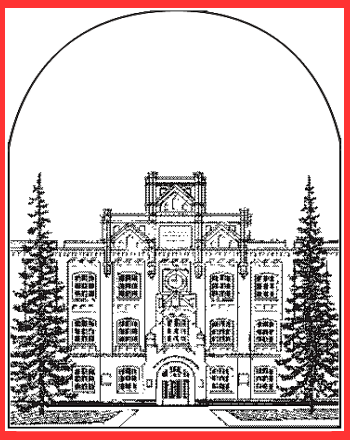
ВІСНИК
НАЦІОНАЛЬНОГО ТЕХНІЧНОГО УНІВЕРСИТЕТУ УКРАЇНИ
“Київський політехнічний інститут”

# Інформатика, управління та обчислювальна техніка



59

2013

Міністерство освіти і науки України
Національний технічний університет України «КПІ»

# ВІСНИК

# НАЦІОНАЛЬНОГО ТЕХНІЧНОГО УНІВЕРСИТЕТА УКРАЇНИ «КПІ»

## Інформатика, управління та обчислювальна техніка

Заснований у 1964 р.
Випуск 59

Київ “ВЕК+” 2013

**УДК 004**

Вісник НТУУ «КПІ». Інформатика, управління та обчислювальна техніка: Зб. наук. пр. – К.: Век+, – 2013. – № 59. – 134 с.

Рекомендований до друку Вченою радою факультету інформатики та обчислювальної техніки, протокол № 5 від 16.12.2013



Описано результати дослідження і створення компонентів обчислювальних й інформаційних систем і комплексів, пристроїв автоматики та передавання даних, систем автоматизації програмування, контролю й діагностики, штучного інтелекту тощо.

Для аспірантів, студентів, фахівців з обчислювальної техніки, систем керування, автоматизації програмування, штучного інтелекту та інших інформаційно-обчислювальних систем.

ISSN 2310-3620

Свідоцтво про державну реєстрацію № 16949-5719 Р від 17.06.2010

Збірник наукових праць українською, англійською та російською мовами

Web-ресурс – http://it-visnyk.kpi.ua

Підп. до друку 16.12.2013. Формат 60×84 1/16. Гарнітура Times. Папір офсетний № 1.
Наклад 150 прим.
Надруковано в ЗАТ “ВІПОЛ”, 03151 м.Київ, вул. Волинська, 60.

# ЗМІСТ

УДК 519.854.2

*ПАВЛОВ А.А.*

# ПРИЗНАКИ ОПТИМАЛЬНОСТИ ДОПУСТИМЫХ РЕШЕНИЙ ТРУДНОРЕШАЕМЫХ ЗАДАЧ КОМБИНАТОРНОЙ ОПТИМИЗАЦИИ

В статье для нескольких видов труднорешаемых одноэтапных задач теории расписаний формулируются признаки оптимальности допустимого решения – теоретические основы построения для этих задач полиномиальной составляющей ПДС-алгоритмов.

In the article for several types of intractable single-stage scheduling problems the optimality signs of a feasible solution are formulated which are the theoretical basis for the construction the polynomial component of the PDC-algorithms for these problems.

## Введение

Большая часть дискретных математических моделей, построенных для анализа, синтеза и функционирования сложных систем, является либо *NP*-полными, либо не проще, чем *NP*-полные задачи. В частности, большинство задач теории расписаний принадлежат этому классу [1]. Анализ трудностей, возникающих при вычислениях на пути создания эффективных методов решения такого рода задач, привел к следующей проблеме: можно ли исключить перебор всех или почти всех вариантов в задаче? Эта проблема исследуется в теории *NP*-полных задач, сформировавшейся на основе работ С. Кука, Р. Карпа, Л. Левина и др. [2]. При принятии гипотезы $P \neq NP$ точных полиномиальных алгоритмов решения этого класса задач не существует [2]. Можно лишь в некоторых классах *NP*-полных задач выделять подклассы (определяемые ограничениями, накладываемыми на параметры комбинаторной задачи), для которых строятся точные полиномиальные алгоритмы [2]. В общем случае эффективными являются лишь приближенные и эвристические алгоритмы.

В [1, 3, 4] изложен новый подход к возможности получения точного решения *NP*-полных задач достаточно большой размерности, сформировавшийся как теория ПДС-алгоритмов. ПДС-алгоритмом называется алгоритм [1], состоящий из полиномиальной и экспоненциальной составляющей, которые могут содержать условия декомпозиции исходной задачи на подзадачи меньшей размерности (обычно полиномиальная составляющая является частью экспоненциальной составляющей). Верхняя оценка сложности полиномиальной составляющей известна. Полиномиальная составляющая порождается логико-аналитическими условиями (*p*-условиями), выполнение которых допустимым решением, полученным в результате реализации полиномиальной составляющей ПДС-алгоритма, определяет его как оптимальное. *p*-условия находятся в результате теоретических исследований соответствующего класса труднорешаемых задач комбинаторной оптимизации. Усредненная эффективность полиномиальной составляющей ПДС-алгоритма находится статистическими методами [1]. Полиномиальная составляющая ПДС-алгоритма синтезируется таким образом, чтобы последовательная процедура конструирования допустимых решений была наиболее эффективной с точки зрения реализации *p*-условий (признаков оптимальности допустимых решений). Иногда экспоненциальная составляющая ПДС-алгоритма заменяется алгоритмом полиномиальной сложности, приводящим к приближенному (субоптимальному) решению [1].

Несмотря на единую методологию построения ПДС-алгоритмов, их конкретная реализация для различных классов труднорешаемых задач комбинаторной оптимизации (в частности, их полиномиальная составляющая) приводит к абсолютно различным алгоритмам [1, 3, 4]. Таким образом, построение ПДС-алгоритма для труднорешаемой задачи комбинаторной оптимизации однозначно определяется возможностью получения для этого класса задач конструктивных (просто проверяемых) признаков оптимальности допустимого решения.

В статье формулируется ряд новых одноэтапных задач календарного планирования, для каждой из которых находятся *p*-условия (признаки оптимальности допустимого решения). Полученные результаты реализуют возможность построения для этих классов комбинаторных задач оптимизации эффективных ПДС-алгоритмов. Подавляющее число исследуемых ниже новых одноэтапных задач календарного

планирования порождены приведенной в [5] многоэтапной сетевой задачей календарного планирования – формальным представлением третьего уровня четырехуровневой модели планирования, принятия решений и оперативного управления в сетевых системах с ограниченными ресурсами [5].

*Примечание.* Приведенные новые одноэтапные задачи календарного планирования не исследовались на предмет того, к какому классу задач ($P$ или не проще, чем $NP$-полные) они относятся. Однако это не суть важно – ПДС-алгоритм может оказаться необходимой вычислительной процедурой и в случаях, когда:

а) неизвестен точный полиномиальный алгоритм решения задачи;

б) точный полиномиальный алгоритм будет построен (обоснование принадлежности задачи классу $P$), либо если будет доказано, что $P = NP$, но ПДС-алгоритм статистически значимо окажется его эффективней. Это возможно в том случае, когда сложность точного полиномиального алгоритма существенно выше сложности полиномиальной составляющей ПДС-алгоритма.

### Задача 1.1

Задано множество заданий $J$, число независимых приборов $m$, для каждого задания $j \in J$ известна длительность выполнения $l_j$, $j = \overline{1,n}$. Все задания имеют общий директивный срок $d$. Процесс обслуживания каждого задания может начаться в любой момент времени, он будет протекать без прерываний до завершения обслуживания задания. Все приборы работают без прерываний с общим моментом запуска.

Необходимо построить допустимое расписание выполнения заданий $j \in J$, у которого момент запуска заданий на выполнение $r$ является максимальным либо суммарное опережение времени завершения выполнения заданий относительно общего директивного срока является минимальным.

Как показано в [6], оптимальное расписание по одному из критериев автоматически является оптимальным расписанием по второму критерию.

В [6] приведен первый признак оптимальности допустимого расписания: на равномерном расписании (время работы всех приборов одинаково) достигается абсолютный оптимум по обоим критериям. В случае невыполнения этого условия оптимальным является допустимое расписание, у которого выполняется второй признак оптимальности: допустимое расписание является оптимальным по обоим критериям, если $\forall C_i$ (где $C_i$, $i = \overline{1,m}$ – суммарное время работы $i$-го прибора) выполняется: для любых $i \neq j$, $i, j = \overline{1,m}$, $|C_i - C_j| = 0 \vee b$, где $b$ – произвольное рациональное число такое, что $\forall i = \overline{1,n}$ числа $l_i / b$ являются целыми; $l_i > 0$, $i = \overline{1,n}$ – длительность $i$-го задания, произвольное рациональное число (пример такого расписания приведен на рис. 1).

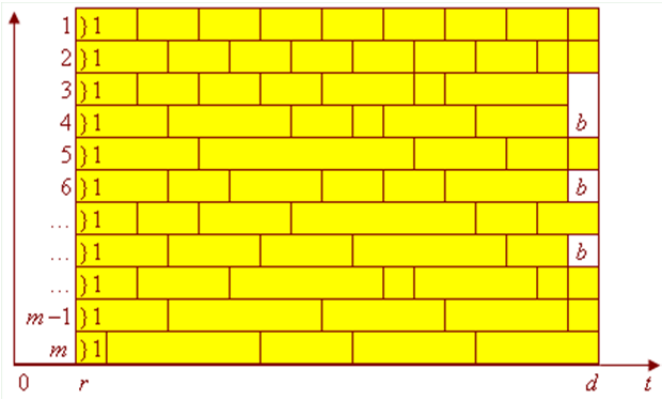



***Рис. 1.***

*Доказательство.* Рассмотрим частный случай такого расписания, когда $l_i$, $i = \overline{1,n}$ – целые числа, а $b$ равняется единице. Это расписание является оптимальным по критерию максимизации момента $r$ запуска заданий на выполнение. Действительно, $r_{\max}$ не удовлетворяет неравенству

$$r_{\max} \geq d - C_{\min}, \quad C_{\min} = \min_{i=\overline{1,m}} C_i. \qquad (1)$$

Действительно, из рис. 1 следует, что для $r \geq d - C_{\min}$ построенное допустимое расписание невозможно, т.к. выполняется неравенство

$$C_{\min} \cdot m < \sum_{i=1}^{n} l_i$$

в силу того, что $l_i$, $C_i$ – целые числа, $r_{max} \leq d - C_{\min} - 1$. Но для $r = d - C_{\min} - 1$ построено допустимое расписание, значит, $r_{max} = d - C_{\min} - 1$.

*Примечание.* В силу того, что $C_i$ – целые числа, $\forall i, j, i \neq j, |C_i - C_j|$ – натуральные числа.

Общий случай. $b > 0$ – рациональное число, $l_i > 0$, $i = \overline{1,n}$ – рациональные числа, $\forall i \; l_i / b$ – натуральные числа. Путем изменения масштаба эту задачу сводим к предыдущей: $\hat{l}_i, i = \overline{1,n}$ – новые длительности, выражаются в натуральных числах, где единица измерения равна $b$. В новых единицах измерения допустимое распи-

сание (рис. 1) сводим к рассматриваемому выше частному случаю.

**Утверждение 1.** Число $b$ является общим делителем чисел $l_i$, $i = \overline{1,n}$, который нацело делится на остальные общие делители $a_j$, $j = \overline{1,k}$, причем $\forall i\ l_i / b$, $\forall i\ l_i / \forall j\ a_j$ являются целыми числами.

*Доказательство*. Если для $\forall i \neq j$ выполняется $C_i \neq C_j$ ($|C_i - C_j| = b > 0$, см. рис. 1), то обязательно $|C_i - C_j| = k_l a_l$, $l = \overline{1,k}$, где $\forall l\ k_l$ – целые числа. Действительно, т.к. $\forall i\ l_i / a_l$, $l = \overline{1,k}$, – целые числа, то $\forall a_l$ число $|C_i - C_j|$ можно представить в виде $k_l a_l$, где $k_l$ – целое число. Отсюда, $b$ нацело делится на $\forall a_l$, $l = \overline{1,k}$.

*Следствие*. $b$ – наибольший общий делитель чисел $l_i$, $i = \overline{1,n}$.

Пусть $k$ – число приборов, у каждого из которых суммарное время работы равно $d - b$ (см. рис. 1), где $d$ – общий директивный срок. Обозначим

$$\hat{l} = \sum_{i=1}^{n} l_i \Big/ b.$$

**Утверждение 2.** а) если $\hat{l} / m$ – целое число, то второй критерий оптимальности не может быть реализован ($k$ не существует);

б) если $\hat{l} / m$ – дробное число, то $k$ – единственное из множества $\{1, \ldots, m-1\}$, при котором $\hat{l} / m - k / m$ – целое число.

*Доказательство.* Из рис. 1 следует, что

$$m \cdot C_{\max} = \sum_{i=1}^{n} l_i - k \cdot b$$

или $d_1 = \hat{l} / m - k / m$, где $d_1 = C_{\max} / b$ – целое число, $\sum_{i=1}^{n} l_i \Big/ b = \hat{l}$ – целое число. Если $\hat{l} / m$ – целое число, то при любом $k = \overline{1,m-1}$ $d_1$ – дробное число, что невозможно. Если $\hat{l} / m$ – дробное число, то только при одном $k \in \{1, \ldots, m-1\}$ $d_1$ является целым числом.

### Задача 1.2

Постановка задачи 1.2 отличается от постановки задачи 1.1 тем, что приборы запускаются в разные моменты времени $t_{1,н} \le t_{2,н} \le \ldots \le t_{m,н}$, $t_{i+1,н} - t_{i,н} = d_i$. $i = \overline{1,m-1}$ – заданные числа. Необходимо найти допустимое расписание, оптимальное по одному из двух критериев:

1) $t_{1,н}$ является максимальным;

2) суммарное опережение директивного срока является минимальным:

$$\min \sum_{i=1}^{n} (d - t_{iк}),$$

где $t_{iк}$ – момент окончания работы $i$-го прибора.

**Утверждение 3.** Если

$$\sum_{i=1}^{n} l_i > \sum_{i=1}^{m} d_i + \sum_{j=2}^{m-1} \left( \sum_{i=1}^{m-1} d_i - \sum_{l=1}^{j-1} d_l \right), \qquad (2)$$

то: а) допустимое расписание, оптимальное по одному из двух критериев, является оптимальным по другому;

б) первый и второй признаки оптимальности допустимого расписания для задачи 1.1 также являются признаками оптимальности допустимого расписания для задачи 1.2.

в) Утверждение 1 для задачи 1.1 справедливо и для задачи 1.2. Утверждение 2 для задачи 1.1 справедливо для задачи 1.2 в случае, когда все числа $d_i$, $i = \overline{1,m-1}$, нацело делятся на $b$.

Утверждение 3 очевидным образом следует из [6], доказательств утверждений 1, 2, приведенных для задачи 1.1, и иллюстрируется рис. 2.

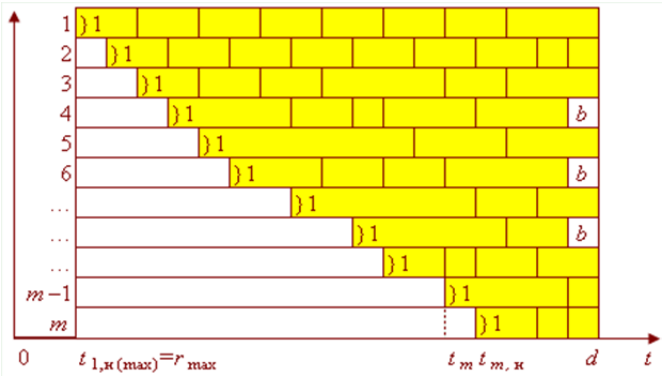



***Рис. 2.***

*Примечание.* Условие (2) необходимо для того, чтобы в любом допустимом расписании всегда выполнялось $\max_i C_i > t_{m,н}$.

### Задача 1.3

Постановка задачи отличается от постановки задачи 1.1 тем, что приборы могут запускаться в произвольные моменты времени. Необходимо найти допустимое расписание, у которого бы выполнялось: $\max t_{i,н}$ является минимально возможным.

**Утверждение 4.** Признаком оптимальности допустимого расписания для задачи 1.3 является выполнение следующих условий:

1) суммарное опережение равно нулю;

2) расписание является равномерным ($t_{i,н} = \text{const}$) либо

$$|t_{i,н} - t_{j,н}| = 0 \vee b, \qquad (3)$$

где $b > 0$ – наибольший общий делитель чисел $l_i$; $l_i / b$ – целые числа, $i = \overline{1,n}$. Число $k$ приборов, у которых $t_{i,\text{н}} < \max\limits_{j=1,m} t_{j,\text{н}}$, является единственным и определяется в утверждении 2 (задача 1.1).

Доказательство утверждения 4 очевидным образом вытекает из [6] и исследования задачи 1.1 и иллюстрируется рис. 3.

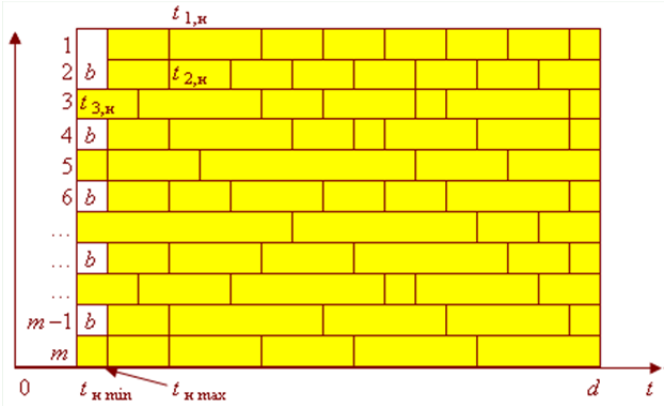



***Рис. 3.***

**Задача 2**

Задано множество заданий $J$, число независимых приборов $m$, для каждого задания $j \in J$ известна длительность выполнения $l_j$, $j = \overline{1,n}$. Процесс обслуживания каждого задания может начаться в любой момент времени, он будет протекать без прерываний до завершения обслуживания заданий. Все приборы работают без прерываний. Заданы ограничения на моменты окончания работы приборов, т.е. должны выполняться условия:

$$t_{i\text{к}} \le d_i, \quad d_1 \ge d_2 \ge \ldots \ge d_m, \quad i = \overline{1,m}$$

где $t_{i\text{к}}$ – момент окончания работы $i$-го прибора.

Директивные сроки для моментов окончания выполнения заданий отсутствуют.

Необходимо найти допустимое расписание, у которого $\min\limits_{i=1,m} t_{i,\text{н}} \to \max$.

**Утверждение 5.** Если

$$\sum_{i=1}^{n} l_i > \sum_{i=1}^{m-1} (d_i - d_{i+1}) + \\ + \sum_{i=2}^{m-1} \left[ \sum_{i=1}^{m-1} (d_i - d_{i+1}) - \sum_{l=1}^{j-1} (d_l - d_{l+1}) \right],$$

то признаком оптимальности допустимого расписания является:

п. 1) $t_{i\text{к}} = d_{i\text{к}}$, $i = \overline{1,m}$;

п. 2) совпадает с п. 2) утверждения 4, причем число $k$ приборов, у которых $t_{i,\text{н}} < \max\limits_{j=1,m} t_{j,\text{н}}$, единственно, если числа $d_i - d_{i+1}$, $i = \overline{1,m-1}$, нацело делятся на $b$.

*Доказательство* утверждения 5 следует из доказательств утверждений 3 и 4 и иллюстрируется рис. 4.

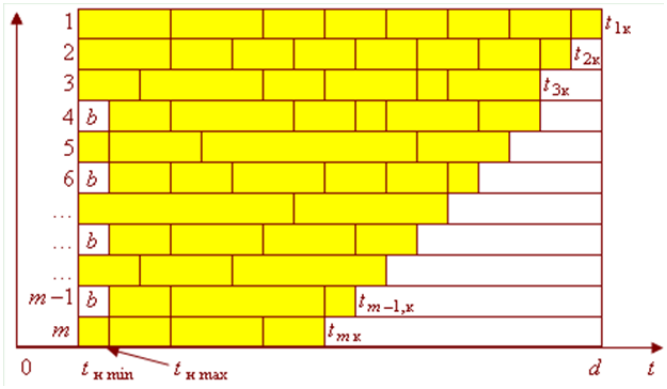



***Рис. 4.***

**Задача 3**

Задано множество независимых заданий $J = \{\overline{1,n}\}$, каждое из которых состоит из одной операции длительности $l_j$, $j = \overline{1,n}$. Заданы директивные сроки выполнения заданий $d_j$, $j = \overline{1,n}$. Прерывания в процессе выполнения задания не допускаются. Имеется один прибор, предназначенный для выполнения заданий. Задания поступают в систему одновременно.

*Примечание*. Нумерация заданий реализует выполнение неравенств: $d_1 \le d_2 \le \ldots \le d_n$.

Необходимо построить допустимое расписание, у которого одновременно выполняются:

1) момент начала выполнения заданий $r$ является максимальным $r_{\max}$ (критерий 1);

2) суммарное опережение выполнения заданий относительно директивных сроков является минимальным (критерий 2).

В [7] показано, что оптимальной последовательностью заданий по критерию 1 является последовательность, упорядоченная по неубыванию значений директивных сроков. В [7] приведен алгоритм А (линейной относительно $n$ сложности) нахождения $r_{\max}$. В [7] также показано, что верхней оценкой отклонения суммарного опережения директивных сроков этого допустимого расписания от допустимого расписания, оптимального по критерию 2, является

$$\sum_{i=1}^{n-1} \left\lfloor \left( d_i - r_{\max} - \sum_{j=1}^{i} l_j \right) \Big/ \min_{j=i+1,n} l_j \right\rfloor \Delta_i, \qquad (4)$$

$$\Delta_i = \max\left(0, \max_{j=i+1,n} l_j - l_i\right),$$

где $\lfloor a \rfloor$ обозначает ближайшее снизу целое от $a$.

**Утверждение 6.** Признаком оптимальности по обоим критериям допустимого расписания является:

п. 1. Последовательность, упорядоченная по неубыванию значений директивных сроков с $r_{\max}$, определенным в [7] (алгоритм А), является также оптимальной по критерию минимизации суммы опережений выполнения заданий относительно директивных сроков, если для него (4) равно нулю.

п. 2. Пусть (4) не равно нулю.

Шаг 1. В допустимом расписании (1, 2, …, $n$) с $r_{\max}$ [7] находим минимальный натуральный индекс $p$, для которого выполняется: в расписании (1, 2, …, $n$) существуют работы с номерами из множества $\{i = \overline{1, p-1}\}$, для которых выполняется неравенство $t_{p\text{к}} \le d_i$, где $t_{p\text{к}}$ – момент окончания выполнения $p$-го задания. Переупорядочиваем эти задания (вместе с заданием $p$) по невозрастанию их длительностей. В [7] показано, что такая перестановка уменьшает суммарное опережение, если есть неупорядоченные задания разной длительности.

Шаг 2. Дальше находим следующий по возрастанию наименьший натуральный индекс $p$, для которого может быть реализована описанная выше процедура, а множество работ, которые будут упорядочены по неубыванию, не совпадает с предыдущим (для дальнейших значений индекса $p$ с предыдущими).

Таких шагов может быть не более, чем $n - 1$. Пусть суммарное результирующее опережение равно $\Delta$. Тогда, если

$$\sum_{i=1}^{n-1} \left\lfloor \left( d_i - r_{\max} - \sum_{j=1}^{i} l_j \right) \Big/ \min_{j=\overline{i+1,n}} l_j \right\rfloor \Delta_i - \Delta = 0, \quad (5)$$

то полученное допустимое расписание является оптимальным по обоим критериям.

*Доказательство* утверждения 6 вытекает из доказательства теорем 4 и 5 [7].

**Следствие.** Если у допустимого расписания, полученного в п. 2 утверждения 6, выражение (5) больше нуля, то полученное допустимое расписание является строго оптимальным по первому критерию, субоптимальным по второму критерию с верхней оценкой отклонения от оптимального значения по второму критерию, которая задается выражением (5).

### Задача 4.1

Имеется $m$ независимых параллельных приборов равной производительности, работающих без прерываний, которые выполняют $n$ работ ($l_i$ – длительность выполнения $i$-й работы, $i = \overline{1,n}$). Работы должны быть выполнены к директивным срокам $d_i$. Моменты запуска станков произвольны. Необходимо построить допустимое расписание, минимизирующее следующий критерий:

$$r_{i_1} = \max\{\min_{i=1,n} r_i\};$$

$$r_{i_l} = \max\{\min_{i} r_i, i = \overline{1,n}, i \ne j_k, k = \overline{1,l-1}\}, l = \overline{2,n}, \quad (6)$$

где $i_1$ – номер прибора, у которого момент запуска в оптимальном расписании самый ранний (он является самым поздним для всех допустимых расписаний); $i_l$, $l = \overline{2,n}$ – номер прибора, у которого момент запуска следующий по величине после приборов $i_k$, $k = \overline{1,l-1}$ (он является самым поздним для всех допустимых расписаний с фиксированным $r_{i_k}, k = \overline{1,l-1}$).

Очевидно, выполняются неравенства $d_i - l_i \ge 0$, $i = \overline{1,n}$. Перенумеруем работы по неубыванию чисел $d_i - l_i$, и пусть при этом выполняются неравенства $d_1 - l_1 < d_2 - l_2 < \ldots < d_n - l_n$.

*Конструирование признака оптимальности №1 допустимого расписания.*

п. 1. На первый прибор первой назначаем работу с индексом 1. $r_1 = d_1 - l_1$. Числу $r_1$ соответствует максимально возможное значение $r_{i_1}$ в (6). Пусть выполняются следующие неравенства: $d_i + l_j > d_j$, $i = \overline{1,m}$, $j = \overline{i+1,m}$. Тогда назначаем на $j$-й прибор, $j = \overline{2,m}$, первой работу с индексом $j$ и моментом запуска прибора $r_j = d_j - l_j$.

**Утверждение 7.** Произвольное допустимое расписание, для которого выполнен п. 1, является оптимальным по критерию (6).

*Доказательство.* Действительно, нарушение в произвольном допустимом расписании алгоритмической процедуры п. 1 немедленно приводит к выполнению:

$$\begin{pmatrix} r_1 \\ \vdots \\ r_m \end{pmatrix} \succcurlyeq \forall \begin{pmatrix} r_{l1} \\ \vdots \\ r_{lm} \end{pmatrix}$$

в соответствии с предпорядком – лексикографическим порядком, где $(r_1 \ \ldots \ r_m)^{\mathrm{T}}$ соответствует произвольное допустимое расписание, у которого выполнен п. 1, а $(r_{l1} \ \ldots \ r_{lm})^{\mathrm{T}}$ – мо-

менты запуска приборов для произвольного допустимого расписания.

*Примечание.* Если в соответствии с п. 1 все работы распределены на меньшем количестве приборов, то соответствующие $r_j$ формально кладутся равными $+\infty$.

*Конструирование признака оптимальности №2 допустимого расписания.*

п. 2. На первый прибор первой назначается работа с индексом 1. $r_1 = d_1 - l_1$. ($d_1 - l_1$ – это максимально большое возможное значение $r_{i_1}$ в (6)). Пусть $k_2 - 1$ – максимальное натуральное число, для которого выполняются неравенства:

$$d_1 + \sum_{j=2}^{l} l_j \le d_l,\ l = \overline{1, k_2 - 1}. \qquad (7)$$

Тогда на первый прибор последовательно назначаются работы с индексами 1, 2, …, $k_2 - 1$. Работа с индексом $k_2$ назначается первой на второй прибор, $r_2 = d_{k_2} - l_{k_2}$. Если неравенство

$$\min\{d_1 + \sum_{j=2}^{k_2-1} l_j + l_{k_2+1},\ d_{k_2} + l_{k_2+1}\} \le d_{k_2+1}$$

не выполняется, тогда на третий прибор первой назначается работа с индексом $k_2 + 1$ в момент времени $r_3 = d_{k_2+1} - l_{k_2+1}$. В этом случае $k_3 = k_2 + 1$. В противном случае должны выполняться неравенства:

$$\min\{d_1 + \sum_{j=2}^{k_2-1} l_j,\ d_{k_2}\} + l_{k_2+1} \le d_{k_2+1}, \qquad (8)$$

$$\max\{d_1 + \sum_{j=2}^{k_2-1} l_j,\ d_{k_2}\} + l_{k_2+1} > d_{k_2+1}. \qquad (9)$$

Тогда работа с индексом $k_2 + 1$ назначается на прибор, которому соответствует минимум в (8). Если неравенство (8) выполнено, а неравенство (9) нарушается, то признак оптимальности №2 допустимого расписания для данной индивидуальной задачи нарушен. Аналогично последовательно назначаются на первый либо второй прибор работы с индексами $\overline{k_2 + 2, k_3 - 1}$ ($k_3$ – максимально возможное натуральное число). При этом, если текущая работа может быть назначена на прибор с меньшим моментом начала обслуживания, то назначение ее на прибор с большим моментом времени начала обслуживания должен приводить к нарушению директивного срока (аналог выполнения неравенств (8), (9)). Работа с индексом $k_3$ первой назначается на третий прибор в момент времени $r_3 = d_{k_3} - l_{k_3}$.

Аналогично происходит дальнейшее последовательное назначение работ с индексами $k_3 + j$ на приборы. Необходимым условием выполнения признака оптимальности №2 является требование, что распределяемая работа может быть назначена только на один прибор (с минимальным временем освобождения) из текущего множества приборов, на которые производится назначение работ. Если ни на один прибор из текущего множества приборов работа не может быть назначена, она назначается первой на следующий прибор в момент времени, равный директивному сроку этой работы минус ее длительность. Распределение работ заканчивается либо когда все работа распределены на $l$ приборов ($l < m$), либо назначением на $m$-й прибор первой $k_m$-й работы $r_m = d_{k_m} - l_{k_m}$. Распределение работ в соответствии с п. 2 завершено.

Пусть $J_1$ – множество работ, которое содержит все работы с индексами из множества $\{\overline{1, k_m}\}$ либо $\{\overline{1, k_l}\}$, если работы распределились на $l$ приборах ($l < m$). Потребуем выполнения следующего условия. Перенумеруем все работы из множества $J_1$ в порядке их назначения на приборы. Тогда для каждой работы с индексом $j$ ($j = \overline{2, k_m \vee k_l}$) должно выполняться

$$d_j - t_{j\text{к}} < l_p,\ p = \overline{l+1, k_m \vee k_l}, \qquad (10)$$

где $t_{j\text{к}}$ – момент окончания выполнения $j$-й работы. Тогда имеет место утверждение:

**Утверждение 8.** Произвольное допустимое расписание, для которого выполнен п. 2 и условие (10), является оптимальным по критерию (6), т.е.

$$\begin{pmatrix} r_1 \\ \vdots \\ r_m \end{pmatrix} \succcurlyeq \forall \begin{pmatrix} r_{l_1} \\ \vdots \\ r_{l_m} \end{pmatrix} \qquad (11)$$

в соответствии с предпорядком – лексикографическим порядком, где $(r_1 \ \dots \ r_m)^{\mathrm{T}}$ соответствует произвольное допустимое расписание, у которого выполнен п. 2 и условие (10), а $(r_{l_1} \ \dots \ r_{l_m})^{\mathrm{T}}$ – моменты запуска приборов для произвольного допустимого расписания.

*Примечание* 1. Если в соответствии с п. 2 загруженными оказались $l$ приборов ($l < m$), то в (11) $r_{l+j}$, $j = \overline{1, m-l}$, формально принимают значения $+\infty$.

Действительно, из логики распределения работ по приборам (п. 2), а также в сочетании с

выполнением условий (10), (8) ÷ (9) и их аналогов на последующих этапах распределения следует, что любое изменение порядка назначения работ (кроме случаев, когда в неравенствах (8), (9) и их аналогах минимум не является единственным) приводит к ухудшению по лексикографическому порядку моментов начала запуска приборов.

*Примечание* 2. Если все работы распределены по $l$ приборам, $l < m$, то в утверждении 7 п. 2 заканчивается назначением первой на $l$-й прибор работы с номером $k_l$.

*Примечание* 3. Признаки оптимальности №1, 2 позволяют конструировать полиномиальную составляющую ПДС-алгоритма для задачи 4.1: конструируется алгоритм полиномиальной сложности для построения допустимого расписания, у которого предварительно работы из множества $J_1$ назначены в соответствии с утверждением 8 (с учетом примечания 2). При этом алгоритм должен учитывать при назначении еще не распределенных работ резервы времени $d_j - t_{j\text{к}}$, $k = \overline{1, k_m \vee k_l}$, оставшиеся после назначения работ из множества $J_1$ в соответствии с утверждением 8. Если полиномиальный алгоритм построил допустимое расписание, то для него признак оптимальности (утверждение 8) выполнен, и это расписание является оптимальным по лексикографическому критерию (6). Если допустимое расписание построить не удалось, то задача 4.1 решается экспоненциальной составляющей ПДС-алгоритма либо приближенным или эвристическим полиномиальным алгоритмом, аппроксимирующим экспоненциальную составляющую ПДС-алгоритма.

*Примечание* 4. Приведем одну из наиболее статистически эффективных стратегий построения приближенного полиномиального алгоритма. Приближенный алгоритм полностью совпадает с полиномиальной составляющей ПДС-алгоритма (примечание 3) за исключением того, что в признаке оптимальности №2 снимается требование, что работа может быть назначена только на один прибор из текущего множества приборов (условия (8), (9) и их обобщение). То есть, текущая работа назначается на прибор с наименьшим временем освобождения, хотя, возможно, ее можно было бы назначить без срыва ее директивного срока первой на другие приборы из текущего множества. Утверждать, что в этом случае построенное допустимое расписание является оптимальным, нельзя, но с учетом выполнения условия (10), а оно приводит к тому, что если текущая работа не назначается на прибор с минимальным временем освобождения, то она может быть назначена только первой на один из текущего множества приборов. А это означает, что количество допустимых вариантов построения начального расписания существенно уменьшается. Очевидно, что при моделировании произвольных индивидуальных задач, для которых выполняется условие (10), логика алгоритма назначать очередную работу на прибор из текущего множества приборов с минимальным временем освобождения практически всегда приводит к выполнению условия (11). Таким образом, построение допустимого расписания с учетом таким образом измененного признака оптимальности №2 (эмпирический признак оптимальности) практически всегда приводит к строгому решению задачи 4.1 по лексикографическому критерию (6).

**Задача 4.2**

Отличается от задачи 4.1 тем, что для работ с индексами из множества $J_1 \in J = \{\overline{1,n}\}$ ограничениями на момент завершения выполнения $t_{l\text{к}}$ $\forall l = \overline{1,n}$ являются не директивные сроки $d_l$ ($t_{l\text{к}} \le d_l$), а ограничения вида $t_{l\text{к}} \in [d_l - \varepsilon_l, d_l]$ $\forall l \in J_1$, см. [5], $\varepsilon_l > 0$.

В частности, если $\varepsilon_l$ мало, такое ограничение соответствует практической реализации формального ограничения – выполнения работы точно в срок ($t_{l\text{к}} = d_l$).

Для задачи 4.2 признак оптимальности №1 остается без изменений (утверждение 7). Признак оптимальности №2, утверждение 8, а также эмпирический признак оптимальности претерпевают следующие изменения. Либо индексы всех работ, назначенных в соответствии с п. 2 на приборы, начиная со второй, не принадлежат $J_1$, либо у этих работ выполняется

$$t_{l\text{к}} \in [d_l - \varepsilon_l, d_l] \; \forall l \in J_1.$$

**Задача 5.1**

Имеется $m$ независимых параллельных приборов разной производительности, работающих без прерываний, которые выполняют $n$ работ ( $l_i^j$ – длительность выполнения $i$-й работы на $j$-м приборе). Работы должны быть выполнены к директивным срокам $d_i$. Моменты запуска приборов произвольны. Необходимо построить

допустимое расписание, оптимальное по критерию (6).

Для задачи 5.1 очевидным образом обобщим признак оптимальности допустимого расписания №1, приведенный для задачи 4.1.

*Конструирование признака оптимальности №1 допустимого расписания.*

п. 1. Рассмотрим следующую монотонно неубывающую последовательность чисел:

$$d_{i_1} - l_{i_1}^{j_1},\ d_{i_2} - l_{i_2}^{j_2},\ ...,\ d_{i_m} - l_{i_m}^{j_m},\ \text{где}$$

$$d_{i_1} - l_{i_1}^{j_1} = \min_i\{(d_i - \min_j l_i^j),\ i = \overline{1,n},\ j = \overline{1,m}\} \quad (15)$$

$$d_{i_p} - l_{i_p}^{j_p} = \min_i\{(d_i - \min_j l_i^j),\ i = \overline{1,n},\ j = \overline{1,m},$$

$$i \notin \{\overline{i_1, i_{p-1}}\},\ j \notin \{\overline{j_1, j_{p-1}}\}\}$$

При этом на каждом приборе достигается только один минимум. Пусть выполняются все неравенства

$$d_{i_l} + l_{i_p}^{j_l} > d_{i_p}\ \ \forall p = \overline{l+1,m},\ l = \overline{1,m-1}.$$

Тогда имеет место утверждение:

**Утверждение 9.** Произвольное допустимое расписание, у которого на прибор $j_l$, $l = \overline{1,m}$, первой назначается работа $i_l$ в момент времени $r_{j_l} = d_{i_l} - l_{i_l}^{j_l}$, является оптимальным расписанием по критерию (6).

### Задача 5.2

Является обобщением задачи 5.1, идентичным обобщению задачи 4.1 на задачу 4.2.

Для задачи 5.2 признак оптимальности допустимого расписания (утверждение 9) остается справедливым. Действительно, все $r_i \in [d_i - \varepsilon_l, d_i]$, $i = \overline{1,m}$, по построению.

### Выводы

В статье приведены основы теории ПДС-алгоритмов. Сделаны новые постановки одноэтапных задач теории расписаний. Найдены для них признаки оптимальности допустимого расписания – основы построения ПДС-алгоритмов.

### Список литературы


1. Згуровский М.З., Павлов А.А. Принятие решений в сетевых системах с ограниченными ресурсами: Монография.– К.: Наукова думка. – 2010. – 573 с.
2. Гери М.Р., Джонсон Д.С. Вычислительные машины и труднорешаемые задачи. – М.: Мир, 1982. – 416 с.
3. Конструктивные полиномиальные алгоритмы решения индивидуальных задач из класса NP. / А.А.Павлов, А.Б.Литвин, Е.Б.Мисюра, Л.А.Павлова, В.И.Родионов, под редакцией А.А.Павлова.– К.: Техника, 1993.– 126 с.
4. Pavlov A., Pavlova L. PDC-algorithms for intractable combinatorial problems. Theory and methodology of design.– Uzhhorod, «Karpatskij region» shelf №15, 1998.– 320 pp.
5. Павлов А.А., Мисюра Е.Б., Лисецкий Т.Н., Сперкач М.О., Халус Е.А. Четырехуровневая модель планирования, принятия решений и оперативного управления в сетевых системах с ограниченными ресурсами // Вісник НТУУ “КПІ”. Серія «Інформатика, управління та обчислювальна техніка». – К.: “ВЕК+”, 2013. – №58 – 14 с.
6. Павлов А.А., Мисюра Е.Б., Сперкач М.О. Исследование свойств задачи календарного планирования выполнения заданий с общим директивным сроком параллельными приборами по разным критериям оптимальности // Вісник НТУУ “КПІ”. Серія «Інформатика, управління та обчислювальна техніка». – К.: “ВЕК+”, 2012. – №57.– С. 15–17.
7. Павлов А.А., Мисюра Е.Б., Халус Е.А. Исследование свойств задачи календарного планирования для одного прибора по критерию минимизации суммарного опережения заданий при условии допустимости расписания // Вісник НТУУ “КПІ”. Серія «Інформатика, управління та обчислювальна техніка». – К.: “ВЕК+”, 2012. – №56.– С. 98–102.

УДК 004.451.24

*СТІРЕНКО С.Г.*

# ОРГАНІЗАЦІЯ ВІДКЛАДЕНИХ ОБЧИСЛЕНЬ В КЛАСТЕРНИХ СИСТЕМАХ

Рассматриваются варианты организации отложенных вычислений в параллельных системах, применяемые в различных инструментальных средствах. Предложены ряд подходов для определения начала передачи данных. Приведен математический формализм и семантика операций, программная реализация которых позволит соответствовать математической модели.

Discusses options for lazy evaluation in parallel systems used in different tools. Proposed several approaches to determine the start of data transmission. The mathematical formalism of this model and semantics of operations on the model are described.

## Відкладення передачі даних

При спробі підвищити ефективність паралельної системи розробники стикаються з проблемою ефективного завантаження її ресурсів та реорганізації обчислювальних процесів. [1] Для виконання такої оптимізації пропонується відкладати початок передачі даних $t_{comm,k}^{\mathrm{i}\to\mathrm{j}}$ на певний проміжок часу, тобто $t_{comm,k}^{\mathrm{i}\to\mathrm{j}} \geq t_{rdy,k}^{\mathrm{i}}$. Це доцільно лише за умови що дані підготовлені раніше ніж запитане перше звернення до них $t_{rdy,k}^{\mathrm{i}} < t_{req,k}^{\mathrm{j}}$. Зокрема, можна відкласти передачу даних до такого моменту часу, у який звільниться якнайменш $m_k$ пам'яті на вузлі $j$, за умови якщо зберігання цього обсягу на вузлі $i$ не матиме негативного впливу на ефективність. Пропонується не відкладати передачу даних, якщо її зберігання на вузлі і має негативний вплив на ефективність

$$M_i'(t) - M_i(t) = m_k \forall: t_{rdy,k}^{i} \leq t \leq t_{comm,k}^{i\to j} + t_{T,k}^{i\to j} \Longrightarrow \begin{cases} \text{якщо } K_E' \geq K_E, \text{ відкладати;} \\ \text{інакше не відкладати.} \end{cases} \quad (1)$$

Вибір моменту часу, у який буде розпочата передача даних, має бути виконаний з ціллю оптимізації виразів з [2] за умови виконання (1). Оскільки на час прийняття рішення $t \leq t_{comm,k}^{\mathrm{i}\to\mathrm{j}}$ в динаміці відомі лише оцінені значення $t_{req,k}^{\mathrm{j}}$ та $t_{T,k}^{\mathrm{i}\to\mathrm{j}}$ (реальні значення можуть відрізнятися через вплив багатьох факторів та будуть відомі лише після завершення передачі даних та початку обчислень над $k$-тим блоком даних, тому їх використання неможливе), необхідно запропонувати евристичні підходи до визначення моменту початку передачі даних.

Розглянемо декілька типових випадків, у яких відкладення передачі може мати позитивний вплив на коефіцієнт ефективності.

Розглянемо виконання обчислень паралельно на двох вузлах обчислювальної системи з локальною пам'яттю. Для спрощення вважаємо, на кожному вузлі виконується лише один процес обчислень, при цьому можуть виконуватись інші системні процеси, можливості вплинути на які в рамках користувацької програми обчислень неможливо. В певному місці обчислень у вузлі 1 генеруються блоки даних (1), (2) та (3), які в подальшому мають бути використані для обчислень, що виконуються на вузлі 2 (рис. 1), причому $t_{rdy,1}^{1} = t_{rdy,2}^{1} = t_{rdy,3}^{1}$. У момент часу $t_{comm,1}^{1\to 2} = t_{rdy,1}^{1}$ починається передача блоку даних (1) даних, яка триває $t_{T,1}^{1\to 2}$, після чого у вузлі 2 можуть бути розпочаті обчислення. Одночасно з цим може виконуватись передача блоку даних (2) $t_{comm,2}^{1\to 2} = t_{rdy,2}^{\mathrm{i}} + t_{T,1}^{1\to 2}$. У певний момент часу

$$t_x : t_{comm,1}^{1\to 2} < t_x < t_{comm,1}^{1\to 2} + t_{T,1}^{1\to 2} \quad (2)$$

закінчуються обчислення над блоком даних (1), які призвели до рішення, що виконувати обчислення над блоком даних (2) не потрібно (наприклад, результат вже достатньо точний). Тому наступними на вузлі 2 будуть виконані обчислення з блоком даних (3). Однак, вони ще не передані на цей вузол і не можуть бути передані, оскільки виконується передача блоку даних (2). Лише після закінчення його передачі можна буде розпочати передачу блоку даних (3) $t_{comm,3}^{1\to 2} = t_{rdy,3}^{1} + t_{T,1}^{1\to 2} + t_{T,2}^{1\to 2}$. Таким чином, час очікування введення та виведення у вузлі 2 складатиме $t_{w,i/o}^{2} = t_{rdy,1}^{i} + \sum_{k=1}^{3} t_{T,k}^{1\to 2} - t_X$.

У випадку ж відкладення передачі даних (рис. 2) до моменту

$$t_{comm,2}^{1\to 2} \geq t_x \geq t_{T,1}^{1\to 2} \quad (3)$$

безпосередня передача даних не розпочнеться. Таким чином, час очікування введення та виведення у вузлі 2 складатиме $t_{w,i/o}^{2\prime} = t_{T,3}^{1\to 2}$.

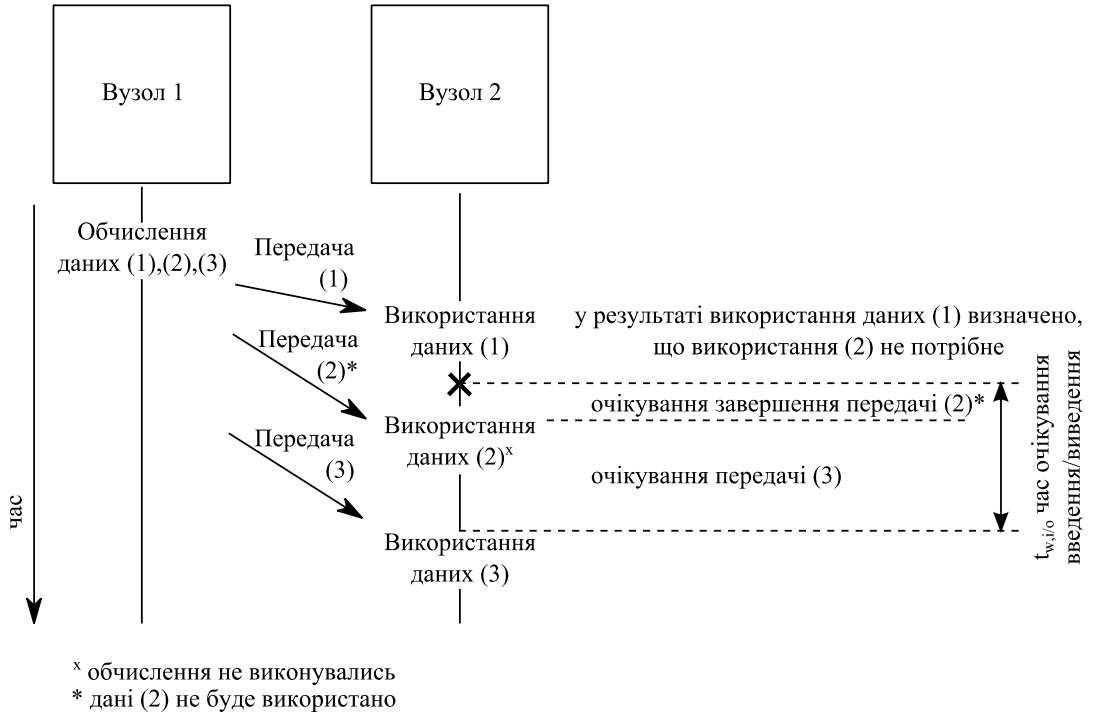



***Рис. 1. Передача даних, які не будуть використані через скасування обчислень***

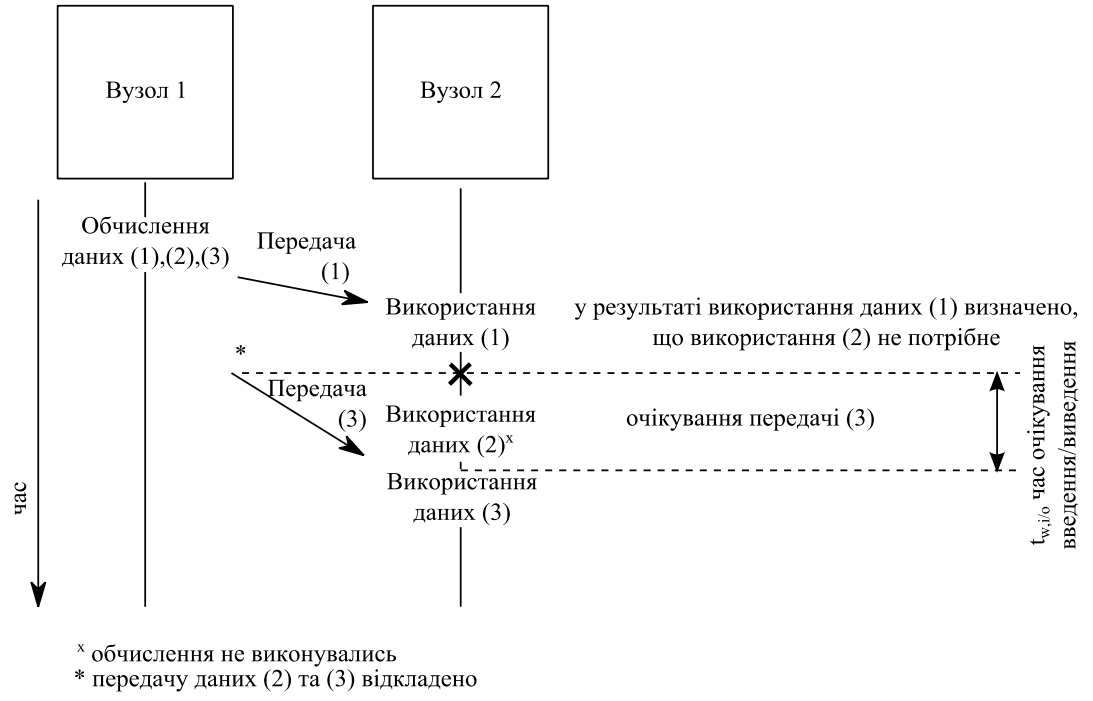



***Рис. 2. Відкладення передачі даних при можливості скасування обчислень***

Різниця часу очікування введення та виведення становитиме

$$t^{2\prime}_{w,i/o} - t^{2}_{w,i/o} = t^{1\to 2}_{T,3} - t^{\text{i}}_{rdy,1} - \sum_{k=1}^{3} t^{1\to 2}_{T,k} + t_x = t_x - t^{\text{i}}_{rdy,1} - t^{1\to 2}_{T,1} - t^{1\to 2}_{T,2}$$

з урахуванням (2) отримаємо

$$t^{2\prime}_{w,i/o} - t^{2}_{w,i/o} \le t^{1\to 2}_{comm,2} + t^{1\to 2}_{T,2} - t^{1\to 2}_{T,1} = t^{1\to 2}_{comm,2} - t^{1\to 2}_{T,1}$$

або

$$-t^{1\to 2}_{comm,2} + t^{1\to 2}_{T,1} \le -t^{2\prime}_{w,i/o} + t^{2}_{w,i/o}$$

звідки за визначенням (3)

$$t^{2\prime}_{w,i/o} - t^{2}_{w,i/o} \ge 0 \qquad (4)$$

Тобто час очікування введення та виведення без відкладення передачі даних *більший* ніж час очікування введення та виведення з відкладен-

ням передачі даних. При цьому, за умови відкладення передачі блоку даних (2) у вузлі 2 не виділяється пам'ять для його зберігання, тобто

$$\begin{cases} M_1'(t) = M_1(t)\ \forall t; \\ M_2'(t) = M_2(t)\ \forall t : t < t_{rdy,1}^1 + t_{T,1}^{1\to 2}; \\ M_2'(t) = M_2\left(t - t_{w,i/o}^2 + t_{w,i/o}^{2\prime}\right) - \\ \quad -m_2 \forall t : t \geq t_{rdy,1} + t_{T,1}^{1\to 2}, \end{cases} \quad (5)$$

що можна скоротити до

$$M_i'(t) - M_i(t) \leq 0\ \forall i\ \in \overline{(1,2)}, t \quad (6)$$

Підставляючи (4), (6) до визначення коефіцієнту ефективності [1], отримаємо

$$K_E'(t) - K_E(t) \geq 0 \quad (7)$$

тобто відкладення передачі даних збільшило коефіцієнт ефективності.

Іншою типовою ситуацією, в якій відкладання передачі даних на певний час може зменшити час очікування введення та виведення, є використання обчислювальної системи, в якій підтримується балансування навантаження. Розглянемо виконання обчислень на системі з трьома вузлами з локальною пам'яттю, які попарно зв'язані між собою. Вузол 1 виконує обчислення з генерації блоків даних (1) та (2) одночасно $t_{rdy,1}^1 = t_{rdy,2}^1$, а на вузлі 2 мають виконуватись обчислення над цими блоками даних послідовно (рис. 3). Передача блоку даних (1) починається відразу по його готовності $t_{comm,1}^{1\to 2} = t_{rdy,1}^1$, після завершення якої розпочинається передача блоку даних (2) $t_{comm,2}^{1\to 2} = t_{rdy,2}^1 + t_{T,1}^{1\to 2}$. В деякий момент часу $t_B$, такий що

$$0 \leq t_B - t_{comm,2}^{1\to 2} = t_{T,2}^{1\to 2} \quad (8)$$

в обчислювальній системі відбувається балансування навантаження, після якого обчислення, з використанням блоку даних (2) переносяться до вузла 3. При цьому, блок даних (2) має бути переданий повторно. Час очікування введення та виведення даних в цьому випадку складатиме

$$t_{w,i/o}^3 = t_{rdy,2}^1 + t_{T,1}^{1\to 2} + t_{T,2}^{1\to 2} + t_{T,2}^{1\to 3} - t_B \quad (9)$$

причому на вузлі 2 було використано пам'ять для тимчасового збереження блоку даних (2), який фактично не використовувався на ньому.

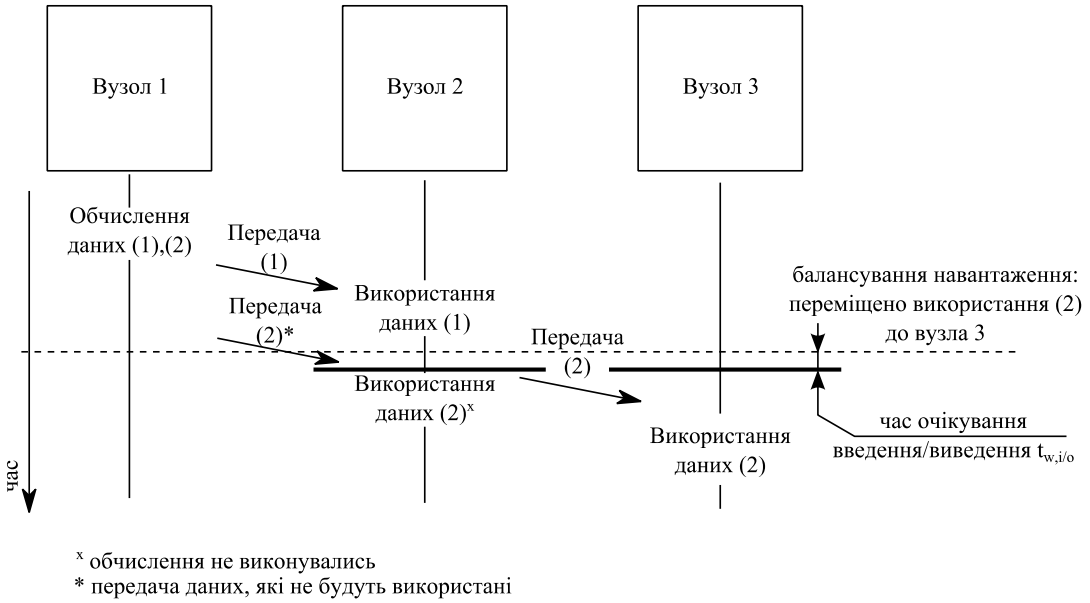



***Рис. 3. Повторна передача даних через балансування навантаження***

У випадку відкладання передачі даних до часу

$$t_{comm,2}^{1\to 2} \geq t_B \quad (10)$$

(рис. 4), тобто балансування навантаження відбудеться до початку передачі даних, можна встановити, що виконувати передачу даних необхідно одразу до вузла (3). В цьому випадку, відсутня необхідність передавати дані на проміжковий вузол (2), та час очікування введення та виведення даних становитиме

$$t_{w,i/o}^{3\prime} = t_{T,2}^{1\to 3} \quad (11)$$

Розрахуємо різницю часу очікування введення та виведення без відкладення обчислень та з таким

$$t_{w,i/o}^3 - t_{w,i/o}^{3\prime} = t_{rdy,2}^1 + t_{T,1}^{1\to 2} + t_{T,2}^{1\to 2} + t_{T,2}^{2\to 3} - t_B - t_{T,2}^{1\to 3}$$

що з урахуванням (8) може бути перетворене на нерівність

$$t_{w,i/o}^3 - t_{w,i/o}^{3\prime} \geq t_{rdy,2}^1 + t_{T,1}^{1\to 2} + t_{T,2}^{1\to 2} + t_{T,2}^{2\to 3} - t_{comm,2}^{1\to 2} - t_{T,2}^{1\to 3}$$

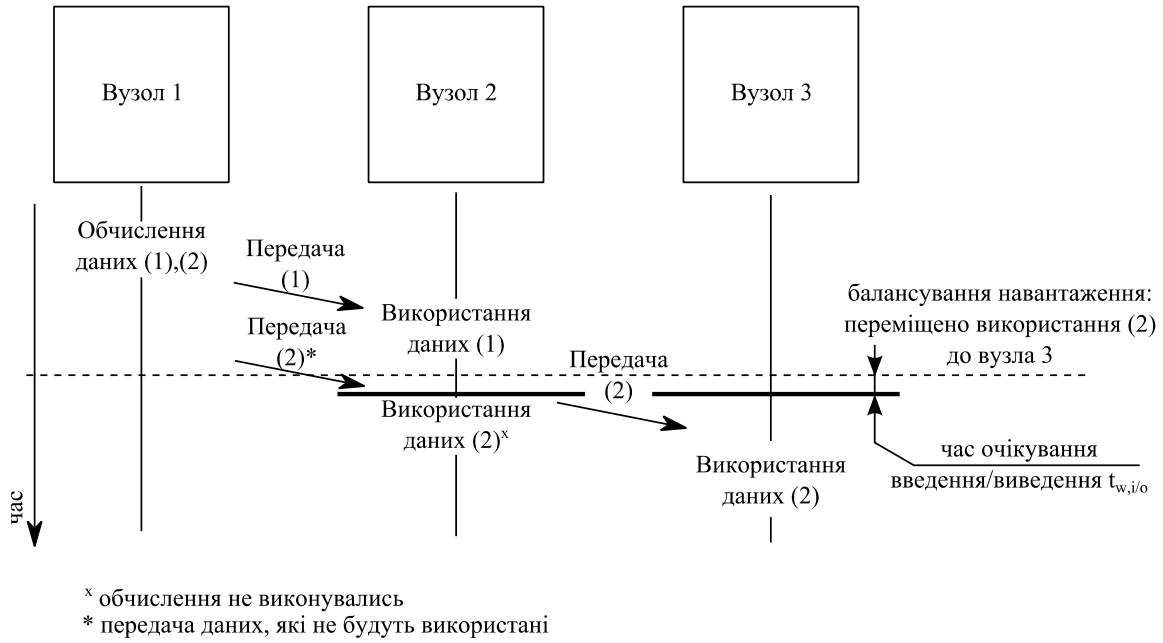



***Рис. 4. Відкладення передачі даних при балансуванні навантаження***

а оскільки за умови негайної передачі даних $t_{comm,2}^{1\to2} = t_{rdy,2}^{1} + t_{T,1}^{1\to2}$ 2, можемо спростити до

$$t_{w,i/o}^{3} - t_{w,i/o}^{3\prime} \geq t_{rdy,2}^{1} + t_{T,1}^{1\to2} + t_{T,2}^{1\to2} + t_{T,2}^{2\to3} - t_{rdy,2}^{1} - t_{T,1}^{1\to2} - t_{T,2}^{1\to3} = = t_{T,2}^{1\to2} + t_{T,2}^{2\to3} + t_{T,2}^{2\to3}$$

Тобто різниця в часі очікування введення та виведення даних залежить від значення виразу $(t_{T,2}^{1\to2} + t_{T,2}^{2\to3}) - t_{T,2}^{1\to3}$, а саме від різниці часу, що необхідний на передачу блоку даних (2) з вузла 1 до вузла 3 через проміжковий вузол 2 або напряму. Причому час передачі розглядається на програмному рівні, тобто не обов'язково залежить від наявності та параметрів фізичних зв'язків між вузлами. Обчислювальна система намагатиметься обрати такий шлях та спосіб передачі даних, який би мінімізував час виконання передачі [2], тому для більшості випадків за умови застосування відомих стратегій організації обчислювального процесу $(t_{T,2}^{1\to2} + t_{T,2}^{2\to3}) - t_{T,2}^{1\to3} \geq 0$, звідки випливає

$$t_{w,i/o}^{3} - t_{w,i/o}^{3\prime} \geq 0 \qquad (12)$$

Таким чином, завдяки відкладенню передачі даних існує можливість зменшити час очікування введення та виведення, який матиме вплив на коефіцієнт ефективності аналогічний розглянутому в (4), (7). Тим не менш у ряді випадків за умови застосування вкрай неоптимальної стратегії вибору шляху передачі даних між вузлами неоднорідної системи, може спостерігатися зворотній ефект. Для запобігання цьому необхідно вести статистику під час виконання та враховувати її при прийнятті рішення про відкладення передачі даних.

Окрім цих типових випадків, які були показані у спрощеному вигляді у порівнянні з реальними обчисленнями на системах з локальною пам'яттю, існує ще багато більш складних випадків. Зокрема випадки, у яких відкладення передачі даних може зменшити обсяги пам'яті, що використовується в певному вузлі, що також може впливати на коефіцієнт ефективності при виконанні обчислень.

## Висновки

При застосуванні низькорівневих моделей програмування, розв'язання цієї задачі покладається на користувача обчислювальної системи, який завдяки знанню особливостей задачі може забезпечити коректну організацію передачі даних, тобто таку, за якої дані доступні на необхідному вузлі перед початком обчислень. Однак, така реалізація не завжди матиме достатню ефективність через невикористання можливостей перевпорядкування передачі, довільного порядку передачі та інших. Тому в сучасних системах застосовуються моделі програмування з більш високим рівнем абстракцій, зокрема такі, що мають переносити дані користувача за описаними ним правилами автоматично на вузол, на якому будуть виконуватися обчислення з використанням цих даних. Використання таких моделей дозволяє виконувати ряд оптимізації, в тому числі врахування особливостей гетерогенної системи, в процесі виконання обчислень з метою збільшення їх ефективності.

Організація передачі даних може впливати на коефіцієнт ефективності через два фактори: час очікування та виконання введення та виведення даних за мережею, яка зазвичай використовується для зв'язку вузлів системи з локальною пам'яттю, та обсягу використаної пам'яті. Останнє впливає на коефіцієнт ефективності опосередковано: так обсяг використаної пам'яті впливає на кількість кеш-промахів, яка в свою чергу визначає час очікування підкачки даних з диску в основну пам'ять, причому останній входить до часу очікування, у який не виконуються обчислення даної задачі, а лише у найкращому випадку деякі системні задачі завдяки принципу розподілу часу.

Запропоновано застосувати технологію відкладення обчислень до передачі даних в системах з локальною пам'яттю, а саме відкладати момент початку передачі даних між вузлами від моменту готовності даних, якщо дані підготовані раніше ніж використовуються перший раз. У зворотньому випадку, відкладення даних недоцільне, оскільки лише збільшить час очікування введення та виведення, під час якого не виконуються обчислення. Відкладення обчислень особливо корисне у випадку наявності в обчислювальній системі механізмів балансування навантаження, які можуть призвести до надлишкового копіювання даних, якщо не було однозначно визначено на якому вузлі будуть виконуватись обчислення. Цей підхід найкраще застосовувати в процесі виконання обчислень, оскільки він може дозволити не передавати дані, якщо в процесі обчислень прийнято рішення про невиконання певної частини обчислень або відсутність необхідності у використанні ряду даних завдяки специфічному для задачі аналізу, наприклад якщо необхідну точність обчислень вже досягнуто.

З іншого боку, відкладення передачі даних передбачає необхідність зберігання їх у вузлі-джерелі впродовж певного часу, що збільшить на цей час обсяг використаної пам'яті в цьому вузлі. Збільшення обсягу використаної пам'яті може зменшити коефіцієнт ефективності та нівелювати ефект, досягнутий завдяки відкладенню передачі даних. Тому необхідно дослідити взаємний вплив цих параметрів та запропонувати методи, які враховують обидва, для розрахунку часу початку передачі даних, такого що мінімізує час очікування передачі даних та середній обсяг використаної пам'яті за час передачі даних.

Запропоновано ряд підходів до визначення часу початку передачі даних. Серед них слід відзначити два підходи, які є відповідно нижнім і верхнім обмеженням на доцільні значення часу початку передачі даних: передавати дані відразу по готовності, тобто фактично на виконувати відкладення, та передавати дані за запитом, тобто відкладати передачу на максимально можливий термін. Обмеження знизу є гарантією збереження коректності обчислень, оскільки дані будуть підготовані перед їх пересилкою для використання, а вихід за обмеження зверху через необхідність додаткового очікування після запиту на використання даних вносить у час виконання затримку, під час якої обчислення не виконуються, що в результаті знижує коефіцієнт ефективності. Запропоновано також два підходи до більш гнучкого обчислення часу початку передачі даних, один з яких базується на статистичній обробці попередніх значень часу очікування введення та виведення та обсягів пам'яті під час розв'язання даної задачі, а інший – на ймовірнісній оцінці часу першого запиту даних та часу, необхідного для передачі цих даних між вузлами. В рамках кожного з підходів може бути запропоновано декілька різних способів визначення невідомих величин, а саме можна змінювати алгоритми статистичної обробки або ймовірнісні моделі прогнозування. Для останніх найбільш простою є використання моделі програмування, що базується на розбитті на підзадачі, яка якісно відображає задачі з великою кількістю даних для обробки.

**Список посилань:**


1. Стиренко, С.Г. Модель организации вычислений в распределенной системе / С.Г. Стиренко, А.И. Зиненко, Д.В. Грибенко // Вісник НТУУ «КПІ». Інформатика, управління та обчислювальна техніка. – 2012. –Т. 57. – С. 101–109.
2. Стіренко С.Г. Модель технології програмування паралельних систем. / С.Г. Стиренко // Вісник НТУУ «КПІ». Інформатика, управління та обчислювальна техніка. – 2013. –Т. 58. – С. 158–164.
3. Симоненко, В. П. Организация вычислительных процессов в ЭВМ, комплексах, сетях и системах / В. П. Симоненко. – К. : ВЕК+, 1997. –304 с.

УДК 004.056.5

*МАРКОВСЬКИЙ О.П.,*
*АБУ-УСБАХ О.Н.,*
*ФЕДОРЕЧКО О.І.,*
*ПЕРЕСЕНЧУК Д.В.*

# МЕТОД КОРЕКЦІЇ БЛОКІВ ДАНИХ ПОШКОДЖЕНИХ ПРИ ПЕРЕДАЧІ В БЕЗДРОТОВИХ КАНАЛАХ

В статті пропонується нова технологія корекції блоків даних при передачі в бездротових лініях глобальних комп'ютерних мереж. Розроблений метод має за основу відновлення пошкодженого при передачі блоку в цілому на противагу корекції окремих символів. Досягнутий ефект зумовлений розділенням процесів виявлення та виправлення помилок. В основі запропонованого методу покладено прості математичні операції, що забезпечує ефективну апаратну реалізацію на програмованих логічних матрицях.

This paper proposes an innovative technique for correcting of erroneous data blocks in transmitted over the wireless channels of global computer network. Proposed method is based on recovering of whole data damaged blocks instead of individual symbols correction. Efficiency is achieved by separating the error detection from the correction process and using different codes for each case. The proposed technique is based on simple mathematical operations and is suitable for implementation in FPGA devices.

## Вступ

Динамічний розвиток засобів і технологій передачі даних значною мірою визначає загальний прогрес систем комп'ютерної обробки інформації.

Тенденцією розвитку засобів передачі цифрових даних є невпинне зростання питомої ваги бездротових технологій. Суттєвою вадою останніх є вразливість до впливу зовнішніх завад. Відповідно, при використанні бездротових технологій передачі даних зростає кількість помилок, які здебільшого мають характер "пачки помилок" (burst errors) - групи суміжних спотворених символів [1].

З іншого боку, значне зростання об'ємів даних, що передаються по радіоканалах має наслідком збільшення інтенсивності зовнішніх електромагнітних полів і, відповідно, зростання впливу зовнішніх завад.

Розвиток можливостей інтегральних технологій стимулює постійне підвищення рівня спектрального ущільнення тобто кількості бітів, що передаються одним сигналом, при передачі даних та підвищення темпу слідування несучих сигналів. Це призводить до помітного збільшення негативного впливу на надійність передачі явищ, пов'язаних з інтерференцією несучих сигналів [2].

Разом з тим, триває процес розширення використання комп'ютерних систем з розподіленими компонентами в усіх сферах людської діяльності, в тому числі тих, які пов'язані з техногенними чи іншими ризиками. Для таких застосувань потрібно забезпечити високий рівень надійності передачі даних.

Наведенні чинники визначають нагальну необхідність постійного вдосконалення засобів виявлення та виправлення помилок передачі даних, в тому числі в глобальних мережах Інтернет.

## Аналіз існуючих технологій корекції передачі даних

В рамках дослідження розглядається ситуація передачі по радіоканалу з використанням спектральної модуляції по мережі інформаційної посилки $D$, що складається $m$ блоків $B_1,B_2,...,B_m$, кожен з яких містить по $n$ символів.

Досліджується типова для практики ситуація виникнення помилок в результаті дії зовнішніх завад, результатом дії яких є виникнення "пачки помилок". В більшості теоретичних моделей [1] виникнення помилок передачі цифрових даних вважається, що кількість виникаючих "пачок" підпорядкована біноміальному закону розподілу, в той час як закон розподілу довжини "пачки" має більш складний характер [3].

Традиційними для корекції помилок передачі даних є дві технології

- виявлення помилок, що виникають при передачі блоку, за допомогою спеціальних кодів з подальшою повторною передачею при наявності помилки;
- використання корегуючих кодів, які дозволяють прямо виправити обмежену кількість помилок передачі даних за рахунок конт-

рольних символів, що передаються разом з блоком.

Вважається [4], що перша технологія більш ефективна при відносно малій інтенсивності виникнення помилок. Відповідно, основною сферою її застосування є кабельні канали передачі цифрових даних. Її перевагами є значно менша в порівнянні з корегуючими кодами кількість контрольних розрядів, а також те, що вона дозволяє виправляти помилки будь-якої кратності, в тому числі "пачки" помилок довільної довжини.

Суттєвими недоліками повторної передачі вважаються [4]:
- необхідність використання зворотного зв'язку при передачі даних, що суттєво уповільнює транзакції в глобальній мережі;
- можливіть виникнення помилки при повторній передачі даних;
- необхідність буферизації даних, що передаються по мережі до отримання підтвердження коректності передачі.

Корегуючі коди [5] вважаються більш ефективним засобом виправлення помилок при відносно великій інтенсивності їх виникнення. Для більшості корегуючих кодів, і зокрема для найбільш поширеного в мережових технологіях коду Ріда-Соломона, кількість контрольних символів вдвічі перевищує кратність $k$ спотворених при передачі символів, що можуть бути скореговані.

Перевагами корегуючих кодів типу Ріда-Соломона є те, що вони дозволяють локалізувати та виправити задану кількість пошкоджених символів блоку з використанням теоретично мінімальної кількості контрольних розрядів.

Суттєвим недоліком згаданих корегуючих кодів є висока обчислювальна складність процесів локалізації і корекції спотворених при передачі символів, пов'язана з необхідністю розв'язання систем лінійних символьних рівнянь в арифметиці полів Галуа [4]. Враховуючи, що архітектура сучасних процесорів не розрахована на виконання операцій на полях Галуа, для швидкої корекції помилок з використанням корегуючих кодів здебільшого використовуються спеціальні апаратні засоби. Обчислювальна складність реалізації корекції залежить від кількості символів, які здатен корегувати код.

Відповідно, основною проблемою практичного застосування корегуючих кодів в сучасних умовах є варіювання в широких межах часового проміжку дії зовнішньої завади і, відповідно числа спотворених символів в "пачці". Для забезпечення високої надійності корекції "пачки помилок" існує дві можливості:
- збільшення кількості контрольних символів на блок, що має наслідком помітне уповільнення процесу корекції і зростання рівня інформаційної надлишковості [5];
- застосування технології створення віртуальних блоків шляхом перемішування символів різних фізичних блоків і обчислення контрольних символів по віртуальним блокам. Це надає змогу статистично вирівняти кількість спотворених символів в кожному з блоків, але, з одного боку не гарантує виправлення заданої кількості помилок, а з другого - суттєво уповільнює процеси кодування та корекції даних [1].

В умовах збільшення інтенсивності виникнення помилок передачі даних в каналах глобальних мереж, існуючі корегуючі коди не дозволяють повною мірою вирішити проблему забезпечення безпомилкової передачі цифрових даних.

Це вимагає розробки нових засобів виправлення помилок, які в більшій мірі враховують особливості їх виникнення в бездротових мережах під дією зовнішніх завад та особливості передачі блоків в глобальних комп'ютерних мережах.

Одним з можливих підходів при цьому є комбіноване використання кодів для виявлення помилок в блоках та кодів для виправлення цілих блоків. Оскільки основний об'єм витрат обчислювальних ресурсів при використанні кодів Ріда-Соломона приходиться на локалізацію спотворених символів, то такий підхід дозволить суттєво спростити та прискорити процес корекції.

Ціллю досліджень є створення ефективного методу виправлення блоків цифрових даних при їх передачі по бездротовим каналам глобальних комп'ютерних мереж.

### Метод корекції блоків даних

Сутність розробленого методу корекції помилок передачі символів в каналах зі спектральною модуляцією полягає в комбінованому застосуванні двох технологій: виявлення помилок передачі блоку за допомогою кодів CRC [4] та передача $h$ контрольних блоків, за допомогою яких виправляється будь-які $h$ з інформаційних блоків. Задля спрощення викладу, в статті розкрито запропоновану технологію відновлення пошкоджених при передачі по безд-

ротовим каналам для $h$=4, тобто в ситуації, коли кількість спотворених блоків не перевищує 4-х.

Як зазначалося вище, інформаційна посилка $D$ передається у вигляді $m$ блоків $B_1,B_2,...,B_m$, кожен з яких складається з $n$ символів. Тобто кожен $j$-тий ($j$=1,...,$m$) блок $B_j$ може бути представлений як набір символів $B_j = \{b_{j1},b_{j2},...,b_{jn}\}$.

Передача кожного $j$-го блоку $B_j$ контролюється кодом виявлення помилок передачі, зокрема CRC або WCS [3], який використовує $k$ контрольних розрядів, що передаються разом з блоком. Додатково передаються також чотирьох контрольних блоків $C_1,C_2,...,C_4$, що дозволяє виправити до 4-х спотворених при передачі блоків інформаційної посилки.

Контрольні блоки пропонується формувати на передавачі наступним чином.

Контрольні блоки $C_1$ та $C_2$ складаються з $n$+$m$-1 символів та формуються наступним чином. Кожен $i$-тий символ $c_{1i}$ , $i$=1,...,$n$+$m$-1 першого контрольного блоку $C_1$ пропонується формувати як суму за модулем певної підмножини символів інформаційних блоків наступним чином:

$$\begin{aligned}
&\forall i \in \{1,..,m\} : c_{1i} = \bigoplus_{q=1}^{i} b_{i-q+1,q} \\
&\forall i \in \{m+1,..,n\} : c_{1i} = \bigoplus_{q=i-m+1}^{i} b_{i-q+1,q} \\
&\forall i > n : c_{1i} = \bigoplus_{q=i-m+1}^{n} b_{i-q+1,q}
\end{aligned} \quad (1)$$

Кожен $i$-тий символ $c_{2i}$ , $i$=1,...,$n$+$m$-1 другого контрольного блоку $C_2$ пропонується формувати як суму за модулем певної підмножини символів інформаційних блоків наступним чином:

$$\begin{aligned}
&\forall i \in \{1,..,m\} : c_{2i} = \bigoplus_{q=1}^{i} b_{m-i+q,q} \\
&\forall i \in \{m+1,..,n\} : c_{2i} = \bigoplus_{q=i-m+1}^{i} b_{m-i+q,q} \\
&\forall i > n : c_{2i} = \bigoplus_{q=i-m+1}^{n} b_{m-i+q,q}
\end{aligned} \quad (2)$$

Контрольні блоки $C_3$ та $C_4$ складаються з $n$+2·($m$-1) символів. Формування кожного з символів третього блоку $C_3$ виконується згідно виразу:

$$\begin{aligned}
&\forall 1 \le i \le 2\cdot m-3 : c_{3i} = \bigoplus_{q=0}^{(i-1)\,div\,2} b_{q+1,i-2\cdot q}, \\
&\forall 2\cdot m-2 \le n : c_{3i} = \bigoplus_{q=0}^{m-1} b_{q+1,i-2\cdot q}, \\
&\forall n < i \le n+2\cdot m-2 : c_{3i} = \bigoplus_{q=(i-n-1)\,div\,2+1}^{m-1} b_{q+1,i-2\cdot q}
\end{aligned} \quad (3)$$

В формулах (3) позначення $\alpha$ div $\beta$ відповідає операції цілочисельного ділення (тобто з відкиданням залишку) $\alpha$ на $\beta$.

Кожен $i$-тий символ $c_{3i}$ , $i$=1,...,$n$+2·($m$-1) четвертого контрольного блоку $C_4$ пропонується формувати як суму за модулем певної підмножини символів інформаційних блоків згідно з наступною формулою:

$$\begin{aligned}
&\forall 1 \le i \le 2\cdot m-2 : c_{4i} = \bigoplus_{q=0}^{(i-1)\,div\,2} b_{m-q,i-2\cdot q}, \\
&\forall 2\cdot m-1 \le i \le n : c_{4i} = \bigoplus_{q=0}^{m-1} b_{m-q,i-2\cdot q}, \\
&\forall n < i \le n+2\cdot m-2 : c_{4i} = \bigoplus_{q=(i-n-1)\,div\,2+1}^{m-1} b_{m-q,i-2\cdot q}
\end{aligned} \quad (4)$$

Сформовані описаним способом на передавачеві контрольні блоки $C_1,...,C_4$ доповнюються кодами для виявлення помилок передачі.

Нехай, в процесі передачі були пошкоджені чотири інформаційні блоки з порядковими номерами $v$,$w$,$u$,$z$, причому $v$<$w$<$u$<$z$. Відповідно, постає задача відновлення символів блоків $B_v=\{b_{v1},b_{v2},...,b_{vn}\}$, $B_w=\{b_{w1},b_{w2},...,b_{wn}\}$, $B_u=\{b_{u1},b_{u2},...,b_{un}\}$ та $B_z=\{b_{z1},b_{z2},...,b_{zn}\}$.

Метод передбачає відновлення зазначених символів в наступному порядку:

1. Відновлюються символи $b_{v1},b_{v2},...,b_{v,w-v}$ за допомогою контрольних символів $c_{3,2\cdot v-1}$, $c_{3,2\cdot v},...,c_{3,w+v-2}$ відповідно. Кожен із вказаних контрольних символів згідно (3) може бути представлений у вигляді

$$\begin{aligned}
&c_{3,2\cdot v-1} = \bigoplus_{q=0}^{v-1} b_{q+1,2\cdot v-1-2\cdot q} = b_{1,2\cdot v-1} \oplus b_{2,2\cdot v-3} \oplus ... \oplus b_{v,1} \\
&c_{3,2\cdot v} = \bigoplus_{q=0}^{v} b_{q+1,2\cdot v-2\cdot q} = b_{1,2\cdot v} \oplus b_{2,2\cdot v-2} \oplus ... \oplus b_{v,2} \\
&\ldots\ldots\ldots\ldots\ldots \\
&c_{3,w+v-2} = \bigoplus_{q=0}^{(w+v-2)\,div\,2} b_{q+1,w+v-1-2\cdot q} = b_{1,w+v-1} \oplus b_{2,w+v-3}, .. \\
&\qquad \oplus b_{v,w-v} \oplus ... \oplus b_{(w+v-2)\,div\,2+1,\varsigma}
\end{aligned} \quad (5)$$

де $\zeta$=1, якщо $w$+$v$-парне і $\zeta$=2, якщо $w$+$v$ - непарне. Кожне з наведених рівнянь (5) містить лише одну невідому компоненту: $b_{v1}$ для першого рівняння, $b_{v2}$ -для другого рівняння, $b_{v,w-v}$ для останнього рівняння. Всі інші компоненти рівнянь належать до інформаційних блоків з номерами меншими $v$ та блоків з номерами меншими за $w$. Виходячи з цього, символи $b_{v1},b_{v2},...,b_{v,w-v}$ відновлюються у вигляді:

$$\begin{aligned}
&b_{v,1} = c_{3,2\cdot v-1} \oplus b_{1,2\cdot v-1} \oplus ... \oplus b_{v-1,3}\\
&b_{v,2} = c_{3,2\cdot v} \oplus b_{1,2\cdot v} \oplus ... \oplus b_{v-1,4}\\
&\cdots\cdots\cdots\cdots\\
&b_{v,w-v} = c_{3,w+v-21} \oplus b_{1,w+v-1} \oplus ...\\
&\oplus b_{v-1,w-v-2} \oplus b_{v+1,w-v+2} \oplus ...\\
&\oplus b_{(w+v-2)div2+1,\varsigma}
\end{aligned} \qquad (6)$$

2. Відновлюються $z$-$u$ символів $b_{z1},b_{z2},...,b_{z,z-u}$ з використанням символів 4-го контрольного блоку: $c_{4,2\cdot(m-z)+1}$, $c_{4,2\cdot(m-z)+2},...,c_{4,2\cdot(m-z)+z-u}$ відповідно. Кожен із вказаних контрольних символів згідно (4) може бути представлений у вигляді:

$$\begin{aligned}
&c_{4,2\cdot(m-z)+1} = \bigoplus_{q=0}^{m-z} b_{m-q,2\cdot(m-z-q)+1} =\\
&= b_{m,2\cdot(m-z)+1} \oplus b_{m-1,2\cdot(m-z)-1} \oplus ... \oplus b_{z+1,3} \oplus b_{z,1}\\
&c_{4,2\cdot(m-z)+2} = \bigoplus_{q=0}^{m-z+1} b_{m-q,2\cdot(m-z-q)+2} =\\
&= b_{m,2\cdot(m-z)+2} \oplus b_{m,2\cdot(m-z)} \oplus ... \oplus b_{z+1,4} \oplus b_{z,2}\\
&\cdots\cdots\cdots\cdots\\
&c_{4,2\cdot(m-z)+z-u} = \bigoplus_{q=0}^{m-z+(z-u)div2} b_{m-q,2\cdot(m-q)-z-u} =\\
&= b_{m,2\cdot m-z-u} \oplus ... \oplus b_{z,z-u} \oplus ... \oplus b_{z-(z-u)div2,\zeta}
\end{aligned} \qquad (7)$$

де $\omega$=1, якщо $z$-$u$-парне і $\omega$=2, якщо $z$-$u$ - непарне. Кожне з наведених рівнянь (7) містить лише одну невідому компоненту: $b_{z,1}$ для першого рівняння, $b_{z,2}$ -для другого рівняння, $b_{z,z-u}$ для останнього рівняння. Всі інші компоненти рівнянь належать до інформаційних блоків з номерами більшими за $z$ та блоків з номерами більшими за $u$. Виходячи з цього, символи останнього з пошкоджених в процесі передачі блоку: $b_{z1},b_{z2},...,b_{z,z-u}$ відновлюються у вигляді:

$$\begin{aligned}
&b_{z,1} = c_{4,2\cdot(m-z)+1} \oplus b_{m,2\cdot(m-z)+1} \oplus\\
&\oplus b_{m-1,2\cdot(m-z)-1} \oplus ... \oplus b_{z+1,3}\\
&b_{z,2} = c_{4,2\cdot(m-z)+2} \oplus b_{m,2\cdot(m-z)+2} \oplus\\
&\oplus b_{m-1,2\cdot(m-z)} \oplus ... \oplus b_{z+1,4}\\
&\cdots\cdots\cdots\cdots\\
&b_{z,z-u} = c_{4,2\cdot(m-z)+z-u} \oplus b_{m,2\cdot m-z-u} \oplus\\
&\oplus b_{z-1,z-u+2} \oplus ... \oplus b_{z-(z-u)div2,\omega}
\end{aligned} \qquad (8)$$

3. Організується цикл відновлення значень спотворених при передачі символів блоків $B_v$, $B_w$, $B_u$, $B_z$ зі змінною $j$, що послідовно приймає значення від одиниці до $n$. До початку циклу відновлені $w$-$v$ перших символів першого з пошкоджених блоків $B_v$ та $z$-$u$ початкових символів останнього з пошкоджених інформаційних блоків - $B_z$ В рамках циклу виконується корекція ($j$+$v$-$w$)-го символу $B_v$, ($j$+$z$-$u$)-го символу $B_z$, $j$-того символу двох пошкоджених при передачі інформаційних блоків: $B_w$ та $B_u$. В циклі виконується наступна послідовність дій:

3.1. Якщо $j$<$n$-$w$+$v$, то відновлюється значення $b_{v,\mu}$ де $\mu$=$j$+($w$-$v$) за формулами:

При $\mu < 2\cdot m$:

$$b_{v,\mu} = c_{3,2\cdot v+\mu-2} \oplus \bigoplus_{q=1}^{v-1} b_{q,\mu-2\cdot q} \oplus \bigoplus_{q=v+1}^{\mu\ div2} b_{q,\mu-2\cdot q}$$

При $2\cdot m \le \mu \le n$ :

$$b_{v,\mu} = c_{3,2\cdot v+\mu-2} \oplus \bigoplus_{q=1}^{v-1} b_{q,\mu-2\cdot q} \oplus \bigoplus_{q=v+1}^{m-1} b_{q,\mu-2\cdot q} \qquad (9)$$

При $\mu > n$ :

$$b_{v,\mu} = c_{3,2\cdot v+\mu-2} \oplus \bigoplus_{q=(\mu-n)div2}^{v-1} b_{q,\mu-2\cdot q} \oplus \bigoplus_{q=v+1}^{m-1} b_{q,\mu-2\cdot q}$$

3.2. Якщо $j$<$n$-$z$+$u$, то відновлюється значення $b_{z,\lambda}$ де $\lambda$=$j$+($z$-$u$) за формулами:

При $\lambda < 2\cdot m$:

$$b_{v,\lambda} = c_{4,2(m-z)+\lambda+1} \oplus \bigoplus_{q=0}^{m-z-1} b_{m-q,\lambda-2\cdot q} \oplus \bigoplus_{q=m-z+1}^{\lambda div2} b_{m-q,\lambda-2\cdot q}$$

При $2\cdot m \le \lambda \le n$ :

$$b_{v,\lambda} = c_{4,2(m-z)+\lambda+1} \oplus \bigoplus_{q=0}^{m-z-1} b_{m-q,\lambda-2\cdot q} \oplus \bigoplus_{q=m-z+1}^{m-1} b_{m-q,\lambda-2\cdot q} \qquad (10)$$

При $\lambda > n$ :

$$b_{v,\lambda} = c_{4,2(m-z)+\lambda+1} \oplus \bigoplus_{q=(\lambda-n)div2}^{m-z+1} b_{m-q,\lambda-2\cdot q} \oplus \bigoplus_{q=m-z+1}^{m-1} b_{m-q,\lambda-2\cdot q}$$

3.3. Відновлюється значення $b_{w,j}$ згідно із формулами:

При $j < m$-$w$:

$$b_{w,j} = c_{1,j} \oplus \bigoplus_{q=1}^{w-1} b_{q,j+w-q} \oplus \bigoplus_{q=w+1}^{j+w-1} b_{q,j+w-q}$$

При $m$-$w \le j \le n$-$w$ :

$$b_{w,j} = c_{1,j} \oplus \bigoplus_{q=1}^{w-1} b_{q,j+w-q} \oplus \bigoplus_{q=w+1}^{m} b_{q,j+w-q} \qquad (11)$$

При $j > n$-$w$ :

$$b_{w,j} = c_{1,j} \oplus \bigoplus_{q=j+w-n}^{v-1} b_{q,j+w-q} \oplus \bigoplus_{q=w+1}^{m} b_{q,j+w-q}$$

3.4. Відновлюється значення $b_{u,j}$ згідно із формулами:

При $j < m$:

$$b_{u,j} = c_{2,j} \oplus \bigoplus_{q=0}^{m-u-1} b_{m-q,j+m-u-q} \oplus \bigoplus_{q=m-u+1}^{m+j-u-1} b_{m-q,j+m-u-q}$$

При $m \le j \le n$-$m$+$u$ :

$$b_{u,j} = c_{2,j} \oplus \bigoplus_{q=0}^{m-u-1} b_{m-q,j+m-u-q} \oplus \bigoplus_{q=m-u+1}^{m-1} b_{m-q,j+m-u-q} \qquad (12)$$

При $j > n$-$m$+$u$ :

$$b_{u,j} = c_{2,j} \oplus \bigoplus_{q=j+m-u-n}^{v-1} b_{m-q,j+m-u-q} \oplus \bigoplus_{q=m-u+1}^{m-1} b_{m-q,j+m-u-q}$$

Таким чином, з використанням гранично простих операцій XOR, запропонована процедура забезпечує рекурсивне виправлення будь-яких $h$ блоків даних пошкоджених в процесі при передачі.

З наведеного вище опису процедури корекції $h$ блоків даних (формули 5-12) слідує, що при відновленні кожного з $n$ символів виконується операція XOR, причому кількість доданків в кожній із сум не перевищує $m$. Звідси очевидним є той факт, що час T корекції $h$ блоків інформаційної посилки не перевищує значення:

$$T \le h \cdot m \cdot n \cdot t_{XOR}, \qquad (13)$$

де $t_{XOR}$ - час виконання операції XOR.

Загальна кількість $N$ контрольних символів, що передаються при використанні запропонованого методу визначається формулою:

$$N = h \cdot (2 \cdot m + n - h) \qquad (14)$$

При використанні кодів Ріда-Соломона загальна кількість $K$ контрольних символів визначається формулою [1]:

$$K = 2 \cdot l \cdot m, \qquad (15)$$

де $l$- максимальна довжина "пачки помилок" в блоці. Аналіз наведених виразів (14) і (15) дозволяє зробити висновок про те, що з точки зору кількості додаткової інформації, що використовується для корекції помилок, запропонований метод більш ефективний а порівнянні з кодами Ріда-Соломона при виконанні умови:

$$l > h \cdot (1 + \frac{n}{2 \cdot m}), \qquad (16)$$

тобто в ситуаціях коли максимальна довжина "пачки помилок" становить значну величину. Наприклад, за умови, що кількість блоків $m$=100, а довжина кожного блоку становить n=256 символів, запропонований метод, розрахований на корекцію 4-х блоків ($h$=4), більш ефективний в плані використання меншого об'єму контрольної інформації вже при $l$ >5.

На практиці вже зараз максимальна тривалість дії зовнішньої завади може перевищувати час передачі 20 символів [2]. Зі зростанням швидкості передачі даних значення $l$ - максимальної кількості символів (несучих сигналів), передача яких підпвдвє під дію зовнішньої завади має стійку тенденцію до зростання. Аналіз формул (15) та (16) свідчить про те, що запропонований метод ефективніший за використання корегуючих кодів і, зокрема, кодів Ріда-Соломона при великій кількості блоків в інформаційній посилці.

Аналіз тенденцій інформаційного обміну в глобальних мережах [4] переконливо свідчить про зростання об'ємів даних в повідомленнях.

Таким чином, аналіз тенденцій розвитку засобів передачі даних в комп'ютерних глобальних мережах свідчить про те, що переваги запропонованого методу в порівнянні з корегуючими кодами в перспективі зростатимуть.

Слід зазначити також і той факт, що на відміну від корегуючих кодів, запропонований метод не накладає жодних обмежень на довжину блоку даних. При використанні кодів Ріда-Соломона довжина блоку не може перевищувати $2^d$-$2 \cdot l$ символів, де $d$ - їх розрядність. Так при використанні амплітудно-фазової модуляції QAM-256 значення розрядності $d$ символу дорівнює 8 і при $l$=20 максимальна довжина інформаційної частини блоку не може перевищувати 216 байтів.

Запропонований метод, на відміну від кодів Ріда-Соломона не накладає обмежень на довжину "пачки" спотворених зовнішньою завадою символів.

Перевагою запропонованого методу в порівнянні з кодами Ріда-Соломона є значне прискорення та спрощення контролю та виправлення помилок. Для виявлення помилок передачі блоку при використанні кодів Ріда-Соломона потрібно обчислити $2 \cdot l$ синдромів, що потребує

часу $T_c = 2 \cdot l \cdot n \cdot (t_m + t_{xor})$, де $t_m$ - час множення на полі Галуа двох $d$-розрядних символів. Враховуючи, що $t_m \approx 2 \cdot d \cdot t_{xor}$, то $T_c \approx 6 \cdot l \cdot n \cdot d \cdot t_{xor}$. Виявлення помилок в блоці для запропонованого методу становить $T_d \approx 4 \cdot n \cdot d \cdot t_{xor}$, що забезпечує прискорення в l раз.

Запропонована процедура корекції значно простіша в порівнянні з тими, що використовуються в корегуючих кодах як з точки зору логіки операцій, так і з точки зору складності операцій, що робить ефективнішим як програмну реалізацію, так і апаратну з використанням FPGA -структур.

## Висновки

В роботі запропоновано метод виправлення спотворених зовнішніми завадами блоків даних, що передаються по радіоналам глобальних комп'ютерних мереж.

В основу методу покладено концепцію розділення функцій виявлення на виправлення помилок: виявлення помилок реалізується з використанням циклічних надлишкових кодів або зважених контрольних сум, а їх виправлення відбувається за рахунок додаткової передачі $h$ контрольних блоків, що забезпечує виправлення "пачки" будь-якої довжини в обмеженій значенням $h$ кількості блоків.

Важливими перевагами запропонованого методу в порівнянні з відомими корегуючими кодами є відсутність обмежень на тривалість дії зовнішньої завади, обмежень на довжину блоку даних, значно менший час виявлення помилок, суттєво простіші процедури корекції, що не передбачають розв'язання символьних систем рівнянь на полях Галуа.

Показано, що переваги запропонованого методу в порівнянні з корегуючими кодами зі збільшенням швидкості передачі даних в перспективі зростатимуть.

Розроблений метод може бути ефективно використано в перспективних засобах забезпечення надійного обміну даними з використанням радіоканалів в глобальних мережах.

## Список літератури

1. Скляр Б. Цифровая связь. Теоретические основы и практическое применение / Скляр Б.- М.: Изд. дом "Вильямс".- 2004.- 1104 с.
2. Yeung R.W. Network error correction. Part 1:Basic conseption and upper bounds/ Yeung R.W., Cai N. // Commun. Information Systems .-Vol. 6.- 2006.- PP. 19-36.
3. Марковський О.П. Ефективний метод корекції "пачки" помилок у каналах зі спектральною модуляцією / Марковський О.П., Рябикіна В.О., Семенюк Ю.В. // Вісник Національного технічного університету України "КПІ" Інформатика, управління та обчислювальна техніка, – Київ: ВЕК+ – 2012 – № 55.- С.28-33.
4. Klove T. Error Detecting Codes: General Theory and Their Application in Feedback Communication Systems / T. Klove, V. Korzhik.- Norwell, MA: Kluwer, 1995. – 433 p.
5. Xuan Guang Constraction of Network Error Correction Codes in Packet Network / Xuan Guang, Fang-Wei Fu, Zhen Zhang // IEEE Transaction on information theory. Vol.59- № 2- 2013 - PP.1030-1047.

УДК 004.724.4(045)

*КУЛАКОВ Ю.А.,*
*ГАБИНЕТ А.В.*

# СПОСОБ ВЫБОРА КЛИЕНТОВ В PEER-TO-PEER СЕТЯХ С УЧЕТОМ ИХ МЕСТОПОЛОЖЕНИЯ И ЗАДЕРЖЕК МЕЖДУ НИМИ

В настоящей статье был предложен способ выбора клиентов, учитывающий физическое местоположение пиров и задержки между ними. Кратчайший путь между клиентами в соответствующих AS определяется на основе построения графа из ближайших автономных систем. Проведен анализ зависимости коэффициента нагрузки от кол-ва клиентов и размещения их в различных автономных системах.

In this article was proposed a method of selecting clients, taking into account the physical location of peers and the delay between them. The shortest path between clients in the relevant AS is based on constructing a graph of the closest autonomous systems. The analysis of the dependence of the load factor on the number of customers and placing them in different autonomous systems is adopted.

## Введение

Функционирование любой peer-to-peer системы распределения контента опирается на узлы и соединения между ними. Эта сеть образуется поверх и независимо от базовой (в типичном случае IP) и поэтому часто называется оверлейной. Топология, структура, степень централизации оверлейной сети, механизмы локализации и маршрутизации, которые в ней используются для передачи сообщений и контента, являются решающими для работы системы, поскольку они воздействуют на ее отказоустойчивость, производительность, масштабируемость и безопасность.

В основном существуют три различные архитектуры p2p систем[1,2]:

1) Централизованные (Napster[3]),
2) Децентрализованные структурированные (Gnutella[4], Edutella[5]),
3) Децентрализованные неструктурированные (Pastry[6], Chord[7], CAN[8], Tapestry[9]).

Децентрализованные структурированные системы используют распределенные хеш-таблицы (DHT[10]) для поиска данных на узлах. Недостатком таких систем является то, что не учитывается физическое расстояние между узлами. Таким образом в статье предложен способ, который позволяет на основании задержек между автономными системами находить кратчайший маршрут и использовать существующие алгоритмы внутри самой автономной системы.

## Модифицированный способ

Сервер, который транслирует потоковое видео при подключении клиента к трансляции определяет по IP адресу, в какой автономной системе он находится. Таким образом будет определяться физическое место клиента в сети Интернет.

Поиск нужных кусков информации осуществляется с помощью связи ключ-значение. Идентификаторы ключа и значения состоят из следующих компонентов: IP адрес транслирующего сервера (32 бита), номер автономной системы (32 бита), IP адрес клиента (32 бита) и случайно сгенерирована хэш функция (32 бита). Последнее значение необходимо для того, чтобы каждый ключ и кусочек мультимедийных данных имели уникальный идентификатор, размером 128 бит.

Во время того, как клиент подключается к серверу, последний строит граф, в вершину которого он помещает себя. Ветки этого графа – это номера автономных систем, которые существуют в сети Интернет. В результате клиент, который послал запрос серверу для получения трансляции, будет помещен в узел дерева с номером соответствующей автономной системы. Пример графа показан на рис. 1.

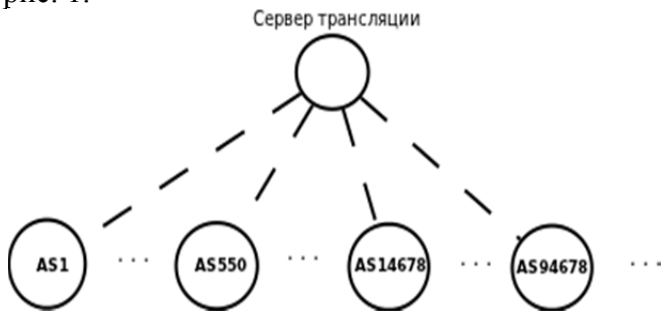



***Рис. 1. Построения графа при подключении клиентов***

Важный момент заключается в том, что при определении AS клиента, сервер также ищет

все соседние автономные системы по отношению к AS клиента (рис. 2)

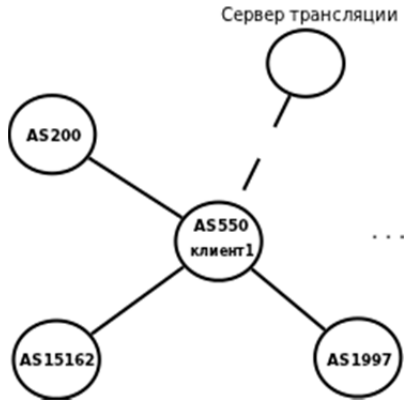



***Рис. 2. Добавление соседних вершин узла***

Таким образом, если соседняя автономная система окажется также соседней по отношению к AS другого клиента, то таким образом появится маршрут между двумя пирами, который будет является кратчайшим. При появлении новых клиентов и добавлении новых вершин к графу (рис.3) будет строиться остовное дерево с помощью алгоритма Прима[11]. Поиск кратчайшего маршрута между пирами будет осуществляться с помощью этого построенного дерева.

Поиск пиров для обмена мультимедийными данными происходим следующим образом. Внутри каждой автономной системе поиск нужной информации осуществляется средствами распределенных хеш-таблиц (DHT), которые используют существующий алгоритм Pastry. То есть здесь используется принцип децентрализации и чем больше активных узлов будет находиться в одной автономной системе, тем эффективнее он будет.

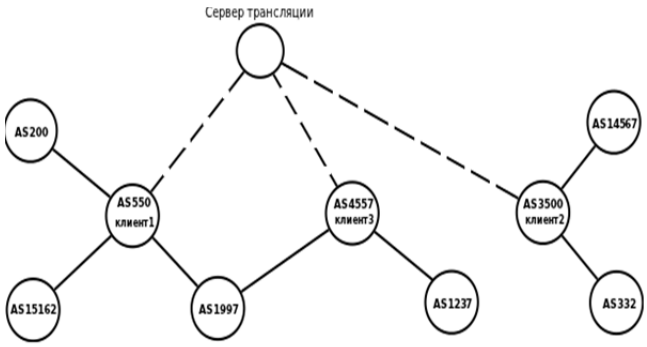



***Рис. 3. Построение графа на сервере трансляции***

Когда клиент найдет всех возможных пиров в локальной автономной системе, он отправляет запрос к серверу трансляции для получения информации о пирах из других автономных систем.

Сервер посылается на свой граф, который был построен в результате подключения новых клиентов. Таким образом он выдает адрес потенциально ближайшего пира к клиенту, который запрашивает информацию. Стоит отметить, что если в этой автономной системе находятся несколько пиров, то будет отдаваться адрес пира случайным образом. Далее по аналогии клиент с помощью алгоритмов DHT ищет пиров в ближайшей автономной системе.

По окончанию поиска клиент опять обращается с запросом о новых пирах. Если нету связи между вершинами автономных систем графа, в которых размещены клиенты, то сервер определяет случайным образом номер автономной системе для дальнейшего поиска.


***Рис. 4. Физическая топология***

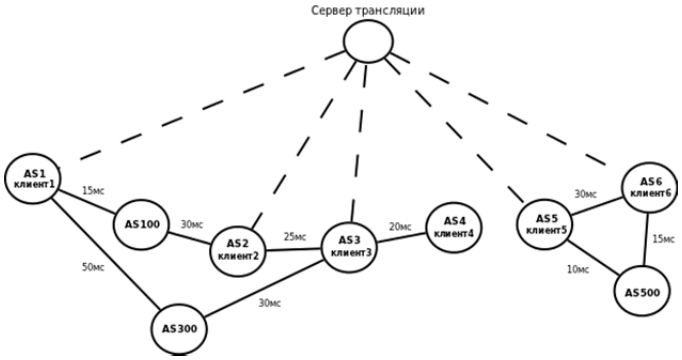



***Рис. 5. Логическая топология для проведения экспериментов с предложенным алгоритмом***

## Моделирования передачи данных в p2p сети

На рис. 4 представлена полносвязная топология между узлами и сервером трансляции. Если рассматривать логические соединения, то для проведения экспериментов

использовались две разные топологии(рис. 5, рис. 6).

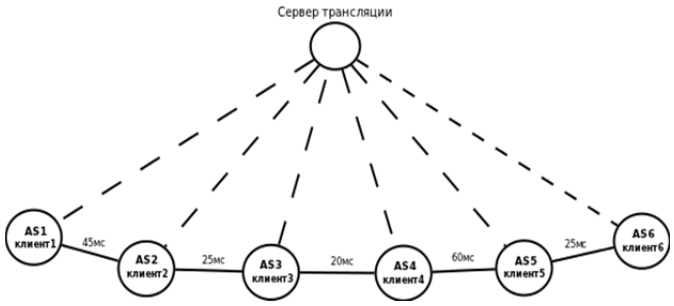



***Рис. 6. Логическая топология для проведения экспериментов с базовым алгоритмом***

### Проведения экспериментов и анализ результатов

Для числового отображения качества алгоритма мы ввели переменную коэффициента нагрузки. Он показывает распределение потока входных и выходных данных на каждом из узлов и считается по формуле:

$$К_н = \frac{V_i}{V_o}, \quad (1)$$

где $V_i$ – количество входящих данных, $V_o$ – количество исходящих данных.

Для сбора статистики входящих и исходящих данных, а также вычисления $К_н$ на каждом узле был написан скрипт, который размещался на каджом из узлов.

Для того, чтобы статистика собиралась централизовано, использовался скрипт, который опрашивал каждый из узлов и получал соответствующие значения $К_н$.

Эксперименты были поделены на 2 этапа: использование готового алгоритма в идеальных условиях (нету задержек между пирами); использование готового и предложенного алгоритмов в реальной сети (в качестве узлов используются автономные системы, между которыми существуют различные задержки).

Результаты всех экспериментов приведены ниже на графиках (рис. 7, 8, 9).

На рис. 7 при проведении эксперимента на основе базового алгоритма Pastry виден экспоненциальный график. Это значит, что при увеличении количества клиентов при трансляции потокового видео коэффициент нагрузки стремится к единице. То есть количество входящего и исходящего трафика практически равно между собой. Получается, что клиент получает столько же данных для просмотра потокового видео, сколько он отдает (ретранслирует) другим пирам в сети.

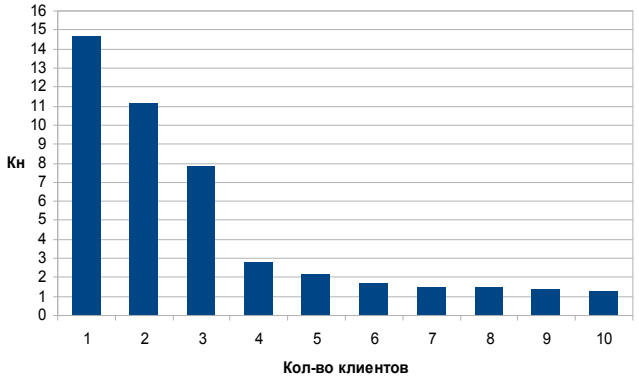



***Рис. 7. Зависимость коэффициента нагрузки от количества клиентов***

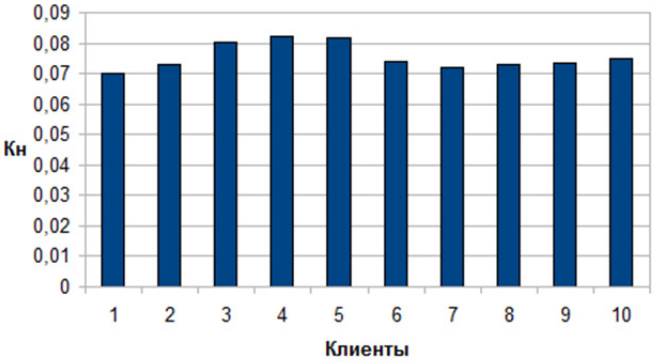



***Рис. 8. Зависимость коэффициента нагрузки на сервере трансляции от количества подключенных клиентов***

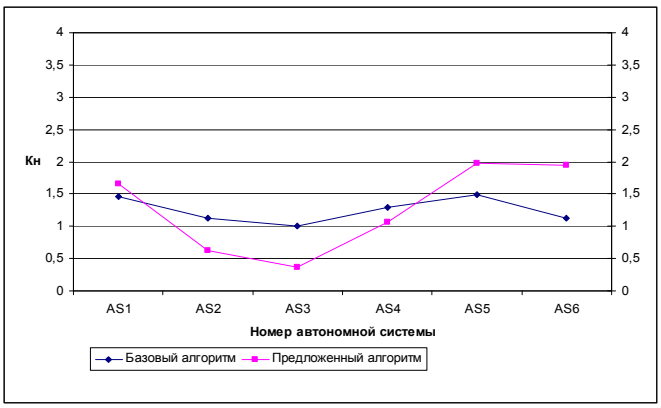



***Рис. 9. Графики зависимости коэффициента нагрузки от автономной системы***

Поэтому можно сделать вывод о том, что при большом количестве клиентов нагрузка на peer-to-peer сеть равномерно распределяется между ними. В этом и заключается основная задача DHT алгоритмов.

На рис. 8 был представлен график зависимости коэффициента нагрузки на сервере трансляции от количества подключенных клиентов. Он приближенный к прямой линии. Это объясняется тем, что коэффициент нагрузки на сервере совсем не зависит от

количества клиентов. При этом эти значения значительно меньше единицы, потому что на сервере трансляции в основном существует только исходящий трафик.

Последние два эксперимента заключались в сравнении базового алгоритма Pastry и предложенного нами алгоритма с учетом задержек между узлами. То есть клиенты находились в разных сетях, а точнее в разных автономных системах. Физическая топология в обоих случая была одинакова (рис. 4), но логические топологии отличались друг от друга (рис. 5, рис. 6). На рис. 9 изображено 2 графика, которые характеризуют зависимость коэффициента нагрузки от номера автономной системы, в которой размещался тот или иной клиент.

Проанализируем эти 2 графика отдельно:

– в базовом алгоритме совсем не учитываются задержки между узлами. Коэффициенты нагрузки на каждом из клиентов практически равны между собой и не зависят от расстояния между клиентами.

– в предложенном алгоритме идет неравномерное распределение коэффициентов нагрузки между автономными системами. Это объясняется тем, что в предложенном алгоритме учитывается расстояние между автономными системами. Поэтому в клиентах, расположенные в автономных системах AS2, AS3, которые имеют соседние вершины совсем близко друг к другу, больше исходящего трафика чем в клиентах из отдаленных автономных системах(AS1, AS4).

Также можно отметить, что клиенты, которые находятся в вершинах графа, который не связан с основным (граф, который возник при появлении первых клиентов, размещенных в соседних автономных системах) является наиболее отдаленным (автономные системы AS5, AS6). На графике видно, что отрезок между этими автономными системами имеет форму прямой линии еще и с довольно большим коэффициентом нагрузки. Это можно объяснить тем, что входящий трафик больше исходящего, потому что передача видео данных клиентам, которые находятся далеко от них, не столь приоритетна, как обмен видео потоком между соседними автономными системами. А вот клиент, который размещен в автономной системе AS3 взаимодействует с другими активными автономными системами больше всего. В этом случае исходящий трафик превосходит входящий.

При сравнении с базовым алгоритмом можно отметить, что коэффициент нагрузки при использовании предложенного алгоритма на клиентах, которые находятся в автономных системах AS2, AS3, AS4 меньше на 24,9%, 33,3%, 8,3% соответственно.

В результате сравнения двух графиков после проведения экспериментов базового и предложенного алгоритмов на основе данной топологии были обнаружены недостатки предложенного нами алгоритма. Они заключаются в том, что коэффициент нагрузки клиентов, которые находятся далеко от других (вершины графа, которые не соединены с основным, а также вершины, которые находятся далеко от основного скопления других активных вершин) достаточно больше по сравнению с базовым алгоритмом (около 10,99%, 19,7%, 32,1% соответственно для AS1, AS5, AS6).

## Выводы

1. При увеличении количества клиентов коэффициент нагрузки на каждом из них стремится к единице, что означает равномерную нагрузку в peer-to-peer сети.

2. На сервере трансляции коэффициент нагрузки не зависит от количества включенных пиров.

3. Предложенный способ за счет учета задержек между клиентами позволяет повысить эффективность передачи видео данных через peer-to-peer сеть.

4. В качестве недостатка предложенного алгоритма можно отметить следующее, что коэффициент нагрузки удаленных клиентов от основного скопления активных пиров, больше по сравнению с базовым алгоритмом Pastry.

## Список литературы

1. D. Chopra, H. Schulzrinne, E. Marocco, and E. Ivov. Peer-to-Peer Overlays for Real-time Communication: Security issues and Solutions. IEEE Communications Surveys Tutorials 11, no. 1, 2009, pages 4–12.
2. Philip A. Bernstein, Fausto Giunchiglia, Anastasios Kementsietsidis, John Mylopoulos, Luciano Serafini, and Ilya Zaihrayeu, “Data management for peer-to-peer computing : A vision,” in WebDB 2002, June 2002, pp. 89–94.

3. Napster: http://www.collegetermpapers.com/viewpaper/1304273730.html.
4. Gnutella: http://web.archive.org/web/20090331221153/http://wiki.limewire.org/index.php?title=GDF.
5. Edutella: http://edutella.jxta.org/
6. A. I. T. Rowstron, P. Druschel. Pastry: Scalable, Decentralized Object Location, and Routing for Large-Scale Peer-to-Peer Systems. In: Proceedings of the IFIP/ACM International Conference on Distributed Systems Platforms (Middleware), 2001, pages 329–350.
7. I. Stoica, R. Morris, D. Liben-Nowell, D. R. Karger, M. F. Kaashoek, F. Dabek, and H. Balakrishnan. Chord: A Scalable Peer-to-Peer Lookup Protocol for Internet Applications. IEEE/ACM Transactions on Networking 11, no. 1, 2003, pages 17–32.
8. S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker. A Scalable Content-addressable Network. In: Proceedings of the 2001 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications (ACM SIGCOMM), 2001, pages 161–172.
9. B. Y. Zhao, L. Huang, J. Stribling, S. C. Rhea, A. D. Joseph, and J. D. Kubiatowicz. Tapestry: A Resilient Global-scale Overlay for Service Deployment. IEEE Journal on Selected Areas in Communications 22, no. 1, 2004, pages 41–53.
10. T. Balke, W. Siberski, DHT Algorithms, Springer LNCS 3485, 2007
11. Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. Introduction to Algorithms, Third Edition. MIT Press, ISBN 0-262-03384-4. Section 23.2: The algorithms of Kruskal and Prim, 2009, pp. 631–638.

УДК 519.854.2

*ПАВЛОВ А.А.,*
*МИСЮРА Е.Б.*

# МИНИМИЗАЦИЯ СУММАРНОГО ЗАПАЗДЫВАНИЯ ПРИ ВЫПОЛНЕНИИ НЕЗАВИСИМЫХ ЗАДАНИЙ С ОБЩИМ ДИРЕКТИВНЫМ СРОКОМ ИДЕНТИЧНЫМИ ПАРАЛЛЕЛЬНЫМИ ПРИБОРАМИ, МОМЕНТЫ ЗАПУСКА КОТОРЫХ ПРОИЗВОЛЬНЫ

Рассматривается задача составления расписания выполнения независимых заданий с общим директивным сроком и равными весами при произвольных моментах запуска приборов. Предложен ПДС-алгоритм решения задачи, сформулированы признаки оптимальности полиномиальной составляющей алгоритма, получена оценка отклонения получаемых решений от оптимального. Приведен пример решения задачи.

The problem of scheduling independent tasks with a common due date and equal weights in the case of arbitrary starting times of machines is considered. The PDC-algorithm to solve it is proposed. The optimality signs of the polynomial component of the algorithm are formulated. The upper bound of deviation of a solution from optimum is obtained. An example of the problem solution is given.

## Введение

ПДС-алгоритмы [1] базируются на идеях выделения полиномиальной составляющей алгоритма и экспоненциального подалгоритма. С этой целью для каждого алгоритма выводятся логико-аналитические условия, проверяемые в процессе решения произвольной индивидуальной труднорешаемой задачи, в случае выполнения которых данная произвольная индивидуальная задача точно решается полиномиальным подалгоритмом.

Существуют два типа ПДС-алгоритмов. В первом типе экспоненциальная составляющая представлена либо аппроксимацией полиномиальной составляющей, либо аппроксимирующим экспоненциальным алгоритмом с верхней заранее заданной оценкой числа итераций (или ограничением по времени решения задачи). В результате получаем: а) оптимальное значение функционала, если в процессе выполнения алгоритма реализованы признаки оптимальности получаемых решений; б) приближенное решение с оценкой отклонения от оптимального; в) эвристическое решение, если такая оценка не может быть получена.

В ПДС-алгоритмах второго типа экспоненциальная составляющая представлена экспоненциальным подалгоритмом, построенным на теоретически обоснованных направленных перестановках и эффективных отсечениях. В результате решения задачи либо реализуется полиномиальная составляющая алгоритма, либо после выполнения всех направленных перестановок и отсечений получаем решение, оптимальность которого доказана в соответствующих теоремах.

Алгоритм решения исследуемой ниже задачи относится к ПДС-алгоритмам первого типа.

## Постановка задачи МСЗПО

Задано множество заданий $J = \{1, 2,..., n\}$, $m$ приборов равной производительности, для каждого задания $j \in J$ известна длительность выполнения $l_j$. Все задания имеют общий директивный срок $d$. Моменты запуска приборов на выполнение работ $T_i \gtreqless d$, $i = \overline{1,m}$, различны. Простои приборов при выполнении заданий запрещены.

Необходимо построить расписание $\sigma$ выполнения заданий $j \in J$ на $m$ приборах такое, чтобы достигался минимум функционала:

$$F(\sigma) = \sum_{j \in J} \max[0; C_j(\sigma) - d],$$

где $C_j(\sigma)$ – момент завершения выполнения задания $j$ в последовательности $\sigma$.

Сформулированная задача относится к классу *NP*-трудных [2]. В [1] представлен ПДС-алгоритм решения сформулированной задачи при условии, что моменты запуска приборов одинаковы (МСЗП). Для удобства изложения материала приведем основные теоретические результаты по решению задачи, приведенной в [1].

*Алгоритм построения начального расписания.* Пронумеруем задания множества $J = \{1, 2,..., n\}$ по неубыванию значений $l_j$, а приборы – по неубыванию значений $T_i$. На первом шаге выбираем задание $j$ с минимальным $l_j$ и назначаем на прибор $i$ с минимальным $T_i$. Определяем время освобождения прибора $i$:

$T_i^{осв} = T_i^{осв} + l_j$. Далее выбираем очередное задание для назначения на выполнение и назначаем на прибор с минимальным временем освобождения $T_i^{осв}$. Такую процедуру выполняем до тех пор, пока не будут распределены на выполнение все задания. Обозначим полученное расписание через $\sigma^{уп}$.

Поиск оптимального расписания в соответствии с известной теоремой можно ограничить рассмотрением расписаний, при которых каждый прибор обслуживает задания в порядке возрастания их номеров [3].

Действительно, если в оптимальной последовательности $\sigma_i^* = (j_1^*, j_2^*, ..., j_n^*)$ для некоторых заданий $j_k$, $j_{k+1}$ выполняется $l_{j_k} > l_{j_{k+1}}$, то последовательности $\sigma_i$, отличающейся от $\sigma_i^*$ транспонированием элементов $j_k^*$ и $j_{k+1}^*$, соответствует меньшее значение рассматриваемого функционала, что противоречит предположению об оптимальности $\sigma_i^*$.

Используемые обозначения:

$P_i(\sigma)$ – множество незапаздывающих заданий в расписании прибора $i$;

$S_i(\sigma)$ – множество запаздывающих заданий в расписании прибора $i$, для которых выполняются условия:

$$S_j^н < d,\ C_j > d,\ \forall j \in S_i(\sigma),$$

где $S_j^н$ — момент начала выполнения задания $j$;

$Q_i(\sigma)$ – множество запаздывающих заданий в расписании прибора $i$, для которых выполняются условия:

$$S_j^н < d,\ \forall j \in Q_i(\sigma),$$

$$P = \bigcup_{i=\overline{1,m}} P_i;\ S = \bigcup_{i=\overline{1,m}} S_i;\ Q = \bigcup_{i=\overline{1,m}} Q_i;$$

$R_i(\sigma)$ – резерв времени прибора $i$ в расписании $\sigma$

$$R_i(\sigma) = d - \sum_{j \in P_i(\sigma)} l_j;$$

$\Delta_i(\sigma)$ – запаздывание в выполнении задания $j \in S_i(\sigma)$ относительно директивного срока:

$$\Delta_i = \sum_{j \in P_i(\sigma) \bigcup S_i(\sigma)} l_j - d.$$

Для случая равных $T_i$ в [3] сформулирована теорема, определяющая класс расписаний, который содержит оптимальное решение по рассматриваемому функционалу на множестве всех возможных расписаний, построенных для фиксированного множества заданий $J$. Используем для удобства изложения принятые нами обозначения.

*Теорема* 1 [3]. Существует оптимальное расписание, при котором выполняются условия:

1) $P \bigcup S = \{1, 2, ..., |P \bigcup S|\}$;
2) если $P \bigcup S < n$, то $\sum_{j \in P_i \cup S_i} l_j \geq d$, и $Q_i \setminus S_i$ содержит те и только те элементы, которые отличаются от $|P \bigcup S| + i$ на величину, кратную $m$, $i = \overline{1,m}$.

Обозначим через $\Psi_{PS}$ класс расписаний, удовлетворяющий условию теоремы 1. Выделим из класса $\Psi_{PS}$ класс расписаний $\Psi_P \in \Psi_{PS}$, удовлетворяющий дополнительно следующим условиям:

1) $P = \{1, 2, ..., |P|\}$;
2) $\min_{j \in S(\sigma)} l_j > \max_{i=\overline{1,m}} R_i(\sigma)$;
3) $S_{j_k}^H \leq S_{j_l}^H$, если $l_{j_k} \leq l_{j_l}$, $\forall j_k, j_l \in S(\sigma)$.

Пусть $|P| < n$. Обозначим: $P_{\min}$ – минимальное количество заданий множества $P$, при котором $\Psi_P \neq \varnothing$; $P_{\max}$ – максимальное количество заданий множества $P$.

В следующих утверждениях обосновывается построение структуры ПДС-алгоритма и определяются правила направленных перестановок, выполняемых в процессе его реализации.

*Утверждение* 1 [1]. Для всех возможных расписаний $\sigma \in \Psi_P$, построенных на множестве заданий $J$, справедливо: $P_{\max} - P_{\min} < m$.

*Утверждение* 2 [1]. При построении оптимального расписания в результате направленных перестановок возможны перемещения заданий только между множествами $P(\sigma)$ и $S(\sigma)$.

*Утверждение* 3 [1]. При перестановке задания $j \in P(\sigma)$ с прибора $i_k$ с бóльшим числом запаздывающих заданий на прибор $i_l$ с меньшим числом запаздывающих заданий уменьшается $\Delta_{i_k}$ на величину, равную $l_j$.

*Утверждение* 4 [1]. Максимальная разность количества запаздывающих заданий на приборах в расписаниях $\sigma \in \Psi_P$ не превышает единицы.

На основании следующих теорем обосновываются признаки оптимальности получаемых расписаний, определяется основная характеристика получаемых расписаний, которая показывает, на сколько можно теоретически уменьшить значение функционала, чтобы получить оптимальное расписание.

*Определение* 1. Расписание с одинаковым числом запаздывающих заданий на приборах назовем равномерным.

*Теорема* 2 [1]. Равномерное расписание $\sigma \in \Psi_P$ является оптимальным.

Введем обозначения:

$L_{\max}$ – максимальное число запаздывающих заданий на приборах;

$L_{\min}$ – минимальное число запаздывающих заданий на приборах;

$i = \overline{1,k}$ – приборы с числом запаздывающих заданий $L_{\max}$;

$|P(\sigma)| = P$ – мощность множества $P(\sigma)$;

$$\Delta_\Sigma(\sigma) = \sum_{i=1}^{k} \Delta_i ;$$

$$R_\Sigma(\sigma) = \sum_{i=k+1}^{m} R_i ; \Omega_\Sigma(\sigma) = \min\{R_\Sigma(\sigma), \Delta_\Sigma(\sigma)\}.$$

*Теорема* 3 [1]. Если в расписаниях $\sigma \in \Psi_P$, $\sigma' \in \Psi_P$, построенных на заданном множестве заданий $J$, максимальное число запаздывающих заданий одинаково, то при $R_\Sigma(\sigma) \neq 0$ и $R_\Sigma(\sigma') \neq 0$ справедливо: $R_\Sigma(\sigma) - R_\Sigma(\sigma') = \Delta_\Sigma(\sigma) - \Delta_\Sigma(\sigma')$.

*Теорема* 4. Для любых двух расписаний $\sigma \in \Psi_P$ и $\sigma' \in \Psi_P$ справедливо следующее:

$$F(\sigma) - F(\sigma') = \Omega_\Sigma(\sigma) - \Omega_\Sigma(\sigma'). \quad (1)$$

*Теорема* 5. Если в расписании $\sigma \in \Psi_P$ выполняется $\Omega_\Sigma(\sigma) = \min\{R_\Sigma(\sigma), \Delta_\Sigma(\sigma)\} = 0$, то расписание $\sigma$ оптимально.

Величина $\Omega_\Sigma(\sigma) = \min\{R_\Sigma(\sigma), \Delta_\Sigma(\sigma)\}$ является основной характеристикой расписания $\sigma \in \Psi_P$, где $\Delta_\Sigma(\sigma)$ показывает, на сколько можно теоретически уменьшить значение функционала $F(\sigma)$, чтобы получить оптимальное расписание. Суммарный резерв $R_\Sigma(\sigma)$ показывает, какие резервы существуют для получения оптимального расписания.

*Утверждение* 5. Пусть $\sigma \in \Psi_P$, тогда если $\Delta_\Sigma(\sigma) = 0$, то $\Omega_\Sigma(\sigma) = 0$.

*Теорема* 6. Для расписания $\sigma \in \Psi_P$ справедливо следующее соотношение:

$$F(\sigma) - F(\sigma^*) \leq \Omega_\Sigma(\sigma).$$

где $\sigma^*$ – оптимальное расписание выполнения множества заданий $J$.

Таким образом, для расписаний $\sigma \in \Psi_P$ величина $\Omega_\Sigma(\sigma)$ является оценкой отклонения значения суммарного запаздывания оптимального расписания.

*Утверждение* 6. Для расписаний $\sigma \in \Psi_P$ выполняется $\Omega_\Sigma(\sigma) \leq \Delta_\Sigma(\sigma)$.

На основании следующих теорем и утверждений обосновываются характеристики расписаний, получаемых в результате перестановок, выполняемых в произвольном порядке и уменьшающих $\Delta_\Sigma(\sigma)$, и, следовательно, $R_\Sigma(\sigma)$ [1]. Эти перестановки последовательно выполняются с некоторого расписания $\sigma \in \Psi_P$ и осуществляются посредством переноса незапаздывающих заданий между приборами $I_\Delta$ и $I_R$ в текущем расписании $\sigma_k$, где $I_R(\sigma)$ – множество номеров приборов расписания $\sigma$, на которых запаздывает меньшее число заданий; $I_\Delta(\sigma)$ – множество номеров приборов расписания $\sigma$, на которых запаздывает большее число заданий. При этом порядок выполнения работ на приборах, кроме указанных выше, не изменяется. В результате проведенной перестановки в полученном расписании $\sigma_{k+1}$ может измениться количество запаздывающих заданий только на одном из приборов из множества $I_\Delta(\sigma_k)$ с номером $i_1$ и на одном приборе из множества $I_R(\sigma_k)$ с номером $i_2$. Перестановка является запрещенной, если в расписании $\sigma_{k+1}$ количество заданий на приборе $i_2$ больше количества заданий на приборе $i_1$.

Обозначим класс таких расписаний через $\Psi(\sigma_P)$.

*Теорема* 7. Для любого расписания $\sigma \in \Psi(\sigma_P)$ справедлива оценка отклонения показателя качества от оптимального значения:

$$F(\sigma) - F(\sigma^*) \leq \Omega_\Sigma(\sigma).$$

*Следствие.* Для расписаний $\sigma \in \Psi(\sigma_P)$ справедливы теоремы 5, 6 и утверждение 6.

Таким образом, в расписаниях $\sigma \in \Psi(\sigma_P)$ изменение значения функционала так же, как и в расписаниях $\sigma \in \Psi_P$, определяется величинами $\Delta_\Sigma(\sigma)$, $R_\Sigma(\sigma)$.

*Теорема* 8. Если в расписании $\sigma \in \Psi(\sigma_P)$ $\Omega_\Sigma(\sigma) = \min\{R_\Sigma(\sigma), \Delta_\Sigma(\sigma)\}$ достигает наименьшего значения, то расписание $\sigma$ оптимально.

В следующих утверждениях, доказательство которых очевидно, сформулированы свойства, характеризующие задания $j \in S \cup Q$ в расписаниях $\sigma \in \Psi(\sigma_P)$.

*Утверждение* 7. Пусть $S'_k(\sigma) \cup Q'_k(\sigma)$, $S''_l(\sigma) \cup Q''_l(\sigma)$ – множества запаздывающих заданий на приборах $k$ и $l$, соответственно. В расписаниях, полученных в результате перестановки множеств запаздывающих заданий между приборами, то есть при выполнении $S''_k(\sigma) \cup Q''_k(\sigma)$, $S'_l(\sigma) \cup Q'_l(\sigma)$, отклонение показателя качества от оптимального расписания определяется функцией $\Omega_\Sigma(\sigma) = \min\{R_\Sigma(\sigma), \Delta_\Sigma(\sigma)\}$.

Пусть на приборах $k$, $l$, $r$ в расписании $\sigma$ задания $j \in S \cup Q$ пронумерованы следующим образом: $j_k, j_{k+m}, j_{k+2m}, \ldots, j_l, j_{l+m}, j_{l+2m}, \ldots, j_r, j_{r+m}, j_{r+2m}, \ldots$

*Определение* 2. Задания $j_k, j_l, j_r$, или $j_{k+m}, j_{l+m}, j_{r+m}, \ldots$ или $j_{k+2m}, j_{l+2m}, j_{r+2m}, \ldots$ назовем запаздывающими заданиями одного уровня.

*Утверждение* 8. На приборах с одинаковым числом запаздывающих заданий задания одного уровня можно менять местами. При таких перестановках значение функционала (показателя качества) не изменяется.

*Утверждение* 9. От расписания $\sigma \in \Psi(\sigma_P)$ всегда можно перейти к расписанию $\sigma \in \Psi_{PS}$ с тем же значением показателя качества посредством перестановки запаздывающих заданий одного уровня.

### Описание ПДС-алгоритма решения задачи МСЗП

В [1] представлены два ПДС-алгоритма решения задачи МСЗП, которые включают полиномиальную составляющую и приближенный алгоритм и строятся только на направленных перестановках. Полиномиальная составляющая алгоритма задается детерминированной процедурой последовательного выполнения направленных перестановок, общее количество которых ограничено полиномом от числа заданий и количества приборов. В результате решения задачи получаем либо строго оптимальное решение полиномиальной составляющей алгоритма (если в процессе вычислений удовлетворялось одно из условий оптимальности), либо приближенное с верхней оценкой отклонения от оптимального.

Полученные в результате исследования свойств задачи признаки оптимальности ее решения справедливы и эффективны как в случае приближенных алгоритмов, так и в случае реализации экспоненциальной составляющей алгоритма.

*Структура экспоненциальной составляющей ПДС-алгоритма.*

Экспоненциальный подалгоритм состоит из следующих основных блоков [1]:

Блок I. Построение начального расписания $\sigma_0 \in \Psi_P$. Если $\Omega_\Sigma(\sigma_0) = 0$, то расписание оптимально, конец. Иначе: 1) если $R_\Sigma(\sigma_0) \ge \Delta_\Sigma(\sigma_0)$, переходим к выполнению блока II; 2) если $R_\Sigma(\sigma_0) < \Delta_\Sigma(\sigma_0)$, переходим к блоку III.

Блок II. Построение равномерного расписания $\sigma \in \Psi(\sigma_P)$ с числом запаздывающих заданий на каждом приборе $i$, равном $L(\sigma) = L(\sigma_0) - 1$, где $L(\sigma_0)$ – максимальное число запаздывающих заданий на одном приборе в расписании $\sigma_0$. Если такое расписание найдено, то оно оптимально, конец. Иначе идем на блок IV.

Блок III. Построение равномерного расписания $\sigma \in \Psi(\sigma_P)$ с числом запаздывающих заданий на каждом приборе, равным $L(\sigma) = L(\sigma_0)$.

Блок IV. Построение расписания $\sigma \in \Psi(\sigma_P)$, для которого $\Omega_\Sigma(\sigma) < \Omega_\Sigma(\sigma_0)$. Если $\Omega_\Sigma(\sigma) = 0$, то расписание оптимально, конец; иначе идем на выполнение блока V.

Блок V. Построение расписания $\sigma \in \Psi_{PS}$, для которого $\Omega(\sigma) < \Omega(\sigma_0)$. Конец.

Каждый блок включает только улучшающие перестановки заданий между приборами [1]. После выполнения каждого блока проверяются признаки оптимальности получаемых решений, и если они выполняются, то получено оптимальное решение, иначе переходим к следующему блоку. Такие процедуры сначала выполняются для отдельных заданий, а затем для групп заданий. С этой целью формируются группы, включающие два, три и более задания. В результате работы алгоритма либо получаем оптимальное решение задачи, либо алгоритм прекращает работу после заданного числа итераций или заданного времени работы и определяется оценка отклонения полученного решения от оптимального.

На основании приведенных теоретических результатов в [1] разработан эффективный ПДС-алгоритм решения задачи МСЗП. Алгоритм состоит из двух этапов: Этап 1 (Алгоритм А0) – построение начального расписания $\sigma^{\text{уп}}$. Этап 2 (Алгоритм А) – построение оптимального расписания $\sigma^*$.

*Описание Алгоритма А0.*

1. Пронумеруем задания множества $J = \{1, 2, ..., n\}$ по неубыванию значений $l_j$.

2. Пронумеруем приборы по неубыванию значений $T_i$.

3. Полагаем $T_i^{осв} = T_i \ \forall i = \overline{1, m}$.

4. Выбираем задание $j$ с минимальным $l_j$ из неназначенных заданий и назначаем на прибор $i$ с минимальным временем освобождения $T_i^{осв}$.

5. Определяем новое время освобождения прибора $i$: $T_i^{осв} = T_i^{осв} + l_j$.

6. Если распределены на выполнение все задания, конец алгоритма. Иначе переход на п. 4.

Обозначим полученное расписание через $\sigma^{\text{уп}}$.

В Алгоритме А используются следующие типы перестановок.

*Перестановка* 1*P*-0*P*-Δ. С прибора $h \in I_\Delta(\sigma)$ на прибор $r \in I_R(\sigma)$ перемещается задание $j$ такое, что $l_j \ge \Delta h(\sigma)$, $l_j \le R_r(\sigma)$.

*Перестановка* 1*P*-0*P*-*R*Δ. С прибора $h \in I_\Delta(\sigma)$ на прибор $r \in I_R(\sigma)$ перемещается задание $j$ такое, что $l_j < \Delta h(\sigma)$, $l_j > R_r(\sigma)$.

*Перестановка* 1*P*-0*P*-*R*. С прибора $h \in I_\Delta(\sigma)$ на прибор $r \in I_R(\sigma)$ перемещается задание $j$ такое, что $l_j \le \Delta h(\sigma)$, $l_j \le R_r(\sigma)$.

*Описание Алгоритма А.*

*Блок I. Построение начального расписания.*

1. Построение начального расписания $\sigma^{уп} \in \Psi_P$ по Алгоритму А0.

2. Определяем $\Omega_\Sigma(\sigma^{уп})$. Если $\Omega_\Sigma(\sigma^{уп}) = 0$, то расписание $\sigma^{уп}$ оптимально, конец алгоритма. В противном случае переходим к пункту 3.

3. Если $R_\Sigma(\sigma) \geq \Delta_\Sigma(\sigma)$, выполняем пункт 4. Иначе полагаем $\sigma = \sigma^{уп}$, переходим к п. 9.

*Блок II. Построение равномерного расписания $\sigma \in \Psi(\sigma_P)$ с числом запаздывающих заданий на каждом приборе i, равном $L(\sigma) = L(\sigma^{уп}) - 1$.*

4. Полагаем $\sigma = \sigma^{уп}$, $h = 1$, $f = f(\sigma)$, где $f$ – количество приборов с большим числом запаздывающих заданий.

5. Если для прибора $h$ возможна перестановка $1P\text{-}0P\text{-}\Delta$, то выполняем ее. Получаем расписание $\sigma'$. Переходим к п. 6, в случае отсутствия перестановки переходим к п. 7.

6. Полагаем $h = h + 1$. Если $h \leq f$, переходим к п. 5, иначе к п. 8.

7. Если для прибора $h$ возможна перестановка $1P\text{-}0P\text{-}R$, то выполняем ее. Переходим к п. 6.

8. Если $\Omega_\Sigma(\sigma) = 0$, то реализовалась полиномиальная составляющая алгоритма, расписание $\sigma$ оптимально. Определяем значение оптимизируемого функционала. Иначе определяем значение оценки $W$ отклонения функционала от оптимального, конец алгоритма.

*Блок III. Построение равномерного расписания $\sigma \in \Psi(\sigma_P)$ с числом запаздывающих заданий на каждом приборе i, равном $L(\sigma) = L(\sigma^{уп})$.*

9. Полагаем $r = f(\sigma) + 1$.

10. Если для прибора $r$ возможна перестановка $1P\text{-}0P\text{-}R\Delta$, выполняем ее. Получаем расписание $\sigma'$, переходим к п. 11. В противном случае переходим к п. 12.

11. Полагаем $r = r + 1$. Если $r \leq m$, идем на п. 10, иначе на п. 13.

12. Если для прибора $r$ возможна перестановка $1P\text{-}0P\text{-}R$, выполняем ее, получаем расписание $\sigma'$. Переходим к п. 11.

13. Если $\Omega_\Sigma(\sigma) = 0$, то реализовалась полиномиальная составляющая алгоритма, расписание $\sigma$ оптимально. Определяем значение оптимизируемого функционала. Иначе определяем значение оценки $W$ отклонения функционала от оптимального, конец алгоритма.

Трудоемкость алгоритма А определяется полиномом $O(mn \log n)$.

**Исследование свойств задачи МСЗПО**

Построим начальное расписание $\sigma^{уп} \in \Psi_P$ по Алгоритму А0. Условно разобьем полученное расписание $\sigma^{уп}$ на $\sigma^1$ и $\sigma^2$, где $\sigma^1$ – расписание выполнения заданий на приборах, для которых $T_i < d$, а $\sigma^2$ – расписание выполнения заданий на приборах, для которых $T_i \geq d$.

*Утверждение* 10. Максимальная разность количества запаздывающих заданий на приборах в расписании $\sigma^1$ не превышает единицы.

*Утверждение* 11. Количество запаздывающих заданий на каждом из приборов, для которых $T_i < d$, больше или равно количеству запаздывающих заданий на каждом из приборов, для которых $T_i \geq d$.

Справедливость утверждений 10 и 11 основана на алгоритме построения последовательности $\sigma^{уп}$.

*Теорема* 9. В расписании $\sigma^{уп}$ не существует перестановок заданий, выполняемых между приборами $i \in \sigma^1$ и $i \in \sigma^2$, приводящих к уменьшению значения функционала.

*Доказательство* основано на утверждениях 1–3 и 11, следствии теоремы 1 и известной теореме [3], в соответствии с которой поиск оптимального расписания можно ограничить рассмотрением расписаний, при которых каждый прибор обслуживает задания в порядке возрастания их номеров.

*Следствие теоремы* 1 [3]. Пусть обслуживание заданий $L$-м прибором не может быть начато раньше момента времени $T_L > 0$, $L = \overline{1, M}$. Расписанию $\sigma$, при котором каждое очередное задание $k = 1, 2, \ldots, n$ назначается на обслуживание на тот прибор, который раньше других оказывается свободным, соответствует наименьшее значение суммы времен завершения обслуживания всех заданий.

Рассмотрим расписание на приборах $i_r \in \sigma^1$ и $i_s \in \sigma^2$. Пусть задание $j_k$ выполняется на приборе $i_r$, а задание $j_p$ – на приборе $i_s$, причем задания $j_k$ и $j_p$ принадлежат одному уровню запаздывающих заданий. Для этих заданий выполняется: $l_{j_k} \leq l_{j_p}$, в соответствии с алгоритмом построения последовательности $\sigma^{уп}$. Поменяем эти задания местами, т.е. задание $j_k$ будет выполняться на приборе $i_s$, а задание $j_p$ – на приборе $i_r$. В результате такой перестановки, в соответствии с утверждением 11, значение функционала увеличивается. Перестановки запаздывающих заданий, принадлежащих разным уровням запаздывания, в соответствии со следствием теоремы 1 [3], также приводит к увели-

чению значения функционала. В соответствии с утверждениями 1–3, задания $j \in \sigma^2$ не могут быть перемещены в множества заданий $P$ или $S$. Следовательно, в расписании $\sigma^{уп}$ не существует улучшающих перестановок между последовательностями $\sigma^1$ и $\sigma^2$. Теорема доказана.

Справедлива следующая теорема, доказательство которой очевидно.

*Теорема* 10. Значение функционала по задаче МСЗПО равно сумме значений функционала последовательностей $\sigma^1$ и $\sigma^2$.

### ПДС-алгоритм решения задачи МСЗПО

1. Построение начального расписания $\sigma^{уп}$ по Алгоритму А0.

2. Разбиваем полученное расписание $\sigma^{уп}$ на $\sigma^1$ и $\sigma^2$, где $\sigma^1$ – расписание выполнения заданий на приборах, для которых $T_i < d$, а $\sigma^2$ – расписание выполнения заданий на приборах, для которых $T_i \geq d$.

3. Выполнение ПДС-алгоритма А, приведенного выше, на последовательности $\sigma^1$.

4. Анализ полученного решения. Если реализовалась полиномиальная составляющая алгоритма, расписание $\sigma^1$ оптимально, переход на п. 5. Иначе переход на п. 6.

5. Определение значения функционала для последовательности $\sigma^1$. Переход на п. 7.

6. Определение значения функционала и оценки отклонения значения функционала от оптимального для последовательности $\sigma^1$.

7. Определение значения функционала последовательности $\sigma^2$, оптимальной по построению.

8. Определение значения функционала задачи МСЗПО в соответствии с теоремой 10. Конец.

*Пример.* Исходные данные: число приборов $m = 5$, число заданий $n = 25$, общий директивный срок $d = 10$, длительности заданий $l_j$:

| **j** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** | **10** | **11** | **12** | **13** |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $l_j$ | 1 | 1 | 1 | 2 | 2 | 2 | 3 | 3 | 3 | 4 | 4 | 4 | 5 |
| **j** | **14** | **15** | **16** | **17** | **18** | **19** | **20** | **21** | **22** | **23** | **24** | **25** | |
| $l_j$ | 5 | 5 | 6 | 6 | 6 | 7 | 7 | 7 | 8 | 8 | 8 | 9 | |

Значения моментов запуска приборов $T_i$:

| **i** | **1** | **2** | **3** | **4** | **5** |
|---|---|---|---|---|---|
| $T_i$ | 0 | 1 | 2 | 10 | 15 |

В табл. 1 приведено начальное расписание $\sigma^{уп}$ (иллюстрируется рис. 1), в табл. 2 – оптимальное расписание $\sigma^*$ (рис. 2), в таблицах $C_j$ – момент окончания выполнения задания $j$, $F$ – величина запаздывания относительно директивного срока для запаздывающих заданий.

**Табл. 1 – Начальное расписание $\sigma^{уп}$**

| $i$ | $j$ | $l_j$ | $d$ | $C_j$ | $F$ | $i$ | $j$ | $l_j$ | $D$ | $C_j$ | $F$ |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 1 | 1 | 10 | 1 | | 2 | 25 | 9 | 10 | 32 | 22 |
| 4 | 4 | 2 | 10 | 3 | | 3 | 3 | 1 | 10 | 3 | |
| 1 | 7 | 3 | 10 | 6 | | 3 | 6 | 2 | 10 | 5 | |
| 1 | 10 | 4 | 10 | 10 | | 3 | 9 | 3 | 10 | 8 | |
| 1 | 13 | 5 | 10 | 15 | 5 | 3 | 12 | 4 | 10 | 12 | 2 |
| 1 | 17 | 6 | 10 | 21 | 11 | 3 | 16 | 6 | 10 | 18 | 8 |
| 1 | 22 | 8 | 10 | 29 | 19 | 3 | 21 | 7 | 10 | 25 | 15 |
| 2 | 2 | 1 | 10 | 2 | | 4 | 14 | 5 | 10 | 15 | 5 |
| 2 | 5 | 2 | 10 | 4 | | 4 | 18 | 6 | 10 | 21 | 11 |
| 2 | 8 | 3 | 10 | 7 | | 4 | 23 | 8 | 10 | 29 | 19 |
| 2 | 11 | 4 | 10 | 11 | 1 | 5 | 19 | 7 | 10 | 22 | 12 |
| 2 | 15 | 5 | 10 | 16 | 6 | 5 | 24 | 8 | 10 | 30 | 20 |
| 2 | 20 | 7 | 10 | 23 | 13 | | | | | | |

$F_\Sigma(\sigma^{уп}) = 169$. Задание 2 с прибора 2 переносим на прибор 3. Получаем расписание $\sigma^*$:

**Табл. 2 – Оптимальное расписание $\sigma^*$**

| $i$ | $j$ | $l_j$ | $d$ | $C_j$ | $F$ | $i$ | $j$ | $l_j$ | $d$ | $C_j$ | $F$ |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 1 | 1 | 10 | 1 | | 3 | 2 | 1 | 10 | 3 | |
| 4 | 4 | 2 | 10 | 3 | | 3 | 3 | 1 | 10 | 4 | |
| 1 | 7 | 3 | 10 | 6 | | 3 | 6 | 2 | 10 | 6 | |
| 1 | 10 | 4 | 10 | 10 | | 3 | 9 | 3 | 10 | 9 | |
| 1 | 13 | 5 | 10 | 15 | 5 | 3 | 12 | 4 | 10 | 13 | 3 |
| 1 | 17 | 6 | 10 | 21 | 11 | 3 | 16 | 6 | 10 | 19 | 9 |
| 1 | 22 | 8 | 10 | 29 | 19 | 3 | 21 | 7 | 10 | 26 | 16 |
| 2 | 5 | 2 | 10 | 3 | | 4 | 14 | 5 | 10 | 15 | 5 |
| 2 | 8 | 3 | 10 | 6 | | 4 | 18 | 6 | 10 | 21 | 11 |
| 2 | 11 | 4 | 10 | 10 | | 4 | 23 | 8 | 10 | 29 | 19 |
| 2 | 15 | 5 | 10 | 15 | 5 | 5 | 19 | 7 | 10 | 22 | 12 |
| 2 | 20 | 7 | 10 | 22 | 12 | 5 | 24 | 8 | 10 | 30 | 20 |
| 2 | 25 | 9 | 10 | 31 | 21 | | | | | | |

$F_\Sigma(\sigma^*) = 168$. Полученное расписание оптимально, т.к. $R_\Sigma(\sigma^1) = 0$. Построено равномерное расписание для $\sigma^1$ (приборы 1–3).

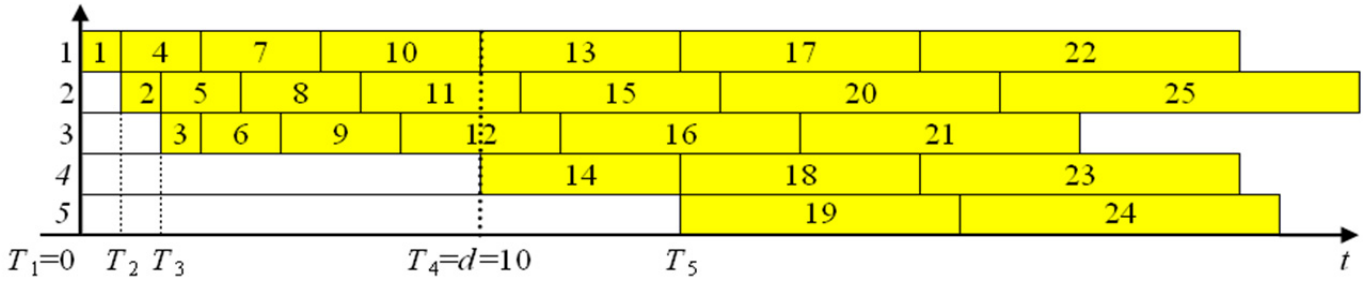



***Рис. 1 – Начальное расписание $\sigma^{уп}$***

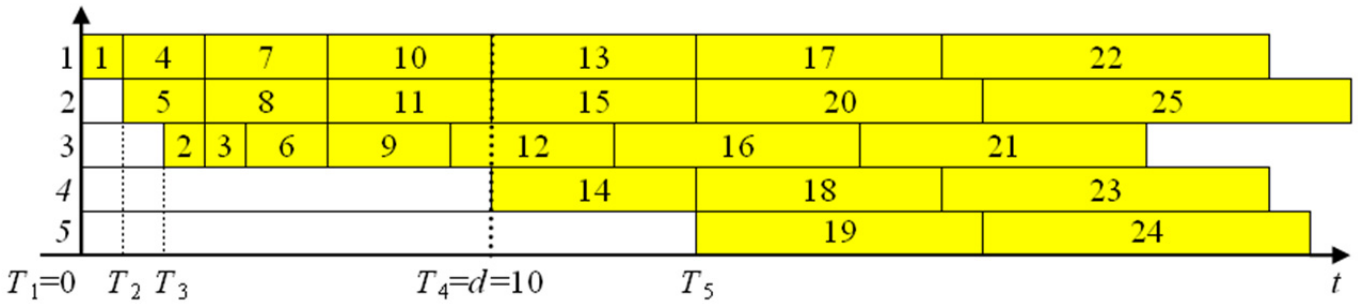



***Рис. 2 – Оптимальное расписание*** $\sigma^*$

## Выводы

Приведены основные теоретические результаты и ПДС-алгоритм решения задачи для случая, когда моменты запуска приборов одинаковы [1]. Исследованы теоретические свойства задачи МСЗПО. Получены признаки оптимальности полиномиальной составляющей ПДС-алгоритма, оценка отклонения от оптимального для экспоненциальной составляющей, приведен ПДС-алгоритм решения задачи, включающий ПДС-алгоритм решения задачи МСЗП с трудоемкостью O($mn$ log $n$). Приведен пример.

## Список литературы

1. Згуровский М.З., Павлов А.А. Принятие решений в сетевых системах с ограниченными ресурсами: Монография.– К.: Наукова думка. – 2010. – 573 с.
2. Гери М.Р., Джонсон Д.С. Вычислительные машины и труднорешаемые задачи. – М.: Мир, 1982. – 416 с.
3. Танаев В.С., Шкурба В.В. Введение в теорию расписаний. – М.: Наука, 1975.– 256 с.

УДК 004.492

*КАЛІНОВСЬКИЙ Я.О.,*
*БОЯРІНОВА Ю.Є.,*
*ХІЦКО Я.В.,*
*ГОРОДЬКО Н.О.*

# МНОЖИННІСТЬ НЕКАНОНІЧНИХ ГІПЕРКОМПЛЕКСНИХ ЧИСЛОВИХ СИСТЕМ СКІНЧЕННОЇ ВИМІРНОСТІ

У статті розглядається спосіб отримання безлічі неканонічних гіперкомплексних систем методом переходу від нескінченновимірної гіперкомплексної системи до скінченновимірних гіперкомплексних систем різного вигляду, залежно від правил множення і методу факторизації. Системи, знайдені запропонованим методом, починаючи з третьої розмірності, є неканонічними. Отримані системи кратній розмірності можна повторно факторизувати і отримати гіперкомплексну числову систему розмірності в два рази меньше.

The subject of the article is a method of obtaining a list of non-canonical hypercomplex systems by the transition from the infinite to the finite hypercomplex number system of different types, depending on the rules of multiplication and factorization method. Systems that have been found by the proposed method, starting from the third dimension are non-canonical. The resulting systems of multiple dimensions can be re-factored to obtain hypercomplex system of dimension in half.

## Вступ

Гіперкомплексна форма представлення даних є широко використовуваної для підвищення ефективності алгоритмів в різних галузях науки і техніки[1]. Найчастіше використовуються канонічні гіперкомплексні числові системи (ГЧС), такі як системи комплексних, подвійних, дуальних чисел, кватерніонів та ін [2]. У деяких задачах можна використовувати ще і різні неканонічні гіперкомплексні числові системи для того, щоб підвищити ефективність алгоритмів, або, як наприклад, в криптографічних задачах, підвищити складність злому даних зловмисником, - залежно від обраної числової системи [3-4].

## Аналіз існуючих методів отримання нових гіперкомплексних числових систем

Ознака канонічності гіперкомплексної числової системи визначається добутком пар базисних елементів: якщо всі ці добутки дорівнюють одному з базисних елементів з коефіцієнтом з множини $\{-1;0;+1\}$, то така система називається канонічною. Якщо ж хоч один добуток базисних елементів є сумою двох або більшої кількості доданків і / або з коефіцієнтом, який за межами множини $\{-1;0;+1\}$, то така гіперкомплексна числова система називається неканонічною [1].

Для дослідження можливості застосування неканонічних ГЧС важливим питанням є побудова таких систем. У самому загальному випадку, побудова неканонічних ГЧС, здійснюється за допомогою перебору структурних констант при базисних елементах в комірках таблиці множення. При цьому встановлюється обмеження на діапазон значень структурних констант, наявність одиничного елемента, лінійну незалежність базисних елементів. Також, відомий метод отримання списку неканонічних ГЧС, ізоморфних конкретній системі, за допомогою перебору коефіцієнтів у системі лінійних рівнянь [4-5].

## Метод факторизації нескінченновимірної гіперкомплексної числової системи

Розглянемо такий підхід отримання безлічі неканонічних гіперкомплексних систем, як перехід від нескінченновимірної гіперкомплексної системи [6] до скінчено-вимірної гіперкомплексної системи різного вигляду, залежно від правил множення і методу факторизації.

Нехай $\Gamma$ - дискретна зліченна множина з нескінченним базисом $\{e_i\}, i = 1,....,n,....$. На ньому введена операція інволюції * така, що

$$*: \Gamma \to \Gamma, \text{ для } \forall e_i \in \Gamma, e_i^* \in \Gamma. \quad (1)$$

Існує елемент базису $e_1 \in \Gamma$ такий, що

$$e_1 \cdot e_i = e_i \cdot e_1 = e_i, \ (e_i^*)^* = e_i \quad (2)$$

Операція множення (згортка) у множині проводиться за наступним правилом:

$$e_i \cdot e_j = \sum_{k=1}^{\infty} C_{ij}^k e_k \quad (3)$$

Множина є гіперкомплексною системою, яка може бути як дискретною, так і безперервною, при виконанні умов (1)-(3).

Перехід від нескінченновимірної гіперкомплексної системи до конечновимірної гіперкомплексної числової системи будемо здійснювати з врахуванням умов коммутативности і позитивності структурних констант:

$$C_{ij}^{k} \geq 0; \quad C_{ii^*}^{1} > 0; \quad C_{ij}^{k} = C_{ki^*}^{j} \geq 0.$$

Сформуємо нескінченновимірну гіперкомплексну систему, від якої пізніше за допомогою факторизації по конкретній підгрупі будемо отримувати гіперкомплексні числові системи кінцевої розмірності.

Задана група $Z = \{-\infty, \infty\}$. На ній обрана деяка підгрупа автоморфізмів $V = \{-1,1\} \in Aut\, Z$, що для $\forall n \in Z$, виконуються

$$1(n) = n \in Z, -1(n) = -n \in Z.$$

Факторизуємо групу $Z$ по підгрупі V та отримаємо множину $Z/V = \Gamma = N \cup \{0\}$. Визначимо для цієї множини правило множення базисних елементів (згортку).

Якщо для групи $Z$ згортка має вигляд

$$\sigma_n \cdot \sigma_m = \sigma_{n+m} \tag{4}$$

або, враховуючи властивості групи $Z$, можна записати $n \cdot m = n + m$.

Тоді згортка для $Z/V = \Gamma$ буде мати вигляд:

$$\begin{aligned} n \cdot m &= \{n,-n\} \cdot \{m,-m\} = \\ &= (n+m) + (m-n) + (-n-m) + (n-m) = \\ &= \{(n+m),-(n+m)\} + \{(n-m),-(n-m)\} \end{aligned}$$

З властивостей згортки та виразу (4) вірне співвідношення:

$$\frac{1}{2}(\sigma_n \cdot \sigma_{-n}) = \sigma_n \tag{5}$$

Якщо врахувати вираз (5), отримаємо згортку:

$$\begin{aligned} \sigma_n \cdot \sigma_m &= \frac{1}{2}(\sigma_n + \sigma_{-n}) \cdot \frac{1}{2}(\sigma_m + \sigma_{-m}) = \\ &= \frac{1}{4}(\sigma_{n+m} + \sigma_{-(n+m)}) + (\sigma_{|n-m|} + \sigma_{-|n-m|}) \end{aligned} \tag{6}$$

Тоді вигляд згортки (4) з врахуванням співвідношення (6) має вигляд:

$$\sigma_n \cdot \sigma_m = \frac{1}{2}(\sigma_{n+m} + \sigma_{|n-m|}) \tag{7}$$

Маючи нескінченновимірну гіперкомплексну систему і згортку, обираємо підгрупи, по яких будемо здійснювати факторизацію.

Можна обирати підгрупи $\{A_2, A_3, A_4, \ldots\}$, де

$A_2 = \{a_k : a_k = 2k, k \in \Gamma\}$,

$A_3 = \{a_k : a_k = 3k, k \in \Gamma\}$,

$A_4 = \{a_k : a_k = 4k, k \in \Gamma\}$ і т.д.

При цьом слід зазначити, что кожна така підгрупа є нескінченновимірною і число таких підгруп зліченна кількість.

(4)

## Аналіз отриманих запропонованим методом гіперкомплексних числових систем

Після факторизації гіперкомплексної системи $\Gamma = N \cup \{0\}$ по підгруппі $A_2 = \{a_k : a_k = 2k, k \in \Gamma\}$ зі згорткою (7) отримаємо скінченновимірну гіперкомплексну числову систему $G_2 = \Gamma/A_2 = \{e_1, e_2\}$ другої розмірності, а саме, систему подвійних чисел, таблиця множення якої має вигляд (рис.1).

| | |
|---|---|
| $e_1$ | $e_2$ |
| $e_2$ | $e_1$ |

***Рис.1. Таблиця множення ГЧС 2-ї вимірності***

Надалі, збільшуючи розмірність, отримуємо неканонічні ГЧС.

При факторизації гіперкомплексної системы $\Gamma = N \cup \{0\}$ по підгрупі $A_3 = \{a_k : a_k = 3k, k \in \Gamma\}$ зі згорткою(7) отримаємо ГЧС $G_3 = \Gamma/A_3 = \{e_1, e_2, e_3\}$ третьої розмірності. Таблиця множення отриманої скінченновимірної гіперкомплексної числової системи буде мати вигляд (рис.2):

| | | |
|---|---|---|
| $e_1$ | $e_2$ | $e_3$ |
| $e_2$ | $\frac{1}{2}(e_1 + e_3)$ | $\frac{1}{2}(e_1 + e_2)$ |
| $e_3$ | $\frac{1}{2}(e_1 + e_2)$ | $\frac{1}{2}(e_1 + e_2)$ |

***Рис.2. Таблиця множення ГЧС 3-ї розмірності***

Після факторизації гіперкомплексної системы $\Gamma = N \cup \{0\}$ по підгрупі $A_4 = \{a_k : a_k = 4k, k \in Q\}$ зі згорткою вигляду (7) отримаємо гіперкомплексну числову систему $G_4 = \Gamma/A_4 = \{e_1, e_2, e_3, e_4\}$ четвертої розмірності (рис.3)

| | | | |
|---|---|---|---|
| $e_1$ | $e_2$ | $e_3$ | $e_4$ |
| $e_2$ | $\frac{1}{2}(e_1+e_3)$ | $\frac{1}{2}(e_2+e_4)$ | $\frac{1}{2}(e_1+e_3)$ |
| $e_3$ | $\frac{1}{2}(e_2+e_4)$ | $e_1$ | $e_2$ |
| $e_4$ | $\frac{1}{2}(e_1+e_3)$ | $e_2$ | $\frac{1}{2}(e_1+e_3)$ |

***Рис.3. Таблиця множення ГЧС 4-ї розмірності***

Так як в ГЧС четвертої розмірності, що отримано, є закономірність, яка полягає в наступному: $e_1 \cdot e_1 = e_1, e_3 \cdot e_3 = e_1$, то можна здійснити повторну факторизацію по підмножині $\{e_1, e_3\}$.

Повторна факторизація $G_4 / \{e_1, e_3\}$ призводить до гіперкомплексної числової системи другої розмірності з таблицею множення(рис.4).

| | |
|---|---|
| $e_1$ | $e_2$ |
| $e_2$ | $\frac{1}{2}(e_1+e_2)$ |

***Рис.4. Таблиця множення неканонічної ГЧС 2-ї розмірності***

Після факторизації гіперкомплексної системи $\Gamma = N \cup \{0\}$ по підгрупі $A_5 = \{a_k : a_k = 5k, k \in \Gamma\}$ з правилами множення вигляду (7) отримаємо скінченновимірну гіперкомплексну числову систему $G_5 = \Gamma / A_5 = \{e_1, e_2, e_3, e_4, e_5\}$ п'ятої розмірності з відповідною таблицею множення (рис.5).

| | | | | |
|---|---|---|---|---|
| $e_1$ | $e_2$ | $e_3$ | $e_4$ | $e_5$ |
| $e_2$ | $\frac{1}{2}(e_1+e_3)$ | $\frac{1}{2}(e_2+e_4)$ | $\frac{1}{2}(e_3+e_5)$ | $\frac{1}{2}(e_1+e_4)$ |
| $e_3$ | $\frac{1}{2}(e_2+e_4)$ | $\frac{1}{2}(e_1+e_5)$ | $\frac{1}{2}(e_1+e_2)$ | $\frac{1}{2}(e_2+e_3)$ |
| $e_4$ | $\frac{1}{2}(e_3+e_5)$ | $\frac{1}{2}(e_1+e_2)$ | $\frac{1}{2}(e_1+e_2)$ | $\frac{1}{2}(e_2+e_3)$ |
| $e_5$ | $\frac{1}{2}(e_1+e_4)$ | $\frac{1}{2}(e_2+e_3)$ | $\frac{1}{2}(e_2+e_3)$ | $\frac{1}{2}(e_1+e_4)$ |

***Рис.5 Таблиця множення неканонічної ГЧС 5-ї розмірності***

Факторизуемо гіперкомплексну систему $\Gamma = N \cup \{0\}$ по підгрупі $A_6 = \{a_k : a_k = 6k, k \in \Gamma\}$ аналогічно попереднім підгрупам та отримаємо ГЧС $G_6 = \Gamma / A_6 = \{e_1, e_2, e_3, e_4, e_5, e_6\}$ шостої розмірності. Згортка ГЧС, що отримано, з врахуванням (7) буде мати вигляд (рис.6)

| | | | | | |
|---|---|---|---|---|---|
| $e_1$ | $e_2$ | $e_3$ | $e_4$ | $e_5$ | $e_6$ |
| $e_2$ | $\frac{1}{2}(e_1+e_3)$ | $\frac{1}{2}(e_2+e_4)$ | $\frac{1}{2}(e_3+e_5)$ | $\frac{1}{2}(e_4+e_6)$ | $\frac{1}{2}(e_1+e_5)$ |
| $e_3$ | $\frac{1}{2}(e_2+e_4)$ | $\frac{1}{2}(e_1+e_5)$ | $\frac{1}{2}(e_2+e_6)$ | $\frac{1}{2}(e_1+e_3)$ | $\frac{1}{2}(e_2+e_4)$ |
| $e_4$ | $\frac{1}{2}(e_3+e_5)$ | $\frac{1}{2}(e_2+e_6)$ | $e_1$ | $e_2$ | $e_3$ |
| $e_5$ | $\frac{1}{2}(e_4+e_6)$ | $\frac{1}{2}(e_1+e_3)$ | $e_2$ | $\frac{1}{2}(e_1+e_3)$ | $\frac{1}{2}(e_2+e_4)$ |
| $e_6$ | $\frac{1}{2}(e_1+e_5)$ | $\frac{1}{2}(e_2+e_4)$ | $e_3$ | $\frac{1}{2}(e_2+e_4)$ | $\frac{1}{2}(e_1+e_5)$ |

***Рис.6. Таблиця множення ГЧС 6-ї розмірності***

Так як в цій ГЧС, що отримано, є закономірність, яка полягає в наступному $e_1 \cdot e_1 = e_4 \cdot e_4 = e_1, e_1 \cdot e_2 = e_4 \cdot e_5 = e_2$, $e_1 \cdot e_3 = e_4 \cdot e_6 = e_3$, то можна здійснити повторну факторизацію по підмножині $\{e_1, e_4\}$. В результаті отримаємо ще одну ГЧС третьої розмірності зі згорткою (рис.7).

| | | |
|---|---|---|
| $e_1$ | $e_2$ | $e_3$ |
| $e_2$ | $\frac{1}{2}(e_1+e_3)$ | $\frac{1}{2}(e_1+e_2)$ |
| $e_3$ | $\frac{1}{2}(e_1+e_2)$ | $\frac{1}{2}(e_1+e_3)$ |

***Рис.7. Таблиця множення ГЧС 3-ї розмірності***

Аналогічно запропонованим методом можна отримати ГЧС більшої розмірності. Як видно з наведених вище прикладів, отримання ГЧС по підгрупах вигляду

$A_6 = \{a_k : a_k = 6k, k \in Q\}$,

$A_8 = \{a_k : a_k = 8k, k \in Q\}$ и т.д., тобто по підгрупам, елементи яких є парними, можна повторно факторизувати та отримувати ГЧС розмірності в 2 рази менше.

## Висновки

Отже, у статті розглянуто метод переходу від нескінченновимірної гіперкомплексної системи до скінченновимірної неканонічної ГЧС методами факторизації.

Показано факторизацію нескінченновимірної гіперкомплексної системи по нескінченновимірним підгрупам, використовуючи згортку ви-

гляду $\sigma_n \cdot \sigma_m = \frac{1}{2}(\sigma_{n+m} + \sigma_{|n-m|})$, з метою отримання скінченномірних гіперкомплексних числових систем, які, починаючи з третьої розмірності, є неканонічними.

Отримані системи парної розмірності (2, 4, 6, 8 ...), можна повторно факторизуватиь і отримати гіперкомплексну числову систему розмірності в два рази менше.

## Список літератури


1. Синьков М.В. Гіперкомплексні числові системи: основи теорії, практичні використання, бібліографія / М.В. Синьков, Ю.Є. Боярінова, Я.О. Каліновський, Синькова Т.В., Федоренко О.В. – К. ІПРІ, 2009. – 44 с. – ( Препр. /НАН України, Інт пробл. реєстрації інформац, 2009).
2. Кантор И. Л. Гиперкомплексные числа/ Кантор И. Л., Солодовников А.С. – М.: Наука, 1973. – 144 с.
3. Синьков М.В. Развитие задачи разделения секрета / Синьков М.В., Бояринова Ю.Е., Калиновский Я.А., Трубников П.В. // Реєстрація, зберігання і обробка даних. – 2003. – Т. 5 – № 4. – С. 90–96.
4. Одарич Я.В. Вычисления в неканонических гиперкомплексных числовых системах / Одарич Я.В., Наливайчук Е.Ю., Наливайчук Н.В. // Радіоелектронні і комп'ютерні системи. –2010. –№5. – С. 75-78.
5. Калиновский Я.В. Высокоразмерные изоморфные гиперкомплексные числовые системы и их использование для повышения эффективности вычислений / Калиновский Я.В., Бояринова Ю.Е – Киев: Инфодрук, 2012. – 82 с.
6. Одарич Я.В. Процедура перечисления гиперкомплексных числовых систем методом линейных преобразований./ Одарич Я.В. // Реєстрація, зберігання і обробка даних. – 2008. – Т. 6 – № 2. – С. 107–112.
7. Березанский Ю. М. Гармонический анализ в гиперкомплексных системах / Березанский Ю. М. , Калюжный А. А. – К.: Наукова думка, 1992. –352 с.

УДК 004.724.4(045)

*КУЛАКОВ Ю.А.,*
*УВАРОВА Н.В.*

# АНАЛИЗ АЛГОРИТМОВ ВЫБОРА ПАКЕТА ДЛЯ ЗАПРОСА ПРИ МНОГОАБОНЕНТСКОЙ ПЕРЕДАЧЕ

Статья посвящена анализу различных алгоритмов выбора пакета для передачи в условиях многоабонентской передачи. Авторы рассматривают существующие подходы к выбору пакетов и дают формальное описание четырех основных – случайного, последовательного, неравномерно случайного и с приоритетами. Проведено моделирование работы указанных алгоритмов и оценена задержка получения пакета относительно времени его генерации для разных алгоритмов. Показано, что лучшие показатели имеют алгоритмы, содержащие как случайную составляющую, так и механизм приоритезации первоочередных пакетов.

The article is devoted to the problem of selection the package for the transmission – so called data scheduling for the multicast systems. According to the article, data scheduling plays important role in p2p multicasting, allowing utilization of available bandwidth. Authors provide information about existing methods and select four basic of them – random, sequential, with non-uniform distribution and with priorities. The simulation of specified algorithm is performed and the delay of package receiving is analyzed. It is shown, that algorithms, that combine the randomness with priority of the first packages, have best results.

## Вступление

Увеличение объемов использования интернета поставило задачу передачи информации значительному количеству получателей. Решением этой проблемы может быть использование многоабонентской доставки информации.

Распространение потокового контента при помощи технологии Peer-To-Peer может быть жизнеспособным решением для обеспечения масштабированности и равномерного распределения нагрузки. Тем не менее нужно для решить целый ряд проблем. Как правило, при распространении потоковых данных, источник данных генерирует определенное число блоков фиксированного или переменного размера, которые должны быть доставлены получателям с фиксированной скоростью и в жесткие сроки. Не полученные вовремя пакеты могут привести к снижению качества, временной или даже полной приостановке воспроизведения [1].

Основной проблемой для p2p передачи есть временные ограничения в условиях динамичности, (так называемой churn), перегруженности сети и ограничениях на общее количество подключений. Необходимым условием тогда есть достижение эффективного использования пропускной ширины канала. Участники сети должны отдавать для нужд системы часть своей пропускной способности за счет передачи уже загруженных пакетов другим участникам [3].

Существует ряд протоколов многоабонентской передачи, которые используют технологию p2p, к ним относят например, Chainsaw, Meshcast, Coolstreaming.

Одним из важных элементов системы многоабонентской передачи алгоритм выбора пакета для передачи (пакета запроса для pull-систем и пакета для передачи для push-систем). Так в системе Chainsaw используется простейший алгоритм - случайный, Coolsteaming - с приоритетами. В [2] исследуется недостатки случайного алгоритма и исследуется разработка алгоритма с приоритетами для обеспечения непрерывности трансляции.

Случайный алгоритм хорошо обеспечивает динамичность за счет использования всех преимуществ топологии и выбора случайного пакета, но может привести к потере пакетов. Поэтому необходимо модифицировать алгоритм так, чтобы не уменьшить устойчивость алгоритма, но увеличить его шансы выбрать правильный пакет [4].

Другим крайним вариантом является выбор не случайного пакета, а первого имеющегося с наименьшим порядковым номером, что может привести к неравномерной нагрузке на узлы, так как порядок запроса пакетов будет одинаковым для всех узлов. Необходимо выбрать средний вариант - между выбором пакетов с равномерной вероятностью и исключения вероятности по выбору пакета вообще. Среди вариантов - выбор пакета с вероятностью, или например, в общем случае выбор пакета случайного, а в случае наличия пакетов, которые скоро достигнут своего таймаута - предоставление им преимущества. Задачей этой статьи исследовать как меняются характеристики получения пакетов при использовании различных алгоритмов выбора следующего пакета для передачи.

### Алгоритмы выбора пакета

Следует отметить, что задача эффективного планирования передачи даже в условиях динамических связей между узлами и возможной неоднородностью узлов относится к классу NP-полных и не имеет оптимального решения, которое могло бы работать в условиях ограничения реального времени на принятие решения. Поэтому используются приближенные эвристические алгоритмы, которые работают с достаточной степенью эффективности для конкретных применений и условий.

Рассмотрены следующие алгоритмы выбора следующего пакета для запроса: равномерно случайный, последовательный, неравномерно случайный и с приоритетами.

Случайный выбор пакета запроса, можно формально описать так:

1. все пакеты из окна необходимости узла, имеющиеся у соседей со свободным подключением, расположить в порядке возрастания номеров

2. выбор случайного числа из множества целых чисел - возможных индексов пакетов в этом массиве (обычно это числа от 0 вплоть до длины массива не включительно).

3. выбор пакета из сформированного массива по случайно выбранному индексу.

Последовательный алгоритм:

1. все пакеты из окна необходимости узла, имеющиеся у соседей со свободным подключением, расположить в порядке возрастания номеров.

2. пакет запроса выбирается с наименьшим порядковым номером (то есть тот, который был сгенерирован раньше).

Одним из вариантов модификации выбора пакетов - использование не равномерного распределения для выбору индекса, а более сложного. В условиях наличия таймаута для получения пакета, следует выбирать те распределения, в которых вероятность выбора меньших индексов больше. Конечно, такой способ не гарантирует полного отсутствия потерянных пакетов, но значительно увеличивает вероятность своевременной доставки пакета. Как пример такого распределения было выбрано beta распределения с параметрами $alpha = 1$ $beta = 3$. Замечательное свойство именно beta распределения, что границы его значений четко фиксированы и становят [0, 1], в отличие от, например, экспоненциального распределения. Тогда выбор элемента из массива с этим распределением тривиальная задача.

1. Используя генератор пседвослучайных чисел, который выдает числа с beta-распределение сгенерировать случайное число.

2. Умножить это число на общую длину массива и округлить до ближайшего целого.

3. Выбрать из упорядоченного массива пакетов по полученному индексу.

Другим вариантом является использование обычного случайного алгоритма с исключениями. Есть при обычных условиях, алгоритм работает в обычном режиме, но при определенных условиях наступают особые случаи. Каждый пакет может находиться в трех состояниях в зависимости от промежутка времени, который остался до окончания таймаута. Если этот промежуток меньше некоторого выбранного $t_1$ , то пакет считается потерянным и удаляется из окна необходимости. Этот случай описывает ситуацию, когда осталось слишком мало времени, чтобы вовремя доставить пакет , поэтому нет смысла его запрашивать и тратить время и пропускную способность. Если же оставшийся промежуток времени, находится в границах между $t_2$ и $t_3$ , то это означает , что пакет скоро будет считаться потерянным и он имеет приоритет на запрос. В противном же случае , пакет является обычным.

Тогда алгоритм выбора пакетов будет выглядеть следующим образом.

1. Назначить каждому пакету статус.

2. Очистить окно необходимых пакетов от пакетов со статусом 1.

3. Сформировать список пакетов, содержащихся в окнах доступности соседей со свободными подключениями.

4. Далее разделить пакеты на пакеты со статусом 2 и со статусом 3.

5. Если есть пакеты со статусом 2 - пакет выбирается из этой группы( случайно или по порядку). Пакеты из второй группы выбираются только в случае, когда нет пакетов в первой. Из этой группы пакеты выбираются исключительно случайно.

К достоинствам первых трех алгоритмов можно отнести простоту реализации и скорость работы, к достоинствам алгоритма с приоритетами – возможность адаптировать алгоритм под конкретное применение через манипуляцию параметрами $t_1$, $t_2$, $t_3$.

### Результаты моделирования

Для проверки эффективности предложенных модификаций была разработана программа на языке программирования Python по моделированию виртуальных сетей и сбора статистики состояния узлов. На вход программы подаются параметры системы: количество узлов, количество пакетов в трансляции, а также алгоритм, который следует моделировать. После модели-

рования в качестве результата возвращается пошаговый лог всех событий системы, статистика по загрузке входного и выходного каналов узлов, время получения каждого пакета различными узлами. Среди свойств сети критических для реализации в программе можно выделить следующие: ограничения на ширину канала узла (как на отдачу, так и на загрузку), ограничения на размер буфера (количества пакетов, которые могут одновременно храниться в узле).

Существует несколько вариантов метрик, которые можно использовать для оценки качества работы алгоритмов. Так как основное предназначение многоабонентской рассылки данных в настоящее время это трансляция медиа данных (видео/аудио поток), то критичным есть получение пакета в ограниченный промежуток времени. Для большинства трансляций начальная задержка начала трансляции не критична, хотя для конференций в режиме реального времени такая задержка может быть критична. В этой статье в качестве метрики была выбрана задержка относительно получения первого пакета каждым узлом, не относительно абсолютного начала трансляции.

На рисунках 1, 2, 3 и 4 представлены результаты моделирования трансляции 100 пакетов 10 узлам для разных алгоритов выбора пакета. Для каждого узла показана задержка получения пакета относительно времени получения предыдущего пакета.

По оси абсцисс - порядковый номер пакета, по оси ординат - время его получения каждым узлом. Диагональная линия, показывает время получения пакета источником - то есть время его создания. Чем больше отклонение от этой линии в большую сторону, тем с большей задержкой был получен пакет.

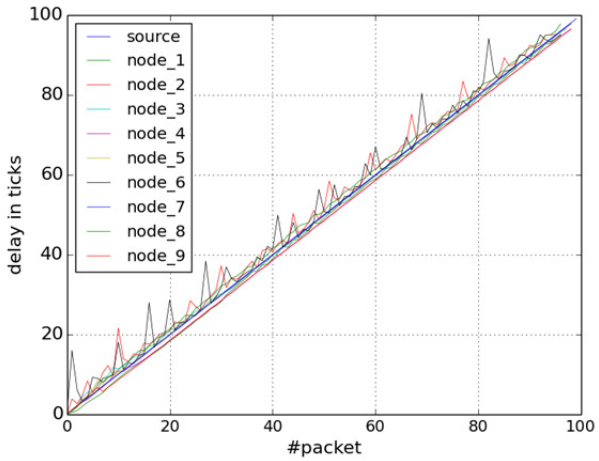



***Рис. 1. Задержка получения пакетов для алгоритма случайного выбора***

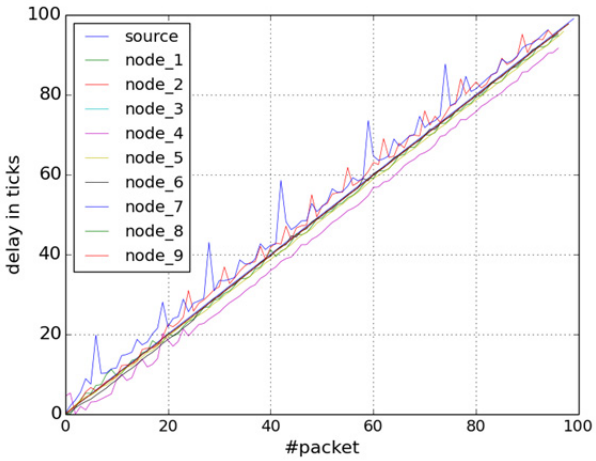



***Рис. 2. Задержка получения пакетов для алгоритма последовательного выбора***

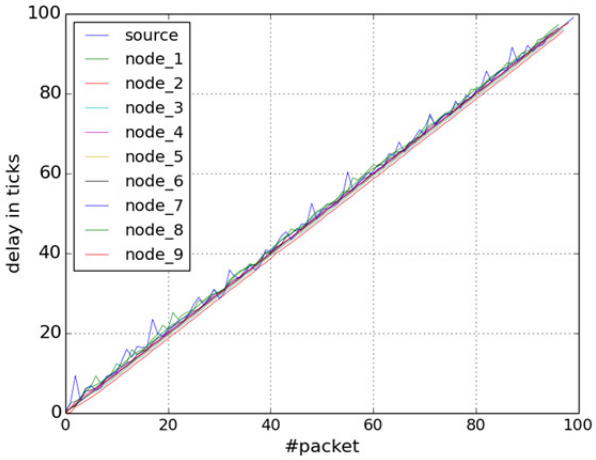



***Рис. 3. Задержка получения пакетов для алгоритма с неравномерным распределением***

Так как для потоковой передачи получения пакета в определенный промежуток времени является критичным, то пакеты, которые были получены с слишком большой задержкой считаются неактуальными и соответственно утраченными.

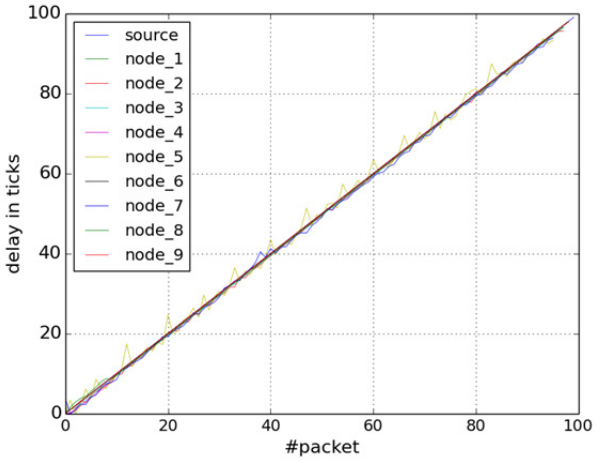



***Рис. 4. Задержка получения пакетов для алгоритма с приоритетами***

В таблице 1 показано начало трансляции для каждого пакета. Видно, что случайные алгоритмы имеют наиболее равномерные результаты, а худший результат у последовательного алгоритма за счет узлов, которые не смогли получить начальные пакеты в заданный промежуток времени.

***Табл. 1. Задержка начала трансляции***

| | **Rand** | **Seq** | **Beta** | **Priority** |
|---|---|---|---|---|
| Mean | 3.23 | 4.01 | 3.23 | 3.66 |
| Std | 1.15 | 3.1 | 1.23 | 2.55 |

## Выводы

Если сравнить задержку пакетов для последовательного и случайного алгоритмов, то можно заметить, что результаты для случайного выбора более одинаковы для разных узлов, чем для последовательного выбора. В случае последовательного выбора существуют узлы , которые постоянно получат пакеты с большим запасом по таймауту, и узлы, которые хронически теряют пакеты. В случае же случайного выбора такая закономерность также присутствует , но значительно меньше выражена. Если у случайного алгоритма, потери пакетов объясняются неоптимальным выбором, то последовательный алгоритм всегда выбирает оптимально, но он ограничен выбором из тех пакетов , которые имеются у его соседей.

Видно, что лучше показали себя более сложные алгоритмы: с неравномерным случайным выбором и приоритетами , которые сочетают случайную составляющую и учитывают необходимость доставить пакеты вовремя.

Выбирая между алгоритмом с beta распределением и с приоритетами, можно отметить, что у beta-алгоритма более быстрый старт трансляции, что может быть критичным для некоторых применений.

Показано, что выбор вероятностного алгоритма дает возможность использовать всю доступную входящую скорость, но увеличивает шансы на потерю пакета. Выбор алгоритма для выбора пакета должен зависеть от параметров системы , в которой планируется применение алгоритма.

## Список литературы

1. Bagheri, Maryam, Ali Movaghar, and Ali Asghar Khodaparast. "Bandwidth Adapted Hierarchical Multicast Overlay. // Computing in the Global Information Technology (ICCGI), 2010 Fifth International Multi-Conference on. IEEE – 2010 – P. 262-267.
2. Kwon, Oh Chan, and Hwangjun Song. Adaptive tree-based P2P video streaming multicast system under high peer-churn rate. // Journal of Visual Communication and Image Representation 24.3 – 2013. – P.203-216.
3. Meng, Xianfu, and Wei Wu. Live P2P Streaming System with High Playback Continuity. // 2014 International Conference on Computer, Communications and Information Technology (CCIT 2014) - Atlantis Press, 2014.
4. Afolabi, Richard O., Aresh Dadlani, and Kiseon Kim. "Multicast scheduling and resource allocation algorithms for OFDMA-based systems: A survey." // Communications Surveys & Tutorials, IEEE 15.1 - 2013 - P.240-254.

УДК 519.6:512.972, 004.27

*КЛИМЕНКО И.А.*

# МЕТОД ОТОБРАЖЕНИЯ ЗАДАЧ НА РЕКОНФИГУРИРУЕМУЮ АРХИТЕКТУРУ ВЫЧИСЛИТЕЛЬНОЙ СИСТЕМЫ

Предложен метод отображения задач на реконфигурируемую архитектуру вычислительной системы и приведена его формализация. Отображение осуществляется на основании требований задачи ко времени вычисления и возможностей реконфигурируемой вычислительной системы с учетом ограничений, определяемых ее архитектурой.

The method of mapping tasks to the reconfigurable architecture of the computer system and its formalization is proposed. Mapping is based on the task requirements to computing time and on the opportunities of reconfigurable computing system within the constraints defined by the architecture.

## 1. Введение

В настоящее время в связи с широким использованием программируемых логических интегральных схем в качестве элементной базы для построения вычислительных систем актуальной становится проблема оптимизации отображения задачи на архитектуру ВС в частности на ее реконфигурируемую часть. При этом возможность физической реконфигурации архитектуры такой системы позволяет предложить различные вычислительные топологии адаптивные к типам и классам решаемых задач. Наиболее явные особенности реконфигурируемой архитектуры оказывают влияние на процесс отображения уже на этапе распараллеливания задачи. На этом этапе нет необходимости придерживаться жесткой заранее фиксированной структуры. Достаточно привести задачу к некоторому промежуточному виду после чего настроить под требования задачи архитектуру системы, для наиболее эффективного ее отображения.

Рассмотренная выше проблема отображения становится актуальной в контексте решения задачи динамической реконфигурации вычислительных систем, построенных на программируемой элементной базе (ПЛИС). Современный класс таких систем, называемый реконфигурируемыми вычислительными системами (РВС), интенсивно исследуется и развивается сообществом высокопроизводительных вычислений с целью дальнейшего повышения производительности суперкомпьютерных и кластерных вычислений. Основная концепция создания РВС обеспечение адаптивности архитектуры согласно требованиям решаемой в данный момент времени задаче при сохранении черт систем широкого использования и обеспечении при этом высокой реальной производительности. Но вместе с чрезвычайной привлекательностью перепрограммирования архитектуры РВС имеют определенный ряд проблем, поиск решений которых обуславливают большую актуальность исследований в данной области. Проблемы с одной стороны связаны с динамическими методами и средствами программирования прикладного уровня для реконфигурируемых архитектур, сложностями низкоуровневого схемотехнического программирования, с другой стороны это проблемы физического уровня реконфигурации, к которым относятся ограничения аппаратного обеспечения, высокие накладные расходы и энергозатраты на реконфигурацию.

## 2. Постановка задачи

На основании вышесказанного в статье предложен метод отображения задач на реконфигурируемую архитектуру вычислительной системы и сделана попытка его формализации. Предложен следующий подход – отображение задачи на реконфигурируемую архитектуру ВС осуществляется на основании заранее требуемых параметров качества решения, которые определяются требованиями со стороны решаемой задачи и возможностями системы с учетом ограничений архитектуры.

Основным требованием со стороны задачи является суммарное время вычисления, которое состоит из времени настройки архитектуры (времени реконфигурации) и времени вычисления.

В качестве ограничений со стороны системы выступают следующие параметры: ограничения аппаратного вычислительного ресурса ПЛИС, минимизация времени реконфигурации, мини-

мизация количества обмена данными на межмодульном уровне.

Любая параллельная задача предполагает операции вычисления и операции обмена данными. Поэтому, для ее представления в реконфигурируемой вычислительной среде необходимо построить такую модель, которая с одной стороны не выходит за пределы поставленных ограничений и при этом имеет минимальное количество связей между уровнями. Данная задача может быть сведена к теории графов.

### 3. Модель задачи

Для описания модели исходной задачи предлагается модель программирования М-задач [1, 2], которая используется для структурирования параллельных программ со смешанной параллельностью. Смешанная параллельность, характерная большинству современных требовательных к продуктивности задач, касается наличия в параллельной программе как параллелизма задач так и параллелизма данных. Множество М-задач, каждая из которых работает над различной частью приложения, объединяется в М-программу. Параллелизм данных позволяет отдельным М-задачам выполнятся в стиле *SPMD* на непересекающемся наборе процессоров, параллелизм задач (стиль *MPMD*) позволяет множеству М-задач выполняться одновременно, обмениваясь данными друг с другом. Преимущество этого подхода, в том что он позволяет увеличивать доступную степень параллелизма, определяя соответствующее строение М-задачи и ограничивая передачу данных в ее пределах. Таким образом, уменьшая коммуникационные издержки и увеличивая масштабируемость.

Значительные преимущества от смешанного параллелизма программ получают многоядерные параллельные вычислительные средства за счет высокой степени параллельности своих архитектур. Такая парадигма многоядерных вычислений может быть расширена до использования в реконфигурируемых параллельных вычислительных системах, если уровень многоядерных узлов представить реконфигурируемыми вычислительными структурами [3]. При этом, по сравнению с многоядерными архитектурами, которые требуют адаптации задачи к жёсткой архитектуре системы с сомнительным достижением максимальной продуктивности, получаем дополнительную степень параллелизма за счет гибкой гетерогенной архитектуры, которая адаптируется к структуре прикладной задачи и позволяет реализовать вычисления с максимальной продуктивностью.

В литературе рассмотрено большое количество различных методов и средств отображения М-задач на архитектуру многоядерных вычислительных структур. Это достаточно актуальная сегодня область в сфере высокопроизводительных вычислений. Однако при использовании в РВС эти методы требуют модификаций с учетом возможностей реконфигурации вычислительных узлов и ограничений, накладываемых архитектурными особенностями РВС.

Параллельные части М-программы представляются макро-графами потоков данных *MDG* (*Macro Dataflow Graphs*) с вершинами, представляющими крупно-модульные М-задачи, и с рёбрами, указывающими на зависимости между этими вершинами (рис.1) [1, 2].

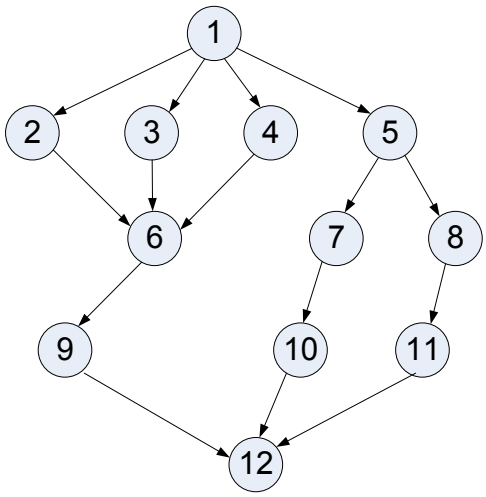



***Рис. 1. Граф М-программы***

В качестве примера рассмотрим граф М-программы, которая представлена графом $G_M = (V, E)$, где $V$ – множество вершин которые представляют собой М-задачи программы, $E$ – множество рёбер, при этом ребро $e \in E$ соединяет М-задачи $M_1$ и $M_2$, если между ними есть отношение по данным или по управлению.

### 4. Метод отображение задач на архитектуру РВС

Для решения задач со смешанной параллельностью в реконфигурируемых вычислительных системах с реализацией многоуровневого параллелизма архитектуры предлагается разбить процесс отображения задачи на несколько этапов, учитывающих возможности и

ограничения каждого уровня параллелизма системы.

Основные этапы метода отображения задачи на реконфигурируемую архитектуру, предлагаемого в данной работе представлены на рис. 2.


***Рис.2. Основные этапы метода отображения задачи архитектуру РВС***

На этапе разбиения графа М-задачи на уровни обеспечиваются такие требования со стороны задачи, как минимизация времени вычисления. Со стороны системы данное требование обеспечивается минимизацией коммуникаций на межмодульном уровне, для достижения чего предлагается такое разбиение графа М-задачи, которое обеспечивает минимальное количество связей между уровнями.

Один из эффективных подходов отображения М-программы на архитектуру мультиядерной вычислительной системы, рассмотренный в работе [1] – *алгоритм отображения по уровням*. В рамках которого граф М-программы делится на уровни независимых М-задач, и каждый уровень отображается один за другим. Предложенный способ используется нами на первом этапе для разбиения графа. Самая большая гибкость для решения задач отображения достигается при использовании как можно меньшего количества уровней. Это может быть реализовано при использовании жадного алгоритма, который работает сверху вниз по графу М-задачи и помещает так много вершин, насколько это возможно, в текущий уровень [1]. Результат такого разбиения представлен на рис. 3. Данный граф $G_M = (V, E)$ будет исходным для дальнейших преобразований.

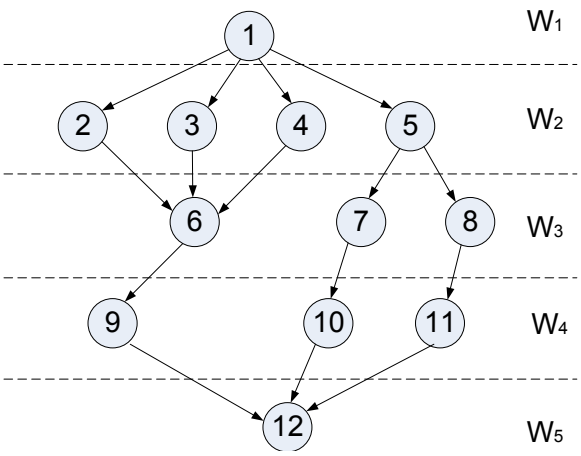



***Рис. 3. Разделение графа М-задачи на уровни***

### 3.1. Кластеризация

Алгоритм отображения по уровням, описанный в работе [1] требует модификации для эффективного использования в реконфигурируемой архитектуре. В связи с этим в работе предлагается объединение нескольких уровней графа $G$ и формирования некоторого множества подграфов – кластеризация. Каждый подграф реализуется на одном и том же наборе конфигураций ПЛИС. Это приведет к локализации операций обмена данными в пределах вычислительного модуля. В то же время для отображения уровней в пределах каждого подграфа на архитектуру вычислительного модуля остается целесообразным реализация отображения по уровням.

Для обеспечения качества сервиса учитываются такие требования со стороны задачи, как минимизация времени вычисления. Со стороны системы данное требование обеспечивается минимизацией коммуникаций на межмодульном уровне, минимизацией коммуникаций на внутримодульном уровне и уменьшением количества последовательных реконфигураций архитектуры. Для достижения чего предлагается такое разбиение графа М-задачи на подграфы, которое обеспечивает минимальное количество связей между кластерами, и минимальное количество уровней подграфа.

Как видно на графе (рис. 3), общий объем связей между М-задачами фиксированный. Согласно модели архитектуры подмножества М-задач отображаются на один/несколько вычислительных модулей, а в их пределах – на од-

ну/несколько конфигураций ПЛИС. Тогда межзадачные связи могут быть поделены на два типа – внешние межмодульные связи и внутримодульные связи между конфигурациями ПЛИС одного модуля. Считаем, что при выполнении кластеризации максимизация коммутаций внутри вычислительного модуля приводит к минимизации межмодульных связей.

Тогда исходный граф $G_M$ разбивается на множество подграфов $C_i \in G_M$ ($i = \overline{1,n}$, где $n$ – количество подграфов), таким образом что бы количество связей внутри подграфа было максимальным [4].

В качестве альтернативного средства может быть предложена кластеризация с минимизацией количества связей между подграфами на базе метода определения минимального сечения подграфа, описанного в работе [5].

Полученный граф представлен на рис. 4 *а*. Для внешнего алгоритма управления, реализуемого центральным процессором граф М-задачи соответствует представленному на рис. 4, *б*. Количество уровней уменьшилось до трех, связи между вершинами соответствуют оставшимся межмодульным связям. Граф М-задачи будет отображаться на архитектуру системы и выполняться согласно алгоритму отображения по уровням. В качестве минимальной вычислительной единицы архитектуры для отображения рассматривается вычислительный модуль. Управление решением подграфа осуществляется соответствующими средствами вычислительного модуля, которые аналогичным образом реализуют алгоритм отображения по уровням.

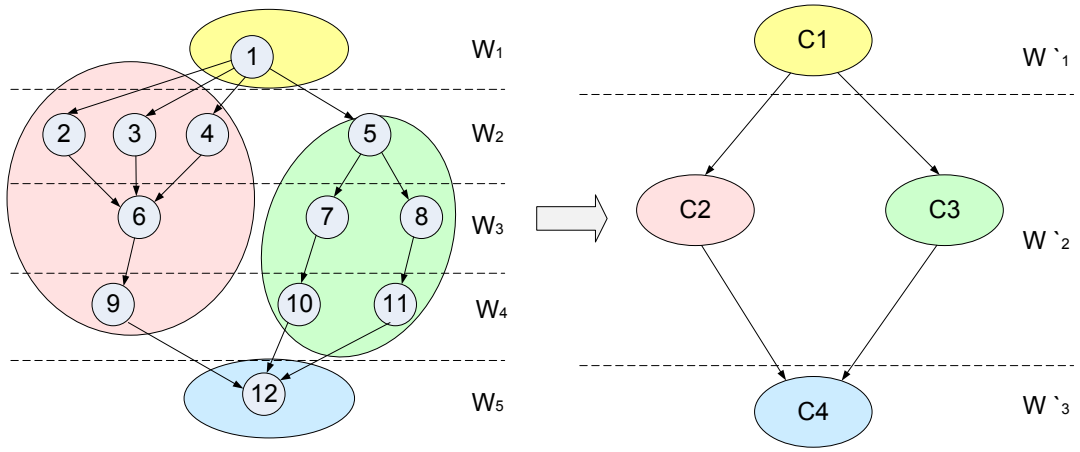



***Рис. 4. Кластеризация графа М-задачи***

Предварительную оценку эффекта от проведенной кластеризации можно оценить на основании параметра выигрыша $P_{COMM}$, который определяет время межзадачного взаимодействия, сохраненное за счет максимизации внутримодульных связей, и выражается следующей формулой [4]:

$$P_{COMM} = \frac{U_{COMM}}{B_{IO}} \times 2,$$

где $U_{COMM}$ – объем межзадачных связей, перешедших на внутримодульный уровень, $B_{IO}$ – доступная пропускная способность ввода\вывода между конфигурациями ПЛИС и основной памятью системы. На данном этапе для упрощения формализации кластеризации мы делаем предположение о том что время внутримодульного взаимодействия минимально и пренебрегаем этой величиной. Такое предположение приемлемо, поскольку топология вычислительного модуля еще не рассматривается и невозможно оценить соотношение коммуникаций между конфигурациями ПЛИС и внутрикристальными взаимодействиями, которые по времени действительно могут быть сведены к нулю. Однако в дальнейшем величина параметра прибыли $P_{COMM}$, должна быть откорректирована с учетом времени внутримодульного обмена для связей $U_{COMM}$ и различных топологий вычислительных модулей.

### 3.2. Оптимизация

Дальнейшая оптимизация структуры кластера локализуется на уровне архитектуры вычислительного модуля. Со стороны задачи учитывается требование минимизации времени вычисления, которое обеспечивается системой

путем уменьшения времени реконфигурации архитектуры, со стороны системы учитываются ограничения аппаратных ресурсов вычислительного модуля за счет введения ограничения на количество вершин каждого уровня подграфа

Не рассматривая на данном этапе топологии вычислительных модулей сделаем некоторые предположения относительно архитектуры вычислительного модуля и введем абстрактный шаблон архитектуры. Это позволит формализовать оптимизацию графа М-задачи и в дальнейшем рассмотреть наиболее эффективные топологии вычислительных модулей для решения различных классов задач. Определим общее количество вычислительного ресурса одного модуля, как некоторую вычислительную площадь $N = \{G, S\}$, состоящую из групп $G$ взаимосвязанных виртуальных вычислительных структур $S$. Количество $g$ групп соответствует количеству конфигураций ПЛИС одного вычислительного модуля, количество $s$ виртуальных вычислительных структур ограничено аппаратными возможностями одной конфигурации ПЛИС. При этом все вычислительные структуры $S_l$ ($l = \overline{1,s}$) однотипны, их количество $s$ в каждой группе одинаково так же, как и количество $g$ групп одинаково во всех вычислительных модулях. Модель обобщенной архитектуры вычислительного узла представлена на рис.5.

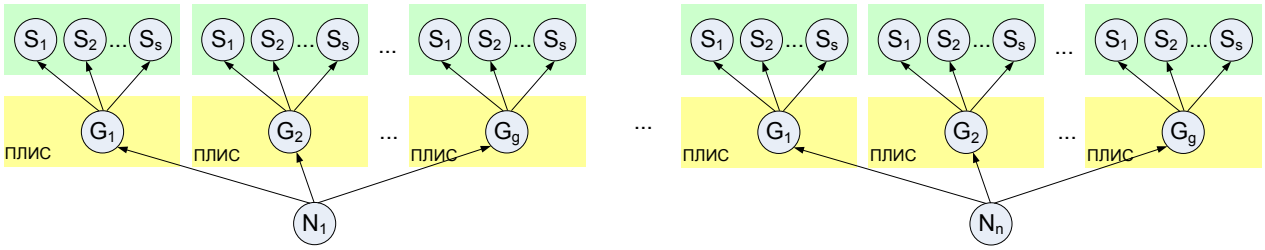



***Рис. 5. Модель архитектуры РВС***

Предопределенное количество вычислительного ресурса одного модуля, равное $(g \times s)$ вычислительных структур, а исходя из этого, и каналов связи между конфигурациями ПЛИС ограничивает количество вершин, которые могут быть размещены на одном уровне подграфа *Ci*. Для достижения цели оптимизации структуры подграфа, согласно ограничениям аппаратных ресурсов, полученный граф может быть трансформирован без нарушения структуры связей – что подразумевает перемещение вершин с одного уровня на другой (рис. 6). С другой стороны, согласно алгоритму отображения по уровням каждый уровень подграфа будет последовательно отображен на архитектуру, которая должна быть предварительно реконфигурирована, и выполнен. Таким образом время выполнения М-задач каждого уровня подграфа будет равно

$$T_{sum\ \max Wj} = (T_{commVCi} + T_{reconfVCi})_{\max}, \qquad (1)$$

где $T_{sum\max Wj}$ – максимальное суммарное время выполнения М-задач, входящих в текущий уровень $W_j$ ($j = \overline{1,w}$, где $w$ – количество уровней), $T_{commVCi}$ – время вычисления и $T_{reconfVCi}$ – время реконфигурации М-задачи соответствующей некоторой вершине $V$ подграфа $C_i$. Согласно этому неограниченное увеличение количества уровней подграфа во время трансформации приведет к значительному увеличению суммарного времени вычисления в первую очередь за счет накладных расходов на реконфигурацию архитектуры. Это нарушает исходное требование минимизации суммарного времени вычисления.

В этой связи определим условия оптимизации структуры подграфа:

1. Количество вершин в одном уровне – максимально возможное, но не больше, чем $(v_{\max} = g \times s)$ вершин.

2. Количество уровней – максимальное количество уровней ограниченно коэффициентом *чувствительности ко времени* М-задачи $K_T$:

– $(K_T = 1)$ – задача требует минимально возможного времени вычисления, что выражается в формировании минимально возможного количества уровней, возможно, с применением дополнительных средств минимизации количества уровней или уменьшения времени реконфигурации;

– ($K_T = 0$) – задача не чувствительна к времени вычисления, допускается произвольное количество уровней, оптимизация выполняется только по критерию 1.

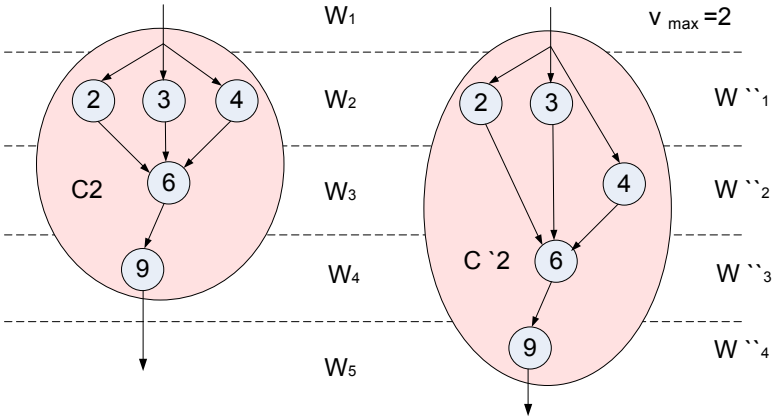



***Рис. 6. Оптимизация структуры подграфа К2***

Количество задач, возложенных на каждую конфигурацию, зависит от выделенных ресурсов для каждой задачи. При этом время реконфигурации каждой задачи $P_{cnf\ task}$ равно доле времени от полной конфигурации всего FPGA устройства $T_{cnf\ FPGA}$ и пропорционально количеству используемых задачей ресурсов [4]:

$$P_{cnf\ task} = \frac{S_{reg}}{S_{full}} \times T_{cnf\ FPGA},$$

где $S_{req}$ – ресурсы используемые одной задачей, $S_{full}$ – полный ресурс FPGA.

Мы рассматриваем только полную конфигурацию FPGA, при этом отображение по уровням, даже в случае решения нескольких задач на одной конфигурации ПЛИС подразумевает их одновременную загрузку в микросхему и выполнение согласно (1). Тогда ($S_{reg} = S_{full}$) и для оценки времени реконфигурации используем следующее выражение:

$$P_{cnf\ task} = T_{cnf\ FPGA}.$$

При таком подходе обеспечение соответствия критерию оптимизации 2 возможно только путем минимизации количества уровней подграфа. Для расширения возможностей с точки зрения уменьшения времени реконфигурации можно рассматривать ряд дополнительных средств. Например, пересмотр структуры подграфа с целью уменьшения количества вершин на критичном уровне, расширение вычислительного ресурса за счет ресурсов других вычислительных модулей, частичная реконфигурация [6], поиск изоморфных структур и реализация их на одних и тех же конфигурациях ПЛИС [6], кеширование конфигураций [7], планирование конфигураций заранее [8], использование адаптивных топологий архитектуры вычислительного модуля [6]. Однако это выходит за рамки данной работы.

Что касается учета ограничений внутримодульных коммуникационных связей при оптимизации подграфа М-программы, вопрос так же остается открытым в данной работе, поскольку они напрямую зависят от топологии архитектуры вычислительного модуля, которая на данном этапе не рассматриваются.

### 3.3. Отображение

Этап отображения представлен в общем виде, поскольку принятая за основу абстрактная модель архитектуры не детализирует топологию вычислительных структур и не рассматривает требований к архитектуре со стороны классов задач. Отображение М-программы на соответствующую реконфигурируемую архитектуру – это отображение $F$ группы М-задач М на структуру $A$, описывающую архитектуру вычислительного модуля, то есть, $F : M \to A.$ Согласно методологии описанной в работе [1], определим последовательность физических виртуальных вычислительных структур для отображения (рис. 5)

$$s_1, s_2, \ldots, s_{(s \times g)} \qquad (2)$$

Отображающая функция $F$ присваивает группам М-задач $M_i$, $i = \overline{1, v}$ (где $v$ – количество М-задач на одном уровне кластера) физические виртуальные структуры из последовательности (2):

$$F(M_i) = \left\{ s_j, s_{j+1}, \ldots, s_{j+|M_i|-1} \,\middle|\, j = 1 + \sum_{k=1}^{i-1} |M_k| \right\}.$$

## 4. Выводы

1. Предложенный метод отображения параллельных задач на реконфигурируемую архитектуру, за счет учета требований со стороны задачи и возможностей вычислительной системы, позволяет получить оптимальное отображение задачи с целью повышения производительности. При этом, в качестве основных критериев оптимизации отображения задачи приняты минимизация времени вычисления, значительным образом определяемое количеством межмодульных взаимодействий и временем реконфигурации архитектуры, и ограничения аппаратных ресурсов, которые являются критическими параметрами, влияющими на производительность реконфигурируемых вычислительных систем.

2. Для решения задач со смешанной параллельностью в вычислительных системах с реализацией многоуровневого параллелизма архитектуры, в том числе с возможностью реконфигурации вычислительных узлов, целесообразно разбить процесс оптимизации отображения задачи на несколько этапов, учитывающих возможности и ограничения каждого уровня параллелизма системы. Предложенный метод, в связи с этим, предполагает нескольких этапов выполнения. На этапах разбиения графа задачи на уровни и кластеризации осуществляется минимизация времени вычисления задачи за счет уменьшения количества операций обмена данными на межмодульном уровне. Дальнейшая оптимизация структуры кластера локализуется на уровне архитектуры вычислительного модуля – учитываются ограничения аппаратных ресурсов вычислительного модуля, осуществляется минимизация времени вычисления, за счет уменьшения времени реконфигурации архитектуры. Минимизация времени реконфигурации в данной работе не рассматривается, для этого предлагается использовать известные, описанные в литературе методы. Этап отображения задачи представлен в общем виде без детализации топологии вычислительного модуля.

3. Топология вычислительных модулей является важным фактором, влияющим на продуктивность вычислений в реконфигурируемых вычислительных системах. В связи с чем дальнейшая работа над методом предполагает разработку типовых архитектур эффективных для решения различных классов задач и на их основе модификацию этапа отображения.

## Список литературы

1. Dümmler J. Scalable computing with parallel tasks / J. Dümmler, T. Rauber, G. Rünger // Proceedings of the 2nd Workshop on Many-Task Computing on Grids and Supercomputers (MTAGS '09), (November 16, Portland, Oregon ) – Article No. 9 – ACM New York, NY, USA, 2009. – P. 1-10.
2. Bansal S., Kumar P., Singh K. An improved two-step algorithm for task and data parallel scheduling in distributed memory machines // Parallel Computing. – Volume 32, Issue 10. – 2006. – P. 759–774.
3. Smith M.C. Optimization of Shared High-Performance Reconfigurable Computing Resources / M.C. Smith, G.D. Peterson // ACM Transactions on Embedded Computing Systems. – Vol. 11, No. 2, Article 36. – 2012. – 22 p.
4. Reconfigurable Computing / M. Huang, V.K. Narayana, H. Simmler [and all] // ACM Transactions on Reconfigurable Technology and Systems (TRETS). – Vol. 3, Issue 4, Article 20. – ACM New York, NY, USA, 2010. – 25 p.
5. Кулаков Ю.А. Формирование структуры корпоративной сети / Ю.А. Кулаков, Халиль Х. А. Аль Шкерат // Гірнича електромеханіка та автоматика: Наук.- техн. Зб. – 2003. – Вип. 70. – С. 86 - 91
6. Panella A. A design workflow for dynamically reconfigurable multi-FPGA systems / A. Panella, M.D. Santambrogio, F.Redaelli [and all] // Proceeding in 18th VLSI System on Chip Conference (VLSI-SoC), (27-29 September, 2010). – IEEE/IFIP, 2010. – P. 414 – 419.
7. El-Araby E. Exploiting Partial Runtime Reconfiguration for High-Performance Reconfigurable Computing / E. El-Araby E., I. Gonzalez, T. El-Ghazawi // ACM Transactions on Reconfigurable Technology and Systems (TRETS). – Vol. 1, Issue 4, Article 36. – ACM New York, NY, USA, 2009. – 23 p.
8. Resano J. Efficiently Scheduling Runtime Reconfigurations / J. Resano, J.A. Clemente, C. Gonzalez, [and all] // ACM Transactions on Design Automation of Electronic Systems (TODAES). – Vol. 13, Issue 4, Article 58. – ACM New York, NY, USA, 2008 – 12 p.

УДК 004.052.042

*МАРКОВСЬКИЙ О.П.,*
*ІВАНОВ Д.Г.,*
*ВАНЧУГОВ Б.Ю.*

# ОРГАНІЗАЦІЯ РЕЗЕРВУВАННЯ ТА ВІДНОВЛЕННЯ ДАНИХ ПРИ ЇХ ВІДДАЛЕНОМУ ЗБЕРІГАННІ

В статье предложен метод резервирования и восстановления данных пользователя, хранящихся на нескольких удаленных узлах. Разработанный метод позволяет восстанавливать данные в наиболее часто встречающихся на практике ситуациях: при потере доступа к одному из узлов хранения или любым двум носителям. Детально описаны математическая идея метода и процедуры восстановления данных из узла хранения, доступ к которому утрачен. Разработанная процедура восстановления иллюстрируется примерами. Приведены теоретические и экспериментальные оценки эффективности предложенного метода.

In paper the method of reservation and recovering of user data stored on some remote memory units is proposed. Developed method makes it possible to recover data for frequent occurrence in practice situation: in case of access loosing for one of remote storage unit or for any two storage devices. The mathematical idea of proposed method and procedure for recovering of data from access lost storage unit are described in details. A numerical example for developed recovering procedure are given.

## Вступ

Початок другого десятиліття XXI-го століття знаменує новий виток спіралі розвитку комп'ютерної обробки даних, а саме: використання технологій віддаленого надання обчислювальних і програмних ресурсів. На початку 80-х років минулого століття за зміну обчислювальним центрам з концентрованими комп'ютерними ресурсами прийшли доступні і максимально наближені до користувачів персональні комп'ютери. Нині, випередження темпів збільшення складності задач користувачів в порівнянні зі зростанням потужності персональних комп'ютерів, а також динамічний прогрес засобів телекомунікацій та мережових технологій, стимулюють відродження концентрації обчислювальних ресурсів з віддаленим доступом до них користувачів.

Технології віддаленого надання користувачеві ресурсів на комерційній основі з використанням Інтернету отримали назву "хмарних обчислень". Поняття ресурсу в цьому контексті включає власне обчислювальні ресурси, програмне забезпечення, а також ресурси пам'яті. Саме віддалене надання ресурсів зовнішньої пам'яті на сьогоднішній день набуло найбільшого поширення [1].

Вузловою проблемою ефективності віддаленого надання ресурсів пам'яті є забезпечення надійного доступу до даних користувача. Причинами втрати доступу до даних можуть бути: фізичне пошкодження носіїв, втрата інформації на них в результаті дії вірусів або некоректних дій програмного забезпечення, тимчасовий вихід з ладу обладнання або перевантаження вузла зберігання даних, втрата доступу вузла до мережі, комерційні чи природні катаклізми. Швидке збільшення кількості користувачів систем віддаленого зберігання інформації загострює проблему ефективного доступу до неї.

Для забезпечення надійного та оперативного доступу користувача до його інформації, що зберігається на віддалених носіях потрібні спеціальні механізми резервування та відновлення даних, доступ до яких постійно чи тимчасово втрачено.

Таким чином, наукова задача створення ефективних методів та засобів резервування і відновлення даних, що зберігаються на віддалених носіях є актуальною та важливою з огляду на особливості сучасного етапу розвитку інформаційних технологій.

## Аналіз технологій резервування даних в системах їх віддаленого зберігання

При віддаленому зберіганні даних користувача вони розподіляються по окремим вузлам зберігання інформації. В рамках окремого вузла організовано розподілення даних по носіях, доступ користувачів до даних, їх захист, резервування в разі втрати доступу до одного або декількох носіїв.

Для забезпечення високого рівня неперервності доступу кожному користувачеві до своєї інформації, що зберігається на віддалених носіях найчастіше використовується їх резервування.

В якості критеріїв ефективності систем резервування найчастіше виступають:
- обчислювальна складність процедури відновлення даних з використанням резервних носіїв;
- кількість носіїв ($К_В$), дані з яких можуть бути відновлені в разі втрати доступу до них;
- відношення кількості резервних носіїв ($К_Р$) до числа носіїв, дані з яких можуть бути відновлені ($\alpha=К_Р/К_В$).

Проведений аналіз літературних джерел [1-3] показав, що найчастіше причинами втрати доступу до даних віддалених користувачів стає вихід з ладу окремих носіїв (або стирання інформації на них), є а також тимчасова нездатність вузлу обслужити запит користувача.

При втраті даних на окремих носіях в результаті чи то виходу їх із ладу, або стирання (помилкового чи цілеспрямованого) даних на них, кількість $q$ таких носіїв залежить від часу $t$, що пройшов від моменту останнього звертання до них. Якщо вважати, що користувач зберігає свої дані на М носіях і залежність ймовірності $P(t)$ втрати даних до них від часу підпорядкована експоненційному закону, то ймовірність того, що до $q$ з них буде втрачено доступ, визначається наступною формулою:

$$P(t) = C_M^q \cdot (1 - e^{-\lambda \cdot t})^q \cdot e^{-(M-q)\cdot\lambda\cdot t}, \qquad (1)$$

де $\lambda$- інтенсивність втрати даних на одному носієві в результаті його виходу з ладу чи стирання.

З формули (1) та аналізу статистики виходу з ладу носіїв [4] випливає, що ймовірність втрати доступу до одного носія доволі невелика, на 2-3 порядки меншою є ймовірність виходу з ладу двох носіїв. Тому, можна вважати, що реально, за умови, коли час між зверненнями не перевищує року, кількість носіїв, доступ до яких втрачено в результаті виходу їх з ладу або стирання даних не перевищує 2-х. Більшість реальних систем резервування даних розраховано саме на таку кількість носіїв, доступ до яких може бути втрачено [4].

Тимчасова нездатність вузлу обслужити запит користувача здебільшого пов'язана з перевантаженістю вузла зберігання даних, виходом з ладу його апаратних чи програмних компонентів, економічними чи природними катаклізмами.

Проблема забезпечення надійного доступу до даних, що містяться на віддалених від користувачів вузлах зберігання інформації спонукала до створення ряду технологій резервування.

Найбільш простою схемою резервування є використання простого дублювання даних на двох носіях. До такого типу відносяться системи Intermemory [2] та RAID-1 [3]. Використання простого дублювання пов'язане зі значними затратами об'єму пам'яті. При цьому, воно не гарантує відновлення даних при втраті доступу до обох носіїв, на яких зберігаються копії даних.

Значно меншого об'єму пам'яті потребує схема резервування, що передбачає для групи носіїв використання одного контрольного, на якому зберігається сума за модулем 2 відповідних даних всіх носіїв групи. Ця схема дозволяє доволі просто відновити дані при втраті доступу до одного з носіїв групи. Найбільш відомим застосуванням описаної схеми резервування є система RAID-1 [3]. Проте ця схема не дозволяє відновлювати дані при втраті доступу до більш як одного носія.

Найбільшого поширення на практиці набули технології відновлення даних на основі корегуючих та erasure кодів [4]. При відновленні даних з носіїв, до яких втрачено доступ, як правило, не має потреби в їх локалізації. Класичні корегуючі коди, такі, як коди Хемінга, БЧХ, Ріда-Соломона орієнтовані на послідовне виконання двох процедур: локалізації спотвореної частини даних та їх виправлення. З цієї причини при використанні загаданих вище класичних корегуючих кодів для відновлення даних з носіїв, до яких втрачено доступ, потрібна їх модифікація. Модифіковані коди Ріда-Соломона, зокрема, використовуються в системі відновлення даних з носіїв RAID-6 [3].

Більш ефективно використання для цієї цілі спеціальних erasure кодів. Більшість таких кодів [4] мають за основу лінійні перетворення, і це зумовлює швидке зростання кількості резервних носіїв при збільшенні числа носіїв до яких втрачено доступ.

Загальною рисою відомих технологій відновлення даних з носіїв, до яких втрачено доступ є те, що вони реалізовані в рамках окремого вузла зберігання інформації. Це означає, що в разі втрати доступу до вузла в результаті тимчасового виходу його з ладу, перевантаження, вірусної атаки, відключення від мережі, техногенних або природних катаклізмів, відомі механізми відновлення даних або доступу до них для конкретного користувача не спрацьовують.

Таким чином, існуючі методи відновлення доступу до даних в системах їх віддаленого зберігання не гарантують вирішення цієї задачі в разі втрати доступу до вузла зберігання інформації.

Ціллю досліджень є розробка та дослідження методу відновлення даних з віддалених від користувача носіїв для найбільш поширених на практиці ситуаціях, включаючи втрату доступу до вузла зберігання інформації.

### Метод відновлення даних з носіїв вузла, до якого втрачено доступ

Проведений аналіз показав, що існуючі системи резервування інформації на носіях, які реалізуються на рівні вузлів зберігання не забезпечують повною мірою надійності та оперативності доступу до даних конкретного користувача - фактичного їх володаря. Тільки останній може у відповідності до цінності для нього даних, особливостей їх використання та вимог щодо оперативності доступу до них визначати політику резервування інформації на віддалених носіях. Це диктує доцільність застосування паралельно з існуючими системами резервування засобів забезпечення надійності та оперативності доступу до даних конкретного користувача.

З позицій користувача найбільш частими причинами відмов у доступі до віддаленої інформації є тимчасова втрата взаємодії з вузлом зберігання та втрата доступу до одного та рідше до двох носіїв.

Нехай інформація користувача, що зберігається на $s$ носіях, розміщена на $n$ вузлах зберігання інформації (надалі ВЗІ або просто вузол). При цьому на кожному з вузлів задіяно для даних конкретного користувача однакова кількість носіїв - $m=s/n$. Дані, що розміщено на $j$-тому носієві, $j\in\{1,..,m\}$, $i$-го вузла, $i\in\{1,..,n\}$, позначаються як $a_{ij}$.

Система віддаленого зберігання даних дає можливість швидкого доступу до будь-якої користувацької інформації. Як зазначалося вище, на практиці для користувача причинами відмові в доступі найчастіше є дві ситуації:

1. Один з вузлів тимчасово не доступний.
2. Не доступними є один або, що щонайбільше, два носія різних вузлів.

Отже, постає необхідність одночасного вирішення 2-ох задач:

1. Відновлення даних будь-якого з $M$ носіїв одного, наприклад, $k$-ого вузла: $a_{k1},a_{k2},...,a_{km}$, де $k\in\{1..n\}$.

2. Відновлення даних будь-яких 2-х носіїв $a_{qe}$ і $a_{\mathrm{gr}}$, де $q,g\in\{1,..,n\}$ і $e,r\in\{1,..,m\}$.

Для вирішення вказаних задач пропонується використати $m$+2 додаткових носіїв, які розділяються на дві групи. Перша група складається з $m$ додаткових носіїв, дані на яких $b_1,b_2,..,b_m$ формуються як суми за модулем два даних користувача з однойменних носіїв всіх вузлів на яких зберігається його інформація:

$$\forall j=1,...,m: b_j=\bigoplus_{i=1}^{n} a_{ij} \qquad (2)$$

Друга група з двох додаткових носіїв, на яких формуються і зберігається сума $c$ за модулем 2 поліноміальних добутків даних всіх можливих пар однойменних носіїв на яких записані дані користувача:

$$c=\bigoplus_{j=1}^{m}\ \bigoplus_{\substack{i=1,..,n-1\\ k=i+1,..,n}} (a_{ij}\otimes a_{kj}), \qquad (3)$$

де символом '⊗' позначено операцію поліноміального множення. В силу того, що розрядність поліноміального добутку вдвічі більша за розрядність множників, то для зберігання суми $c$ добутків використовується два додаткових носія. Практично, код $c$ являє собою набір сум поліноміальних добутків фрагментів, на які розбиваються дані.

Додаткові носії для зберігання даних $b_1,b_2,..,b_m$ і коду $c$ можуть бути розміщені як на окремому вузлі, так і безпосередньо у користувача.

При втраті користувачем доступу до $k$-го вузла, $k\in\{1,..,n\}$ дані $a_{k1},a_{k2},...,a_{km}$ з його носіїв відновлюються у відповідності з наступним виразом:

$$\forall j\in\{1,..,m\}: a_{kj}=b_j\oplus\bigoplus_{l=1,l\neq k}^{n} a_{lj}. \qquad (4)$$

При втраті доступу до двох довільних носіїв, на яких розміщено інформацію користувача, постає задача відновлення даних $a_{qe}$ і $a_{gr}$ і тут можна розглядати три випадки:

- втрачено доступ до двох носіїв, що знаходяться на одному вузлі, тобто $q=g$. Для цього випадку відновлення $a_{q,e}$ і $a_{gr}$ реалізується за допомогою формули (3):

$$\begin{aligned} a_{qe}&=b_e\oplus\bigoplus_{l=1,l\neq q}^{n} a_{l,e},\\ a_{gr}&=b_r\oplus\bigoplus_{l=1,l\neq g}^{n} a_{lr} \end{aligned} \qquad (5)$$

- втрачено доступ до двох носіїв на різних вузлах, тобто $q\neq g$ і при цьому номери $e$ і $r$ носіїв зайнятих даними користувача на різних вузлах також різні: $e\neq r$. Для цього випадку відновлення $a_{q,e}$ і $a_{gr}$ також реалізується за формулою (4).

- втрачено доступ до двох носіїв на різних вузлах, тобто $q \neq g$ але при цьому номери $e$ і $r$ носіїв зайнятих даними користувача на різних вузлах однакові: $e=r$. У цьому випадку відновлення $a_{qr}$ і $a_{gr}$ відбувається за дещо складнішою процедурою. Спочатку визначається значення суми за модулем 2 вказаних кодів $a_{qr}$ і $a_{gr}$ :

$$\delta = a_{qr} \oplus a_{gr} = b_r \oplus \bigoplus_{l=1, l\neq q, l\neq g}^{n} a_{lr} \qquad (6)$$

Значення поліноміального добутку η кодів $a_{qr}$ і $a_{gr}$ : $\eta = a_{qr} \otimes a_{gr}$ визначається наступним чином. Сума $c$ за модулем 2 поліноміальних добутків даних всіх можливих пар одноіменних носіїв на яких записані дані користувача може бути розділена на дві компоненти, з яких перша γ не залежить від $a_{qr}$ і $a_{gr}$, а друга - залежить від них:

$$\begin{aligned} c &= \bigoplus_{j=1}^{m} \bigoplus_{\substack{i=1,..,n-1 \\ k=i+1,..,n}} (a_{ij} \otimes a_{kj}) = \\ &= \bigoplus_{j=1, j\neq r}^{m} \bigoplus_{\substack{i=1,..,n-1 \\ k=i+1,..,n}} (a_{ij} \otimes a_{kj}) \oplus \bigoplus_{\substack{i=1,..,n-1 \\ k=i+1,..,n}} (a_{ir} \otimes a_{kr}) \end{aligned} \qquad (7)$$

Очевидно, що перша компонента γ суми (7) має вигляд:

$$\gamma = \bigoplus_{j=1, j\neq r}^{m} \bigoplus_{\substack{i=1,..,n-1 \\ k=i+1,..,n}} (a_{ij} \otimes a_{kj}) \qquad (8)$$

Тоді сума за модулем 2 (7) може бути представлена у наступному вигляді:

$$\begin{aligned} c &= \gamma \oplus \bigoplus_{\substack{i=1,..,n-1 \\ k=i+1,..,n}} (a_{ir} \otimes a_{kr}) = \\ &= \gamma \oplus (a_{qr} \oplus a_{gr}) \otimes \bigoplus_{i=1, i\neq q, i\neq g}^{n} a_{ir} \oplus (a_{qr} \otimes a_{gr}) = \\ &\gamma \oplus \delta \otimes \bigoplus_{i=1, i\neq q, i\neq g}^{n} a_{ir} \oplus (a_{qr} \otimes a_{gr}) \end{aligned} \qquad (9)$$

Відповідно, чисельне значення γ визначається за наступною формулою:

$$\eta = a_{qr} \otimes a_{gr} = c \oplus \gamma \oplus \delta \otimes \bigoplus_{i=1, i\neq q, i\neq g}^{n} a_{ir} \qquad (10)$$

Таким чином, значення кодів $a_{qr}$ і $a_{gr}$ з пари носіїв, до яких втрачено доступ може бути віднайдено як розв'язання системи рівнянь:

$$\begin{cases} a_{qr} \oplus a_{gr} = \delta \\ a_{qr} \otimes a_{gr} = \eta \end{cases} \qquad (11)$$

Кожен з кодів $a_{qr}$ і $a_{gr}$ має певну розрядність $w$, яка вимірюється мільйонами бітів. При обчисленні поліноміального добутку коди $a_{qr}$ і $a_{gr}$ розділяються на $d$-розрядні фрагменти, так, що код η добутку складається з $(2 \cdot d\text{-}1)$-розрядних фрагментів.

В силу симетричності операцій додавання за модулем 2 та поліноміального множення система рівнянь (11) має два розв'язки.

Можна показати, що для кожного з $d$-розрядних фрагментів система (11) може бути зведена до системи з $2 \cdot d$-1 лінійних рівнянь. Наприклад, якщо $d$=4, і біти фрагменту $a_{qr}$ позначаються як $x_1, x_2, x_3, x_4$, біти фрагменту $a_{gr}$ позначаються як $y_1, y_2, y_3, y_4$, біти фрагменту δ позначаються як $\beta_1, \beta_2, \beta_3, \beta_3$ , біти η позначаються як $\rho_1, \rho_2, ..., \rho_7$, то друге рівняння системи (11) може бути представлено у вигляді наступних бітових рівнянь:

$$\begin{cases} x_1 \oplus y_1 = \beta_1 \\ x_2 \oplus y_2 = \beta_2 \\ x_3 \oplus y_3 = \beta_3 \\ x_4 \oplus y_4 = \beta_4 \\ x_1 \cdot y_1 = \rho_1 \\ x_2 \cdot y_1 \oplus x_1 \cdot y_2 = \rho_2 \\ x_3 \cdot y_1 \oplus x_2 \cdot y_2 \oplus x_1 \cdot y_3 = \rho_3 \\ x_4 \cdot y_1 \oplus x_3 \cdot y_2 \oplus x_2 \cdot y_3 \oplus x_1 \cdot y_4 = \rho_4 \\ x_4 \cdot y_2 \oplus x_3 \cdot y_3 \oplus x_2 \cdot y_4 = \rho_5 \\ x_4 \cdot y_3 \oplus x_3 \cdot y_4 = \rho_6 \\ x_4 \cdot y_4 = \rho_7 \end{cases} \qquad (12)$$

Підстановкою виразів для $y_1, y_2, y_3, y_4$ з перших 4-х рівнянь: $y_1 = \beta_1 \oplus x_1, ..., y_4 = \beta_4 \oplus x_4$ системи (11) в наступні 7 рівнянь цієї системи отримується наступна система лінійних бітових рівнянь:

$$\begin{cases} x_1 \oplus y_1 = \beta_1 \\ x_2 \oplus y_2 = \beta_2 \\ x_3 \oplus y_3 = \beta_3 \\ x_4 \oplus y_4 = \beta_4 \\ x_1 \cdot \beta_1 \oplus x_1 = \rho_1 \\ x_2 \cdot \beta_1 \oplus x_1 \cdot \beta_2 = \rho_2 \\ x_3 \cdot \beta_1 \oplus x_2 \cdot \beta_2 \oplus x_2 \oplus x_1 \cdot \beta_3 = \rho_3 \\ x_4 \cdot \beta_1 \oplus x_3 \cdot \beta_2 \oplus x_2 \cdot \beta_3 \oplus x_1 \cdot \beta_4 = \rho_4 \\ x_4 \cdot \beta_2 \oplus x_3 \cdot \beta_3 \oplus x_3 \oplus x_2 \cdot \beta_4 = \rho_5 \\ x_4 \cdot \beta_3 \oplus x_3 \cdot \beta_4 = \rho_6 \\ x_4 \cdot \beta_4 \oplus x_4 = \rho_7 \end{cases} . \qquad (13)$$

Очевидно, що система (12) може бути доволі просто розв'язана і результаті можуть бути отримані бітові значення $x_1, x_2, x_3, x_4$, поточного фрагменту $a_{qr}$ і біти як $y_1, y_2, y_3, y_4$ одноіменного фрагменту $a_{gr}$.

Таким чином доведено, що запропонований метод забезпечує відновлення даних при втраті доступу до будь-яких двох носіїв, на яких розміщено дані користувача.

### Оцінка ефективності

Нескладно показати, що обчислювальна складність розв'язання системи лінійних рівнянь (13) визначається як $O(2\cdot d^2+d)$ так, що обчислювальна складність розв'язання системи (11) становить $O(2\cdot w\cdot d+w)$. В перших двох випадках обчислювальна складність відновлення даних з двох носіїв визначається складністю операції додавання за модулем 2: $O(2\cdot w/d)$. Враховуючи, що ймовірність третього випадку при відновленні даних з двох носіїв приблизно оцінюється як $n^{-1}$, то, в середньому, обчислювальна складність операцій відновлення даних з двох носіїв при використанні запропонованого методу визначається як $O(2\cdot w\cdot(n+d^2)/(n\cdot d)) \approx O(2\cdot w/d)$. Обчислювальна складність відновлення даних з носіїв вузла, доступ до якого втрачено визначається як $O(m\cdot w/d)$.

На практиці, час відновлення залежить не скільки від обчислювальної складності операцій реконструкції даних, стільки від кількості носіїв, до яких треба звернутися, щоб отримати дані, потрібні для відновлення втраченої інформації.

Кількість додаткових носіїв для зберігання резервних кодів $b_1,b_2,..\ ,b_m$ дорівнює $m$, а для збереження коду $c$ - два носії, вважаючи на те, що розрядність поліноміального добутку вдвічі перевищує довжину множників. Таким чином, кількість $h$ резервних носіїв, що використовуються у запропонованому методі визначається формулою:

$$h = m + 2 \qquad (14)$$

Як зазначалося вище, практично всі відомі засоби відновлення доступу до віддалених даних користувача реалізуються в рамках окремого вузла зберігання даних. Це означає, що резервування відбувається на рівні самого вузла і, відповідно, відомі засоби практично не забезпечують вирішення вказаної задачі в разі втрати доступу користувача до вузла або тимчасовій непрацездатності самого вузла. Разом з тим, статистика відмов в доступі до даних, що віддалено зберігаються, свідчить про те, що такий тип втрати доступу зустрічається доволі часто. Реалізація запропонованого методу резервування та відновлення доступу до даних орієнтована на рівні користувача і передбачає, що зберігання основних і резервних його даних відбувається на різних вузлах. Відповідно, викладений метод забезпечує ефективне відновлення для користувача доступу даних, які зберігаються на тимчасово недоступному віддаленому вузлі.

Крім того, запропонований метод дозволяє ефективно для користувача вирішувати задачу забезпечення неперервності доступу до своїх даних в разі втрати доступу до двох довільних носіїв, локалізованих як на одному вузлі зберігання даних, так і на різних вузлах.

Таким чином, запропонований метод дозволяє ефективно вирішувати задачу резервування і відновлення даних користувача для найбільш поширених на практиці випадків втрати доступу до них.

Задача відновлення даних при втраті доступу до одного вузла та двох довільних носіїв може бути вирішена з використанням відомих корегуючих кодів, таких, зокрема як коди Ріда-Соломона, циклічні та БЧХ коди, а також erasures коди. При цьому, для вирішення, за допомогою зазначених кодів, задачі відновлення даних з носіїв одного вузла потребує $2\cdot m$ додаткових носіїв. При цьому зазначені вище корегуючі коди забезпечують відновлення даних і при втраті інформації з двох довільних носіїв. Таким чином, загальна кількість носіїв для резервування даних для їх відновлення при втраті доступу до носіїв одного вузла зберігання або двох довільних носіїв становить $2\cdot m$, тобто менше в порівнянні з оцінкою (13) для запропонованого методу.

Процедура відновлення даних з використанням корегуючих кодів передбачає розв'язання систем лінійних рівнянь. Для відновлення інформації з носіїв одного вузла потрібно розв'язувати систему з $m$ лінійних рівнянь. Обчислювальна складність розв'язання такої системи становить $O(2\cdot m^2)$ для одного $d$-розрядного фрагменту. Обчислювальна складність відновлення даних з носіїв вузла, доступ до якого втрачено, визначається як $O(m^2\cdot w/d)$. Тобто, використання запропонованого методу дозволяє зменшити обчислювальну складність приблизно в $m$ раз.

При вирішенні задачі відновлення даних з двох довільних носіїв за допомогою корегуючих кодів, розв'язується система з 4-х рівнянь, обчислювальна складність цієї процедури становить $O(8\cdot w/d)$. Порівняння з наведеним вище аналогічним показником для запропонованого методу доводить, що його застосування дозво-

ляє зменшити обчислювальну складність приблизно в 4 рази.

В відомих корегуючих кодах, одні і ті ж самі механізми використовуються для відновлення даних при всіх варіантах втрати доступу до носіїв. Відповідно, для вирішення різних задач витрачаються однакові обчислювальні ресурси. В запропонованому методі передбачена більш гнучка процедура відновлення даних, яка дозволяє витрачати різні за об'ємом обчислювальні ресурси в різних випадках втрати доступу до віддалених носіїв.

Таким чином, запропонований метод відновлення даних при втраті доступу до носіїв одного вузла зберігання даних або двох довільних носіїв забезпечує зменшення числа резервних носіїв та обчислювальної складності процедури відновлення в порівнянні з корегуючими кодами. Досягнутий ефект забезпечується за рахунок вузької спеціалізації запропонованого методу на найбільш поширених на практиці ситуаціях втрати доступу користувача до даних, що віддалено зберігаються.

Важливим і принципово новим моментом є те, що в рамках запропонованого методу з'являється можливість відновлення інформації цілого вузла за рахунок використання резервних даних користувачів, що користуються цим вузлом для віддаленого зберігання своїх даних. Це значно підвищує живучість систем віддаленого зберігання інформації.

## Висновки

Таким чином, в роботі запропоновано метод організації резервування та відновлення даних при їх зберіганні на віддалених від користувача носіях з урахуванням найбільш поширених на практиці ситуацій: втраті доступу до окремого вузла зберігання або не більше ніж двох віддалених носіїв.

Дослідження запропонованого методу забезпечення неперервності доступу користувача до його віддалених даних довели, що за рахунок спеціалізації та використання більш гнучких процедур відновлення даних, він забезпечує більшу ефективність резервування в порівняння з відомими методами, корегуючими та ensure кодами.

Розроблений метод може бути ефективно використано в перспективних "хмарних" технологіях віддаленого зберігання даних.

## Список літератури

1. Иванов Д. Г. Организация резервирования в системах распределенного хранения данных // Вісник Національного технічного університету України "КПІ" Інформатика, управління та обчислювальна техніка, – Київ: ВЕК+ – 2012 – № 56. с.160-164.
2. Луцкий Г.М., Иванов Д.Г. Метод восстановления данных на основе скошенных матриц в системах распределенного хранения // Известия высших учебных заведений. Проблемы полиграфии и издательского дела. Информационные технологии.- М.:УПИПК МГУП им. И.Федорова.- 2013.- № 2. - С.47-52
3. Heuert U., Ivanov D. Paladin: Secure and redundant cloud storage // Herald of the Merseburg University of Applied Sciences. - Merseburg: Elbe Drucketei Wittenberg GmbH.- 2011.-№ 8.-S.220-228.
4. Plank J. S., Thomasson M.G., On Practical Use of LDPC Erasure Codes for Distribute Storage Application: Technical Report UT-CS-03-510.- Department of Computer Science, University of Tennessee.-2004.

УДК 681.3.07

*ПОЛТОРАК В.П.,*
*СТЕПАНЕНКО О.Ю.*

# РІШЕННЯ ЗАДАЧІ ДИНАМІЧНОГО БАЛАНСУВАННЯ В ОБЛАСТІ МЕРЕЖІ OSPF

В статті представлені результати дослідження області мережі, яка функціонує на протоколі маршрутизації OSPF, на предмет можливості створення динамічного алгоритму автоматичного розподілу навантаження над протоколом, що забезпечив би стабільне функціонування мережевої структури. Створено моделі об'єктів та проведено моделювання роботи запропонованого алгоритму.

The article presents an area network studying results which operates under the routing protocol OSPF. It is made for possibility of creating a dynamic algorithm for automatic load distribution over an OSPF that would ensure the stable operation of the network structure. An object simulation models of proposed algorithm are created and implemented.

## Постановка задачі в загальному вигляді та її актуальність

Відкритий протокол вибору першого найкоротшого шляху (Open Shortest Path First – OSPF) – протокол динамічної маршрутизації, заснований на технології відстеження стану каналу (link-state technology), що використовує для знаходження найкоротшого шляху Алгоритм Дейкстри (Dijkstra's algorithm). Протокол OSPF був розроблений IETF ще в 1988 році. На теперешній час OSPF широко використовується в корпоративних мережах через свої переваги: високу швидкість збіжності, підтримка мережних масок змінної довжини (VLSM), відсутність обмежень досяжності, оптимальне використання пропускної здатності мережі, оптимальний вибір шляху маршрутизації, можливість застосування на обладнанні від різних виробників, підтримка різних вимог IP-пакетів до якості обслуговування (пропускна здатність, затримка і надійність) та можливість статичного розподілу навантаження. Але, не дівлячись на перелічені переваги, є і недоліки такі, як розподіл навантаження за замовчуванням лише по чотирьом маршрутам з рівними метриками. Звичайно можливо за допомогою штатних методів і додаткових налаштувань обійти данні вимоги. Таке рішення не може бути використане, як уніфіковане, та вимагає багато часу і творчого підходу.

## Мета

Дослідження мережі, яка функціонує на протоколі маршрутизації OSPF, на предмет можливості створення динамічного алгоритму автоматичного розподілу навантаження над протоколом, що забезпечив би стабільне функціонування мережевої структури.

## Огляд існуючих рішень

У цьому розділі ми розглянемо статичну задачу балансування навантаження, в якій вимоги до трафіку відомі.

OSPF використовує дворівневу ієрархію мережі. Ця ієрархія містить два основних елементи:

Область. Область – це група суміжних мереж. Області являють собою логічні розділи автономної системи.

Автономна система. Автономна система – це сукупність мереж із спільним управлінням і зі спільною стратегією маршрутизації. Автономні системи, також відомі як домени, можна логічно розділити на кілька областей.

OSPF використовує різні типи областей для обмеження лавинної розсилки оновлень станів каналів (link-state flooding) через всю мережу. Лавинна розсилка і обчислення алгоритму Дейкстри на маршрутизаторі обмежені змінами в межах області. Маршрутизатори, пов'язані з кількома областями, називають граничними маршрутизаторами мережі (ABR). Тому ABR повинен містити інформацію про всі області, до яких він підключений. Маршрутизатор називають ASBR граничним маршрутизатором автономної системи, коли він виконує функції шлюзу між OSPF та іншим протоколом маршрутизації. Маршрутизатор називають внутрішнім маршрутизатором (IR), коли всі його OSPF інтерфейси відносяться до однієї області.

OSPF передбачає кілька типів областей: магістральна область (backbone area), стандартна область (standard area), кінцева

область (stub area) і неповністю кінцева область (not-so-stubby area).

Магістральна область є ядром всієї OSPF мережі, всі області з'єднані з магістральною областю. Вона має бути зпроектована та розрахована виходячи з вимог до конкретної мережі. Нижче наведені рекомендації, використовувані при проектуванні опорної області мережі OSPF (області 0) [1]:

Враховувати, що область 0 є транзитною і не може служити для розміщення одержувачів трафіку.

Вимагати того, щоб забезпечувалася і контролювалася стабільність базової області.

Забезпечити найбільш повне застосування в проекті засобів резервування.

Прагнути до того, щоб магістральні області OSPF були зв'язаними.

Намагатися підтримувати якомога більш просту структуру області. Чим менше в ній маршрутизаторів, тим краще.

Підтримувати симетрію при визначенні пропускної здатності, щоб протокол OSPF міг забезпечувати розподіл навантаження .

Стежити за тим, щоб всі інші області з'єднувалися безпосередньо з областю 0 .

Не допускати розміщення в області 0 ресурсів кінцевих користувачів (хостів).

Можна зробити висновок, що магістральна область має бути стійкою до зростання трафіку за рахунок резервування та симетричності пропускної здатності. З чого витікає, що застосування динамічного балансування до опорної області мережі OSPF не є доцільним.

Найбільш ймовірним є варіант, при якому може виникнути потреба в балансуванні саме не магістральної області, адже саме в областях цього типу мають бути розташовані ресурси кінцевих користувачів.

Модель мережі. Розглянемо IP мережу на основі протоколу OSPF маршрутизації (OSPF мережі). Нехай $N$ позначає множину вузлів (маршрутизаторів), $n$ і $L$ набір посилань $l$ мережі. Як альтернативу, використаємо позначення $(i, j)$ для позначення зв'язку від вузла $i$ до вузла $j$. Ємність каналу $l$ позначимо $b_l$. Множину пар входу-виходу $k = (s_k; t_k)$ позначимо як $s_k$ для вхідних вузлів і $t_k$ для вихідних вузлів з індексом $K$ вузлів для пари $k$. Нехай $P_k$ – множина всіх можливих шляхів $p$ від вузла $s_k$ до вузла $t_k$. Використаємо позначення $l \in p$ , якщо канал (link) $l$ належить до шляху $p$. Запит трафіку вхідної-вихідної пари позначимо $d_k$.

У протоколах маршрутизації на основі стану каналу, таких як OSPF, кожна ланка $l$ асоціюється з фіксованою вагою $w_l$ і трафік передається по найкоротшому з шляхів. Нехай $P_k^{SP}$ - множина найкоротших шляхів від вузла $s_k$ до вузла $t_k$ по відношенню до ваг зв'язків $w_l$, тоді отримаємо [3]:

$$P_k^{SP} = \left\{ p \in P_k \Big| \sum_{l \in p} w_l = \min_{p' \in P_k} \sum_{l' \in p'} w_{l'} \right\}. \qquad (1)$$

Стандартним є вибір $w_l = 1$ для всіх $l$ результатів в шляхах на мінімальній відстані, що мінімізує таким чином загальну необхідну смугу пропускання каналу зв'язку.

У кожному вузлі $i$, вхідний трафік з тим же призначенням $t$ агрегується, а потім розщеплюється в зв'язки *(i; j)*, які належать до одного з найкоротших шляхів з вхідної-вихідної пари *(i; t)*. Такі сусідні вузли $j$ називаються допустимими наступними ланками (хопами). Нехай $\phi_{ij}^t$ це відповідні коефіцієнти розподілу. Таким чином, $\phi_{ij}^t$ відноситься до частини всього трафіку, що проходить через вузол $i$, та трафіку призначеного вузлу $t$, що відправлено через зв'язок *(i; j)*. Необхідна умова [3]:

$$\sum_{j:(i;j)\in} \phi_{ij}^t = 1 \qquad (2)$$

На рисунку 1 ілюстровано математичну модель мережі описану вище. Як, наприклад, зазначено в [4], як правило, передбачається, що ці відносини розщеплення на $\phi_{ij}^t$ дорівнюють:

$$\phi_{ij}^t = \frac{1}{\left|\{j : (i;j) \in p\}\right|} \qquad (3)$$

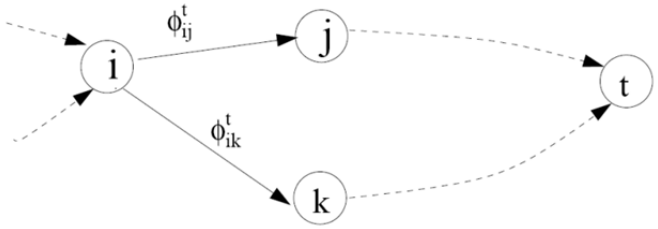



***Рис. 1. Модель мережі***

Постановка задачі розподілу статичного балансування. Розподіл навантаження може базуватись на мінімізації середньої затримки або мінімізації максимального використання лінії зв'язку. Оптимальне рішення для задачі мінімізації максимального використання зв'язків не може бути унікальним в цілому. Серед оптимальних рішень, те, яке мінімізує загальне використання ресурсів є найбільш розумним. Таким чином сформулювалася постановка задачі, рішення якої має мінімізувати максимальне використання зв'язків у мережі з великою вагою, але і

враховує загальне використання ресурсів з меншою вагою, як, наприклад, в [5]:

$$\begin{gathered}\min\left(\mathrm{a}+r\sum_{k\in K}\sum_{l\in L}x_l^k\right);\\ a\geq 0, x_l^k\geq 0;\Leftrightarrow k\in K, l\in L;\\ \sum_{k\in K}x_l^k\leq ab_l;\Leftrightarrow l\in L;\\ Ax^k=R^k;\Leftrightarrow k\in K;\end{gathered}\qquad(4)$$

де $a$ і $x_l^k$ є вільні змінні, що описують мінімум максимальної утилізації каналу і навантаження трафіку пари вхід-вихід $k$ на лінії $l$ відповідно, а $r$ деяка мала константа. Крім того, $A\in R^{N\times L}$, де $N=|N|$ і $L=|L|$, позначає матрицю, для якої виконується $A_{nl}=-1$, якщо канал $l$ прямує до вузла $n$, та $A_{n\,l}=1$, якщо канал $l$ виходить з вузла $n$, і $A_{nl}=0$ в іншому випадку; $x^k\in R^{L\times 1}, k\in K,$ відноситься до вектора навантаження каналу з елементами $x_l^k$; $R^k\in R^{N\times 1}, k\in K,$ позначає вектор, для якого: $R_{sk}^k=d_k, R_{tk}^k=-d_k$ і $R_{sk}^k=0$ в іншому випадку.

В [5] доведено, що існує множина позитивних ваг каналів $w_l$ таких, що оптимальні шляхи в задачі балансування навантаження (4) це найкоротші шляхи щодо цих ваг каналів. Іншими словами, множина рішень задачі балансування буде завжді міститися в множині (1) до для всіх $k$. Процедура визначення цих ваг каналів приведена нижче.

Нехай $y_l=\sum_{k\in K}x_l^k$ навантаження трафіку, на каналі $l$, а оптимальне рішення $x_l^k$ проблеми балансування навантаження (4). Сформулюємо відповідно нову первинну задачу та її двоїсту задачу. У первинній задачі викликані навантаження трафіку $y_l$ використовуються, як нові обмеження пропускної здатності [5]:

$$\begin{gathered}\min\left(\sum_{k\in K}\sum_{l\in L}x_l^k\right);\\ x_l^k\geq 0;\Leftrightarrow k\in K, l\in L;\\ \sum_{k\in K}x_l^k\leq y_l;\Leftrightarrow l\in L;\\ Ax^k=R^k;\Leftrightarrow k\in K;\end{gathered}\qquad(5)$$

Двоїста задача:

$$\begin{gathered}\max\left(\sum_{k\in K}d_kU_{tk}^k-\sum_{l\in L}y_lW_l\right);\\ W_l\geq 0;\Leftrightarrow l\in L;\\ U_{sk}^k;\Leftrightarrow k\in K;\\ U_j^k-U_i^k\leq W_{(i,j)}+1;\Leftrightarrow k\in K;(i,j)\in L.\end{gathered}\qquad(6)$$

Необхідні ваги каналу потім додаються $w_l = W_l + l$, де змінні $W_l$ визначаються як рішення двоїстої задачі.

Крім того, оптимальне призначення трафіку обчислюється на основі розподілу метрик $\phi_{ij}^t$ визначаних з навантаження каналів $x_l^k$ отриманих при вирішенні прямої задачі. Ці відносини розподілу розраховуються наступним чином [6]:

$$\phi_{ij}^t=\frac{\sum_{k:t_n=t}x_{(i,j)}^k}{\sum_{j:(i;j)\in L}\sum_{k:t_n=t}x_{(i,j)}^k}.\qquad(7)$$

### Алгоритм динамічного балансування в області мережі OSPF

Проблема статичного балансування навантаження представлена в попередньому розділі може бути поставлена і вирішена тільки тоді, коли запити до обсягу трафіку $d_k$ відомі.

Постановка задачі динамічного балансування. Наші припущення полягають у наступному. Вимоги трафіку $d_k$ фіксовані, але невідомі. Навантаження каналу періодично вимірюється в час $t_n$. Нехай $y_l(n)$ позначає виміряне навантаження каналів $l$ за період часу $(t_{n-1}, t_n)$. Інформація про виміряне навантаження збирається з усіх вузлів в мережі. Час, необхідний для збору інформації незначний (ним можна знехтувати) в порівнянні з довжиною періоду вимірювання.

Загалом основне завдання динамічного алгоритму балансування навантаження полягає в наступному: на підставі виміряних навантажень на каналах, метрики каналів $w_l$ і відношення розподілу трафіку $\phi_{ij}^t$ слід скоригувати таким чином, щоб мережа дійшла в стан конвергентності якомога швидше, щоб досягалися (невідомі) оптимальні значення показників відповідної задачі статичного балансування навантаження, представленої вище.

Однак це дуже не бажано модифікувати ваги каналів. Мета в даному випадку полягає у коригуванні співвідношення розподілу трафіку в мережі таким чином, щоб мережа переходила в конвергентний стан і досягалися (невідомі) оптимальні значення показників відповідної обмеженої (5) та двоїстої (6) задачі оптимізації.

Адаптивний і розподілений алгоритм балансування навантаження. Ми припускаємо, що $w_l$ ваги каналу фіксовані. Для кожної пари вхід-вихід, позначимо множину найкоротших шляхів $P_k^{SP}$ від вузла $s_k$ до вузла $t_k$ по відношенню до ваг зв'язків $w_l$ .

Нехай $\phi_{ij}^t(n)$ позначають коефіцієнти поділу трафіку, які основані на вимірах навантажень каналу зв'язку $y(n) = (y_l(n); l \in L)$. Відзначимо, що навантаження каналів виміряно на всіх вузлах. Таким чином , в нашому адаптивному і розподіленому алгоритмі, для кожного вузла $i$ самостійно визначаються коефіцієнти розщеплення трафіку $\phi_{ij}^t(n)$ для всіх вузлів призначення $t$ і припустимих наступних хопів $j$.

Рішення алгоритму засновані на фукнції вартості $D_p(y)$ [3], визначеної для кожного шляху $P_k^{SP}$:

$$D_p(y) = \max_{l \in p} \frac{y_l}{b_l} \qquad (8)$$

де $y = (y_l; l \in L)$. Це натуральний вибір, оскільки мета в тому, щоби мінімізувати максимальне використання каналу зв'язку (утилізацію). Ідея алгоритму проста і полягає в тому, щоби знизити використання «найдорожчих» каналів шляхом розподілу трафіку на інші канали. Це звичайно має бути зроблено за рахунок збільшення коефіцієнта вартості каналу (метрики), пов'язаної з яким-небудь іншим каналом. В адаптивному регулюванні співвідношення розщеплення у короткий часовий проміжок можливі втрати пакетів. Тим не менш, це може бути вирішено шляхом зміни тільки частини співвідношень розщеплення за цей час, наприклад.

Так як алгоритм є адаптивним, у нас є проблеми управління із зворотним зв'язком: відносини розщеплення (метрики), які залежать від вимірюваних навантажень істотно впливають на результати вимірювань майбутніх навантажень на каналах. Добре відомо, що системи управління зі зворотним зв'язком схильні до нестабільності, якщо посилення в контурі системи занадто великі. Таким чином, щоб уникнути шкідливих коливань, дозволяється змінювати метрики тільки з дуже невеликими кроками. Розмір кроку визначається параметром деталізації $g$. Дрібніша деталізація досягається за рахунок збільшення вартості $g$. Період виміру (5 хвилин, за використання SNMP [2]) має бути достатньо коротким, щоб отримати достатньо швидку збіжність.

Для моніторингу мережі будемо використовувати простий протокол керування мережею (Simple Network Management Protocol — SNMP) — це протокол керування мережами зв'язку на основі архітектури TCP/IP. Саме за допомогою SNMP отримаємо інформацію про різкі підйоми та падіння трафіку.

Алгоритм. Розглянемо роботу алгоритму на невеликому простому прикладі. Модель області мережі OSPF зображено на рисунку 2. Це ієрархічна стандартна область. Нехай було зафіксовано різке критичне зростання трафіку від ланки 0 до ланки 4 через канал 0-5-4. Канали мають різні метрики і пропускні здатності, отже OSPF не зможе виконати статичне балансування самостійно.

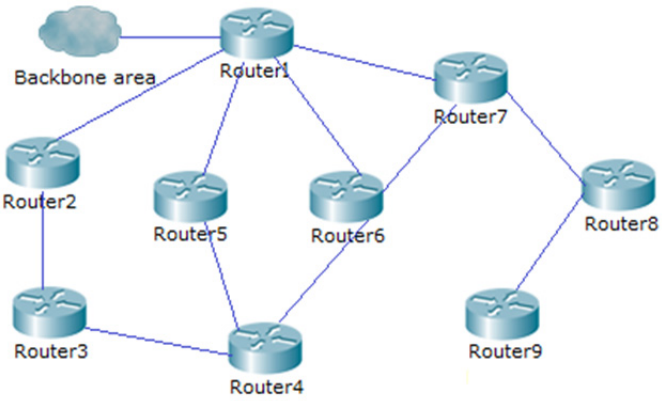



***Рис. 2. Модель області OSPF***

Відомі дані: симетрична матриця зв'язків *AA* (9), матриця *RT*, що складається з масивів, єлементами якого є наступні характеристики ланок: ідентифікаційний номер роутера, номер каналу, метрики, пропускної здатності. Вхідні данні: це ідентифікаційні номери початкового та кінцевого роутерів.

$$AA = \begin{bmatrix} 0 & 1 & 0 & 0 & 1 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix} \qquad (9)$$

Хотілося б звернути увагу, що для кожного випадку алгоритм зберігає рішення в базі даних і перед початком роботи перевіряє чи не має даних по такому випадку в базі. Дані зберігаються і будуть валідними для конкретної області мережі OSPF поки *RT* матриця залишається без змін.

Спочатку маємо проаналізувати, які є канали від початкового роутера 0 до роутера 4. То ж, Алгоритм пошуку шляху в графі буде наступним:

Крок 1. Знаходимо всі роутери, які мають з'єднання з початковим та кінцевим роутером (one-hop routers). Та запам'ятовуємо шлях в окрему матрицю, розмірність якої дорінює розмірності вектору матриці *AA*. В нашому випадку:

$$\begin{bmatrix} 1 & 5 & 4 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 6 & 4 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \qquad (10)$$

Крок 2. Знаходимо всі інші шляхи, але алгорим керується правилом: одна ланка потрапляє в таблицю лише один раз. Пріоритетними є шляхи, що вже потрапили в таблицю на першому кроці, а отже маршрут 1-7-6-4 не потрапить в таблицю. У випадку, коли маршрути не потрапили до таблиці на першому кроці, але всеодно мають спільні ребра, то маршрут обираємо за допомогою алгоритму SPF. Отримуємо наступну матрицю для нашого випадку:

$$\begin{bmatrix} 1 & 2 & 3 & 4 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \qquad (11)$$

Крок 3. Знову використаємо SPF. Проаналізумо отримані маршрути та відсортуємо в порядку, який вкаже алгоритм. Перевіримо чи не має навантаження по трафіку на данних каналах. Так як OSPF підтримує балансування навантаження між чотирма маршрутами з рівною метрикою, то в результуючій таблиці отримаємо, як максимум, чотири найкращі (з точки зору SPF) маршрути. Для прикладу, маємо три наступних:

$$\begin{bmatrix} 1 & 5 & 4 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 6 & 4 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 2 & 3 & 4 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \qquad (12)$$

Маршрути знайдено. На наступному кроці, віддаленим скриптом активуємо підінтерфейси на роутері 0 та роутері 2 та об'єднуємо їх в єдину мережу. Оскільки параметри каналу мають бути однакові, то обираємо канал з найменшими метрикою і пропускною здатністю та встановлюємо такі показники примусово на даному віртуальному каналі. Аналогічні дії виконуємо для роутера 0 та роутера 6. Тепер трафік має піти маршрутами виділеними на рисунку 3.

Визначимо показники продуктивності змодельованої мережі. Ефективність функціонування мережі за час моделювання $t_m$, який містить $k$ тимчасових інтервалів ($t_m = k\Delta t$), оцінювалася за показниками, запропонованими в [7]:

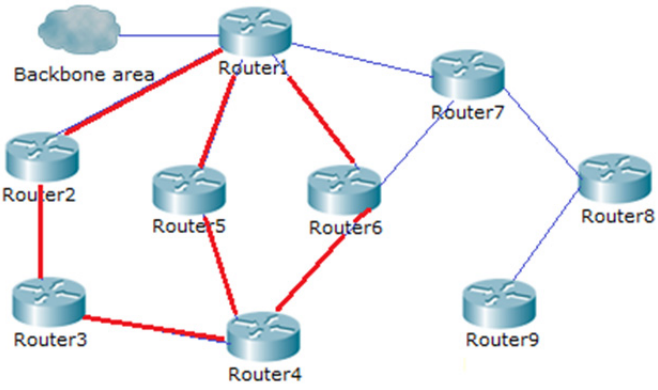



***Рис. 3. Модель балансування трафіку в області OSPF***

1) Нормований обсяг навантаження, що надійшло в мережу ($Y_{вх\ н}$, %):

$$Y_{вх\ н} = \frac{\sum_{k=1}^{\kappa} Y_{вх}(k)}{\Delta t \sum_{k=1}^{\kappa} C(k)} 100\%; \qquad (13)$$

$$Y_{вх}(k) = \sum_{i=1}^{N}\sum_{j=1}^{N} y_{вхi,j}(k)$$

де $Y_{вх}(k)$ величина сумарного навантаження, що надходить в мережу на $k$-му інтервалі; $y_{вхi,j}(k)$ обсяг навантаження, що надходить до вузлів мережі; $C(k)$ пропускна здатність мережі на $k$-му інтервалі .

2) Нормований обсяг навантаження, що вийшов з мережі ($Y_{вих\ н}$, %):

$$Y_{вих\ н} = \frac{\sum_{k=1}^{\kappa} Y_{вих}(k)}{\Delta t \sum_{k=1}^{\kappa} C(k)} 100\%; \qquad (14)$$

$$Y_{вих}(k) = \sum_{i=1}^{N}\sum_{j=1}^{N} y_{вихi,j}(k)$$

де $Y_{вих}(k)$ величина сумарного навантаження, обслугованого мережею на $k$-му інтервалі;

$y_{вихі,j}(k)$ обсяг навантаження, що виконано на окремому вузлі мережі.

3) Нормоване значення продуктивності мережі ($P_н$, %):

$$P_H = \frac{\sum_{k=1}^{\kappa} \mathrm{P(k)}}{\sum_{k=1}^{\kappa} \mathrm{C(k)}} 100\%; \qquad \mathrm{P(k)} = \sum_{i=1}^{N} \sum_{\substack{j=1 \\ j \neq i}}^{N} c_{\mathrm{i,j}}(k) u^{j}{}_{\mathrm{i,j}}(k) \tag{15}$$

де $P(k)$ продуктивність мережі на $k$-му часовому інтервалі.

4) Нормований коефіцієнт використання окремих канальних ресурсів мережі ($\mathrm{K_{i,j\ н}}$) :

$$\mathrm{K_{i,jH}} = \frac{\sum_{k=1}^{\kappa} \mathrm{K_{i,j}(k)}}{K}; \qquad \mathrm{K_{i,j}(k)} = \frac{c_{\mathrm{i,j}}(k) u^{j}{}_{\mathrm{i,j}}(k)}{C(k)} \tag{16}$$

де $\mathrm{K_{i,j}(k)}$ коефіцієнт використання окремих канальних ресурсів.

5) Нормований коефіцієнт використання канальних ресурсів мережі в цілому ($K_{вик\ н}$) :

$$\mathrm{K_{викН}} = \frac{\sum_{k=1}^{\kappa} \mathrm{K_{вик}(k)}}{K} \tag{17}$$

де $K_{вик}(k)$ коефіцієнт використання канальних ресурсів мережі на k-му інтервалі.

Результати моделювання наведені в таблиці 2.

## Висновки

У рамках пропонованої моделі за заданих початкових умов були отримані результати моделювання. Нормований показник продуктивності за весь час моделювання при використанні запропонованого методу динамічного балансування навантаження склав $P_H$ =38.7%, а при відсутності алгоритму балансування в мережі OSPF, $\mathrm{P_H}$ становив 17%. З отриманих результатів видно, що змодельована мережа успішно справляється з навантаженням, що надходить та рівномірно розподіляє його відповідно до штатного алгоритму SPF.

У роботі проаналізоване завдання балансування навантаження і варіанти його вирішення в мережах OSPF , в ході чого було встановлено, що балансування навантаження по маршрутах з різною вартістю в цьому протоколі не вирішується, а винятком є лише випадок з рівними метриками.

У статті запропоновано метод динамічного балансування навантаження для вирішення поставленого завдання. Новизна методу полягає в тому, що він на відміну від використовуваних в протоколі маршрутизації OSPF ресурсів враховує динаміку зміни завантаження буфера і інтерфейсів маршрутизатора. Він здійснює також балансування навантаження між маршрутами з різною метрикою. У загальному випадку пропонований метод балансування може бути застосований в якості налаштування над існуючим протоколом маршрутизації, реалізованим в маршрутизаторі.

Проведено моделювання роботи мережі на основі запропонованої моделі в рамках одної області мережі. За отриманими результатами зроблено висновок про те, що управління на основі запропонованої моделі успішно вирішує завдання балансування навантаження, рівномірно завантажуючи всі доступні канали зв'язку, тим самим підвищуючи показники продуктивності мережі в цілому.

***Табл. 2. Результати моделювання.***

| Порівняння функціонування мережі | Показники продуктивності | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | **У*вх н*, %** | **У*вих н*, %** | **Р*н*, %** | **К*вик н*** | **$К_{12}$** | **$К_{15}$** | **$К_{16}$** | **$К_{64}$** | **$К_{54}$** | **$К_{23}$** |
| До балансування | 17.5 | 6.4 | 12.6 | 0.22 | 0 | 0.9 | 0 | 0 | 0.9 | 0 |
| Після балансування | 17.5 | 15.1 | 38.7 | 0.57 | 0.6 | 0.75 | 0.71 | 0.64 | 0.69 | 0.61 |

## Перелік посилань

1. Томас М. Т. Структура и реализация сетей на основе протокола OSPF. – М.: Издательский дом "Вильяме", 2004. – 816 С.

2. Шмидт К. Дж., Мауро Д. Р. Основы SNMP. – М.: Символ-Плюс, 2005. – 455 С.
3. Susitaival R., Aalto S. Adaptive load balancing with OSPF // HET-NETs 2004. – 2004. – P. 9/1-9/10.
4. Fortz B., Thorup M. Internet Traffic Engineering by Optimizing OSPF Weights // in Proc. Of IEEE Infocom 2000. – 2000. – P 3-5.
5. Wang Y., Wang Z., Zhang L. Internet Traffic Engineering without Full Mesh Overlaying // Proc. of IEEE Infocom 2001. – 2001. – P 1-6.
6. Sridharan A., Guerin R., Diot C. Achieving Near-Optimal Traffic Engineering Solutions for Current OSPF-IS-IS Networks // in Proc. of IEEE Infocom 2003. – 2003. – P 1-8.
7. Евсеева О.Ю. Методы адаптивной маршрутизации в гибридных телекоммуникационных сетях с гарантированным качеством обслуживания: автореф. дис. канд. техн. наук: 05.12.02, Укр. гос. академия железнодорожного транспорта – Х., 2004. – 20 с.

УДК 004.056.53::511.52

*БЕЗШТАНЬКО. В.М.*

# МЕТОД ОПРЕДЕЛЕНИЯ ДОПУСТИМЫХ ЧАСТОСТЕЙ ВОЗНИКНОВЕНИЯ УЩЕРБА ПРИ ОЦЕНКЕ РИСКОВ ИНФОРМАЦИОННОЙ БЕЗОПАСНОСТИ С ПОМОЩЬЮ ГЕНЕТИЧЕСКОГО АЛГОРИТМА

В настоящей статье предлагается метод, позволяющий определять допустимые значения частости возникновения ущерба при оценивании рисков информационной безопасности в условиях неопределенности. Метод основан на использовании линейного неоднородного диофантового уравнения в положительных числах. Его реализацию предложено осуществить с использованием математического аппарата генетического алгоритма. На основе полученных результатов возможно принятие решения о необходимости обработки рисков в системах управления информационной безопасностью.

In this paper we propose a method to determine valid values relative frequency of occurrence of harm when assessing information security risks in an uncertain environment. The method is based on the use of linear inhomogeneous Diophantine equations in positive numbers. Invited to implement its implementation using a genetic algorithm mathematical apparatus. Based on these results possible decision on the need to handle risks in information security management systems.

## 1. Введение

В настоящей статье предлагается метод, позволяющий определять допустимые количественные оценки частости возникновения ущерба в модели для оценки рисков информационной безопасности.

Оценка риска выражается как комбинация частости возникновения ущерба вследствие реализации угрозы и его абсолютной величины. Использование традиционных подходов к решению этой задачи ограничено сложностью накопления статистики нанесения ущерба. Для преодоления этого ограничения разработан метод определения допустимых значений частости на основе линейного, неоднородного диофантового уравнения в положительных числах. Реализацию метода предлагается осуществить с использованием математического аппарата генетического алгоритма. На основе полученных результатов возможно принятие решения о необходимости обработки рисков в системах управления информационной безопасностью.

## 2. Постановка задачи

Перспективным подходом к обеспечению конфиденциальности, целостности и доступности государственных информационных ресурсов является внедрение систем управления информационной безопасностью. Такой подход предполагает оценивание рисков для принятия решения о необходимости их обработки и, как следствие, о выборе соответствующих средств и мероприятий. Для этого руководством организации выбирается метод определения оценок и устанавливаются критерии принятия риска [1, 2].

В большинстве случаев оценка риска выражается качественно или количественно как комбинация частости возникновения ущерба вследствие реализации угрозы и его абсолютной величины. Абсолютная величина ущерба может устанавливаться экспертами. Частость возникновения ущерба определяется по накопленным результатам статистических данных о проишедших инциндентах на эквивалентных обьектах. Такие данные на первоначальном этапе нахождения оценок риска является неизвестным. Поэтому выбор метода определения оценок риска в конечном итоге сводиться к выбору метода определения частости возникновения ущерба [3]. Вместе с тем, применение вероятностного, статистического и экспертного подходов для решения этой задачи ограничено сложностью выполнения условий [4, 5]:

а) стационарности наблюдений за реализациями угроз для накопления статистики возникновения ущерба;

б) использования заимствованной статистики возникновения ущерба вследствие реализации угроз, а именно:

- объекты, к которым предполагается применять статистику, и объекты, на которых собрана статистика, являются эквивалентными (требование эквивалентности объектов);

– условия, при которых предполагается применять статистику и условия ее сбора явля-

ются эквивалентными (требование эквивалентности условий);

– объемы выборок статистики являются достаточными, методы обработки – корректными, а источники сведений – заслуживающие доверия (требование убедительности).

Таким образом, получение соответствующих частостных характеристик осуществляется на основе статистики недостаточного объема, то есть в условиях неопределенности. В связи с этим, целью данной работы является разработка метода определения допустимых частостей нанесения ущерба, вследствие реализации угрозы информационной безопасности для получения модельных оценок рисков в условиях неопределенности.

### 3. Решение задачи

Для достижения поставленной цели предположим, что установлено приемлемое значение величины риска $r_{np}$. Это значение является оценкой возможности обеспечения требуемого уровня информационной безопасности и целей деятельности организации [6]. Тогда сума допустимых значений величины риска для каждого из $n$ информационных ресурсов должна быть меньшей или равной $r_{np}$, то есть

$$r_1 + r_2 + ... + r_j + ... + r_n \le r_{np},\ j \in (1; n), \qquad (1)$$

Исходя из этого предположения, выразим $r_j$ через произведение ущерба $a_j$ на частость $x_j$ его возникновения [7]

$$a_j \cdot x_j \le r_j, \qquad (2)$$

где $a_j \in Z_+$, $x_j \in Z_+$, $r_j \in Z_+$, $Z_+$ – множество положительных целых чисел.

Путем подстановки (2) в (1) получим

$$a_1 \cdot x_1 + a_2 \cdot x_2 + ... + a_j \cdot x_j + ... + a_n \cdot x_n \le r_{np} \qquad (3)$$

Вследствие этого, предельную форму записи (3) можно трактовать как линейное, неоднородное диофантовое уравнение относительно $x_j$ в положительных целых числах

$$a_1 \cdot x_1 + a_2 \cdot x_2 + ... + a_j \cdot x_j + ... + a_n \cdot x_n = r_{np} \qquad (4)$$

где $j$ - номер информационного ресурса или группы информационных ресурсов, для которых проводится оценка рисков.

Предположим, что осуществляется оценка рисков в системе, состоящей из $n$ информационных ресурсов или групп информационных ресурсов (4) и предлагаемый метод определения допустимых частостей возникновения ущерба с помощью математического аппарата генетических алгоритмов [8, 9] состоит из 11 шагов.

**Шаг 1.** Допустимые значения частости возникновения ущерба $x_j$ ($j = 1,2...n$) могут приобретать величины ограниченные следующими условиями:

$$\begin{cases} 1 \le x_1 \le \dfrac{r_{np} - a_2 - a_3 - ... - a_j - ... - a_n}{a_1} \\ 1 \le x_2 \le \dfrac{r_{np} - a_1 - a_3 - ... - a_j - ... - a_n}{a_2} \\ ..... \\ 1 \le x_j \le \dfrac{r - a_1 - a_2 - a_3 - ... - a_n}{a_j} \\ ..... \\ 1 \le x_n \le \dfrac{r - a_1 - a_2 - a_3 - ... - a_j}{a_n}. \end{cases} \qquad (5)$$

**Шаг 2.** Для решения уравнения (4) посредством генетического алгоритма создадим популяцию $P_1$, которая состоит из $k$ - хромосом $p_i$, представляющих собой предполагаемые решения $x_{i1}, x_{i2}, ..x_{ij}...x_{in}$ уравнения (4), где $i$ - номер хромосомы в популяции $(i = 1, 2, ...k)$.

**Шаг 3.** Для каждой хромосомы $p_i$ созданной отцовской популяцией $P_1$ определим значение целевой функции $f_i$ как разницу между суммой произведений значений ущерба $a_j$ на предполагаемые значения частости $x_j$ возникновения ущерба и приемлемым значением риска $r_{np}$. Тогда, для $f_i$ можно записать:

$$f_i(x_{i1}, x_{i2}, ..x_{ij}...x_{in}) = \\ = \left|(a_1 x_{i1} + a_2 x_{i2} + ... + a_j x_{ij} + .... + a_j x_{in}) - r_{np}\right|, \qquad (6)$$

где $f_i(x_{i1}, x_{i2}, ..x_{ij}...x_{in}) \to 0$.

Следовательно, решением уравнения (4) будут $x_{i1}, x_{i2}, ..x_{ij}...x_{in}$ при которых значения целевой функции $f_i(x_{i1}, x_{i2}, ..x_{ij}...x_{in}) = 0$.

**Шаг 4.** Если $f_i(x_{i1}, x_{i2}, ..x_{ij}...x_{in}) \ne 0$, то для оценки сходимости генетического алгоритма [9] определим среднее значение целевой функции популяции

$$f_{P_{ie}} = \frac{\sum_{i=1}^{k} f_i(x_{i1}, x_{i2}, ..x_{ij}...x_{in})}{n} \qquad (7)$$

**Шаг 5.** Далее, выполним селекцию хромосом путем подбора пары хромосом для выполнения операций скрещивания. На практике ча-

сто применяют различные комбинации существующих методов селекции [10]:

1) в случае использования случайной селекции на основе круговой диаграммы – каждая хромосома имеет точку на колесе рулетки, обратно пропорциональную значению целевой функции $f_i(x_{i1}, x_{i2}, ..x_{ij}...x_{in})$. Соответственно, при случайном выборе хромосомы для дальнейшего скрещивания решение имеет тем большую вероятность выбора, чем меньше значение целевой функции $f_i(x_{i1}, x_{i2}, ..x_{ij}...x_{in})$ хромосомы, то есть чем больше величина занимаемого сектора;

2) при селекции по заданной шкале – производится предварительная сортировка множества решений на основе заданного критерия. Каждой хромосоме ставится в соответствие некоторое число, определяющее его роль в популяции;

3) элитная селекция – для дальнейших преобразований на каждом шаге для формирования следующей популяции выбираются хромосомы с наименьшими значениями целевой функции $f_i\ (x_{i1}, x_{i2}, ...x_{ij}...x_{in})$.

4) турнирная селекция – из популяции случайно выбирается некоторое число индивидов, количество которых ограниченно размером “турнира”, поэтому из этой группы хромосомы с наименьшими значениями целевой функции $f_i(x_{i1}, x_{i2}, ...x_{ij}...x_{in})$ отбираются для дальнейших преобразований;

5) инбридинг (близкородственное скрещивание) - в случаях, когда необходимо закрепить какой-либо признак (свойство), полученный в процессе применения генетических операторов, при выполнении селекции выбираются “родственные” хромосомы в наибольшей степени, обладающие рассматриваемым свойством;

6) гибридизация – этот вид селекции применяется в тех случаях, когда в результате применения генетических операторов необходимо получить какое-то новое свойство. В этом случае для выполнения скрещивания выбираются разнородные индивиды - носители свойств, комбинацию которых предполагается получить в итоге.

Для предлагаемого метода определения допустимых значений частости возникновения ущерба выбираем случайную селекцию, как наиболее часто используемую [11]. Для этого, вначале определим обратное значение целевой функции $\frac{1}{f_i\ (x_{i1}, x_{i2}, ..x_{ij}...x_{in})}$ каждой хромосомы $p_i$ в рассматриваемой популяции и сумму их обратных значений $\sum_{i=1}^{k} \frac{1}{f_i(x_{i1}, x_{i2}, ..x_{ij}...x_{in})}$.

Далее найдем вес $w_i$ каждой хромосомы в популяции

$$w_i = \frac{\frac{1}{f_i(x_{i1}, x_{i2}, ..x_{ij}...x_{in})}}{\sum_{i=1}^{k} \frac{1}{f_i(x_{i1}, x_{i2}, ..x_{ij}...x_{in})}}.$$

Используя полученные значения весов $w_i$ хромосом в популяции, построим круговую диаграмму, в которой каждому значению веса $w_i$ хромосомы $p_i$ соответствуют части круговой диаграммы. Создадим материнскую популяцию $P_2$. Для этого, на построенной круговой диаграмме, случайным образом произведем отбор хромосом. Отобранные хромосомы составляют материнскую популяцию. Проведем случайным образом операцию скрещивания между хромосомами популяций отцовской $P_1$ и материнской $P_2$. Количество выбираемых хромосом $p_i$ для скрещивания ограниченно размером $k$ отцовской популяции $P_1$. Соответственно при случайном отборе хромосом, представляющих собой предполагаемое решение $x_{i1}, x_{i2}, ..x_{ij}...x_{in}$ уравнения (4), преимущество выбора имеет та хромосома, у которой больше величина занимаемого сектора круговой диаграммы. Далее, случайным образом выбираем пары хромосом для скрещивания в существующей популяция отцовских $P_1$ хромосом и созданной популяции материнских $P_2$ хромосом. Очевидно, что выбранные для скрещивания хромосомы чаще будут скрещиваться с хромосомами, вес $w_i$ которых больше. С помощью выбранного нами оператора кроссинговера произведем скрещивания хромосом.

Известно достаточно много различных операторов кроссинговера, но наиболее часто используются следующие [8 – 11]:

- стандартный (одноточечный, двухточечный и многоточечный);
- универсальный;
- упорядочивающий;
- циклический;
- жадный.

В работе использован стандартный одноточечный оператор кроссинговера [10] как наиболее простой и часто используемый. Скрещивание проводиться хромосом, представленных в

десятичной системе исчисления. Для проведения такой операции, случайным образом выбирается точка скрещивания между генами в хромосоме из популяции отца и хромосоме из популяции матери. Затем часть генов первого родителя, расположенная левее точки скрещивания, и часть генов второго родителя, расположенная правее точки скрещивания, копируются в первого потомка. Второй потомок формируется из правой части генов первого родителя и левой части генов второго родителя. Таким образом, путем перекомпоновки генов двух старых родительских хромосом генерируются две новые хромосомы потомков. Родительские хромосомы перед операцией кроссинговера будут выглядеть, как показано в таблице 1. Точка скрещивания выбирается случайным образом. Например, пусть, родительская хромосома $p_1$ из популяции отца $P_1$ имеет вид $p_1(P_1):\langle x_1;x_2;x_3\rangle$, хромосома $p_1$ из созданной популяции матери $P_2$ примет вид $p_1(P_2):\langle x_1';x_2';x_3'\rangle$. Случайным образом выберем точку скрещивания в $p_1(P_1)$ и $p_1(P_2)$. Пусть хромосомы родителей будут скрещиваться между вторым и третьим генами. Меняя гены двух родительских хромосом, до или после одноточечного кроссинговера, создадим новые хромосомы потомков $p_1'$ и $p_2'$.

***Табл.1. Кроссинговер***

| Хромосомы | Гены | | |
|---|---|---|---|
| $p_1(P_1)$ | $x_1$ | $x_2$ | $x_3$ |
| $p_1(P_2)$ | $x_1'$ | $x_2'$ | $x_3'$ |
| $p_1'(P_1)$ | $x_1$ | $x_2$ | $x_3'$ |
| $p_1'(P_2)$ | $x_1'$ | $x_2'$ | $x_3$ |

Тогда, в популяцию потомков $P_3$ войдут хромосомы имеющие вид $p_1'(P_1):\langle x_1;x_2;x_3'\rangle$ и $p_1'(P_2):\langle x_1';x_2';x_3\rangle$.

**Шаг 6.** Вычислим значение целевой функции (5) хромосом популяции потомков $P_3$.

**Шаг 7.** Из хромосом, которые составляют популяции родителей $P_1$, $P_2$ и потомков $P_3$ сформируем новую популяцию $P_4$. В популяцию $P_4$ отберем неповторяющиеся хромосомы с наименьшими значениями целевой функции $f_i\ (x_{i1},x_{i2},..x_{ij}...x_{in})$. Количество отобранных хромосом ограничим размером $k$ отцовской популяции $P_1$.

**Шаг 8.** Вопрос преодоления локальных минимумов и зацикливания работы алгоритма решается введением в популяцию новых хромосом. Для этого в генетическом алгоритме используются операторы мутации и инверсии. Следует отметить, что на сегодняшний день, в основном используются следующие операторы мутации [9, 11]:

1. **Простой мутации.** При простой мутации в векторной хромосоме случайным образом выбирается ген, а затем производится случайное изменение его числового значения. Известен также модифицированный оператор простой мутации, называемый **точечной мутацией**, который отличается тем, что в хромосоме подвергается мутации не один, а несколько генов с заданной вероятностью.

2. **Мутации обмена**. Если используется оператор **мутация обмена**, то в хромосоме случайным образом выбираются два гена, которые меняются местами. Можно также выделить **одноточечную мутацию обмена**, где, в отличие от двухточечной, обмениваются местами только соседние гены, и точка мутации выбирается между двумя генами.

3. **Золотого сечения.** В подобных операторах выбор точки мутации осуществляется на основе правила “золотого сечения”, т.е. точка мутации определяется отношением $\frac{L}{1{,}61803}$; где $L$ – длина хромосомы, определяемая количеством генов. Так, для хромосомы, состоящей из трех генов $L=3$, $\frac{3}{1{,}61803}=1{,}85$ точка мутации будет находиться между первым и вторым геном. Это значит, что при выполнении мутации золотого сечения в хромосоме первый и второй ген поменяются местами. На практике применение тех или иных операторов мутации обуславливается характером решаемой задачи. В предлагаемом методе определения допустимых значений частостей возникновения ущерба мутация генов происходит в одной хромосоме с наибольшим (наихудшим) значением функции приспособленности $f_i\ (x_{i1},x_{i2},..x_{ij}...x_{in})$, созданной популяции $P_4$. Выполнение операции

происходит посредством использования простой мутации или правила золотого сечения, с вероятностью 0,5 наступления того или иного события.

**Шаг 9.** Для воздействия на хромосому оператором инверсии [11] определим две точки в хромосоме, которые являются границами инверсии. Затем определим точку в середине участка хромосомы выделенного для инверсии. Далее гены хромосомы повернем относительно точки инверсии на 180°.

В предлагаемом методе инверсия проводится в каждой сформированной популяции $P_4$ в одной хромосоме выбранной случайным образом. Для этого, в хромосоме случайным образом определяются две точки среза и участок, ограниченный точками среза, поворачивается на 180° (инвертируется). Определим значение целевой функции $f_i$ (6) хромосом популяции $P_4$, на которых воздействовали операторами мутации и инверсии.

**Шаг 10.** Определим среднее значение целевой функции $f_{p_{4c}}$ популяции $P_4$ (7). Проведем сравнение значений целевых функций первой $f_{p_{1c}}$ и созданной $f_{p_{4c}}$ популяций. Если с каждым поколением среднее значение целевой функции популяции стремится к нулю $f_{p_{ie}} \to 0$, то алгоритм работает успешно.

**Шаг 11**. В случае если среднее значение целевой функции $f_{p_{4c}}$ популяции сформированной из наилучших хромосом популяций родителей $P_1$ и $P_2$, детей $P_3$, после применения операторов мутации и инверсии увеличивается или не изменяется $f_{p_{1c}} \geq f_{p_{4c}}$, то в популяции $P_4$ проводится мутация $n$ хромосом с наибольшими значениями целевой функции за правилом, определенным в шаге 7.

Рассмотрим примение предлагаемого метода поиска допустимых значений частости возникновения ущерба на примере модели оценки риска состоящей из трех активов.

$$2x_1 + 3x_2 + 4x_3 = 54 \qquad (8)$$

где, приемлемое значение риска $r_{np} = 54$, значения ущерба $a_j = \{2,3,4\}$ соответственно, $x_1, x_2, x_3$ - неизвестные значения частости возникновения ущерба.

**Шаг 1** Допустимые частости возникновения ущерба $x_j$ могут принимать значения в пределах накладываемых ограничений обусловленных видом уравнения (5):

$$1 \leq x_1 \leq \frac{r - a_2 - a_3}{a_1} \text{ или } 1 \leq x_1 \leq \frac{54-3-4}{2};$$

$$1 \leq x_2 \leq \frac{r - a_1 - a_3}{a_2} \text{ или } 1 \leq x_2 \leq \frac{54-2-4}{3};$$

$$1 \leq x_3 \leq \frac{r - a_1 - a_2}{a_3} \text{ или } 1 \leq x_3 \leq \frac{54-2-3}{4}.$$

Следовательно, $1 \leq x_1 \leq 24$, $1 \leq x_2 \leq 16$, $1 \leq x_3 \leq 12$.

**Шаг 2.** Создадим популяцию $P_1$ с шести хромосом, состоящей из произвольных чисел в заданных для $x_1, x_2, x_3$ пределах (табл.2).

***Табл. 2. Популяция*** $P_1$

| № п./п. хромосомы | $P_1 \in (x_{i1}, x_{i2}, x_{i3})$ |
|---|---|
| 1 | 20, 15, 6 |
| 2 | 23, 12, 3 |
| 3 | 5, 7, 11 |
| 4 | 11, 3, 8 |
| 5 | 2, 13, 10 |
| 6 | 7, 8, 5 |

**Шаг 3** Определим значения целевой функции $f_i$ для каждой из хромосом популяции $P_1$.

$$f_1 = (2 \cdot 20 + 3 \cdot 15 + 4 \cdot 6) - 54 = 109 - 54 = 55$$
$$f_2 = (2 \cdot 23 + 3 \cdot 12 + 4 \cdot 3) - 54 = 100 - 54 = 46$$
$$f_3 = (2 \cdot 5 + 3 \cdot 7 + 4 \cdot 11) - 54 = 75 - 54 = 21$$
$$f_4 = (2 \cdot 11 + 3 \cdot 3 + 4 \cdot 8) - 54 = 63 - 54 = 9$$
$$f_5 = (2 \cdot 2 + 3 \cdot 13 + 4 \cdot 10) - 54 = 83 - 54 = 29$$
$$f_6 = (2 \cdot 7 + 3 \cdot 8 + 4 \cdot 5) - 54 = 58 - 54 = 4.$$

Так как (8) не имеет решения, значения целевых функций хромосом $f_i$ популяции $P_1$ не равны нулю, то переходим к шагу 4.

**Шаг 4**. Найдем среднее значение целевой функции $f_{P_{1c}}$ популяции.

$$f_{p_{1c}} = \frac{55 + 46 + 21 + 9 + 29 + 4}{6} = 27{,}33$$

**Шаг 5**. Из популяции $P_1$ выполним отбор хромосом для скрещивания методом случайной селекции. Для этого определим вес $w_i$ каждой хромосомы популяции $P_1$ (табл. 3).

***Табл.3. Вес хромосом $w_i$ популяции $P_1$***

| № п./п. хромосомы | $\frac{1}{f_i(x_{in})}$ | $\sum_{i=1}^{6}\frac{1}{f_i(x_{in})}$ | $w_i$ |
|---|---|---|---|
| 1 | 0,018 | 0,497 | 0,036 |
| 2 | 0,022 | | 0,044 |
| 3 | 0,048 | | 0,097 |
| 4 | 0,111 | | 0,223 |
| 5 | 0,048 | | 0,097 |
| 6 | 0,250 | | 0,503 |

Используя полученные значения весов хромосом $w_i$ популяции $P_1$, построим круговую диаграмму (рис.1).

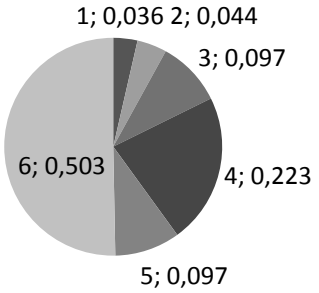



***Рис. 1 Круговая диаграмма веса хромосом популяции $P_1$.***

Из рисунка видно, что части круговой диаграммы соответствуют значениям веса хромосом. Соответственно, при случайном выборе, решение уравнения (8) обладает тем больше вероятностью выбора, чем больше величина занимаемого сектора круговой диаграммы.

Для выполнения операций скрещивания случайным образом выбираем на интервале $\{0,\ 1\}$ хромосому, занимающую соответствующий сектор на круговой диаграмме. Очевидно, что выбранные для скрещивания хромосомы чаще будут скрещиваться с хромосомами, вес $w_i$ которых больше. В рассматриваемом случае необходимо выполнить шесть операций. Выбранные для скрещивания хромосомы будут представителями популяции матери $P_2$ (табл.4).

***Табл.4. Популяция $P_2$***

| № п/п хромосомы | $P_2 \in (x_{i1}, x_{i2}, x_{i3})$ |
|---|---|
| 1 | 7, 8, 5 |
| 2 | 2, 13, 10 |
| 3 | 7, 8, 5 |
| 4 | 5, 7, 11 |
| 5 | 7, 8, 5 |
| 6 | 11, 3, 8 |

С помощью одноточечного кроссинговера произведем скрещивание хромосом из отцовской $P_1$ и материнской $P_2$ популяций. Точку скрещивания определим произвольно. Вследствие проведенных операций получим популяцию потомков $P_3$ (табл. 5, / - точка скрещивания).

***Табл.5. Скрещивание хромосом***

| № п./п. | Хром. $P_1$ | Хром. $P_2$ | Хром. $P_3$ |
|---|---|---|---|
| 1 | 20, 15, / 6 | 7, 8, / 5 | 20, 15, 5<br>7, 8, 6 |
| 2 | 23, / 12, 3 | 2,/ 13, 10 | 23, 13, 10<br>2, 12, 3 |
| 3 | 5, / 7, 11 | 7, / 8, 5 | 5, 8, 5<br>7, 7, 11 |
| 4 | 11, 3, /8 | 5, 7, /11 | 11, 3, 11<br>5, 7, 8 |
| 5 | 2, 13,/ 10 | 7, 8,/ 5 | 2, 13, 5<br>7, 8, 10 |
| 6 | 7, / 8, 5 | 11, / 3, 8 | 11 8, 5<br>7, 3, 8 |

**Шаг 6**. Для каждой из хромосом популяции потомков $P_3$ определим значение целевой функции $f_i$ .

$$f_1 = (2\cdot 20 + 3\cdot 15 + 4\cdot 5) - 54 = 105 - 54 = 51$$
$$f_2 = (2\cdot 7 + 3\cdot 8 + 4\cdot 6) - 54 = 79 - 54 = 8$$
$$f_3 = (2\cdot 23 + 3\cdot 13 + 4\cdot 10) - 54 = 125 - 54 = 71$$
$$f_4 = (2\cdot 2 + 3\cdot 12 + 4\cdot 3) - 54 = 52 - 54 = -2$$
$$f_5 = (2\cdot 5 + 3\cdot 8 + 4\cdot 5) - 54 = 54 - 54 = 0$$
$$f_6 = (2\cdot 7 + 3\cdot 7 + 4\cdot 11) - 54 = 79 - 54 = 25$$
$$f_7 = (2\cdot 11 + 3\cdot 3 + 4\cdot 11) - 54 = 75 - 54 = 21$$
$$f_8 = (2\cdot 5 + 3\cdot 7 + 4\cdot 8) - 54 = 63 - 54 = 9$$
$$f_9 = (2\cdot 2 + 3\cdot 13 + 4\cdot 5) - 54 = 63 - 54 = 9$$
$$f_{10} = (2\cdot 7 + 3\cdot 8 + 4\cdot 10) - 54 = 78 - 54 = 24$$
$$f_{11} = (2\cdot 11 + 3\cdot 8 + 4\cdot 5) - 54 = 66 - 54 = 12$$
$$f_{12} = (2\cdot 7 + 3\cdot 3 + 4\cdot 8) - 54 = 55 - 54 = 1$$

Значение целевой функции пятой хромосомы из популяции детей $P_3$ $f_5 = 0$. Значит, частным решением уравнения (8) будут числовые значения $\{5, 8, 5\}$.

Для демонстрации всех этапов работы предлагаемого метода поиска частостей возникновения значений ущерба с помощью генетического алгоритма рассмотрим возможные варианты решения уравнения (8). Для этого, повторим шаги 2-6. Результаты операций представлены в табл. 6, 7 и 8.

***Таб.6. Вес хромосомы в популяции***

| № | $P_1$ | $f_i$ | $f_{p_1}$ | $\frac{1}{f_i(x_{in})}$ | $\sum_{i=1}^{k}\frac{1}{f_i(x_{in})}$ | $w_i$ |
|---|---|---|---|---|---|---|
| 1 | 1, 8, 4 | -12 | 19,5 | 0,083 | 0,441 | 0,188 |
| 2 | 14, 7, 6 | 19 | | 0,053 | | 0,120 |
| 3 | 19, 2, 7 | 18 | | 0,056 | | 0,127 |
| 4 | 10, 8, 1 | -6 | | 0,167 | | 0,379 |
| 5 | 7, 11, 4 | 17 | | 0,059 | | 0,137 |
| 6 | 22, 9, 7 | 45 | | 0,023 | | 0,052 |

Для популяции $P_1$ построим круговую диаграмму (рис.2):

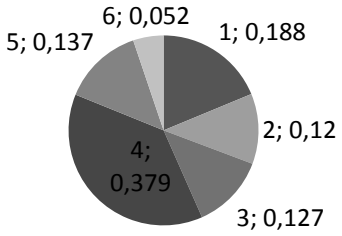



***Рис. 2 Круговая диаграмма веса хромосом популяции $P_1$.***

Выберем для скрещивания хромосомы в материнскую популяцию $P_2$ . Произведем скрещивание и получим популяцию детей $P_3$ (табл. 7).

***Табл.7. Скрещивание хромосом***

| № п/п | $P_1$ | $P_2$ | $P_3$ |
|---|---|---|---|
| 1 | 1, 8, / 4 | 10, 8, / 1 | 1, 8, 1<br>10, 8, 4 |
| 2 | 14, 7, / 6 | 7, 11, /4 | 14, 7, 4<br>7, 11, 6 |
| 3 | 19, / 3, 7 | 10, / 8, 1 | 19, 8, 1<br>10, 3, 7 |
| 4 | 10,/ 8, 1 | 1, /8, 4 | 10, 8, 4<br>1, 8, 1 |
| 5 | 7, 11, /4 | 14, 7, /6 | 7, 11, 6<br>14, 7, 4 |
| 6 | 22,/ 9, 7 | 19, /2, 7 | 22, 2, 7<br>19, 9, 7 |

Примечание: / - точка скрещивания.

Для каждой хромосомы из популяции детей $P_3$ определим значение целевой функции $f_i$ (табл. 8).

***Табл.8. Значение целевой функции популяции $P_3$***

| № п/п | $P_3$ | $f_i$ |
|---|---|---|
| 1 | 1, 8, 1 | -24 |
| 2 | 10, 8, 4 | 6 |
| 3 | 14, 7, 4 | 11 |
| 4 | 7, 11, 6 | 71 |
| 5 | 19, 8, 1 | 12 |
| 6 | 10, 3, 7 | 3 |
| 7 | 10, 8, 4 | 6 |
| 8 | 1, 8, 1 | -24 |
| 9 | 7, 11, 6 | 17 |
| 10 | 14, 7, 4 | 11 |
| 11 | 22, 2, 7 | 24 |
| 12 | 19, 9, 7 | 39 |

**Шаг 7**. Далее, из хромосом популяции родителей $P_1$, $P_2$ и детей $P_3$ сформируем новую популяцию $P_4$. Для этого отберем шесть неповторяющихся хромосом (табл. 7 и 8) с наименьшими (наилучшими) значениями целевой функции $f_i$ (табл. 9).

***Табл.9. Отбор хромосом в новую популяцию $P_4$***

| № п/п хромосомы | $P_4$ | $f_i$ |
|---|---|---|
| 1 | 1, 8, 4 | -12 |
| 2 | 10, 8, 1 | -6 |
| 3 | 14, 7, 4 | 11 |
| 4 | 19, 8, 1 | 12 |
| 5 | 10, 3, 7 | 3 |
| 6 | 10, 8, 4 | 6 |

**Шаг 8.** В популяции $P_4$ проведем мутацию первой хромосомы, имеющую наибольшее по модулю (наихудшее) значение целевой функции. Допустим, что необходимо провести мутацию с использованием правила золотого сечения. Так, как хромосома состоит из трех генов $L=3$, то по правилу золотого сечения $\frac{3}{1{,}61803}=1{,}85$ точка мутации будет находиться между первым и вторым геном. Это значит, что при выполнении мутации золотого сечения в хромосоме первый и второй ген поменяются местами. То есть, первая хромосома популяции $P_4$ примет вид $p_1\langle 8,1,4\rangle$.

**Шаг 9.** Для проведения инверсии случайным образом выберем хромосому из популяции $P_4$. Пусть это будет хромосома $p_6\langle 10,8,4\rangle$ (табл. 9). Далее предположим, что случайным образом определили точки среза в выбранной хромосоме. Пусть, они будут перед первым и между вторым и третьим генами $p_6\langle /10,8,/4\rangle$.Тогда, после инверсии, шестая хромосома будет иметь вид $p_6\langle 8,10,4\rangle$. Определим значение целевой функции хромосом популяции $P_4$, на которых воздействовали операторами мутации и инверсии (табл. 10).

В результате выполнения операций мутаций и инверсий частного решения уравнения (8) найдено не было.

**Шаг 10**. На основе полученных значений целевой функции хромосом популяции $P_4$

(табл. 10) определим среднее значение целевой функции популяции $P_4$:

$$f_{P_{4c}} = \frac{\sum_{i=1}^{k} f_i(x_{in})}{k} = \frac{19+6+11+12+3+8}{6} = 9{,}84.$$

***Табл.10. Значение целевой функции хромосом популяции*** $P_4$

| № п/п хромосомы | $P_4$ | $f_i$ |
|---|---|---|
| 1 | 8, 1, 4 | -19 |
| 2 | 10, 8, 1 | -6 |
| 3 | 14, 7, 4 | 11 |
| 4 | 19, 8, 1 | 12 |
| 5 | 10, 3, 7 | 3 |
| 6 | 8, 10, 4 | 8 |

Проведем сравнение значений целевых функций первой $f_{p_1} = 27{,}33$ и созданной $f_{p_4} = 9{,}84$ популяций. Среднее значение целевой функции четвертой популяции $f_{p_4} < f_{p_1}$, значит, алгоритм работает успешно, переходим к выполнению шагов 5-10.

**Шаг 11**. В случае если среднее значение целевой функции $f_{p_4}$ популяции, сформированной из наилучших хромосом популяций родителей $P_1$, $P_2$ и детей $P_3$, после применения операторов мутации и инверсии увеличивается или не изменяется $f_{p_4} \geq f_{p_4}$, то в популяции $P_4$ проводится мутация $n$ хромосом с наибольшими значениями целевой функции за правилом определенным на шаге 7. Выполнение циклов алгоритма приведет к поиску всех значений частости возникновения ущерба.

### 3. Экспериментальная часть

В программной реализации метода поиска допустимых значений частости возникновения ущерба произведем подсчет количества операций умножения/деления $K_{gu}$ и сложения/вычитания $K_{gu}$.

Во время проведения эксперимент значения частости возникновения ущерба $x_j \ (j = 1,2...n)$ ограниченны условиями (5).

Опыты проведем для уравнения (4), где количество информационных ресурсов или групп информационных ресурсов $n$ изменяется от 2 до 6. Приемлемое значение риска $r_{пр} = 129$. Величина ущерба $a_j$ приобретает значения $\{2;3;4;5;6;7\}$.

Для рассматриваемых уравнений вида (4) согласно (5) определим ограничения для $x_j$, табл. 11.

Для метода поиска допустимых значений частостей, основанного на генетическом алгоритме, проведем выбор параметров в соответствии с результатами ранее полученны опытов [11]. При проведения эксперимента введем следующие ограничения:

– хромосомы состоят из чисел, представляющие собой допустимые значения частости возникновения ущерба $x_1; x_2; ...; x_j ...; x_n$ в десятичной форме исчисления;

– популяция предполагаемых допустимых значений частости ограничивается 200 хромосомами [11];

– точка кроссинговера в хромосоме определяется случайным образом с помощью встроенного в C# генератора случайных чисел;

***Таб.11. Ограничения для*** $x_j$ ***при изменяющихся*** $n$

| Количество информационных ресурсов $n$ | Вид уравнения | Ограничения $x_j$ |
|---|---|---|
| 2 | $2 \cdot x_1 + 3 \cdot x_2 = 129$ | $1 \le x_1 \le 63$<br>$1 \le x_2 \le 42$ |
| 3 | $2 \cdot x_1 + 3 \cdot x_2 + 4 \cdot x_3 = 129$ | $1 \le x_1 \le 61$<br>$1 \le x_2 \le 43$<br>$1 \le x_3 \le 31$ |
| 4 | $2 \cdot x_1 + 3 \cdot x_2 + 4 \cdot x_3 + + 5 \cdot x_4 = 129$ | $1 \le x_1 \le 58$<br>$1 \le x_2 \le 39$<br>$1 \le x_3 \le 29$<br>$1 \le x_4 \le 24$ |
| 5 | $2 \cdot x_1 + 3 \cdot x_2 + 4 \cdot x_3 + + 5 \cdot x_4 + 6 \cdot x_5 = 129$ | $1 \le x_1 \le 55$<br>$1 \le x_2 \le 37$<br>$1 \le x_3 \le 28$<br>$1 \le x_4 \le 22$<br>$1 \le x_5 \le 19$ |
| 6 | $2 \cdot x_1 + 3 \cdot x_2 + 4 \cdot x_3 + + 5 \cdot x_4 + 6 \cdot x_5 + 7 \cdot x_6 = 129$ | $1 \le x_1 \le 52$<br>$1 \le x_2 \le 35$<br>$1 \le x_3 \le 26$<br>$1 \le x_4 \le 21$<br>$1 \le x_5 \le 18$<br>$1 \le x_6 \le 15$ |

– мутация генов происходит в каждой итерации (поколении) в одной хромосоме с худшим значением целевой функции, в популяции с уже отобранными лучшими решениями, с использованием простой мутации или правила золотого сечения, с вероятностью 0,5 наступления того или иного события;

– инверсия проводится в каждой итерации (поколении) в одной хромосоме, выбранной генератором случайных чисел. Для этого в ней, с помощью генератора случайных чисел, определяются две точки среза и участок, ограниченный точками среза, поворачивается на 180° (инвертируется);

– значение величин ущерба $a_1; a_2; ..a_j ...; a_n$ не меняется во время поиска решений, и задается только с добавлением нового актива в исследуемое уравнение;

– величина приемлемого риска $r_{np}$ определяется пользователем;

– при попадании алгоритма в локальный минимум проводится мутация 10 хромосом с худшим значением целевой функции по правилу определенном в абзаце 4 данных ограничений;

– работа алгоритма прекращается при нахождении всех возможных допустимых значений частости возникновения ущерба.

Результаты исследования уравнения (4) с помощью генетического алгоритма представлены в таблице 12.

***Табл.12. Значения $K_{gu}$ и $K_{gs}$ при изменяющихся $n$***

| $n$ | $K_{gu}$ | $K_{gs}$ |
|---|---|---|
| 2 | 46800 | 23400 |
| 3 | 954100 | 408900 |
| 4 | 6645600 | 2492100 |
| 5 | 24792200 | 11597400 |
| 6 | 151924000 | 45577200 |

Графическое отображение результатов исследования отображено на рисунке 3.

## *4. Заключение*

Таким образом, в работе предложен метод определения допустимых значений частости возникновения ущерба вследствие реализации угрозы информационной безопасности на основе линейного, неоднородного диофантового уравнения в положительных целых числах. Для сокращения полного перебора множества решений этого уравнения использован математический аппарат генетических алгоритмов. Результаты проведенных экспериментов подтверждают работоспособность предложенного метода.

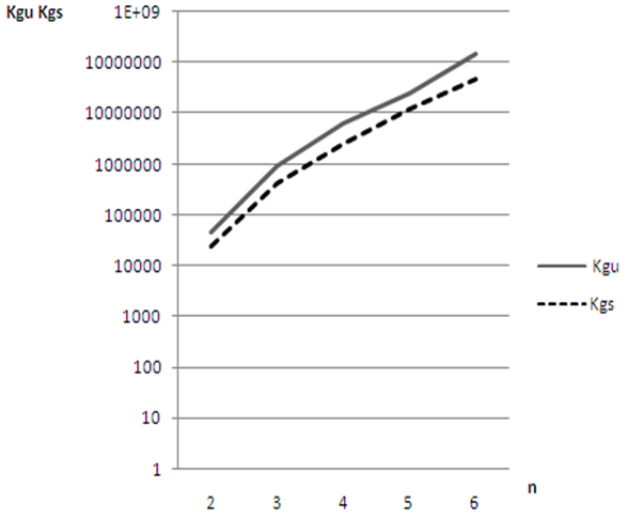



***Рис 3. Зависимость $K_{gu}$ и $K_{gs}$ от $n$***

Определение вычислительной сложносности и сравнительный анализ эффективности предложенного метода с известными методами решения неоднородных положительных диофантовых уравнений является целью дальнейших исследований. Предложенный метод позволяет определять допустимые значений частости возникновения ущерба для получения модельных оценок рисков, в условиях неопределенности. Благодаря этому возможно принятие решения о необходимости их обработки в системах управления информационной безопасностью.

## Список литературы

1. Методи захисту в банківській діяльності. Система управління інформаційною безпекою. Вимоги: (ISO/IEC 27001:2005, MOD): СОУ Н НБУ 65.1 СУІБ 1.0:2010 – Чинний з 2010-10-28. – К.: Національний банк України, 2010. – 59 с. – (Стандарт організації України).
2. Методи захисту в банківській діяльності. Звід правил для управління інформаційною безпекою: (ISO/IEC 27002:2005, MOD): СОУ Н НБУ 65.1 СУІБ 2.0:2010 – Чинний з 2010-10-28. – К. : Національний банк України, 2010. – 195 с. – (Стандарт організації України).
3. Методы и средства обеспечения безопасности. Менеджмент риска информационной безопасности [Электронный ресурс]: (ISO/IEC 27005:2008, IDT): ГОСТ Р ИСО/МЭК 27005-2010 – Действующий с 2010-11-30. –

М.: Стандартинформ, 2011. – Режим доступа: http://docs.cntd.ru/document/1200084141. – Дата доступа: май. 2013. – Название с экрана.

4. Вишняков Я.Д. Общая теория рисков: учеб.пособие для студ. высш. учеб. заведений / Я. Д. Вишняков, Н. Н. Радаев. – М.: Издательский центр «Академия», 2007. – 368 с.
5. Мохор В.В. Построение оценок рисков безопасности информации на основе динамического множества актуальных угроз / В.В. Мохор, А.М.Богданов, О.Н. Крук, В.В. Цуркан // Збірник наукових праць Інституту проблем моделювання в енергетиці ім. Г.Є. Пухова. – К.: ІПМЕ ім. Г. Є. Пухова НАН України, 2010. – Вип. 56. – С. 87–99.
6. Безштанько В. М. Определение приемлемого значения риска для информационных активов организации/ В. М. Безштанько //Збірник наукових праць ІПМЕ ім. Г.Є. Пухова НАН України. –2013 (в друку).
7. Безштанько В. М. Анализ условий разрешимости неоднородного положительного диофантового уравнения при моделировании рисков безопасности информации / В. М. Безштанько // Моделювання та інформаційні технології. – К.: ІПМЕ ім. Г.Є. Пухова НАН України,2012. – Вип. 66. – С. 92 – 96.
8. Рутковская Д. Нейронные сети, генетические алгоритмы и нечеткие системы: /Рутковская Д., Пилинський М., Рутковский Л./ Пер с польского И.Д. Рудинского – М.:Горячая линия – Телеком, 2006. – 452с.
9. Генетический алгоритм: боремся с преждевременной сходимостью. [Електронний ресурс]. Режим доступапhttp://habrahabr.ru/post/122222/ – Дата доступа: июнь 2013. – Название с экрана.
10. Емельянов В.В. Теория и практика эволюционного моделирования. / В.В. Емельянов, В.В. Курейчик, В.М. Курейчик – М: ФИЗМАТЛИТ, 2003-432с.
11. Безштанько В.М. Анализ результатов испытаний генетических алгоритмов при количественном расчете рисков информационной безопасности в условиях отсутствия статистических данных о частости реализации угроз/ В.М.Безштанько, С.И.Душкевич // Національна академія наук України, Збірник наукових праць, Інститут проблем моделювання в енергетиці ім. Г.Є.Пухова. Випуск 60, Київ 2011г., с 69-75.

УДК 004.89

*ДОРОГИЙ Я.Ю.,*
*ЦУРКАН В.В.,*
*ХРЕНОВ О.І.*

# РОЗРОБКА АРХІТЕКТУРИ СИСТЕМИ ДЛЯ МОДЕЛЮВАННЯ ЗГОРТОЧНИХ НЕЙРОННИХ МЕРЕЖ

Розглядається процес розробки архітектури системи, що допоможе у побудові згорточних нейронних мереж (ЗНМ), їх моделюванні та отриманні оброблених результатів.

The process of developing the system architecture, which will help in building of convolutional neural networks (CNN), their modeling and getting the processed results.

## Вступ

У наш час, коли майже у кожному портативному пристрої є цифрова камера з можливістю зйомки фото та відео матеріалів, обсяги інформації зростають з величезною швидкістю, тому виникає проблема у автоматичному розпізнаванні зображень та їх класифікації. Глобально ці проблеми відносяться до терміну «машинне навчання» («Machine Learning»). Постійні публікації у даній сфері підтверджують актуальність даної проблеми.

Як приклад, у жовтні 2012 року відбувся конкурс ImageNet [1], що був присвячений класифікації об'єктів на фотографіях. У конкурсі було потрібно розпізнавання образів в 1000 категорій. Команда переможця Джефрі Хинтона використовувала методи поглибленого вивчення (Deep Learning) та згорточних нейронних мереж (Convolutional Neural Networks) [2].

У березні 2013 року Google інвестував в проект Хінтона, заснований при університеті Торонто. Протягом шести місяців був розроблений сервіс пошуку по фотографіях photos.google.com. Напевно, у кожного є величезний архів фотографій, тепер можна з легкістю їх класифікувати, що допоможе у майбутньому використанні цих матеріалів.

Та повернімось до методів, за допомогою яких ця можливість тепер є досяжною. Зокрема, це згорточні нейронні мережі – окремий клас нейронних мереж, який найкращим чином підходить для інтелектуальної обробки візуальних даних. ЗНМ були розроблені професором Яном Лекуном в кінці 1990-х років. Вже тоді ця технологія дозволяла надійно вирішувати задачі розпізнавання рукописних текстів. З тих пір значно збільшилася потужність комп'ютерів і з'явилися нові алгоритми навчання нейронних мереж.

Також варто зазначити, що Джефрі Хінтон був одним з дослідників, хто запропонував використовувати метод зворотного поширення помилки для тренування нейронних мереж.

## Мета дослідження

Об’єктом дослідження даної роботи є процеси розпізнавання та класифікації зображень, а предметом дослідження – використання ЗНМ для розпізнавання та класифікації зображень.

Мета даної роботи – розробка інструментарію для моделювання ЗНМ. Для досягнення поставленої мети виконаний пошук і порівняння існуючих бібліотек для роботи з ЗНМ, наступним кроком був вибір додаткових компонент, а після цього – розробка зручного інтерфейсу, що об’єднує обрані інструменти.

## ЗНМ та її застосування

***Історія виникнення.*** Згорточна нейронна мережа представляє особливий клас нейронних мереж, який найкращим чином підходить для інтелектуальної обробки візуальних даних.

У 1981 році нейробіологи Торстен Візел та Девід Хабель досліджували зорову кору головного мозку кішки і виявили, що існують так звані прості клітини, які особливо сильно реагують на прямі лінії під різними кутами і складні клітини, що реагують на рух ліній в одному напрямку.

Пізніше Ян Лекун запропонував використовувати так звані згорточні нейронні мережі, як аналог зорової кори головного мозку для розпізнавання зображень [3,4,5].

Ян Лекун і його співробітники зробили дійсно гарну роботу по розпізнанню почерку, використовуючи ЗНМ, що були застосовані на практиці. Це один з небагатьох на той час прикладів

з застосуванням комп'ютера для тренування нейронних мереж, що мали добрий результат.

***LeNet-5.*** Перший вражаючий приклад використання ЗНМ належить Яну Лекуну та його співробітниками, які створили дійсно добрий розпізнавач написаних від руки цифр. Ця мережа була використана для читання ~10% чеків в Північній Америці [6, 8].

Розроблена ЗНМ мала наступні особливості:

- велика кількість прихованих шарів;
- багато карт ознак у кожному шарі;
- об'єднання виходів близьких виділених ознак;
- виконувала розпізнавання, навіть, якщо символи перекривались;
- навчання виконувалось не окремих символів, а повного напису;
- у якості тренувального алгоритму використовувався алгоритм зворотного поширення помилки.

На рис.1. наведена архітектура LeNet-5.

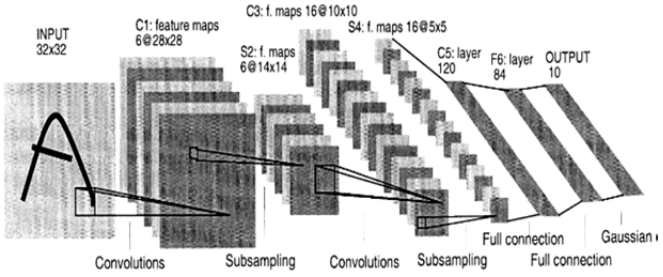



**Рис. 1. Архітектура LeNet-5**

При проходженні тесту на 10000 зображень помилка виникла лише у 82 випадках, це більш ніж 99% правильних відповідей. Тим не менше, більшість помилок, що робить LeNet-5, люди досить легко розпізнають, отже ці випадки також можна подолати.

### Огляд існуючих бібліотек

Розглянемо основні бібліотеки для створення ЗНМ.

***CudaCnn.*** CudaCnn бібліотека для ЗНМ, написана на C++/CUDA з інтерфейсом MATLAB. Підтримує наступні методи навчання: стохастичний градієнтний спуск, стохастичний Левенберга — Марквардта. Опціонально підтримка CUDA. Містить приклад для роботи з датасетом MNIST. Бібліотека не оновлюється з кінця 2012 року.

***cuda-convnet.*** Являє собою швидку (заявлено розробником) C++/CUDA реалізацію ЗНМ. Навчання проводиться з використанням алгоритму зворотного поширення. Бібліотека не оновлюється з липня 2012 року.

***nnForge.*** nnForge – це бібліотека для тренування згорточних та повнозв'язних нейронних мереж. Для обчислення може використовувати CPU і GPU (CUDA). Дана бібліотека написана на мові C++. Це відкрите програмне забезпечення поширюється під ліцензією Apache License v2.0.

***Convolupy.*** Бібліотека написана на мові Python для роботи зі ЗНМ. Підтримка обчислень GPU відсутня. Не оновлюється з 2009 року. Сам розробник порадив використовувати рішення, над яким він зараз працює – Pylearn2 (докладніше у наступному розділі).

***Pylearn2.*** Pylearn2 – бібліотека машинного навчання написана на мові Python [10]. Містить у собі так моделі: ЗНМ, багатошаровий перцептрон, softmax-регресія (як випадок багатошарового перцептрона), обмежена машина Больцмана та інші. Підтримує наступні методи навчання: стохастичний градієнтний спуск, пакетний градієнтний спуск.

Дана бібліотека використовує додаткову інструментарій Theano для оптимізації та обчислення математичних виразів, що включають багатовимірні масиви ефективно. Також ця бібліотека дозволяє використовувати GPU для обчислення складних операцій. Розробник заявляє, що у майбутньому для цього буде використовуватись не лише CUDA, а й OpenCL. Це означає можливу підтримку GPU не тільки компанії Nvidia, а й інших компаній.

Також варто зазначити, що Theano містить у собі реалізацію ЗНМ, та навіть існує бібліотека pynnet, що реалізує інтерфейс для роботи з ними, та вона не підтримується розробником з 2011 року. Сам розробник тепер приймає участь у покращенні та розробці Theano.

Заявленими перевагами Pylearn2 є зрозуміла структура з можливістю створення підкомпонентів та доступність для використання в навчальному процесі (використовується в Монреальському університеті). Бібліотека знаходиться у стадії «бурхливої» розробки.

Мінусом даної бібліотеки є відсутність повної документації, але цей недолік виправляється. Також для основних моделей існують навчальні матеріали у форматі ipynb (IPython Notebook).

Враховуючи основні особливості, було вирішено використовувати саме цю бібліотеку.

### Процес розробки архітектури

***Вимоги до системи.*** Необхідно створити архітектуру, яка дозволить виконувати моделювання ЗНМ для користувача без особливих проблем, реалізувати зрозумілий та простий інтерфейс. Так як майбутньою сферою використання даної системи є використання у навчальному класі для початкового знайомства зі ЗНМ, виникає наступна вимога – можливість використовування кількома користувачами одночасно. Але це не виключає використання системи для більш серйозних задач, таких як знаходження оптимальної архітектури ЗНМ для конкретних наборів даних, вдосконалення тренувальних алгоритмів та інше.

***Клієнт-серверна архітектура.*** Система передбачає клієнт-серверну архітектуру (рис. 2). Моделювання ЗНМ покладено на сервер, де виконуються необхідні обчислення. Користувач працює з системою через web-браузер, що встановлений на персональному комп'ютері. Через web-браузер задаються параметри моделювання, та переглядається отриманий результат. Необхідний доступ до мережі Інтернет.

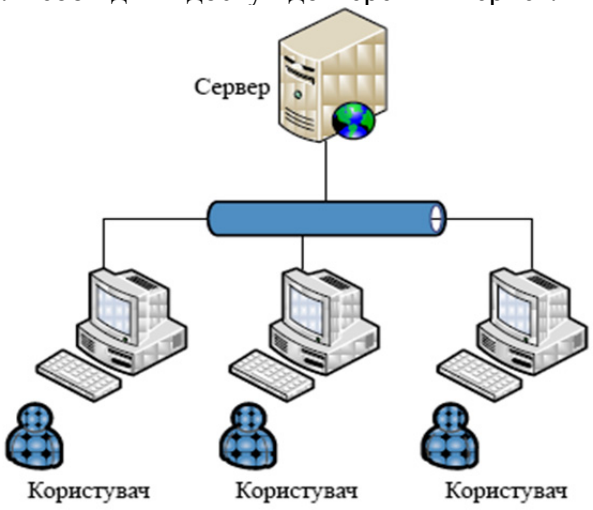



**Рис.2. Приклад архітектури**

Дана архітектура має такі переваги:

- для роботи користувачу необхідний лише web-браузер;
- відсутність жорстких технічних вимог до комп'ютера користувача;
- модернізація та оновлення програмного продукту відбувається без втручання користувача;
- безпека та цілісність даних покладена на сервер.

Та наступні недоліки:

- повна залежність від підключення до Інтернет;
- обмеженність серверних ресурсів.

***Вибір ОС.*** Враховуючи обрану бібліотеку Pylearn2 для ЗНМ, та компоненти, необхідні для її роботи, операційну систему для сервера було обрано CentOS 6.2 64-bit.

CentOS частіше всього використовують, як серверну операційну систему для веб-хостингу. Багато великих хостингових компаній використовують CentOS для забезпечення стабільної роботи для своїх веб-застосунків. Даний дистрибутив Linux базується на основі комерційного дистрибутиву Red Hat Enterprise Linux компанії Red Hat, що гарантує безпеку і стабільність роботи. Життєвий цикл CentOS складає 10 років, підтримка 6-ої версії буде відбуватись до 2020 року.

***Фреймворк для розробки web-системи.*** Враховуючи, що обрана бібліотека для роботи зі ЗНМ написана на Python, фреймворк був обраний теж написаний на цій мові.

Один з варіантів – Flask, що є мікрофреймворком для створення web-сайтів мовою Python. Основна причина чому Flask називається "мікрофреймворком" - це ідея зберегти ядро простим, але розширюваним. У ньому немає абстрактного рівня бази даних, немає валідації форм або всього такого, що вже є в інших бібліотеках до яких ви можете звертатися. Однак Flask підтримує розширення, які можуть додати необхідну функціональність.

Але вибір був зупинений на Django, що має усі необхідні компоненти «з коробки» та такі основні особливості [11]:

- ORM, API доступу до БД з підтримкою транзакцій;
- вбудований інтерфейс адміністратора, з уже наявними перекладами на більшість мов;
- диспетчер URL на основі регулярних виразів;
- розширювана система шаблонів з тегами та наслідуванням;
- система кешування.

***Вибір СКБД***. Обрана бібліотека для web-системи Django дозволяє використовувати наступні СКБД: PostgreSQL (8.2 та вище), MySQL (5.0 та вище), SQLite, Oracle Database Server (9i та вище). Оскільки очікуване навантаження на СКБД має бути невеликим та функції покладені на неї не для моделювання ЗНМ, а тільки для забезпечення роботи з web-системою, було обрано найпоширеніше рішення для web-платформ – MySQL 5.5 [12].

***Web-сервер.*** Для обраної ОС розглядалось два найпопулярніші web-сервери: Apache та nginx. Обидва рішення надають надійність та гнучкість у конфігурації. Але головна особливість, в першу чергу в тому, що Apache працює за принципом: новий запит - новий потік; nginx працює з використанням моделі подій, відповідно новий запит не породжує нових потоків. Завдяки цій особливості, web-сервер потребує менше оперативної пам'яті, а швидкість його роботи стає вищою [13].

***Gateway Interface HTTP-сервера.*** Враховуючи обрані вище рішення, в якості Python web-сервера, до якого відбувається звернення з nginx, було вирішено використовувати Gunicorn [14]. Він має наступні особливості:

- підтримка Django з коробки;
- автоматичне управління робочим процесом;
- проста конфігурація Python;
- можливість використовувати кілька конфігурацій;
- різні hooks для розширюваності;
- сумісність з Python 2.x >= 2.5.

***Фоновий процес моделювання.*** Оскільки процес моделювання досить ресурсномісткий, його необхідно виконувати у фоні та асинхронно, бажано навіть на окремому сервері. Тому було обрано бібліотеку ZeroMQ [15].

ZeroMQ (також називають OMQ) – високопродуктивна бібліотека, що призначена для асинхронного обміну повідомленнями. Ця бібліотека призначена для використання в масштабованих розподілених системах чи в паралельних додатках.

### Схема отриманої архітектури

На рис. 3. представлена схема отриманої архітектури ПЗ.


**Рис.3. Отримана архітектура**

Через web-браузер користувач може створювати задачі, що будуть змодельовані. Після створення задачі, вона додається до черги, звідки її на моделювання отримає ZeroMQ. Після моделювання, результати записуються до бази даних, звідки користувач може їх переглядати.

### Приклад моделювання

Для перевірки працездатності отриманої архітектури було здійснено наступний експеримент [16].

Тренувальний датасет MNIST – набір написаних від руки цифр, складається з 60 тисяч зображень для тренування, та 10 тисяч для тестування, відповідно це складає 85% та 15% від загального обсягу зображень. Конфігурація шарів: вхідний шар [28, 28], ConvRectifiedLinear (кількість вихідних каналів 64, форма ядра згортки [5, 5], форма вікна для розрідженого max pooling [4, 4], відстань між початком кожного вікна згортки [2, 2], норма ядра згортки 1.9365), ConvRectifiedLinear - дублює параметри попереднього шару, Softmax (норма кожного стовпця матриці ваг 1.9365).

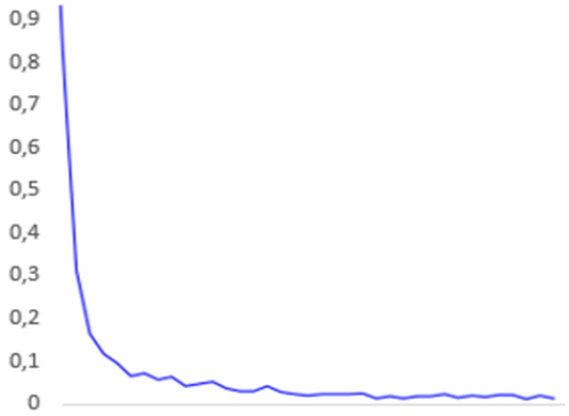



**Рис.4. Графік помилки розпізнання**

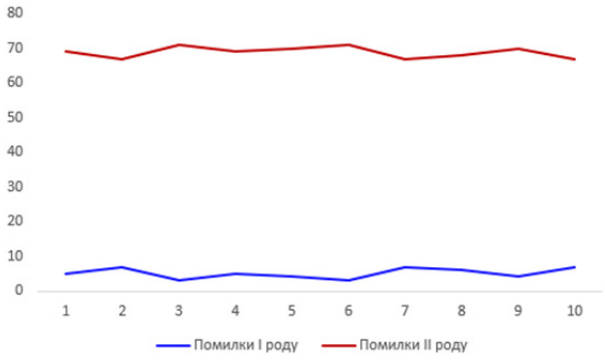



**Рис.5. Графік помилки I та II роду**

Результат моделювання для тестової вибірки - помилка знизилася до 0.74% (рис. 4). Для 10 тестових вибірок середнє значення помилки I роду складає 6,89% від загальної помилки, а помилки II роду 93,1% (рис. 5).

### Висновки

В даній роботі було описано процес розробки архітектури системи для моделювання згорточних нейронних мереж.

Тема даної роботи є актуальною, що підтверджується великою кількістю новин та публікацій за 2012-2013 роки.

На основі виконаного аналізу, було обрано бібліотеку pylearn2, написану на мові Python. Враховуючи поставлену вимоги, обрано допоміжні компоненти для реалізації клієнт-серверної архітектури.

Отримане рішення має переваги у вигляді доступності та простоти у користуванні, але існує недолік, такий як залежність від ресурсів сервера. З іншого боку, сервером може виступати будь-який комп'ютер з ОС Linux, або навіть GPU-кластер.

## Список літератури


1. ImageNet Large Scale Visual Recognition Competition 2012 (ILSVRC2012) [Электронний ресурс]. – Режим доступу: http://www.image-net.org/challenges/LSVRC/2012/
2. Команда Джеффри Хинтона победила в конкурсе компьютерного зрения ImageNet с двукратным преимуществом / Хабрахабр [Электронний ресурс]. – Режим доступу: http://habrahabr.ru/post/183380/
3. LeCun Y. A theoretical framework for backpropagation // Proc. of IEEE. - 1998. - P.21-28.
4. LeCun Y., Bottou L., Bengio Y., Haffne P. Gradient-Based Learning Applied to Document Recognition // Proc. IEEE. - 1998. - P.59-67.
5. Дорогий Я.Ю. Модифицированный алгоритм обучения сверточных нейронных сетей. //Сборник материалов. VII Международная научно-практическая конференция "ПЕРСПЕКТИВЫ РАЗВИТИЯ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ". 18 апреля 2012, г. Новосибирск: Издательство НГТУ. – с 34-38.
6. Convolutional nets for digit recognition [16 min] | Neural Networks for Machine Learning [Электронний ресурс]. – Режим доступу: https://class.coursera.org/neuralnets-2012-001/lecture/73
7. Feature extraction using convolution - Ufldl [Электронний ресурс]. – Режим доступу: http://ufldl.stanford.edu/wiki/index.php/Feature_extraction_using_convolution
8. Yann LeCun's Home Page [Электронний ресурс]. – Режим доступу: http://yann.lecun.com
9. Convolutional neural network class - Mikhail Sirotenko's home page [Электронний ресурс]. – Режим доступу: https://sites.google.com/site/mihailsirotenko/projects/convolutional-neural-network-class
10. Welcome - Pylearn2 dev documentation [Электронний ресурс]. – Режим доступу: http://deeplearning.net/software/pylearn2/
11. Using the Django authentication system | Django documentation | Django [Электронний ресурс]. – Режим доступу: https://docs.djangoproject.com/en/dev/topics/auth/default/#topic-authorization
12. MySQL — Вікіпедія [Электронний ресурс]. – Режим доступу: http://uk.wikipedia.org/wiki/MySQL
13. nginx — Вікіпедія [Электронний ресурс]. – Режим доступу: http://uk.wikipedia.org/wiki/Nginx
14. Gunicorn (HTTP server) - Wikipedia, the free encyclopedia [Электронний ресурс]. – Режим доступу: http://en.wikipedia.org/wiki/Gunicorn_%28HTTP_server%29
15. ØMQ - Wikipedia, the free encyclopedia [Электронний ресурс]. – Режим доступу: http://en.wikipedia.org/wiki/ZeroMQ
16. pylearn2 tutorial: Convolutional network [Электронний ресурс]. – Режим доступу: http://nbviewer.ipython.org/github/lisa-lab/pylearn2/blob/master/pylearn2/scripts/tutorials/convolutional_network/convolutional_network.ipynb

УДК 681.5.13

*КРАВЕЦЬ П.І.,*
*ШИМКОВИЧ В.М.,*
*ОМЕЛЬЧЕНКО П.*

# НЕЙРОМЕРЕЖЕВІ КОМПОНЕНТИ СИСТЕМ КЕРУВАННЯ ДИНАМІЧНИМИ ОБЄКТАМИ З ЇХ АПАРАТНО-ПРОГРАМНОЮ РЕАЛІЗАЦІЄЮ НА FPGA

В даній роботі розроблені та досліджені нейромережеві компоненти систем керування динамічними об'єктами з їх апаратно програмною реалізацією на FPGA, а саме розроблено метод синтезу таких типових моделей як пряма та інверсна модель об'єкта управління та алгоритм апаратно-програмної реалізації фільтра Калмана.

In this paper neural networks are designed and research components of control systems dynamic objects with their hardware program implementation on FPGA, namely those developed typical model as direct and inverse model of the object management and algorithm for hardware-software implementation of Kalman filter.

Нейромережеві системи управління являють собою новий високотехнологічний напрямок в теорії управління та відносяться до класу нелінійних динамічних систем [1,2]. Висока швидкодія за рахунок розпаралелювання вхідної інформації в поєднанні зі здатністю до навчання нейронних мереж робить цю технологію вельми привабливою для створення пристроїв управління в автоматичних системах [2]. Нейронні мережі можуть бути використані для побудови регулюючих та коректуючих пристроїв, еталонної, адаптивної, номінальної та інверсно-динамічної моделей об'єкта, на основі яких виконується спостереження та оцінка параметрів об'єкта керування (ОК), спостереження та оцінка величини діючих в системі збурень, пошук або обчислення оптимальної програми зміни керуючого впливу, ідентифікація ОК, прогнозування стану ОК та інше [2,3]. Здатність до навчання на будь-який заданий принцип функціонування дозволяє створити системи автоматичного керування, оптимальні по швидкодії, по енергоспоживанню і т. д., при цьому, природно, можлива реалізація декількох принципів функціонування та перехід з одного на інший. Навчені нейронні мережі не потребують для обчислень значних часових затрат, а тому системи з нейронними мережами мають значно кращу динаміку. Вони є універсальним засобом для моделювання складних нелінійних ОК та знаходження рішень в некоректних задачах [2-5].

Засоби реалізації нейромережевих систем управління повинні орієнтуватися на широке застосування в промислових умовах, бути універсальними і гнучкими, навчатися і адаптуватися в реальному часі, бути простими і дешевими, тому найбільш перспективними засобами можна вважати FPGA [6,7]. З появою FPGA проектування цифрових мікросхем перестало бути долею виключно великих підприємств з обсягами випуску в десятки і сотні тисяч кристалів. Проектування і випуск невеликої партії унікальних цифрових пристроїв став можливий в умовах проектно-конструкторських підрозділів промислових підприємств, в дослідницьких і навчальних лабораторіях і навіть в умовах домашніх радіоаматорських місць. Промислово випускаються «заготовки» програмованих мікросхем з електричним програмуванням і автоматизованим процесом перекладу схеми користувача в послідовність імпульсів програмування роблять проектування нових цифрових пристроїв порівнянним з розробкою програмного забезпечення [8-10].

Розглянуті далі моделі систем керування включають елементарні схеми, які можуть бути базовими для структурного синтезу функціонально більш складних систем керування. Властивості ШНМ з динамічними алгоритмами навчання дозволяють моделювати складні нелінійні динамічні об'єкти управління – у вигляді прямих та інверсних моделей по вимірах «вхід-вихід» цього об'єкта. Обидві моделі використаються для обчислення векторів стану об'єкта і формування функції управління ним. Також важливим завданням в теорії управління є побудова ефективних алгоритмів оцінювання стану динамічної системи [11]. Один з підходів до оцінювання вектора стану в умовах невизначеності – імовірнісний, згідно з яким обурення і перешкоди є випадковими величинами з відомими функціями розподілу. Завдання фільтрації, оцінки стану динамічної системи за вимірюваннями, при випадкових збуреннях допус-

кає практично вичерпне рішення за допомогою фільтра Калмана [12].

**Метою роботи** є розробка методів та алгоритмів апаратно-програмної реалізації нейромережевих компонентів систем керування динамічними об'єктами на FPGA.

Пряму модель об'єкта управління по вимірах вхідних $u(k)$ та вихідних $y(k)$ даних можна отримати по схемі зображеній на рис. 1. Відома в теорії ідентифікації як схема з налагоджуваною моделлю [13], реалізується в даному випадку рекурентною штучною мережею, що навчається на основі помилки $\hat{e}(t)$:

$$\hat{y}(k+1) = \mathbf{F}\begin{pmatrix} u_k, z^{-1}u_k, ..., z^{-m}u_k; \\ \hat{y}_k, z^{-1}\hat{y}_k, z^{-n}\hat{y}_k; \\ \mathbf{w}_i^{(l)} \end{pmatrix} \quad (1)$$

В якості вектора стану мережі обрано вектор $col(u, z^{-1}y, ..., z^{-n}y) = col(\hat{x}_n(k), \hat{x}_{n-1}(k), ..., \hat{x}_1(k))$, де $z^{-1}$ – оператор затримки. Результатом ідентифікації динамічної моделі поведінки реального ОК в сенсі близькості функцій виходів $\hat{y}(t)$ та $y(t)$ з точністю до помилки навчання $\hat{e}(t) = y(t) - \hat{y}(t)$, або мінімуму деякого функціонала $J\left(e(t)\right)$, можуть бути налаштовані значення вагових коефіцієнтів $\mathbf{w}_i^{(l)}$ у шарах $l = \overline{1, K}$ та оцінки вектора стану об'єкта, який в загальному випадку може бути представлений параметрично недовизначеним нелінійним диференціальним рівнянням:

$$y(k+1) = f\begin{bmatrix} y(k), y(k-1), ..., y(k-n+1); \\ u(k), ..., u(k-m+1) \end{bmatrix}.$$

Структурна схема динамічної нейромережі з одним входом та одним виходом, що відповідає схемі прямої моделі навчання з використанням вимірів «вхід-вихід об'єкта» керування зображена на рис. 1. Схема прямої моделі навчання з використанням вимірів «вхід-вихід об'єкта» керування зображена на рис. 2.

Засновані на ШНМ дискретні ідентифікаційні моделі називаються нейроемуляторами або предикторами. У загальному вигляді вони описуються нелінійним рівнянням (1).

Для одержання прямої моделі в заданому класі функцій $\mathbf{u}(t) \in \mathbf{U}$ не потрібно повної апріорної інформації про структуру зв'язків й їхніх операторів для ОК, окрім інформації про стійкість та обмеженість всіх траєкторій $\mathbf{y}(t)$ для $t \geq 0$. Варто зауважити, що навчена мережа побічно «враховує» вплив на реальний об'єкт управління зовнішніх збурень.

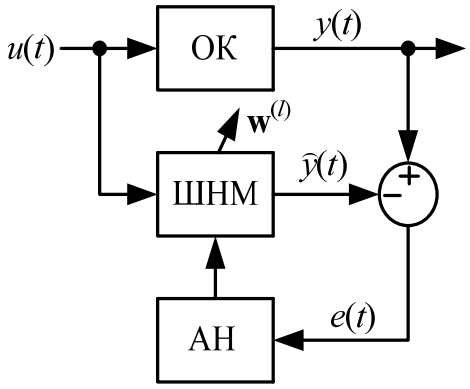



***Рис. 1. Структурна схема адаптивної ідентифікації***

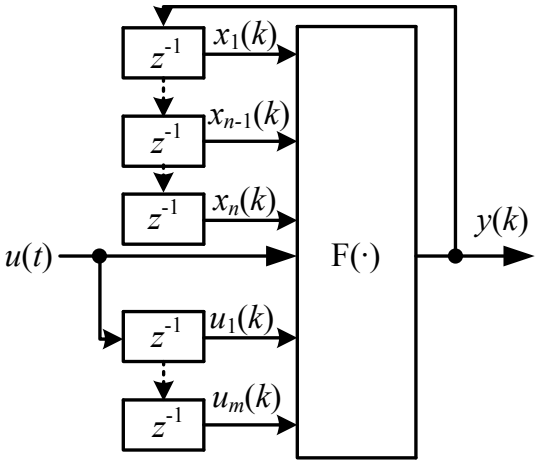



***Рис. 2. Структурна схема прямої моделі ОК***

Для побудови інверсної моделі об'єкта управління можуть бути використані два підходи навчання ШНМ: узагальнене та спеціалізоване інверсне навчання

В схемі узагальненого інверсного навчання, рис. 3, в якості вхідних даних використається тестовий сигнал $\mathbf{u}^*(t)$. Це може бути, наприклад, значення відомої функції оптимального керування об'єктом. Вихід реального ОК $\mathbf{y}(t)$ при підключенні виходу навченої мережі до входу об'єкта відтворює значення функції $\mathbf{r}(t)$, тобто якщо $A$ – оператор ОК, то при $\mathbf{e}_u(t)=0$ нейромережа відображає інверсний оператор $A^{-1}$, що й пояснює термін «інверсне навчання».

Тестовий сигнал $\mathbf{u}^*(t)$ повинен задовольняти вимозі повного перекриття можливого діапазону зміни керуючого впливу в реальній системі керування.

Другий підхід до інверсного моделювання об'єкта управління реалізується схемою спеціа-

лізованого інверсного навчання, зображеною на рис 4. Ця ж схема дозволяє відтворювати задану функцію **r**(*t*).

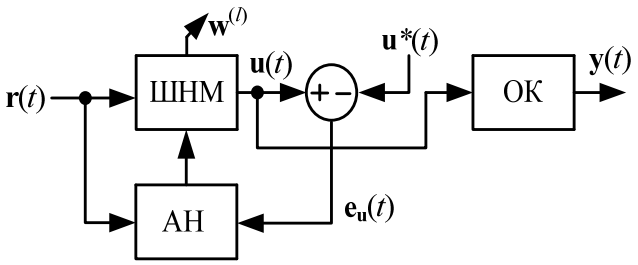



***Рис. 3. Структурна схема загального інверсного навчання ШНМ***

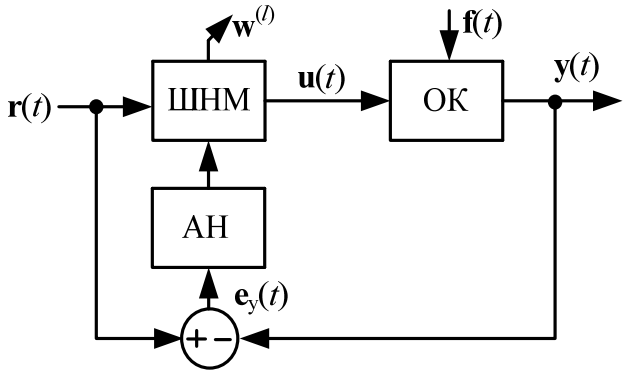



***Рис. 4. Структурна схема спеціалізованого інверсного навчання ШНМ***

У даній схемі сигнал **r**(*t*) виконує роль тестового в завданні інверсного моделювання об'єкта керування за допомогою нейромережі і його форма відповідає класу відтворених у завданні управління функцій.

На відміну від узагальненої схеми в спеціалізованій використовується помилка між заданою функцією **r**(*t*) і виходом **y**(*t*), нейронна мережа налагоджується за динамічним алгоритмом навчання. Якщо нейромережа навчена і помилка $\mathbf{e}_y(t)=0$, то функція **r**(*t*) у точності відтворюється на виході об'єкта. В [17] стверджується, що схема спеціалізованого інверсного навчання дозволяє, відтворити точну інверсну модель об'єкта управління при використанні реального виходу **y**(*t*).

Інверсні моделі самі по собі можуть бути застосовні для управління динамічними об'єктами, у тому числі і для адаптивного управління [2,17-19], але прямі й інверсні моделі можуть служити типовими «будівельними блоками» для синтезу багатомірних систем управління об'єктами з більш складною структурою.

Апаратно-програмна реалізація моделей та елементів систем управління на основі ШНМ на прикладі реалізації узагальненої нейромережевої моделі об'єкта управління рис.2, виконується за наступним алгоритмом:

**Крок 1.** Проведення експерименту.

Основним завданням проведення експерименту є побудова інформативної множини даних *Z*, придатної для побудови працездатної моделі:

$$Z^N = \{[u(t),\, y(t)], t = \overline{1, N}\}, \qquad (2)$$

де: $u(t)$ – тестовий сигнал;

$y(t)$ – реакція об'єкту на тестовий сигнал;

$t$ – час;

$N$ – розмірність множини *Z*.

Побудова інформативної множини даних *Z* може бути виконана шляхом імітаційного моделювання (у випадку якщо модель об'єкту відома) або шляхом проведення серії експериментів з реальним об'єктом (у випадку, якщо модель об'єкту не відома.).

Для нелінійних об'єктів надзвичайно важливо, що б в множині експериментальних даних $Z^N$ були представлені всі можливі комбінації амплітуд і частот з робочого діапазону системи. Один з можливих варіантів тестового сигналу, що задовольняє вказаним вимогам, описується виразом:

$$u(t) = u(t-N) + e(\mathrm{int}\left[\frac{t-1}{N}\right]+1),\ t = 1,2,\ldots \qquad (3)$$

де: $e(t)$ – білий шум з дисперсією $\sigma_e^2$.

Іншим варіантом тестового сигналу є синусоїда з наростаючою частотою $\omega_c$ і амплітудою $A_c$, що змінюється:

$$u(t) = u_0 + A_c \sin(\omega_c t T),$$
$$\omega_c = \omega_н + \frac{(\omega_к - \omega_н)}{N}, \qquad (4)$$
$$A_c = A_н + \frac{(A_к - A_н)}{N},$$

де $\omega_н$, $\omega_к$ – відповідно початкове і кінцеве значення частоти синусоїдального сигналу;

$A_н$, $A_{\hat{e}}$ – відповідно початкове і кінцеве значення амплітуди синусоїдального сигналу.

Для ефективного використання отриманої тестової множини експериментальних даних необхідно виконати його попередню обробку (фільтрацію, видалення надмірних даних і викидів, масштабування), метою якого є отримання найбільш значущої інформації і приведення її до певного вигляду, необхідного для навчання нейромережевої моделі.

**Крок 2**. Вибір області можливих модельних структур.

Під модельною структурою розумітимемо структуру нейронної мережі, яка може бути умовно розділена на «внутрішню структуру» і «зовнішню структуру».

Внутрішня структура нейронної мережі визначається: топологією мережі, кількістю прихованих шарів, числом нейронів і видом активаційних функцій в кожному шарі.

Зовнішня структура нейронної мережі визначається вектором входу $\varphi(t)$ (регресором). Вектор входу $\varphi(t)$ нейронної мережі можна представити як:

$$\phi(t) = [\phi_1 \dots \phi_k]^T = \\ = [\phi_1(t-1) \dots \phi_1(t-d_1) \dots \phi_k(t-1) \dots \phi_k(t-d_k)]^T, \quad (5)$$

де: $\varphi_k$ – $k$-а компонента регресора;

$d_k$ – «глибина» регресора.

Під вибором регресора мається на увазі визначення компонент регресора $\varphi_k$ і глибини регресії $d_k$, тобто кількості $d$ значень $k$-ої компоненти регресора в попередні значення часу. У якості компонент регресора зазвичай використовуються ті параметри системи (процесу), які можуть бути безпосередньо виміряні (або оцінені) в режимі функціонування. Наприклад, для одновимірних об'єктів в якості компонентів регресора використовується значення входу $u(t)$ і виходу $y(t)$ об'єкту. Вибір глибини регресії визначається динамікою об'єкта .

Таким чином, завдання вибору структури нейронної мережі зводиться до визначення «глибини» регресора $\varphi(t)$ («зовнішня» структура) і кількості нейронів прихованого шару $n$ («внутрішня» структура) багатошарової нейронні мережі, що містить один прихований шар нейронів з сигмоїдальними функціями активації і вихідний шар з лінійними функціями активації.

Визначення області «можливих» модельних структур $V(n, \varphi_1, \dots, \varphi_k)$ в області модельних структур $M(n, \varphi_1, \dots, \varphi_k)$, виконується на основі аналізу апріорної інформації про об'єкт і його динаміку.

**Крок 3.** Оцінка моделей з області «можливих» модельних структур.

У роботі [15] розглядається комплексний формалізований підхід до реалізації багатоетапної процедури побудови нейромережевих моделей складних динамічних об'єктів, розроблено програмний пакет «MIMO-Plant» в середовищі MatLab. Для кожної моделі з області $V(n, \varphi_1, \dots, \varphi_k)$ обчислюється значення критерію адекватності, у якості котрого можна використовувати значення середньоквадратичної помилки. Представлено детальний опис програмного пакету та приклади дослідження.

**Крок 4.** Реалізація нейромережевої моделі на ПЛІС.

При апаратно-програмній реалізації нейромережевих моделей на FPGA використовується метод апаратно-програмної реалізації ШНМ описаний в [14].

**Крок 5.** Оцінка «вартості» реалізації моделі на FPGA.

Для вибраної моделі обчислюється значення критерію «вартості» апаратної реалізації нейромережевої моделі на ПЛІС. Під «вартістю» реалізації $C$ в даному випадку розуміється частина ресурсів (Slices), кількість D–тригерів (Flip Flops), об'єм вбудованої блокової пам'яті (BRAM), кількість чотиревходових таблиць перетворення (Look-up Table)), необхідних для реалізації отриманої моделі на вибраному типі кристала ПЛІС

$$C = \max\left( \frac{\sum_{i=1}^{I} S_i}{S_X}, \frac{\sum_{i=1}^{I} F_i}{F_X}, \frac{\sum_{i=1}^{I} R_i}{R_X}, \frac{\sum_{i=1}^{I} L_i}{L_X} \right) \quad (6)$$

де: $S_i, F_i, R_i, L_i$ – кількість Slices, Flip Flops, BRAM, Look-up Table, відповідно, що необхідна для реалізації нейромережевої моделі;

$S_X$, $F_X$, $R_X$, $L_X$ – кількість Slices, Flip Flops, BRAM, Look-up Table, відповідно, яка міститься у вибраному кристалі ПЛІС;

$I$ – сумарна кількість елементів (ваги, зсуви, функції активації, затримки) тих, що реалізовують нейронну мережу.

Умовою реалізуємості нейромережевої моделі на вибраному кристалі ПЛІС є виконання нерівності

$$C < 0.7 . \quad (7)$$

**Крок 6.** Прийняття рішення про можливість практичної реалізації.

Якщо отримана модель не задовольняє якомусь критерію, то необхідно виконати попередні кроки алгоритму побудови моделі, аж до проведення нової серії експериментів.

Відповідно до даного алгоритму, якщо умова (7) виконується, то дану модель можна реалізувати на вибраному кристалі ПЛІС, якщо ж умо-

ва (7) не виконується, то модель на такому кристалі ПЛІС реалізувати неможливо. В цьому випадку забезпечити виконання умови (7) можна наступним чином: 1) виконати процедуру оптимізації структури моделі; 2) вибрати кристал ПЛІС більшої ємності і/або іншого сімейства; 3) збільшити число кристалів ПЛІС.

**Крок 7**. Оптимізація структури моделі

Оптимізація структури нейромережевої моделі проводиться шляхом видалення ряду малозначних вагових коефіцієнтів. Для реалізації цього можуть бути використані такі алгоритми:

- алгоритм послідовного зменшення структури;
- алгоритм послідовного збільшення структури;
- генетичний алгоритм;
- Optimal Brain Damage (OBD);
- Optimal Brain Surgeon (OBS) [5].

Останній алгоритм є найбільш відомим і широко поширеним. Після виконання мінімізації структури необхідно заново оцінити адекватність моделі і виконати кроки 5, 6.

Розглянемо приклад реалізації прямої та інверсної моделі такого динамічного об'єкту як індукційна піч. Динаміка індукційної печі детально описана в [20].

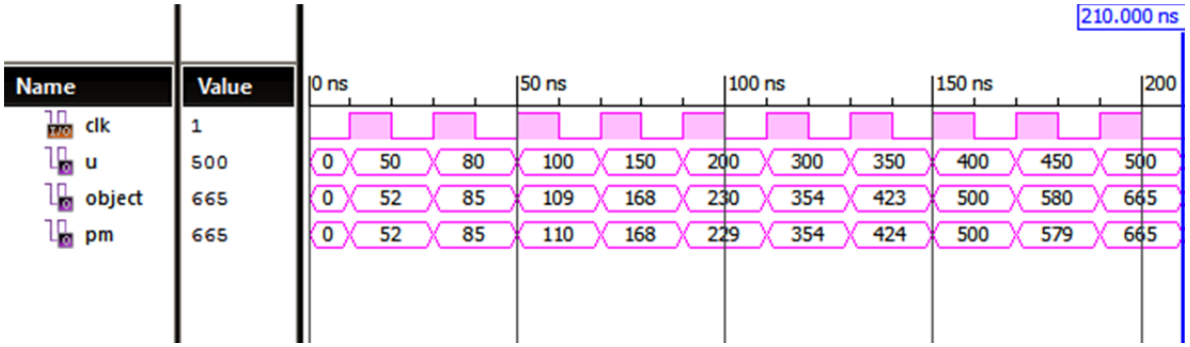



***Рис. 5. Моделювання роботи прямої моделі***

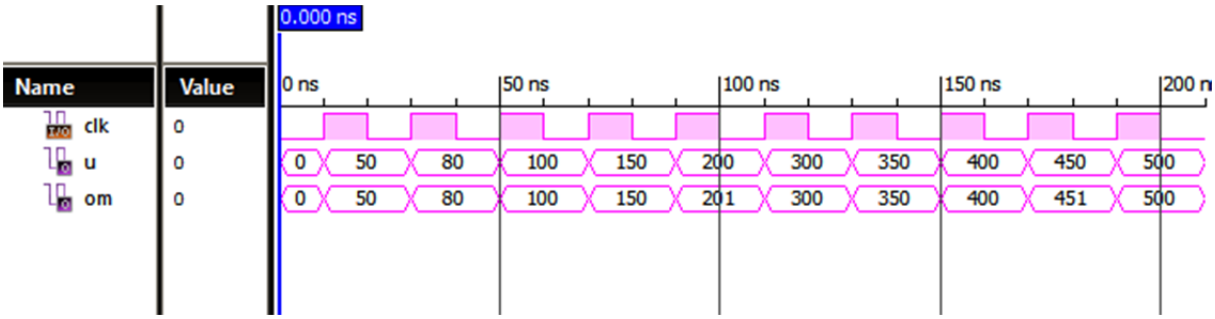



***Рис. 6. Моделювання роботи інверсної моделі***

На рисунках 5 та 6 представлено моделювання роботи структурних схем зображених на рисунках 1 та 4 відповідно. Обидві схеми побудовані в середовищі ISE Design Suite 13.2 та промодельовані в середовищі ISim. Як видно з рисунків 5 та 6 отримані нейромережеві моделі є адекватними.

Фільтр Калмана призначено для рекурсивного дооцінювання вектора стану апріорно відомої динамічної системи, тобто для розрахунку поточного стану системи необхідно знати поточний вимір, а також попередній стан самого фільтра. Фільтр Калмана реалізований в тимчасовому, а не в частотному поданні. Наочний приклад можливостей фільтра – отримання точних, безперервно обновлюваних оцінок положення і швидкості деякого об'єкта за результатами часового ряду неточних вимірювань його місця розташування. Фільтр Калмана оперує поняттям вектора стану системи і його статистичними описом. Вони базуються на дискретизованих за часом лінійних динамічних системах. Стан системи описується вектором кінцевої розмірності – вектором стану. У кожен такт часу лінійний оператор діє на вектор стану і переводить його в інший вектор стану, додається деякий вектор нормального шуму і в загальному випадку вектор управління, що моделює вплив системи управління. Опис фільтра Калмана представлено в [20]. Апаратно-програмна реалізація фільтра Калмана на FPGA дозволить синтезувати велику кількість паралельно пра-

цюючих фільтрів на одному кристалі, що в свою чергу дає можливість синтезу систем управління багатовимірними об'єктами(MIMO).

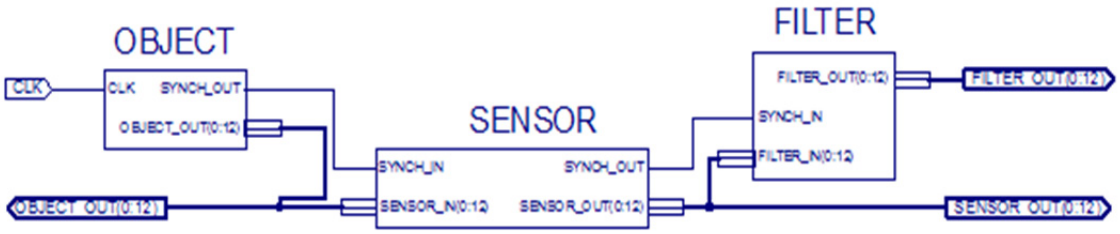



***Рис. 7. Схема електрична принципова моделі для дослідження роботи фільтра Калмана***

Апаратно-програмна реалізація фільтра Калмана на FPGA, виконується за наступним алгоритмом:

**Крок 1:** Виконати ініціалізацію змінних.

Задати дисперсію похибки датчика *sigmaeta*, середнє значення квадрату похибки на *k*-й ітерації *eopt_prev*, дисперсію похибки моделі *sigmaksi*. Задати *a,* коефіцієнт деякої відомої керуючої функції $u_k=a\cdot k$, де *k* – номер ітерації. При обчисленні використовувались десяткові числа, проте, так як в мові VHDL немає вбудованих засобів представлення не цілих десяткових чисел, то в програмному коді фільтра застосовуються коефіцієнти-змінні *h*=10000 і *l*=10 для перенесення коми, множення не цілих змінних в формулах. Таким чином отримуються великі цілі числа типу integer, з якими можна за допомогою вбудованих арифметичних операцій виконати необхідні обчислення. Отже в обчисленнях використовуються змінні з точністю до 0.0001. Результат представлений тільки цілою частиною.

**Крок 2:** Якщо на синхровхід надходить імпульс(перехід з 0 в 1) – перейти до **Крок 3**, інакше – чекати, поки не надійде імпульс (повторити **Крок 2**).

**Крок 3:** Зчитати з входу сигнал FILTER_IN покази датчика *z*<=FILTER_IN. *z* – покази датчика.

**Крок 4:** Обчислити квадрат середнього значення квадрату похибки

*eopt*:=*divide*(
(*sigmaeta*$^2$)·(*eopt_prev*+*sigmaksi*$^2$),
(*sigmaeta*$^2$+*eopt prev*+*sigmaksi*$^2$) )

Вбудованої операції ділення двох десяткових чисел в VHDL не передбачено, тому операцію ділення (ф-я *divide*) в вищенаведеній формулі було реалізовано відновлювальним алгоритмом ділення [21]. Ця функція приймає два числа типу integer (ділене та дільник) і приводить їх до типу unsigned. Тому отримавши два двійкових числа, дуже легко знайти частку маніпулюючи бітовими зсувами, ця операція швидка й використовує манімальну кількість ресурсів.

**Крок 5:** Обчислити коефіцієнти підсилення Калмана

*k*:=*divide*((*eopt*·*h*),(*sigmaeta*$^2$));

В цій формулі також використовується перенесення коми і функція діленні *divide*.

**Крок 6:** Обчислити оптимальне відфільтроване значення

*xopt*:=(*divide*(*xopt_prev*·*l*, *h*)+*a*·*t*·*l*)·(*1*·*h*-*k*)/*l*+*k*·*z;*

де *z*, як було вказано вище, покази датчика.

**Крок 7:** Ділимо на коефіцієнт *h*

*xoptout*:=*xopt/h;*

і результат фільтра (*xoptout*). Дрібна частина з точністю до 0.0001 враховується в обчисленнях, але в результаті нехтується.

**Крок 8:** Видати результат на вихід фільтра *FILTER_OUT*:

*FILTER_OUT <= xoptout*

**Крок 9:** Інкремент кроку ітерації *k*=*k*+1.

Розглянемо приклад реалізації фільтра Калмана на FPGA. Для моделювання його роботи побудуємо схему електричну принципову в програмному забезпеченні Xilinx ISE Design Suite 13.2, на якій в свою чергу змоделюємо роботу ОК, датчика та фільтра рис.6. На рис. 6 блок Object – модель об'єкта. Входи: CLK – вхід синхронізації. Виходи: SYNCH_OUT – вихід синхронізації; OBJECT_OUT – вихідна величина об'єкта. Блок SENSOR – модель датчика. Входи: SYNCH_IN – вхід синхронізації (з об'єктом); SENSOR_IN – вхід, на який надходить вимірювана величина. Виходи: SYNCH_OUT – вихід синхронізації(з фільтром); SENSOR_OUT – вихід, «виміряна» величина. Блок FILTER – фільтр Калмана. Отримує результати вимірювання датчика та за методом Калмана генерує значення, які ближчі до реальної вимірюваної величини**.** Входи: SYNCH_IN – синхронізація; FILTER_IN – вхід фільтра (для датчика з похибкою). Виходи: FILTER_OUT–

оптимальне згенероване значення вимірюваної величини.

Нехай динаміка об'єкта управління описується фізичним законом руху. Характеризується деякою вихідною величиною, яка змінюється за заданим законом $x_{k+1} = x_k + v_k dt$, де $k \in [0,100]$ – номер ітерації; $x_{k+1} \in [0,4950]$ – вихідна(вимірювана) величина на ітерації $k$+1; $a$=1 – коеф. нелінійного закону; $v_k = a \cdot k$ – відома керуюча функція. В побудованій моделі похибка датчика не перевищує $\eta_k = 50$, генерується псевдовипадкова послідовність значень.

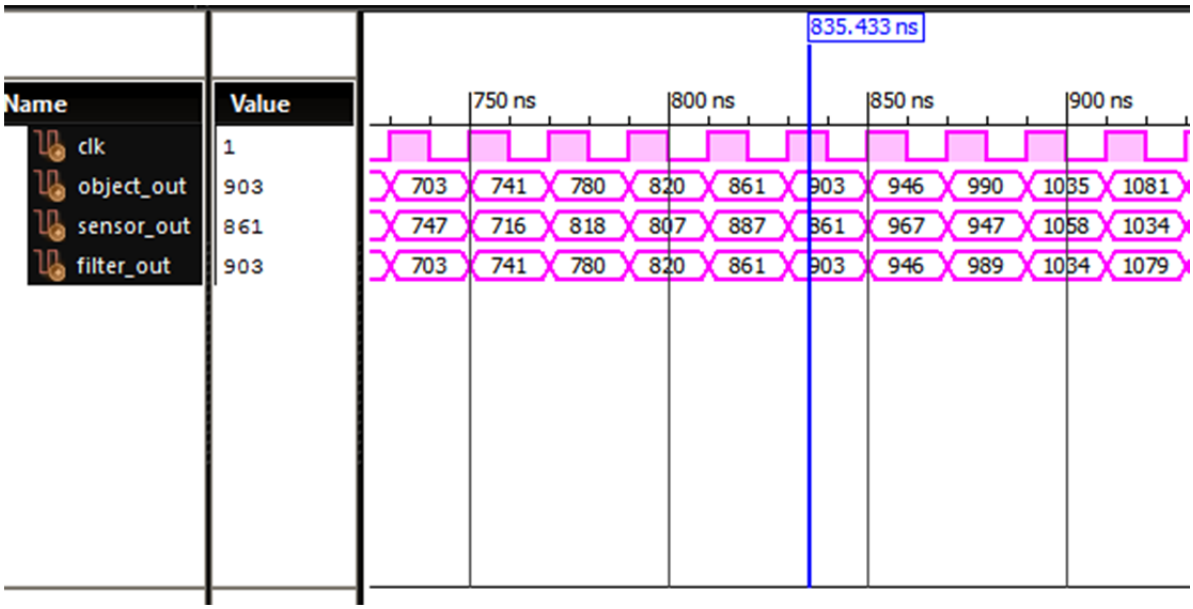



***Рис. 8. Моделювання роботи фільтра Калмана в програмному пакеті ISim***

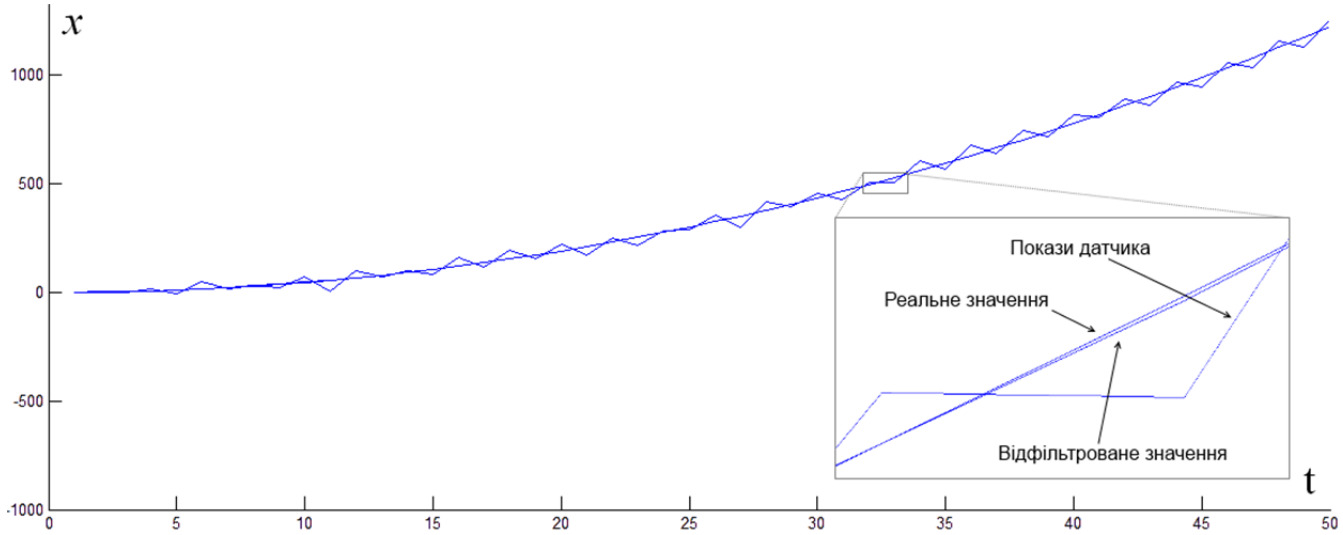



***Рис. 9. Графік фільтрації сигналу по Калману***

На часовій діаграмі, рис. 7, видно, що сигнал на виході датчика (sensor_out) спотворений похибкою, видно сигнал без похибки та роботу фільтра Калмана. При точності 0.0001 використовуваній в обчисленнях та досить великому проміжку значень вимірюваної величини, точність на виході фільтра знаходиться в межах 0.0001. Наглядний графік фільтрації сигналу по Калману(рис. 8) датчика (sensor_out), відфільтрованого значення(filter_out) та реальної величини(object_out) побудований в Matlab.

## Висновок

В даній роботі розроблені та описані методи апаратно-програмної реалізації нейромережевих компонентів систем керування динамічними об'єктами. Розроблено метод апаратно-програмної реалізації прямої та інверсної моде-

лі об'єкта управління. На прикладі показана реалізація моделей такого об'єкта як індукційна піч. Отримані нейромережеві моделі є адекватними. Розроблено алгоритм реалізації фільтра Калмана, розглянуто приклад його реалізації та промодельовано його роботу. Розроблені в даній роботі нейромережеві компоненти дозволять будувати системи управління, які функціонують та адаптуються в режимі реального часу та формують функції управління будь-якої складності. Розробку та моделювання виконано на програмному забезпеченні Xilinx ISE Design Suite 13.2 та чіпі сімейства Spartan 3 – XC3S200.

## Перелік посилань


1. Саймон Хайкин. Нейронные сети. Полный курс. – М.: Вильямс, 2006. – 1104с.
2. Терехов В.А., Ефимов Д.В., Тюкин И.Ю. Нейросетевые системы управления: Учеб. пособие для вузов. – М: Высшая школа. 2002. -183с.
3. Сигеру Омату. Нейроуправление и его приложения. – М.: ИПРЖР, 2000. – 272с.
4. Рутковская Д., Пилиньский М., Рутковский Л. Нейронные сети, генетические алгоритмы и нечеткие системы: Пер. с польск. И.Д. Рудинского. – М.: Горячая линия – Телеком, 2007. – 452 с.
5. Егупов Н.Д. Методы робастного, нейро-нечеткого и адаптивного управления. – М.: Изд-во МГТУ им. Н.Э. Баумана, 2001. – 744 с.
6. Гильгурт С.Я. Анализ применения реконфигурируемых вычислителей на базе ПЛИС для реализации нейронных сетей // Моделювання та інформаційні технології. Зб. наук. пр. ІПМЕ НАН України. – Вип. 37. – Київ: 2006. – С. 168-174 с.
7. Логовский А. Технология ПЛИС и ее применение для создания нейрочипов. // Открытые системы. – 2000. – № 4. – С. 100-102 с.
8. Соловьев В. Проектирование цифровых систем на основе ПЛИС. – М.: Радио и связь, 2003. – 376с.
9. Тарасов И.Е. Разработка цифровых устройств на основе ПЛИС Xilinx® с применением языка VHDL. – М.: Горячая линия – Телеком, 2005. – 252 с.
10. Сергиенко А.М. VHDL для проектирования вычислительных устройств – К ЧП «Корнейчук», ООО «ТИД «ДС», 2003. – 208 с.
11. Кунцевич В.М. Управление в условиях неопределенности: гарантированные результаты в задачах управления и идентификации. Монография. — К.: Наук. думка, 2006. — 264 с.
12. Kalman R.E. Identification of Noisy Systems. Russian Mathematical Surveys, 1985, no. 40 (4), pp. 25–42.
13. Льюнг Л. Идентификация систем. Теория для пользователя. – М.: Наука, 1991.
14. Кравець П.І., Шимкович В.М., Зубенко Г.А. Технологія апаратно-програмної реалізації штучного нейрона та штучних нейронних мереж засобами FPGA / Вісник НТУУ «КПІ». Інформатика, управління та обчислювальна техніка: Зб. наук. пр. – К.: Век+, – 2012. – №55. 174-180с.
15. Кравець П.І., Лукіна Т.Й., Шимкович В.М., Ткач І.І. Розробка та дослідження технології оцінювання показників нейромережевих моделей MIMO-об'єктів управління / Вісник НТУУ «КПІ». Інформатика, управління та обчислювальна техніка: Зб. наук. пр. – К.: Век+, – 2012. – №57. 144-150с.
16. Кравец П.И., Шимкович В.Н. Метод оптимизации весовых коэффициентов нейронных сетей с помощью генетического алгоритма при реализации на программируемых логических интегральных схемах / Международный научно-технический журнал «Электронное моделирование». – 2013. – 35, №3. – С. 65-75.
17. Neural networks for control systems: A survey / K. J. Hunt, D. Sbarbaro, R. Zbikowski, P. J. Gawthrop // Automatica. 1992. Vol. 28. № 6. P. 1083 – 1112.
18. Горбат А. Н. Обучение нейронных сетей. – М.: СП «ParaGraph», 1990. – 160с.
19. Intelligent control systems: Theory and applications. Madan M. Gupta, Naresh K. Sinha. 1995. P. 820.
20. Пузырев В.А. Управление технологическими процессами производства микроэлектронных приборов. – М.: Радио и связь, 1984 – 160 с.
21. http://habrahabr.ru/post/166693
22. http://en.wikipedia.org/wiki/Division_algorithm#Restoring_division

УДК 004.091.043

*МАРКОВСКИЙ А.П.,*
*ШЕВЧЕНКО О. Н.,*
*ФАНЬ ЧУНЬЛЭЙ*

# ИНТЕРАКТИВНО-ШАБЛОННЫЙ МЕТОД КОМПЬЮТЕРНОГО ПЕРЕВОДА НАУЧНО-ТЕХНИЧЕСКИХ ПУБЛИКАЦИЙ

В статье предложен подход к организации компьютерного перевода научно-технических публикаций. В основу подхода положено интерактивное использование лингвистических шаблонов, позволяющее повысить качество перевода за счет рационального разделения функций специалиста в предметной области, переводчика-лингвиста и компьютерных программ. Разработаны процедуры шаблонирования текста публикации и компьютерного перевода с использованием шаблонов. Приведен пример перевода и результаты экспериментальных исследований. Выполнена оценка эффективности предложенного подхода по сравнению с существующими системами компьютерного перевода.

The paper proposes a new approach to computer translation of technical scientific publications. The described method is based on the interactive use of translation templates which allow to improve the translation quality due to the rational distribution of functions between a domain expert, a human translator (a linguist) and computer software.The procedures for creating text templates and for template-based computer translation of publications were developed . An example of the translation and the results of experimental studies were given. The evaluation of the effectiveness of the proposed approach compared to the existing systems of computer translation was made.

## Введение

Начало третьего тысячелетия знаменует динамичный процесс углубления и расширения интеграционных процессов во всех сферах человеческой деятельности. В значительной степени процесс информационной интеграции обусловлен быстрым развитием средств телекоммуникации и компьютерных сетей и, в первую очередь, Интернета, которые технически решают задачу доступа к информации.

Однако, существенным препятствием на пути обмена информацией, главным образом, связанной с научными и техническими достижениями, остается языковой барьер. Развитие технологий электронных публикаций педалировало в начале 21-го века заметный рост числа публикаций научного характера в странах азиатского региона, в первую очередь таких как Китай и Индия [1]. Это обстоятельство обостряет проблему межъязыкового взаимопонимания в процессе обмена научно-технической информацией.

Значительным потенциалом в решении этой проблемы является использование возможностей современных компьютерных технологий.

Таким образом, проблема создания эффективных средств компьютерного перевода научно-технических публикаций является важной и актуальной для расширения и углубления информационной интеграции во многих областях человеческой деятельности.

## Анализ современного состояния проблемы компьютерного перевода

Проблема компьютеризованного перевода является одной из наиболее традиционных в компьютерных технологиях. Первые системы машинного перевода появились еще в 60-х годах прошлого столетия [2]. С тех пор проведено большое число исследований, опубликовано тысячи статей, создано ряд действующих систем, на порядок выросли возможности компьютерной техники.

Все системы компьютеризованного перевода, с позиций участия в их функционировании человека, принято [1] разделять на два класса: машинного (или автоматического) перевода и автоматизированного перевода (CAT – computer-aided translation). Системы первого класса осуществляют перевод текстов с одного естественного языка на другой без участия человека. Примерами подобных систем являются Systran [3] и программа-переводчик Google [4]. Первая из них строится на на основе грамматических правил (Rule-Based Machine Translation, RBMT), обеспечивающей перевод на базе встроенных словарей и грамматических правил двух языков. Основным недостатком этой системы является недостаточное для практиче-

ского использования семантическое качество перевода. Основной причиной этого является отсутствие пригодных для практического использования формальных моделей естественных языков [5]. В некоторых системах полностью машинного перевода, в частности, в упоминавшейся выше программе перевода Google, отсутствие такой модели заменяется статистическим анализом (Statistical Machine Translation, SMT) огромного количества текстов и их переводов, выполненных человеком. Системы этого типа анализируют статистику межъязыковых соответствий и используют эту информацию при выборе вариантов перевода. Их очевидным недостатком является потребность в значительном объеме ресурсов (памяти и процессорного времени). Опыт практического использования показал [4], что знамена формальной модели языка статистикой не позволяет заметно улучшить семантическое качество перевода. Такой же результат имеют и попытки заменить отсутствие формальной модели естественного языка использованием самообучающихся систем и систем на основе нейронных сетей [6]. Фактически такие системы не вышли к настоящему времени за границы экспериментальных разработок.

Таким образом, полностью компьютеризированные системы перевода, работающие без участия человека, не способны к настоящему времени обеспечить в полной мере семантическое соответствие между входным и выходным текстами.

Поэтому на практике такие системы требуют редакторской правки опытного переводчика и специалистов предметной области.

Более широко используются системы автоматизированного перевода, в которых основная роль принадлежит переводчику, труд которого автоматизируется компьютерными средствами. В простейшем случае такие средства представляют собой компьютерные словари. Более сложные системы предоставляют переводчику ряд вариантов перевода отдельных предложения и средства интерактивного редактирования. Существенным недостатком таких систем является недостаточная скорость перевода. Фактически системы автоматизированного перевода применительно к научным и техническим публикациям ориентированы на специалистов в предметной области и предполагают знание ими базовых основ входного языка. Практика использования подобных систем для перевода публикаций научного и технического характера показала значительную зависимость семантического качества перевода от квалификации специалиста и знания им входного языка. Это существенно ограничивает сферу их эффективного использования. В частности, автоматизированные системы практически не пригодны для перевода с китайского, японского и корейского языков для широкого круга европейских специалистов.

Таким образом, несмотря на достаточной широкий фронт проводимых исследований, решенной проблему эффективного перевода информации научного и технического характера считать нельзя [1].

Целью настоящей работы является повышение эффективности компьютерных систем многоязыкового перевода научно-технической информации.

## Шаблонный метод представления и перевода языковых конструкций

Как отмечалось выше, фундаментальной причиной относительной низкой эффективности существующих систем компьютерного перевода научно-технических публикаций является отсутствие в достаточной мере адекватной модели естественных языков, которая позволяла бы семантически точно транслировать предложения входного языка в выходные. Одной из возможностей обойти это препятствие является введение ограничений на количество конструкции предложений входного языка.

Основная идея предлагаемого подхода состоит в том, что публикации научного и технического характера, без заметного ущерба для понимания, могут быть оформлены с использованием ограниченного набора конструкций предложений. Анализ текстов таких публикаций [6,7] показал, что число наиболее употребляемых в них синтаксических конструкций предложений не превышает 200-600. Такое ограниченное число синтаксических конструкций предложений входного языка вполне может переведено специалистами-лингвистами в адекватные конструкции одного или нескольких выходных языков. Таким образом, ограничение возможных вариантов синтаксических конструкций текстов научно-технических публикаций позволяет в определенной мере обойти проблему отсутствия моделей естественных языков.

Реализация такой идеи потребует от автора оформления текста научного или технического

характера изначально в виде предложений, синтаксические конструкции которых выбираются из определенного множества. В известном смысле можно говорить о том, на современном уровне межнациональной информационной интеграции написание и оформление текста научно-технических публикаций должно быть изначально ориентировано для перевода его на иные языки.

Фактически, при реализации предлагаемой идеи две наиболее сложные операции процесса перевода - распознавание входных синтаксических конструкций и синтаксически корректная трансформация конструкций входного языка в конструкции выходного языка выполняются человеком. Причем первая из операций выполняется автором текста непосредственно в процессе его оформления, а вторая - единоразово специалистом-лингвистом.

Для того, чтобы облегчить автору выбор синтаксической конструкции, позволяющей наиболее точно выразить семантический смысл предложения предлагается использовать специальные шаблоны предложений.

Шаблон представляет собой семантико-синтаксическую конструкцию, доминантой которой является семантическая составляющая с синтаксически изменяемыми компонентами. При переводе шаблона семантическая составляющая, являющаяся основой предложения, переводится специалистом-лингвистом, а синтаксические составляющие - заменой слов входного языка на слова выходного языка с использованием компьютерных словарей.

Автор осуществляет выбор одного из прототипов шаблонов. После этого, автором выполняется замена слов шаблона на слова, соответствующие семантическому смыслу формируемого предложения. При этом номер используемого шаблона и список заменяемых слов записываются в спецификацию предложения. Фактически автором выполняется только подходящая по смыслу замена слов, а спецификация оформляется автоматически.

Практически шаблон предъявляется автору в виде предложения естественного языка иной предметной области, в котором, сохраняя семантическую основу, необходимо заменить слова. Например, если семантический смысл предложения состоит в том, что для повышения эффективности исправления “пачки” ошибок в каналах со спектральной модуляцией автором предлагается использовать взвешенные контрольные суммы, то запрос на поиск наиболее подходящего шаблона по базовой семантической составляющей может иметь вид: “**Для повышения эффективности ... предлагается использовать ...** ”.

В ответ на данный запрос система предъявляет ряд пронумерованных шаблонов, один из которых, наиболее близкий по версии автора к желаемому имеет вид: “**Для повышения эффективности** идентификации (1) абонентов (2) в (3) интегрированных (4) базах (5) данных (6) **предлагается использовать** необратимые (7) булевы (8) преобразования (9)”. В приведенном шаблоне выделена семантическая основа и пронумерованы компоненты, которые могут быть заменены.

Автор выполняет замену слов в шаблоне в соответствии со смыслом формируемого предложения, которое после этого приобретает вид: “**Для повышения эффективности** исправления (1) пачки (2) ошибок (3) в (4) каналах (5) со (6) спектральной (7) модуляцией (8) **предлагается использовать** взвешенные (9) контрольные (10) суммы (11)”.

Если предположить, что в рамках рассмотренного выше примера выбранный автором шаблон имеет номер 172, то его спецификация может быть представлена в виде: {**172**, <1-1>, <2-2,3>, <3-4>, <4,5,6-5,6,7,8>, <7,8,9-9,10,11>}. Это означает, что используемая шаблонная конструкция имеет номер 172, и в процессе ее наполнения новым семантическим смыслом первое слово шаблона заменено первым словом предложения, второе слово шаблона заменено вторым и третьим словами предложения и так далее.

При написании каждого предложения публикации автором формируется по семантической основе запрос на выбор шаблона, выбирается наиболее подходящий из предъявляемых по запросу и корректируется путем заменой слов. В результате формируется предложение текста публикации и спецификация предложения, которые сохраняются в одном файле.

Наиболее сложным этапом при компьютерном переводе является синтаксический анализ входного предложения. Это этап предлагается выполнять в интерактивном режиме путем шаблонирования - замены произвольных предложений входного текста на идентичные в семантическом плане предложения-шаблоны. При этом одному предложению исходного текста могут соответствовать несколько предложений шаблонированного текста. Шаблонирование входного текста публикации может вы-

полняться как самим автором, так и квалифицированным специалистом в данной предметной области. Процесс шаблонирования предполагает создание двух синхронизируемых документов: самого текста, предложения которого принадлежат набору шаблонов и спецификации текста, которая каждое из предложений входного текста описывает номером используемого шаблона и списком слов, которые постанавливаются в шаблон.

Схематично процесс шаблонирования показан на рис.1. Вначале при написании предложения интерактивно выбирается базовая языковая конструкция, подходящая для выражения семантического смысла формируемого предложения. Технологически для выбора базовой конструкции автором вводится последовательность ее базовых слов.

Система выполняет по введенной последовательности поиск ряда отвечающих запросу шаблонов, которые предъявляются автору.

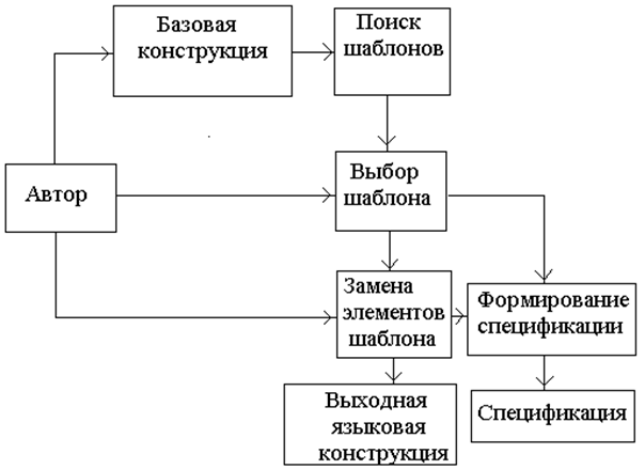



***Рис. 1. Организация формирования текста на основе шаблонов***

Шаблоны состоят из элементов базовой семантической конструкции и пронумерованных заменяемых элементов. Как указывалось выше, список прототипов шаблонов состоит из тысяч предложений на входном языке. Список шаблонов формируется путем выбора наиболее часто встречающихся в научных публикациях языковых конструкций, которые в своей совокупности достаточны для адекватной передачи положений научно-технических публикаций. На практике список шаблонов может пополняться авторами публикаций в случае, если ни один из шаблонов не способен адекватно отразить смысл предложения.

Выбрав наиболее подходящий по смыслу формируемого предложения шаблон на родном языке, автор выполняет замену в нем заменяемых слов, формируя языковую конструкцию предложения и соответствующему ему спецификацию. Фактически результатом описанного процесса является текст публикации и соответствующий ему список спецификаций, используемых только при переводе.

Специалистами-лингвистами выполняется перевод ограниченного списка шаблонов на несколько языков. Если предположить, что система ориентирована на перевод публикаций с использованием трех языков: русского, английского и итальянского, то в рамках рассмотренного выше примера, шаблон с номером 172 представляет собой совокупность семантически одинаковых предложений на трех языках:

Русском: “**Для повышения эффективности** идентификации (1) абонентов (2) в (3) интегрированных (4) базах (5) данных (6) **предлагается использовать** необратимые (7) булевы (8) преобразования (9)”. Базовыми элементами этой конструкции являются: “**Для повышения эффективности** ... **предлагается использовать**”.

Английском: “**To improve the efficiency of** subscriber (2) identification (1) in (3) integrated (4) databases (5-6) **it is proposed to use** irreversible (7) Boolean (8) manipulations (9)”. с базовыми элементами: “**To improve the efficiency of….. it is proposed to use…**”.

Итальянском: “**Per migliorare l’efficienza di** identificazione (1) dell’abbonato (2) nei (3) database (5-6) integrati (4), **si propone di utilizzare** manipolazioni (9) booleane (8) irreversibili (7)”. Базовые элементы семантики конструкции имеют вид: “**Per migliorare l’efficienza di….. si propone di utilizzare…”.**

Структурно процесс перевода предложения с использованием спецификации показан на рис.2.

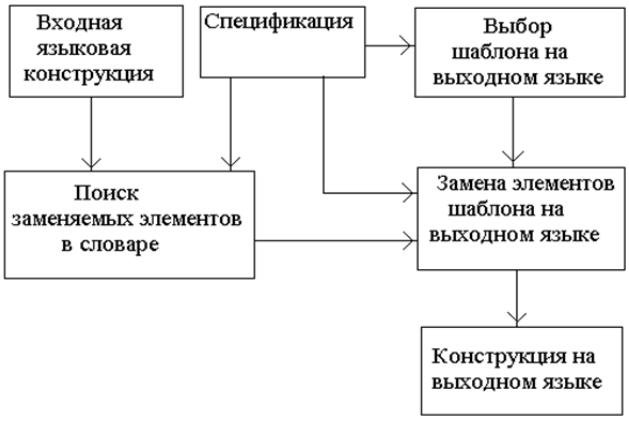



***Рис. 1. Организация перевода с использованием языковых шаблонов***

В отличие от формирования текста, его перевод на один из языков выполняется в автоматическом режиме.

При формировании текста научно-технической публикации на выходном языке каждая языковая конструкция (предложение) переводится отдельно. С использованием спецификации входной языковой конструкции выполняется выделение в ней заменяемых элементов. Для таких элементов в выполняется поиск в словаре соответствующих слов выходного языка. Но номеру из спецификации выбирается шаблон языковой конструкции на выходном языке.

В выбранном шаблоне на основе ссылок подстановок осуществляется замена переменных элементов на перевод слов входной языковой конструкции. В результате такой замены формируется языковая конструкция (предложение) на выходном языке, которая семантически идентичная входной.

В рамках рассматриваемого примера при переводе с русского на английский конструкции: "**Для повышения эффективности** исправления (1) пачки (2) ошибок (3) в (4) каналах (5) со (6) спектральной (7) модуляцией (8) **предлагается использовать** взвешенные (9) контрольные (10) суммы (11)" с использованием словаря осуществляется перевод заменяемых слов, отмеченных цифрами. Далее, по номеру 172, указанном в спецификации выбирается англоязычный шаблон "**To improve the efficiency of** subscriber (2) identification (1) in (3) integrated (4) databases (5-6) **it is proposed to use** irreversible (7) Boolean (8) manipulations (9)" в котором, в соответствии со спецификацией выполняется замена пронумерованных слов, в результате чего формируется предложение: "**To improve the efficiency of** bust (2) error (3) correction (1) in (4) channels (5) with (6) spectral (7) modulation (8) **it is proposed to use** weighted (9) checksums (10-11)".

Аналогично, при переводе на итальянский замена слов в шаблоне формирует выходное предложение в виде: "**Per migliorare l'efficienza di** correzione (1) di errori (3) del cluster (2) in (4) canali (5) con (6) modulazione (8) spettrale (7)**, si propone di utilizzare** checksum (10-11) ponderate (9)".

Как показал проведенный анализ, основная часть семантических ошибок при переводе публикаций научного и технического характера связаны с неправильным пониманием конструкции предложения на неродном для специалиста языке. Реализация предложенного подхода позволяет в значительной мере свести к минимуму такие ошибки перевода.

## Анализ эффективности и результаты экспериментальных исследований

Вводимое в рамках подхода ограничение на конструкции входного текста позволяет достаточно просто, без использования значительных вычислительных ресурсов реализовать на приемлемом для практики уровне семантическое соответствие синтаксических конструкций различных языков. Фактически, в рамках предлагаемого подхода задача перевода разделяется на две составляющие: перевод языковой конструкции с одного естественного языка на другой и перевод отдельных слов заполнения этой конструкции. В рамках предложенного подхода первая, наиболее сложно формализуемая часть перевода, выполняется квалифицированным переводчиком, владеющим как входным, так и выходным языками. Вторая, существенно более простая часть: замена слов одного языка словами другого - осуществляется компьютерными средствами. Это позволяет оптимизировать разделение человеческой и компьютерной составляющих в процессе перевода: плохо формализуемая часть выполняется человеком, а более трудоемкая, но относительно простая часть реализуется компьютерными средствами.

При этом важным является то, что наиболее сложная часть работы, требующая труда высококвалифицированных переводчиков, выполняется одноразово в процессе инициализации системы. После инициализации, в процессе которой подбираются и переводятся шаблоны, система переводит научно-технические публикации автоматически.

Для практического исследования эффективности предложенного подхода была разработана экспериментальная система, ориентированная на перевод научно-технических текстов ограниченной тематики: компьютерные технологии. Это позволило ограничить количество шаблонов трехстами и использовать терминологически более качественный тематический словарь. В разработанной системе использованы три языка: русский, английский и итальянский. В перспективе создание экспериментального образца системы, ориентированной для перевода публикаций на китайский язык. Экспериментальные исследования показали, что система обеспечивает достаточно высокое

для адекватного понимания качество перевода для ограниченной предметной области.

Достоинствами предложенного подхода к компьютерному переводу текстов научно-технического характера по сравнению с известными системами являются:

1. Ориентация на использование относительно небольших ресурсов компьютерных систем; это позволяет реализовать систему перевода на простых, в том числе мобильных, вычислительных платформах с высокой производительностью.

2. Возможность перевода с языков, которыми пользователь совершенно не владеет, что особенно важно для преодоления языкового барьера между восточными и западными специалистами.

3. Изначальная ориентация на многоязыковый перевод.

Вместе с тем, необходимо отметить, что высокий уровень эффективности компьютерного перевода достигнут за счет ограничений на возможность использования языковых конструкций при написании публикации автором или редактировании готовой статьи специалистом. Экспериментальное исследование показало, что такие ограничения не сказываются существенным образом на качестве публикации, а скорость подготовки текста несколько возрастает. Фактически процесс написания публикации в значительной степени подобен современным технологиям интерактивного программирования, поэтому психологически не вызывает дискомфорта, сопровождается снижением уровня грамматических и стилистических ошибок.

## Заключение

Таким образом, в работе предложен подход к повышению эффективности компьютерного перевода. Разработанный подход позволяет в значительной мере обойти узловую для компьютерных систем перевода проблему отсутствия как формальной модели естественного языка, так и моделей трансформации конструкций с одного языка на другой за счет ограничений на возможный набор языковых конструкций. Проведенные теоретически и экспериментальные исследования показали, что указанные ограничения вполне оправданы для определенной сферы переводимого материала, в частности, для публикаций научного и технического характера. Именно на этот, один из наиболее практически важных классов переводимых материалов ориентирован предлагаемый подход.

## Литература

1. Хроменков П.Н. Современные системы машинного перевода /Хроменков П.Н. -М.: Мысль, 2005 –191 с.
2. Сдобников В.В.Теория перевода./Сдобников В.В., Петрова О.В.-М:Высшая школа: , 2008 – 254 с.
3. Марчук Ю.Н. Модели перевода /Марчук Ю.Н.- М.: Академия, 2010 - 188 с.
4. Briscoe T. Lexical Issues in Natural Language Processing //Natural Language and Speech, Springer-Verlag, 1991, - P. 39-40.
5. Кисленко Ю.И. Системна організація мови / Кисленко Ю.І.-К: Український літопис. 1997.- 217 с.
6. Белоногов Г.Г. Компьютерная лингвистика и перспективные информационные технологии: теория и практика построения систем автоматической обработки текстовой информации / Белоногов Г.Г. - М.: Русский мир, 2004 - 358 с.
7. Шевченко О.Н., Шевченко Д.С. Компьютеризированная система обучения иностранному языку/ Шевченко О.Н., Шевченко Д.С.//Системний аналіз та інформаційні технології: матер.наук-технічної конф. 26-30 квітня 2013: тез.допов.- К.:НТУУ “КПІ”, ІПСА.-2013- С.263.

УДК 621.396.946.2

*ВОРОТНІКОВ В. В.*

# ВПЛИВ СТУПЕНЯ ЗВ'ЯЗНОСТІ РЕЗЕРВНИХ ШЛЯХІВ НА НАДІЙНІСТЬ БАГАТОШЛЯХОВОЇ МАРШРУТИЗАЦІЇ В MANET-МЕРЕЖАХ

У роботі проведено аналіз ефективності багатошляхової маршрутизації в MANET-мережах. Більшість протоколів маршрутизації в мобільних мережах у зв'язку із можливою високодинамічною зміною топології використовують реактивні методи. В реактивних протоколах вузлом-джерелом під час процедури розрахунку маршруту, може бути визначено декілька резервних шляхів до вузла-адресата. В таких випадках, багатошляхову маршрутизацію може бути використано для зменшення затримки при відновленні процесу передачі даних після виходу із ладу елементів мережі за рахунок використання резервних шляхів шляхом перемиканням поточного трафіка на інші альтернативні шляхи.

Досліджено вплив ступеня зв'язності резервних шляхів на надійність з'єднання. Доведено, що ефективним рішенням при багатошляховій маршрутизації є використання Low Coupling (слабкопов'язаних) шляхів, що дозволяє у порівнянні з одношляхової маршрутизацією для ненадійних з'єднань отримати приріст в ймовірності передачі даних до 20%.

The paper analyzes the effectiveness multipath routing in MANETs. Most routing protocols in mobile networks due to the topology change high dynamic possible using reactive methods. In reactive protocols, a source node during route calculation procedure can be defined several ways to backup node addressed. In such cases, multipath routing can be used to reduce the delay in the recovery process data after a system network elements through the use of backup paths by switching this traffic to other alternative routes.

The influence degree of connectivity backup paths for reliable connection. It is proved that an effective solution for multipath routing is the use of low coupling (LowCoup) tract, which allows for comparison with singlepath routing for unreliable connections to get a boost in the probability of data transmission up to 20%.

## Постановка проблеми

Мобільні ad-hoc мережі (MANET) – вид безпроводових ad-hoc мереж із довільною структурою, що складається із мобільних маршрутизаторів (і прилеглих хостів), з'єднаних безпроводовими каналами зв'язку. Маршрутизатори можуть вільно пересуватись у будь-якому напрямку і організовуватись довільно, тому топологія безпроводової мережі може змінюватись швидко і непередбачено.

Особливостями організації MANET є те, що кожен вузол є потенційним маршрутизатором; на відміну від проводових мереж із статичною топологією, топологія MANET мережі має набагато більше надмірних з'єднань і є динамічно змінюваною; характеристики каналів (пропускна здатність, частота появи помилок тощо) є асиметричними і залежать від напрямку передачі.

Однією із центральних проблем MANET мереж залишається проблема оптимального використання мережних ресурсів. Шляхи її вирішення лежать у напрямку використання засобів і механізмів багатошляхової маршрутизації та розподілу трафіку [ 1, 2 ].

## Анализ останніх досліджень і публікацій

Більшість протоколів маршрутизації, що було запропоновано для ad hoc мереж є одношляховими, що використовують лише один маршрут передачі даних між вузлом-джерелом і узлом-адресатом.

Питанням використання протоколів багатошляхової маршрутизації та розподілу трафіку у ad hoc мережах присвячений ряд робіт.

У роботі, що присвячено управлінню потоками в мережах [3] під фіксованою маршрутизацією розуміють використання єдиного шляху доставки повідомлень. Альтернативна маршрутизація (багатошляхова) передбачає наявність декількох шляхів. При фіксованій маршрутизації затримки при передачі даних мають менші значення у порівнянні із альтернативною маршрутизацією. Проте, переваги фіксованої маршрутизації в мережі зникають, якщо її структура не оптимальна за критерієм зв'язності або якщо топологія мережі змінюється так швидко, що алгоритми альтернативної маршрутизації не справляються з регулюванням потоків у зв'язку із більшими затримками в мережі.

В роботі [ 4 ] розглянуто можливість застосування протоколу маршрутизації OLSR у багато-

хопових ad-hoc мережах в умовах відносної стаціонарності вузлів. Визначено вплив швидкості пересування вузлів на характеристики мережі.

В роботі [5] вирішується задача маршрутизації і планування використання ліній радіозв'язку для мереж пакетної передачі даних за допомогою методу нейронної мережі. Проте, автором зауважено, що при еволюції мережі гарантується тільки локальний мінімум цільової функції.

В [6] досліджено статичний протокол маршрутизації, що не перенаправляє потоки у разі несправних шляхів. Показано, що у випадку короткотермінової відмови вузла або з'єднання, ремаршрутизація не є необхідною при наявності декількох шляхів. Якщо відмови з'єднань, що виникають у реальних мережах, не відновлюються протягом певного часу, ремаршрутизація є вкрай необхідною. У результаті виникнення відмов погіршується ефективність «маршрутизації без пам'яті», тому що у ациклічного графа, повинен бути, принаймні, один вузол з єдиним маршрутом до місця призначення.

В [7] показано, що вартість утримання двох незалежних шляхів при маршрутизації від джерела в реактивному протоколі SMR, вища, ніж побудова двох шляхів, за припущенням, що вони існують поки не вийдуть обидва із ладу.

Тому, час необхідний для встановлення нових маршрутів після виявлення відмов, може бути зменшений за рахунок більшої надмірності з'єднань між джерелом і адресатом, що притаманно MANET мережам.

У роботі [8], розрізняють декілька типів багатошляхових маршрутів за ступенем зв'язності:

1) незалежні (непов'язані) шляхи, що не перетинаються (не мають ні загальних транзитних вузлів, ні загальних транзитних ребер, крім початкового (джерела) і кінцевого (адресата) вузлів;

2) LoCoup-залежні (Low Coupling, слабкопов'язані) шляхи, що не мають загальних ребер, але можуть мати загальні транзитні вузли;

3) зоннозалежні шляхи, що мають загальні транзитні ребра в межах деякої зони (числа хопів).

**Метою** роботи є підвищення надійності багаошляхового маршруту.

### Формулювання завдання дослідження

Більшість протоколів багатошляхової маршрутизації вирішують три основні задачі: створення маршруту, обслуговування маршруту і розподілення трафіку [9, 10]. У роботі оцінюється процедура формування багатошляхового маршруту.

Для використання багатошляхової маршрутизації необхідною умовою є наявність декількох альтернативних шляхів, а не лише одного оптимального. Крім того, багатошляхова маршрутизація є прийнятною при відсутності надмірних накладних витрат на управління і обслуговування таких маршрутів.

Тому, при постановці завдання будемо вважати, що такі умови, як наявність альтернативних шляхів і прийнятність їх обслуговування виконані.

У відповідності до цього задача досліджень формулюється наступним чином.

Нехай MANET мережа задана графом $G(V,E,d(e))$, де: $V$ – множина вузлів графа; $E$ – множина ребер графа, *d(e)* – вага шляху (сума ваг ребер даного шляху) від початкового вузла $v_s$ до вузла кінцевого $v_d$; $w(v_s,v_d)$ - вага ребра між $v_s$ -м і $v_d$ -м вузлами.

Для графа $G$ визначено множину можливих маршрутів $\Pi=\{\pi_1,...,\pi_N\}$ між $v_s, v_d$.

Необхідно: з множини $\Pi=\{\pi_1,...,\pi_N\}$ сформувати множини з незалежними і LowCoup-залежними і зоннозалежними шляхами одного маршруту і оцінити ефективність їх використання для багатошляхової маршрутизації.

Кожна із множин шляхів трьох типів (незалежні, LoCoup-залежені, зоннозалежні) містить не більше $k$ альтернативних шляхів.

### Виклад основного матеріалу

Із загальної універсальної множини шляхів маршруту $\Pi=\{\pi_1,...,\pi_N\}$ сформуємо:

1) множину $\Pi_{disj}^{s,d}\in\Pi$, що складається із $k$ - альтернативних незалежних шляхів $\pi_i$ між вузлом-джерелом $v_s$ і вузлом-адресатом $v_d$:

$$\Pi_{disj}^{s,d}=\{\pi_i(V,E,d(e)\},\qquad(1)$$

$$\pi_i=\{(V_s,\{\overline{V}_{\pi_i}\},V_d),\overline{E}_{\pi_i},d_{\pi_i}(e)\}$$

де $i,j=\overline{1,k}$, $i\neq j$; $\overline{V_{\pi_i}}$, $\overline{E}_{\pi_i}$, $d_{\pi_i}(e)$ – вектор транзитних вузлів, ребер і вартість шляху $\pi_i$ потужністю $l$-2, $l$ - кількість вузлів в маршруті, відповідно.

Незалежні шляхи маршруту повинні задовольняти умові:

$$\forall \pi_i : \begin{cases} \overline{V_{\pi_i}} \cap \overline{V_{\pi_j}} / \{v_s, v_d\} = \mathrm{O}; \\ \overline{E}_{\pi_i} \cap \overline{E}_{\pi_j} = \mathrm{O}; \\ d_{\pi_i}(e) = \min\limits_{1 \le i \le k} (d(\Pi_{disj}^{s,d})). \end{cases} \quad (2)$$

2) множину LoCoup-залежних шляхів із можливими загальними транзитними вузлами $\Pi_{mesh}^{v} \in \Pi$, що задовольняє умові:

$$\forall \pi_i \in \Pi_{mesh}^{v} : \begin{cases} (\overline{V_{\pi_i}} \cap \overline{V_{\pi_j}}) / \{v_s, v_d\} \neq \mathrm{O}; \\ \overline{E}_{\pi_i} \cap \overline{E}_{\pi_j} = \mathrm{O}; \\ d_{\pi_i}(e) = \min\limits_{1 \le i \le k} (d(\Pi_{disj}^{s,d})). \end{cases} \quad (3)$$

3) множину зоннозалежних шляхів із можливими загальними транзитними вузлами і транзитними ребрами $\Pi_{mesh}^{v,e} \in \Pi$, що задовольняє умові:

$$\forall \pi_i \in \Pi_{mesh}^{v,e} : \begin{cases} (\overline{V_{\pi_i}} \cap \overline{V_{\pi_j}}) / \{v_s, v_d\} \neq \mathrm{O}; \\ \overline{E}_{\pi_i} \cap \overline{E}_{\pi_j} \neq \mathrm{O}; \\ d_{\pi_i}(e) = \min\limits_{1 \le i \le k} (d(\Pi_{disj}^{s,d})). \end{cases} \quad (3)$$

Ефективність використання $\Pi_{disj}^{s,d}$, $\Pi_{mesh}^{v}$ і $\Pi_{mesh}^{v,e}$ для багатошляхової маршрутизації оцінимо за значенням полінома надійності [8].

При формалізації поняття полінома надійності, скористаємось визначеннями алгебри логіки: для множини мінімальних шляхів $\Pi_{disj}^{s,d}$ уведемо подію $E_i$, що всі ребра шляху $\pi_i$, між $v_s$ и $v_d$ працездатні. Події $\{E_i\}$ не є незалежними.

Нехай $\overline{E}_i$ − доповнення події $E_i$. Тоді визначимо подію $D_1 = E_1$, у цілому:

$$D_i = \overline{E_1} \cap \overline{E_2} \cap \ldots \cap \overline{E}_{i-1} \cap E_i . \quad (4)$$

Події $D_i$ є незалежними і часто називаються подіями «незалежного добутку». У такому випадку надійність багатошляхового маршруту мережі розраховується, як:

$$P^{s,d}(G) = \sum_{i=1}^{k} P[D_i]. \quad (5)$$

Проаналізуємо надійність багатошляхового маршруту, що складається із LoCoup-залежних шляхів. Для цього використаємо вираз включення/виключення:

$$\sum_{j=1}^{k} (-1)^{j+1} \sum_{i \subseteq \{1,\ldots k\}} P(E_i) \quad (6)$$

де $E_i$ - подія, що всі шляхи $\forall \pi_i \in \Pi_{mesh}^{v,e}$ при $i \in \overline{1,k}$ формуються не триваліше ніж $t$.

Приклад. Нехай задано мережу, що подано графом (рис. 1).

Визначимо ефективність використання багатошляхової маршрутизації ($k = 3$) для передачі даних між вузлом-джерелом $v_s = 1$ і вузлом-адресатом $v_d = 8$ відповідно до (1-3).

Множину незалежних шляхів багатошляхового маршруту $\Pi_{disj}^{1,8} = \{\pi_1, \pi_2, \pi_3\}$ наведено на рис. 1а-1в. До неї включено три шляхи:

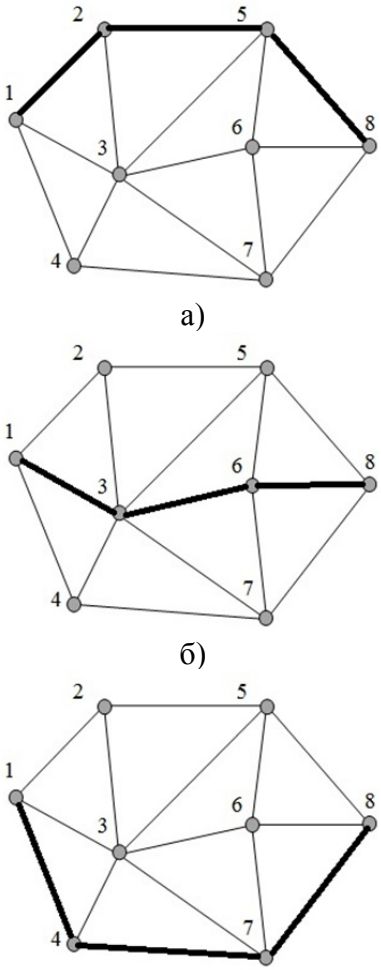


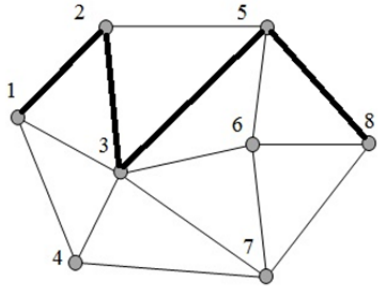

***Рис. 1. Незалежні шляхи багатошляхового маршруту: а) -*** $\pi_1 = \{1,2,5,8\}$ ***; б) -*** $\pi_2 = \{1,3,6,8\}$ ***; в) -*** $\pi_3 = \{1,4,7,8\}$

Для спрощення розрахунків введемо обмеження, що всі ребра графа рівнонадійні, тобто ймовірність функціонування кожного хопа однокова і дорівнює $p$.

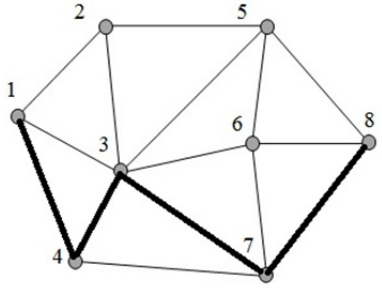
Визначимо надійність кожного шляху, використавши метод включення / виключення (4-5). Шляхи $\pi_1, \pi_2, \pi_3$ складаються із трьох хопів і надійність кожного дорівнює $p^3$. Поліном надійності багатошляхового маршруту (1), що складається із трьох незалежних шляхів:

$$P_{disj}^{1,8}(G,p) = 3p^3 - p^6 - p^9.$$

2. Визначимо надійність багатошляхової маршрутизації, що складається із LoCoup-залежних шляхів із можливими спільними транзитними вузлами (2). Множину $\Pi_{mesh}^{v,e} \in \Pi$ буде сформовано із шляхів: $\Pi_{mesh}^{v,e} = \{\pi_2, \pi_4, \pi_5\}$.




а)




б)

***Рис. 2. LoCoup-залежні шляхи багатошляхового маршруту із спільним вузлом***
***а) -*** $\pi_4 = \{1,2,3,5,8\}$ ***; б) -*** $\pi_5 = \{1,4,3,7,8\}$

Граф мережі багатошляхового маршруту з LoCoup-залежним шляхами може бути поданий у вигляді (рис. 3):

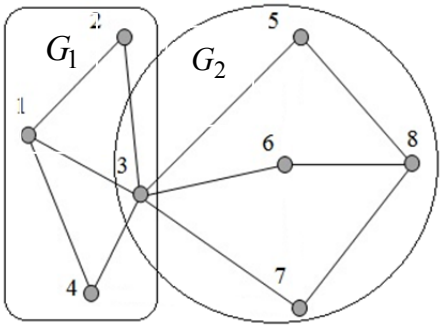



***Рис. 3. LoCoup-залежні шляхи багатошляхового маршруту із спільним вузлом*** $v_3$

Поліном надійності маршруту $\Pi_{mesh}^{1,8}$ для такої мережі розраховується як надійність послідовного з'єднання двох сегментів $G_1$ і $G_2$, що обумовлюють передачу даних маршрутом з $v_1$ у $v_3$ і далі з $v_3$ у $v_8$. Аналіз сегментів свідчить, що передачу даних з $v_1$ у $v_3$, з $v_3$ у $v_8$ можна реалізувати незалежними шляхами відповідних багатошляхових маршрутів $P_{disj}^{1,3}(G_1,p)$ і $P_{disj}^{3,8}(G_2,p)$. Тоді $P_{mesh}^{3}(G,p)$ буде розраховано як:

$$P_{mesh}^{3}(G,p) = P_{disj}^{1,3}(G_1,p) \cdot P_{disj}^{3,8}(G_2,p),$$

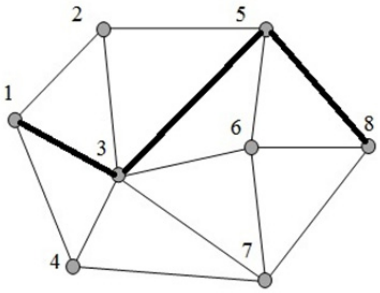
де $P_{disj}^{1,3}(G_1,p) = p + 2p^2 - 2p^3 - p^4 + p^5$;

$$P_{disj}^{3,8}(G_2,p) = 3p^2 - 5p^4 + p^6.$$

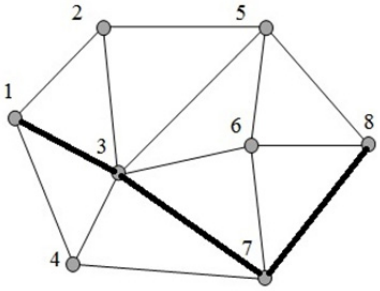
3. Для тієї ж мережі визначимо поліном надійності багатошляхового маршруту, що складається із зоннозалежних шляхів із можливими спільними транзитними ребрами.

В даному прикладі обмежимось шляхами без петель, що складаються не більш ніж із трьох хопів; відповідно до (3), резервні шляхи не повинні мати з основними незалежними шляхами більш одного спільного ребра.

Множина трьох-хопових шляхів багатошляхового маршруту $\Pi_{mesh}^{1,8}$, за умови $\Pi_{disj}^{v} \in \Pi_{mesh}^{1,8}$, додатково буде складатись із двох резервних шляхів (рис. 4).




а)




б)

***Рис. 4. LoCoup-залежні шляхи багатошляхового маршруту із спільним ребром*** $e_{1-3}$ ***:***
***а) -*** $\pi_6 = \{1,3,5,8\}$ ***; б) -*** $\pi_7 = \{1,3,7,8\}$

У знайдених шляхів існує загальне ребро $e_{1-3}$. Для такого випадку поліном надійності необхідно розрахувати як для графа мережі, що подано на рис. 5. Маршрут можна представити як послідовне з'єднання двох сегментів: ребра $e_{1-3}$ і сегмента, що можна представити як паралельне з'єднання ребер $(e_{3-5}, e_{5-8})$, $(e_{3-6}, e_{6-8})$, $(e_{3-7}, e_{7-8})$.

Граф мережі багато шляхового маршруту із зоннозалежним шляхами поданий на рис. 5.

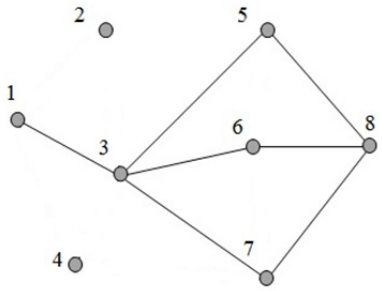



***Рис. 5. LoCoup-залежні шляхи багатошляхового маршруту із спільним ребром*** $e_{1-3}$

Поліном надійності наступний:

$$P_{mesh}^{e_{1-3}}(G,p) = P(\pi\{1,3\}) \cdot P_{disj}^{3,8}(G_2,p) =$$
$$= p(3p^2 - 3p^4 + p^6)$$

На рис. 6 показано залежність надійності багатошляхового маршруту $P(G,p)$ від ймовірність функціонування типового з'єднання $p$. Результати отримані за умови абсолютно надійних вузлів.

Переваги використання LoCoup-залежних шляхів у багатошляховій маршрутизації обумовлені більшою кількістю варіантів побудови маршруту (степінь полінома надійності).

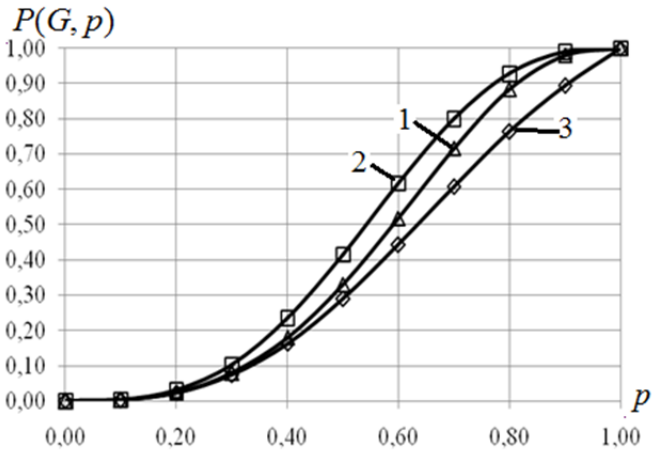



***Рис. 6. Залежність надійності багатошляхового маршруту*** $P(G,p)$ ***від надійності*** $p$ ***для шляхів: 1) – незалежних; 2) – LoCoup-залежних; 3) – зоннозалежних.***

## Висновки

Реактивні протоколи, що використовуються у високодинамічних MANET мережах, дозволяють під час процедури розрахунку маршруту вузлом-джерелом визначати декілька резервних шляхів до вузла-адресата.

Вид зв'язності резервних шляхів має прямий вплив на надійність маршруту. Використання незалежних шляхів має кращі показники надійності ніж використання зоннозалежних. Проте, незалежні шляхи для мережі з інтенсивним трафіком явище рідке, а одночасне їх утримання у монопольному доступі неефективно для багатопродуктових потоків.

Більш доцільним є використання LoCoup-залежних шляхів, що одночасно знімає обмеження обслуговування шляхів і збільшує надійність багатошляхового маршруту за рахунок значно більшої їх кількості у порівнянні із кількістю незалежних шляхів.

## Список літератури

1. Вишневський В. М., Ляхов А. И., Портной С. Л., Шахнович И. В. Широкополосные беспроводные сети передачи информации. / В. М. Вишневський, А. И. Ляхов, С. Л. Портной, И. В. Шахнович // – М.: Техносфера, 2005. – 592 с.
2. Кульгин М. Технологии корпоративных сетей / М. Кульгин. – СПб.: Питер, 2000. – 704с.
3. Хелебі С. Принципы маршрутизации в Internet, 2-е издание.: пер. с англ. / С. Хелебі, Д. Мак –Ферсон // М.: Издательский дом "Вильямс", 2001. — 448 с.
4. Алєксєєва Я. А. Аналіз алгоритмів маршрутизації в ad-hoc мережах // Электроника и связь. Тематический выпуск «Проблемы электроники» / Я. А. Алєксєєва, М. Ю. Терновой // ч. 2. – 2008. – С. 204-207.
5. Сонькин М.А. Информационная технология интеграции компонентов многоуровневых систем с пакетной передачей данных / М.А. Сонькин, Е.Е. Слядников // Изв. ТПУ. – 2006. – Т. 309, № 6. – С. 93–101.

6. Гринберг Я. Р. Применение метода агрегации данных в задаче нахождения узлов с дополнительной функциональностью / Я. Р. Гринберг, И. И. Курочкин, А. В. Корх, Р. М. Алыгулиев, М. А. Гашимов // Информационные технологии и вычислительные системы [Электронный журнал]. – 2013. – №2.– стр. 27-39. Режим доступа к журн.: http://www.jitcs.ru.
7. S.-J. Lee, M. Gerla. "Split multipath routing with maximally disjoint paths in ad hoc networks". IEEE International Conference on In Communications, Vol. 10, 07 August 2002, pp. 3201-3205.
8. Marc Mosko, J.J. Garcia-Luna-Aceves. Multipath Routing in Wireless Mesh Networks. irst IEEE Workshop on Wireless Mesh Networks, IEEE WiMesh 2005, Held in conjunction with IEEE SECON 2005, 26 September 2005, Santa Clara, California USA, pp.64-70.
9. C. Fonseca, J.Mocito, L. Rodriges. Low-Coupling Cluster-based Multipath Routing for Wireless Network., 20th ICCCN, Maui, Hawaii, July, 2011. p. 216.
10. Кулаков Ю.А. Многопутевая маршрутизация в mesh сетях с использованием резервных слабосвязанных путей. / Ю.А. Кулаков, В.В. Воротников // Информационные технологии моделирования и управления: науч.-теорет. журн. – Воронеж: Научная книга – 2014. – №1(85). – С. 68-77.

УДК 681.518

*РОЛИК А.И.*

# УПРАВЛЕНИЕ УРОВНЕМ УСЛУГ КОРПОРАТИВНОЙ ИТ-ИНФРАСТРУКТУРЫ НА ОСНОВЕ КООРДИНАТОРА

Предложено осуществлять управление уровнем услуг в корпоративной ИТ-инфраструктуре посредством двухуровневой системы управления с координатором. Доказана целесообразность использования комбинированного принципа управления на основе обратной связи с учетом возмущающего воздействия. Показана возможность реализации принципа координации на основе системы правил.

Service level management of corporate IT infrastructures was considered. Architecture of the service level management system as a two-level system of IT infrastructure management with coordinator was offered. Appropriateness of combined management principle based on the feedback with consideration of disturbing action was proved. Implementation of the principle of coordination based on the system of rules was shown.

## Введение

Существенная зависимость успешности ведения бизнеса от качества и стабильности предоставления используемых ИТ-услуг делают для ИТ-департамента актуальным постоянное поддержание уровня ИТ-услуг на согласованном с бизнесом уровне [1]. Конкуренция на рынке ИТ-услуг, а также высокая стоимость ресурсов современных ИТ-инфраструктур, посредством которых предоставляются эти услуги, вынуждает ИТ-департамент изыскивать пути решения проблемы по обеспечению поддержания качества ИТ-услуг на согласованном уровне с минимальным количеством используемых для этого ресурсов ИТ-инфраструктуры. Основное направление решения данной проблемы связано с разработкой перспективных систем управления ИТ-инфраструктурой (СУИ), являющихся не столько инструментом автоматизации процессов управления, сколько средством повышения эффективности использования дорогостоящих информационно-телекоммуникационных ресурсов [2]. Внедрение систем проактивного управления распределенной ИТ-инфраструктурой позволяет существенно сократить затраты на ИТ, обеспечивая при этом стабильно высокое качество предоставляемых ИТ-сервисов. Для создания СУИ, ориентированной на управление качеством ИТ-услуг при рациональном использовании ресурсов ИТ-инфраструктуры, необходимо решение ряда задач, связанных, прежде всего, с разработкой методов управления уровнем ИТ-услуг. Поэтому данная статья, посвященная определению принципа построения и функционирования СУИ при управлении уровнем услуг, является актуальной.

## Модель двухуровневой системы управления ИТ-инфраструктурой

В связи с тем, что в настоящее время исследования в области теории управления техническими системами акцентированы на многообъектности, распределенности, большой размерности, возникает потребность выделения специального класса многообъектных распределенных систем управления [3, 4], к которым относятся СУИ. Одним из важнейших вопросов, решаемых при проектировании и эксплуатации таких систем управления, кроме разработки архитектуры, является задача принятия решений. Сложность принятия решений в иерархических системах управления, таких как СУИ, обусловлена тем, что решения принимаются на большинстве уровней иерархии СУИ, а также с ограничением времени на принятие решения. В [4] задачи принятия решений с ограничением времени называются задачами принятия оперативных решений. В [5] предложено такие системы рассматривать как двухуровневые системы управления с координатором.

В [1] предложен декомпозиционно-компенсационный подход к управлению уровнем ИТ-услуг в корпоративных ИТ-инфраструктурах, предполагающий декомпозицию задач управления уровнем услуг и компенсацию негативного влияния различных факторов, таких, как увеличение количества пользователей, отказы в ИТ-инфраструктуре и пр., выделением дополнительных ресурсов критичным приложениям. Подход основан на интегрированном взаимодействии трех иерархических процессов — согласования уровня услуг, планирования ресурсов и управления уровнем услуг с учетом иерархии ИТ-инфраструктуры.

Для реализации этого подхода в данной статье предлагается строить СУИ в виде двухуровневой системы управления с координатором.

Основанием для выделения в СУИ двух уровней является то, что при управлении уровнем услуг СУИ функционирует в различных режимах в условиях неопределенности, неполноты и недостоверности информации, наличия факторов риска, множества конфликтующих критериев и целей подсистем СУИ. От таких систем управления требуется не достижение оптимального функционирования ИТ-инфраструктуры, что практически невозможно, а улучшение качественных характеристик работы ИТ-инфраструктуры. В таких случаях оправдано построение двухуровневых систем с координатором, когда координатор согласовывает самостоятельные решения и действия подсистем СУИ для улучшения работы ИТ-инфраструктуры в целом с точки зрения качества предоставляемых ИТ-услуг. При этом действия координатора должны быть направлены на улучшение глобальной функции качества предоставления услуг, а принятие им решений осуществляется в условиях неопределенности. На рис. 1 приведена модель СУИ, управляющей информационно-телекоммуникационной системой (ИТС), в виде двухуровневой системы с координатором [3, 5].

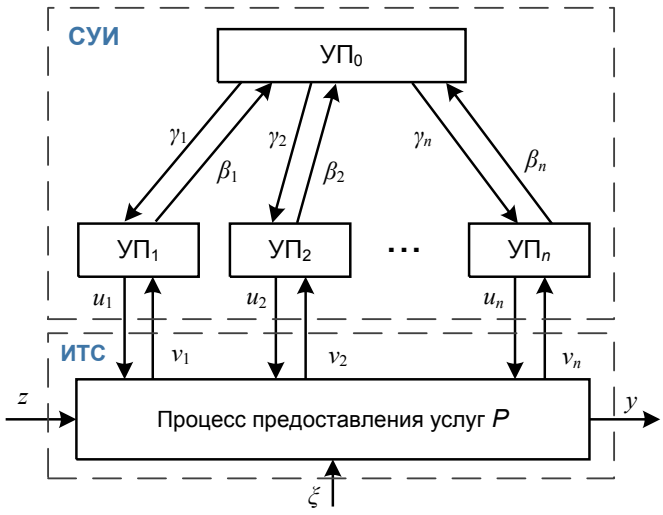



***Рис. 1. Представление СУИ в виде двухуровневой системы***

Расположение управляющих подсистем (УП) отображает иерархическую структуру СУИ. Модель состоит из вышерасположенной по иерархии управляющей подсистемы ($УП_0$), $n$ нижерасположенных управляющих подсистем ($УП_1,\ldots,УП_n$) и управляемого процесса $P$, протекающего в ИТ-инфраструктуре.

Взаимодействие УП по вертикали осуществляется следующим образом. Команды, сигналы, воздействия или вмешательства (входы) $\gamma_1,...,\gamma_n$, передаваемые от $УП_0$ к $УП_1,\ldots,УП_n$, являются координирующими. Командные сигналы или воздействия (входы) $(u_1,...,u_n)$ от $УП_1,\ldots,УП_n$ к процессу $P$ являются управляющими. Снизу вверх поступают сигналы обратной связи или информационные сигналы: от процесса $P$ к $УП_1,\ldots,УП_n$ — $v_1,...,v_n$, и от управляющих подсистем к координатору — $\beta_1,...,\beta_n$. Описание двухуровневой СУИ может быть осуществлено посредством терминальных переменных (входов и выходов). В этом случае УП описываются как функциональные подсистемы, выходы которых однозначно определяются входами [5].

Процесс $P$ можно описывать как управляемую подсистему, на которую воздействуют управляющие сигналы $u$ от $УП_1,\ldots,УП_n$, $u \in U$, $U$ — множество управляющих воздействий; поступают входные сигналы $z$, $z \in Z$, представляющие собой запросы пользователей; и сигналы $\xi$, $\xi \in \Xi$, являющиеся возмущающими воздействиями. К возмущающим воздействиям $\Xi$ относятся неисправности в ИТ-инфраструктуре, функциональные отказы в элементах ИТ-инфраструктуры, запросы других пользователей, которые, являясь помехой для запросов рассматриваемых пользователей, затрудняют достижение целей управления. Выходом процесса $P$ является $y$, $y \in Y$, где $Y$ — множество выходов процесса $P$, ответы ИТ-инфраструктуры на запросы пользователей.

Процесс P можно представить в виде следующего отображения на основе декартова произведения:

$$P : U \times Z \times \Xi \to Y. \qquad (1)$$

Множество $U$ управляющих сигналов, воздействующих на процесс $P$ со стороны $УП_1,\ldots,УП_n$, удобно представлять в виде декартова произведения $n$ множеств [3, 5]

$$U = U_1 \times U_2 \times ... \times U_n. \qquad (2)$$

При этом каждая $g$-я управляющая подсистема $УП_g$, $g = \overline{1,n}$, обладает полномочиями выбора $g$-й компоненты $u_g$ управляющего воздействия $u$ для оказания непосредственного влияния на процесс $P$.

На входы каждой $g$-й управляющей подсистемы $УП_g$, $g = \overline{1,n}$, поступают два сигнала: координирующий сигнал $\gamma_g \in \Gamma$ от $УП_0$ и ин-

формационный сигнал обратной связи $v_g$ в виде данных мониторинга. Управляющим выходом УП$_g$, $g=\overline{1,n}$, является воздействие $u_g$, выбираемое УП$_g$ из множества $U_g$. Предположим, что каждая из УП$_g$, $g=\overline{1,n}$, реализует отображение $C_g$, $g=\overline{1,n}$, такое, что

$$C_g : \Gamma \times V_g \to U_g, \qquad (3)$$

где $V_g$ — множество данных мониторинга $v_g$, поступающих в СУИ от ИТ-инфраструктуры, $v_g \in V_g$. Данные мониторинга $V_g, g=\overline{1,n}$, являются сигналами обратной связи для локального контура управления на основе УП$_g$, $g=\overline{1,n}$.

Сигналы обратной связи $v_g$, поступающие на вход УП$_g$, $g=\overline{1,n}$, получены в результате мониторинга ИТ-инфраструктуры. Они содержат информацию относительно протекания процесса $P$. Естественно, эти сигналы функционально зависят от управляющих сигналов $u$, входов $z$, возмущений $\xi$ и выходов $y$. Эту зависимость можно представить отображением [3, 5]

$$f_g : U \times Z \times \Xi \times Y \to V_g. \qquad (4)$$

Управляющая подсистема УП$_0$ является координатором и вырабатывает координирующие сигналы $\gamma_g \in \Gamma$, $g=\overline{1,n}$, причем сигнал $\gamma_g$ с $g$-го выхода УП$_0$ поступает только на $g$-й вход нижерасположенной управляющей подсистемы УП$_g$, $g=\overline{1,n}$. Координатор УП$_0$ вырабатывает сигнал на основе анализа информации, поступающей на его вход от УП$_g$, $g=\overline{1,n}$, и представляющей собой сигналы обратной связи и обобщенную информацию о состоянии и функционировании ИТ-инфраструктуры. В таком случае можно считать, что в координаторе реализуется отображение $C_0$ такое, что

$$C_0 : \mathrm{B} \to \Gamma, \qquad (5)$$

где B — множество информационных сигналов $\beta$, реализующих обратную связь. Причем $\beta=(\beta_1,...,\beta_n)$ представляет собой совокупность сигналов $\beta_g, g=\overline{1,n}$, обратной связи, поступающих в координатор УП$_0$ от подсистем УП$_g$, $g=\overline{1,n}$.

Аналогично (4) сигнал обратной связи $\beta$, поступающий в УП$_0$, несет в себе информацию о состоянии всех нижерасположенных подсистем, поэтому он определяется отображением

$$f_0 : \Gamma \times V \times U \to \mathrm{B}, \qquad (6)$$

где $V = V_1 \times ... \times V_n$. Таким образом, B является функцией координирующих сигналов $\gamma_g$, $g=\overline{1,n}$, сигналов обратной связи $v=(v_1,...,v_n)$, поступающих в УП$_g$, $g=\overline{1,n}$, и управляющих воздействий $u=(u_1,...,u_n)$.

На модели, изображенной на рис. 1, не показано в явном виде взаимодействие между подсистемами УП$_1$,…,УП$_n$, а также не показано и непосредственное влияние УП$_0$ на функционирование ИТ-инфраструктуры и получение координатором УП$_0$ сигналов обратной связи непосредственно от элементов ИТ-инфраструктуры, что имеет место в реальных СУИ.

В соответствии с [5] координация заключается в воздействии на подсистемы управления УП$_g$, $g=\overline{1,n}$, заставляющем действовать их согласованно, подчиняя действия УП единой политике, ориентированной на достижение глобальной цели системы, несмотря на то, что эта цель может противоречить локальным целям подсистем. Координацию осуществляет УП$_0$, причем именно координатор должен преодолеть противоречия между локальными целями подсистем УП$_g$, $g=\overline{1,n}$.

Успешность деятельности координатора по организации согласованных действий УП$_g$, $g=\overline{1,n}$, оценивается тем, насколько успешно достигается глобальная цель управления ИТ-инфраструктурой. Достижение цели координатором можно рассматривать как решение задачи, которая формализуется как задача принятия решения и заключается в оценке результативности координации. Поскольку эта задача определяется относительно всех подсистем управления, включая протекающий в ИТ-инфраструктуре процесс $P$, то она называется глобальной решаемой задачей [3, 5].

Для двухуровневых систем должны быть обеспечены координируемость по отношению к задаче, решаемой УП$_0$, и координируемость по отношению к глобальной задаче [5]. Первое означает, что сигналы УП$_0$ оказывают координирующее воздействие на задачи, решаемые УП$_g$, $g=\overline{1,n}$, а второе, что координатор способен влиять на УП$_g$, $g=\overline{1,n}$, так, что их совместное воздействие на процесс $P$ направлено на решение глобальной задачи.

Успешное функционирование СУИ, отвечающей двухуровневой модели, может быть

обеспечено только тогда, когда цели подсистем согласованы между собой и согласованы с глобальной целью системы [3, 5]. В двухуровневой системе выделяют три типа целей (задач): глобальную цель, цель координатора $УП_0$ и цели управляющих подсистем $УП_g$, $g=\overline{1,n}$, а необходимость совместимости целей или задач вытекает из следующих особенностей. На процесс $P$ непосредственно воздействуют только $УП_g$, $g=\overline{1,n}$, поэтому глобальная цель может быть достигнута только опосредованно через действия $УП_g$, $g=\overline{1,n}$, которые должны быть скоординированы относительно глобальной цели, а также цели координатора. Глобальная цель — повышение эффективности выполнения бизнес-процессов, выходит за рамки непосредственной деятельности двухуровневой системы, приведенной на рис. 1, и ни одна из подсистем $УП_g$, $g=\overline{0,n}$, не ориентирована на достижение глобальной цели или решение глобальной задачи. Глобальная задача может быть решена только совместными воздействиями всех управляющих подсистем $УП_g$, $g=\overline{0,n}$.

*Глобальная цель*. Учитывая тот факт, что ИТ-инфраструктуры создаются для повышения эффективности производственных и бизнес-процессов, глобальную цель СУИ можно определить как обеспечение максимального качества $Q$ ИТ-сервисов с минимальными затратами $C$. Так, целью процессного управления в соответствии с ITSM и ISO является постоянное повышение уровня ИТ-услуг [6, 7, 8], что формально можно записать как $\max Q$.

Максимальное качество предоставления услуг ИТ-инфраструктурой будет достигаться в том случае, когда

$$\max Q \Leftrightarrow \max Q_i, \forall i=\overline{1,K} \Leftrightarrow \max q_{ki}, \quad \forall i=\overline{1,K}, \forall k=\overline{1,M_i}, \tag{7}$$

где $Q_i, i=\overline{1,K}$ — качество $i$-й услуги; $q_{ki}, k=\overline{1,M_i}$ — значение $k$-го показателя качества $i$-й услуги.

Для достижения цели процессного управления необходимо непрерывно наращивать ресурсы ИТ-инфраструктуры, что неприемлемо прежде всего с экономической точки зрения. С другой стороны, повышение экономической эффективности ведения бизнеса требует сокращения затрат на ИТ-инфраструктуру, т. е. действий, нацеленных на достижение $\min C$. Поддержание качества услуг на этом уровне является основной задачей координатора.

*Цель координатора*. Целью координатора является поддержание качества $Q$ услуг на согласованном уровне с минимальными затратами $C$ на задействуемые ресурсы. Цель координатора можно формализовать следующим образом

$$Q=\text{const}\big|_{\min C}. \tag{8}$$

Выражение (8) означает, что координатор из всех возможных вмешательств будет выбирать такие, которые требуют минимальной стоимости реализации.

Требование поддержания согласованного уровня услуг касается всех услуг и отдельных показателей качества услуг:

$$Q=\text{const} \Leftrightarrow Q_i=\text{const}, \forall i=\overline{1,K} \Leftrightarrow q_{ki}= =\text{const}, \forall k=\overline{1,M_i}, \forall i=\overline{1,K}. \tag{9}$$

Здесь необходимо оговорить следующее обстоятельство. Основным способом повышения качества $i$-й услуги, $i=\overline{1,K}$, является выделение дополнительных ресурсов приложениям, поддерживающим работу $i$-й услуги. При превышении уровнем $i$-й услуги целевого значения производится сокращение ресурсов, выделенных соответствующим приложениям, как этого требует критерий $\min C$. В то же время, последний сервер, предоставляющий $i$-ю услугу, не может быть отключен, несмотря на то, что качество этой услуги по-прежнему выше требуемого, поскольку это приведет к полному прекращению предоставления услуги. Таким образом, всегда будет какой-то фиксированный минимум затрат $C$, после чего дальнейшее сокращение затрат будет невозможно.

*Локальные цели*. Целью локального управления является поддержание заданных значений параметров функционирования ИТ-инфраструктуры с минимальными затратами, т.е.

$$q_{ki}=\text{const}\big|_{\min C}, \forall k=\overline{1,M_i}, \forall i=\overline{1,K}. \tag{10}$$

В модели СУИ, приведенной на рис. 1, управляющие подсистемы $УП_g$, $g=\overline{1,n}$, могут иметь собственные различающиеся цели функционирования.

## Определение принципа координации и синтез координатора

Выполнение требований координируемости и совместимости выступает ограничением при определении стратегий, которыми может руко-

водствоваться координатор. Предложенные в [3, 5] принципы координации, основанные на постулате совместимости, не могут быть использованы в СУИ, поскольку они предполагают либо получение и использование точного прогноза значений параметров процесса $P$, либо требуют знания вида функций или аналитических выражений для решения задачи координации. Для решения проблемы координации необходимо после декомпозиции глобальной задачи произвести синтез координатора и определиться с методами, процедурами или алгоритмами координации.

Перепишем цель (9) координатора в следующем виде:

$$\min \Delta Q_i = \min(Q_i - Q_i^*), \forall i = \overline{1,K} \Leftrightarrow$$
$$\Leftrightarrow \min \Delta q_{ki} = \min(q_{ki} - q_{ki}^*), \quad (10)$$
$$\forall k = \overline{1,M_i}, \forall i = \overline{1,K},$$

где $Q_i$ и $Q_i^*$ — целевое и фактическое значение качества $i$-й услуги; $q_{ki}$ и $q_{ki}^*$ — целевое и фактическое значение показателя качества $i$-й услуги, причем фактическое качество считаем хуже требуемого при $Q_i > Q_i^*$ и, соответственно, при $q_{ki} > q_{ki}^*$.

На входы процесса $P$ поступают управляющие и возмущающие воздействия, а задача СУИ сводится к выбору управления, противодействующего возмущению. Исходя из (11), координатор должен сравнивать текущие значения показателей качества $q_{ki}^*$, $k = \overline{1,M_i}$, $i = \overline{1,K}$, услуг, предоставляемых процессом $P$ пользователям, с целевыми значениями $q_{ki}$, $k = \overline{1,M_i}$, $i = \overline{1,K}$, и вырабатывать координирующие сигналы, минимизирующие отклонение. При управлении, направленном на поддержание согласованного уровня, естественным является использование принципа управления по отклонению [9]. В этом случае выходы процесса $P$ после соответствующих преобразований, заключающихся в сведении метрик для определения значений $q_{ki}, k = \overline{1,M_i}, i = \overline{1,K}$, по цепи обратной связи поступают в координатор, где сравниваются с целевыми значениями. На основании отклонения $\Delta q_{ki}, k = \overline{1,M_i}, i = \overline{1,K}$, вырабатываются координирующие сигналы для УП$_g$, $g = \overline{1,n}$.

В СУИ могут быть измерены основные возмущающие воздействия, к которым относится количество $\hat{a} = \{a_l, l = \overline{1,I}\}$ пользователей услуг, влияние неисправностей на качество сервиса, загруженность каналов связи и др. Это позволяет совместно с управлением по отклонению использовать принцип управления по возмущению и реализовать в СУИ комбинированное управление, при этом больший вес имеет управление по отклонению. На рис. 2 приведен результат декомпозиции координатора, реализующего комбинированный принцип управления уровнем услуг.

Координатор на рис. 2 содержит контур отрицательной обратной связи и цепи для компенсации возмущающих воздействий $\xi \in \Xi$. Компенсационный контур оценивает основные возмущения, которые учитываются при выборе корректирующих сигналов.

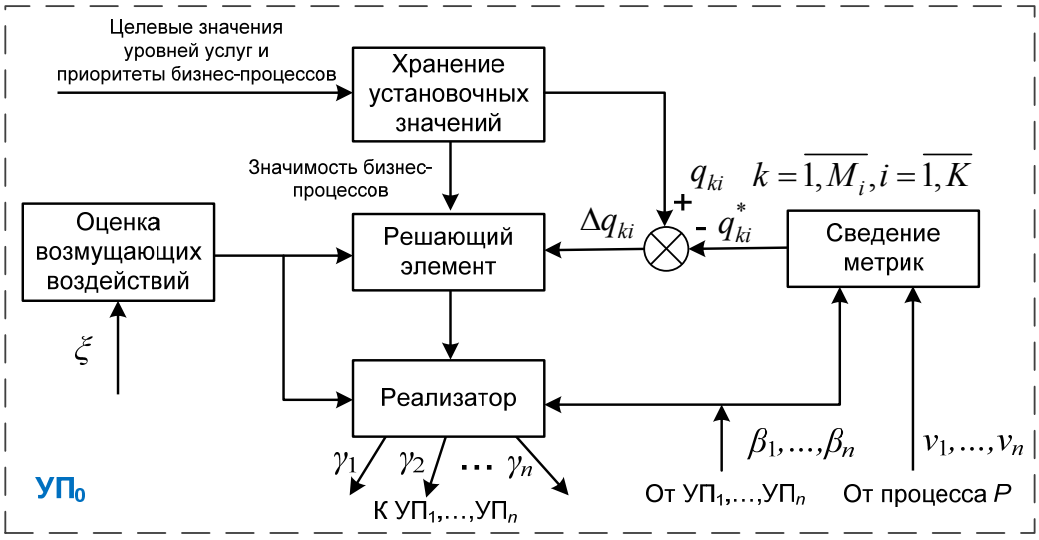



***Рис. 2. Структура координатора, использующего обратную связь и учитывающего возмущающие воздействия***

Глобальная цель управления ИТ-инфраструктурой может изменяться, что проявляется для координатора изменением приоритетов $Pr$ приложений $\{A_l, l=\overline{1,I}\}$ и целевых значений $Q_i, i=\overline{1,K}$ уровня услуг. В этом случае выражение (5) примет вид:

$$C_0 : Pr \times Q_i \times \mathrm{B} \times V \times \Xi \to \Gamma, i=\overline{1,K}. \quad (12)$$

После определения принципа управления необходимо определить стратегию и правила координатора, а также условия применения стратегии или правил.

Выбор координирующего воздействия определяет система

$$< Pr, \Delta\hat{q}, \mathrm{B}, V, \Xi, \mathcal{N} >, \quad (13)$$

где $\Delta\hat{q}$ — вектор отклонения; $\mathcal{N}$ — ситуационная неопределенность.

Проанализируем систему (13). Приоритеты $Pr$ приложениям задаются подсистемой управления производительностью выполнения бизнеса (BPM) и принимают значения из множества $\{1, 2, ..., Pr_m\}$, где $Pr_m$ — максимальное значение приоритета. В процессе функционирования ИТ-инфраструктуры приоритеты приложений $Pr$ изменяются, отслеживая изменения значимости бизнес-процессов.

Для характеристики степени отклонения значений элементов вектора $\Delta\hat{q} = (\Delta q_{1,1}, ..., \Delta q_{k,i}, ..., \Delta q_{M_K,K})$ от целевых значений по аналогии с [10] введем функцию $w(\Delta q_{ki})$, $\forall k = \overline{1,M_i}$, $\forall i = \overline{1,K}$, принимающую значения на отрезке [-1,1] и определяющую степень близости фактического уровня качества к целевым значениям

$$w(\Delta q_{k,i}) = (q_{ki} - q_{ki}^{*}) / q_{\mathrm{кр}_{ki}}, \quad (14)$$

где $q_{\mathrm{кр}_{k,i}}$ — критическое значение показателя качества *i*-й услуги, при котором качество считается неудовлетворительным. Причем при $w(\Delta q_{k,i}) \in (0,1]$ фактическое значение показателя качества *i*-й услуги лучше, чем требуемое, при $w(\Delta q_{k,i}) \in [-1,0)$ качество услуги хуже согласованного.

Значения показателей качества $q_{ki}$, $\forall k = \overline{1,M_i}, i = \overline{1,K}$ от объема выделенных ресурсов *r* в общем виде можно представить следующей зависимостью:

$$q = f_{qr}(r), \quad (15)$$

где $q$ — показатель качества услуги. Для увеличения значения показателя качества $q$ соответствующему приложению из множества $\{A_l\}$ необходимо выделить дополнительные ресурсы. Тогда новое значение $q'$ показателя качества услуг

$$q' = f_{qr}(r + \Delta r). \quad (16)$$

Если $\Delta r > 0$, то $q' \geq q$, что позволяет говорить о монотонном характере функции $f_{qr}$.

Аналогичным образом можно утверждать, что функция

$$q = f_{qa}(\hat{a}) \quad (17)$$

также является монотонной.

Если функции (14) и (16) монотонные, то функция

$$q = f_q(r, \hat{a}) \quad (18)$$

также будет монотонной [1]. Здесь вектор $\hat{a} = \{a_l, l = \overline{1,I}\}$ определяет текущее значение количества запросов к приложениям $\{A_l, l = \overline{1,I}\}$.

Из всех видов неопределенностей [11] наиболее характерной для ИТ-инфраструктуры являются ситуационная неопределенность $\mathcal{N}$, характеризуемая непредвиденной активностью и действиями пользователей, непрогнозируемостью нештатных ситуаций, трудностью определения реакции ресурсов на комбинацию воздействующих факторов. При этом возникает задача выработки координирующих $\gamma = (\gamma_1, ..., \gamma_n)$ и управляющих $u = (u_1, ..., u_n)$ воздействий по сигналам обратной связи $v = (v_1, ..., v_n)$ и $\beta = (\beta_1, ..., \beta_n)$ при воздействии возмущений $\xi \in \Xi$ в условии неопределенности $\mathcal{N}$. Поскольку в аналитическом виде определить соответствующее отображение не представляется возможным, то выходом из ситуации является использование итеративной процедуры координации [3, 5], предполагающей участие всех процессов и подсистем, реализующих управление уровнем услуг, приведенных на схемах на рис. 1 и рис. 2. Применение итеративной процедуры позволяет выработать приемлемые координирующие воздействия ввиду монотонности функций (15)—(18), а также монотонности влияния ситуационной неопределенности $\mathcal{N}$ на уровень услуг. Таким образом, предлагается раскрытие неопределенности осуществлять использованием итеративных процедур управления.

Основная функция координатора заключается в согласовании деятельности УП$_g$, $g=\overline{1,n}$, при генерации ими собственных решений так, чтобы повысить суммарный эффект от их совместных действий. Поэтому решения, принимаемые координатором, оказывают влияние на выбор координирующих, а не управляющих воздействий [5]. Для выбора координирующих воздействий необходимо определить принцип координации. В СУИ координирующие воздействия указывают на то, какому из УП$_g$, $g=\overline{1,n}$, отдается предпочтение при восстановлении качества услуг и какие методы целесообразно использовать (рис. 3). Например, при перегрузке каналов связи выделение дополнительных вычислительных ресурсов приложениям из множества $\{A_l\}$ не восстановит уровень услуг. Поэтому координатор УП$_0$ сообщает УП$_g$, отвечающей за управление потоками в сети, на необходимость ограничения исходящего трафика приложений из $\{A_l\}$, имеющих низший приоритет. Такую координацию можно реализовать, например, путем использования основанного на системе продукций принципа координации. Правила в нотации Бэкуса-Наура [12] имеют следующий вид:

$$
\begin{aligned}
&<\textit{система\_правил}>::=[<\textit{продукция}>]\\
&<\textit{продукция}>::=<\textit{условие}>\rightarrow<\textit{следствие}>\\
&<\textit{условие}>::=[<\textit{простое\_условие}>]\\
&<\textit{простое\_условие}>::=<\textit{объект}><\textit{атрибут}><\textit{предикат}><\textit{значение}>\\
&<\textit{объект}>::=\textit{УП}_g,\ i=\overline{1,n}\\
&<\textit{следствие}>::=[<\textit{указание}>|<\textit{формула}>|<\textit{программа}>]\\
&<\textit{предикат}>::=|\neq|<|>|\leq|\geq\\
&<\textit{атрибут}>::=<\textit{приложение}>|<\textit{приоритет\_приложения}>|<\textit{услуга}>|\\
&|<\textit{бизнес\_процесс}>|<\textit{важность\_процесса}>|<\textit{параметр}>|<\textit{статус\_ИТС}>.
\end{aligned}
\qquad (19)
$$

Применение основанного на продукциях принципа координации оправдано в случаях, когда ставится цель улучшения качества предоставления услуг, а не достижения оптимальных показателей функционирования ИТ-инфраструктуры, в условиях недостаточности информации о факторах, влияющих на результаты координирующих и управляющих воздействий.

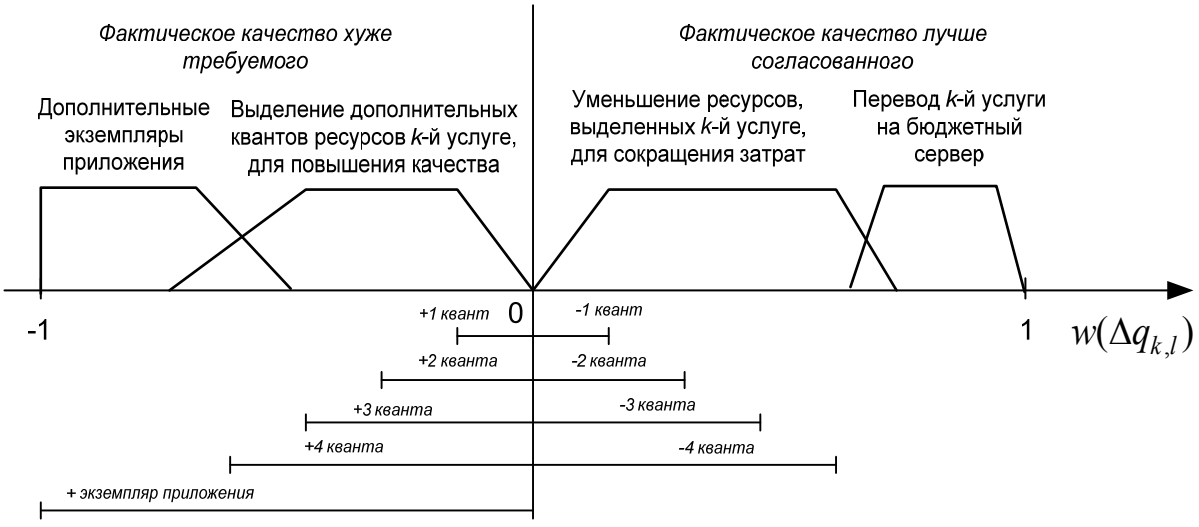



***Рис. 3. Пример выбора действий УПg, $g=\overline{1,n}$, в зависимости от значения функции $w(\Delta q_{ki})$***

Отображение (5) может иметь очень сложный вид, а для системы (13) результирующую взаимосвязь между координирующими $\gamma=(\gamma_1,...,\gamma_n)$, управляющими $u=(u_1,...,u_n)$ воздействиями и выходом процесса $P$ получить аналитически невозможно. В этом случае можно использовать программное управление [13].

Основными процедурами координации являются использование итеративных процедур по улучшению координирующих сигналов на основании анализа результатов координации либо использование обратной связи для коррекции координирующего сигнала [5]. Целесообразно применять комбинацию этих типов

процедур. В обоих случаях для определения сигнала ошибки при оценке воздействия необходимо произвести сведение метрик, измеряемых на уровне процесса $P$, к метрикам, которыми оперирует координатор [14, 15].

Необходимо отметить, что координатор используется при автоматическом режиме управления уровнем услуг, а при автоматизированном управлении роль координатора выполняет администратор уровня услуг, использующий СУИ в качестве системы поддержки принятия решений.

## Вывод

Доказана целесообразность представления модели СУИ при управлении уровнем услуг в виде двухуровневой системы управления с координатором. Комбинированный принцип управления на основе обратной связи с учетом возмущающего воздействия и основанный на правилах принцип координации позволяют поддерживать согласованный уровень услуг в автоматическом режиме работы СУИ.

## Список литературы


1. Ролик А.И. Декомпозиционно-компенсационный подход к управлению уровнем услуг в корпоративных ИТ-инфраструктурах / А.И. Ролик // Вісник НТУУ «КПІ». Інформатика, управління та обчислювальна техніка: Зб. наук. пр. – К.: Век+, – 2013. – № 58. – С. 78–88.
2. Ролик А.И. Система управления корпоративной информационно-телекоммуникационной инфраструктурой на основе агентского подхода / А.И. Ролик, А.В. Волошин, Д.А. Галушко, П.Ф. Можаровский, А.А. Покотило // Вісник НТУУ «КПІ»: Інформатика, управління та обчислювальна техніка. – К.: «ВЕК+», 2010. – № 52. – С. 39–52.
3. Алиев Р.А. Методы и алгоритмы координации в промышленных системах управления / Р.А. Алиев, М.И. Либерзон. – М.: Радио и связь, 1987. – 208 с.
4. Методы робастного, нейро-нечеткого и адаптивного управления: Учебник / Под ред. Н.Д. Егупова. – М.: Изд-во МГТУ им. Н.Э. Баумана, 2001. – 744 с.
5. Месарович М. Теория иерархических многоуровневых систем: пер. с англ. / М. Месарович, Д. Мако, И. Такахара; пер. с англ. под ред. И. Ф. Шахнова. Предисл. чл.-кор. АН СССР Г.С. Поспелова. – М.: «Мир», 1973. – 344 с.
6. Information technology. Service management. Part 1: Specification: ISO/IEC 20000-1:2005. – ISO/IEC, 2005. – 16 p.
7. Information technology. Service management. Part 2: Code of practice: ISO/IEC 20000-1:2005. – ISO/IEC, 2005. – 34 p.
8. Ингланд Р. Введение в реальный ITSM: пер. с англ. / Р. Ингланд. – М.: Лайвбук, 2010. – 132 с.
9. Бесекерский В.А. Теория систем автоматического регулирования, издание третье, исправленное/ В.А. Бесекерский, Е.П. Попов. – М., «Наука». – 1975. – 768 с.
10. Згуровський М.З. Стратегія технологічного передбачення в інноваційній діяльності / М.З. Згуровський, Н.Д. Панкратова// Науково-технічна інформація.– 2006. – 2 (28). – С. 3–10.
11. Згуровский М.З. Системный анализ: проблемы, методология, приложения / М.З. Згуровский, Н.Д. Панкратова. – Киев: Наук. думка. – 2011. – 728 с.
12. Naur P. Revised report on the algorithmic language ALGOL 60// P. Naur // Communications of the ACM 6. – 1 Jan. 1963. – P. 1–17.
13. Моисеев Н.Н. Численные методы в теории оптимальных систем/ Н.Н. Моисеев. – М.: Наука, 1971. – 424 с.
14. Ролік О.І. Метод зведення метрик якості функціонування компонентів ІТ-інфраструктури за допомогою апарату непараметричної статистики / О.І. Ролік, П.Ф. Можаровский, В.М. Вовк, Д.С. Захаров // Вісник НТУУ «КПІ»: Інформатика, управління та обчислювальна техніка. – К.: «ВЕК+», 2011. – № 53. – С. 160–169.
15. Теленик С.Ф. Зведення метрик оцінювання рівня обслуговування користувачів на основі експертних оцінок / С.Ф. Теленик, О.І. Ролік, О.М. Моргаль, О.С. Квітко// Вісник Вінницького політехнічного інституту. – 2011. – №1. – С. 112–123.

УДК 004.67

*МАЛЮКОВ П.Н.,*
*РАДЕР Р.И.*

# СИСТЕМА ОБНАРУЖЕНИЯ АНОМАЛЬНОГО ПОВЕДЕНИЯ АБОНЕНТОВ ТЕЛЕФОННОЙ СЕТИ

Предложен алгоритм и система для обнаружения аномального поведения абонентов с учётом групповых изменений в поведении. Написан прототип и имитатор поведения пользователей для анализа алгоритма. Проанализирована работа алгоритма в трех режимах: однопользовательском, многопользовательском без учета тренда и многопользовательском с учетом тренда.

Algorithm and anomaly behavior of end users detection system is proposed. Program for detecting anomalous behavior and user behavior simulator for analyzing algorithm is developed. Algorithm is analyzed in 3 modes: single user, multi user without considering trend, multi user with trend considering.

## 1. Введение

Операторы связи по всему миру испытывают значительные потери из-за мошенников. По данным исследования CFCA (международная организация для контроля мошенничества, обеспечения доходов и предотвращения потерь) в 2013 году, потери составляют 46.3 миллиардов долларов в год, что больше на 15% по сравнению с аналогичным исследованием в 2011 году. С увеличением потерь, 8% компаний переложило функции по борьбе с мошенничеством из финансовых отделов в отделы IT и безопасности (сейчас 38% от всех компаний) [1].

Игнорировать проблему мошенничества невозможно, так же как и прекратить, поэтому цель – обнаружить вредоносное вмешательство в каналы и средства связи, неправомерное использование услуг как можно раньше и предотвратить его.

## 2. Анализ проблемы

Системы мониторинга трафика распознают аномальный трафик, используя анализ и построение шаблона поведения отдельного пользователя [2]. Такие системы обычно являются частью системы безопасности и не являются самодостаточными — при обнаружении интервенции, в зависимости от политики безопасности, абонент сразу блокируется или подаётся сигнал сотруднику оператора для ручной обработки.

Проблемой такого подхода являются периодические и единоразовые массовые изменения поведения абонентов системы. Примером могут служить праздники, как Новый год, социально-политические события и многое другое. При этом такие события по-разному влияют на пользователей с разными шаблонами поведения, например корпоративных пользователей редко затронут семейные праздники, а такие дни как «черная пятница» повлияют на количество исходящих вызовов корпоративного сегмента, но не частных пользователей.

Учёт этих факторов позволит уменьшить процент ложных срабатываний системы, соответственно уменьшить количество ручной работы или недовольных абонентов из-за автоматической блокировки, что позволит уменьшить затраты на поддержание системы безопасности.

Целью работы является разработка системы мониторинга журналов CDR (Call Detail Record) для обнаружения аномального поведения абонентов с учётом массовых изменений шаблонов использования системы.

Как показало исследование телефонного оператора AT&T, достаточно эффективным способом составления шаблона поведения пользователей является построение вектора 24x7 элементов с количеством звонков за каждый час каждого дня недели [3].

## 3. Предлагаемый метод

Метод заключается в построении шаблона поведения пользователя на основе нескольких недель наблюдения, и впоследствии обнаружения звонков, которые выходят за рамки текущего шаблона. Соответственно, работа алгоритма делится на 2 этапа: режим обучения и рабочий режим.

Учитывая случайную природу совершения телефонного звонка по отношению к оператору, для анализа поведения можно исходить из предположения, что число звонков за определённый промежуток времени будет распределено по распределению Пуассона.

Также учтём, что поведение пользователя зависит от дня недели. Тогда запись поведения

пользователя можно определить как вектор длиной L=H*7, где H – число разбиений суток, а 7 – количество дней в недели. Для уменьшения случайной составляющей, составляется несколько записей поведения по неделям. Количество хранимых записей W влияет на точность конечного шаблона поведения P

$$P = \left(\overline{\lambda_1}, \overline{\lambda_2}, \ldots, \overline{\lambda_L}\right) \quad (1)$$

который можно определить как усреднённый вектор записей поведения за предыдущие W недель, где среднее значение считается как экспоненциально взвешенное скользящее среднее с окном в количество записей:

$$EMA_n = (1-\alpha)x_n + \alpha EMA_{n-1} \quad (2)$$

что можно привести к не рекурсивной формуле [5]:

$$EMA = \frac{x_n + \alpha x_{n-1} + \alpha^2 x_{n-2} + \ldots + \alpha^{n-1} x_1}{1 + \alpha + \alpha^2 + \ldots + \alpha^{n-1}} \quad (3)$$

Это позволяет уменьшить запаздывание, придавая большее значение последним значениям.

$$\overline{\lambda_i} = \frac{\lambda_{iw} + \alpha\lambda_{i(w-1)} + \ldots + \alpha^{n-1}\lambda_{i1}}{1 + \alpha + \alpha^2 + \ldots + \alpha^{w-1}} \quad (4)$$

В режиме обучения система принимает записи о звонках, измеряет частоту звонков и фиксирует её в записях поведения. Когда нужное количество записей сохранено (в зависимости от выбранного критерия, или достигается заданная дисперсия, или задается необходимое количество записей), система переводится в рабочий режим для конкретного абонента. То есть в один момент времени часть абонентов может обрабатываться в режиме обучения, а часть — в рабочем режиме. Это необходимо, так как во время работы системы могут подключиться новые абоненты.

Подсчёт текущей частоты $f_H$ делается на основе времени инициации последних K звонков для каждого абонента отдельно. Имея вектор $t_i^{phone}, i = \overline{1..K}$ отметок времени инициации звонков (в секундах) рассчитываем предполагаемую частоту за определённый промежуток времени T:

$$T = \frac{24 \cdot 60 \cdot 60}{H} \quad (5)$$

Предполагаемая частота за время одной ячейки шаблона:

$$f_H = T \cdot f \quad (6)$$

где $f$ – текущая частота за секунду

$$f = \frac{1}{T_{avg}} \quad (7)$$

$T_{avg}$ - среднее время между звонками. Для уменьшения эффекта запаздывания за изменением частоты, здесь также целесообразно использовать экспоненциально взвешенное среднее значение:

$$T_{avg} = EMA_n(t_n - t_{n-1}, t_{n-1} - t_{n-2}, \ldots, t_2 - t_1) \quad (8)$$

Тогда:

$$f_H = \frac{24 \cdot 60 \cdot 60}{H * EMA(t_n - t_{n-1}, \ldots, t_2 - t_1)} \quad (9)$$

В рабочем режиме система продолжает фиксировать записи поведения, то есть обучение не останавливается. В этом режиме начинает работать алгоритм кластеризации, который классифицирует шаблоны пользователей на k кластеров. Количество классов может быть варьировано с измерением для обеспечения точного разделения абонентов по характеру использования системы. Для кластеризации можно использовать алгоритм k-means, запускаемый периодически после записи очередной записи поведения (раз в неделю), но в виду его сложности для большого количества абонентов целесообразно использовать его потоковую модификацию [4], что позволит классифицировать шаблоны сразу после получения новых данных.

Помимо этого включается проверка каждого звонка на соответствие шаблону поведения. Проверка на соответствие может осуществляться как с использованием доверительных интервалов, так и проверкой с учетом дисперсии. Пусть предполагаемый доверительный интервал $(f_{min}, f_{max})$ с необходимой надёжностью, задаваемой оператором, а отклонение от предполагаемого значения для интенсивности рассматриваемого временного промежутка $\lambda_i$ из шаблона поведения P:

$$d = \frac{f_H^i - \lambda_i}{f_{max} - f_{min}} \quad (10)$$

где i – номер ячейки шаблона. Тогда при $-1 \le d \le 1$ значение текущей частоты находится в пределах ожидаемой.

Для установившегося состояния системы, среднее значение отклонений:

$$trend_C = \frac{\sum d_i^C}{N^C} \quad (11)$$

где i – номера абонентов класса C, будет около нуля. Если же будет происходить сезонное изменение или некоторый иной фактор, который влияет на характер использования си-

стемы класса или нескольких классов пользователей, тренд покажет характер этих изменений.

Смыслом функции тренда является процент отклонения класса абонентов от предыдущего характера использования системы. Поэтому, для уменьшения ложных срабатываний системы обнаружения аномального поведения, необходимо расширить доверительный интервал на вычисленный тренд.

Таким образом, интервенция может быть обнаружена сравнением текущей частоты с границами доверительного интервала, который равен:

$$\begin{cases} trend > 0, (f_{min}, f_{max} + trend) \\ trend < 0, (f_{min} + trend, f_{max}) \end{cases} \quad (12)$$

Собирая все части воедино, получаем алгоритм:

```
1. for каждая новой записи CDR:
2.  if (шаблон абонента в рабочем режиме) и (CDR не соответствует шаблону) then
3.   подать сигнал об интервенции
4.  end if
5.  модифицировать запись поведения
6.  пересчитать текущую частоту
7.  пересчитать тренд
8.  if прошло время Tcluster then
9.   инициировать задачу кластеризации абонентов
10.  end if
11. end for
```

Где Tcluster – время с последней кластеризации.

Последний пункт может быть модифицирован для потоковой кластеризации [5].

### 4. Модель для проверки метода

Для проверки метода были написаны система обнаружения аномального поведения пользователей, а также имитатор поведения пользователей.

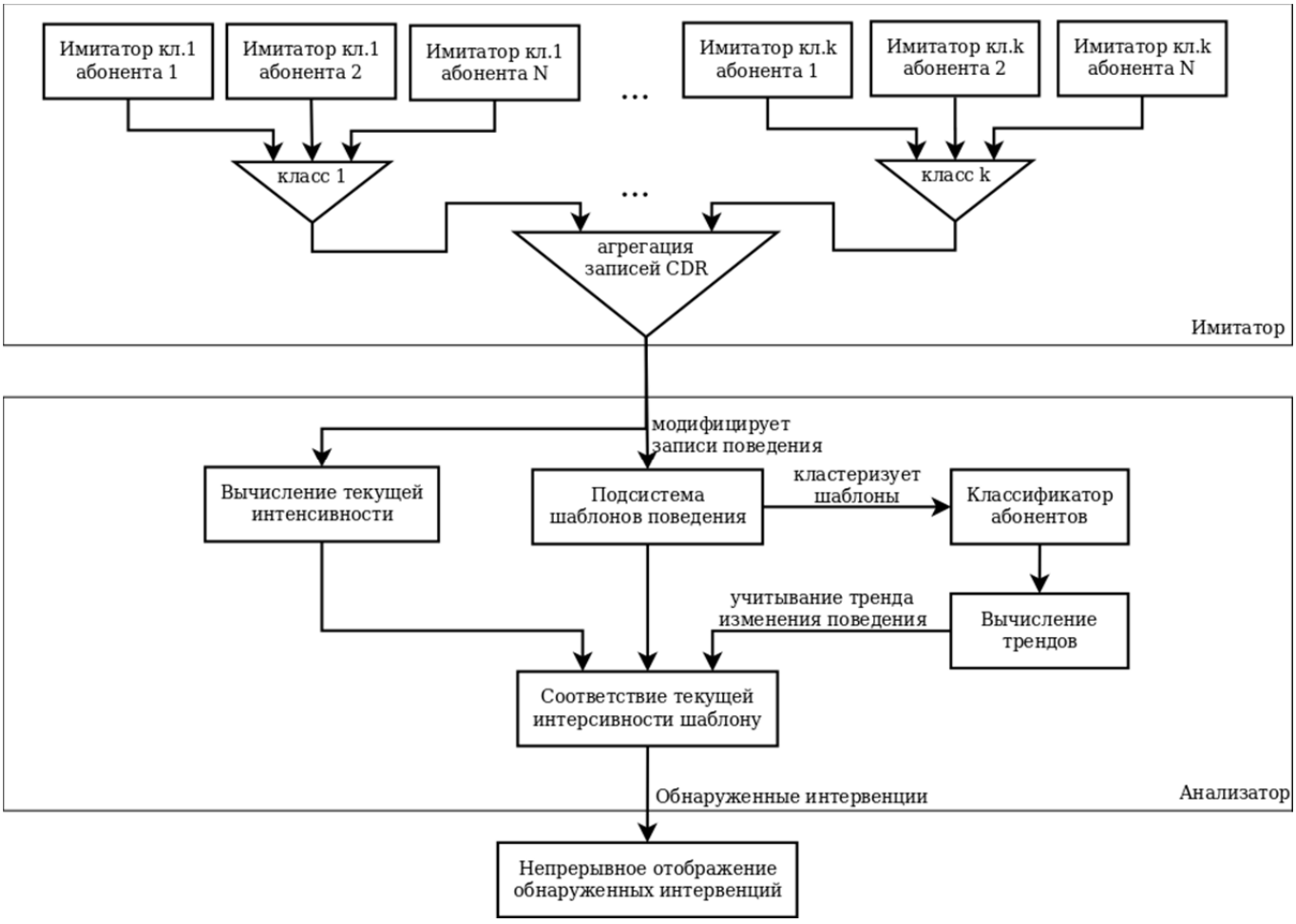



***Рис. 1: Компоненты системы***

Имитатор генерирует телефонные звонки, промежутки между которыми распределены по распределению Пуассона, с изменяемой интенсивностью по дням недели и временем суток.

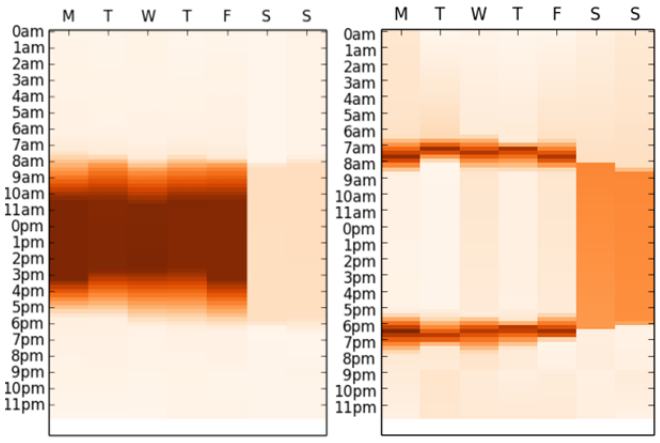



***Рис. 2: Два класса имитируемых абонентов***

Визуализация шаблонов двух классов пользователей на ***Рис. 2***. Первый класс – абоненты, производящие звонки в начале и в конце рабочего дня, а также активные в выходные дни. Второй класс – типично корпоративные пользователи, производящие звонки в бизнес-часы. Как видно, вносится случайная составляющая по дням недели, и для каждого абонента она своя.

В имитаторе есть возможность произвести интервенцию одной группы пользователей, либо нескольких, переводя из одного установившегося состояния в другое. Тем самым есть возможность протестировать работу системы обнаружения аномального поведения.

В анализаторе H=24, это значит что каждая ячейка шаблона отображает один час использования, коэффициент EMA $\alpha_1 = \alpha_2 = 0.8$.

Для подсчёта текущей интенсивности звонков для конкретного абонента всегда хранится K=10 последних записей времени инициации звонка, производится по формуле (9).

Подсистема шаблонов поведения каждый час сохраняет текущую частоту в запись поведения.

Классификатор абонентов запускается раз в неделю и запускает алгоритм кластеризации k-means++, основанный на расстоянии Евклида. Вычисление тренда делается на основе формулы (11).

Проверка соответствия текущей интенсивности шаблону происходит используя расширенный доверительный интервал с надёжностью p=0.997, расширенный по формулам (12) по рассчитанному тренду.

Перед началом работы системы, оператор задаёт параметры системы: $\alpha_1$ для вычисления экспоненциально взвешенного скользящего среднего (EMA) для шаблона поведения, H — количество разбиений суток, W — количество недель работы системы в режиме обучения для конкретного абонента, $\alpha_2$ для вычисления EMA текущей частоты, K — ширина окна для вычисления текущей частоты, p - надёжность доверительного интервала, C — количество кластеров абонентов. При внедрении данной системы, последний параметр может быть опущен и добавлен этап подбора параметра для большей точности работы системы.

Проанализирована работа алгоритма в трех случаях:

1) Интервенция в работу только одного абонента (***Рис. 3***)

2) Интервенция в работу одного класса абонентов, работа без учёта тренда (***Рис. 4***)

3) Интервенция в работу одного класса абонентов, работа с учётом тренда (***Рис. 5***)

На верхних графиках показана зависимость количества звонков, максимально допустимое количество звонков и точками обозначены обнаруженные интервенции. На нижних графиках показан тренд.

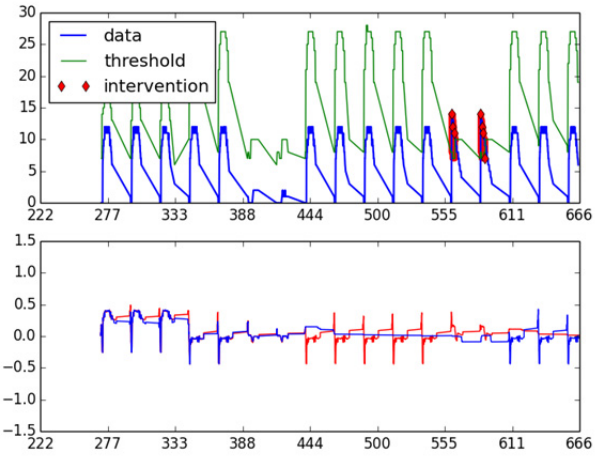



***Рис. 3: Единичная интервенция одного телефонного номера в выходные дни***

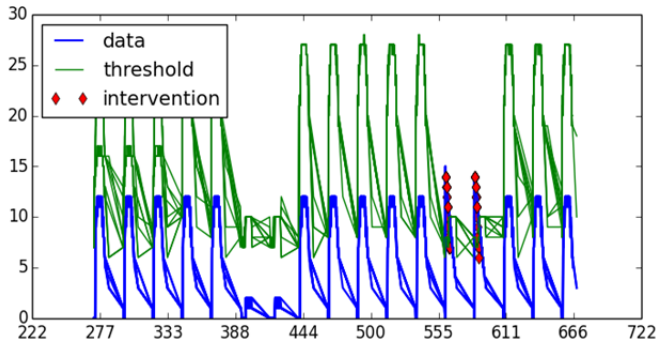



***Рис. 4: Интервенция в работу одного класса абонентов без учета тренда. По оси абсцисс – время в часах (в модели) с начала***

***работы системы, по оси ординат – количество звонков.***

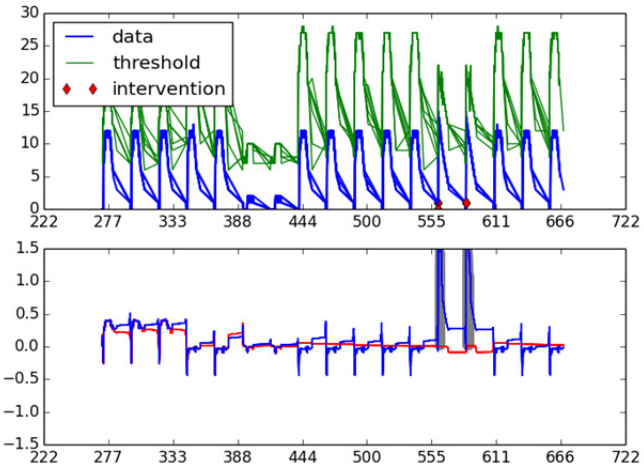



***Рис. 5: Интервенция в работу класса абонентов с учетом тренда***

Как видно, при интервенции в работу одного абонента (***Рис. 3***) тренд не изменяется и интервенция обнаруживается легко (количество подозрительных звонков практически совпадает с фактическим).

При групповом изменении поведения (***Рис. 5***) видно два пика линии тренда, приходящиеся на выходные дни (в данном опыте пользователи, которые никогда не звонили по выходным меняют это поведение). На верхнем графике видно, что уровень интервенций очень низкий, что означает правильную работу алгоритма – групповое изменение поведение не считается подозрительным.

Для проверки, второй опыт был проведён с теми же исходными данными, но без учёта тренда (***Рис. 4***). На графике видно, что уровень интервенций практически совпадает с количеством фактических звонков в рассматриваемом участке, что значит высокую вероятность мошенничества. Таким образом, способ учёта групповых изменений действительно уменьшает количество ложных срабатываний.

## 5. Выводы

Алгоритм потоковый, не требует накопления данных, все хранимые данные можно записывать в циклический список, тем самым потребляя стабильное количество памяти и имея небольшой отпечаток памяти (footprint).

Проанализирована работа алгоритма в многопользовательском режиме с учетом тренда, а также проведено сравнение с однопользовательским режимом и многопользовательским режимом без учета тренда режимах для оценки эффективности метода.

Метод применим к многим алгоритмам и легко модифицируется под конкретные задачи и требования, такие как способ определения интервенции или список отслеживаемых параметров.

## Список литературы

1. Communications Fraud Control Association (CFCA) 2013 Global Fraud Loss Survey [Электронный ресурс] // Communications Fraud Control Association: [сайт]. [2013]. URL: http://www.cfca.org/pdf/survey/Global%20Fraud_Loss_Survey2013.pdf
2. Rosas É.., Analide C. Telecommunications Fraud: Problem Analysis - an Agent-based KDD Perspective [Электронный ресурс] // Fourteenth Portuguese Conference on Artificial Intelligence: [сайт]. [2009]. URL: http://epia2009.web.ua.pt/onlineEdition/402.pdf
3. Becker R.A., Cáceres R., Hanson K., Loh J.M., Urbanek S. Clustering Anonymized Mobile Call Detail Records to Find Usage Groups [Электронный ресурс] // AT&T Researchers — Inventing the Science Behind the Service: [сайт]. URL: http://www.research.att.com/techdocs/TD_100397.pdf
4. Cargal J.M. Discrete Mathematics for Neophytes: Number Theory, Probability, Algorithms, and Other Stuff - Chapter 32 out of 37 // Books in the Mathematical Sciences. 1988. URL: http://www.cargalmathbooks.com/32%20Averages%20.pdf
5. Lecture 6 — Online and streaming algorithms for clustering Sanjoy Dasgupta [Электронный ресурс] // Department of Computer Science and Engineering, University of California, San Diego: [сайт]. URL: http://cseweb.ucsd.edu/~dasgupta/291/lec6.pdf

УДК 004.724.4(045)

*КОГАН А.В.*

# ОРГАНИЗАЦИЯ МНОГОПУТЕВОЙ МЕЖДОМЕННОЙ МАРШРУТИЗАЦИИ С ОПТИМАЛЬНЫМ ЧИСЛОМ ГРАНИЧНЫХ МАРШРУТИЗАТОРОВ

Предложен способ формирования доменов маршрутизации с оптимальным числом граничных маршрутизаторов. Предложен и обоснован способ формирования области минимального сечения графа, которое осуществляется с помощью операций над множеством вершин графа *G* и его подграфов. На междоменном уровне граничные маршрутизаторы объединяются между собой в виртуальную сеть с непересекающимися связями внутри каждой подсети.

Ключевые слова: многопутевая маршрутизация, минимальное сечение графа, граничные маршрутизаторы

Method of forming routing domains with optimal number of edge routers. Proposed and justified method of forming a minimum cross-section area of the graph, which is carried out by the operations on the set of vertices of G and its subgraphs. On cross-domain level boundary routers combined with each other in a virtual network with disjoint links within each subnet.

Keywords: multi-path routing, minimum cross-section of the graph, edge routers

## 1. Введение

Для мобильных сетей большой размерности актуальной является задача формирование оптимальной структуры доменов маршрутизации. В работе [1] показано, что известные методы формирования доменов и выбора месторасположения их контроллеров не достаточно эффективны для мобильных сетей, структура которых постоянно изменяется. В первую очередь это связано с тем, что известные методы не обеспечивают выбора оптимального месторасположения контроллера домена. Это, как правило, приводит к необходимости более частой реконфигурации. С увеличением числа реконфигураций домена, объем служебного трафика в домене увеличивается по нелинейному закону, за счет чего резко падает эффективность задачи маршрутизации [2]. Большинство известных протоколов междоменной маршрутизации строятся на основе протокола BGP [3,4].

В настоящее время с целью повышения эффективности передачи данных, равномерной загрузки сети и повышения уровня безопасности широко используется многопутевая маршрутизация. При рассмотрении вопросов междоменной многопутевой маршрутизации одним из основных вопросов является разбиение сети на домены маршрутизации и определение оптимального числа граничных маршрутизаторов [5].

## 2. Формирования области минимального сечения графа

Следует учитывать, что максимальная эффективность процесса маршрутизации достигается при разбиении сети на домены одинаковой величины [6]. В связи с этим, поиск минимального сечения целесообразно осуществлять ближе к середине графа. С этой целью на начальном этапе формирования области минимального сечения определяется диаметр *D* графа и относительно его середины D/2 формируется область поиска минимального сечения (рис.1).

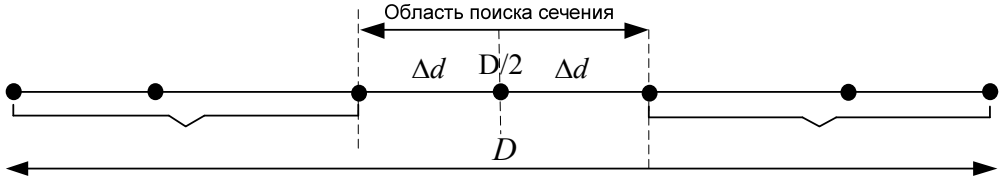



***Рис. 1. Область поиска сечения графа***

На рис. 2 представлен пример разбиения графа на два подграфа *G*1 и *G*2, а также область их пересечения *G*1/2. Вершины S, B2, B6, B10, B14, B17,T образуют диаметр графа. Вершина

В10 является центральной вершиной на диаметре графа.

Формирование минимальной области пересечения осуществляется с помощью операций над множеством вершин графа *G* и его подграфов. Под внутренними вершинами подграфа будем понимать вершины данного подграфа не связанные с внешними вершинами относительно данного подграфа. В свою очередь, вершины подграфа, связанные с внешними вершинами, будем называть граничными вершинами. Например, для подграфа *G*1 внутренними вершинами будут вершины S и В4, а граничными будут вершины В1, В3 и В8. Граничными вершинами подграфа *G*1/2 расположенными на диаметре графа *G* будут вершины В2 и В17, относительно которых будем формировать область минимального сечения.

Рассмотрим работу алгоритма формирования области минимального сечения на примере графа, представленного на рис. 2.

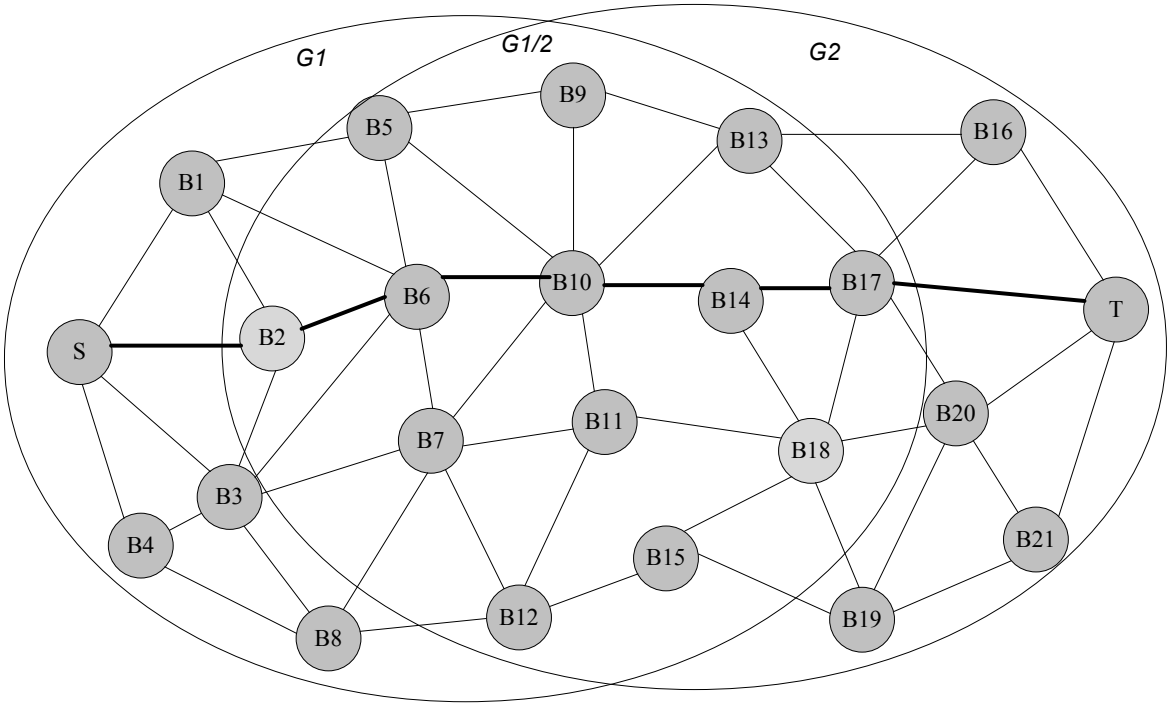



***Рис. 2. Разбиение графа сети на два подграфа***

1. Делим граф *G* сеть на два подграфа *G*1 и *G*2.
2. Выбираем граничные узлы В2 и В17 подграфа *G*1/2, расположенные на диаметре графа *G*.
3. Для вершины В2 множество смежных с ней вершин $B_{C1}$={В6, В3, В1}.
4. Вершины В1, В3 являются внутренними вершинами для подграф $G_S$, и не входят в множество узлов минимального сечения.
5. Для вершины В6 множество смежных с ней вершин равно $B_{C2}$={В1, В5, В10, В7, В3}. В1, В3 не входят в множество узлов минимального сечения.
6. Для вершины В5 множество смежных с ней вершин равно $B_{C3}$={В9, В10, В6}.
7. Для вершины В7 множество смежных с ней вершин равно $B_{C4}$={В6, В10, В14, В11, В12}.
8. Для вершины В12 множество смежных с ней вершин равно $B_{C5}$={В7, В11, В15}.
9. Для вершины В9 множеств о смежных с ней вершин равно $B_{C6}$={В13, В10}.
10. Для вершины В10 множество смежных с ней вершин равно $B_{C7}$={В5, В6, В9, В14, В7}.
11. Для вершины В11 множество смежных с ней вершин равно $B_{C8}$={В7, В12, В18}.
12. Для вершины В15 множество смежных с ней вершин равно $B_{C9}$={В12, В18}.
13. Для вершины В13 множество смежных с ней вершин равно $B_{C10}$={В9}.
14. Для вершины В14 множество смежных с ней вершин равно $B_{C11}$={В10, В7, В18}.

15. Для вершины B18 множество смежных с ней вершин равно $B_{C5}$={B14, B11, B15}.

16. Подмножество сечения $B_{min}$ = {B5, B6, B7, B12} является минимальным сочленением.

Следующим минимальным сечением будет множество вершин $B_{min}$ = {B9, B10, B11, B12} (рис 3). В этом случае граф разбивается практически на два равных подграфа, поэтому данное множество вершин и выбирается в качестве оптимального сечения исходного графа.

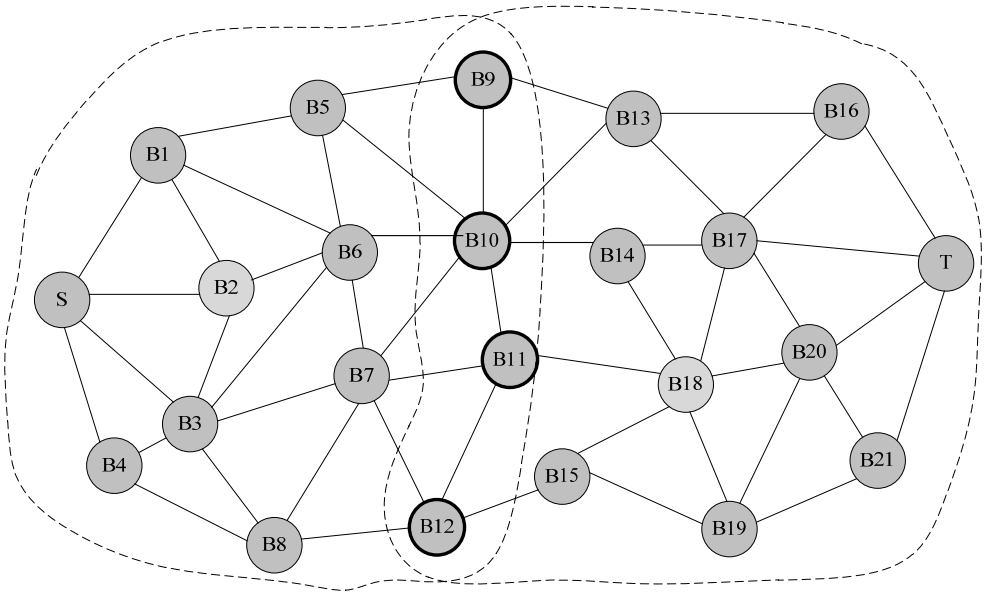



***Рис. 3. Область оптимального сечения графа***

### 3. Формирование структуры междоменной виртуальной сети

Каждой вершине $Bi \in B_{min}$ соотносится граничный маршрутизатор с помощью которых осуществляется междоменная многопутевая маршрутизация.

На рис. 4 представлен пример разбиения сети на 5 подсетей с множеством граничных маршрутизаторов, расположенных в узлах {B5, B6, B7, B8, B9, B14, B15, B16, B19, B25, B28}.

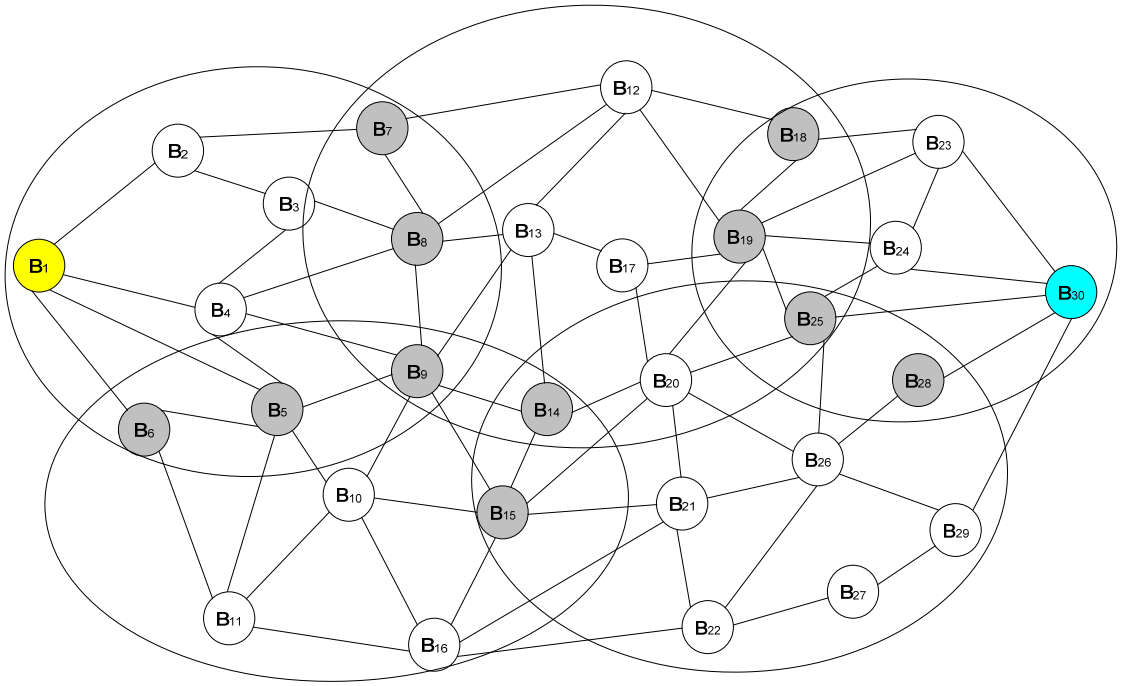



***Рис. 4. Множество граничных маршрутизаторов***

На междоменном уровне граничные маршрутизаторы объединяются между собой в виртуальную сеть с непересекающимися связями внутри каждой подсети (рис. 5).

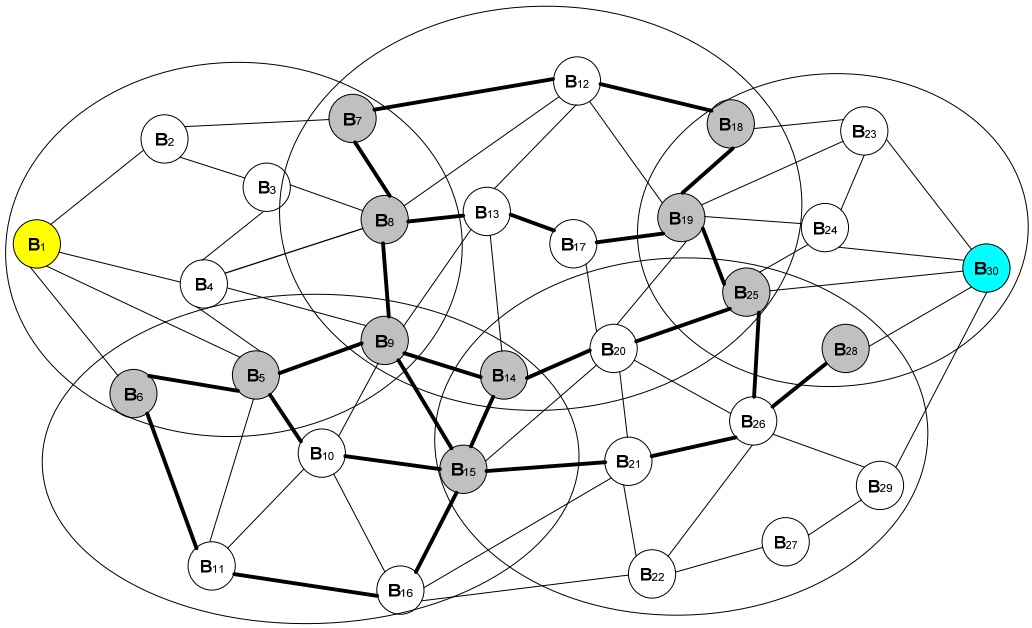



***Рис. 5. Виртуальная междоменная сеть***

В данном случае на междоменном уровне между узлами B1 и B30 можно сформировать 4 виртуальных непересекающихся маршрута, так как степень вершины B1 равна 4, а минимальный разрез на пути из вершины B1 в вершину B30 больше 4.

Для формирования множества непересекающихся маршрутов на междоменном уровне предлагается использовать классический алгоритм Дейкстры. После того как основной путь сформирован, необходимо найти все остальные маршруты. Для этого необходимо удалить все ребра, которые соединены с узлами основного пути, а узлы, которые являются соседними для данного маршрута пометить, как узлы что не могут быть включены в остальные независимые маршруты. Поиск оставшихся путей продолжается в двух сформированных независимых зонах.

В данном случае будут сформированы следующие непересекающиеся маршруты (рис 6):

1. B1>B7>B18>B30;
2. B1>B8>B19>B30;
3. B1>B6>B15>B28>B30.
4. B1>B5>B9>B14>B25>B30.

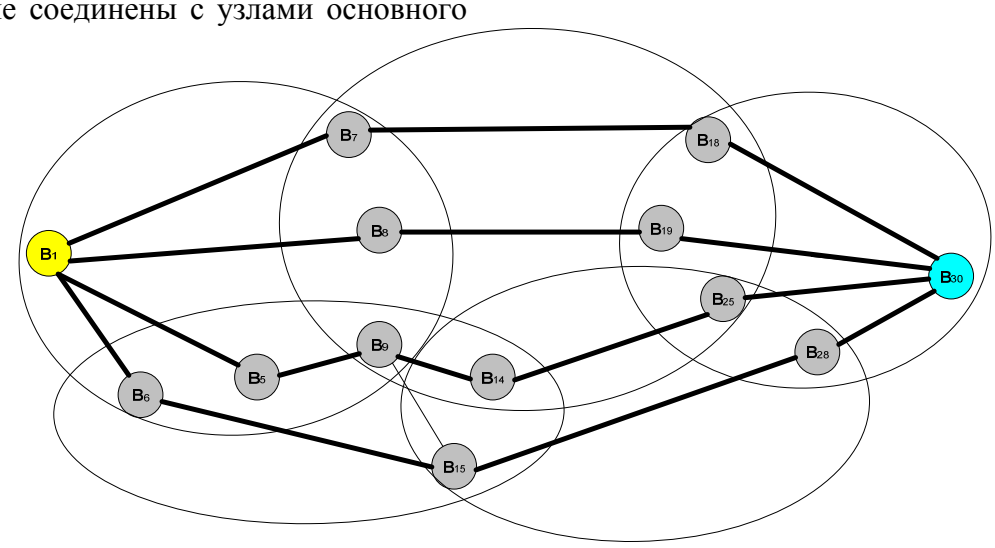



***Рис. 6. Непересекающиеся маршруты***

## 4. Выводы

Многопутевая междоменная маршрутизация позволяет обеспечить равномерную загрузку сети и повысить безопасность передачи информации за счет разбиения сообщения на части и передачи их по нескольким не пересекающимся путям как внутри доменов, так и между доменами. Уменьшение временной сложности алгоритма формирования множества не пересекающихся путей достигается за счет определения минимального сечения внутри доменов и между доменами. Предложенный в работе алгоритм формирования минимального сечения позволяет определить оптимальное число граничных маршрутизаторов, которые объединяются в виртуальную междоменную сеть. Это позволяет свести задачу формирования множества не пересекающихся путей внутри доменов к задаче нахождения непересекающихся путей между граничными маршрутизаторами. Для формирования множества непересекающихся маршрутов на междоменном уровне предлагается использовать модифицированный алгоритм Дейкстры.

## Список литературы


1. Li Y., Kim D., Zou F.,. Du D-ZConstructing Connected Dominating Sets with Bounded Diameters in Wireless Networks //WASA2007 : International Conference on Wireless Algorithms, Systems and Applications, 1-3 Aug. 2007. : thesis rep. – Chicago (USA), 2007. – P. 89–94.
2. Давиденко И.Н. Способ организации динамической структуры мобильной компьютерной сети большой размерности / И.Н. Давиденко, А.В. Левчук // Интеллектуальный анализ информации ИАИ-2009: IX международная научная конференция имени Т.А. Таран, 19-22 мая 2009г. – К.: Просвіта, 2009. – С. 94-103.
3. Juniper, "Configure BGP to Select Multiple BGP Paths, " February 2013. [Online]. Available: http://www.juniper.net/techpubs/software/ junos/junos53/swconfig53-ipv6/html/ipv6-bgp-config29.html.
4. J. Rexford and C. Dovrolis, “Future Internet Architecture: Clean-Slate Versus Evolutionary Research,” Communications of the ACM, vol. 53, no. 9, pp. 36–40, 2010.
5. Кулаков Ю.А. Определение оптимального числа граничных маршрутизаторов
6. при организации безопасной многопутевой маршрутизации в мобильной компьютерной сети большой размерности / Кулаков Ю.А., Коган А.В Вісник НТУУ «КПІ». Інформатика, управління та обчислювальна техніка: Зб. наук. пр. – К.: Век+, – 2013. – № 58. – С. 73-77
7. Лукашенко В.В. Анализ эффективности способов разбиения сети на зоны маршрутизации / В.В. Лукашенко, А.В. Левчук // Електроніка та системи управління. – 2010. – № 1(23). – С. 112-118.

УДК 004 (517.5)

*ПРИЙМАК М.В.,*
*ВАСИЛЕНКО Я.П.,*
*ДМИТРОЦА Л.П.*

# СИГНАЛИ ЗІ ЗМІННИМ ПЕРІОДОМ ТА ЇХ МОДЕЛЬ

В роботі звернена увага на те, що хоча в прикладних дослідженнях зустрічаються сигнали (емпіричні функції) із змінним періодом, проте теорія та методи дослідження таких сигналів на даний час відсутні. У зв'язку з цим в статті вирішується одна із основних задач на шляху побудови теорії сигналів із змінним періодом – обґрунтовується модель таких сигналів у вигляді функції із змінним періодом та розглядаються деякі властивості періоду. Наведено також аналітичні приклади таких функцій – це тригонометричні функції із змінним періодом та записані формули їх періодів. Отримані результати є важливим кроком до розробки конструктивної теорії функцій із змінним періодом, зокрема побудови теорії рядів Фур'є таких функцій.

The article stresses that though some signals (empirical functions) with a variable period occur in the applied research, there is neither theory nor methods of research of such signals at present. Due to this the article is solving one of the main tasks on the way to the signals theory with a variable period development, namely a model of such signals as a function with a variable period is substantiated and some characteristics of period are considered. The analytical examples of such functions, i.e. trigonometrical functions with a variable period are given, and the formulae of their periods are derived. The received results is an important step to the development of the structural function theory with a variable period, namely the development of Fourier series theory of such functions.

## 1. Вступ

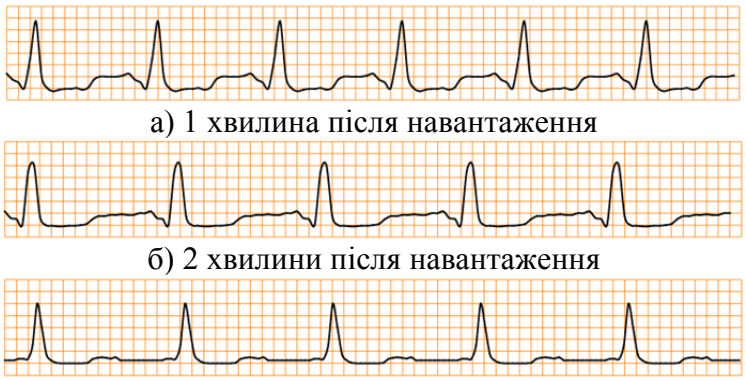
Крім періодичних сигналів (емпіричних функцій), період яких без будь-яких застережень вважається постійним, в прикладних дослідженнях зустрічаються сигнали, які теж можна вважати періодичними, однак при наступній умові: їх період вже не є постійним, а певним чином змінюється. Яскравим приладом тут є добре всім відома електрокардіограма, але отримана не в стані спокою, а після дії на пацієнта певного збудника, найпростіше – фізичного навантаження. Приклад такої електрокардіограми поданий на рис. 1.

Як вивчати сигнали із змінним періодом? Як це не дивно, але огляд наукових джерел засвідчує, що на даний час математична теорія таких сигналів відсутня, а значить відсутні будь-які аналітичні методи їх дослідження.

**Мета роботи:** розглянути приклади сигналів із змінним періодом, навести означення функції із змінним періодом, як їх математичної моделі, розглянути приклади тригонометричних функцій із змінним періодом та записати формули їх періодів, що в сукупності може бути покладене в основу наступних кроків створення теорії таких сигналів.

## 2. Приклад сигналів із змінним періодом

Вище вже йшла мова про наявність сигналів із змінним періодом. Наглядним прикладом тут є електрокардіограма, отримана під час чи після дії на організм пацієнта фізичного навантаження, і яку розглядають протягом деякого проміжку часу, поки пульс приходить в «норму». На рис. 1 наведені три відрізки електрокардіограми, кожний тривалістю 3 сек., взяті через певні проміжки часу після дії навантаження. На рис.1а) – електрокардіограма, отримана через 60 сек. після дії навантаження, на рис 1б) і 1в) – відповідно через 120 сек. і 180 сек. після навантаження. Аналізуючи графіки, видно, що форма електрокардіограми приблизно повторюється як на кожному із графіків, так і на різних графіках. Але при цьому легко бачити, що період повторюваності змінюється, а саме збільшується, та із плином часом стабілізується, що відповідає зменшенню частоти пульсу, або те саме, що збільшенню періоду повторюваності електрокардіограми. Очевидно, подібною до кардіограми буде поведінка спірограми, теж отриманої після дії навантаження чи іншого збудника психофізичного стану людини. Приклади аналогічних сигналів можна також навести із функціонування деяких технічних систем. Це може бути робота двигунів, дизель-генераторів в перехідних режимах, наприклад, після зміни зовнішнього навантаження. За даними Міжнародної служби обертання Землі добовий період обертання Землі є змінним.



а) 1 хвилина після навантаження

б) 2 хвилини після навантаження

в) 3 хвилини після навантаження

***Рис. 1. Відрізки електрокардіограми, отриманої після дії на організм пацієнта фізичного навантаження***

Наявність сигналів із змінним періодом ставить природні запитання щодо підходів та методів їх вивчення. Як показує огляд літературних джерел, ні теорії, ні загальних аналітичних методів дослідження таких сигналів не існує. Щодо електрокардіограм, як сигналів із змінним періодом, то, звичайно, що спеціалісти, а це, як правило, лікарі-терапевти, певним чином їх аналізують. Суть аналізу загалом зводиться до аналізу $R$-$R$ інтервалів. Зауважимо [1, с.85], що $R$-$R$ інтервал – це медичне поняття (термінологія), і являє собою віддаль між сусідніми $R$-зубцями електрокардіограми. В свою чергу $R$-зубець означає точку на електрокардіограмі, в якій вона приймає екстремальне, а саме, максимальне значення. $R$-зубці і $R$-$R$ інтервали добре видно на рис.1. На відрізку електрокардіограми 1а) таких $R$-зубців шість, а $R$-$R$ інтервалів п’ять, на відрізках 1б) і 1в) знаходиться по п’ять $R$-зубців і відповідно по чотири $R$-$R$ інтервали. При аналізі електрокардіограм спостерігається два підходи. Це традиційний підхід, коли спеціалісти зосереджуються на візуально-описовому аналізі $R$-$R$ інтервалів. В останній час широко використовується інформаційно-обчислювальні пристрої, з допомогою яких здійснюється вимірювання $R$-$R$ інтервалів та проводиться їх аналіз обчислювальними методами, зокрема методами математичної статистики. Звичайно, що ці та подібні їм методи загальної проблеми дослідження сигналів із змінним періодом не вирішують. Тому постає цілком закономірне питання – як досліджувати сигнали із змінним періодом, який для цього вибрати шлях, яким підходам віддати перевагу?

### 3. Загальний підхід до аналітичного вивчення сигналів із змінним періодом

Досвід багатьох науковців свідчить, що надзвичайно плідним тут вважається підхід, суть якого зводиться до тріади «модель – алгоритм – програма» [2, стр.8-9]. Згідно цього підходу на першому етапі вибирається (або обґрунтовується нова) модель («еквівалент») об’єкта, що відображає в математичній формі найважливіші його властивості: закони якими він підпорядковується; зв’язки, властиві його складовим тощо. Другий етап – вибір (чи розробка) алгоритмів для реалізації моделі на комп’ютері. Модель подається у формі, придатній для застосування числових методів, визначається послідовність обчислювальних і логічних операцій, які потрібно здійснити, щоб знайти шукані величини із заданою точністю. На третьому етапі створюються програми, що «перекладають» модель і алгоритми на доступну комп’ютерові мову. Основним в цьому підході є, безумовно, перший етап – обґрунтування моделі, оскільки від адекватності об’єкта і його моделі залежить успішність та достовірність розв’язків наступних задач тріади. Що стосується сигналів із змінним періодом, то вперше модель була запропонована в [3]. Розглянемо це важливе питання більш детально.

### 4. Поняття функції із змінним періодом

Для сигналів із змінним періодом найбільш характерними є дві особливості: повторюваність значень сигналу та змінність періоду цієї повторюваності. Яким чином врахувати ці особливості сигналу в його моделі, розглядалося в [3]. Як один із основних результатів цієї роботи – був введений клас функцій із змінним періо-

дом. Нагадаємо це поняття та деякі властивості змінного періоду.

**Означення**. Функція $f(x)$ дійсного аргументу $x \in I \subseteq R$ називається функцією із змінним періодом, якщо існує така функція $T(x) > 0$, якщо для всіх $x \in I$, таких, що $x + T(x) \in I$, виконується рівність

$$f(x) = f(x + T(x)) \qquad (1)$$

Функція $T(x)$, яку назвемо змінним періодом, вважається диференційовною функцією.

Вважатимемо надалі, що область визначення $I = [a,b]$ в кожному конкретному випадку повинна уточнюватися, що буде зустрічатися нижче.

Із (1) випливає, що при $T(x) = T = const$ функція $f$ є звичайною періодичною функцією з постійним періодом $T$.

Приклад змінного періоду $T(x)$ показано на рис. 2. В точці $x_1$ період функції $f(x)$ дорівнює $T(x_1)$, тому значення функції в точках $x_1$ і $x_1 + T(x_1)$ рівні: $f(x_1) = f(x_1 + T(x_1))$. В точці $x_2$ періодом є число $T(x_2)$, причому значення періоду $T(x)$ в точках $x_1$ і $x_2$ різні: $T(x_1) > T(x_2)$.

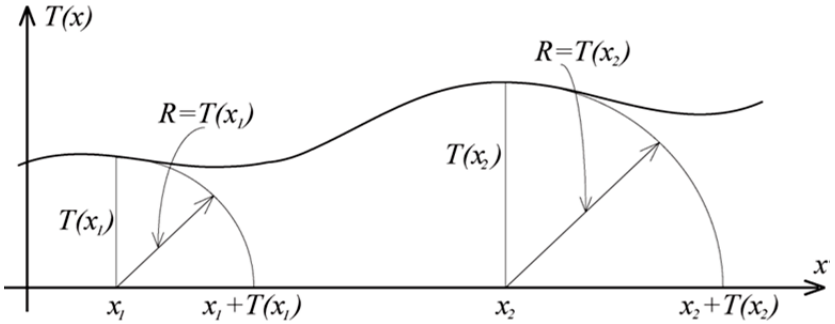



***Рис. 2. Змінний період $T(x)$, його значення в точках $x_1$ і $x_2$ та відповідні їм точки $x_1 + T(x_1)$ і $x_2 + T(x_2)$, в яких значення функції $f(x)$ повторюються.***

### 5. Змінний період $T^-(x)$ та його взаємозв'язок із періодом $T(x)$

Відомо, що для періодичної функції $g(x)$ з постійним періодом $T$ виконується рівність $g(x) = g(x+T) = g(x-T)$. Нескладні міркування, зокрема звернення до рис. 2, показують, що для функції $f(x)$ із змінним періодом $T(x)$ аналогічна рівність $f(x) = f(x + T(x)) = f(x - T(x))$ в загальному не виконується. Тому для випадку, коли аргумент $x$ зменшується, змінний період повторюваності функції $f(x)$ позначимо через $T^-(x)$. При цьому, якщо $x$ і $x - T^-(x)$ належать області визначення $I$, то

$$f(x) = f(x - T^-(x)) \qquad (2)$$

Наявність функцій із змінним періодом, як моделей відповідних сигналів, відкриває перспективу розробки методів їх дослідження – аналітичних, числових тощо. Враховуючи загальні положення теорії конструктивних функцій [4,5], для цього найперше необхідно, щоб в кожному конкретному випадку досліджувана функція була аналітичною. Це означає, що або вона сама безпосередньо є аналітичною, тобто задана деякою формулою, або подається (зображується) як певне наближення у вигляді деякого ряду, наприклад, ряду Фур'є, доданками якого є найпростіші аналітичні функції, що враховують змінність періоду. Але тут постає нове питання щодо наявності найпростіших (елементарних) аналітичних функцій із змінним періодом, які можна було б використати в задачах наближення.

Важливо відзначити, що такі функції знайдені, і ними є тригонометричні функції із змінним періодом. Приклади таких функції а також їх змінні періоди розглянемо більш детально.

### 6. Тригонометричні функції із змінним періодом та їх періоди

Один із методів отримання функцій із змінним періодом – це нелінійне перетворення (трансформація) аргументу періодичних функцій, і в першу чергу тригонометричних функцій. Для прикладу розглянемо найпростіші із періодичних функцій – $\sin t$ та $\cos t$. Нехай в свою чергу $t = x^\alpha$, $\alpha > 0, \alpha \neq 1$. В цьому випадку тригонометричні функції $\sin x^\alpha$, $\cos x^\alpha$, $\alpha > 0, \alpha \neq 1, x \in I$, є періодичними із змінними періодами. Область визначення

$I$ кожної із цих функцій залежить від значення $\alpha$ та парності функції, і може бути одним із інтервалів $[0,\infty)$ або $(-\infty,\infty)$. Для спрощення деяких міркувань будемо вважати, що $I=[0,\infty)$

Можна показати, що для тригонометричних функцій із змінним періодом має місце наступне

**Твердження.** Для функцій $\sin x^{\alpha}$ та $\cos x^{\alpha}$, $\alpha>0$, $x\ge 0$, їх змінні періоди $T_{\alpha}(x)$ та $T_{\alpha}^{-}(x)$ визначаються наступним чином:

$$T_{\alpha}(x)=-x+\left(x^{\alpha}+2\pi\right)^{1/\alpha},\ x\in[0,\infty) \qquad (3)$$

$$T_{\alpha}^{-}(x)=x-\left(x^{\alpha}-2\pi\right)^{1/\alpha},\ x\in[T(0),\infty) \qquad (4)$$

До цього твердження зробимо деякі зауваження.

- ✓ Коли не буде виникати непорозумінь, індекс $\alpha$ у виразах змінних періодів $T_{\alpha}(x)$ та $T_{\alpha}^{-}(x)$ іноді може опускатися.
- ✓ Якщо в виразах змінного періоду (3) і (4) параметр $\alpha=1$, то легко бачити, що в цьому частинному випадку періоди набувають значень $T_1(x)=2\pi$ і $T_1^{-}(x)=2\pi$, тобто отримуємо період функцій $\sin x$ і $\cos x$.
- ✓ Умова $x\in\left[T(0),\infty\right)$ у виразі (3) означає наступне. Коли рухатися вздовж осі $OX$ в сторону зменшення аргументу $x$, то враховуючи, що для функцій $\sin x^{\alpha}$ і $\cos x^{\alpha}$ їх область визначення $I=[0,\infty)$, для значень аргументу $x$, таких, що $0\le x<T(0)$, не існує точок зліва від них, в яких би значення функцій $\sin x^{\alpha}$ і $\cos x^{\alpha}$ повторювалися через період $T_{\alpha}^{-}(x)$

### 7. Приклади тригонометричних функцій із змінним періодом та їх періоди

**Приклад 1.** На рис. 3 зображена функція $f_1(x)=\sin x^{3/4}$, $x\ge 0$ та для порівняння функція $f_2(x)=\sin x$.

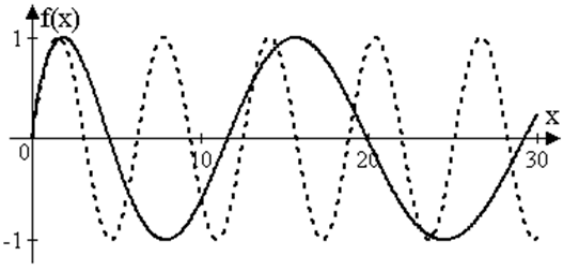



***Рис. 3. Функції*** $f_1(x)=\sin x^{3/4}$ ***(неперервна лінія),*** $f_2(x)=\sin x$ ***(пунктирна лінія).***

Аналізуючи графіки, видно, що функція $f_1(x)=\sin x^{3/4}$ із зростанням аргументу «розтягується», тобто її період при цьому зростає. На відрізку $[0,30]$ для цієї функції вкладається два періодичних коливання, для функції $\sin x$ на цьому ж інтервалі розміщується більше чотирьох коливань.

Враховуючи (3) і (4), для функції $f(x)=\sin x^{3/4}$ її зміні періоди

$$T(x)=-x+\left(x^{3/4}+2\pi\right)^{4/3},\ x\in[0,\infty),$$

$$T^{-}(x)=x-\left(x^{3/4}-2\pi\right)^{4/3},\ x\in[T(0),\infty),$$

Оскільки $T(0)=(2\pi)^{4/3}\approx 11.594$, то останній вираз ще можемо записати у вигляді

$$T^{-}(x)=x-\left(x^{3/4}-2\pi\right)^{4/3},\ x\in[11.594,\infty),$$

або $T^{-}(x)=x-\left(x^{3/4}-2\pi\right)^{4/3}$, $x\ge 11.594$.

Графіки цих періодів показані на рис. 4. Для порівняння подано також період $T(x)=T=2\pi$ функції $\sin x$.

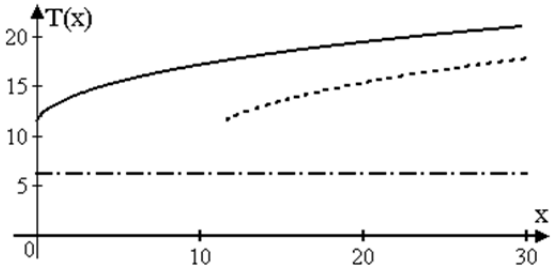



***Рис 4. Змінні періоди для функції*** $\sin x^{3/4}$ ***:*** $T(x)$ ***(неперервна лінія),*** $T^{-}(x)$***,*** $x\ge(2\pi)^{4/3}\approx 11.594$ ***(пунктирна лінія) та період*** $T(x)=2\pi$ ***для функції*** $\sin x$ ***(штрих-пунктирна лінія).***

Поведінка періодів $T(x)$ та $T^{-}(x)$ підтверджує проведені вище міркування в прикладі 1 щодо поведінки самої функції $\sin x^{3/4}$. Так в точці $x=0$ період $T(0)=11.594$, для $x=15$ період $T(15)=18.427$, що перевищує значення $T(0)$ і свідчить про його зростання. В свою чергу період $T^{-}(15)=13.519$, і є меншим, ніж

$T(15)=18.427$, що підтверджує поведінку самої функції $\sin x^{3/4}$: якщо в сторону зростання аргументу вона «розтягується», то в сторону зменшення аргументу ця функція, навпаки, – «стискається».

**Приклад 2.** Для випадку, коли $\alpha>1$, на рис. 5 показана функція $f_1(x)=\sin x^{4/3}$, $x\ge 0$, (неперервна лінія), і для порівняння – функція $f_2(x)=\sin x$ (пунктирна лінія). Із рисунка видно, що із зростанням аргументу функції $\sin x^{4/3}$ «стискається», тобто період зменшується: коли на проміжку [0,15] вкладається дещо більше двох коливань функції $\sin x$, то функція $\sin x^{4/3}$ робить більше п'яти з половиною коливань.

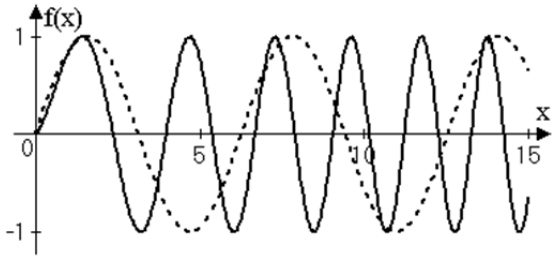



***Рис. 5. Функції:*** $f_1(x)=\sin x^{4/3}$ ***(неперервна лінія);*** $f_2(x)=\sin x$ ***(пунктирна лінія).***

Для функції $f(x)=\sin x^{4/3}$ її змінні періоди

$$T(x)=-x+\left(x^{4/3}+2\pi\right)^{3/4},\ x\in[0,\infty),$$

$$T^{-}(x)=x-\left(x^{4/3}-2\pi\right)^{3/4},\ x\in\left[T(0)\approx 3.986,\infty\right).$$

Графіки цих періодів зображені на рис. 6. Для порівняння подано також період $T(x)=T=2\pi$ функції $\sin x$.

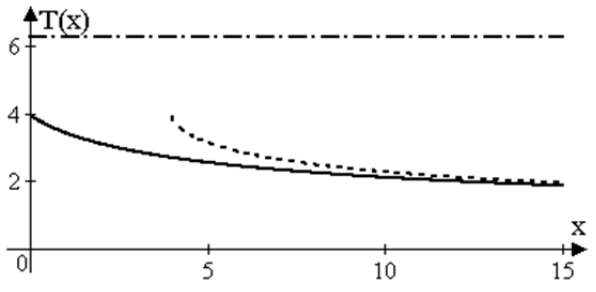



***Рис. 6. Змінні періоди для функції*** $\sin x^{4/3}$ ***:*** $T(x)$ ***(неперервна лінія);*** $T^{-}(x)$***,*** $x\ge(2\pi)^{3/4}\approx 3.986$ ***(пунктирна лінія). Період*** $T=2\pi$ ***для функції*** $\sin x$ ***(штрих-пунктирна лінія).***

Із рисунків видно, що із зростанням аргументу $x$ період $T(x)=-x+\left(x^{4/3}+2\pi\right)^{3/4}$ спадає. Так, коли для $x=0$ період $T(0)=3.986$, то для $x=10$ період $T(10)=2.115$. Період $T^{-}(x)=x-\left(x^{4/3}-2\pi\right)^{3/4}$, $x\in[3.986,\infty)$, навпаки, із зменшенням аргументу $x$ зростає.

Подібною до поведінки функцій $\sin x^{3/4}$ і $\sin x^{4/3}$ та їх періодів буде поведінка функцій $\cos x^{3/4}$ і $\cos x^{4/3}$ та їх періодів.

**Приклад 3.** Розглянемо ще тригонометричну функцію $tgx^{\alpha}$, $x\ge 0$. На рис.7 зображено графік функції $f(x)=tgx^{3/5}$ та для порівняння графік функції $tgx$.

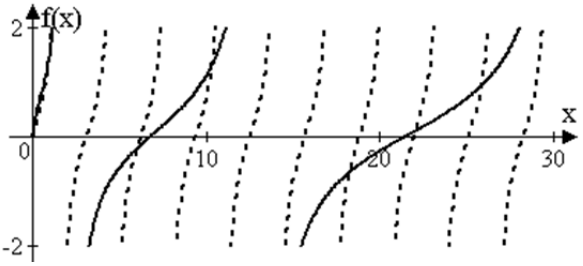



***Рис. 7. Графіки функцій*** $tgx^{3/5}$ ***(неперервні лінії) та*** $tgx$ ***(пунктирні лінії)***

Для функції $tgx^{3/5}$, $x\ge 0$, її зміні періоди

$$T(x)=-x+\left(x^{3/5}+\pi\right)^{5/3},\ x\in[0,\infty),$$

$$T^{-}(x)=x-\left(x^{3/5}-\pi\right)^{5/3},\ x\in\left[T(0)\approx 6.739,\infty\right),$$

подані на рис. 8. Для порівняння показано також період $T(x)=T=\pi$ функції $tgx$.

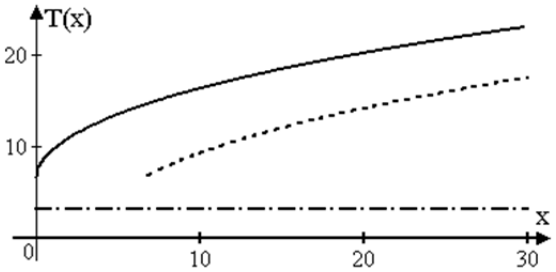



***Рис.8. Змінні періоди для функції*** $tgx^{3/5}$ ***:*** $T(x)$ ***(неперервна лінія),*** $T^{-}(x)$***,*** $x\ge\pi^{5/3}\approx 6.739$ ***(пунктирна лінія). Період*** $T(x)=\pi$ ***(штрих-пунктирна лінія) для функції*** $tgx$***.***

Узагальнюючи міркування, висловлені у розглянутих вище прикладах, можна стверджувати, що для тригонометричних функцій $\sin x^{\alpha}$, $\cos x^{\alpha}$, $tgx^{\alpha}$, $ctgx^{\alpha}$, $x \in [0,\infty)$, при значеннях $\alpha$, таких, що $0 < \alpha < 1$, їх графіки із зростанням аргументу розтягуються, відповідно періоди є зростаючими функціями. При $\alpha > 1$ графіки цих функцій стискаються, а їх періоди є спадними функціями. В частинному випадку, коли $\alpha = 1$, отримаємо добре відомі функції $\sin x$, $\cos x$, $tgx$, $ctgx$, період яких є постійними: для функцій $\sin x$ і $\cos x$ період $T = 2\pi$, для $tgx$ і $ctgx$ період $T = \pi$.

Наявність розглянутих вище тригонометричних функцій із змінним періодом наштовхує на думку про побудову на цій основі тригонометричної системи функцій із змінним періодом та дослідження такої системи. Зрозуміло, що позитивне вирішення цих питань є в свою чергу передумовою розвитку теорії наближення функцій із змінним періодом, зокрема теорії рядів цих функцій. В цьому напрямку вже отримано ряд результатів, які планується опублікувати в наступних роботах.

## 8. Висновки

Звернено увагу на наявність сигналів із змінним періодом та наведено конкретний приклад таких сигналів. Запропоновано їх математичну модель сигналів у вигляді функції із змінним періодом та розглянуто деякі властивості змінного періоду. Розглянуто аналітичні приклади – тригонометричні функції із змінним періодом та в явному вигляді записані їх періоди. Наголошено, що отримані в роботі результати є основою розвитку теорії функцій із змінним періодом, зокрема теорії рядів Фур'є цих функцій та їх практичного використання в задачах електротехніки, кардіології тощо..

## Перелік посилань


1. Морман Д. Физиология сердечно-сосудистой системы / Д. Морман, Р. Хеллер. – СПб: Издательство «Питер», 2000. – 256 с.
2. Самарский А.А. Математическое моделирование: Идеи. Методы. Примеры / А.А. Самарский, А.П. Михайлов. – 2-е изд., испр. – М. : Физматлит, 2001. – 320 с.
3. Приймак М.В. Умовно періодичні випадкові процеси із змінним періодом / М.В. Приймак, І.О. Боднарчук, С.А. Лупенко // Вісник Тернопільського державного технічного університету. – 2005. – Т.10, №2. – С. 132-141.
4. Бернштейн С.Н. Конструктивная теория функций / Бернштейн С.Н. – Издательство АН СССР, 1954. – 628 с. – (Собрание сочинений. [1931-1953]; Т.2).
5. Натансон И.П. Конструктивная теория функций / Натансон И.П. – М.-Л.: Гос. Изд-во технико-теоретической литературы, 1949. – 454 с.

УДК 004.2

*ЖАБИН В.И.,*
*ЖАБИНА В.В.*

# ПОВЫШЕНИЕ ЭФФЕКТИВНОСТИ ПАРАЛЛЕЛЬНЫХ ВЫЧИСЛЕНИЙ В ПОТОКОВЫХ СИСТЕМАХ

Исследуются параллельные системы для реализации мелкозернистых алгоритмов на базе вычислительных модулей, выполняющих операции в избыточных системах счисления с фиксированной и с плавающей запятой при поразрядном вводе и выводе данных. Показано, что применение таких вычислительных модулей позволяет частично совмещать во времени выполнение зависимых по данным операции и упростить коммутационную среду систем. Выполнение операций с плавающей запятой позволяет расширить диапазон представления чисел и обеспечивает асинхронный режим передачи данных, что дает возможность ускорить вычисления.

In respect of fine-grained algorithms implementation there are considered parallel systems on the base of processors which perform operations using fixed and floating point arithmetic as well as redundant numerical systems for bit-by-bit data input and output. It is shown that such computing modules enable to partly execute simultaneously sequences of operations as well as to simplify respective system communication structure. The use of floating point arithmetic for operation execution enables to extend the range of number presentation as well as to provide the asynchronous mode for data transfer giving an opportunity to make calculations faster.

## Введение

Время решения задач в параллельных вычислительных системах во многом зависит от реализуемого уровня параллелизма, который определяется зернистостью представления графа задач. В системах реального времени, например, связанных с траекторными вычислениями, необходимо обеспечить высокую скорость решения систем алгебраических уравнений, интерполяции функций различными методами и т.д. [1]. Наиболее высокая степень распараллеливания может быть достигнута при мелкозернистом представлении алгоритмов, когда вершинам графа задачи соответствуют отдельные операции. При этом максимально увеличивается число параллельных ветвей, что дает потенциальную возможность использовать в системе большее количество параллельно работающих вычислительных модулей (ВМ).

Однако, скорость обработки данных в параллельных системах связана не только с длительностью выполнения операций, но и с затратами времени на обмен информацией между параллельными ветвями. Как известно, увеличение степени распараллеливания вычислений сопряжено с ростом интенсивности обмена данными между ВМ. Этот фактор является весьма важным и должен учитываться при выборе архитектуры систем и организации вычислений.

Одна из возможностей уменьшения затрат времени на обмен данными состоит в использовании потоковых систем с непосредственными связями (ПСНС) между ВМ [2, 3]. В ПСНС выходы одних ВМ подключаются к входам других в соответствии с графом потока данных (ГПД). В процессе вычислений данные пересылаются от одних ВМ к другим, преобразуясь на каждом шаге в соответствии с определенной операцией, заданной ГПД. При такой организации вычислительного процесса не требуются сложные процедуры пересылки данных между ВМ, что создает предпосылки к уменьшению непроизводительных затрат времени в процессе обмена информацией.

Уменьшение сложности средств коммутации, в свою очередь, позволяет упростить и ускорить проектирование систем на базе заказных и программируемых СБИС с использованием современной технологии SoC (System on a Chip – система на кристалле). Однако со стороны элементной базы накладывается ряд ограничений, связанных с числом выводов микросхем, наличием встроенных функциональных узлов и устройств. Например, ресурсов одной микросхемы может не хватить для погружения всей системы, то есть возникает необходимость применения нескольких корпусов микросхем, связанных между собою внешними линиями связи. При передаче информации параллельным кодом возникают проблемы, связанные с возможной нехваткой выводов микросхем. Кроме того, снижается надежность систем, так как контактные соединения между микросхе-

мами относятся к ненадежным элементам. Возрастает также энергопотребление и увеличиваются габаритные размеры системы.

Учитывая важность этих проблем, в начале 90-х годов компания Virtual Machine Works обнародовала свою новую технологию под названием VirtualWire (виртуальные соединения), представленную как технология производства больших устройств, для реализации которых приходится использовать несколько ПЛИС [4]. Идея, заложенная в основе технологии VirtualWire, заключается в следующем. Часть ресурсов микросхем используется для реализации специальных цепей, которые позволяют подключать выводы микросхем попеременно к разным источникам сигналов внутри ПЛИС. Данная технология, хотя и помогает решить проблему нехватки выводов микросхем, не позволяет в полной мере использовать незадействованные ресурсы микросхем для решения заданной задачи. Кроме того, мультиплексирование создает дополнительные временные задержки продвижения потоков данных, что противоречит самой идее потоковой модели вычислений.

Одним из эффективных подходов к решению проблемы сокращения средств коммутации в ПСНС является использование в качестве ВМ операционных устройств, выполняющих операции в неавтономном режиме при поразрядном вводе и выводе данных, начиная со старших разрядов. Благодаря поразрядному обмену данными существенно сокращаются аппаратурные затраты на средства коммутации. Однако в работах, посвященных использованию неавтономных вычислений в ПСНС, рассматриваются операции с фиксированной запятой, что приводит к ряду недостатков.

В статье анализируются недостатки применения в ПСНС арифметики с фиксированной запятой и исследуется возможность повышения эффективности вычислений за счет выполнения арифметических операций с плавающей запятой.

## Организация неавтономных вычислений в ПСНС

При реализации конкретных алгоритмов в ПСНС требуется обеспечить передачу данных (промежуточных результатов) от одних ВМ к другим в соответствии с ГПД. В реконфигурируемых ПСНС (рис. 1) необходимое соединение между ВМ обеспечивает коммутационная среда, которая настраивается в соответствии с ГПД. Проблемам реконфигурации систем посвящено много научных работ, в том числе, монографий, например, [5, 6]. В системах с жесткими связями ВМ соединяются между собой непосредственно на стадии изготовления, что ограничивает функциональные возможности систем, делая их фактически специализированными. Известны формальные методики перехода от ГПД к структуре системы, например, [2].

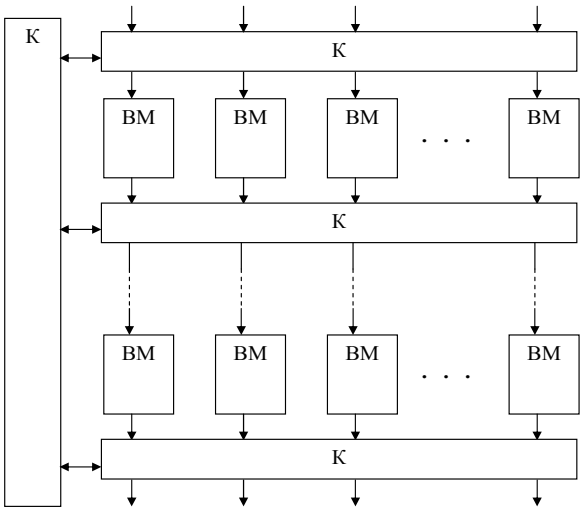



***Рис. 1. Реконфигурированная параллельная вычислительная система: К – коммутатор***

Любой путь в ГПД представляет собой цепочку связанных между собой по данным операций, которые нельзя распараллелить, так как предыдущая операция формирует операнд для следующей. Параллельно во времени могут выполняться только операции, которые принадлежат разным путям. Арифметические операции со старших разрядов выполняются с использованием однородной избыточной системы счисления с целым основанием $k$. Известны алгоритмы обработки данных как в симметричных [7, 8], так и в смещенных [9, 10] системах счисления соответственно с цифрами $\{\overline{-g, g}\}$ и $\{\overline{0, g}\}$.

Последовательность зависимых по данным операций выполняется в неавтономном режиме с помощью цепочки ВМ, которые позволяют совмещать выполнение зависимых операций на уровне обработки разрядов слов следующим образом. На каждом шаге в ВМ вводится по одному разряду операндов и формируется один разряд результата, начиная со старших разрядов. При этом разряд промежуточного результата, полученный на $i$-м шаге в одном ВМ при выполнении $j$-й операции, может быть исполь-

зован на $(i+1)$-м шаге в другом ВМ при выполнении $(j+1)$-й операции. При таком режиме вычислений выполнение последующей операции будет начинаться не после завершения выполнения предыдущей операции, а сразу же после получения первого разряда результата этой операции. Каждый ВМ начинает формировать разряды операндов с некоторой задержкой на $p_j$ шагов после получения первых разрядов операндов. В таких ВМ информация представлена последовательным кодом только на входах и выходах. Между узлами (регистрами, сумматорами) информация передается параллельным кодом, в связи с чем такие устройства называют "квазипараллельными". Режим работы ВМ называют неавтономным, так как для выполнения последовательности операций необходима совместная работа нескольких ВМ, которые синхронно обмениваются информацией в процессе работы. Если ГПД содержит критический путь из $N$ операций, то при максимальном распараллеливании алгоритм может быть реализован за время [3]

$$T = \left[ n - 1 + \sum_{j=1}^{N} \left( p_j + 1 \right) \right] T_{\text{ц}},$$

где $n$ – разрядность операндов, $T_{\text{ц}}$ – длительность цикла формирования цифры результата в ВМ.

Частичное совмещение выполнения зависимых операций дает потенциальную возможность ускорения обработки данных в потоковых системах на базе квазипараллельных ВМ, по сравнению с параллельными ВМ [2, 3]. Для сравнения времени решения задач в системах с параллельными и квазипараллельными ВМ необходимо определить значения указанных выше параметров $k$, $p_j$ и $T_{\text{ц}}$ для конкретной структурной организации модулей. Можно указать следующие недостатки синхронных вычислений с фиксированной запятой:

– существенные ограничения на диапазон представления чисел при заданной разрядности устройств;

– накопление погрешности при выполнении последовательности операций вплоть до полной потери значимости результатов вычислений;

– сложность масштабирования исходных операндов для обеспечения необходимого соотношения их величин в определенном месте цепочки операций (например, при делении делимое должно быть меньше делителя, а при сложении разряды слагаемых должны иметь одинаковый вес);

– необходимость обеспечения одновременного поступления разрядов операндов на входы операционных устройств из разных источников, что требует выбора максимального по длительности такта из всех возможных для разных операций.

Для повышения эффективности ПСНС целесообразно реализовать следующее:

– разработать методы неавтономного выполнения операций с плавающей запятой с целью повышения точности вычислений и расширения диапазона представления чисел;

– разработать методику реализации вычислений в асинхронном режиме с целью упрощения системы синхронизации и ускорения вычислений.

### Обработка данных в форме с плавающей запятой в неавтономном режиме

Число $X$ в позиционной канонической (неизбыточной) системе счисления с основанием $k$ можно записать в форме с плавающей запятой как

$$X = k^P M \ ,$$

где $M$ -мантисса, а $P$ - порядок числа $X$.

В канонической системе счисления порядок $P$ является целым числом, а мантисса – нормализованной дробью, то есть

$$k^{-1} \le |M| \le 1.$$

Если для записи порядка используется $m$ разрядов, а для записи мантиссы – $n$ разрядов, то максимальное по модулю число в форме с плавающей запятой может быть представлено следующим образом

$$k^{k^m - 1}(1 - k^{-n}) = k^{k^m - 1} - k^{k^m - n - 1}.$$

В свою очередь, минимальное по абсолютной величине число, которое отличается от нуля, записывают как

$$k^{-k^m + 1} k^{-1} = k^{-k^m}.$$

Для компьютеров разработан стандарт ANSI/IEEE 754 представления чисел с плавающей запятой. В нем используют скрытый старший разряд мантиссы с весом 1 (разрядность мантиссы увеличивается на один разряд). Для упрощения работы с порядками при выполнении операций используется смещенный порядок, позволяющий работать с целыми

числами без знаков. Заметим, что при $k > 2$ старший значащий разряд мантиссы может иметь различные значения, то есть не может быть скрытым из-за неоднозначности.

Алгоритмы выполнения операций с плавающей запятой сложнее, чем с фиксированной. Однако при этом увеличивается точность вычислений, диапазон представления чисел, а также устраняются трудности, связанные с масштабированием данных.

Рассмотрим возможный вариант представления чисел с плавающей запятой в избыточной системе счисления.

С порядками операции могут выполняться в автономном режиме, то есть в каждом вычислительном модуле операции выполняются независимо. В связи с этим нет необходимости использовать для представления порядков избыточный код. Порядок может быть представлен в виде целого числа в любом коде, в том числе, в канонической двоичной системе. Обработка порядков, которая сводится к сложению, вычитанию и сравнению кодов, обычно выполняется быстрее, чем поразрядная обработка мантисс и, как правило, может с ней совмещаться.

Более сложной является обработка мантисс. При выполнении арифметических операций возникает проблема, связанная с возможным нарушением нормализации. В канонических системах нормализация заключается в сдвиге мантиссы влево или вправо и коррекции порядка. При неавтономных вычислениях разряды мантиссы формируются поочередно, начиная со старших. Сдвиг можно осуществить только путем задержки разрядов при приеме в вычислительный модуль, что эквивалентно сдвигу кодов вправо. Сдвиг влево таким же образом невозможен.

Вторая проблема заключается в том, что наличие значащего разряда мантиссы справа от запятой не означает, что мантисса является дробным числом. Это связано с тем, что цифры в избыточном представлении могут превышать основание системы счисления. Например, в двоичной системе с цифрами {2,1,0} число 0,22102=1,10110 больше единицы, а в системе с цифрами {-1,0,1} число $0,1\bar{1}\bar{1}\bar{1}\bar{1} = 0,00001$ значительно меньше единице. Все это требует специальных алгоритмов обработки чисел в избыточных системах.

Будем считать, что числа с плавающей запятой имеют вид

$$X = k^P M ,$$

где порядок $P$ – целое число в неизбыточном коде, а мантисса $M$ представлена в избыточном коде и находится в пределах

$$k^{-1} \le |M| < M_{\max} .$$

Значение $M_{\max}$ определяется основанием системы счисления и диапазоном изменения значений цифр.

Пусть мантисса числа $X$ представлена в форме

$$M_x = \sum_{i=1}^{n} x_i k^{-i} ,$$

где $x_i \in \{\overline{g,0}\}$ или $x_i \in \{\overline{-g,g}\}$.

В обоих случаях $M_{\max}$ будет иметь вид

$$0, gg...g = g\sum_{i=1}^{n} k^{-1} . \qquad (1)$$

Определим предел суммы в правой части (1), представленной в виде ряда

$$\frac{1}{k} + \frac{1}{k^2} + \frac{1}{k^3} + ... + \frac{1}{k^n} .$$

Находим частичные суммы:

$$s_1 = \frac{1}{k},\ s_2 = \frac{k+1}{k^2},\ s_3 = \frac{k^2+k+1}{k^3},\ldots,$$
$$s_n = \frac{k^{n-1} + k^{n-2} + ... + k + 1}{k^n} .$$

Исследуя сумму $s_n$, можно найти

$$\lim_{n\to\infty} s_n = \frac{1}{k-1} .$$

С учетом полученного предела суммы и (1) определим диапазон изменения мантиссы в избыточном представлении как

$$k^{-1} \le |M| < \frac{g}{k-1} . \qquad (2)$$

Рассмотрим возможные интервалы изменения значений мантисс результатов различных операций.

Мантисса произведения, исходя из диапазона изменения мантисс (2), находится в пределах

$$k^{-2} \le |M| < \frac{g^2}{(k-1)^2}. \qquad (3)$$

Анализируя (3), можно показать, что при умножении, в отличие от канонической системы счисления, нормализация относительно (2) может быть нарушена как вправо, так и влево.

В соответствии с (2) мантисса частного от деления изменяется в диапазоне

$$\frac{k-1}{gk} \le |M| < \frac{gk}{k-1}. \qquad (4)$$

Можно показать, что нарушение нормализации может произойти как влево, так и вправо. Следует указать, что при делении мантисс необходимо, чтобы делимое было меньше делителя. Поэтому разряды делимого надо принимать в ВМ с задержкой, величина которой определяется значениями $g$ и $k$. При каждой задержке надо прибавлять единицу к порядку результата.

При сложении (вычитании) мантисс результат может находиться в пределах

$$0 \le |M| < \frac{2g}{k-1}. \qquad (5)$$

Влево нормализация может быть нарушена на один разряд, а вправо – на все $n$ разрядов.

В соответствии с (2) можно определить интервал изменения значений мантиссы для других функций. Например, для квадратного корня получим

$$\sqrt{k^{-1}} \le M < \sqrt{\frac{g}{k-1}}. \qquad (6)$$

Учитывая диапазоны изменения чисел (3), (4), (5) и (6), рассмотрим алгоритмы выполнения операций в ВМ.

Считаем, что ВМ содержит блоки для обработки мантисс и порядков, которые взаимодействуют между собой. Цифры мантисс принимаются в модуль из предыдущего модуля, начиная со старших разрядов. Передача первой значащей цифры стробируется. До начала обработки разряды операндов могут накапливаться в буфере модуля. Порядки могут передаваться и обрабатываться в неизбыточном коде. Обработка порядков во времени может совмещаться с обработкой мантисс.

***Алгоритм операции умножения*** $Z = YX$**.**

1. Принять порядки $P_x$ и $P_y$.

2. Получить предварительный порядок результата $P_z = P_x + P_y + 1$.

3. Принимать в цикле цифры мантисс операндов $x_i$ и $y_i$; формировать цифры мантиссы результата $z_i$; пока $z_i = 0$ корректировать порядок $P_z := P_z - 1$; цифру $z_i = 0$ не выдавать.

4. При формировании первой ненулевой цифры результата $z_i$ выдать окончательный порядок $P_z$ и данную первую ненулевую цифру мантиссы результата.

5. Принимать очередные цифры мантисс операндов и выдавать очередные цифры мантиссы результата до получения $n$ разрядов мантиссы результата.

Здесь считаем, что нарушение нормализации влево возможно на один разряд, что определяется значениями $g$ и $k$.

***Алгоритм операции деления*** $Z = Y / X$**.**

1. Принять порядки $P_x$ и $P_y$.

2. Получить предварительный порядок результата $P_z = P_y - P_x + 1$.

3. В первом цикле, принять старшую цифру мантиссы делителя и 0 в качестве старшей цифры мантиссы делимого.

4. Принимать в цикле цифры мантисс операндов $x_i$ и $y_i$; формировать цифры мантиссы результата $z_i$; пока $z_i = 0$ корректировать порядок $P_z := P_z - 1$; цифру $z_i = 0$ не выдавать.

5. При формирования первой ненулевой цифры результата $z_i$ выдать окончательный порядок $P_z$ и данную первую ненулевую цифру мантиссы результата.

6. Принимать очередные цифры мантисс операндов и выдавать очередные цифры мантиссы результата до получения $n$ разрядов мантиссы результата.

Считаем, что для рассматриваемой системы счисления делимое достаточно задержать на один разряд.

***Алгоритм операции сложения*** $Z = Y + X$**.**

1. Принять порядки $P_x$ и $P_y$.

2. Получить предварительный порядок результата $P_z = \max(P_x, P_y) + 1$ и разность порядков $\Delta P = P_{\max} - P_{\min}$.

3. Пока $\Delta P \neq 0$ принимать в цикле цифру мантиссы операнда с большим порядком и 0 в качестве цифры другого операнда; выполнять $\Delta P := \Delta P - 1$.

4. В последующих циклах ($\Delta P = 0$) принимать цифры обоих слагаемых и формировать цифру мантиссы результата $z_i$.

5. Если $z_i \neq 0$, то выдать окончательный порядок $P_z$ и $z_i$ в качестве старшей цифры мантиссы результата; в противном случае корректировать порядок $P_z := P_z - 1$, цифру $z_i \neq 0$ не выдавать.

6. Принимать очередные цифры мантисс операндов и выдавать очередные цифры мантиссы результата до получения $n$ разрядов мантиссы результата.

Возможность переполнения разрядной сетки учитывается в п. 2 добавлением 1 к порядку. Выравнивание порядков выполняется в п. 4. Нормализацию результата обеспечивает п. 5. При нулевом значении $n$ разрядов результата в ВМ формируется отображение машинного нуля.

***Алгоритм извлечения квадратного корня*** $Y = \sqrt{X}$**.**

1. Принять порядок операнда $P_x$.

2. Если порядок $P_x$ нечетный, то принять 0 в качестве старшей цифры мантиссы операнда; скорректировать порядок $P_x := P_x + 1$, иначе коррекция порядка не нужна.

3. Сформировать и выдать порядок результата $P_z := k^{-1} P_x$.

4. Принимать цифры операнда $x_i$, формировать и выдавать цифры мантиссы результата $z_i$ до получения $n$ разрядов мантиссы результата $Z$.

Коррекция порядка операнда в п. 2 обеспечивает получения целого значения порядка результата.

При использовании симметричных систем счисления возможны ситуации, когда первая значащая цифра результата в каждом цикле фактически сдвигается вправо, то есть в данном цикле должна иметь нулевое значение. Например, $\bar{1}\bar{1}\bar{1}\bar{1}1 = 00001$. В этом случае выдача цифры из ВМ задерживается, а в следующем цикле проверяется значение следующей за ней цифры. Цифра выдается из модуля, когда превращение ее в 0 невозможно. Аппаратная реализация этого процесса достаточно проста.

Таким образом, применение плавающей запятой расширяет диапазон представления чисел по сравнению с фиксированной запятой. Если для записи порядка используется $m$ разрядов, а для записи мантиссы – $n$ разрядов, то число $X$ в форме с плавающей запятой в системе с основанием $k$ может находиться в пределах

$$k^{-k^m+1} k^{-1} \leq X < k^{k^m-1} \frac{g}{k-1}.$$

При одинаковой длине разрядной сетки обеспечивается повышение точности вычислений по сравнению с фиксированной запятой.

Предложенная методика выполнения операций с плавающей запятой позволяет асинхронно передавать данные между ВМ, то есть каждый ВМ может иметь собственную частоту тактирования. Это создает предпосылки для ускорения вычислений в ПСНС.

## Выводы

В работе исследована возможность повышения эффективности параллельных вычислений в потоковых системах с непосредственными связями между вычислительными модулями, работающими в избыточных системах счисления в неавтономном режиме.

Разработаны алгоритмы неавтономного выполнения основных арифметических операций с плавающей запятой, позволяющие совмещать процессы поразрядного ввода операндов, их обработки и поразрядного вывода результата, начиная со старших разрядов.

Благодаря этому имеется потенциальная возможность ускорения вычислений за счет выполнения цепочек зависимых по данным операций в режиме совмещения, что нельзя обеспечить в автономном режиме.

Показано, что по сравнению с использованием фиксированной запятой применение операций с плавающей запятой в потоковых системах позволяет расширить диапазон представления чисел, повысить точность получения результатов при заданной длине разрядной сетки и ускорить вычисления за счет обработки информации в асинхронном режиме.

## Список литературы


1. Байков В.Д. Решение траекторных задач в микропроцессорных системах ЧПУ / В.Д.Байков, С.Н.Вашкевич. – Л.: Машиностроение, 1986, 105 с.
2. Жабин В.И. Построение быстродействующих специализированных вычислителей для реализации многоместных выражений / В.И.Жабин, В.И.Корнейчук, В.П.Тарасенко // Автоматика и вычислительная техника. – 1981. - №6. – с. 18-22.
3. Жабин В.И. Выполнение последовательностей зависимых операций в режиме совмещения / В.И.Жабин. // Вісник НТУУ «КПІ». Інформатика, управління та обчислювальна техніка: Зб. Наук. Пр. – К.: Век+. – 2007. - №46. – С. 226-233.
4. Максфилд К. Проектирование на ПЛИС. Архитектура, средства и методы / К.Максфилд. – М.: Издательский дом «Додэка-XXI», 2007, 408 с.
5. Палагин А.В. Реконфигурируемые вычислительные системы: Основы и приложения / А.В.Палагин, В.Н.Опанасенко. – К.: Просвіта, 2006, 280 с.
6. Каляев И.А. Архитектура семейства реконфигурируемых вычислительных систем на основе ПЛИС / И.А. Каляев, И.И. Левин, Е.А. Семерников // Искусственный интеллект. – 2008. - № 3. – с. 663-674.
7. Жабин В.И. Некоторые машинные методы вычисления рациональных функций многих аргументов / В.И. Жабин, В.И.Корнейчук, В.П.Тарасенко // Автоматика и телемеханика. – 1977. - №12. – С. 145-154.
8. Жабин В.И. Методы быстрого неавтономного воспроизведения функций / В.И.Жабин, В.И.Корнейчук, В.П.Тарасенко // Управляющие системы и машины. – 1977. - №3. – С. 96-101.
9. Дичка И.А. Совмещение зависимых операций на уровне обработки разрядов операндов / И.А.Дичка, В.В.Жабина. // Искусственный интеллект. – 2008. - №3. – С. 649-654.
10. Дичка И.А. Метод вычисления функций в неавтономном режиме / И.А.Дичка, В.В.Жабина // Искусственный интеллект. – 2009. – №4. – С. 409-414.

УДК 004.056.5

*МАРКОВСЬКИЙ О.П.,*
*ВИНОГРАДОВ Ю.М.,*
*САЛОХА О.Є.,*
*ТКАЧЕНКО І.М.*

# ЕФЕКТИВНЕ ОБЧИСЛЕННЯ КВАДРАТНОГО КОРЕНЯ НА ПОЛЯХ ГАЛУА GF($2^m$)

У статті запропоновано спосіб прискореного обчислення кореня на полях Галуа GF ($2^m$). Показано, що задача обчислення коренів на полях Галуа може бути зведена до розв'язання системи лінійних бітових рівнянь. Запропоновано технологію реалізації цієї теоретичної ідеї. Доведено, що обчислювальна складність O*(m)* запропонованого способу істотно менша, ніж складність відомих способів, що становить O*($m^2$)*.

In article, the method of accelerated calculation of square root on Galois fields GF ($2^m$) has been proposed. By the theoretical way, it has been shown that computing roots on Galois fields' calculation can be reduced to solving system of linear bits equations. New technology of this theoretical idea was proposed. It has been proved, that calculation complexity O*(m)* of proposed method is much smaller in comparing to known methods, which equals O*($m^2$)*.

## Вступ

Розв'язання алгебраїчних рівнянь на кінцевих полях і, зокрема, на полях Галуа GF GF($2^m$), є однією з найважливіших задач обчислювальної алгебри і теорії чисел [1]. Перші роботи, присвячені вирішенню цієї задачі, були опубліковані більше ста років тому [2].У наш час, крім теоретичного інтересу, ця задача має велику практичну цінність, оскільки вона відіграє ключову роль в комп'ютерних технологіях корекції помилок, кодування і стиснення даних. Потужним імпульсом розвитку комп'ютерних технологій у розв'язанні таких рівнянь стало практичне використання криптографічних систем захисту інформації, що базуються на еліптичних кривих та інших різновидах Абелевих груп [1].

Найважливішою складовою частиною технологій вирішення алгебраїчних рівнянь на кінцевих полях є обчислення квадратного кореня. Типовим застосуванням операції добування кореня є стиснення і відновлення точки на еліптичній кривій [3]. Точка з координатами *(x, y)* на кривій стискається до виду *(x, β)* де $\beta \in \{0,1\}$. Для відновлення чисельного значення *y* по *(x,β),* необхідно вирішити квадратне рівняння $y^2 = P(x)$, тобто обчислити квадратний корінь $\sqrt{P(x)}$. Подібна ситуація виникає і при хешуванні на еліптичних кривих, яке застовується в ряді криптосистем [4,5].

Ефективність засобів корекції помилок, кодування і стиснення даних, а також систем криптографічного захисту інформації значною мірою визначається можливістю досягнення високої продуктивності при програмній та апаратній реалізації. Операція добування квадратного кореня на полях Галуа GF($2^m$) є однією з найскладніших задач в обчислювальному плані, тому від швидкості її реалізації значною мірою залежить продуктивність зазначених вище засобів. При використанні існуючих методів добування квадратного кореня на полях GF($2^m$) час виконання цієї операції пропорційний $m^2$.

З розвитком технологій розподілених обчислень істотно зросли можливості комп'ютерних систем, які потенційно можуть бути використані для порушення захисту. Найпростішим заходом підвищення криптостійкості систем, що ґрунтуються на використанні полів Галуа GF($2^m$), є збільшення розрядності *m* чисел, що використовуються. Це значною мірою уповільнює продуктивність засобів криптографічного захисту. Тому, в сучасних умовах важливою і актуальною є проблема розробки нових підходів до прискорення програмної та апаратної реалізації операції обчислення квадратного кореня на кінцевих полях. Основним резервом зменшення обчислювальної складності цієї важливої для практичних застосувань операції є врахування особливостей її використання в реальних системах [6].

## Аналіз відомих методів обчислення коренів на кінцевих полях

Практична значимість задачі обчислення квадратного кореня на кінцевих полях, особливо для систем криптографічного захисту інформації на основі еліптичних кривих, стимулює інтенсивні дослідження в області методів вирі-

шення цієї задачі. Як вже зазначалося, більше ста років тому були запропоновані два базових методи обчислення квадратного кореня на полях Галуа GF($2^m$): Tonelli [2] та Cipolla [5]. Пізніше ці методи були розширені для випадку поля GF($q^m$), де $q$ – просте число і отримали відповідні назви: Tonelli-Shanks [4] і Cipolla-Lehmer [1].У 1977 році в роботі [5] метод Tonelli-Shanks був розширений для випадку добування кореня довільного ступеня. В роботі [6] був розроблений спеціалізований метод обчислення кубічного кореня, що відрізняється підвищеною швидкодією.

Базовими операціями на полях Галуа GF($2^m$) є додавання та множення їх елементів. Операція додавання відповідає додаванню в поліноміальній математиці і далі позначена як '+'. Операція множення на полях GF *($2^m$)* фактично складається з двох операцій: поліноміального множення (множення без переносів), позначеного далі символом '⊗' і редукції, тобто знаходження залишку від поліноміального ділення добутку на утворюючий поліном $P(x)$ поля. Операція редукції позначена далі як 'rem', на відміну від арифметичної редукції 'mod'.

Для кожного елементу поля GF($2^m$), що утворюється нерозкладним поліномом *P(x)* ступеню *m,* якому відповідає число *p,* існує мультиплікативна циклічна група, порядок *n* якої не перевищує $2^m-1$. Наприклад, для поля Галуа, утвореного нерозкладним поліномом $P(x) = x^4 + x^3 + 1$ ($p = 25_{10} = 11\ 001_2$) генеруються циклічні групи. Порядок циклічної групи дорівнює $2^m-1$, якщо її генератор не має спільних дільників з $2^m-1$.

У кожній циклічній групі поля GF($2^m$) може бути виділена циклічна підгрупа, кожен елемент якої є квадратом попередньої. При цьому порядок кожної з квадратичної підгруп не перевищує $log_2 n$, тобто менше або дорівнює $m$.

Основна ідея знаходження квадратного кореня $\sqrt{A}$ на полях Галуа GF($2^m$) методом Tonelli-Shanks полягає в проходженні квадратичної циклічної підгрупи до знаходження її елемента, що передує шуканому. У процедурному значенні прохід по квадратичній циклічній підгрупі еквівалентний операції експоненціювання[10].

$$B = A^{2^{m-1}} rem(p). \qquad (1)$$

Таким чином, ідея добування квадратного кореня на полях GF($2^m$) теоретично є досить простою, однак її практична реалізація пов'язана зі значними витратами обчислювальних ресурсів, оскільки обчислення (1) передбачає виконання $m$-1 операцій піднесення до квадрату і редукції.

Операції піднесення до квадрату виконуються за правилами поліноміального множення, тобто без урахування переносів. Операція піднесення до квадрату може бути ефективно реалізована з використанням важливої властивості: в двійковій формі представлення квадрата числа $A$ розряди, що знаходяться на парних позиціях, дорівнюють нулю, а непарні розряди дорівнюють відповідним розрядам числа $A$, тобто, якщо $A=a_0+a_1\cdot 2+a_2\cdot 2^2+...+a_{m-1}\cdot 2^{m-1}$, то:

$$A \otimes A = A^2 = a_0 + a_1 \cdot 2 + a_2 \cdot 2^2 + ... + a_{m-1} \cdot 2^{2m-2}. \qquad (2)$$

Найважливішим наслідком цієї властивості є той факт, що обчислення поліноміального квадрату не вимагає для своєї реалізації ніяких обчислювальних операцій, а зводиться лише до перестановки розрядів вихідного числа [7].

При оцінці обчислювальної складності операції добування квадратного кореня за методом Tonelli-Shanks слід враховувати те, що на практиці розрядність елементів поля $m$ істотно перевищує розрядність процесора $w$, тому при виконанні операцій елементи поля розбиваються на $s$ секцій *(s = m/w).*

Операція редукції, тобто приведення результату поліноміального множення в рамки поля Галуа виконується шляхом обчислення залишку від поліноміального ділення *(2·m-1)*-розрядного результату поліноміального піднесення до квадрату на *(m+1)*-розрядний код утворюючого поліному поля Галуа . Реалізація операції поліноміального ділення включає виконання *(m-1)* циклів, в кожному з яких здійснюється зсув на один розряд *(m+1)*- розрядного коду $p$ і логічне додавання його з кодом поточного залишку в разі, якщо старший розряд останнього дорівнює одиниці. Для зсуву *(m+1)*- розрядного коду $p$ на один розряд необхідно виконати *(s+1)* процесорних операцій зсуву. Так як ця операція виконується в кожному з *(m - 1)* циклів редукції, то сумарна кількість процесорних операцій зсуву становить *(s+1)·(m-1)*. Виходячи з того, що в процесі редукції операція додавання виконується, в середньому, в половині циклів поліноміального ділення, то середня кількість таких операцій становить *(m-1)/2*. Беручи до уваги, що для реалізації цієї операції на $w$- розрядному процесорі треба виконати *(s+1)* процесорних операцій логічного додавання для редукції результату множення складає: *(s+1)·(m-1)/2*. Відповідно, середній час виконання однієї ре-

дукції результату піднесення до квадрату становить $1.5\cdot(s+1)\cdot(m-1)\cdot\tau$, де $\tau$-час виконання на процесорі логічної операції. Враховуючи, що вилучення квадратного кореня вимагає $(m-1)$ операції зведення в квадрат, середнє число $N_T$ логічних операцій, потрібних для добування квадратного кореня на полі GF($2^m$) становить:

$$N_T = 1.5\cdot(s+1)\cdot(m-1)^2. \qquad (3).$$

Аналогічна оцінка обчислювальної складності $O(m^2)$ наведена в [1] і для методу Cipolla-Lehmer. У сучасних умовах збільшення продуктивності розподілених комп'ютерних систем, які потенційно можуть бути використані для порушення захисту, найбільш простим способом підвищення крипостійкості є збільшення розрядності хисел. При цьому, як випливає з (3) квадратично зростає обчислювальна складність реалізації операції отримання квадратного кореня на полях GF($2^m$).

Відомі методи обчислення квадратного кореня на полях Галуа GF($2^m$) розглядають цю важливу для практики задачу незалежно від особливостей її практичного використання в реальних криптосистемах. Разом з тим, особливості практичного використання операції дозволяють істотно зменшити її обчислювальну складність.

Метою досліджень є розробка способів прискорення вилучення квадратного кореня на полях Галуа, орієнтованих на апаратну реалізацію та широке розпаралелювання.

### Обчислення кореня рішенням системи булевих рівнянь

Обчислення квадратного кореня на полях Галуа GF($2^m$) може бути зведене до розв'язання системи лінійних бітових рівнянь. Нехай задано значення $A=a_0+a_1\cdot 2+a_2\cdot 2^2+...+a_{m-1}\cdot 2^{m-1}$, $a_0,a_1,...,a_{m-1}\in\{0,1\}$. Необхідно визначити $B=b_0+b_1\cdot 2+b_2\cdot 2^2+...+b_{m-1}\cdot 2^{m-1}$, $b_0,b_1,...,b_{m-1}\in\{0,1\}$ таке, що $(B\otimes B) rem\ P = A$ чи $B\otimes B = P\otimes D + A$, де $D=d_0+d_1\cdot 2+d_2\cdot 2^2+...+d_{m-2}\cdot 2^{m-2}$. Оскільки двійкові розряди поліноміального квадрата $B\otimes B$ числа $B$ стоять на парних позиціях рівні нулю, а розряди з непарним номером рівні двійковим розрядам числа $B$, тобто $B\otimes B = b_0+b_1\cdot 2^2+b_2\cdot 2^4+...+b_{m-1}\cdot 2^{m-2}$, $b_0 = p_0\cdot d_0 + a_0$ оскільки наступний розряд $B\otimes B$ рівний нулю, то $p_0\cdot d_1 + p_1\cdot d_0 + a_1 = 0$. Аналогічним чином, може бути отримана система бітових рівнянь:

$$\begin{cases} b_0 = p_0\cdot d_0 + a_0 \\ 0 = p_0\cdot d_1 + p_1\cdot d_0 + a_1 \\ b_1 = p_0\cdot d_2 + p_1\cdot d_1 + p_2\cdot d_0 + a_2 \\ 0 = p_0\cdot d_3 + p_1\cdot d_2 + p_2\cdot d_1 + p_3\cdot d_0 + a_3 \\ \dots \\ b_{m/2-1} = p_0\cdot d_{m-2} + \dots + p_{m-2}\cdot d_0 + a_{m-2} \\ 0 = p_1\cdot d_{m-2} + p_2\cdot d_{m-3} + \dots + p_{m-1}\cdot d_0 + a_{m-1} \\ b_{m/2} = p_2\cdot d_{m-2} + p_3\cdot d_{m-3} + \dots + p_m\cdot d_0 \\ \dots \\ 0 = p_{m-1}\cdot d_{m-2} + p_m\cdot d_{m-3} \\ b_{m-1} = p_m\cdot d_{m-2} \end{cases} . \qquad (4)$$

При постійному утворюючому поліномі $P(x)$ поля Галуа GF($2^m$) система (4) може бути приведена до вигляду:

$$\begin{cases} b_0 = \lambda_0(a_0, a_1, ..., a_{m-1}) \\ b_1 = \lambda_1(a_0, a_1, ..., a_{m-1}) \\ \dots \\ b_{m-1} = \lambda_{m-1}(a_0, a_1, ..., a_{m-1}) \end{cases} \qquad (5)$$

де $\lambda_0, \lambda_1, \dots, \lambda_{m-1}$ – лінійні булеві функції. Із системи (5) безпосередньо обчислюються значення бітів коду квадратного кореня на полі Галуа.

Наприклад, якщо $m = 4$, $P(x) = x^4 + x^3 + 1$, $p_0 = 1$, $p_1 = 0$, $p_2 = 0$, $p_3 = 1$, $p_4 = 1$, система (5) приймає вигляд:

$$\begin{cases} b_0 = a_0 + a_3 \\ b_1 = a_0 + a_1 \\ b_2 = a_1 + a_3 \\ b_3 = a_1 \end{cases} \qquad (6)$$

Наприклад, якщо $A = 15$ $(a_0 = 1, a_1 = 1, a_2 = 1, a_3 = 1)$, з системи (6) відповідно: $b_0 = 0$, $b_1 = 0$, $b_2 = 0$ і $b_3 = 1$, тобто шуканий корінь $B = 8$.

При програмній реалізації рішення системи (5) можуть бути заздалегідь заготовлені по $u$ бітових масок $M_{j1}, M_{j2}, \dots, M_{js}$ для виділення розрядів коду A, які входять в функцію $\lambda_j$ для кожного із значень $b_j$, $j\in\{0, .., m-1\}$. Відповідно, обчислення значення $b_j$ здійснюється у вигляді логічної суми бітів парності побітових добутків фрагментів коду A на відповідні маски:

$$b_j = (a_0, a_1, ..., a_{w-1})\cdot M_{j1} + (a_w, a_{w+1}, ..., a_{2w-2})\cdot M_{j2} + .. \\ .+ (a_{m-w-1}, a_{m-w}, ..., a_{m-1})\cdot M_{js}$$

Очевидно, що загальна кількість $N_L$ логічних операцій необхідних для обчислення значень $m$ бітів $b_0, b_1, \ldots, b_{m-1}$ визначається як

$$N_L = s \cdot m \qquad (7)$$

Порівняння отриманого виразу з оцінкою (3) числа операцій, необхідних для обчислення квадратного кореня на GF($2^m$) для відомих способів, показує, що запропонований спосіб виконання цієї операції забезпечує зменшення обчислення обчислювальної складності приблизно в $1.5 \cdot m$ разів:

$$\beta = \frac{N_T}{N_L} = \frac{1.5 \cdot (s+1) \cdot (m-1)}{s \cdot m} \approx 1.5 \cdot m$$

Враховуючи, що в алгоритмах захисту інформації значення $m$ складає сотні і тисячі біт, виграш в обчислювальній складності і, відповідно, у часі обчислення квадратного кореня на GF($2^m$) досягається використанням запропонованого способу, складає 2-3 порядки. Ще більший виграш як у часі обчислення квадратного кореня на полях Галуа, так і по складності схеми, застосування запропонованого способу забезпечується при апаратній реалізації, оскільки вирази (5) представляють собою гранично просту форму обчислення бітів кореня, кожен з яких може обчислюватися паралельно.

## Висновки

Запропоновано спосіб прискореного обчислення квадратного кореня на полях Галуа GF($2^m$), який базується на зведенні цієї задачі до розв'язання системи лінійних бітових рівнянь. Доведено, що запропонований спосіб має обчислювальну складність O($m$), істотно меншу, ніж відомі методи - O($m^2$). Запропонований спосіб орієнтований на апаратну реалізацію та паралельні обчислення бітів кореня. Проведені дослідження показали, що використання запропонованого способу забезпечує виграш у часі на 2-3 порядки.

## Література


1. Menezes A. Elliptic Curve Public Key Cryptosystems. / Menezes A. - Kluwer Academic Published.-1993 - 422 p.
2. Tonelli A. Bemerkung über die Auflösung quadratischer Congruenzen / Tonelli A. // Göttinger Nachrichten.- 1891.- PP.344-346.
3. Boneh D. Identity-based encryption from the Well pairing / Boneh D., Franclin M. // SIAM Journal of Computing.- Vol.23.- 2003.- № 3.- PP. 354-368.
4. Barreto P.S.L.M. Efficient Computation of Root in Finite Fields / Barreto P.S.L.M., Voloch J.F. // Designs, Codes and cryptography.- 2006.- № 39.- pp. 275-280.
5. Cipolla M. Un metodo per la risolutione della congruenza di secondo grado / Cipolla M. // Rendiconto dell'Accademia Scienze Fisiche e Matematiche.- Napoli.-1903.- Ser.3- Vol.IX.- PP.154-163.
6. Aldeman L.M. On taking root in finite fields / Aldeman L.M., Manders K. Miller G. // Proc. 18-th IEEE Symposium on Foundations of Computer Science.-1977.-PP.175-177.
7. Марковський О.П. Спосіб прискореного обчислення коренів на полях Галуа GF($2^m$) / Марковський О.П., Виноградов Ю.М., Косейкіна Г.С. // Вісник Національного технічного університету України "КПІ" Інформатика, управління та обчислювальна техніка, – Київ: ВЕК+ – 2012 – № 56. с.165-168.