответствующего i-той в блоке кодовой группе X_i предлагается формировать в виде логического произведения младшего бита x_1^i кода кодовой группы X_i на разряды двоичного кода порядкового номера W_i кодовой группы в блоке: $V_i= x_1^i \cdot W_i$, где $W_i = \{w_1^i, w_2^i, \ldots, w_l^i\}, l = \log_2 t, \forall e \in \{1,..,l\} : w_e \in \{0,1\}$, причем $\sum_{e=1}^{l} w_e^i \cdot 2^{e-1} = i$. В качестве однобитовых полей z_2 и z_3 предлагается использовать младший x_1^i и следующий за ним - x_2^i биты кодовой группы X_i соответственно. Использование такой модификации взвешенной Контрольной суммы позволяет гарантированно обнаруживать все ошибки, доминирующие в сетевых линиях с амплитудно-импульсной модуляцией.

При возникновении под воздействием гауссова шума однократной ошибки передачи канального сигнала с большой вероятностью p_1 амплитуда канального импульса изменится на величину ε, лежащую в интервале $s/2<\varepsilon<3\cdot s/2$. Поскольку, при этом исказится младший бит x_1 передаваемой кодовой группы X, то биты поля z_2 соответствующего контрольного кода V на приемнике и передатчике будут отличаться, равно как, будут отличаться и контрольные суммы на приемнике и передатчике. Это значит, что такая ошибка будет безусловно обнаружена.

При возникновении двух ошибок передачи канального сигнала, при которых в обоих случаях изменение ε амплитуды импульса находится в интервале $s/2<\varepsilon<3\cdot s/2$, поля z_2 контрольных сумм на передатчике и приемнике будут идентичными. Покажем, что при этом код поля Z_1 на приемнике и передатчике будут отличаться. Пусть, ошибки имели место при

передаче j–го и i–го канальных сигналов, $i,j\in\{1,\ldots,t\}$ (соответственно, кодовых групп – X_i и X_j). Поскольку, значения битов x_1^j на приемнике и передатчике отличны, то поле Z_1 контрольного кода V_j на передатчике состоит из l нулевых битов (если на передатчике $x_1^j=0$), в то время как на приемнике поле Z_1 контрольного кода V_j равно W_j, поскольку $x_1^j=1$ (или наоборот, в случае, если на передатчике $x_1^j=1$, а на приемнике $x_1^j=0$). Аналогично обстоит дело и с контрольными кодами на передатчике и приемнике и для контрольного кода V_i : либо на приемнике поле Z_1 этого кода равно W_i, при том, что на передатчике оно равно нулю, либо наоборот. Из изложенного очевидно, что поле Z_1 суммы по модулю 2 контрольных кодов приемника V_{jS} и передатчика V_{iR} равно W_j,, равно как поле Z_1 суммы по модулю 2 контрольных кодов приемника V_{iS} и передатчика V_{iR} равно W_i. Для всех остальных канальных сигналов суммы по модулю 2 контрольных кодов приемника и передатчика равны нулю: $\forall u\in\{1,\ldots,t\},u\neq i,u\neq j$: $V_{uS}\oplus V_{uR}=0$. Из этого следует, что поле Z_1 контрольных сумм приемника и передатчика для рассматриваемого случая двойной ошибки будет равно $W_j\oplus W_i\neq 0$. Это означает, что двойные ошибки рассматриваемого типа, которые доминируют по сравнению с другими типами двойных ошибок будут гарантированно обнаруживаться предлагаемой модификацией взвешенной контрольной суммы, в отличие от применения CRC.

При возникновении под воздействием гауссова шума однократной ошибки передачи канального сигнала, с вероятностью $p_2<<p_1$ амплитуда канального импульса изменится на величину ε, лежащую в интервале $3\cdot s/2<\varepsilon<7\cdot s/2$. Поскольку при этом исказится второй бит x_2 передаваемой кодовой группы X, то биты поля z_3 соответствующего контрольного кода V на приемнике и передатчике будут отличаться, равно как, будут отличаться и контрольные суммы на приемнике и передатчике. Это значит, что такая ошибка гарантированно обнаруживается предложенной модификацией взвешенной контрольной суммы.

При возникновении двух ошибок передачи канального сигнала, при которых в одном случае изменение ε амплитуды импульса находится в интервале $s/2<\varepsilon<3\cdot s/2$, а во втором – в интервале $3\cdot s/2<\varepsilon<5\cdot s/2$. В этом случае искажение битов передаваемого блока также гарантированно обнаруживается с использованием предлагаемой модификации взвешенной контрольной суммы. Действительно, пусть указанные ошибки имели место при передаче j–го и i–го канальных сигналов, $i,j\in\{1,\ldots,t\}$ (соответственно, кодовых групп – X_i и X_j). Пусть, например, при передаче j–го канального сигнала изменение ε его амплитуды находится в интервале: $3\cdot s/2<\varepsilon<5\cdot s/2$, а при изменении i–го канального сигнала – в интервале

$s/2<\varepsilon<3\cdot s/2$. Тогда, при передаче кодовой группы X_i исказится ее второй бит – x_2^j, при том, что младший – x_1^j не исказится. При передаче кодовой группы X_i точно исказится ее младший бит $-x_1^i$, в то время, как следующий $-x_2^i$ исказится с вероятностью 0.5. Из изложенного следует, что битовое поле z_2 контрольного кода V_i будет отличаться на передатчике и приемнике, в то время, как для остальных контрольных кодов $V_1,\ldots,V_{i-1},V_{i+1},\ldots,V_t$ значения битового поля z_2 на передатчике и приемнике будут одинаковыми. Следовательно, поле z_2 кода Δ разности контрольных сумм передатчика и приемника не равно нулю, что означает гарантированное обнаружение двукратной ошибки рассматриваемого типа.

Предложенная модификация взвешенной контрольной суммы позволяет также гарантированно обнаруживать ошибки передачи канальных сигналов нечетной кратности, при условии, что изменения ε амплитуды импульсов находится в интервале $s/2<\varepsilon<3\cdot s/2$. Как уже было показано выше, это наиболее вероятное изменение амплитуды, имеющее следствием искажение битов контролируемого блока данных. Важным является то, что при использовании CRC обнаружение ошибок этого класса, при условии, что число ошибочно переданных импульсов больше единицы, не гарантируется: они обнаруживаются с вероятностью, зависящей от степени образующего полинома CRC [4].

Таким образом, предложенная модификация взвешенной контрольной суммы, ориентированная на использование в сетевых линиях с амплитудно-импульсной модуляцией позволяет, за счет оптимизации кодирования контрольной информации, гарантированно обнаруживать доминирующие в таких линиях типы ошибок. По сравнению с традиционно используемыми для обнаружения ошибок в таких линиях CRC предложенный способ обладает следующими преимуществами:

1. Расширением класса гарантированно обнаруживаемых ошибок за счет двукратных ошибок передачи канальных сигналов, а также ошибок передачи канальных сигналов нечетной кратности, которые выявляются CRC с вероятностью $1-2^n$, где n-степень образующего полинома. Действительно, при наиболее часто встречающемся типе двойной ошибке передачи канальных сигналов, когда изменение ε амплитуды двух искаженных импульсов находится в интервале $s/2<\varepsilon<3\cdot s/2$, с вероятностью 0.75 искажаются более, чем 2 бита контролируемого блока. Согласно свойствам CRC, при его применении искажение более, чем 2-х бит может быть обнаружено с вероятностью $1-2^n$.
2. Уменьшение количества контрольных разрядов для относительно коротких блоков передаваемых данных. Если при использовании CRC число контрольных разрядов равно n (на практике, в зависимости от применения стандартов CRC-16 или CRC-31 n=16 или n=32), то при

использовании предлагаемой модификации взвешенной контрольной суммы число контрольных разрядов равно $2+\log_2 t$. Например, при контроле передачи блока длиной 64 байта с использованием 8-ми уровневой амплитудно-импульсной модуляции, $t = 2^{11}$, соответственно требуемое число контрольных разрядов равно 13.

3. Ускорение выполнения контроля ошибок передачи данных за счет того, что, во-первых, в предлагаемой модификации контрольной суммы контролируются только 2 младших разряда символа, а во-вторых, за счет того, что при аппаратной реализации существует возможность распараллеливая вычислений, связанных с контролем отдельных фрагментов передаваемого блока.

Выводы

Проведенные исследования, направленные на повышение эффективности контроля ошибок в линиях с амплитудно-импульсной модуляцией цифровых данных показали, что традиционно используемые для обнаружения ошибок технологии CRC не гарантируют обнаружения важных классов ошибок, и в частности, битовых искажений, вызванных ошибками передачи канальных сигналов, кратность которых превышает единицу. Для повышения эффективности обнаружения ошибок в линиях с амплитудно-импульсной модуляцией проанализированы их особенности. При этом, ошибки, возникающие в таких линиях исследованы в двух ракурсах: с точки зрения кратности ошибок передачи канальных сигналов и с точки зрения вероятностных характеристик изменения амплитуды канального импульса, а также влияния этого изменения на искажение отдельных битов кодовой группы, модулируемой одним канальным сигналом. Доказано, что в линиях с амплитудно-импульсной модуляцией вероятности искажений отдельных битов кодовой группы при ошибке передачи модулирующего канального сигнала сильно отличаются, соответственно отличается объем информации, который связан с их искажением. Это открывает потенциальные возможности повышения эффективности средств обнаружения средств в таких линиях за счет оптимизации кодирования контрольной информации.

Предложен способ повышения эффективности применения CRC для контроля ошибок в сетевых линиях с амплитудно-импульсной модуляцией цифровых данных за счет расширения класса гарантированно обнаруживаемых ошибок, а именно - обнаружения двойных ошибок передачи канальных сигналов.

Разработан и исследован способ обнаружения ошибок на основе использования взвешенной контрольной суммы, который позволяет по сравнению с традиционной CS и CRC расширить класс гарантированно обнаруживаемых ошибок, снизить количество контрольных битов для относительно коротких блоков, а также уменьшить время выполнения

вычислений, связанных с контролем ошибок в темпе передачи цифровых данных в линиях с амплитудно-импульсной модуляцией.

Список использованной литературы

1. Скляр Б. Цифровая связь. Теоретические основы и практическое применение. М.: Издательский дом ”Вильямс”, 2004.- 1104 с.
2. Blahut R.E. Theory and Practice of Error Control Codes. Mass.:Addison-Wesley Publishing Company, 1983 – 452 p.
3. Ирвин Дж., Харль Д. Передача данных в сетях: инженерный подход. СПб.: БХВ-Петербург, 2002.- 448 с.
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.
5. Klove T., Korzhik V. Error Detecting Codes: General Theory and Their Application in Feedback Communication Systems. Norwell, MA: Kluwer, 1995. – 433 p.

УДК 004.052.42

МАРКОВСКИЙ А.П.,
МУЛКИ АХМЕД ЯССИН АЛ БАДАЙНЕХ.,
ПУЯ СОЛЕЙМАНИ НЕЖАДИАН

ОБ ОДНОМ ПОДХОДЕ К ПОВЫШЕНИЮ ЭФФЕКТИВНОСТИ ОБНАРУЖЕНИЯ И ИСПРАВЛЕНИЯ ОШИБОК ПЕРЕДАЧИ ДАННЫХ

Статья посвящена решению проблемы повышения эффективности обнаружения и исправления ошибок передачи данных за счет расширения класса гарантированно обнаруживаемых ошибок и класса ошибок, которые могут быть исправлены без повторной передачи. Для решения проблемы предложено использовать двумерную взвешенную контрольную сумму. Разработан алгоритм обнаружения и коррекции ошибок. Показано, что предложенная технология обеспечивает большую эффективность обнаружения и исправления ошибок по сравнению с CRC и традиционной двумерной контрольной суммой.

Paper is dedicated to solving the problem of increasing the efficiency of data transmission error detecting and correcting by expanding the class of guaranteed detecting errors and class of ones which can be corrected without repeat transmission. For solving the problem the utilization of two-dimension weighed check sum has been proposed. Algorithm for error detection and correction has been worked out. It has been shown that proposed techniques ensure superior data transformation error detecting and correcting effectiveness in compare to CRC and traditional two-dimension checksum.

Введение

Передача цифровых данных является одним из наименее надежных среди компонент систем обработки информации. Трудности обеспечения высокой достоверности передачи данных обусловлены сложной природой физических процессов в длинных линиях, отражением сигналов и их взаимным влиянием (межсигнальная интерференция), влиянием внешних помех. Поэтому, с конца 40-х годов интенсивно развивается технология обнаружения и исправления ошибок передачи данных.

В последнее десятилетие отмечается качественный прогресс с области технологии передачи данных: многократно возросли объемы и скорость передачи информации. Вместе с тем, проблема обеспечения эффективного контроля ошибок приобрела большую важность. Уменьшение временных интервалов между сигналами, расширение применения спектрального уплотнения имеют следствием рост числа ошибок интерференционной природы [1]. Увеличение интенсивностей электромагнитных полей, обусловленных расширением применения беспроводных линий передачи данных вызывает рост числа ошибок, вызванных внешними помехами. Для новых технологий передачи данных характерны специфические типы

возникающих искажений информации. Все это диктует необходимость развития средств контроля и исправления ошибок, адекватного прогрессу технологии передачи данных.

Таким образом, проблема повышения эффективности обнаружения и исправления ошибок с учетом особенностей современных технологий передачи данных является важной и актуальной для развития компьютерных систем и сетей.

Анализ средств контроля ошибок передачи данных

В современных шинах (USB, PCI) компьютерных систем и большинстве сетевых линий информация передается блоками, которые могут разделяться на символы [2]. Пусть передаваемый блок B состоит из m бит: $B=\{b_1,b_2,\ldots,b_m\}$, $b_l\in\{0,1\}$, $l=1,\ldots,m$ и разделяется на $t=m/n$ n-разрядных символов. Если обозначить j–тый символ передаваемого блока в виде $X_j=\{x_1,x_2,...,x_n\}=\{b_{(j-1)\cdot n+1},b_{(j-1)\cdot n+2},...,b_{j\cdot n}\}$, $j=1,2,..,t$, $\forall i\in\{1,\ldots,n\}$: $x_i\in\{0,1\}$, то блок B может быть представлен в виде последовательности $t = m/n$ символов: $B=\{X_1,X_2,\ldots,X_t\}$, $X_j\in\{0,1,\ldots,2^n-1\}$.

Для обеспечения надежности передачи цифровых данных используется две базовые технологии - прямое исправление ошибок (FEC-forward error correction) с применением корректирующих кодов и обнаружение ошибок с автоматическое повторной передачей при их обнаружении (ARQ-Automatic Repeat Request) [1]. При использовании технологии FEC контрольный код передается после каждого из символов блока, а технология ARQ предусматривает передачу контрольного кода только после всего блока.

В рамках технологии *FEC* для исправления ошибок используется широкая гамма корректирующих кодов, наиболее известными из которых являются коды Голея, Рида-Маллера, Боуза-Чоудхури-Хоквингема (БЧХ), турбо-коды [1]. Недостатками корректирующих кодов является относительно большой объем контрольной информации, сложность процедур кодирования и декодирования. Число контрольных разрядов быстро растет с увеличением кратности исправляемых ошибок, поэтому в рамках технологии FEC обычно используют перемешивание битов символов блока с тем, чтобы многократная ошибка возникающая, обычно, внутри символа распределялась по многим символам. Это существенно усложняет процедуры кодирования и декодирования. Поэтому, практическое использование корректирующих кодов оправдано, если интенсивность возникающих ошибок достаточно высока, а также, в случае отсутствия возможности для повторной передачи блока. Соответственно, основными сферами практического применения корректирующих кодов являются системы беспроводной связи, для которых характерным является отно-

сительно высокая интенсивность ошибок, вызванных внешними помехами, а также системы хранения информации на дисках.

Главное преимущество ARQ перед FEC заключается в том, что обнаружение ошибок производится гораздо быстрее и проще по сравнению с коррекцией ошибок. При использовании технологии ARQ объем контрольной информации, передаваемой по линии существенно меньше. В проводных, кабельных и опто-волоконных линиях передачи цифровых данных интенсивность возникновения ошибок относительно мала, поэтому использование корректирующих кодов оказывается неэффективным. Для обеспечения безошибочной передачи в таких линиях преимущественно используется технология ARQ [3].

Для обнаружения ошибок передачи данных в рамках технологии ARQ используются циклические коды (CRC- Cyclic Redundancy Check), и контрольные суммы (CS-Check Sum). Использование CRC позволяет обнаруживать все ошибки нечетной кратности, все двойные ошибки при передаче блока данных, а также ”пачку” ошибок, локализованных в участке длиной не превышающего степень образующего полинома CRC [4]. Традиционная *CS*, вычисляемая как побитовая сумма по модулю 2 всех символов блока, позволяет гарантированно обнаруживать только все ошибки нечетной кратности. Однако, модификации *CS* обеспечивают существенно большую достоверность обнаружения ошибок. Так, двумерная CS (TDCS-Two Dimension Check Sum) позволяет гарантированно обнаруживать ошибки нечетной кратности и двукратные ошибки. Некоторые варианты взвешенной CS позволяют обнаруживать, кроме ошибок нечетной кратности, все ошибки кратностью 2 и 4 [5].

Существенным отличием перечисленных кодов для обнаружения ошибок передачи информации от корректирующих кодов является отсутствие функциональной зависимости между длиной m контролируемого блока и разрядностью k контрольного кода.

Как уже отмечалось выше, в современных условиях существуют объективные факторы, действие которых имеет следствием увеличение числа возникающих ошибок в линиях передачи данных. Это требует адекватного повышения эффективности контроля правильности передачи, а также технологии исправления возникающих ошибок. В частности, рост интенсивности внешних электро-магнитных полей требует увеличения кратности гарантированно обнаруживаемых ошибок, а также повышения достоверности выявления ”пачек” ошибок.

Кратность ошибок, гарантированно обнаруживаемых CRC и известными модификациями CS не превышает 4-х, поэтому необходимо создание средств с большей кратностью гарантированно обнаруживаемых ошибок. Другой важной проблемой, которая не решается с использованием существующих средств обнаружения ошибок является обеспечение эффективности исправления выявленных ошибок. CRC и известные моди-

фикации CS не позволяют локализовать часть блока, содержащей неверно переданные биты. Поэтому для исправления выявленных ошибок выполняется повторная передача всего блока. Это сопряжено во-первых, с заметными затратами времени, а во-вторых, не исключает возможности возникновения ошибок при повторной передаче. Учитывая тенденцию роста объемов передаваемых блоков, значимость указанного недостатка также увеличивается.

Целью работы является увеличение кратности гарантированно обнаруживаемых ошибок и повышение эффективности исправления выявленных ошибок.

Модификация TDCS на основе использования взвешенных контрольных сумм

Повышение кратности обнаруживаемых ошибок передачи данных необходимо связано с увеличением объема передаваемой контрольной информации.

В качестве основы для разработки эффективных средств обнаружения ошибок большой кратности выбрана TDCS. Это обусловлено тем, что стандарты большинства линий передачи вычислительных систем, в частности COM-портов и шин PCI, а также линий компьютерных сетей предусматривают пересылку бита четности на каждый передаваемый байт [2].

При контроле с использованием *TDCS* передаваемых блок данных можно представить в виде матрицы *D*, имеющей *n* столбцов *m* строк. Вместе с блоком передается *m* битов четности $p^S_{r1}, p^S_{r2},..., p^S_{rm}$ строк и *n* битов четности $p^S_{C1}, p^S_{C2},..., p^S_{Cn}$ столбцов матрицы D^S передатчика. При этом, бит четности *i*–той, $i\in\{1,\ldots,m\}$, строки $\{d^S_{i1}, d^S_{i2},\cdots, d^S_{in}\}$ матрицы D^S вычисляется в виде: $p^S_{ri} = d^S_{i1} \oplus d^S_{i2} \oplus \cdots \oplus d^S_{in}$, а бит четности *j*-того, $j\in\{1,\ldots,n\}$, столбца $\{d^S_{1j}, d^S_{2j},\cdots, d^S_{mj}\}$ матрицы D^S вычисляется в виде: $p^S_{Cj} = d^S_{1j} \oplus d^S_{2j} \oplus \cdots \oplus d^S_{mj}$. На приемнике биты четности строк $p^R_{r1}, p^R_{r2},..., p^R_{rm}$ и биты четности столбцов $p^R_{C1}, p^R_{C2},..., p^R_{Cn}$ вычисляются по матрице D^R принятого блока и сравниваются с принятыми от передатчика. Решение о наличии ошибок передачи производится путем анализа вектора $\Delta_R=\{r_1,r_2,\ldots,r_m\}$ несовпадения битов четности строк на приемнике и передатчике, а также вектора $\Delta_C=\{c_1,c_2,\ldots,c_m\}$ несовпадения битов четности столбцов матриц D^R и D^C. Это означает, что каждый *i*-тый бит Δ_R формируется как: $r_i = p^S_{ri} \oplus p^R_{ri}$ и каждый *j*–тый бит Δ_C вычисляется в виде: $c_j = p^S_{Cj} \oplus p^R_{Cj}$.

При использовании такой TDCS можно гарантированно обнаруживать все ошибки нечетной кратности и двойные ошибки. 4-кратные ошибки,

попарно локализованные в одноименных разрядах двух строк матрицы D не обнаруживаются. Кроме обнаружения ошибок указанной выше кратности, TDCS позволяет локализовать битовую позицию однократной ошибки. Это позволяет, в принципе, использовать TDCS для исправления наиболее часто встречающихся в проводных линиях однократных ошибок. При этом существует риск не различить однократную ошибку от пятикратной. Действительно, если две пары ошибок локализованы в одинаковых битах двух строк, а пятая ошибка локализована в j-том бите i-той строки, то по два ошибочных бита q_1 и q_2 локализованы в строке и их строковые биты четности равны нулю.

Основным недостатком описанной традиционной TDCS является малая кратность гарантированно обнаруживаемых ошибок.

Для повышения достоверности обнаружения ошибок передачи данных и повышения эффективности их исправления с использованием двумерной контрольной суммы предлагается ее модификация, отличающаяся наличием контроля с использованием взвешенных сумм битов столбцов матрицы D. Предлагаемая модификация предполагает вычисление для каждого j–го столбца матрицы D^S h-разрядного кода S_j^S взвешенной контрольной суммы $S_j^S = d_{1j}^S \cdot W_1 \oplus d_{2j}^S \cdot W_2 \oplus \cdots \oplus d_{mj}^S \cdot W_m$, где W_1, $W_2,\ldots,W_m$ – h-разрядные коды весовых коэффициентов строк матрицы D. В простейшем случае $h=\log_2 m$, а W_i – двоичный код номера i строки. Аналогичным образом вычисляется значение h-разрядного кода S_j^R взвешенной суммы для j–го столбца матрицы D^R приемника.

Вычисленные на передатчике взвешенные контрольные суммы $S_1^S, S_2^S, ..., S_n^S$ передаются на приемник, где сравниваются с взвешенными контрольными кодами $S_1^R, S_2^R, ..., S_n^R$, вычисленными на приемнике. Результатом сравнения является набор Δ_V h-разрядных кодов $\{V_1,V_2,\ldots,V_n\}$, каждый j–тый из которых вычисляется как поразрядная сумма по модулю 2 соответствующих взвешенных сумм передатчика и приемника $V_j = S_j^S \oplus S_j^R$. Передача считается выполненной без ошибок, если все компоненты множеств $\Delta_R=\{r_1,r_2,\ldots,r_m\}$, $\Delta_C=\{c_1,c_2,\ldots,c_m\}$ и $\Delta_V=\{V_1,V_2,\ldots,V_n\}$ равны нулю. Суммарное число k передаваемых контрольных бит в предлагаемой модификации TDCS определяется формулой:

$$k = m + n(1 + \log_2 m) \qquad (1)$$

Введение взвешенных контрольных сумм столбцов матрицы D позволяет гарантированно обнаруживать двойную ошибку в столбце. Действительно, если ошибка происходит в q-том и g–том, $q,g \in \{1,\ldots,m\}$, битах j-того столбца матрицы D, то $V_j = W_q \oplus W_g \neq 0$, в силу того, что не равна нулю сумма по модулю 2 двух различных кодов.

Анализ возможностей модифицированной TDCS по обнаружению многократных ошибок

Предложенная модификация TDCS позволяет гарантированно обнаруживать все ошибки нечетной кратности, поскольку при возникновении таких ошибок не менее, чем один из битов $\{r_1,r_2,\ldots,r_m\}$ и не менее чем один из битов $\{c_1,c_2,\ldots,c_m\}$ равен единице.

Для того, чтобы ошибка четной кратности не была обнаружена при использовании предлагаемой TDCS, то есть, для того, чтобы ее проверочные коды $\Delta_R=\{r_1,r_2,\ldots,r_m\}$, $\Delta_C=\{c_1,c_2,\ldots,c_m\}$ и $\Delta_V=\{V_1,V_2,\ldots,V_n\}$ не были равны нулю необходимо выполнение 3-х условий:

1. Для того, чтобы $\Delta_R=\{r_1,r_2,\ldots,r_m\}$ не содержал единичных компонент, необходимо, чтобы число ошибочных бит в каждой из искаженных строк матрицы D^R было четным.
2. Для того, чтобы $\Delta_C=\{c_1,c_2,\ldots,c_m\}$ не содержал единичных компонент, необходимо, чтобы число ошибочных бит в каждом из искаженных столбцов матрицы D^R было четным.
3. Для того, чтобы все компоненты $\Delta_V=\{V_1,V_2,\ldots,V_n\}$ были равны нулю, необходимо, чтобы число ошибочных бит в каждом из искаженных столбцов матрицы D^R было больше 2-х.

Из первых двух условий следует, что кратность ошибок должна нацело делиться на 4, а из третьего следует, что она должна быть больше 4-х. Минимальная кратность ошибок, удовлетворяющих приведенным выше условиям равна 8-ми. Действительно, если 8 ошибочных бит локализованы попарно в разрядах x_1 и x_2 4-х строках с номерами y_1,y_2,y_3,y_4 , то все битовые компоненты $\Delta_R=\{r_1,r_2,\ldots,r_m\}$ и $\Delta_C=\{c_1,c_2,\ldots,c_m\}$ равны нулю и существует ненулевая вероятность того, что сумма 4-х весовых коэффициентов равна нулю: $V_{X_1}=W_{Y1}\oplus W_{Y2}\oplus W_{Y3}\oplus W_{Y4}=0$. Количество E_8 вариантов таких локализаций определяется формулой:

$$E_8=\binom{m}{4}\cdot\binom{n}{2} \qquad (2)$$

Поскольку сумма по модулю 2 4-х h–разрядных двоичных кодов $V_{X_1}=W_{Y1}\oplus W_{Y2}\oplus W_{Y3}\oplus W_{Y4}$ может с равной вероятностью принимать $2^h = m$ значений, то вероятность p_{E8} того, что эта сумма равна нулю составляет: $p_{E8} = 1/m$. Из этого и выражения (2) следует, что вероятность P_{N8} того, что возникшая 8-кратная ошибка не будет обнаружена определяется в виде:

$$P_{N8}=\frac{E_8\cdot p_{E8}}{\binom{m\cdot n}{8}}\approx\frac{m^4\cdot n^2\cdot 8!}{m^9\cdot n^8\cdot 4!\cdot 2}=\frac{480}{(m\cdot n)^5\cdot n} \qquad (3)$$

Например, при контроле правильности передачи блока длиной 1 Кбайт (n=8, m=1024), вероятность P_{N8} того, что возникшая 8-кратная ошибка не будет обнаружена составляет $3\cdot10^{-18}$.

Рассуждая аналогичным образом, несложно показать, что для 12-кратной ошибки, вероятность P_{N12} того, что она не будет обнаружена определяется формулой:

$$P_{N12} = \frac{\binom{m}{6}\cdot\binom{n}{2}\cdot m^{-1}}{\binom{m\cdot n}{12}} \approx \frac{m^6\cdot n^2\cdot 12!}{m^{13}\cdot n^{12}\cdot 6!\cdot 2} = \frac{332640}{(m\cdot n)^7\cdot n^3} \qquad (4)$$

Так, для n=8, m=1024, в соответствии с (3), вероятность $P_{N8} = 2.6\cdot10^{-25}$. Для больших кратностей 16,20,…ошибок, вероятности того, что они не будут обнаружены, быстро убывают.

Таким образом, по сравнению с CRC и обычной TDCS, использование предложенной модификации TDCS позволяет существенно расширить класс гарантированно обнаруживаемых ошибок за счет безусловного выявления ошибок кратности 4 и 6.

С практической точки зрения важной является оценка возможности гарантированного обнаружения ”пачки” ошибок, вызванной внешними помехами и наводками. Анализ сформулированных выше трех условий при выполнении которых ошибки не обнаруживаются, позволяет сделать вывод о том, что ”пачка” битовых ошибок гарантированно обнаруживается, если ее длина не превышает $4\cdot n$-1. Следовательно, по возможности обнаружения ”пачек” ошибок предложенная модификация TDCS практически соответствует CRC-32 и существенно превышает обычную TDCS, которая позволяет гарантированно обнаруживать ”пачки” ошибок, длина которых не превышает $4\cdot n$-1.

Организация исправления ошибок передачи данных в TDCS на основе взвешенных сумм

Использование модифицированной TDCS позволяет существенно расширить класс ошибок, которые могут быть исправлены без повторной передачи всего блока.

При организации исправления возникших ошибок возможными являются три режима:

- исправление без передачи дополнительной информации, то есть на основе наличных проверочных кодов $\Delta_R=\{r_1,r_2,\ldots,r_m\}$, $\Delta_C=\{c_1,c_2,\ldots,c_m\}$ и $\Delta_V=\{V_1,V_2,\ldots,V_n\}$;
- формирование требований на передачу отдельных фрагментов блока и исправление выявленных ошибок на их основе.
- формирование требования на повторную передачу всего блока.

Для исправления возникших ошибок передачи блока предлагается следующий алгоритм обработки на приемнике проверочных кодов $\Delta_R=\{r_1,r_2,\ldots,r_m\}$, $\Delta_C=\{c_1,c_2,\ldots,c_n\}$ и $\Delta_V=\{V_1,V_2,\ldots,V_n\}$. Алгоритм реализует коррекцию тех ошибок, которые могут быть исправлены с использованием векторов Δ_R, Δ_C и Δ_V, а также составление списка Ξ скорректированных строк матрицы D^R. Предлагаемый алгоритм включает две базовых процедуры:

А. Процедуру коррекции ошибок по проверочным кодам Δ_R, Δ_C и Δ_V.

В. Процедуру коррекции ошибок за счет повторной передачи.

Процедура **А** коррекции ошибок по проверочным кодам Δ_R, Δ_C и Δ_V, в свою очередь, состоит из двух циклов:

А.1. Цикла коррекции ошибок нечетной кратности в строках матрицы D^R.

А.2. Цикла коррекции одиночных ошибок в столбцах матрицы D^R.

Реализация первого из перечисленных циклов – **А.1** состоит в последовательном выполнении следующей последовательности действий:

1. Построить множество Ω номеров строк, у которых проверочный бит четности равен единице: $r_i=1$: $i\in\Omega$. $\Xi=\varnothing$. Установить счетчик s скорректированных битов в нуль: $s=0$.

2. Сформировать множество $\Theta = \{\vartheta_1,\vartheta_2,\ldots,\vartheta_t\}$ всех возможных подмножеств множества Ω. Для каждого u-го из подмножеств Θ, $u\in\{1,\ldots,t\}$, сформировать:

- бит c'_u, равный единице, если число элементов ϑ_u –нечетное и равный нулю, если количество элементов ϑ_u – четное.
- h-разрядный код $V'_u = \bigoplus_{g\in\vartheta_U} W_g$.

3. Установить в нуль флаг f корректирования матрицы D^R - $f:=0$.
4. Установить индекс столбца $j:=1$.
5. Если $c_j = 0$ и $V_j = 0$ перейти на пп.9.
6. Установить индекс $u=1$.
7. Если $\vartheta_u\neq\varnothing$, $c_j = c'_u$ и $V_j = V'_u$ то:

Перебрать все t-1 возможные пары $<\vartheta_u,\vartheta_l>$, $l\in\{1,\ldots,t\}$ подмножеств множества Θ. Если среди подмножеств Θ нет пары $<\vartheta_u,\vartheta_l>$ для которой выполняются условия: $c'_u=c'_l$ и $V'_u = V'_l$ то выполнить коррекцию:

7.0. Установить $f:=0$.

7.1. Инвертировать j–тый разряд строк матрицы D^R, номера которых принадлежат множеству ϑ_u: $\forall g\in\vartheta_u$: $d^R_{gj} = 1\oplus d^R_{gj}$.

7.2. Установить в нуль c_j и V_j.

7.3. Включить в список Ξ номера всех строк, принадлежащих ϑ_u: $\Xi=\Xi\cup\vartheta_u$.

7.4. Исключить из множества Θ все подмножества, которые имеют общие номера строк с подмножеством ϑ_u: $\forall l=1,\ldots,t$: если $\vartheta_l \cap \vartheta_U \neq \varnothing \Rightarrow \Theta = \Theta - \vartheta_l$. Скорректировать значение t.

7.5. Установить в нуль проверочные биты четности строк, номера которых принадлежат ϑ_u: $\forall g \in \vartheta_u$: $r_g=0$.

7.6. Увеличить значение счетчика s на число, соответствующее количеству элементов множества ϑ_u; перейти на пп.9.

8. Если $u < t$: $u:=u+1$, возврат на пп.7.

9. Если $j < n$: $j:=j+1$, возврат на пп.5.

10. Если матрица D^R корректировалась, то есть $f=1$, возврат на пп.3., иначе конец цикла **А.1**.

Реализация цикла **А.2** коррекции ошибок единичной кратности в столбцах матрицы D^R состоит в выполнении следующей последовательности действий:

11. Установить флаг φ коррекции в цикле **А.2**. в нуль: φ:=0.

12. Установить счетчик j цикла столбцов в единицу: $j=1$.

13. Если $c_j = 0$ и $V_j = 0$ перейти на пп.15.

14. Если $c_j = 1$ и $V_j \neq 0$, то :

14.1. Инвертировать j–тый разряд строки матрицы D^R с номером V_j : $d^R_{Vj} = 1 \oplus d^R_{V_j j}$

14.2. Установить в нуль c_j , V_j : $c_j=0$, $V_j=0$

14.3. Включить в список Ξ номер V_j скорректированной строки: $\Xi=\Xi\cup V_j$.

14.4. Установить φ:=1.

14.5. Инвертировать бит четности скорректированной строки $r_{V_j} = 1 \oplus r_{V_j}$.

14.6. Инкрементировать значение счетчика s числа исправленных битов.

15. Если $j < n$: $j:=j+1$, возврат на пп.12.

16. Если φ = 1, возврат на пп.1, иначе конец цикла **А.2**.

После завершения выполнения процедуры **А**, реализуется процедура **В** повторной передачи отдельных строк или всего блока, которая осуществляется в следующем порядке:

1. Если все компоненты Δ_R, Δ_C и Δ_V равны нулю, то, в зависимости от значения s – количества скорректированных бит матрицы D^R выполнить запрос на повторную передачу строк, номера которых составляют множество Ξ.

2. Если существуют не равные нулю компоненты Δ_R, Δ_C и Δ_V, выполнить запрос на повторную передачу всего блока.

3. По получении копий строк, составляющих множество Ξ, сравнить их с одноименными скорректированными строками матрицы D^R: если они совпадают, то исправления выполнены правильно, если не совпадает – выполнить запрос на повторную передачу блока.

Предложенный алгоритм иллюстрируется следующими примерами. Пусть, передается блок данных объемом 16 байт, причем вместе с каждым байтом передается его бит четности. Соответственно матрица D содержит 16 строк (m=16) по 8 бит в каждой (n=8). Рассмотрим работу описанного выше алгоритма для нескольких вариантов локализации ошибок.

Пример 1. Пусть позиции двух искаженных бит в матрице D задаются двумя тремя парами координат: <3,3>,<7,3>, причем первая цифра соответствует номеру строки, а вторая – номеру столбца. Фактически искажен третий бит двух строк: 3-й и 7-й матрицы D^R. В этом случае на приемнике сформируются следующие не равные нулю проверочные коды: r_3=1, r_7=1, $V_3=W_3 \oplus W_7$=0011⊕0111=0100.

В процессе выполнения цикла **A** строится множество $\Omega=\{3,7\}$ и его подмножества $\vartheta_1=\{3\}, \vartheta_2=\{7\}$, $\vartheta_3=\{3,7\}$, то есть t=6. Вычисляются c_1'=1,V_1'=0011, c_2'=1,V_2'=0111, c_3'=0, V_3'=0100. Очевидно, что операции пп.7 коррекции ошибок будут выполняться только при j=3 u=3 $c_j=c_u'$=0, $V_j=V_u'$. В п.7.1. инвертируются третьи биты строк 3 и 7 матрицы D^R. В пп.7.2 устанавливается в нуль V_3. В пп.7.3 в множество Ξ дописываются номера скорректированных строк: Ξ={3,7}. В пп.7.5. обнуляются биты r_3 и r_7.

Поскольку матрица D^R корректировалась и f=1, то цикл А1 выполняется еще раз. Поскольку все проверочные коды равны нулю, то цикл реализуется без коррекции D^R. Последующий цикл А2 также выполняется без коррекции, поскольку $\forall j \in \{1,\ldots,16\}$: c_j=0 и V_j=0.

В рамках процедуры **B,** в зависимости от значения s, можно осуществить запрос на повторную передачу 3-й и 7-й строк матрицы D и по их получении проверить правильность выполненной выше коррекции ошибок в означенных строках.

Пример 2. Пусть позиции трех искаженных бит в матрице D задаются двумя тремя парами координат: <3,3>,<5,3>,<5,7>. Фактически искажен третий бит двух строк: 3-й и 5-й матрицы D^R, а также 7-й бит 5-й строки. В этом случае на приемнике сформируются следующие, не равные нулю проверочные коды: r_3=1, $V_3 = W_3 \oplus W_5$=0011⊕0101=0110, а также c_7=1, V_7= W_5 = 0101.

В процессе выполнения цикла **A.1** строится множество $\Omega=\vartheta_1=\{3\}$, t=1, c_1'=1, V_1'= W_3 = 0011. Однако, коррекции матрицы D^R, в этом цикле не производится, поскольку не существует пары $<j,u>$, $j \in \{1,\ldots,16\}$, u=1 : $c_j = c_1' \; V_j = V_1'$. При реализации цикла **A.2**. при j=7 выполняется условие пп.14: c_7=1, V_7= W_5 = 0101≠0. Соответственно, осуществляется коррекция 7-го бита строки матрицы D^R с номером V_7 = 5 (пп.14.1). В пп.14.2 c_7, V_7 устанавливаются в нуль. В список Ξ дописывается номер скорректированной строки: Ξ={5}. Флаг φ устанавливается в единицу. Инвертируется проверочный бит r_5 четности 5-той строки: r_5=1.

Поскольку φ=1, по пп.16 осуществляется возврат на повторное выполнение цикла **A1**. При переходе к выполнению этого цикла установлены в единицу проверочные биты четности 3-й и 5-той строк: r_3=1 и r_5=1. Проверочные коды всех столбцов, кроме 3-го равны нулю. Для 3–го столбца c_3=0, V_3 = 0110. Эта ситуация полностью аналогична описанной выше в рамках примера 1. Соответственно корректировка двух искаженных битов производится по описанному в примере 1 способу. После выполнения цикла **A1** множество Ξ={3,5}. Эти строки могут быть переданы повторно для проверки правильности выполненной коррекции.

Анализ предложенного алгоритма коррекции ошибок показывает, что он позволяет гарантированно корректировать без повторной передачи искажения до 4-х бит передаваемого блока. Единственным вариантом локализации 4-кратных ошибок, при котором не обеспечивается их исправление без повторной передачи является ситуация, когда их матричные координаты могут быть представлены в виде: $<b_1, e_1>$,$<b_1, e_2>$, $<b_2,e_1>$, $<b_2,e_2>$, $b_1,b_2\in\{1,\ldots,m\}$, $e_1,e_2\in\{1,\ldots,n\}$. Поскольку количество вариантов такой локализации равно $m\cdot(m-1)\cdot n\cdot(n-1)/4$, то вероятность P_{NC4} появления такой локализации четверки искаженных бит определяется формулой:

$$P_{CN4} = \frac{m\cdot(m-1)\cdot n\cdot(n-1)\cdot 3!}{m\cdot n\cdot(m\cdot n-1)\cdot(m\cdot n-2)\cdot(m\cdot n-3)} \approx \frac{6}{m^2\cdot n^2} \qquad (5)$$

Анализ выражения (5) свидетельствует о том, что практически все 4-кратные ошибки также могут быть исправлены без повторной передачи.

Анализ достоверности коррекции ошибок

Исправление ошибок на основе анализа на приемнике проверочных кодов $\Delta_R=\{r_1,r_2,\ldots,r_m\}$, $\Delta_C=\{c_1,c_2,\ldots,c_m\}$ и $\Delta_V=\{V_1,V_2,\ldots,V_n\}$, как и при использовании всех корректирующих кодов, сопряжено с риском неверной классификации ошибок. Так, однократная ошибка передачи бита, локализованного в строке x и столбце y матрицы D, при которой r_y=1, c_x=1 и W_x=y по значениям проверочных кодов не отличима от 7-кратной ошибки, при которой 4 искаженных бита локализованы в одном столбце z, причем одна из этих четырех ошибок находится в строке y, а остальные три – в строках y_1,y_2,y_3, а другие три искаженных бита находятся в столбце x и строках y_1,y_2,y_3. Очевидно, что при такой локализации 7-ми искажаемых битов c_z=0, r_{y1}=0, r_{y2}=0, r_{y3}=0, c_x=1, r_y = 1, а $W_z = W_{y1} \oplus W_{y2} \oplus W_{y3} \oplus W_x$. Существует вероятность (численно равная $1/m$) того, что W_z= 0, тогда $W_z = W_{y1} \oplus W_{y2} \oplus W_{y3} = y$ и тогда такая 7-кратная ошибка по значению проверочных кодов не отличима от однократной. Проведя рассуждения, аналогичные использованным при выводе формулы (1) можно показать, что вероятность P_{S7} того, что возникшая 7-кратная ошибка будет иметь

проверочные коды совпадающие с однократной определяется выражением:

$$P_{S7} = \frac{105}{(m \cdot n)^4 \cdot n} \qquad (6)$$

Так, для n=8, m=1024, в соответствии с (6), вероятность $P_{S7} = 2.9 \cdot 10^{-15}$. Необходимо отметить, что рассмотренная конфигурация 7-кратной ошибки не может возникнуть в ”пачке”, вызванной внешней помехой. С учетом того, вероятность возникновения 7-кратной ошибки, в проводных линиях передачи данных достаточно мала.

Аналогично можно показать, при коррекции двукратной ошибки существует риск, связанный с тем, что такая ошибка, исходя из проверочных кодов не отличима от 6-кратной. Вероятность P_{S6} такого риска определяется вероятностью специфической локализации в матрице D 6-ти искаженных бит, которая определяется формулой:

$$P_{S6} = \frac{60}{(m \cdot n)^3 \cdot n} \qquad (7)$$

Соответственно, при коррекции 3-кратной ошибки существует риск не отличить ее от 5-кратной. Сопоставление формул (6) и (7) показывает, что вероятность риска неверной коррекции на основе проверочных кодов растет с увеличением количества s скорректированных битов матрицы D^R.

Устранение риска неверной коррекции достигается повторной передачей скорректированных строк матрицы D. Целесообразность повторной передачи определяется значением s, зависимостью вероятностей появления ошибок от их кратности для конкретного типа канала, требованиям к достоверности передачи данных.

Для оценки эффективности предложенного алгоритма коррекции ошибок проводилось статистическое моделирование его работы. Для матрицы D, состоящей из 32-х строк и 8-ми столбцов сгенерировано 40000 ошибок кратностей от 1 до 8. Полученные результаты приведены в таблице 1. Анализ результатов экспериментальных исследований показывает, что предложенный алгоритм коррекции без повторной передачи исправленных строк позволяет надежно корректировать ошибки, кратность которых не превышает 3-х.

Табл. 1. Результативность алгоритма коррекции ошибок без повторной передачи исправленных строк

Результат работы алгоритма коррекции	Кратность ошибок							
	1	2	3	4	5	6	7	8
Исправлены без повторной передачи (%)	100	100	100	99	94	81	58	38
Исправлено за счет повторной передачи (%)	0	0	0	1	5.5	16	35	50
Неверно исправлены без повторной передачи (%)	0	0	0	0	0.5	2	7	12

Выводы

В результате проведенных исследований, направленных на повышение эффективности обнаружения и коррекции ошибок передачи данных, предложена модификация TDCS. За счет использования взвешенных контрольных сумм, применение предложенной TDCS позволяет расширить класс гарантированно обнаруживаемых ошибок по сравнению с традиционной TDCS и CS. Разработан алгоритм коррекции ошибок для предложенной TDCS без повторной передачи блока данных. Выполненный теоретический анализ и экспериментальные исследования показали его высокую эффективность для исправления ошибок, кратность которых не превышает 3-х.

Предложенная модификация TDCS позволяет повысить эффективность обнаружения и коррекции ошибок в линиях с передачей бита четности.

Список использованной литературы:

1. Скляр Б. Цифровая связь. Теоретические основы и практическое применение. М.: Издательский дом ”Вильямс”, 2004.- 1104 с.
2. Петров С.В. Шины PCI, PCI Express. Архитектура, дизайн, принципы функционирования.-СПб.:БХВ-Петербург.-2006.-416 с.
3. Дансмор Б., Скандьер Т. Справочник по телекоммуникационным технологиям. М.: Издательский дом ”Вильямс”, 2004.-640 с.
4. Klove T., Korzhik V. Error Detecting Codes: General Theory and Their Application in Feedback Communication Systems. Norwell, MA: Kluwer, 1995. – 433 p.
5. Али Тауфик Окла Аль-Хавальди, Мулки Ахмед Яссин Ал Бадайнех, Антоненко А.А. Об одном подходе к повышению надежности ошибок передачи данных методом контрольных сумм // Вісник Національного технічного університету України ”КПІ” Інформатика, управління та обчислювальна техніка, – Київ: ВЕК+ – 2006 – № 45. с.27-35.

УДК 004.056.53

ИЯД МОХД МАДЖИД АХМАД ШАХРУРИ

ИСПОЛЬЗОВАНИЕ ХЕШ-ПАМЯТИ ДЛЯ БЫСТРОЙ АУТЕНТИФИКАЦИИ АБОНЕНТОВ МНОГОПОЛЬЗОВАТЕЛЬСКИХ СИСТЕМ

В статье предлагается подход к повышению скорости аутентификации абонентов многопользовательских систем. Этот подход имеет в основе использование хеш-памяти и учитывает многоуровневый характер организации памяти. Разработана вероятностная модель хеш-поиска в квазипостоянных массивах ключей, хранящихся в двухуровневой памяти. На основе этой модели предложена эффективная организация хеш-доступа к аутентификационной информации. Показано, что предложенная организация хеш-поиска требует не более одного обращения к медленной памяти нижнего уровня.

In paper an approach for increasing the rate of multi-user system abonent authentication has been proposed. This approach is based on utilization of hash-memory and takes into account multilevel modern memory organization. The probabilistic model of hash-searching in near-permanent keys arrays which are storages in two-level memory has been developed. On the base of this model the effectiveness organization of hash-access to authentication information has been proposed. It has been shown that proposed hash-access organization requires ensure no more then one access to slow low-level memory.

Введение

Одним из определяющих факторов прогресса во всех областях человеческой деятельности является развитие и расширение информационной интеграции на основе сетевых технологий. Компьютерные сети, предоставляя качественно новые возможности доступа к информации, позволяют заметно повысить эффективность интеллектуальной деятельности. В настоящее время процесс информационной интеграции продолжает динамично углубляться и охватывать новые области применения на основе расширения использования беспроводных линий передачи данных [1]. Тенденцией последних лет стал рост доли среди терминальных устройств сетей портативных микроконтроллеров: датчиков и управляющих устройств.

Важнейшим условием эффективности информационной интеграции на основе компьютерных сетей является обеспечение надежной защиты данных и разграничение прав доступа в интегрированной информационной среде. Расширение использования беспроводных технологий передачи данных, резкий рост числа абонентов компьютерных сетей, повышение производительности компьютерных систем, используемых для нарушения защиты – являются, в современных условиях, факторами повышения риска информационной безопасности интегрированных систем. С другой

стороны, процесс расширения использования интегрированных систем обработки данных все в большей степени охватывает области человеческой деятельности, связанные с техногенным и иными видами рисков. Указанные факторы определяют необходимость совершенствования средств защиты данных в интегрированных информационных средах и, в частности, повышения эффективности аутентификации удаленных абонентов.

В ряду основных направлений совершенствования аутентификации удаленных абонентов весомую роль играет повышение оперативности реализации функций разграничения прав доступа. Быстрый рост числа абонентов интегрированных систем повышает важность проблемы обеспечения высокой скорости их аутентификации.

Таким образом, актуальной и важной составляющей для современного этапа развития технологии информационной интеграции является повышение скорости аутентификации удаленных абонентов интегрированных систем хранения и обработки информации.

Анализ существующих решений проблемы эффективной аутентификации удаленных пользователей

Практически все современные многопользовательские системы имеют средства защиты от несанкционированного доступа к их ресурсам. В большинстве систем информация, связанная с предоставлением прав пользования ресурсами, представлена в форме закрытого пароля пользователя [2]. Обычно в системе имеется список легальных пользователей, в котором им сопоставляются пароль и перечни ресурсов, к которым возможен доступ. При этом пароль выполняет функции аутентификации пользователя, входящего в систему.

В частности, в широко известной системе UNIX [1] поиск выполняется по идентификатору пользователя, введенный им пароль шифруется с использованием модифицированного алгоритма DES и сравнивается с зашифрованным паролем, хранящимся в списке. Существует ряд многопользовательских систем для которых поиск производится по паролю [2].

При реализации аутентификации пользователей системы возникает технологическая проблема поиска по паролю или идентификатору пользователя. В большинстве систем способ реализации процедуры поиска не специфицирован. Для систем с небольшим числом пользователей поиск вполне может выполняться последовательно. Однако, с развитием информационной интеграции, растет число систем, количество легальных пользователей которых превышает десятки и сотни тысяч [1]. Большинство таких систем можно рассматривать как системы массового обслуживания, для которых важным показателем является время реакции на запрос Пользователя. Существует вполне реальная опасность уменьшения времени реакции системы на запросы пользователя, вызванная умышленным

или случайным ростом интенсивностей потока обращений со стороны несанкционированных пользователей. Следует иметь в виду еще один важный аспект рассматриваемой проблемы: повышение скорости аутентификации положительно сказывается на безопасности многопользовательских систем, поскольку ряд технологий незаконного проникновения в такие системы основаны на задержках аутентификации [3].

Поэтому, для рассматриваемого класса многопользовательских систем актуальной является проблема ускорения аутентификации пользователей за счет повышения скорости поиска по идентификатору или паролю.

Необходимо отметить, что проблема повышения эффективности аутентификации пользователей систем и сетей включает ряд других задач, решение которых может быть достигнуто за счет ускорения поиска в больших массивах данных. К таким задачам, в частности, относится проверка качества пароля при его генерации системой или выборе пользователем [4]. Одним из наиболее перспективных способов решения этой задачи, а также задачи селекции паролей, используемых для несанкционированного доступа к ресурсам системы, является использование списка потенциально опасных паролей. Для того, чтобы такой подход был достаточно эффективным, этот список должен быть достаточно большим и составлять более 1 Гбайта. Поиск в таком списке должен осуществляться при каждом цикле аутентификации пользователя, следовательно, процедуру поиска необходимо реализовать достаточно быстро.

Память всех современные компьютерных систем имеет виртуальную организацию, которая предполагает наличие быстродействующей кэш-памяти относительно небольшого объема и медленной основной памяти большого объема. Обмен данными между кэш-памятью и основной памятью производится страницами. Эту важную особенность необходимо учитывать при организации поиска пароля в списке тех, доступ к ресурсам для которых разрешен. Для интегрированных систем с большим числом абонентов списки паролей и соответствующих им кодов доступа хранятся в основной памяти. Организация хранения паролей должна быть выполнена таким образом, чтобы поиск идентифицирующей информации пользователя требовал перемещения в кэш-память не более, чем одной страницы. При этом полагается, что для реализации поиска в кэш-памяти выделяется определенное количество страниц. При поиске идентифицирующего кода на соответствие одному из хранящихся в памяти системы, этот код играет роль ключа поиска, поэтому в дальнейшем изложении целесообразно придерживаться традиционной для поиска терминологии, понимая под ключом поиска идентификатор или пароль удаленного пользователя. В общем виде можно полагать, что поиск ключа производится в массиве из m ключей, причем в одной странице можно максимально разместить w ключей вместе со связанной с ними информацией, используемой для указания прав доступа.

Основные особенности поиска при аутентификации пользователей состоят в следующем:

1. Интенсивность использования ключей для поиска на несколько порядков превышает интенсивности их использования, что позволяет считать массив ключей и связанной с ними информации квазипостоянным. Эта особенность характерна и для указанной выше задачи поиска пароля в списке потенциально опасных.

2. Для исключения несанкционированного доступа к паролям, хранящимся в памяти, необходимо исключить их хранение там в явном виде.

Приемлемая скорость сравнения паролей может быть достигнута при использовании двух технологий поиска:

1. Применение логического (с использованием бинарных деревьев) или физического упорядочения массивов ключей и бинарного поиска.

2. Применение хеш-адресации.

Реализация бинарного поиска с учетом перемещения в кэш-память не более одной страницы из основной памяти требует организации упорядоченного каталога страниц в кэш-памяти. В таком каталоге необходимо хранить коды первых ключей каждой страницы, то есть m/w ключей. Существенные затраты времени может потребовать изменение ключа или добавление нового. Для их уменьшения используют резерв [1] свободной памяти в каждой из страниц. Коэффициент β загрузки памяти определяется в виде отношения максимального числа w ключей, хранящихся в странице к числу w_R ключей, хранящихся в странице с учетом резерва: $\beta=w_R/w<1$. .

Существенно большую эффективность поиска по ключу обеспечивает хеш-адресация. При квазипостоянном характере массива ключей-паролей существует возможность подбора хеш-преобразования, не образующего коллизий, что обеспечивает потенциально наибольшую скорость поиска, поскольку гарантирует необходимость перемещения в кэш-память не более одной страницы. Присущая хеш-адресации неполная загрузка памяти позволяет достаточно просто организовать добавление новых ключей. Важным достоинством применения хеш-памяти по сравнению с упорядочением ключей представляется отсутствие необходимости в каталоге, занимающем кэш-память.

К настоящему времени подробно исследованы вопросы получения хеш-преобразования, не образующего коллизий для постоянных массивов ключей. Такие хеш-преобразования получили название совершенных (perfect hash transformation). В работе [5] намечены подходы к организации хеш-адресации динамических массивов ключей с учетом виртуальной организации хеш-памяти. Однако хеш-адресация квазипостоянных массивов ключей в системах с виртуальной памятью имеет ряд важных особенностей, не позволяющих прямо использовать существующие решения.

Таким образом, для достижения потенциально высоких характеристик поиска, обеспечиваемых применением хеш-адресации при аутентификации пользователей систем и сетей, необходимо решить ряд задач, связанных с организацией хеш-поиска с учетом специфики указанного ее применения. Решение этих задач требует проведения специальных исследований и может быть получено на основе модели хеш-поиска, учитывающей особенности задач аутентификации.

Целью работы является разработка математической модели хеш-поиска в квазистатических массивах идентификаторов пользователей, учитывающей особенности виртуальной организации памяти, а также создание на основе этой модели технологии аутентификации абонентов многопользовательских систем с использованием хеш-поиска.

Разработка модели хеш-поиска в квазистатических массивах

Задачей разрабатываемой модели хеш-поиска является получение аналитических зависимостей между характеристиками хеш-памяти, определяющими ее организацию, позволяющих эффективно решать задачи оптимизации структуры хеш-памяти на этапе ее проектирования исходя из заданных критериев эффективности хеш-поиска.

В основу разрабатываемой модели положена концепция получения хеш-преобразования $H(X)$, обеспечивающего отображение заданного множества Ω из m ключей на s страниц хеш-памяти таким образом, чтобы количество ключей, адресуемых в каждую из страниц, не превышало $(\alpha+\delta)\cdot w$, где α – коэффициент загрузки хеш-памяти, δ - допустимая вариация загрузки страницы хеш-памяти, $\alpha+\delta\leq 1$. Коэффициент α загрузки хеш-памяти определяется отношением количества m сохраняемых в ней записей к максимально возможному их количеству $M = s\cdot w$, исходя из объема хеш-памяти:.

$$\alpha = \frac{m}{M} = \frac{m}{s \cdot w} \qquad (1)$$

Под записью понимается информация о абоненте системы, сохраняемая в хеш-памяти. Запись содержит блок идентифицирующей информации и блок данных, связанных с пользователем. Этот блок содержит данные о правах доступа и другую, связанную с пользователем информацию. Вместо блока данных в записи может содержаться адресная ссылка на место его хранения.

Получение хеш-преобразования $H(X)$, удовлетворяющего указанному выше условию может быть получено путем подбора. В качестве механизма подбора хеш-преобразования целесообразным представляется использование стандартизированного шифроблока типа DES или Rijndael, который осуществляет криптографическое однозначное шифрование с использованием ключа K данных D в код C: $C = H_K(D)$ [6]. Ключевая для поиска информация X, в таком варианте, используется в качестве входных

данных шифроблока, ключ K шифроблока выступает в роли настроечного кода и является, по существу, номером хеш-преобразования. Выходной код C шифроблока разделяется на две части: h-разрядный фрагмент используется в качестве хеш-адреса $A_K(X)$ страницы, а оставшиеся разряды представляют собой хеш-свертку $S_K(X)$ ключа X поиска. Соответственно, подбор хеш-преобразования $H_K(X)$ выполняется путем изменения ключа K шифроблока.

Считая, что на странице может разместиться w ключей, хеш-адрес $A_K(X)$ страницы представляет собой $h=\log_2 s$ – разрядный код. Хеш-функция распределяет m ключей по s группам, которые содержат $\eta_1,\eta_2,\ldots,\eta_s$ ключей, причем $\sum_{j=1}^{s}\eta_j = m$. Для того, чтобы хеш-преобразование распределяло ключи по страницам хеш-памяти, необходимо, чтобы количество хеш-адресов, каждой из страниц не превышало максимально допустимого числа $u=(\alpha+\delta)\cdot w$ записей в странице: $\forall j\in\{1,\ldots,s\}$: $\eta_j \le u$. При этом, в каждой из страниц хеш-памяти обеспечивается наличие свободного объема памяти, достаточного для размещения $w\cdot(1-\alpha-\delta)$ записей.

Если исходить из того, что хеш-преобразование $H_K(X)$ формирует равномерно распределенные хеш-адреса, то в одну страницу, в среднем, попадает m/s хеш-адресов. В качестве теоретической модели распределения хеш-адресов представляется корректным использовать вероятностную модель Бернулли. Согласно этой модели, попадание хеш-адресов m ключей в пределы фиксированной страницы памяти можно рассматривать как m опытов, в каждом из которых с вероятностью $1/s$ происходит событие – попадание хеш-адреса в адресное пространство этой страницы. Тогда, согласно свойствам рассматриваемой модели, можно считать, что математическое ожидание числа хеш-адресов, попадающих в фиксированную страницу равно m/s с дисперсией $m\cdot\frac{1}{s}\cdot(1-\frac{1}{s})$. Тогда, согласно теореме Муавра-Лапласса, количество хеш-адресов, попадающих в рамки страницы, подчинено распределению Гаусса с математическим ожиданием m/s и дисперсией $m\cdot\frac{1}{s}\cdot(1-\frac{1}{s})$. Если учесть, что число s страниц хеш-памяти достаточно велико, то дисперсию числа хеш-адресов, попадающих в фиксированную страницу, можно приближенно считать равной m/s.

Вероятность P_{os} переполнения страницы, то есть, вероятность того, что число хеш-адресов, попадающих в фиксированную страницу хеш-памяти, превысит u, для нормального распределения с учетом (1) определиться следующим выражением:

$$P_{os} = 0.5-\Phi(\frac{u-m/s}{\sqrt{m/s}}) = 0.5-\Phi(\frac{w\cdot(\alpha+\delta)-m/s}{\sqrt{m/s}}) = 0.5-\Phi(\delta\cdot\sqrt{\frac{w}{\alpha}}) \qquad (2)$$

Для постоянного массива ключей, то есть, в ситуации, когда в странице нет необходимости иметь запас свободной памяти для $w\cdot(1-\alpha-\delta)$ записей, $\delta = 1-\alpha$ и вероятность P_{osw} переполнения страницы, то есть, вероятность того, что число хеш-адресов, попадающих в фиксированную страницу хеш-памяти, превысит w, определиться следующим выражением:

$$P_{osw} = 0.5 - \Phi(\sqrt{w}\cdot\frac{1-\alpha}{\sqrt{\alpha}}) \tag{3}$$

Из выражения (3) следует, что вероятность переполнения страницы в определяющей степени зависит от значения коэффициента δ запаса свободной памяти в странице, а также от коэффициента α заполнения хеш-памяти, а также от объема страницы и памяти, требующейся для хранения одной записи.

Для того, чтобы исключить переполнение всех s страниц при из заполнении фиксированным множеством Ω из m ключей, необходимо подобрать соответствующее хеш-преобразование. Вероятность P_0 того, что за одну пробу хеш-преобразования удастся исключить переполнение всех страниц хеш-памяти определяется произведением вероятностей того, что каждая из s страниц не будет переполнена:

$$P_0 = (1-P_{OS})^S = \left[0.5 + \Phi(\delta\cdot\sqrt{\frac{w}{\alpha}})\right]^S \tag{4}$$

Среднее число g проб, которое необходимо выполнить для подбора хеш-преобразования, при котором исключается переполнение страниц хеш-памяти, определяется следующим выражением:

$$g = \sum_{j=1}^{\infty} j\cdot P_0\cdot(1-P_0)^{j-1} = \frac{1}{P_0} \tag{5}$$

Проведенные экспериментальные исследования на основе статистического моделирования доказали адекватность изложенной математической модели хеш-адресации квазистатического массива ключей в хеш-памяти, разделенной на страницы. Предложенную модель можно использовать для оптимизации параметров хеш-памяти.

Из (5) следует, что при заданном числе g_z проб для подбора хеш-преобразования, обеспечивающего распределение записей по страницам с заполнением последних не более, чем на $(\alpha+\delta)\cdot 100\%$ значения α, δ и w должны удовлетворять условию:

$$\sqrt{\frac{w}{\alpha}}\cdot\delta = \Phi^{-1}(\sqrt[s]{\frac{1}{q_Z}} - 0.5) \tag{6}$$

Если необходимо обеспечить быстрый подбор хеш-преобразования, то можно исходить из условия $s\cdot P_{os} \approx 1$. Тогда при больших значениях s справедливо следующее приближение:

$$P_0 = (1 - P_{OS})^S = 1 - \sum_{i=1}^{s} (-1)^i \cdot \binom{s}{i} \cdot P_{OS}{}^i \approx 1 - \sum_{i=1}^{s} (-1)^i \cdot \frac{1}{i!} = 1 - \frac{1}{e} \approx 0.632 \cdot$$

соответственно, $q \approx 1/P_{os} = 1.58$. Это означает, что для того, чтобы обеспечить быстрый подбор хеш-преобразования, значения α, δ и w должны удовлетворять условию:

$$\Phi(\sqrt{\frac{w}{\alpha}} \cdot \delta) = 0.5 - \frac{1}{s} \qquad (7)$$

Анализ полученной математической модели показывает, что наиболее компромиссным вопросом организации хеш-поиска при аутентификации пользователей является выбор объема страниц, которыми производится обмен между основной и кэш-памятью. Очевидно, что скорость поиска существенно зависит от времени загрузки адресуемой хеш-адресом страницы основной хеш-памяти в кэш-память, которое, в свою очередь, зависит от размера страницы. Следовательно, с точки зрения достижения высокой скорости хеш-поиска, целесообразно уменьшать размер страницы, в то время как, с позиций снижения времени на подбор хеш-преобразования, согласно (5) оправдано увеличение размера страницы. Решение указанного компромисса может быть найдено исходя из заданной интенсивности обновления массива ключей: чем чаще изменяются ключи, тем меньшим должно быть время подбора хеш-преобразования и тем большим должна быть величина δ. Разрешение этих противоречивых требований можно достичь либо за счет увеличения размера страницы, либо за счет снижения коэффициента α загрузки основной хеш-памяти.

Организация хеш-поиска

Сущность организации хеш-поиска идентификатора пользователя состоит в следующем. Основной массив идентификаторов, в виде хеш-сверток и связанной с ними информации сохраняется по хеш-адресам в основной памяти. Подбором хеш-преобразования для квазипостоянного массива ключей обеспечивается распределение записей по страницам основной памяти с соответствии с хеш-адресами.

В кэш-памяти помещается активная страница основной памяти, а также область записей, которые добавляются в процессе регистрации новых пользователей и для которых не хватает места в соответствующих их идентификаторам страницах основной памяти.

Кроме того, в кэш-памяти может быть организовано хранение кодов хеш-преобразований, наиболее часто используемых записей.

Перед началом сеанса работы системы, либо в специально выделенное время обновления системы для множества Ω идентификаторов-ключей m пользователей подбором находится хеш-преобразование $H_K(X)$, которое распределяет записи пользователей по s страницам хеш-памяти таким образом, что в каждой из них имеется свободный объем, позволяющих

записать в каждую страницу не менее $w\cdot(1-\alpha-\delta)$ записей. Найденное подбором хеш-преобразование $H_K(X)$, в виде кода K ключа фиксируется и используется при хеш-поиске. Таким образом, множество Ω идентификаторов-ключей разделяется на s подмножеств, каждое из которых включает не более $w\cdot(\alpha+\delta)$ элементов. Это означает, в каждой странице сохраняется не более $w\cdot(\alpha+\delta)$ записей пользователей. Кроме того, шифроблоком формируется $\log_2 w$–разрядный хеш-адрес $U_K(X)$, определяющий адрес записи, соответствующей ключу X внутри страницы. Код хеш-свертки $S_K(X)$ вместе с хеш-адресом $A_K(X)$ страницы и хеш-адресом $U_K(X)$ внутри страницы, в силу однозначности преобразований, реализуемых шифроблоком, однозначно определяют идентификатор X пользователя. Внутри страницы записи размещаются по их хеш-адресам $U_K(X)$ с допустимостью возникновения коллизий, которые разрешаются с использованием известных технологий [6], в частности, пробинга.

Каждая из записей содержит код хеш-свертки $S_K(X)$ идентификатора, код пароля, коды прав доступа, другая информация, относящаяся к пользователю. Для сокращения длины записи, и, соответственно, увеличения w, вместо всех кодов, кроме $S_K(X)$ может храниться код адресной ссылки на место их хранения.

После сохранения в основной хеш-памяти записей всех, зарегистрированных на момент обновления пользователей, система их аутентификации готова к работе.

При удалении записи X производится ее исключение из страницы путем обнуления занимаемой ею памяти с учетом перемещения цепочки коллизий.

При добавлении новой записи X производится перемещение адресуемой $A_K(X)$ страницы из основной памяти в кэш-память. Если страница не заполнена, то по адресу $U_K(X)$ производится попытка дописать новую запись X. Если соответствующая область страницы занята, то выполняется пробинг до нахождения свободной области. Если страница, адресуемая $A_K(X)$ оказывается полностью занятой, то запись X, адресуемая частью разрядов хеш-адреса $A_K(X)$ помещается в специально отведенную область кэш-памяти, являющуюся, по-существу областью переполнения. Если адресуемая ячейка занята, то осуществляется разрешение коллизий с использованием одного из видов пробинга. При этом запись, помещаемая в область переполнения содержит не хеш-свертку идентификатора X, а его полный код. Если выделенная область переполнения оказывается полностью занятой, то необходимо произвести цикл обновления.

Технологически описанная процедура размещения в хеш-память новой записи может производиться двумя способами, отличающихся приоритетом при размещения записи в одной из двух областей. При регистрации новых пользователей в период между циклами обновления системы аутентификации, соответствующие записи могут быть помещены:

- в область переполнения относительно небольшого объема, локализованной в кэш-памяти;
- в основной хеш-памяти, занимая свободные зоны страниц, адресуемых хеш-адресом идентификатора вновь регистрируемого пользователя.

Теоретически можно рассмотреть два варианта организации дописывания:

1. Вначале записи дописываются в свободные зоны соответствующих страниц основной памяти, а при отсутствии в таких зон – в область переполнения кэш-памяти.

2. Вначале в область переполнения кэш-памяти, а по заполнении последней – в свободные зоны соответствующих страниц основной хеш-памяти.

Очевидно, что первый вариант обеспечивает большее количество новых записей, которые могут быть помещены в хеш-память до ее обновления. Преимуществом второго варианта является более эффективное использование объема области переполнения кэш-памяти, что имеет следствием большую скорость аутентификации пользователей.

При поступлении n-разрядного идентификатора X пользователя шифроблоком с использованием в качестве ключа кода K выполняется вычисление h-разрядного хеш-адреса $A_K(X)$ страницы, хеш-адреса внутри страницы $U_K(X)$ и кода хеш-свертки $S_K(X)$. Формируется также хеш-адрес $Q_K(X)$ записи с идентификатором X в области переполнения кэш-памяти. Адрес $Q_K(X)$ представляет собой подмножество разрядов кодов $A_K(X)$, $U_K(X)$ и $S_K(X)$. По адресу $Q_K(X)$ производится хеш-поиск в обрасти переполнения кэш-памяти. Если поиск завершается успешно, то есть в области переполнения найдена запись с идентификатором X, то пользователю разрешается доступ к ресурсам системы в пределах выделенных прав.

Одновременно с поиском в области переполнения кэш-памяти, страница, адресуемая кодом $A_K(X)$ загружается из основной памяти в кэш-память. Затем по адресу $U_K(X)$ из страницы считывается запись, ее хеш-свертка сравнивается с кодом $S_K(X)$. В случае совпадения указанных кодов выполняется сравнение паролей и осуществляется предоставление прав доступа. В случае несовпадения кодов хеш-сверток выполняется пробинг до нахождения совпадения либо пустой записи. Последнее означает, что запись с ключом X не хранится в памяти. Время t_1 поиска по ключу-идентификатору X пользователя определяется суммой времен: t_H – выполнения хеш-преобразования с использованием шифроблока, t_S - времени свопинга адресуемой $A_K(X)$ страницы из основной памяти в кэш-память, t_X -временем хеш-поиска в выбранной странице кэш-памяти: $t_1 = t_H + t_S + t_X$. Учитывая, что $t_S > t_H >> t_X$, время аутентификации, в основном, определяется временем, затрачиваемым на свопинг страницы из основной памяти в кэш-память.

Выводы

Итогом проведенных исследований, направленных на повышение эффективности аутентификации удаленных абонентов многопользовательских систем за счет ускорения поиска по идентифицирующему коду являются следующие результаты:

1. Выявлены специфические особенности поиска по идентифицирующему коду удаленного пользователя.

2. Разработана модель хеш-поиска в квазистатических массивах идентификационных кодах пользователей, учитывающая многоуровневую организацию памяти.

3. На основе разработанной модели предложена организация хеш-поиска в квазипостоянных массивах идентифицирующих пользователей кодов. Доказано, что время аутентификации определяется обращением не более, чем одной странице памяти нижнего уровня.

Предложенная организация хеш-памяти может быть эффективно использована для повышения скорости аутентификации пользователей как компьютерных систем коллективного пользователей, так и абонентов сетей.

Список использованной литературы

1. Столлингс В. Криптография и защита сетей: принципы и практика. М.: Изд.дом."Вильямс".- 2001-621 с.
2. Соколов А.В., Шаньгин В.Ф. Защита информации в распределенных корпоративных сетях и системах. М.: ДМК.-2002.- 655 с.
3. Шнайер Б. Прикладная криптография. Протоколы, алгоритмы, исходные тексты на языке Си. М.:"Триумф". - 2002.-816 с.
4. Stafford E. Preventing Weak Password Choices // Computers and security.-1992.-№ 3.- P.46-53.
5. Мнацаканов А.В. Организация хеш-поиска в системах с виртуальной памятью // Матеріали VII Міжнародної науково-технічної конференції "Системний аналіз та інформаційні технології" (13-16 вересня 2006 р., м.Київ). К.: НТУУ "КПІ".-2006.- С.208-211.
6. 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.

УДК 004.7

КОРОЛЬКОВ І.В.

СКОРОЧЕННЯ ДОВЖИН МАРШРУТІВ, ОТРИМАНИХ МЕТОДОМ АЛГОРИТМІЧНОЇ МАРШРУТИЗАЦІЇ, ШЛЯХОМ МОДИФІКАЦІЇ ТОПОЛОГІЧНИХ ДЕРЕВ

Розглянуто метод алгоритмічної маршрутизації як механізм абстракції при моделюванні маршрутизації в обчислювальних мережах. Запропоновано два ітераційних метода поліпшення дерева топології мережі. Перший метод передбачає перевірку кожної дійсної модифікації, після чого модифікація відразу ж приймається або відхиляється. Другий метод включає перевірку вузла та відновлення усіх можливих розірваних зв'язків, після чого модифікація приймається або відхиляється на підставі підсумкової зміни загальної кількості пересилань, необхідних для з'єднання усіх пар окремих вузлів у мережі, далі процедура повторюється для кожного вузла. Проведено аналіз можливості застосування обох методів, а також їх комбінації для модифікації маршрутних дерев мереж різних розмірів.

The Algorithmic Routing method as the abstraction technique for computer networks simulation is considered. Two iterative methods of topology trees improvement are proposed. The first method tests each modification and accepts or rejects it immediately. The second method examines a node, reconnects all possible broken links and after that accepts or rejects the modification based on the overall change in the sum total of the number of hops needed to join all distinct pairs of nodes in the network, and than repeats this for every node. The analysis of routing trees modification for networks of different sizes by both methods and its combination is done.

Вступ

Ключова особливість маршрутизації в Інтернет полягає в тому, що кожен маршрутизатор підтримує свою власну таблицю маршрутизації, навіть якщо вона побудована за допомогою обміну повідомленнями зі своїми сусідами. Тим не менш, на відміну від проектування реального маршрутизатора, при моделюванні можна не брати до уваги деякі наявні обмеження.

Метод алгоритмічної маршрутизації (АМ) було висунуто в [1], далі розвинуто у [2]. Ідея методу заснована на роботі з мережами бінарних дерев [3] та передбачає зниження використання пам'яті; при цьому дещо зростає обчислювальна складність.

Алгоритмічна маршрутизація передбачає накладення вихідної топології мережі на *k*-е дерево. Це дерево може бути сформовано шляхом пошуку в ширину (ПВШ) чи в глибину (ПВГ) мережі. При цьому вибір алгоритму пошуку впливає на точність маршрутизації. Значення *к* відповідає

максимальній кількості потомків, які охоплені будь-яким вузлом у дереві. Якщо в мережі наявні петлі, вони руйнуються на етапі відображення.

У [4] нами було запропоновано декілька модифікацій вихідного методу АМ для поліпшення продуктивності та масштабованості моделювання великих обчислювальних мереж. Однак слід відмітити, що отримані маршрути не завжди є найбільш короткими, в них можуть виникати деякі спотворення.

У мережі може існувати безліч петель, при цьому кожна петля робить вклад у збільшення середньої відносної різниці довжин маршрутів *R*. Також важливим є показник довжини петель – взаємний вплив та накладення петель ускладнюють завдання аналізу порівняно з простою кільцевою мережею. На додачу до кількості та розміру петель фактор їх розташування в мережі також має значний вплив на величину *R*. З одного боку, якщо петлі перебувають на периферії дерева, вони будуть впливати тільки на збільшення довжин маршрутів, які закінчуються там. З іншого боку, оскільки більшість маршрутів перетинають магістральні канали мережі, петлі, які перебувають на основних магістралях, будуть справляти значний вплив на *R*. Також, оскільки між будь-якими двома гілками дерева існує тільки один шлях, трафік може концентруватися на деяких лініях зв'язку та маршрутизаторах, у той час як інші будуть простоювати.

Постановка завдання

Вибір кореневого вузла в дереві, отриманому шляхом ПВШ, справляє вплив як на якість отриманих маршрутів, так і на обчислювальну складність операції розрахунку наступного пересилання. Як корінь доцільно обрати вузол, при якому середня глибина дерева буде мінімальною, або перебудувати дерево після його формування.

Нехай *H* – загальна кількість пересилань, необхідних для з'єднання всіх пар окремих вузлів у мережі:

$$H = \sum_{i=1}^{N} \sum_{j=i+1}^{N} Dist(i,\ j).$$

На прикладі декількох мереж різного розміру в табл. 1 наведені середні значення *H*/*D*. Тут *H*/*D* являє собою відношення загальної кількості пересилань у маршруті до кількості пересилань у найкоротшому маршруті та є показником якості маршрутів у дереві. В усіх випадках пріоритет при формуванні маршрутного дерева методом ПВШ було надано візлам з високою зв'язністю. Відносна якість маршрутів варіюється у межах $1{,}1 \div 1{,}27$ від довжини найкоротшого маршруту. На жаль, явного розв'язку задачі вибору кореневого вузла та порядку ПВШ для мінімізації H не існує.

Табл.1. Приклад значень *H*/*D* для мереж,
які включають від 100 до 6400 вузлів

Кількість вузлів	Середнє *H/D*
100	1,23
200	1,10
400	1,12
800	1,11
1600	1,16
3200	1,17
6400	1,27

Знаходження оптимального дерева для АМ є окремим випадком більш загальної проблеми теорії графів, відомої як задача оптимально комунікативного зв'язного дерева (Optimal Communication Spanning Tree, OCST), висунута у [5].

Формально: неорієнтований граф G заданий як $G = (V, E)$, де V – множина вершин, а E – множина ребер, які з'єднують усі пари вершин. Кількість вершин позначено як $n = |V|$. Ребро між вершинами v_i і v_j позначено як (v_i, v_j). Зв'язне дерево графа G визначено як $T = (V, F)$, де $F \subseteq E$, $|F| = |V| - 1$ і всі вершини зв'язані.

Розв'язання задачі OCST передбачає знаходження зв'язного дерева для графа G, що відповідає певним вимогам. Матриця запитів $R = (r_{ij})$ визначає кількість трафіка між парами вершин. Матриця R має розмір $n \times n$, а r_{ij} – трафік, який підлягає передачі між вузлами v_i і v_j. Матриця відстаней $W = w_{ij}$ також має розмір $n \times n$ та містить у собі вагові коефіцієнти відстаней між кожною парою вузлів. Нехай $P_{ij}(T) \subseteq F$ - множина ребер, які з'єднують вершини v_i і v_j у дереві T. Вага зв'язного дерева $w(T)$ визначено як

$$w(T) = \sum_{v_i, v_j \in V} \left(r_{ij} \sum_{(v_p, v_q) \in P_{ij}(T)} w_{pq} \right).$$

Дерево T є розв'язком задачі OCST, якщо $w(T) \leq w(T')$ для всіх інших зв'язних дерев T'.

Оптимальне маршрутне дерево для АМ є окремим розв'язком задачі OCST. Нехай G_n – граф топології мережі, у якому ребра E_n відповідають лініям зв'язку. У вихідну задачу OCST вводяться додаткові умови:

$|G| = |G_n|$,

$r_{ij} = 1$, якщо $i < j$,

$r_{ij} = 0$, якщо $i \geq j$,

$w_{ij} = 1$, якщо $(v_i, v_j) \in E_n$,

$w_{ij} = \infty$, якщо $(v_i, v_j) \notin E_n$.

Як і раніше, G – неорієнтований граф.

Основна задача OCST є *NP*-повною [6]. Сучасні підходи до розв'язання задачі OCST базуються на еволюційних алгоритмах [7]. Але розв'язки задачі OCST обмежились невеликими графами, які зазвичай містять у собі

менше ніж 25 вершин [8]. Такі підходи не дають змоги оперувати великими графами, які використовуються в алгоритмічній маршрутизації.

Нещодавні дослідження були спрямовані на пошук алгоритмів для отримання розв'язків, близьких до оптимальних, але запропоновані алгоритми не підходять для графів великих розмірів, оскільки потребують розрахунку найменших маршрутів для всіх пар вузлів, що обчислювально не є доцільним.

Запропонований далі ітераційний метод формування дерев для АМ, на відміну від згаданих вище, не гарантує перебування H у заданих межах, але дає змогу оперувати графами більших розмірів. На кожному кроці проводиться модифікація дерева, метрикою при цьому є H: якщо H зменшується, модифікація зберігається, інакше вона скасовується.

Формування зв'язного дерева для отримання найкоротших маршрутів

Мережа сама по собі не може бути модифікована. Зміні може бути підданий лише спосіб відображення топології на дерево. Якщо мережа містить у собі петлі, то є більше ніж один спосіб відображення на дерево, який впливає на значення H. Розглянемо невелику мережу на рис. 3а. Ця мережа може бути відображена на дерево методом ПВШ двома способами, як показано на рис. 3б і 3в. Обидва дерева дійсні, але мають різні значення H. Проте, призначуючи вузол z замість вузла y предком вузла x, можна перейти від одного дерева до іншого. При цьому зміни не мають уводити петлі чи робити дерево не повністю зв'язаним.

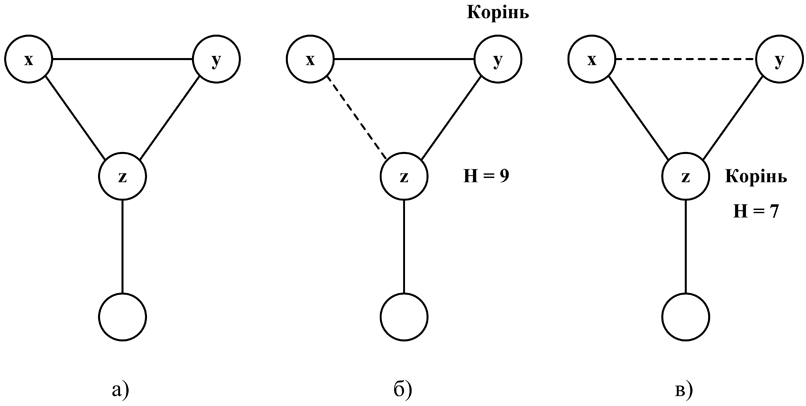


Рис. 3. Приклад альтернативного вибору кореневого вузла у простій мережі

Як і раніше, $G(V, E)$ – граф топології мережі, а (T, r) – маршрутне дерево, в якому T – зв'язне дерево, а r – кореневий вузол. $S(T, a)$ – множина вузлів у піддереві T з коренем в a. Нехай $x, y, z \in V$, а y – потомок x.

Позначимо модифікацію дерева як

$$T_{new} = \frac{T}{\{(x, y)\} \cup \{(z, y)\}},$$

де $z \in S(T, x)$. Позначимо значення H для дерев T і T_{new} як $H(T)$ і $H(T_{new})$ відповідно.

Розглянемо два ітераційних методи поліпшення дерева АМ. Перший метод передбачає перевірку кожної дійсної модифікації, після чого модифікація відразу приймається або відхиляється. Другий метод включає перевірку вузла та відновлення усіх можливих розірваних зв'язків, після чого модифікація приймається або відхиляється на підставі загальної зміни H, потім процедура повторюється для кожного вузла. Алгоритми цих двох методів наведено на рис. 5 і 6 відповідно.

```
   Задано. Граф G = (V, E). Маршрутне дерево T = (V, F, r), де
F ⊆ E. H(T) – сума відстаней між усіма невпорядкованими парами
вузлів у дереві T.
   Визначити. Маршрутне дерево Tnew, для якого H(Tnew) ≤ H(T).
   (1) Ttest ← T
   (2) Tnew ← T
   (3) foreach z ∈ V
   (4)   foreach (x, y) ∈ Fnew, (x, y) ∈ E, z ∉ S(x, Tnew)
   (5)     Ttest ← Tnew / {(x, y)}∪{(x, z)}
   (6)     if H(Ttest) ≤ H(Tnew)
   (7)       Tnew ← Ttest
   (8) return Tnew
```

Рис. 5. Ітераційний алгоритм, у якому кожна модифікація відразу приймається або відхиляється

```
   Задано. Граф G = (V, E). Маршрутне дерево T = (V, F, r), де
F ⊆ E. H(T) – сума відстаней між усіма невпорядкованими парами
вузлів у дереві T.
   Визначити. Маршрутне дерево Tnew, для якого H(Tnew) ≤ H(T).
   (1) Ttest ← T
   (2) Tnew ← T
(3)    foreach z ∈ V
(4)      foreach (x, y) ∈ Fnew, (x, y) ∈ E, z ∉ S(x, Tnew)
   (5)     Ttest ← Tnew / {(x, y)}∪{(x, z)}
(6)      if H(Ttest) ≤ H(Tnew)
   (7)     Tnew ← Ttest
(8)    return Tnew
```

Рис. 6. Ітераційний алгоритм, у якому одночасно перевіряються декілька модифікацій

Для вироблення рішення щодо прийняття або відхилення кожної модифікації дерева необхідно визначати зміну величини H. У мережі з

$b = |E| - |V| - 1$ розірванними зв'язками необхідно $O(Nb)$ розрахунків, що обчислювально є дуже витратним, якщо значення b є великим.

Нехай $d(x, y)$ – відстань між вузлами x та y уздовж маршруту, обраного методом АМ. Для двох множин вузлів X та Y нехай $D(X, Y)$ – сума довжин маршрутів, які з'єднують кожну пару вузлів x, y ($x \in X, y \in Y, x \neq y$):

$$D(X, Y) = \sum_{x \in X,\, y \in Y} d(x, y) - \frac{1}{2} \sum_{x, y \in Y \cap X} d(x, y). \quad (1)$$

Другий член цього виразу введено для компенсації подвійного підрахунку вузлів, які належать як X, так і Y.

Необхідна наявність можливості швидкого оновлення значення H, коли у піддереві з коренем в a, предком b та розірваним зв'язком з вузлом c розривається зв'язок з b і c стає предком (передбачається, що результатом цієї модифікації є повнозв'язне дерево).

Нехай $S(a, T)$ – множина вузлів у піддереві дерева T з коренем в a. Нехай $S'(a, T) = V \backslash S(a, T)$. Нехай T_1 – маршрутне дерево мережі. Для стислості запишемо $S_1 = S(a, T_1)$ и $S'_1 = S'(a, T_1)$. Нехай a – вузол з предком b в дереві T_1.

Тоді

$$H(T_1) = \frac{1}{2} \sum_{x, y \in V} d(x, y) = \frac{1}{2} \left[\sum_{x, y \in S_1} d(x, y) + 2 \sum_{x \in S_1,\, y \in S'_1} d(x, y) + \sum_{x, y \in S'_1} d(x, y) \right]. \quad (2)$$

Оскільки S_1 і S'_1 є неперетинними (непересічними) множинами, якщо $x \in S_1$ і $y \in S'_1$, тоді:

$$d(x, y) = d(x, a) + d(a, b) + d(b, y) = d(x, a) + 1 + d(b, y) \quad (3)$$

і

$$\sum_{x \in S_1, y \in S'_1} d(x, y) = N(S_1) N(S'_1) + \sum_{x \in S_1, y \in S'_1} \left(d(x, a) + d(b, x) \right) \quad (4)$$

$$= N(S_1) N(S'_1) + N(S'_1) \sum_{x \in S_1} d(x, a) + N(S_1) \sum_{y \in S'_1} d(b, y)$$

$$= N(S_1) N(S'_1) + N(S'_1) D(\{a\}, S_1) + N(S_1) D(\{b\}, S'_1)$$

Використовуючи вирази 1, 2 і 4, отримаємо:

$$H(T_1) = D(S_1, S_1) + D(S'_1, S'_1) + N(S_1) N(S'_1) + N(S'_1) D(\{a\}, S_1) + N(S_1) D(\{b\}, S'_1) \quad (5)$$

Нехай T_2 – така модифікація дерева T_1, що c є предком a, де $c \notin S_1$. Передбачається, що $S_2 = S(a, T_2) = S_1$. Позначимо функції відстаней мереж для дерев T_1 і T_2 як $H(T_1)$ і $H(T_2)$ відповідно. Тоді $H(T_2)$ може бути представлене аналогічно виразу 5:

$$H(T_2) = D(S_1, S_1) + D(S_1', S_1') + N(S_1)N(S_1') + N(S_1')D(\{a\}, S_1) + N(S_1)D(\{c\}, S_1'). \quad (6)$$

Враховуючи 5 і 6, отримаємо:

$$\Delta H = H(T_2) - H(T_1) = N(S_1)[D(\{c\}, S_1') - D(\{b\}, S_1')], \quad (7)$$

де:

$$D(\{b\}, S_1') = D(S_1 \cup S_1', \{b\}) - D(S_1, a) - N(S_1),$$

$$D(\{c\}, S_1') = D(S_1 \cup S_1', \{c\}) - N(S_1)d(b, c) - N(S_1) - D(S_1, a).$$

Таким чином, отримаємо:

$$\Delta H = N(S_1)[D(V, c) - D(V, b) - N(S_1)d(b, c)]. \quad (8)$$

Практичні результати

На рис. 7–9 наведені результати застосування алгоритмів поліпшення на вихідному дереві, сформованому методом ПВШ. Для кожної мережі застосовувались три методи. Необхідно відзначити, що поліпшення Н, описані нижче, будуть варіюватися від мережі до мережі. Отримані результати не є показовими для всіх мереж цього розміру, а лише на прикладі демонструють деяку типовість поведінки та характеристики масштабованості. Далі в цьому розділі будуть використані більш реалістичні мережі, отримані на основі знімків Інтернет та за допомогою сучасних засобів генерування мереж.

Перший метод (рис. 7а, 8а, 9а) відповідає описаному раніше алгоритму, наведеному на рис. 5. Провадиться інтеграція за кожним вузлом у дереві, яка робить цей вузол предком будь-яких сусідніх вузлів, з якими він зв’язаний розірваним зв’язком, потім модифікація негайно приймається або відхиляється. Загальне поліпшення залежить від вихідного дерева, яке на кожному кроці залежить від вибору кореневого вузла. Н приймає мінімальне значення з випадковим локальним максимумом. Цей максимум відбувається, коли дві чи більше зміни разом викликають зниження Н, а окремо спричиняють підвищення, а отже, відхиляються алгоритмом.

Другий метод (рис. 7б, 8б, 9б) відповідає описаному раніше алгоритму, наведеному на рис. 6. У цьому випадку вузол стає предком для всіх можливих сусідніх вузлів і загальна модифікація дерева приймається або відхиляється. Результати не дуже відрізняються від результатів першого методу. Локальний мінімум Н все ще наявний.

Третій метод є більш цікавим. Він являє собою комбінацію двох перших методів. Спочатку застосовується перший метод, далі другий, а потім повторно перший метод. Результати, наведені на рис. 7в, 8в і 9в, помітно перевершують попередні – локальних мінімумів значно менше.

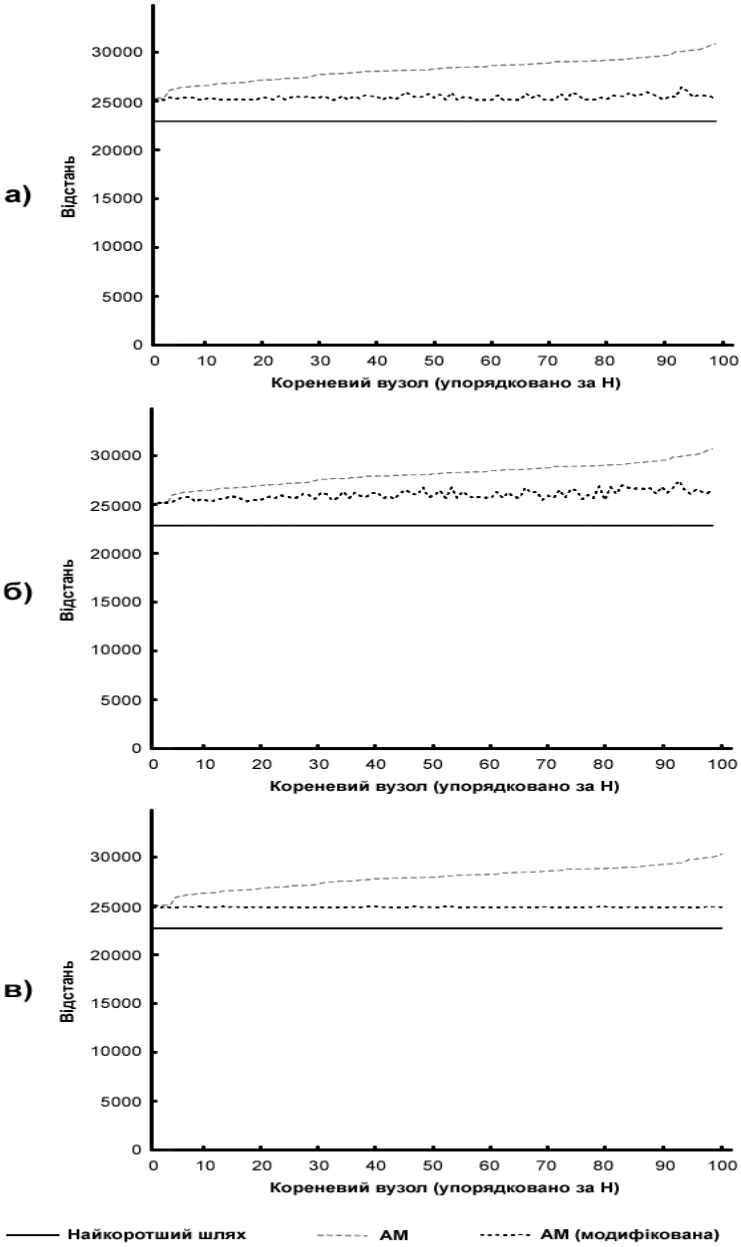


Рис. 7. Приклад поліпшення H для мережі, яка містить у собі 100 вузлів

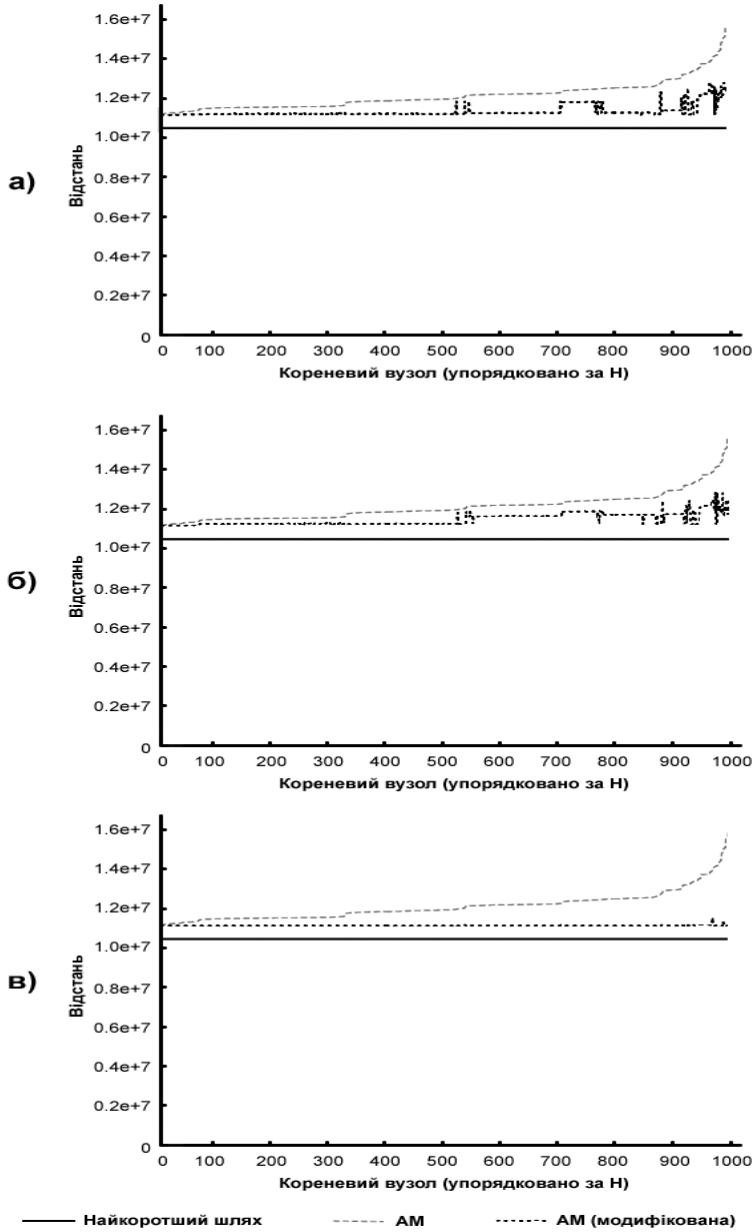


Рис. 8. Приклад поліпшення H для мережі, яка містить у собі 1500 вузлів

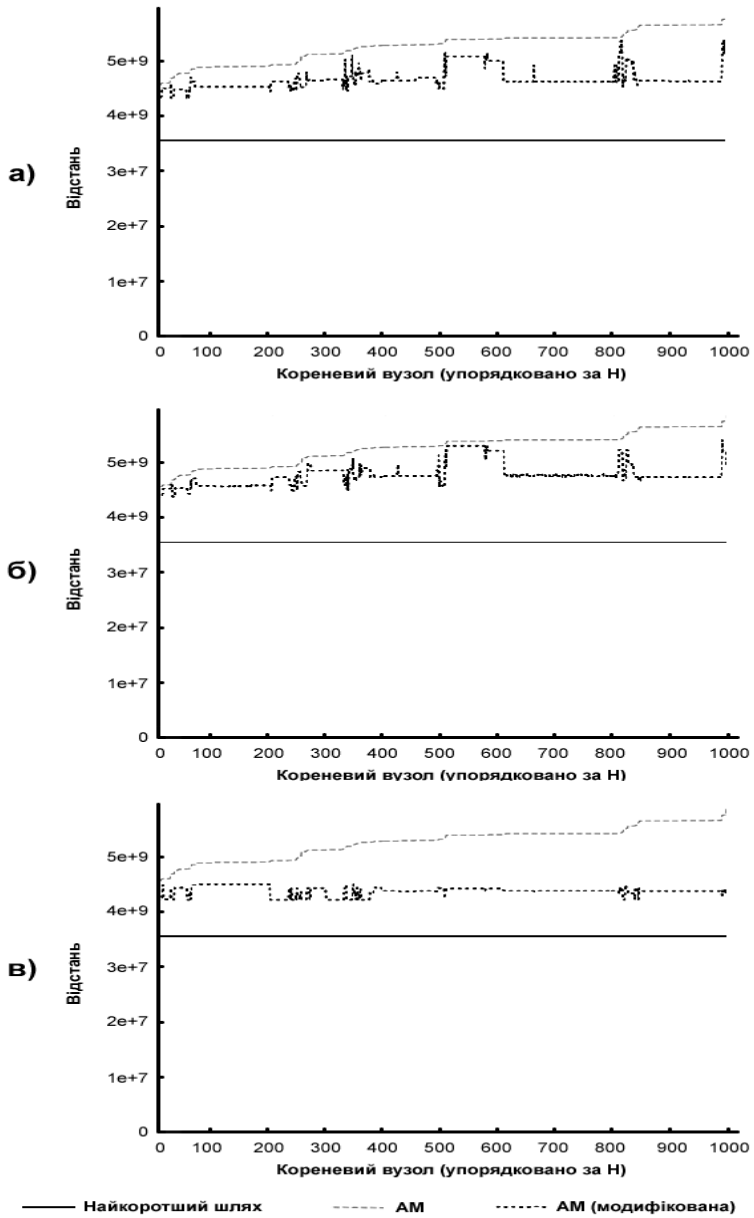


Рис. 9. Приклад поліпшення H для мережі, яка містить у собі 25000 вузлів

У середньому для розглянутих мереж алгоритми поліпшення знижують значення H у межах 6 ÷ 17 відсотків. Хоча таке поліпшення і не є настільки значним, його невелика обчислювальна складність виправдовує хоча б один прохід процесу поліпшення.

Час, необхідний для модифікації дерева за допомогою алгоритмів, описаних вище, суттєво не відрізнявся.

Рис. 10 ілюструє витрати часу на модифікацію дерев ПВШ мереж різних розмірів. Ці показники були отримані шляхом поодинокого вимірювання та ілюструють порядок величини часу, а не конкретне значення продуктивності. Для виконання тестів використовувався ПК на базі процесора AMD Athlon 1800МГц з 512Мб ОЗП під управлінням ОС FreeBSD 5.3 з компілятором GCC 3.4.2.

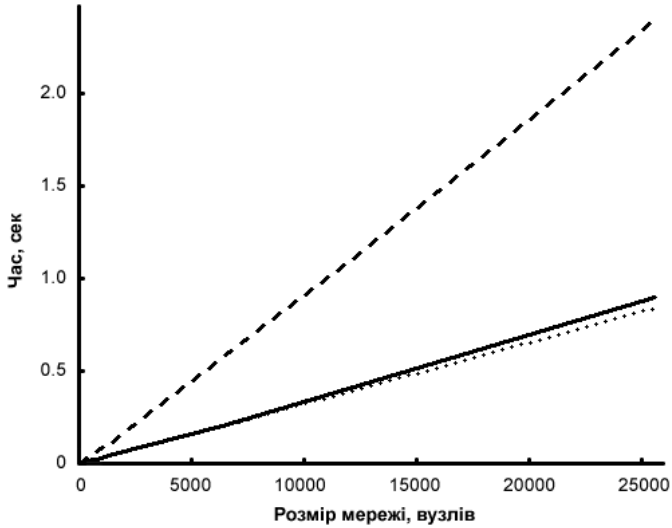


Рис. 10. Час, витрачений на поліпшення якості маршрутного дерева

Висновки

Важливою складовою моделювання обчислювальних мереж є моделювання механізмів маршрутизації. При виконанні моделювання мають бути збалансовані такі фактори: швидкість, точність і масштабованість. Найбільш точний метод передбачає докладне моделювання протоколів маршрутизації для кожного вузла мережі. Проте, якщо кожен з модельованих маршрутизаторів підтримує повні таблиці маршрутизації, значні вимоги щодо пам’яті перешкоджають моделюванню великих мереж. Для вирішення цієї проблеми у модель може бути введений певний рівень абстракції.

Метод АМ передбачає відображення мережі на дерево та використовує простий алгоритм розрахунку маршруту між двома заданими вузлами. При цьому отримані маршрути не завжди є найбільш короткими, у них можуть виникати деякі спотворення.

У цій статті запропоновані швидкі та ефективні методи підвищення якості маршрутних дерев. Модифікація дерев, які використовуються в АМ, дає можливість помітно скоротити довжину маршрутів, що отримуються, а це особливо важливо у завданнях моделювання надвеликих обчислювальних мереж. У статті продемонстровано результати практичного застосування описаних методів для модифікації маршрутних дерев мереж різних розмірів. Відносна простота методів дає змогу застосувати їх у випадках, коли використання АМ декількох дерев обчислювально не є виправданим.

Перелік посилань

1. *Huang P.* Enabling Large-scale Network Simulations: A Selective Abstraction Approach. PhD thesis, University of Southern California, 1999. – P. 129.
2. *Huang P., Heidemann J.* Minimizing routing state for light-weight network simulation // In Proceedings of the International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems, Cincinnati, Ohio, USA. - August 2001. – P. 108 – 116.
3. *Raman S., McCanne S., Shenker S.* Asymptotic scaling behavior of global recovery in SRM // In Proceedings of SIGMETRICS/PERFORMANCE 98, Joint International Conference on Measurement and Modeling of Computer Systems. - 1998. – P. 90 – 99.
4. *Корольков И. В.* Повышение производительности моделирования маршрутизации в больших вычислительных сетях // Радіоелектронні і комп’ютерні системі. – 2007 - №7(26). – С. 20 – 26.
5. *Hu T. C.* Optimum communication spanning trees // SIAM Journal of Computing. - 3(3). - 1974. - P. 188 – 195.
6. *Кормен Т., Лейзерсон Ч., Ривест Р.* Алгоритмы: построение и анализ. - М.: МЦНМО, 1999. – С. 1085 – 1151.
7. *Li Y., Bouchebaba Y.* A new genetic algorithm for the optimal communication spanning tree problem // In Artificial Evolution, volume 1829 of Lecture Notes in Computer Science. - 1999. – P. 162 – 173.
8. *Rothlauf F., Gerstacker J., Heinz A.* On the optimal communication spanning tree problem. IlliGAL Report No. 2003015. - May 2003. – P. 12

УДК 004.415.2

АВРАМЕНКО Е.А.

СРЕДСТВО РЕДОКУМЕНТИРОВАНИЯ НАСЛЕДУЕМОГО ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

В статье рассматривается метод редокументирования наследуемого программного обеспечения, ориентированный на технологию разработки; модель документа программного обеспечения. Предложен язык описания документов для редокументирования RDDL.

The article considers development technology oriented method of legacy software redocumentation; software document model It is proposed redocumentation document description language RDDL.

Введение

Редокументирование – это создание документации наследуемого программного обеспечения (ПО), отвечающей установленным требованиям или стандартам [1]. Задача редокументирования ПО возникла в 80-х годах прошлого столетия в связи с возросшим количеством наследуемого ПО, требующего реинженерии [1-3]. Такое ПО, чаще всего, не имеет документации или имеющаяся документация является устаревшей и неполной.

Существующие методы редокументирования ПО (например, структурное редокументирование [4], инкрементное редокументирование на основе Web-технологий [5]) направлены на решение задачи автоматизации отдельных аспектов документирования. Они также не учитывают требований к составу, структуре и содержанию документов.

В статье предлагаются средства для реализации метода редокументирования, применяемого при реинженерии наследуемого ПО. Средства позволяют повысить эффективность редокументирования путем автоматизированного создания документов, отвечающих требованиям технологии и фазы, выбранных для прямой инженерии. Рассмотрен язык описания документов Redocumentation Document Description Language (RDDL), построенный на основе модели документа ПО и обеспечивающий проведение редокументирования.

1. Метод редокументирования

Сущность метода редокументирования состоит в создании документации наследуемого ПО, состав, структура и содержание которой регламентируются требованиями технологии разработки, выбранной для прямой инженерии.

Состав документации при редокументировании определяется особенностями фазы, начиная с которой проводится прямая инженерия ПО. Структура и содержание документов определяются требованиями выбранной технологии разработки. Содержимое документов формируется на основе результатов обратной инженерии.

Реализация метода состоит в следующем. Для каждой технологии создаются комплекты метаописаний документов, относящихся к фазам (рис.1).

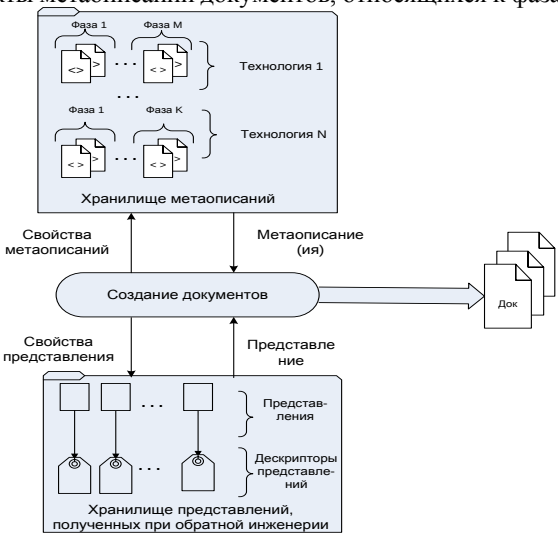


Рис.1. Схема использования метаописаний при редокументировании

Представления, извлеченные из наследуемого ПО с помощью обратной инженерии, снабжаются дескриптором и размещаются в хранилище представлений. По метаописанию и представлению для заданных технологии и фазы генерируется документ.

2. Модель документа ПО

Документ – это материальный объект, содержащий информацию в зафиксированном виде и специально предназначенный для её передачи во времени и пространстве [6]. В отраслевых стандартах документ ПО определяется как уникально идентифицируемая единица информации, разработанная для определенного круга пользователей с определенной целью и записанная на любом носителе информации [7-9].

Документ ПО D представлен тройкой $D = < S, C, P >$, где S – структура документа, C – содержимое документа (информационное наполнение), P – внешнее представление документа.

Структура документа S представлена множеством $S = \{se_i | i = 1..M\}$, где se_i – структурные элементы документации (СЭД). Множество всех СЭД документа $S = S_b \cup S_{in} \cup S_g$ можно рассматривать как объединение трех непересекающихся подмножеств:

- основных СЭД, определяемых информационным содержимым документа (например, главы, параграфы и пункты документа);

- идентификационных СЭД, предназначенных для идентификации документа (например, название и номер версии документа);
- общих СЭД, необходимых для поиска и навигации в документе (например, глоссарий и содержание документа).

Множество S образует дерево G, состоящее из узлов (СЭД) и ребер, отображающих отношение иерархической упорядоченности между СЭД :

$$G = \langle S, E \rangle ,$$

где E – множество линейно упорядоченных ребер дерева.

Общие и основные СЭД могут содержать вложенные элементы (СЭД), порядок расположения которых задается списком ребер. Например, для корневого СЭД дерева se_d список ребер имеет вид:

$$((se_d, se_1), (se_d, se_2), ..., (se_d, se_i), ..., (se_d, se_j), (se_d, se_n)) .$$

Содержимое документа C представлено множеством $C = \{ie_j \mid j = 1..N\}$, где ie_j – информационный элемент документа (ИЭД) – логически завершенный инкапсулированный элемент, содержащий информацию о ПО, например, описание структуры классов, объектов, описание интерфейсов.

Между элементами множеств C и S определено такое соответствие R, что если задан некоторый ИЭД $ie \in C$, то и определен СЭД $se \in S$. R определено для любого элемента из множества C и является отображением $R : C \to S$.

При редокументировании ИЭД формируется путем включения в него представления, которое извлекается из наследуемого ПО, и является инкапсулированной информацией, отображающей часть ПО или ПО в целом в том или ином аспекте. Перечень требований, диаграмма вариантов использования, диаграмма компонентов, описание класса, диаграмма классов, изображение пользовательского интерфейса, схема БД являются примерами представлений ПО. Представление (v) имеет значение (z) (рисунок, текст, диаграмма, анимация или звук) и характеризуется свойствами ($p_1, ..., p_n$) (тип, нотация, физическое размещение значения представления):

$$v = \langle z, p_1, ..., p_n \rangle .$$

ИЭД, кроме представления, может включать пояснение, которое описывает представление, например, примечание, подписи таблиц и рисунков.

Форма представления документа определяется правилами внешнего оформления, зависящими от формата носителя документа, которые обычо формируются путем установки значений для стандартизированных или корпоративных элементов стиля [10,11].

3. Язык метаописания лркументов

Для построения метаописаний документов предлагается специально разработанный язык RDDL, реализованный на основе XML [12]. Язык

RDDL базируется на рассмотренной модели документа и содержит средства для описания СЭД, ИЭД, представлений и пояснений.

Метаописание на языке RDDL состоит из элементов, принадлежащих следующим типам: корневому, идентификационному, общему и основному СЭД, ИЭД, представлению и пояснению, причем идентификационный и общий СЭД имеют подтипы. Элемент, принадлежащий корневому типу, в метаописании может быть только один, и по одному элементу каждого подтипа идентификационных и общих СЭД. Количество элементов, принадлежащих типам основного СЭД, ИЭД, представления и пояснения, в метаописании описании не ограничено.

Предлагаемый формальный язык характеризуется следующим:

- множествами меток и атрибутов;
- правилами, определяющими согласование и порядок следования элементов в метаописании;
- правилами описания элементов метаописания.

Множество меток определено для типов и подтипов элементов метаописания (табл. 1). Метка представляется идентификатором, обозначающим начало и конец описания элемента в метаописании.

Табл. 1

Метка	Подтип элемента метаописания	Тип элемента метаописания
document	–	корневой
projectName	название проекта	идентификационный СЭД
title	название документа	
version	номер версии документа	
revisionHistory	лист изменений	
tableofContents	содержание	общий СЭД
glossary	глоссарий	
introduction	введение	
listOfFigures	список иллюстраций	
listOfTables	список таблиц	
index	указатель	
bibliography	список литературы	
appendix	приложение	
item	–	основной СЭД
infoItem	–	ИЭД
view	–	представление
explanation	–	пояснение

Правила вложенности устанавливают элементы метаописания, имеющие вложенные элементы определенного типа (табл.2).

Табл. 2

Тип элемента метаописания	Тип вложенных элементов

корневой	идентификационный, общий и основной СЭД
общий СЭД	основной СЭД, ИЭД
основной СЭД	основной СЭД, ИЭД
ИЭД	представление, пояснение

Правило следования СЭД определяет порядок расположения вложенных СЭД в содержимом корневого элемента (рис.2). Порядок следования идентификационных СЭД устанавливается следующим образом: элементы, принадлежащие подтипам название проекта, название документа, номер версии документа и лист изменений. Порядок следования общих СЭД задается требованиями технологии. Порядок расположения основных СЭД друг относительно друга (следование и вложенность) устанавливается экспертом при создании метаописаний на языке RDDL.

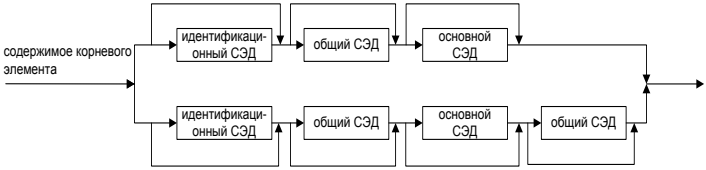


Рис.2. Правило следования СЭД в метаописании

Правил описания элементов каждого типа, базирующихся на структуре элемента метаописания (рис.3).

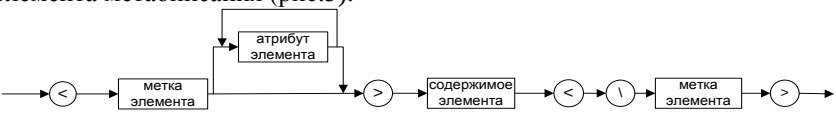


Рис.3. Структура элемента метаописания

Название метки выбирается из предопределенного в языке RDDL множества меток (табл.1), а содержимое элемента зависит от его вида, который может быть следующим:

- пустым – разрешены только атрибуты (содержимое элементов идентификационного СЭД и представления);
- вложенным – вложенные элементы и атрибуты (содержимое элементов корневого типа и типа ИЭД (рис.4));
- текстовым – текст (содержимое элемента, принадлежащего типу пояснение);
- смешанным – текст и вложенные элементы (содержимое элементов типов общий и основной СЭД (рис.5)).

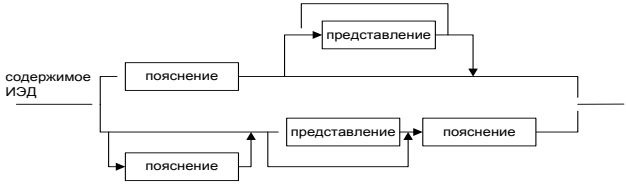


Рис.4. Описание содержимого ИЭД на языке RDDL

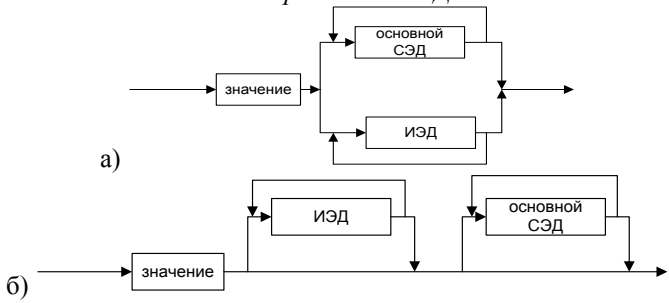


Рис.5. Описание содержимого СЭД на языке RDDL
а) для общего СЭД; б) для основного СЭД.

Описанные средства были применены при редокументирования одной из подсистем ПО, входящего в состав автоматизированной системы изготовления и учета документов государственного образца [13]. При реализации проекта были созданы метаописания документов фаз технологии RUP, которые использовались средствами автоматической генерации документов (рис.6).

```
<?xml version="1.0" encoding="utf-8"?>
<document technology="RUP" phase ="Elaboration">
        <projectName></projectName>
        <title>Software Architecture Document</title>
        <version></version>
        <revisionHistory><\revisionHistory>
        ...
        <item> 5. Logical View
                A description of the logical view of the architecture. Describes the most important classes, their organization in
                service packages and subsystems, and the organization of these subsystems into layers. Also describes the most
                important use-case realizations, for example, the dynamic aspects of the architecture. Class diagrams may be
                included to illustrate the relationships between architecturally significant classes, subsystems, packages and layers.
                <item> 5.1 Overview
                        <infoItem>
                        <view type="ListOfPackages" notation="UML"></view>
                        <explanation>Package and Subsystem Layering <\explanation>
                        </infoItem>
                </item>
        </item>
        ...
<\document>
```

Рис. 6. Пример метаописания документа «Архитектура ПО» фазы Elaboration технологии RUP

Заключение

Редокументирование при реинженерии наследуемого ПО обеспечивает создание документации, которая используется в прямой инженерии. В методе редокументирования, ориентированном на технологию разработки, состав получаемой документации определяется фазой прямой инженерии, а структура и содержание документов – требованиями технологии. В качестве средства реализации метода предлагается использовать метаописания документов технологий, создаваемые на языке RDDL, разработанном на основе модели документа с помощью расширяемого языка разметки XML. Метаописания документов позволяют автоматизировать процесс редокументирования путем применения средств автоматической генерации документов ПО.

Список использованной литературы

1. Software Reengineering Assessment Handbook (SRAH). Version 3.0. JLC-HDBK-SRAH. March 1997.– 98p.
2. Chikofsky, Elliot J.; Cross II, James H. Reverse Engineering and Design Recovery: A Taxonomy. IEEE Software, January 1990.– pp.13-17
3. Tilley, Scott R. Perspective on Legacy System Reengineering (Draft Version 3), SEI Reengineering Center Publication, Summer 1996. – 124p.
4. Wong, Kenny; Tilley Scott R.; Müller, Hausi A. ; Storey, Margaret-Anne D. Structural Redocumentation: A Case Study. IEEE Software, Vol. 11, no. 6, January 1995. – pp. 501-520
5. Rajlish V. Incremental Redocumentation Using the Web. IEEE Software, September/October 2000. – pp.2-6
6. Советский энциклопедический словарь / Гл.ред. А.М.Прохоров.- 3-е изд. –М.: Сов. Энциклопедия, 1984. – 1600с.
7. ISO/IEC 15910:1999(E). Information technology – Software user documentation process
8. ISO/IEC 6592:2000(E). Information technology – Guidlines for the documentation of computer-based application systems
9. Software Engineering Institute, “Capability Maturity Model Integration, v1.1,” CMU/SEI-2002-TR- 002, ESC-TR-2002-002, December 2001
10. ISO/IEC FDIS 18019:2003(E). Software and systems engineering — Guidelines for the design and preparation of user documentation for application software
11. Microsoft® Manual of Style for Technical Publications, Third Edition: Microsoft Corporation Editorial Style Board, 2004 – 381 p.
12. Бин Д. XML для проектировщиков. Повторное использование и интеграция. – М.:КУДИЦ-ОБРАЗ, 2004. – 256с.
13. Сидоров М.О., Хоменко В.А., Авраменко Е.А. „Реинженерия проектов программного обеспечения”. Проблемы программирования. – НАНУ. – 2,3, 2006. – с.31-38

УДК 681.3.

АРТЕМЕНКО В. А.,
БОГДАНОВА Н. В.

УПРАВЛЕНИЕ ТРАФИКОМ С УЧЕТОМ ЕГО САМОПОДОБИЯ

Составлена имитационная модель функционирования модифицирован- ной схемы и проведен анализ эффективности работы такой схемы по сравнению с классической.

The simulation model of functioning of the modified chart is made and the analysis of work efficiency of such chart in comparison to classic is produced.

В интегрированной телекоммуникационной сети можно выделить следующий важный аспект это информационные и управляющие сигналы передаются в одних и тех же физических, а иногда и логических каналах связи. Возможность их взаимного влияния может приводить к существенному изменению параметров приложений и является причиной возникновения сложных бифуркационных явлений [1], которые проявляется в неравномерном или взрывном характере протекания сетевых процессов.

Поэтому для управления такими процессами необходимо использовать адекватные математические модели, учитывающие в конструктивной форме как статистический характер возмущении, так и динамику передачи пакетов на различных уровнях протоколов межсетевого взаимодействия, включая временные и пространственные характеристики. В основе данных моделей лежит принцип самоподобия с учетом которого и рассматриваются математические модели трафика и задержек в узлах коммутации.

В статье [2] произведен расчет временных параметров телекоммуникационных сетей при самоподобном трафике. В статье [3] подробно рассмотрены методы управления трафиком по виртуальному пути, а также описан механизм управления трафиком для модифицированной схемы. Ссылаясь на вышеуказанные работы в этой статье будет составлена имитационная модель функционирования модифицированной схемы и произведен анализ эффективности работы такой схемы по сравнению с классической.

Рассмотрим алгоритм формирующий “окно” перегрузки по сигналу обратной связи в виде пакетов подтверждения, который захватывает очередные данные, и разрешает передавать их без подтверждения.

Допустим, что из-за очередей в промежуточных узлах – маршрутизаторах, в компьютерах источника и приемника, а также из-за переполнения буферов в этих узлах соединение сети не справляется с нагрузкой. Вследствие этого часть пакетов чрезмерно задерживается в пути и даже, может быть, теряются. В этом случае пакеты подтверждения не отсылаются и протоколом TCP на стороне источника на следующем цикле передачи

формируется окно уменьшенного размера. На этом заканчивается фаза медленного старта и начинается фаза управления перегрузкой, сопровождающаяся для налаживания и поддержания соединения локальными воздействиями и определением каждый раз текущего окна соединения. Интервал времени между моментами посылки пакета в направлении приемника и получения пакета подтверждения определяется ранее введенным параметром задержки (RTT). Чтобы избежать длительных простоев из-за ожидания потерянных или задержавшихся пакетов вводится пороговое значение RTT0. При превышении задержкой RTT порогового значения RTT0 пакеты считаются потерянными. Величина RTT0 определяется с помощью адаптивного алгоритма TCP/IP протокола, порядок работы которого включает следующие операции. Измеряются ряд значений RTT-задержки, полученные значения усредняются с весовыми коэффициентами, возрастающими от предыдущего измерения к последующему, а затем полученный результат умножается на некоторый коэффициент.

В связи с уменьшением окна перегрузки пропускная способность соединения снижается. Анализ работы существующего алгоритма работы протокола показывает, что имеются значительные резервы уменьшения потерь в пропускной способности в режимах медленного старта и управления перегрузкой. Во-первых, перегрузка не прогнозируется, а обнаруживается, по самому факту отсутствия пакетов подтверждения после очередного перемещения окна, во-вторых по этому факту нельзя судить о величине перегрузки и следующее состояние “окно” устанавливается методом «проб и ошибок». Задача заключается в формировании заблаговременно, не заходя в критическую область обнаружения потерянных пакетов, оценок как месторасположения временного интервала возможного проявления перегрузки, так и величины самой перегрузки. Решение такой задачи можно получить в рамках методов идентификации процессов, использования при моделировании RTT-задержек фрактального броуновского движения и формирования оценок прогноза. Оптимальная оценка прогноза RTT-задержки для момента времени tn+k принимает вид [4]:

$$\hat{T}_{n+k} = \hat{B}_H(t_{n+k}) - \hat{B}_H(t_{n+k-1}) + T_0 + \Delta T_{cp}; \quad (1)$$

где: $\hat{T}_{n+k}$ – оптимальная оценка прогноза RTT для момента времени t_{n+k};

RTT- интервал между посылкой пакета и получением подтверждения;

$\hat{B}_H(t_{n+k})$ – оптимальная оценка процесса дробного броуновского движения с параметром Херста для момента времени t_{n+k} ;

$\hat{B}_H(t_{n+k-1})$ – оптимальная оценка процесса дробного броуновского движения с параметром Херста для момента времени tn+k-1 ;

T_0 – минимальная составляющая интервала времени без очереди;

ΔT_{cp} – среднее приращение RTT задержки.

Остановимся подробней на оценке прогноза следующего за последним измеренным для момента времени tn значением фрактального броуновского движения. На основании формулы (1) она равна

$$\hat{T}_{n+1} = \hat{B}_H(t_{n+1}) - B(t_n) + T_0 + \Delta T_{cp}; \quad (2)$$

С учетом формулы rн(1; Δ) = 22Н-1-1 оценка прогноза RTT-задержки принимает вид:

$$\hat{T}_{n+1} = \left(2^{2H-1} - 1\right)\left[T_n - \left(T_0 + \Delta T_{cp}\right)\right] + T_0 + \Delta T_{cp}; \quad (3)$$

Например, наличие возможной перегрузки на n+l цикле могло быть обнаружено и тем самым с большой вероятностью предотвращено по результатам измерений RTT-задержек или фрактального броуновского движения на m предшествующих циклах, формирования согласно формуле (2) соответствующей оценки прогноза и сравнения ее с тайм-аутом. Кроме того, по величине спрогнозированной RTT-задержки на n+l цикле можно судить об уровне перегрузки и по сигналу обратной связи соответственно уменьшить (перенастроить) величину окна, то есть определить закон изменения окна перегрузки.

Ввиду того, что счетные статистики первого и второго порядков (интенсивность и корреляционная функция точечного процесса) экспериментально могут быть определены только на отдельных участках сети, предметом оптимизации и управления становятся потоки информации между отдельными узлами виртуального соединения сети. Выберем два узла, из которых i-ый узел является источником, а j-ый узел - приемником. Допустим, интенсивность потока (пропускная способность участка сети между i- и j-ми узлами) определяется очередью в узле j, возникшей, например, из-за ограниченного объема памяти буфера в этом узле, низкой интенсивности разгрузки этого буфера, из-за прибывших в этот узел пакетов с других соединений сети и т.д.

В связи с этим интенсивность потока информации от узла i к узлу j понижается, а в случае переполнения буфера в узле j передача информации прекращается, что сопровождается потерей части пакетов. Для предотвращения полной потери пропускной способности необходимо регулировать уровень загрузки буфера в узле j . Для этого воспользуемся про-

гнозируемыми оценками счетных характеристик. Измеряют на рассматриваемом участке сети число отсчетов Xn на интервале (tn,tn-T). Одновременно осуществляется прогноз на некотором интервале упреждения kT, где k – параметр смещения. Ограничимся для примера прогнозом на один шаг вперед. Прогнозируемая оценка Xn+1 определяется или по последнему результату измерений в соответствие с формулой оценки прогноза отчета:

$$\hat{X}_{n+k} = r(k;T)(X_n - \lambda T) + \lambda T, \quad (4)$$

где: λ – интенсивность поступления пакетов; k – параметр смещения; Xn –число отсчетов; $\hat{X}_{n+k}$ – оптимальное значение оценки прогноза отсчета; r(k;T) – коэффициент корреляции.

Если величина прогноза оказывается больше порога, зависящего от уровня загрузки буфера и определяемого некоторым адаптивным алгоритмом, то по сигналу обратной связи интенсивность генерации информации с узла i уменьшается на величину, зависящую от уровня загрузки буфера и значений оценок прогноза. Пропускная что способность из-за этого на участке сети и понижается, но в связи с сохранением процесса передачи информации и уменьшением числа потерянных пакетов удается в среднем уменьшить потери в пропускной способности этого соединения. Если необходимо сохранить пропускную способность, то по сигналу обратной связи доступные сетевые ресурсы перераспределяются в пользу этого участка виртуального соединения сети.

На основании вышеизложенного, была составлена имитационная модель функционирования модифицированной схемы. Результаты моделирования показали следующую эффективность работы такой схемы по сравнению с классической рисунок 1.

Результаты моделирования показывают, что возможны случаи, когда наиболее вероятное время задержки при работе модернизированной схемы будет меньше чем фиксированное минимальное время передачи для классической схемы.

Следует отметить, что при высоком уровне загрузки сети время задержки будет приближаться к задержкам в классической схеме. Это объясняется тем, что при высоких загрузках предложенная система управления трафиком будет вынуждена уменьшать скорости передачи некоторых источников, для которых эти скорости были увеличены, что повышает непроизводительные расходы. Однако, как показывает моделирование, при правильном выборе параметров модернизированная схема работает более эффективно, чем классическая рисунок 2.

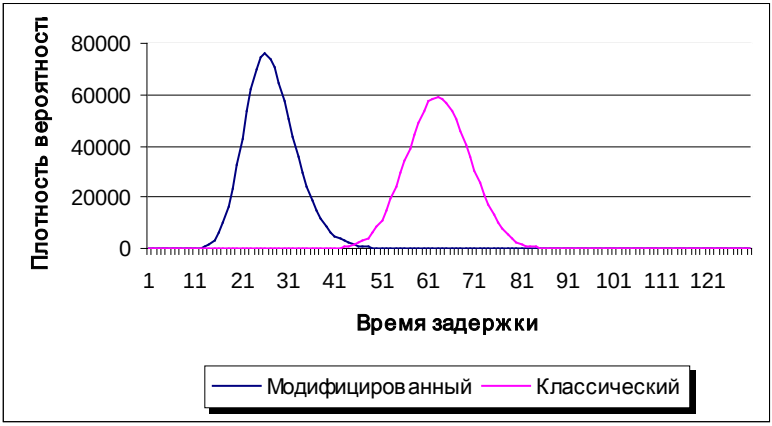


Рис. 1 – Эффективность работы модифицированной схемы по сравнению с классической

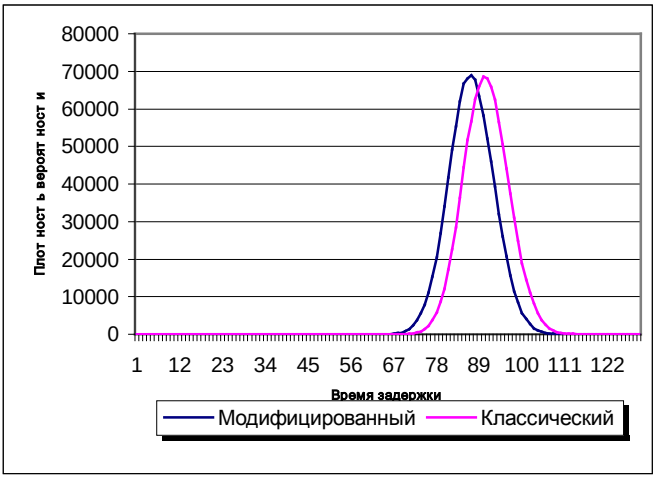


Рис. 2 - Эффективность работы модифицированной схемы по сравнению с классической при выборе параметров

На основании вышеизложенного, была составлена имитационная модель функционирования модифицированной схемы. Результаты моделирования показали следующую эффективность работы такой схемы по сравнению с классической.

Параметр средней загрузки при различной вероятности поступления запросов на соединение определяет, насколько оптимально используется пропускная способность сети при повышенных нагрузках. Вероятности поступления запросов на соединение означает вероятность возникновения

запроса на соединение в каждый момент времени (для модели – в каждый такт модельного времени).

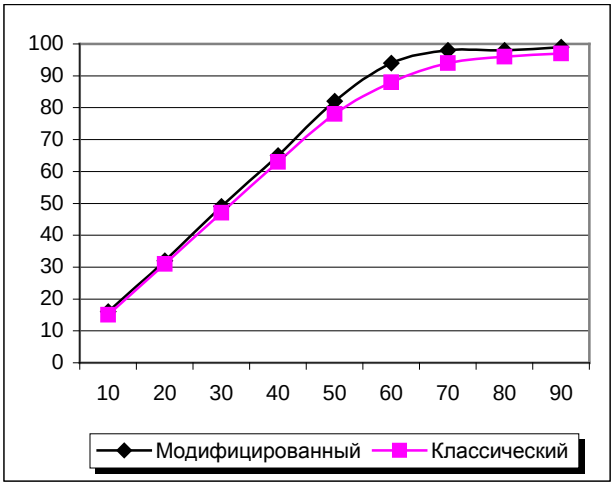


Рис. 3 – Сравнительный график средней загрузки при различной вероятности поступления запросов на соединение.

Как показывают результаты моделирования (рисунок 3), классическая схема менее эффективно использует полосу пропускания за счет того, что при большой загруженности сети остается нераспределенная пропускная способность, величина которой недостаточна для установки новых соединений. В модернизированной схеме эта пропускная способность передается тем соединениям, которые могут получить некоторую выгоду от передачи своего трафика с более высокой скоростью.

Параметр отказов в обслуживании при различной вероятности поступления запросов на соединение, определяет вероятность того, что поступившему запросу на соединение будет отказано в обслуживании.

Вероятность удачного выбора альтернативного маршрута зависит от многих параметров, как, например, связности сети, загрузки коммутационного оборудования, эффективности алгоритмов маршрутизации, а также от заданных параметров для искомого маршрута. На рисунке 5 представлена зависимость отказов в обслуживании от вероятности удачного выбора альтернативного маршрута.

Результаты моделирования показывают, что даже при минимальной вероятности удачной ремаршрутизации модернизированная схема управления трафиком дает положительный результат в виде минимизации отказов в обслуживании.

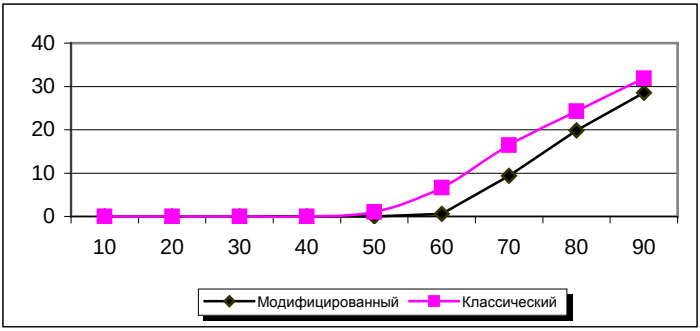


Рис. 4 – Сравнительный график средней загрузки при различной вероятности поступления запросов на соединение.

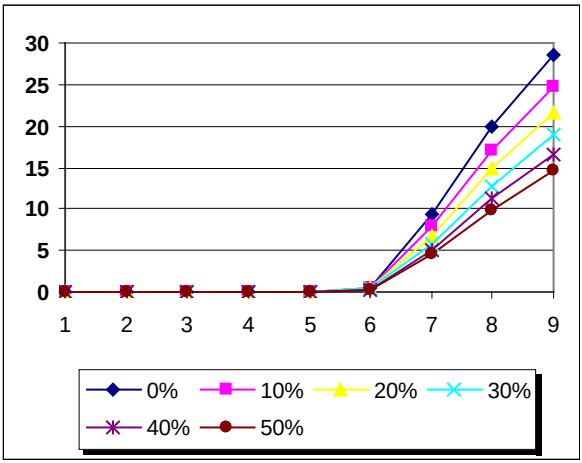


Рис. 5 - Зависимость отказов в обслуживании от вероятности удачного выбора альтернативного маршрута

Характер графиков объясняется тем, что при средней загруженности сети (рисунок 6) модернизированная схема может работать очень эффективно за счет увеличения пропускной способности для тех соединений, которые могут получить от этого выгоду. При этом, для низкой и средней загруженности сети существует низкая вероятность того, что впоследствии возникнет необходимость снижать скорость этих соединений или производить ремаршрутизацию, а следовательно не потребуются дополнительные непроизводительные затраты.

Результаты моделирования показывают, что возможны случаи, когда наиболее вероятное время задержки при работе модернизированной схемы будет меньше чем фиксированное минимальное время передачи для классической схемы.

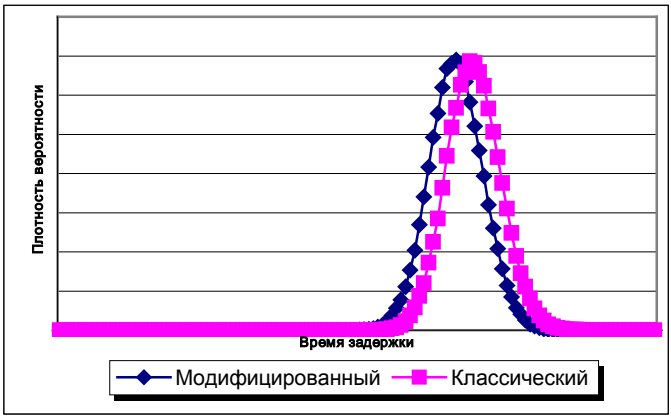


Рис. 6 - Плотность вероятности времени задержки для высокого уровня загрузки сети

Следует отметить, что при высоком уровне загрузки сети время задержки будет приближаться к задержкам в классической схеме. Это объясняется тем, что при высоких загрузках предложенная система управления трафиком будет вынуждена уменьшать скорости передачи некоторых источников, для которых эти скорости были увеличены, что повышает непроизводительные расходы. Однако, как показывает моделирование, при правильном выборе параметра Δ модернизированная схема работает более эффективно, чем классическая схема.

Список используемой литературы

1. Городецкий А.Я., Иванов А.В. Фрактальные процессы в компьютерных сетях: методы идентификации моделей трафика // Научно-технические ведомости СПбГТУ, 2001, № 3 (25).
2. Богданова Н.В., Мухаммед Ель Амин Бабикер. Анализ временных параметров телекоммуникационных сетей при самоподобном трафике.//Системні технології. Регіональний міжвузівський збірник наукових праць №1(48).-Дніпропетровськ, 2007 ст. 101-111.
3. Богданова Н.В., Способы повышения эффективности системы управления телекоммуникационными сетями.// Адаптивні системи автоматичного управління. Регіональний міжвузівський збірник наукових праць. Випуск 9(29).-Дніпропетровськ, 2006 ст.23-32.
4. Шелухин О.И., Тенякшев А.М., Осин А.В. Фрактальные процессы в телекоммуникациях. Монография /Под ред. О.И.Шелухина - М.: Радиотехника, 2003.- 480 с.

УДК 681.518

РОЛИК А.И.,
СОКОЛОВСКИЙ Р.Л.

РАСПРЕДЕЛЕНИЕ МОБИЛЬНЫХ КОМПОНЕНТОВ СИСТЕМЫ УПРАВЛЕНИЯ ИНФОРМАЦИОННО-ТЕЛЕКОММУНИКАЦИОННОЙ СИСТЕМОЙ

Показана целесообразность применения агентских технологий в системах управления функционированием крупномасштабных информационно-телекоммуникационных систем. Предложена математическая модель оптимального распределения агентов системы управления.

Expedience of the use of agent technologies in functioning management systems of large-scale information-telecommunication systems is shown. The management system agents optimal assignment mathematical model is offered.

Введение

В настоящее время практически все предприятия и организации для управления своими процессами деятельности используют различные информационные технологии. Организации с распределенной структурой широко используют информационно-телекоммуникационные системы (ИТС), в которых для объединения рассредоточенных подразделений часто используется сеть Интернет.

Эффективность работы ИТС существенно зависит от эффективности и бесперебойности функционирования информационных и телекоммуникационных ресурсов, предоставляемых ИТС пользователям. Особенностью современных ИТС является интеграция разнообразных, порой сильно различающихся компонентов и технологий, которая создает множество проблем в обслуживании ИТС, проявляющихся в сложности организации мониторинга и управления, контроля качества услуг, предоставляемых пользователям и т. д. Поэтому особую значимость приобретает разработка новых или совершенствование существующих систем управления ИТС.

Проблемы управления ИТС

Перспективы роста и развития крупных организаций непосредственно зависят от эффективности использования информационных и сетевых технологий, позволяющих накапливать и обрабатывать большие объемы аналитической информации, обеспечить эффективность процессов управления и использования информационных ресурсов, повысить надежность и оперативность информационного обмена. Информационно-телекоммуникационная система, которая призвана интегрировать и поддерживать

деятельность объектов информационной и телекоммуникационной структуры организаций, представляет собой гетерогенную среду, стремительно развивающуюся и требующую эффективного управления [1].

Проблемы повышения эффективности работы как информационных, так и сетевых технологий приобретают особую актуальность в связи с ростом стоимости программного обеспечения, серверных платформ и других инфраструктурных компонентов ИТС, когда производительность, надежность, время обработки запросов пользователей и стабильность работы при пиковых нагрузках не удовлетворяет требованиям со стороны бизнес-процессов, и не оправдывает больших инвестиций в информационные технологии. Для управления ИТС разрабатываются и внедряются системы управления функционированием (СУФ), которые не только осуществляют мониторинг и управление информационными и телекоммуникационными технологиями, но и управляют распределением ограниченных ресурсов ИТС, а также решают и другие задачи [1–4].

Сложность разработки СУФ ИТС больших корпораций, государственных и правительственных структур обусловлена множеством применяемых информационных технологий, сложностью и распределенностью ИТС, большим количеством используемого сетевого оборудования различных производителей, разнообразием ресурсов и услуг, предоставляемых пользователям, необходимостью совместной работы различных систем управления, непосредственно управляющих только отдельными технологиями или оборудованием без учета проблем, возникающих в других системах управления, динамики значимости процессов деятельности и пр. Эти и множество других проблем делают невозможным решение задачи создания единой универсальной централизованной СУФ ИТС, способной эффективно управлять различными крупномасштабными ИТС. Поэтому СУФ большими ИТС строятся по иерархическому принципу с делегированием функций удаленным компонентам СУФ, осуществляющим мониторинг и управление фрагментом ИТС в отдельной географической зоне или находящимся в ведении одного администратора. В условиях высокой степени интеграции и взаимосвязей компонентов ИТС особое значение приобретает вопрос определения количества и размещения удаленных компонентов СУФ. Несмотря на существование большого количества разнообразных решений в области мониторинга и управления ИТС, отсутствует методический аппарат, позволяющий решать задачи оптимизации размещения компонентов СУФ на аппаратных средствах ИТС.

Применение агентских технологий в системах управления ИТС

Для решения задач мониторинга и управления большими ИТС целесообразно использовать агентские технологии [2, 5], которые применяются для управления программным обеспечением в Grid-системах [6], автоматизации деловых процессов поставщиков услуг и операторов связи [7], а

также для решения других задач. Основой агентских технологий является использование программных агентов. Программные агенты – это часть программного обеспечения, действующая от имени другой программы, т. е. может запустить программу от ее имени и с ее полномочиями, и способна участвовать в агентских взаимоотношениях.

Суть агентских технологий применительно к СУФ ИТС заключается в следующем.

Автономные программные агенты самостоятельно решают задачи СУФ для отдельной географической области или функциональной группы, а также координируют собственные действия для решения задач, поставленных перед СУФ системными администраторами.

В отличие от активных объектов и других типов программных компонентов, автономные агенты, обладая интеллектуальными свойствами и способностью перемещаться, могут самостоятельно инсталлироваться и автономно действовать для достижения целей функционирования и решения поставленных перед ними задач, при этом координируя свои действия, как с другими агентами, так и с центральным узлом СУФ. Одной из таких задач является контроль функционирования аппаратно-программного обеспечения телекоммуникационных узлов и серверов приложений с возможностью, например, самостоятельного принятия решения и осуществления перезагрузки или реализации других мероприятий для восстановления эффективной работы серверов и других устройств. Автоматизация таких процессов позволяет существенно сократить расходы на эксплуатацию ИТС.

Автономные программные агенты должны обладать свойствами интеллектуальности [8], мобильности [9] и способности взаимодействовать [10].

Различная степень интеллектуальности позволяет агентам генерировать цели функционирования в зависимости от поставленных перед ними администраторами СУФ задач. При этом агенты могут иметь упреждающее поведение, обладать способностью обучения, самостоятельного принятия решений на основании собираемой от элементов ИТС информации о функционировании, распределять и перераспределять между собой задания.

Программные агенты могут быть стационарными или мобильными. Стационарные агенты не способны к перемещению в ИТС. Интеллектуальные мобильные агенты могут самостоятельно перемещаться в ИТС с учетом загруженности сети, неисправности или отключения отдельных узлов или фрагментов сети и пр., управляя своей миграцией. Мобильные агенты могут прекратить работу на одном узле и продолжить на другом. Естественно, что при этом на узлах должна иметься программная платформа, способная принять агента. Мобильные агенты представляют собой эффективное решение для автоматизации выполнения действий, связан-

ных со сбором и обработкой информации, необходимой для эффективного управления ИТС.

Способность взаимодействовать позволяет агенту осуществлять координированные действия совместно с другими агентами для достижения общих целей.

В отличие от технологии клиент-сервер, когда клиент через сеть обменивается с сервером запросами и ответами, в технологии мобильных агентов возможна миграция части распределенного приложения на сервер, в результате чего удаленное взаимодействие заменяется локальным.

Для того чтобы агенты могли перемещаться по сети и выполнять какие-либо действия в узлах, на которых они будут располагаться, должна существовать платформа, способная принять агента и создать ему условия, необходимые для решения задач, обеспечить его коммуникацию с другими агентами, а также идентификацию, управление агентом, целостность информации и пр. В качестве такой платформы может быть использована платформа разработки программного обеспечения SmartBase [1].

На основе стационарных и мобильных агентов строятся многоагентские СУФ, являющиеся агентскими системами и обладающие высокой гибкостью и степенью автоматизации при решении различных заданий СУФ в отдельном элементе, подсети, домене или зоне.

Преимуществами агентской технологии является:

- уменьшение загруженности ИТС за счет сокращения служебного трафика СУФ;
- возможность автономного выполнения задач управления функционированием;
- возможность работы без постоянной связи с центральным узлом СУФ;
- возможность конкурентного выполнения заданий в многоагентской среде;
- миграция агентов вместе с решаемыми ими задачами позволяет выровнять использование вычислительных ресурсов и снизить вероятность перегрузки отдельных узлов;
- использование мобильных агентов освобождает администраторов от операций инсталляции, запуска и завершения использования программных компонентов СУФ.

Агентские технологии с мобильными агентами отлично подходят для решения множества задач в СУФ специализированных ИТС, объединяющих множество различных информационных и сетевых технологий и предоставляющих конвергентные информационно-телекоммуникационные услуги. Кроме того, технология мобильных агентов прекрасно подходит для решения задач управления функционированием в специализированных ИТС с мобильными пользователями и группами пользователей, а также позволяет осуществлять адаптацию к изменяющимся характери-

стикам технических средств и изменяющимся потребностям пользователей при их перемещении.

При создании СУФ на основе агентской технологии необходимо решить множество вопросов, в частности, определить количество агентов, их расположение, выполняемые ими функции, правила взаимодействия с агентами, высшими по уровню иерархии – мета-агентами. Мета-агенты получают информацию от множества стационарных и мобильных агентов, обобщают эту информацию и обеспечивают выполнение функций СУФ в отдельной географической зоне или функциональной группе элементов ИТС, а также могут решать задачи управления функциональными или обеспечивающими подсистемами в АСУ больших организаций и предприятий. При этом мета-агенты, кроме взаимодействия с агентами и, при необходимости, с другими мета-агентами, поддерживают взаимодействие с главным мета-агентом, находящимся на центральном узле СУФ ИТС.

Целью статьи является разработка математической модели определения мест оптимального расположения мета-агентов системы управления информационно-телекоммуникационной системой.

Математическая модель определения мест оптимального расположения мета-агентов

Использование агентских технологий в СУФ является эффективным средством решения множества проблем управления ИТС больших организаций, корпораций, государственных структур, возникающих в результате использования большого количества разнообразных информационных и телекоммуникационных технологий, объединения множества локальных сетей и пр. Агентская система может повторять структуру ИТС, а агентов можно выстраивать в необходимые уровни иерархии, выделяя мета-агенты, главные мета-агенты и пр., устанавливать любые связи и зависимости между ними, добавлять, перемещать, совмещать и т. д. При этом на каждом аппаратном сервере или узле ИТС, в котором может быть размещен агент, должна быть установлена соответствующая платформа – специальное программное обеспечение, предоставляющее функциональность для создания, запуска и удаления агента, а также среду для выполнения агентов (набор необходимых библиотек и пр.).

Мета-агент самостоятельно выполняет задание, если ему выделены достаточные для работы ресурсы, он имеет соответствующие права и получает необходимую информацию от агентов. В противном случае он ищет другого мета-агента, способного выполнить задание или предоставить ему нужную информацию. Например, если мета-агент, отвечающий за работоспособность функциональной или обеспечивающей подсистемы АСУ, находящийся в одной подсети, не может непосредственно получить информацию о работоспособности элемента ИТС, находящегося в другой

подсети, он находит агента, который такую информацию может предоставить. После чего организовывается взаимодействие, при котором агенты, кроме решения собственных задач, действуют в интересах других агентов, т. е. получают информацию, обрабатывают ее или передают без обработки для обобщения мета-агентами.

Учитывая тот факт, что функциональность мета-агентов не зависит от их месторасположения и они могут перемещаться в сети, возникает задача оптимизации мест расположения мета-агентов.

Для решения этой задачи разработана следующая математическая модель.

Пусть $E = \{e_l\}$, $l = 1,...,L$, – множество элементов ИТС, в качестве которых выступает только активное оборудование (серверы, рабочие станции, коммутаторы и пр.), L – общее количество элементов.

$S = \{s_r\}$, $r = 1,...,R$, – множество групп элементов, R – количество групп. Состав каждой группы формируется администратором системы управления ИТС. В группы могут объединяться элементы, обеспечивающие отдельные сервисы ИТС, входящие в состав подсетей, выполняющие определенные функции ИТС и пр. Причем $s_r \subset E$, но не обязательно $e_l \in s_r$, для всех $l = 1,...,L$ и $r = 1,...,R$, т. е. могут быть элементы, не входящие ни в одну из групп множества S.

Состояние элементов каждой группы контролируется отдельным мета-агентом, который может самостоятельно собирать информацию, характеризующую работоспособность элементов группы, или получать обобщенную информацию от стационарных или мобильных агентов, располагающихся на элементах групп.

Бинарная матрица $G = \|g_{rl}\|$, $r = 1,...,R$, $l = 1,...,L$ – фиксирует принадлежность *l*-го элемента *r*-й группе:

$$g_{rl} = \begin{cases} 1, \text{если } e_l \in s_r, \\ 0 \text{ — в противном случае.} \end{cases}$$

Зададим $Z = \{z_k\}$, $k = 1,...,K$ – множество линий связи между элементами, *k* – номер линии связи, *K* – количество линий связи между элементами ИТС.

Определим квадратную бинарную матрицу $W = \|w_{ij}\|$, $i, j = 1,...,L$, *ij*-й элемент которой фиксирует наличие связи между *i*-м и *j*-м елементами множества *E*

$$w_{ij} = \begin{cases} 1, \text{если между элементами } e_i \text{и } e_j \\ \quad \text{имеется непосредственная связь,} \\ 0 \text{—в противном случае.} \end{cases}$$

Бинарная матрица $C^a = \left\| c_{rl}^a \right\|$, $r = 1,...,R$, $l = 1,...,L$ – фиксирует расположение r-го мета-агента на l-м элементе для отдельной a-й комбинации, $a = 1,...,A$, из всех возможных комбинаций A размещения мета-агентов на элементах ИТС

$$c_{rl}^a = \begin{cases} 1, \text{если } r\text{-й мета-агент} \\ \quad \text{располагается на элементе } e_l, \\ 0 \text{—в противном случае.} \end{cases}$$

Подмножество $H \subset E$ из N элементов определяет те элементы ИТС, на которых установлена соответствующая платформа и могут быть размещены мета-агенты групп.

С учетом того, что на отдельном элементе ИТС может быть установлено несколько мета-агентов, количество возможных комбинаций A размещений R мета-агентов на одной из N допустимых позиций, вычисляется следующим образом

$$A = N^R. \tag{1}$$

Зададим трехмерную матрицу $U_{R \times L \times K}^a = \left\| u_{rlk}^a \right\|$, $r = 1,...,R$, $l = 1,...,L$, $k = 1,...,K$, элемент u_{rlk}^a которой определяет, задействована ли линия связи z_k для передачи информации r-му мета-агенту о функционировании элемента e_l для a-й комбинации размещения мета-агентов. Причем

$$u_{rlk}^a = \begin{cases} 1 & \text{если информация от элемента } e_l \\ & \text{передается соответствующему мета-} \\ & \text{агенту } r\text{-й группы по каналу } z_k; \\ 0 & \text{– в противном случае.} \end{cases}$$

Зададим матрицу $V^a = \left\| v_{rk}^a \right\|$, $r = 1,...,R$, $k = 1,...,K$, элемент v_{rk}^a которой определяет, задействована ли k-я линия связи при передаче информации о функционировании r-й группы элементов r-м мета-агентом главному мета-агенту или в центральный узел СУФ ИТС, в который стекается вся

информация о функционировании групп элементов, и, после обобщения этой информации, делается вывод о функционировании ИТС. Значение элементов матрицы V^a определяется следующим образом

$$v_{rk}^a = \begin{cases} 1, & \text{если информация от } r\text{-го мета-агента группы к главному мета-агенту проходит через канал } z_k; \\ 0, & \text{– в противном случае.} \end{cases}$$

Будем считать оптимальным такое расположение мета-агентов, при котором суммарные затраты D^a на произведение вычислений мета-агентами при a-м расположении мета агента $D^a_{\text{ðàñ÷}}$, и передачу всей информации в агентской системе при a-м расположении мета-агента $D^a_{\text{ï ðð}}$ будут наименьшими для всех возможных комбинаций A размещения мета-агентов, т.е. будет выполнено условие

$$\min D^a = \min_{a \in A} (D^a_{\text{ðàñ÷}} + D^a_{\text{ïåð}}), \quad . \qquad a = \qquad A \qquad (2)$$

Введем матрицу $P = \left\| p_l \right\|, l = 1,...,L$, элемент p_l которой равен объему информации о функционировании l-го элемента множества E, передаваемому мета-агенту каждой из групп, в которую входит элемент e_l, и матрицу $Q = \left\| q_r \right\|, r = 1,...,R$, элемент q_r которой соответствует объему данных передаваемых от r-го мета-агента главному мета-агенту (в центральный узел СУФ ИТС).

Зададим вектор $\overline{Y} = \{y_k\}$, $k = 1,...,K$, элемент y_k которого определяет стоимость передачи единицы информации по каналу z_k.

Тогда суммарные затраты на передачу всей информации в агентской системе при a-м расположении мета агента $D^a_{\text{ï ðð}}$ можно определить следующим образом:

$$D^a_{\text{ïåð}} = \sum_{k=1}^{K} \left(\sum_{r=1}^{R} \sum_{l=1}^{L} p_l \cdot u^a_{rlk} + \sum_{r=1}^{R} q_r \cdot v^a_{rk} \right) \cdot y_k \ . \quad (3)$$

Для расчета затрат на выполнение вычислений r-м мета-агентом группы необходимо знать стоимость вычислительных ресурсов элемента множества E, на котором расположен r-й мета-агент и объем выполняемых агентом вычислений.

Зададим вектор $\overline{X} = \{x_r\}$, $r = 1,...,R$, где x_r – доход от сервисов, предоставляемых элементами r-ой группы, и вектор $\overline{M} = \{m_l\}$, $l = 1,...,L$, где m_l – дополнительный доход от l-го элемента множества E.

Матрицей $T = \|t_r\|, r = 1,...,R$ зададим потребности в вычислительной емкости для каждого r-го мета-агента.

Тогда

$$D^{a}_{расч} = \sum_{r=1}^{R}\sum_{l=1}^{L}(x_r \cdot g_{rl} + m_l)\cdot t_r \,. \qquad (4)$$

Для определения оптимальных мест расположения мета-агентов в ИТС необходимо произвести перебор по всем возможным комбинациям из A с учетом критерия (2) и составляющих (3) и (4).

Для примера рассмотрим структуру ИТС предприятия, приведенную на рис. 1.

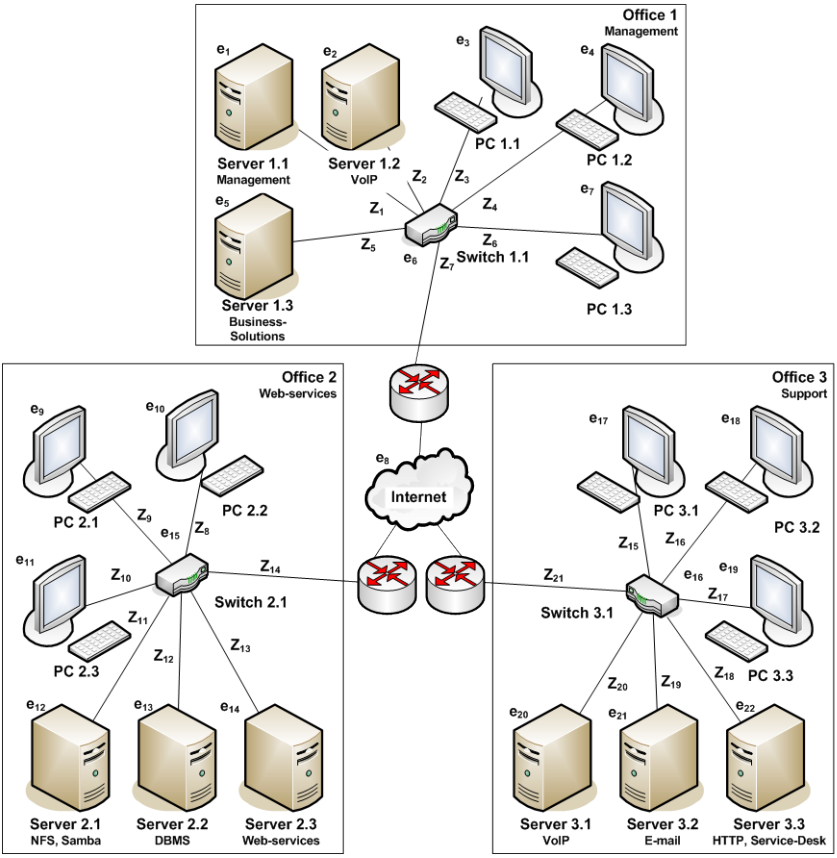


Рис. 1. Пример структуры ИТС предприятия

Предположим, что предприятие имеет три офиса, каждый из которых выполняет определенные функции. Подсети офисов объединены посредством Интернет. Причем первый офис выполняет функции управления, второй – хранилища и web-сервисов, а третий – поддержки клиентов. В

каждом офисе имеется по три сервера, три рабочих станции и одному коммутатору. Подключение к Интернет производится через соответствующие маршрутизаторы. Для рассматриваемого примера $L = 22$.

Сервисы ИТС предприятия группируются по выполняемым функциям. В табл. 1 приведен состав групп сервисов и серверов, которые их предоставляют.

Табл. 1. Группы сервисов ИТС и серверов их обеспечивающих

№ группы	Сервисы (серверы)
1	Management (Server 1.1), DBMS (Server 2.2)
2	Business solutions (Server 1.3), DBMS (Server 2.2), e-mail (Server 3.2)
3	Web-services (Server 2.3), DBMS (Server 2.2)
4	VoIP (Server 3.1), DBMS (Server 2.2)

Каждую группу сервисов, а также каждую группу элементов отдельной подсети будет контролировать собственный мета-агент, собирающий информацию от элементов ИТС, посредством агентов, расположенных в соответствующих узлах, либо непосредственно опрашивая элементы. Для четырех групп сервисов и трех подсетей необходимо семь мета-агентов ($R = 7$).

Распределение элементов ИТС по группам приведено в табл. 2.

Табл. 2. Состав групп, контролируемых мета-агентами

r	Состав групп множества S
1	$\{e_1, e_{13}\}$
2	$\{e_5, e_{13}, e_{21}\}$
3	$\{e_{14}, e_{13}\}$
4	$\{e_{20}, e_{13}\}$
5	$\{e_1, e_2, e_3, e_4, e_5, e_6, e_7\}$
6	$\{e_9, e_{10}, e_{11}, e_{12}, e_{13}, e_{14}, e_{15}\}$
7	$\{e_{16}, e_{17}, e_{18}, e_{19}, e_{20}, e_{21}, e_{22}\}$

Пусть $H = \{e_1, e_2, e_5, e_{12}, e_{13}, e_{14}, e_{20}, e_{21}, e_{22}\}$. Тогда $N = 9$.

Предположим, что сервер «Server 1.1» является центральным узлом СУФ и на нем располагается главный мета-агент.

По выражению (1) определим значение A. Количество возможных комбинаций расположения мета-агентов $A = 4782969$.

Для упрощения расчетов, предположим, что значения элементов матрицы P и Q будут равны единице. Данное предположение вполне соответствует действительности, поскольку объемы информации о функцио-

нировании элементов ИТС, а также данные передаваемые от мета-агентов главному мета-агенту могут быть приблизительно одинаковыми.

Пусть $y_7 = 5,\ y_{14} = 5,\ y_{21} = 5$, а все остальные значения вектора $\overline{Y}$ – нулевые, в связи с тем, что остальные линий связи являются фрагментами локальных сетей.

Примем $\overline{X} = \{4, 8, 25, 10, 0, 0, 0\}$, а все значения вектора $\overline{M}$ – нулевыми.

Предположим, что потребности в вычислительной емкости для каждого мета-агента пропорциональны количеству элементов в группе, которую контролирует данный мета-агент. Тогда $T = \|2, 3, 2, 2, 7, 7, 7\|$.

Результаты, полученные после алгоритмизации и программирования представленной математической модели, приведены в табл. 3.

Табл. 3. Результаты моделирования

D^a	a	Элементы, на которых располагаются мета-агенты						
		$r = 1$	$r = 2$	$r = 3$	$r = 4$	$r = 5$	$r = 6$	$r = 7$
80	614662	e_2	e_2	e_{12}	e_{22}	e_2	e_{12}	e_{22}
80	611017	e_{22}	e_2	e_{12}	e_2	e_2	e_{12}	e_{22}
80	612475	e_2	e_2	e_{12}	e_{12}	e_2	e_{12}	e_{22}

Как видно из табл. 3 минимальное значение D^a, равное 80, получено сразу для трех возможных комбинаций расположения мета-агентов.

Вывод

Разработанная модель позволяет оптимальным образом расположить мета-агенты, являющиеся компонентами системы управления функционированием информационно-телекоммуникационной системой. Интерес для дальнейших исследований представляет совершенствование модели, с целью уменьшения количества вычислений, необходимых для поиска оптимального решения, а также ее доработка, позволяющая определить оптимальное количество мета-агентов.

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. — К.: «ВЕК+», — 2006. — № 45. — С. 112—126.
2. Теленик С. Ф., Ролік О.І., Терещенко П.І., Літвінцов О.В. Модель управління розподілом ресурсів інформаційно-телекомунікаційної системи збройних сил України// Збірник науков. праць ОТ і ВБ України. — 2006. — №5 (34). — С. 117—124.
3. Теленик С.Ф., Ролік О.І., Букасов М.М., Терещенко П.І. Управління доступом до обмежених ресурсів інформаційно-телекомунікаційної мережі АСУ спеці-

ального призначення // Зб. наук. праць ЦНДІ ЗС України. — 2006. — № 3 (37). — С. 33—43.
4. 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. — К.: Екотех, — 2006. — № 44. — С. 234—239.
5. Бурков В.Н., Новиков Д.А. Теория активных систем: Состояние и перспективы. М.: Синтег, 1999. — 128 с.
6. Тржец К., Ежич Г., Кушек М., Дешич С. Дистанционное управление программным обеспечением с помощью агентов/ Ericsson Nikola Tesla, REVIJA 18 (2005) 1. — pp. 57—67.
7. Fricke, S., Bsufka, K., Keiser, J., Schmidt, T., Sesseler, R., Albayrak, S., “Agent-based Telematic Services and Telecom Applications”, Communications of the ACM, vol. 44 (2001), no. 4., pp. 43—48.
8. Knapik, M., Johnson, J., Developing Intelligent Agents for Distributed Systems: Exploring Architecture, Technologies, and Applications, McGraw-Hill, New York, 1998.
9. Cockayne, W. R., Zyda, M., Mobile Agents. Prentice Hall, Upper Saddle River, New Jersey, 1997.
10. D’Inverno, M., Luck, M., Understanding Agent Systems. Springer-Verlag, Berlin, 2001.

УДК 681.03

ВИНОГРАДОВ Ю.Н.

СПОСОБ ФОРМАЛЬНОГО ОПИСАНИЯ ФУНКЦИОНИРОВАНИЯ РЕКОНФИГУРИРУЕМЫХ ВЫЧИСЛИТЕЛЬНЫХ УСТРОЙСТВ

В статье предложена формализованная модель построения и функционирования реконфигурируемых вычислительных устройств, которая в полной мере соответствует их структурной и алгоритмической организации. Исследована взаимная согласованность и корректность всех введенных в модели понятий. Предложенная модель может быть основой для анализа и синтеза реконфигурируемых вычислительных устройств.

The formalized model of construction and functioning of configurable computing devices is offered in the article, which to a full degree corresponds their structural and algorithmic organization. Mutual coordination and correctness of all of the concepts entered in models is investigational. The offered model can be basis for an analysis and synthesis of reconfigurability computing devices.

1. Введение

Реконфигурируемость структуры вычислительного устройства является естественной потребностью при создании сложных систем, которые должны обладать высокой надежностью, гибкостью и адаптацией к решаемым задачам.

В настоящее время превалирует тенденция выполнения научных исследований и практических разработок в области реконфигурируемых вычислительных устройств (РВУ) с перспективной элементной базой (в частности, ПЛИС), которые призваны удовлетворить требования, возникающие при создании высокопроизводительных устройств обработки данных, цифровой обработки сигналов, поддержки телекоммуникаций, сопряжением комплексов и систем и т.д..

Исследования комплекса прикладных задач анализа, синтеза и оптимизации системного и логического проектирования реконфигурируемых устройств [1 - 3] показали, что сдерживающим фактором построения РВУ является отсутствие системы формальных математических моделей, которые учитывали бы характеристики особенности их функционирования, определяемые особенностями новой элементной базой (в частности, ПЛИС).

Таким образом, задача построения формализованной системы согласованных математических понятий, адекватно описывающих характерные особенности функционирования РВУ, является актуальной и важной для современного этапа развития теории проектирования вычислительных устройств.

2. Анализ существующих моделей функционирования РВУ

Принцип реконфигурируемости, положенный в основу проектирования РВУ, прошел определенную эволюцию и переосмысление, что, в первую очередь, связано с прогресом элементной базы. Слово реконфигурируемый трактуется двояко: есть функциональные элементы структуры, которые объединяются тем или иным способом, при помощи каналов связи, для решения конкретной задачи, либо в системе для каждой конкретной задачи используется новая структура с новыми функциональными элементами. Такая форма реконфигурации выражается в том, что для реализации различных процессов возникают структуры в произвольном порядке.

В [1] рассмотрена классификация и проанализированы основные особенности РВУ, которые в наибольшей мере определяют способ их функционирования:

- отсутствие заранее известной структуры решения задачи;
- отсутствие единого событийного времени;
- внутренний и внешний недетерминизм при организации вычислительного процесса;
- наличие конфликтов и совместно используемых ресурсов;
- независимость структурной организации от времени;
- наличие двух видов рекофигурирования.

Ни одна из известных моделей функционирования РВС [2] [3] и изложенная в [4] не учитывает все эти особенности РВС в полной мере. Упомянутые модели создавались с целью спецификации структуры и исследования вычислительных процессов при решении задач в ней. Основные недостатки указанных моделей – это отсутствие связи с количественным и алгоритмическим анализом вычислительных устройств. Модели частично описывают влияние функциональных элементов структуры на динамику взаимодействия структурных элементов системы. Они не содержит множественного времени как количественной сущности, что затрудняет описание процессов временной реконфигурации. В известных моделях отсутствует иерархичность и структуризация функциональной организации вычислительной системы с точки зрения ее реконфигурируемости.

Таким образом, проанализированные выше модели лишь частично формализуют процесс описания РВУ и требуют дополнительных инструментов для логического и системного проектирования вычислительных устройств.

Целью данной статьи является построение математической модели динамики функционирования реконфигурируемых вычислительных устройств, которая достигается путем разработки согласованной системы математических понятий, адекватно описывающих их характерные особенности.

3. Разработка модели функционирования реконфигурируемых вычислительных устройств.

Основу модели функционирования реконфигурируемых вычислительных устройств, которые учитывают влияние функциональных элементов структуры на динамику взаимодействия структурных элементов, составляют три понятия:

1. поведение – модель динамики организации реконфигурируемых вычислительных процессов;
2. среда – модель аппаратных средств с реконфигурируемой структурой;
3. реализация – исчисление, определяющее интерпретацию поведения в конкретной среде при заданных исходных данных.

3.1. Основные понятия и определения предлагаемой модели

В составе РВУ будем выделять три основные части:

- физическую среду – аппаратные средства, определяемые составом функциональных элементов и коммутирующей сетью;
- логическую среду – управляющие средства, которые поддерживают работоспособность физической среды и реализуют выполнение той или иной задачи и способ реконфигурации;
- поведенческую среду, которая определяет конкретную реализацию вычислительного процесса на РВУ.

В предлагаемой модели под словом реконфигурация понимается децентрализация управления и вычислений в физической и логической средах. Это значит, что в физической среде есть устройства способные функционировать автономно, т.е. решать индивидуальную “перегружаемую” задачу, а логическая среда обеспечивает использование реконфигурируемости физической среды. В качестве таких устройств мы будем иметь ввиду ПЛИС. Такое понимание охватывает широкий класс вычислительных систем, к которым относятся: измерительно-вычислительные системы, бортовые вычислительные комплексы, системы управления экспериментом, технологическими процессами, высокопроизводительные вычислительные комплексы на базе спецпроцессоров.

Динамику функционирования РВУ определяет взаимодействие процессов реализации конкретной вычислительной подзадачи при ее решении и совместная работа с логической и физической средами. Схематично наше представление о динамике можно охарактеризовать так. Каждый отдельно взятый вычислительный процесс определяет логическую последовательность действий. Эти действия есть либо реализация статического, либо динамическое использование логического ресурса. Использование сопровождается передачей параметров. Так выглядит выполнение совокупности процессов в РВУ с точки зрения глобальной организации вычислительного процесса.

Результатом выполнения действия по отношению к логической среде является формирование сообщения определенного типа и имени процесса, которому необходимо передать управление и само сообщение. Под сообщением мы будем понимать совокупность параметров обращения. Тип сообщения – это класс эквивалентности на множестве допустимых сообщений.

Передача управления может происходить по инициативе самого процесса, а может произойти и в результате воздействия на исполнителя извне (прерывания), например со стороны другого исполнителя под влиянием выполняемого на нем процесса. Одному исполнителю может быть сопоставлено несколько процессов. Их количество ограничено физическими характеристиками исполнителя.

Исполнитель – это модель физической среды. Исполнители бывают динамические и статические. Статический исполнитель может выполнять только фиксированные задачи из заранее определенного конкретного набора. Динамический может менять набор задач. Исполнители, объединенные каналами связи, могут обмениваться сообщениями и образуют реконфигурируемый исполнитель. Канал передает эти сообщения без каких либо изменений от одного исполнителя к другому, внося определенную временную задержку. Величина этой задержки и объем единовременно передаваемой информации есть постоянные характеристики данного канала.

Каждый реконфигурируемый исполнитель связан с определенным набором исполнителей. Этот набор ограничен и не изменяется в процессе реализации задачи. Последовательность логических действий по реализации процесса также фиксирована.

Так кратко, на содержательном уровне, можно описать наше представление о существе рассматриваемого явления и те концепции, на которых создается модель. Подчеркнем, что мы задаем модель не какой то конкретной вычислительной системы, а модель класса реконфигурируемых вычислительных устройств, характеристики которого были сформулированы в [1].

3.2 Математическая модель функционирования реконфигурируемых вычислительных устройств

Конкретизируем понятие поведения, т.е. детально опишем модель динамики организации реконфигурирования вычислительного процесса.

Поведение процесса будем описывать в форме тензорного уравнения как умножение со сверткой подмножества области отправления отображения на график этого отображения. Результатом при этом, является подмножество области прибытия отображения. [5]

Использование такого описания поведения процесса позволяет отказаться от меток шагов преобразования, т.к. их роль представляют обозна-

чения подмножеств областей отправления. Более того, в этом случае отпадает необходимость в описании момента передачи управления при описании поведения процесса, т.к. каждое отображение начинает выполняться только тогда, когда в результате выполнения других отображений сформируется подмножество области отправления данного отображения.[6].

Таким образом, поведение процесса можно описать тензорным уравнением преобразования его шагов, что позволяет рассматривать множество взаимодействий реализации алгоритмов конкретной задачи.

Пусть M – множество типов сообщений в РВУ и P – множество наблюдаемых процессов в системе. Множество P состоит из элементов, которые принадлежат либо множеству процессов логической среды P_L, либо множеству подзадач вычислительного алгоритма P_A, причем P = P_L $\bigcup$ P_A и P_L $\bigcap$ P_A = 0.

Выполнение шага процесса P_i определяется следующим образом:

1. инициализируется воздействие, которое заключается в получении сообщения типа a из множества M_I и управления от процесса P_j;
2. выполняется реализация, которая состоит из последовательности внутренних действий $\{q_i\}$, множество которых обозначим через Q, направленных на определение отклика на инициализируемое воздействие.
3. формируется реакция, которая заключается в посылке нового сообщения b из множества M_O.

Шаг процесса S определим как тройку S = (M_I, M_O, P), где M_I - множество исходных сообщений, M_O – множество выходных сообщений (M = M_I $\bigcup$ M_O), P – множество состояний наблюдаемых процессов.

В этом случае, выполнение шага описывается тензорным уравнением

$$S^{M\_O,P_j} = S^{M\_I,P_i} Q^{M\_O,P_j}_{M\_I,P_i} \qquad (1).$$

Важной характеристикой шага является его сложность. Сложность шага – величина, определяющая время выполнения его внутренних действий. Эти действия суть использования статических логических ресурсов, поэтому можно определить тензор W_i^{qi}, который однозначно задает сложность $W_i^{qi} = W_1^{q1} + W_2^{q2} + \ldots = W^q$. Очевидно, что если внутренние действия отсутствуют, то $W_i^{qi} = 0$.

Для оценки времени на передачу сообщений между исполнителями мы введем такую характеристику сообщения, как временная сложность шага. Определим ее с помощью тензорного уравнения $C\_T^b = M\_I_a^i Q_i^{ab}$,

которое реализует график M_O^b на множестве типов сообщений с областью значений – множество натуральных чисел *i*.

Историей процесса P назовем непустую конечную последовательность его шагов, замкнутую слева, т.е. если $H_P S^P_i = S_1\ S_{2....}\ S_k$ история, то $\forall i : i \le k$ и последовательности $S_1\ S_{2....}\ S_k$ – история. Наше предположение о конечности истории основано на представлении о реализуемости вычислений под надзором наблюдателя, а всякое наблюдение конечно.

Для выполнения процесса P потребуется фиксированное число шагов его истории, которые можно охарактеризовать длиной истории K_P. Истории процессов P_1 и P_2 равны тогда и только тогда, когда равны их длины $K_{P1} = K_{P2}$ и $\forall i : 1 \le i \le K_{P1}$, $S_{1i} = S_{2j}$, где S_{ij} –

i-тый шаг j-го процесса.

Два шага S_1 и S_2 равны тогда и только тогда, когда совпадают типы сообщений и типы процессов. Равенство сложностей шагов в этом случае не требуется.

Подисторией процесса P называется любой постфикс этой истории. Цепь – это любое подслово слова H_P. Поведением процесса P назовем множество всевозможных историй этого процесса, которое мы будем обозначать BH_P.

Проанализируем структуру поведения процесса BH_P с целью его описания. Для этого историю процесса $H_P S^P_i = S_1\ S_{2...}\ S_k$, правая часть которой представляет цепь $k = S_1\ S_{2...}\ S_k$ представим в блочном виде: $S_1\ S_{2...}\ S_l = l_1$, $S_{l+1}\ S_{l+2...}\ S_k = l_2$.

Введем две операции: следования $\rightarrow$ и альтернативы $\neg$. Цепь $k = l_1 \rightarrow l_2$ такова, что

$k = l_1 \neg l_2$, а l_1 – префикс, l_2 – постфикс k, т.е. операция следования есть конкатенация цепей, рассматриваемых как последовательность символов в алфавите S^P_i. Нетрудно видеть, что операция $\rightarrow$ ассоциативна, но не коммутативна.

Если поведение процесса P задается выражением $H_{P1} \neg H_{P2}$, то сам процесс, в зависимости от тех или иных условий, может быть реализован либо историей H_{P1} либо H_{P2}. Правило выбора способа реализации операция $\neg$ не определяет. Основные свойства операции $\neg$ следующие:

1. $l_1 \neg l_2 = l_2 \neg l_1$;
2. $(l_1 \neg l_2) \neg l_3 = l_1 \neg (l_2 \neg l_3)$;
3. $l_1 \rightarrow (l_2 \neg l_3) = l_1 \rightarrow l_2 \neg l_1 \rightarrow l_3$.

Таким образом, операция $\neg$ коммутативна, ассоциативна и дистрибутивна справа относительно операции $\rightarrow$.

Теорема 1. Любой процесс P, который имеет историю H_P ($P : H_P \in BH_P$) может быть представлен в виде: $H_P = H_{P1} \rightarrow \sim H_P$.

Теорема 2. Пусть BH_P' множество всевозможных историй $H_P' = H_{P0} \to \sim H_P'$, причем $\sim H_P' \in BH_P'$, где BH_P' – множество подисторий историй из BH_P'. Тогда $BH_P' = H_0 \to \Sigma \sim H_{Pi}$, где сумма берется по $\sim H_{Pi} \in \sim BH_P'$.

Обозначим элементы $t^* H_{P0}$ такими, что они не принадлежат $H_{P0} \to l$, где $l \in S^*_P$.

Теорема 3. Для любого фиксированного $H_{P0} \in BH_P$ представление $t^* H_{P0} \in BH_P$ в виде $H_0 \to \Sigma \sim H_{Pi}$ единственно с точностью до ассоциативной перестановки слагаемых.

Следствие. Префикс H_P' любой истории из BH_P задает на истории класс эквивалентности $t^* H_P' = \{H_P \mid H_P = H_P' \to \sim H_P\}$. Причем для $\forall$ H_P', H_P'' из BH_P таких, что $H_P' \neq H_P''$ следует, что $t^* H_P' \cap t^* H_P'' = \varnothing$.

Теорема 4. Для любого множества BH_P существует тензор преобразования (возможно инвариант), который устанавливает взаимно-однозначное соответствие между множествами путей историй процесса и множества историй из BH_P:

$$S_{P_i}^k T_k^{P_i} = BH_{P_i}$$

Одной из основных особенностей поведения реконфигурируемых вычислительных процессов является недетерминизм. Можно определить два его вида: внутренний и внешний. Будем говорить, что выражение $S_i + S_j$ описывает внешний недетерминизм, если множество M_I из процесса S_i не равно соответствующему множеству исходных сообщений из процесса S_j и внутренний, если они равны.

Способ реализации задачи определяется множеством поведений процессов $BH_p = \bigcup_{P_i \in p} bhp_i$ образующих ее решения. Множество шагов $S^P{}_i$ в поведении BH_P частично упорядочено. Это отношение отражает взаимодействие между процессами и порядок следования шагов в истории каждого процесса. Проанализируем это отношение более подробно, т.к. оно описывает причинно-следственные связи на множестве действий при решении задачи и во многом определяет семантику реконфигурации в РВУ. По определению, из цепочки S_i следует цепочка S_j тогда и только тогда когда

$\exists$ $H_P \in BH_P : (S_i \neg S_j) \in H_P$ и , $S_j \in S^P{}_k$; где S_i – причина, а S_j – следствие. Это отношение обозначим R □ (P) = { (S_i, S_j) | из S_i следует S_j }. Как видно, оно означает причинно-следственные связи на $S^P{}_k$, где $k \in P$.

Это отношение можно распространить на случай событий. Выполнение шага процесса P согласно (1) состоит в инициализации воздействия путем получения сообщения из M_I и реакции, которая заключается в формировании нового сообщения из M_O. Если обозначить событие через *e,* с надлежащим индексом, то получим: из цепочки e_i следует цепочка e_j тогда и только тогда когда

$\exists\ S_k : (e_i = M\_I(S^P{}_k)\ \&\ e_j = M\_O(S^P{}_k) \vee \exists\ S_l :\ e_j \in S_k\ \&\ e_j \in S_l\ \&\ (S_k, S_l) \in R\ \square\ (P))$

Определим отношение

$R\sim(P) = \{(S_i, S_j) \mid \exists Hp_1,\ \exists Hp_2 : S_i \in Hp_1\ \&\ S_j \in Hp_2\ \&\ M\_O(S_i^P) = M\_I(S_j^P)\}$,

т.е. реакция на шаге S_i в поведении процесса P_1 есть обращение к шагу S_j в поведении процесса P_2. Если $(S_i, S_j) \in R\sim(P)$, то этот факт будем записывать как тензорное уравнение $S_j = S_i\ R^i_j$. Отношение $R\sim(P)$ несимметрично, иррефлексивно.

Обозначим $R > (P) = R\sim(P) \bigcup \{ \bigcup_{P_i = P} R\ \square\ (P) \}$. Не трудно показать, что эти отношение также несимметрично и иррефлексивно.

По аналогии с ранее сделанным, распространим отношение $R\sim(P)$ на случай событий:

из цепочки e_i следует цепочка e_j тогда и только тогда когда

$\exists\ S_k\ \exists\ S_l : (S_k\ S_m) \in R\sim(P)\ \&\ e_i \in S_k\ \&\ e_j \in S_m$

Это отношение обозначим $Re\sim(P)$.

Историей выполнения вычисления называется тройка $(H^i{}_P,\ M\_I_i{}^j,\ R\sim P)$, т.е. это совокупность историй процессов $H^i{}_P$, удовлетворяющих отношению $R\sim(P)$; $M\_I_i{}^j$ – сообщение типа a_i из множества исходных сообщений процесса P^j, которое задает первый шаг в том процессе, с которого начинаются вычисления.

Рассмотрим различия в описании поведения процессов логической среды и описания вычислений, проводящихся в ней.

1. В поведении прикладных процессов вместо некоторого шага может быть указан тензор процесса, который свернут до шага. Это означает, что процесс начиная с этого места ведет себя так как свернутый процесс.

2. В уравнениях выполнения процессов из логической среды, последовательности шагов вычислений представляются переменными с именами специального фиксированного вида, которые образуют множество Dm с элементами dmi. Реакция в которой указано dmi, означает передачу управления прикладному процессу, сопоставленному этому имени. Соответствие между Dm и процессами из P устанавливается уравнением $SHD = P^i\ Dm_i$. Здесь предполагается, что SHD не зависит от времени. Однако, наделяя SHD различными свойствами, например, вводя индексы времени, можно описывать динамические процессы развивающиеся во времени.

3. Считаем, что множества Dm и P_L фиксированы для задачи. В этом случае, исходя из ограничения Dm, следует ограничение на множество допустимых к реализации вычислений, а именно $|Dm| \geq |P|$.

4. В логической среде есть процесс с именем Stop, обращение к которому завершает процесс. В реакциях шагов может присутствовать предо-

пределенная переменная back, которую можно представить, как вершину стека. Для работы с ней есть две функции put – поместить в стек и get – выдать значения.

Проанализируем виды реконфигурирования и их семантику. Есть два вида реконфигурации: функциональная и полная. Первая соответствует случаю, когда функциональные элементы, объединяются при помощи коммутационной сети для решения той или иной подзадачи. В этом случае, выполнение вычислений происходит одним исполнителем, т.е. процессы разделяют ресурсы одной и той же логической и физической сред. Во втором случае, вычисления выполняются независимо на разных процессах, обмениваясь при необходимости сообщениями.

Введем две операции композиции процессов: $P_1 \sim P_2$ – совместное использование функциональных элементов для решения ряда подзадач; $P_1 \parallel P_2$ – полная реконфигурация. Существует несколько подходов в описании способа реконфигурирования. Например, один из самых распространенных это использования описания типа семантики применяемой в языках программирования, таких как CSP [7]. Другой, это семантики описания операторов работы с разделяемыми переменными, который чаще всего используется в теории сетей [8].

Для описания процессов при объединении функциональных элементов с помощью коммутирующей сети будем использовать семантику чередования. Это означает, что события которые задаются процессами HP_1 и HP_2, образуют цепочку, в которой они не имеют предпочтения в порядке следования между собой. Ограничение на чередование определяет некоторое отношение независимости, задаваемое на S^P. Такая семантика учитывает единство временной оси исполнителя. Это выражается в том, что все события, происходящие на данном исполнителе, собираются в цепочки.

Семантику независимых реконфигурируемых структур, организующих выполнение вычислительных процессов, будем описывать отношениями частичного порядка на множестве их событий. Запись $P_1 \parallel P_2$ означает, что цепочка событий, определяемая историей HP_1 , появляется между точками взаимодействия вместо и независимо от цепочки, определяемой историей HP_2. Анализ взаимодействия этих видов реконфигурации и их эквивалентность будет проведен ниже.

Конкретизируем понятие физической среды, которая в предлагаемой модели реконфигурируемых устройств представляет исполнитель. Прежде всего, рассмотрим реконфигурируемые устройства у которых функциональные элементы структуры объединяются тем или иным способом при помощи коммуникационных каналов. Для этого введем понятие пространственно реконфигурируемый исполнитель.

Пространственно реконфигурируемый исполнитель SE – это математическая структура:

SE=<Pa, Tm_n^e, $M^P{}_i$, C_S , A_{Map}^{Tm} > (2)

у которой Pa – атомарный процесс; Tm_n^e – тензор модельного времени; $M^P{}_i$ – память исполнителя; C_S – пространственно реконфигурируемый вычислительный узел; A_{Map}^{Tm} – арбитр. Определим строго эти понятия.

В множестве наблюдаемых процессов P можно выделить подмножество атомарных процессов Pa. Атомарным называется процесс, который для конкретного устройства имеет минимальное количество шагов истории процесса. В этом случае атомарный процесс задает частичное отображение множества исходных сообщений M_I в множество выходных сообщений M_O и образует множество типов атомарных сообщений. M_A.

Основные свойства атомарных процессов таковы:

1) $\forall a_i : \exists M\_I : \exists! M\_O : a_i \in M\_A$,

$\forall b_i : \exists M\_O : \exists! M\_I : b_i \in M\_A$

т.е. для любого $a_i \in M\_A$ определено множество исходных сообщений множество выходных сообщений. Между M_I и M_O тензор $S^{M\_I,Pa}$ задает однозначное соответствие. Поэтому недетерминизма на уровне исполнителя нет.

2) Выполнение атомарного процесса не прерывается, т.е. начавшись, атомарный процесс будет закончен за определенное фиксированное, конечное время без останова.

3) После выполнения атомарного процесса управление всегда возвращается тому процессу, который инициировал его выполнение, если ему на вход поступило сообщение допустимого типа.

4) У атомарного процесса моменты передачи сообщения результата и моменты возврата управления совпадают; если управление передано, то сообщение доступно.

Каждый атомарный процесс характеризуется сложностью, которая определяет время выполнения его внутренних действий в тактах. Суть этих действий, заключается в использовании статических физических ресурсов сети. Поэтому можно определить тензор W_i^a который однозначно задает время выполнения атомарного процесса множеством натуральных единиц времени (тактов) по часам исполнителя. В этом случае можно отметить, что время выполнения любого атомарного процесса конечно, постоянно и равно $P_a W_i^a$.

Модельное время есть тензор $\mathrm{Tm}_{\mathrm{n}}^{\mathrm{e}}$, который сопоставляет событию *e* количество тактов прошедших к моменту его возникновения на данном исполнителе. Момент наступления события считается с момента начала выполнения первого атомарного процесса. Область отправления отображения модельного времени определена на множестве всех событий e_i в процессах P_a, приписанных данному исполнителю и представляет собой, в общем случае, множество натуральных чисел. Постулируем свойства тензора Tm.

Постулат 1. $\forall e_i, e_j : e_i, e_j \in BHp_m \,\&\, e_i \sim\!\!\rightarrow e_j$, т.е. в одном процессе причина всегда наступает раньше следствия $\mathrm{Tm}_i < \mathrm{Tm}_j$.

Постулат 2.

$$\forall e_i, e_j : e_i \in BHp_k \,\&\, e_j \in BHp_m \,\&\, e_i \sim\!\!\rightarrow e_j \,\&\, p_m \sim p_k,$$

т.е. если причина и следствие принадлежат разным $\mathrm{Tm}_i < \mathrm{Tm}_j$ процессам и эти процессы выполняются на одном и том же исполнителе (устройстве), то по часам этого исполнителя причина наступит всегда раньше следствия.

Из приведенных выше постулатов можно сформулировать следующие следствия:

Следствие 1. Каждый процесс имеет длительность, т.е. между двумя последовательными событиями одного процесса (между причиной и следствием) всегда происходит хотя бы один такт времени.

Следствие 2. $\forall e_i, e_j : e_i \rightarrow e_j$, и $\mathrm{Tm}_i < \mathrm{Tm}_j$ т.е. любой процесс выполняется последовательно.

Следствие 3. $\forall \mathrm{e_j} : \mathrm{e_0} \rightarrow \mathrm{e_j}$ и $\mathrm{Tm}_0 < \mathrm{Tm}_j$, т.е. обращение к процессу всегда наступает раньше (по часам данного исполнителя), чем возникает какое либо событие в этом процессе.

Следствие 4. Если

$$\forall \mathrm{e_i}, \mathrm{e_j} : (\mathrm{e_i}, \mathrm{e_j}) \in \mathrm{R} \sim \mathrm{P} \,\&\, \mathrm{e_i} \in \mathrm{BHp_k} \,\&\, \mathrm{e_j} \in \mathrm{BHp_m} \,\&\, \mathrm{P_k} \sim \mathrm{P_m},$$

то из этого следует $\mathrm{Tm}_i < \mathrm{Tm}_j$, причем R~P транзитивное замыкание. Данное следствие означает, что если два события связаны причинно – следственной связью, принадлежат разным процессам, которые выполняются на одном и том же исполнителе, то причина по часам этого исполнителя наступает всегда раньше следствия.

Теорема 5. Тензор $\mathrm{Tm}_{\mathrm{n}}^{\mathrm{e}}$ не нарушает отношения причинно – следственного порядка цепочки процессов, которые выполняются одним исполнителем.

Обозначим через t переменную на множестве вещественных чисел R, значение которой равно величине астрономического времени, код которо-

го можно определить с помощью астрономических наблюдений. Такое задание времени никоим образом не зависит от свойств вычислительного устройства, но это время важно для организации конкретных вычислений.

Рассмотрим определение множества отправлений тензора Tm_n^e, а именно, обозначим через Tm_t количество тактов наступивших к моменту времени t. При этом постулируем следующие свойства:

Постулат 3. $\forall t, \exists P, \forall \Delta t : \rho, t, \ \Delta \in R \ \& \ 0 < \Delta < \rho$

В этом случае Tm_t(t + Δt) – Tm_t = 1& Tm_t(t + Δt) – Tm_t = 0. т.е. часы идут с минимально различимым тактом ρ. На практике точность измерения ρ флюктуирует. Поэтому для практического применения Постулат 3 удобнее представить в следующем виде:

Постулат 3_1. $\forall t, \forall \Delta t \forall \rho, \exists \zeta : 0 < \Delta < (\rho - \zeta) \ \& \ \rho - \zeta < \rho < \rho + \zeta$

В этом случае Tm_t(t + Δt) – Tm_t = 0& Tm_t(t + ρ) – Tm_t = 1.

При таком подходе можно однозначно обеспечить показания часов в разных экспериментах только лишь в рамках определенного интервала наблюдений δt, такого, что $|[\frac{\delta t}{\rho - \zeta}|] = |[\frac{\delta t}{\rho + \zeta}|]$, где |[…]| - целая часть выражения. Отсюда, зная ρ и ζ можно вычислить δt, либо зная заранее δt, определить длительность шага моделирования.

Для хранения и правильного функционирования процесса требуется определенный объем памяти в котором хранятся параметры реконфигурации. Определим тензор M^P_i, который в качестве области отправления отображения на график этого отображения имеет множество процессов и множество натуральных чисел соответственно. Такое задание тензора M^P_i позволяет определить необходимый объем памяти для хранения и выполнения процесса. В тоже время сам исполнитель может иметь определенное (возможно большее) количество единиц памяти N.

На данном конкретном исполнителе может выполняться только такое множество процессов, у которых наличная память N будет больше чем необходимая.

Понятие вычислительного узла является ключевым для построения модели исполнителя. Пространственно реконфигурируемый вычислительный узел C_S это математическая структура C_S = <IN, ENT_{IN}^n, OUT, EXT_k^{OUT}>, где IN – множество полюсов, которые мы будем называть входами, тензор ENT_{IN}^n – определяет назначение входов; тем самым, каждому входу ставится во взаимно – однозначное соответствующие полюсы n вычислительного узла, OUT – множество полюсов,

которые мы будем называть выходами, тензор EXT_k^{OUT} – определяет назначение выходов.

Полюса $out_i \in OUT$ обеспечивают передачу воздействий на процессы, размещенные на других исполнителях. Каждому out_i по определенному правилу или закону сопоставляется атомарный процесс P_out_i. Время срабатывания полюса out_i определяется тензором $Tm_i^{P-out_i}$. Также с каждым полюсом связана функция Map(n):N $\rightarrow$ {0,1}; Map(n) = 1, если $n \in [Tm\_t_0, Tm\_t_0 + P_{P\_out_i} \bullet Tm_i^{P-out_i}]$, в противном случае – 0. Здесь Tm_t_0 задает воздействие на P_out_i. Назовем эту функцию характеристической.

Один out_i может быть соединен с несколькими входами in_j своего исполнителя или других исполнителей. Всем этим in_j одновременно будет передано одно и тоже сообщение. Время передачи того или иного сообщения определяется характеристикой соответствующей связи. (Способ задания характеристики будет рассмотрен отдельно). Объем и тип передаваемого за одно обращение к out_i сообщения определяет атомарный процесс P_out_i.

Каждому входу in_j можно сопоставить (способ задания сопоставления будет рассмотрен отдельно) только один выход out_i своего либо другого исполнителя и один процесс из множества типов сообщений M, управление которому будет передано, если данный вход возбужден и выиграл арбитраж. Вход возбужден, если Map(n)=1 у выхода, связанным с этим входом.

Тензор A_{Map}^{Tm} описывает процедуру арбитража входов пространственно реконфигурируемого исполнителя. Область отправления отображения это множество значений тензор Tm, а множество характеристических функций определяет график этого отображения. Таким образом, в зависимости от состояния выходов, арбитраж позволяет однозначно передать управление на вход исполнителя.

Более сложная модель временного реконфигурируемого исполнителя. Конкретизируем эту модель.

Временный реконфигурируемый исполнитель DE это математическая структура:

$$DE = < \{SE_i\}, C\_D, M\_S_M^E > \qquad (3)$$

где $\{SE_i\}$ – множество пространственно реконфигурируемых исполнителей, C_D – распределенный вычислительный узел, M_S – поток сообщений.

Понятие распределенного вычислительного узла C_D определим как гиперграф, множество вершин которого образуют полюса C_S, входящие в состав соответствующего CE_i, т.е. это математическая структура: C_D = <{C_S_i}, E, ENT_{IN}^{n}, EXT_k^{OUT} >,

где {C_S_i} – множество реконфигурируемых вычислительных узлов, которое обозначим через V. По сути, множество вычислительных узлов, определяется следующим образом:

$V = \cup\{OUT_i \cup IN_i \mid OUT_i \in C\_S_i \,\&\, OUT_i \in C\_S_i\}$; $C\_S_i \in$ C_D;

E – множество дуг, которые мы будем обозначать ARC, и которое определятся, как

$E = \{ARC_j \mid ARC_j \in V \,\&\, \forall j \exists! OUT \in ARC_j\}$, т.е. дугу в гиперграфе образует множество полюсов, среди которых есть только один полюс типа выход;

ENT_{IN}^{n} – тензор разметки входов C_D : IN – вход в узел C_D, если

$\exists C\_S_j : C\_S_j \in C\_D \Rightarrow \forall ARC : IN \notin ARC \,\&\, IN \in IN_j \,\&\, IN_j \in C\_S_j$; т.е. это один из входов C_S, не использованных в дугах C_D;

EXT_k^{OUT} тензор разметки выходов C_D : OUT, если

$\exists C\_S_j : OUT_j \in OUT_j \,\&\, OUT_j \in C\_S_j \Rightarrow \forall ARC : OUT_j \notin ARC$, т.е. это один из выходов C_S, не использованных в дугах C_D;

Поток M_S формируется множеством дуг E по которым за время tt блоками m_v передается множество M сообщений. Каждому конкретному сообщению m_v_i соответствует время передачи tt_i, измеряемое в тактах исполнителя. Пару вида < m_v_i, tt_i, > будем называть пропускной способностью канала.

Модели пространственного и временного реконфигурируемых исполнителей позволяют при помощи конструктивных правил описать любую структуру вычислительного устройства. Конкретизируем и определим алгебру носителей для построения модели структуры такого устройства.

Обозначим через C множество вычислительных узлов: C = $CS \cap CD$, где CS(m,n) – множество пространственно реконфигурируемых вычислительных узлов CS, имеющих не более n входов и m выходов; CD – множество временных реконфигурируемых узлов. Для простоты изложения, без потери общности, введем следующие обозначения: C_S(i,j) – пространственно реконфигурируемый вычислительный узел с i входами и j выходами; Plug – пространственно реконфигурируемый вычислительный узел вида C_S(1,1). Будем считать, что $CD_1 = CD_2$, если:

1) $\forall C\_S_{1i} : C\_S_{1i} \in CD_1 \Rightarrow \exists C\_S_{2j} : C\_S_{2j} \in CD_2 \,\&\, C\_S_{1i} = C\_S_{2j}$;

2) $E_1 = E_2$; $ENT_{IN}^{n1} = ENT_{IN}^{n2}$; $EXT_{k1}^{OUT} = EXT_{k2}^{OUT}$;

т.е. вычислительные узлы равны с точностью до изоморфизма.

Определим на множестве вычислительных узлов: С операции объединения, слияния входов, композиции и замыкания. Везде далее считаем заданными вычислительные узлы $C\_D_1 = <V_1, E_1, ENT_{IN}^{n1}, EXT_{k1}^{OUT}>$, $C\_D_2 = <V_2, E_2, ENT_{IN}^{n2}, EXT_{k2}^{OUT}>$, $C\_D_1, C\_D_2 \in C$.

Объединением вычислительных узлов $C\_D_1 + C\_D_2$ будем называть узел $C\_D_3 = <V_3, E_3, ENT_{IN}^{n3}, EXT_{k3}^{OUT}> \in CD$ такой, что: $V_3 = V_1 \cup V_2$; $E_3 = E_1 \cup E_2$; $n_3 = n_1+n_2$; $k_3 = k_1 + k_2$;

$ENT_{IN}^{n3(i)} = ENT_{IN}^{n1(i)}$, если $1 \le i \le n_1$, и $ENT_{IN}^{n3(i)} = ENT_{IN}^{n2(i-n_1)}$ если $n_1 \le i \le n_1 + n_2$;

$EXT_{k3}^{OUT} = EXT_{k1}^{OUT}$, если $OUT \in V_1$, и $EXT_{k3}^{OUT} = k_1 + EXT_{k2}^{OUT}$, если $OUT \in V_2$.

Рис. 1 иллюстрирует эту операцию графически.

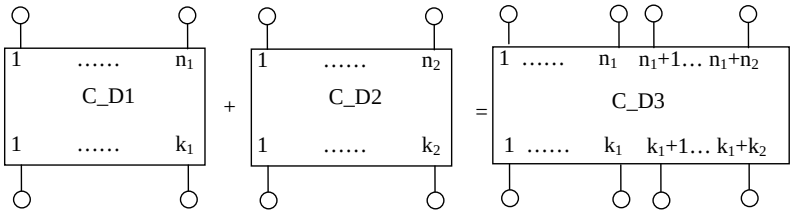


Рис 1.

Слиянием входов вычислительных узлов C_D1 # C_D2 будем называть узел $C\_D_3 = <V_3, E_3, ENT_{IN}^{n3}, EXT_{k3}^{OUT}> \in CD$ такой, что: $V_3 = ENT_{IN}^{n1} \cup \{EXT_{k1}^{OUT} \cup EXT_{k2}^{OUT}\}$; $E_3 = E_1 \cup E_2$; причем $ENT_{IN}^{n2} = ENT_{IN}^{n1}$ и $EXT_{k3}^{OUT} = EXT_{k1}^{OUT}$, если $EXT_{k3}^{OUT} \in EXT_{k1}^{OUT}$, и $EXT_{k3}^{OUT} = EXT_{k2}^{OUT}$, если $EXT_{k3}^{OUT} \in EXT_{k2}^{OUT}$. Эта операция определена для тех вычислительных узлов, у которых количество входов

равно друг другу, т.е. $ENT_{IN}^{n1} = ENT_{IN}^{n2}$. Сущность этой операции состоит в том, что входы C_D_1 отождествляются с входами C_D_2. Тем самым допускается многополюсное возбуждение входов узла C_D_3. Графическое представление этой операции представлено на рис. 2.

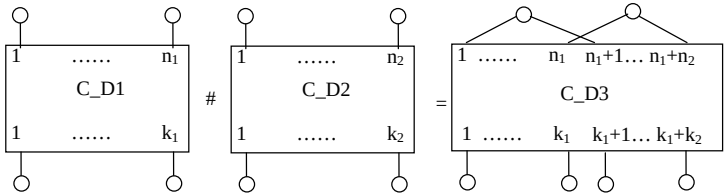


Рис. 2

Композицией носителей $C\_D_1 * C\_D_2$ будем называть вычислительный узел $C\_D_3 = <V_3, E_3, ENT_{IN}^{n3}, EXT_{k3}^{OUT}> \in CD$ такой, что: $ENT_{IN}^{n3} = ENT_{IN}^{n1}$, и $ENT_{IN}^{n3} = ENT_{IN}^{n2}$; $ENT_{IN}^{n1} = ENT_{IN}^{n3}$ $(n_1=n_3)$, $EXT_{k3}^{OUT} = EXT_{k2}^{OUT}$ $(k_3=k_2)$; $E_3 = E_1 \cup E_2 \cup (EXT_{k1}^{OUT} \cup EXT_{k1}^{OUT})$, где $k_1 = n_2$. Эта операция определена только для тех операндов, у которых число выходов первого вычислительного узла равна числу входов второго. Графически эту операцию можно проиллюстрировать рис. 3.

Для определения операции замыкания вычислительных узлов введем предварительные определения. Обозначим через Z_{IM}^{D} тензор, который сопоставляет множеству чисел $D_g \subseteq \{1, \ldots, k\}$ график с областью значений $IM_g \subseteq \{1, \ldots, n\}$. Теперь замыканием C_D_1 по Z_{IM}^{D} называется вычислительный узел: $C\_D_2 = [C\_D_1]_{g(i)} = <V_2, E_2, ENT_{IN}^{n2}, EXT_{k2}^{OUT}>$ такой, что

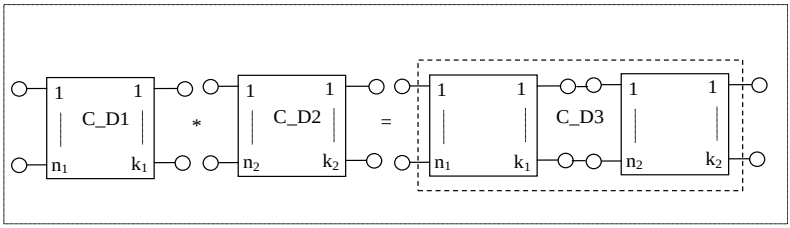


Рис. 3

$$EXT_{k2}^{OUT} = EXT_{k1}^{OUT} - EXT_{k1_{IM}}^{OUT_D} \; Z_{IM}^{D} \subseteq D_g,$$

$$ENT_{IN}^{n2} = ENT_{IN}^{n1} - ENT_{IN_D}^{n1_{IM}} \; Z_{IM}^{D} \subseteq IM_g,$$

$$E_2 = E_1 \cup \{(EXT_{ki}^{OUT} , \; ENT_{IN_D}^{n1_{IM}} \; Z_{IM}^{D})| \; EXT_{k1_{IM}}^{OUT_D} \subseteq D_g\},$$

Таким образом, при выполнении операции замыкания $[C\_D_1 \;]_{g(i)}$ вычислительного узла C_D_1 по Z_{IM}^{D} происходит перенумерация с уплотнением входов/выходов результирующего вычислительного узла C_D_2 . Графически эта операция представлена на рис. 4.

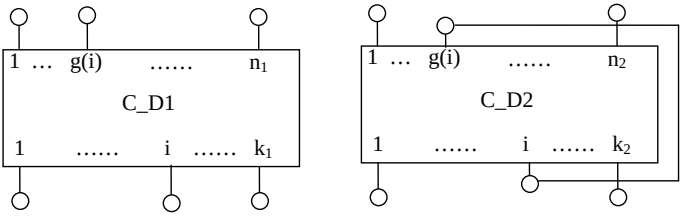


Рис. 4

Из выше приведенного определения вычислительного узла и операций объединений, слияний, композиции, замыкания следует, что структура < C, + , # , * , [] > образует частичную алгебру.

Теорема 6. При фиксированных m и n для любого временного реконфигурируемого вычислительного узла из CD существует разложение на множество пространственно реконфигурируемых вычислительных узлов CS с помощью операций объединений, слияний, композиции и замыкания.

Эта теорема гарантирует возможность представление структуры любого временного реконфигурируемого исполнителя в виде алгебраического выражения над пространственно реконфигурируемыми вычислительными узлами CS. Доопределим понятие исполнителя на случай алгебраического описания его структуры. В этом случае основную проблему представляет проблема переопределения потока сообщений M_S на случай вычислительных узлов типа Plug. Так как определение потока сообщений M_S на множестве дуг E определяет время tt, и блоки m_v, которыми передаются сообщения из M и задает пропускную способность канала (m_v, tt), то на случай узла типа Plug этот поток будет иметь следующий вид: $M\_S(Plug) = (m\_v, tt)_i$, если $Plug \in ARC_i$. Будем говорить, что это

имеет место, т.е. Plug $\in$ ARC_i , если выход узла EXT_i замкнут на вход ENT_j , такой что пара (EXT_i , ENT_j) $\in$ ARC_i.

Теперь постулируем свойства тензора Tm_n^e, который сопоставляет событию *e* количество тактов n наступивших к моменту его появления на данном исполнителе на случай временного реконфигурируемого исполнителя.

Постулат 4.

$\forall\ Tm_n^e : Tm_n^e \in SE_i \in DE_i \Rightarrow (t_0 - tt) - t_0 \geq 1$, где t_0 – момент астрономического времени начала передачи единичного блока m_v (момент начала передачи – возврат управления от процесса сопоставленного EXT)

Функционирование реконфигурируемого вычислителя определяется тем, как в каждом конкретном случае будет происходить интерпретация поведения вычислительного процесса на исполнителе. Поэтому для определения способа функционирования вычислителя необходимо построить историю вычислительного процесса по исходному поведению логической среды, исполнителю и начальному воздействию.

Конкретизируем связь логической среды и исполнителя. Для этого введем тензор привязки логической среды к исполнителю $B_{DE_i}^{LE_i}$, который устанавливает взаимно – однозначное соответствие между множеством поведений процессов логической среды LE = {BHp_i | Pa_i $\in$ {P_L $\cup$ Pa}} и множеством входов / выходов исполнителя .{SE_i} $\in$ DE_i; EXT = {ext_i | $\exists$ }SE : SE $\in$ DE&ext_i $\in$ SE}; .{SE_i} $\in$ DE_i; ENT = {ent_j | $\exists$ }SE : SE $\in$ DE&ent_j $\in$ SE}.

Основные свойства такого отображения следующие: 1) одному входу соответствует один процесс; 2) нет двух или более исполнителей, которым был бы приписан процесс с одним и тем же именем; 3) тензор привязки логической среды к исполнителю $B_{DE_i}^{LE_i}$ не меняется в течении всего периода наблюдений.

Для описания функционирования реконфигурируемого вычислителя определим вычислительное устройство как тройку COM = <BHp, SHD, CE>, где BHp – поведение вычислительного процесса; SHD – значение тензорного уравнения, которое задает соответствие между множеством переменных фиксированного вида Dm и процессами; SHD т.е. это по сути функция распределения вычислительного процесса в логической среде; CE – вычислительная среда. CE = <LE, $B_{DE_i}^{LE_i}$, DE>, а свойства всех этих объектов были уже определены.

Организация вычислительного процесса по выполнению вычислений на вычислительном устройстве есть исчисление, которое будем называть временно реконфигурируемым наблюдателем. Зададим его в виде набора

идентичных алгоритмов и правил взаимодействий между ними. Набор этих алгоритмов будем называть пространственно реконфигурируемым наблюдателем (или просто наблюдателем, если это не вызывает неоднозначности). Каждый из этих алгоритмов определяет выбор очередного шага из поведения каждого из процессов порождаемых прикладными вычислениями и логической средой.

Каждому пространственно реконфигурируемому исполнителю сопоставлен свой наблюдатель, который будем обозначать OBS. По существу каждое OBS задает частичное отображение $M \times P\_A \times A \to S$, которое определяется согласно (1). В этом случае, в зависимости от шага процесса S^*, процессов P и состояния арбитра A_{Map}^{Tm} формируется следующий шаг S следующим образом:

1 Если $A_{Map}^{Tm} \neq 0$, то P = $B_{Tm}^{Map}\ A_{Map}^{Tm}$ и $S \in S_i \in BHp = S_{P_i}^{k}\ T_k^{P_i}$,

где S_i – очередной шаг процесса P, соответствующий воздействию поступившему на вход A_{Map}^{Tm}. При этом если $p^* \in P\_L$, то put(p^*, s^*), иначе,

если $p^* \in P$, то $dm_j = p^*\ s^*$, где p^* определяется из SHD = $P^j Dm_j$, put(dm_j) и $S^* \in BHp$.

2. Если $A_{Map}^{Tm} = 0$, то

2.1) если переменная специального вида dm_k определяет реакцию на шаге s^*, т.е. $dm_k \in rpl(s^*)$, то p определяется из SHD = $P^k Dm_k$ и выбирается шаг следующий за указанным в dm_k воздействии, которое соответствует $rpl(s^*)$ (если таких шагов несколько, то выбор случаен);

2.2) если back $\in rpl(s^*)$, то p = get(back) и выбирается шаг, который следует за указанным в back и воздействие, которое соответствует $rpl(s^*)$ (если таких шагов несколько, то выбор случаен);

2.3) если p $\in rpl(s^*)$, то берем очередной шаг процесса P, если их несколько, то тот, у которого воздействие соответствует реакции (если таких шагов несколько, то выбор случаен);

2.4) если $rpl(s^*)$ содержит обращение к атомарному процессу, то сообщение в $rpl(s^*)$ преобразуется согласно определению данного атомарного процесса и следующим будет выбран шаг, который следует за s^* в поведении этого процесса, из которого данный s^*;

2.5) если в выбранном процессе нет шага с воздействием, соответствующим $rpl(s^*)$, то работа данного OBS блокируется.

Текущим значением p^* становится s^*.

Рассмотрим свойства наблюдателя на предмет его корректности. Под корректностью наблюдателя мы будем понимать то, что последователь-

ность цепочек шагов, которую порождает OBS, является историей программы, т.е. удовлетворяет отношению R > (P) = R~(P) $\cup$ R □ (P).

Пусть $S^*(P)$ – множество цепочек в алфавите S(P) и w $\in$ $S^*(P)$. Обозначим $[w]_{Pi}$, где $p_i \in P$, последовательность только тех шагов из w, которые принадлежат $S(p_i)$, и в том порядке, в котором они расположены в w, т.е. $[w]_{Pi} \in S^*(p_i)$. Назовем $[w]_{Pi}$ – проекцией w на p_i.

Для обоснования корректности наблюдателя надо показать то, что при $\forall i : p_i \in P, \forall j : OBS_j$ он порождает:

1) цепочку шагов, в которых не нарушены ни R □ (P), (причинно-следственные связи на BHpi), ни R ~ (P) (причинно-следственные связи между процессами), т.е. w удовлетворяет R ~ (P) и $\forall p_i : \exists Hp_i :$[w]Pi =Hpi, где w – цепочка шагов, получаемая при интерпретации. Таким образом, это будет означать, что OBSi корректно воспроизводит реконфигурацию основанную на совместном использовании функциональных элементов для решения ряда подзадач: P1~P2.

2) если процессы прикладных вычислений P распределены между двумя и более OBS, то порождаемые совокупности цепочек $\{w_k\}$ обладают тем свойством, что $p_i : p_i \in P_i, \exists w_k : [w_k]_{P_i} = Hp_i \,\&\, \{w_k\}$ удовлетворяют R ~ (P), т.е. OBS корректно воспроизводит реконфигурацию в форме полной реконфигурации P1 || P2.

Для корректного наблюдателя характерно то, что $\forall i : OBS$ не нарушает отношения R>(P), а это будет тогда, когда для любых SHD и $B_{DE_i}^{LE_i}$ множество OBS корректно порождает историю вычислительного процесса. Таким образом, введенные нами понятия корректны при корректности прикладных вычислений.

4. Заключение

В результате проведенных исследований разработана формализованная модель построения и функционирования реконфигурируемых вычислительных устройств. В строго математическом смысле в качестве модели предложена структура с исчислением, правилами вывода. Разработанная модель описывает все характерные особенности функционирования РВУ. В отличии от модели описанной в[4], предложенный нами временный реконфигурируемый исполнитель позволяет описать децентрализацию управления, конфликты на уровне физической среды и задать оба типа реконфигурации. Предложенная форма описания поведения вычислительного процесса отражает независимость вычислений от функционального элемента и времени. Исследована взаимная согласованность и корректность всех введенных в модели понятий. Все это в целом, являет-

ся основой для создания новой методики проектирования РВУ с более глубокой степенью оптимизации проектных решений.

Таким образом, разработанная формализованная модель построения и функционирования РВУ позволяет описать алгоритмические свойства и поведение реконфигурируемых вычислительных процессов и устройств. Предложенная модель может быть основой для анализа и синтеза реконфигурируемых устройств.

Список использованной литературы

1. Виноградов Ю.Н., Иванов Д.Г., Кушниренко Н.В. Структурный способ иерархической классификации реконфигурируемых вычислительных систем. // Вісник Національного технічного університету України "КПІ" Інформатика, управління та обчислювальна техніка. К.: ТОО "ВЕК+" № 46 . – 2007. С. 133 – 140 .
2. Палагин А.В., Опанасенко В.Н., Сахарин В.Г. Реконфигурируемые структуры на ПЛИС // УсиМ. – 2000. - №3 - С. 32 – 39.
3. Хорошевский В.Г. Инженерный анализ функционирования вычислительных машин и систем. - М.: Радио и связь, 1987.- 256 с.
4. Палагин А.В., Опанасенко В.Н. Реконфигурируемые вычислительные системы. – К.: Просвіта, 2006. – 280 с.
5. Чижухин Г.Н., Панферов В.П. Структурное проектирование вычислительных систем. – Пенза Пенз., политехн. Ин-т, 1985, - 88с.
6. Крон Г. Тензорный анализ сетей: Пер. с англ. / Под ред. Л.Т. Кузина, П.Г. Кузнецова. – М: Сов. Радио 1987. – 720 с.
7. Plotkin G. An operational semantics for GSP // Formal description of programming concepts. N.-H. Press, 1983. p 199 – 223.
8. Котов В.Е. Сети Петри. М.: Наука 1989. – 210 с.

УДК 004.415.2.043

САЛАПАТОВ В.І.

РЕЛЯЦІЙНА МОДЕЛЬ ДАНИХ ПРО ВНУТРІШНЄ УЯВЛЕННЯ ПРОГРАМ

В цій статті запропоновано реляційну структуру даних для внутрішнього подання програм. Це дає можливість формалізувати процес машинно-незалежної оптимізації програм із застосовуванням операцій реляційної алгебри та передати оптимізоване внутрішнє подання програми на останню фазу компіляції – синтез машинних кодів.

There are proposed the relation data structure for internal program imaging in this article. That give possibility formalize the machine independed optimization program process due to relation algebra operations and pass the internal optimized program imaging to the last phase of compiler work – machine code synthesis.

Класичний трьохадресовий код, що використовується для подання внутрішнього уявлення програм [1, 2], зручний лише для виразів, а для таких конструкцій, як присвоєння, цикли та розгалуження через свою жорстку структуру він є надмірним і тому не зовсім зручним. Реляційна модель подання даних дозволяє визначити для кожної конструкції мови семантику складових операторів, операнд sd разом з їхніми типами, а також місце розташування результату без жорсткої прив'язки до певних шаблонів. Крім того, стандартні реляційні операції над відношеннями дозволяють достатньо просто виконати машинно-незалежну оптимізацію програм з використанням операції пошуку за зразком [3]. Таким чином, внутрішнє подання програм доцільно представити у вигляді нормАлізованої реляційної БД, яка максимально компактно та стисло подає інформАцію без втрат і усуває суперечність даних. Тому буде цілком доречно застосувати таку модель даних і для внутрішнього подання програм та в подальшому використовувати реляційні операції на фазі машинно-незалежної оптимізації [3]. ER–модель внутрішнього уявлення програми має враховувати крім семантик операторів та операндів з їхніми атрибутами ще й порядковий номер оператора. З урахуванням цього така модель виглядає так, як наведено на малюнку 1. Сутність TOPUP відображає оператор внутрішнього подання. Через параметр SEM встановлюються зв'язок з сутністю OPERAND, яка визначає для оператора із заданою семантикою припустимий список операндів з атрибутами: TOP – типом операнда (приймальник, джерело тощо) та LTD – обмеження значення операнду. Для верхівки TOPUP визначається конкретний операнд (сутність DATA) через зв'язок PR та VN між сутностями OPERAND та DEFINE через сутність DATA. Безпосередній зв'язок конкретного опе-

ранду DATA з верхівкою TOPUP здійснюється через номер оператора NUM і далі через ім'я змінної або константи VN – із змінними та константами (сутність DEFINE).

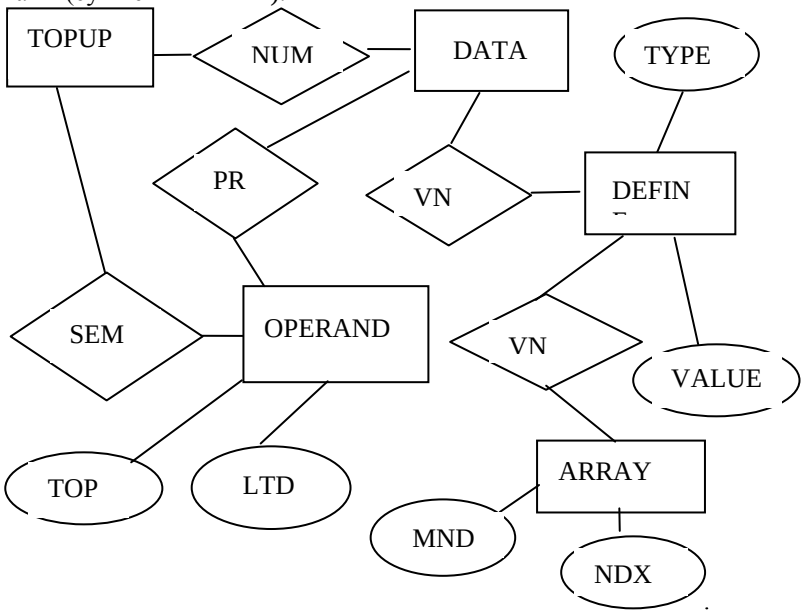


Рис.1. ER–модель даних про внутрішнє уявлення програми

Властивості TYPE та VALUE визначають тип операнду та його поточне значення. Сутність ARRAY через зв'язок VN пов'язана із сутністю DEFINE – ім'ям змінної та використовується, якщо операнд являє собою елемент масиву. При цьому атрибути MND та NDX вказують на номер індексу масиву та максимальне значення індексу відповідно.

Ідеальним варіантом щодо семантик операторів внутрішнього подання програм є наявність операторів машинних команд з такими самими семантиками в системі команд ЕОМ. А такі складні конструкції, як цикли та розгалуження можуть подаватися у вигляді низки операторів з елементарними семантиками [4]. Таким чином, передбачається подальше перетворення синтаксичного дерева в процесі його обходу, тобто аналізу, у низку операторів з певними елементарними семантиками. До таких семантик можна віднести арифметичні та логічні операції над цілими, дійсними і символьними даними, операції пересилок, операції зсуву, операції умовних та безумовних переходів і деякі специфічні операції (перекодування, пошук розподільника і т.і.). Як вже було зазначено [4, 5], в разі відсутності команди з певною семантикою в системі команд ЕОМ для неї може бути передбачено спеціальне макровизначення. Для подання внутрішнього уявлення програми необхідно пов'язати номер оператора із

заданою семантикою з оператором, що описується сутністю **TOPUP**. Тому у базовому відношенні основним ключовим атрибутом, що являє собою сутність оператора, обрано номер оператора NUM. Код SEM семантики вказує на тип операції внутрішнього оператора і в подальшому використовується для перетворення синтаксичного дерева у внутрішній код. Тому для внутрішнього подання програм таких даних нижче наведено відношення **TOPUP** із структурою, яка відображена на рисунку 2.

TOPUP	
NUM *	Номер верхівки
SEM *	Код семантики

Рис. 2. Структура відношення опису верхівок синтаксичного дерева

З кожною такою верхівкою дерева синтаксичного розбору мають бути пов'язані підлеглі верхівки – операнди. Ці підлеглі верхівки можуть бути кінцевими (листами) і тоді вони вказують безпосередньо на операнд (змінну або константу), або вказувати на іншу операцію і тоді вони вказують на верхівку типу TOPUP. Семантика кінцевої верхівки передбачає наявність конкретних операндів і тому потребує додаткового відношення для них. Для подальшого синтаксичного аналізу коди семантик кінцевих верхівок мають відрізнятися від кодів семантик операторів. Для таких кодів, що являють собою операнди, необхідно спеціальне відношення для опису даних. Це відношення повинно відображати константи та змінні, які були визначені у програмі користувача. Останні визначаються у програмі за допомогою декларативних операторів визначення змінних, констант і т.і. Для даних типу ***масив*** мають бути враховані індекси та їхні максимальні значення. Для даних типу ***запис*** необхідно ще вказати тип кожного елементу запису або його довжину та вказівник на елемент в середині запису. Остання вимога легко реалізується при використанні составного імені. Таким чином, кожен елемент запису визначається як звичайна змінна певного типу з певною довжиною. З оглядом усього сказаного нижче, на рисунку 3. представлена структура відношення **DEFINE** для визначення даних програм разом з їх типами та значеннями.

DEFINE	
VN *	Ім'я змінної
TYPE	Тип змінної
VALUE	Значення

Рис. 3. Структура відношення опису даних програми користувача

Ім'я змінної VN разом з її типом TYPE та початковим або поточним значенням VALUE разом з можливими індексами повністю описують

прості дані програми. Ключовий параметр VN дозволяє зв'язати ім'я масиву з його параметрами (кількістю індексів та їхніми максимальними значеннями. При визначенні змінних, масивів та констант у програмі створюється відповідний запис у відношенні **DEFINE**. Під час виконання програми або під час її часткового моделювання при компіляції значення кожної змінної може змінюватися і ці зміни мають враховуватися у відношенні **DEFINE.** В разі визначення даних у вигляді масиву необхідно задати його розмірність та максимальні значення індексів. Значення конкретного елементу масиву має визначається параметром VN через ім'я разом з усіма індексами. Для контролю значень індексів та їхньої кількості у процесі компіляції необхідно додаткове відношення. Нижче на рис. 4 наводиться структура відношення **ARRAY**, яка дозволяє виконувати контроль за зміною індексів масивів.

ARRAY	
VN *	Ім'я змінної
NDX	Номер індексу (для масиву)
MND	Максимальне значення індексу

Рис. 4. Структура відношення опису додаткових даних для масиву

Ключовий атрибут VN імені масиву пов'язує конкретний масив за його сутністю з номером індексу NDX, починаючи із старшого індексу, та максимальним значенням цього індексу MND. В разі опису структур або масивів структур, що визначається атрибутом TYPE у відношенні **DEFINE**, кожен елемент таких структур визначається як змінна із составним ім'ям, типом та можливим початковим значенням. Фактичне значення кількості записів у масиві структур має відповідати параметру MND у відношенні **ARRAY**. При цьому значення атрибуту NDX має бути рівним 1, тобто вказувати на одновимірний масив.

Для створення компілятора мови програмування необхідно обрати стандартний за семантиками набір операторів внутрішнього подання програм [6]. Цей набір складається із стандартних арифметико-логічних операторів для цілих і дійсних даних, операторів зсуву, пересилок, деяких спеціальних операторів, операторів розгалуження та операторів введення-виведення. Кожен з цих операторів так чи інакше пов'язаний з певними операндами. Для того, щоб задати синтаксис операторів мови програмування необхідно визначити правила граматики такої мови. З точки зору синтаксичної конструкції кожного з операторів мови програмування правила граматики визначають коректність операндів у семантичних верхівках при побудові синтаксичного дерева. Оскільки за основу подання даних в роботі було обрано реляційну модель, то цілком логічним є використання цієї моделі і для визначення синтаксичних правил мови програму-

вання. Так, якщо в деякій мові програмування маємо якусь обмежену множину операторів з їх семантичними верхівками S_i, то необхідно визначити операнди для цих операторів OP_j^i як підлеглі до них верхівки разом з типами, порядком розташування та можливими обмежувальними значеннями. З метою наближення семантик внутрішніх операторів мови до семантик машинних команд ЕОМ необхідно із загальної множини стандартних семантик S внутрішніх операторів мови обрати семантики тих команд, які присутні в системі команд ЕОМ. Таке наближення суттєво спрощує роботу компілятора при породженні машинних кодів програм [5]. Для спеціальних мовних конструкцій, що визначають тип оператора мови (умовний оператор, оператори циклу різного виду, тощо) на фазі синтезу кодів програм цілком припустимим є використання макросів. Відповідні макровизначення, які необхідно заздалегідь підготувати, мають відповідати семантиці такого оператора і повинні бути внесені до списку семантик команд ЕОМ.

Як вже було зазначено, трьохадресовий код для внутрішнього подання програм, у який перетворюється синтаксичне дерево програм і який широко застосовується в сучасних компіляторах, має стандартну жорстку структуру і для багатьох операторів є надлишковим і тому не зовсім зручним для подальшої машинно-незалежної оптимізації програм [7]. До того ж, кожен оператор внутрішнього подання програм має свій власний формат і може бути представлений як частинка графу, у верхівці якого знаходиться оператор з певною семантикою, а гілками його є операнди. На рисунку 5 показано структуру відношення **OPERAND**, що фактично описує семантичні правила щодо операндів в операторах мови програмування.

OPERAND	
SEM *	Код семантики оператора
PR *	Порядковий номер операнду
TOP	Тип операнду
LTD	Обмеження значення операнду

Рис. 5. Структура відношення опису операнду для оператора внутрішнього подання программ

За даною логікою кожен оператор із певною семантикою може мати стільки записів, скільки в нього є операндів. Це робить таке уявлення дуже гнучким та зручним для внутрішнього подання програми за допомогою БД. Набір базових семантик внутрішнього подання операторів разом з описом їхніх операндів можна вважати постійним або умовно постійним. Зміни у наборі базових семантик можуть вноситись по мірі подальшого розширення та удосконалення систем команд ЕОМ.

Але структура БД, що формується при синтаксичному аналізі для внутрішнього подання програм, має враховувати прив'язку операторів до операндів та їхній опис щодо конкретного елементу внутрішнього подання програми.

Для зв'язку внутрішніх операторів з конкретними даними потрібно відношення, що відповідає сутності DATA в ER-моделі даних про внутрішнє уявлення програми. На малюнку 6 наведена структура даних такого відношення DATA.

DATA	
NUM *	Номер оператора
PR *	Код призначення операнду
VN *	Ім'я змінної

Рис. 6. Структура відношення зв'язку оператора із змінними

Як бачимо, наведене відношення повністю складається з ключових атрибутів і має використовуватися з одного боку для зв'язку операторів з конкретними операндами, а з іншого боку через зв'язок PR – з відповідним операндом відношення OPERAND, що дозволяє контролювати наявність операнда згідно з його призначенням.

Повна структура БД для опису внутрішнього уявлення програми з наведеними вище відношеннями має вигляд, що представлений на рис. 7.

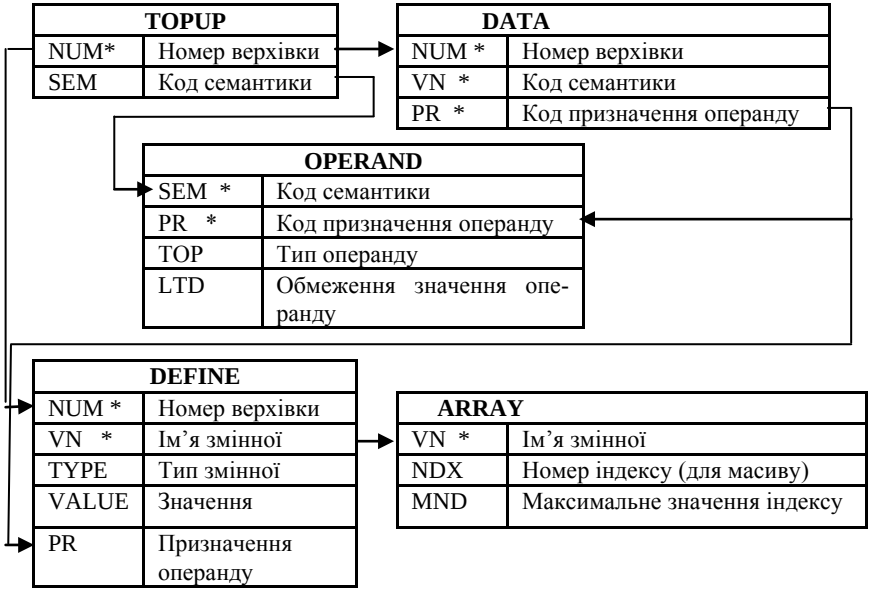


Рис. 7. Структура БД для опису внутрішнього уявлення програми

Даний малюнок відображає структуру БД для внутрішнього уявлення операторів мови. Верхівка дерева, що описується відношенням TOPUP, за своїм номером NUM вказує на конкретну змінну з іменем VN, типом змінної TYPE, її призначенням PR та поточним значенням VALUE. Для комутативних операторів порядковий номер операнду не має значення і визначається кодом семантики SEM. Відношення **OPERAND** фактично частково визначає семантику оператора мови, тому невідповідність типів операндів вказує на семантичну помилку. Якщо операнд є елементом масиву, то відношення **ARRAY** визначає його розмірність та обмеження на індекси.

Спеціальні мовні конструкції за допомогою макросів можуть перетворюватися у оператори виразу, але їхній синтаксис має описуватись також відношенням **OPERAND**, яке визначає типи операндів.

Запропонована структура даних для внутрішнього подання програм користувача дозволяє формалізувати процес їхньої машинно-незалежної оптимізації їа допомогою реляційних операторів [3]. Така структура використовується в компіляторі мови С мікро-контролера AVR, який розроблений на кафедрі кібернетики Черкаського національного університету імені Богдана Хмельницького.

Список посилань

1. А.Ахо, Р.Сети, Дж.Ульман. Компиляторы: принципы, технологии и инструменты. М.: Вильямс, 2001. – 768с.
2. А.В.Гордеев, А.Ю. Молчанов. Системное программное обеспечение. СПб.: Питер. 2001. – 736 с.
3. Салапатов В.І. Еквівалентні перетворення внутрішнього уявлення програм. Вісник ЧДТУ. № 45. Черкаси. 2004. с. 128-132
4. Салапатов В.І. Особливості формування інформаційних ресурсів для різних блоків програм. Вісник Херсонського державного технічного університету, Математичне моделювання та інформаційні технології. 2006. Херсон. с. 141-144.
5. Салапатов В.И, Кузнецов А.В. Структурированный макроассемблер виртуальной ЭВМ. Вестн. Киев. политехн. ин-та. Автоматика и электроприборостроение. К., Вып 22. 1986. с. 28-31
6. Салапатов В.І. Адаптація процесу синтезу кодів програм під систему команд цільової ЕОМ. Вісник національного технічного університету України «КПІ», Інформатика, управління та обчислювальна техніка. № 43. Київ. 2005 с. 75-79
7. Салапатов В.И. Особливості внутрішнього уявлення програм для подальшого синтезу його коду. Вісник ЧІТІ. № 3. 0.30 Черкаси. 2002. с. 78-84.

УДК 681.513

ЗАЙЧЕНКО О.Ю.,
КУХАРЄВ С.О.
КУХАРЄВА О.В.

ІМІТАЦІЙНЕ МОДЕЛЮВАННЯ МЕРЕЖ З ТЕХНОЛОГІЄЮ MPLS З МЕТОЮ КЕРУВАННЯ РІЗНОТИПНИМ ТРАФІКОМ

Сформульовано задачу розробки комплексу програм для розробки топології комп'ютерних мереж з технологією MPLS, моделювання поведінки мережі в залежності від налаштувань її складових та керування різнотипним трафіком. Наведено опис алгоритму імітації.

The article is formulated the problem of development a program system for MPLS networks topology designing, modeling network behaviour depending on its component setting, and traffic engineering. Simulation algorithm description is represented.

1. Вступ

За останні роки було зроблено багато спроб використати багатопротокольну комутацію за мітками (MPLS), що значно вплинуло на використання IP-мереж. MPLS являє собою технологію, з якою у перспективі будуть працювати більшість IP-мереж, у тому числі Internet. Використання технології MPLS надає мережі Internet новий принцип передачі пакетів, що впливає на перерозподіл потоків даних та на реалізацію віртуальних приватних мереж, а також дозволяє провайдерам більш ефективно забезпечувати задану якість обслуговування [1, 2].

Сучасна комп'ютерна мережа з технологією MPLS є дуже складною системою, в ній одночасно передається інформація різних типів з різними показниками якості обслуговування. На її роботу істотно впливають методи керування трафіком, а саме, організація та обслуговування черг, способи організації буферів комутаторів [3]. Побудова аналітичної моделі для дослідження впливу цих факторів на показники якості функціонування мереж в даній ситуації неможлива, обладнання для побудови мереж з технологією MPLS є досить дорогим, тому для попереднього проектування, а також для моделювання роботи мереж, аналізу та оптимізації їх характеристик пропонується імітаційне моделювання – один з найпотужніших інструментів дослідження складних систем [4].

Метою даної роботи є опис імітаційної моделі спеціалізованої комп'ютерної мережі з технологією MPLS, яка дозволяє задавати топологію мережі, характеристики її складових, моделювати поведінку мережі із заданою топологією з метою подальшого обчислення реальних характеристик мережі (середньої затримки, варіації затримки, долі відкинутих пакетів), виявлення "вузьких" місць.

2. Постановка задачі

Задається топологія спеціалізованої комп'ютерної мережі з технологією MPLS, що складається з вузлів зв'язку – LSR (маршрутизаторів комутації за мітками) та LER (прикордонних вузлів), а також каналів зв'язку відповідної пропускної спроможності. Модель мережі включає також зовнішні вузли зв'язку.

Зовнішні вузли є генераторами трафіка різних категорій сервіса (передача з постійною, змінною та доступною швидкостями – CBR, VBR та ABR відповідно).

Вводяться наступні показники якості обслуговування трафіка різних категорій:

- середня затримка пакетів (CTD),
- варіація середньої затримки (CDV),
- відсоток втрачених пакетів (CLR).

При передачі трафіка категорії CBR резервується фіксована смуга в каналі, не залежно від того, чи йде передача даних. Залишок смуги ділять між собою трафіки категорій VBR та ABR, причому VBR є більш пріоритетним [3].

Передача трафіків CBR, VBR та ABR здійснюється по віртуальних каналах, котрі встановлюються між джерелом та адресатом. Вони задаються разом з іншими параметрами мережі.

Задача даної статті – описати розроблену імітаційну модель комп'ютерної мережі з технологією MPLS, яка реалізує різні технології передачі трафіка та дозволяє керування ним.

3. Технології та показники якості обслуговування

Якість обслуговування – це здатність мережних засобів забезпечити необхідний сервіс для визначення класів трафіка в різних мережних середовищах. Основне значення технологій якості обслуговування полягає у забезпеченні пріоритетів, необхідної смуги пропускання, контрольованої затримки й варіації затримки й зменшенні відсотка втрат пакетів при передачі, а також у забезпеченні пріоритетності обслуговування деяких потоків з можливістю одночасної передачі інших потоків.

Завдання технології якості обслуговування полягає у забезпеченні усталеної роботи програм, що мають тверді вимоги до смуги пропускання й часу затримки проходження інформації мережею. Роль технології якості обслуговування в справедливому розподілі мережних ресурсів і забезпеченні параметрів, запитаних програмою і здатних бути виконаними мережею.

Функції якості обслуговування полягають у забезпеченні диференційованого й гарантованого обслуговування мережного трафіка шляхом передачі контролю за використанням ресурсів і завантаженості

мережі її операторові. Якість обслуговування являє собою набір вимог, пред'явлених до ресурсів мережі при транспортуванні потоку даних.

Ключовими характеристиками якості обслуговування є час затримки пакетів (CTD – максимальна затримка у мережі при передаванні пакетів між відправником та отримувачем), варіація затримки (CDV – різниця між максимальним і мінімальним часом доставлення пакетів від відправника до отримувача), відсоток загублених пакетів (CLR). Залежно від вимог до цих параметрів весь трафік може бути розділений на кілька основних класів – наприклад на трафік реального часу, трафік транзакцій і трафік даних. Згодом приналежність пакету до деякого класу визначає політику обслуговування даного пакету.

Основні технології забезпечення якості обслуговування – формування трафіка, організація й обслуговування черг (або розподіл ресурсів), керування навантаженням і резервування ресурсів мережі. Технології якості обслуговування відрізняються за способом керування – керування за швидкістю (за каналом) (формування трафіка, керування навантаженням) або керування чергою, а також змішані.

4. Опис імітаційної моделі

4.1. Вхідні дані моделі

Топологія мережі з технологією MPLS: n вузлів мережі (транзитні – LSR, прикордонні – LER), k каналів зв’язку. Модель мережі включає також зовнішні вузли зв’язку, що є генераторами трафіка різних категорій сервіса.

Перед початком процесу імітації роботи мережі проводиться встановлення значень параметрів вузлів, каналів зв’язку, мережі, а також подій, які відбуватимуться в мережі.

Параметри об’єктів, що беруть участь в імітаційній моделі:

- для вузла: ідентифікатор, внутрішня затримка, об’єм буфера, інтенсивність передавання даних.
- для канала зв’язку: ідентифікатор канала, ідентифікатори двох вузлів, що з’єднані даним каналом , пропускна спроможність канала.

Параметри мережі: тривалість процесу імітації, кількість тактів імітації, максимальна кількість вузлів та каналів зв’язку у моделі.

Параметри подій (однієї чи декількох), що мають відбуватися в мережі: вузол-відправник, вузол-отримувач, час настання події, об’єм трафіка, що передається від вузла-відправника до вузла-отримувача, категорія якості обслуговування, віртуальні канали.

Значення міток задає адміністратор.

4.2. Імітаційний алгоритм

При імітації роботи мережі використовується метод задання часу за допомогою фіксованих інтервалів часу (механізм фіксованих прирістов часових інтервалів), тобто відлік часу ведеться через заздалегідь визначе-

ні часові інтервали постійної довжини [4]. Усі події, які мають місце впродовж цього проміжку часу, будемо вважати такими, що відбулись одночасно. Отже, з точки зору процесу імітації події, які відповідають певному проміжку часу, починаються з його початку незалежно від того, в який момент часу вони відбуваються насправді.

Упродовж часу моделювання виконуються маніпуляції над пакетами. Під маніпуляцією будемо розуміти процедуру, яка перевіряє пакет за критерієм можливості операції пересилання та виконує безпосередньо процес пересилання. Перед проведенням маніпуляції пакети сортуються за певними критеріями, які визначають порядок їх обробки. Далі йде маніпуляція з першим пакетом у черзі, що може бути обробленим, потім виконується нове сортування для отримання нового «першого пакету». Ця процедура виконується доти, доки буде існувати перший пакет, який може бути оброблений.

Під час імітації організується цикл з багатьма вкладеннями (зовнішній цикл – за часом, наступний – за тактом, останній – за пакетами, котрі знаходяться у черзі пакетів для можливої обробки). Розглянемо окрему ітерацію.

Крок 1. На початку циклу в облікові параметри елементів моделі встановлюємо початкові значення для поточного такту.

Крок 2. До початку передачі через мережу MPLS пакетів трафіка будь-якого вигляду маршрутизатори LSR встановлюють відповідність між мітками та FEC (класами еквівалентності при пересиланні) у своїх таблицях. Крім того, проводимо узгодження характеристик трафіка та функціональних можливостей MPLS.

Крок 3. При одержанні даних про прив'язку міток до FEC кожен маршрутизатор LSR створює запис в таблиці LIB. Зміст таблиці відбиває відповідність між мітками та FEC і ставить у відповідність кожній парі «вхідний інтерфейс, вхідна мітка» пару «вихідний інтерфейс, вихідна мітка». При будь-якому новому узгодженні прив'язки міток до FEC записи в таблиці оновлюються. Таблиці міток, згідно яким кожен пакет направляється по відповідному LSP (тракту, що комутується за мітками), завжди повинні бути задані до того, як пакет почне свій шлях по мережі.

Крок 4. Тракти LSP створюються в напрямку, зворотному створенню записів у таблицях LIB. Кожен LSR отримує мітку від нижнього маршрутизатора. LSP створюється шляхом послідовної маршрутизації по ділянках, а якщо потрібна оптимізація розподілу трафіка, для визначення тракту використовується протокол CR-LDP, який гарантує виконання вимог до QoS, або протокол RSVP-TE.

Крок 5. Наповнюємо мережу пакетами.

Відповідно до розкладу подій відбувається генерація пакетів, що мають бути передані в мережу у поточному такті, тобто пакети потрапляють до зовнішнього вузла.

Крок 6. Вхідний маршрутизатор, визначивши, якому FEC належить прийнятий ним ззовні пакет, використовує таблицю LIB, щоб відшукати потрібну прив'язку «FEC-мітка», і інкапсулює цю мітку способом, який відповідає технології на рівні 2.

Крок 7. Для всіх існуючих пакетів стан активності переводимо у TRUE (стан активності говорить про те, що даний пакет ще може бути оброблений упродовж поточного такту).

Крок 8. Сортуємо пакети за порядком їхньої обробки залежно від накопиченої затримки. Результатом сортування є черга пакетов для обробки. Після обробки кожного пакета виконуємо повторне сортування. Таким чином, ми завжди працюємо з пакетом, що стоїть першим у черзі. Крок 8 проводимо доти, доки існує хоча б один активний пакет, над яким виконуємо маніпуляцію. Якщо пакет далі не може бути оброблений упродовж поточного такту, його стан активності переводиться у FALSE.

Одержавши пакет, маршрутизатор LSR витягає з нього мітку і використовує її як індекс у своїй таблиці пересилання. Як тільки знайдено запис, у якому значення вхідної мітки дорівнює значенню мітки, що була витягнута з пакету, маршрутизатор, згідно підзапису цього запису, заміняє вхідну мітку в пакеті вихідною міткою та пересилає пакет через вихідний інтерфейс, зазначений у підзапису, до наступного LSR, також зазначеного в тому ж під запису та збільшує накопичену затримку пакета.

Крок 9. Видаляємо з черги відкинуті (DROPED), доставлені (DELIVERED) пакети та ті, у яких стан ENABLED=FALSE.

Далі переходимо на наступну ітерацію. Кожна ітерація відповідає окремому такту конкретного часу.

В імітаційному алгоритмі реалізовано наступні технології забезпечення якості обслуговування:

- формування трафіка, що передбачає буферизацію згенерованого трафіка, який неможливо у даний момент передавати;
- контроль за встановленням з’єднання (якщо мережа не може забезпечити задану якість обслуговування, то трафіку відмовляється в обслуговуванні);
- резервування ресурсів мережі;
- організація та обслуговування черг (пріоритетне та непріоритетне обслуговування черг) [3].

5. Приклад результатів моделювання

Наведена вище імітаційна модель була використана у розробленому комплексі програм. Засобами програм цього комплексу можна спостерігати завантаженість окремих вузлів чи каналів зв’язку мережі з заданою топологією при заданих її параметрах (для подальшого виявлення найбільш та найменш завантажених елементів), а також кількість доставлених пакетів, досліджувати поведінку трафіка різних класів при використанні різних технологій забезпечення якості обслуговування, обчислювати ключові показники якості обслуговування з метою подальшого їх аналізу.

Структура мережі, на якій проводилися експерименти, представлена на рис. 1. Для кожного вузла та каналу зв'язку було задано свої параметри. Також було задано події, що повинні відбуватися в мережі: моменти передавання даних, їх об'єми, вузли-відправники та отримувачі інформації, категорії сервісу для даних, що передаються (рис. 2).

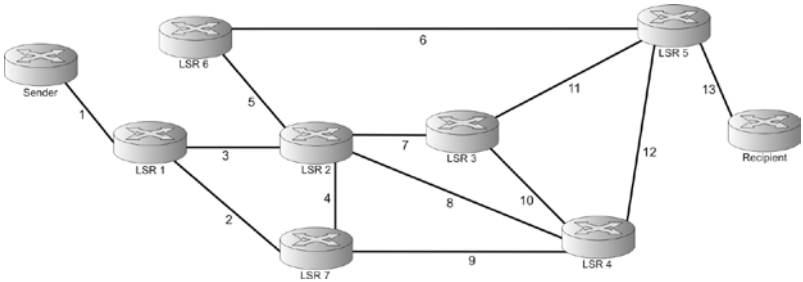


Рис. 1. Топологія експериментальної мережі

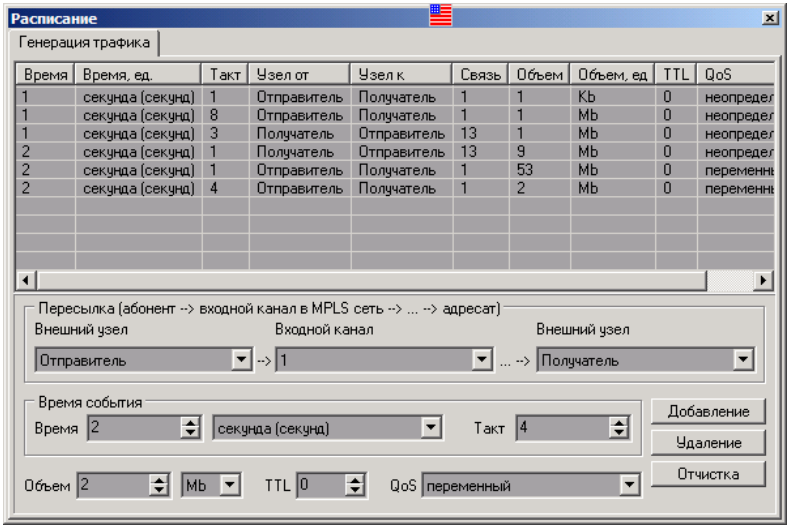


Рис.2. Розклад генерування трафіка

Досліджувалася завантаженість окремих вузлів та каналів зв'язку з метою виявлення найбільш та найменш завантажених елементів.

На рис. 3 наведено кількість пакетів у буфері комутатора маршрутизації за мітками LSR2 на перших 40 тактах імітації.

На рис. 4 наведено завантаженість каналу зв'язку 3 на перших 40 тактах імітації.

Аналогічним чином можна побудувати графіки завантаженості усіх вузлів та каналів зв'язку та, виходячи з цього, зробити висновок про

доцільність зміни топології мережі. Зокрема, з рис. 3 та 4 можна зробити висновок про те, що завантаженість розглянутого вузла та канала не є граничною.

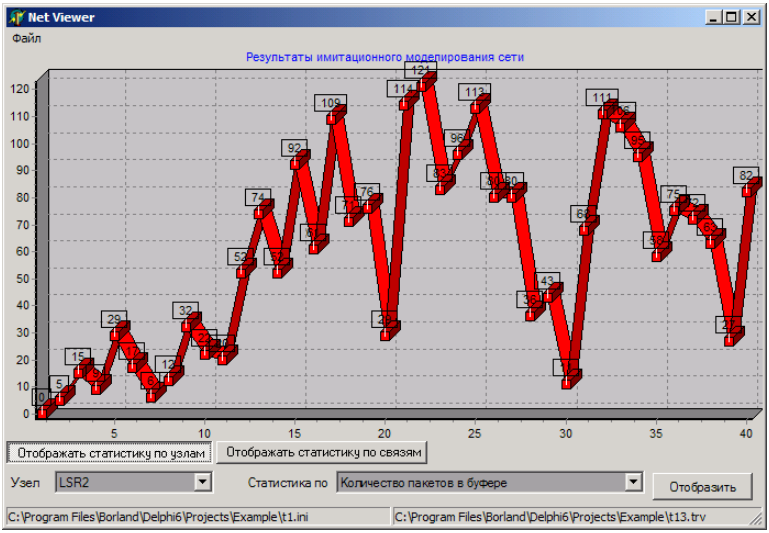


Рис. 3. Завантаженість буфера LSR2

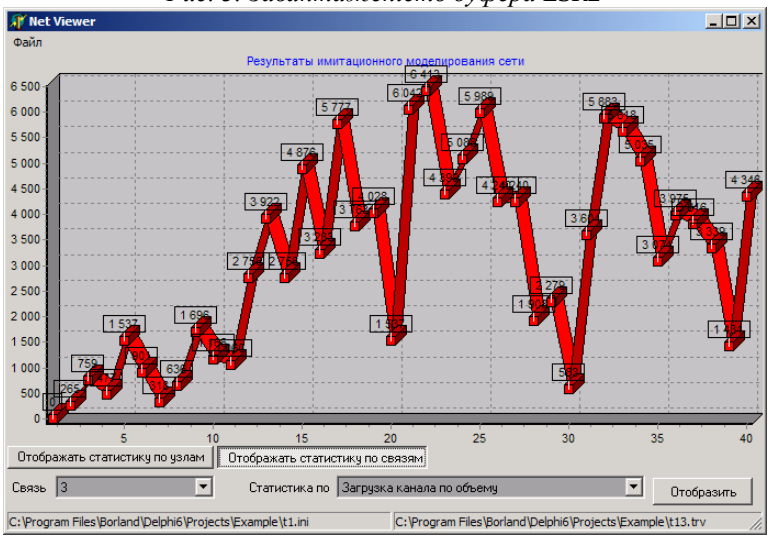


Рис. 4. Завантаженість каналу зв'язку 3

6. Висновки

1. У статті сформульовано задачу розробки програмного комплексу, що дозволяє здійснювати розробку топології комп'ютерних мереж з

технологією MPLS, моделювати поведінку мережі в залежності від налаштувань її складових.

2. Наведено детальний опис імітаційного алгоритму та приклад результатів моделювання.

3. В цілому створена імітаційна модель є зручним засобом для проектувальників спеціалізованих комп'ютерних мереж з технологією MPLS. Вона дозволяє будувати топологію мережі, досліджувати поведінку трафіка різних класів при використанні різних технологій забезпечення якості обслуговування, аналізувати ключові показники якості обслуговування.

Список посилань

1. Гольдштейн А.Б., Гольдшейн Б.С. Технология и протоколы MPLS. СПб.:БХВ – Санкт – Петербург, 2005.-304 с.: ил.
2. Олвейн В. Структура и реализация современной технологии MPLS.: Пер. с англ. – М.: Издательский дом "Вильямс", 2004. – 480 с.: ил. – Парал. тит. англ.
3. О.Ю. Зайченко, Ю.П. Зайченко, Д.Ю. Круглов, «Імітаційне моделювання корпоративної мережі АТМ із різними методами управління трафіками», Наукові вісті НТУУ "КПІ".
4. Шеннон Р. Имитационное моделирование систем – искусство и наука. М.: Мир, 1978.-422 с.

УДК 681.3.06

СТИРЕНКО С.Г.,
МИТИН С.В.

ПАРАЛЛЕЛЬНЫЕ ВЫЧИСЛЕНИЯ ПРИ РАСПОЗНОВАНИИ ИЗОБРАЖЕНИЙ НА ОСНОВЕ КЛЕТОЧНОЙ НЕЙРОННОЙ СЕТИ

Предложен подход для эффективной реализации параллельных вычислений на основе клеточной нейронной сети. Исследование выполнялось с использованием изображений в оттенках серого цвета. Описанный подход подразумевает использование SPMD парадигмы для распараллеливания данных. Процесс распараллеливания осуществлялся при помощи интерфейса передачи сообщений MPI, а производительность оценивалась реализацией на двух кластерных архитектурах.

In this paper a simple but effective approach for parallelization of cellular neural networks for image processing is developed. Digital gray-scale images were used to evaluate the program. The approach uses the SPMD model and is based on the structural data parallel approach. The process of parallelizing the algorithm employs HPF to generate an MPI-based program and the performance behavior was analyzed on two different cluster architectures.

Введение

Клеточная нейронная сеть (*Cellular Neural Network*) (КНС), объединяет в себе черты клеточных автоматов (локальность взаимодействия) и нейронных сетей и является элементарной математической моделью, ячейкой которой является аппаратная или программная модель нейрона. КНС может быть использована для широкого спектра задач, от приложений для сигнальных процессоров (выделение признаков, выявление движения и т.п.) и анализа трехмерный поверхностей (сцен) до решения дифференциальных уравнений в частных производных [1]. За последние годы в этой области появилось достаточно много публикаций [2-4].

В общем КНС может быть представлена как массив идентичных динамичных систем, которые локализуются подключением ячеек. Архитектурно КНС представляет собой объединение двух подходов: массивно-параллельных вычислений, типичных для полносвязных нейронных сетей и локальным взаимодействием между ячейками, характеризующий клеточные автоматы. Любая ячейка имеет связи только с соседними, взаимодействуя непосредственно только с ними. Ячейки не имеющие непосредственного контакта (находящиеся в окрестности) имеют косвенную связь.

Основной идеей представленной параллельной КНС является моделирование подхода решения задач с дискретным временем, которая может быть описана следующими выражениями: [5]

$$x_i(n+1) = \sum_{k \in N_r(j)} A_{jk} y_k(n) + \sum_{k \in N_r(j)} B_{jk} u_k(n) + I_j \quad (1)$$

$$y_i(n) = f[x_j(n)] \quad (2)$$

где x, y, u, I – состояния ячеек: выходные, входные, и значения по диагонали соответственно;

A и B – шаблон обратной связи окрестности и входной шаблон. Выходная функция определяется как $f(x)$, которая может быть функцией знака:

$$f(x) = \begin{cases} 1 \; при \; x \geq 0 \\ 0 \; при \; x < 0 \end{cases} \quad (3)$$

или кусочно-линейная функция выхода:

$$y_i(n) = 0.5*(||x(n)+1|| - ||x(n)-1|| \quad (4)$$

Этот подход довольно удобен для распараллеливания на классических супер-ЭВМ [6]. До сих пор не было сделано какого-либо серьезного анализа для парадигмы SPMD [7] (Single Program Multiple Data) КНС реализаций на кластерных архитектурах, а между тем, она значительно упрощает решение задач обработки изображений.

Механизм распараллеливания

При моделировании КНС будем использовать параллелизм на уровне данных. Этот подход упрощает процесс параллельного нейро-сетевого моделирования и приближает его по сложности к последовательному. Таким образом конечный пользователь имеет возможность более эффективно использовать высокопроизводительные параллельные вычисления работая с нейронными сетями.

Подход предполагает использование парадигмы SPMD, которая сочетает в себе возможности специализированных высокопроизводительных языков для распараллеливания процесса решения задачи. Процесс моделирования системы сводится к разработке последовательной программы, которая манипулирует распределенными данными. Поэтому трудоемкий процесс ручного физического распараллеливания смещается в сторону среды программирования и к компилятору. Этот подход с тем же успехом может быть применен при моделирования на супер-ЭВМ типа MPP и кластерных архитектурах.

Структурный подход параллелизма на уровне данных сводится к четырем стадиям, которые могут использоваться при переходе от последовательного описания программы к параллельной с целью эффективной реализации нейронной сети:

- Последовательное кодирование нейронной сети;
- Определение структуры данных;

- Выделение параллельных участков;
- Автоматизированная генерация параллельного кода и анализ.

С нашей точки зрения КНС можно представить в виде N-мерного регулярного (распознаваемого конечным автоматом) массива элементов. Определены три коррелированных массива: входной *u* (формула 1) и два выходных, один (*x*) для состояния ячеек в автоматное время *n* и второй (*y*) для состояния ячеек в предыдущий момент времени *n*–1. Каждый элемент массива *u* представляется одним пикселем входного изображения.

Придерживаясь SPMD модели сконцентрируем свое внимание на распараллеливании этих трех массивов. Предложим возможные схемы (рис. 1).

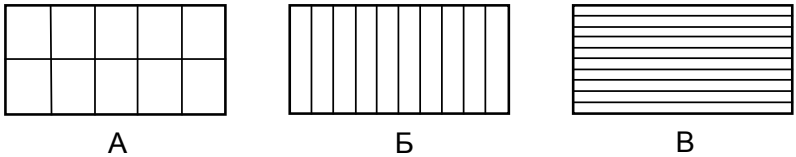


Рис. 1. Возможные схемы распараллеливания при обработки изображений для 10 процессоров: неперекрывающее расположение (А – поблочно, Б – поколоночно, В – построчно)

Исследование эффективности метода

Исследование включало в себя две стадии. На первой оценивались схемы распараллеливания данных изображения (см. рис. 1) и на второй стадии было проанализировано ускорение и масштабируемость лучшей схемы распараллеливания.

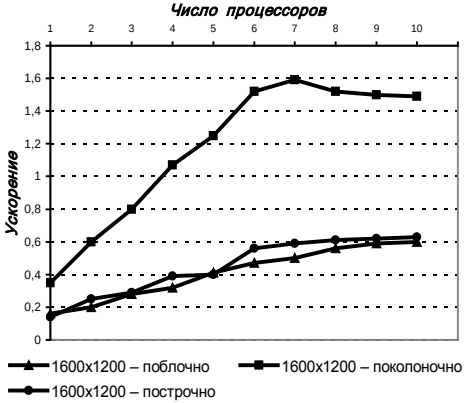
В ходе работы будем опираться на два различных критерия при анализе ускорения. С одной стороны – *абсолютное ускорение,* которое вычисляется сравнением времени выполнения параллельного программного кода на нескольких процессорах с временем выполнения последовательной программы на одном процессоре [8]. Таким образом, получим абсолютное ускорение за счет распараллеливания. С другой стороны – *относительное ускорение,* вычисляя время выполнения параллельной программы, запуская ее на нескольких процессорах с временем выполнения параллельной программы на одном процессоре.

Основой для работы являлся *исследовательский* кластер установленный в национальном техническом университете Украины «КПИ» [9]. Он имеет следующие характеристики: содержит восемьдесят четыре узла, каждый из которых имеет два двухядерных процессора Intel Xeon 5160 с тактовой частотой 3.00 ГГц и 4 ГБ DDR оперативной памяти, частота системной шины – 1333 МГц, коммутационная сеть – Infiniband 4x SDR.

Второй – *лабораторный* кластер был создан на базе научно-исследовательской лаборатории кафедры вычислительной техники НТУУ «КПИ»

[10] и имеет следующую конфигурацию: процессоры – Celeron 2.6 ГГц с 1 ГБ DDR оперативной памяти с частотой системной шины – 800 МГц, коммутационная сеть – Gigabit Ethernet.

Алгоритм КНС был запрограммирован с использованием интерфейса передачи сообщений MPI (реализация MPICH). Были допущены перекрытия размеров блоков на один пиксель с использованием параметра трансляции. Вдоль массива блочных границ, компилятор локализует вычислимую часть, которая, в противном случае, требовала бы передачу данных через коммутационную среду.



Оценка распараллеливания данных (стадия 1)

Оценка различных схем распараллеливания исследовался только на исследовательском кластере. В качестве входных данных использовалось стандартное изображение разрешением 1600×1200 пикселей. На рис. 2. показано абсолютное ускорение для различных схем распараллеливания. Легко заметить, что поколоночная схема распараллеливания имеет наилучшие результаты. С одной стороны это объясняется тем, что реже приходится обращаться к основной памяти, за счет, так называемого, «теплого кэша». Другими словами, более естественного механизма кэширования, т.е. потеря данных в кэш памяти минимальна. С другой стороны – поколоночное распараллеливание данных в соответствии с вычислительным потоком программы, обрабатывающей изображение, является более естественным.

Помимо этого можно увидеть, что издержки, связанные с дополнительной нагрузкой на коммуникационную среду и всю параллельную инфраструктуру в параллельной программе по сравнению с последовательной находится в интервале от 70 до 92 процентов. А также, что в худшем случае, разница между схема распараллеливания превышает 20 процентов. В результате анализа результатов показанных на рис. 2 можно сделать вывод, что распараллеливание нужно увязывать с соответствующей схемой распределения данных, а также с количеством узлов (процессоров) обработки.

Вращение изображения не дает эффекта ускорения обработки, если данные имеют поколоночное распределение. Имеется в виду, что разница во времени обработки различных частей изображения не влияет на время выполнения всей программы, при условии что данные распределены в

соответствии с поколоночной схемой. Хотя две другие схемы распределения данных имеют определенную зависимость времени выполнения программы от подстройки изображения.

Оценка ускорения и масштабируемости (стадия 2)

На второй стадии оценивалось ускорение и масштабируемость для изображений с различной разрешением. В соответствии с результатами первой стадии, использовалась схема только с поколоночным распределением данных.

Для исследования во второй стадии использовались оба кластера. Оценивалась параллельная обработка с использованием КНС для изображений от 400×300 до 4800×3600.

На рис. 3 показана абсолютное ускорение для изображений с разным разрешением. Лабораторный кластер показал ускорение более чем в два раза при использовании более чем трех вычислительных узлов (процессоров) для изображений более чем 800×600 пикселей.

На рис. 4 показано относительное ускорение параллельной программы для изображений с различным разрешением. Изображения с большим разрешением имеют лучшие результаты .

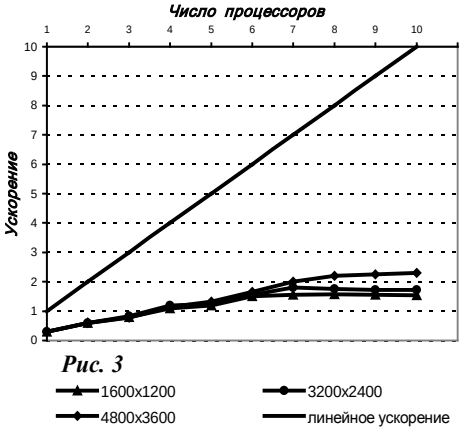


Рис. 3

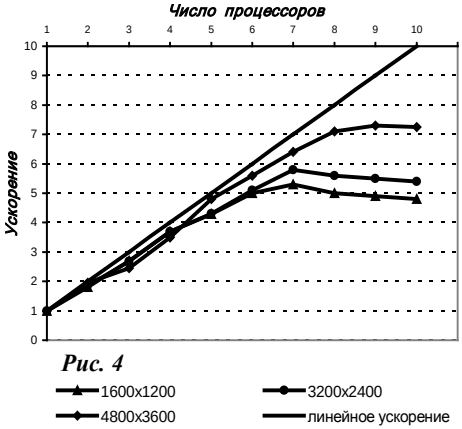


Рис. 4

Для больших изображений достигается более высокое значение ускорения. Однако, наблюдается некий процесс *насыщения*[1] абсолютного ускорения. Для изображений с меньшим разрешением насыщенность достигается раньше и дальнейшее увеличение числа узлов обработки не

[1] Под понятием насыщение понимается процесс прекращения роста параметра ускорения, при дальнейшем повышении количества узлов обработки.

приводит к повышению ускорения, поскольку коммуникационные затраты времени превышают время обработки. Таким образом, параметр ускорение существенно зависит от аппаратной и программной реализации коммуникационной среды (до 70%).

Программа показала довольно хорошую масштабируемость. При обработке КНС легко распараллеливается, тем самым, достигается довольно хорошая балансировка нагрузки между узлами.

Все, до сих пор, представленные результаты получены на лабораторном кластере [10]. Для верификации этих результатов, все оценки выполнялись повторно на более производительном исследовательском кластере [9].

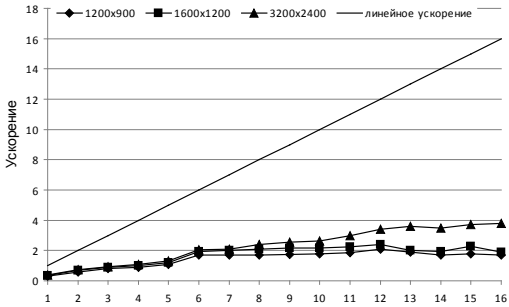


Рис. 5.

Выполнение параллельной программы на исследовательском кластере подтвердило наши предположения. Благодаря более мощным аппаратным средствам и архитектуре программного обеспечения мы получили более высокие значения ускорения (рис. 5).

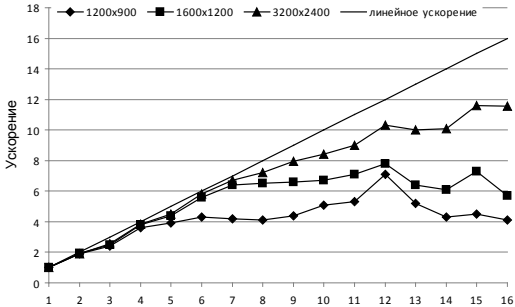


Рис. 6.

На исследовательском кластере использовалось до 16 узлов и полученные результаты сопоставимы с результатами лабораторного кластера (рис. 5 и 6). Здесь также наблюдался процесс насыщения ускорения, который для изображений с небольшим разрешением наблюдался раньше, чем для изображений с бóльшим разрешением.

Очевидны некоторые локальные оптимумы, которые наблюдаемые после 12 и 15 процессорами, в которых данные изображения имеют наилучшее распределение между вычислительными узлами, что приводит к оптимальному балансировке нагрузки. Однако детальный анализ этих оптимумов – тема дальнейшего исследования.

Выводы

Как побочный эффект исследования в работе представлен вариант реализации параллельного моделирования нейронных сетей с помощью

кластерных систем, который может быть использован в качестве базовой архитектуры. С развитием более быстрых и дешевых кластеров и сетевых интерфейсов может быть получено наилучшее соотношение цена/производительность для многих приложений, которые раньше считались прерогативой исключительно суперкомпьютеров, как в нашем случае при моделировании больших клеточных нейронных сетей.

Список использованной литературы

1. T. Roska, J. Vandewalle. Cellular Neural Network. John Wiley and Sons, 1993.
2. E. Schkuta, T. Fuerle, H. Wanec. Strutural data parallel simulation of neural networks. System Research and Info. Systems, 9:149-172, 2000.
3. Руденко О.Г., Бодянский Е.В. Основы теории искусственных нейроных сетей. – Харьков, НТУ «ХПИ», 2002. – 320 с.
4. Li Guo. The application of modified neural network ART1 in the fault diagnosis of air engine [J]. Microcompuer information, 2005(9-1):156-158.
5. V. Cimagalli and M. Balsi. Cellular neural networks: A review. In 6th Italian Workshop on Parallel Architectures and Neural Networks, Vietri sul Mare, May 1993. World Scientific.
6. E. Schikuta. Data parallel software simulation of cellular neural networks. In CNNA’96 - 1996 Fourth IEEE Internationl Workshop on Cellular neural networks and their applications, pages 267-271, 1996.
7. Таненбаум Э. Архитектура компьютера. – М, «Питер», 2005. – с. 581.
8. J. Ortega, R. Voigt. Solution of partial differential equationson vector and parallel computers. Society for Industrial& Applied Mathematics, July 1985.
9. http://www.zn.ua/3000/3100/60893/
10. http://www.comsys.ntu-kpi.kiev.ua

УДК 683.519

ЗАЙЧЕНКО Ю.П.,
МАЛИЛАХ ЕСФАНДИЯРФАРД,
ЗАИКА А.И.

АНАЛИЗ ИНВЕСТИЦИОННОГО ПОРТФЕЛЯ НА ОСНОВЕ ПРОГНОЗИРОВАНИЯ КУРСОВ АКЦИЙ

Работа посвящена исследованию в области портфельной оптимизации, проводимой в расплывчатых информационных условиях. Рассмотрена задача формирования оптимального портфеля акций максимальной доходности при заданном уровне риска на основе метода нечетко-множественной оптимизации. Интервальная оценка доходности каждой акции на следующий период определяется нечетким методом группового учета аргументов по предыстории. Лаги переменных индексов доходности, которые принимают участие в построении модели, определяются при помощи корреляционного анализа.

Представлены результаты экспериментальных исследований предлагаемого подхода к портфельной оптимизации.

The article is devoted to portfolio optimization under fuzzy conditions. The optimal portfolio with maximal profit is considered under given level of risk based on the fuzzy optimization of stock exchange. The interval estimates of shares profitability at the next period is determined by fuzzy group method of data handling. The lags of variables profitability, included in the model are defined by correlation analyses. The results of experimental investigations of the suggested approach to fuzzy portfolio optimization are presented.

1. Введение

В последние годы задачи портфельной оптимизации представляют большой интерес в связи с появлением и развитием рынков ценных бумаг на Украине. Исторически первой работой посвященной оптимизации инвестиционного портфеля явилась работа Г. Марковитца [1], которая стала классической и послужила отправной точкой для дальнейших исследований. Однако существенные недостатки классической постановки в частности допущения о нормальном законе распределения и стационарности процессов, описывающих финансовые ряды, которые на практике не выполняются, потребовали разработки новых подходов к портфельной оптимизации.

Одним из новых направлений в этой области является использование нечетко-множественного подхода к портфельной оптимизации, свободного от недостатков классической модели Марковитца. В работах [1-4] была исследована нечетко-множественная модель оптимизации инвестиционного портфеля на аппарате нечетких множеств. Вместе с этим в этих работах построение интервальных оценок доходности ценных бумаг бази-

руется на ретроспективных данных, которые не всегда соответствуют будущей доходности в момент реализации инвестиционного портфеля.

Поэтому значительного повышения эффективности построенного портфеля в нечетких условиях можно ожидать при использовании адекватного метода прогнозирования будущих доходностей ценных бумаг. Целью настоящей статьи является исследование метода оптимизации инвестиционного портфеля в нечетких условиях на основе прогнозирования курсов акций. В качестве соответствующего метода прогнозирования доходностей акций предлагается использовать нечеткий метод учета аргументов (НМГУА), предложенный и исследованный в работах [6, 7].

Главной особенностью работы является то, что прогнозирование, оптимизация и анализ портфеля производится от начала и до конца без внешнего оценивания (т.е. привлечения экспертов). Не требуются экспертные оценки доходности акций, определение существенных входных переменных. Все этапы происходят только на основании исторической выборки в следующем порядке:

1. С помощью нечеткого метода группового учета аргументов на основе существенных индексов доходности акций строится полиномиальная интервальная модель будущего поведения доходности каждой акции. Предусмотрена возможность адаптации коэффициентов модели при поступлении новых значений без необходимости пересчета всей модели.

2. После получения интервальной оценки доходности акции производится решение задачи нечетко-множественной оптимизации портфеля при заданном уровне риска и критической доходности. В результате находится долевое распределение акций в портфеле и интервальная оценка доходности портфеля в целом.

2. Постановка задачи

Рассматривается фондовый портфель, состоящий из $N = 5$ компонент, акций крупных энергетических компаний: GAZP («ГазПром»), LKON («Лукойл»), EESP (РАО «ЕЭС»), MSNG («МосЭнерго»), TANT («Татнафт») и его поведение на интервале времени $[0, T]$. Данные получены по результатам торгов ценными бумагами компаний, взятых на Московской межбанковской валютной биржи (ММВБ) в период с 1.03.2007 по 30.03.2007.

Каждая из компонент портфеля характеризуется своей финансовой доходностью (оцененной в точке T как относительное приращение цены актива за период). Держатель фондового портфеля – частный вкладчик, инвестиционная компания, взаимный фонд – управляет своими инвестициями, руководствуясь определенными соображениями. С одной стороны, инвестор старается максимизировать свою доходность. С другой сто-

роны, он фиксирует предельно допустимый риск неэффективности своих инвестиций.

Примем капитал инвестора равным 1. Задача оптимизации фондового портфеля заключается в нахождении вектора долевого распределения бумаг в портфеле $x = \{x_i\}$, $i = \overline{1, N}$, который максимизирует доход инвестора при заданном уровне риска (очевидно, что $\sum_{i=1}^{N} x_i = 1$).

Эта задача в свою очередь состоит из подзадач:

1. Нахождение существенных индексов доходности акций, используя корреляционный анализ.
2. Нахождение полиномиальной интервальной модели индексов доходностей акций с применением нечеткого метода группового учета аргументов.
3. Нахождение долевого распределения акций в портфеле и интервальной оценки его доходности, используя нечетко-множественный метод оптимизации фондового портфеля.

3. Построение нечеткой модели прогнозирования индексов доходности акций с использованием нечеткого МГУА

Постановка задачи

Заданы входные временные процессы $x_1(t), x_2(t), ..., x_n(t)$ – котировки курсов ценных бумаг (ЦБ) на фондовом рынке в дискретные моменты времени $t = 0,1,2,...,T$. Требуется построить прогнозирующую модель для оценки котировок ЦБ в следующий период времени, то есть найти $y_i(t) = x_i(t+1)$. Учитывая нестационарность финансовых процессов, а также сложные неизвестные взаимозависимости между входными переменными и выходной $y_i(t) = x_i(t+1)$ предлагается для этих целей использовать нечеткий метод группового учета аргументов (НМГУА), предложенный в [6, 7]. Этот метод использует основные идеи и принципы классического МГУА и позволяет синтезировать прогнозирующую модель по обучающей выборке при минимальном участии человека – ЛПР. Достоинством МГУА является то, что не требуется задавать априори структуру модели, алгоритм строит её сам в процессе работы, то есть метод решает задачу структурной идентификации при самых общих предложениях относительно класса моделей.

Описание алгоритма НМГУА

Выбор общего вида модели, которым будет описываться искомая зависимость. Из множества выходов $X = \{X_1, X_2, ..., X_n\}$ выбираются пары аргументов X_i и X_j и составляются частичные описания вида $Y_k^{(1)} = \varphi(X_i, X_j)$, $i \neq j$, $i, j = \overline{1, N}$ при этом используют квадратичные частичные описания (либо линейные):

$$Y_k^{(1)} = a_0 + a_i X_i + a_j X_j + a_{ij} X_i X_j + a_{ii} X_i^2 + a_{jj} X_j^2 .$$

Число частичных описаний 1-го ряда равно $M = n(n-1)/2$.

Коэффициенты – интервалы, которые можно задать треугольными нечеткими числами и записать в виде центра α_i и ширины c_i: $a_i = (\alpha_i, c_i)$.

2. Поиск наилучшей модели осуществляется таким образом:

- Вся выборка делится на обучающую и проверочную: $N_{выб} = N_{уч} + N_{пров}$.
- На обучающей выборке $N_{уч}$ определяются значения a_0, a_i, a_{ij}, используя симплекс метод.
- Конструируется модель вида:

$$\min\left(C_0 \cdot M + C_1 \sum_{k=1}^{M} \left|x_{ki}\right| + C_2 \sum_{k=1}^{M} \left|x_{kj}\right| + C_3 \sum_{k=1}^{M} \left|x_{ki} \cdot x_{kj}\right| + C_4 \sum_{k=1}^{M} \left|x_{ki}^2\right| + C_5 \sum_{k=1}^{M} \left|x_{kj}^2\right|\right) \quad (1)$$

при условиях:

$$\alpha_0 + \alpha_1 x_{ki} + \alpha_2 x_{kj} + \alpha_3 x_{ki} x_{kj} + \alpha_4 x_{ki}^2 + \alpha_5 x_{kj}^2 - (C_0 + C_1 \left|x_{ki}\right| + C_2 \left|x_{kj}\right| + C_3 \left|x_{ki} \cdot x_{kj}\right| + C_4 \left|x_{ki}^2\right| + C_5 \left|x_{kj}^2\right|) \le y_k$$

$$\alpha_0 + \alpha_1 x_{ki} + \alpha_2 x_{kj} + \alpha_3 x_{ki} x_{kj} + \alpha_4 x_{ki}^2 + \alpha_5 x_{kj}^2 + (C_0 + C_1 \left|x_{ki}\right| + C_2 \left|x_{kj}\right| + C_3 \left|x_{ki} \cdot x_{kj}\right| + C_4 \left|x_{ki}^2\right| + C_5 \left|x_{kj}^2\right|) \ge y_k \quad (2)$$

$$k = \overline{1, M}, \quad C_p \ge 0, \quad p = \overline{0,5},$$

где *к* – номер измерения, данные из которого используются; M – объем выборки.

Задача состоит в том, чтобы минимизировать суммарную ширину интервала для выходных значений Y за счет нахождения таких значений ширины интервалов искомых коэффициентов c_i и таких значений центров интервалов α_i, $i=\overline{1,C_{r+1}^2}$, которые бы обеспечивали минимальное рассеивание величины Y одновременно с выполнением условия, что измеряемые значения искомой величины находятся в этом интервале – условие (2). Данная задача (1), (2) является задачей линейного программирования.

3. По критерию минимума $\overline{\varepsilon}^2$ на проверочной последовательности выбирается F лучших моделей, что реализует процедуру селекции. Величина F называется *свободой выбора*, при этом $F < C_{r+1}^2$. Выходы этих моделей служат аргументами-входами для конструирования моделей следующего ряда. Для отбора лучших моделей используются следующие критерии:

а) *критерий регулярности* (точности):

$$\overline{\varepsilon}_{пр}^2: \quad \overline{\varepsilon^2} = \frac{1}{N_{пр}} \cdot \sum_{i=1}^{N_{пр}} (y_i - y_i^*)^2 \quad .(3)$$

б) критерий несмещенности:

$$n_{см} = \frac{1}{N} \sum_{i=1}^{R} \left(y_i^* - y_i^{**} \right)^2$$

, при этом, чем меньше $n_{см}$, тем более несмещенной является модель.

4. Находится $\overline{\varepsilon}^2(m) = \min\limits_{k} \overline{\varepsilon}^2(m)$. Проверяется условие $\overline{\varepsilon}^2(m) > \overline{\varepsilon}^2(m-1)$, где $\overline{\varepsilon}^2(m)$, $\overline{\varepsilon}^2(m-1)$ – величины критерия точности для лучших моделей m-го ряда селекции соответственно. Если да, то конец. Искомая модель выбирается из частичных описаний $(m-1)$-го уровня, на котором достигается минимальная ошибка $\overline{\varepsilon}^2(m-1)$.

Иначе переход к конструированию следующего ряда частичных описаний. При этом проводится отбор (селекция) *F* лучших описаний.

5. Заключительный этап.

Из F лучших моделей по критерию регулярности выбираем лучшую модель. Восстанавливаем аналитический вид лучшей модели, используя геделевскую нумерацию Двигаясь от конца до начала и делая последовательную замену переменных, вычисляются выражения для искомой модели в исходном пространстве описаний.

Нечетко-множественный метод оптимизации фондового портфеля

Пусть имеется фондовой портфель из N активов на интервале $[0,T]$. Прогнозное поведение каждой из компонент портфеля $i = 1,\ldots,N$ на момент T характеризуется свой финальной расчетной доходностью r_i (оцененной в точке T как относительное приращение цены актива за период). Поскольку доход по ЦБ случаен, его точное значение в будущем неизвестно, а вероятностное описание такого сорта случайности не вполне корректно, то в качестве описания доходности уместно использовать треугольные нечеткие числа. Таким образом, для i-ой ценной бумаги имеем:

$\overline{r_i}$ – ожидаемая доходность по i-ой ценной бумаги;

r_{i1} – нижняя граница доходности i-ой ценной бумаги;

r_{i2} – верхняя граница доходности i-ой ценной бумаги.

$r_i = \left(r_{1i}, \overline{r_i}, r_{2i}\right)$ – доходность по i-ой ценной бумаге, треугольное нечеткое число.

Тогда доходность по портфелю $r = (r_{\min} = \sum_{i=1}^{N} x_i r_{1i}; \overline{r} = \sum_{i=1}^{N} x_i \overline{r_i}; r_{\max} = \sum_{i=1}^{N} x_i r_{2i})$ также является треугольным нечетким числом (как линейная комбинация треугольных нечетких чисел).

Задача оптимизации имеет вид [2-4]:

$$\tilde{r} = \sum_{i=1}^{N} x_i \cdot \tilde{r}_i \to \max,$$

$$\frac{1}{\sum_{i=1}^{N} x_i r_{i2} - \sum_{i=1}^{N} x_i r_{i1}} \left(\left(r^* - \sum_{i=1}^{N} x_i r_{i1} \right) + \left(\sum_{i=1}^{N} x_i \tilde{r}_i - r^* \right) \cdot \ln \left(\frac{\sum_{i=1}^{N} x_i \tilde{r}_i - r^*}{\sum_{i=1}^{N} x_i \tilde{r}_i - \sum_{i=1}^{N} x_i r_{i1}} \right) \right) = \beta$$

$$\sum_{i=1}^{N} x_i = 1, \; x_i \geq 0, \quad i = \overline{1, N},$$

где β – степень риска портфеля, x_i – вес i-го актива в портфеле, r^* – критическое значение доходности. Задача может решаться с помощью R – алгоритма оптимизации [4,5].

5. Построение оптимального инвестиционного портфеля по результатам торгов и его анализ

В качестве исходных данных использовались результаты, итоги торгов на Московской межбанковской валютной бирже (ММВБ), фрагмент которых приводится в таблице 1. На основе этих данных были рассчитаны индексы доходности акций (в %) от текущей цены акций. Результаты приводятся в таблице 2.

Табл. 1. Итоги торгов по ценным бумагам компании

Дата/Акции	GAZP	LKOH	EESR	MSNG	TATN
01.03.2007	260,4	2047,71	30,54	6,155	112,71
02.03.2007	257,88	2031,66	29,933	6,05	110,48
05.03.2007	245,03	1953,66	28,281	5,641	102,56
06.03.2007	254,28	2034,8	29,0507	5,947	105,66
07.03.2007	253,1	2046,26	28,97	6,162	106,75
09.03.2007	263,68	2101,08	29,666	6,26	110,75
12.03.2007	263,77	2081,31	31,395	6,337	112,41
13.03.2007	260,78	2075,97	32,458	6,276	11,77
14.03.2007	249,8	2002,34	30,768	5,894	105,62
15.03.2007	255,52	2045,45	31,616	6,089	11,54
16.03.2007	254,83	2041,39	31,89	6,141	114,63
19.03.2007	256,56	2057,14	32,888	6,213	115
20.03.2007	254,48	2063,41	33,558	6,258	113,37
21.03.2007	253,38	2077,53	34,191	6,522	114,18
22.03.2007	258,91	2125,4	35,38	6,568	117,15
23.03.2007	264,53	2163,93	35,205	6,501	119,28
26.03.2007	272,8	2194,52	35,062	6,489	122,45
27.03.2007	270,32	2176,29	34,408	6,385	121,22
28.03.2007	271,81	2203,67	35,069	6,429	123,26

Дата/Акции	GAZP	LKOH	EESR	MSNG	TATN
29.03.2007	272,67	2234,55	35,751	6,565	124,24
30.03.2007	274,2	2258,55	35,546	6,638	124,22

Табл. 2.

Дата /Акции	GAZP	LKOH	EESR	MSNG	TATN
1	-0,009677	-0,007838	-0,019876	-0,017059	-0,019785
2	-0,049829	-0,038392	-0,05519	-0,067603	-0,071687
3	0,03775	0,041532	0,027439	0,054246	0,030226
4	-0,004641	0,005632	-0,002994	0,036153	0,010316
5	0,041802	0,02679	0,024025	0,015904	0,037471
6	0,000341	-0,009409	0,058282	0,0123	0,014989
7	-0,011336	-0,002566	0,033859	-0,009626	-0,005693
8	-0,042104	-0,035468	-0,052067	-0,060867	-0,055024
9	0,022898	0,02153	0,027561	0,033084	0,05605
10	-0,0027	-0,001985	0,008666	0,00854	0,027703
11	0,006789	0,007715	0,031295	0,011724	0,003228
12	-0,008107	0,003048	0,020372	0,007243	-0,014174
13	-0,004323	0,006843	0,018863	0,042186	0,0071145
14	0,021825	0,023042	0,034775	0,007053	0,026014
15	0,021706	0,018128	-0,004946	-0,010201	0,018182
16	0,031263	0,014136	-0,004062	-0,001846	0,026576
17	-0,009091	-0,008307	-0,018653	-0,016027	-0,010045
18	0,005512	0,012581	0,019211	0,006891	0,016829
19	0,003164	0,014013	0,019447	0,021154	0,007951
20	0,005611	0,01074	-0,005734	0,01112	-0,000161

Используем корреляционный анализ временных рядов $x_i(t)$ и определим оптимальные лаги по каждой входной переменной. Так, например при прогнозировании цены акций GAZP лаги для входных переменных были таковы (см.таблицу 3).

Табл. 3.

Переменная	Лаг	Корреляция
LKOH	2	0,330041858
TATN	5	0,251828399
MSNG	2	0,210079481
EESR	5	0,127865395
GAZP	2	0,127548329

6. Построение полиномиальной модели индекса доходности акции

Используя результаты корреляционного анализа, а также метод НМГУА, были получены следующие модели, синтезированные на 12 точках из 15. Остальные 3 точки в каждом эксперименте использовались для прогнозирования. Например, для GAZP были получены следующие модели:

- верхняя граница интервала:

GAZP+=0,0174+0.4342*LIKOH(-2)^1+12,0603*LKOH(-2)^1*TATN(-5)^1+
+0,1076*MSNG(-2)^1+3,4033*LKOH(-2)^2+189,0276*LKOH(-2)^2*TATN
(-5)^1+2624,7284*LKOH(-2)^2*TATN(-5)^2

- центр интервала :

GAZP=0,0021+0.4342 *LIKOH(-2)^1+12,0355*LKOH(-2)^1*TATN(-5)^1+
+0,1076*MSNG(-2)^1+3,4033*LKOH(-2)^2+189,0276*LKOH(-2)^2*TATN
(-5)^1+2624,7284*LKOH(-2)^2*TATN(-5)^2

- нижняя граница интервала:

GAZP-=0,0131+0.4342*LIKOH(-2)^1+12,0603*LKOH(-2)^1*TATN(-5)^1+
+0,1076*MSNG(-2)^1+3,4033*LKOH(-2)^2+189,0276*LKOH(-2)^2*TATN
(-5)^1+2624,7284*LKOH(-2)^2*TATN(-5)^2

Результаты прогнозирования для трех точек приводятся в таблице 4, а итоговые в таблице 5.

Табл. 4.

	GAZP-	GAZP=	GAZP+	GAZP (реальное значение)
1-я точка, попала	-0,005042	0,010273	0,025589	0,005512
2-я точка, попала	-0,020419	-0,005103	0,010212	0,003164
3-я точка, попала	-0,002993	0,012322	0,027638	0,005611
	LKOH-	LKOH=	LKOH+	LKOH (реальное значение)
1-я точка, попала	-0,005027	0,008949	0,022925	0,012581
2-я точка, попала	-0,003154	0,011394	0,025943	0,014013
3-я точка, попала	-0,003862	0,009448	0,022759	0,01074
	EESR-	EESR=	EESR+	EESR (реальное значение)
1-я точка, попала	-0,013872	0,015114	0,044101	0,019211
2-я точка, попала	-0,013749	0,013238	0,040225	0,019447
3-я точка, не попала	-0,002977	0,009009	0,020996	-0,005734
	MSNG-	MSNG=	MSNG+	MSNG (реальное значение)
1-я точка, попала	0,002955	0,008155	0,013355	0,006891
2-я точка, попала	0,007468	0,014598	0,021728	0,021154

3-я точка, попала	0,002158	0,008448	0,014738	0,01112
	TATN-	TATN=	TATN+	TATN (реальное значение)
1-я точка, попала	0,004206	0,01975	0,035295	0,0168229
2-я точка, попала	-0,007756	0,007788	0,023333	0,007951
3-я точка, попала	-0,004858	0,007686	0,020231	-0,000161

Табл. 5

Акция	СО	СКО
LKOH	0,00458	0,004934
TATN	0,00218	0,002326
MSNG	0,006135	0,006275
EESR	0,003978	0,004108
GAZP	0,003714	0,00382

Здесь, СО – средняя ошибка прогноза, СКО – средне-квадратическое отклонение.

Как видно из результатов приведенных выше, в большинстве случаев действительное значение индекса доходности акции попадало в прогнозный интервал.

Существенно повысить точность помогла адаптация коэффициентов при поступлении новых значений.

Результаты использования нечетко-множественного метода оптимизации портфеля

Завершающий этап в нахождении долевого распределения акций в портфеле на основе интервальной оценки доходности каждой акции, выполняется с помощью нечетко-множественного метода оптимизации портфеля.

Например, для 1-й точки были получены следующие результаты, приведенные в таблице 6.

В данном случае для действительных значений доходности:

GAZP = 0,005512 EESR = 0,019211

MSNG = 0,006891 TANT = 0,016829

при найденном долевом распределении акций в портфеле:

$$X_{GAZP} = 0{,}101 \qquad X_{EESR} = 0{,}135$$

$$X_{MSNG} = 0{,}122 \qquad X_{TATN} = 0{,}642$$

доходность реального портфеля составила 0,014795.

Как видим, доходность реального портфеля попала в оценочный интервал доходности (см. таблицу 6).

Табл. 6

	Нижняя граница	Ожидаемая доходность	Верхняя граница	Доля
LKOH	-0,005042	0,010273	0,025589	0,101
TATN	-0,005027	0,008949	0,022925	0
MSNG	-0,013872	0,015114	0,044101	0,135
EESR	0,002955	0,008155	0,013355	0,122
GAZP	0,004206	0,01975	0,035295	0,642
Портфель	0,000678	0,016752	0,032826	1

Выводы

1. Использование данного метода нечетко-множественной оптимизации имеет разумное обоснование в сравнении с четкой задачей Марковитца, так как риск портфеля – это не его волатильность, а возможность того, что ожидаемая доходность портфеля окажется ниже некоторой предустановленной плановой величины.

2. В нечетко-множественном методе под риском понимается ситуация, когда ожидаемая доходность портфеля ниже заданного критического уровня. Со снижением ожидаемой доходности возрастает риск того, что доход от портфельных инвестиций окажется меньше критического значения. В модели Марковитца риск рассматривается как степень колебания ожидаемого дохода по портфелю, причем как в меньшую, так и в большую сторону, что противоречит здравому смыслу.

3. Использование нечеткого МГУА для прогнозирования доходностей акций в следующий период дает достаточно надежную основу для оптимизации инвестиционного портфеля в нечетких условиях. Эксперту достаточно рассчитать значении коридора, в котором ожидаемо колеблется будущий доход по ценным бумагам при помощи НМГУА.

Список использованной литературы

4. Недосекин А.О. Система оптимизации фондового портфеля от Сименс Бизнес Сервисез // Банковские технологии. – 2003. – № 5. – Также на сайте: http: //www. finansy. ru/publ/fin/004. htm
5. Недосекин А.О. Применение теории нечетких множеств к задачам управления финансами. Раздел 3 // Аудит и финансовый анализ. – 2000. – №2. - Также на сайте: http://www.cfin.ru/press/afa/2000-2/08-3.shtml
6. Недосекин А.О. Монотонные портфели и их оптимизация // Аудит и финансовый анализ. – 2002. – №2. – Также на сайте: http://sedok.narod.ru/s_files/PF_Article_4.zip
7. Зайченко Ю.П., Малихех Есфандиярфард. Анализ и сравнение результатов оптимизации инвестиционного портфеля при применении модели Марковица и нечетко-множественного метода // XIII-th International Conference KDS-2007. SOFIA, 2007.-Vol.1, pp.278-286

8. Зайченко Ю. П., Малихех Есфандиярфард. Оптимизация инвестиционного портфеля с использованием аппарата нечетких множеств. Труды Винницкого национального технического университета.-2006.-Вып 12
9. Зайченко Ю. П., Заєць *І. О*. Синтез і адаптація нечітких прогнозуючих моделей на основі методу самоорганізації. // Наукові вісті НТУУ "КПІ". – 2001. – №3. – с. 34 – 41.
10. Ю.П. Зайченко та І.О. Заєць. Порівняльний аналіз алгоритмів МГУА з використанням різних методів покрокової адаптації коефіцієнтів //Вісник Національного технічного університету України, сер. Інформатика, управління та обчислювальна техніка.-2005, № 43.- с. 167-180.

УДК 681.513

ПІНЧУК Д.А.

НЕЧІТКИЙ МЕТОД ІНДУКТИВНОГО МОДЕЛЮВАННЯ З НЕЧІТКИМИ ВХІДНИМИ ЗМІННИМИ

Представлен нечеткий метод группового учета аргументов с нечеткими входами.

Fuzzy group method of data handling with fuzzy inputs is presented.

Вступ

Сьогодні прийняття кардинальних управлінських рішень – дуже відповідальне та має базуватися на достовірній інформації. Часто осягнути всю повноту доступної інформації одній людині не під силу. Тоді на допомогу їй приходять системи підтримки прийняття рішень, які базуються на математичних методах моделювання та прогнозування. Над написанням алгоритмів обробки інформації працює безліч вчених багатьох країн.

Дуже цінними у площині вирішення цих задач є результати розробки методу групового урахування аргументів (МГУА) [1,2] та нечіткого методу індуктивного моделювання Інститутом прикладного системного аналізу (ІПСА) [3]. Ці методи дозволяють будувати модель досліджуваного об’єкту використовуючи відносно малу кількість вхідної інформації.

Метод групового урахування аргументів у [1,2,3] розрахований на чіткі вхідні змінні. Однак дуже багато реальних задач сьогодні мають на вході нечітку інформацію.

Дана робота є результатом вирішення проблеми побудови моделі об’єкту, що описується нечіткими параметрами. Кінцевою метою досліджень у даному напрямку є отримання моделі об’єкту для прогнозування його поведінки у різних умовах, що описуються нечіткими параметрами.

Постановка задачі

Об’єкт, що досліджується, описується множиною параметрів $X = \left\{ x_i \middle| i = \overrightarrow{1, n}; y \right\}$. Де x_i – вхідні параметри, y – вихідний параметр.

Маємо множину вимірів параметрів $\Omega = \left\{ X^{(k)} \middle| k = \overrightarrow{1, N} \right\}$.

Значення (кількісне або якісне вираження) параметрів носить оціночний характер – є нечіткою величиною. Це може бути інтервал; впорядкована чи невпорядкована лінгвістична змінна; величини, параметри яких можуть бути виражені їх статистичними оцінками; нечіткі числа з

різними видами функції належності, які представлені параметрами цих розподілів.

Необхідно побудувати модель об'єкту $y = f(x_i | i = \overrightarrow{1, n})$.

Вихідним положенням будемо вважати повну відсутність уяви про структуру моделі та її належність до певного класу. Перелік параметрів, який визначено, може бути неповним та виключати базові параметри. В перелік можуть бути включені залежні між собою параметри.

Обґрунтування вибору напряму досліджень

За умов невідомої структури моделі потрібно використовувати метод що володіє алгоритмом самоорганізації. Одним з таких методів є Метод групового урахування аргументів. Для гарантованої побудови моделі зручно використовувати Нечіткий метод групового урахування аргументів. Від дозволяє побудувати модель навіть тоді, коли матриця вхідних параметрів є виродженою.

Однак обидва методи працюють лише з чіткими вхідними даними. При застосуванні цих методів, нечітка інформація, що міститься у вхідних параметрах, втрачається на етапі дефазифікації.

У статті [4] було модифіковано Метод групового урахування аргументів для використання нечіткої вхідної інформації. Перевагами такої модифікації стало зменшення ширини інтервалу прогнозу та підвищення коефіцієнту кореляції моделі з реальними даними. Недоліком методу стало підвищення ширини інтервалу на окремих прогнозованих точках.

Сподіваючись усунути недолік Методу групового урахування аргументів з нечіткими входами було вирішено модифікувати Нечіткий метод групового урахування аргументів для використання нечіткої інформації на вході.

Нечіткий метод групового урахування аргументів з нечіткими входами

Основна ідея методу

Маємо об'єкт, що характеризується множиною вхідних та вихідних параметрів $X = \{x_i | i = \overrightarrow{1, n}; y\}$. Де x_i – вхідні параметри, y – вихідний параметр та N спостережень $\Omega = \{X^{(k)} | k = \overrightarrow{1, N}\}$.

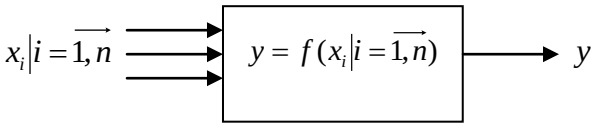


Рис. 1.

За результатами спостережень необхідно визначити $f(x_i | i = \overrightarrow{1,n})$. Структура моделі $f(x_i | i = \overrightarrow{1,n})$ невідома.

Параметри $x_i | i = \overrightarrow{1,n}, y$ – нечіткі величини, кількісне або якісне вираження яких носить оціночний характер. Це можуть бути інтервали; впорядковані чи невпорядковані лінгвістичні змінні; випадкові величини, виражені їх статистичними оцінками; нечіткі числа, представлені видами функцій належності та їх параметрами.

Найбільш повна залежність між входами $x_i | i = \overrightarrow{1,n}$ та виходом y може бути представлена за допомогою узагальненого поліному Колмогорова-Габора:

$$y = a_0 + \sum_{i=1}^{n} a_i x_i + \sum_{i=1}^{n} \sum_{i \le j} a_{ij} x_i x_j + \sum_{i=1}^{n} \sum_{i \le j} \sum_{j \le k} a_{ijk} x_i x_j x_k + \ldots \quad (1),$$

де коефіцієнти $a, a_i, a_{ij}, a_{ijk}, \ldots$ невідомі.

Нормування вхідних даних

Нормування вхідних даних проводиться зведенням всіх вхідних даних до інтервалів.

Зведення нечітких вхідних даних до інтервального виду проводиться за допомогою взяття інтервалу α-рівня від функцій належності нечітких величин.

Дійсні числа

Замінюються нечітким інтервалом шириною 0. Взяття α-рівня від такого числа завжди буде інтервалом шириною 0.

Нечіткі числа

Описуються своєю функцією належності та її параметрами. Взяття α – рівня від нечіткого числа буде інтервалом, який буде визначатися точками перетину лінії α-рівня з функцією належності.

Лінгвістичні змінні

Впорядковані лінгвістичні змінні, з *N* можливими значеннями p = 1,2K *N* – замінюються нечіткими числами з трикутною функцією належності, що має параметри *A* – центр та *C* – ширина інтервалу. При чому: *A* = *p*, *C* = 2.

Потім значення всіх інтервалів нормуються до інтервалу $[0.2;0.8]$.

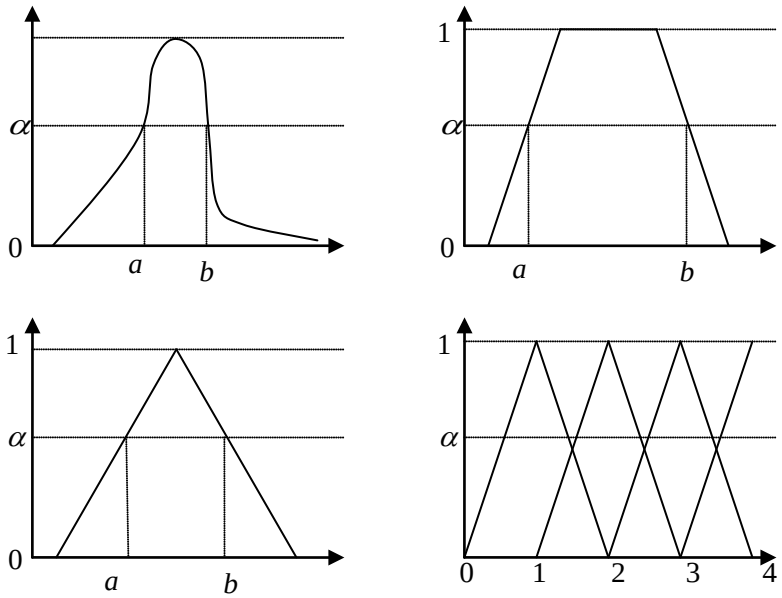


Рис. 2.

Основний алгоритм методу

Для побудови моделі метод використовує ідею самоорганізації та механізми живої природи – схрещування (гібридизацію) та селекцію (відбір).

Вся вибірка наявних даних (довжиною N) ділиться на навчальну та перевірочну вибірки: $N_{навч}$ та $N_{перев}$.

На першому етапі, використовуючи навчальну вибірку, будуються прості моделі-претенденти $y1_s = f(x_i, x_j)$, $i \neq j$, $i, j = \overrightarrow{1,n}$, $s = \overrightarrow{1, C_n^2}$.

Серед них обираються F найкращих моделей за критерієм регулярності, який базується на умові найменшого середньоквадратичного відхилення границь інтервалу, що видає модель $\lfloor \underline{y}_s ; \overline{y}_s \rfloor$ від реального значення нормованого інтервалу $\lfloor \underline{y}; \overline{y} \rfloor$ на перевірочній вибірці $N_{перев}$:

$$\varepsilon_s{}^2 = \frac{1}{N_{перев}} \sum_{k=1}^{N_{перев}} \left(\left(\underline{y}^{(k)} - \underline{y}_s^{(k)} \right)^2 + \left(\overline{y}^{(k)} - \overline{y}_s^{(k)} \right)^2 \right) \quad (2),$$

де s – номер часткового опису, k – номер перевірочної точки; $\underline{y}_s^{(k)}$, $\overline{y}_s^{(k)}$ – вихід s-го часткового опису на k-ій перевірочній точці.

Найкращі F моделей передаються на вхід другого етапу побудови моделі: $y2_s = f(y1_i, y1_j)$, $i \neq j$, $i, j = \overrightarrow{1,F}$, $s = \overrightarrow{1,C_F^2}$.

Після чого повторюємо процедуру побудови моделей-претендентів.

Після кожного етапу побудови моделей-претендентів та відбору найкращих, перевіряємо зменшення середнього критерію незміщеності для кращих моделей даного етапу складності r:

$$n(r)_{сер}^{незм} = \frac{1}{F}\sum_{i=1}^{F} n(r)_i^{незм} \quad (3).$$

В основі критерію незміщеності є той дуже важливий факт, що для одного й того ж об'єкту дослідження, по різних вибірках даних, при інших рівних умовах, повинні бути отримані близькі моделі, що описують поведінку об'єкта. Критерій має наступний вигляд:

$$n_i^{незм} = \frac{1}{R_1 + R_2} \cdot \sum_{k=1}^{R_1+R_2} \left(\left(\underline{y}_{i_1}^{(k)} - \underline{y}_{i_2}^{(k)} \right)^2 + \left(\overline{y}_{i_1}^{(k)} - \overline{y}_{i_2}^{(k)} \right)^2 \right) \quad (4)$$

Тут R_1, R_2 – дві довільно обрані однакові за розмірами вибірки даних, що не перетинаються; $\underline{y}_{i_1}^{(k)}, \overline{y}_{i_1}^{(k)}, \underline{y}_{i_2}^{(k)}, \overline{y}_{i_2}^{(k)}$ – оцінки значень отримані на коефіцієнтах, що вирахувані на моделях, отриманих відповідно на вибірці R_1 та R_2.

Якщо $n(r)_{сер}^{незм} > n(r-1)_{сер}^{незм}$, або $\left| n(r)_{сер}^{незм} - n(r-1)_{сер}^{незм} \right| \leq \varepsilon$, де ε – заздалегідь прийнята мала величина точності, то це означає, що ми досягнули оптимального рівня складності моделі на $r-1$ етапі. У цьому випадку потрібно за модель об'єкту прийняти ту модель $r-1$ етапу, яка забезпечує мінімальне значення критерію регулярності.

Якщо $n(r-1)_{сер}^{незм} - n(r)_{сер}^{незм} > \varepsilon$, то необхідно продовжити будувати моделі-претенденти наступного етапу.

Алгоритм побудови моделі-претенденту

Модель-претендент будується з використанням поняття лінійної інтервальної моделі:

$$Y = A_0 + A_1 z_1 + \ldots + A_m z_m; \quad (5)$$

змінні $z_1 \ldots z_m$ – нечіткі числа інтервального вигляду, що описуються парою параметрів $z_i = \left[\underline{z}_i, \overline{z}_i \right]$.

A_0 – нечітке число інтервального вигляду, що описуються парою параметрів $A_0 = (a_0, c_0)$, $c_0 > 0$. a_0 – центр інтервалу, c_0 – його ширина;

A_i – нечіткі числа інтервального вигляду, що описуються парою параметрів $A_i = \left[\underline{a}_i, \overline{a}_i \right]$.

Тоді Y – нечітке інтервальне число, параметри якого визначаються наступним чином:

Ліва границя інтервалу: $\underline{y} = \sum_{i=1}^{m} \hat{a}_i \underline{z}_i + \sum_{i=1}^{m} \underline{a}_i \hat{z}_i + a_0 - \frac{c_0}{2}$; (6)

Права границя інтервалу: $\overline{y} = \sum_{i=1}^{m} \hat{a}_i \overline{z}_i + \sum_{i=1}^{m} \overline{a}_i \hat{z}_i + a_0 + \frac{c_0}{2}$ (7),

де $\hat{a}_i = \frac{\underline{a}_i + \overline{a}_i}{2}$, $z_i = \frac{\underline{z}_i + \overline{z}_i}{2}$.

Для того, щоб інтервальна модель (5) була коректна, необхідно, щоб справжнє значення залежної величини навчальної вибірки $Y_k = \left(\underline{y}^{(k)}, \overline{y}^{(k)} \right)$ належало інтервалу, що визначається формулами (6), (7):

$$\begin{cases} \sum_{i=1}^{m} \hat{a}_i \underline{z}_i^{(k)} + \sum_{i=1}^{m} \underline{a}_i \hat{z}_i^{(k)} + a_0 - \frac{c_0}{2} \le \underline{y}^{(k)} \\ \sum_{i=1}^{m} \hat{a}_i \overline{z}_i^{(k)} + \sum_{i=1}^{m} \overline{a}_i \hat{z}_i^{(k)} + a_0 + \frac{c_0}{2} \ge \overline{y}^{(k)} \end{cases} \quad k = \overrightarrow{1, M} \quad (8)$$

При побудові моделі оптимальної складності будемо розглядати часткові описи наступного вигляду:

$$A_0 + A_1 x_1 + A_2 x_2 + A_3 x_1 x_2 + A_4 x_1^2 + A_5 x_2^2 \quad (9)$$

Через це, змінні z_i з формул (5)–(8) записуються наступним чином:

$$z_1 = x_1, z_2 = x_2, z_3 = x_1 x_2, z_4 = x_1^2, z_5 = x_2^2 \quad (10)$$

де $z_i = x_j x_k$ – нечітке число інтервального вигляду ліва границя інтервалу якого $\underline{z}_i = \hat{x}_j \underline{x}_k + \underline{x}_j \hat{x}_k$, права $\overline{z}_i = \hat{x}_j \overline{x}_k + \overline{x}_j \hat{x}_k$.

Метод оцінювання параметрів лінійної інтервальної моделі

Нехай маємо M вимірів (n+1)-ї змінної, при чому перші n – незалежні вхідні змінні, а (n+1)-ша залежить від них, при чому характер залежності невідомий. Оцінювання параметрів лінійної інтервальної моделі для часткового опису зводиться до знаходження параметрів $\overline{a}_i, \underline{a}_i$, a_0 та c_0, таких, що виконуються наступні умови:

1. Точні значення функціонально залежної величини $\left[\underline{y}^{(k)}, \overline{y}^{(k)} \right]$ повинні знаходитись в інтервалі $\left[\underline{y}(z^{(k)}), \overline{y}(z^{(k)}) \right]$;

2. Ширина інтервалу $\left[\underline{y}(z^{(k)}), \overline{y}(z^{(k)})\right]$ повинна бути мінімізована.

Такі вимоги можуть бути зведені до наступної задачі лінійного програмування:

$$\min\left(\sum_{k=1}^{M}\left(\left(\sum_{i=1}^{n}\hat{a}_i\overline{z}_i^{(k)}+\sum_{i=1}^{n}\overline{a}_i\hat{z}_i^{(k)}\right)-\left(\sum_{i=1}^{n}\hat{a}_i\underline{z}_i^{(k)}+\sum_{i=1}^{n}\underline{a}_i\hat{z}_i^{(k)}\right)\right)+Mc_0\right)$$

$$\sum_{i=1}^{n}\hat{a}_i\underline{z}_i^{(k)}+\sum_{i=1}^{n}\underline{a}_i\hat{z}_i^{(k)}+a_0-\frac{c_0}{2}\le\underline{y}^{(k)}$$

$$\sum_{i=1}^{n}\hat{a}_i\overline{z}_i^{(k)}+\sum_{i=1}^{n}\overline{a}_i\hat{z}_i^{(k)}+a_0+\frac{c_0}{2}\ge\overline{y}^{(k)} \qquad (11)$$

$$k=\overline{1,M}$$

$$c_0\ge 0$$

Згрупувавши невідомі:

$$\min\left(\sum_{k=1}^{M}\left(\left(\sum_{i=1}^{n}\frac{\underline{a}_i+\overline{a}_i}{2}\overline{z}_i^{(k)}+\sum_{i=1}^{n}\overline{a}_i\hat{z}_i^{(k)}\right)-\right.\right.$$

$$\left.\left.-\left(\sum_{i=1}^{n}\frac{\underline{a}_i+\overline{a}_i}{2}\underline{z}_i^{(k)}+\sum_{i=1}^{n}\underline{a}_i\hat{z}_i^{(k)}\right)\right)+Mc_0\right)$$

$$\sum_{i=1}^{n}\frac{\underline{a}_i+\overline{a}_i}{2}\underline{z}_i^{(k)}+\sum_{i=1}^{n}\underline{a}_i\hat{z}_i^{(k)}+a_0-\frac{c_0}{2}\le\underline{y}^{(k)}$$

$$\sum_{i=1}^{n}\frac{\underline{a}_i+\overline{a}_i}{2}\overline{z}_i^{(k)}+\sum_{i=1}^{n}\overline{a}_i\hat{z}_i^{(k)}+a_0+\frac{c_0}{2}\ge\overline{y}^{(k)}$$

$$k=\overline{1,M}$$

$$c_0\ge 0$$

$$\min\left(\sum_{i=1}^{n}\overline{a}_i\left(\sum_{k=1}^{M}\overline{z}_i^{(k)}\right)-\sum_{i=1}^{n}\underline{a}_i\left(\sum_{k=1}^{M}\underline{z}_i^{(k)}\right)+Mc_0\right)$$

$$\sum_{i=1}^{n}\overline{a}_i\frac{\underline{z}_i^{(k)}}{2}+\sum_{i=1}^{n}\underline{a}_i\left(\underline{z}_i^{(k)}+\frac{\overline{z}_i^{(k)}}{2}\right)+a_0-\frac{c_0}{2}\le\underline{y}^{(k)} \quad k=\overline{1,M} \quad c_0\ge 0$$

$$\sum_{i=1}^{n}\overline{a}_i\left(\overline{z}_i^{(k)}+\frac{\underline{z}_i^{(k)}}{2}\right)+\sum_{i=1}^{n}\underline{a}_i\frac{\overline{z}_i^{(k)}}{2}+a_0+\frac{c_0}{2}\ge\overline{y}^{(k)}$$

Оскільки на параметри $\overline{a}_i$, $\underline{a}_i$ та a_0 не накладаються умови невід'ємності, то для розв'язку задачі оцінювання параметрів більш зручно перейти до двоїстої задачі лінійного програмування:

$$\max(-\sum_{k=1}^{M}\underline{y}^{(k)}\cdot\delta_k+\sum_{k=1}^{M}\overline{y}^{(k)}\cdot\delta_{k+M})$$

$$-\sum_{k=1}^{M}\frac{\underline{z}_i^{(k)}}{2}\delta_k+\sum_{k=1}^{M}\left(\overline{z}_i^{(k)}+\frac{\underline{z}_i^{(k)}}{2}\right)\delta_{k+M}=\sum_{k=1}^{M}\overline{z}_i^{(k)}$$

....

$$-\sum_{k=1}^{M}\left(\underline{z}_i^{(k)}+\frac{\overline{z}_i^{(k)}}{2}\right)\delta_k+\sum_{k=1}^{M}\frac{\overline{z}_i^{(k)}}{2}\delta_{k+M}=-\sum_{k=1}^{M}\underline{z}_i^{(k)} \qquad (12)$$

....

$$-\sum_{k=1}^{M}\delta_k+\sum_{k=1}^{M}\delta_{k+M}=0$$

$$\frac{1}{2}\sum_{k=1}^{M}\delta_k+\frac{1}{2}\sum_{k=1}^{M}\delta_{k+M}\le M$$

$$\delta_k\ge 0$$

Ця задача завжди має розв'язок. У відповідності до теореми двоїстості, оптимальні рішення прямої та двоїстої задачі співпадають. Тому, розв'язавши двоїсту задачу симплекс-методом та знайшовши оптимальні значення двоїстих змінних, ми зможемо знайти значення змінних $\overline{a}_i$, $\underline{a}_i$, a_0, c_0, $i=\overline{1,C_{r+1}^2}$, тобто визначити параметри лінійної інтервальної моделі.

Порівняння маиематичної складності методів побудови моделей-претендентів

Параметр → **Метод ↓**	**Складність математичної задачі знаходження моделі-претенденту**	**Обмеження на вхідні дані**	**Кількість параметрів математичної моделі**
МГУА з нечіткими входами	Зводиться до задачі лінійного програмування	Немає	n+1
НМГУА з нечіткими входами	Зводиться до задачі лінійного програмування	Немає	2(n+1)

Де n – кількість вхідних змінних у задачі.

Висновки

Задача знаходження моделі об’єкту розбивається на певну кількість задач лінійного програмування, що вирішуються двоїстим симплекс методом. Зважаючи на умови їх постановки, ці задачі завжди мають розв’язок.

За своєю складністю отриманий метод суттєво не відрізняється від Методу групового урахування аргументів з нечіткими входами. Відмінність у тому, що підчас побудови моделі-претенденту кількість змінних у задачі лінійного програмування НМГУАнв удвічі більша у порівнянні з чітким методом. Однак це не суттєво впливає на складність вирішення таких задач.

Напрямок подальших досліджень

Провести розробку програмного продукту. Побудувати модель конкретного економічного об’єкту. Отримати прогноз за допомогою отриманої моделі. Порівняти отримані результати з методом групового урахування аргументів з нечіткими входами за середньоквадратичним відхиленням, коефіцієнтом кореляції та середньою шириною прогнозованого інтервалу. Порівняти результати роботи методу на різних видах часткових описів.

Перелік посилань на літературні джерела

1. Ивахненко А.Г., Зайченко Ю.П., Димитров В.Д. Принятие решений на основе самоорганизации. М.:Соврадио, 1967г
2. Іваненко О.Г., Іваненко Г.О. Індуктивні методи прогнозування та аналізу складних економічних систем. – К., НІСД, 1997 р.
3. Зайченко Ю.П., Кебкал О.Г., Крачковський В.Ф. Нечіткий метод групового врахування аргументів та його застосування в задачах прогнозування макроекономічних показників. //Наукові вісті НТУУ «КПІ», №2, 2000р. с.18-26.
4. Пінчук Д.А. Розробка та дослідження методу індуктивного моделювання з нечіткими вхідними змінними. // Системні дослідження та інформаційні технології.// Системні дослідження та інформаційні технології, №1, 2008р.
5. Yuriy Zaychenko. The Fuzzy Group Method of Data Handling with Fuzzy Input Variables. //Fifth International Conference “Information Research applications”, proceedings vol.1-pp99-110-2007, ITHEA, Sofia, 2007.

УДК 681.14

МУХИН В.Е.

СРЕДСТВА УПРАВЛЕНИЯ РИСКАМИ БЕЗОПАСНОСТИ КОМПЬЮТЕРНЫХ СИСТЕМ

Одним из важных аспектов повышения защищенности компьютерных систем является анализ существующих рисков безопасности информационных ресурсов. Предлагается комплексная система управления рисками безопасности, позволяющая анализировать возможные вторжения в компьютерные системы и выбирать наиболее эффективные средства защиты информации.

The prominent aspect of computer systems security level increasing is the analysis of existing risks for information resources safety. The complex safety risks control system is suggested. This system allows analyze the potential intrusions into computer systems and makes recommendation how to choose the most effective mechanisms for the information security.

Введение

Риск безопасности компьютерных систем представляет собой возможность использования уязвимостей компьютерных систем с целью вторжения в них и нанесения определенного ущерба, при этом под уязвимостью понимаются ресурсы системы, используемые для реализации вторжений [1].

Для повышения защищенности компьютерных систем необходимо выполнять анализ рисков их безопасности, т.е. оценивать в какой степени применяемые средства безопасности информационных ресурсов способны защитить систему от угроз [2]. Процедура анализа рисков безопасности включает в себя: выделение защищаемых ресурсов, выявление угроз этим ресурсам и оценку вероятности реализации угроз безопасности. В процессе анализа рисков также оцениваются возможные потери вследствие реализации вторжений в систему, а также формируются рекомендации по применению средств защиты. Конечной целью процедуры анализа рисков является выбор эффективных средств защиты компьютерной системы, т.е., в частности, таких средств защиты, стоимость которых не превышает величины возможных потерь в результате совершения атак на систему.

В целом, анализ и управление рисками безопасности направлены на минимизацию возможных материальных и нематериальных потерь вследствие атак на систему.

Методы анализа рисков безопасности

Современные методы анализа рисков безопасности подразделяются на две категории – количественные и качественные. [3]

Количественные методы анализа рисков основываются на оценке уровня риска для каждой угрозы с последующим расчетом суммарной величины возможных потерь в случае реализации вторжения. Средства на основе количественных методов анализа рисков позволяют сгенерировать специальный сводный отчет возможных потерь по каждой угрозе, в который входят два параметра:

1. вероятность реализации вторжения за определенный период времени;
2. величина потенциальных потерь в случае успешной реализации вторжения.

Далее, путем перемножения данных показателей, формируется обобщенная оценка возможных потерь вследствие совершенной атаки.

Качественные методы анализа рисков, в отличие от количественных, не предусматривают расчет точных значений параметров, а наличие риска выражается качественными параметрами, такими как "высокая", "средняя" или "низкая" степень риска. Данный подход обусловлен еще и тем, что некоторые потери ввиду вторжений, такие как искажение или модификация данных, не всегда могут быть выражены в терминах конкретных единиц (например, финансовых).

Сравнение количественных и качественных методов и средств анализа рисков безопасности

В настоящее время расчет величины возможных потерь ввиду наступления страхового случая широко используется страховыми компаниями, т.е. фактически они применяют количественные методы анализа рисков. Однако, в отличие от страховых случаев, для которых необходимо лишь оценить затраты по выплате страховых премий, риски безопасности компьютерных систем имеют более сложный характер и оцениваются на основе анализа всего комплекса возможных последствий вследствие вторжений. Кроме того, риски безопасности компьютерных систем не всегда могут быть выражены лишь в финансово-кредитных терминах. Потенциальные потери часто могут быть вызваны такими факторами, как, например, взаимное доверие субъектов в системе или подобными.

Преимущество *количественных методов* состоит в возможности соотносить затраты на реализацию средств защиты с потенциальными потерями вследствие вторжения и выбирать эффективные средства защиты системы. Таким образом, анализ величины возможных потерь ввиду успешной атаки позволяет рассчитать величину требуемых затрат для предотвращения угроз, но с другой стороны, при расчетах используются данные, которые не всегда имеют достаточный эмпирический базис, и, в результате, полученные количественные оценки могут оказаться некорректными.

Главное преимущество *качественных методов* – минимизация времени и затрат на проведение анализа рисков. Количественные методы тре-

буют проведения детального анализа параметров безопасности компьютерной системы для выявления угроз и уязвимостей, а также оценки вероятности их реализации и величины требуемых затрат на средства защиты системы. На практике данный анализ требует значительных ресурсов. Качественные методы позволяют быстрее получить результат с меньшими затратами ресурсов, т.к. данные методы не требуют сбора дополнительных данных и сложных вычислений. Однако, анализ рисков безопасности по качественным параметрам (“высокий”, “средний” и “низкий” риск) представляет собой не строго детерминированный подход. Одну и ту же угрозу разные администраторы безопасности могут оценивать различно: одни ее воспринимают как “высокий” риск для системы, тогда как другие, наоборот, как “низкий” риск.

В целом, при выборе средств анализа рисков необходимо учитывать следующие факторы [4,5]:

- стоимость – величину затрат на организацию средств анализа и управления рисками;
- эффективность – необходимо выбрать наиболее эффективный метод анализа риска с учетом специфики функционирования данной компьютерной системы;
- адаптируемость – средства должны быть настраиваемыми для данной компьютерной системы.
- полноту – применяемые средства должны выявлять все возможные риски безопасности в системе.

Второй фактор (эффективность) особенно важен для администраторов безопасности, принимающих решения по организации системы безопасности. Оценка эффективности средств анализа и управления рисками безопасности включает в себя следующие действия:

1. устанавливается ряд показателей, которые характеризуют эффективность средства;

2. выполняются количественные оценки показателей эффективности с учетом вероятности реализации вторжения;

3. формируется интегральная оценка показателя эффективности, которая используется для сравнения различных методов и средств управления рисками безопасности.

В качестве показателей эффективности средств анализа и управления рисками безопасности предлагается использовать следующие 7 параметров:

1. постоянство – полученные результаты при независимом анализе варьируются незначительно при различных системных конфигурациях;

2. оптимальность – затраты на реализацию средств анализа и управления рисками безопасности не превышают величину потерь при вторжении;

3. гибкость – методы и средства должны быть применимы к различным конфигурациям компьютерной системы, при этом входные данные должны легко обновляться, т.к. они периодически изменяются;

4. доступность – требуемые для средств данные должны быть постоянно доступны;

5. комплексность – необходимо выполнять анализ всех параметров, характеризующих риски;

6. подлинность – результаты анализа рисков должны быть корректными;

7. доверие – результаты анализа рисков безопасности должны быть реалистичными и нести семантическую нагрузку.

Предлагается использовать комбинацию качественных и количественных методов анализа рисков для создания системы управления рисками безопасности информационных систем.

Система управления рисками безопасности компьютерных систем

Предлагается система управления рисками безопасности, позволяющая оценивать применяемые средства защиты с учетом рисков безопасности, анализировать возможные вторжения в компьютерные системы и выбирать наиболее адекватные и эффективные средства защиты компьютерных систем. Структура данной системы управления рисками безопасности показана на рис.1.

Рассмотрим детальнее все пять подсистем в предложенной системе управления рисками безопасности.

1. *Подсистема анализа стоимости ресурсов и информации компьютерных систем.*

Данный анализ выполняется в два этапа. Вначале оценивается критичность обрабатываемой информации и каждому программному приложению присваивается уровень критичности, который связан с требуемой степенью защищенности обрабатываемых данных. На втором этапе оценивается стоимость аппаратных ресурсов компьютерной системы.

2. *Подсистема анализа вторжений и рисков безопасности.*

Данный анализ проводится в три этапа:

- выявляются уязвимости: устанавливаются “узкие” места или недостатки в проектировании или реализации средств управления безопасностью системы; проверяется совпадают ли данные уязвимости с раннее выявленными угрозами безопасности;
- оцениваются веса уязвимостей: для выявленных уязвимостей устанавливаются их взаимосвязи, после чего они ранжируются по степеням опасности их реализации;
- оценивается вероятность реализации угрозы.

3. Подсистема оценки потерь вследствие вторжений в компьютерные системы и затрат на реализацию средств защиты.

Потери вследствие реализации вторжений и затраты на средства защиты компьютерных систем оцениваются с учетом предоставляемого сред-

ствами защиты уровня безопасности. Учитывается стоимость средств защиты, вероятность того, что угрозы безопасности будут реализованы, а также возможные потери в результате реализации угрозы и определяется требуемый уровень защищенности системы с учетом минимизации затрат на реализацию средств защиты.

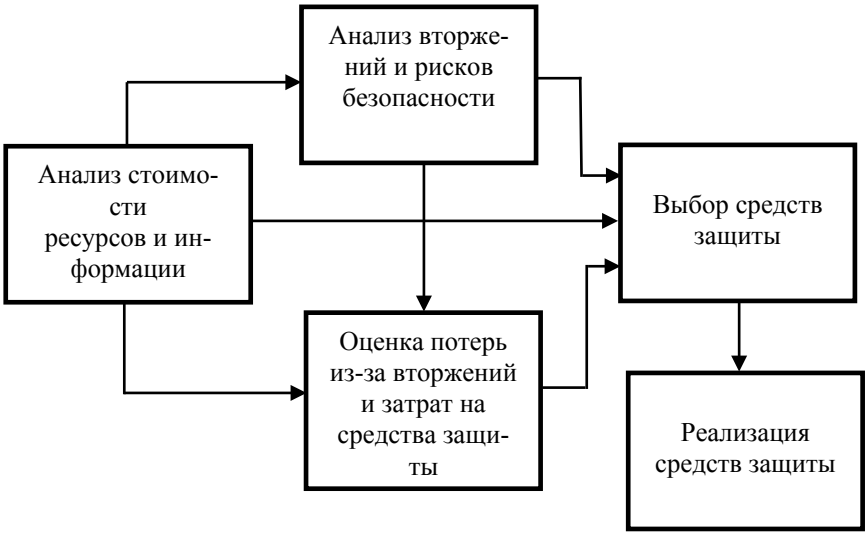


Рис. 1. Система управления рисками безопасности

4. *Подсистема выбора средств защиты компьютерных систем.*

Данный блок позволяет выбрать эффективные средства защиты. Для выбора средств защиты компьютерных систем предлагается использовать метод поиска в пространстве решений.

В процессе выбора средств защиты используются следующие параметры:

i: i - тое средство защиты, $i = \overline{1 \ldots I}$;

j: j - тая угроза, $j = \overline{1 \ldots J}$;

c_i : стоимость i-того средства защиты;

f_{ij} : эффективность i-того средства защиты для предотвращения j –той угрозы (предполагается независимость средств защиты друг от друга);

l_j : возможные потери, вызванные j –той угрозой;

P_j : вероятность реализации j –той угрозы;

B_i : ожидаемый эффект от применения i-того средства защиты;

L : суммарные возможные потери вследствие реализации угроз безопасности;

C : суммарная стоимость средств защиты;

ТС : суммарная величина стоимости средств защиты и потерь вследствие реализации угроз безопасности;

v_j : рейтинговая оценка.

Алгоритм выбора средств защиты компьютерной системы включает следующие шаги:

1. Оценка возможных потерь вследствие реализации угрозы:

$$l_j = P_j * v_j \tag{1}$$

2. Расчет эффективности средств защиты:

$$B_i = \sum_{j=1}^{J} l_j * \left\{ 1 - \prod_{i=1}^{I} (1 - f_{ij}) \right\} \tag{2}$$

3. Расчет общих возможных потерь по всем средствам защиты:

$$L = \sum_{j=1}^{J} l_j * \prod_{i=1}^{I} \left(1 - f_{ij}\right) \tag{3}$$

4. Оценка суммарной стоимости средств защиты:

$$C = \sum_{i=1}^{I} c_i \tag{4}$$

5. Оценка величины суммарных возможных затрат по всем средствам защиты с учетом угроз безопасности:

$$TC = C + L \tag{5}$$

Главная задача средств анализа и управления рисками состоит в минимизации параметра ТС путем выбора соответствующих средств защиты.

Реализация средств защиты компьютерных систем.

Осуществляется в три этапа. На первом этапе устанавливается специфика реализации средств защиты, при этом учитывается степень опасности осуществления несанкционированных действий. На втором этапе собственно имплементируются средства защиты. На третьем этапе параметры средств защиты оцениваются и верифицируются. Критичные компьютерные системы, к безопасности которых предъявляются особые требования, проходят оценку и тестирование применяемых в них средств защиты в процессе разработки и начальной эксплуатации системы. Задача тестирования состоит в подтверждении эффективности применяемых средств защиты и отсутствия нежелательных побочных эффектов.

Для демонстрации предложенной системы управления рисками безопасности проведем экспериментальные исследования. Предположим, что существует 5 возможных угроз: 1) вирус, 2) превышение полномочий легальным пользователем, 3) вход злоумышленника в систему при отказе в обслуживании легальному пользователю, 4) нарушение функционирования системы и 5) несанкционированный доступ незарегистрированным пользователем.

Рассмотрим три варианта реализации средств управления доступом к ресурсам компьютерных систем: 1) Контролируемое управление доступом (DAC), 2) Мандатное управление доступом (MAC) и 3) Ролевое управление доступом (RBAC).

В табл. 1 и табл. 2 представлена информация по вероятностям реализации угроз безопасности и соответствующим возможным потерям, а также по эффективности средств защиты в зависимости от типа угроз безопасности. Оценка стоимости суммарных затрат по всем угрозам безопасности для различных средств защиты информации проводится с учетом вероятности реализации угроз безопасности и величины возможных потерь и составляет: для средства на основе DAC – 3840 условных единиц (у.е.), на основе MAC – 2020 у.е., на основе RBAC – 1350 у.е. Таким образом, для средств на основе RBAC характерны наименьшие возможные потери в случае попыток реализации угроз безопасности и данные средства являются наиболее эффективными.

Табл. 1. Угрозы безопасности, вероятности их реализации и возможные потери

Тип угрозы	Вероятность реализации	Возможные потери, у.е.
Вирус	0.3	2000
Превышение полномочий легальным пользователем	0.1	4000
Вход при отказе в обслуживании	0.1	3000
Нарушение функционирования системы	0.2	5000
Несанкционированный доступ незарегистрированным пользователем (НП)	0.5	4000

Табл. 2. Эффективность средств защиты (управления доступом) в зависимости от типа угроз безопасности

Угроза/ Средство защиты	Вирус	Превышение полномочий легальным пользователем	Вход при отказе в обслуживании	Нарушение функционирования системы	Несанкционированный доступ НП
DAC	0.1	0.1	0.2	0.1	0.1
MAC	0.5	0.5	0.6	0.4	0.6
RBAC	0.8	0.9	0.7	0.7	0.6

Таким образом, предлагаемая система управления рисками безопасности является вспомогательным инструментом администратора безопасности для определения требуемого уровня безопасности ресурсов компьютерной системы, выявления угроз, для оценки величины потерь, которые

могут возникнуть в случае реализации угроз безопасности, и, наконец, для выбора эффективных средств защиты.

Заключение

В настоящий момент весьма актуальными являются исследования в области:

- разработки эффективных методов принятия решения в условиях множественных угроз безопасности;
- дифференциации средств защиты компьютерных систем в соответствии с используемыми в них механизмами защиты в зависимости от типа угроз безопасности;
- разработки методов анализа требуемого уровня защищенности компьютерных систем с учетом оценки вероятности реализации угроз безопасности.

Также актуальным является усовершенствование системы управления рисками безопасности для более корректной работы с расширенным набором исходных данных по событиям безопасности в компьютерных системах, в частности, на основе статистических данных компаний, например, занимающихся электронной коммерцией, для верификации корректности полученных результатов и рекомендаций на практике.

Список использованной литературы

1. C. Alberts, A. Dorofee. Managing Information Security Risks. The Octave Approach. SEI Series, Addison Wesley, 2003. – 205 p.
2. F. Farahmand, S. B. Navathe, P. Sharp Gunter, P.H. Enslow. Managing Vulnerabilities of Information Systems to Security Incidents.//ACM ICEC September 2003, Pittsburg, USA. – pp. 125-133.
3. A. Freeman. Cyber Risk Management and National Strategy to Secure Cyberspace.// The Open Group Conference Proceedings, Feb. 2003. – pp. 63-69.
4. L.A. Gordon, M.P. Loeb, T. Sohail. A Framework for Using Insurance for Cyber-Risk Management. //Communications of ACM, Vol. 46, N. 3, March 2003. – pp. 81-85.
5. П. Покровский. Оценка информационных рисков. // LAN, 2004. – Т. 10, N.10. – с.90-95.

УДК 683.519

ЗАЙЧЕНКО Ю.П.,
ГОНЧАР М.А.

НЕЧЕТКИЕ МЕТОДЫ КЛАСТЕРНОГО АНАЛИЗА В ЗАДАЧАХ АВТОМАТИЧЕСКОЙ КЛАССИФИКАЦИИ В ЭКОНОМИКЕ

В статье рассмотрена задача кластерного анализа в условиях неопределенности и описаны нечеткие методы кластерного анализа – *k*-средних и Густавссона-Кесселя. Проведены экспериментальные исследования предложенных алгоритмов на примере автоматической классификации стран СНГ по макроэкономическим показателям.

The problem of cluster analysis is considered and fuzzy clustering methods C-means and of Gustavssona-Kessel are described. The experimental investigations of the fuzzy clustering algorithm are described and it's application for automatic classification of CIS countries is presented.

Введение

Задачи кластер-анализа, или автоматической классификации получили широкое применение в экономике, социологии, медицине, геологии и других отраслях, всюду где имеются множества объектов произвольной природы, описываемых в виде векторов $x = \{x_1, x_2, \ldots, x_N\}$, которые необходимо автоматически разбить на группы однородных объектов по признакам «сходства-различия». В последние годы эти методы широко применяются в задачах интеллектуального анализа данных и Data mining. Традиционные методы кластер-анализа предполагают четкое разбиение исходного множества на подмножества, при котором каждая точка после разбиения попадает только в один кластер. Однако такое ограничение не всегда верно.

Зачастую необходимо произвести разбиение так, чтобы определить степень принадлежности каждого объекта к каждому множеству. В этом случае целесообразно использовать нечеткие методы кластер-анализа.

Цель настоящей статьи – провести исследование и анализ нечетких методов и алгоритмов кластер-анализа в экономике и финансовой сфере.

Рассмотрим постановку задачи нечеткого кластер-анализа.

1. Постановка задачи

Имеется N объектов $x = \{x_1, x_2, \ldots, x_N\}$, где $x_j = \lfloor x_{j1}, x_{j2}, \ldots, x_{jk} \rfloor$. Необходимо разбить их на k кластеров и определить места размещения центров кластеров c_i, $i = \overline{1,k}$.

Обозначим через u_{ij} – степень принадлежности точки x_j j-му кластеру.

На величины u_{ij} накладываются следующие ограничения [1]:

$$0 \le u_{ij} \le 1 \text{ и } \sum_{i=1}^{k} u_{ij} = 1 \text{ для всех } j. \qquad (1)$$

Требуется найти такое размещение центров кластеров c_i в пространстве объектов ($x_1,\ldots, x_n$) и величины $\{u_{ij}\}$ при которых величина критерия (средневзвешенное отклонение точек x_j от центров кластеров) было минимально, т.е.:

$$\min_C E = \sum_{i=1}^{k} \sum_{j=1}^{n} u_{ij}^m \left\| x_j - c_i \right\|^2, \quad (2)$$

при условии (1), где $m > 1$, m – целое.

Для решения этой задачи разработаны нечеткий метод k-средних и алгоритм Густавссона-Кесселя.

2. Алгоритм нечёткой самоорганизации k-средних

Допустим, что в сети существует K нечетких нейронов с центрами в точках c_i (i = 1, 2, …, K). Начальные значения этих центров могут быть выбраны случайным образом из областей допустимых значений соответствующих компонентов векторов x_j $j = \overline{1, p}$, использованных для обучения. Пусть функция фаззификации задана в форме обобщенной функции Гаусса.

Подаваемый на вход сети вектор x_j будет принадлежать к различным группам, представляемым центрами c_i, в степени u_{ij}, причем $0 < u_{ij} < 1$, а суммарная степень принадлежности ко всем группам, очевидно, равна 1.

Поэтому

$$\sum_{i=1}^{K} u_{ij} = 1, \qquad (3) \qquad \text{для } j = \overline{1, p}.$$

Функцию погрешности, соответствующую такому представлению, можно определить как сумму частных погрешностей принадлежности к центрам c_i с учетом степени принадлежности u_{ij}. Следовательно,

$$E = \sum_{i=1}^{K} \sum_{j=1}^{p} u_{ij}^m \left\| c_i - x_j \right\|^2, \qquad (4)$$

где m – это весовой коэффициент, который принимает целочисленные значения из интервала [2,N].

Цель обучения с самоорганизацией состоит в таком подборе центров c_i, а также состава кластеров, чтобы для заданного множества обучающих векторов x_j – обеспечить достижение минимума функции (4) при одновременном соблюдении условий ограничения (3). Таким образом возникает задача минимизации нелинейной функции (4) с p ограничениями

типа (3). Решение этой задачи можно свести к минимизации функции Лагранжа, определенной в виде [1]:

$$LE = \sum_{i=1}^{K}\sum_{j=1}^{N} u_{ij}^{m}\left\|c_i - x_j\right\|^2 + \sum_{j=1}^{p}\lambda_j\left(\sum_{i=1}^{K} u_{ij} - 1\right), \quad (5)$$

Где λ_j (j=1,2, …, p) – это множители Лагранжа. В [1] доказано, что решение задачи (5) можно представить в виде

$$c_i = \frac{\sum_{j=1}^{p} u_{ij}^{m} x_j}{\sum_{j=1}^{p} u_{ij}^{m}}, \quad (6)$$

и

$$u_{ij} = \frac{1}{\sum_{k=1}^{K}\left(\frac{d_{ij}^2}{d_{kj}^2}\right)^{\frac{1}{m-1}}}, \quad (7)$$

где d_{ij} – это эвклидово расстояние между центром c_i, и вектором x_j, $d_{ij} = \left\|c_i - x_j\right\|$. Поскольку точные значения центров c_i, в начале процесса не известны, алгоритм решения данной задачи должен быть итерационным. Он может быть сформулирован в следующем виде:

Выполнить случайную инициализацию переменных u_{ij}, выбирая их значения из интервала [0,1] таким образом, чтобы соблюдалось условие (3).

Определить k центров c_i, в соответствии с (6).

Рассчитать значение функции погрешности согласно выражению (4). Если ее значение окажется ниже установленного порога либо, если уменьшение этой погрешности относительно предыдущей итерации пренебрежимо мало, то завершить вычисления. Последние значения центров составляют искомое решение. В противном случае перейти к п. 4.

Рассчитать новые значения u_{ij} по формуле (7) и перейти к п. 2.

Такую процедуру нечеткой самоорганизации будем называть алгоритмом C-means.

Многократное повторение итерационной процедуры ведет к достижению минимума функции E, которая необязательно будет глобальным минимумом. Качество находимых центров, оцениваемое значением функции погрешности E существенным образом зависит от предварительного

подбора как значений u_{ij}, так и центров c_i. Наилучшим может быть признано такое размещение центров, при котором они располагаются в областях, содержащих наибольшее количество предъявленных векторов x_j. При таком подборе центров они будут представлять векторы данных x_j с наименьшей суммарной погрешностью.

Поэтому начало итерационной процедуры расчета оптимальных значений центров должно предваряться процедурой их инициализации. К наиболее известным алгоритмам инициализации относятся алгоритмы пикового группирования и разностного группирования данных.

3. Алгоритм нечеткого кластер-анализа Густавссона-Кесселя

В классическом алгоритме *k*-средних (c-means) элементы выбираются с помощью обычного евклидового расстояния между вектором *x* и центром кластера *c*:

$$d(x,c) = \|x - c\| = \sqrt{(x-c)^T (x-c)}\,.$$

При таком задании расстояния между двумя векторами множество точек, равноудаленных от центра, принимает вид сферы с одинаковым масштабом по всем осям. Но если данные создают группы, форма которых отличается от сферической или если шкалы отдельных координат вектора сильно отличаются, в этом случае метрика становится неадекватной. В этом случае качество кластеризации можно значительно повысить за счет улучшенной версии алгоритма самоорганизации, который называется алгоритмом Густавссона-Кесселя.

Основные изменения относительно базового алгоритма *k*-средних в ведении в формулу расчета метрики масштабирующей матрицы *A*. При таком масштабировании расстояние между центром *c* и векторами *x* определяется формулой:

$$d(x,c) = \|x - c\| = \sqrt{(x-c)^T A(x-c)}$$

В качестве масштабирующей обычно используется положительно-определенная матрица, то есть матрица, у которой все собственные числа действительные и положительные.

Аналогично базовому алгоритму *k*-средних цель обучения с использованием алгоритма Густавсона-Кесселя в таком размещении центров, при котором минимизируется критерий:

$$E = \sum_i \sum_j u_{ij}^m d^2(x_j, c_i)\,. \qquad (8)$$

Описание алгоритма Густавссона-Кесселя:

Провести начальное размещение центров в пространстве данных. Создать элементарную форму масштабирующей матрицы А.

Сформировать матрицу коэффициентов принадлежностей всех векторов x к центрам c по формуле:

$$u_{ij} = \frac{1}{\sum_{k=1}^{K} \left(\frac{d^2(x_j, c_i)}{d^2(x_j, c_k)} \right)^{\frac{1}{m-1}}}. \qquad (9)$$

Рассчитать новое размещение центров c в соответствии с формулой:

$$c_i = \frac{\sum_{j=1}^{N} u_{ij}^m x_j}{\sum_{j=1}^{N} u_{ij}^m}. \qquad (10)$$

Сгенерировать для каждого вектора матрицу ковариаций:

$$S_i = \sum_{j=1}^{N} u_{ij}^m (x_j - c_i)(x_j - c_i)^T. \qquad (11)$$

Рассчитать новую масштабную матрицу для каждого i-го центра по формуле:

$$A_i = \sqrt[n]{\det(S_i)} \cdot S_i^{-1}. \qquad (12)$$

Если последние изменения положений центров и матрицы ковариации достаточно малые по отношению к предыдущим значениям и не превышают заданной величины, то – завершить итерационный процесс.

Данные методы обладают следующим недостатком – число кластеров k должно быть известным. Для устранения первого недостатка и решения задачи кластерного анализа при незаданном числе кластеров разработан подход, основанный на вычислении величины: $\Delta E(k)=E(k-1)-E(k)$,
где $E(k-1)$ и $E(k)$ – оптимальное значение критерия для числа кластеров $(k-1)$ и (k).

Как только величина $\frac{\Delta E(k)}{E_0} \le \Theta$, где $\Theta = (0{,}2 \div 0{,}3)$ – величина заданного порога, процесс увеличения числа кластеров прекращается.

4. Применение нечетких методов *k*-средних и Густавссона-Кесселя в задачах автоматической классификации

Кластеризация регионов Украины по данным Госкомстата за январь-март 2005 года.

Исходные данные:

Табл.1

	Темпы роста промышленного производства	Темпы роста сельского хозяйства	Темпы роста строительства	Темпы роста розничной торговли
Автономная Республика Крым	**118,6**	**110,5**	**81.2**	**127,5**
Винницкая	**108,5**	**101,9**	**100,1**	**116,8**
Волынская	**117,2**	**105,7**	**83.1**	**117,9**
Днепропетровская	**112,3**	**117,1**	**126,9**	**115,4**
Донецкая	**94.4**	**113,3**	**115,9**	**120,2**
Житомирская	**114,8**	**109,5**	**86.7**	**113,6**
Закарпатская	**113,2**	**98.9**	**72.9**	**118,0**
Запорожская	**112,8**	**114,1**	**106,2**	**132,2**
Ивано-Франковкая	**106,2**	**96.4**	**91.5**	**120,7**
Киевская	**115,9**	**110,4**	**106,8**	**104,5**
Кировоградская	**115,1**	**82.3**	**118,9**	**120,3**
Луганская	**100,0**	**98.2**	**111,5**	**142,9**
Львовская	**101,3**	**100,1**	**102,7**	**114,4**
Николаевская	**110,9**	**101,6**	**106,5**	**120,1**
Одесская	**103,9**	**102,9**	**153,6**	**116,1**
Полтавская	**101,4**	**103,4**	**115,5**	**106,5**
Ровенская	**116,2**	**102,3**	**102,6**	**118,2**
Сумская	**117,4**	**100,7**	**90.5**	**115,1**
Тернопольская	**85.3**	**98.1**	**93.8**	**120,3**
Харьковская	**113,4**	**101,3**	**82.1**	**122,6**
Херсонская	**138,0**	**118,3**	**157,7**	**115,1**
Хмельницка	**133,2**	**94.5**	**78.3**	**112,6**
Черкасская	**112,4**	**112,5**	**94.5**	**110,5**
Черновецкая	**94**	**98.4**	**114,8**	**115,2**
Чернигивская	**109,6**	**103,3**	**115,7**	**108,6**
г.Киев	**117,2**	**100**	**310**	**119,7**
г.Севастополь	**107,3**	**100**	**107,7**	**128,7**

Табл. 2. Результаты кластеризации

Автономная Республика Крым	**0.980**	**0.018**	**0.001**	**0.000**
Винницкая	**0.000**	**0.996**	**0.002**	**0.001**

Волынская	**0.000**	**0.318**	**0.678**	**0.004**
Днепропетровская	**0.000**	**0.009**	**0.001**	**0.990**
Донецкая	**0.000**	**0.037**	**0.942**	**0.020**
Житомирская	**0.000**	**0.050**	**0.946**	**0.003**
Закарпатская	**0.576**	**0.419**	**0.003**	**0.002**
Запорожская	**0.988**	**0.010**	**0.000**	**0.001**
Ивано-Франковкая	**0.795**	**0.199**	**0.003**	**0.003**
Киевская	**0.000**	**0.142**	**0.812**	**0.046**
Кировоградская	**0.000**	**0.005**	**0.005**	**0.990**
Луганская	**0.000**	**0.991**	**0.007**	**0.002**
Львовская	**0.000**	**0.991**	**0.001**	**0.008**
Николаевская	**0.000**	**0.986**	**0.013**	**0.001**
Одесская	**0.000**	**0.814**	**0.108**	**0.078**
Полтавская	**0.000**	**0.985**	**0.002**	**0.012**
Ровенская	**0.000**	**0.988**	**0.009**	**0.003**
Сумская	**0.000**	**0.937**	**0.057**	**0.006**
Тернопольская	**0.000**	**0.992**	**0.001**	**0.007**
Харьковская	**0.000**	**0.916**	**0.082**	**0.002**
Херсонская	**0.000**	**0.016**	**0.000**	**0.984**
Хмельницка	**0.000**	**0.065**	**0.929**	**0.006**
Черкасская	**0.000**	**0.019**	**0.976**	**0.005**
Черновецкая	**0.000**	**0.617**	**0.002**	**0.381**
Чернигивская	**0.000**	**0.975**	**0.011**	**0.014**
г.Киев	**0.000**	**0.103**	**0.002**	**0.895**
г.Севастополь	**0.000**	**0.993**	**0.006**	**0.001**

Центры кластеров:

Первый	**113,38**	**107,19**	**90,46**	**126,35**
Второй	**105,90**	**100,98**	**104,07**	**119,21**
Третий	**114,49**	**107,71**	**94,33**	**113,18**
Четвертый	**119,57**	**104,36**	**124,62**	**117,42**

Из расположения центров кластеров видно:

В первом и третьем кластере находятся области с высокими темпами роста производства и сельского хозяйства и низкими темпами роста строительства. Различие между кластерами в темпах роста розничной торговли (для АР Крыма, Закарпатской, Запорожской и Ивано-Франковской областей (первый кластер) они высокие, для Волынской, Донецкой, Житомирской, Киевской, Хмельницкой, Черкасской (третий кластер) – ниже).

Во втором кластере находятся области с низкими темпами роста (Винницкая, Луганская, Львовская, Николаевская, Одесская, Полтавская, Ровенская, Сумская, Тернопольская, Харьковская, Сумская, Черновицкая, Черниговская, г. Севастополь).

Днепропетровская, Кировоградская, Херсонская области и г. Киев находятся в четвертом кластере (высокие темпы роста).

Значение минимизируемого критерия ошибки: E=4287.

Табл. 3. Расстояние между центрами кластеров

первый – второй	18,18588
второй – третий	15,82117
Третий – четвертый	31,18478
первый – третий	13,78145
второй – четвертый	24,97599
первый – четвертый	35,95797

Как видим, наименьшее расстояние между первым и третьим кластерами: собственно различия только в значениях четвертого параметра. Наибольшее – между первым и четвертым и между третьим и четвертым кластерами.

Заключение

1. В статье сформулирована задача нечеткой кластеризации экономических объектов по их характеристикам.
2. Описаны алгоритмы нечеткой классификации *k*-средних и Густавсона-Кесселя и описан подход к их использованию в случаях, когда число кластеров *k* заранее не задано.
3. Проведены экспериментальные исследования разработанных алгоритмов кластер-анализа на примере автоматической классификации стран СНГ.

Список использованной литературы

1. Зайченко Ю.П. Основи проектування інтелектуальних систем. Навчальний посібник. – К.: Видавничий Дім «Слово», 2004. – 352с.
2. Никифорова Н.С. Кластерный анализ в задачах социально-экономического прогнозирования.

УДК 519.854.2

ПАВЛОВ А.А.,
ИВАНОВА А.А.,
ЗИГУРА Р.А.

МЕТОД ГРУППОВОГО УЧЁТА АРГУМЕНТОВ И АНАЛИЗА ИЕРАРХИЙ (МГУАИАИ) В ЗАДАЧАХ ПРИНЯТИЯ РЕШЕНИЙ

Метод анализа иерархий (МАИ) Саати [1–4] в настоящее время является наиболее употребимым для решения реальных задач многокритериального выбора. Если альтернативы в МАИ задаются векторами, то формально задача многокритериального выбора может решаться так же методом группового учета аргументов (МГУА) А.Г. Ивахненко.

В статье предлагается метод группового учета аргументов и анализа иерархий (МГУАиАИ), объединяющий в себе преимущества и особенности МГУА и МАИ.

Nowadays Saaty's Analytical Hierarchy Process (AHP) is the most applicable one for solving practical problems of multicriteria choice. In case the alternatives in AHP are vectors the problem of multicriteria choice can be also solved with the help of the Method of Group Accounting of Arguments by Ivahnenko.

The Method of Group Accounting of Arguments and Hierarchy Analysis that combines the advantages and peculiarities of both methods mentioned above is proposed in the article.

Постановка задачи

Есть глобальная цель, которую количественно можно задать числовой скалярной ограниченной непрерывной функцией *f*(*A*), A – произвольная альтернатива, выражаемая n – мерным вектором $\overline{x} = (x_1, \ldots x_n)^{\top}$, x_j, $j = \overline{1, n}$ ограниченное численное значение *j*-того показателя, характеризующего альтернативу A.

Примечание. Числовой скалярной непрерывной функции *f*(*A*) не существует, если с точки зрения глобальной цели может быть нарушено условие транзитивности: существуют альтернативы A_i, A_j, A_e для которых в соответствии с глобальной целью выполняется $A_i \succ A_j, A_j \succ A_e, A_e \succ A_i$.

Необходимо найти наилучшую альтернативу из множества $\{A_i, i = \overline{1, m}\}$ с точки зрения глобальной цели. В такой постановке наиболее употребимый метод решения – метод анализа иерархий Саати [1–4]. При этом задание альтернативы A вектором $\overline{x}$ не является существенным.

Вкратце основные этапы решения задачи многокритериального выбора с помощью метода анализа иерархий Т. Саати [1–4]следующие:

1. Очертить проблему и определить, что необходимо узнать.

2. Построить иерархию, начиная с вершины (цели – с точки зрения управления), через промежуточные уровни (критерии, от которых зависят следующие уровни) к самому нижнему уровню, который обычно является перечнем альтернатив.

3. Построить, используя эксперта (экспертов) множество матриц парных сравнений для каждого из нижних уровней – по одной матрице для каждого элемента примыкающего сверху уровня.

4. По матрицам парных сравнений найти значения весов объектов.

5. Найти согласованность всей иерархии.

6. Найти наилучшую альтернативу.

Заложенный в МАИ принцип идентичности и декомпозиции предусматривает структурирование проблем в виде иерархии или сети. Обычно в наиболее элементарном виде иерархическая структура строится с вершины, через промежуточные уровни к самому низкому уровню. Поэтому первый шаг при решении задачи многокритериального выбора состоит в декомпозиции и представлении задачи в иерархической форме, что показано на рис.1.

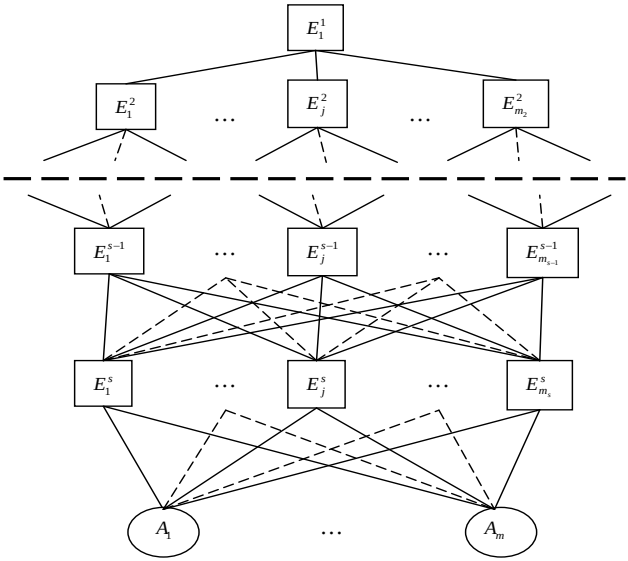


Рис. 1. Пример иерархического представления задачи принятия решений

В представленной на рисунке задаче имеем *m* альтернатив $A_1 ... A_m$ и *s* уровней критериев $E_j^i,\ i = \overline{1,s}\quad j = \overline{1,m_i}$

На последующих этапах МАИ элементы иерархии сравниваются попарно по отношению к их воздействию на общую характеристику (гло-

бальную цель либо критерий верхнего уровня). Из группы матриц парных сравнений формируется набор локальных приоритетов, которые выражают относительное влияние множества элементов на элемент примыкающего сверху уровня, т.е. в нашем случае компоненты векторов будут иметь следующий вид [6]:

$$W^A_{E_1^{s-1}} = [W^A_{E_1^s} W^A_{E_2^s} \ldots W^A_{E_{m_s}^s}] W^E_{E_1^{s-1}}$$

$$\ldots$$

$$W^A_{E_{m_{s-1}}^{s-1}} = [W^A_{E_1^s} W^A_{E_2^s} \ldots W^A_{E_{m_s}^s}] W^E_{E_{m_{s-1}}^{s-1}}$$

$$\ldots$$

$$W^A_{E_1^{s-j}} = [W^A_{E_1^{s-j+1}} W^A_{E_2^{s-j+1}} \ldots W^A_{E_{m_{s-j+1}}^{s-j+1}}] W^E_{E_1^{s-j}}$$

$$\ldots$$

$$W^A_{E_{m_{s-j}}^{s-j}} = [W^A_{E_1^{s-j+1}} W^A_{E_2^{s-j+1}} \ldots W^A_{E_{m_{s-j+1}}^{s-j+1}}] W^E_{E_{m_{s-j}}^{s-1}}$$

Последняя формула

$$W^A_{E_1^1} = [W^A_{E_1^2} W^A_{E_2^2} \ldots W^A_{E_{m_2}^2}] W^E_{E_1^1}, \qquad (1)$$

$W^E_{E_1^1} = \begin{pmatrix} w^{E_1^2}_{E_1^1} \\ \vdots \\ w^{E_{m_2}^2}_{E_1^1} \end{pmatrix}$, где $w^{E_j^2}_{E_1^1}$ – вес критерия E_j^2 в глобальную цель E_1^1, $j = \overline{1, m_2}$

$W^A_{E_1^1} = \begin{pmatrix} w^1_{E_1^1} \\ \vdots \\ w^m_{E_1^1} \end{pmatrix}$, где $w^j_{E_1^1}$ – результирующий вес j-ой альтернативы в глобальную цель E_1^1, $j = \overline{1, m}$

$W^A_{E_l^{s-j}} = \begin{pmatrix} w^1_{E_l^{s-j}} \\ w^2_{E_l^{s-j}} \\ \vdots \\ w^m_{E_l^{s-j}} \end{pmatrix}$, где $w^i_{E_l^{s-j}}$ – вес (вклад) i-ой альтернативы в критерий E_l^{s-j}, $i = \overline{1, m}$, $j = \overline{1, s-1}$, $l = \overline{1, m_{s-j}}$

$$W_{E_l^{s-j}}^{E} = \begin{pmatrix} w_{E_l^{s-j}}^{E_1^{s-j+1}} \\ w_{E_l^{s-j}}^{E_2^{s-j+1}} \\ \vdots \\ w_{E_l^{s-j}}^{E_{m_{s-j+1}}^{s-j+1}} \end{pmatrix}$$, где $w_{E_l^{s-j}}^{E_p^{s-j+1}}$ – вес E_p^{s-j+1} критерия в критерий E_l^{s-j},

$j=\overline{1,s-1},\ \ l=\overline{1,m_{s-j}},\ \ p=\overline{1,m_{s-j+1}}$.

Таким образом, j-я компонента (1) – это $W_{E_1^1}^{j}$.

Принимается та альтернатива, на которой достигается максимум

$\max\limits_{j} w_{E_1^1}^{j}$

Если в дереве иерархий не все связи имеют место, то соответствующие веса принимаются равными нулю. Поэтому в данной статье рассматривается только общий случай.

Рассмотрим, каким образом дерево иерархий **с** весами приписанными каждой из ветвей, аппроксимирует неизвестные значения $f(A_i),\ \ i=\overline{1,m}$. Для этого рассмотрим следующие выражения:

$E_j^s(A_i)$ – значение, которое принимает критерий E_j^s – на альтернативе A_i,

$$E_j^s(A_i) = w_{E_j^s}^{i},\ \ i=\overline{1,m},\ \ j=\overline{1,m_s}.$$

При этом для фиксированных s, j $w_{E_j^s}^{i}$ измеряются в одних и тех же единицах и пронормированы : $\sum\limits_{i=1}^{m} w_{E_j^s}^{i} = 1$

$$E_j^{s-1}(A_i) = \sum_{l=1}^{m_s} w_{E_j^{s-1}}^{E_l^s} E_l^s(A_i).$$

Веса $w_{E_j^{s-1}}^{E_l^s}$ при фиксированных s и j измеряются в одних и тех же единицах и пронормированы $\sum\limits_{j=1}^{m_s} w_{E_j^{s-1}}^{E_l^s} = 1$.

В общем случае $E_t^{s-j}(A_i)=\sum_{l=1}^{m_{s-j+1}} w_{E_t^{s-j}}^{E_l^{s-j+1}} E_l^{s-j+1}(A_i)$ Веса $w_{E_t^{s-j}}^{E_l^{s-j+1}}$ при фиксированных *s, j, t* измеряются в одних и тех же единицах и пронормированы $\sum_{l=1}^{m_{s-j+1}} w_{E_t^{s-j}}^{E_l^{s-j+1}}=1$.

Примечание: в иерархической системе Саати все веса считаются неотрицательными.

И наконец:

$$E_1^1(A_i)=\sum_{l=1}^{m_2} w_{E_1^1}^{E_l^2} E_l^2(A_i),\ i=\overline{1,m} \tag{2}$$

Выражение (2) и есть аппроксимация неизвестного значения $f(A_i)$, $i=\overline{1,m}$.

$E_1^1(A_i),\ i=\overline{1,m}$ можно также представить следующим образом:

$$E_1^1(A_i)=\sum_{t=1}^{m_s} \alpha_t w_{E_t^s}^{i},\ i=\overline{1,m}, \tag{3}$$

где коэффициенты α_t очевидным образом определяются из выражения (2).

Постановка задачи 2

С учетом дополнительной информации эксперт (эксперты) задают значения $f(A_i)$, $i=\overline{1,m}$ неизвестной функции цели на альтернативах A_i $i=\overline{1,m}$, ставится задача не нахождения наилучшей альтернативы, а построение такой оценочной функции $\hat{f}(A)=\hat{f}(x_1\ldots x_n)$ неизвестной функции $f(A)=f(x_1\ldots x_n)$, по которой можно было бы находить адекватные оценки $f(A)$ (A – произвольная альтернатива $\overline{\in}\{A_1\ldots A_m\}$) либо находить отношение порядка на произвольном множестве альтернатив в общем случае не совпадающем с множеством A_i, $i=\overline{1,m}$ для нахождения наилучшей альтернативы.

В этом случае наиболее конструктивным методом решения является широко известный метод эвристической самоорганизации МГУА (метод группового учета аргументов) Ивахненко А.Г. [10–12], суть которого в общих чертах заключается в следующем.

Пусть неизвестная функция $f\left(x_1 \ldots x_n\right)$ задана таблично: $f\left(x_{1i}, x_{2i} \ldots x_{ni}\right),\ i = \overline{1, p}$.

Тогда инвариантной составляющей любого алгоритма, реализующего МГУА является:

1. Разбиение множества векторов $\overline{x_i} = \left(x_{1i}, \ldots x_{ni}\right)^{\top},\ i = \overline{1, p}$ на два подмножества P_1 и P_2.

Примечание. Различия алгоритмов, реализующие МГУА, различаются правилами построения P_1 и P_2.

2. Процедура МГУА является многорядной. Каждый ряд генерирует множество частных аппроксимаций неизвестной функции $f\left(x_1, \ldots x_n\right)$. Обычно этими аппроксимациями являются полиномы от аргументов функции $f\left(x_1, \ldots x_m\right)$ и, как правило, степень этих полиномов от ряда к ряду увеличивается. В каждом ряду частные аппроксимации обычно строятся методом наименьших квадратов, используя значения неизвестной функции и соответствующие им значения аргументов из множества P_1.

Примечание. Различные алгоритмы МГУА различаются видом задания частных аппроксимаций $f\left(x_1, \ldots x_n\right)$ в каждом ряду многорядной процедуры МГУА.

3. В следующем ряду используются не все частные описания предыдущего ряда, а только часть из них, которые наилучшим образом аппроксимируют неизвестную функцию $f\left(x_1, \ldots x_n\right)$ на множестве значений аргументов P_2.

Примечание. Различные алгоритмы МГУА отличаются критериями отбора частных аппроксимаций предыдущего ряда на последующий. Есть алгоритмы в которых критерий отбора определяется на множествах P_2 и P_1.

4. Результирующей аппроксимацией является та, на которой достигается минимум значения критерия отбора частных описаний от предыдущего ряда к последующему. Эмпирически показано, что если множество P_2 является представительным, то изменение значений критерия отбора на «лучших» частных аппроксимациях каждого ряда «грубо» описывается одномодальной функцией, минимуму которой соответствует наиболее «близкое» к $f\left(x_1, \ldots x_n\right)$ частное описание.

Основным общепризнанным отличием (преимуществом) МГУА, как метода эвристической самоорганизации является использование критерия

отбора по множеству P_2, которые часто приводят к установлению истиной закономерности, выражаемой функцией $f\left(x_1,\ldots x_m\right)$.

Постановка задачи 3

Совпадает с постановкой задачи 2, однако исходные данные являются другими.

По прежнему необходимо восстановить эффективную оценку $\hat{f}\left(A\right)$ неизвестной функции $f\left(A\right)$ отражающую глобальную цель. В этом случае автоматизируется процесс выбора наилучшей альтернативы из произвольного набора альтернатив без привлечения экспертов, т.е. без временных и финансовых затрат.

Постановка задачи 2 и применение МГУА предполагает возможность непосредственного измерения значений функции $f\left(A\right)$ на множестве альтернатив A_i, $i=\overline{1,m}.$

Постановка задачи 1 и использование МАИ основано на том, что непосредственно эксперт не может измерить значения $f\left(A_i\right), i=\overline{1,m}.$ Однако естественно предположить (в отличие от постановки 1), что эксперт (эксперты) могут задать множество неравенств $G\left\{f\left(A_i\right)\text{или}>f\left(A_j\right)\text{или}=f\left(A_j\right)\text{или}\ f\left(A_i\right)\geq f\left(A_j\right)\right\}$ $i\neq j,(ij)\in I$; $i,j\in\left\{\overline{1,m}\right\}$. Каждое неравенство эксперт (эксперты) задает тогда, когда он уверен, что с точки зрения глобальной цели альтернатива A_i лучше либо не хуже либо равна по эфективности альтернативе A_j.

Примечание. Неравенства из множества *G* не должны нарушать условие транзитивности, в противном случае числовой скалярной ограниченной непрерывной функции $f\left(A\right)$ не существует и задача теряет смысл.

В общем случае множество *G* не позволяет найти наилучшую альтернативу: *G* может не задавать полный порядок (предпорядок) на множестве альтернатив $A_1\ldots,A_m$.

В такой постановке задача нахождения эффективного приближения неизвестной функции $f\left(A\right)=f\left(\overline{x}\right)$, (альтернатива *A* задается вектором $\overline{x}$), а также нахождение наилучшей альтернативы из множества $A_1\ldots A_m$ может быть решено методом объединяющим в себе преимущества метода группового учета аргументов и метода анализа иерархий, названной авторами методом группового учета элементов и анализа иерархий (МГУАиАИ).

Суть метода заключается в следующем:

1. Сначала реализуется метод анализа иерархий Саати – первые 5 из шести пунктов, сформулированные в начале данной статьи.

Примечание 1. Матрицы парных сравнений строятся не для всех m альтернатив (если m достаточно большое число), а только для тех (по Саати [1–2] не более 10), которые приводят к хорошо согласованным матрицам парных сравнений последнего уровня дерева иерархий). Выбор этих альтернатив из множества $A_1 K A_m$ является отдельной задачей. Возможность увеличения числа хорошо согласованных альтернатив можно получить с помощью задач оптимизации (модель 5) [7], (модели 6–7, 8-9) [9].

Примечание 2. Проблема реверса весов альтернатив, точнее установление факта ее отсутствия, может быть решена с помощью очевидных модификаций соответствующих моделей оптимизации [7–8].

Пусть $A_{i1},\ldots A_{ip}$ – альтернативы, заданные векторами $\overline{x_{i1}},\ldots \overline{x_{ip}}$, составили нижний ряд иерархии дерева Саати (рис. 1). С помощью МАИ найдены значения оценок $f\left(A_{ij}\right)= f\left(\overline{x_{ij}}\right),\ j=\overline{1,p}$.

2. Используем метод группового учета аргументов [10–12] в следующей модификации.

Как известно, частные описания на каждом ряду МГУА строятся по обучающей последовательности данных, а отбираются на следующий ряд по проверочной последовательности данных (в соответствии с заданным критерием отбора). В МГУАиАИ могут быть использованы любые алгоритмы его реализации, для которых выполнены следующие условия.

1) Обучающейся последовательностью данных являются значения $f\left(\overline{x_{ij}}\right),\ j=\overline{1,p}$, полученные с помощью метода анализа иерархий Саати.

2) Проверочной последовательностью данных служит множество G, которое трактуется (в случае не нарушения условий транзитивности) как множество неравенств вида $\left(f\left(\overline{x_i}\right) \text{ или } f\left(\overline{x_j}\right) \quad f\left(\overline{x_i}\right) \text{ или } f\left(\overline{x_j}\right) \quad f\left(\overline{x_i}\right)=\left(\overline{x_j}\right)\right) \quad i\neq j;(ij)\in I; i,j\in\left\{\overline{1,m}\right\}$.

Тогда любой критерий отбора частных описаний с предыдущего ряда на последующий ряд МГУА основан на анализе выражений

$M_{kp}^{(ij)}=\{0,$ *если* $f^{kp}\left(\overline{x_i}\right)> f^{kp}\left(\overline{x_j}\right)$ *или* $f^{kp}\left(\overline{x}\right)\geq f^{kp}\left(x_j\right)$ *или* $f^{kp}\left(x_i\right)= f^{kp}\left(x_j\right)$ в зависимости от типа элемента $(ij)\in I;\ \left|f^{kp}\left(\overline{x_i}\right)- f^{kp}\left(\overline{x_j}\right)\right|$ в противном случае $\}$ $(ij)\in I$; k – номер ряда МГУА, p – номер частного описания на k-том ряду МГУА. $f^{kp}\left(\overline{x}\right)$ это p – тое частное описание на k-том ряду МГУА.

Например, можно ввести суммарную оценку нарушения ограничений из множества G.

$M_2^{kp} = \frac{1}{|I|}\sum_{(ij)\in I} M_{kp}^{(ij)}$, $|I|$ – число элементов в I и на следующий уровень пропустить заданное число частных описаний предыдущего уровня с наименьшими значениями M_2^{kp}.

3) В качестве критерия остановки алгоритма может быть использован критерий остановки любой модификации метода группового учета аргументов. Учитывая специфику рассматриваемой задачи, можно предложить следующие критерии остановки:

а) Пусть M_1^{kp} – мера отклонения p – того частного описания на *к* уровне МГУА, вычисленная по обучающейся последовательности $\frac{1}{p}\sum_{j=1}^{p}\left| f\left(\overline{x_{ij}}\right) - f^{kp}\left(\overline{x_{ij}}\right)\right|$.

Тогда решение задачи – это частное описание уровня иерархии МГУА с минимальным номером на котором достигается $M_{kp}^{1} = M_{kp}^{2} = 0$.

б) Решение задачи – частное описание уровня иерархии МГУА с минимальным номером на котором достигается локальный минимум функции $\min_p M_1^{kp}$ при условии, что $M_2^{kp} = 0$.

в) Решение – частное описание уровня иерархии МГУА с минимальным номером на котором достигается локальный минимум функции $\min_p \left\{a_1 M_1^{kp} + a_2 M_2^{kp}\right\}$ $a_1,\ a_2 > 0$ – весовые коэффициенты.

Примечание. Возможна ситуация при которой существуют несколько наборов альтернатив

$\left\{A_{i_1}^e, \ldots A_{i_{ke}}^e\right\}$ $e = \overline{1, L}$ $\forall A_{ij}^e \in \{A_1, \ldots, A_m\}$, для каждой из которых выполняется:

1) имеют общие альтернативы;

2) Каждому набору альтернатив соответствуют хорошо согласованные матрицы парных сравнений последнего уровня дерева иерархий;

3) для любых двух произвольных наборов альтернатив существует не менее одного критерия предпоследнего уровня иерархии МАИ (рис. 1); не менее одной общей альтернативы, веса которой, найденные по двум соответствующим матрицам парных сравнений являются различными. Т.е. присутствует эффект реверса весов альтернатив.

В этом случае в МГУАиАИ предлагается в качестве обучающей последовательности использовать поочередно все наборы ($e = \overline{1, L}$) альтернатив, а решение задачи соответствует тому набору альтернатив, на котором достигается наилучшее значение критерия остановки, определяемое по трем параметрам *к*, M_1^{kp}, M_2^{kp}.

Список используемой литературы

1. *Saaty T.L.* Multycriteric Decision Making. The Analytic Hierarchy Process,

McGraw Hill International. – New York, 1980. Translated to Russian, Portuguese, and Chinese. Revised edition, Paperback. – Pittsburgh, PA: RWS Publications, 1990,1996.
2. *Саати Т., 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. *Ларичев О.И.* Теория и методы принятие решений. – М.:Логос, 2000.
6. *Андрейчиков А.В., Андрейчикова О.Н.* Анализ, синтез, планирование решений в экономике. – Москва: Финансы и статистика. – 2001.
7. *А.А. Павлов, Е.И. Лищук, В.Н. Кут* Математические модели оптимизации для обоснования и нахождения весов в методе парных сравнений. Системні дослідження та інформаційні технології 2007р. №2.
8. *А.А. Павлов, Е.И. Лищук, В.Н. Кут* Математические модели оптимизации для обоснования и нахождения весов объектов по неоднородным матрицам парных сравнений. Системні дослідження та інформаційні технології 2007р. №3
9. *А.А. Павлов, Е.И. Лищук, В.Н. Кут* Многокритериальный выбор в задаче обработки данных матрицы парных сравнений. Вісник НТУУ „КПІ” Інформатика, управління та обчислювальна техніка, Київ 2007р. №46.
10. *Ивахненко А.Г., Мюллер И.А.* Самоорганизация прогнози-рующих моделей. Киев. Техника 1985г.
11. *Ивахненко А.Г.* Моделирование сложных систем. Киев. Высшая школа, 1997г.
12. *Ивахненко А.Г., Юрочковский Ю.В.* Моделирование сложных систем по экспериментальным данным М. Радио и связь 1986 г.

УДК 519.854.2

ПАВЛОВ О.А.,
МІСЮРА О.Б.,
ХАЛУС О.А.,
БЕНЬКОВСЬКИЙ С.Б.,
КОСТИК Д.Ю.,
ЛИСЕЦЬКИЙ Т.М.

СИСТЕМА МОДЕЛЮВАННЯ ДЛЯ ДОСЛІДЖЕННЯ ЕФЕКТИВНОСТІ ПДС-АЛГОРИТМУ ЗАДАЧІ МІНІМІЗАЦІЇ СУМАРНОГО ЗАПІЗНЕННЯ ВИКОНАННЯ ЗАВДАНЬ

Розроблено систему моделювання для дослідження ефективності ПДС-алгоритму (алгоритм із поліноміальною й експоненційною складовими) задачі мінімізації сумарного запізнення виконання завдань, яка відноситься до NP- складних; методику дослідження задачі, а також генератор довільних задач, які охоплюють усі підкласи задачі мінімізації сумарного запізнення (МСЗ). Побудовано таку схему дослідження ефективності ПДС-алгоритму, яка дозволяє отримати статистичні характеристикі розв'язку задачі поліноміальною та експоненційною складовими (оцінки трудомісткості, відсоток задач, що розв'язуються поліноміально, оцінка реального часу розв'язання, відсоток задач, що розв'язуються реальним повним перебором).

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.

Огляд відомих методів розв'язання задачі МСЗ 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, що неможливо існуючими методами [2]. В процесі розв'язання задачі перевіряються умови [1], при виконанні яких оптимальний розклад досягається за поліноміальний час (поліноміа-

льна складова алгоритму). При невиконанні умов задача розв'язується експоненціальною складовою алгоритму.

"Мінімізація сумарного запізнення виконання незалежних завдань з директивними строками одним приладом"

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 $j=1\ldots n$. 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.

Концепція моделювання

У загальному вигляді підхід до моделювання роботи алгоритмів, як і будь-яких інших процесів, виглядає наступним чином [3, 4].

Побудовано модель із підсистемою перевірки адекватності, яка моделює роботу досліджуваної задачі в усіх аспектах, що цікавлять дослідника. Для встановлення зв'язку моделі із реальною ситуацією, розроблено генератор задач, який відтворює всі сторони реальних задач, які обробляються реальним об'єктом, і створює таку їх послідовність, яка охоплює усі підкласи задачі МСЗ.

Іншою підсистемою є підсистема аналізу. Вона використовується для статистичної, прогностичної та іншої обробки результатів роботи моделі, створюючи на її основі аналітичні висновки самостійно або друкуючи звіти для аналітиків, які досліджують роботу ПДС-алгоритму методом його моделювання.

Запропонована концепція є загально-дослідницькою і може використовуватися для імітаційного дослідження у практичних умовах для ширшого класу задач (наприклад, при умові наявності ваг завдань). Її реалізація як системи є досить простою і з незначними доповненнями дає значний синергетичний ефект у експериментальних дослідженнях.

Основні принципи системи моделювання:

- перевірка адекватності протягом циклу життя моделювання і імітації;
- результати не розглядаються як абсолютно правильні або абсолютно хибні;
- система моделювання будується відносно імітації роботи алгоритму, тому довіра до неї оцінюється по якості алгоритму;
- система моделює складний алгоритм і вимагає творчого потенціалу і проникнення в проблему;

- закінчене дослідження в системі має бути можливе;
- для прийнятності і довіри до результатів моделювання проблема формулюється методами системи.

Для забезпечення необхідної якості моделювання і імітації необхідні зусилля направлені на вимірювання і оцінку різноманітних показників якості таких як точність, ефективність виконання, практична значимість генерованих задач, їх взаємозамінність, можливість багатократного використання алгоритму, і застосовність (узгодження людина - комп'ютер).

Структура системи моделювання

Структура системи моделювання пропонує користувачам-дослідникам спілкуватися із системою через інтерфейс (ГІК), який висуває на передній план функціональні налаштування та широкий доступ до будь-якої інформації, яка створюється у системі, а також дає можливість отримувати прості статистичні звіти по розв’язуванню тих чи інших класів задач мінімізації сумарного запізнення.

Реалізований варіант є ефективним дослідницьким кіоском, що дає можливість оперативно отримувати інформацію про якість роботи ПДС-алгоритму на всіх можливих, або конкретних підкласах задач і направляти дослідження у тих напрямках, які виявлятимуть нові особливості розкладів.

Система має оформлений інтерфейс користувача, інтегроване середовище статистики та паралельне розв’язування множини задач. Можливість одночасного запуску кількох копій програми алгоритму робить систему моделювання такою, що оперативніше реагує на зміну вимог до задач та конструює розв’язки у режимі он-лайн.

Особливу увагу в системі приділено формуванню автоматизованого банку інформації за різноманітними класами задач з наступними вимогами:

- можливість опису і реорганізації файлів за допомогою інтерфейсу системи моделювання;
- можливість колективного використання файлів;
- забезпечення нагромадження, зберігання і видачі файлів із постановкою задачі, її розв’язком і статистичним звітом отримання такого розв’язку;
- забезпечення необхідного сервісу під час доступу до файлів, їх редагуванні, видача повних довідок.
- Завдання планування експериментів у межах системи моделювання полягає в отриманні максимальної інформації засобами імітації. Перевагами розробленої системи є:
- простота проведення експериментів;
- можливість управління експериментами, включаючи їх переривання і відновлення;

- легкість варіювання умовами проведення експериментів;
- відсутність кореляції між послідовністю генерованих задач, що досягається застосуванням незв'язних послідовностей псевдовипадкових чисел;
- визначення тривалості роботи алгоритму.

Загальний підхід у схемі дослідження

Система моделювання використовується для дослідження алгоритму розв'язання задачі МСЗ для одного приладу.

Методика вибору значень параметрів вхідних задач і їх розмірностей ґрунтується на методиці проведення наукового дослідження для підтвердження результатів, викладеній у [5].

Процедура дослідження алгоритму виглядає наступним чином.

1. Генеруємо завдання.
2. Розв'язуємо задачу.
3. Формуємо вихідний текстовий файл.
4. Будуємо задану кількість інших розкладів таким же чином.
5. Аналізуємо вихідні файли за різних параметрів та робимо висновки.

Генерація завдань проходить за схемою запропонованою Д. Фішером у 1976 році [6] і традиційно використовується у літературі з тих пір для перевірки різноманітних алгоритмів. Серед наукової спільноти домінує думка про універсальність такої схеми, а результати отримувані на задачах, генерованих з однаковими параметрами, вважаються таким, що отримані на однакових задачах.

Задачі різного ступеню складності генеруються із послідовності за допомогою двох факторів: фактору запізнення (середній коефіцієнт) T і діапазону директивних строків R.

Для кожного завдання j із множини розмірності n генерується цілий час виконання p_j із діапазону рівномірного розподілу із параметрами $[p_{min}, p_{max}]$.

Відносний діапазон директивних строків R і середній коефіцієнт запізнювання T вибираються із набору {2,4,6,8,1}. Цілий директивний строк d_j генерується із діапазону рівномірного розподілу

$$\left[P\left(1 - T - R/2\right)\ P\left(1 - T + R/2\right)\right],$$

де P – час виконання усіх завдань $P = \sum_{j=1}^{n} p_j$.

Унікальний підхід у схемі дослідження

Унікальність підходу у схемі дослідження полягає у специфічному використанні схеми довільної генерації задач, яка охоплює усі підкласи задачі МСЗ. Ця специфіка викликана необхідністю побудувати таку схему

дослідження ефективності ПДС-алгоритму, яка дозволить отримати статистичні характеристики розв’язку задачі поліноміальною та експоненційною складовими:

- оцінки трудомісткості;
- відсоток задач, що розв’язуються поліноміально;
- статистична оцінка реальної складності обчислення поліміальної складової алгоритму;
- відсоток задач, що розв’язуються декомпозиційною складовою алгоритму;
- статистична оцінка реальної складності обчислення декомпозиційної складової алгоритму;
- відсоток задач, які вирішуються реально повним перебором;
- відсоток задач, які мають верхньою оцінкою експоненту, але реально вирішуються за коротший час.

Методологія дослідження складена з двох методів: динамічного і формального.

Динамічний метод призначений для оцінки алгоритму, заснованої на поведінці алгоритму при розв’язанні задач МСЗ. Цей метод вимагає вимірювань у системі моделювання.

З метою виявлення можливої залежності приналежності генерованих задач до певних цікавих для нас класів задач (наприклад, задач, які розв’язуються за поліноміальний час) від значень параметрів генератора, проведено серію досліджень характеристик генерованих задач.

Формальний метод заснований на математичному доказі правильності ПДС-алгоритму: формалізації оптимальності сконструйованих розкладів.

Методологічною основою дослідження є діалектичний метод пізнання і системний підхід. У процесі дослідницької роботи застосовані сукупність методів імовірнісно-статистичного аналізу, методи аналізу і синтезу інформації, наукового експерименту, імітаційного моделювання та інноваційної концепції.

Висновок

Розроблено систему моделювання для дослідження ефективності ПДС-алгоритму для задачі мінімізації сумарного запізнення для одного приладу. Обґрунтовано концепцію моделювання та програмно її реалізовано, що дозволяє застосовувати її для експериментального дослідження роботи ПДС-алгоритму на задачах великої розмірності з метою аналізу отриманих розв’язків. Розроблено методику використання системи моделювання для дослідження ефективності роботи ПДС-алгоритму, яка полягала у генеруванні 200 задач кожної із розмірностей, їх розв'язанні алгоритмом та зборі статистичної інформації.

Список посилань

1. Павлов А.А., Теленик С.Ф. Информационные технологии и алгоритмизация в управлении.– К.: Техника.– 2002.– 344 с.
2. Tansel B.C., Kara B.Y., Sabuncuoglu I. An Efficient Algorithm for the Single Machine Total Tardiness Problem: IEEE Transactions, August 2001 33-8-661
3. Математическое моделирование в исследовании процессов: Методы математич. моделирования и решения процессных задач/ И.М. Федоткин, И.Ю. Бурляй, Н.А. Рюмшин. – К.: Техніка, 2002. – 407 с.
4. Моделирование структуры статистических систем и процессов / Л.Н. Сергеева. – Запорожье, 2002. – 87 с.
5. Шейко В., Кушнаренко Н. Організація та методика науково-дослідницької діяльності: Підручник. – К.: Знання-Прес, 2002. – 295 с.
6. Fisher M.L., A dual algorithm for the one-machine scheduling problem // Math. Programming 11, 1976 – pp. 229–251

УДК 681.3

СЕРГІЄНКО А.М.

СИНТЕЗ СТРУКТУР ДЛЯ ВИКОНАННЯ ПЕРІОДИЧНИХ АЛГОРИТМІВ З ОПЕРАТОРАМИ КЕРУВАННЯ

Запропоновано метод відображення алгоритму, заданого умовним графом синхронних потоків даних, в конвейерний обчислювальний пристрій, що описаний на мові VHDL. Приведено приклад синтезу пристрою для обчислення модуля комплексних чисел, який реалізовано в програмованій логічній інтегральній схемі з високою пропускною спроможністю і малими апаратними витратами.

A method of mapping of Boolean SDF graph into pipelined datapath which is described by VHDL program. A synthesis example of the complex number magnitude calculator illustrates the method. This calculator is configured in FPGA and provides both high throughput and small hardware volume.

До періодичних алгоритмів належать алгоритми, що виконують один і той самий алгоритм з повторенням для різних груп вхідних даних. До них належать алгоритми цифрової обробки сигналів, керування, вирішення задач лінійної алгебри, тощо. Такі алгоритми прийнято задавати у вигляді графів потоків даних. Використання графа синхронних потоків даних (ГСПД) дає змогу формально будувати на його основі оптимальну обчислювальну схему зі статичним розкладом виконання алгоритму [1]. В роботах [2,3] запропоновано метод відображення ГСПД, представленого в просторі ресурси – час у вигляді конфігурації алгоритму (КА), в структуру обчислювального пристрою на рівні регістрових передач. Суть метода полягає в тому, що актор графа – оператор алгоритму, якому відповідає вектор $\boldsymbol{K}_{p,q}$ в КА – відображається в *p*-у вершину процесорного елемента (ПЕ) і в деякий такт свого виконання. В даній статті цей метод вдосконалений, щоб відображати алгоритми з операторами керування.

Для представлення і відображення алгоритмів з умовними розгалуженнями в роботі [4] були запроваджені умовні ГСПД (Boolean, conditional SDF graphs), які відрізняються від звичайних ГСПД тим, що вони мають декілька паралельних (альтернативних) гілок (див. рис. 1,а). Ці гілки починаються в вершині перемикача SW і сходяться в вершині селектора SEL, причому одна з них активізується тільки тоді, коли актор умови *T* є істинним а інша – коли хибним.

При інтерпретації ГСПД, тобто при моделюванні алгоритму, кожна вершина активізується, якщо на її входах є певна кількість токенів, що асоціюються з даними, після чого вона видає на свої виходи токени – результати. Статичний розклад для ГСПД можна скласти, якщо граф є узгодженим, тобто якщо при його інтерпретації кількість токенів на дугах залишається постійною і не виникає блокування. В роботі [4] доведено,

що в загальному випадку за жений час перевірити те, що умовний ГСПД є узгодженим, неможливо.

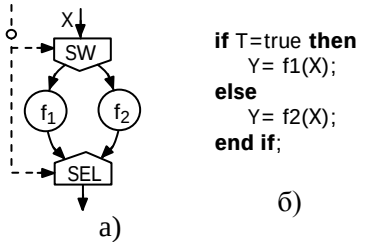


Рис.1. Фрагмент умовного ГСПД (а) і відповідний йому умовний оператор (б)

Квазістатичним називається такий розклад ГСПД, коли момент запуску деякого актора знаходиться динамічно, тобто під час інтерпретації ГСПД, але запуски решти акторів можна знайти за статичним розкладом. В роботі [4] показано, що для умовного ГСПД, у якого альтернативні гілки відповідають оператору if-then-else (див. рис.1,б), або оператору case з усіма можливими комбінаціями умови селектора, можна скласти квазістатичний розклад. При цьому, наприклад, моменти запусків акторів f_1, f_2 (рис.1,а) знаходяться динамічно в залежності від токена T, а запуск актора SEL знаходиться з урахуванням максимальної з затримок виконання акторів f_1, f_2. Таким чином, враховуючи вищеназвані особливості умовного ГСПД, можна будувати КА з умовними операторами, що буде коректно відображатись в структуру обчислювального пристрою (ОП) і його розклад роботи.

Умовний ГСПД з квазістатичним розкладом взаємно однозначно відповідає деякій VHDL–програмі, в якій та чи інша гілка алгоритму вибирається за оператором if-then-else або case. При цьому можна наполягти, щоб логічна умова цих операторів не була константою, або виразом, що спрощується до константи, а в гілках операторів були розглянуті всі без винятку логічні альтернативи. При таких умовах одержимо узгоджений умовний ГСПД, як і в [4]. При відображенні програми на VHDL в умовний ГСПД слід дотримуватись таких самих правил відображення, що представлені в [3]. Доповненням є те, що вершиною перемикача є вершина, яка генерує відповідний операнд, альтернативам в операторах if-then-else, case ставляться у відповідність вершини операцій, що виконуються при визначених умовах, а фразі end if або end case відповідає вершина селектора, в яку зливаються альтернативні гілки.

При синтезі ОП з опису на мові VHDL оператори if-then-else або case можуть відображатися з розділенням ресурсів [3]. Якщо при відображенні умовного ГСПД оператори присвоювання, що стоять в альтернативах логічних операторів будуть розділяти апаратні ресурси, то по-перше, можна одержати статичний розклад, по-друге, апаратні витрати в результуючому ОП будуть мінімізованими. Для цього доведемо наступне твердження.

Твердження. Якщо в умовному ГСПД підграфи умовних розгалужень відповідають операторам if-then-else або case, в гілках яких розглянуті всі без винятку логічні альтернативи, які відображаються в єдиний апаратний ресурс, то для такого ГСПД можна скласти статичний розклад.

Доведення. Виходячи зі згаданого вище, даний ГСПД має квазістатичний розклад. Якщо крім цього можна визначити моменти запуску акторів для кожної з альтернативних гілок до інтерпретації ГСПД, то тоді розклад буде статичним. Дійсно, за умовою, коли всі без винятку актори в альтернативних гілках відображаються в єдиний апаратний ресурс, всім цим акторам можна призначити один і той самий момент запуску. При цьому вершина селектора буде вибирати результат того актора, для якого токен-селектор буде істинним. Через те, що в альтернативних гілках розглянуті всі без винятку логічні альтернативи, при будь-якому значенні токена-селектора на відповідному вході вершини селектора завжди буде з"являтись рівно один токен результату. Це є умовою того, що в ГСПД не виникає ні блокування, ні накопичення токенів у вхідних дугах вершини селектора, тобто ГСПД є узгодженим. Так як актори в альтернативних гілках відображаються в єдиний апаратний ресурс, тобто ПЕ, а не в два або більше ПЕ, то це є умовою мінімізації апаратних витрат.

Розглянемо метод синтезу структур ОП для виконання періодичних алгоритмів з операторами керування, що оснований на методі відображення КА в структуру конвейєрного ОП. Начальними даними для синтезу структури ОП приймаються умовний ГСПД алгоритму та період його виконання L. Процес одержання ефективної конфігурації алгоритму має три етапи.

На першому етапі вершини-оператори 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 виконується мінімізація числа ПЕ шляхом виконання вимог $|\boldsymbol{K}_{p,q}|\rightarrow L$, тобто число вершин, що відображаються в один ПЕ, прямує до L. Крім того, при формуванні КА бажано будувати досконалий кістяк ГСПД, як це пропонується в [5].

При розташуванні логічних операцій, вершині селектора ставиться у відповідність вершина-оператор мультиплексора, до входів якого під'єднуються кінці альтернативних гілок. Цей мультиплексор керується сигналом, що приходить по керуючій дузі з вершини, що генерує логічний сигнал. Початки альтернативних гілок під'єднуються до відповідних вершин – джерел даних для цих гілок.

На другому етапі мінімізуються оператори в альтернативних гілках. Так як згідно з вищеприведеним твердженням оператори з альтернативних гілок відображаються в один і той самий ПЕ, а також виконуються в одному такті, то вони повинні мати однакові координати, тобто їх можна зклеїти в одну вершину. При цьому вершина оператора після зклеювання повинна мати входи як для операндів, так і для керуючого сигнала, що перемикає функції оператора у відповідності з виконанням альтернативних гілок, а вершина мультиплексора альтернатив зникає.

Слід відмітити, що розділяти один і той самий ресурс можуть не тільки вершини-оператори, що стоять в одному ярусі альтернативних

гілок (з однаковими часовими координатами), а і взяті з різних ярусів. Більш того, це можуть бути вершини-оператори, що належать альтернативним гілкам з різних підконфігурацій КА. Але умови виконання цих гілок не повинні перекриватись, щоб не було випадків, коли вершини, що зливаються, можуть виконуватись одночасно. Треба додати, що вказані випадки злиття вершин є можливими, але можуть бути недоцільними через те, що для перетвореної КА не можна буде скласти ефективний розклад. Також злиття вершин може не привести до зменшення апаратних витрат, якщо ці вершини відповідають різнорідним операціям.

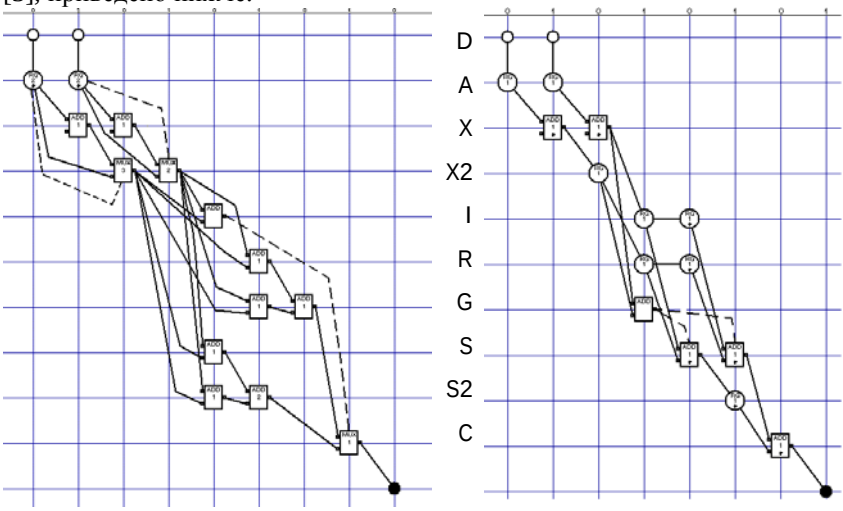
На третьому етапі виконується врівноважування КА. Після цього виконується оптимізація КА шляхом взаємних перестановок векторів-вершин з одного ярусу з метою мінімізації числа регістрів та числа входів мультиплексорів в результуючій структурі і/або з застосуванням інших стратегій, наприклад, ресинхронізації [3,6]. Крім того, положення вершин-мультиплексорів може бути оптимізоване таким самим чином, як і положення інших вершин, вихідні дуги з них можуть мати довільну часову складову, в тому числі і нульову. В останьому випадку вони не відображатимуться в мультиплексор з регістром і їх можна буде мінімізувати в структурі ОП шляхом злиття з вхідним мультиплексором ПЕ, що стоїть після нього. Одержана оптимізована КА відображається в граф структури ОП шляхом склеювання векторів-вершин з одинаковими координатами *k, l*. КА перетворюється в розклад виконання операторів, використовуючи ту властивість, що момент виконання оператору, що відповідає вектору $\boldsymbol{K}_i = \langle k_i, l_i, t_i \rangle$, дорівнює t_i. При цьому можна не будувати структуру і розклад, якщо зразу описати ОП на мові VHDL [3].

Розглянемо застосування методу на прикладі синтезу ОП для обчислення модуля комплексного вектора $c = |a+jb|$ за алгоритмом Понселе [7]. Цей алгоритм полягає в кусково-лінійній апроксимації функції двох змінних в залежності від відношення амплітуд значень *a,b* і може бути представлений наступною VHDL-програмою:

```
    entity MAGNITUDE is
        generic(n: integer:=16);        – розрядність даних
        port (    A: in INTEGER range -2**n to 2**n-1;
                  B: in INTEGER range  -2**n to 2**n-1;
                  C: out INTEGER range  0 to -1+2**n+ 2**n);
end MAGNITUDE;
architecture MAGN_BEH of MAGNITUDE is
        signal Z,R,I : integer        range -2**n to 2**n-1:=0;
    begin
        process(A,B) begin
                  if A<0 then  – обчислення абсолютної величини операндів
                            R<=-A; – в залежності від їхніх знакових розрядів
                  else
                            R<=A;
                  end if;
```

```
                if B<0 then
                        I<=-B;
                else
                        I<=B;
                end if;
                if R<I then              – визначення більшого з операндів
                  C<=(I + R/4) -(I/32-R/8);    – апроксимація по Понселе
                else  – ділення на 4,8,32 виконується шляхом зсуву даного
                  C<=(R + I/4) -(R/32-I/8);
            end if;
        end process;
    end MAGN_BEH;
```

Нехай вхідні дані *a* і *b* приходять по одній вхідній шині послідовно і алгоритм виконується з періодом $L = 2$ такти синхросерії CLK в конвейєрному режимі. На першому етапі будується КА, що відповідає цьому алгоритму, яка показана на рис. 2,а. Слід відмітити, що на рисунку дуги, що відповідають керуванню мультиплексорами, зображені пунктиром, а зсув операндів, тобто ділення на 4,8,32 – не відображений. На другому етапі виконується розділення ресурсів вершин-суматорів. І нарешті, після третього етапу будується урівноважена оптимізована КА, що показана на рис. 2,б. Опис результуючого ОП, одержаний з КА за методикою [3], приведено нижче.



а) б)

Рис.2. Начальна (а) і кінцева (б) конфігурація алгоритму Понселе

```
entity MAGNITUDE2 is generic(n: integer:=16); – розрядність даних
    port (CLK: in BIT;
          RST:in BIT;                          – начальне встановлення
          D: in INTEGER range -2**(n-1) to 2**(n-1)-1;
```

```
                C: out INTEGER range  0 to -1+2**n);
end MAGNITUDE2;
architecture MAGN_SYNT of MAGNITUDE2 is
        signal X,X2,R,I,A,S,S2 : integer range -2**n to 2**n-1:=0;
        signal ct: bit:='0';
        signal G:boolean:=false;
   begin
  CT2:process(CLK,RST) begin  – лічильник тактів
            if CLK='1' and CLK'event then
                    if RST='1' then              ct<='0';
                    else  ct<=not ct ;
                    end if;
            end if;
        end process;
MAIN:process(CLK) begin
            if CLK='1' and CLK'event then
                    A<=D; – вхідний регістр
                    if A<0 then
                       X<=0-A;    – абсолютна величина
                    else
                       X<=A;
                    end if;
                    case ct is  – те, що виконується в парних і непарних тактах
                    when '0' =>X2<=X;     –  парний такт
                            if G = true then
                                S<= R + I/4;
                            else
                                S<= R/4 +I;
                            end if;
                            C<=S2 – S;          – регістр результата
                    when others=>  I<=X;   – непарний такт
                            if (X2 >X) then
                                G<= true;
                            else
                                G<=false;
                            end if;
                            R<=X2;
                            S2<=S;
                            if G = true then
                                S<= R/32-I/8;
                            else
                                S<= I/32-R/8;
                            end if;
                    end case;
            end if;
        end process;
   end MAGN_SYNT;
```

Результуючий ОП MAGNITUDE2 для 16-розрядних даних компілятором-синтезатором Synplify перетворюється в схему на рівні вентилів для

ПЛІС Virtex-4 фірми Xilinx з апаратними витратами 122 LUT, яка може виконувати розрахунки в конвейерному режимі з тактовою частотою до 200 МГц. При включенні режиму ресинхронізації компілятор Synplify дає схему, що працює з частотою до 235 МГц. Для порівняння, ОП MAGNITUDE, що виконує начальний алгоритм, після конфігурації в ПЛІС має апаратні витрати 219 LUT і тактову частоту лише 100 МГц. В порівнянні з аналогічним ОП, що також виконує алгоритм Понселе [7], цей ОП має більшу точність (менше 4%) і побудований за формальним методом.

В даній статті запропоновано метод синтезу структур для виконання періодичних алгоритмів з операторами керування, що є удосконаленням методу відображення графу синхронних потоків даних, представленого у вигляді конфігурації алгоритму. Метод дає змогу формально відображати алгоритми з операторами керування в структуру конвейерних обчислювальних пристроїв з заданим періодом обчислень, що мають мінімізовані апаратні витрати і високу тактову частоту. При цьому через те, що результатом є опис ОП на VHDL, метод дає змогу не будувати власне структуру ОП і розклад виконання алгоритму, а перекласти це завдання на компілятор-синтезатор. Наведений приклад синтезу ОП для обчислення модуля комплексного числа доводить високу ефективність метода.

Список посилань

1. Bhattacharyya S.S., Leupers R., Marwedel P. Software Synthesis and Code Generation for Signal Processing Systems // IEEE Trans. on Circuits and Systems—II: Analog and Digital Signal Processing, – 2000. –V47. –№9. –p.849-875.
2. 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.
3. Сергиенко А.М. VHDL для проектирования вычислительных устройств. – Киев: Диасофт. –2002. – 210 с.
4. 4.J. T. Buck. Scheduling Dynamic Dataflow Graphs with Bounded Memory using the Token Flow Model. PhD thesis, Dept. El. Engineering and Computer Sci., Berkeley Univ. -1993.-173p.
5. Сергієнко А.М. Досконалий кістяк графа алгоритма// Вістник НТУУ"КПІ". Інформатика, управління та обчислювальна техніка. -2007. -№46. –с.62-67.
6. Сергиєнко А.М., Симоненко В.П. Отображение периодических алгоритмов в программируемые логические интегральные схемы //Электронное моделирование. –Т.29. -2007. -№2. –с.49-62.
7. Каневский Ю.С., Клименко М.К., Сергиенко А.М. Устройство для вычисления функции $\sqrt{Y^2+X^2}$. А.С. СССР N 1541601, Б.И. №5, 1990.

УДК 681.322

СЕРГІЄНКО А.М.,
ЛЕПЕХА В.Л.,
ЛЕСИК Т.М.

СПЕЦПРОЦЕСОРИ ДЛЯ ДВОВИМІРНОГО ДИСКРЕТНОГО КОСИНУСНОГО ПЕРЕТВОРЕННЯ

Описано хід розробки спецпроцесорів для прямого і оберненого двовимірного дискретного косинусного перетворення з використанням метода синтезу конвейерних обчислювальних пристроїв. Показані переваги цих процесорів як за швидкодією, так і за апаратними витратами при їхній реалізації в програмованих логічних інтегральних схемах.

The process of development of 2-d DCT and IDCT processors using the method of pipeline processor synthesis is described. The advantages of these processors both due to its speed and due to its hardware volume by their implementation in FPGA are proven.

Двовимірне дискретне косинусне перетворення (ДКП) виконує відображення масиву дійсних чисел розмірами $n \cdot n$ в масив косинусного спектру розмірами $n \cdot n$. Пара двовимірних ДКП і інверсного ДКП (ІДКП) використовується для ущільнення зображень за стандартами JPEG і MPEG, для яких $n = 8$. За звичайним алгоритмом двовимірне ДКП обчислюється як n-точкове ДКП над стовпчиками начальної матриці, а потім – n-точкове ДКП над строчками матриці проміжних результатів. Так, восьмиточкове ДКП обчислюється за формулами:

$$Y(0) = \frac{1}{\sqrt{8}} \sum_{m=0}^{7} X(m), \quad Y(k) = \frac{1}{2} \sum_{m=0}^{7} X(m) \cos \frac{(2m+1)k\pi}{16}, \quad k = 1,2,...,7, \qquad (1)$$

де $X(m)$ – начальні дані, $Y(k)$ - k–й коефіцієнт косинусного спектру.

Восьмиточкове ІДКП обчислюється за формулою:

$$X(m) = \frac{1}{2} \sum_{k=0}^{7} Y(k) \cos \frac{(2m+1)k\pi}{16}, \quad m = 0,1,...,7, \qquad (2)$$

де $Y(k)$ – k-й коефіцієнт косинусного спектру, $X(m)$ – m-й результат ІДКП.

Авторами були розроблені спецпроцесори для ДКП і ІДКП. Метою розробки було створення процесорів, що конфігуруються в програмованій логічній інтегральній схемі (ПЛІС) і забезпечують максимальну пропускну спроможність при умові, що дані поступають в кожному такті. Іншою метою була перевірка ефективності методики синтезу конвейерних обчислювальних пристроїв [2,3].

Структури процесорів складаються з вхідної буферної пам’яті, двох блоків конвеєрної обробки, що виконують ДКП або ІДКП розміру 8 і

буферної пам'яті між ними для транспозиції матриці проміжних результатів. Загальний блок керування призначений для генерації необхідних адресних послідовностей і керуючих сигналів.

Швидкі алгоритми обчислення одновимірних ДКП і ІДКП основані на факторизації формул (1) і (2) і в більшості випадків є різновидами алгоритма Чена [1]. Ці алгоритми характеризуються мінімальною кількістю операцій (рекорд – 11 операцій множення). Але через неоднородність графа алгоритма утруднюється побудова конвейерного обчислювача ДКП.

При виборі алгоритма бажано приймати до уваги, що в ПЛІС фірм Xilinx, Altera складності суматора, регістра і двохвходового мультиплексора приблизно однакові, а складність блока множення вдесятеро більша за складність суматора; регістрові затримки SRL16 глибиною до 16 мають складність одного регістра [3]. В результаті, були розроблені алгоритми з 24 множеннями, 44 додаваннями і відніманнями, що виконують 8-точкове ДКП і ІДКП і які мають більш регулярну структуру своїх графів.

Згідно з методом синтезу конвейерних обчислювачів [2], граф синхронних потоків даних (ГСПД) алгоритму представляється в трьохвимірному просторі у вигляді конфігурації алгоритму (КА) $K_G = (K, D, A)$, де K – матриця векторів-вершин $\boldsymbol{K}_i$, що відповідають операторам алгоритма, D – матриця векторів-дуг $\boldsymbol{D}_j$, безпосередніх інформаційних зв'язків між операторами, A – матриця інцидентності ГСПД. В векторі-вершині $\boldsymbol{K}_i = <k_i, s_i, t_i>$ координати k_i, s_i, t_i дорівнюють типу оператора, номеру елементарного процесорного елемента (ПЕ), в якому виконується цей оператор і такту, в якому результат цього оператора записується в регістр.

Пошук оптимального структурного рішення полягає в знаходжденні такої матриці K, яка мінімізує заданий критерій якості. При пошуку ефективних рішень слід керуватись рядом закономірностей про умови коректності КА, розкладу алгоритму, умов мінімізації апаратурних витрат і умов виконання алгоритму в конвейерному режимі з періодом L тактів [2,3] .

Ефективну КА одержують в два етапи. На першому етапі вершини ГСПД разом з дугами розміщуються в просторі як множина векторів $\boldsymbol{K}_i$ і $\boldsymbol{D}_j$ з урахуванням умов коректності КА, тобто формується начальна КА. Мінімізується число ПЕ в структурі через виконання вимоги, коли число вершин, що відображаються в один ПЭ, прямує до L.

На другому етапі виконується врівноваження КА, коли у всі дуги ГСПД включаються проміжні вершини операторів затримки (регістрів). В результаті, вершини-оператори формують яруси, відстань між якими по координаті часу t дорівнює 1 такт. Врівноважена КА оптимізується через взаємні перестановки векторів-вершин з одного яруса, в результаті чого мінімізується число регістрів і входів мультиплексорів.

На рис. 1 показана КА алгоритму ІДКП, що одержана на другому етапі. Формування КА виконувалось за допомогою редактора КА Paredit, розробленого авторами.

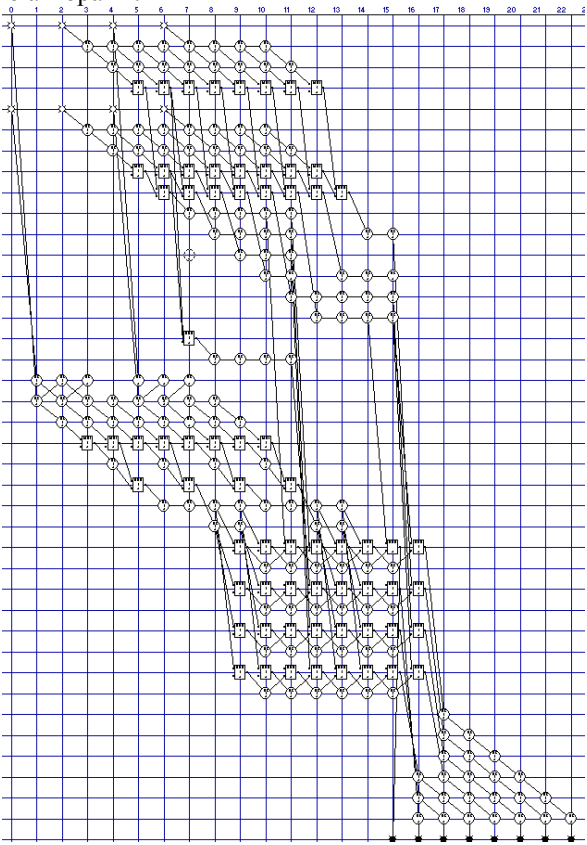


Рис. 1

Конфігурація алгоритму ІДКП

На основі цієї КА за методикою, приведеною в [3], був складений VHDL-опис блоку конвейерної обробки, що був включений в проект процесора. Результати конфігурування процесорів ДКП і ІДКП в різні ПЛІС фірми Xilinx приведені в таблиці 1.

Табл. 1. Апаратні витрати процесорів в кількості еквівалентних конфігурованих логічних блоків (CLB slices) і максимальна тактова частота f_C при реалізації в ПЛІС фірми Xilinx.

Процесор	xc2vp4-7		xc4vlx15-12		xc5vlx30-3	
	CLB slices	f_C, МГц	CLB slices	f_C, МГц	CLB slices	f_C, МГц
ДКП	583	260	549	320	194	400
ІДКП	771	255	705	305	233	400

Процесор ДКП має вхідну розрядність даних – 8 і вихідну розрядність – 12. Максимальна похибка не перевищує 5-кратного значення молодшого розряду в 12-розрядному результаті, що відповідає максимальній похибці 0,25 %. Процесор ІДКП приймає 12-розрядні дані спектру і видає 8-розрядні відновлені дані. Максимальна похибка обчислень не перевищує одного молодшого розряду. Таким чином, розрядність проміжних результатів і коефіцієнтів підібрана такою, щоб забезпечити мінімальні апаратні витрати, максимальну швидкодію при задовільній точності розрахунків.

В табл.2 розроблений процесор ДКП порівнюється з аналогами, які пропонуються відомими фірмами, що виконані на тій самій елементній базі – ПЛІС xc2v-6. В колонках таблиці крім кількості еквівалентних конфігурованих логічних блоків і максимальної тактової частоти f_C вказана кількість блоків множення MPU і блоків пам'яті BRAM. Порівняння процесорів показує екстремально високу швидкодію і низькі апаратні витрати розробленого процесора.

Табл. 2. Характеристики процесорів ДКП

Виробник	CLB slices	MPU	BRAM	f_C, МГц	Примітка
НТУУ"КПІ"	599	6	0	230	
CAST [4]	743	8	1	161	
BARCO [5]	1190	0	2	133	
Xilinx [6]	2295	0	0	141	Розрядність результ. 20

Таким чином, розробка спецпроцесорів ДКП і ІДКП показала високу ефективність метода синтезу конвейерних обчислювальних пристроїв, запропонованого в [2,3]. Ці процесори можна використовувати в усіх нових пристроях для обробки фото- і відео зображень, включаючи такі, для яких потрібна обробка з високою пропускною спроможністю.

Список посилань

1. Liang J., Tran T.D. Fast Multiplierless approximations of the DCT With the Lifting Scheme //IEEE Trans. on Signal Processing. – V49. –№12. –2001. –p.3032-3044.
2. Сергиенко А.М. VHDL для проектирования вычислительных устройств. – Киев: Диасофт. –2002. – 210 с.
3. Сергиенко А.М. Симоненко В.Н. Отображение периодических алгоритмов в ПЛИС // Электрон. моделирование.
4. DCT-FI 2D Forward and Inverse Discrete Cosine Transform Core. - Sept. – 2007.- Available at http://www.cast-inc.com/cores/dct/cast_dct-x.pdf
5. Forward and inverse 2D DCT // BA118DCT/IDCT Factsheet. – 01 March 2005 . – 4p. – Available at www.barcodesignservices.com
6. 2-D Discrete Cosine Transform (DCT). V2.0// Xilinx Inc Product Specification . - March 14, 2002.-11p. -Available at http://www.xilinx.com.

УДК 519.876.5

НЕСТЕРЕНКО Б.Б.,
НОВОТАРСЬКИЙ М.А.

APRO-МЕРЕЖІ ДЛЯ МОДЕЛЮВАННЯ СКЛАДНИХ ДИСКРЕТНИХ СИСТЕМ

Запропонована нова версія формального опису складних дискретних систем - APRO-мережі. Викладено структурні й алгоритмічні особливості APRO-мереж. Наведено приклад опису моделі розподіленої обчислювальної системи, орієнтованої на обробку реального робочого навантаження.

New version of formal description for complex discrete systems (APRO-nets) is offered. Structural and algorithmic features of APRO-nets are stated. The example of model description for distributed computing system focused on processing of real working loading is resulted.

Вступ

Сучасні дискретні системи, у відповідності до загальної теорії систем, задають деякою сукупністю станів, модифікація яких можлива шляхом виконання переходів під дією активностей. Оскільки значна частина пристроїв, машин та механізмів, створених людиною, відповідає означенню дискретних систем, розробці інструментів для їх дослідження приділяють значну увагу. Однією з важливих проблем є створення формальних засобів опису дискретних систем з метою побудови моделей, які б адекватно відображали властивості об’єкта моделювання.

Формальні засоби для створення моделей послідовних систем добре вивчені. Наприклад, теорія скінченних автоматів [1] і подальший її розвиток у вигляді теорії агрегативных систем [2] дозволяють вичерпно описати детерміновані моделі систем, функціонування яких зазвичай задають у вигляді єдиного процесу. Стохастичні моделі систем подібного типу добре представляє теорія масового обслуговування [3]. Однак для адекватного представлення паралельних або розподілених систем семантичні правила формального опису повинні враховувати істотно більше інформації про організацію обчислень. Корисними можуть виявитися дані про те, які процеси розвиваються незалежно, якщо необхідно враховувати причинно-наслідкові зв'язки або робити вибір між альтернативними варіантами.

Для опису паралельних систем широкого застосування набули формальні засоби, представлені різними версіями мереж Петрі [4]. Їх успіх базується на тому факті, що опис об’єкта моделювання множиною станів і алгоритмів обробки уніфікує процес переходу від конкретної моделі до абстрактної, тобто такої, яка дозволяє застосувати формальні процедури аналізу. В результаті накопичення досвіду застосування мереж Петрі по-

ряд з безумовними перевагами поступово проявилися також і їх недоліки, основний з яких полягає у використанні фіксованих правил спрацювання переходів. Незважаючи на те, що доведена функціональна повнота цих правил, застосування їх до опису роботи реальних об'єктів призводить до невиправдано громіздких моделей. Цей факт став причиною виникнення як численних модифікацій мереж Петрі, так і нових мереж. Очевидно, що все розмаїття навколишнього світу неможливо вкласти в прокрустове ложе обмеженої множини алгоритмів, що і є внутрішнім стимулом розвитку теорії мережного моделювання.

Згадані обмеження стали однією з причин подальшого розвитку паралельних засобів формального опису, що розширюють можливості мереж Петрі. Такі формальні засоби одержали назву PRO-мережі [5], яка походить від слів «процеси» та «процесори». В основі PRO-мереж лежить відмова від задавання жорстких правил функціонування переходів за рахунок введення спеціальних процедур, що керують режимами роботи переходів і обробкою інформації. Асинхронні PRO-мережі або APRO-мережі є новим кроком розвитку PRO-мереж і орієнтовані на опис паралельних обчислювальних асинхронних структур.

Структура APRO-мережі

Представимо APRO-мережу у вигляді кортежу:

$$\Phi = (P, T, F, M, V), \qquad (1)$$

де $P = \{ p_i \}_{i=1}^{n}$ – скінченна множина позицій,

$T = \{ t_j \}_{j=1}^{m}$ – скінченна множина переходів,

$F = (P \times T) \cup (T \times P)$ – множина ребер між переходами й позиціями,

$M = \left\{ \left(p_k, \{ \mu_l \}_{l=1}^{Max\_p_k} \right) \right\}_{k=1}^{n}$ – скінченна множина маркувань,

$V = (\Delta, \Psi, \Lambda)$ – множина глобальних змінних.

Подібно графічним позначенням мереж Петрі, позиція APRO-мереж позначається кружком, простий перехід - лінією, зв'язки - лініями зі стрілками, а мітка - крапкою. Додатково введені позначення, пов'язані з елементами операторного переходу. Сам операторний перехід зображується у вигляді прямокутника, а входи й виходи операторного переходу представлені у вигляді правого й лівого півкіл.

Позиції APRO-мережі: $p_i = \{d_i, q_i\}$, де d_i – множина параметрів позиції, q_i – множина міток, розміщених на даній позиції.

Переходи APRO-мережі включають два класи переходів: $t = \{\tau, \theta\}$, де τ – клас простих переходів, θ – клас операторних переходів. Простий перехід описує множина елементів: $\tau_j = \{ \chi_j, N_j \}$, де χ_j – множина параме-

трів переходу, $N_j = \{\rho_j, \pi_j, \gamma_j, \omega_j, A_j, O_j\}$ – функціональне ядро переходу, що включає такі елементи: ρ_j – процедура активації переходу, π_j – процедура обслуговування переходу, γ_j – процедура деактивації переходу, ω_j – процедура очікування, A_j – частково впорядкована послідовність активностей, O_j– частково впорядкована послідовність вихідних міток.

Клас переходів θ забезпечує композиційні властивості мережі і представлений множиною:

$$\theta = (P_\theta, T_\theta, E_\theta, X_\theta, F_\theta) \ , \qquad (2)$$

де $P_\theta = \{(p_\theta)_1, \cdots (p_\theta)_a\}$, $a \in N$ – множина позицій;

$T_\theta = \{(\tau_\theta)_1, \cdots (\tau_\theta)_b\}$, $b \in N$, $T_\theta \neq \varnothing$ – непуста множина переходів;

$E_\theta = \{(e_\theta)_1, \cdots (e_\theta)_c\}$, $c \in N$, $E_\theta \neq \varnothing$ – непуста множина входів;

$X_\theta = \{(x_\theta)_1, \cdots (x_\theta)_d\}$, $d \in N$, $X_\theta \neq \varnothing$ – непуста множина виходів;

$F_\theta = (P_\theta \times T_\theta) \cup (T_\theta \times P_\theta) \cup (E_\theta \times T_\theta) \cup (T_\theta \times X_\theta)$ – множина ребер.

Таким чином, операторний перехід θ містить APRO-мережу Φ_θ, яка є мережею нижнього рівня стосовно мережі Φ. Переходи мережі Φ_θ також можуть бути операторними переходами, що дозволяє будувати вертикально стратифіковані моделі з необмеженою кількістю рівнів.

Щоб уникнути плутанини, елементи операторного переходу будемо позначати індексом з його найменуванням. Формальний опис довільної внутрішньої позиції $(p_\theta)_i \in \Phi_\theta$ операторного переходу θ відповідає формальному опису довільної позиції верхнього рівня $p_i \in \Phi$. Внутрішні переходи $(\tau_\theta)_j \in \Phi_\theta$ також мають ідентичний формальний опис із простими переходами верхнього рівня $\tau_j \in \Phi$.

Входи операторного переходу: $(e_\theta)_j = \left\{ (\eta_\theta)_j, \left(N_\theta^e\right)_j, \mu_\theta \right\}$, де $(\eta_\theta)_j$ – множина параметрів входу переходу θ; $N_\theta^e = \{(\rho_\theta)_j, (w_\theta)_j\}$ – ядро входу, що включає процедуру активації входу $(\rho_\theta)_j$ та процедуру очікування $(w_\theta)_j$; μ_θ – поточна вхідна мітка операторного переходу θ. Виходи операторного переходу: $(x_\theta)_i = \left\{(o_\theta)_i, \left(N_\theta^x\right)_i, \left(q_\theta^x\right)_i\right\}$, де $(o_\theta)_i$ – множина параметрів виходу переходу θ; $(N_\theta)_i = \{(\gamma_\theta)_i\}$ – ядро виходу, яке містить

процедуру деактивації виходу $(\gamma_\theta)_i$; $\left(q_\theta^x\right)_i$ – множина міток на виході операторного переходу θ. Ребра мережі задають матрицею інцидентності H з елементами:

$$H(p_i,t_j)=\begin{cases}-1, & (p_i,t_j)\in \mathbf{F},\\ +1, & (p_i,t_j)\in \mathbf{F}^{-1},\\ 0, & (p_i,t_j)\notin \mathbf{F},(p_i,t_j)\notin \mathbf{F}^{-1}\end{cases}, \quad \begin{matrix}1\le i\le n,\\ 1\le j\le m.\end{matrix}$$

Мітки APRO-мережі: $\mu_k=\{\lambda_k,\alpha_k\}$, де λ_k – множина параметрів мітки, α_k – множина атрибутів мітки. Множина глобальних змінних: V=(Δ, Ψ, Λ), де Δ – підмножина показників продуктивності, Ψ – підмножина показників реактивності, Λ – підмножина показників використання.

Алгоритмічні аспекти функціонування APRO-мережі

Розглянемо основні аспекти семантики APRO-мережі, що базуються на процесах, активностях і подіях. Активності APRO-мережі виражають сукупність типових дій переходу на заданому рівні опису. За умови одночасного використання операторних і простих переходів мережа буде містити активності верхнього рівня й внутрішні активності операторних переходів, утворюючи дворівневу структуру активностей. Якщо операторний перехід нижнього рівня також містить операторні переходи, то кількість рівнів активностей збільшується на одиницю. Таким чином, одержуємо деревоподібну структуру активностей. Логічна послідовність активностей утворює процес. Кожний з процесів, що існують у мережі, може входити в процес більш високого рівня як субпроцес.

Будемо розглядати множину активностей A, що включає такі підмножини: $A=\{compute, activate, deactivate, wait\}$. Підмножина *compute* містить активності, що виконують дії, пов'язані з обробкою інформації під час роботи переходу. Підмножина *activate* представляє активності, що реалізують дії переходу, необхідні для його активації, активності підмножини *deactivate* завершують роботу переходу. Активності, задані підмножиною *wait*, відображають дії переходу в режимі очікування. Упорядкована послідовність активностей може утворювати процес.

Множині переходів $T=\{t_1, \Lambda, t_j, \Lambda, t_m\}$ поставимо у відповідність множину процесів Pr=$\{Pr_1, \Lambda, Pr_j, \Lambda, Pr_m\}$ і задамо підмножини активностей:

$$activate := \{a_j\}_{j=1}^m,\ compute := \{c_j\}_{j=1}^m,\ deactivate := \{d_j\}_{j=1}^m,$$

$$wait := \{w_j\}_{j=1}^m.$$

Тоді процес Pr_j переходу t_j можна представити послідовністю активностей: $Pr_j := \{a_j, c_j, d_j, w_j\}$. Процес Pr_j відображає типовий процес роботи

переходу, що спочатку активується, виконує обробку інформації, а потім завершується й переходить у режим очікування.

Народження й зникнення активностей відбувається шляхом здійснення подій, кожна з яких є миттєвою зміною стану APRO-мережі. Будемо розглядати тільки події *start* та *stop*, що забезпечують виникнення й завершення активностей. Тоді процес Pr_j може бути записаний з урахуванням даних подій:

$$Pr_j := \{w_j.stop, a_j.start, a_j.stop, c_j.start, c_j.stop, d_j.start, d_j.stop, w_j.start\}$$

Сукупний процес Pr_j завжди однозначно задає поточний стан відповідного переходу t_j. Тому загальною ознакою впорядкування послідовності Pr_j виступає час виникнення активностей, що входять у дану послідовність.

На рис.1 показана структура APRO-мережі, що складається із простого переходу τ_j, вхідних позицій $p_a, \cdots, p_b$ і вихідних позицій $p_c, \cdots, p_n$.

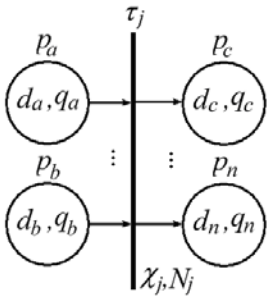


Рис. 1. Структура простого переходу APRO-мережі

Відповідно до опису структури, основними елементами позицій APRO-мережі є параметри стану $d_a, \cdots d_b, d_c, \cdots d_n$ й списки активних міток $q_a, \cdots q_b, q_c, \cdots q_n$. Простий перехід τ_j містить параметри стану χ_j і ядро N_j. Послідовний розвиток процесу Pr_j відбувається за рахунок того, що здійснення поточної події супроводжується створенням нової події з новим часом виконання. Стартова подія завжди запускає нову активність за допомогою виконання відповідної процедури переходу. Структура логічних зв'язків процедур переходу, що входять до складу ядра N_j, показана на рис. 2.

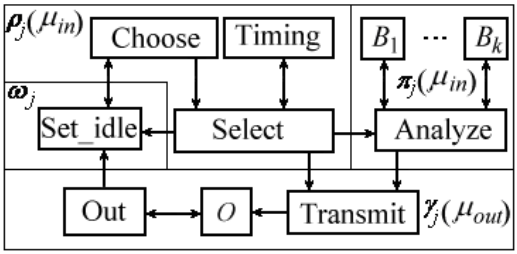


Рис. 2. Структура процедур $\rho_j(\mu_{in})$, $\pi_j(\mu_{in})$, $\gamma_j(\mu_{out})$ *і* ω_j

Процедура активації $\rho_j(\mu_{in})$ забезпечує виконання комплексу операцій, що відповідають активності a_j й складається із процедур Choose, Select і Timing. Процедура Choose призначена для виконання дій, пов'язаних з аналізом міток, розміщених на вхідних позиціях переходу τ_j. Процедура Select реалізує алгоритм перевірки мітки, обраної за допомогою процедури Choose. Одна з важливих умов перевірки полягає в збереженні часової цілісності моделі шляхом порівняння локального часу переходу й часу створення мітки.

Цю функцію виконує окрема процедура Timing. Запуск процедури $\rho_j(\mu_{in})$ відбувається в момент завершення активності w_j і початку активності a_j. Механізм цієї зміни полягає в тому, що завершальна подія $w_j.stop$ активності w_j породжує стартову подію $a_j.start$ активності a_j. Існує чотири можливих завершення процедури $\rho_j(\mu_{in})$ залежно від результатів аналізу вхідної мітки μ_{in}:

1. Якщо пошук активної мітки на вхідних позиціях завершився безрезультатно, то подія завершення активності a_j породжує подію $w_j.start$.

2. Якщо активна мітка μ_{in} знайдена процедурою Choose, але значення атрибутів цієї мітки не дозволяють запустити процедуру обробки $\pi_j(\mu_{in})$, то подія завершення активності a_j також породжує подію $w_j.start$.

3. Якщо активна мітка μ_{in} знайдена процедурою Choose і має коректні параметри, то подія завершення активності a_j породжує подію $c_j.start$.

4. Якщо активна мітка μ_{in}, яка знайдена процедурою Choose, є транзитною міткою, то подія завершення активності a_j породжує подію $d_j.start$.

Процедура $\pi_j(\mu_{in})$ реалізує сукупність операцій, що відповідають активності c_j переходу τ_j й складається із процедури Analyze і множини

процедур $\{B_i(\alpha)|\alpha \in \mu\}_{i=1}^{k}$. Запуск цієї процедури відбувається у випадку, якщо мітка коректна й може бути оброблена в даному переході, що підтверджується створенням події $c_j.start$. Процедура Analyze забезпечує оцінку параметрів поточної мітки μ_{in} з метою вибору відповідної процедури $B_k(\alpha)\big|_{\alpha\in\mu_{in}}$. Робота процедури Analyze завершується подією $c_j.stop$, яка породжує подію $d_j.start$.

Процедура $\gamma_j(\mu_{out})$ включає операції виводу міток із простого переходу τ_j. До складу $\gamma_j(\mu_{out})$ входять процедури Transmit і Out. Існує два варіанти запуску процедури $\gamma_j(\mu_{out})$, кожний з яких ініціюється відповідною подією $d_j.start$. Процедура Transmit формує параметри й атрибути вихідної мітки й розміщує їх у черзі вихідних міток O_j. Процедура Out вибирає першу в черзі вихідну мітку μ_{out} й намагається встановити її на відповідну вихідну позицію. Якщо мітка з тих чи інших причин не встановлена на зазначеній вихідній позиції, то вона знову повертається в чергу вихідних міток. За умови відсутності черги вихідних міток O_j робота процедури $\gamma_j(\mu_{out})$ завершується, про що свідчить подія $d_j.stop$, здійснення якої породжує подію $w_j.start$. Відповідно до рис.2, подія $w_j.start$ завжди запускає процедуру очікування ω_j. Основною складовою даної процедури є процедура Set_idle, що встановлює дії переходу після завершення повного циклу обробки інформації.

Приклад опису APRO-мережі

Розглянемо приклад композиційної побудови APRO-мережної моделі розподіленої обчислювальної системи з використанням операторних переходів (рис. 3).

Об'єкт моделювання включає простий перехід «*distributor*», що виконує функції розподілу обчислювального навантаження між N вузлами, заданими за допомогою множини операторних переходів $\{node\,i\,|\,1 \le i \le N\}$. Формування загального результату із часткових результатів обчислень у вузлах виконує простий перехід «*collector*». Переходи «*distributor*» і «*collector*» реалізують обмін даними між вузлами шляхом використання зв'язків через позиції «*straight*» і «*back*». Перехід «*collector*» також завершує процес моделювання шляхом установки стану «*stop*».

Структурно APRO-мережа $\Phi_d = (P, T, F, M, V)$ включає такі елементи: $P = \{p_1, p_2, \cdots p_{2N+4}\}$,

Позиція $p_1 = \{\{start, data, 1, N, \delta\_\mu_1\}, \{\mu_1\}\}$ містить параметри $d_1 = \{start, data, 1, N, \delta\_\mu_1\}$ і мітки $q_1 = \{\mu_1\}$. Елементи множини параметрів, зокрема, мають такі значення: *start* – назва позиції, *data* – назва типу міток, що можуть розміщуватись на даній позиції, 1– поточна кількість міток на позиції, N – максимальна кількість міток на позиції, δ_μ_1– значення локального лічильника часу, що дорівнює часу створення поточної мітки. За даною схемою сформовані параметри решти позицій:

$p_2 = \{\{input\,1, data, 0,1,0\}, \varnothing\}, \ldots,\ p_{2N} = \{\{input\,N, data, 0,1,0\}, \varnothing\},$

$p_3 = \{\{output\,1, data, 0,1,0\}, \varnothing\}, \ldots,\ p_{2N+1} = \{\{output\,N, data, 0,1,0\}, \varnothing\},$

$p_{2N+2} = \{\{straight, data, 0, N, 0\}, \varnothing\},\ p_{2N+3} = \{\{stop, data, 0, N, 0\}, \varnothing\},$

$p_{2N+4} = \{\{back, data, 0, N, 0\}, \varnothing\},$

$T = \{\tau_1, \tau_2, \theta_1, \cdots \theta_N\}$

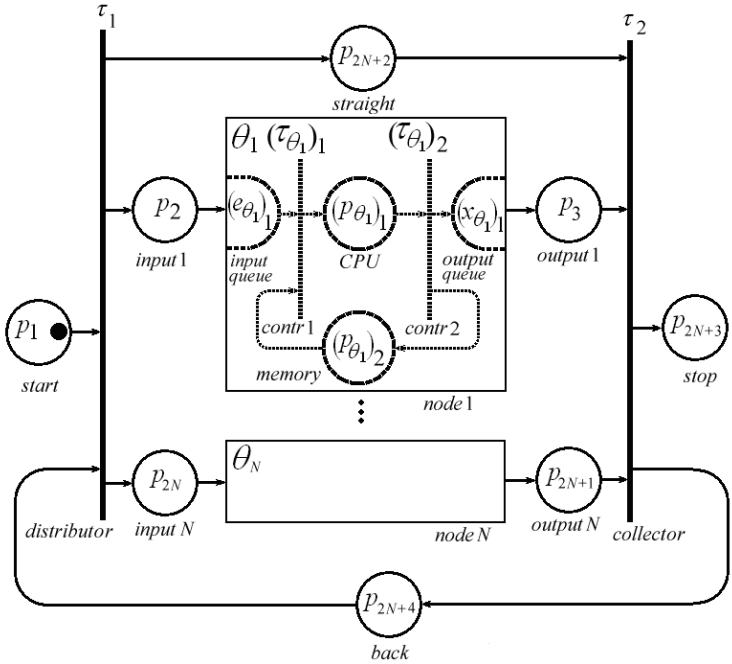


Рис. 3. Модель розподіленої обчислювальної системи

Перехід $\tau_1 = \{\{distributor, 0,0,0\}, N_1\}$ включає параметри $\chi_1 = \{distributor, 0,0,0\}$ і ядро N_1. Елементи множини параметрів мають такі значення: *distributor* – назва переходу, 0 – значення локального лічильника часу переходу, 0 – загальний час простою переходу, 0 – номер рівня переходу. За такими ж правилами сформований перехід

$\tau_2 = \{\{collector, 0,0,0\}, N_2\}$. Операторні переходи $\theta_i = \left(P_{\theta_i}, T_{\theta_i}, E_{\theta_i}, X_{\theta_i}, F_{\theta_i}\right)$, $1 \le i \le N$ містять наступні елементи:

$P_{\theta_i} = \left\{\left(p_{\theta_i}\right)_1, \left(p_{\theta_i}\right)_2\right\}$, $\left(p_{\theta_i}\right)_1 = \{\{CPU, data, 0,1,0\}, \varnothing\}$,

$\left(p_{\theta_i}\right)_2 = \{\{memory, data, 0,1,0\}, \varnothing\}$,

$T_{\theta_i} = \left\{\left(\tau_{\theta_i}\right)_1, \left(\tau_{\theta_i}\right)_2\right\}$, $\left(\tau_{\theta_i}\right)_1 = \left\{\{contr\,1,0,0,1\}, \left(N_{\theta_i}\right)_1\right\}$,

$\left(\tau_{\theta_i}\right)_2 = \left\{\{contr\,2,0,0,1\}, \left(N_{\theta_i}\right)_2\right\}$,

$$E_{\theta_i} = \left\{\left(e_{\theta_i}\right)_1\right\}, \left(e_{\theta_i}\right)_1 = \left\{\{input\ queue, 0,0,0,10,1\}, \left(N^e_{\theta_i}\right)_1, \varnothing\right\},$$

$$X_{\theta_i} = \left\{\left(x_{\theta_i}\right)_1\right\}, \left(x_{\theta_i}\right)_1 = \left\{\{output\ queue, data, 0,10\}, \left(N^x_{\theta_i}\right)_1, \varnothing\right\},$$

$$F_{\theta_i} = \begin{pmatrix} & \left(e_{\theta_i}\right)_1 & \left(p_{\theta_i}\right)_1 & \left(p_{\theta_i}\right)_2 & \left(x_{\theta_i}\right)_1 \\ \left(\tau_{\theta_i}\right)_1 & -1 & 1 & -1 & 0 \\ \left(\tau_{\theta_i}\right)_2 & 0 & -1 & 1 & 1 \end{pmatrix}.$$

$$F = \begin{pmatrix} & p_1 & p_2 & p_3 & \cdots & p_{2N} & p_{2N+1} & p_{2N+2} & p_{2N+3} & p_{2N+4} \\ \tau_1 & -1 & 1 & 0 & \cdots & 1 & 0 & 1 & 0 & -1 \\ \tau_2 & 0 & 0 & -1 & \cdots & 0 & -1 & -1 & 1 & 1 \\ \theta_1 & 0 & -1 & 1 & \cdots & 0 & 0 & 0 & 0 & 0 \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ \theta_N & 0 & 0 & 0 & \cdots & -1 & 1 & 0 & 0 & 0 \end{pmatrix}$$

Запропонована APRO-мережна модель описує дискретну розподілену систему обробки даних. Особливість даного підходу полягає в можливості імітаційного моделювання систем з реальним робочим навантаженням, що обумовлено використанням довільних структур даних в атрибутах міток і реалізацією довільних алгоритмів обробки цих даних за допомогою процедур обробки переходів. APRO-мережний опис є основою вхідної мови задавання параметрів моделей для моделюючого комплексу APRO_Simulator. За допомогою формування масивів статистичних даних відповідно до показників, об'єднаних в множину глобальних змінних V , даний комплекс дозволяє аналізувати різні характеристики складних дискретних систем як на етапі їх розробки, так і на етапі експлуатації.

Висновки

Стрімке ускладнення дискретних систем стимулює пошук нових підходів до створення їх формальних описів, що лежать в основі побудови різного роду моделей. Такі описи повинні мати високий ступінь абстрагування з метою спрощення моделі і одночасно забезпечувати високу адекватність опису об'єктів моделювання для отримання коректних результатів. З урахуванням даних вимог розроблений інструмент формального опису складних дискретних систем – APRO-мережі.

Наукова новизна запропонованої версії мережного формального опису складних дискретних систем обумовлена сукупністю таких властивостей:

- гнучкого механізму ієрархічного представлення моделей;
- можливості опису паралельно функціонуючих і асинхронно взаємодіючих процесів з використанням сучасних підходів до просування модельного часу;
- можливості побудови імітаційних моделей з використанням реального робочого навантаження.
- Практична цінність роботи полягає у тому, що побудовано програмний комплекс, який є середовищем моделювання з вхідною мовою, створеною на основі синтаксису та семантики APRO-мереж.

Список посилань

1. Баранов С.И. Синтез микропрограммных автоматов. – Л.: Энергия, 1979. – 232 с.
2. Бусленко Н.П. Моделирование сложных систем.– М.: Наука, 1978.– 400 с.
3. Гнеденко Б.В., Коваленко И.Н. Введение в теорию массового обслуживания.– М.: Наука, 1987.– 336 с.
4. Питерсон Дж. Теория сетей Петри и моделирование систем.– М.: Мир, 1984.– 264 с.
5. Нестеренко Б.Б., Новотарский М.А. Мультипроцессорные системы.− К.: Институт математики АН Украины, 1995. − 408 с.

УДК 681.3.06

КУЗНЕЦОВ А.В.

СМЫСЛОВАЯ АМОРФНОСТЬ В ГРАММАТИЧЕСКИХ СИГНАТУРАХ ПРЕДЛОЖЕНИЙ ЕСТЕСТВЕННОГО ЯЗЫКА

Рассмотрена технология анализа предложений естественного языка на основе грамматики непосредственных составляющих при раздельном представлении словаря и грамматической сигнатуры. Описана структура модифицированного словаря и смысловые агенты-процедуры, которые связывают лексемы анализируемых предложений с моделью проблемной области знаний и дополняют грамматическое согласование частей речи смысловым согласованием. Показано, что в данном случае грамматическая сигнатура теряет чувствительность к словарному наполнению и обретает свойство смысловой аморфности, сохраняя грамматический строй естественного языка. Такое свойство позволяет в рамках единой грамматической сигнатуры выполнять смысловой машинный анализ предложений с формально заданным стилем написания, но относящихся к разным проблемным областям, которые требуют отличающихся словарей.

Consideration of technology natural language sentences computerized analysis based on definite clause grammar provided separate notation of dictionary and grammar signature is represented. There is describing modified dictionary and agents-procedures of meaning which associate the lexical elements inside sentence with problem area model and supplement the grammar harmonization with semantic harmonization between ones. In this case the grammar signature gets lost sensitivity to word's filling due that acquire amorphous semantics but keep natural language grammar order. Like distinctive feature inside uniform grammar signature provided the computerized analysis of meaning the sentences which have formal definite style for writing but associated with different problem knowledge areas that requires unlike dictionaries.

В прикладной лингвистике широко используется грамматика непосредственных составляющих (ГНС), которая дает возможность структурного описания синтаксических объектов предложений естественного языка на основе деревьев непосредственных составляющих, где ветви и листья ассоциируются с допустимыми списками лексем. В Прологе правильность использования форм таких лексем проверяется не преобразуемыми подцелями, которые заключены в фигурные скобки внутри грамматических правил и содержат необходимые атрибуты для согласования частей речи в предложении. Другими словами, ГНС представляет собой одновременно и словарь и грамматику [1]. Подстановка вместо списков лексем правил обращения к словарю проблемной области (ПО) логически разделяет ГНС на две составные части – словарь и грамматическую сигнатуру, которая, таким образом, теряет чувствительность к словарному наполнению ГНС, но сохраняет грамматический строй выбранного есте-

ственного языка. Такая подстановка дает возможность в пределах единой грамматической сигнатуры (т.е. без изменения грамматических правил) обращаться к разным проблемным словарям одного и того же языка, а, значит, выполнять машинный анализ предложений, которые передают разный смысл или даже относятся к разным ПО, если выполнить необходимые изменения в запрограммированной модели ПО. Т.е. отдельно представленная в ГНС грамматическая сигнатура приобретает свойство смысловой аморфности, и может использоваться как основа для организации обращений к различным моделям ПО на ограниченном естественном языке с формализованным стилем записи предложений.

Следует учесть, что одна из существенных особенностей, касающихся исследования возможностей ГНС, сводится к иллюстрации отдельных примеров, так как ГНС уже представляет собой формализм для описания грамматики ограниченного естественного языка и не требует дополнительных синтаксических надстроек для собственного описания. Таким образом, отображая обобщение ГНС на конструкции Пролога, определим подход к проектированию системы обработки естественного языка, ориентированной на машинный анализ семантики заданного подмножества предложений, которые передают разный смысл, но могут быть представлены единой грамматической сигнатурой.

Абстрагируясь от внутренних разделителей и знаков препинания, схематически зададим состав некоторой группы предложений в виде последовательности нетерминальных символов, перечисляющих части речи:

<прилагатеное$_1$><прилагательное$_2$><существительное$_1$><глагол><существительное$_2$>.

Эта схема соответствует корневой вершине дерева разбора, которая в Прологе задается типовым определением контекстно-свободной грамматики:

```
'предложение'-->'фраза_существительного','фраза_глагола'.
```

В одном из вариантов грамматического разбора такое разбиение предложения ассоциируется с двумя ветвями, исходящими из вершины первого уровня иерархии дерева разбора, а 'фраза_существительного' потребует одного листа (для первого определителя существительного) и ветви, исходящей из вершины второго уровня, где представлен первый остаток списка лексем фразы существительного.

Подобным же образом разбирается и 'фраза_глагола'.

Далее, рассмотрим несколько предложений естественного языка:

```
Смешная рыжая обезьяна держит банан.
Знаменитый талантливый пародист рассмешил публику.
Легкомысленный халатный охранник прозевал грабителя.
Опасный вооруженный гангстер грабит банки.
```

Очевидно, все эти предложения имеют общую характерную черту – они полностью вписываются в приведенную выше схему порядка следования частей речи естественного языка, хотя передают совершенно разный смысл. Для того чтобы в Прологе привести систему машинного анализа этих предложений к единой грамматической сигнатуре, остановимся, прежде всего, на организации словаря, вынесенного за пределы ГНС. Этот словарь потребует некоторой модификации по сравнению со словарем, обычно встроенным в ГНС.

В качестве основного требования к словарю определим общность структур, используемых для представления различных частей речи. Тогда любые факты, задающие требуемый словарь, могут быть записаны в Прологе следующим образом:

<функтор>(<атрибут$_1$>:<атрибут$_2$>: ...:<атрибут$_m$>,[<слово$_1$>,<слово$_2$>, ...,<слово$_n$>]).

В угловых скобках, как всегда в подобных случаях, представлены нетерминальные символы.

Для описания структурных составляющих словаря введем соглашение обозначать совпадающие по назначению атрибуты разных частей речи одними и теми же большими буквами латиницы. В программе на Прологе с больших букв начинаются имена переменных, содержащих значение того или иного атрибута. Что касается представления самих значений используемых атрибутов, то они задаются атомами Пролога, которые ассоциируются с названиями этих значений на естественном языке, но с приемлемыми сокращениями, допускающими сохранение читабельности. Таким образом, в условиях ввода ограничений на естественный язык при построении ГНС, будем использовать атрибуты, перечисленные в табл. 1., а их допустимые значения представлены в табл. 2. Тогда, в соответствии с данными из таблиц 1,2, для грамматического согласования частей речи в предложении будем использовать наборы атрибутов, которые сведены в табл. 3.

Табл.1

	Существительное	Прилагательное	Местоимение	Глагол	Параметр смыслового агента
G – род	×	–	×	×	–
Q – число	×	×	×	×	–
A – падеж	×	×	×	–	–
R – разряд или лицо	–	–	×	×	–
T – время	–	–	–	×	–
V – вид	–	–	–	×	–
D – залог	–	–	–	×	–

	Существительное	Прилагательное	Местоимение	Глагол	Параметр смыслового агента
C – класс применимости	×	–	–	×	–
F – функция применимости	–	–	–	–	×
S – субъект действия	–	–	–	–	×

Табл. 2

G	Q	A	R	T	V
'мужск' 'женск' 'средн' 'неопфм'	'единств' 'мнжеств' 'неопфм'	'именит' 'родит' 'дател' 'винит' 'творит' 'прелож'	'личн'('первое') 'личн'('второе') 'личн'('третье') 'безличн' 'возвр' 'притяж' 'указ' 'опред' 'притяж' 'указ' 'опред' 'вопрос' 'относит' 'отриц' 'неопфм'	'прош' 'наст' 'будущ' 'неопф м'	'сверш' 'несверш'

Табл.2 (продолжение)

D	C	F	S
'действит' 'страдат' 'возврзн' 'взаимнзн' 'неопфм'	<классы>	<функции>	<субъекты>

Табл. 3

Существительное	Прилагательное	Местоимение	Глагол
G:Q:A:C:GB	G:Q:A	G:Q:A:R	G:Q:R:T:V:D:C

Для значений разрядов местоимений и лиц глагола применяются следующие сокращения:

Л и ч н ы е: 'личн'('первое') – первое лицо глагола,
'личн'('второе') – второе лицо глагола,
'личн'('третье') – третье лицо глагола,

'беличн'	– безличные глаголы.
В о з в р а т н о е:	– 'возвр'.
П р и т я ж а т е л ь н ы е:	– 'притяж'.
У к а з а т е л ь н ы е:	– 'указ'.
О п р е д е л и т е л ь н ы е:	– 'опред'.
В о п р о с и т е л ь н ы е:	– 'вопрос'.
О т н о с и т е л ь н ы е:	– 'относит'.
Н е о п р е д е л е н н ы е:	– 'неопфм'.
О т р и ц а т е л ь н ы е:	– 'отриц'.

В дальнейших примерах использована лишь часть лиц глаголов, но здесь перечислены все разряды и лица, чтобы показать возможность полноты представления.

Теперь обратимся к самой сложной части речи – глаголу.

У глагола, помимо типовых признаков – рода, числа и лица, характерных для его связи с другими частями речи, присутствуют время, вид, залог, и класс применимости, для дополнительной смысловой связи с субъектом в предложении.

Наличие у глагола неопределенной формы приводит к необходимости использовать значение `'неопфм'` в атрибутах, определяющих время, число, лицо, род и залог. Обратим также внимание, что при использовании личных местоимений с глаголами значение разряда местоимения и лица глагола должны совпадать (согласование по переменной `R`).

Переменная `D` применяется для хранения значений залога (`'действит'` – действительный, `'страдат'` – страдательный), неопределенной формы (`'неопфм'`) и возможных дополнительных значений (`'возврзн'` – возвратное значение, `'взаимнзн'` – взаимное значение), которые зависят от роли глагола в предложении.

Переменная `C` определяет класс применимости глагола и обеспечивает смысловую связку с существительным, выражающим субъект действия в предложении.

Для пояснения необходимости введения такого атрибута как класс применимости рассмотрим три предложения, совершенно корректные с грамматической точки зрения.

```
Знаменитый клоун держит в доме смешную обезьяну.
Смешная обезьяна держит банан.
Смешная обезьяна читает книгу.
```

Очевидно, первые два предложения с глаголом `'держит'` следует считать семантически правильными, тогда как третье предложение совершенно лишено смысла ввиду отсутствия в человеческой практике грамотных обезьян. Это значит, что глагол `'держит'` применим, например, как к классу `'люди'` так и к классу `'животные'`, тогда как глагол `'читает'`

может быть использован только с классом 'люди'. В терминах Пролога применимость требуемого глагола к одному или нескольким классам описывается в словаре следующим образом.

```
'перех_глаг'('мужск':'единств':'лицн'('третье'):'наст':'несоверш':'действит':C:,
   ['держит'],Lc) :- ['животные', 'люди']=Lc,'принадлежит'(C,Lc).
'перех_глаг'('мужск':'единств':'лицн'('третье'):'наст':'несоверш':
   'действит':'люди',['читает'],['люди']).
'принадлежит'(X,[ X|_ ]).
'принадлежит'(X,[_|T]) :- 'принадлежит'(X,T).
```

При задании словаря глаголов следует также обратить внимание на использование функторов, которые описывают такие свойства глаголов как переходность и непереходность. Переходные глаголы имеют функтор 'перех_глаг', а непереходные – 'неперех_глаг'. Для словарей других частей речи функторы не несут дополнительной смысловой нагрузки и обозначаются как 'словарь_существит' и 'словарь_прилагат'.

В первом определении структуры с функтором 'перех_глаг' на месте обозначения класса использована переменная C, которая в ходе анализа предложения унифицируется некоторым значением, соответствующим названию класса существительного (субъекта действия в предложении). Тогда смысловая правильность использования глагола 'держит' по отношению к существительному будет проверена предикатом 'принадлежит', который просмотрит список классов применимости. Второе определение указывает на то, что глагол 'читает' может использоваться только с уникальным классом 'люди', подстановка названия которого сразу сделана на месте переменной C.

Подобно классу применимости C, переменные F и S необходимы для смысловой проверки анализируемых лексем. Эти переменные распространяют свои значения по цепочке определитель→субъект→глагол→объект в ходе грамматического разбора предложения. Такая информация передается друг другу смысловыми агентами-процедурами (далее, просто *смысловые агенты*), которые используют переменные F и S в качестве параметров по следующей схеме:

```
'агент_о'(Ld,S)→'агент_с'(n,S,F)→'агент_ф'(Lv,F,Fo)→'агент_фо'(Lo,Fo).
```

Предикаты с соответствующими функторами приведены далее в расширенном примере, содержащем полную грамматическую сигнатуру для разбора предложений, которые имеют предусмотренный в ГНС стиль написания.

В ходе грамматического разбора смысловые агенты обеспечивают связь с моделью ПО для выяснения правильного смыслового использования каждой лексемы (Ld, Ls, Lv, Lo) во взаимном окружении, путем проверки допустимых смысловых отношений составленных из их базовых

форм. Смысловой агент, отвечающий за работу с текущей анализируемой лексемой, отыскивает в словаре ее базовую форму, и обращается с запросом к той части модели ПО, которая охватывает множество сопутствующих грамматическому разбору структур, попадающих под признаки связи с анализируемым контекстом. Эти структуры составляются на шаге проектирования модели ПО и образуют допустимые смысловые отношения на множестве включенных в них атомов. Значения атомов соответствуют базовым формам лексем, охваченных словарем. Тогда, ответ на запрос смыслового агента подтвержден успехом в том случае, когда найден атом, значение которого в допустимом смысловом отношении эквивалентно лексеме, приведенной к базовой форме. Т.е. успешная работа смысловых агентов обеспечивает однозначную привязку лексем в грамматической структуре предложения к атомам сопутствующих структур модели ПО. Если после завершения грамматического разбора все лексемы предложения охвачены допустимыми связями с атомами сопутствующих структур, то разобранное предложение принимается, как семантически правильное, так как лексемы во взаимном окружении образуют допустимые смысловые отношения в контексте заданных правил ГНС.

Более детально алгоритм работы смысловых агентов выглядит следующим образом.

Перед вызовом процедуры 'агент_о'(Ld,S) формируется входной список лексем-определителей субъекта действия, который извлекается из анализируемого предложения путем отсечения предусмотренного в ГНС количества элементов от головы списка лексем (в нашем случае – два элемента). Полученные элементы приводятся к своей базовой форме. Далее, процедура 'агент_о'(Ld,S) через параметр Ld принимает входной список лексем-определителей и при помощи встроенного предиката bagof в базе знаний собирается информация о всех субъектах, которые попадают под заданные в определителях признаки. В результате формируется список структур, содержащих информацию о субъектах действия, для унификации с переменной S. Если этот список не пустой, то дальше его обрабатывает 'агент_с'(Ls,S,F), отвечающий за работу с субъектами.

Текущая лексема предложения Ls, в соответствии с грамматическими правилами, представляет собой субъект действия и, 'агент_с'(Ls,S,F) попытается отыскать в списке S структуру с названием объекта, которое эквивалентно базовой форме Ls. Если субъект действия найден, то 'агент_с'(Ls,S,F) унифицирует переменную F со списком функций, которые может выполнять найденный субъект. Затем, 'агент_ф'(Lv,F,Fo) обрабатывает полученный от 'агент_с'(Ls,S,F) список F.

Следующая лексема Lv представляет собой глагол, обозначающий действие, или, в нашей интерпретации, функцию применимости субъекта. Тогда 'агент_ф'(Lv,F,Fo) попытается отыскать в списке F такую функцию, функтор которой эквивалентен глаголу Lv в базовой форме. Если функция найдена, переменная Fo унифицируется со структурой, выражающей эту функцию, и завершение обработки выполняет 'агент_фо'(Lo,Fo).

Процедура 'агент_фо'(Lo,Fo) попытается отыскать среди аргументов функции применимости такой, который эквивалентен базовой форме последней лексемы Lo, представляющей собой объект действия. Если эквивалентность установлена, то смысловой анализ предложения принимается как успех.

В расширенном примере проиллюстрирована работа всей цепочки смысловых агентов. Например, благодаря их внедрению в систему грамматического разбора, для глагола 'читать' в переходном значении объектом может быть только предмет, подлежащий чтению ('книга' или 'журнал', а не 'обезьяна').

Подводя некоторые итоги, отметим, что помимо формально определенного набора грамматических правил и словаря, еще одним важнейшим компонентом для анализа предложений естественного языка является используемая модель ПО, которая служит опорой для проверки семантической правильности текста. Обычно такую проверку осуществляют, непосредственно добавляя логические отношения между формами лексем внутри фигурных скобок [2]. Но, в этом случае требуемые для проверки лексемы попадают внутрь грамматической сигнатуры, что нарушает принцип обобщения. Поэтому, проверку допустимых по смыслу лексем следует осуществлять с помощью смысловых агентов, связывающих анализируемые предложения с отдельно заданной моделью ПО, которая в своих структурах содержит атомы, соответствующие базовым формам тех или иных лексем, а грамматическое согласование осуществляется путем унификации атрибутов лексем, хранимых в словаре. Базовые формы имеют именительный падеж единственного числа для существительного и прилагательного, а для глагола – неопределенную форму. Тогда, для проверки семантической правильности предложения, текущую анализируемую лексему следует привести к базовой форме, предусмотренной в модели ПО, так как внутри предложения, в общем случае, все лексемы представлены в согласованной форме. Попутно заметим, что для преобразования существительных используется дополнительный атрибут GB, который сохраняет признак рода в базовой форме. Это необходимо для приведения к базовой форме существительных, когда отличается значение их рода в согласованной форме от рода в базовой форме. Например, слово 'обезьяны' относится к среднему роду, а в базовой форме должно иметь женский род, единственное число, именительный падеж, которыми представлено соответствующее значение атома в структуре модели ПО. В то же время, для определителей (прилагательных) дополнительного атрибута, хранящего значение рода в базовой форме, не требуется, так как они имеют все родовые текстовые формы, а в базовой форме принят мужской род в соответствии с родом понятия 'объект'.

Обычно в Прологе модель ПО описывается декларативно-процедурным методом. Например, факты для предложений о гангстере и обезьяне выглядят следующим образом:

```
'объект'(['гангстер','определитель'(['вооруженный','опасный']),
```

```
   'функция'(['грабить'(['банк','инкассатор','казино'])])]).
'объект'('обезьяна','определитель'(['смешной','рыжий']),
          'функция'(['держать'(['банан','палка']))]).
```

Тогда для успешной работы процедур `'агент_ф'(Lv,F,Fo)` и `'агент_с'(Ls,S,F)` переходные глаголы `'грабит'` и `'держит'` предварительно приводятся к неопределенной форме, а существительное `'банки'` к единственному числу именительного падежа. Существительное `'обезьяна'` также обрабатывается процедурой приведения к базовой форме, вызываемой в соответствии с работой обобщенного алгоритма, хотя в данном случае такое преобразование приводит к тавтологии, так как исходная текстовая форма этого слова соответствует базовой.

Исходя из описанного подхода к построению грамматической сигнатуры, словаря и модели ПО, обратимся к разработке ГНС, где все перечисленные компоненты представлены раздельно. Это позволяет осуществлять смысловой анализ таких фраз, которые вписываются в приведенную ранее грамматическую схему. Ограниченный объем статьи, разумеется, не дает возможности продемонстрировать детальную семантическую проверку большого количества предложений, покрываемых выбранной грамматической сигнатурой. Но, в проиллюстрированном далее образце программы очевиден резерв для расширения пределов как модели ПО, так и словаря, что ведет к более развитому диалогу пользователя с машиной. В приведенной программе можно, допустим, предусмотреть углубленный анализ связи слова `'банки'` с моделью ПО. Тогда система будет точнее реагировать, например, на некоторое предложение о финансисте, открывающем банки, и не перепутает его работу с употреблением консервов, если по каким-то причинам понадобится объединить в одну две разные модели с настолько противоположным смыслом слов при их одинаковой текстовой форме. В этом случае смысловой агент, способный к углубленному анализу связи предложения с моделью ПО, потребуется, скажем, для фразы:

```
Успешный состоятельный финансист открывает банки.
```

Приведенный ниже пример вполне допускает необходимое дополнение.

```
% интерпретатор Amzi!Prolog 7.2.1
% выполните consult данного файла и проверьте запросы для разбора предложений
% ?-'предлож'(['смешная','рыжая','обезьяна','держит','банан'],[]).
% ?-'предлож'(['знаменитый','талантливый','пародист','рассмешит','публику'],[]).
% ?-'предлож'(['легкомысленный','халатный','охранник','прозевал','грабителя'],[]).
% ?-'предлож'(['опасный','вооруженный','гангстер','грабит','банки'],[]).
:- op(500,xfy,:).
'предлож'-->'фраза_существит'(G,Q,'именит',C,F),'фраза_глаг'(G,Q,_,_,_,_,C,F).
'фраза_существит'(G,Q,A,C,F)-->[W1],[W2],[W3],
   {'прилагат_индекс'(W1,G:Q:A,_,I1),
    'прилагат_индекс'(W2,G:Q:A,_,I2),
    'существит_индекс'(W3,G:Q:A:C:GB,_,I3),
```

```
    'прилагат_индекс'(_,'мужск':'единств':'именит',K1,I1),
    'прилагат_индекс'(_,'мужск':'единств':'именит',K2,I2),
    'существит_индекс'(_,GB:'единств':'именит':C:_,K3,I3),
    'агент_о'([K1,K2],S),'агент_с'(K3,S,F)}.
'фраза_существит'(G,Q,A,C,F)-->[W],
   {'существит_индекс'(W,G:Q:A:C:GB,_,I),
   'существит_индекс'(_,GB:'единств':'именит':C:_,K,I),
    ((is_list(F)->'агент_фо'(K,F);'агент_с'(K,S,F)))}.
'фраза_глаг'(G,Q,_,_,_,_,C,F)-->[W],
   {'перех_глаг_индекс'(W,G:Q:'лицо'('третье'):_:V:'действит':C,_,I,Lc),
    'перех_глаг_индекс'(_,'неопфм':'неопфм':'неопфм':'неопфм':V:'неопфм':C,K,I,Lc),
    'агент_ф'(K,F,Fo)},'фраза_существит'(_,_,'винит',_,Fo).
'существит_индекс'(W,G:Q:A:C:GB,K,I):-atomic(W),atomic(I),!,fail.
'существит_индекс'(W,G:Q:A:C:GB,K,I):-(atomic(W)->
('словарь_существит'(G:Q:A:C:GB,L),'элемент'('принадлежать'(W,L)),'индекс'(W,L,I));
   (atomic(I),atomic(G),atomic(Q),atomic(A),'словарь_существит'(G:Q:A:C:GB,L),
    'искать_слово'(K,L,I))).
'прилагат_индекс'(W,G:Q:A,K,I):-atomic(W),atomic(I),!,fail.
'прилагат_индекс'(W,G:Q:A,K,I):-
   (atomic(W)->('словарь_прилагат'(G:Q:A,L),
                'элемент'('принадлежать'(W,L)),
                'индекс'(W,L,I));
   (atomic(I),atomic(G),atomic(Q),atomic(A),
                'словарь_прилагат'(G:Q:A,L),
                'искать_слово'(K,L,I))).
'перех_глаг_индекс'(W,G:Q:R:T:V:D:C,K,I):-atomic(W),
   atomic(I),!,fail.
'перех_глаг_индекс'(W,G:Q:R:T:V:D:C,K,I,Lc):-
   (atomic(W)->('перех_глаг'(G:Q:R:T:V:D:C,L,Lc),
                'элемент'('принадлежать'(W,L)),
                'индекс'(W,L,I));
   (atomic(I),atomic(G),atomic(Q),atomic(R),
    atomic(T),atomic(V),atomic(D),atomic(C),
                'перех_глаг'(G:Q:R:T:V:D:C,L,Lc),
                'искать_слово'(K,L,I))).
'элемент'( P ) :- P, !.
'подмножество'(L,[]).
'подмножество'(L,[X|T]):-'элемент'('принадлежать'(X,L)),'подмножество'(L,T).
'принадлежать'( X, [ X|_ ] ).
'принадлежать'( X, [ _|T ] ) :- 'принадлежать'( X, T ).
'индекс'( X, [ ], I ) :- I is 0.
'индекс'( X, [ X|T ], I ) :- !, I is 1.
'индекс'( X, [ _|T ], I ) :- 'индекс'( X, T, Ic ),(Ic =:= 0->I is Ic;I is Ic + 1).
'искать_слово'( X, [ _|_ ], 0 ):- !, fail.
'искать_слово'( X, [ X|_ ], 1 ).
'искать_слово'( X, [ _|T ], I ) :- Ic is I - 1,'искать_слово'( X, T, Ic ).
'агент_о'([K1,K2],S):- Obj='объект'(N,'определитель'(L),F),
   bagof(Obj,(Obj,'подмножество'(L,[K1,K2])),S).
'агент_с'(Ls,[],_):-!,fail.
'агент_с'(Ls,S,F):-'функция_субъект'(Ls,S,[],F).
'функция_субъект'(Ls,[],F,F).
'функция_субъект'(Ls,['объект'(Ls,_,'функция'([Fs|_]))|St],Ft,F):-
      'функция_субъект'(Ls,St,[Fs|Ft],F).
```

```
'агент_ф'(Lv,F,Fo):-'аргум_функция_субъект'(Lv,F,Fo).
'аргум_функция_субъект'(Lv,[],_):-!,fail.
'аргум_функция_субъект'(Lv,Lf,Fo):- Lf=[Lfh|Lft],
   Lfh=..[Fn,Fo],(Lv=Fn->!;'аргум_функция_субъект'(Lv,Lft,Fo)).
'агент_фо'(Lo,Fo):-'элемент'('принадлежать'(Lo,Fo)).
'объект'('гангстер','определитель'(['вооруженный','опасный']),
         'функция'(['грабить'(['банк','инкассатор','казино'])])).
'объект'('обезьяна','определитель'(['смешной','рыжий']),
         'функция'(['держать'(['банан','палка'])])).
'объект'('пародист','определитель'(['знаменитый','талантливый']),
         'функция'(['рассмешить'(['публика'])])).
'объект'('охранник','определитель'(['легкомысленный',
         'халатный']),'функция'(['прозевать'(['грабитель'])])).
'словарь_существит'('мужск':'единств':'именит':'люди':'мужск',
   ['пародист','охранник','гангстер','грабитель']).
'словарь_существит'('мужск':'единств':'родит':'люди':'мужск',
   ['пародиста','охранника','гангстера','грабителя']).
'словарь_существит'('мужск':'единств':'дател':'люди':'мужск',
   ['пародисту','охраннику','гангстеру','грабителю']).
'словарь_существит'('мужск':'единств':'винит':'люди':'мужск',
   ['пародиста','охранника','гангстера','грабителя']).
'словарь_существит'('мужск':'единств':'творит':'люди':'мужск',
   ['пародистом','охранником','гангстером','грабителем']).
'словарь_существит'('мужск':'единств':'предлож':'люди':'мужск',
   ['пародисте','охраннике','гангстере','грабителе']).
'словарь_существит'('женск':'единств':'именит':'люди':'женск',['публика']).
'словарь_существит'('женск':'единств':'родит':'люди':'женск',['публики']).
'словарь_существит'('женск':'единств':'дател':'люди':'женск',['публике']).
'словарь_существит'('женск':'единств':'винит':'люди':'женск',['публику']).
'словарь_существит'('женск':'единств':'творит':'люди':'женск',['публикой']).
'словарь_существит'('женск':'единств':'предлож':'люди':'женск',['публике']).
'словарь_существит'('женск':'единств':'именит':'животные':'женск',['обезьяна']).
'словарь_существит'('женск':'единств':'родит':'животные':'женск',['обезьяны']).
'словарь_существит'('женск':'единств':'дател':'животные':'женск',['обезьяне']).
'словарь_существит'('женск':'единств':'винит':'животные':'женск',['обезьяну']).
'словарь_существит'('женск':'единств':'творит':'животные':'женск',['обезьяной']).
'словарь_существит'('женск':'единств':'предлож':'животные':'женск',['обезьяне']).
'словарь_существит'('мужск':'единств':'именит':'растения':'мужск',['банан']).
'словарь_существит'('мужск':'единств':'родит':'растения':'мужск',['банана']).
'словарь_существит'('мужск':'единств':'дател':'растения':'мужск',['банану']).
'словарь_существит'('мужск':'единств':'винит':'растения':'мужск',['банан']).
'словарь_существит'('мужск':'единств':'творит':'растения':'мужск',['бананом'] ).
'словарь_существит'('мужск':'единств':'предлож':'растения':'мужск',['банане']).
'словарь_существит'('мужск':'единств':'именит':'учереждения':'мужск',['банк']).
'словарь_существит'('мужск':'единств':'родит':'учереждения':'мужск',['банка']).
'словарь_существит'('мужск':'единств':'дател':'учереждения':'мужск',['банку']).
'словарь_существит'('мужск':'единств':'винит':'учереждения':'мужск',['банк']).
'словарь_существит'('мужск':'единств':'творит':'учереждения':'мужск',['банком'] ).
'словарь_существит'('мужск':'единств':'предлож':'учереждения':'мужск',['банке']).
'словарь_существит'('средн':'множеств':'именит':'учереждения':'мужск',['банки']).
'словарь_существит'('средн':'множеств':'родит':'учереждения':'мужск',['банков']).
'словарь_существит'('средн':'множеств':'дател':'учереждения':'мужск',['банкам']).
'словарь_существит'('средн':'множеств':'винит':'учереждения':'мужск',['банки']).
```

```
'словарь_существит'('средн':'множеств':'творит':'учреждения':'мужск',['банками'] ).
'словарь_существит'('средн':'множеств':'предлож':'учреждения':'мужск',['банках']).
'словарь_прилагат'('мужск':'единств':'именит',['вооруженный','знаменитый',
   'легкомысленный','опасный','рыжий','смешной','талантливый','халатный']).
'словарь_прилагат'('мужск':'единств':'родит',['вооруженного','знаменитого',
   'легкомысленного','опасного','рыжего','смешного','талантливого','халатного']).
'словарь_прилагат'('мужск':'единств':'дател',['вооруженному','знаменитому',
   'легкомысленному','опасному','рыжему','смешному','талантливому','халатному']).
'словарь_прилагат'('мужск':'единств':'винит',['вооруженного','знаменитого',
   'легкомысленного','опасного','рыжего','смешного','талантливого','халатного']).
'словарь_прилагат'('мужск':'единств':'творит',['вооруженным','знаменитым',
   'легкомысленным','опасным','рыжим','смешным','талантливым','халатным']).
'словарь_прилагат'('мужск':'единств':'предлож',['вооруженном','знаменитом',
   'легкомысленном','опасном','рыжем','смешном','талантливом','халатном']).
'словарь_прилагат'('женск':'единств':'именит',['вооруженная','знаменитая',
   'легкомысленная','опасная','рыжая','смешная','талантливая','халатная']).
'словарь_прилагат'('женск':'единств':'родит',['вооруженную','знаменитую',
   'легкомысленную','опасную','рыжую','смешную','талантливую','халатную']).
'словарь_прилагат'('женск':'единств':'дател',['вооруженной','знаменитой',
   'легкомысленной','опасной','рыжей','смешной','талантливой','халатной']).
'словарь_прилагат'('женск':'единств':'винит',['вооруженную','знаменитую',
   'легкомысленную','опасную','рыжую','смешную','талантливую','халатную']).
'словарь_прилагат'('женск':'единств':'творит',['вооруженной','знаменитой',
   'легкомысленной','опасной','рыжей','смешной','талантливой','халатной']).
'словарь_прилагат'('женск':'единств':'предлож',['вооруженной','знаменитой',
   'легкомысленной','опасной','рыжей','смешной','талантливой','халатной']).
'перех_глаг'('неопфм':'неопфм':'неопфм':'неопфм':
   'несоверш':'неопфм':'люди',['грабить'],['люди']).
'перех_глаг'('неопфм':'неопфм':'неопфм':'неопфм':
   'несоверш':'неопфм':C,['держать','зевать','смешить'],Lc):-
   ['животные','люди'] = Lc,'элемент'('принадлежать'(C,Lc)).
'перех_глаг'('неопфм':'неопфм':'неопфм':'неопфм':
   'соверш':'неопфм':C,['прозевать','продержать','рассмешить'],Lc):-
   ['животные','люди'] = Lc,'элемент'('принадлежать'(C,Lc)).
'перех_глаг'('мужск':'единств':'лицо'('третье'):'наст':
   'несоверш':'действит':'люди',['грабит'],['люди']).
'перех_глаг'('мужск':'единств':'лицо'('третье'):'наст':
   'несоверш':'действит':C,['держит','зевает','смешит'],Lc):-
   ['животные','люди']=Lc,'элемент'('принадлежать'(C,Lc)).
'перех_глаг'('женск':'единств':'лицо'('третье'):'наст':
   'несоверш':'действит':C,['держит','зевает','смешит'],Lc):-
   ['животные','люди']=Lc,'элемент'('принадлежать'(C,Lc)).
'перех_глаг'('мужск':'единств':'лицо'('третье'):'прош':
   'соверш':'действит':C,['прозевал','продержал','рассмешил'],Lc ):-
   ['животные','люди']= Lc,'элемент'('принадлежать'(C,Lc)).
'перех_глаг'('женск':'единств':'лицо'('третье'):'прош':
   'соверш':'действит':C,
   ['прозевала','продержала','рассмешила'],Lc ) :-
   ['животные','люди']= Lc,'элемент'('принадлежать'(C,Lc)).
```

Выводы

Раздельное представление грамматической сигнатуры и словаря в ГНС обеспечивает обращение к различным моделям проблемных областей знаний в рамках ограниченного естественного языка с формализованным стилем записи предложений. Это дает возможность пользователю в своих запросах придавать предложениям разный смысл путем дополнения или замены словаря и модели ПО без изменения грамматических правил, что упрощает разработку гибких систем анализа естественного языка.

Грамматика непосредственных составляющих допускает расширение грамматических признаков частей речи смысловыми признаками, которые посредством связанных переменных Пролога передаются для анализа смысловым агентам.

Внедрение в систему грамматического разбора смысловых агентов позволяет осуществлять проверку не только грамматической, но и смысловой правильности лексем в их контекстной взаимосвязи внутри предложений естественного языка. В этом случае смысловые агенты служат каналом обмена семантической информацией между анализируемыми предложениями и моделью ПО вдоль всей цепочки грамматического разбора. Это позволяет удерживать смысл анализируемых предложений, в пределах ограничений, предусмотренных на сопутствующих грамматическому разбору структурах модели ПО, которая в целом образует семантическую опору системы обработки естественного языка.

Рассмотренная модель ПО охватывает базовые формы всех помещенных в словарь лексем, объединяя их в допустимые смысловые отношения, и в типовом случае описывается декларативно-процедурным методом, что позволяет адаптировать ГНС к формальному семантическому анализу предложений естественного языка в пределах заданных ограничений.

Список использованной литературы

1. Дж. Малпас. Реляционный язык Пролог и его применение: Пер. с англ. – М.: Наука, 1990. – 464с.
2. Ulf Nilsson and Jan Maluszynski. Logic, Programming and Prolog.– Hoboken.: John Wiley & Sons Ltd., 1995.–294c.

УДК 681.327

БЛІНОВА Т. О.

ВИКОРИСТАННЯ КОДІВ ФІБОНАЧЧІ ТА ЗОЛОТОЇ ПРОПОРЦІЇ В ЦИФРОВИХ ОПТИЧНИХ КАНАЛАХ ПЕРЕДАЧІ ІНФОРМАЦІЇ

Аналізуються класичний код Фібоначчі та код золотої 1-пропорції з метою визначення найприйнятніших параметрів коду, орієнтованого на використання в цифрових оптичних каналах передачі інформації. На прикладі класичного коду Фібоначчі здійснена оцінка ефективності за чотирма критеріями.

The classic Fibonacci code and gold 1-proportion code are analyzed with the purpose of determination of the most acceptable parameters of code, oriented to the use in the digital optical interconnects. On the classic Fibonacci code example the estimation of efficiency is carried out after four criteria.

Розвиток технологій виготовлення інтегральних оптичних мікросхем робить цифрові оптичні обчислення все більш перспективними. Роботи по розробці цифрових оптичних каналів передачі інформації й оптичних комп'ютерів сьогодні широко ведуться в США, Європі, Японії, Росії [1].

Інтерес до використання цифрових оптичних каналів передачі інформації при створенні не тільки обчислювальних мереж, але й окремих комп’ютерів і навіть СБІС зумовлений, зокрема, наступним.

Навіть у персональних комп'ютерах довжина з'єднань між контактами мікросхем, розташованих на різних платах, вимірюється дециметрами, а інколи й метрами. Використовування оптичного каналу для організації "міжчіпового" обміну дозволяє на порядок скоротити затримку розповсюдження сигналу. З іншого боку, одним зі шляхів вирішення проблеми з'єднань у СБІС є використання оптоволоконних мікроліній із метою зниження енергоспоживання, підвищення швидкодії й надійності СБІС, у тому числі в екстремальних умовах експлуатації. Оптичні системи нечутливі до електромагнітних перешкод і нічого не випромінюють у зовнішнє середовище, забезпечуючи захист комп'ютера від перехоплення інформації.

При розробці цифрових оптичних каналів зв’язку, незалежно від місця використання, чи то, наприклад, у локальній мережі, чи то при організації “міжчіпового” обміну, виникає задача вибору відповідного методу кодування інформації. Автору невідомий код, що був би найкращим у будь-якому випадку.

Ще в кінці минулого століття вінницьким СКТБ "Модуль" і київським НВО "Маяк" була розроблена волоконно-оптична лінія зв'язку підвищеної пропускної спроможності, в якій використовувалися два способи кодування інформації: біфазний код типу "Манчестер" й один із різновидів

кодів Фібоначчі. При роботі цієї системи в коді Фібоначчі збільшувалася пропускна здатність, а ймовірність помилки в каналі становила 10^{-11} (при роботі в коді "Манчестер"– 10^{-9}). Досить широко відомі інженерні розробки СКТБ "Модуль" по проектуванню “фібоначчієвих” аналого-цифрових та цифро-аналогових перетворювачів, що самокоректуються. Вони показали, що застосування кодів Фібоначчі та золотої пропорції дозволяє одночасно поліпшити всі технічні параметри АЦП та ЦАП, зокрема, точність, швидкодію, а насамперед, їхню температурну й временну метрологічну стабільність [4].

Коди Фібоначчі та золотої пропорції, маючи всі основні переваги класичних двійкових кодів (простоту виконання арифметичних і логічних операцій тощо), володіють надмірністю. Це створює можливості для вирішення ряду задач, зокрема, задач точності, синхронізації, контролю й діагностики. Слід підкреслити, що технології волоконно-оптичних комунікацій добре відповідають послідовній архітектурі. У [2,3] показано, що використання 1-кодів золотої пропорції дозволяє виконувати послідовні обчислення, починаючи зі старших розрядів, причому апаратурні затрати на такі обчислення, порівняно з іншими методами кодування, невеликі.

Існує багато кодів золотої пропорції та кодів Фібоначчі.

Метою даної роботи є оцінка ефективності використання класичного коду Фібоначчі та коду золотої 1-пропорції в цифрових оптичних каналах передачі інформації.

При оцінці ефективності бралися до уваги наступні основні показники: кодова надмірність; кодова самосинхронізація; здатність кодів до виявлення й корекції помилок; прийнятна складність кодування-декодування.

У надмірній двійковій позиційній системі числення з ірраціональною основою типа класичної золотої пропорції (золотої 1-пропорції) задане число D може бути представлене як

$$D = \sum_{i=-\infty}^{+\infty} d_i \alpha_1^i,$$

де d_i={0,1} – i-й розряд кодового слова;

$\alpha_1 = (1+\sqrt{5})/2$ – основа системи числення;

при цьому ваги сусідніх розрядів у кодовому слові зв’язані співвідношенням:

$$\alpha_1^i = \alpha_1^{i-1} + \alpha_1^{i-2}. \qquad (1)$$

У надмірній системі числення на основі класичного коду Фібоначчі натуральне число B може бути представлене як

$$B = \sum_{i=0}^{n-1} b_i \varphi_1(i),$$

де b_i={0,1} – i-й розряд кодового слова;

φ_1 (i) – i-е 1-число Фібоначчі, що обчислюється за наступною рекурентною формулою:

$$\varphi_1(i) = \begin{cases} 0 \text{ при } i < 0; \\ 1 \text{ при } i = 0; \\ \varphi_1(i-1) + \varphi_1(i-2) \text{ при } i > 0. \end{cases}$$

Для кодування m двійкових розрядів потрібно n розрядів коду класичного коду Фібоначчі. У [5] доведено, що відносна надмірність R_ϕ цього коду, яка визначається як $R_\phi = (n\text{-}m)/m$, при $n \to \infty$ дорівнює відносній надмірності коду золотої 1-пропорції, і приблизно становить 0,44.

Особливістю цих кодів є багатозначність кодового представлення одного й того ж числа. Перехід від однієї форми представлення до іншої здійснюється за допомогою операцій згортки та розгортки.

Для 1-коду Фібоначчі та коду золотої 1-пропорції операція згортки означає заміну (відповідно до (1) та (2)) одиниць двох молодших (i-1)-го та (i-2)-го розрядів одиницею старшого i-го розряду, а операція розгортки – виконання зворотної заміни.

Велика кількість форм представлення одного й того ж числа дозволяє вибрати такі форми, які якнайповніше задовольняють вимогам конкретного застосування.

Серед безлічі форм представлення слід виокремити ряд основних, що мають певні властивості.

У першу чергу, це мінімальна форма (М-форма), що має мінімальну кількість одиниць.

З М-форми можна отримати (за допомогою операції розгортки) інші форми представлення (табл. 1): частковорозгорнену (ЧР-форму), повністю розгорнену (максимальну) і рівноважну (РВ-форму).

У коді М-форми приблизно 27% одиниць і 73% нулів [3].

Основна властивість М-форми – наявність у кодовому зображенні між двома одиницями не менше одного нуля.

У коді максимальної форми міститься максимально можлива кількість одиниць, вона може бути отримана при виконанні над кодом М-форми операції розгортки.

РВ-форма – це форма з рівною кількістю одиниць і нулів. Якщо код М-форми не є рівноважним, необхідно доповнити недостатню кількість одиниць виконанням операції розгортки.

При неможливості отримання за допомогою розгортки М-форми потрібної кількості одиниць вводиться одиниця в старший розряд перед розрядною сіткою (одиниця зсуву), яка також піддається розгортці. Відновлення інформації здійснюється шляхом виконання операції згортки. При цьому введена одиниця зсуву піде за межі розрядної сітки. Слід підкреслити, що РВ-форма може мати декілька представлень, які відрізняються довжиною серій нулів (одиниць) (см. табл. 1).

Табл. 1. Приклад форм представлення числа 31 у 1-коді Фібоначчі

Вага розрядів	**144**	**89**	**55**	**34**	**21**	**13**	**8**	**5**	**3**	**2**	**1**	**1**
М-форма	0	0	0	0	1	0	1	0	0	1	0	0
Максимальна форма	1	0	1	1	0	1	1	1	1	0	1	1
РВ-форма	1	1	0	0	1	0	0	1	1	1	0	0
	1	1	0	0	1	0	1	0	0	0	1	1
	1	0	1	1	1	0	1	0	0	1	0	0

Окрім цих форм представлення існує безліч проміжних, так званих ЧР-форм.

При виборі форми представлення кодів Фібоначчі та золотої пропорції, що використовуватиметься в каналах передачі інформації, головну увагу треба приділити забезпеченню самосинхронізації.

Для кількісної оцінки самосинхронізованості кодів використовується коефіцієнт синхронізованості $S(n)$. Ідеально самосинхронізованим ($S = 1$) можна вважати регулярний одноінтервальний сигнал із тривалістю інтервалу T_0. Чим частіше в сигналі зустрічатимуться інтервали із тривалістю $T_1 > T_0$, і чим вони будуть довшими, тим менша здатність до самосинхронізації у сигналу. Таким представленням відповідає наступне визначення коефіцієнта синхронізованості:

$$S(i) = 1 - \sum_{i=0}^{L} a_i P_i,$$

де P_i – вірогідність появи в сигналі інтервалу тривалістю $T_1 > T_0$;

a_i – ваговий коефіцієнт, приписуваний тривалості T_1 [4].

Для прикладу розглянемо, як у мережах Ethernet можна було би застосувати код Фібоначчі.

У теперішній час у мережах Ethernet стандарту 802.3 на основі оптичного волокна манчестерське або двопозиційне кодування використовується тільки для інтерфейсів стандартів 10BASE-FB/FP/FL. Вище відзначалася перевага коду Фібоначчі над кодом «Манчестер».

За стандартом Fast Ethernet (стандарт 802.3u, що являє собою доповнення до існуючого стандарту 802.3 у вигляді глав із 21 по 30) один з інтерфейсів – багатомодове оптоволокно (100BaseFX). За цим стандартом пристрій фізичного рівня PHY приймає дані в паралельній формі від MAC-підрівня (канального рівня), кодує їх за методом 4B/5B і передає послідовно по оптоволокну, використавши метод кодування NRZI (Non-Return to Zero Inverted), що, як і метод 4B/5B, був визначений у стандарті FDDI. Через особливості методу NRZI для забезпечення частих змін сигналу потрібно виключити з кодів занадто довгі послідовності нулів. Отже, у методі 4B/5B кожні 4 біти даних MAC-підрівня представляються 5 біта-

ми так, щоб гарантувати наявність не більше трьох нулів підряд при будь-якому сполученні бітів у вхідних даних.

Однак, при застосуванні коду 4B/5B не забезпечується баланс по постійному струму, тому що його таблиця кодування містить більше 1, чим 0. Потенційно це може привести до залежного від переданих даних нагрівання лазерних діодів, оскільки передавач може передавати більше бітів "1" (випромінювання є), чим "0" (випромінювання немає), що, у свою чергу, може спричини появу помилок при високих швидкостях передачі [6].

Отже, для кодування сигналів даних в інтерфейсах стандарту 100BaseFX слід вибрати таку форму представлення коду Фібоначчі, яка містила би багато 0, але не мала би послідовності із чотирьох і більше нулів підряд, у тому числі й на стику кодових комбінацій.

Здійснимо вибір форми представлення класичного коду Фібоначчі, яка відповідає цьому критерію, на прикладі кодування десяткових чисел від 0 до 255.

Очевидно, що М-форма та максимальна форма не є оптимальними, оскільки перша містить забагато нулів, а друга – забагато одиниць. Потрібно шукати таку ЧР-форму представлення, котра не містила би послідовності із трьох і більш нулів підряд, і мала послідовності одиниць обмеженої довжини (для підвищення коефіцієнта самосинхронізації). Назвемо для визначеності таку ЧР-форму представлення мінімізованою (МН-формою).

У зв'язку з тим що код МН-форми представлення містить послідовності із двох одиниць, діапазон представлення чисел у МН-формі більше ніж у М-формі. Для того щоб представити число С = 255 за допомогою М-форми, потрібно 13 розрядів (1 0000 1000 0010), а за допомогою МН-форми – 12 розрядів (1100 0110 0010).

Враховуючи, що два молодші розряди МН-форми мають рівну вагу, і наймолодший із них завжди – 0, цей розряд є сенс не передавати по каналу. Отже, коефіцієнт кодової надмірності для коду Фібоначчі (у даному випадку) дорівнює $R_ф$ = (11-8)/8 = 0,375, а для коду 4B/5B – 0,25.

Для збільшення коефіцієнта синхронізованості МН-форми представлення всі кодові комбінації, що починаються послідовністю із трьох і більше нулів, доповнюються розгорненою одиницею зсуву (тобто такі кодові комбінації починаються з наборів 11 або 1010...101011).

Результати аналізу розробленої автором 11-розрядної таблиці кодування десяткових чисел від 0 до 256 (восьмирозрядних двійкових) за допомогою МН-форми 1-коду Фібоначчі, та 10-розрядної таблиці кодування цих же чисел за допомогою методу 4B/5B наступні. Відсутні послідовності із трьох і більше нулів, у тому числі на стику. Кількість одиниць при кодуванні за методом 4B/5B значно більша (табл. 2), як і максимальна довжина серій одиниць на стику – 8 проти 4 у коді Фібоначчі.

Табл.2. Розподіл комбінацій (за кількістю одиниць у комбінації)

у кодових таблицях

Кількість 1 у комбінації	*8*	*7*	*6*	*5*	*4*
Кількість комбінацій (1-код Фібоначчі),%	***0***	***1***	***30***	***60***	***9***
Кількість комбінацій (код 4В/5В),%	***8***	***28***	***33***	***24***	***7***

Таким чином, порівняно з 4В/5В, кодування за допомогою МН-форми 1-коду Фібоначчі забезпечує кращий баланс по постійному току та кращу синхронізованість, а також більші можливості для контролю та керування (за рахунок більшої кількості незадіяних кодових комбінацій).

Для інтерфейсів 1000Base-FX/LX/CX/ZX замість методу кодування 4В/5В було запропоновано використовувати метод кодування 8В/10В, в якому 8 біт інформації, що поступають із МАС-підрівня, кодуються 10-бітами таким чином, щоб забезпечити збалансовану кількість 0 та 1 у потоці сигналів.

Отже, обрана для використання в інтерфейсі 1000Base-FX/LX/CX/ZX форма представлення коду Фібоначчі повинна бути рівноважною. Аналіз таблиці кодування 12-розрядних чисел Фібоначчі показав, що з можливих РВ-форм певного числа завжди можна обрати таку РВ-форму, яка не містить послідовності із чотирьох та більше нулів підряд, а також не має набору бітів 00 у двох наймолодших розрядах та набору бітів 000 у трьох найстарших розрядах, що забезпечує відсутність набору 0000 й на стику кодових комбінацій.

Аналіз властивостей класичного коду Фібоначчі та коду 1-пропорції показує, що їхнє застосування в цифрових оптичних каналах передачі інформації дозволяє: забезпечити кодову самосинхронізацію потоків даних; знаходити помилки, які порушують порядок проходження одиниць та нулів у коді; забезпечити передачу керуючої інформації; порівняно легко виконувати операції кодування за допомогою розгортки.

Найбільш ефективним буде використання системи АЦП(ЦАП)-оптичний канал, яка працює цілком у класичному коді Фібоначчі та золотої 1-пропорції, адже тоді можна буде уникнути зайвих операцій по перекодуванню сигналів у(із) двійковий код. Така система, яка містить АЦП(ЦАП), що самокоректуються, та цифровий оптичний канал, який нечутливий до електромагнітних перешкод, може знайти широке застосування в середовищах із великими індустріальними перешкодами тощо.

Список посилань

1. Никоноров Н.В., Козлов С.А., Никифоров В.О., Белашенков Н.Р. Разработка модели и создание интегрированного Учебно-научно-производственного центра СПбГУ ИТМО - ОАО «ЛОМО» «оптические технологии и системы» (УНПЦ «ОТИС») //Интеллектуальные информационные системы. №3. – 2005. www.fasi.gov.ru.
2. Блинова Т.А. О возможности применения золотого кода в рекурсивных кон-

вейерных преобразователях информации //Вестн. КПИ. Сер. автоматика и электроприборостроение. – N22. – 1985. – С.17 – 19.
3. Авт. св. СССР N 1693601 А1, кл. G06F7/49. БИ. 47, 1991. Конвейерное вычислительное устройство/Блинова Т.А.
4. Стахов А.П., Лихтциндер Б.Я., Орлович Ю.П., Сторожук Ю.А. Кодирование данных в информационно-регистрирующих системах. –К.: Издательство "Техника". –1985. – 127 с.
5. 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.
6. Высокоскоростные сети. http://www.quick-net.nm.ru/

УДК 004.94

КІР‘ЯНОВ О.В.

ПОКРАЩЕННЯ РЕЗУЛЬТАТІВ ОТРИМАНИХ ПОЛІНОМІАЛЬНИМ БАГАТОРЯДНИМ МГУА ДЛЯ ЗАДАЧ З ВЕЛИКОЮ КІЛЬКІСТЮ ЗМІННИХ ЗА ДОПОМОГОЮ ДОДАВАННЯ ЗВОРОТНИХ ЗМІННИХ

В даній статті запропоновано алгоритм, який за допомогою додавання зворотних змінних покращує результати отримані поліноміальним багаторядним МГУА для задач з великою кількістю змінних. Особливістю цього алгоритму є те, що для відбору найінформативніших змінних використовуються результати отримані за допомогою МГУА . Проведено порівняльний аналіз наведеного алгоритму з поширеним алгоритмом з використанням коефіцієнтів кореляції; експериментально доведено, що наведений алгоритм дає кращі результати.

This article describes algorithm that improves results found with the help of polynomial multilayered GMDH by introducing reverted variables. Peculiarity of this algorithm is that it uses results got by GMDH to select the most informative variables. Comparative analysis of the described algorithm and widespread algorithm that use correlation coefficients was done. It was proved experimentally that described algorithm gave better results.

Вступ

Метод групового урахування аргументів (МГУА) був запропонований наприкінці 60-х – початку 70-х років академіком О.Г. Івахненко. Цей метод використовує ідеї самоорганізації і механізми живої природи – схрещування (гібридизацію) і селекцію (добір). МГУА це не один алгоритм, а цілий спектр алгоритмів, кожен з яких вирішує задачі при певних умовах. Найбільш поширеними варіаціями є: комбінаторний, багаторядний, з послідовним виділенням трендів, нечіткий та інші [1].

МГУА показав високу ефективність при застосуванні в різних сферах: аналізу та прогнозуванні економічних систем; медичної діагностиці; аналізу та прогнозуванні екологічних систем; прогнозу погоди, тощо [2].

Індуктивні алгоритми МГУА передбачають перебір усіх можливих варіантів (моделей) і вибір найкращої моделі. Для отримання найкращих результатів додають якомога більше вхідних змінних. Однак, через перебір усіх варіантів час розрахунків залежить експоненціально від кількості вхідних змінних. Для задач з великою кількістю змінних (до 500) не можна просто додавати зворотні змінні на вхід МГУА, тому що це «значно» збільшить час розрахунків. Тому використовують різні методи для відсіювання найменш інформативних змінних. Рекомендований спосіб відсіювання це спосіб з використанням коефіцієнтів кореляції, коли визначаються коефіцієнти кореляції вхідних змінних з виходом моделі, розміщу-

ються змінні у порядку спадання коефіцієнтів кореляції та відбираються перші n змінних [1, 3, 4].

Існує інше перспективне рішення для комбінаторного МГУА – це комбінаторний МГУА з послідовним відбором змінних (Combinatorial GMDH algorithm with successive selection of arguments). Основна ідея полягає в тому, що відбираємо найінформативніші змінні за допомогою самого ж МГУА [3, 4]. У цій статті ця ідея була розвинена для поліноміального багаторядного алгоритму МГУА.

Постановка задачі

Припустимо, що є багатовимірна вибірка даних W, яка складається з n спостережень за багатовимірною змінною $X = \{X_1, X_2, ..., X_m\}$. Необхідно знайти залежність кожної змінної X_i від усіх інших: $X_i = f_i(X / X_i), i = 1..m$, використовуючи поліноміальний багаторядний алгоритм МГУА. Результатом роботи поліноміального багаторядного алгоритму МГУА є поліном Колмогорова-Габора [1], в нашому випадку він буде мати наступний вигляд:

$$f_i(X / X_i) = a_0 + \sum_{j=1}^{m-1} a_j x_j + \sum_{j=1}^{m-1} \sum_{k=1}^{m-1} a_{jk} x_j x_k + ... \qquad (1)$$

Якість знайденої залежності f_i повинна оцінюватись за зовнішнім критерієм регулярності $AR(f_i)$. Критерій регулярності визначається наступним чином: вибірка W_i формується з W видаленням спостережень за змінною X_i; далі W_i поділяється на дві підвибірки W_{iA}(навчальну) та W_{iB}(перевірочну). f_{iA} – це залежність f_i коефіцієнти $a_j, a_{jk}, ...$ якої знайдені на вибірці W_A. Тоді значення критерію регулярності для залежності f_i знаходиться за наступною формулою: $AR(f_i) = \| y_B - f_{iA}(W_B) \|$. При порівнянні двох залежностей кращою залежністю вважається та, значення критерію якої менше.

Необхідно покращити знайдені залежності за допомогою додавання зворотних змінних.

Розв‘язання задачі
Багаторядні алгоритми МГУА

Відповідно до поставленої задачі, для розв’язку скористуємося багаторядним алгоритмом МГУА, який широко застосовується для рішення некоректних чи недовизначених задач моделювання, тобто у випадку, коли число точок у таблиці дослідних даних менше числа аргументів, що

входять у синтезовану модель. Але він також успішно застосовується для задач, коли вихідних даних досить для застосування однорядного МГУА.

Багаторядні алгоритми працюють за наступною схемою:

Будуються часткові описи від усіх попарних комбінацій початкових даних $y_1 = f(x_1, x_2), y_2 = f_2(x_1, x_3), ..., y_k = f_k(x_{n-1}, x_n)$. Коефіцієнти знаходяться за допомогою МНК (метод найменших квадратів). З цих моделей вибирається деяке число кращих за зовнішнім критерієм селекції.

Змінні, отримані на попередньому кроці, разом з початковими даними формують вхідні данні для нових часткових описів: $z_1 = \varphi(x_1, y_1), z_2 = \varphi_2(x_1, y_2), ..., z_l = \varphi_k(y_{k-1}, y_k)$. Знов відбираються найкращі моделі за зовнішнім критерієм селекції і передаються на наступний крок.

Якщо значення зовнішнього критерію зменшується, то переходимо на крок 2, інакше алгоритм зупиняється.

Кожний частковий опис може бути лінійним:

$$f = a_0 + a_1 x_i + a_2 x_k$$

або нелінійним

$$f = a_0 + a_1 x_i + a_2 x_j + a_3 x_i x_j .$$

Критерій селекції вибирається в залежності від «якості» вхідних даних та мети побудови поліномів (знаходження зв‘язків, побудова короткострокового прогнозу, тощо). В нашому випадку, в постановці було визначено, що необхідно використовувати зовнішній критерій регулярності.

Тепер зробимо припущення, які допоможуть нам розв‘язати поставлену задачу.

Припущення №1. Припустимо, що за допомогою поліноміального багаторядного МГУА знайдено оптимальну за зовнішнім критерієм залежність (1). Усі x_l при яких стоїть ненульовий коефіцієнт a_l будуть найінформативнішими змінними для y.

Наприклад, знайдено залежність $y = a_0 + a_1 x_2 + a_2 x_4 x_5$. Тоді для y інформативними змінними будуть x_2, x_4, x_5.

Дійсно, якщо МГУА побудував оптимальну (чи одну з оптимальних) залежність, то ці змінні будуть найінформативніші. Якщо ми візьмемо інші змінні або викинемо одну з них, то отримаємо «гіршу» залежність, тому що при переборі МГУА відібрав залежність з мінімальним значенням зовнішнього критерію.

Припущення №2. Припустимо, що за допомогою поліноміального багаторядного МГУА знайдено оптимальну за зовнішнім критерієм залеж-

ність (1). Відбираємо усі одночлени в які входить змінна x_1 і при яких стоїть ненульовий коефіцієнт a_1. Усі інші змінні, що також входять в відібрані одночлени, будуть інформативними для змінної x_1, але їх спочатку потрібно обернути ($\frac{1}{x_i}$).

Наприклад, нехай знайдено залежність $y = a_0 + a_1 x_2 x_4 x_6 + a_2 x_4 x_5$. Тоді для змінної x_4 інформативними змінними будуть $\frac{1}{x_2}, \frac{1}{x_5}, \frac{1}{x_6}$, для змінної x_2 – $\frac{1}{x_4}, \frac{1}{x_5}, \frac{1}{x_6}$ тощо.

Припустимо, що у знайденому поліномі змінна x_k присутня лише у одному одночлені: $y = a_0 + x_k f_1(X/x_k) + f_2(X/x_k) + f_3(X/x_k) + \ldots$, де $f_i(X/x_k) = a_i \prod_{j=1, j\neq k}^{k} x_j$.

Перенесемо x_k в ліву частину, а все інше у праву:

$$x_k = \frac{y - a_0 - f_2(X/x_k) - f_3(X/x_k) - \ldots}{f_1(X/x_k)}.$$

Тобто x_k залежить від $\frac{1}{f_1(X/x_k)} = \frac{1}{a_1 \prod_{j=1, j\neq k}^{k} x_j}$. З цього випливає, що x_k залежить від усіх змінних з $f_1(X/x_k)$, але взятих оберненими. Зробимо евристичне припущення, що такий самий результат ми отримуємо і у випадку коли змінна x_k присутня в декількох одночленах і її не можна виразити через інші змінні.

На основі наведених вище припущень розроблено алгоритм, який складається з трьох кроків:

Алгоритм

Знаходимо p найкращих залежностей для кожної змінної X_i використовуючи поліноміальний багаторядний МГУА:

$$f_{ij}(X/X_i), i = 1..m, j = 1..p$$

Для $\forall X_i$ визначаємо множину найінформативніших змінних X_{Ei}, що складається як зі звичайних вхідних змінних X_j так і зворотних $\frac{1}{X_j}$:

Всі X_j, що входять у p найкращих поліномів для X_i, потрапляють у X_{Ei} (на основі Припущення №1)

Знаходимо всі X_j, що помножуються на X_i в будь якому поліномі. $\frac{1}{X_j}$ потрапляють у X_{Ei} (на основі Припущення №2).

Використовуючи багаторядний поліноміальний алгоритм МГУА і X_{Ei} як набір вхідних змінних знов знаходимо залежності.

Недоліком цього алгоритму є те, що кількість вхідних змінних, які використовуються на кроці 3, може бути набагато більше ніж на 1-ому кроці. А так як час роботи використаного алгоритму МГУА залежить експоненціально від кількості вхідних змінних, то це може призвести до значного збільшення часу розрахунків. Тому X_{Ei} повинна містити не більше змінних ніж початкова кількість змінних m, щоб не збільшити «значно» час розрахунків за цим алгоритмом. Якщо кількість змінних у X_{Ei} перевищує m, тоді є 2 варіанти:

Зменшити кількість поліномів, з яких ми отримуємо найінформативніші змінні (рекомендований варіант)

Відсіяти змінні з найменшими коефіцієнтами кореляції.

Таким чином, час розрахунків в середньому збільшиться приблизно у 2 рази, так як при другому розрахунку кількість змінних не перевищує початкової кількості.

Експерименти

Експерименти проводилися на даних з сайту http://www.gmdh.net/GMDH_dat.htm. Мета експериментів була виявити який з двох алгоритмів кращий: алгоритм з використанням коефіцієнтів кореляції чи наведений вище алгоритм. В таблицях попарно порівнюються результати отримані за допомогою багаторядного поліноміального МГУА: без використання зворотних змінних (далі буде називатися «МГУА без звор. змін.»), з використання зворотних змінних відібраних за допомогою алгоритму з коефіцієнтами кореляції («МГУА зі звор. змін. за коеф. корел.»), з використанням зворотних змінних відібраних за допомогою наведеного вище алгоритму («МГУА зі звор. змін. за алгоритмом»). Будемо вважати, що поліном покращився, якщо значення зовнішнього критерію селекції зменшилося більше ніж на 5%. І відповідно, будемо вважати, що поліном погіршився, якщо значення зовнішнього критерію селекції збільшилося більше ніж на 5%.

Таблиці мають наступну структуру: в першому стовпчику дається назва моделі, в наступних – кількість залежностей, які покращились або погіршились.

Назва моделі	**«МГУА зі звор. змін. за коеф. корел.»**				**«МГУА зі звор. змін. за алгоритмом»**			
	Порівняння з «МГУА без звор. змін.»		**Порівняння з «МГУА зі звор. змін. за алгоритмом»**		**Порівнянні з «МГУА без звор. змін.»**		**Порівнянні з «МГУА зі звор. змін. за коеф. корел.»**	
	Покращилось	**Погіршилось**	**Покращилось залеж-**	**Погіршилось**	**Покращилось**	**Погіршилось**	**Покращилось**	**Погіршилось**

	залежно-стей	залеж-ностей	ностей	залежно-стей	залеж-ностей	залеж-ностей	залеж-ностей	залеж-ностей
Демогра-фія	5	0	6	2	5	1	2	6
Сумарна демогра-фія	12	2	6	9	12	2	9	6
Сумарні макрое-кон. показни-ки	7	3	5	8	12	0	8	5
Показни-ки індуст-рії	2	11	1	13	10	4	13	1
Кількість зайнятих в індустрії	4	5	3	4	4	5	4	3
Інфляція	7	4	0	9	12	1	9	0
Випуск продукції	3	3	3	0	3	3	0	3

Якщо порівнювати розглянуті алгоритми з «МГУА без звор. змін.», то з **табл.1** ми бачимо, що більшість поліномів покращились після використання наведеного вище алгоритму (72% залежностей покращилось). Трохи гірший результат дає алгоритм з використанням коефіцієнтів кореляції (покращилось майже 50%залежностей). Якщо ж порівняти результаті цих двох алгоритмів, то ми бачимо, що покращилось 56% поліномів проти 30%, які погіршились. Таким чином, можна зробити висновок, що наведений вище алгоритм дає кращі результати ніж алгоритм з використанням коефіцієнтів кореляції.

Необхідно додати, що поліноми знайдені на першому кроці алгоритму не повинні бути загублені. Маючи повний набір поліномів отриманих як з використанням зворотних змінних так і без них, ми повинні відібрати найкращі залежності. Таким чином в нас не буде поліномів, які погіршились після введення зворотних змінних.

Висновки

Використання алгоритму МГУА для визначення найінформативніших змінних було започатковано в комбінаторному МГУА з послідовним відбором змінних. В цій статті запропоновано метод відбору звичайних і зворотних змінних з використанням поліноміального багаторядного МГУА. Був проведений порівняльний аналіз з алгоритмом відбору змінних з використанням коефіцієнтів кореляції і показано, що запропонований алгоритм дає кращі результати, тобто для більшості змінних знаходить залежності з меншим значенням критерію регулярності.

Даний напрямок є дуже перспективним. Подальша робота може бути спрямована на поширення використання МГУА для відбору інформативних змінних як для інших алгоритмів (як пов’язаних з МГУА так і не пов‘язаних).

Список використаної літератури

1. Зайченко Ю.П. Основи проектування інтелектуальних систем – К.: Видавничий дім «Слово», 2004. – 352 с.
2. Madala H.R ., Ivakhnenko A.G. Inductive Learning Algorithms for Complex Systems Modeling. Boca Raton: CRC Press Inc., 1994.
3. Samoilenko O.A., Stepashko V.S. Combinatorial GMDH algorithm with successive selection of arguments. IWIM, Prague, 2007
4. Ivakhnenko A.G., Ivakhnenko G.A., Savchenko E.A., and Wunsch D. Problems of Further Development of GMDH Algorithms: Part 2 // Pattern Recognition and Image Analysis , Vol. 12, № 1, 2002, p.6-18.

УДК 681.3.06

ПУСТОВАРОВ В.І.

СУПРОВОДЖЕННЯ КОНТРОЛЮ ВІДПОВІДНОСТІ ПРОГРАМ І МОДЕЛЕЙ ФОРМАЛЬНИМ СПЕЦИФІКАЦІЯМ ЗАДАЧ

Запропоновано механізми автоматизації доказового об'єктно-орієнтованого програмування задач за специфікаціями на базі контролю семантичної коректності взаємних перетворень кодів програм та специфікацій. Через ці механізми обґрунтовано використання аналітичного типу даних, який можна вбудувати до мов і систем програмування. Запропонований підхід дозволяє автоматизувати доказові перетворення кодів і специфікацій в системах автоматизації програмування і моделювання для задач формальної верифікації та синтезу кодів.

The proofing mechanisms of object-oriented programming are proposed for tasks with mutual transformations of program codes and formal specifications. Those mechanisms were used for justification to analytical data type which maybe built into programming languages and systems. Proposed approach allows automated proofing transformations of program codes and specifications in programming and simulation system for tasks of formal verification and code synthesis.

Однією з цілей систем автоматизації підготовки програм і моделей обладнання [3, 4, 7] є визначення підтримки і супроводження відповідності програм розв'язання задач формальним і формалізованим специфікаціям. Хоча сучасні системи програмування і моделювання (СПМ) не включають комплексів засобів достатньо повної перевірки відповідності кодів специфікаціям, комп'ютерна реалізація взаємних перетворень кодів і специфікацій [4, 6, 7] може дозволити організувати контроль і супроводження відповідності виконавчих кодів і специфікацій в процесі реалізації різноманітних програмних проектів. Створення автоматизованих засобів формальних перетворень дозволить скоротити витрати праці на створення нових і модифікацію попередніх проектів.

Перевірка коректності текстів програм і моделей, а також результатів формальних математичних перетворень, досі не автоматизовані, хоча створено ряд схем формальних специфікацій [2, 6, 7], які принципово можуть зберігатися на рівні комп'ютерних мов. Наприклад, мова Z-специфікацій [7] одержала ґрунтовну підтримку для задач управління базами даних. Механізми формальних аналітичних перетворень, стали основою математичних, а точніше, алгебраїчних методів розв'язання задач і побудови теорій типів даних [4, 6, 7] аж до взаємних перетворень математичних виразів і виконавчих кодів комп'ютера. Саме такі перетворення над моделями використовуються для формального доведення вручну тверджень відносно математичних формул і програмних кодів [4, 6, 7].

З самого початку формалізації та автоматизації задач аналітичних перетворень для полегшення впорядкування, систематизації та порівняння аналітичних виразів в логіці почали використовувати спеціальні форми [2, 4, 6], які сприяють зменшенню обсягів правил і таблиць перетворень при формальному синтезі та верифікації кодів. Найбільш загальними є графові схеми внутрішнього подання з підграфами G_{lj}, які поєднують підлеглі та рекурсивно зв'язані вузли включно з термінальними вузлами лексем G_{0j} та використовуються в різних трансляторах [1, 5]. Найбільш зручні спеціальні форми загальних конструкцій виразів і операторів комп'ютерних мов і мов специфікацій базуються на базовому порядку виконання операцій, основаному на їх передуваннях або пріоритетах. Ці форми є відображенням повного різноманіття конструкцій мов і відповідають таким вимогам:

- впорядкування складних структурованих конструкцій за послідовністю виконання елементів конструкції відповідно графового подання;
- визначення загального відношення порядку $\mathbf{S}_s(G_{lj})$ для всіх елементарних та структурованих або складених однотипних та однойменних об'єктів G_{lj} спрямованих графів (СГ) текстів комп'ютерних мов та мов специфікацій з розміщенням більш складних об'єктів після простих;
- впорядкування однорідних або подібних елементів конструкцій згідно з відношенням порядку $\mathbf{S}_s$ відносно кореневого вузла підграфа G_{lj} для конструкцій зі списками або операціями, повторюваними на одному рівні виразу;
- визначення спеціального порядку $\mathbf{S}_c(G_{lj})$ типів даних за їх структурною складністю та різноманіттям і впорядкованістю значень в доменах;
- впорядкування однорідних елементів конструкцій, що припускають довільну послідовність обробки або перестановку в списках аргументів та специфікацій даних, в порядку зменшення важливості їх змістовної обробки;
- використання єдиної системи впорядкування, нумерації і кодування однорідних та однотипних елементів конструкцій, в тому числі, операцій з можливістю перестановки аргументів (впровадження такої системи може зробити подання надлишковим, але повинно спростити порівняння за наявності спеціального механізму прискорення порівнянь);
- можливість побудови спеціально обмежених спеціальних форм для будь-яких підмножин операцій з високим пріоритетом.

Традиційна методика доказових перетворень та синтезу програм за специфікаціями [7] базується на операціях тотожних та імплікативних перетворень, які можна розглядати як узагальнення логічних операцій рівнозначності, нерівнозначності та імплікації [4], що використовується при аналітичних доведеннях. Щоб полегшити розрізнення дій перетворення в ланцюжках доведень від операцій відношення рівності виразів,

що операцію тотожності (тотожної рівності, рівності, визначення рівняння, рівнозначності) будемо позначати у формі $x \equiv y$ на відміну від $x = y$, використаного в [4], що співпадає з рівністю, а операнди називатимемо лівою частиною та правою частиною. За Хехнером [4] імплікацією називають перетворення, що визначає істинність слідування довільного математичного виразу y з виразу такого ж типу x, і записують її в формі $x \Rightarrow y$ або $y \Leftarrow x$. При цьому операнд x називають антецедентом мети (попередником), а y – консеквентом (наслідком). Далі називатимемо ці операції правосторонньою та лівосторонньою імплікаціями.

Група ключових трансформаційних дій утворює послідовні ланцюжки доказових перетворень, що зберігаються в протоколах доведень. Тому їх необхідно включити до списку основних операцій внутрішнього подання процесів перетворень, показані на Табл.1 в порівнянні з відповідними логічними операціями. Семантика цієї групи дій задається відношеннями тотожності та імплікативності між попередньою S_{j-1} та наступною S_j специфікацією або виконавчим кодом в ланцюжку перетворень.

Табл. 1. Типові операції перетворень специфікацій, моделей, програм та скриптів запитів в порівнянні з відповідними логічними операціями

Типи операцій	**Мови або процеси**	**Варіанти відповідності**
Y1. Двостороння еквівалентність або тотожність: $\equiv$	Перетворення специфікацій і кодів	Доцільність аналізу другого аргументу після першого
Y2. Лівостороння імплікація: $\Leftarrow$	Перетворення специфікацій і кодів	Необхідність аналізу першого аргументу після другого
Y3. Правостороння імплікація: $\Rightarrow$	Перетворення специфікацій і кодів	Необхідність аналізу другого аргументу після першого
X1. Рівнозначність: $\equiv$ ~^	Специфікацій; моделювання і програмування	Доцільність обчислення другого аргументу після першого
X2. Лівостороння імплікація: $\Leftarrow$: -	Специфікацій і логічного програмування	Необхідність обчислення першого аргументу після другого
? X3. Правостороння імплікація: $\Rightarrow$	? Специфікацій	? Необхідність обчислення другого аргументу після першого

В протоколах ланцюжків перетворень виконуються і фіксуються результати операцій еквівалентних перетворень '$\equiv$' і операцій лівосторонньої '$\Leftarrow$' та правосторонньої '$\Rightarrow$' імплікації, в яких стрілка вказує на менш загальну специфікацію S_j. Тобто на всій області визначення елементів та параметрів типу, або аргументів та параметрів фрагмента коду, ця специфікація S_j завжди має ті самі значення істинності, що відповідають обмеженням адекватності виконаних перетворень, але за межами цієї області S_j може не відповідати цим обмеженням.

В процесі доведення тотожність та імплікації реалізуються як композиції відповідних операцій для даних за типами з відокремленими елемен-

тами функціональних перетворень. При цьому ланцюжки тотожностей виконують тотожні перетворення, а ланцюжки односпрямованих імплікацій разом з тотожними перетвореннями – відповідні імплікативні доведення. В наведених нижче формулах збільшені знаки ‘≡’, ‘⇐’ і ‘⇒’ визначають тотожність та імплікативність ланцюжків перетворень і мають нижчий пріоритет, ніж звичайні логічні операції з такими ж назвами, що можуть зустрітися в специфікаціях і виконавчих кодах.

$$S_0 \equiv S_1 \equiv \ldots \equiv S_j \quad \equiv \quad S_0 \equiv S_j \qquad (1)$$

$$S_0 \equiv S_1 \Leftarrow \ldots \equiv S_j \quad \equiv \quad S_0 \Leftarrow S_j \qquad (2)$$

$$S_0 \equiv S_1 \Rightarrow \ldots \equiv S_j \quad \equiv \quad S_0 \Rightarrow S_j \qquad (3)$$

$S_0 \equiv S_1 \Rightarrow \ldots \Leftarrow S_j$ або $S_0 \equiv S_1 \Leftarrow \ldots \Rightarrow S_j$ – ланцюжки, що не мають сенсу.

Інша група ключових дій доказових перетворень протоколює тотожні перетворення специфікацій та їх перетворення імплікаціями на коди програм у комп’ютерній мові або в зворотному напрямку. Таблиці таких перетворень для специфікацій, кодів і тверджень відносно елементарних та складних об’єктів включають з одного боку, специфікації даних та зв’язків і обмежень, а з іншого боку відповідні коди виконавчих операторів та специфікації даних.

Для скорочення витрат на виконання доведень та зберігання їх результатів обґрунтована доцільність використання даних, інкапсульованих в об’єктно-орієнтованих типах. Якщо організувати традиційне успадкування разом з обмеженням множин значень окремих полів додержанням і визначити обмеження на правила композиції типів, то виникає можливість успадкування доведень відповідності типів об’єктів їх специфікаціям. Аксіоматичний базис доказових аналітичних перетворень повинен включати типи, що відповідають числовим математичним абстракціям, механізми для доведення відповідності та впорядкованості домену значень і комплексу методів класу об’єктів специфікаціям відповідного типу [4, 6, 7]. Це створює можливість поширення доведення коректних методів обробки загальних та математичних типів даних на похідні типи даних з обмеженими та успадкованими множинами значень їх елементів.

Можливі базові шляхи для первинних імплікативних доказових перетворень наведено в Табл. 2. Напрямки комплексування типів окремих полів при створенні класу визначаються обмеженнями: 1) односпрямованої впорядкованості всіх полів комплексного типу; 2) однакових наборів операцій батьківських типів; 3) доведеними наборами операцій похідного класу пересіченням наборів операцій батьківських типів. Для показаних напрямків можна довести імлікативність множин даних та коректність всіх методів типу або класу (абстрактні математичні типи, які не мають комп’ютерної реалізації виділено сірим кольором).

Табл. 2. Напрямки перетворення типів зі скороченням кардинальних чисел доменів за правосторонньою імплікацією $T_a \Rightarrow T_i$

Континуальні ⇒	Цілі необмежені ⇒	Натуральні необмежені ⇒	Необмежені без знака ⇒	Перенумеровані (кодовані)
Континуальні ⇒	Обмежені з плаваючою точкою ⇒	Цілі обмежені ⇒	Обмежені без знака ⇒	Перенумеровані (кодовані)
Континуальні ⇒	Обмежені з плаваючою точкою ⇒	З обмеженням порядку плаваючої точки ⇒	З обмеженням порядку і мантиси ⇒	Перенумеровані (кодовані)
Дискретні з довільними доменами ⇒	Дискретні з обмеженими доменами ⇒	Дискретні з впорядкованими доменами ⇒	Впорядковано кодовані ⇒	Перенумеровані (кодовані)

Для визначення діапазону числових або впорядкованих множинних даних в позначеннях [4] x_{imin},.. x_{imax}: T_i необхідно задати нижню x_{imin} і верхню x_{imax} межі значень конкретизованого типу T_i. Ці межі повинні гарантувати або імплікативно доводити адекватну роботу всіх методів (операцій і функцій) типу T_i відносно типу T_a в рамках обмежень, визначених аналітиком. Тобто, якщо для цілих числових типів границі їх діапазонів відповідають відношенням: $x_{imin} \le x_{amin}$ та $x_{amax} \le x_{imax}$, всі сусідні значення x_i відрізняються на 1, а для всіх операцій та функцій, зв'язаних з цими типами, можна довести за [4, 6, 7], що $f_a \Rightarrow f_i$, то $T_a \Rightarrow T_i$, і коректність типу T_i слідує з коректності типу T_a. Тобто, всі специфікації, операції, дії, функції, оператори, підпрограми, тощо, коректні для типу T_a, що відрізняються лише заміною типу даних на тип T_i, також будуть коректними.

Континуальні або дійсні числові типи повинні забезпечувати задану точність наближеного відтворення $x_i \approx x_a$ даних так, щоб $|x_a: T_a - x_i: T_i| \le \varepsilon$ та необхідний діапазон припустимих значень, а також додержання таких самих або не гірших ніж задані в специфікаціях точності та діапазонів значень при виконанні базових операцій над даними відповідного типу. В таких випадках конкретизовані специфікації доводяться правосторонньою імплікацією зі специфікацій базових абстрактно математичних типів з додаванням умов точності.

З позицій об'єктно-орієнтованого програмування будь-які класи об'єктів C повинні бути реалізовані, як структурне об'єднання полів даних F базового типу T з комплексом методів f=(f_o, f_f, f_s, f_p), до яких відносяться операції (оператори [1]) f_o, функції f_f (як математичні, так і програмні), підпрограми f_s, предикати f_p (в тому числі, предикати обмежень і відповідності специфікаціям). Розширення такого типу або класу за допомогою позначень визначення структур [4] може бути записане, як $C = F \rightarrow T \mid f_1$

$\rightarrow T \,|\ldots|\, f_k \rightarrow T\,|$, де k – кількість методів в типі. Відповідність первинних методів розширених типів або класів їх специфікаціям доводиться базовими методами доведення, що і було зроблено Е.Хехнером для базових абстрактних математичних типів [4].

Для обґрунтування подальших доказових перетворень важливо довести набір теорем для переходу по ієрархії типів від абстрактно математичних типів до поширених в сучасних мовах програмування комп'ютерних типів або класів даних:

1. ***Теорема обмеження доменів значень типів даних***, яка визначає імпликативне доведення коректності типів зі скороченими доменами з коректності базового типу при використанні такого самого набору методів.

2. ***Теорема скорочення кількості полів структурованих типів даних***, яка визначає імпликативне доведення коректності з вилученням наборів полів з коректності базового типу при незмінному наборі методів.

3. ***Теорема побудови композиційних методів в рамках одного типа***, яка визначає умови імпликативного доведення коректності композиційних методів з коректності методів базового типа.

4. ***Теорема побудови композиційних типів з окремих полів***, яка визначає імпликативне або тотожне доведення коректності композиційних типів з коректності базових типів при незмінному наборі методів.

5. ***Теорема доведення функціональних розширень типів***, яка визначає умови імпликативного доведення коректності композиційних методів з методів базового типа та операцій розширення, взятих з зовнішніх типів.

Для спрощення доказів важливо, щоб в обмеженому типі T_{i} використовувались ті самі значення, що і в базовому типі T_{a}, тоді в типі T_{i} залишаться справедливими всі відношення порядку. В Табл. 3 зібрані основні алгебраїчні методи ручної мінімізації, формальної верифікації та синтезу виразів і кодів засобів розв'язання задач (ЗРЗ) і визначено основні шляхи, правила і шаблони різних функцій та операцій $\mathrm{F_a}(G_{\mathrm{SIn}}, G_{a\mathrm{C}\bullet})$ семантичних перетворень. Аналітичні дані специфікацій G_{SIn} і правила перетворень $G_{\mathrm{C}\bullet lj}$ задаються кореневими вузлами відповідних СГ.

Базовий порядок обробки встановлюється за правилами управління Cnt, поданими через кореневі вузли $G_{\mathrm{C}\bullet lj}$ графів спрямованих перетворень відповідно з послідовністю їх використання в ланцюжках перетворень. Спочатку вони найчастіше перетворюють вхідні вирази на проміжну внутрішню форму Inout, а потім – на заключну цільову форму перетворення.

Табл. 3. Базові типові шляхи семантичних перетворень специфікацій, моделей та програм

Тип	Спосіб перетворення	Внутрішня інформаційна база	Ланцюжки
Мінімізація подання кодів	1. Перетворення фрагмента на спеціальну форму ⇒ виключення повторень ⇒ перетворення на мінімальну форму	Cnt: Правила перетворення: 1.1.G_{Cntlj} – вхідних виразів G_{In} на часткову спеціальну форму G_{Int}; 1.2.G_{Cmnlj} – обмеженої спеціальної форми G_{Int} на мінімальну G_{Mn}. Inout: поточне внутрішнє подання $G_{In} \Rightarrow G_{Int} \Rightarrow G_{Mn}$;	$G_{In} \Rightarrow$ $G_{Int}= F_{Cn}(G_{In}, G_{Cntlj}) \Rightarrow$ $G_{Mn}= F_{Mi}(G_{Int}, G_{Cmnlj})$
Мінімізація з наближеннями	2. Спосіб 1 ⇒ імплікативне перетворення мінімальної форми	Cnt: Правила перетворення способу 1 (1.1 і 1.2) з доданням: 1.3. Правил перетворення G_{Cmalj} мінімальної форми G_{Mn} на мінімальну наближену форму G_{Mna}. Inout: $G_{In} \Rightarrow G_{Int} \Rightarrow G_{Mn} \Rightarrow G_{Mna}$;	Ланцюжки способу 1 ⇒ $G_{Mna}=F_{Ma}(G_{Mn}, G_{Cmalj})$
Формальна верифікація	3. Формування виразів для кодів визначення цільових змінних ⇒ підстановка до специфікації ⇒ доведення істинності	Cnt: 3.1.Правила підстановки G_{Csslj} виразів програм G_{ECd} до специфікацій G_{Sp}; правила перетворення: 3.2.(1.1)G_{Cntlj} – результатів підстановки G_{Rs} на обмежену спеціальну форму G_{Int}; 3.3.(1.2)G_{Cmnlj} – часткової спеціальної форми G_{Int} на мінімальну G_{Mn} з результатом *true* Inout: $G_{Sp} \cup G_{ECd} \Rightarrow G_{Rs} \Rightarrow G_{Int} \Rightarrow$ *true*;	$G_{Sp} \cup G_{ECd} \Rightarrow$ $G_{Rs}=F_{Sb}(G_{ECd}, G_{Sp}, G_{Sbnlj}) \Rightarrow$ $G_{Int}=F_{Cn}(G_{Rs}, G_{Cntlj}) \Rightarrow$ $G_{Mn}= F_{Mi}(G_{Int}, G_{Cmnlj}) \equiv$ *true*
	4. Декомпозиція задачі ⇒ доведення відповідності окремих фрагментів ⇒ доведення відповідності композиції	Cnt: Правила: 4.1.Розширення G_{Gelj} специфікацій проміжними цілями; 4.2.G_{Dtslj} – декомпозиції специфікацій відносно проміжних цілей G_{Gin}; Доведення частин за способом 3. Inout: $G_{Sp} \cup G_{ECd} \Rightarrow G_{Sp} \cup G_{ECd} \cup G_{Gin} \Rightarrow G_{Rsn} \Rightarrow G_{Int} \Rightarrow$ *true*;	$G_{Sp} \Rightarrow G_{Gin}$; ⇒ $G_{Rsn}= F_{Dc}(G_{Sp} \cup \cup G_{ECd} \cup G_{Gin})$ $G_{Int}=F_{Sb}(G_{Rsn}, G_{Sp}, G_{Cntlj}) \Rightarrow$ $G_{Mn}= F_{Mi}(G_{Int}, G_{Cmnlj}) \equiv$ *true*
Формальний синтез	5. Декомпозиція специфікацій задачі на послідовність ⇒ синтез окремих фрагментів коду	Cnt: Правила декомпозиції 4.1 і 4.2. 5.1. Правила перетворення G_{Cvcnlj} специфікацій G_{Sp} на коди G_{ECdn}. Inout: $G_{Sp} \Rightarrow G_{Sp} \cup G_{Gin} \Rightarrow G_{Rsn} \Rightarrow G_{ECdn}$;	$G_{Sp} \Rightarrow G_{Gin}$; ⇒ $G_{Rsn}= F_{Dc}(G_{Sp} \cup \cup G_{ECd} \cup G_{Gin})$ $G_{ECdn}= F_{Cv}(G_{Rsn}, G_{Cvcnlj})$

	6. Пошук аналогів серед наявних ЗРЗ ⇒ визначення різниці типів і зв'язків специфікацій ⇒ синтез окремих фрагментів коду	Cnt: 6.1.Критерії близькості $G_{\mathrm{Cnre}lj}$ специфікацій задач G_{Sp} до наявних ЗРЗ; правила $G_{\mathrm{Sd}n}$ визначення різниці між специфікаціями і кодом ЗРЗ Inout: $G_{\mathrm{Sp}} \Rightarrow G_{\mathrm{Sp}} \cup G_{\mathrm{ECp}} \Rightarrow G_{\mathrm{Sp}} \cup G_{\mathrm{ECp}} \cup \cup G_{\mathrm{ECd}n} \Rightarrow G_{\mathrm{ECp}} \cup G_{\mathrm{ECd}n}$	$G_{\mathrm{Sp}} \Rightarrow$ $G_{\mathrm{ECp}}=\mathrm{R}_{\mathrm{Cv}}(G_{\mathrm{Sp}}, G_{\mathrm{Cnre}lj})$; $G_{\mathrm{Sd}n}= G_{\mathrm{Sp}} \backslash G_{\mathrm{ECp}}$; $G_{\mathrm{ECd}n}= \mathrm{F}_{\mathrm{Cv}}(G_{\mathrm{Sd}n}, G_{\mathrm{Cvc}nlj})$

В задачах **мінімізації або оптимізації внутрішнього подання** вхідними даними є вузли СГ G_{In} внутрішнього подання, утвореного синтаксичним аналізатором. Потім формуються проміжні дані внутрішньої обмеженої спеціальної форми як вузли СГ G_{Int} за правилами $\mathrm{F}_{\mathrm{Cn}}(G_{\mathrm{In}}, G_{\mathrm{Cnt}lj})$ перетворення на цю форму. Вони також зберігають інформацію управління у формі вузлів СГ $G_{\mathrm{Mn}}= \mathrm{F}_{\mathrm{Mi}}(G_{\mathrm{Int}}, G_{\mathrm{Cmn}lj})$ послідовність використання яких визначається режимами перетворення. Насамкінець вирази мінімальної наближеної форми в форматі вузлів СГ G_{Mna} за правилами перетворення обмеженої спеціальної форми на мінімальну $\mathrm{F}_{\mathrm{Ma}}(G_{\mathrm{Mn}}, G_{\mathrm{Cma}lj})$, які також зберігають інформацію управління у формі вузлів СГ $G_{\mathrm{Cma}lj}$, послідовність використання яких визначається режимом наближеного перетворення.

Один з підходів до **формальної верифікації** методом аналітичних перетворень технологічно дуже схожий на базовий підхід до мінімізації внутрішнього подання, але використовує дещо іншу послідовність дій. Спочатку вузли СГ об'єднання $G_{\mathrm{Sp}} \cup G_{\mathrm{ECd}}$ специфікацій та виконавчих кодів цілеспрямовано реконфігуруються на подання формул результатів G_{Rs} виразами вхідних даними за правилами підстановки $\mathrm{F}_{\mathrm{Sb}}(G_{\mathrm{In}}, G_{\mathrm{Sp}}, G_{\mathrm{Sbn}lj})$ виразів присвоювань та інших операторів виконавчої мови. Далі необхідно виконати співставлення або доведення відповідності результатів G_{Rs} специфікації за правилами $\mathrm{F}_{\mathrm{Cn}}(G_{\mathrm{Int}}, G_{\mathrm{Cnt}lj})$, сформувавши спеціальну форму подання кодів G_{Int} як проміжну. Після двох заключних кроків, повністю аналогічних останнім крокам мінімізації 1.2 та 1.3 (за табл. 3) ми повинні одержати або тотожне логічне значення *true*, або вираз, що визначає умови відповідності кодів специфікаціям.

Більш загальний підхід до **формальної специфікації** як і задачі **формального синтезу** відрізняються від попередніх задач семантичних перетворень необхідністю декомпозиції задач на частини, починаючи з текстів початкових специфікацій. Ці перетворення виконуються за допомогою загальних правил декомпозиції $G_{\mathrm{Rs}n}=\mathrm{F}_{\mathrm{Dc}}(G_{\mathrm{Sp}} \cup G_{\mathrm{ECd}} \cup G_{\mathrm{Gi}n})$ через визначення проміжних цільових змінних $G_{\mathrm{Gi}n}$ або на базі пошуку ЗРЗ, найближчих за специфікацією, і розміщених в базі знань СПМ як обґрунтовані ЗРЗ. Для формального синтезу ключове значення має повнота правил еквівалент-

них перетворень елементів специфікації на коди G_{ECdn}= $F_{Cv}(G_{Sdn}, G_{Cvcnlj})$ [7], де всі ключові та функціональні елементи подаються у вигляді СГ.

Розв'язання задач семантичних перетворень можна полегшити за допомогою створення аналітичного типу даних. ***Аналітичний тип даних*** (як вбудований тип даних комп'ютерної мови) з комплексом параметрів, який визначає таблиці передувань і таблиці синтаксичних та семантичних правил, дозволяє одержати узагальнений механізм, доступний для використання на етапах проектування, аналізу та виконання операторів програм і математичних виразів в програмах і книжкових текстах. Поняття аналітичного типу визначає шлях реалізації узагальнення формальних і формалізованих математичних та комп'ютерних моделей, яке полегшує обробку виразів і всіх видів операторів програм на всіх етапах проектування.

Зовнішні властивості використання ***аналітичного типу*** спрощують використання і обробку **математичних виразів будь-яких типів і складності** на етапі проектування програмних об'єктів та накопичення баз знань. Схожим чином перетворюються окремі конструкції і фрагменти програм, процедур, функцій, специфікацій будь-яких об'єктів та інших модулів програм, моделей та запитів як інтегрованих об'єктів і забезпечують можливість їх модифікації, контроль спрямованості модифікацій. Крім того, за рахунок спеціального кодування полів вони можуть дозволити стандартизувати всі базові види семантичної обробки СМП.

Додаткові особливості власної поведінки аналітичного типу та даних такого типу визначають реакцію середовища обробки на етапі проектування або перестроювання і спирається на можливість їх використання для програмування та задач аналітичної обробки, а також для виконання різних видів навчання шляхом накопичення правил і уточнення коефіцієнтів.

Таким чином, дані ***аналітичного типу*** формуються на етапах лексичного та синтаксичного аналізу специфікацій проблемної галузі аналізу і реконфігурації конструкцій. Вони представляють різні типи текстів на комп'ютерних і природних мовах і використовуються в процесі перетворень на етапі оптимізації та таких інших етапах семантичної обробки, як генерація виконавчих кодів або формальне доведення коректності та складання відповідних протоколів.

Прагматична корисність аналітичного типу визначається **іменуванням** або **кодуванням** аналітичних об'єктів, як єдиного цілого, що здатне відтворювати семантичні характеристики, призначені для аналізу можливості та доцільності використання аналітичних моделей даних. Інший напрямок використання прагматичної корисності полягає в можливості вживання аналітичного типу не тільки в комп'ютерних мовах моделювання, програмування та запитів для цілей специфікації. Прагматична корисність проявляється і в формально визначеному розширенні мови математики на кшталт мови специфікації Z, яка може розглядатися як розширення формалізованої природної мови.

Особливі зовнішні **властивості** ***аналітичного типу*** даних пов'язані з поліморфізмом і можливістю супроводження еквівалентних і наближених формул, виразів і правил. Вони забезпечують визначення та дотримання напрямків перетворень аналітичних виразів і задаються наборами правил та послідовностями їх застосування.

Аксіоматика ***аналітичного типу*** повинна визначати правила або властивості для забезпечення повноти функціональних та багатозначних перетворень в межах цього типу. Це можна зробити на основі використання аксіоматики теорії множин та теорії наближених обчислень для комплексів операцій та функцій перетворення, еквівалентності, імплікативності та/або наближеності. Крім того, необхідно визначити аксіоми підтримки структурованості подання, кодування або іменування, цільової спрямованості перетворень, можливості доказового (обґрунтованого) обчислення при наявності необхідних даних. Включення наборів аксіом для оцінок характеристик реалізації перетворень за різними види складності дозволяє попередньо оцінити можливий час розв'язання задач доведення і ліквідації фундаментальних суперечностей визначень, неоднозначності спрямувань в перетвореннях.

Власна поведінка та реакції на середовище даних ***аналітичного типу*** визначаються операціями та функціями, які в основному схожі на множини операцій та операторів алгебри множин і мови маніпуляції даних в SQL. Крім того до набору операцій повинні включати правила підстановок та перетворень, зібрані в Табл. 3. На основі послідовностей таких операцій, визначених для кожної задачі семантичної обробки, виконуються еквівалентні перетворення різних напрямків.

Аналітичний тип природно розширюються виразами на метамовах типами, які визначають комп'ютерні та природні мови. При цьому похідні функціональні або операційні типи від аналітичного типу визначають базові механізми внутрішніх перетворень СПМ. Вони можуть задавати механізми автоматизованої генерації методів та структур даних будь-яких інших типів та класів об'єктів, що вбудовуються в мову або реалізуються за її допомогою.

Область визначення та припустимих значень даних аналітичного типу при реалізації визначається областями значень окремих полів вузла графу виразів мов і метамов. Ряд полів визначає покажчики на підлеглі або батьківські об'єкти цього ж типу, які можуть бути рекурсивними. Значення кодів полів операцій, відношень, зв'язків або термінальних позначень і полів типів їх результатів повинні однозначно кодуватися в СПМ і у випадках можливості різної довжини результатів включати додаткові коди довжини.

Висновки

Сучасні узагальнення внутрішнього подання моделей і програм дозволяють зберігати не тільки проміжні результати їх взаємних перетворень, але і протоколи обґрунтувань та доведень при перетвореннях. Викорис-

тання схожих форматів внутрішнього подання для даних, виразів і операторів широкого спектру задач дозволяє створити набори базових методів аналітичного типу для внутрішніх перетворень при мінімізації кодів, формальній верифікації та формальному синтезі кодів. Механізми перетворень мають однакову структуру правил роботи ядра при різних підходах та методиках побудови та перетворень програм і моделей. Однак результати автоматизованих доказових перетворень істотно залежать числа варіантів заповнення вбудованих баз знань, а час вибору найкращого варіанту перебудови – від числа варіантів послідовності застосування правил доведення. Тому можливість та доцільність використання наведеного підходу визначається характеристиками часової складності окремих задач доведення і попереднього накопичення в базі знань найбільш корисних варіантів перетворень для уникнення повного перебору варіантів перетворень за можливими напрямками.

Список посилань

1. Ахо А., Сети Р., Ульман Дж. Компиляторы: принципы, технологии, инструменты: Пер. с англ. – М.: Издательский дом Вильямс, 2001. – 768 с.
2. Лисков Б., Гатэг Дж. Использование абстракций и спецификаций при разработке программ. Пер. с англ. М.: Мир, 1989. 424 с.
3. Пустоваров В.І. Особливості трансформаційного підходу до реалізації мов програмування і моделювання. Вісник Національного технічного університету “Київський політехнічний інститут”. Інформатика, управління та обчислювальна техніка. К., «Век+», 2002, 38, с. 73-80.
4. Hehner E.C.R. Practical theory of programming. Springer-Verlag, New York, 1993 – 243 p.
5. Muchnick S.S. Advanced compiler design and implementation – San Francisco, Morgan Kaufmann Publishers, 1997. - 856 p.
6. Pierce Benjamin C. Types and Programming Languages. MIT Press, Cambridge, 2002. 620 p
7. Woodcock J., Davies J. Using Z. Specification, Refinement, and Proof. C.A.R. Hoare Series editor, 1995 – 390 p.

УДК 004.724.4(045)

КУЛАКОВ А.Ю.

СПОСОБ ПОВЫШЕНИЯ ЭФФЕКТИВНОСТИ GRID СИСТЕМ НА БАЗЕ ВИРТУАЛЬНЫХ СЕТЕЙ VPN

Предложен способ повышения эффективность функционирования GRID систем. Проведен анализ изменений пропускной способности каналов передачи данных на базе объединенных деревьев доставки информации для сетей VPN. Показано, что распределение нагрузки по маршрутизаторам DR, а также балансировка пропускной способности при многоабонентской доставке информации существенно зависит не только от физической, но и от логической организации распределенной системы

A way to improve the effectiveness of the GRID systems is proposed. The analysis of changes in channel bandwidth capacity based on the combined trees information delivery of LSP tunnels VPN networks is adopted. It has been shown that load distribution of the DR as well as balancing bandwidth capacity of multicast information delivery depends not only on the physical but also on the logical organization of a distributed system

Введение

В связи с расширением области применения GRID технологий, увеличения доли мультимедийного трафика в компьютерных сетях, актуальной становится задача повышения эффективности многоабонентской доставки информации. Особую важность эта задача приобретает в GRID системах, где состав группы абонентов может меняться динамически. В связи с этим, статические алгоритмы многоабонентской маршрутизации не эффективны, так как не могут в полной мере обеспечить надежную доставку информации при динамическом изменении топологии GRID системы. Наиболее простым и надежным способом доставки информации является лавинообразное распространение пакетов [1]. Существенным недостатком использования лавинного алгоритма для формирования маршрутной информации в GRID системах является то, что с увеличением количества и подвижности узлов, резко возрастает объем служебной информации в сети. В свою очередь, основным недостатком алгоритма маршрутизации от источника [2] является сложность реализации маршрутизаторов и значительный объем передаваемой системной информации внутри пересылаемого пакета.

В настоящее время для построения GRID систем все чаще используется многопротокольная коммутация на основе меток (MPLS – Multiprotocol Label Switching).

Формирование виртуального дерева VPN

Для динамической маршрутизации в GRID системах наиболее эффективно использовать многоабонентскую передачу информации на базе протокола «точка – многоточие» с использованием меток. Существуют два протокола рассматриваемые IETF (Internet Engineering Task Force) для построения LSP в сетях MPLS [3]: протокол резервирования ресурсов RSVP-TE (Resource Reservation Protocol – Traffic Engineering) и протокол распределения меток LDP (Label Distribution Protocol). Оба протокола могут быть расширены для передачи информации в LSP режиме «точка – многоточие», однако, RSVP-TE строит деревья «точка – многоточие» от корня к конечным вершинам, а LDP наоборот. В случае одновременной передачи одного IP-пакета по нескольким адресам, распределение меток LDP формируется после групповой передачи. В свою очередь, протокол RSVP-TE специально разработан для масштабируемого группового обслуживания с использованием сетей VPN и мульти-VPN на основе группового дерева. Поэтому, использование протокола RSVP-TE для совмещения резервирования ресурсов и организации LSP для различных потоков данных является более эффективным для поставленной задачи.

Применение RSVP-TE позволяет оптимизировать GRID систему под конкретные задачи путем создания туннелей LSP на основе VPN. Сообщения этих протоколов передаются от одного узла сети к другому в соответствии с данными об IP-адресах маршрута.

Пусть существует множество сетей VPN, объединенных в дерево доставки, в которое входят деревья с объединенными или единичными групповыми деревьями (рис.1).

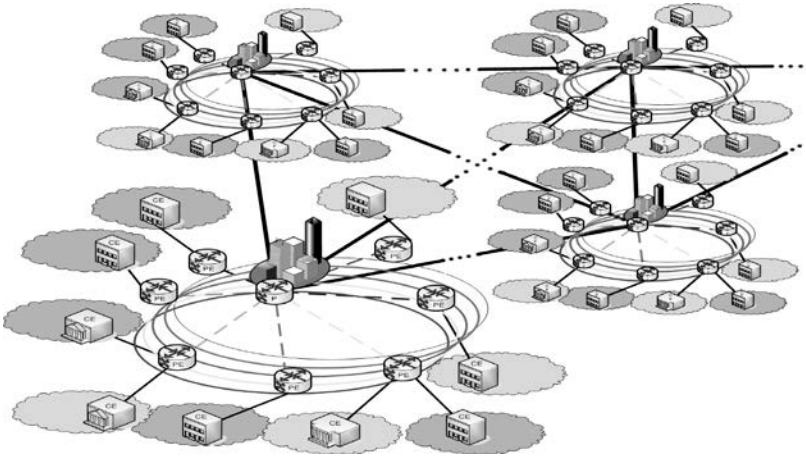


Рис. 1. Множество сетей VPN, объединенных в дерево доставки

Необходимо отметить, что каждый граничный маршрутизатор должен располагаться как можно ближе к корню дерева (Root), обычно, такой граничный маршрутизатор является абонентом многоабонентской доставки. В этом случае, все граничные маршрутизаторы (PE) посылают единому Root свои пакеты через LSP туннель ядра для получения внешней метки, осуществляя проверку предварительно назначенной внутренней метки соответствующему маршрутизатору доставки (DR) для данной VPN группы, и заменяют метку VPN группы на метку всего дерева. В таблице маршрутизатора DR существует вторая метка (выходная метка) для передачи информации вниз по дереву доставки другим VPN, как показано на рис. 2.

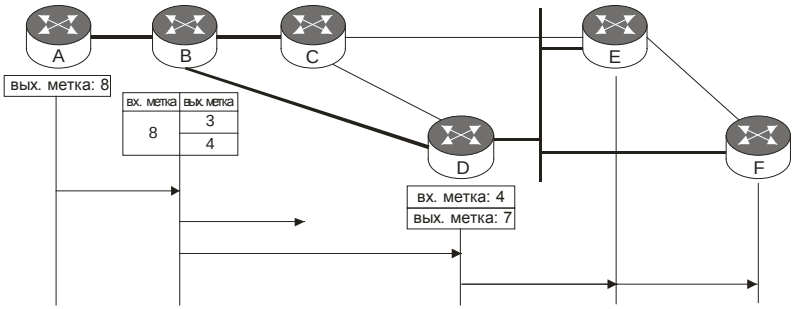


Рис. 2. Процесс передачи информации вниз по дереву доставки другим VPN

Выбор между единичным деревом и объединенным деревом доставки зависит от доступных ресурсов. Объединенное дерево обеспечивает надежность и минимальную задержку, за счет своей сложности и состояния передачи, поэтому защищенное одиночное групповое дерево в нашем случае является наиболее приемлемым. Перестройка дерева осуществляется при помощи сигнального протокола «точка-многоточие» LSP поддерживаемого RSVP-TE.

Для оценки состояния пропускной способности будем рассматривать модель, полученную в результате объединения маршрутизаторов PE с абонентами в сети VPN по не которым признакам. Предлагается не изменять физическую топологию системы, а применять VPN сети для поддержки необходимой структуры. Информация, предназначенная всем узлам GRID системы, которые принадлежат множеству сетей VPN объединенных в дерево доставки, передается при помощи протокола «точка-многоточие». Для этого необходимо, чтобы все маршрутизаторы PE передавали информацию корню дерева (Root). Если маршрутизатор не владеет информацией о составе группы, то передача информации осуществляется вниз по дереву граничным маршрутизаторам.

Для примера рассмотрим доставку пакета от абонента AMb VPN сети A всем абонентам VPN сети B (рис. 3) и определим значение пропускной способности, которое не эффективно используется во время передачи.

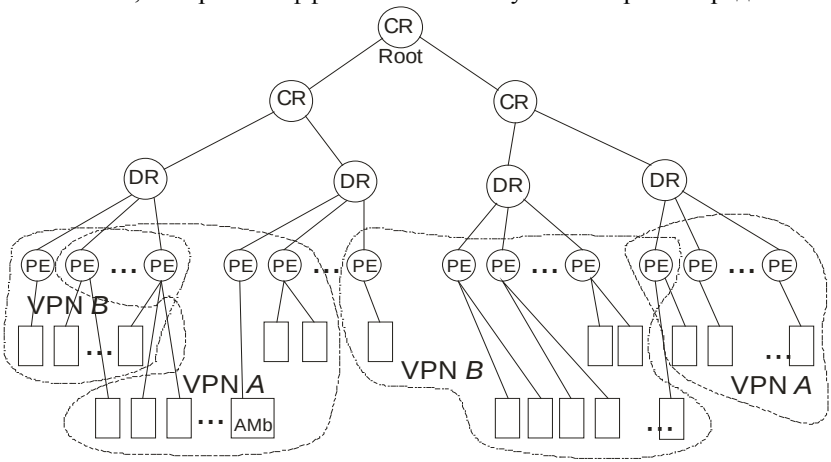


Рис. 3. Пример сети

Для этого необходимо отметить, что VPN сети A и B объединены между собой маршрутизаторами DR, которые в свою очередь связаны между собой корневыми маршрутизаторами CR. Для передачи информации абонент AMb VPN сети A передает информацию в маршрутизатор CR Root через маршрутизаторы PE и DR. Корневой маршрутизатор передает информацию маршрутизаторам DR, которые входят в состав VPN B, и далее осуществляется передача информации вниз по дереву через маршрутизаторы PE до абонентов (рис.4).

Общую пропускную способность (BW_{LSP}) туннеля LSP сети VPN можно вычислить аналитически относительно соединения «точка-точка». Если свести многоадресную и широковещательную передачу всех VPN к единице, то пропускную способность можно определить по формуле:

$$BW_{LSP} = K * \left(N * M * 2 + M \sum_{i=1}^{\log_2(N)} 2^{i-1} * (2i-1) \right), \qquad (1)$$

где: K – количество корневых маршрутизаторов CR,

N – количество маршрутизаторов DR,

M – количество абонентов.

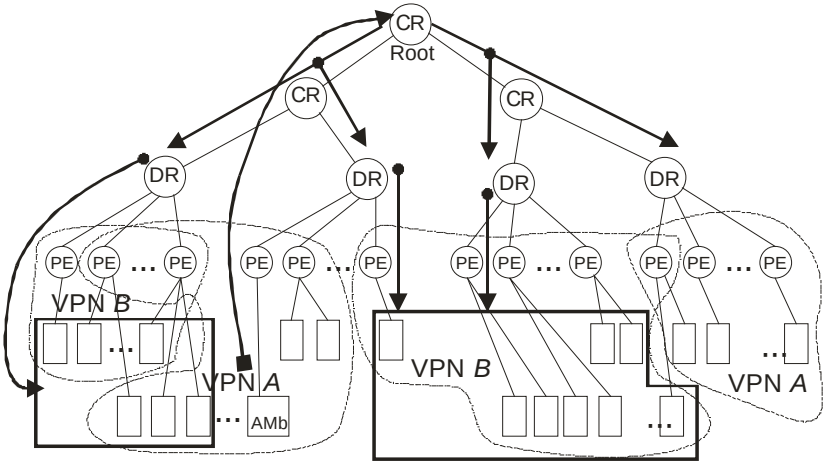


Рис. 4. Процесс доставки пакета от абонента AMb VPN сети A всем абонентам VPN сети B

С другой стороны, если дерево передает информацию по соединению «точка-многоточие» для каждого из туннелей LSP сетей VPN, общая пропускная способность (BW_{mLSP}) может быть представлена в виде:

$$BW_{mLSP} = K * \left(1 + \log_2(N) + \sum_{i=1}^{\log_2(N)} 2^i + M * N\right), \qquad (2)$$

где K – количество корневых маршрутизаторов CR,
N – количество маршрутизаторов DR,
M – количество абонентов.

После объединения сетей VPN в дерево доставки, изменяется только последнее слагаемое в (2), которое принимает вид $Z * N$, где Z ($Z \geq M$) обозначает среднее количество вершин, которые зависят от маршрутизатора DR в «точка-многоточие» туннеля LSP, после равномерного распределения K деревьев на W групповых деревьев ($W \leq K$).

Увеличение пропускной способности (BW_{abLSP}) возникает при условии, когда объединенное дерево VPN A не содержит в себе маршрутизаторов PE из группы VPN B. При этом передача информации через такой маршрутизатор PE считается бесполезной. В этой ситуации, маршрутизаторы DR поддерживают количество предающих связей равное $W * Z$. Необходимо отметить, что $M \leq Z \leq L$.

$$BW_{abLSP} = K * \left(1 + \log_2(N) + \sum_{i=1}^{\log_2(N)} 2^i + Z * W\right), \qquad (3)$$

где: K – количество корневых маршрутизаторов CR;
N – количество маршрутизаторов DR;
Z – среднее количество вершин, зависящих от маршрутизатора DR;
W – количество групповых деревьев.

В результате, для предложенного примера сети (рис. 3) значение пропускной способности ($BW_{(ab)}$), которое не эффективно используется во время передачи от абонента AMb VPN сети A всем абонентам VPN сети B можно определить как разность (2) от (3), представленное в виде:

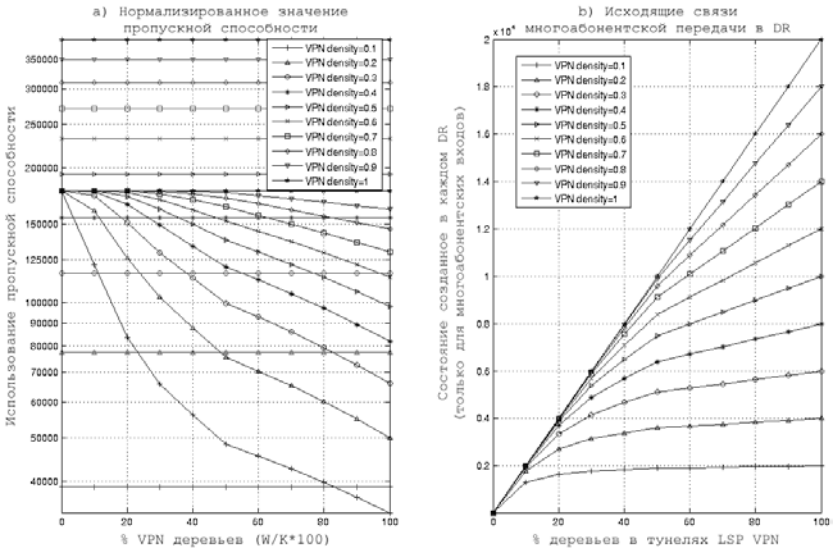
$$BW_{(ab)} = K*(M*N - Z*W) \quad (4)$$

Необходимо отметить, что распределение сетей VPN случайным образом является не эффективным для построения больших распределенных систем. Невзирая на то, что пропускная способность зависит от вида топологии сети, необходимо проводить анализ распределения VPN сетей для повышения скорости процесса передачи данных. Таким образом, можно сделать вывод, что анализ общей пропускной способности туннелей LSP сетей VPN во многом зависит от правильной организации распределенной системы.

Моделирование

Для оценки преимущества объединенных деревьев, проведем моделирование распределенной системы при условии, что количество VPN равно 1000. Возьмем по 20 граничных маршрутизаторов для каждого из P маршрутизаторов DR. Расположим сети VPN по случайному закону распределения таким образом, чтобы они были равномерно разделены между всем маршрутизаторами DR. Для большей эффективности будем рассматривать дополнительные параметр – плотность распределения LSP туннелей VPN по маршрутизаторами DR (VPN density). Предположим, что все VPN сети имеют одинаковую плотность. Определим, что количество объединенных деревьев W изменяется в приделах от 1 до 1000. На каждой итерации, значение Z рассчитывается путем группировки случайным образом разных VPN сетей в объединенное дерево, которое в свою очередь относится к другой группе VPN сети в маршрутизаторе DR.

Результаты моделирование представлены на рис.5. Рисунок 5.a показывает отношения использование пропускной способности от количества объединенных деревьев системы. Параллельные линии отображают состояние пропускной способности при использовании метода передачи информации «точка-точка», а кривые – «точка-многоточие» с использованием групповых деревьев. На рисунке 5.b показано количество исходящих связей маршрутизаторов DR при многоабонентской доставке информации.



а) б)

Рис. 5. Оценка пропускной способности и количества исходящих связей в GRID системе с использованием объединенных деревьев на базе туннелей LSP сетей VPN.

Выводы

Предложенный метод многоабонентской доставки информации позволяет повысить эффективность функционирования GRID систем за счет более эффективного использования пропускной способности каналов передачи данных на базе объединенных деревьев доставки информации туннелей LSP сетей VPN.

Список использованных источников

1. Perkins C., Belding-Royer E. and Das S., “Ad hoc on demand distance vector routing protocol,” July 2003, IETF RFC 3561, July 2003.
2. Broch J., Johnson D. B. and Maltz D. A., “The dynamic source routing protocol for mobile ad hoc networks,” IDV 08, IETF, February 2003.
3. A.Chaak, “Quality of service and Traffic Engineering in Consolidated Core and Metro Networks”, University of Toronto, September 2004

УДК 004.724.4(045)

ДАВИДЕНКО И.Н.

СПОСОБ ПОВЫШЕНИЯ ЭФФЕКТИВНОСТИ ПРОЦЕССА МАРШРУТИЗАЦИИ В МОБИЛЬНЫХ СЕТЯХ БОЛЬШОЙ РАЗМЕРНОСТИ

The method of routing process effectiveness increasing based on agent systems is offered. Analysis is taken on dependency between network neighborhood distribution of first and second network topology degree and agent place. As shown, topology modification causes distribution density changes by non-linear law.

Предложен способ повышения эффективности процесса маршрутизации на основе системы агентов. Проведен анализ зависимости изменения распределения сетевого окружения первого и второго порядков топологии сети и место расположения агентов. Показано, что с изменением топологии плотность распределения изменяется по нелинейному закону.

Введение

Для современных компьютерных сетей характерным является наличие разнородного трафика, при этом существует устойчивая тенденция увеличения доли мультимедийного трафика [1, 2, 3]. Каждый вид трафика предъявляет свои требования к качеству обслуживания (*QoS*) [4]. К ним относятся требования к пропускной способности, задержкам, изменению (флуктуации) задержки, надежности и относительной загруженности узла.

В отличие от мультимедийного трафика данные передаются, как правило, с неравномерной плотностью потока, пачками, и прибывают в непредсказуемые промежутки. В связи с этим, полоса пропускания, используемая для мультимедийного трафика, должна быть защищена от трафика данных, и наоборот.

Для этих целей, иерархическая модель сети является наиболее предпочтительной, так как позволяет создать наиболее устойчивую структуру сети и более рационально распределить ресурсы [5]. Также достоинством иерархической модели сети является более высокий уровень защиты данных. Все узлы такой сети мобильны, связываются друг с другом динамически произвольным образом без участия централизованного управления и базовых станций.

Основным фактором, который влияет на процесс передачи информации в мобильных компьютерных сетях, является их динамическая топология, поэтому эффективность функционирования таких сетей в значительной степени зависит от решения задачи маршрутизации. Кроме того, для обеспечения гарантированного качества обслуживания в мобильных

сетях наиболее эффективным подходом является интеграция элементов поддержки качества обслуживания в протоколы маршрутизации.

Обзор и анализ существующих решений

Для компьютерных систем большой размерности не существует единого эффективного алгоритма маршрутизации. В связи с этим компьютерные системы разбиваются на отдельные подсистемы – домены. При этом задача маршрутизации делится на [6] задачи внутридоменной и междоменной маршрутизации.

Большая часть алгоритмов внутридоменной маршрутизации используют алгоритмы маршрутизации по состоянию канала *(link-state protocol)* для распространения топологической информации по всей сети с помощью служебных сообщений [7]. Для распространения служебных сообщений чаще всего используется лавинный алгоритм (*flooding*). Суть алгоритма заключается в том, что получив управляющий пакет, система рассылает его во всех возможных направлениях, кроме линии связи, по которой пакет пришел. К основным недостаткам алгоритма маршрутизации по состоянию канала можно отнести низкую масштабируемость , по причине большого количества служебного трафика создаваемого в сетях большого размера сообщениями об изменении состояния узлов. Такая же проблема возникает при частых изменениях топологии сети.

В отличие от этого, при междоменной маршрутизации управление трафиком обеспечивает снижение уровня загруженности каналов путем отправки одного потока данных группе получателей, а не каждому получателю индивидуально с использованием многоадресной маршрутизации. Вследствие чего, задача междоменной маршрутизации состоит в построении максимально устойчивого минимального покрывающего дерева.

Особую актуальность принимают задачи оптимизации управления трафиком в мобильных сетях, связанных с использованием динамических алгоритмов маршрутизации, при которых объем управляющего трафика зависит от частоты изменений в сети.

Одной из основных задач управления функционированием компьютерной сети при передаче мультимедийного трафика является организация эффективной системы доставки информации. Решение данной задачи осуществляется с помощью процедуры конструирования трафика (*TE – Traffic Engineering*) [8] основной целью, которой является обеспечение равномерной загрузки сети. В этом заключается основное отличие данной задачи от задачи оптимальной маршрутизации.

Средства традиционной маршрутизации не удовлетворяют требованиям *QoS*, требованию равномерного распределения нагрузки по каналам связи, не обеспечивают достаточной оперативности при изменении маршрутов в сети, вызванных перемещением абонентских систем [9]. Еще одним недостатком является необходимость передачи регулярных обнов-

лений маршрутной информации даже при незначительном колебании загрузки каналов передачи или изменением топологии системы. [10].

Одним из подходов к решению данной задачи является оптимизация сети путем динамического распределения трафика, т.е. в режиме реального времени решать задачу динамического назначения маршрутов, удовлетворяющую требованиям к параметрам *QoS*, и обеспечивать равномерную загрузку сети.

В этом случае эффективность задачи маршрутизации и, в целом, задачи конструирования трафика в значительной мере зависит от формирования оптимальной структуры сети.

В работе [11] предложен механизм междоменного конструирования трафика на основании технологии MPLS, удовлетворяющую требованиям к параметрам *QoS*, и обеспечивать равномерную загрузку сети.

В [12] для решения задачи *TE* предлагается использоваться расширения протоколов маршрутизации работающих на основе алгоритма состояния каналов, к ним относятся протоколы *OSPF* и *IS-IS*, которые лавинообразно распространяют информацию для каждого маршрутизатора, содержащую полную информацию о топологии сети, метрике сети по всем каналам, с целью обеспечения вычисления кратчайшего пути до адресата.

В качестве одного из основных условий уменьшения сложности внутридоменной маршрутизации в работе [13] рассматривается ограничение на число пересылок между любыми двумя абонентскими системами домена. Предполагается, что внутри каждого домена абонентские системы могут связаться друг с другом самое большее через две пересылки. При этом алгоритм должен разбивать сеть на непересекающиеся домены. Однако существующие структуры формирования непересекающихся доменов не позволяют осуществлять динамическую реконфигурацию сети.

Постановка задачи

Одним из характерных признаком современных компьютерных систем является их динамическая реконфигурация. Это отражается на структуре доменов компьютерной системы [14] и, в большинстве случаев, сказывается на эффективности системы конструирования трафика.

В самом деле, в процессе динамической реконфигурации перестраивается структура доменов, что приводит к необходимости реконфигурации самой системы конструирования трафика, а существующие методы конструирования трафика не позволяют это сделать.

В связи с тем, что динамическую систему жестко разбить на статические подсети нельзя, необходимо отказаться от явного разбиения, и необходимо предложить механизм распределенного управления трафиком. В связи с этим, предложен способ организации распределенной системы

управления, при котором промежуточным узлам предоставляется возможность самостоятельно осуществлять реконфигурацию канала. С этой целью формируется распределенная система агентов управления трафиком.

Данная задача является NP-полной и решается с помощью линейного программирования.

Пусть сеть представлена в виде нагруженного графа с ограничением пропускной способности узла и всей системы передачи данных.

Граф сети обозначим, как $G = (V, E)$, где V обозначает множество узлов, и E множество связей. В качестве альтернативы, будем использовать представление (i, j) для связи узла i с узлом j. Пропускная способность связи (i, j) от узла i до узла j определена, как $c(i, j)$.

Пусть множество узлов V, разбивается на подмножества $V = \{V_1, V_2, V_3\}$. Ставим в соответствие каждому K_1 некоторое множество V_1 узлов, $V_1 = \{v_1, v_2, ..., v_i\}$, а K_2 ставится в соответствие множество V_2 узлов, $V_2 = \{v_i, ..., v_{i+k}\}$. Пусть есть какой-то узел v_i, который может входить в разные множества с различной вероятностью, $v_i \in V_1,\ v_i \in V_2$.

В работе [15] проведен анализ известных алгоритмов формирования доменов и предложен распределенный алгоритм *DDR* (*Distributed Dynamic Routing Algorithm*) для сетей с фиксированной структурой. В данной работе предлагается модифицированный алгоритм формирования доменов, основной целью которого является определение количества агентов. Необходимо учитывать динамический характер сети, и что от расположения агентов зависит объем передаваемого управляющего трафика.

Таким образом, определим какие из абонентских систем домена могут выступать в качестве агентов и смогут разделять между собой нагрузку.

В качестве дополнительного условия выбора абонентской системы v_i в качестве агента a_j определим следующее условие:

$$a_j = \{v_i \mid (1 - p'_j) = max\ \forall\ v_i \in V_j\}, \quad (1)$$

где p – вероятность того, что выбранная абонентская система может быть агентом. Учитывая устойчивость узла, вероятность выбора агента можно представить в виде:

$$K_1 = K_0 \cdot \delta + K' \cdot a_j, \quad (2)$$

где K_1 – коэффициент выбора агента; K_0 – весовой коэффициент степени; K' – весовой коэффициент a_j.

Выражения (1) и (2) обеспечивают формирование максимально устойчивого дерева от одного отправителя информации до одного или нескольких получателей информации, при этом могут быть использованы алгоритмы, приведенные в работах [15, 16].

Для разбиения графа на домены заполним матрицу инцидентности MI, и с ее помощью определим максимальную степень вершин графа v_m, и определим сможет ли одна из таких вершин стать агентом.

Для этой вершины определим плотность сетевого окружения δ_{v_m} первого и второго порядка, по формуле:

$$\delta_{v_i} = \frac{S + \sum_{S} E_{i,j}}{S}, \qquad (3)$$

при условии, что $E_{i,j} \neq 0$ в MI,

где S – максимальная степень связности вершины, $E_{i,j}$ – связь между смежными ребрами вершины с максимальной степенью.

Из всех полученных δ_{v_i} выбираются максимальные значения первого и второго порядка, и такая вершина становится агентом.

Приведем пример вычислений для регулярного графа представленного на рисунке 1.

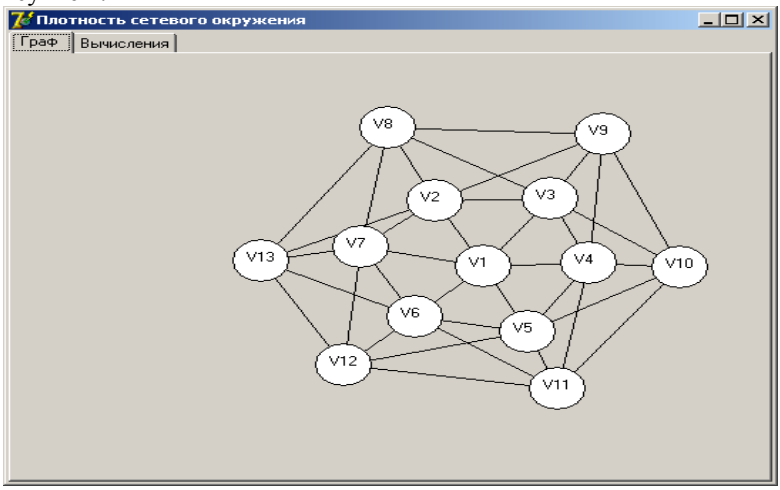


Рис. 1. Пример сети, представленного в виде регулярного графа

Максимальная степень вершины v_m = 6

Количество вершин, имеющих максимальную степень вершины = 7

Плотность сетевого окружения 1 порядка для вершины 1 $\delta_{v_m}(1) = 2$

Плотность сетевого окружения 2 порядка для вершины 1

$\delta_{v_m}(2) = 1{,}5$ $\delta_{v_m}(4) = 1{,}5$ $\delta_{v_m}(6) = 1{,}5$

$\delta_{v_m}(3) = 1{,}5$ $\delta_{v_m}(5) = 1{,}5$ $\delta_{v_m}(7) = 1{,}5.$

Это вершины с номерами: 1, 2, 3, 4, 5, 6, 7

Плотность сетевого окружения 1 порядка: $\delta_{v_m}(1) = 2$

$\delta_{v_m}(2) = 2{,}33$ $\delta_{v_m}(4) = 2{,}33$ $\delta_{v_m}(6) = 2{,}33$

$\delta_{v_m}(3) = 2{,}33$ $\delta_{v_m}(5) = 2{,}33$ $\delta_{v_m}(7) = 2{,}33$

Плотность сетевого окружения 2 порядка для вершины 1

$\delta_{v_m}(2) = 2{,}33$ $\delta_{v_m}(4) = 2{,}33$ $\delta_{v_m}(6) = 2{,}33$

$\delta_{v_m}(3) = 2{,}33$ $\delta_{v_m}(5) = 2{,}33$ $\delta_{v_m}(7) = 2{,}33$

Плотность сетевого окружения 2 порядка для вершины 2

$\delta_{v_m}(1) = 2$ $\delta_{v_m}(7) = 2{,}33$ $\delta_{v_m}(9) = 2{,}2$

$\delta_{v_m}(3) = 2{,}33$ $\delta_{v_m}(8) = 2{,}2$ $\delta_{v_m}(13) = 2{,}2$

и т.д.

На рис. 2. приведен пример расчета графа в моделирующей программе.

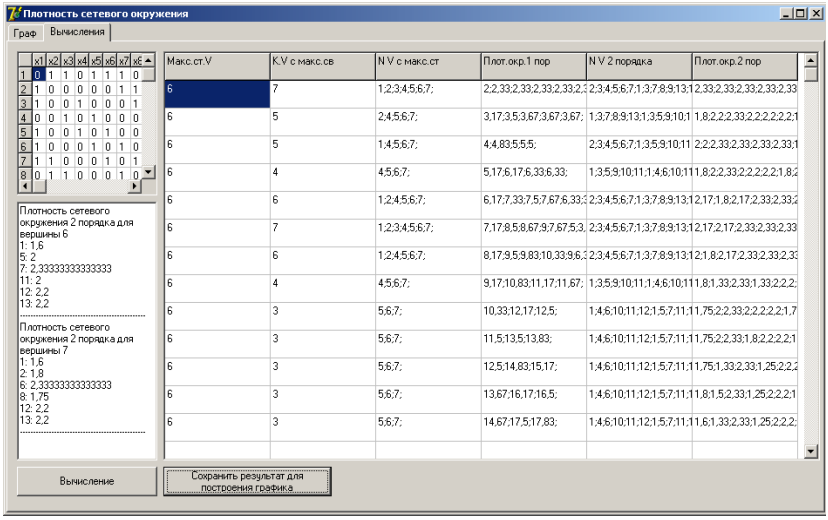


Рис. 2. Пример расчета графа в моделирующей програмне

Приведем пример еще один пример на рис. 3, однако в нем мы выберем нерегулярную структуру графа.

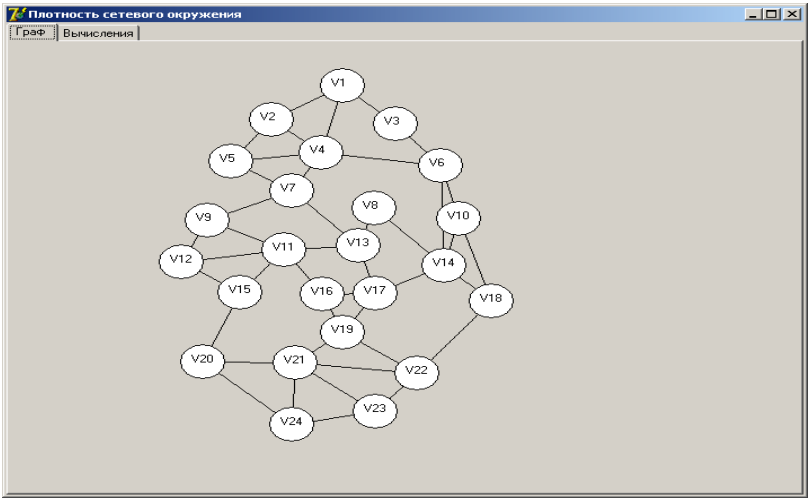


Рис. 3. Пример сети 2, представленной в виде графа

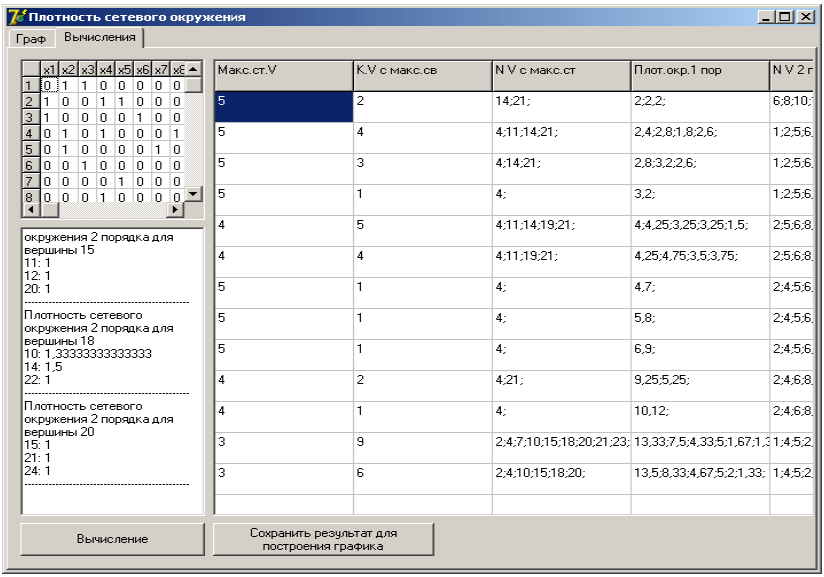


Рис. 4.Пример расчета нерегулярного графа в моделирующей програмне

В данном случае видно, что плотность сетевого окружения второго порядка выше плотности окружения первого порядка. В таком случае при

выходе из стоя некоторых ребер графов 1 и 2, эффективность работы топологии оценим на рис.5.

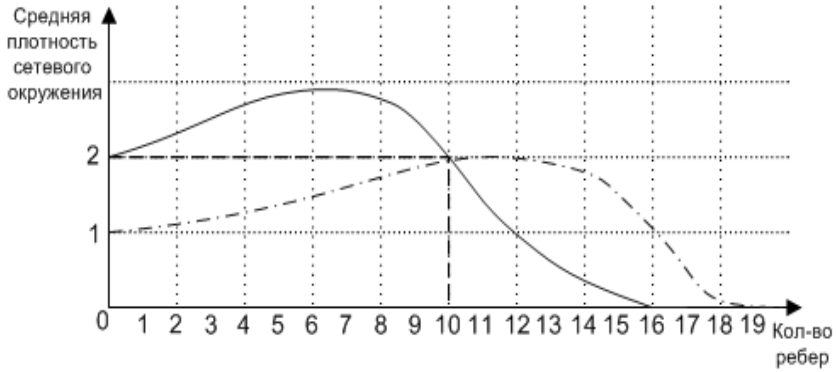


Рис. 5. Отношение средней плотности сетевого окружения к количеству вышедших из строя ребер (—— граф 1, —·- граф 2).

Проведем анализ зависимости изменения распределения сетевого окружения первого и второго порядков топологии сети в зависимости от места расположения агентов. Так как характер графиков для регулярной и нерегулярной топологии одинаковый, выберем для примера 1 граф. Предположим, что агентами назначены вершины v_1 и v_4, и проанализируем графике поведение топологии, рис. 6 и рис. 7.

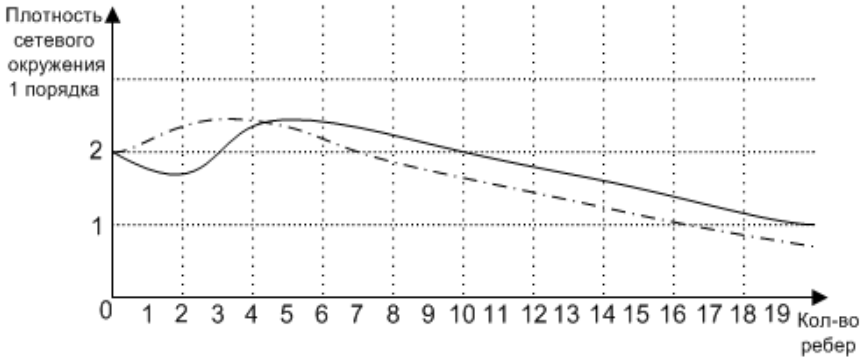


Рис. 6. График зависимости плотности сетевого окружения первого порядка от степени вершины (—— v_1 и —·- v_4).

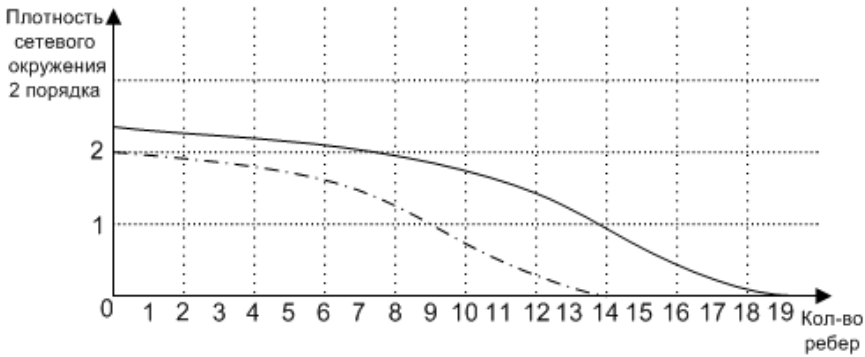


Рис. 7. График зависимости плотности сетевого окружения второго порядка от степени вершины (—— v_1 и —·- v_4).

После ряда экспериментов можно сделать вывод, что при выходе из строя ребер из первого порядка сетевого окружения, плотность сетевого окружения второго порядка увеличивается, а при выходе из строя ребер из второго порядка сетевого окружения, плотность сетевого окружения первого порядка повышается. При выходе из строя вершин из первого и второго сетевого окружения, плотность сетевого окружения первого порядка намного увеличивается, по отношению к плотности сетевого окружения второго порядка.

Кроме того, при неправильно выбранном местоположении агента, в данном случае v_4, показатели сетевого окружения ниже, в следствие чего можно сделать вывод, что агентом необходимо выбирать вершину не только у которой максимальная степень связности, а также вершину у которой плотность сетевого окружения первого и второго порядков наибольшая.

Выводы

В данной работе предложен способ повышения эффективности процесса маршрутизации на основе системы агентов. Для этого требуется меньше объем информации связанной с реконфигурацией сети. Преимуществом данного подхода является возможность двух смежных агентов содержать в своей области смежные узлы, или смежные узлы могут находиться на границе.

Проведен анализ зависимости изменения распределения сетевого окружения первого и второго порядков в зависимости от вида топологии сети. По результатам моделирования построены графики, доказывающие, что агентом необходимо выбирать вершину не только у которой максимальная степень связности, а также вершину, у которой плотность сетевого окружения первого и второго порядков наибольшая. Показано, что с

изменением топологии плотность распределения изменяется по нелинейному закону.

Список использованной литературы

1. F. Yu, V. Wong, V. Leung, A New QoS Provisioning Method for Adaptive Multimedia in Cellular Wireless networks, INFOCOM 2004.
2. Y. Xiao, H. Li, S. Choi, Protection ND Guarantee for Voice and Video Traffic in IEEE 802.11e Wireless LANs, INFOCOM 2004.
3. W. Sheikh, B. Shafig, R. Paul, A. Ghafoor,Provision of Multimedia Service in a Mobile Ad Hoc Network, Computer Society, IEEE 2004.
4. G. Yang, D. Shen, V. Li, UEP for Video Transmission in Space-Time Coded OFDM Systems, INFOCOM 2004.
5. Жуков И.А., Клименко И.А. Обеспечение заданного уровня качества обслуживания в объединенных сетях // Проблеми інформатизації та управління: Зб. наук. пр. – К.: НАУ, 2005. – Вип. 13. – С.5–14.
6. Таненбаум Э. «Компьютерные сети» 4-е издание, Питер 2003.-992с.
7. Adjih C., Clausen T., Jacquet P., Laouiti A., Minet P., Muhlethaler P., Qayyum A. and Viennot L. “Optimized link state routing protocol”, IDV 08, IETF, September 2003.
8. D. Awduche, A. Chiu, A. Elwalid, I. Widjaja, X. Xiao, “Overview and Principles of Internet Traffic Engineering”, May 2002, IETF RFC 3272
9. Y.H. Zhang, D. Makrakis and D. Hatzinakos, Supporting of QoS and Micro-Mobility in MPLS-based IPv6 Wireless Networks, Europa2004
10. Germán Goldszmidt, Yechiam Yemini, “Delegated Agents for Network Management”, IEEE Communications Magazine, March 1998
11. X.Xiao, A.Hannan, B.Bailey, “Traffic engineering with MPLS in the Internet”, IEEE Network Magazine, p.28-33, March 2000
12. Chakrabarti S. and Mishra A., “QoS issues in Ad Hoc Wireless”, IEEE Communications Magazine, Februar 2001.
13. B. Albert, FDDI and FDDI-II: Architecture, Protocols and Performance. Artech House, 1994
14. P. Vilà, J.L. Marzo, A. Bueno, “Automated Network Management Using a Hybrid Multi-Agent System”, In rtificial Intelligence and Applications (AIA 2002), September 9-12, 2002. Málaga, Spain. ISBN: 0-88986-352-0.
15. Mobile Ad Hoc Networking & Computing at Eurocom: “On Designing routing protocol for mobile Ad Hoc networks”.
16. C.-K. Toh, "Long-lived Ad-Hoc Routing based on the concept of Associativity" March 1999 IETF Draft, 8 pages.
17. T. Imielinski and J. C. Navas, "Geographic addressing, routing, and resource discovery with the global positioning system", Communications of the ACM Journal, 1997.

УДК 004.724.4(045)

КУЛАКОВ Ю.А.
МАКСИМЕНКО Е.В.,
РУЩАК О.А.

ПОВЫШЕНИЕ УРОВНЯ БЕЗОПАСНОСТИ ПЕРЕДАЧИ ИНФОРМАЦИИ В МОБИЛЬНЫХ СЕТЯХ

Рассмотрены вопросы повышения уровня безопасности в мобильных сетях. Предложен способ оптимального разбиения сообщения на отдельные блоки данных. Предложен алгоритм многопутевой маршрутизации для повышения безопасности передачи данных.

Questions of rise of level of safety in transportable networks are considered. The way of optimal splitting of the message on separate bulks is offered. The algorithm of multipath routeing for rise of safety of data transfer is offered.

Введение

В связи с расширением области применения мобильных компьютерных сетей, а также интеграции их в глобальные компьютерные сети, особую актуальность приобретает обеспечение безопасности передачи информации в сетях данного типа. Использование беспроводных каналов передачи, а также динамически изменяющаяся топология мобильных сетей не позволяют в полной мере использовать способы защиты информации, применяемые в сетях с проводными каналами связи и фиксированной структурой. Большинство существующих схем защиты информации для мобильных сетей ориентировано на обеспечение корректности маршрутной информации. В работах [1,2,3] предложены различные схемы защиты на уровне протоколов маршрутизации. Как правило, эти схемы предназначены для обеспечения корректности маршрутизации в мобильных сетях. Некоторые из них также включают решение проблемы некорректного поведения узла и способы обнаружения вторжения. В то же время практически ни одна из схем не рассматривает вопросы обеспечения безопасности самих передаваемых данных.

В данной работе предлагается передавать предварительно разбитое сообщение по нескольким путям.

На первом этапе производится разбиение сообщения на оптимальное количество частей. Для получения частей исходного сообщения используется пороговый алгоритм разделения секрета [4]. Пороговый алгоритм разделения секрета делит секретное сообщение на N частей, называемых *долями (share* или *shadow)* . При этом, имея в своем распоряжении любое число частей, меньшее T, нельзя получить никаких данных о секретном сообщении. В то же время при использовании

соответствующего алгоритма можно восстановить секретное сообщение из любого числа T (или больше) частей. Такой подход называют пороговой схемой разделения секрета (T, N) (threshold secret sharing). Таким образом, при использовании пороговой схемы разделения (T, N), секретное сообщение может быть разделено на N частей, при чем для того, чтобы перехватить сообщение, противник должен перехватить как минимум T частей. При перехвате числа частей меньше порогового, T, противник не может получить никаких данных о сообщении и фактически шансы его восстановить сообщение не больше, чем у того, кто вообще ничего не знает о сообщении. В рамках данной работы будет использована пороговая схема Шамира с использованием интерполяционных многочленов Лагранжа.

На втором этапе происходит выбор набора путей, соответствующих значений (T, N), и распределение частей на каждый из выбранных путей для достижения максимальной безопасности. Фундаментальная цель состоит в том, чтобы максимизировать безопасность путем распределения частей таким образом, чтобы противнику пришлось перехватить все пути, чтобы восстановить сообщение.

Формулировка проблемы

Предположим, что пороговый алгоритм разделения (T, N) применен к сообщению, которое нужно защитить, в исходном узле. Пусть на сетевом уровне есть всего M непересекающихся путей, путь 1, путь 2, …, путь M, доступных от источника до адресата. Для обозначения характеристик безопасности путей используется вектор $\underline{p} = [p_1, p_2, \ldots, p_M]$, где $pi(i= 1,2,\ldots,M)$ – это вероятность, что путь i скомпроментирован. Не отходя от обобщения, далее принимается $p_1 \le p_2 \le \ldots \le p_M$, что означает, что пути упорядочиваются от более безопасного до менее безопасного. При этом информация о безопасности пути p доступна в источнике из протоколов маршрутизации. Предполагается, что, если узел перехвачен, все части сообщения, проходящие через этот узел, перехвачены. Следовательно, путь скомпрометирован тогда, когда один или более любых узлов по пути скомпрометированы. Для каждого пути, предполагается, что, если он скомпрометирован, то все части сообщения, направленные по этому пути скомпрометированы. Поскольку используются непересекающиеся пути, то вероятность компроментации отдельного пути независима от вероятности компроментации других путей. Вероятности pi не включают вероятности того, что источник или адресат скомпрометированы, то есть, предполагается, что и источник, и адресат надежны.

Схема распределения используется, чтобы распределить N частей на M доступных путей. Обозначим распределение частей как $\underline{n} = [n_1, n_2, \ldots, n_M]$, где n_i – число частей, распределенных по пути i, n_i – целое число, $n_i \ge 0$,

$$\sum_{i=1}^{M} ni = N \qquad (1)$$

В соответствии с алгоритмом разделения, вероятность, что сообщение скомпрометировано, равняется вероятности, что *T* или больше частей скомпрометированы. Обозначим вероятность, что сообщение скомпрометировано как $P_{msg}(n)$. Тогда, распределение частей может быть сформулировано в виде проблемы оптимизации:

минимизировать $P_{msg}(n)$ при $\sum_{i=1}^{M} ni = N$,

n_i – целое число, $n_i \geq 0$.

Достижение максимальной безопасности без избыточности

Определим

$$r = 1 - \frac{T}{N} \qquad (2)$$

как коэффициент избыточности схемы разделения (*T, N*). Безизбыточной является схема с *r=0*, то есть *N = T*. При наличии *M* доступных путей и соответствующих характеристик безопасности, $\underline{p} = [p_1, p_2, \ldots, p_M]$, безизбыточная (*N, N*) (*N≥M*), схема разделения сообщения обеспечивает максимальную защиту, то есть, минимальную вероятность перехвата сообщения, когда не менее одной и не более *T-1* частей распределены по каждому из доступных путей, то есть

$$\begin{cases} 1 \leq n_i \leq T - 1, \ i = 1, \ldots m \\ \sum_{i=1}^{m} n_i = N \end{cases} \qquad (3)$$

Это распределение вынуждает противника перехватывать все пути, чтобы перехватить сообщение. Вероятность перехвата равняется вероятности, что все пути скомпрометированы, т.е.

$$P_{msg}(\underline{n}) = \prod_{i=1}^{M} p_i \qquad (4)$$

Заметим, что максимальная безопасность зависит только от выбранных путей. Так как *pi* – вероятность, удовлетворяющая условию $0 \leq pi \leq 1$, то чем больше путей используется для распределения ресурсов, тем меньше вероятность, и тем более безопасна доставка сообщения. Таким образом, если требуемый уровень защиты γ_{Pn} , схема должна выбрать для доставки сообщения первые m путей, путь *1*, путь *2*, …, путь *m*, которые удовлетворяют условию

$$P_{msg}(\underline{n}) = \prod_{i=1}^{m} p_i \le \gamma_{Pn} \qquad (5)$$

Для достижения максимальной безопасности с избыточностью, общее количество частей, распределенных по любым *m-1* или меньшему количеству путей, должно быть меньше чем *T*. Опять-таки, это распределение частей вынуждает противника перехватить все *m* путей, чтобы перехватить сообщение. Это также необходимое и достаточное условие для достижения максимальной безопасности.

Способ многопутевой маршрутизации

Рассмотрим алгоритм нахождения максимального количества путей, используемый для оптимизации набора путей для системы. Алгоритм использует частичное представление топологии сети, которое может быть предоставлено любым основным протоколом маршрутизации.

Предположим, что q_i – вероятность того, что узел перехвачен. Тогда вероятность того, что путь (s, t), состоящий из узлов $s, n_1, n_2, \ldots, n_l$, скомпроментирован, равна

$$p = 1-(1-q_1)\cdot(1-q_2)\cdot\cdots\cdot(1-q_l) \qquad (6)$$

Так как рассматривается безопасность сообщений во время передачи по сети, то предполагается, что источник и адресат надежны $qs=qd=0$. Вероятность qi показывает уровень защиты узла i и может быть оценена при помощи некоторого контролирующего программного обеспечения с обратной связью и/или аппаратных средств типа файерволов и устройств обнаружения вторжения. Вероятность также может быть назначена вручную администраторами на основании уровня физической защиты узлов, расположения или ранжирования узлов, и т. д. Например, штаб в тыле имеет самый высокий уровень защиты (наиболее низкое значение q_i), в то время как отдельные узлы, проникающие в зону неприятеля, будут иметь более низкий уровень защиты (высокое значение q_i).

В сети необходимо найти оптимальный набор путей, такой, чтобы вероятность P_{msg} была минимизирована. Вероятность перехвата сообщения

$$P_{msg}(\underline{n}) = \prod_{i=1}^{M} p_i$$

P_i – вероятность, которая всегда меньше чем 1. Чем больше частей p_i, тем меньше вероятность, и лучше защита. Таким образом, цель алгоритма поиска путей состоит в том, чтобы найти как можно больше путей, которые в то же время будут как можно более безопасными.

Алгоритм нахождения максимального количества путей, используемый для схемы распределения, является модификацией алгоритма алгоритма Дейкстры. Модифицированный алгоритм Дейкстры отличается от классического алгоритма тем, что позволяет помеченному узлу

вернуться к временной метке в случае нахождения к этому узлу пути с меньшей стоимостью. Для того, чтобы использовать модифицированный алгоритм нахождения кратчайшего пути во взвешенном графе, необходимо преобразовать характеристики безопасности отдельных узлов в совокупную функцию стоимости связи

Функция стоимости связи между узлами n_i и n_j определяется как:

$$c_{ij} = -\log\sqrt{(1-q_i)(1-q_j)} \qquad (7)$$

Тогда, стоимость пути (s, t) при применении алгоритма поиска кратчайшего пути определяется как

$$\cos t(s,t) = c_{s1} + c_{12} + \ldots + c_{l-1,l} + c_{lt}.$$

учитывая, что источник и приемник надежны, т.е. $q_s = q_t = 0$

тогда

$$\cos t(s,t) = -\log\sqrt{(1-q_1)^2(1-q_2)^2\ldots(1-q_l)^2} =$$
$$= -\log((1-q_1)(1-q_2)\cdots(1-q_l))$$

значение $\cos t(s,t)$ будет минимально при минимальном значении f

$$f = -\log((1-q_1)(1-q_2)\cdots(1-q_l)),$$

которое минимально при максимальном значении произведения

$$(1-q_1)(1-q_2)\cdots(1-q_l)$$

Алгоритм поиска максимального числа заключается в следующем.

При поиске набора путей сначала находится первый самый безопасный путь при помощи модифицированного алгоритма Дейкстры. После этого выполняется преобразование графа следующим образом:

1. Для каждого выбранного пути связи, используемые в пути, заменяются направленными дугами - дуге, которая направлена к источнику, присваивается ее исходная стоимость со знаком «-»; дуге, направленной к адресату, присваивается бесконечная стоимость (таким образом дуга фактически удаляется, поскольку критерием выбора является минимальная стоимость дуги).

2. Каждый узел на выбранных путях (кроме источника и адресата) разбивается на два разнесенных подузла, соединенных дугой нулевой стоимости, направленный к узлу-источнику.

3. Каждая внешняя связь, соединенная с узлом на выбранном пути, заменяется двумя составными дугами со стоимостью, равной стоимости связи - одна дуга заканчиваться на одном подузле, а другая выходит от другого подузла таким образом, что вместе с дугой нулевой стоимости, не получается замкнутый цикл.

4. С помощью модифицированного алгоритма Дейкстры, нахождение самого безопасного пути в преобразованном графе.

5. Если больше не может быть найдено ни одного пути, то заканчиваем моделирование с сохраненным набором путей.

6. Выполняется преобразование графа к первоначальному виду; при этом удаляются чередующиеся ребра (ребра, соединяющие два узла и направленные в противоположные стороны); остающиеся ребра группируются для формирования нового набора путей.

7. Вычисляется общая безопасность нового набора путей, она сравнивается с безопасностью набора путей, полученного на предыдущей итерации.

8. Если новый набор путей не обеспечивает лучшую безопасность, чем предыдущий, то поиск заканчивается с сохраненным набором путей, в противном случае сохраняется найденный набор путей и осуществляется переход к первоначальной трансформации графа.

На рис. 1(а) показана первая итерация работы алгоритма, показан первый найденный путь. На рис. 1(б) изображена трансформация графа – каждый узел, кроме источника и адресата, заменяется двумя подузлами, соединенными дугой нулевой стоимости, направленной к источнику; каждая внешняя связь, соединенная с узлом на выбранном пути, заменяется двумя составными дугами со стоимостью, равной стоимости связи - одна дуга заканчивается на одном подузле, а другая выходит из другого подузла. На рис. 1(в) показана вторая итерация работы алгоритма, найден еще один путь. Рис. 1(г) иллюстрирует трансформацию графа уже для этого набора путей, и применение модифицированного алгоритма Дейкстры. На рис. 1(д) показана перегруппировка ребер и формирование окончательного набора путей. Рис. 1(е) иллюстрирует результирующий набор путей.

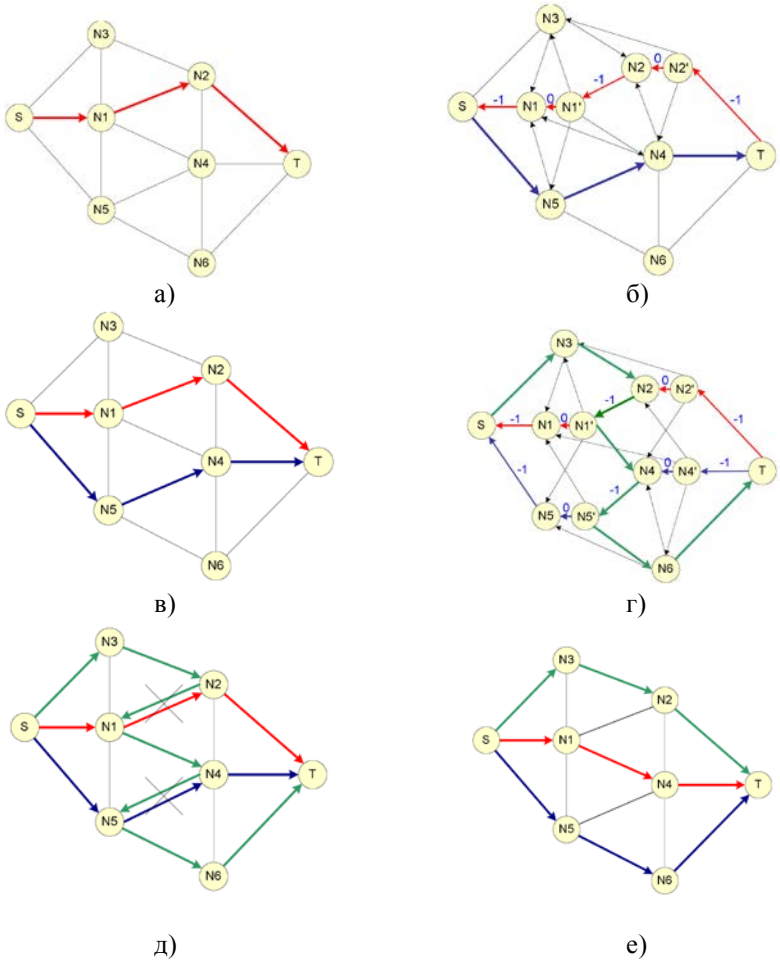


Рис.1

Использование многопутевой маршрутизации позволяет повысить защищенность информации при передаче ее по сети. Эффективность предложенного подхода зависит от топологии мобильной сети и степени ее узлов. При высокой степени узлов сети (порядка 15 и выше) передача информации по нескольким непересекающимся путям позволяет снизить вероятность перехвата передаваемой по сети информации, по сравнению с отправкой по одному пути, в среднем на 30%. При степени узлов от 6 до 12 выигрыш составляет порядка 20%. При степени узлов сети, меньшей 6, применение многопутевой маршрутизации дает те же самые результаты, что и применение однопутевой маршрутизации. Это объясняется тем, что

при этом слишком низка вероятность нахождения набора независимых путей.

Эффективность предложенного подхода также зависит от средней вероятности перехвата узлов. Как показали исследования, при средней вероятности перехвата узлов 0,3 на исследуемом диапазоне степеней (от 6 до 12) применение предложенного подхода дает выигрыш по безопасности порядка 12%.

Исследование вероятности того, что противник подслушает сообщение при передаче его по сети, также показало эффективность предложенного подхода. Применение многопутевой маршрутизации позволяет значительно снизить вероятность подслушивания сообщения (в среднем на 30%). Это снижение фактически зависит от количества используемых для доставки информации путей. Данная вероятность имеет некоторую нижнюю границу, что связано с тем, что противник, находясь в пределах зоны покрытия узла-источника или узла-приемника, может подслушать все части передаваемого сообщения.

Заключение

Предложен и разработан алгоритм нахождения приемлемого с точки зрения безопасности набора независимых путей для доставки сообщения.

Предложен и обоснован способ разделения передаваемого сообщения в исходном узле, и распределения полученных частей сообщения на выбранные пути позволяющий повысить безопасность передачи информации, и обеспечить в то же время определенный уровень надежности доставки информации путем введения некоторой избыточности.

Список использованных источников

1. Hu Y.-C., Johnson D. B., Perrig A. SEAD: secure efficient distance vector routing for mobile wireless ad hoc networks // Proceedings of the 4th IEEE Workshop on Mobile Computing Systems & Applications (WMCSA 2002) – 2002.- Calicoon(NY,USA)- P.3-13.
2. Papadimitratos P., Haas Z.J. Secure routing for mobile ad hoc networks // SCS Communication Networks and Distributed Systems Modeling and Simulation Conference (CNDS 2002) – San Antonio(TX, USA) - 2002
3. Venkatraman L., Agrawal D.P. Strategies for enhancing routing security in protocols for mobile ad hoc networks // Journal of Parallel and Distributed Computing – 2003-Vol.63, №2,P.214 – 227
4. Marti S., Giuli T., Lai K., Baker M. Mitigating routing misbehavior in mobile ad hoc networks // the 6th annual ACM/IEEE International Conference on Mobile Computing and Networking (Mobi-Com'OO) - 2000 - Boston(MA, USA) - P.255-265

УДК 004.724.4(045)

КУЛАКОВ Ю.А.,
ЛАНГЕ А.Й.

ФОРМИРОВАНИЕ ДИНАМИЧЕСКОЙ ИНФРАСТРУКТУРЫ КРУПНОМАСШТАБНЫХ МОБИЛЬНЫХ СЕТЕЙ

Предложен способ распределенного формирования доменов, оптимальных с точки зрения минимального времени маршрутизации.

Предложен и обоснован выборе контролера домена на основе критерия плотности сетевого окружения.

A method of forming a distributed domains, the best in terms of minimum time routing.

A justified and choosing a domain controller on the basis of density network environment.

Введение

В связи с улучшением параметров беспроводных каналов передачи данных расширяется область использования мобильных компьютерных сетей, увеличивается их размерность. Это в свою очередь сказывается на их инфраструктуре, которая перестает быть одноуровневой. В работе [1] рассмотрены вопросы многоуровневой структуры сети и вопросы оптимизации процедуры маршрутизации в масштабируемых мобильных сетях. Дальнейшее развитие эти вопросы получили в работе [2].

С целью повышения эффективности конструирования трафика компьютерная сеть разбивается на домены, при этом конструирование трафика осуществляется на внутридоменном и междоменном уровнях. Как правило, способы конструирования трафика внутри доменов отличаются от способов конструирования трафика между доменами. При этом эффективность каждого из этих методов зависит от размера и структуры доменов.

В настоящее время существует ряд подходов [3,4,5,6,7] к решению задачи формирования доменов и их контролеров. Все известные решения направлены на определение подмножества узлов в пределах компьютерной сетисети и связывание их с контролером домена. Каждый контролер домена ответственен за управление связью между узлами в доменах также как маршрутизация информации к другим контролерам в других доменах.

В качестве критерия выбора структуры и размера доменов служит эффективность передачи данных, которая равна: $E=1+W_d/W_c$, где: W_d – количество переданных полезных данных, $W_{сл}$ – количество служебной информации. В свою очередь, объем служебного трафика в домене V_i зависит от количества узлов N_{Vi} в нем и частоты (F_r) его реконфигурации.

С увеличением числа реконфигураций домена, объем служебного трафика в домене увеличивается по нелинейному закону [8], за счет чего резко падает эффективность передачи данных. При определенных условиях это может привести к неустойчивой работе сети. Объем служебной информации зависит также от диаметра домена и степени связности его узлов.

Таким образом, чем больше степень вершины, выбранной в качестве контролера домена, тем меньше объем управляющего трафика и время его распространения по сети. В связи с этим в большинстве случаев степень вершины выбирается в качестве критерия при определении месторасположения контролера домена.

В работе [9]показано, что известные методы формирования доменов и выбора месторасположения их контролеров не всегда эффективны для мобильных сетей, структура которых постоянно изменяется. В первую очередь это касается процедуры выбора узла компьютерной сети в качестве контролера домена. В большинстве известных алгоритмов месторасположение контролера домена определяется на основании степени вершины, то есть выбирается вершина с максимальной степенью, однако, при регулярной топологии сети степени всех вершин равны.

Определение оптимального месторасположение контролера домена маршрутизации

Основная цель заключается в проектировании эвристики, которая будет выбирать некоторые узлы как доменные заголовки и подсчитывать домена в крупной мобильной сети. определение домена не должно быть определено априори каким-то постоянным критерием, оно должно отображать плотность сети.

В связи с этим для выбора узла компьютерной сети в качестве контролера домена предложен и обоснован критерий $d_k(v_i)$ плотности k-го порядка сетевого окружения вершины v_i:

$$d_k\,(v_i) = 1 + \frac{\sum_{v_m \in \Gamma_k, v_j \in \Gamma_k} l_{m,j}}{\delta\ (v_i)},$$

где: $\delta(v_i)$ – степень вершины v_i ;

$l_{m,j}$ – эвклидово расстояние между вершинами v_m и v_j.

$\Gamma_k(v_i)$ – многозначное соответствие k-го порядка для вершины v_i .

Критерий $d_k(v_i)$ характеризует относительную связность вершины v_i в своем k-ом окружением и представляет собой отношение общего числа ребер в подграфе, образованном k-ым окружением вершины v_i , к ее степени.

В качестве контролера домена выбирается узел сети, соответствующий вершине с максимальным значением $d_k(v_i)$. $\Gamma_k(v_i)$ характеризует отношение между количеством связей и количеством узлов в k-сетевым

окружением. Это способствует повышению устойчивости функционирования сети и снижению объема управляющего трафика.

Рассмотрим диапазон изменения значений критерия $d_k(v_i)$ в зависимости от структуры сети и степени связности ее узлов. Для линейной или древовидной структуры графа компьютерной сети, величина параметра $d_k(v_i)$ будет минимальной и равной для всех вершин $d_k(v_i)=1$.

Для компьютерной сети с полносвязной топологией и единичном сетевом окружении $\Gamma(v_i)$ все вершины имеют равные степени $\delta(v_i)= n-1$, где n – число вершин графа. В этом случае $d_k(v_i) =n/2$.

Для компьютерной сети с полносвязной плоской топологией плотность *1*-го порядка сетевого окружения для центральной равна двум.

В свою очередь, плотность *1*-го порядка сетевого окружения для боковой вершины равна 5/3.

В общем случае для нерегулярного плоского графа значения величины $d_1(v_i)$ будет различно для разных вершин. Например, для графа (рис. 1) значения степени вершин и величин $d_1(v_i)$ приведены в табл.1.

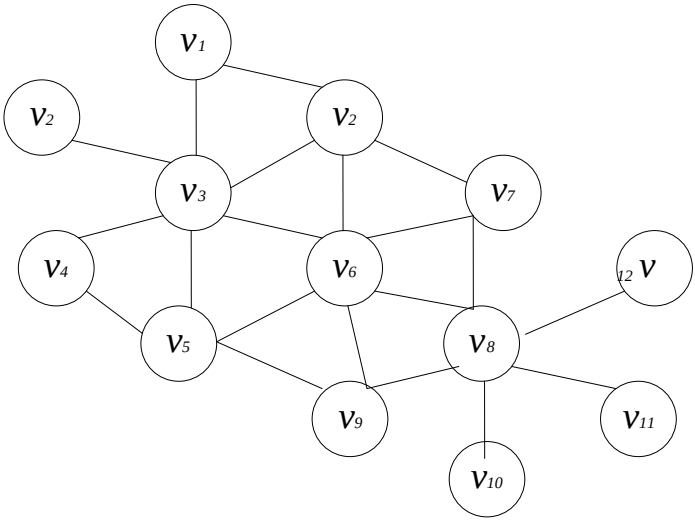


Рис.1

Табл. 1

№ вершины	1	2	**3**	4	5	**6**	7	**8**	9	10	11	12
степень	2	1	**6**	2	4	**6**	3	**6**	3	1	1	1
$d_1(v_i)$	1.5	1	**1.7**	1.5	1.8	**2**	1.7	**1.3**	1.7	1	1	1

В данном случае три вершины: 3,6 и 8 имеют максимальную степень, равную шести, однако предпочтительней является вершина 6, так как у нее максимальное значение $d_1(v_i)$=2 и она находиться в центре графа. Это обеспечивает минимальный управляющий трафик и минимальное время обновления маршрутной информации. При лавинообразном алгоритме распространения маршрутной информации, что характерно для алгоритмов внутридоменной маршрутизации, выбор контроллера домена на основе критерия плотности сетевого окружения позволяет сократить объем управляющей информации при реконфигурации сети примерно в $d_k(v_i)$ раз.

Это достигается за счет то, что в данном случае уменьшается дублирование управляющего трафика.

Рассмотрим загрузку каналов передачи информации при лавинообразном распространении одного кадра управляющей информации.

В начальный момент времени t_1 количество K_1 кадров управляющей информации равно степени $\delta(v_i)$ вершины v_i, выбранной в качестве контролера домена, т.е. $K_1 = \delta(v_i)$. В следующий момент времени t_2 количество кадров управляющей информации равно:

$$K_2 = \sum_{\delta_i \in \Gamma_1(v_i)} (\delta(v_i - 1).$$

В общем случае:

$$K_{i+1} = \sum_{\delta_i \in \Gamma_1(v_i)} (\delta(v_i - 1)$$

.

Уменьшению объема управляющей информации также способствует сокращение времени распространения управляющего трафика. Так, например, в данном случае (см. рис. 1) при выборе вершины v_6 в качестве контролера домена время распространения управляющей информации до наиболее удаленного узла равно двум тактам, а при выборе вершины v_3 или v_8 в качестве контролера домена время распространения управляющей информации до наиболее удаленного узла равно трем тактам.

В общем случае порядок (k) плотности сетевого окружения выбирается в зависимости от размера сети, на начальном этапе формирования инфраструктуры сети значение k принимается равным единицы. При нескольких вершинах с одинаковым значением $d_1(v_i)$ значение k задается равным 2 и для вершин с максимальным значением $d_1(v_i)$ вычисляется значение $d_2(v_i)$. Среди этих вершин выбирается вершина с максимальным значением $d_2(v_i)$. При наличие нескольких вершин с максимальным значением $d_2(v_i)$ задается k=3 и процедура повторяется до выявления одной вершины с максимальным значением $d_k(v_i)$.

На рис.2 представлен граф, у которого $d_1(v_3)$= $d_1(v_6)$=2, а величина $d_2(v_3)$=19/6=3.2, соответственно $d_2(v_6)$=28/6=4.67. Следовательно, в качестве контролера домена выбирается вершина v_6.

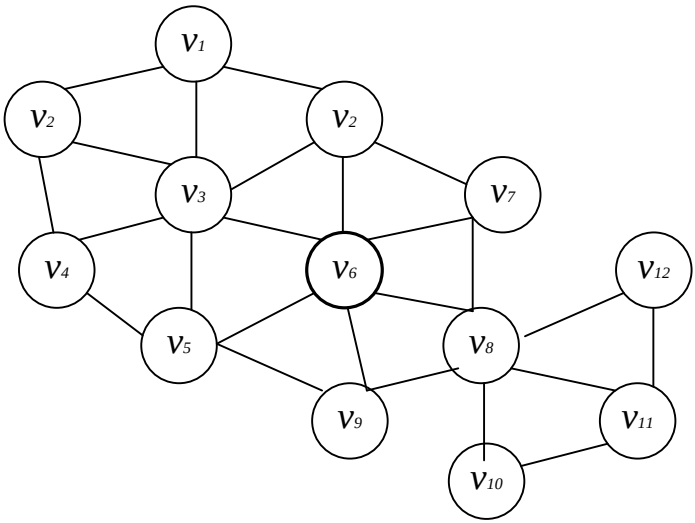


Рис.2.

В связи с этим предлагается следующий способ формирования инфраструктуры компьютерной сети.

Алгоритм формирования структуры сети

На начальном этапе формирования структуры компьютерной сети определяется количество и месторасположение агентов управления трафиком на основе критерий $d_k(v_i)$ плотности k-го порядка. С этой целью выполняется следующая последовательность операций:

1. Каждый узел $v_i \in V$ компьютерной сети формирует множество $\Gamma(v_i)$ смежных с ним узлов.
2. На основании множества $\Gamma(v_i)$ и степени $\delta(v_i)$ каждый узел v_i формирует свой коэффициент плотности $d_k(v_i)$.
3. Узлы $v_i \in V$ обмениваются между собой значениями $d_k(v_i)$.
4. Каждый узел $v_i \in V$ сравнивает собственное значение $d_k(v_i)$ со всеми значениями $d_k(v_j)$ узлов $v_j \in \Gamma_k(v_j)$.
5. Если $d_k(v_i) > d_k(v_j)$ узлов $v_j \in \Gamma_k(v_j)$, то узел v_i объявляет себя контролером домена. Являясь контролер домена, узел v_i начинает формировать домен $D(v_i)$, включая в свой состав все узлы $v_j \in \Gamma_k(v_j)$.
6. Затем аналогичным образом из узлов $v_m \notin D(v_i)$ формируется следующий домен и так далее.

7. В результате формируется множество $\{D_j(v_i)\}$ доменов конструирования трафика.

Процедура реконфигурации доменов

В результате перемещения узлов происходит изменение топологии компьютерной сети, которая в некоторых случаях может вызвать необходимость реконфигурации доменов. Данная процедура заключается в следующем:

1. При перемещении узла v_m проверяется принадлежность его к одному из доменов $\{D_j\ (v_i)\}$.
2. При отсутствии признака принадлежности к одному из доменов узел v_m начинает процедуру поиска ближайшего домена, которая заключается в обмене информацией с контролерами смежных доменов.
3. При выполнении условия вхождения в домен, АС подключается к домену, при этом контролер домена уведомляет всех членов домена о новом члене домена.
4. Если АС не может быть включена хотя бы в один из доменов, она начинает процедуру формирования собственного домена, объявляя себя головой домена.
5. За счет перераспределения АС, входящих в состав смежных с вновь образованным доменом, осуществляется динамическая реконфигурация сети и обновление таблиц маршрутизации.

Таким образом, в результате процесса самоорганизации мобильная компьютерная разбивается на оптимальное число доменов, количество и состав которых меняется в зависимости от перемещения мобильных узлов. При этом каждый из контролеров домена обеспечивает конструирование трафиком в определенной группе узлов, объединенных в соответствующий домен конструирования трафиком.

Выводы

1. Использование критерия плотности сетевого окружения при выборе контролера домена позволяет сформировать максимально устойчивую инфраструктуру мобильной сети и снизить объем управляющего трафика.
2. Предложенный способ формирования доменов маршрутизации по сравнению с известными способами позволяет реализовать сформировать и поддерживать оптимальную инфраструктуру крупномасштабных мобильных сетей с точки зрения минимизации объема управляющего трафика.

Список использованной литературы

1. N. Mitton, A. Busson, and E. Fleury. Self-organization in large scale ad hoc networks**.** In *The Third Annual Mediterranean Ad Hoc Networking Workshop, MED-HOC-NET 04*, Bodrum, Turkey, June 2004

2. B. Goncalves, N. Mitton, and I. Guérin-Lassous. Comparison of two Self-Organization and Hierarchical Routing Protocols for Ad Hoc Networks**.** In *Second International Conference on Mobile Ad Hoc and Sensor Networks (MSN)*, Hong-Kong, China, December 2006
3. P. Basu, N. Khan, and T. Little. A mobility based metric for clustering in mobile ad hoc networks. In Proceedings of Distributed Computing Systems Workshop 2001, 2001.
4. G. Chen and I. Stojmenovic. Clustering and routing in mobile wireless networks. Technical Report TR-99-05, SITE, June 1999.
5. A. Amis, R. Prakash, T. Vuong, and D. Huynh. Max-min dcluster formation in wireless ad hoc networks. In Proceedings of the IEEE INFOCOM, march 2000.
6. B. Das and V. Bharghavan. Routing in ad-hoc networks using minimum connected dominating sets. In ICC, 1997.
7. Y. Fernandess and D. Malkhi. K-clustering in wireless ad hoc networks. In Proceedings of the second ACM international workshop on Principles of mobile computing, 2002.
8. Кулаков Ю.А., Мухамед Ель Амин Бабикер Адаптивный алгоритм формирования доменов в динамических системах // Вісник національного технічного університету України “КПИ”, Інформатика, управління та обчислювальна техніка, 2006. - № 45 С.57-66
9. Кулаков Ю.А. , Мухамед Ель Амин Бабикер, Рамессар Сурийур Кумар. Адаптивный распределенный алгоритм формирования **к**ластеров в мобильных сетях// Труды шестой международной научно-практической конференции "Современные информационные и электронные технологии" (23–27 мая 2005 г. г. Одесса. Украина). – 2005.

УДК 004.724.4(045)

БАХТАРИ ХЕДАЯТОЛЛАХ

МНОГОАБОНЕНТСКАЯ ДОСТАВКА ИНФОРМАЦИИ В СЕТЯХ AD HOC

Рассмотрен вопрос организации многоабонентской доставки информации в мобильных сетях. Для маршрутизации предлагается использовать устойчивый подграф многоадресной доставки сообщений. Предложен алгоритм формирования максимально устойчивого пути.

The question of increase of efficiency of transfer of the information is considered due to formation of the steady virtual channel. For routing it is offered to use a steady subgraph of multicast messages. The algorithm of formation of maximum steady way is offered.

Введение

Под многоабонентской доставкой сообщений понимается передача сообщения от абонентской системы (АС) – источника информации к определенной группе АС – получателей информации, называемой группой многоабонентской доставки сообщений. Процесс формирования группы многоабонентской доставки заключается в построении частичного покрывающего дерева.

Существует ряд алгоритмов [1] формирования дерева многоабонентской доставки сообщений, основанных на дереве Штейнера. В качестве критерия оптимизации используется общая стоимость дерева многоабонентской доставки сообщений. Если группа многоабонентской доставки сообщений включает все вершины сети, решение задачи построения дерева Штейнера сводится к поиску дерева кратчайшего пути. Однако известные точные методы решения этой задачи они не учитывают ограничения, накладываемые на деревья, сетевыми задачами. В этом случае используются эвристические алгоритмы формирования дерева Штейнера. Так, в работе [2] предложен эвристический алгоритм Прима по поиску минимального остовного дерева для полного *пространственного графа*, где полный *пространственный граф* – это граф, который содержит все вершины в сети и имеет связь между каждой парой вершин, представляющих кратчайший путь между ними.

В работе [3] предложен распределенный алгоритм для решения задачи построения дерева Штейнера. Для решения проблемы оптимизации дерева по критерию задержки в работе [4] представлен эвристический алгоритм, названный *самый короткий мультикаст алгоритм с ограничениями.* Работа алгоритма начинается с того, что вычисляется дерево с наименьшей задержкой, внедренное в данный источник и охватывающее

всех членов группы. После этого итерационно заменяются главные рёбра в дереве более дешевыми главными рёбрами не в дереве, при не нарушении ограничения по задержке; это выполняется до тех пор, пока общая стоимость дерева не может быть далее сокращена. алгоритм по поиску кратчайшего пути в дереве. Алгоритм сходится, так как выполнение начинается с дерева с наименьшей задержкой.

В работе [5] предложен централизованный эвристический алгоритм, названный *KPP эвристика*. Зная источник v_s и группу многоабонентской доставки сообщений B, с помощью метода динамического программирования, формируется подграф G с учетом ограничения по задержке. Затем с помощью алгоритма Прима формируется остовое дерево подграфа.

В работе [6] рассмотрена задача построения дерева Штейнера с учетом величины задержки передачи информации. Вначале конструируется опорное *дерево*, T_r, путей минимальной стоимости от исходной вершины v_s ко всем вершинам получателям. После этого, для каждой вершины-получателя v_i выполняется попытка создать дерево T_i, которое изначально содержит путь в T_r от вершины v_s к вершине v_i. Затем алгоритм увеличивает T_i, прибавляя "хорошие" пути от вершин, входящих в дерево, до вершин-получателей, которые не входят в дерево, пока все вершины-получатели не будут включены в дерево. Если в результате будет создано более одного подходящего дерева, то будет выбрано дерево с минимальной стоимостью.

Особую актуальность многоабонентская передача приобретает в мобильных сетях [7], так как перемещение АС между зонами действия различных базовых станций (БС) может привести к разрыву виртуального соединения и, как следствие, к потере информации. Многоабонентская передача еще более усложняет ситуацию, так как миграция АС определяет необходимость реконфигурации multicast группы.

С точки зрения временной сложности и минимизации времени ремаршрутизации в мобильных компьютерных сетях наиболее эффективными алгоритмами формирования дерева многоадресной доставки являются алгоритмы, формирующие общий путь до группы многоабонентской доставки. Основным недостатком данных алгоритмов является увеличение пути до наиболее удаленной вершины группы многоабонен многоабонентской доставки, которое может превысить предельно допустимое значение. В этом случае целесообразно использовать комбинированный алгоритм формирования дерева многоабонентской доставки, используя минимальный путь до наиболее удаленной вершины группы многоабонентской доставки.

При передаче пакетов в мобильной сети в режиме виртуальных соединений дополнительным фактором, влияющим на эффективность передачи, является устойчивость сформированного канала передачи информации. Желательно чтобы виртуальный канал обеспечивал безразрывную

передачу информации во время сеанса обмена. В связи с этим в качестве основного критерия формирования виртуального соединения рассматривается его устойчивость. Таким образом, в мобильных сетях актуальной является задача формирования устойчивого виртуального канала передачи информации.

В общем случае, компьютерную сеть можно представить в виде нагруженного графа $G(V, E)$, где $V= \{ v_i | i=1,2…n\}$ – множество вершин графа, $E=\{e_{i,j} | i,j=1,2…n\}$ – множество ребер графа. Каждое ребро $e_{i,j}$ графа характеризуется весом $w_{i,j}$. В зависимости от решаемой задачи величина $w_{i,j}$ может характеризовать длину $l_{i,j}$ ребра, задержку $t_{i,j}$ или стоимость $c_{i,j}$ передачи информации по каналу, соответствующему ребру $e_{i,j}$.

В случае многоабонентской доставки сообщений с источником в вершине v_s и множеством вершин $V_m = \{ v_i | i=1,2…n\}$, определяющих группу АС, получателей информации, задача маршрутизации состоит в нахождении дерева $T_s = (V_s, E_s)$ доставки сообщений для которого выполняется условие:

$$W_s = \min \sum_{e_{i,j} \in E_s} w_{i,j}, \qquad (1)$$

где: W_s – вес всех ребер дерева доставки сообщений.

В качестве ограничения на включение ребра $e_{i,j}$ в состав дерева T_s рассматриваются условия:

- $B_{i,j} > b_s$, где: $B_{i,j}$ – значение пропускной способности канала; b_s – требуемое значение пропускной способности для многоабонентской доставки сообщений по дереву графа T_s.
- $t_d > t_{max}$, где: t_d – значение допустимой задержки передачи пакетов; t_{max} – максимальная задержка передачи пакетов между узлами дерева многоабонентской передачи данных.

Таким образом, при значительном удалении источника от группы получателей информации целесообразно задачу формирования дерева многоабонентской доставки информации разбить на задачу формирования кратчайшего пути L_0 от вершины v_s до вершины v_m ближайшей к множеству вершин V_m и задачу формирования структуры группы получателей информации с корневой вершиной v_m. В этом случае вес ребер W_d дерева многоабонентской доставки информации включает в себя вес ребер W_0 пути до группы многоабонентской доставки информации и вес ребер W_m внутри группы:

$$W_d = \sum_{e_{i,j} \in E_0} w_{i,j} + \sum_{e_{k,l} \in E_m} w_{k,l} \qquad (2)$$

При нахождении вершины источника внутри группы доставки множество $V_0 = \varnothing$ и $L_0 = 0$, а величина W_d равна:

$$W_d = \sum_{e_{k,l} \in E_m} w_{k,l}. \qquad (3)$$

При значительном удалении источника от группы многоадресной доставки информации выполняется условие:

$$\sum_{e_{i,j} \in E_0} w_{i,j} >> \sum_{e_{k,l} \in E_m} w_{k,l} \qquad (4)$$

Отсюда следует, что основное внимание в данном случае необходимо уделять вопросам оптимального построения минимального пути и определения вершины v_m.

Для мобильных компьютерных систем и сетей с постоянно меняющейся топологией актуальным является формирование максимально устойчивого пути

Устойчивость пути будем оценивать с помощью коэффициента устойчивости: $k= (T_0 - T_S)/T_0$,где: T_s – время сеанса обмена (требуемое время виртуального соединения); T_0 – время жизни маршрута, зависящее от устойчивости выбранного пути.

Как правило, удаление некоторого ребра в графе связности мобильной сети связано с перемещением смежной ему АС. Каждой вершине v_i графа $G = (V, E)$ сопоставим величину $p(v_i)$, представляющую собой вероятность удаления или перемещения вершины v_i. Соответственно величина $q(v_i) = 1- p(v_i)$ характеризует устойчивость вершины v_i.

Далее каждому ребру $e_{i,j}$ сопоставим величину $p(e_{i,j})$, характеризующую вероятность удаления ребра $e_{i,j}$ из графа. Очевидно, что вероятность удаления ребра определяется вероятностями удаления смежных вершин, то есть:

$$p(e_{i,j})= 1-(1- p(v_i)(1- p(v_j)). \qquad (5)$$

Соответственно, выражение:

$$q(e_{i,j}) = 1- p(e_{i,j}) \qquad (6)$$

характеризует устойчивость ребра $e_{i,j}$ между смежными вершинами v_i и v_j.

Подставив выражение (5) в выражение (6) получим:

$$q(e_{i,j}) =(1- p(v_i)(1- p(v_j))$$

или

$$q(e_{i,j}) =q(v_i)\, q(v_j)) \qquad (7)$$

В свою очередь устойчивость пути $S_{n,m}$ является функцией от устойчивости всех ребер данного пути, то есть: $S_{n,m} = f(S_{i,j} \mid \forall\ e_{i,j} \in P_{n,m})$. В общем случае устойчивость пути определяется следующим выражением:

$$S_{n,m} = \prod_{\forall e_{i,j} \in P_{n,m}} (1 - p(e_{i,j})). \qquad (8)$$

Подставив в выражение (8) выражение (6) получим:

$$S_{n,m} = \prod_{\forall e_{i,j} \in P_{n,m}} q(e_{i,j})). \qquad (9)$$

С учетом выражения (7) имеем:

$$S_{n,m} = q(v_n)q(v_m) \prod_{\substack{\forall e_k \in P_{n,m} \\ k \neq n, k \neq m}} (q(v_k))^2. \qquad (10)$$

Рассмотрим вопрос формирования максимально устойчивого пути от источника (вершины v_1) до некоторой вершины $v_j \in \Gamma(V_m)$, где: V_m – множество вершин группы многоабонентской доставки.

В результате работы алгоритма формируется некоторое множество вершин V_S к которому, на каждом шаге вычисления, добавляется вершина из множества $\Gamma(V_S)=\{v_j \in V \backslash V_S\}$ вершин, смежных с вершинами множества V_S.

Алгоритм заключается в следующем:

1. Формируется подмножество $V_S=\{v_1\}$.
2. Для всех вершин $v_j \in \Gamma(v_1)$, вычисляем $S_{1,j} = q(e_{1,j})$ и формируем множество $B=\{b_j\}$ упорядоченных пар $b_j=(v_1, S_{1,j})$.
3. Среди вершин множества смежности $\Gamma(v_1)$, находим вершину $v_j \in V \backslash V_S$, с максимальным значением $S_{1,j}$ и включаем ее во множество V_S
4. Для всех вершин $v_j \in \Gamma(V_S)$, вычисляем $S_{1,j} = S_{1,i} \times q(e_{i,j})$ и формируем множество $B=\{b_j\}$ упорядоченных пар $b_j=(v_i, S_{1,j})$. При этом из двух пар b_j выбирается пара с максимальным значением $S_{1,j}$.
5. Среди вершин множества смежности $\Gamma(V_S)$, находим вершину $v_j \in V \backslash V_S$, с максимальным значением $S_{1,j}$ и включаем ее во множество V_S.
6. Если $V \backslash V_S \neq \varnothing$, то переход к пункту 4, иначе алгоритм завершает свою работу.

В результате формируется путь между вершинами v_1 и v_m с максимальным значением $S_{1,m}$.

В мобильных сетях с относительно низкой устойчивостью виртуальных соединений предлагается формировать устойчивый подграф многоадресной доставки сообщений, в основу формирования которого положен приведенный выше алгоритм формирования максимально устойчивого пути. Основное отличие заключается в том, что при формировании множества $B=\{b_j\}$ упорядоченных пар $b_j=(v_i, S_{1,j})$ для каждой вершины v_j выбирается несколько пар b_j с наибольшими значениями $S_{1,j}$, сумма которых стремиться к единице. Таким образом, вместо одного пути формируется граф доставки сообщений.

Заключение

1. Предложенный в работе подход к организации многоабонентской доставки на основе формированию максимально устойчивых виртуальных соединений позволяет повысить эффективность передачи информации за счет снижения вероятности ремаршрутизации информации в процессе ее передачи.

2. С целью повышения эффективности процедуры маршрутизации в мобильной сети в настоящей работе предлагается строить дерево доставки не на всем исходном графе, а на предварительно сформированном максимально устойчивом подграфе $G^s = (V^s, E^s)$.

Список использованной литературы

1. H. F. Salama, “Multicast Routing for Real-Time Communication on High Speed Networks,” Ph.D. dissertation, NC State Univ., Raleigh, 1996.
2. L. Kou, G. Markowsky, and L. Berman, “A Fast Algorithm for Steiner Trees,” *Acta Informatica*, 1981, pp. 141–45.
3. F. Bauer and A. Varma, “Distributed Algorithms for Multicast Path Setup in Data Networks,” Comp. Eng. Dept., UC Santa Cruz, UCSC-CRL-95-10, Aug. 1995.
4. Mario Gerla , Ching-Chuan Chiang , Lixia Zhang, “Tree multicast strategies in mobile, multishop wireless networks”, Mobile Networks and Applications, v.4 n.3, p.193-207, Oct 1999.
5. Guangyu Pei and Mario Gerla and Tsu-Wei Chen, “Global State Routing: A New Routing Scheme for Ad-hoc Wireless Networks”,{ICDCS} Workshop on Wireless Networks and Mobile Computing", pages D71-D78,2000.
6. B. Wang and J.C. Hou, “Multicast routing and its QoS extension: problems, algorithms, and protocols”, IEEE Network, vol. 14, Jan.2000.
7. Кулаков Ю.А., Способ повышения эффективности доставки сообщений в мобильной компьютерной сети// Вісник національного технічного університету України “КПИ”, Інформатика, управління та обчислювальна техніка, 2002. - № 38. -С. 99-105.

УДК 004.724.4(045)

САДЕГ РАСТГУ МОХАММАД

СПОСОБ УПРАВЛЕНИЯ ТРАФИКОМ В МОБИЛЬНЫХ СЕТЯХ АТМ

Предложен эффективный способ управления трафиком в беспроводных сетях АТМ. Рассмотрен способ прогнозирования трафика с учетом его самоподобия. Приведен способ расчета времени задержки передачи ячеек в сети АТМ.

An effective way to manage traffic in wireless networks ATM. We consider the method of forecasting traffic in view of its samopodobiya. A method of calculating the time delay in the transmission network ATM cells.

Актуальность работы обусловлена тем, что современные сетевые технологии, такие как асинхронный режим передачи Asynchronous Transfer Mode (ATM), в сочетании с беспроводными каналами передачи данных позволяют создавать достаточно высокоскоростные мобильные сети, ориентированные на мультимедийные приложения.

В связи с миграцией абонентских систем возникает необходимость постоянной корректировки маршрутов, что в свою очередь оказывает влияние на сетевой трафик. Таким образом, повышение эффективности управления трафиком в беспроводной сети АТМ является актуальной задачей.

Механизмы управления трафиком должны включать в себя средства обратной связи для всех классов QoS, т.к. при необходимости нужно передавать источникам требования снизить скорость передачи. Одним из способов управления трафиком является управление на основе контроля параметров задержки *CTD* со стороны источника. Это позволяет предсказывать перегрузки в сети и предпринимать соответствующие меры, вплоть до заблаговременной ремаршрутизации или обхода перегруженного участка. Это особенно актуально в случаях, когда перегрузки вызваны отказами или перемещением промежуточных узлов. Обход загруженного участка может также осуществляться для увеличения скорости передачи с целью минимизировать общую задержку передачи трафика для данного соединения и более равномерного распределения загрузки по сети.

Конкретная задержка складывается из задержек при передаче по линии связи между устройствами и задержек на каждом из промежуточных коммутаторов в АТМ:

$$CTD = T_o + T_n,$$

где: T_o – время обслуживания в коммутаторах;

T_n – суммарное время передачи.

Величина *CTD* определяется разницей во времени между максимальной и минимальной задержкой при доставке ячеек от отправителя к получателю. Вариация задержки зависит от процедуры мультиплексирования в один физический канал связи потоков ячеек, принадлежащих множеству виртуальных соединений, и непостоянства задержки, вносимой очередями коммутаторов АТМ.

Для стандартной схемы параметр задержки может быть представлен в следующем виде:

$CTD = (T_{о.мин} + T_п) + T_{о.доп}$.

В то же время, для модифицированной схемы возможна минимизация задержек за счет минимизации времени передачи:

$$CTD = (To.мин + \frac{1}{k} \cdot Tп) + To.доп$$

Уменьшение времени задержки *CTD* осуществляется различным образом для различных классов обслуживания. Так, для классов гарантированного сервиса (CBR, VBR) такое уменьшение может осуществляться за счет выбора альтернативных маршрутов передачи, которые являются в данный текущий момент времени более выгодными, чем те, которые используются. Для сервиса ABR возможно уменьшение интенсивности трафика, вплоть до минимально допустимого значения. Следует отметить, что уменьшение интенсивности трафика возможно также для классов обслуживания с гарантированным качеством сервиса в случаях, когда их коэффициент ускорения больше единицы, а также для VBR классов с некоторым снижением качества передачи мультимедийной информации, телефонии и т.п.

Вышеперечисленные действия могут дать желаемый результат только при достаточно эффективной схеме управления обратной связью. Далее описаны дополнения этого алгоритма работы цепи обратной связи в ключе формирования трафика и управления сетевыми параметрами, необходимыми для модифицированной схемы:

Ввести два пороговых значения $CTDw_1$ и $CTDw_2$.

При $CTD > CTDw_1$, текущий коммутатор генерирует и отправляет предыдущему коммутатору на маршруте ячейку RM с установленным флагом W_1 и текущим значением *CTD*. Следует отметить, что здесь подразумевается общее время задержки доставки, однако пороговые значения *CTD* достигается на определенном коммутаторе при его чрезмерной загруженности или невозможности передать данные определенного соединения из-за отказов сети. Поэтому следует предпринять попытку «разгрузить» этот коммутатор или выбрать альтернативный маршрут для его обхода.

При получении W_1, коммутатор пытается определить альтернативный маршрут доставки, используя ***только*** свои связи, но за исключением двух – ту, которая идет к перегруженному коммутатору и входящую связь для

данного маршрута. Альтернативный маршрут должен удовлетворять условию: $CTDnew \le CTD + \Delta$, где: Δ – параметр, учитывающий временные издержки на установку нового соединения, реконфигурации маршрутных таблиц, фактор риска (чтобы избежать новых перегрузок и реконфигураций) и т.п.

1. Если такой маршрут найден, то производится переустановка соединения и все сопутствующие ей действия, чтобы «перенести» перегруженное соединение на новый маршрут. Если маршрут не найден, то W_1 игнорируется.
2. При $CTD > CTDw_2$, текущий коммутатор генерирует и отправляет предыдущему коммутатору на маршруте ячейку RM с установленным флагом *W2* и текущим значением *CTD*. W_2 сигнализирует о перегрузке близкой к критической.
3. При получении W_2, коммутатор выполняет действия аналогичные пункту 3. Если необходимый маршрут не может быть найден средствами данного коммутатора, то ячейка с W_2 передается коммутатору предыдущему с точки зрения перегруженного маршрута, вплоть до источника. Источник при этом может реагировать на получение предупреждающих ячеек как изменением скорости передачи (ее понижением), так и выбором альтернативных маршрутов.
4. Вышеперечисленные пункты алгоритма выполняются только для трафика с гарантированным качеством обслуживания. Для трафика «лучшей попытки» передается только ячейка W_2, и передается она по цепочке коммутаторов к источнику с требованием понизить интенсивность передачи.
5. Источники трафика ABR и UBR могут также попытаться изменить маршрут, однако такое решение должно приниматься на уровне самих источников и поддерживаться сетью.

Управление трафиком с учетом его самоподобия

В работе [1] предложены механизмы управления трафиком для асинхронного режима передачи информации. Очевидно, что они должны включать в себя средства обратной связи для всех классов QoS, т.к. при необходимости нужно передавать источникам требования снизить скорость передачи. Такая система обратной связи имеет также дополнительные преимущества. Одним из них является контроль параметров задержки со стороны источника, что позволяет предсказывать перегрузки в сети и предпринимать соответствующие меры, вплоть до обхода перегруженного участка. Это особенно актуально в случаях, когда перегрузки вызваны отказами промежуточных коммутаторов. Обход загруженного участка может также осуществляться для увеличения скорости передачи с целью минимизировать общую задержку передачи трафика для данного соединения и более равномерного распределения загрузки по сети.

Для эффективной работы схемы управления нагрузкой необходимо обеспечить «справедливую» дисциплину обслуживания, чтобы иметь некоторую гарантию того, что увеличение трафика для определенного направления связано с общей перегрузкой коммутатора, а не дискриминацией из-за присутствия более приоритетного трафика. Кроме того, она должна эффективно работать с передачей на повышенных скоростях и быстро реагировать на необходимость изменения скорости, чтобы предпринимать соответствующие меры. Для определения наиболее подходящей схемы рассмотрим математическую модель трафика в сети. Последние исследования [2] в области передачи данных показали, что используемый ранее пуассоновский закон распределения дает достаточно высокую погрешность, более лучшие результаты дает использование принципа самоподобия.

Возможность их взаимного влияния может приводить к существенному изменению параметров приложений и является причиной возникновения сложных бифуркационных явлений [3], которые проявляется в неравномерном или взрывном характере протекания сетевых процессов.

Поэтому для управления такими процессами необходимо использовать адекватные математических моделей, учитывающие в конструктивной форме как статистический характер возмущении, так и динамику передачи пакетов на различных уровнях протоколов межсетевого взаимодействия, включая временные и пространственные характеристики. В основе данных моделей лежит принцип самоподобия с учетом которого и рассматриваются математические модели трафика и задержек в узлах коммутации.

Алгоритм, формирующий “окно” перегрузки по сигналу обратной связи в виде пакетов подтверждения, называется алгоритмом скользящего “окно”, так как при очередном получении пакета подтверждения “окно” перемещается (и увеличивается на один пакет на каждый шаг алгоритма), захватывая очередные данные, которые разрешается передавать без подтверждения. Чтобы избежать длительных простоев из-за ожидания потерянных или задержавшихся пакетов вводится пороговое значение RTT_0. При превышении задержкой RTT порогового значения RTT_0 пакеты считаются потерянными. Величина RTT_0 определяется с помощью адаптивного алгоритма TCP/IP протокола, порядок работы которого включает следующие операции. Измеряются ряд значений RTT-задержки, полученные значения усредняются с весовыми коэффициентами, возрастающими от предыдущего измерения к последующему, а затем полученный результат умножается на некоторый коэффициент.

Очевидно, в связи с уменьшением окна перегрузки пропускная способность соединения снижается. Анализ работы существующего алгоритма работы протокола показывает, что имеются значительные резервы уменьшения потерь в пропускной способности в режимах медленного старта и управления

перегрузкой. Во-первых, перегрузка не прогнозируется, а обнаруживается, по самому факту отсутствия пакетов подтверждения после очередного перемещения окна, во-вторых, по этому факту нельзя судить о величине перегрузки и следующее состояние “окно” устанавливается методом «проб и ошибок». Задача заключается в формировании заблаговременно, не заходя в критическую область обнаружения потерянных пакетов, оценок как месторасположения временного интервала возможного проявления перегрузки, так и величины самой перегрузки. Решение такой задачи можно получить в рамках методов идентификации процессов, использования при моделировании RTT-задержек фрактального броуновского движения и формирования оценок прогноза. Оптимальная оценка прогноза RTT-задержки для момента времени t_{n+k} принимает вид [4]:

$$\hat{T}_{n+k} = \hat{B}_H\left(t_{n+k}\right) - \hat{B}_H\left(t_{n+k-1}\right) + T_0 + \Delta T_{cp}\,; \qquad (1)$$

где: $\hat{T}_{n+k}$ – оптимальная оценка прогноза RTT для момента времени t_{n+k};

RTT- интервал между посылкой пакета и получением подтверждения;

$\hat{B}_H\left(t_{n+k}\right)$ – оптимальная оценка процесса дробного броуновского движения с параметром Херста для момента времени t_{n+k} ;

$\hat{B}_H\left(t_{n+k-1}\right)$ – оптимальная оценка процесса дробного броуновского движения с параметром Херста для момента времени t_{n+k-1};

T_0 – минимальная составляющая интервала времени без очереди;

ΔT_{cp} – среднее приращение RTT задержки.

Остановимся подробней на оценке прогноза следующего за последним измеренным для момента времени t_n значением фрактального броуновского движения. На основании формулы (1) она равна

$$\hat{T}_{n+1} = \hat{B}_H\left(t_{n+1}\right) - B\left(t_n\right) + T_0 + \Delta T_{cp}\,; \qquad (2)$$

С учетом формулы $r_н(1; \Delta) = 2^{2H-1}-1$ оценка прогноза RTT-задержки принимает вид:

$$\hat{T}_{n+1} = \left(2^{2H-1} - 1\right)\left[T_n - \left(T_0 + \Delta T_{cp}\right)\right] + T_0 + \Delta T_{cp}\,; \qquad (3)$$

Например, наличие возможной перегрузки на n+l цикле могло быть обнаружено и тем самым с большой вероятностью предотвращено по результатам измерений RTT-задержек или фрактального броуновского движения на m предшествующих циклах, формирования согласно формуле (2) соответствующей оценки прогноза и сравнения ее с тайм-аутом. Кроме того, по величине спрогнозированной RTT-задержки на n+l цикле можно судить об уровне перегрузки и по сигналу обратной связи соответственно уменьшить (перенастроить) величину окна, то есть определить закон изменения окна перегрузки.

Прогнозируемая оценка X_{n+1} определяется или по последнему результату измерений в соответствие с формулой оценки прогноза отчета:

$$\hat{X}_{n+k} = r(k;T)(X_n - \lambda T) + \lambda T \,, \qquad (4)$$

где: λ – интенсивность поступления пакетов; *k* - параметр смещения;

X_n – число отсчетов;

$\hat{X}_{n+k}$ – оптимальное значение оценки прогноза отсчета; *r(k;T)* – коэффициент корреляции.

Если величина прогноза оказывается больше порога, зависящего от уровня загрузки буфера и определяемого некоторым адаптивным алгоритмом, то по сигналу обратной связи интенсивность генерации информации с узла *i* уменьшается на величину, зависящую от уровня загрузки буфера и значений оценок прогноза. Хотя пропускная способность из-за этого на участке сети и понижается, но в связи с сохранением процесса передачи информации и уменьшением числа потерянных пакетов удается в среднем уменьшить потери в пропускной способности этого соединения. Если необходимо сохранить пропускную способность, то по сигналу обратной связи доступные сетевые ресурсы перераспределяются в пользу этого участка виртуального соединения сети.

Выводы

1. Для мобильных сетей АТМ на базе стандартного протокола PNNI разработаны процедуры управления виртуальными соединениями, учитывающие особенности предлагаемых способов формирования и реконфигурации виртуальных соединений.
2. Предложен и обоснован математический аппарат расчета времени задержки передачи ячеек в сети АТМ с использованием которого проведен анализ эффективности известных способов уменьшения времени задержки передачи информации.
3. Предложен способ и разработан алгоритм управления потоком на основе формирования цепи обратной связи сигнальных ячеек, позволяющий с помощью введения пороговых значений задержки оптимизировать сетевой трафик.

Список использованной литературы

1. Кулаков Ю.А., Асад Махмуд Асад Аль Насер, Муаффаг Ахмед Абдул-Рахман Абу-Алхайжа. Управление трафиком в беспроводных сетях АТМ //Вимірювальна та обчислювальна техніка в технологічних процесах.–2001.-№ 4.-С. 99-104.
2. Fractional Levy motion and its application to network traffic modelling". N. Laskin, I. Lambadaris, F.Harmantzis, M. Devetsikiotis // Computer Networks 40, 2002, 363-375
3. Городецкий А.Я., Иванов А.В. Фрактальные процессы в компьютерных сетях: методы идентификации моделей трафика // Научно-технические ведомости СПбГТУ, 2001, № 3 (25).
4. Шелухин О.И., Тенякшев А.М., Осин А.В. Фрактальные процессы в телекоммуникациях. Монография / Под ред. О.И.Шелухина - М.: Радиотехника,

2003.- 480 с

УДК 519.854.2

ЖУРАКОВСЬКИЙ Я.Ю.,
ЖУРАКОВСЬКА О.С.

ЗАСТОСУВАННЯ МАТЕМАТИЧНИХ МОДЕЛЕЙ ТА МЕТОДІВ У СИСТЕМАХ MRP II, ERP, APS

В статті розглянуто принципи побудови систем MRP II, ERP, APS та проаналізовано математичні моделі та методи, що знаходять застосування в таких системах. Зокрема, сформульовано ряд задач теорії розкладів, що виникають на різних рівнях планування вказаних систем.

Principles of construction of the MRP II, ERP, APS systems are considered in the article. Mathematical models and methods which are used in such systems are analyzed. The row of tasks of Scheduling theory which presents at different levels of planning is formulated.

1. Принципи побудови систем MRP II, ERP, APS

В останні десятиліття у зв'язку із зростанням складності промислового виробництва, підвищенням вимог до якості продукції виникла потреба в удосконаленні методології та технології управління, а також у застосуванні обчислювальної техніки до підтримки розв'язання цих задач. Сучасні системи управління спираються на методологію планування, що охоплює коло задач управління промисловим підприємством на оперативному рівні, в основі якої лежить стандарт MRP II (Manufacturing Resource Planning). MRP II базується на таких принципах [3]:

- ієрархічність, тобто розділення планування на рівні, що відповідають різним ступеням управлінської відповідальності. Плани підприємства розробляються зверху вниз, при цьому передбачено забезпечення зворотного зв'язку;
- інтегрованість, тобто об'єднання на оперативному рівні основних функціональних сфер діяльності підприємства, пов'язаних із матеріальними та фінансовими потоками на підприємстві;
- інтерактивність, тобто можливість дослідження впливу ймовірних ситуацій на результати діяльності підприємства та його підрозділів на різних рівнях ієрархії планових рішень.

До основних обов'язкових модулів системи MRP II належать, зокрема:

Головний календарний план виробництва (Master Production Schedule, MPS), що описує план, виходячи із номенклатурних позицій незалежного попиту (що, коли та скільки виробляти). Решта планів базується на MPS, який, в свою чергу, розробляється на основі плану виробництва (плану продаж та операцій), а також детальних планів продаж для кожної номенклатурної позиції, включеної до MPS. Детальний план продаж визначає

пріоритети MPS, тобто порядок та терміни виробництва продукції в межах планового періоду. Розробка MPS є поєднуючою ланкою між різними функціями підприємства: маркетингом та збутом, виробництвом, а також конструкторсько-технологічною підготовкою виробництва;

Оперативне управління виробництвом (Shop Floor Control або Production Activity Control), або планування та диспетчеризація роботи цеху. Цей модуль дозволяє побудувати календарний план роботи цеху (пооперацій-но) та відстежувати його фактичне виконання. Графіки виконання виробничих замовлень (завдань) у розрізі технологічних операцій є найбільш деталізованим планом. Рівень деталізації цього графіка (по годинам, по хвилинам) залежить як від вимог виробництва у деталізації, так і від можливостей інформаційної системи.

Основними функціями запуску замовлень та диспетчеризації є такі [3]:

- управління пріоритетами замовлень;
- управління чергами замовлень до робочих центрів;
- планування та контроль вхідних/вихідних потоків на робочих центрах;
- формування та направлення послідовності замовлень;
- призначення робочим центрам виконання замовлень.

Управління пріоритетами замовлень є функцією призначення та модифікації пріоритетів виробничих замовлень відповідно до політики, що встановлена на підприємстві. Можливими критеріями для оцінки успішності системи управління пріоритетами можуть бути, наприклад, такі [3]:

- доля виконаних у відповідності із графіком замовлень;
- середнє запізнення замовлення (в днях);
- обсяг незавершеного виробництва;
- час простоювання робочих центрів;
- мінімізація підготовчого часу.

Управління чергами замовлень до робочих центрів є актуальним питанням, оскільки довжина черги безпосередньо впливає на обсяг незавершеного виробництва та тривалість виробничого циклу. Отже, ця задача зводиться до досягнення найбільш повного використання критичних робочих центрів.

Наступним кроком у розвитку систем планування та управління є створення ERP-систем (Enterprise resource planning). За визначенням [1], ERP є фінансово-орієнтованою інформаційною системою для визначення та планування ресурсів усього підприємства, необхідних для того, щоб прийняти, виконати, відвантажити та відобразити в обліку замовлення клієнтів.

Зокрема, ERP системи мають такі характеристики [4]:

- готове програмне забезпечення, розроблене для середовища клієнт-сервер (традиційного та побудованого на інтернет технологіях);
- інтеграція більшості бізнес-процесів;

- охоплення більшої частини ділових операцій організації;
- використання бази даних усього підприємства;
- забезпечення доступу до даних в режимі реального часу;
- в деяких випадках можливість інтеграції обробки ділових операцій та планування (наприклад, виробниче планування).

Узагальнено структуру управління в системах ERP подано на рис. 1.

Таким чином, фактично ERP система є розширенням системи MRP II, оскільки її мета – інтеграція управління усіма ресурсами підприємства, а не тільки матеріальними, як це було в MRP II.

Слід відзначити особливість ERP систем, яка полягає фактично у збереженні підходів до планування виробництва, прийнятих у MRP II. Основна причина цього полягає у тому, що на початковому етапі переходу від MRP II до ERP потужність обчислювальних систем була недостатньою для забезпечення широкого застосування методів моделювання та оптимізації.

Необхідно також відзначити, що ERP системи не позбавлені недоліків. Наприклад, підвищення стійкості оперативних планів не завжди можна досягти збільшенням тривалості виробничого циклу, оскільки це, в свою чергу, викликає зростання обсягу незавершеного виробництва, зростання запасів та черг. Із зростанням черг збільшується неточність графіків, що не дозволяє підвищити точність поставок продукції замовникам. Крім вказаного недоліку слід також відзначити, що модель планування сучасних ERP систем можна також представити як набір окремих процесів, що мають все більш деталізовані обчислення, але які не завжди базуються на достатньо точних даних, що надходять від процесів верхніх рівнів ієрархії.

Можна відзначити, що сучасним напрямком розвитку ERP систем є APS системи (Advanced Planning and Scheduling System). При впровадженні нових принципів організації та управління виробництвом виявляються недоліки існуючих систем управління, що формує потребу в нових, складніших системах для планування та прийняття рішень в ході управління виробництвом.

Потенціальний ефект від впровадження APS систем, що базуються на нових методах управління, може полягати в наступному:

- суттєво вища точність планування;
- скорочення запасів;
- значне покращення діяльності в сфері поставок продукції;
- краще використання основних фондів;
- підвищення завантаження ресурсів;
- гнучкіше реагування на вимоги, викликані конкуренцією.

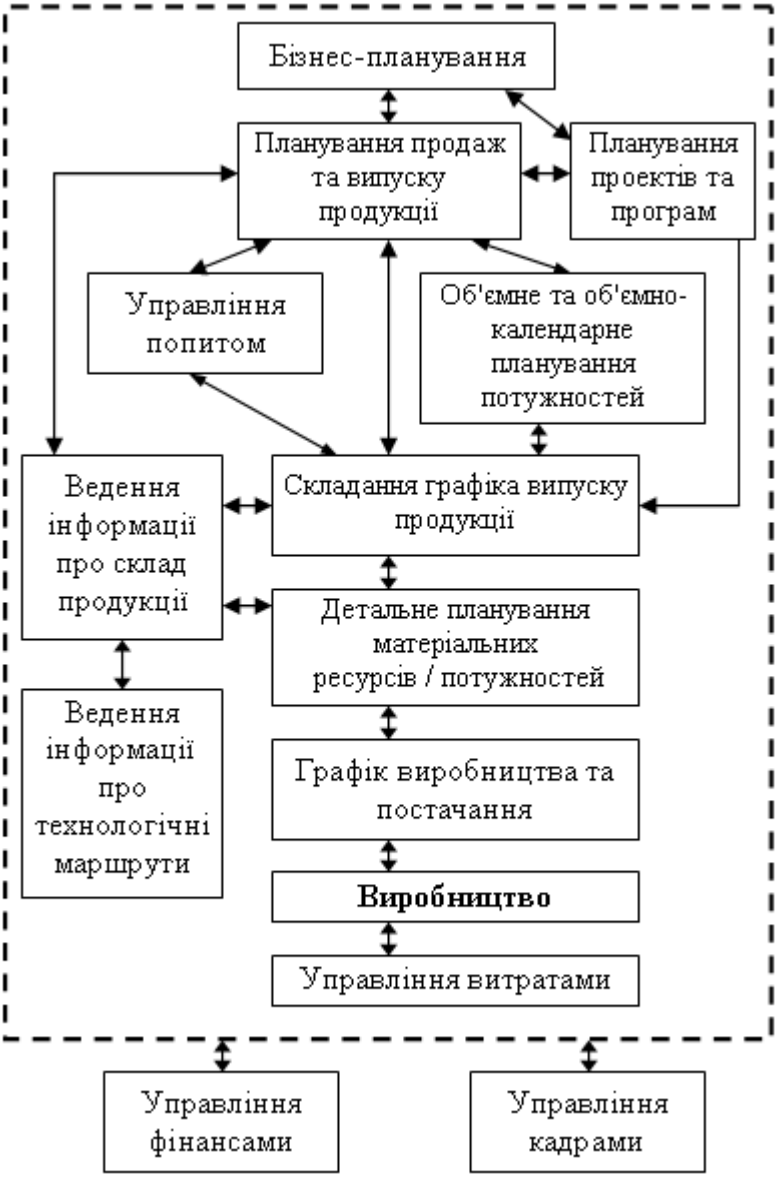


Рис. 1. Структура управління в системах ERP

Як правило, APS системи направлені на створення кращих планів з урахуванням усіх факторів, що обмежують можливості підприємства зі своєчасної поставки продукції. При цьому одночасно розглядаються ма-

теріальні потреби та виробничі потужності для оцінки існуючого плану та генерації нового. Деякі APS системи, крім цього, покращують цей план шляхом створення деталізованого виробничого графіку, що оптимізує черговість виконання робіт за критерієм пропускної здатності та обмеженнях на терміни виконання замовлень.

APS системи звичайно є композицією різних процесів, які використовують ті ж самі підходи до планування, але із різними обмеженнями та вхідними даними [5]. Структуру моделі APS зображено на рис. 2.

Рис. 2. Структура моделі APS

Найвищим рівнем планування в рамках розглянутої моделі є планування виробничого ланцюжка. Плани, що розробляються на цьому рівні, враховують обмеження на виробничі потужності по всьому виробничому ланцюжку.

Рівень планування діяльності підприємства об’єднує та оптимізує виконання функцій, що традиційно виконуються модулями: планування потреби в ресурсах; об’ємно-календарне планування потужностей; MPS; планування потреб розподільчої мережі; планування матеріальних потреб.

Модуль виробничого планування на підставі побудованого плану роботи підприємства розв’язує задачу календарного планування на більш деталізованому рівні з метою виконання термінів завершення замовлень.

Модуль оцінки можливості виконання забезпечує функціонування трьох попередніх рівнів. На цьому рівні використовується інформація із уже побудованого виробничого плану, а також дані про ресурси, необхідні для виробництва наявних, але не включених в план замовлень. Цей рівень підвищує точність визначення термінів виконання замовлень.

Необхідно відзначити, що оптимізація розв’язків є обов’язковою концепцією, закладеною в системи APS.

2. Математичні моделі та методи, що знаходять застосування в системах MRP II, ERP, APS

Оптимальне планування (в першу чергу в системах APS) може бути здійснено за допомогою економіко-математичних методів, під якими розуміємо комплекс математичних методів, що дозволяють знаходити оптимальні або наближені до оптимальних розв’язки економічних задач.

Вкажемо деякі з методів, що можуть застосовуватись для оптимального планування.

Лінійне програмування. В термінах лінійного програмування може бути сформульовано широке коло задач планування виробництва, фінансової діяльності, техніко-економічного планування, планування НДДКР.

Дискретне програмування. До таких задач можна віднести, наприклад, задачі розміщення обладнання, формування портфелю замовлень тощо.

У випадку, коли параметри моделі є випадковими величинами із відомими функціями розподілу, використовуються моделі *стохастичного програмування*.

Мережеві моделі та методи застосовуються, коли процес управління може бути представлений у вигляді графа, що описує взаємозв'язок робіт, ресурсів, часових витрат тощо.

Динамічне програмування. Моделі динамічного програмування можуть застосовуватись при розробці правил управління запасами, в процесі календарного планування виробництва або поточного та капітального ремонту складного обладнання та його заміни тощо.

Методи *багатокритеріальної оптимізації* дозволяють одержати розв'язок задачі за комплексом критеріїв, що відображають різні аспекти діяльності підприємства.

Методи *математичної статистики* застосовуються для розв'язання задач аналізу та прогнозування процесів на підприємстві.

Теорія управління запасами дозволяє визначати рівні запасів матеріалів, напівфабрикатів, виробничих потужностей та інших ресурсів в залежності від попиту на них.

Теорія розкладів є методологічною основою для розв'язання задач про впорядкування послідовності робіт. При цьому враховуються структура та параметри технологічного процесу. Оскільки більшість задач календарного планування, розв'язання яких необхідне для оптимального планування виробництва на різних рівнях планування в системах MRP II, ERP, APS, належать до задач теорії розкладів (ТР), то наведемо математичні постановки деяких із них.

3. Математичні постановки задач теорії розкладів

Для запису математичних постановок задач ТР будемо використовувати трьохпозиційну нотацію [2].

Для забезпечення сформульованих у розділі 1 вимог до планів, що формуються на різних рівнях планування системами MRP II, ERP, APS, доцільним є розгляд і розв'язання таких задач:

- однопроцесорні розклади:

$1\,|prec;\ r_i\,|\ C_{\max}$ (максимальний час виконання);

$1\,|\,r_i\ |\ L_{\max}$ (максимальне відхилення від директивних строків);

$1 \,|\,|\, \sum T_i$ (сумарне запізнення відносно директивних строків);

$1 \,|\, r_i \,|\, \sum C_i$ (сумарний час виконання);

$1 \,|\, p_i = p \,;\, r_i \,|\, \sum \omega_i C_i$ (сумарний зважений час виконання);

$1 \,|\,|\, \sum \omega_i T_i$ (сумарне зважене запізнення)

$1 \,|\,|\, \sum \omega_i U_i$ (сумарна (зважена) кількість завдань, завершених із запізненням);

$1 \,|\,|\, \sum \omega_i D_i$ (зважене випередження та запізнення відносно директивних строків);

- розклади для паралельних процесорів (без переривань):

$Q|\, p_i = p \,;\, r_i \,|\, C_{\max}$, $P|\, p_i = p \,;intree|\, L_{\max}$, $Pm|\, p_i = p \,;tree|\, \sum C_i$,

$Pm|\, p_i = p \,;\, r_i \,|\, \sum \omega_i U_i$, $Q|\, p_i = p \,|\, \sum \omega_i T_i$, $P|\, p_i = p \,;\, r_i \,|\, \sum \omega_i C_i$, $R|\,|\, \sum C_i$;

- розклади для паралельних процесорів (з перериваннями):

$P|tree;pmtn|\, C_{\max}$, $P|intree;pmtn|\, L_{\max}$, $P|\, p_i = p \,;pmtn|\, \sum \omega_i C_i$,

$Q|\, p_i = p \,;pmtn|\, \sum U_i$, $Q|\, p_i = p \,;prec;pmtn;\ r_i \,|\, \sum C_i$,

$Qm|\, p_i = p \,;pmtn;\, r_i \,|\, \sum \omega_i U_i$, $Rm|\, pmtn|\, \sum T_i$, $Q|\, p_i = p \,;\, pmtn|\, \sum \omega_i T_i$;

- розклади для систем довільного типу (open shop):

$O|\,|\, C_{\max}$, $O|\, p_{ij} = 1;\, outtree|\, L_{\max}$, $O|\, p_{ij} = 1;\ r_i \,|\, \sum U_i$,

$O|\, p_{ij} = 1 \,|\, \sum \omega_i U_i$, $O|\, p_{ij} = 1;\ r_i \,|\, \sum \omega_i T_i$;

- розклади для поточного виробництва (flow shop):

$F|\, p_{ij} = 1;\, tree|\, C_{\max}$, $F|\, p_{ij} = 1;\, intree|\, L_{\max}$,

$F2|\, p_{ij} = 1;\, chains;\, |\, \sum T_i$, $F2|\, p_{ij} = 1;\, chains;\, |\, \sum \omega_i C_i$,

$F|\, p_{ij} = 1;\, chains;\ r_i \,|\, L_{\max}$, $F|\, p_{ij} = 1;\, intree|\, \sum C_i$,

$F|\, p_{ij} = 1;\ r_i \,|\, \sum \omega_i U_i$, $F|\, p_{ij} = 1;\ r_i \,|, \sum \omega_i T_i$.

Висновки

Розглянуто принципи побудови систем MRP II, ERP, APS. Сформульовано вимоги до планів, що генеруються на різних рівнях планування вказаних систем. Наведено критерії, якім повинні задовольняти ці плани.

Наведено ряд математичних моделей та методів, що знаходять застосування у системах MRP II, ERP, APS з метою здійснення оптимального управління виробництвом.

Сформульовано математичні постановки задач теорії розкладів, які виникають на різних рівнях планування розглянутих систем, у відповідності до наведених критеріїв. Розв’язання цих задач необхідне для досяг-

нення мети побудови оптимальних планів в процесі управління виробництвом.

Список посилань

1. APICS Dictionary, 10th ed. American Priduction and Inventory Control Society, 2002.
2. Graham, R.L., Lawler, E.L., Lenstra, J.K., Rinnooy Kan, A.E.G.: Optimization and approximation in deterministic sequencing and scheduling: a survey, Ann. Discrete Math. 5, 287–326, 1979.
3. Гаврилов Д.А. Управление производством на базе стандарта MRP II. 2-е изд.–СПб.:Питер, 2005.–416с.
4. О'Лири, Дэниел. ERP системы. Современное планирование и управление ресурсами предприятия. Выбор, внедрение, эксплуатация/ Дэниел О'Лири;[Пер. с англ. Ю.И. Водяновой].– М.:ООО «Вершина», 2004.– 272с.
5. Рыбников А.И. Система управления предприятием типа ERP – М.: Аэроконсалт, 1999 – 214с.

УДК 681.3.06

БОЛДАК А.А. ,
НЕВДАЩЕНКО М.В.

МАТЕМАТИЧЕСКИЙ АППАРАТ ФОРМАЛИЗАЦИИ МОДЕЛЕЙ, ИСПОЛЬЗУЕМЫХ ПРИ ПРОЕКТИРОВАНИИ ИНФОРМАЦИОННЫХ СИСТЕМ

Формализованы понятия класса и отношений между классами, которые соответствуют терминам объектно-ориентированного подхода к проектированию информационных систем. Показано, что классы и операции, определенные на них, являются алгебраической системой, которая позволяет эффективно решать задачи, связанные с взаимнооднозначными преобразованиями моделей в рамках объектно-ориентированного подхода к проектированию информационных систем. Приведены примеры решения некоторых типичных задач с применением предложенного математического аппарата.

Classes and its relations concept that correspond the terms of the object-oriented information systems' design approach. Classes and its operations are the algebraic system. The system allows effective tasks' solving for model transformations in object-oriented design approach.

Проектирование программных систем связано с использованием различных моделей: концептуальных, проектных и моделей реализации.

Сложность задач, связанных с переходом от проектной модели к модели реализации, определяется необходимостью учета дополнительных ограничений, связанных с использованием выбранных программных технологий и инструментальных средств разработки. В случае использования многоуровневой архитектуры, ее решение становится еще более сложным, т.к. необходимо определить, каким образом проектные классы будут отображаться в одном или нескольких слоях такой архитектуры, тем более что на различных уровнях могут использоваться собственные формализмы для описания моделей.

Типичная в этом смысле ситуация возникает при реализации персистентных объектов, которые должны быть представлены как в программной среде, так и в СУБД.

Во многих источниках [1 – 5] описаны подходы и методы, которые могут применяться для решения подобных задач. Однако, по признанию самих авторов статей, область применимости предложенных ими методов ограничивается оптимизацией отдельных видов запросов [1, 3] и не может применяться для решения задачи отображения программных классов в СУБД в общем случае.

Целью работы является формулировка такого математического аппарата, который мог бы эффективно применяться для решении задач, свя-

занных с эквивалентными преобразованиями моделей в рамках объектно-ориентированного подхода к проектированию информационных систем.

В основе разрабатываемого математического аппарата лежит понятие класса, определяющего посредством спецификации типизированное множество и операции, которые образуют кольцо на множестве таких классов.

Алгебра Классов

Определение класса

Классом будем называть множество, определенное с помощью задания свойства

$$C = \{I \mid P(I)\}, \ (1)$$

где I – (от *Instance*) элемент (экземпляр) класса, в качестве которых могут выступать любые множества, в том числе и пустое, такие, что для них имеет место

$$\{i \in I \mid P(i)\} = \{i \mid P(i)\};$$

$P(I) = I \xrightarrow{P} \{0,1\}$ – (от *Property*) характеристическая функция, которая принимает значение 1, если I принадлежит классу, и 0 – в противном случае.

Характеристическая функция определяет наличие у элементов $i \in I$ некоторого свойства. Если экземпляры имеют несколько свойств, для каждой из которых можно определить $p_k, k = \overline{1,n}$, то общая характеристическая функция может быть представлена их конъюнкцией:

$$P(I) = \mathop{\&}_{k=1}^{n} p_k(I). \ (2)$$

Благодаря тому, что для логической операции конъюнкции имеет место коммутативность, функция $P(I)$ может взаимно однозначно отображаться во множество $\{p_1(I), p_2(I), ..., p_n(I)\}$, заданное перечислением свойств, которое будем называть спецификацией класса. Т.е.,

$$P(I) \leftrightarrow S = \{p_1(I), p_2(I), ..., p_n(I)\},$$

где S – (от *Specification*) спецификация класса.

Поскольку спецификация однозначно определяет характеристическую функцию класса, то она также определяет и класс. Класс, определенный с помощью спецификации, будем обозначать $C(S)$ или $C(S = \{p_1(I), p_2(I), ..., p_n(I)\})$.

Переход к спецификации класса, заданного перечислением свойств, дает возможность достаточно просто решать задачи классификации экземпляров.

По аналогии с классической теорией множеств определим понятие нуль-класса и универсального класса.

Нуль-классом является класс $C^0(S^0 = \varnothing)$.

Таким образом, если спецификация класса является пустым множеством, т.е. к элементам класса не выдвигается никаких требований, то любой экземпляр может рассматриваться как представитель этого класса.

Универсальным классом является класс $C^U(S^u = U_p)$, где S^u – универсальная спецификация, которая содержит множество всех свойств (универсум свойств).

Понятно, что U_p является конечным множеством, которое определяется в рамках каждой конкретной задачи. Однако, условие конечности U_p не нарушает условий универсума, т.е. всегда (в рамках задачи, которая рассматривается) имеет место

$$\forall S : S \cap S^U = S .$$

Смысл универсальной спецификации определяется тем, что к объекту в процессе его проектирования применяют макроподход [6], в рамках которого предполагается, что объект рассматривается как “черный ящик”, свойства которого определены и известны, но нет никаких предположений о его строении, что соответствует определению класса $C^U(S^U)$.

Суперклассом класса C есть класс $C_1(S_1) = SUPER[C(S)] : S_1 \subseteq S$, спецификация которого является подмножеством спецификации класса C.

Потомком класса C есть класс $C_1(S_1) = CHILD[C(S)] : S_1 \supseteq S$, если спецификация класса C является подмножеством спецификации класса C_1.

Очевидно, что отношение потомков и суперклассов являются обратными.

Операции с классами

Дополнения класса

Дополнением класса $C(S)$ до универсального класса $C^U(S^U)$ (или просто – дополнением класса $C(S)$) есть класс $\overline{C}(S^U \setminus S)$, определяемый спецификацией, которая является дополнением S до универсальной спецификации S^U.

Разностью классов $C_1(S_1)$ и $C_2(S_2)$ является класс $C_1 \setminus C_2(S_1 \setminus S_2)$.

Обобщение классов

Обобщением классов $C_1(S_1)$ и $C_2(S_2)$ является класс $C_1 \overset{G}{-} C_2(S_1 \cap S_2)$, определяемый спецификацией, которая получается в результате пересе-

чения спецификаций S_1 и S_2. Знак $\overset{G}{—}$ обозначает операцию обобщения (от *Generalization*).

Название операции отвечает процедуре определения общих (совместных) свойств объектов, в данном случае – классов $C_1(S_1)$ и $C_2(S_2)$. Очевидно, что множество этих свойств содержится в пересечении соответствующих спецификаций.

Расширение классов

<u>Расширением</u> классов $C_1(S_1)$ и $C_2(S_2)$ является класс $C_1 \overset{E}{—} C_2(S_1 \cup S_2)$, определяемый спецификацией, которую получают в результате объединения спецификаций S_1 и S_2. Знак $\overset{E}{—}$ обозначает операцию расширения (от *Extension*).

Свойства операции дополнения

Утверждение 1. Расширение класса и его дополнения образуют универсальный класс.

Действительно, если рассматривать класс $C(S)$ и класс $\overline{C(S)}$, то

$$C(S) \overset{E}{—} \overline{C(S)} = C(S \cup \overline{S}) = C(S^U) = C^U .$$

Утверждение 2. Обобщение класса и его дополнения образуют нуль-класс.

По аналогии с доказательством предыдущего утверждения

$$C(S) \overset{G}{—} \overline{C(S)} = C(S \cap \overline{S}) = C(\varnothing = S^0) = C^0 .$$

Утверждение 3. Дополнение к дополнению класса является самим классом.

Т.е., $\overline{\overline{C(S)}} = \overline{C(\overline{S})} = C(\overline{\overline{S}}) = C(S)$.

Утверждение 4. Для классов имеет место соотношение $C_1 \setminus C_2 = C_1 \overset{G}{—} \overline{C_2}$.

Учитывая свойства обобщения, получим

$$C_1(S_1) \overset{G}{—} \overline{C_2(S_2)} = C(S_1 \cap \overline{S_2}) = C(S_1 \cap S^u \setminus S_2) = C((S_1 \cap S^u) \setminus S_2) =$$

$$C(S_1 \setminus S_2) = C(S_1) \setminus C(S_2).$$

Свойства операции обобщения

Утверждение 5. Обобщение классов является коммутативным $C_1 \overset{G}{—} C_2 = C_2 \overset{G}{—} C_1$.

Справедливость этого утверждения следует из коммутативности операции пересечения множеств.

$$C_1(S_1) \overset{G}{—} C_2(S_2) = C(S_1 \cap S_2) = C(S_2 \cap S_1) = C_2 \overset{G}{—} C_1$$

Утверждение 6. Обобщение классов является суперклассом для всех его потомков.

Основываясь на определении суперкласса и операции обобщения, получим

$$C_1(S_1) \overset{G}{—} C_2(S_2) = C(S = S_1 \cap S_2), S \subseteq S_1, S \subseteq S_2 \Rightarrow$$

$$C_1(S_1) \overset{G}{—} C_2(S_2) = SUPER(C_1), C_1 = CHILD(C_1(S_1) \overset{G}{—} C_2(S_2));$$

$$C_1(S_1) \overset{G}{—} C_2(S_2) = SUPER(C_2), C_2 = CHILD(C_1(S_1) \overset{G}{—} C_2(S_2)) .$$

Утверждение 7. Обобщение класса с собою является самим классом $C \overset{G}{—} C = C$.

Утверждение 8. Обобщение класса и нуль-класса является нуль-классом $C \overset{G}{—} C^0 = C^0$.

Доказательства утверждений 7 и 8 тривиальны.

Свойства операции расширения

Утверждение 9. Расширение классов является коммутативным $C_1 \overset{E}{—} C_2 = C_2 \overset{E}{—} C_1$.

По аналогии с доказательством утверждения 5 получаем

$$C_1(S_1) \overset{E}{—} C_2(S_2) = C(S_1 \cup S_2) = C(S_2 \cup S_1) = C_2 \overset{E}{—} C_1$$

Утверждение 10. Любой класс является суперклассом для его расширения с любым другим классом.

Действительно,

$$C_1(S_1)\overset{E}{—}C_2(S_2) = C(S = S_1 \cup S_2), S_1 \subseteq S, S_2 \subseteq S \Rightarrow$$

$$C_1(S_1)\overset{E}{—}C_2(S_2) = CHILD(C_1), C_1 = SUPER(C_1(S_1)\overset{E}{—}C_2(S_2));$$

$$C_1(S_1)\overset{E}{—}C_2(S_2) = CHILD(C_2), C_2 = SUPER(C_1(S_1)\overset{E}{—}C_2(S_2)).$$

Утверждение 11. Расширение класса с собой – есть сам класс $C\overset{E}{—}C = C$.

Утверждение 12. Расширение класса и нуль-класса – есть сам класс $C\overset{E}{—}C^0 = C$.

Связь прямого произведения множеств с операцией расширения классов

Известно, что операция прямого произведения множеств является единственным средством создания агрегатов, т.е. множеств, элементы которых состоят из нескольких частей. Рассмотрим, как влияет операция прямого произведения множеств на классы, с помощью которых эти множества задаются.

Допустим, что есть множества $A = \{a \mid \underset{S_A}{\&}\, p(a)\}$ и $B = \{b \mid \underset{S_B}{\&}\, r(b)\}$, которые описываются соответствующими классами $C_A(S_A = \{p(A)\})$, $C_b(S_b = \{r(B)\})$. Аргументами функций $p(a)$ и $r(b)$ являются, соответственно, элементы множеств *A* и *B*.

Вследствие прямого произведения $A \times B$ получаются элементы ⟨*a*,*b*⟩, для которых одновременно должны выполнятся ограничения $\underset{S_A}{\&}\, p(a)$ и $\underset{S_B}{\&}\, r(b)$, наложенные отдельно на множества *A* и *B*, т.е. $S_{A\times B} = \underset{S_A}{\&}\, p(a) \,\&\, \underset{S_B}{\&}\, r(b)$, но в этом случае функции *p* и *r* своими аргументами имеют кортежи ⟨*a*,*b*⟩. Приведение функций *p* и *r* к общей области определения выполняется с помощью проекций

$$S_{A\times B} = \underset{S_A}{\&}\, p(\Pi_A\langle a,b\rangle) \,\&\, \underset{S_B}{\&}\, r(\Pi_b\langle a,b\rangle).$$

Учитывая то, что $p(\Pi_A\langle a,b\rangle) = \Pi_A p(\langle a,b\rangle) = p'(\langle a,b\rangle)$ – некоторая функция, определенная на множестве $A \times B$ (как и для r), предыдущую формулу можно переписать в виде

$$S_{A\times B} = \mathop{\&}_{S_A} p(\langle a,b\rangle) \,\&\, \mathop{\&}_{S_B} r(\langle a,b\rangle)\,.$$

Учитывая ассоциативность и коммутативность операции конъюнкции, можно утверждать, что

$$C_{A\times B} = C_A \overset{E}{—} C_B\,.$$

Последнее имеет важное практическое значение, поскольку позволяет упрощенно (по отношению к операции прямого произведения) определять агрегатные объекты с помощью операции расширения, в основе которой лежит объединение соответствующих конечных спецификаций, заданных набором свойств.

Определение алгебры классов

Утверждение 13. Операции расширения, обобщения и дополнения образуют, по крайней мере, кольцо на множестве классов.

Для доказательства этого утверждения необходимо проверить ограничения, которым должна удовлетворять рассматриваемая система.

<u>Ассоциативность</u> операции расширения классов следует из свойств спецификаций, а именно

$$C_1 \overset{E}{—} (C_2 \overset{E}{—} C_3) = C_1 \overset{E}{—} C(S_2 \cup S_3) = C(S_1 \cup (S_2 \cup S_3)) = C((S_1 \cup S_2) \cup S_3) =$$

$$C(S_1 \cup S_2) \overset{E}{—} C_3 = (C_1 \overset{E}{—} C_2) \overset{E}{—} C_3.$$

<u>Единицей</u> для операции расширения является ноль-класс C^0 (см. Утверждение 12).

<u>Противоположным элементом</u> класса C по операции расширения является его дополнение $\overline{C}$ (см. Утверждение 2).

Операция расширения является <u>коммутативной</u> (см. Утверждение 9).

Таким образом, для множества классов и операции расширения выполняются условия существования группы, т.е. $(\overset{E}{—}, \{C\}, C^0)$ является абелевой группой.

Для операции обобщения очевидна <u>замкнутость</u>, <u>ассоциативность</u> доказывается аналогично предыдущему.

Установлено, что обобщение является коммутативной операцией (см. Утверждения 5), т.е. левый и правый дистрибутивные законы объединяются.

Покажем, что для расширения и обобщения имеет место дистрибутивный закон.

$$C_1(S_1)\overset{G}{-}[C_2(S_2)\overset{E}{-}C_3(S_3)] = C_1(S_1)\overset{G}{-}C(S_2 \cup S_3) = C(S_1 \cap (S_2 \cup S_3)) =$$

$$C((S_1 \cap S_2) \cup (S_1 \cap S_3)) = C(S_1 \cap S_2)\overset{E}{-}C(S_1 \cap S_3) = \qquad (3)$$

$$[C_1(S_1)\overset{G}{-}C_2(S_2)]\overset{E}{-}[C_1(S_1)\overset{G}{-}C_3(S_3)]$$

Таким образом, выполняются также требования, обеспечивающие существования кольца, т.е. $(\overset{G}{-},\overset{E}{-},\{C\},C^0,C^U)$ является коммутативным кольцом с единицей C^U.

Преобразования диаграмм классов

Приведенная выше теория классов, по своей терминологии совпадает с терминами объектно-ориентированного проектирования, которое в настоящее время используется в качестве методологической основы при проектировании информационного и программного обеспечения [7].

Выводы о том, что $(\overset{G}{-},\overset{E}{-},\{C\},C^0,C^U)$ является замкнутой алгебраической системой, позволяет получить преобразования классов и их взаимоотношений, которые, с одной стороны, сохраняют свойства классов, с другой – учитывают дополнительные ограничения.

В качестве графического представления классов и отношения между ними будем использовать UML–диаграммы [8]. Пример такой диаграммы и необходимые объяснения изображены на Рис. 1.

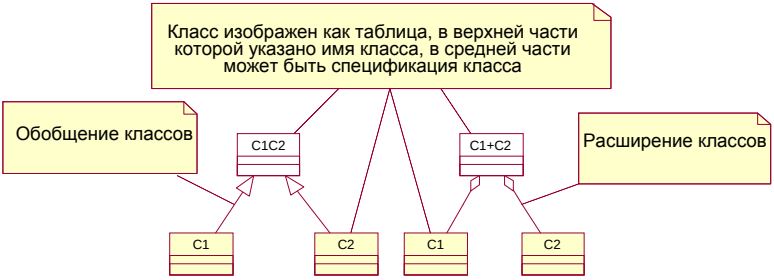


Рис. 1. Пример UML–диаграммы

Взаимное преобразование обобщения и расширения классов

Необходимость таких преобразований обусловлена тем, что в некоторых моделях, а, соответственно, и в языках программирования, не используется понятие обобщений, например в реляционной модели данных [9], которая поддерживается большинством СУБД.

В этом случае необходимы преобразования обобщений классов, т.е. такое эквивалентное представление, в котором сохраняются свойства классов и отсутствуют их обобщения.

Утверждение 14. Любой класс может быть представлен расширением своего суперкласса и его дополнения к рассматриваемому классу.

Пусть рассматривается класс *C*, при чем известно, что $C = CHILD(C_S)$. Тогда:

$$C_S \overset{E}{—} (C \setminus C_S) = C_S \overset{E}{—} (C \overset{G}{—} \overline{C_S}) \quad (4)$$

Учитывая то, что $S \supseteq S_S$, имеет место поглощение $C_S \overset{G}{—} C = C_S$.

Таким образом, использование поглощения, дистрибутивного закона и свойств обобщения в (4) дает следующий результат:

$$C_S \overset{E}{—} (C \overset{G}{—} \overline{C_S}) = (C \overset{G}{—} C_S) \overset{E}{—} (C \overset{G}{—} \overline{C_S}) = C \overset{G}{—} (C_S \overset{E}{—} \overline{C_S}) = C \overset{G}{—} C^U = C$$

Изобразим этот вывод в виде UML– диаграммы (Рис. 2).

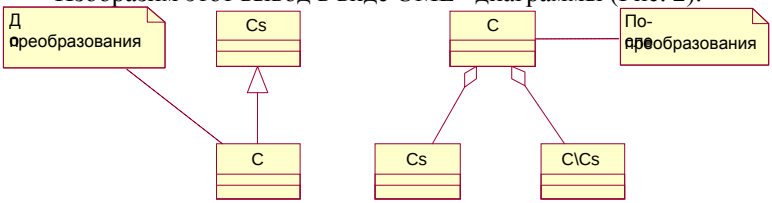


Рис. 2. Взаимные преобразования обобщения и расширения классов

Приведем пример использования эквивалентных преобразований для обобщения и расширения.

При разработке базы данных для учета успеваемости учеников были определены классы УЧЕНИК ({ФИО, Адрес, Класс}) и УЧИТЕЛЬ ({ФИО, Адрес, Должность}), на которых можно получить обобщение ПЕРСОНА ({ФИО, Адрес}). Соответствующая диаграмма изображена на Рис. 3.

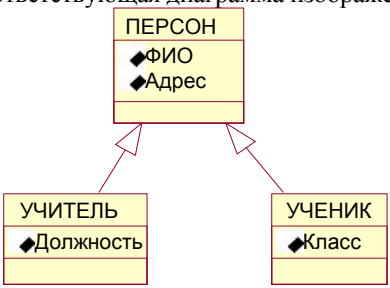


Рис. 3. Диаграмма классов до преобразования

Однако, для того, чтобы определить эти классы в реляционной модели необходимо избавится от обобщений, но таким образом, чтобы класс ПЕРСОНА сохранился. Это возможно благодаря утверждению 4 (Рис. 4).

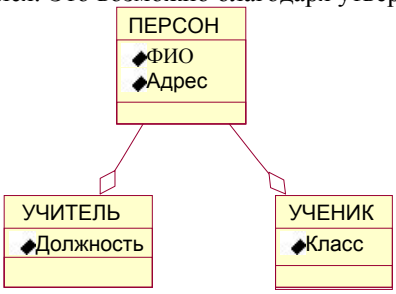


Рис. 4. Диаграмма классов после преобразования

Таким образом, в соответствии с реляционной моделью, этот фрагмент можно представить тремя таблицами, которые связаны между собой внешними ключами. Внешние ключи реализуют в данном случае операцию расширения.

Соответственно, на языке SQL эта диаграмма описывается следующим образом:

```
CREATE TABLE ПЕРСОНА    (     Id NUMBER PRIMARY KEY,
                              ФИО TEXT,
                              Адрес TEXT
                        );
CREATE TABLE УЧЕНИК     (     Id NUMBER PRIMARY KEY,
                              Класс TEXT,
                              Персона          NUMBER
                              REFERENCES ПЕРСОНА(Id)
                        );
CREATE TABLE УЧИТЕЛЬ    (     Id NUMBER PRIMARY KEY,
                              Должность TEXT,
                              Персона          NUMBER
                              REFERENCES ПЕРСОНА(Id)
                        );
```

Разрешение проблемы множественного наследования

При построении проектной модели на множестве классов могут выполняться обобщения, в том числе и множественные.

С другой стороны, большинство объектно-ориентированых языков программирования реализуют только простое наследование. Следовательно, необходимы преобразования, которые позволят избавиться от множественного наследования при условии сохранения свойств классов.

Непосредственное использование дистрибутивного закона алгебры классов позволяет решить эту задачу.

Начальная диаграмма изображена на Рис. 5, на котором класс C_1 имеет два суперкласса.

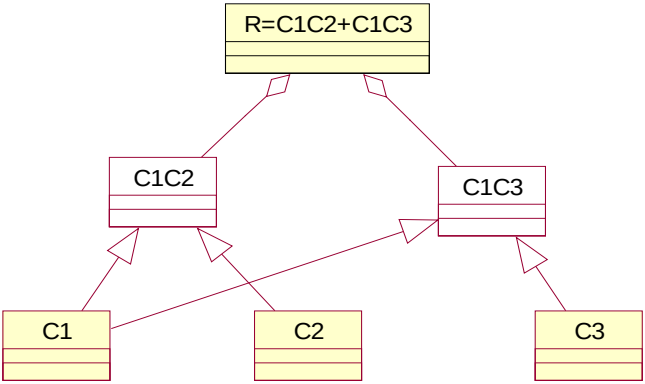


Рис. 5. Пример множественного наследования

Применение дистрибутивного закона позволяет решить эту задачу так, как это показано на Рис. 6.

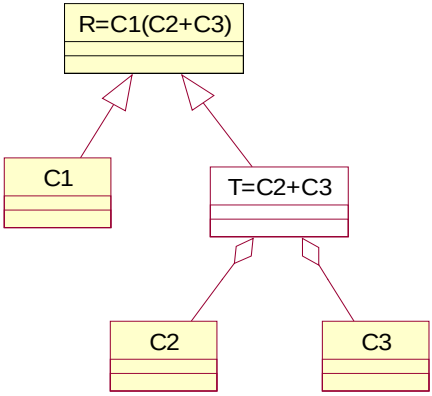


Рис. 6. Решение задачи исключения множественного наследования

Используя модель, показанную на Рис. 6 в языке Object Pascal [9], классы могут быть описаны следующим образом:

```
Type
R = class
…
end;
```

```
C1 = class( R )
…
end;
C2 = class
…
end;
C3 = class
…
end;
T = class( R )
InstanceC2 : C2;
InstanceC3 : C3;
…
end;
```

Следует отметить, что для схема, изображенная на Рис. 5 в языке Object Pascal вообще не может быть описана.

Наследование расширения классов

При проектировании структуры классов часто обнаруживается подобность агрегатных отношений. В контексте обобщения структур интересен ответ на вопрос: являются ли агрегатные отношения свойствами и могут ли они наследоваться.

Рассмотрим диаграмму классов, показанную на Рис. 7.

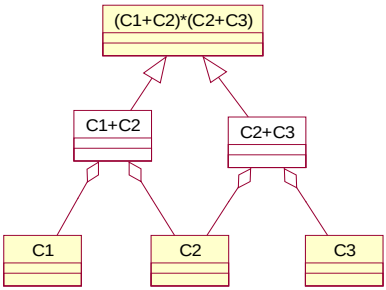


Рис. 7. Пример однотипных расширений.

Как видно на рисунке, расширение с классом C_2 является общим свойством для C_1+C_2 и C_2+C_3.

Применив преобразования (3), (Утверждение 7), (Утверждение 9), (Утверждение 11) и (Утверждение 13), получим

$$(C_1 \overset{E}{-} C_2) \overset{G}{-} (C_2 \overset{E}{-} C_3) = (C_1 \overset{G}{-} C_2) \overset{E}{-} (C_1 \overset{G}{-} C_3) \overset{E}{-} (C_2 \overset{G}{-} C_2) \overset{G}{-} (C_2 \overset{E}{-} C_3) =$$

$$\left[(C_1 \overset{G}{-} C_2) \overset{E}{-} (C_1 \overset{G}{-} C_3) \right] \overset{E}{-} \left[C_2 \overset{G}{-} (C_2 \overset{E}{-} C_3) \right] =$$

$$(C_1 \overset{G}{-} C_2) \overset{E}{-} (C_1 \overset{G}{-} C_3) \overset{E}{-} C_2 = \left[C_2 \overset{E}{-} (C_1 \overset{G}{-} C_2) \right] \overset{E}{-} (C_1 \overset{G}{-} C_3) =$$

$$C_2 \overset{E}{-} (C_1 \overset{G}{-} C_3).$$

На Рис. 8 показан результат.

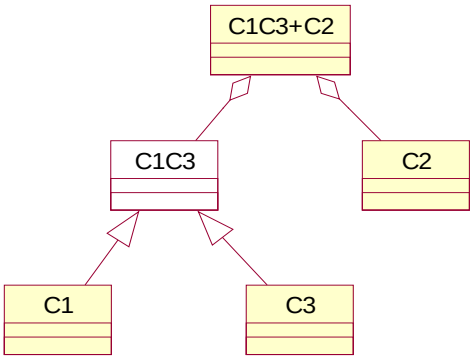


Рис. 8. Наследование общих расширений

Таким образом, расширения могут рассматриваться как особенные свойства (в объектно-ориентированном программировании их называют агрегатными) и они могут наследоваться, как и все остальные.

Выводы

Разработанный математический аппарат, представляющий собой алгебру, определенную на множестве типизированных классов, может использоваться для решения задач, возникающих в процессе проектирования информационных систем и связанных с взаимнооднозначными преобразованиями моделей. Его формальность создает предпосылки для автоматизации решения таких задач на ранних стадиях проектирования.

Список использованной литературы

1. Niki Trigoni, Yong Yao, Alan Demers, Johannes Gehrke, ajmohan Rajaraman. Multi-Query Optimization for Sensor Networks. International Conference on Distributed Computing in Sensor Systems, 2005.

2. Dunren Che, Karl Aberer. A Heuristics-Based Approach to Query Optimization in Structured Document Databases. IEEE Computer Society, 1999.

3. Vijay M. Sarathy, Lawrence V. Saxton, Dirk Van Gucht. An Object Based Algebra for Parallel Query Processing and Optimization. Technical Report TR368, Dept. of Computer Science, Indiana University, December 1992.

4. Nick Taylor. Chaudhuri: An Overview of Query Optimization in Relational Systems. University of Pennsylvania, October 2005.

5. Jef Wijsen, Alexis B`es. On Query Optimization in a Temporal SPC Algebra. Elsevier, 2003.

6. Бузовский О.В. Формальное определение понятий объекта и процесса проектирования.: Весник КПИ. Серия «Автоматика и электроприборостроение», №29: К.,1993.

7. Буч Г. Объектно-ориентированное проектирование с примерами применения: Пер.с анл. – М.:Конрод, 1992.

8. Буч Г., Рамбо Д., Джекобсон А. Язык UML. Руководство пользователя.: Пер.с анл. – М.:ДМК, 2000.

9. Дейт К. Дж. Введение в системы баз данных, 6-е издание: Пер.с анл. – К.;М.;СПб: Издательский дом «Вильямс», 1999.

З М І С Т

Павлов А.А., Аракелян Г.А., Кут В.И. Нахождение весов объектов по матрице парных сравнений, не содержащей цифровой информации 3
Коломоец Г.П. Исследование статических параметров передачи пакетов с данными HTTP 11
Марковский А.П.Мухаммад Мефлех Алиса Абабне., Зюзя А.А., Гаразд В.М. К проблеме защиты операндов модулярного экспоненцирования от их реконструкции анализом динамики потребления мощности 22
Зохре Карим Заде, Мельник А.П., Самковский К.С. Метод построения нелинейного генератора двоичных последовательностуй на сдвиговом регистре 33
Мнацаканов А.В., Кузнецов А.А. К проблеме хеш-адресации без коллизий постоянного массива ключей 42
Марковский А.П., ТурченкоЮ.А., Ковальчук А.В. Повышение эффективности обнаружения ошибок передачи данных в линиях с амплитудно-импульсной модуляцией 52
Марковский А.П., Мулки Ахмед Яссин Ал Бадайнех, Пуя Солеймани Нежадиан. Об одном подходе к повыщению эффективности обнаружения и исправления ощибок передачи данных 62
Ияд Мохд Мадрид Ахмад Шахрури. Использование хеш-памяти для быстрой аутентификации абонентов многопользовательских систем 75
*Корольков І.В. С*корочення довжин маршрутів, отриманих методом алгоритмічної маршрутизації, шляхом модифікації топологічних дерев 86
Авраменко Е.А. Средство редокументирования наследуемого программного обеспечения 98
Артеменко В. А., Богданова Н. В. Управление трафиком с учетом его самоподобия 105
Ролик А.И., Соколовский Р.Л. Распределение мобильных компонентов системы управления информационно-телекоммуникационной системой 113
Виноградов Ю.Н. Способ формального описания функционирования реконфигурируемых вычислительных устройств 125
Салапатов В.І. Реляційна модель даних про внутрішнє уявлення програм 146
Зайченко О.Ю., Кухарєв С.О., Кухарєва О.В. Імітаційне моделювання мереж з технологією MPLS з метою керування різнотипним трафіком 153
Стиренко С.Г., Митин С.В. Параллельные вычисления при распознавании изображений на основе клеточной нейронной сети 161
Зайченко Ю.П., Малихах Есфандиярфард, Заика А.И. Анализ инвестиционного портфеля на основе прогнозирования курсов акций 168
*Пінчук Д.А. Н*ечіткий метод індуктивного моделювання з нечіткими вхідними змінними 180

Мухин В.Е. Средства управления рисками безопасности компьютерных систем 189
Зайченко Ю.П., Гончар М.А. Нечеткие методы кластерного анализа в задачах автоматической классификации в экономике 197
Павлов А.А., Иванова А.А., Зигура Р.А. Метод группового учёта аргументов и анализа иерархий (МГУАиАИ) в задачах принятия решений 205
Павлов О.А., Місюра О.Б., Халус О.А., Беньковський С.Б., Костик Д.Ю., Лисенький Т.М. Система моделювання для дослідження ефективності пдс-алгоритму задачі мінімізації сумарного запізнення виконання завдань 215
Сергієнко А.М. Синтез структур для виконання періодичних алгоритмів з операторами керування 221
Сергієнко А.М., Лепеха В.Л., Лесик Т.М. Спецпроцесори для двовимірного дискретного косинусного перетворення 228
Нестеренко Б.Б., Новотарський М.А. APRO-Мережі для моделювання складних дискретних систем 232
Кузнецов А.В. Смысловая аморфность в грамматических сигнатурах предложений естественного языка 242
Блінова Т. О. Використання кодів Фібоначчі та золотої пропорції в цифрових оптичних каналах передачі інформації 255
Кір'янов О. В. Покращення результатів отриманих поліноміальним багаторядним МГУА для задач з великою кількістю змінних за допомогою додавання зворотних змінних 262
Пустоваров В.І. Супроводження контролю відповідності програм і моделей формальним специфікаціям задач 269
Кулаков А.Ю. Способ повышения эффективности GRID систем на базе виртуальных сетей VPN 280
*Давиденко И.Н.*Способ повышения эффективности процесса маршрутизации в мобильных сетях большой размерности 287
Кулаков Ю.А., Максименко Е.В., Рущак О.А. Повышение уровня безопасности передачи информации в мобильных сетях 297
Кулаков Ю.А., Ланге А.Й. Формирование динамической инфраструктуры крупномасштабных мобильных сетей 305
Бахтари Хедаятоллах. Многоабонентская доставка информации в сетях AD HOC 312
Садег Растгу Мохаммад. Способ управления трафиком в мобильных сетях ATM 318
Жураковський Я.Ю., Жураковська О.С. Застосування математичних моделей та методів у системах MRP II, ERP, APS 324
Болдак А.А., Невдащенко М.В. Математический аппарат формализации моделей, используемых при проектировании информационных систем 332

Відомості про авторів

Авраменко Е.А.	кафедра инженерии программного обеспечения, НАУ
Аракелян Г.А.	каф. АУТС НТУУ ”КПІ”
Артеменко В.А.	канд. техн. наук., доцент, каф. АУТС НТУУ ”КПІ”
Бахтари Хедаятоллах	
Беньковський С.Б.	аспирант кафедры АСОИУ НТУУ ”КПІ”
Блінова Т.О.	канд. техн. наук, кафедра вычислительной техники НТУУ ”КПИ”
Богданова Н.В.	кафедра ААЕ НТУУ ”КПІ”
Болдак А.А.	канд. техн. наук, кафедра обчислювальної техніки НТУУ ”КПІ”
Виноградов Ю.Н.	кафедра вычислительной техники НТУУ ”КПИ”
Гаразд В.М.	студент кафедры вычислительной техники НТУУ ”КПИ”
Гончар М.А.	кафедра ММСА ІПСА НТУУ «КПІ»
Давиденко И.Н.	аспирант кафедры информационных технологий НАУ
Жураковський Я.Ю.	каф. АХВ НТУУ «КПІ»
Жураковська О.С.	канд. техн. наук., каф. АСОІУ НТУУ ”КПІ”, e-mail: oxana@asu.ntu-kpi.kiev.ua
Заика А.И.	кафедра ММСА ІПСА НТУУ «КПІ»
Зайченко О.Ю.	д-р техн. наук, доцент, кафедра ММСА ІПСА НТУУ «КПІ»
Зайченко Ю.П.	д-р техн. наук, профессор, кафедра ММСА ІПСА НТУУ «КПІ»
Зигура Р.А.	студент кафедры АСОИУ НТУУ ”КПІ”
Зохре Карим Заде	канд. техн. наук, стажер кафедры вычислительной техники НТУУ ”КПИ”, e-mail: markovskyy@mail.ru
Зюзя А.А.	аспирант кафедры вычислительной техники НТУУ ”КПИ”
Иванова А.А.	аспирант кафедры АСОИУ НТУУ ”КПІ”
Ияд Мохд Мадрид Ахмад Шахрури	аспирант кафедры вычислительной техники НТУУ ”КПИ”, e-mail: markovskyy@mail.ru
Кір‘янов О.В.	аспірант, кафедра АСОИУ НТУУ ”КПІ”, e-mail: Alexndr.Kiryanov@infopulse.com.ua
Ковальчук А.В.	студент кафедры вычислительной техники НТУУ ”КПИ”
Коломоец Г.П.	канд. физико-математических наук, доцент, ка-

	федра программирования и информационных технологий, классический приватный университет, г.Запорожье, e-mail: kgp@mail.zp.ua
Корольков І.В.	аспирант кафедры компьютерных систем и сетей, Запорожский национальный технический университет , e-mail: korolkov@gmail.com
Костик Д.Ю.	студент кафедры АСОИУ НТУУ ”КПІ”
Кузнецов А.А.	студент кафедры вычислительной техники НТУУ ”КПИ”
Кузнецов А.В.	кафедра обчислювальної техніки НТУУ ”КПІ”
Кулаков А.Ю.	аспирант кафедры вычислительной техники НТУУ ”КПИ”
Кулаков Ю.А.	д-р техн. наук, доцент, кафедри обчислювальної техніки НТУУ ”КПІ”
Кут В.И.	кафедра АСОИУ НТУУ ”КПІ”
Кухарєв С.О.	кафедра ММСА ІПСА НТУУ «КПІ»
Кухарєва О.В.	кафедра ММСА ІПСА НТУУ «КПІ»
Ланге А.Й.	аспирант кафедры вычислительной техники НТУУ ”КПИ”
Лепеха В.Л.	кафедра обчислювальної техніки НТУУ ”КПІ”
Лисецький Т.М.	студент кафедры АСОИУ НТУУ ”КПІ”
Лесик Т.М.	кафедра обчислювальної техніки НТУУ ”КПІ”
Максименко Е.В.	кафедра использование автоматизированных систем и технологий, ИССЗИ НТУУ “КПИ”
Малихах Есфандиярфард	кафедра ММСА ІПСА НТУУ «КПІ»
Марковский А.П.	канд. техн. наук, кафедра вычислительной техники НТУУ ”КПИ”,e-mail: markovskyy@mail.ru
Мельник А.П.	аспирант кафедры вычислительной техники НТУУ ”КПИ”
Місюра О.Б.	канд. техн. наук, ст. научн. сотр., кафедра АСОИУ НТУУ ”КПІ”
Митин С.В.	кафедра использование автоматизированных систем и технологий, ИССЗИ НТУУ “КПИ”
Мнацаканов А.В.	аспирант кафедры вычислительной техники НТУУ ”КПИ”.”,e-mail: markovskyy@mail.ru
Мулки Ахмед Яссин Ал Бадайнех	аспирант кафедры вычислительной техники НТУУ ”КПИ”
Мухаммад Мефлех Алиса Абабне	аспирант кафедры вычислительной техники НТУУ ”КПИ”
Мухин В.Е.	канд. техн. наук, доцент, кафедра вычислительной техники НТУУ ”КПИ”, e-mail: mukhin@comsys.ntu-kpi.kiev.ua,

Невдащенко М.В.	студент кафедры вычислительной техники НТУУ ”КПИ”
Нестеренко Б.Б.	д-р техн. наук, Інститут математики НАН України
Новотарський М.А.	канд. техн. наук, ст. наук. спів., Інститут математики НАН України
Павлов О.А.	д-р техн. наук, професор, кафедра АСОИУ НТУУ «КПІ»
Пінчук Д.А.	аспірант, кафедра ММСА ІПСА НТУУ «КПІ», e-mail: pinchukd@ua.fm
Пустоваров В.І.	канд. техн. наук, кафедра обчислювальної техніки НТУУ ”КПІ”.
Пуя Солеймани Нежадиан	студент кафедры вычислительной техники НТУУ ”КПИ”
Ролик А.И.	канд. техн. наук, кафедра АУТС НТУУ ”КПІ”
Рущак О.А.	кафедра использование автоматизированных систем и технологий, ИССЗИ НТУУ “КПИ”
Садег Растгу Мохаммад	гражданин Иордании
Салапатов В.І.	канд. техн. наук, доцент, кафедра кібернетики, ЧНУ
Самковский К.С.	студент кафедры вычислительной техники НТУУ ”КПИ”
Сергієнко А.М.	канд. техн. наук, ст. наук. спів., кафедра обчислювальної техніки НТУУ ”КПІ”
Соколовский Р.Л.	студент кафедры АУТС НТУУ ”КПИ”
Стіренко С. Г.	канд. техн. наук, кафедра обчислювальної техніки НТУУ ”КПІ”
Турченко Ю.А	аспирант кафедры вычислительной техники НТУУ ”КПИ”
Халус О.А.	кафедра АСОИУ НТУУ «КПІ»