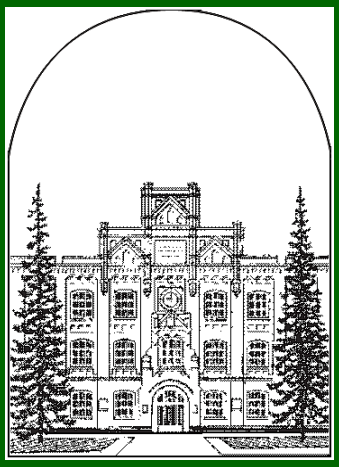
ВІСНИК

НАЦІОНАЛЬНОГО ТЕХНІЧНОГО УНІВЕРСИТЕТУ УКРАЇНИ
"Київський політехнічний інститут"

Інформатика, управління та обчислювальна техніка



61

2014

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 61

Київ “ВЕК+” 2014

УДК 004

Вісник НТУУ «КПІ». Інформатика, управління та обчислювальна техніка: Зб. наук. пр. – К.: Век+, – 2014. – № 61. – 179 с.

Рекомендований до друку Вченою радою факультету інформатики та обчислювальної техніки, протокол № 1 від 28.07.2014

Головний редактор: Луцький Г.М., д.т.н., проф.

Заст. гол. ред.: Стіренко С.Г., к.т.н., доц.
Відповідальний секретар: Поспішний О.С.

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

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.

Журнал індексується наступними наукометричними базами даних: Directory of Open Access Journals (DOAJ), Російський індекс наукового цитування (РІНЦ), Google Scholar.

Журнал входить до переліку наукових фахових видань МОН України.

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 26.05.2014. Формат 60×84 1/8. Гарнітура Times. Папір офсетний № 1.
Наклад 150 прим.
Надруковано в ЗАТ “ВІПОЛ”, 03151 м.Київ, вул. Волинська, 60.

© Національний технічний університет України “КПІ”, 2014

ЗМІСТ

Згуровский М.З., Павлов А.А., Мисюра Е.Б. Минимизация лексикографического критерия для допустимого расписания на параллельных приборах с произвольными директивными сроками 4

Кулаков Ю.О.,Клименко І.А. Організація багаторівневої пам'яті в реконфігурованих обчислювальних системах ……..…….…….. 18

Павлов А.А., Халус Е.А. Составление допустимого расписания выполнения работ на одном приборе оптимального по критерию минимизации суммарного опережения работ……. 27

Стіренко С.Г. Реалізація методу визначення моменту відкладення пересилання даних в комп'ютерних кластерних системах ……..……...…….. 35

Коваль О.В.,Зайцева К.А. Верифікація комп'ютерної моделі системи інформаційного управління… 43

Боброва Н.Л. Обоснование использования комплекса диагностических методик для оценки психофизиологического состояния человека ...……..…….… 49

Роковой А.П., Габинет А.В., Петрук В.А. Метод локализации мультимедийного трафика в одноранговых сетях ……..… 54

Згуровский М.З., Павлов А.А., Мисюра Е.Б., Мельников О.В., Лисецкий Т.Н. Методология построения четырехуровневой модели планирования, принятия решений и оперативного планирования в сетевых системах с ограниченными ресурсами……..…… 60

Широчин С.С.,Сулема Є.С. Спосіб стеганографічного захисту даних в аудіо-файлах на основі комплементарного образу……..……. 85

Павлов А.А., Жданова О.Г., Сперкач М.О. Задача составления допустимого расписания с максимально поздним моментом запуска выполнения идентичными параллельными приборами работ с общим директивным сроком ……..…. 93

Марковський О.П., Невдащенко М.В., Білашевська А.М. Захищена реалізації фільтрації зображень в grid-системах ……..... 105

Демчинский В.В. Особенности применения механизмов формирования и ограничения трафика…... 110

Денисюк А.В., Кавицкая В.С., Любченко В.В. Определение адекватной модели для представления знаний в современных информационных системах …….. 114

Бойко Ю.В., Глибовець М.М., Єршов С.В., Кривий С.Л., Погорілий С.Д., Ролік О.І., Теленик С.Ф., Ясочка М.В. Управління високопродуктивними ІТ-інфраструктурами……. 120

Михайленко В. В., Виноградов Ю. М., Бабенко А. А. Комп'ютерний аналіз напівпровідникових перетворювачів електроенергії з багаторозгалуженими структурами …… 143

Ролик А.И. Комплексный подход к управлению уровнем ИТ-услуг в корпоративных ИТ-инфраструктурах ……... 149

Кулаков Ю.А., Диброва М.А. Способ формирования виртуальной структуры grid, ориентированной на многопутевую маршрутизацию……... 162

Дорогий Я.Ю. Алгоритм структурної оптимізації нейронної мережі …….…… 169

Салапатов В.І. Моделювання, верифікація та розробка програм ……... 174

Відомості про авторів ..……..…... 178

Summary ……...……. 179

УДК 519.854.2

ЗГУРОВСКИЙ М.З.,
ПАВЛОВ А.А.,
МИСЮРА Е.Б.

МИНИМИЗАЦИЯ ЛЕКСИКОГРАФИЧЕСКОГО КРИТЕРИЯ ДЛЯ ДОПУСТИМОГО РАСПИСАНИЯ НА ПАРАЛЛЕЛЬНЫХ ПРИБОРАХ С ПРОИЗВОЛЬНЫМИ ДИРЕКТИВНЫМИ СРОКАМИ

В статье разработаны ПДС-алгоритмы решения задач составления расписаний выполнения независимых работ с директивными сроками параллельными приборами равной и разной производительности по векторному (лексикографическому) либо скалярному критерию максимизации минимального момента начала работы приборов. На основе созданных ранее признаков оптимальности допустимых решений разработаны алгоритмы, реализующие полиномиальную составляющую ПДС-алгоритма и полиномиальную аппроксимацию точного алгоритма решения рассматриваемых задач – эвристические или приближенные алгоритмы с оценкой отклонения от оптимального решения.

In the paper the PDC-algorithms are developed for scheduling problems of processing independent jobs with due dates on parallel machines with equal and different performance ratios with the vector (lexicographical) or scalar criterion of maximization the minimum moment of the processing start on machines. On the basis of previously created signs of optimality of the feasible solutions the algorithms are developed that implement the polynomial component of the PDC-algorithm and polynomial approximation of exact algorithm for the problems solution – heuristic or approximate algorithms with the estimate of deviation from the optimal solution.

Введение

В [1, 2, 3] изложен новый подход к решению *NP*-полных задач достаточно большой размерности, сформировавшийся как теория ПДС-алгоритмов. ПДС-алгоритмом называется алгоритм [1], состоящий из полиномиальной составляющей и экспоненциального подалгоритма либо полиномиальной аппроксимации точного алгоритма (приближенного или эвристического алгоритма), которые могут содержать условия декомпозиции исходной задачи на подзадачи меньшей размерности [1, гл. 3]. 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. Полиномиальная составляющая ПДС-алгоритма синтезируется таким образом, чтобы последовательная процедура конструирования допустимых решений была наиболее эффективной с точки зрения реализации *p*-условий (достаточных признаков оптимальности допустимых решений). Верхняя оценка сложности полиномиальной составляющей известна. Иногда экспоненциальная составляющая ПДС-алгоритма заменяется алгоритмом полиномиальной сложности, приводящим к приближенному (субоптимальному) решению [1].

В [4] 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.

Таким образом, расширяется класс задач, для которых могут быть построены ПДС-алгоритмы. Приведенные в данной статье задачи 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 известен, но его вычислительная сложность существенно выше вычислительной сложности полиномиальной составляющей ПДС-алгоритма, и при этом статистически значимо полиномиальная составляющая реализуется при произвольном моделировании индивидуальных задач.

В данной статье разработаны ПДС-алгоритмы решения задач 4.1 и 5.1 из [4]. Каждый из них включает как несколько алгорит-

мов, реализующих полиномиальную составляющую (p-составляющую), так и несколько алгоритмов, реализующих полиномиальную аппроксимацию точного алгоритма решения рассматриваемой задачи (предлагаются эвристические алгоритмы и приближенные алгоритмы с оценкой отклонения от оптимального решения). Это связано с тем, что в зависимости от исходных данных конкретный алгоритм дает лучшее решение, чем другие (заранее неизвестно, какой алгоритм окажется лучшим).

Задача 4.1 [4]. Приборы равной производительности

Определение 1. Расписание называется допустимым, если в нем нет запаздывающих работ. Работа называется допустимой, если она выполняется без запаздывания.

Постановка задачи. 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 приборов произвольны. Необходимо построить допустимое расписание, максимизирующее один из двух критериев:

Критерий 1.

$$r_{j_1} = \max\{\min_{i=\overline{1,m}} r_i\}$$
$$r_{j_l} = \max\{\min_i r_i, i=\overline{1,m}, i \neq i_k, k=\overline{1,l-1}\}, l=\overline{2,m}, \quad (1)$$

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

Определение 2. Допустимое расписание, у которого моментами запуска приборов являются $r_{j_l}, l=\overline{1,m}$, называется оптимальным по прямому лексикографическому порядку.

Примечание. Допустимое расписание называется оптимальным по обратному лексикографическому порядку, если для него выполняется:

$$r_{j_1} = \max\{\max_{i=\overline{1,m}} r_i\}$$
$$r_{j_l} = \max\{\max_i r_i, i=\overline{1,m}, i \neq i_k, k=\overline{1,l-1}\}, l=\overline{2,m}.$$

Критерий 2. Допустимое расписание является оптимальным по критерию 2, если для него выполняется:

$$r_{j_1} = \max\{\min_{i=\overline{1,m}} r_i\}. \quad (2)$$

Признаки оптимальности допустимого решения по векторному критерию (1).

Обозначим через I множество всех работ $\{i$: $i = \overline{1,n}\}$.

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

В статье [4] сформулированы и доказаны достаточные признаки оптимальности №1 и №2, которые позволили сконструировать полиномиальную составляющую ПДС-алгоритма для рассматриваемой задачи.

Признак 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_{j_1} в (1). Пусть выполняются следующие неравенства: $d_i + l_k > d_k$, $i = \overline{1,m}$, $k = \overline{i+1,m}$. Тогда назначаем на k-й прибор, $k = \overline{2,m}$, первой работу с индексом k и моментом запуска прибора $r_k = d_k - l_k$.

Утверждение 1 [4]. Произвольное допустимое расписание, для которого выполнен п. 1, является оптимальным по критерию (1).

Признак 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_{j_1} в (1)). Пусть $k_2 - 1$ – максимальное натуральное число, для которого выполняются неравенства:

$$d_1 + \sum_{i=2}^{l} l_i \leq d_l, l=\overline{1,k_2-1}. \quad (3)$$

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

$$\min\{d_1 + \sum_{i=2}^{k_2-1} l_i + l_{k_2+1}, d_{k_2} + l_{k_2+1}\} \leq d_{k_2+1}$$

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:

$$\min\{d_1 + \sum_{i=2}^{k_2-1} l_i, d_{k_2} + l_{k_2+1}\} \le d_{k_2+1}, \qquad (4)$$

$$\max\{d_1 + \sum_{i=2}^{k_2-1} l_i, d_{k_2} + l_{k_2+1}\} > d_{k_2+1}. \qquad (5)$$

Тогда работа с индексом $k_2 + 1$ назначается на прибор, которому соответствует минимум в (4). Если неравенство (4) выполнено, а неравенство (5) 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 (4), (5)). 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 $k_3 + i$ 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 I_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 I_1 в порядке их назначения на приборы. Тогда для каждой работы с индексом i ($i = \overline{2, k_m \vee k_l}$) должно выполняться

$$d_i - C_i < l_p, \; p = \overline{i+1, k_m \vee k_l}, \qquad (6)$$

где C_i – момент окончания выполнения i-й работы. Тогда имеет место утверждение:

Утверждение 2 [4]. Произвольное допустимое расписание, для которого выполнен п. 1 либо п. 2 и условие (6), является оптимальным по критерию (1), т.е.

$$\begin{pmatrix} r_1 \\ r_2 \\ \vdots \\ r_m \end{pmatrix} \succcurlyeq \forall \begin{pmatrix} r_{q_1} \\ r_{q_2} \\ \vdots \\ r_{q_m} \end{pmatrix} \qquad (7)$$

в соответствии с предпорядком – лексикографическим порядком, где $(r_{q_1} \ldots r_{q_m})^{\mathrm{T}}$ – вектор моментов запуска приборов произвольного допустимого расписания, компоненты которого расположены в прямом лексикографическом порядке, т.е. первая компонента наименьшая, вторая – следующая по величине и т.д., а $(r_1 \ldots r_m)^{\mathrm{T}}$ соответствует допустимому расписанию, у которого выполнен п. 1 либо п. 2 и условие (6) (компоненты вектора $(r_1 \ldots r_m)^{\mathrm{T}}$ по построению расположены в соответствии с прямым лексикографическим порядком).

Условие (7) соблюдается, если $r_1 > r_{q_1}$, либо если $r_1 = r_{q_1}$ и $r_2 > r_{q_2}$, либо $r_1 = r_{q_1}$ и $r_2 = r_{q_2}$, но $r_3 > r_{q_3}$, и т.д., либо $r_j = r_{q_j}, j = \overline{1, m}$.

Аналогично (7) по прямому лексикографическому порядку сравниваются два произвольных вектора.

Примечание. Если в соответствии с п. 2 загруженными оказались l приборов ($l < m$), то в (7) r_{l+i}, $j = \overline{1, m-l}$, формально принимают значения $+\infty$.

Примечание. Если <u>все</u> работы распределены по l приборам, $l < m$, то построено оптимальное расписание по критерию (1).

Признак оптимальности допустимого решения по скалярному критерию (2).

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 (2).

Утверждение 3. Произвольное допустимое расписание, у которого $r_i \ge r_1$, $j = \overline{2, m}$, является оптимальным по критерию (2).

ПДС-алгоритм решения задачи 4.1 [4] по критерию (1) (алгоритм А)

p-составляющая ПДС-алгоритма. Полиномиальная составляющая ПДС-алгоритма состоит из пяти алгоритмов, которые реализуются последовательно в произвольном порядке. Первый алгоритм, построивший допустимое расписание, для которого выполнен признак опти-

мальности №1 или №2, реализует оптимальное решение задачи по критерию (1).

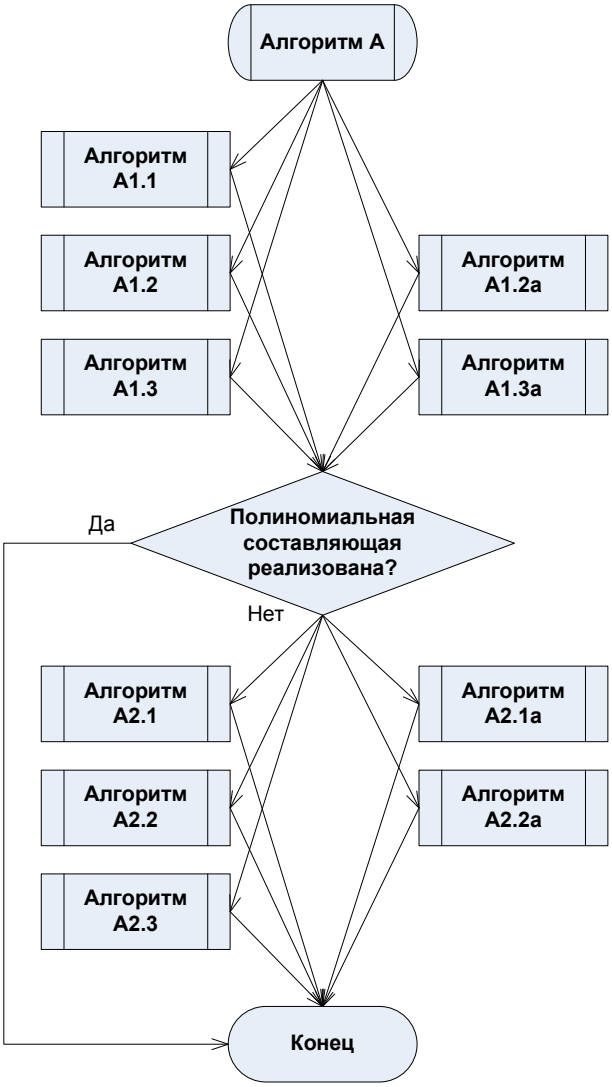


Рис. 1. Схема ПДС-алгоритма А

Алгоритм А1.1

Алгоритм реализует полиномиальную составляющую для построения допустимого расписания, если построено начальное расписание в соответствии с признаком оптимальности допустимого решения №1 либо №2. В результате исходное множество работ I разбивается на два подмножества: I_1 – множество работ, предварительно назначенных на выполнение, и I_1' – множество еще не назначенных работ, которые упорядочиваются по возрастанию значений $d_i - l_i$ (последовательность $\sigma^{\text{уп}}$). Для каждого прибора k известен момент освобождения после выполнения назначенных работ C_k. Дальнейшее назначение работ из последовательности $\sigma^{\text{уп}}$ на выполнение осуществляется следующим образом: каждая очередная работа назначается на выполнение на прибор с минимальным моментом освобождения. Если работа не запаздывает, итерация окончена, пересчитываем момент освобождения, переходим к следующей работе из упорядоченной последовательности работ $\sigma^{\text{уп}}$. Если работа запаздывает, полиномиальная составляющая не реализована, алгоритм окончен. Если в результате такого распределения все работы выполняются без запаздывания, то полиномиальная составляющая реализована.

Алгоритмы А1.2, А1.2а

Алгоритмы применяются для случая, когда начальное расписание реализует признак оптимальности №1.

Алгоритмы А1.2, А1.2а – это алгоритмы А2.1, А2.1а (см. ниже на стр. 7) с учетом следующего ниже примечания А.

Примечание А. Алгоритм распределяет все работы, кроме работ, принадлежащих начальному расписанию. После выполнения алгоритма проверяется выполнение следующего условия: на каждом приборе момент запуска прибора не должен быть меньше, чем момент окончания выполнения первой работы на этом приборе в соответствии с начальным расписанием. Если это условие не выполняется, полиномиальная составляющая не реализована данным алгоритмом. Иначе для получения оптимального расписания необходимо сдвинуть полученное приближенным алгоритмом расписание на этом приборе влево так, чтобы момент начала выполнения первой работы приближенного расписания на каждом приборе совпадал с моментом окончания выполнения первой работы начального расписания на этом приборе (чтобы все приборы работали без прерываний).

Алгоритмы А1.3, А1.3а

Алгоритмы применяются для случая, когда начальное расписание реализует признак оптимальности №1.

Алгоритмы А1.3, А1.3а – это алгоритмы А2.2, А2.2а (см. ниже на стр. 9) с учетом примечания А (стр. 7).

Полиномиальная аппроксимация точного алгоритма решения задачи состоит из пяти алгоритмов, задача решается каждым из них и выбирается решение, наилучшее в соответствии с прямым лексикографическим порядком (7).

Алгоритмы А2.1, А2.1а

В основу алгоритма А2.1 положен принцип последовательного группового назначения с использованием локально оптимального обратного лексикографического порядка.

Алгоритм состоит из ряда однотипных итераций. Каждая итерация, кроме первой, включает предварительный этап и назначение работ на выполнение, которое осуществляется по групповому принципу (группами из m работ – по одной работе на каждый прибор). Назначение работ осуществляется в обратном порядке: на первой итерации алгоритма на каждый прибор назначается по одной работе, момент окончания выполнения которой совпадает с ее директивным сроком, а моменты запуска их на выполнение соответствуют глобальному оптимуму по обратному лексикографическому порядку (для каждой работы находится прибор, момент запуска которого для этой работы является самым поздним; первой назначается работа на прибор с самым поздним моментом запуска, затем следующая работа – на прибор со следующим по величине моментом запуска, и так, пока не будет назначено m работ на m приборов).

Примечание. Множество приборов, из которых находится прибор со следующим по величине моментом запуска, не включает в себя приборы, на которых ранее были назначены работы на первой итерации.

На текущей итерации распределяется множество работ из еще не назначенных на приборы на предыдущих итерациях. На предварительном этапе находятся работы, для каждой из которых существует не менее одного прибора, на котором она является допустимой (работа, назначенная на этот прибор, выполняется без срыва ее директивного срока). Сначала осуществляется назначение допустимых работ в соответствии с локально оптимальным обратным лексикографическим порядком (первой назначается допустимая работа на прибор с самым поздним моментом запуска, затем следующая работа на прибор со следующим по величине максимальным моментом запуска, и т.д., пока все допустимые работы не будут назначены).

Примечание. Множество приборов, из которых находится прибор со следующим по величине моментом запуска, не включает в себя приборы, на которых ранее были назначены работы на текущей итерации.

Если на все приборы назначено по одной работе, текущая итерация закончена.

В результате такого распределения могут остаться работы, для каждой из которых не существует приборов, на которых они допустимы. Они объединяются во вторую группу работ для текущей итерации. Распределение работ на приборы второй группы также осуществляется локально оптимально в соответствии с обратным лексикографическим порядком (аналогично распределению по приборам допустимых работ).

Примечание. В отличие от первой группы работ, назначение работы на прибор осуществляется с учетом сдвига момента запуска работы (и всего расписания работ, уже назначенных на этот прибор предыдущими итерациями) на величину срыва ее директивного срока.

Примечание. Множество приборов, из которых находится прибор со следующим по величине моментом запуска, не включает в себя приборы, на которых ранее были назначены работы на текущей итерации. Итерации продолжаются до получения моментов начала выполнения первых работ на каждом приборе.

Для повышения эффективности алгоритма А2.1 его последним шагом является алгоритм корректировки моментов начала выполнения работ АК.

Алгоритм корректировки АК.

1. Выбираем прибор с минимальным значением начала выполнения работ $r_{\min}$ и первую работу на этом приборе последовательно ставим на каждый прибор (если работа запаздывает, то сдвигаем ее и работы, назначенные на этот прибор на предыдущих итерациях, влево на величину запаздывания), определяя, на каком приборе у этой работы будет максимально поздний момент начала выполнения, и если он больше $r_{\min}$, то осуществляем перестановку ее на этот прибор. Определяем новые моменты начала выполнения работ.

2. Аналогичным образом, ищем следующий прибор с минимальным значением начала выполнения работ и пытаемся переставить первую работу с этого прибора на прибор, на котором момент ее начала выполнения будет больше.

3. Такие перестановки выполняются, пока возможно увеличение $r_{\min}$.

Примечание. Алгоритм корректировки является последним шагом всех алгоритмов полиномиальной аппроксимации точного алгоритма для задач 4.1 и 5.1.

Алгоритм А2.1а отличается тем, что последним шагом текущей итерации является проверка выполнения неравенства

$$\max_{i=1,m} r_i^{\text{тек}} - \min_{i=1,m} r_i^{\text{тек}} > K_{\text{эксп}},$$

где $r_i^{\text{тек}}$ – момент запуска прибора i после текущей итерации, $K_{\text{эксп}}$ – допустимая величина, определяемая экспериментально. Если неравенство выполняется, то реализуется алгоритм АК для расписания, построенного итерациями с

первой до текущей включительно, после чего выполняется переход к следующей итерации.

Обоснование группового назначения алгоритма А2.1. Логично на каждой итерации назначать работы на приборы локально оптимально по прямому лексикографическому порядку. Однако реализация такой стратегии приведет к полному перебору всех вариантов назначения. Поэтому эвристическое назначение выполняется локально оптимально по обратному лексикографическому порядку в предположении, что назначение это мало отличается от назначения локально оптимально по прямому лексикографическому порядку. Кроме того, эффективность такого назначения подтверждается следующими тремя утверждениями.

Утверждение 4. Рассмотрим текущую итерацию алгоритма и множество приборов J, для каждого из которых существует непустое множество I_j допустимых работ. Для $\forall j \in J$ допустимые работы из I_j на данной итерации назначаются на прибор j без запаздывания. Для каждого прибора $j \in J$ находим работу $i \in I_j$ с максимальным моментом начала выполнения, пусть это работа $i_j^{\max}$. Если все работы $i_j^{\max}$ различны, то моменты начала выполнения работ $i_j^{\max}$, назначенных на текущей итерации на прибор j, $\forall j \in J$, являются оптимальными как по обратному, так и по прямому лексикографическому порядку, по сравнению с любым другим набором допустимых работ, назначенных на приборы $j \in J$ в любом допустимом порядке.

Доказательство утверждения очевидным образом вытекает из правил нахождения работ $i_j^{\max}$.

Утверждение 5. Пусть для каждого прибора второй группы находится работа с самым поздним моментом начала выполнения, и эти работы различны на всех этих приборах. Тогда назначение на эти приборы одинаково как по прямому, так и по обратному лексикографическому порядку.

Доказательство вытекает из правил построения расписания.

Утверждение 6. Множество допустимых работ, назначаемых алгоритмом А2.1 на данной итерации, моменты начала выполнения которых являются оптимальными по обратному лексикографическому порядку, одновременно являются лучшими по прямому лексикографическому порядку по сравнению с моментами начала выполнения любого другого допустимого множества работ, назначенных на те же приборы, что и назначенные алгоритмом А2.1 на данной итерации.

Доказательство. Пусть $C = B \setminus A \neq \varnothing$, где A – множество допустимых работ, назначенных алгоритмом А2.1, B – произвольное множество допустимых работ, назначенных на те же приборы, что и работы, назначенные алгоритмом А2.1 из множества A, и при этом для назначенных работ множества C выполняется: наиболее поздний момент начала выполнения каждой допустимой работы из множества C не принадлежит множеству моментов начала выполнения работ из множества A, назначенных алгоритмом А2.1. Утверждение доказано.

Предположительно, при решении задачи алгоритмом А2.1 большее число работ выполняется точно в срок, чем при решении задачи другими аппроксимационными алгоритмами.

Алгоритм А2.2

Алгоритм А2.2 основан на построении в явном виде локально оптимального обратного лексикографического порядка.

Алгоритм состоит из первой и ряда однотипных итераций. Первая итерация полностью совпадает с первой итерацией алгоритма А2.1.

На текущей итерации распределяется множество работ из еще не назначенных на приборы на предыдущих итерациях. Для каждой работы находим момент запуска на каждый прибор с учетом сдвига всего расписания, назначенного на этот прибор, на величину запаздывания этой работы (если оно существует). Далее назначение работ на приборы осуществляется оптимально в соответствии с обратным лексикографическим порядком (сначала назначается работа на прибор с максимально поздним моментом запуска, затем следующая работа на прибор со следующим максимально поздним моментом запуска и т.д., пока не будут назначены работы на каждый прибор).

Примечание. Множество приборов, из которых находится прибор со следующим по величине моментом запуска, не включает в себя приборы, на которых ранее были назначены работы на текущей итерации.

Итерации продолжаются до получения моментов начала выполнения первых работ на каждом приборе. Последним шагом является алгоритм корректировки АК.

Алгоритм А2.2а отличается тем, что последним шагом текущей итерации является проверка выполнения неравенства

$$\max_{i=\overline{1,m}} r_i^{\text{тек}} - \min_{i=\overline{1,m}} r_i^{\text{тек}} > K_{\text{эксп}}$$

и реализация алгоритма АК, если оно выполняется.

Утверждение 7. Пусть для каждого прибора найдены работы с самым поздним моментом начала выполнения, и пусть для всех приборов эти работы различны. Тогда распределение одинаково как по прямому, так и по обратному лексикографическому порядку.

Доказательство вытекает из правил построения расписания.

Алгоритм А2.3

Приближенный алгоритм А2.3 – алгоритм полиномиальной сложности для построения допустимого расписания, у которого предварительно работы из множества всех работ I назначены в соответствии с признаком оптимальности допустимого решения №1 либо №2. Если такое расписание построить не удалось, оно строится по признаку оптимальности №2 без учета условий (4), (5) или их аналогов, а также условия (6), при этом строгие неравенства не обязательны: $d_1 - l_1 \le d_2 - l_2 \le \quad \le \ldots \le d_n - l_n$. Алгоритм А2.3 является эвристическим.

В результате построения начального расписания исходное множество работ I разбивается на два подмножества: I_1 – множество работ, предварительно назначенных на выполнение, и I_1' – множество еще не назначенных работ, которые упорядочиваются по неубыванию значений $d_i - l_i$ (последовательность $\sigma^{\text{уп}}$, строгие неравенства не обязательны). Для каждого прибора k известен момент освобождения после выполнения назначенных работ C_k. Дальнейшее назначение работ осуществляется следующим образом. Выбираем очередную работу из последовательности $\sigma^{\text{уп}}$ и назначаем на прибор с минимальным временем освобождения. Если работа не запаздывает, итерация окончена, пересчитываем момент освобождения прибора, переходим к следующей работе. Если работа запаздывает, то она назначается на каждый прибор, все назначенные на этот прибор работы сдвигаются влево на величину запаздывания назначаемого задания. Получаем m расписаний, которым соответствуют m векторов текущих моментов запуска приборов. Выбираем наилучший вектор по прямому лексикографическому порядку (7) и назначаем запаздывающую работу на прибор, которому соответствует наилучший по (7) вектор моментов запуска, с учетом сдвига расписания на величину срыва директивного срока. Последним шагом является алгоритм корректировки АК.

ПДС-алгоритм решения задачи 4.1 [4] по критерию (2) (алгоритм Б)

Критерий (2) описан на стр. 5, признаки оптимальности допустимого решения – на стр. 6.

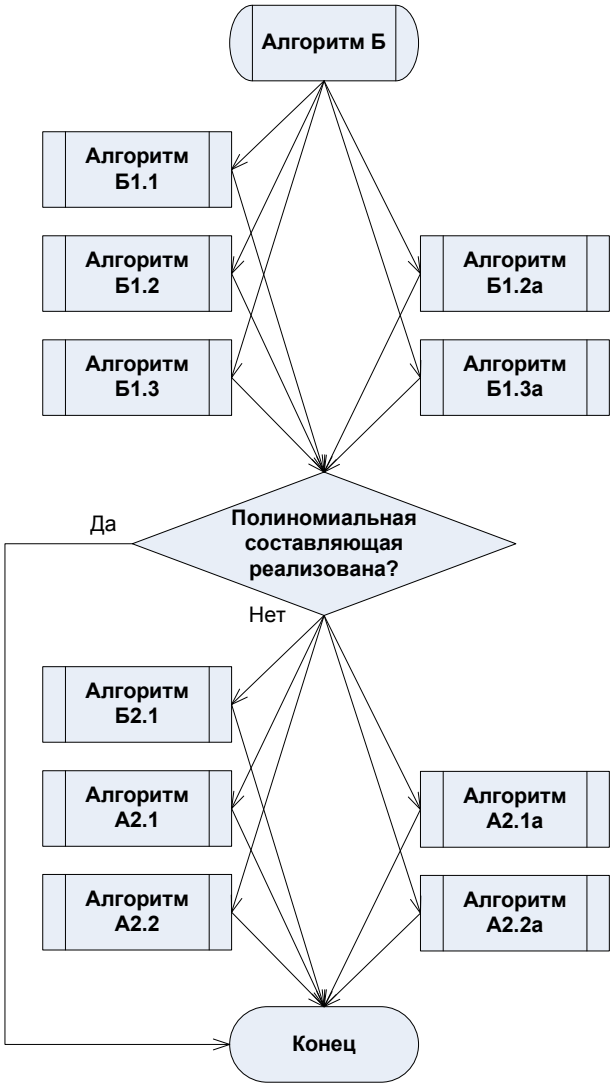


Рис. 1. Схема ПДС-алгоритма Б

р-составляющая ПДС-алгоритма. Полиномиальная составляющая ПДС-алгоритма состоит из пяти алгоритмов, которые реализуются последовательно в произвольном порядке. Первый алгоритм, построивший допустимое расписание, для которого выполнен признак оптимальности, реализует оптимальное решение задачи по критерию (2).

Алгоритм Б1.1

Алгоритм реализует полиномиальную составляющую для построения допустимого расписания, если начальное расписание формируется в соответствии с признаком оптимальности №2 для критерия (1) без учета выполнения условий (4), (5) или их аналогов, а также условия (6), при этом строгие неравенства не обязательны: $d_1 - l_1 \le d_2 - l_2 \le \ldots \le d_n - l_n$. В результате исходное множество работ I разбивается на два под-

множества: I_1 – множество работ, предварительно назначенных на выполнение, и I_1' – множество еще не назначенных работ, которые упорядочиваются по неубыванию значений d_i – – l_i (последовательность $\sigma^{\text{уп}}$). Для каждого прибора k известен момент освобождения после выполнения назначенных работ C_k. Дальнейшее назначение работ осуществляется следующим образом. Выбираем очередную работу из последовательности $\sigma^{\text{уп}}$ и назначаем на прибор с минимальным временем освобождения, если она на нем оказывается допустимой (если прибор с минимальным временем освобождения – первый, то на него назначается только допустимая работа, если на первом приборе работа запаздывает, то она назначается на другой прибор со следующим по величине минимальным временем освобождения). Если на приборе с минимальным временем освобождения она запаздывает, то расписание всех назначенных на приборе работ сдвигается влево на величину ее запаздывания и проверяется новый момент запуска: если $r^{\text{нов}} \geq r_1$, работу назначаем, если $r^{\text{нов}} < r_1$, полиномиальная составляющая этим алгоритмом не реализована, конец алгоритма. В противном случае, переходим к аналогичному распределению следующей работы из последовательности $\sigma^{\text{уп}}$, и т.д. Алгоритм завершается построением оптимального расписания по критерию (2) либо установлением факта невозможности его построения этим алгоритмом.

Алгоритмы Б1.2, Б1.2а

Алгоритмы Б1.2, Б1.2а – это алгоритмы А2.1, А2.1а (см. выше) с учетом следующего ниже примечания Б.

Примечание Б. Начальное расписание состоит из первой работы, назначенной на первый прибор с моментом запуска r_1 (см. признак оптимальности по критерию (2)). После выполнения алгоритма, который распределяет все работы, кроме первой, проверяются следующие условия: в полученном расписании момент начала выполнения первой работы приближенного расписания на первом приборе должен быть не меньше, чем d_1 (момент окончания выполнения первой работы начального расписания на первом приборе, назначенной в соответствии с признаком оптимальности), а момент запуска начальных работ приближенного расписания на остальных приборах должен быть не меньше, чем r_1. Если эти условия выполняются, то для получения оптимального расписания необходимо сдвинуть влево расписание работ на первом приборе, полученное приближенным алгоритмом, так, чтобы момент начала выполнения первой работы приближенного расписания совпадал с моментом окончания выполнения первой работы начального расписания d_1. Если момент начала выполнения первой работы приближенного расписания на первом приборе меньше, чем d_1, а моменты запуска остальных приборов не меньше, чем r_1, то первую работу приближенного расписания на первом приборе поочередно ставим на остальные приборы и переставляем ее на тот прибор, на котором момент начала выполнения этой работы будет самым большим и не меньше, чем r_1. Если такой прибор найти не удалось, полиномиальная составляющая не реализована. Если же первая работа переставлена на другой прибор, проверяем условие: если момент запуска второй работы на первом приборе больше, чем d_1, то путем сдвига расписания на первом приборе до момента d_1 получаем оптимальное расписание. Если же в приближенном расписании момент запуска второй работы на первом приборе меньше, чем d_1, пытаемся перенести на другой прибор вторую работу приближенного расписания на первом приборе. Описанная процедура повторяется, пока либо не получено оптимальное расписание, либо полиномиальная составляющая этим алгоритмом не реализуется.

Алгоритмы Б1.3, Б1.3а

Алгоритмы Б1.3, Б1.3а – это алгоритмы А2.2, А2.2а (см. выше) с учетом примечания Б (стр. 11).

Полиномиальная аппроксимация точного алгоритма решения задачи содержит пять алгоритмов – приближенные алгоритмы А2.1, А2.1а, А2.2, А2.2а (см. выше), в силу того, что производится корректировка моментов начала выполнения работ алгоритмом АК (стр. 8), а также алгоритм Б2.1. Задача решается каждым из этих алгоритмов и выбирается решение, которому соответствует наилучший $\max\{\min_j r_j\}$, $j = \overline{1,n}$.

Алгоритм Б2.1

Алгоритм Б2.1 является модификацией алгоритма Б1.1 (см. выше на стр. 10) со следующим отличием. Если новый момент запуска прибора $r^{\text{нов}} < r_1$, запаздывающая работа ставится на первый прибор и сдвигается расписание первого прибора на величину ее запаздывания, после чего эта работа назначается либо на первый прибор, либо на прибор с текущим моментом запуска $r^{\text{нов}}$, в зависимости от того, где больше текущий момент запуска. Для дальнейших итераций этот

прибор становится аналогом первого. Процедура назначения продолжается аналогично до распределения всех работ. На последнем шаге алгоритма применяем алгоритм корректировки АК.

Оценка отклонения решения от оптимального. Алгоритм полиномиальной аппроксимации является приближенным по критерию (1) или (2), если реализуется начальная часть признака оптимальности (реализуется назначение начального расписания). Тогда существует оценка отклонения решения, полученного приближенным алгоритмом, от оптимального.

По критерию (1): с этой целью производим сравнение векторов по условию (7).

Примечание. Пусть $r_l > r_{q_l}$, где l – номер компоненты, для которой достигается первая положительная разность, т.е. $r_j = r_{q_j}, j = \overline{1, l-1}$. Тогда знаки разностей остальных компонент ($j = \overline{l+1, m}$) могут быть любыми. Лицо, принимающее решение, анализирует полученное решение и выбирает оптимальное или приближенное решение, исходя из практических соображений.

По критерию (2): оценка задается разностью $r_1 - r_{q_1} \geq 0$ (все алгоритмы Б2.1, А2.1, А2.1а, А2.2, А2.2а являются приближенными по критерию (2)).

Примечание. Для критерия (1) алгоритмы А2.1, А2.1а, А2.2, А2.2а являются приближенными, если существует начальное расписание в соответствии с признаком оптимальности №1. Алгоритм А2.3 является приближенным, если существует начальное расписание в соответствии с признаком оптимальности №1 или №2.

Задача 5.1 [4]. Приборы разной производительности

Постановка задачи. 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, длительности работ не связаны между собой (возможна ситуация, что одна работа выполняется быстрее другой на одном приборе и медленнее другой на другом приборе). Работы должны быть выполнены к директивным срокам d_i. Моменты запуска приборов произвольны. Необходимо построить допустимое расписание, оптимальное по критерию (1) либо (2).

Обозначим через I множество всех работ $\{i\colon i = \overline{1, n}\}$.

Для задачи 5.1 очевидным образом обобщен признак оптимальности допустимого расписания №1, приведенный для задачи 4.1 [4].

Конструирование признака оптимальности №1 допустимого расписания по критерию (1).

п. 1. Рассмотрим следующую монотонно неубывающую последовательность чисел:

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

$$d_{i_1} - l_{i_1}^{j_1} = \min_i \{(d_i - \min_j l_i^j), i = \overline{1, n}, j = \overline{1, m}\} \quad (9)$$

$$d_{i_p} - l_{i_p}^{j_p} = \min_i \{(d_i - \min_j l_i^j), i = \overline{1, n}, j = \overline{1, m},$$
$$i \notin \{\overline{i_1, i_{p-1}}\}, j \notin \{\overline{j_1, j_{p-1}}\}\}, p = \overline{2, m}$$

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

$$d_{i_l} + l_{i_p}^{j_l} > d_{i_p} \ \forall p = \overline{l+1, m}, l = \overline{1, m-1}. \quad (10)$$

Тогда имеет место утверждение:

Утверждение 8 [4]. 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 (1).

Эвристический признак оптимальности №2 по критерию (1): построение начального расписания.

Пусть $\sigma^{\text{уп}}$ – последовательность работ i_1, i_2, …, i_n, определяемая формулами (9) без учета выполнения условия $j \notin \{\overline{j_1, j_{p-1}}\}, p = \overline{2, m}$, и если пар ($i_p$, j_p) несколько, то выбирается любая. Припишем каждой i_l-й работе j_l-й прибор. Первую работу i_1 назначаем на приписанный к ней прибор j_1 с моментом начала выполнения r_{j_1}. Переходим к следующей работе из $\sigma^{\text{уп}}$ и проверяем, может ли она быть назначена на прибор j_1, и если она не запаздывает, назначаем ее на прибор j_1, и т.д., пока не придем к запаздывающей работе. Запаздывающая работа назначается на приписанный к ней прибор с моментом окончания, равным ее директивному сроку. Назначение следующей работы из $\sigma^{\text{уп}}$ осуществляется, начиная с прибора j_1, в той же последовательности, в которой шло назначение на приборы первых работ. Если ни на один из этих приборов работу назначить без запаздывания не удалось, назначаем эту работу на приписанный к ней прибор с моментом окончания, равным ее директивному сроку. Если на этот прибор уже назначена первая работа, все неназначенные работы из $\sigma^{\text{уп}}$, включая текущую, и все приборы, не содержащие назначенных работ, переупорядочиваются в соответствии с

правилами построения $\sigma^{уп}$. Получаем новую последовательность $\sigma^{уп}$. Выбираем следующую работу из $\sigma^{уп}$ и назначаем на приписанный к ней прибор. Дальше эта процедура повторяется до момента назначения первой работы на последний прибор.

Дополнительное условие 1. Моменты запуска всех приборов не меньше, чем $r_{i_1} = d_{i_1} - l_{i_1}^{j_1}$.

Дополнительное условие 2. Начальное расписание удовлетворяет условию: в соответствии с очередностью назначения первых работ, моменты запуска приписанных к ним приборов реализуют числовую неубывающую последовательность.

Примечание. Дополнительное условие 2 включает в себя дополнительное условие 1.

Конструирование признака оптимальности №1 допустимого расписания по критерию (2).

На прибор j_1 назначена работа i_1 с моментом запуска $r_{j_1} = d_{i_1} - l_{i_1}^{j_1}$ и существует допустимое расписание, у которого $r_k \geq r_{j_1}$, $k = \overline{1,m}$.

Для критерия (2) начальное расписание всегда существует, т.к. всегда можно реализовать (9) для нахождения только работы i_1 на приборе j_1.

Эвристический признак оптимальности №2 допустимого расписания по критерию (2).

Совпадает с эвристическим признаком оптимальности №2 для критерия (1) без учета выполнения дополнительного условия 2.

ПДС-алгоритм решения задачи 5.1 [4] по критерию (1) (алгоритм В)

p-составляющая ПДС-алгоритма. Полиномиальная составляющая ПДС-алгоритма состоит из семи алгоритмов, которые реализуются последовательно в произвольном порядке. Первый алгоритм, построивший допустимое расписание, для которого выполнен признак оптимальности, реализует оптимальное решение задачи по критерию (1).

Алгоритм В1.1

Алгоритм реализует полиномиальную составляющую для построения допустимого расписания, если построено начальное расписание в соответствии с признаком оптимальности №1. В результате исходное множество работ I разбивается на два подмножества: I_1 – множество работ, предварительно назначенных на выполнение, и I_1' – множество еще не назначенных работ, которые упорядочиваются по неубыванию <u>директивных сроков</u> (последовательность $\sigma^{уп}$; далее под $\sigma^{уп}$ понимается последовательность работ, упорядоченная по неубыванию

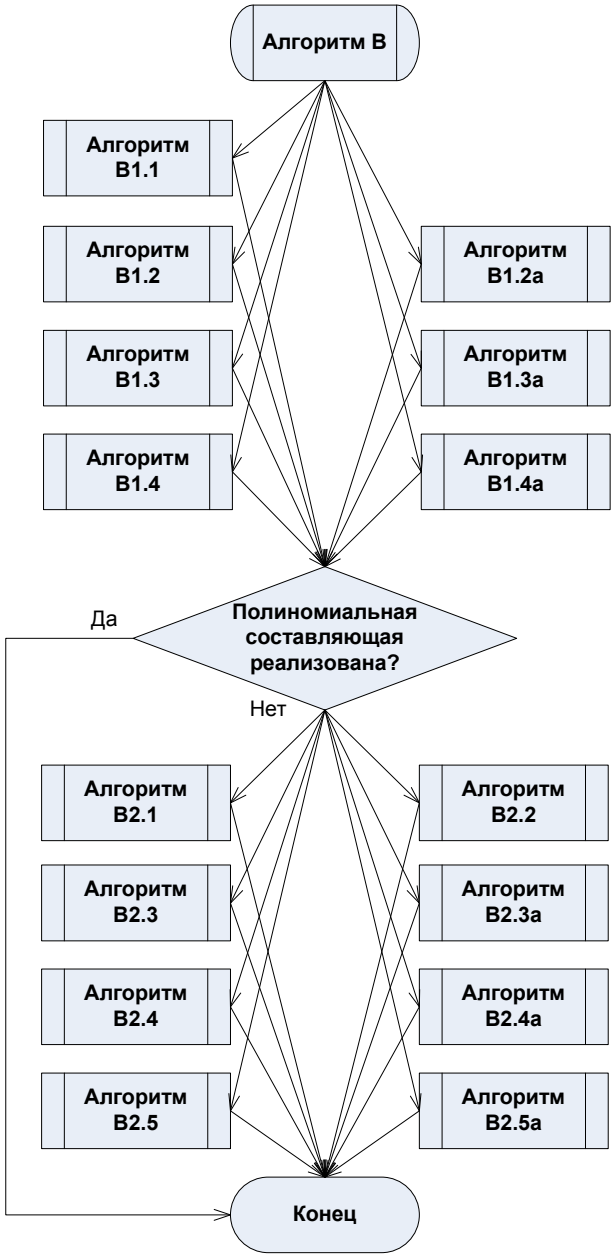


Рис. 3. Схема ПДС-алгоритма В

директивных сроков). Дальнейшее назначение неназначенных работ на выполнение осуществляется следующим образом: выбираем из последовательности $\sigma^{уп}$ очередную работу и назначаем на прибор, на котором опережение относительно ее директивного срока будет максимальным. Аналогичная процедура повторяется для всех работ. Если на очередной итерации алгоритма на всех приборах работа запаздывает, конец алгоритма, алгоритм не реализовал полиномиальную составляющую. В противном случае, назначение продолжается с работы, следующей за назначенной в последовательности $\sigma^{уп}$, пока не будут распределены все работы последовательности. Если все работы распределены без запаздывания, полиномиальная составляющая реализована.

Алгоритмы B1.2, B1.2a

Алгоритмы B1.2, B1.2a – это алгоритмы A2.1, A2.1a с учетом примечания A (стр. 7), при этом начальное расписание строится в соответствии с признаком оптимальности №1.

Алгоритмы B1.3, B1.3a

Алгоритмы B1.3, B1.3a – это алгоритмы A2.2, A2.2a с учетом примечания A (стр. 7), при этом начальное расписание строится в соответствии с признаком оптимальности №1.

Алгоритмы B1.4, B1.4a

Алгоритмы B1.4, B1.4a – это алгоритмы B2.5, B2.5a (см. ниже) с учетом примечания A (стр. 7), при этом начальное расписание строится в соответствии с признаком оптимальности №1.

Полиномиальная аппроксимация точного алгоритма решения задачи состоит из восьми алгоритмов, задача решается каждым из них и выбирается решение, наилучшее в соответствии с прямым лексикографическим порядком (7).

Алгоритм B2.1

Алгоритм B2.1 – алгоритм полиномиальной сложности для построения допустимого расписания, у которого предварительно работы из множества I (начальное расписание) назначены в соответствии с признаком оптимальности №1 для критерия (1). Алгоритм B2.1 – это алгоритм B1.1 со следующим изменением: если очередная работа из последовательности $\sigma^{\text{уп}}$ запаздывает на каждом приборе, то она последовательно ставится на каждый прибор, эта работа и все назначенные ранее на этот прибор работы сдвигаются влево на величину запаздывания назначаемого задания. Получаем m расписаний, которым соответствуют m векторов моментов запуска всех приборов. Выбираем наилучший вектор по прямому лексикографическому порядку (7) и назначаем текущую работу из $\sigma^{\text{уп}}$ на прибор, соответствующий наилучшему по (7) вектору начальных моментов запуска приборов. Последним шагом является алгоритм корректировки АК.

Алгоритм B2.2

Алгоритм B2.2 – алгоритм полиномиальной сложности для построения допустимого расписания, у которого предварительно работы из множества I назначены в соответствии с эвристическим признаком оптимальности №2 для критерия (1). Текущая итерация алгоритма B2.2 совпадает с текущей итерацией алгоритма B2.1.

Алгоритмы B2.3, B2.3a

Алгоритмы B2.3, B2.3a – это алгоритмы A2.1, A2.1a для приборов разной производительности.

Алгоритмы A2.1 и A2.1a, их теоретическое обоснование (утверждения 4–6), а также алгоритм корректировки АК, приведенные для параллельных приборов равной производительности, являются универсальными и справедливыми также для приборов разной производительности.

Алгоритмы B2.4, B2.4a

Алгоритмы B2.4, B2.4a – это алгоритмы A2.2, A2.2a для приборов разной производительности.

Алгоритмы A2.2 и A2.2a, их теоретическое обоснование (утверждение 7), а также алгоритм корректировки АК, приведенные для параллельных приборов равной производительности, являются универсальными и справедливыми также для приборов разной производительности.

Алгоритмы B2.5, B2.5a

Алгоритмы B2.5, B2.5a – это адаптация алгоритмов A2.1, A2.1a для приборов разной производительности с тем изменением, что на текущей итерации допустимые работы назначаются в порядке невозрастания длительностей их выполнения на самых выгодных для них допустимых приборах (минимально время выполнения).

Алгоритм B2.5 состоит из ряда однотипных итераций. Каждая итерация, кроме первой, включает предварительный этап и назначение работ на выполнение, которое осуществляется по групповому принципу (группами из m работ – по одной работе на каждый прибор). Назначение работ осуществляется в обратном порядке: на первой итерации алгоритма на каждый прибор назначается по одной работе, момент окончания выполнения которой совпадает с ее директивным сроком, а моменты запуска их на выполнение соответствуют глобальному оптимуму по обратному лексикографическому порядку (для каждой работы находится прибор, момент запуска которого для этой работы является самым поздним; первой назначается работа на прибор с самым поздним моментом запуска, затем следующая работа – на прибор со следующим по величине моментом запуска, и так, пока не будет назначено m работ на m приборов).

Примечание. Множество приборов, из которых находится прибор со следующим по величине моментом запуска, не включает в себя приборы, на которых ранее были назначены работы на первой итерации.

На текущей итерации распределяется множество работ из еще не назначенных на приборы на предыдущих итерациях. На предварительном этапе находятся работы, для каждой из которых существует не менее одного допустимого прибора (работа, назначенная на этот прибор, выполняется без срыва ее директивного срока). Для каждой допустимой работы определяется прибор, на котором ее длительность выполнения является минимальной. Выбирается та допустимая работа, у которой самая короткая длительность выполнения является максимальной, и назначается на прибор, на котором ее длительность выполнения минимальна. Эта работа и прибор исключаются из множества допустимых работ и неназначенных приборов, описанная процедура повторяется вплоть до назначения всех допустимых работ.

В результате такого распределения могут остаться работы, для каждой из которых не существует приборов, на которых они допустимы. Они объединяются во вторую группу работ для текущей итерации. Распределение работ второй группы на приборы осуществляется локально оптимально в соответствии с обратным лексикографическим порядком с учетом сдвига момента запуска работы (и всего расписания работ, уже назначенных на этот прибор предыдущими итерациями) на величину срыва ее директивного срока, пока не будет назначено по одной работе на каждый прибор.

Примечание. Множество приборов, из которых находится прибор со следующим по величине моментом запуска, не включает в себя приборы, на которых ранее были назначены работы на текущей итерации.

Итерации продолжаются до получения моментов начала выполнения первых работ на каждом приборе. Последним шагом является алгоритм корректировки АК.

Алгоритм В2.5а отличается тем, что последним шагом текущей итерации является проверка выполнения неравенства

$$\max_{i=\overline{1,m}} r_i^{\text{тек}} - \min_{i=\overline{1,m}} r_i^{\text{тек}} > K_{\text{эксп}}$$

и реализация алгоритма АК, если оно выполняется.

Обоснование эвристики назначения допустимых работ. С учетом того, что на текущей итерации в первую очередь назначается та допустимая работа, у которой самое короткое время выполнения является самым длинным, предполагается, что на последующих итерациях при назначении работ количество запаздывающих работ будет минимальным.

ПДС-алгоритм решения задачи 5.1 [4] по критерию (2) (алгоритм Г)

Критерий (2) описан на стр. 5, признаки оптимальности допустимого решения – на стр. 13.

р-составляющая ПДС-алгоритма. Полиномиальная составляющая ПДС-алгоритма состоит из семи алгоритмов, которые реализуются последовательно в произвольном порядке. Первый алгоритм, построивший допустимое расписание, для которого выполнен признак оптимальности, реализует оптимальное решение

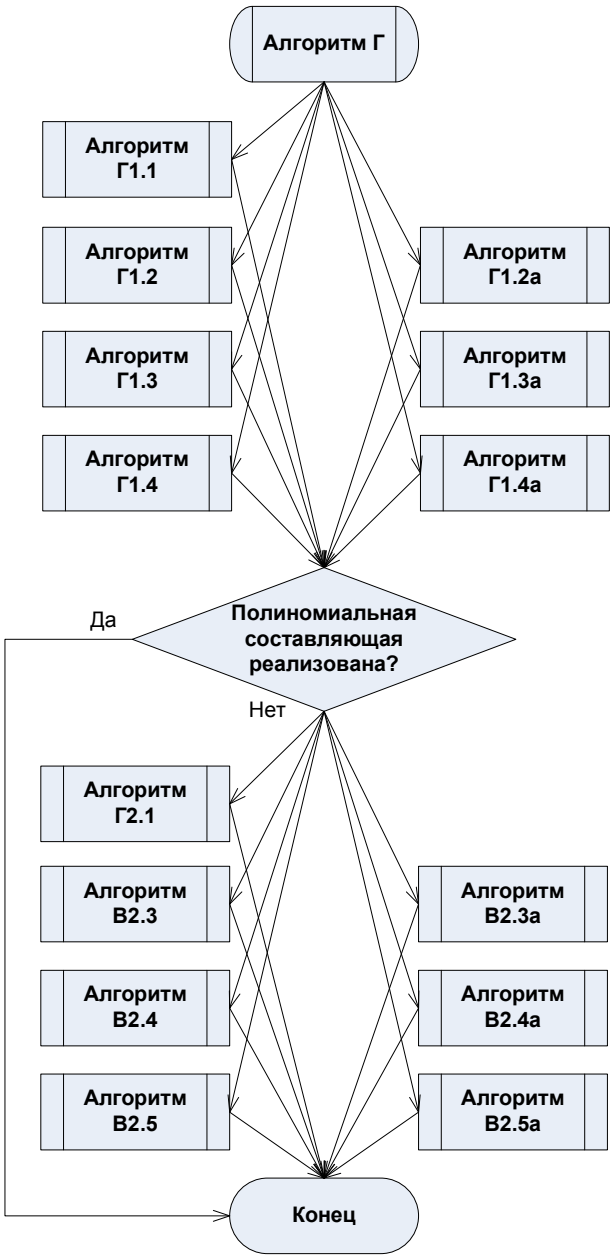


Рис. 4. Схема ПДС-алгоритма Г

Алгоритм Г1.1

Алгоритм реализует полиномиальную составляющую для построения допустимого расписания, если построено начальное расписание в соответствии с эвристическим признаком оптимальности №2 для критерия (2). В результате исходное множество работ I разбивается на два

подмножества: I_1 – множество работ, предварительно назначенных на выполнение, и I_1' – множество еще не назначенных работ, которые упорядочиваются по неубыванию директивных сроков (последовательность $\sigma^{\text{уп}}$). Дальнейшее назначение работ осуществляется следующим образом. Выбираем очередную работу из последовательности $\sigma^{\text{уп}}$ и назначаем на прибор, на котором достигается максимальное опережение относительно ее директивного срока или минимальное запаздывание, если все приборы для нее недопустимы (если этот прибор – j_1-й, то на него назначается только допустимая работа. Если на j_1-м приборе эта работа имеет минимальное запаздывание, то она назначается на следующий прибор с минимальным по величине запаздыванием). Если текущая работа назначается на прибор с минимальным запаздыванием, то расписание ранее назначенных на приборе работ, включая текущую, сдвигается влево на величину запаздывания, и для нового момента запуска проверяется условие: если $r^{\text{нов}} \geq r_{j_1}$, назначение работы произведено, если $r^{\text{нов}} < r_{j_1}$, полиномиальная составляющая этим алгоритмом не реализована, конец алгоритма. Процедура повторяется со следующей работой из последовательности $\sigma^{\text{уп}}$. Если все работы распределены без запаздывания, то получено оптимальное расписание.

Алгоритмы Г1.2, Г1.2а

Алгоритмы Г1.2, Г1.2а – это алгоритмы В2.3, В2.3а с учетом примечания В (см. ниже).

Примечание В. Алгоритм распределяет все работы, кроме i_1. Начальное расписание состоит из одной работы i_1, назначенной на прибор j_1 с моментом запуска r_{j_1}. Для расписания, полученного приближенным алгоритмом, проверяется выполнение следующего условия: момент начала выполнения первой работы приближенного расписания на приборе j_1 должен быть не меньше, чем d_{i_1} (момент окончания выполнения первой работы начального расписания на приборе j_1, назначенной в соответствии с признаком оптимальности), а момент запуска начальных работ приближенного расписания на остальных приборах должен быть не меньше, чем r_{j_1}. Если это условие выполняется, то для получения оптимального расписания необходимо сдвинуть влево расписание работ на приборе j_1, полученное приближенным алгоритмом, так, чтобы момент начала выполнения первой работы приближенного расписания совпадал с моментом окончания выполнения первой работы начального расписания d_{i_1}. Если момент начала выполнения первой работы приближенного расписания на приборе j_1 меньше, чем d_{i_1}, а моменты запуска остальных приборов не меньше, чем r_{j_1}, то первую работу приближенного расписания на приборе j_1 поочередно ставим на остальные приборы и переставляем ее на тот прибор, на котором момент начала выполнения этой работы будет самым большим и не меньше, чем r_{j_1}. Если такой прибор найти не удалось, полиномиальная составляющая не реализована. Если же первая работа перемещена на другой прибор, проверяем условие: если момент запуска второй работы на приборе j_1 больше, чем d_{i_1}, то путем сдвига расписания на приборе j_1 до момента d_{i_1} получаем оптимальное расписание. Если же момент запуска второй работы приближенного расписания на приборе j_1 меньше, чем d_{i_1}, аналогично реализуем перенос второй работы приближенного расписания на приборе j_1. Описанная процедура повторяется, пока либо не получено оптимальное расписание, либо полиномиальная составляющая этим алгоритмом не реализуется.

Алгоритмы Г1.3, Г1.3а

Алгоритмы Г1.3, Г1.3а – это алгоритмы В2.4, В2.4а с учетом примечания В (стр. 16).

Алгоритмы Г1.4, Г1.4а

Алгоритмы Г1.4, Г1.4а – это алгоритмы В2.5, В2.5а с учетом примечания В (стр. 16).

Полиномиальная аппроксимация точного алгоритма решения задачи содержит семь алгоритмов. В качестве приближенных алгоритмов используются алгоритмы В2.3, В2.3а, В2.4, В2.4а, В2.5, В2.5а (см. выше) в силу того, что производится корректировка моментов начала выполнения работ алгоритмом АК (стр. 8), а также алгоритм Г2.1. Задача решается каждым из алгоритмов и выбирается решение, у которого отклонение от r_{j_1} является минимальным.

Алгоритм Г2.1

Алгоритм Г2.1 является модификацией алгоритма Г1.1 (см. выше на стр. 15) со следующим отличием текущей итерации. Если новый момент запуска прибора $r^{\text{нов}} < r_{j_1}$, запаздывающая работа ставится на j_1-й прибор и его расписание сдвига-

ется на величину запаздывания текущей работы, после чего она назначается либо на j_1-й прибор, либо на прибор с моментом запуска $r^{нов}$, в зависимости от того, где больше текущий момент запуска. Для дальнейших итераций этот прибор становится аналогом j_1-го прибора. Процедура назначения продолжается аналогично до распределения всех работ. На последнем шаге алгоритма применяем алгоритм корректировки АК.

Оценка отклонения решения от оптимального. Алгоритм полиномиальной аппроксимации является приближенным по критерию (1) или (2), если реализуется начальная часть признака оптимальности (реализуется назначение начального расписания). Тогда существует оценка отклонения решения, полученного приближенным алгоритмом, от оптимального.

По критерию (1): с этой целью производим сравнение векторов по условию (7).

Примечание. Пусть $r_l > r_{q_l}$, где l – номер компоненты, для которой достигается первая положительная разность, т.е. $r_j = r_{q_j}, j = \overline{1, l-1}$. Тогда знаки разностей остальных компонент ($j = \overline{l+1, m}$) могут быть любыми. Лицо, принимающее решение, анализирует полученное решение и выбирает оптимальное или приближенное решение, исходя из практических соображений.

По критерию (2): оценка задается разностью $r_1 - r_{q_1} \geq 0$ (алгоритмы В2.3, В2.3а, В2.4, В2.4а, В2.5, В2.5а, Г2.1 являются приближенными по критерию (2)).

Примечание. Для критерия (1) алгоритмы В2.1, В2.3, В2.3а, В2.4, В2.4а, В2.5, В2.5а являются приближенными, если существует начальное расписание в соответствии с признаком оптимальности №1.

Примечание. Все остальные алгоритмы полиномиальной аппроксимации точных алгоритмов для задачи 5.1 являются эвристическими.

Примечание. Если в конечном расписании, полученном любым алгоритмом полиномиальной аппроксимации точного алгоритма решения задачи, у хотя бы одного из приборов момент запуска отрицательный, то решение задачи данным алгоритмом не получено.

Выводы

На основе разработанных и доказанных ранее признаков оптимальности допустимых решений [4] для задач составления расписаний выполнения независимых работ на параллельных приборах равной (задача 4.1 [4]) и разной (задача 5.1 [4]) производительности по векторному (лексикографическому) критерию (1) и скалярному критерию (2) разработаны ПДС-алгоритмы их решения. Создано 39 алгоритмов, реализующих полиномиальную составляющую ПДС-алгоритма и полиномиальную аппроксимацию точных алгоритмов решения рассматриваемых задач – эвристические или приближенные алгоритмы с оценкой отклонения от оптимального решения. Задача решается по каждому из алгоритмов и выбирается наилучшее из решений.

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. 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.
3. Pavlov A., Pavlova L. PDC-algorithms for intractable combinatorial problems. Theory and methodology of design.– Uzhhorod, «Karpatskij region» shelf №15, 1998.– 320 pp.
4. Павлов А.А. Признаки оптимальности допустимых решений труднорешаемых задач комбинаторной оптимизации // Вісник НТУУ “КПІ”. Серія «Інформатика, управління та обчислювальна техніка». – К.: “ВЕК+”, 2013. – №59 – С.4–11.

УДК 519.6:512.972, 004.27

КУЛАКОВ Ю.О.,
КЛИМЕНКО І.А.

ОРГАНІЗАЦІЯ БАГАТОРІВНЕВОЇ ПАМ'ЯТІ В РЕКОНФІГУРОВАНИХ ОБЧИСЛЮВАЛЬНИХ СИСТЕМАХ

Запропоновані засоби організації багаторівневої пам'яті реконфігурованої обчислювальної системи, для підтримки розкладу зберігання конфігурацій апаратних задач на локальному рівні обчислювального модуля, зменшуючи накладні витрати реконфігурації і вирішуючи проблему обмеженості внутрішньої пам'яті ПЛІС.

The multilevel memory of reconfigurable computing system to support the scheduling of storage hardware tasks configuration at the local level computing module are proposed, reducing the reconfiguration overhead and solving the problem of limited internal memory of the FPGA.

1. Вступ

На протязі останніх декількох років реконфігуровані обчислювальні системи все більше проявляють риси паралельних обчислювальних систем загального призначення і носять назву реконфігурованих суперкомп'ютерів або високопродуктивних реконфігурованих комп'ютерів (*High-Performance Reconfigurable Computers, HPRC*) [1, 2, 3].

Ефективним механізмом, що забезпечує гнучкість архітектури високопродуктивних реконфігурованих комп'ютерів є динамічна реконфігурація часу виконання (*RunTime* реконфігурація) [1, 4, 5, 6]. При цьому негативним фактором, що впливає на загальну продуктивність обчислень є накладні витрати реконфігурації, найбільш критичною складовою яких є час, витрачений на реконфігурацію і її підтримку.

Сучасні технології провідних виробників ПЛІС пропонують ефективні механізми зниження затримок реконфігурації за рахунок часткової динамічної реконфігурації, коли перезавантажується лише частина мікросхеми [7]. Вбачається, що саме використання часткової динамічної реконфігурації лежить в основі концепції підвищення продуктивності реконфігурованих суперкомп'ютерів [1, 2, 3, 4, 8].

Надалі, засоби часткової реконфігурації надають можливість внутрішнього конфігурування мікросхеми [4, 9]. Це сприяє значному зменшенню часу реконфігурації, за умови розміщення бітових конфігураційних потоків у внутрішній пам'яті мікросхеми. Але інша проблема обмеженості ресурсів внутрішньої пам'яті мікросхеми ПЛІС не дозволяє зберігати велику кількість конфігураційних даних і все ж вимагає залучення зовнішньої пам'яті, що спричинює додаткові часові витрати.

Таким чином, незважаючи на переваги технології часткової динамічної реконфігурації витрати часу на реконфігурацію і обмежені апаратні ресурси внутрішньої пам'яті залишаються значною проблемою, що впливає на досягнення високої продуктивності реконфігурованих обчислювальних систем. Вирішення цієї проблеми є важливим аспектом в сфері високопродуктивних реконфігурованих обчислень, що обумовлює актуальність та доцільність проведених досліджень.

2. Огляд відомих рішень

На ліквідацію накладних витрат реконфігурації в останні роки покладено багато дослідницьких та наукових зусиль. В ряді робіт [10, 11, 12, 13] дослідження спрямовані на рішення проблеми зменшення накладних витрат реконфігурації в області вбудованих реконфігурованих систем, які базуються, як на повній так і на частковій реконфігурації.

Також в літературі представлено багато рішень для підвищення ефективності використання часткової динамічної реконфігурації в високопродуктивних реконфігурованих комп'ютерах. В цій області зазвичай дослідження спрямовані на розробки алгоритмів планування і розміщення задач, що водночас реалізують різні методи зменшення витрат реконфігурації. Найбільш розповсюдженими методами є попередня вибірка конфігурацій [1, 14], повторне використання обчислювальних ресурсів [3, 6, 8], поєднання конфігурацій з метою зменшення реконфігураційних витрат і руху даних [1, 2]. Алгоритми представлені авторами робіт [1, 2, 14] випробувані на існуючих

високопродуктивних платформах *Cray XD*1, *SRC*-6 [15], обмеження передумовленої архітектури яких негативно вплинули на гнучкість реалізації та отримані результати. Переважна кількість оглянутих розробок базуються на програмному або експериментальному моделюванні процесу реконфігурації, часто абстраговано від фізичної сторони реалізації [3, 6, 8, 10]. Інші розглядають загальну архітектурну модель обчислювальної системи, ігноруючи її реальні особливості, на кшталт структури реконфігурованої області, організації комунікаційного середовища та пам'яті, комунікаційних процесів, конфліктів загальних ресурсів, тощо [2, 8].

Ряд робіт, які пропонують архітектурні та технічні вдосконалення базових технологій часткової реконфігурації, базуються на апаратних методах кешування та віртуальних моделях пам'яті, засобах збільшення пропускної здатності інтерфейсів реконфігурації. В роботі [8] на ряді з існуванням двох процесорів для здійснення управління реконфігурацією і виконання програмних функцій передбачено окремий апаратний контролер реконфігурації, який звільнює центральний управляючий процесор від витрат на реконфігурацію і збільшує пропускну здатність реконфігурації, сприяючи прискоренню обчислень. Автори робіт [16, 17] для підвищення швидкості завантаження конфігураційних потоків пропонують підхід прямого доступу до пам'яті конфігурацій. Розміщення блоку кеш пам'яті поряд з портом *ICAP* призводить до подальшого збільшення продуктивності [17]. Одначе, розміщення пам'яті великого об'єму на ПЛІС виявляється неефективним. В роботі [9] прискорення реконфігурації здійснюється за рахунок реалізації пристрою реконфігурації для зв'язку *ICAP* та зовнішньої *SRAM* в режимі прямого доступу, що дозволяє забезпечити швидкість передавання конфігураційного потоку близької до ідеальної – 400 Мбайт/с. Для нових поколінь ПЛІС автори очікують досягнення пропускної здатності в декілька Гбайт/с.

Таким чином, підтримка засобів зменшення накладних витрат, вбудована в алгоритми планування та розміщення задач, зазвичай покладається на рівень програмної або програмно-апаратної надбудови операційної системи [8], що потребує додаткових витрат продуктивності.

З іншого боку, переважно всі відомі архітектурні засоби відокремлені від рішення задач планування та розміщення, що неефективно для реконфігурованих обчислювальних систем, де всі засоби забезпечення послідовності реконфігурації сильно залежать один від одного і у більшості випадків представляють єдиний механізм.

3. Постановка задачі

Визначимо сумарний час виконання задачі T_{Sum}, як суму двох складових часу, які впливають на загальну продуктивність реконфігурованих обчислень:

$$T_{Sum} = T_{Count} + T_{Rconf},$$

де T_{Count} – час виконання задачі апаратними засобами, T_{Rconf} – час реконфігурації апаратної задачі на ПЛІС.

Складова T_{Count} визначає позитивний ефект від традиційних способів зменшення часу виконання задач, шляхом поглиблення паралелізму та їх апаратного прискорення. Таке прискорення очікуване від застосування реконфігурованих обчислень. Складова T_{Rconf} є непродуктивною складовою, яка визначає час витрачений саме на процес реконфігурації апаратури. Ця складова обумовлює втрату продуктивності під час реконфігурованих обчислень і вимірюється накладними видатками. При цьому в деяких випадках може бути порівняний час апаратного прискорення задач і час їх конфігурації на апаратурі [1].

Час реконфігурації в свою чергу складається наступних узагальнених компонент: часу виконання логічної та фізичної послідовності реконфігураційного процесу, що забезпечується на програмному рівні алгоритмами планування та розподілу задач; часу передавання конфігураційних даних, які можуть завантажуватись із загальної системної бібліотеки конфігураційних файлів, із зовнішньої локальної пам'яті обчислювального модуля або із внутрішньої пам'яті ПЛІС, часу прошивання мікросхеми. Тоді до найбільш вагомих компонент накладних витрат віднесемо втрати продуктивності на забезпечення обчислювальної складності алгоритмів планування та розміщення задач, та час витрачений на передавання і завантаження конфігураційних даних.

В якості реконфігурованого простору приймемо динамічну частину ПЛІС, призначену для реконфігурації [4], яку визначимо як реконфігуровану область. Реконфігурована область має розмір і структуру. Розмір залежить від можливостей використовуваного сімейства ПЛІС і

архітектури реконфігурованої системи. Структура визначається використовуваною моделлю розміщення задач. Сучасні технології ПЛІС пропонують 1*D* і 2*D* моделі розміщення задач [3, 4], коли задачі можуть бути розміщені стовбцями різної ширини на всю висоту реконфігурованої області або прямокутниками фіксованого або довільного розміру та місця розташування.

В якості абстрактної моделі обчислювальної задачі розглядаємо функцією або так зване функціональне ядро, синтезоване в цифрову схему з метою розміщення в реконфігурованій області. Визначимо її, як апаратну задачу. Застосуємо традиційну модель задачі абстрактного рівня [3], коли задача характеризується розміром, формою відображення і часом виконання. Розмір визначає просторові вимоги до реконфігурованої області. Форма відображення залежить від моделі розміщення задач в реконфігурованій області. Для всіх моделей розміщення узагальнимо, що задача має форму прямокутника. Тоді задача визначається як вектор

$$T_i = \{S_i, T_{Sum\,i} \mid (T_{Count\,i} + T_{\mathrm{R}config\,i}), I_i, N_i\}, \quad (1)$$

де S_i – площина прямокутника, що містить задачу T_i, а час реконфігурації $T_{\mathrm{R}config\,i}$ це частина часу її виконання $T_{Sum\,i}$, I_i – тип операції, що виконує задача, і N_i – унікальний ідентифікатор задачі у вихідному графі алгоритму, $i = \overline{1, N_{\max}}$, $N_{\max}$ – кількість вершин графу.

Вважаємо що задачі розміщуються на поверхні реконфігурованої області згідно деякому алгоритму планування та розміщення задач з урахуванням їх просторових параметрів (1). Основною ціллю досліджень є зменшення, що сприятиме загальному прискоренню обчислень.

Для забезпечення очікуваного ефекту від застосування реконфігурованих обчислень з точки зору продуктивності, має бути вирішена задача зменшення компонент часу реконфігурації $T_{\mathrm{R}config}$, принаймні в степені порівняній з прискоренням обчислень, що зводитиме до нуля накладні витрати реконфігурації.

4. Вирішення проблеми обмеженого об'єму внутрішньої пам'яті ПЛІС

Алгоритми попередньої вибірки конфігурацій, повторного використання обчислювальних ресурсів, динамічної самореконфігурації, засоби дефрагментації реконфігурованої області і витіснення задач надають максимальні показники продуктивності забезпечуючи швидкість завантаження конфігурацій, шляхом зберігання їх у внутрішні пам'яті ПЛІС. Але це досягаються ціною непродуктивного використання обмежених та коштовних ресурсів внутрішньої пам'яті ПЛІС. Кількісні оцінка використання внутрішньої пам'яті для забезпечення переваг часткової реконфігурації описана в роботі [18], видно, що обмежені ресурси внутрішньої пам'яті ПЛІС є основною перешкодою, яка не дозволяють зберігати велику кількість конфігураційних даних.

Об'єм внутрішньої блочної пам'яті розповсюджених в реалізації реконфігурованих пристроїв сімейств мікросхем Virtex 4 108Кб – 756Кб, Virtex E 4Кб – 16Кб. Розмір повного конфігураційного потоку мікросхем сімейства Virtex E досягає 68Кб – 748 Кб [19]. За даними роботи [18] розмір конфігураційних файлів реконфігурованих лічильників, що відрізняються довжиною лічильного регістру 24Кб – 131Кб. При цьому найменший зконфігурований модуль займає 0.36 % логічного ресурсу мікросхеми Virtex-5 FX70, найбільший – 10 %.

З вищесказаного видно, що ціна використаних ресурсів ПЛІС під час зберігання вже зконфігурованих апаратних задач значно менша, ніж за зберігання їх у вигляді конфігураційних файлів у внутрішній пам'яті. На цьому базується запропонована в статті концепція, подолання накладних витрат реконфігурації за рахунок зменшення часу передавання і завантаження конфігураційних даних. При цьому для тимчасового зберігання конфігурацій запропоновано використовувати поверхню реконфігурованої області і зберігати апаратні ресурси в заздалегідь зконфігурованому вигляді, вирішуючи проблему обмежених ресурсів внутрішньої пам'яті.

5. Модель реконфігурованого обчислювального модуля

Представлені в статті дослідження орієнтовані на застосування в масштабованих реконфігурованих суперкомп'ютерах, побудованих за принципом відкритої архітектури та модульної організації. Такі архітектури зазвичай містять центральний процесор і загальну системну пам'ять. Обчислювальні модулі під'єднані до центральних засобів управління та загальної пам'яті за допомогою високошвидкісних мережевих засобів (4).

Структура запропонованого обчислювального модуля представлена на рис. 1.

На відміну від загальноприйнятої моделі обчислювального модуля, у склад якої зазвичай входить управляючий процесор (УП), енергонезалежна пам’яті для збереження локальних програм та даних (ЛП) і реконфігурований модуль, що реалізований на ПЛІС (РМ), у структурі обчислювального модуля запропоновано використання багаторівневої пам’яті. Багаторівнева пам’ять розміщується між основною системною пам’яттю та внутрімодульними засобами управління логікою реконфігурації (Рис.1). Для реалізації багаторівневої пам’яті запропоновано використання традиційної технології кеш-пам’яті. До складу реконфігурованого модуля також додано спеціальний контролер реконфігурації, з метою зменшення часу реконфігурації за рахунок розділення логічного і фізичного рівнів реконфігурації і розвантажування управляючого процесора [8].

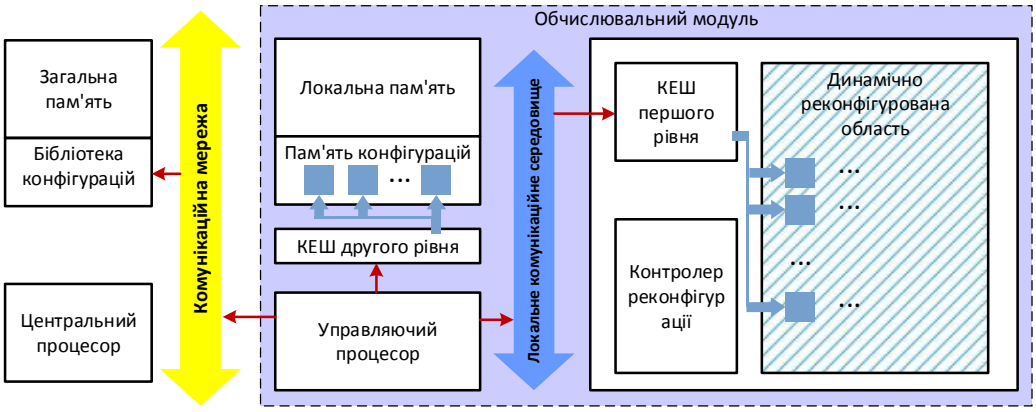


Рис. 1. Архітектура реконфігурованої обчислювальної системи

Реконфігурований модуль складається зі статичної і динамічної області. Динамічна область призначена для розміщення апаратних задач у вигляді відповідно зконфігурованих функціональних блоків (ФБ). Під ФБ ми розуміємо структурне визначення задачі, що є частиною реконфігурованої області в яка містить апаратну задачу. В статичній області ПЛІС розміщується кеш пам’ять першого рівня і контролер управління фізичною послідовністю реконфігурації.

Динамічна область разом із своїм основним обчислювальним призначенням одночасно виконує функцію зберігання функціональних блоків для їх подальшого використання. Конфігурації ФБ зберігаються на поверхні реконфігурованої області за певним розкладом. Кеш пам’ять використовується для обліку управляючої інформації і підтримки розкладу реконфігурації функціональних блоків. Фактично кеш пам’ять містить покажчики на функціональні блоки, що зберігаються в реконфігурованій області, та інформацію щодо їх поточного стану.

Первинне завантаження конфігурацій ще більше уповільнює реконфігурацію за рахунок звернення до загальної системної пам’яті, де зберігається централізована бібліотека конфігурацій функціональних блоків. Для запобігання великої кількості звернень до загальної пам’яті запропонований кеш другого рівня, де зберігаються одного разу вже застосовані конфігураційні файли. Локальна пам’ять конфігурацій реалізована у складі локальної пам’яті обчислювального модуля, яка не так критична до об’єму, ніж внутрішня пам’ять ПЛІС. Але, як вже було зазначено, обмін конфігураційними даними між локальною пам’яттю і реконфігурованою областю буде спричиняти накладні витрати реконфігурації. Кеш пам’ять другого рівня містить покажчики на конфігураційні файли апаратних задач та їх поточний стан.

Алгоритм запуску апаратної задачі на виконання в реконфігурованій області наступний:

Крок 1. Управляючий процесор запускає чергову задачу для обчислення, яка представлена словом команди і словами даних. У форматі команди присутнє поле типу операції, що має бути виконана над вхідними даними. На підставі чого УП формує виконавчу адресу, для пошуку конфігураційних файлів в пам’яті.

Крок 2. Виконавча адреса поступає на вхід кеш-пам’яті першого рівня, здійснюється пошук раніш зконфігурованого функціонального блоку. У випадку «успішного звернення» УП ініціює процедуру запуску задачі в існуючому функціональному блоці.

Крок 3. У випадку «промаху» виконавча адреса поступає на вхід кеш-пам’яті другого рівня в пошуку тимчасово збереженого конфігура-

ційного файлу апаратної задачі. У випадку «успішного звернення» УП ініціює процедуру повторного завантаження конфігураційного файлу із локальної пам'яті на ПЛІС і запуску задачі.

Крок 4. У випадку «промаху» УП завантажує конфігураційний файл на ПЛІС із бібліотеки конфігурацій загальної пам'яті і запускає задачу.

Повернення до кроку 1.

6. Модель пам'яті стану функціональних блоків

Виконувана програма у вихідному стані подана у вигляді ациклічного графу G, в вершинах якого розміщуються обчислювальні задачі. Максимальна кількість вершин графу обмежена величиною $N_{\max}$. Задачі однозначно визначаються унікальним ідентифікатором задачі у вихідному графі $N_g \mid g = \overline{1, N_{\max}}$.

Кожна задача асоціюється з параметром «Тип операції». Тип операції відповідає функції, яку виконує задача, що синтезована в цифрову схему і завантажується в функціональний блок під час реконфігурованих обчислень. Апаратні реалізації задач створюються заздалегідь і зберігається в централізованій бібліотеці конфігурацій у вигляді конфігураційних файлів, кількість яких дорівнює M_I – по одній для кожного типу операції. Таким чином тип операції $I_j \mid j = \overline{1, I_{\max}}$, де $I_{\max}$ – кількість апаратно реалізованих типів операцій, однозначно визначає конфігураційний файл, що завантажується в функціональний блок і використовується як основний параметр для пошуку його конфігурації в пам'яті, або на мікросхемі ПЛІС.

Технологічні обмеження площини реконфігурованої області та розміри конфігурацій обмежують кількість розміщуваних функціональних блоків параметром $F_{\max}$. Тоді кожний функціональний блок, що розміщений на поверхні реконфігурованої області визначається ідентифікатором $F_s \mid s = \overline{1, F_{\max}}$.

Співвідношення між описаними параметрами наведені на рис. 2. Означені співвідношення важливі для формування виконавчої адреси пошуку конфігураційних файлів в локальній та загальній пам'яті і функціональних блоків на поверхні реконфігурованої області. Кожний конфігураційний файл однозначно визначається типом операції. Декілька функціональних блоків можуть виконувати один і той самий тип операції, відповідно до типу задач. Кожна обчислювальна задача N_g в визначений момент часу T_n виконується в окремому функціональному блоці F_s. Протягом часу в кожному функціональному блоці може бути виконано послідовність задач, у випадку їх повторного використання (рис. 3). Відповідно до цього, конфігураційні файли ідентифікуються значенням типу операції I. Для ідентифікації зконфігурованих на ПЛІС функціональних блоків використаємо складений з двох полів ідентифікатор адреси $I.F$ (рис. 4).

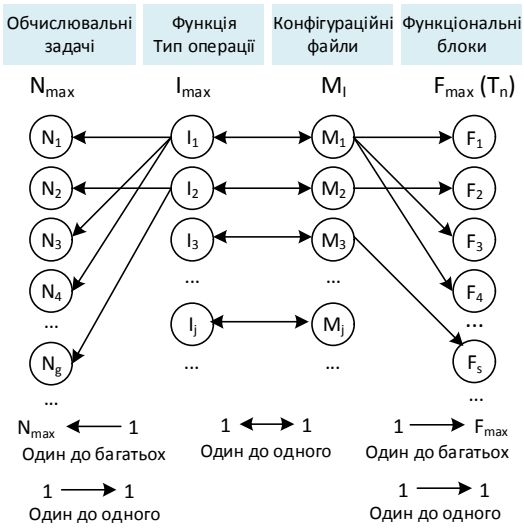


Рис. 2. Співвідношення між основними параметрами

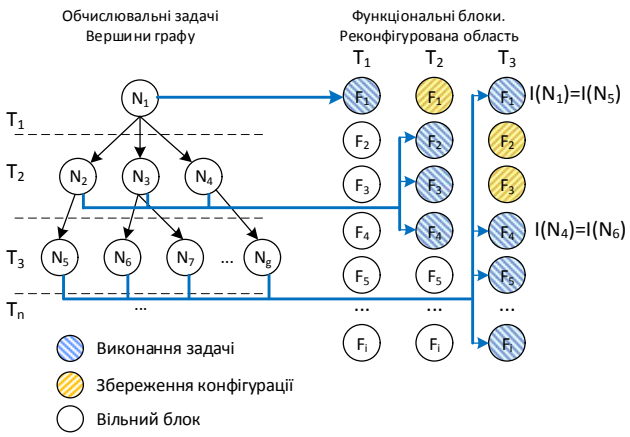


Рис. 3. Принцип завантаження функціональних блоків

Припустимо, що максимальна кількість виконуваних типів операції $M_{\max}$ = 64, максимальна кількість конфігурацій ФБ в одній ПЛІС F_{max} = 63. Тоді модель пам'яті для зберігання стану розміщених на площині реконфігурованої області ПЛІС знадобиться мінімум 4Кб пам'яті з адресним доступом, за умови мінімального обсягу управляючої інформації, що дорівнює одному байту. За збереження, наприклад чотирьох байтів управляючої інформації знадобить-

ся 16Кб пам’яті. Модель пам’яті з адресним доступом зображена на рис. 5.

Концепцію асоціативного пошуку в багаторівневій пам’яті реалізують класичні технології кеш-пам’яті. Використання їх особливостей, таких як локальність розміщення, прозорість для програмного шару, апаратна реалізація пошуку, швидкість пошуку, відсутність збиткових записів, мобільний розмір, вирішують ряд проблем пов’язаних з накладними видатками реконфігурації і обмеженим ресурсом ПЛІС.

7. Вибір моделі кеш пам’яті

Для вирішення поставленої в роботі задачі розглянемо дві типові технології організації кеш-пам’яті – повністю асоціативна кеш-пам’ять і кеш-пам’ять із прямим відображенням адрес.

Після формування виконавчої адреси в повністю асоціативній пам’яті її старші біти, що утворюють тег, одночасно порівнюються з всіма тегами записів пам’яті. Якщо всі порівняння призводять до негативного результату, фіксується так званий «кеш-промах», в іншому випадку, коли знайдений один шуканий запис фіксується «кеш-успіх».

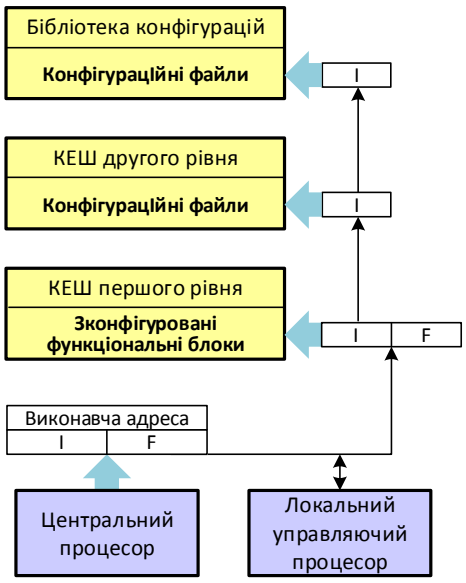


Рис. 4. Формат виконавчої адреси

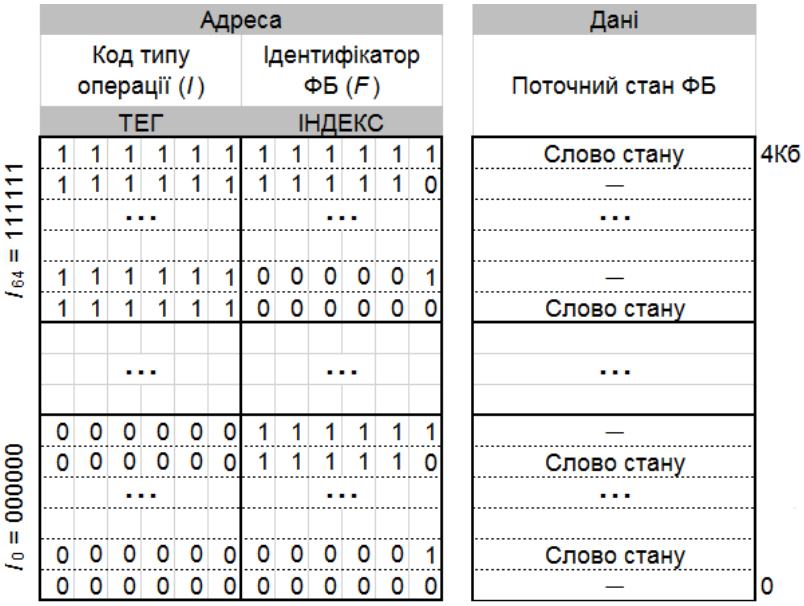


Рис. 5. Модель пам’яті

З огляду на проблему зберігання ресурсів внутрішньої пам’яті ПЛІС, враховуючи, що максимум шістдесят чотири слова даних із доступних 4Кб будуть використовуватись для збереження станів, в роботі запропоновано використання пам’яті з асоціативним пошуком інформації. Це скоротить об’єм пам’яті станів до 64 байтів. За збільшення довжини слова стану, наприклад до чотирьох байтів об’єм асоціативної пам’яті збільшиться до 256 байт. Що потребуватиме доволі незначних запам’ятовуючих ресурсів мікросхеми ПЛІС.

З іншого боку повторне використання раніш зконфігурованих функціональних блоків та підтримка їх розкладу зберігання, вимагає здійснення пошуку в пам’яті станів шляхом перебору всіх записів пам’яті. Звичайно за застосування пам’яті з адресним доступом час виконання такого пошуку буде пропорційний кількості записів, що потребуватиме значного часу.

Асоціативний пошук це типовий засіб спрямований на прискорення пошуку інформації. Для здійснення пошуку в асоціативні пам’яті процесор видає в інтерфейс пам’яті спеціальне слово, яке називається тегом. Під час виконання пошуку заданий тег порівнюється з тегом кожного запису асоціативної пам’яті. Таким чином здійснення пошуку зводиться до одного звернення до пам’яті з боку процесора, що прискорює процедуру пошуку і розвантажує комунікаційне середовище. Сама процедура асоціативного пошуку підтримується апаратно на рівні інтерфейсу пам’яті.

При цьому, якщо в якості тегу використовується ідентифікатор адреси функціонального блоку, складений з двох полів $I.F$, ми отриму-

ємо максимальне зменшення об'єму пом'яті. Тоді за видалення збиткових записів об'єм пом'яті дорівнюватиме шістдесят чотири записи, по одній для кожного ФБ. При цьому прискорення пошуку буде досягнуте лише за рахунок зменшення кількості записів. Пошук збереже всі риси пошуку в пам'яті з адресним доступом, таким чином, значне прискорення неможливе.

Пояснимо це наступним чином. Як вже було зазначено між типом виконуваних операції і ФБ зконфігурованими на ПЛІС і існує зв'язок один-до-багатьох, тобто можлива наявність певної кількості однотипних блоків. Ціллю пошуку є знаходження всіх ФБ на ПЛІС, що виконують певний тип операції. Технологія же повністю асоціативної кеш-пом'яті дозволяє знаходження лише одного запису. В цьому випадку процедуру пошуку доведеться покласти на процесор, що суперечить ідеї досліджень. Використання же в якості тегу лише поля типу операції, можливе, але слова станів всіх однотипних блоків доведеться зберігати в одному рядку кеш-пам'яті. Це потягне за собою ускладнення й уповільнення процедури пошуку та потребуватиме додаткового вдосконалення інтерфейсу кеш-пам'яті. Окрім всього типова апаратура повністю асоціативної кеш-пам'яті будується на застосуванні компараторів, кількість яких дорівнює кількості записів кеш-пам'яті.

Приведемо також оцінку апаратних ресурсів для реалізації асоціативної кеш-пам'яті. За зроблених припущень знадобиться 64 дванадцятибітні компаратори. Приблизна оцінка затрат устаткування для реалізації одного компаратора складає 10 транзисторів на один біт компаратора [19], таким чином нам знадобиться 7680 транзисторів для реалізації лише схеми порівняння в інтерфейсі кеш-пам'яті. Для реалізації схеми порівняння в типових реалізаціях пам'яті, що використовується в найбільш розповсюджених процесорах, об'ємом 256 Кб з довжиною рядка 32 байти знадобиться 2 211 840 транзисторів.

Поставленим в роботі цілям щодо виконання пошуку найбільш відповідає асоціативна пам'ять з прямим відображенням адрес.

Рядки кеш пам'яті адресуються асоціативним тегом, в якості якого приймаємо тип операції, а ряд індексів, що переадресовується на кожний рядок пам'яті представляє ідентифікатори функціональних блоків. Схема організації кеш-пам'яті наведена на рис. 6. За зроблених припущень кількість рядків, яку визначає розрядність тегу, дорівнює шістдесят чотири.

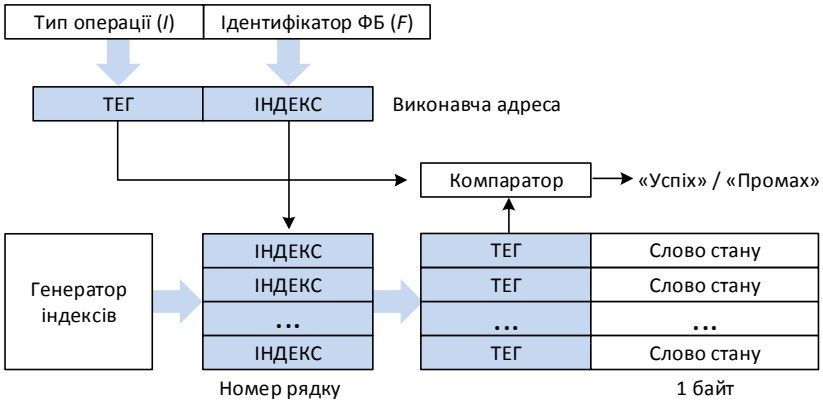


Рис. 6. Структурна схема кеш-пам'яті першого рівня

Стандартний алгоритм функціонування наступний. На адресний вхід інтерфейсу пам'яті поступає виконавча адреса, що складається з полів [$I.F$] = [ТЕГ.ІНДЕКС]. Аналізується поле ІНДЕКС, яке вказує на один з ФБ. Надалі старші розряди адреси (ТЕГ), що вказують на шуканий тип операції, порівнюються з тегом, який зберігається в рядку. За збігу формується сигнал «Успіху», інакше – «Промаху». Така пам'ять потребує мінімального об'єму устаткування. Для порівняння знадобиться лише один компаратор, на вхід якого подається тег КЕШ-рядка обраного за полем «ІНДЕКС».

Таким чином можливо реалізувати наступну логіку цільового пошуку. Фактично задача стоїть в переборі всіх функціональні блоків і пошуку серед них тих, що виконують шуканий тип операції. Надалі розподілювач задач аналізує стан вибраних блоків і обирає серед них блок для завантаження чергової задачі.

Для звільнення процесору від виконання послідовного перебору адрес функціональних блоків, до складу інтерфейсу пам'яті вводиться спеціальний пристрій – генератор адрес (рис. 6). Тоді виконавча адреса, що видається процесором на початку пошуку складається з полів [*I*.0]. генератор адрес послідовно видає індекси для пошуку, а значення коду операції порівнюється з тегами рядків.

Відомий недолік традиційного застосування кеш-пам'яті з прямим відображенням адрес, коли неможливо одночасно зберігати два рядки з одним і тим самим індексом і різними значеннями тегів, не заважає рішенню цільової задачі дослідження, бо така ситуація ніколи не виникає – неможливо виконання декількох операцій одним функціональним блоком.

Записи, що видаляються із кеш-пам'яті потрапляють в кеш-пам'ять другого рівня. Відповідно до цього конфігурації функціональних блоків видаляються з поверхні реконфігурованої області, а конфігураційні дані зберігаються у локальній пам'яті обчислювального модуля.

8. Кеш-пам'ять другого рівня

Призначення кеш-пам'яті другого рівня в наданні додаткового часу для зберігання конфігураційних даних на локальному рівні обчислювального модуля. Це сприятиме додатковому зменшенню накладних видатків реконфігурації за рахунок відсутності комунікаційних процедур міжмодульного рівня.

Для реалізації такої кеш-пам'яті застосуємо повністю асоціативну кеш-пам'ять. Обмежена кількість виділеної пам'яті для зберігання конфігураційних файлів обмежує об'єм кеш-пам'яті декількома рядками, що обумовлює доволі незначні апаратні витрати. В якості асоціативного тегу для пошуку застосовується поле типу операції, що однозначно визначає файл конфігурації функціонального блоку.

Розклад зберігання конфігураційних файлів в кеш-пам'яті другого рівня підтримується автоматично стандартними засобами без участі планувальника. В даному випадку доцільна стратегія обмеження кількості рядків пам'яті і витіснення із пам'яті записів або найбільш давніх записів, або таких що мають меншу ймовірність використання. Для цього існують відомі алгоритми, які можуть бути успішно використані: алгоритм заміщення на основі найбільш давнього застосування (***Least Recently Used, LRU***), - заміщення найменш часто використовуваного рядку (*Least Frequently Used, LFU*).

9. Висновки

Накладні витрати реконфігурації є значною проблемою, яка негативним чином впливає на продуктивність реконфігурованих обчислювальних систем. Всі відомі алгоритми планування й розподілу задач, що підтримують різні методи зменшення накладних витрат, реалізовані на абстрактному рівні операційної системи засобами центрального управляючого процесора. Це потребує додаткового часу на їх реалізацію.

Технічна підтримка здійснювана запропонованими засобами спрямована на зменшення часу виконання реконфігурації методом повторного використання ресурсів функціональних блоків.

Запропонована структура обчислювального модуля на базі багаторівневої пам'яті вирішує задачу управління розкладом розміщення та підтримки заздалегідь завантажених функціональних блоків, прозоро для програмного шару, локалізовано на рівні апаратури обчислювального модуля. Таким чином, запропоновані засоби зменшують обчислювальну складність алгоритмів управління логічною послідовністю реконфігурації.

Запропонований новий спосіб зберігання конфігурацій апаратних задач на поверхні реконфігурованої області ПЛІС на базі застосування багаторівневої пам'яті, який забезпечує зменшення часу реконфігурації і вирішує проблему обмеженості ресурсів внутрішньої пам'яті ПЛІС. Запропонований спосіб може бути використаний для забезпечення методів повторного використання і попереднього завантаження обчислювальних ресурсів, які широко застосовуються для зменшення накладних витрат реконфігурації.

Список посилань

1. El-Araby E. Exploiting Partial Runtime Reconfiguration for High-Performance Reconfigurable Computing / E. El-Araby, I. Gonzalez, T. El-Ghazawi // ACM Transactions on Reconfigurable Technology and Systems (TRETS). - USA, NY, New York, ACM, 2009. – Vol. 1, Issue 4, Article № 21.

2. Huang M. Reconfiguration and Communication-Aware Task Scheduling for High-Performance Reconfigurable Computing / M. Huang, V.K. Narayana, H. Simmler, O. Serres, T. El-Ghazawi // Transactions on Reconfigurable Technology and Systems (TRETS). – USA, NY, New York, ACM, 2010. – Vol. 3, Issue 4, Article № 20.
3. 3. Bassiri M.M. Mitigating Reconfiguration Overhead In On-Line Task Scheduling For Reconfigurable Computing Systems / M.M. Bassiri, S.H. Shahriar // Proceeding of 2nd International Conference on Computer Engineering and Technology (ICCET), (Chengdu, 16-18 April 2010). – IEEE, 2010. – Vol. 4. – P. V4-397. – V4-402.
4. 4. Клименко І.А. Класифікація реконфігурованих обчислювальних систем / І.А. Клименко, М.В. Рудницький // Вісник Вінницького політехнічного інституту. – Вінниця: ВНТУ, 2014. - №5 (116). – С. 120 – 128.
5. 5. Danne K. Periodic Real-Time Scheduling for FPGA Computers / K. Danne, M. Platzner // Published in Third International Workshop on Intelligent Solutions in Embedded Systems (Germany, Hamburg, 2005). – IEEE, 2005. – P. 117 – 127.
6. 6. Panella A. A Design Workflow for Dynamically Reconfigurable Multi-FPGA Systems / A. Panella, F. Redaelli, F. Cancare, D. Sciuto // Published in 18th IEEE/IFIP VLSI System on Chip Conference (VLSI-SoC), (Spain, Madrid, 27-29 Sept. 2010). – IEEE, 2010. – P. 114 – 119.
7. 7. What's New in Xilinx ISE Design Suite 12 [Електронний ресурс]. – Xilinx, 2014. – Режим доступу: http://www.xilinx.com/support/documentation/sw_manuals/xilinx12_3/whatsnew.htm.
8. 8. Al-Wattar A. Efficient On-line Hardware/Software Task Scheduling for Dynamic Run-time Reconfigurable Systems / A. Al-Wattar, S. Areibi, F. Saffih // Proceeding in 26th International Parallel and Distributed Processing Symposium Workshops & PhD Forum (IPDPSW). – IEEE, 2012. - P 401 – 406.
9. 9. Liu S. Achieving Energy Efficiency through Runtime Partial Reconfiguration on Reconfigurable Systems / S. Liu, R.N. Pittman, A. Forin, J.-L. Gaudiot // Transactions on Embedded Computing Systems (TECS). – USA, NY, New York, ACM, 2013. – Volume 12, Issue 3, Article № 72. – 21 p.
10. 10. Pellizzoni R. Adaptive allocation of software and hardware real-time tasks for FPGA-based embedded systems / R. Pellizzoni, M. Caccamo // Proceeding of the 12th Real-Time and Embedded Technology and Applications Symposium, 2006. – IEEE, 2006 – P. 208 – 220.
11. 11. Extending Embedded Computing Scheduling Algorithms for Reconfigurable Computing Systems Saha, P. ; El-Ghazawi, T. Proceedeng in 3rd Southern Conference on Programmable Logic (SPL '07) (Mar del Plata, 28-26 Feb. 2007). – IEEE, 2007 – P. 87 – 92.
12. 12. Hasan M.Z. Runtime partial reconfiguration for embedded vector processors / M.Z. Hasan, S.G. Ziavras // Journal of Computers. – Vol. 2, №9. – Academy Publisher, 2007. – P. 60 – 66.
13. 13. Jara-Berrocal A. VAPRES: A Virtual Architecture for Partially Reconfigurable Embedded Systems / A. Jara-Berrocal, A. Gordon-Ross A.// Proceeding of Design, Automation & Test in Europe Conference & Exhibition (DATE), (Germany, Dresden, 8-12 March 2010) – IEEE, 2010. – P. 837 – 842.
14. 14. Taher M. Virtual Configuration Management: A Technique for Partial Runtime Reconfiguration / M. Taher, T. El-Ghazawi // IEEE Transactions on Computers. – IEEE, 2009. – Vol. 58, Issue: 10. – P. 1398 – 1410.
15. 15. Buell D. High-Performance Reconfigurable Computing // D. Buell, T. El-Ghazawi, K. Gaj, V. Kindratenko // Journal Computer. – Volume 40 Issue 3 – USA, CA, IEEE Computer Society Press Los Alamitos, 2007. – P. 23 – 27.
16. 16. Koch D. Fine-Grained Partial Runtime Reconfiguration on Virtex-5 FPGAs // D. Koch, C. Beckhoff, J. Torrison // Proceeding of 18th IEEE Annual International Symposium on Field-Programmable Custom Computing Machines (FCCM) (NC, Charlotte, 2-4 May 2010). – IEEE, 2010, - P. 69 – 72.
17. 17. Liu S. Minimizing the runtime partial reconfiguration overheads in reconfigurable systems / S. Liu, R.N. Pittman, A. Forin, J.-L. Gaudiot // The Journal of Supercomputing, 2012. – Volume 61, Issue 3. – P. 894-911
18. 18. Дунець Р. Б. Проблеми побудови частково реконфігурованих систем на ПЛІС / Р. Б. Дунець, Д. Я. Тиханський // Радіоелектронні і комп'ютерні системи, 2010. – № 7 (48). – с. 200 - 204.
19. 19. Virtex TM 2.5 V Field Programmable Gate Arrays. DS003-1 (v4.0) [Електронний ресурс]. – Xilinx, 2013. – Режим доступу: http://www.xilinx.com/support/documentation/data_sheets/ds003.pdf.

УДК 519.854.2

ПАВЛОВ А.А.,
ХАЛУС Е.А.

СОСТАВЛЕНИЕ ДОПУСТИМОГО РАСПИСАНИЯ ВЫПОЛНЕНИЯ РАБОТ НА ОДНОМ ПРИБОРЕ ОПТИМАЛЬНОГО ПО КРИТЕРИЮ МИНИМИЗАЦИИ СУММАРНОГО ОПЕРЕЖЕНИЯ РАБОТ

Рассмотрена задача составления допустимого расписания выполнения независимых работ с различными длительностями и директивными сроками одним прибором, при котором минимизируем суммарное опережение директивных сроков. Предложен ПДС – алгоритм нахождения оптимального расписания по критерию минимизации суммарного опережения.

We considered a problem of creation of feasible schedule of performing an independent operations with different durations and due dates of a single device, in which the minimum of the total earliness of due dates is achieved. We consider PDS – an algorithm for finding an optimal schedule on criteria of minimizing the total earliness.

Введение

Работа посвящена широко распространенной на практике задаче составления расписания выполнения множества работ с различными директивными сроками одним прибором, в которых все работы должны быть выполнены без нарушения их директивных сроков, и при этом должно достигать минимума суммарное отклонение моментов окончания работ от их директивных сроков.

Постановка задачи

Задано множество независимых работ $J=\{1,2,...,n\}$, каждая из которых состоит из одной операции. Для работы $j \in J$ известны длительность выполнения p_j и директивный срок выполнения d_j. Прерывания работ не допускаются. Работы поступают в систему одновременно. Процесс выполнения работ является непрерывным: после выполнения первой по порядку работы сразу же начинает выполняться вторая и т.д. Необходимо найти допустимое расписание, в котором суммарное опережение моментов окончания выполнения работ относительно директивных сроков принимает минимальное значения.

Определение 1. Расписание называется *допустимым*, если в нем удовлетворены все директивные сроки (все задания не запаздывают).

Определение 2. Для заданного расписания под *моментом запуска* r понимается момент начала выполнения работы, которая в расписании стоит на первой позиции [3].

Введем обозначения:

C_j – момент окончания работы j;

E_j – значение опережения для j - ой работы: $E_j = \max\{0; d_j - C_j\}$.

В допустимом расписании $d_j \geq C_j \ \forall j$, т.е. $E_j = d_j - C_j$.

Эта задача является близкой к двухкритериальной задаче, рассмотренной в [2,3], в которой требуется найти допустимое расписание, в котором момент запуска работ является максимально поздним (***критерий 1***), а суммарное опережение выполнения работ относительно директивных сроков принимает минимальное значения (***критерий 2***). В рамках решения двухкритериальной задачи предполагалось, что первичным критерием являлся ***критерий 1***. В некоторых практических задачах основным критерием оценки расписания является ***критерий 2*** – минимизация суммарного опережения, именно этот критерий и рассматривается в данной работе.

Для решения рассматриваемой задачи были использованы результаты, полученные при исследовании двухкритериальной задачи [1,2,3]. Так, в работе [2] был разработан алгоритм ***A*** определения самого позднего момента запуска выполнения работ в системе $n/1/r \to \max$, при котором расписание остается допустимым. Суть алгоритма состоит в следующем. Вначале работы упорядочиваются по неубыванию директивных сроков, затем определяется момент запуска выполнения работ, при котором в расписании хотя бы одна работа будет запаздывающей. Определяется максимальное из запазды-

ваний и вся последовательность работ сдвигается влево на эту величину.

Теорема 1. Алгоритм ***A*** строит допустимое расписание, в котором момент r начала выполнения работ (момент запуска) является самым поздним [2].

Следующая теорема описывает достаточные условия оптимальности двухкритериальной задачи.

Теорема 2. Допустимое расписание с моментом запуска $r_{\max}$, построенное по неубыванию директивных сроков, является оптимальным по критерию минимизации суммарного опережения при выполнении условий [2]:

$$d_{j_1} \le d_{j_2} \le \ldots \le d_{j_n}, p_{j_1} \ge p_{j_2} \ge \ldots \ge p_{j_n}, \qquad (1)$$

где j_i - это номер задания, которое в расписании занимает i-ую позицию.

Если сохранять установленный момент запуска $r_{\max}$ и при этом стремиться минимизировать и суммарное опережение заданий, то упорядочение должно осуществляться в соответствии с теоремой 3.

Теорема 3. Пусть σ - допустимое расписание выполнения n работ с максимально поздним моментом запуска $r_{\max}$, тогда допустимое расписание $\sigma_1 = \{j_1, j_2, \ldots j_n\}$ (j_i - это номер работы, которая в σ_1 занимает i-ую позицию) построенное по следующему правилу:

$$p_{j_n} = \min_{s \,/\, d_S \ge \sum_{j=1}^{n} p_j + r_{\max}} \{p_s\}, \qquad (2)$$

$$p_{j_i} = \min_{s \,/\, d_S \ge \sum_{j=1}^{n} p_j - \sum_{l=0}^{n-i-1} p_{j_{n-l}} + r_{\max}} \{p_s\}, \; i = \overline{n-1,1} \qquad (3)$$

является оптимальным по критерию минимизации суммарного опережения работ с максимально поздним моментом запуска $r_{\max}$ [3].

Следствие 3.1. Для любого заданного момента запуска $\forall r < r_{\max}$ расписание, построенное в соответствии с рекуррентными соотношениями (2)-(3), имеет минимальное суммарное опережение для данного r [3].

В работе [3] был разработан полиномиальный алгоритм ***A1*** нахождения оптимального расписания по критерию суммарного опережения для заданного момента запуска r. Схема алгоритма базируется на равенствах (2)-(3), в которых вместо $r_{\max}$, подставляется $\forall r < r_{\max}$.

Отметим, что при ухудшении расписания по ***критерию 1*** (при смещении его влево) можно получить расписание, которое по ***критерию 2*** лучше предыдущего. Следующий пример иллюстрирует это утверждение.

Пример 1.

Пусть имеется 5 работ с такими длительностями выполнения и директивными сроками:

$n = 5$;

$p_1 = 1$; $p_2 = 1$; $p_3 = 1$; $p_4 = 1$; $p_5 = 100$;

$d_1 = 110$; $d_2 = 111$; $d_3 = 112$; $d_4 = 113$; $d_5 = 114$.

В соответствии с алгоритмом ***A*** определено, что $r_{\max} = 10$. По Алгоритму ***A1*** для заданного момента запуска построено расписание, оптимальное по ***критерию 2***, у которого суммарное опережение равно 396 (рисунок 1).

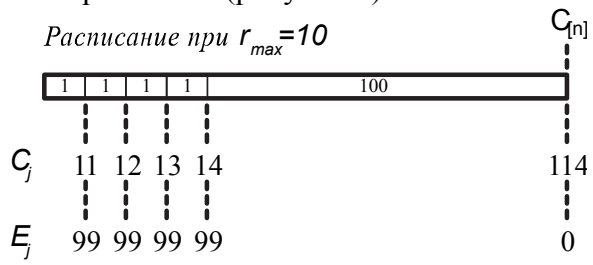


Рис. 1. Расписание при максимальном моменте запуска $r_{\max} = 10$

После этого для момента запуска $r = 9$ по алгоритму ***A1*** построено расписание, оптимальное по ***критерию 2*** , у которого суммарное опережение равно 5 (рисунок 2).

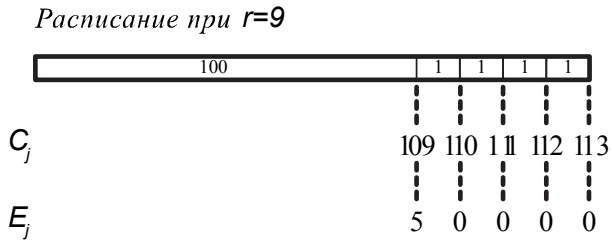


Рис. 2. Расписание при моменте запуска $r = 9$

Как видим, при ухудшении на единицу значения ***критерия 1***, значение ***критерия 2*** улучшилось на величину 396-5=391.

Одним из подходов к эффективному решению как труднорешаемых (NP-полных) задач комбинаторной оптимизации, так и в некоторых случаях P – разрешимых задач комбинаторной оптимизации является разработка ПДС – алгоритмов [1]. В [4] приведена классифика-

ция ПДС- алгоритмов. ПДС – алгоритмы бывают двух типов. Необходимым и достаточным условием построения ПДС – алгоритма, является нахождение теоретически обоснованных признаков оптимальности допустимого решения (достаточные условия оптимальности), которые конструктивно проверяются в процессе ее решения полиномиальным алгоритмом (Р – составляющая ПДС - алгоритма). Данная задача решается ПДС – алгоритмом второго типа: кроме Р – составляющей он содержит полиномиальную аппроксимацию точного алгоритма (приближенное решение, полученное в случае не выполнения достаточных условий оптимальности).

Признак оптимальности допустимого расписания

Обозначим через:

$E(r)$ – значение критерия 2 в расписании, построенном по теореме 3 (минимальное суммарное опережение при заданном моменте запуска r);

r_{opt} – момент запуска в расписании с минимальным суммарным опережение среди расписаний со всеми возможными моментами запуска.

Пусть имеем расписание, построенное по теореме 3 для момента запуска $r_{\max}$. Очевидно, что при уменьшении момента запуска на произвольную величину Δ (смещении полученной последовательности работ влево на Δ) суммарное опережение увеличится на $n\Delta$. При дальнейшем увеличении величины Δ может наступить такой момент, когда для некоторого момента запуска $r = r_{\max} - \Delta \geq 0$ после применения теоремы 3 произойдет смена структуры расписания (изменение порядка выполнения работ) и при этом $E(r) < E(r^+)$. $E(r^+)$ - точки разрыва в графике зависимости $E(r)$ от r, которые представлен на рисунке 3.

Признак 1) Пусть имеем расписание, у которого момент запуска равен $r_{\max}$ и для которого выполняется равенство:

$$E(r_{\max}) + nr_{\max} = E(0), \qquad (4)$$

(где $E(r_{\max})$, $E(0)$ - значение ***критерия 2*** для моментов запуска $r_{\max}$ и 0 соответственно), то это расписание является оптимальным по двум критериям. То есть, имеем, что $r_{opt} = r_{\max}$.

Отметим также, что для любого r (и в том числе и для $r_{\max}$) невозможно выполнение неравенства: $E(r) + nr < E(0)$.

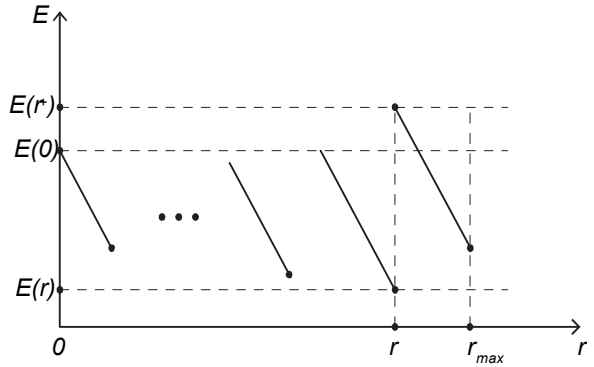


Рис. 3. Изменение суммарного опережения при изменении момента запуска

Признак 2) Пусть равенство (4) не выполняется и пусть при этом существует фиксированный момент запуска $r < r_{\max}$, для которого выполняется равенство:

$$E(r) + nr = E(0) \qquad (5)$$

Тогда оптимальный момент запуска r_{opt} (соответствующий оптимальному суммарному опережению) удовлетворяет неравенству:

$$r \leq r_{opt} < r_{\max}.$$

Доказательство выполнения признаков проведем методом от противного.

Пусть выполняется признак 1, но при этом $r_{opt} \neq r_{\max}$. Это означает, что существует момент запуска $r_{opt} < r_{\max}$, для которого $E(r_{opt}) < E(r_{\max})$.

С учетом того, что

$$E(r_{\max}) + nr_{\max} = E(0)$$

имеем:

$$nr_{\max} > nr_{opt}.$$

Следовательно

$$E(r_{opt}) < E(r_{\max}),$$

$$E(r_{opt}) + nr_{opt} < E(r_{\max}) + nr_{\max} = E(0)$$

Пришли к противоречию. Справедливость признака 1) доказано.

Пусть выполняется признак 2), но $r_{opt} < r$, тогда $E(r_{opt}) < E(r)$. Следовательно, $E(r_{opt}) + nr_{opt} < E(r) + nr = E(0)$, что невозможно. Справедливость признака 2) доказано. ■

Принципиальная схема ПДС-алгоритма

Вначале для $r_{\max}$ строится оптимальное по критерию 2 расписание. Если для него выпол-

няется равенство $E(r_{\max}) + nr_{\max} = E(0)$, то полученное расписание оптимально. Если не выполняется признак 1) признака оптимальности, то применяется полиномиальный (со сложностью $kO(n^2)$) алгоритм, который изменяя моменты запуска выполнения заданий от $r_{\max}$ до 0 и применяя рекуррентно процедуру из теоремы 3 позволяет получить ряд расписаний, оптимальных для моментов запуска

$$r_{\max},\ r_1,\ r_2, ..., r_k,$$

где: $r_1 = r_{\max} - \Delta_1, r_2 = r_1 - \Delta_2, ..., r_k = r_{k-1} - \Delta_k$.

$r_1,\ r_2, ..., r_k$ – дискретные моменты изменения структуры оптимального по критерию 2 расписания, когда точно меняется позиция хотя бы одной работы (k - заданное число).

Длины интервалов $\Delta_l, l = \overline{1,k}$ с неизменной структурой расписания определяем следующим образом:

$$\Delta_l = \min_{2 \le i \le n} \left\{ C_{j_i} - \max_{\substack{s / p_s < p_{j_i}: \\ d_s < C_{j_i}}} \{d_s\} \right\}. \qquad (6)$$

Величина $\max\limits_{\substack{s / p_s < p_{j_i}: \\ d_s < C_{j_i}}} \{d_s\}$ – максимальный директивный срок из тех работ, длительность которых меньше длительности текущей работы j_i, и которые имеют директивный срок меньше, чем момент окончания работы j_i.

Процесс заканчивается, если для любого i в (6) максимум не определен (нет работ s, которые удовлетворяют условиям: $p_s < p_{j_i}$, $d_s < C_{j_i}$.

Для всех $r \in [r_l, r_{l-1})$ $l = \overline{1,k}$ ($r_0 = r_{\max}$) структура оптимального расписания постоянна и $E(r_l) < E(r_{l-\Delta})$ для всех $0 < \Delta < r_l - r_{l+1}$.

Оптимальное расписание построено, если найдено $r_l, l \le k$ такое, что:

$$E(r_l) + nr_l = E(0) \qquad (7)$$

Это означает, что выполняется признак 2) оптимальности допустимого расписания и r_{opt} - это одно из значений $r_l,\ r_{l-1}, ..., r_{\max}$, если $l < k$.

Алгоритм А2. Полиномиальная составляющая ПДС – алгоритм нахождения оптимального расписания по критерию минимизации суммарного опережения

ШАГ 1. Построить расписания, оптимальные по критерию 2 для моментов запуска $r_{\max}$ и 0. Определить соответствующие им суммарные опережения $E(r_{\max})$ и $E(0)$.

ЕСЛИ выполняется равенство $E(r_{\max}) + nr_{\max} = E(0)$, **ТО КОНЕЦ** – полученное для $r_{\max}$ расписание оптимально.

Полагаем: $l = 0$ – порядковый номер смены структуры расписания; $r_0 = r_{\max}$.

ШАГ 2.

2.1. $l = l + 1$

2.2. Найти длину l -го интервала Δ_l (по формуле (6)).

2.3. Сдвинуть влево текущую последовательность работ на величину Δ_l: $r_l = r_{l-1} - \Delta_l$

2.4. Построить для полученного момента запуска r_l оптимальное по критерию 2 расписание.

2.5. **ЕСЛИ НЕ** выполняется равенство $E(r_l) + nr_l = E(0)$,

ТО перейти к п.2.1.

ИНАЧЕ (выполняется признак оптимальности 2). Найти $E(r_{opt}) = \min\limits_{1 \le l \le k} E(r_l)$

КОНЕЦ вычислений.

Сложность полиномиальной составляющей $kO(n^2)$, где k - заданное количество изменений структур расписания. Покажем это. Наиболее трудоемкой частью алгоритма является пункт 2.2 ШАГА 2, который состоит в нахождении длины Δ_l l -го интервала (в котором структура расписания неизменна) и соответственно момента запуска $r_l = r_{l-1} - \Delta_l$, когда после применения алгоритма А1 в оптимальном расписании по критерию 2 для хотя бы одна работа изменит позицию относительно расписания с моментом запуска r_{l-1}. Для фиксированной пары работ s и j_i, проверка: $p_s < p_{j_i}$, $d_s < C_{j_i}$ осуществляется за время $\Theta(1)$. Всего существующих пар $\frac{n(n-1)}{2}$. Таким образом, верхняя оценка количества сравниваемых пар $O(n^2)$. Пусть общее количество смен структур k. Следовательно

сложность полиномиальной составляющей $kO(n^2)$.

Если на каком-то этапе выполняется признак оптимальности 2), то точное решение находится в отрезке $[r_l, r_{\max}]$. Если же на k структурах изменения расписаний признак оптимальности 2) не выполняется, то находим минимальное значение суммарного опережения на всех дискретных моментах изменения структур. И таким образом, получим приближенное решение, но точное $\forall r \in [r_l, r_{\max}]$. Это и есть полиномиальная аппроксимация точного алгоритма. ПДС - алгоритма второго типа построен.

Примеры нахождения оптимального расписания по критерию 2.

Пример 2.

Дано: $n = 6$;

$p_1 = 99$; $p_2 = 28$; $p_3 = 62$; $p_4 = 98$; $p_5 = 45$; $p_6 = 74$;

$d_1 = 240$; $d_2 = 260$; $d_3 = 249$; $d_4 = 514$; $d_5 = 481$; $d_6 = 449$.

В соответствии с алгоритмом ***A*** найдено расписание σ с максимально возможным моментом запуска $r_{\max} = 71$. После применения алгоритма ***A1*** минимальное суммарное опережение при максимальном моменте запуска $E(r_{\max}) = 264$. Окончательный вид расписания с минимальным суммарным опережением при максимальном моменте запуска приведен в таблице 1.

Табл. 1. Расписание, оптимальное по критерию минимизации суммарного опережения при $r_{\max}$

Порядковый номер в расписании	Номер работы, j	Длительность работы, p_j	Директивный срок, d_j	Момент окончания работы, C_j	Опережение работы, E_j
1	1	99	240	170	70
2	3	62	249	232	17
3	2	28	260	260	0
4	4	98	514	358	156
5	6	74	449	432	17
6	5	45	481	477	4
					$E(r_{\max}) = E(71) = 264$

Расписание с минимальным суммарным опережением с моментом запуска 0 приведен в таблице 2.

Табл. 2. Расписание, оптимальное по критерию минимизации суммарного опережения с моментом запуска 0

Порядковый номер в расписании	Номер работы, j	Длительность работы, p_j	Директивный срок, d_j	Момент окончания работы, C_j	Опережение работы, E_j
1	1	99	240	99	141
2	3	62	249	161	88
3	2	28	260	189	71
4	4	98	514	287	227
5	6	74	449	361	88
6	5	45	481	406	75
				E_0	690

Данный пример на первом шаге применения алгоритма *A2* дает оптимальное расписание по критерию 2. Проверим это: $E(r_{\max}) + nr_{\max} = E(0)$, $264 + 6 \cdot 71 = 690$. Из признака оптимальности 1) получили оптимальное расписание по ***критерию 2***, $E(r_{\max}) = E(r_{opt})$. Задача решена.

Пример 3.

Найдем оптимальное расписание по критерию 2 для следующей задачи:

$n = 6$;

$p_1 = 2$; $p_2 = 7$; $p_3 = 5$; $p_4 = 3$; $p_5 = 1$; $p_6 = 20$;

$d_1 = 30$; $d_2 = 43$; $d_3 = 49$; $d_4 = 52$; $d_5 = 54$; $d_6 = 56$.

Посчитаем, применяя алгоритм ***A2***, суммарное опережение с моментом запуска 0. В таблице 3 приведено расписание с моментом запуска 0, оптимальное по ***критерию 2***.

Суммарное опережение $E(0) = 99$.

Для нахождения максимального момента запуска применим алгоритм ***A1***. В таблице 4 приведено расписание с $r_{\max} = 18$, оптимальное по ***критерию 2***.

Суммарное опережение $E(r_{\max}) = 73$. Признак оптимальности 1) не выполняется ($E(r_{\max}) + nr_{\max} \neq E(0)$: $73 + 6 \cdot 18 = 181 \neq 99$).

Табл. 3. Оптимальное расписание с моментом запуска 0

Порядковый номер в расписании	Номер работы, j	Длительность работы, p_j	Директивный срок, d_j	Момент окончания работы, C_j	Опережение работы, E_j
1	6	20	56	20	36
2	2	7	43	27	16
3	1	2	30	29	1
4	3	5	49	34	15
5	4	3	52	37	15
6	5	1	54	38	16

Найдем минимальное значение среди всех возможных моментов изменения структуры расписания:

$$\Delta_1 = \min_{2\le i\le n}\left\{\underbrace{56-\max\{54,52,49,30,43\}}_{соотв.i=6};\ \underbrace{35-30}_{соотв.i=4};\ \underbrace{32-30}_{соотв.i=3}\right\} \quad (8)$$

Табл. 4. Оптимальное расписание с моментом запуска $r_{\max}$

Порядрядковый номер в расписании	Номер работы, j	Длительность работы, p_j	Директивный срок, d_j	Момент окончания работы, C_j	Опережение работы, E_j
1	2	7	43	25	18
2	1	2	30	27	3
3	3	5	49	32	17
4	4	3	52	35	17
5	5	1	54	36	18
6	6	20	56	56	0

Минимум в выражении (8) соответствует позиции 6: $\Delta_1 = 56-54=2$. Составляющие выражения (8), соответствующие позициям 2 и 5, не определены в силу того, что работы, стоящие на этих позициях, не удовлетворяют условиям (6). Сдвигаем последовательность работ на величину 2: $r_1 = r_{\max} - \Delta_1 = 18-2=16$. Построим оптимальное по критерию 2 расписание для $r_1 = 16$ (таблица 5).

Суммарное опережение $E(r_1)=63$. Признак оптимальности 2) не выполняется ($E(r_1)+nr_1 \ne E(0)$, $63+6\cdot 16=159\ne 99$). Найдем минимальное значение среди всех возможных моментов изменения структуры расписания:

$$\Delta_2 = \min_{2\le i\le n}\left\{\underbrace{53-\max\{52,49,30,43\}}_{соотв.i=5};\ \underbrace{33-30}_{соотв.i=4}\right\} \quad (9)$$

Табл. 5. Оптимальное расписание с моментом запуска r_1

Порядрядковый номер в расписании	Номер работы, j	Длительность работы, p_j	Директивный срок, d_j	Момент окончания работы, C_j	Опережение работы, E_j
1	2	7	43	23	20
2	3	5	49	28	21
3	1	2	30	30	0
4	4	3	52	33	19
5	6	20	56	53	3
6	5	1	54	54	0

Минимум в выражении (9) соответствует позиции 5: $\Delta_2 = 53-52=1$. Составляющие выражения (9), соответствующие позициям 2, 3 и 5, не определены в силу того, что работы, стоящие на этих позициях, не удовлетворяют условиям (6). Сдвигаем последовательность работ на величину 1: $r_2 = r_1 - \Delta_2 = 16-1=15$. Построим оптимальное по критерию 2 расписание для $r_2 = 15$ (таблица 6).

Табл. 6. Оптимальное расписание в момент запуска r_2

Порядрядковый номер в расписании	Номер работы, j	Длительность работы, p_j	Директивный срок, d_j	Момент окончания работы, C_j	Опережение работы, E_j
1	2	7	43	22	21
2	1	2	30	24	6
3	6	20	56	44	12
4	3	5	49	49	0
5	4	3	52	52	0
6	5	1	54	53	1

Суммарное опережение $E(r_2)=40$. Признак оптимальности 2) не выполняется ($E(r_2)+nr_2 \ne E(0)$, $40+6\cdot 15=130\ne 99$). Найдем минимальное значение среди всех возмож-

ных моментов изменения структуры расписания:

$$\Delta_3 = \min_{2\le i\le n}\left\{\underbrace{52-30}_{соотв.\,i=5};\ \underbrace{49-30}_{соотв.\,i=4};\ \underbrace{44-\max\{30,43\}}_{соотв.\,i=3}\right\} \qquad (10)$$

Минимум в выражении (10) соответствует позиции 3: $\Delta_3 = 44-43=1$. Составляющие выражения (10), соответствующие позициям 2 и 6, не определены в силу того, что работы, стоящие на этих позициях, не удовлетворяют условиям (6). Сдвигаем последовательность работ на величину 1: $r_3 = r_2 - \Delta_3 = 15-1=14$. Построим оптимальное по критерию 2 расписание для $r_3 = 14$ (таблица 7).

Табл. 7. Оптимальное расписание с моментом запуска r_3

Порядковый номер в расписании	Номер работы, j	Длительность работы, p_j	Директивный срок, d_j	Момент окончания работы, C_j	Опережение работы, E_j
1	1	2	30	16	14
2	6	20	56	36	20
3	2	7	43	43	0
4	3	5	49	48	1
5	4	3	52	51	1
6	5	1	54	52	2

Суммарное опережение $E(r_3)=38$. Признак оптимальности 2) не выполняется ($E(r_3)+nr_3 \ne E(0)$, $38+6\cdot 14=122\ne 99$). Найдем минимальное значение среди всех возможных моментов изменения структуры расписания:

$$\Delta_4 = \min_{2\le i\le n}\left\{\underbrace{53-30}_{соотв.\,i=3};\ \underbrace{36-30}_{соотв.\,i=2}\right\} \qquad (11)$$

Минимум в выражении (11) соответствует позиции 2: $\Delta_4 = 36-30=6$. Составляющие выражения (11), соответствующие позициям 4, 5 и 6, не определены в силу того, что работы, стоящие на этих позициях, не удовлетворяют условиям (6). Сдвигаем последовательность работ на величину 6: и $r_4 = r_3 - \Delta_4 = 14-6=8$. Построим оптимальное по критерию 2 расписание для $r_4 = 8$ (таблица 8).

Табл. 8. Оптимальное расписание с моментом запуска r_4

Порядковый номер в расписании	Номер работы, j	Длительность работы, p_j	Директивный срок, d_j	Момент окончания работы, C_j	Опережение работы, E_j
1	6	20	56	28	28
2	1	2	30	30	0
3	2	7	43	37	6
4	3	5	49	42	7
5	4	3	52	45	7
6	5	1	54	46	8

Суммарное опережение $E(r_4)=56$. Признак оптимальности 2) не выполняется ($E(r_4)+nr_4 \ne E(0)$, $56+6\cdot 8=104\ne 99$).

Найдем минимальное значение среди всех возможных моментов изменения структуры расписания:

$$\Delta_5 = \min_{2\le i\le n}\left\{\underbrace{45-30}_{соотв.\,i=5};\ \underbrace{42-30}_{соотв.\,i=4};\ \underbrace{37-30}_{соотв.\,i=3}\right\} \qquad (12)$$

Минимум в выражении (12) соответствует позиции 3: $\Delta_5 = 37-30=7$. Составляющие выражения (12), соответствующие позициям 6 и 2, не определены в силу того, что работы, стоящие на этих позициях, не удовлетворяют условиям (6). Сдвигаем последовательность работ на величину 7: и $r_5 = r_4 - \Delta_5 = 8-7=1$. Построим оптимальное по критерию 2 расписание для $r_5 = 1$ (таблица 9).

Табл. 9. Оптимальное расписание с моментом запуска r_5

Порядковый номер в расписании	Номер работы, j	Длительность работы, p_j	Директивный срок, d_j	Момент окончания работы, C_j	Опережение работы, E_j
1	6	20	56	21	35
2	2	7	43	28	15
3	1	2	30	30	0
4	3	5	49	35	14
5	4	3	52	38	14
6	5	1	54	39	15

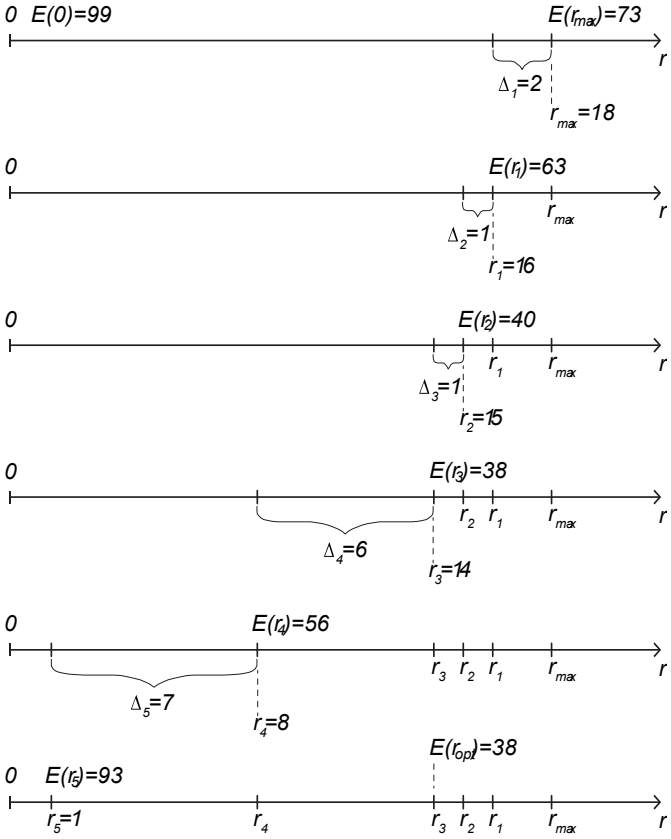


Рис.4. Пошаговый процесс нахождения оптимального расписания по критерию минимизации суммарного опережения

Суммарное опережение $E(r_5)=93$. Проверим, выполнение признака оптимальности 2). Признака оптимальности 2) допустимого расписания выполняется ($E(r_5)+nr_5 \neq E(0)$ $E_{r_5}+nr_5=E_0$, $93+6\cdot 1=99$). Оптимальное расписание построено. При этом r_{opt} находится в отрезке $[1,18]$. Найдем $E(r_{opt})=\min\limits_{1\le l\le 5} E(r_l)=38$. Таким образом, оптимальное расписание по критерию 2 совпадает с расписанием в момент запуска r_3 (см.таблица 7). Проиллюстрируем процесс решения графически (рисунок 4).

Выводы

Рассмотрена актуальная задача теории расписаний выполнения множества работ с различными директивными сроками одним прибором, в которых все работы должны быть выполнены без нарушения их директивных сроков и минимизировано суммарное отклонение моментов окончания работ от их директивных сроков. Данная задача найдет широкое применение в современных условиях планирования точно в срок. Проведено исследование свойств этой задачи. Определены условия признака оптимальности расписания по критерию минимизации суммарного опережения выполнения работ и на их основе разработан ПДС - алгоритм нахождения оптимального расписания.

Список литературы

1. Згуровский М. З. Принятие решений в сетевых системах с ограниченными ресурсами [Текст]: монография / М. З. Згуровский, А. А. Павлов. – К.: Наукова думка, 2010.– 573 с.
2. Павлов А.А. Исследование свойств задачи календарного планирования для одного прибора по критерию минимизации суммарного опережения заданий при условии допустимости расписании / А. А. Павлов, Е.Б. Мисюра, О.А. Халус // Вісник НТУУ “КПІ”. Серія «Інформатика, управління та обчислювальна техніка». – К.: “ВЕК+”, 2012. – №56.– С. 98–102.
3. Згуровский М.З. Задача построения допустимого расписания с максимально поздним моментом запуска и минимальным суммарным опереженем / М.З.Згуровский, А. А. Павлов, О.А. Халус // // Системні дослідження та інформаційні технології. — 2014. — № 3. — 12 с.
4. Павлов А.А. ПДС-алгоритмы решения задач составления расписаний по критерию опережения/запаздывания на одном приборе [Текст] / А.А. Павлов, Е.Б. Мисюра // Вісник НТУУ “КПІ”. Серія «Інформатика, управління та обчислювальна техніка». – К.: “ВЕК+”, 2014. – №60. – 16 с.

УДК 004.451.24

СТІРЕНКО С.Г.

РЕАЛІЗАЦІЯ МЕТОДУ ВИЗНАЧЕННЯ МОМЕНТУ ВІДКЛАДЕННЯ ПЕРЕСИЛАННЯ ДАНИХ В КОМП'ЮТЕРНИХ КЛАСТЕРНИХ СИСТЕМАХ

В роботі розглядаються реалізація методу визначення моменту відкладення передачі даних, що ефективно використовуються під час виконання балансування навантаження. На основі проведених експериментів та аналізу їх результатів можна стверджувати, що метод визначення моменту відкладання пересилання даних може бути ефективно застосовано для підвищення функціонування паралельних кластерних систем.

We consider the implementation of the method for determining the date of the deposition of data that are being used while performing load balancing. Based on the experiments and analyze their results could be argued that the method of determination of a data transmission delay can be effectively used to improve the functioning of parallel cluster systems.

Особливості реалізації в кластерних системах

Основними системами з локальною пам'яттю на даний момент є кластерні системи, що складаються з вузлів, кожен з яких має спільну пам'ять для декількох процесорів, з'єднаних високопродуктивною мережею. Переважна більшість існуючих кластерних систем надає доступ до низькорівневого інтерфейсу передачі повідомлень MPI як до інтерфейсу програмування обчислювальної системи. Структура цього інтерфейсу дозволяє абстрагувати апаратні особливості тієї чи іншої системи, зокрема систему зв'язків та нестрогу неоднорідність обчислювальних вузлів, дозволяючи виконувати обчислення на будь-якому з них та організовувати передачу даних між будь-якою парою вузлів. Таким чином, MPI є фактично інтерфейсом до більшості кластерних систем, відповідно доцільним є використовувати його для безпосередньої передачі даних, тобто ввести додатковий рівень абстракції над MPI. Змінення поведінки безпосередньо інтерфейсу MPI небажана, оскільки може вплинути на вже існуючі програми з його використанням, які розраховують на відому поведінку.

Додатковий рівень абстракції може також забезпечити підтримку моделі, що використовує підзадачі зі шляхами розбиття для забезпечення прогнозування часу передачі даних між вузлами та часу виконання обчислень підзадачі. На цьому рівні можна виконувати збір різного роду статистичної інформації із застосуванням інтерфейсу визначення продуктивності MPI performance API, який надає платформонезалежні методи доступу до інформації про показники продуктивності, що є вкрай важливим для підтримки достатньо великої кількості обчислювальних систем. Робота на тому ж самому рівні, що й MPI, вимагала б повторної реалізації тих самих механізмів. Аналогічним чином, додатковий рівень абстракції з використанням поділу задач на частини був запропонований в технології Intel Threading Building Blocks [1] для обчислювальних систем зі спільною пам'яттю, однак його модель не передбачає необхідність передачі даних. Тому доцільно зберігати сумісність інтерфейсів з TBB, якщо можливо. З точки зору доступних користувачу дій, основними механізмами розпаралелювання є паралельні згортка та сканування, причому конкретна реалізація паралельного обчислення не визначається, тому існує можливість використовувати такий самий інтерфейс. З іншого боку, підтримка атомарних операцій або взаємного виключення технічно неможлива в системах з локальною пам'яттю, тому ця частина інтерфейсу TBB не має підтримуватись.

Таким чином найбільш доцільним підходом до реалізації запропонованого відкладення передачі даних [2,3,4] є введення додаткового рівня абстракції над MPI, який дозволить виконувати необхідні дії та вимірювати певні метрики ефективності. Це також дозволить користувачу обчислювальної системи описувати способи передачі даних з використанням стандартизованих засобів MPI, однак момент безпосереднього виклику цих засобів буде визначатися системою під час виконання.

На цьому рівні необхідно реалізувати підтримку необхідних механізмів для відкладення передачі даних. Оскільки на даний

момент безпосередньо MPI не надає подібної можливості окрім дуже обмеженої підтримки буферизованих передач, необхідно ввести додатковий керуючий процес, який буде оброблювати запити за передачу даних. За умови відкладення початку передачі даних, у процес, що має їх приймати, замість безпосередньо даних відправляється спеціальна структура даних – дескриптор, – яка містить інформацію про обсяг даних та їх розташування на деякому вузлі. Ця структура може бути використана для виконання явного запиту даних, якщо вони необхідні раніше, ніж були передані. Також такий підхід дозволяє передавати лише дескриптори, розміри яких зазвичай значно менше обсягів даних, що передаються, при перенесенні обчислень між вузлами. У керуючому процесі підтримується впорядкований за часом список відкладених передач. Для визначення часу використовуються високоточні таймери, наявні в сучасних процесорах, роздільна здатність яких досягає наносекунд. [5] Так як в рамках вузла використовується система, побудована за принципами розподілу часу, необхідно врахувати її особливості під час ініціювання початку передачі даних. Для цього виявляється кількість процесів, що знаходяться в активному стані. Якщо передача даних має бути ініційована менш ніж через час, що дорівнює отриманій кількості процесів на квант часу, передачу можна ініціювати негайно.

Типи задач, для яких виконувалось тестування

Для демонстрації принципів роботи запропонованих підходів відкладення початку передачі даних необхідно застосувати їх до ряду типових обчислювальних задач, які виконуються на системах з локальною пам’яттю. Оскільки запропонована для досліджень реалізація використовує додатковий рівень абстракції над низькорівневим інтерфейсом передачі повідомлень, неможливо безпосередньо використати її для існуючих програм із застосуванням MPI, а необхідно виконати їх адаптацію. Тому було обрано ряд типових задач, які мають характерні особливості, поведінку та схеми взаємодії, та адаптовано для використання дослідницької реалізації.

З точки зору організації взаємодії та обсягів даних, що передаються між вузлами обчислювальної системи з локальною пам’яттю, а також обсягів пам’яті, що використовуються у вузлах, можна виділити наступні широкі класи задач.

а) Задачі з рівномірним використанням пам’яті на вузлах та рівномірним обсягом даних, що передаються між вузлами. При цьому, взаємодія може відбуватись лише між обраними парами вузлів, однак необхідною умовою є стабільність середньої кількості переданих даних за певний проміжок часу між всіма парами вузлів. Прикладами таких задач є операції лінійної алгебри із щільними векторами, матрицями та тензорами, що зберігаються простим способом, зокрема матрично-тензорна алгебра, одно- та багатовимірні згортки, гаусово розмиття, дискретні перетворення в системах ортогональних функцій, зокрема перетворення Фур’є; розв’язок диференційних рівнянь на рівномірних сітках методами кінцевих різниць тощо.

б) Задачі, в паралельних реалізаціях яких комунікація між процесами здійснюється відповідно до певного невипадкового закону, зокрема поліноміальному, причому обсяги комунікації між певними парами процесів значно більше ніж між іншими парами процесів. Використання пам’яті у вузлах також може бути нерівномірним, але невипадковим, причому не є обов’язково безпосередньо пов’язаним з обсягами комунікації. До цього класу можуть бути віднесені задачі чисельного математичного аналізу, зокрема чисельного інтегрування зі складними характеристиками сходимості; ітераційні методи, що виконуються на інтервалах до сходимості, зокрема методи розв’язку систем лінійних рівнянь; розв’язок диференційних рівнянь методами кінцевих різниць на кінцевих елементів з адаптивними або нерівномірними сітками тощо.

в) Задачі, в яких обсяги комунікації між процесами при паралельному розв’язані можуть варіюватися випадково в залежності від значень вхідних даних, а не від їх обсягу, або задачі, в яких обсяги використаної пам’яті можуть випадковим чином залежати від значень вхідних даних або результатів, отриманих при попередніх обчисленнях. Прикладами таких задач є задачі лінійної алгебри над довільними розрідженими матрицями з випадковим заповненням; ймовірнісні методи, зокрема методи Монте-Карло; моделювання випадкових процесів; деякі обчислювально складні задачі на графах.

Така класифікація не претендує на повноту або покриття всіх можливих обчислювальних задач, але відображає достатньо велику кількість типових обчислень, що виконуються

на кластерних системах. Крім того, вона дозволить достатньо швидко оцінити ефективність застосування запропонованих підходів до відкладення передачі даних без необхідності реалізовувати велику кількість задач із застосуванням цих підходів.

Для проведення експерименту з ціллю підтвердження впливу на коефіцієнт ефективності запропонованих підходів та перевірки висунутих гіпотез, було обрано та реалізовано із застосуванням моделі розбиття на підзадачі та зв'язані з ними дані, по дві наступні задачі кожного класу:

а) виконання дискретного перетворення Фур'є; чисельне розв'язання диференційного рівняння квантової оптики методом кінцевих різниць з рівномірною сіткою;

б) чисельне інтегрування методом кривих вищих порядків; чисельне розв'язання диференційного рівняння квантової оптики методом кінцевих різниць з адаптивної сіткою;

в) знаходження власних чисел та векторів розрідженої матриці з випадковим заповненням; моделювання випадкового фізичного процесу методом Монте-Карло.

Обрані задачі використовуються у багатьох більш складних застосуваннях обчислювальних систем для розв'язку наукових та технічних задач, тому можливість збільшення коефіцієнту ефективності їх розв'язання дозволить також збільшити коефіцієнти ефективності виконання обчислень, які активно використовують розв'язання цих задач.

Крім того, оскільки обрані задачі та особливо їх конкретні можуть мати чітко виражені особливості організації взаємодії між паралельними процесами, в яких виконується їх обчислення, та схеми використання пам'яті, додатково було побудовано програмні моделі задач кожного класу.

Так для моделювання задачі класу (а) для кожної пари процесів визначається, чи буде виконуватися моделювання взаємодії між ними: якщо рівномірно розподілене на проміжку [0; 1) випадкове значення r перевищує певне порогове значення $r_{thresh} = 0.75$, то процеси вважаються взаємодіючими. Обрання порогового значення може залежати від кількості паралельних процесів, оскільки зазвичай у разі великої кількості процесів, взаємодія відбувається лише між окремими парами процесів. Кожний процес, що моделюється, являє собою чергування фази обчислень та фази передачі даних. Для моделювання обчислень обирається час їх виконання tc як випадкове значення, розподілене за нормальним законом $N(\mu,\sigma)$, зі значеннями $\mu = 10$ мс, $\sigma = 1$ мс, після чого в процесі виконується цикл, умовою виходу з якого є досягнення встановленого часу t_c. В циклі відбувається звернення до випадкових адрес пам'яті, яка доступна даній програмі, причому наступна адреса обирається як випадкова величина, розподілена за нормальним законом, з математичним очікуванням рівним наступній адресі для задовільнення принципу локальності звернення до даних. Під час обчислень виділяється певний обсяг пам'яті M, який обчислюється як $M = \frac{19m}{20} + m_{rand}$, де $m = 2^{29}$ байт – фіксований обсяг виділеної пам'яті, а m_{rand} – випадкова компонента обсягу пам'яті, що розподілена за нормальним законом $N(\frac{m}{10}, \frac{m}{100})$ та має на меті урахування побічних факторів, що можуть впливати на обсяги виділеної пам'яті. Виділена пам'ять заповнюється випадковими даними. Моделювання фази взаємодії виконується наступним чином: з використанням рівномірно розподіленого випадкового значення та порогу rthresh визначається, чи буде відбуватися взаємодія між парою процесів в даній фазі, якщо так, випадковим чином визначається процес-джерело та процес-приймач даних. Після чого визначається обсяг даних, які будуть передаватися $S = \frac{19m}{20} + s_{rand}$, де $s = 2^{22}$ байт – фіксований обсяг даних, що будуть передаватися, а s_{rand} – його випадкова компонента, розподілена за нормальним законом аналогічним до обсягів пам'яті. Виконується передача необхідного обсягу даних між вузлами, значення даних генеруються випадково.

Моделювання задачі класу (б) аналогічним до (а) чином визначає пари процесів, між якими відбуватиметься взаємодія. Однак в цьому випадку необхідно внести нерівномірність обсягів використаної пам'яті та даних. Для цього перед початком моделювання обирається два поліноми PM та PS від номеру процесу i: $P_4(i) = p_4 i^4 + p_3 i^3 + p_2 i^2 + p_1 i + p_0$, де p_i – коефіцієнти, що є випадковими значеннями, розподіленими за нормальним розподілом на проміжку [0; 3). В подальшому поліном $P^M(i)$ використовується для визначення обсягів пам'яті, що виділяються в i-тому вузлі, а поліном $P^S(i)$ – для визначення обсягів даних, що мають бути передані з i-го вузла у певний інший. Для отримання абсолют-

них значень обсягів даних значення поліномів домножаться на $\frac{10m}{P^M(N)}$ та $\frac{10s}{P^S(N)}$ відповідно, де N – кількість процесів.

Моделювання задач класу (в) відбувається наступним чином. На кожній фазі передачі даних для кожної пари процесів обирається чи буде відбуватися між ними взаємодія аналогічно до (а), однак на відміну від нього, будь-яка пара процесів потенційно може виконувати обмін даними. Моделювання процесу обчислень також відбувається за тими самими принципами, однак обсяги виділеної пам'яті визначаються як випадкова величина, розподілена за рівномірним законом на проміжку $[0.1m, 10m)$. Аналогічно обсяги даних, що будуть передані між кожною парою вузлів обираються як незалежні випадкові значення, розподілені за рівномірним законом на проміжку $[0.1s, 10s)$.

Наведені конкретні значення m та s, а також проміжки, у яких вибираються випадкові значення для обсягів пам'яті та даних, обрані у відповідності до статистики виконання задач на кластері Суперкомп'ютерного центру НТУУ «КПІ», типової конфігурації обладнання для обчислювальних систем такого типу (1 Гб оперативної пам'яті на один обчислювальний процес, та використання 10 Гб/с зв'язку між вузлами) а також з урахуванням загальної статистики [5], зокрема різниця між мінімальним та максимальним обсягом даних обрана як два десяткових порядки.

Використання пам'яті у вузлах системи

Процес виконання обчислень на обчислювальній системі з локальною пам'яттю передбачає насамперед необхідність зберігання даних в локальній пам'яті одного з вузлів цієї системи. Оскільки дані є необхідними для виконання обчислень або є їх результатом, зменшити обсяги пам'яті шляхом відмови від їх зберігання неможливо, за винятком випадків відміни обчислень, розглянутих у другому розділі. Однак, у час між готовністю даних до передачі в одному вузлі та їх безпосереднім використанням у іншому вузлі, існує можливість обрати один з них для зберігання цих даних, що в даній роботі виконувалось з огляду на обсяг пам'яті, зайнятої на кожному вузлі з ціллю його мінімізації. Зберігання занадто великих обсягів може непрямо впливати на ефективність виконання обчислень, а в деяких випадках навіть привести до зупинки обчислень через перевищення доступних обсягів пам'яті, так як ряд обчислювальних систем з локальною пам'яттю, зокрема великі кластерні системи, можуть не мати вузлів, обладнаних дисками, що використовуються для підкачки даних в основну пам'ять.

На рис. 1 зображено обсяги пам'яті, що в середньому були використані одним процесом за весь час роботи програми, усереднена для всіх програм, для яких виконувалось тестування, та згрупована за способами визначення тривалості відкладення передачі.

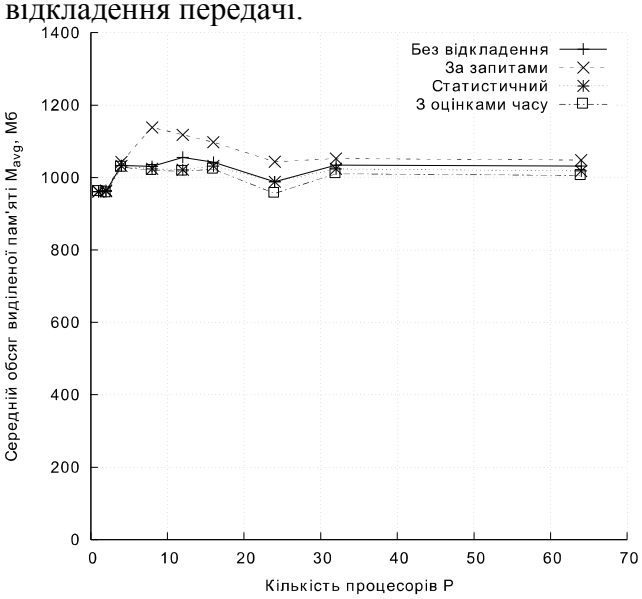


Рис. 1. Середній за час виконання обчислень та за всіма процесами обсяг виділеної пам'яті в перерахунку на один процес

Окремо зазначимо, що обсяг фізичної пам'яті на вузлах системи, яка використовувалась для тестування, складає 1024 Мб. на кожний наявний процесор. Програми виконання обчислень отримували цю інформацію від системи керування задачами на кластері, однак не враховували обсяги пам'яті, що використовуються системними процесами. Крім того, оскільки один вузол обчислювальної системи містить декілька процесорів, а обчислення могли виконуватись на одному або меншій кількості, існувала можливість використовувати всю пам'ять, наявну у вузлі без задіяння механізмів підкачки.

Відкладення передачі даних до запиту на їх використання прогнозовано збільшило середній обсяг використаної пам'яті у вузлах, оскільки дані, передачу яких відкладено, необхідно зберігати у джерелі. Слід відмітити, що при меншій кількості процесорів, обсяг зайнятої пам'яті суттєво збільшився, в деяких випадках збільшення склало до 13%. Це може бути пояснено більшими обсягами даних, що мають передаватися між парами вузлів, якщо їх невелика кількість, та меншою кількістю вузлів, які мог-

ли вже запитати дані на момент їх готовності. Загалом, використання підходу за запитами збільшує середні обсяги використаної пам'яті на 2 - 4%, однак в деяких випадках може зменшити пікові обсяги виділення пам'яті.

Використання підходу з відкладенням початку передачі на час, що визначається статистично дозволило незначно зменшити середній обсяг використаної пам'яті, на 1 - 2%, що викликано здебільшого наявністю правила про заборону відкладення при значній різниці в обсягах виділеної пам'яті. Використання підходу із застосуванням оцінок часу передачі даних та виконання дозволило зменшити обсяги використаної пам'яті до 5.3% при наявності балансування навантаження в системах з лінеарізуємими залежностями швидкостей передачі даних, та до 2.7% в системах з довільними залежностями швидкостей передачі даних.

Провал в обсягах використаної пам'яті для 24 процесорів, найбільш ймовірно пояснюється нерівномірністю розбиття на підзадачі, оскільки використовуваний алгоритм розділення на половини має продукувати кількість підзадач з приблизно однаковими обсягами даних, що є певним ступенем двійки.

За відсутності балансування навантаження у задачах, які не допускають відміну обчислень, середні обсяги використаної пам'яті залишилися незмінними.

Час очікування введення та виведення

Очікування початку передачі даних за мережею, на відміну від часу безпосередньої передачі даних, передбачає відсутність виконання обчислень через неготовність даних, що в свою чергу може суттєво вплинути на час виконання обчислень та на коефіцієнт ефективності.

На рис. 2 наведено залежність відношення часу очікування введення та виведення, до якого входить як час передачі даних мережею, так і робота з дисковою підсистемою, за її наявності на локальних вузлах, до загального часу виконання програми від кількості процесорів, що використовуються для обчислень. Тобто за віссю ординат знаходиться відсоток часу, який, в середньому для всіх процесів, займає час очікування введення та виведення від часу виконання обчислень. При цьому на графіку наведено усереднене для всіх задач та способів визначення часу передачі даних значення. Доцільно зображати саме відсоток, а не абсолютні значення, оскільки тривалість виконання обчислень сильно змінюється зі зміною кількості процесорів, причому різним чином для різних класів задач.

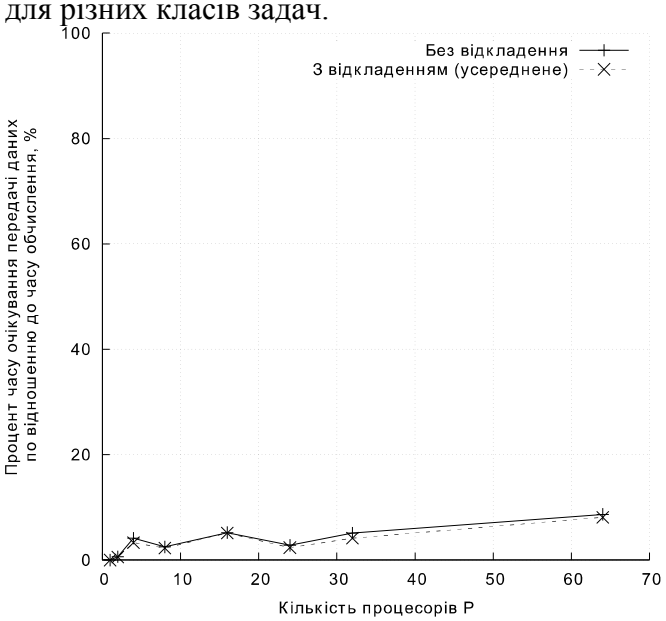


Рис. 2. Відношення часу очікування передачі даних до загального часу виконання програми, усереднене за всіма процесами

При невеликих кількостях процесорів, що використовуються для виконання обчислень, відсоток часу очікування передачі даних майже не змінився. Це пояснюється відносно великим часом виконання обчислень по відношенню до очікування. З іншого боку, час очікування передачі даних із застосуванням відкладення передачі даних зазнав зменшення до 43% по відношенню до цієї ж величини без відкладення, однак це не мало суттєвого впливу на ефективність виконання обчислень. Із збільшенням кількості процесорів, на яких виконуються обчислення, вплив на відсоток часу, що займає очікування введення та виведення, стає більш суттєвим завдяки двом факторам: по-перше, при великій кількості процесорів збільшується кількість пар вузлів, що комунікують один з одним, та потенційно створюють ситуації очікування, і по-друге, зменшується загальний час виконання обчислень і очікування має більш суттєве значення. Крім того, зростає час очікування введення та виведення через збільшення варіантів обрання пар процесів для передачі даних. При використанні 32 процесорів відсоток часу очікування введення та виведення по відношенню до загального часу виконання обчислень було зменшено з 5.1% до 4.1%. Очікується, що з подальшим збільшенням кількості процесорів за умови ба-

лансування навантаження і невпорядкованих передач, різниця в часі очікування лише збільшуватиметься.

Вплив на коефіцієнт ефективності

Покращення двох вищерозглянутих факторів: обсягів використаної у вузлах пам'яті та часу очікування введення та виведення даних – мали на меті вплинути на ефективність виконання обчислень, яка в свою чергу виражається як у швидкості виконання обчислень, так і у завантаженості апаратного забезпечення. Необхідно розглянути вплив на ефективність виконання обчислень окремо для різних класів задач, оскільки вони самі по собі демонструють значення коефіцієнту ефективності, що суттєво відрізняються, особливо при масштабуванні за кількістю процесорів, на яких виконуються обчислення. Тим не менш, необхідно згрупувати результати, отримані для різних обмежень на швидкості передачі даних між вузлами.

На рис. 3 зображено яким чином змінюється коефіцієнт ефективності в задачах, які обмінюються однаковими обсягами даних між обраними парами вузлів. Неважко помітити, що коефіцієнт ефективності незначно спадає при збільшенні кількості процесорів завдяки рівномірності розподілу обсягів даних, що дозволяє статистично їх зпрогнозувати. При невеликих кількостях процесорів, найбільш суттєве збільшення коефіцієнту ефективності досягається при використанні відкладення на статичний визначений час, при цьому із збільшенням кількості процесорів таке відкладення стає менш ефективним, зокрема для 64 процесорів його застосування взагалі стає недоцільним через зменшення коефіцієнту ефективності. Це має бути пояснене двома факторами: з одного боку, зі збільшенням кількості процесорів зменшується мінімальний розмір блоку даних, для яких виконується обчислення, тобто зменшується час виконання обчислень, а отже статично заданий час стає порівняно більшим; з іншого боку, більша ймовірність того, що один із процесорів надішле запит на використання даних за проміжок часу, на який була відкладена їх передача.

З іншого боку, застосування підходів, що базуються на зборі статистичної інформації про виконання обчислень та передачі даних демонструють незначне збільшення коефіцієнту ефективності в одних випадках та зменшення в інших. Зменшення переважно спостерігається при виконанні обчислень, в яких відбувається передача фіксованих обсягів даних між впорядкованими парами процесів, якщо канали передачі даних мають майже однакові характеристики. Разом з тим при використанні каналів передачі, що мають суттєво різні обмеження, застосування таких підходів дозволяє збільшити коефіцієнт ефективності.

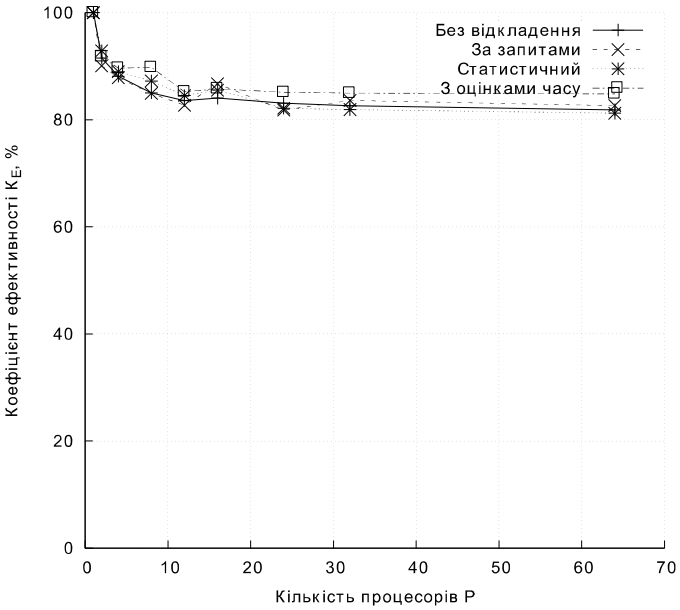


Рис. 3. Залежність коефіцієнту ефективності від кількості процесорів для обчислень з однаковими обсягами даних, що передаються між вузлами

Застосування підходу з оцінками часу майбутньої передачі даних дозволяє збільшити коефіцієнт ефективності на 4 - 6%, оскільки ймовірнісна модель, що використовується для оцінки легко адаптується до рівномірних обсягів та типів даних, що передаються, тому оцінки часу передачі, виконані з використанням цієї моделі, близькі до реальної тривалості передачі даних.

На рис. 4 показано зміну коефіцієнту ефективності при виконанні обчислень на різній кількості процесорів у випадку задач, в яких обсяг обчислень або обсяги даних, що передаються між вузлами, можуть поліноміально залежати від номерів вузлів. Результати згруповані за підходами, що використовувалися для обчислення тривалості відкладення передачі даних, та усереднені за усіма задачами, що входять до даного класу. Слід зазначити, що експерименти виконувались для задачі чисельного інтегрування, для якої обсяги обчислень змінюються лінійно на всьому проміжку, та для чисельного розв'язання диференційного рівняння в двовимірному просторі, в якому обсяги даних, що передаються, змінюються квад-

ратично від центру області розв'язання. Програмна модель, з іншого боку, дозволила моделювати виконання обчислень з поліноміальними залежностями до п'ятого ступеню, як описано в попередньому розділі.

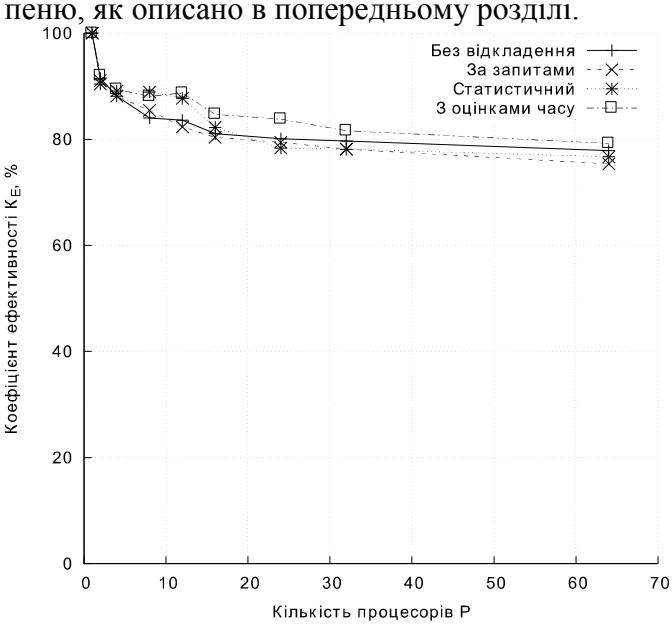


Рис. 4. Залежність коефіцієнту ефективності від кількості процесорів для обчислень з поліноміальною залежністю обсягів даних від номеру процесу

Для невеликої кількості процесорів, до шістнадцяти, усі підходи до обчислення тривалості відкладення початку передачі даних демонструють приблизно однакове збільшення коефіцієнту ефективності. Найбільш істотне збільшення, з 83.5% до 88.7% зафіксовано при використанні 12 процесорів. Підхід до визначення тривалості відкладення за запитами зі збільшенням кількості процесорів призводить до зменшення коефіцієнту ефективності, навіть у порівнянні з відсутністю відкладення передачі даних через збільшення часу очікування передачі. Тим не менш, при невеликій кількості процесорів через більш суттєву нерівномірність розподілу обсягів даних, що передаються, цей підхід дозволяє збільшити коефіцієнт ефективності на 0.2% завдяки перерозподілу обсягів виділеної пам'яті.

Статистичний підхід при великій кількості процесорів може також призвести до незначного зменшення коефіцієнту ефективності, так як передачі даних можуть виконуватися між більшою кількістю пар процесорів що вимагає більш тривалого часу збирання статистики для визначення часу передачі, тому статистичні оцінки можуть бути невірними. Також зі збільшенням кількості процесорів зменшується перевага від використання відкладених передач даних у відносних одиницях, що не перешкоджає однак подальшому зменшенню часу виконання обчислень.

Зміна коефіцієнту ефективності, усередненого за всіма задачами, для яких виконувались експерименти, та згрупована за підходами до визначення тривалості відкладення передачі даних, в залежності від кількості процесорів представлена на рис. 5. Слід відмітити більш різкий спад ефективності при нарощуванні кількості процесорів через явну нерівномірність обсягів обчислень або даних, що передаються в цьому випадку.

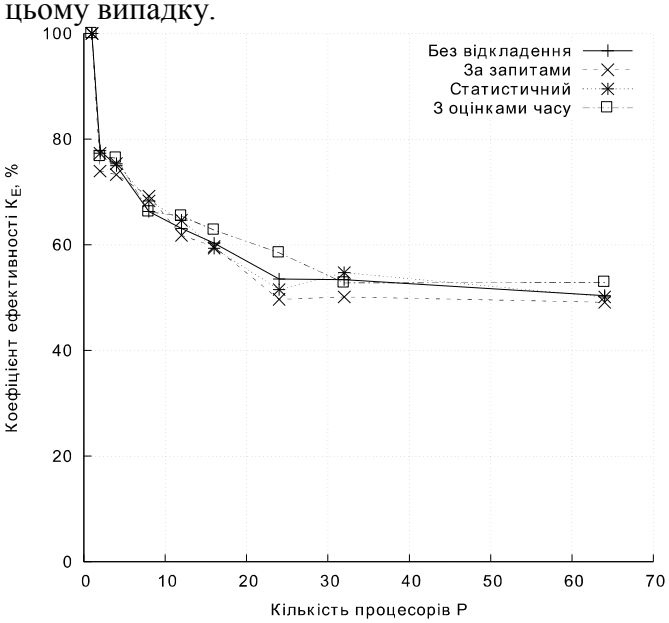


Рис. 5. Залежність коефіцієнту ефективності від кількості процесорів для обчислень, в яких обсяги даних не залежать від номеру процесу

Застосування підходу визначення тривалості відкладення передачі даних за запитами майже в усіх випадках призвела до зменшення коефіцієнту ефективності, яке не було нівельоване ефектами від зміни обсягів пам'яті через випадковий вплив на них в процесі обчислень. Для невеликої кількості процесорів, більшість методів дозволяють отримати коефіцієнти ефективності в межах ±1% у порівнянні до відсутності відкладення, що дозволяє використовувати запропоновані підходи навіть у цьому випадку без суттєвих втрат ефективності.

Використання підходу з оцінками часу при наявності 8 або менше процесорів, здебільшого призвело до зменшення коефіцієнту ефективності, що пояснюється неадаптованістю моделі оцінки до випадкових обсягів даних, які пере-

силаються, тому ймовірнісна модель оцінки могла зафіксувати неіснуючу залежність між обсягами даних. Однак зі збільшенням кількості процесорів, завдяки збільшенню кількості факторів, що впливають на процес обчислень загалом, оцінка часу передачі даних наблизилась до нормального розподілу, що дозволило підходу з оцінками часу демонструвати збільшення ефективності до 4.6%.

Застосування статистичного підходу без оцінок в деяких випадках дозволило збільшити коефіцієнт ефективності до 1.3%, що однак не є загальною тенденцією для всіх обчислювальних задач, для яких проводились експерименти.

Висновки

Запропоновані підходи до відкладання початку передачі даних між вузлами обчислювальної системи на певний час з огляду на збільшення ефективності та оптимізацію використання пам'яті можуть бути реалізовані як додатковий рівень абстракції над низькорівневим інтерфейсом передачі повідомлень MPI, оскільки останній являє собою інтерфейс для організації взаємодії між вузлами в більшості сучасних кластерних систем. Крім того, це дозволить використовувати існуючі оптимізовані реалізації MPI та дозволить користувачеві системи описувати спосіб передачі даних із використанням відомих засобів. При цьому також доцільно реалізувати одну із абстракцій більш високого рівня над передачею повідомлень, зокрема абстракцію розбиття і об'єднання підзадач, яка дозволить у більш простому вигляді описувати структуру задач для подальших експериментів.

Реалізація для сучасних обчислювальних систем з локальною пам'яттю має ряд особливостей, зокрема необхідність урахувати під час реалізації наявність в системі процесів, що виконують функції операційної системи. Також можна урахувати оптимізації реалізацій низькорівневого інтерфейсу передачі повідомлень для випадків, коли передача повідомлень здійснюється між процесами, які фактично мають доступ до спільної пам'яті. В цьому випадку недоцільно виконувати відкладення, оскільки безпосередньої передачі даних не відбувається, а лише надається доступ до певної частини пам'яті для іншого процесу на тому ж самому вузлі.

Для реалізації передачі даних за запитом запропоновано передавати короткі дескриптивні структури даних, в яких міститься інформація про реальне місцезнаходження даних, їх структуру ту та обсяг. При цьому, у випадку переміщення процесу обчислень на інший вузол, необхідно перемістити лише цей дескриптор, а безпосередньо дані можуть переміщені пізніше. Додатково, у моделях, що підтримують розбиття задачі за даними, наприклад у задачах масової обробки даних, існує можливість алгоритмічно визначати нові дескриптори без звернення безпосередньо до даних, якщо це допускається обчислювальною задачею.

Щоб отримати статистику виконання обчислень на реальній системі, використовується ряд інтерфейсів доступу до апаратних лічильників продуктивності, а також інструментування викликів інтерфейсу передачі повідомлень. Дані про виконання обчислень збираються безпосередньо під час їх виконання та зберігаються у файл трасування для подальшої обробки, оскільки аналіз під час виконання обчислень може вплинути на вимірювання через необхідність проведення складних обчислень.

Список посилань

1. Pheatt, Chuck. Intel ® threading building blocks / Chuck Pheatt //Journal of Computing Sciences in Colleges.– 2008.– Vol. 23, no. 4.– P. 298–298.
2. Стіренко С.Г. Метод перебалансування навантаження в кластерних комп'ютерних системах в динамічному режимі // Вісник НТУУ «КПІ». Інформатика, управління та обчислювальна техніка. – 2014. –№. 60. – С. 55-59.
3. Стіренко С.Г. Модель технології програмування паралельних систем // Вісник НТУУ «КПІ» Інформатика, управління та обчислювальна техніка. – 2013. – Т.58. – С.158-164.
4. Стіренко С.Г. Організація відкладених обчислень в кластерних системах // Вісник НТУУ «КПІ» Інформатика, управління та обчислювальна техніка. – 2013. – Т.59. – С.12-16.
5. Hennessy, John L. Computer architecture: a quantitative approach / John L Hennessy, David A Patterson. – Amsterdam, Noord-Hooland, Netherlands: Elsevier, 2011. – 824 p.

УДК 004.93(015.7)

КОВАЛЬ О.В.,
ЗАЙЦЕВА К.А.

ВЕРИФІКАЦІЯ КОМП'ЮТЕРНОЇ МОДЕЛІ СИСТЕМИ ІНФОРМАЦІЙНОГО УПРАВЛІННЯ

Важливим етапом процесу розробки складних програмних систем (ПС) на сучасному етапі є комп'ютерне моделювання . Верифікація комп'ютерної моделі СІУ є важливим етапом методології модельної розробки ПС, що дозволяє оцінити комплексність ПС. Запропоновано аналітичну модель СІУ з дієвою аналітикою, якою передбачено основні складові елементи дієвої аналітики: ціль – сценарій – технологічне середовище – метадані – дані. Пропонується три критерії перевірки комп'ютерної моделі СІУ:цілісність, повнота, несуперечність. Розглянуто основні етапи перевірки сценарної моделі, відповідно до поставлених критеріїв.

An important stage in the development of complex software systems (PS) nowadays is a computer simulation. Verification of MIS computer model is an important step of model-driven methodology to assess the complexity of the PS. Analytical model of MIS with advanced analytics, which provides the basic components of an advanced analytics: goal- scenario - software factory - metadata - data. Proposed three criteria of MIS computer model verification: integrity, completeness, consistency. The main stages of verification scenario models are considered according to set criteria.

Вступ

Ефективність прийняття як стратегічних так і тактичних управлінських рішень напряму залежить від якості інформації, що аналізується. Фіксовані правила та підготовлена стратегія поступається більш інформативним та гнучким рішенням за допомогою достовірної інформації, що надається у необхідний час, чи в системі управління взаємодією з клієнтом, чи в системі управління ресурсами підприємства або в інших управлінських інформаційних системах.

Якість інформації може визначатись багатьма критеріями, але для прийняття рішень важливим фактором виступає оперативність забезпечення агрегованою інформацією в контексті предметної області, що є елементом технології дієвої аналітики в програмних системах інформаційного управління (СІУ). Протягом останніх років, починаючи з 2012, аналітична компанія Gartner [1,2] ставить дієву аналітику в десятку найбільш затребуваних інформаційних технологій. Для того щоб аналітика була дієвою, вона повинна мати змістовне наповнення, бути цілеспрямованою, а також надавати засоби моделювання та оптимізації сценаріїв розвитку ситуації або бізнес-процесів. Тому питання формування, реалізації та верифікації сценаріїв розвитку ситуації, розвитку бізнес-процесів та аналізу інформації є актуальною проблемою.

Постановка задачі

В процесі розробки СІУ з дієвою аналітикою (СІУ ДА) необхідно передбачити наступні складові елементи: ціль – сценарій – технологічне середовище – метадані – дані [3]. Сценарії аналізу інформації для здійснення дієвої аналітики мають реалізуватись в реальному часі, що потребує їх верифікації.

Важливим етапом процесу розробки складних програмних систем (ПС) на сучасному етапі є комп'ютерне моделювання, а для визначення рівня достовірності (validated modeling [4]) та коректності ПС проводиться верифікація її комп'ютерної моделі, аналіз поведінки комп'ютерної моделі та формування висновків за результатами моделювання.

Мета – дослідження та розробка аналітичної моделі СІУ ДА для визначення критеріїв верифікації комп'ютерної моделі СІУ.

Аналіз останніх наукових досліджень та праць

Методологія модельної розробки ПС дозволяє оцінити її комплексність через аналіз її комп'ютерної моделі на коректність, повноту та цілісність [5]. Базовим принципом модельної методології є те, що ПС відображаються однією або декількома моделями, кожній з яких відповідає своя метамодель (метаопис).

В зарубіжних працях виділяють дві основні методології розробки з використанням моделей:

1) модельна розробка (model-driven development, MDD);

2) модельне проектування (model-based development).

Обидві методології використовуються до розробки та проектування як фізичних об'єктів, так і до розробки ПС. Основною відмінністю є те, що в модельній розробці перехід від моделей до генерації програмного коду є автоматизованим, в той час як в модельному проектуванні моделі використовуються лише на етапі комп'ютерного моделювання систем. В рамках даної статті використовується та уточнюється методологія модельної розробки до створення СІУ ДА.

Модельною розробкою ПС займались такі зарубіжні вчені як J.L.G. Dietz [6], Thomas Erl [7], Johan Den Haan [8] та ін. Узагальнюючі підходи цих авторів можна визначати такі етапи розробки ПС, які вони пропонують:

– реверсивне проектування – мануальна побудова організаційно-структурної моделі об'єкту автоматизації;

– функціональне проектування – модель сервісів вилучається з організаційної моделі;

– структурне проектування – модель компонент вилучається з моделі сервісів (вручну);

– розробка та управління – модель реалізації компонент є деталізацією моделі компонент;

– реалізація ПС за відповідними інформаційними технологіями.

В основі модельної розробки, що пропонується, лежить структурно-функціональний підхід, для його первинним є архітектура ПС, що визначається організаційно-структурною моделлю об'єкта автоматизації. Зміна організаційно-структурної моделі об'єкту автоматизації викликає необхідність реінжинірингу ПС, що є суттєвим недоліком цього підходу адже для дієвої аналітики первинним є сценарій аналізу даних, а не її організаційна структура (склад її компонентів та інформаційних потоків, що пов'язують компоненти між собою та з її оточенням). На етапі проектування СІУ ДА її організаційна структура створюється "під сценарій".

Найбільш близькою методологією до MDD у вітчизняних працях є модельне конструювання. Вивченням методів модельного конструювання програмних додатків займались такі вчені як Самойлов В.Д [10] та Писаренко А.П. [11].

Самойлов В.Д. визначає модельне конструювання як інформаційну технологію конструювання моделей на базі структур з причинно-наслідковими зв'язками (функціональні моделі). Сценарій в даній технології задає процес взаємодії користувача з системою та передбачає заздалегідь заплановану реакцію на дії користувача [12]. Модель предметного середовища, як складовий елемент модельного конструювання, описує алгебраїчні структури та обчислення та є відображенням основної функціональності комп'ютерної системи.

В якості функціонального елементу виступає оператор присвоєння. Комп'ютерна реалізація додатку на основі принципів модельного конструювання зводиться до побудови його моделі у вигляді послідовно-функціональних елементів. Запропонований підхід до конструювання допомагає неспеціалістам у програмуванні спростити алгоритмічне представлення предметної області.

Недоліки даного підходу полягають у наступному:

1. В описаному підході надається можливість створювати та змінювати функціональні моделі відповідно до графічної специфікації, але нема програмного середовища для імітації та верифікації сценарних моделей, що є важливою умовою вирішення поставленої задачі.
2. Не враховується інформаційна складова дієвої аналітики – знання та мета опис, в той час як функціональна складова представлена лише оператором присвоєння, що є недостатнім для розробки СІУ з дієвою аналітикою.

Виклад основного матеріалу дослідження

Двома основними вимогами розробки ПС, що базується на моделях є [9]:

1. Кожна модель повинна бути формальною (визначеною на добре відомій мові), на прикладі запропонованої послідовності моделей:
   – організаційна модель – BPMN;
   – модель сервісів – таблиця з властивостями (специфікація сервісів);
   – модель компонент – SCA (Компонентна Архітектура Сервісів);
   – модель реалізації компонент – набір з'єднаних DCL (предметно-орієнтована мова);
2. Кожна модель повинна взаємодіяти інформаційно з попередніми моделями.

Визначимо ці моделі для СІУ ДА. Узагальнена аналітична модель СІУ може бути визначена у наступному вигляді [13]:

$$MI^{MIS} = \bigcup_k \left\{ \begin{matrix} F_k^{MIS}, S_t^{MIS}, I_t^{nU}, Rg_k, V_k^{In}, V_k^{Out} \mid \\ k = \overline{1,N} \end{matrix} \right\},$$

де F_k^{MIS} - набір функцій *к*-ої задачі інформаційного управління;

St_k^{MIS} - структурні елементи СІУ;

I_k^{nU} - набір початкових налаштувань *к*-ої задачі інформаційного управління;

Rg_k - регламент виконання *к*-ої задачі інформаційного управління;

V_k^{In} - внутрішній інформаційний вплив на хід виконання *к*-ої задачі;

V_k^{Out} - зовнішній інформаційний вплив на хід виконання *к*-ої задачі

Але для СІУ ДА дана Узагальнена модель СІУ не враховує наявність або необхідність побудови сценаріїв розвитку бізнес-процесів аналізу інформації, що не дозволяє ефективно побудувати сценарії аналізу інформації, базуючись на технологічних (інформаційних) потужностях організації. Крім того, така аналітична модель не відповідає методології модельної розробки ПС та більш прийнятна за використанням структурно-функціонального підходу до проектування СІУ [14].

Тому з урахуванням параметрів узагальненої аналітичної моделі СІУ в якості концептуальної моделі СІУ ДА запропоновано аналітичну модель у вигляді послідовності наступних моделей СІУ ДА:

$$M^{MIS} : \left\{ \begin{matrix} Kn^M, D_i^M, \mathrm{Pr}_i^M, Sc_i^M, C_i^M, Dv^M \mid \\ i = \overline{1,N} \end{matrix} \right\},$$

де Kn^M - модель знань, що визначає дві категорії знань: знання про технології СІУ: сервіси, інформаційні технології, модулі, програмні засоби, БД, БЗ, сховища даних, Big Data, знання про організаційну структуру: виконавці сценаріїв, бізнес-правила);

D_i^M - об'єктна модель i-ої предметної області, що відповідає за опис понять та метаданих для доступу до даних;

Pr_i^M - модель прецедентів використання, що є декларативним описом функціональних вимог до системи та логіки взаємодії користувача з елементами графічного інтерфейсу;

Sc_i^M - сценарна модель i-ої задачі аналізу інформації, що формалізує процес аналізу інформації та виражається в логічно-впорядкованій послідовності подій в процесі аналізу даних з метою отримання необхідної інформації для прийняття обґрунтованих рішень;

C_i^M – компонентна модель СІУ, як сукупність компонентів, необхідних для забезпечення функціональних та інших вимог, яка побудована і функціонує відповідно до певних правил створення компонентних конфігурацій і компонентного взаємодії [15].

Dv^M - модель реалізації компонент, що також включає опис інформаційної взаємодії об'єктів та елементів системи.

На рисунку 1 показано послідовність моделей СІУ ДА. На першому етапі будується модель знань Kn^M, що включає в себе знання про технологічну базу в організації: програмні модулі, комп'ютерні системи та інформаційні технології, що є в наявності, знання про виконавців (кадрів) та їх кваліфікацію та знання про методи обробки та аналізу даних. Такі знання надалі використовуються для побудови сценаріїв аналізу інформації, визначення елементів сценарію для яких необхідно розроблювати додаткові програмні засоби, сервіси та для яких необхідно залучити експерта з певної проблеми. На другому етапі будується об'єктна модель предметної області, що представляє собою набір понять спеціалізованої області дослідження [16]:

$$D^M = \bigcup_i \left\{ En_i, R_{ij} \mid i = \overline{1,N}; j = \overline{1,N}; i \neq j \right\},$$

де En_i - деяке поняття предметної області;

R_{ij} - зв'язок між i-им та j-им поняттями предметної області.

Згідно концепції побудови адаптивних об'єктних моделей [17] атрибути об'єктів та типи атрибутів є окремими від об'єкта елементами:

$$En = \bigcup_k \left\{ At_k^{En}, Atp^{At}, Vl^{At} \mid k = \overline{1,N} \right\},$$

де At_k^{En} - атрибут поняття, що відноситься до визначеної предметної області;

Atp^{At} - тип відповідного атрибуту;

Vl^{At} - значення відповідного атрибуту (через засоби доступу до даних).

Сценарна модель відображає порядок та зміст проведення інформаційного управління на функціональному рівні.

$$Sc^M = \bigcup_i \left\{ TSc_i, ASc_i, Ex_i, G_i \mid i = \overline{1,N} \right\},$$

де TSc_i - типовий сценарій виконання i-го завдання аналізу інформації;

ASc_i - розширений (вторинний) сценарій виконання i-го завдання аналізу інформації;

G_i - ціль сценарію виконання i-го завдання аналізу інформації, що має такі характеристики як: критерії, час та ресурси.

Сценарій аналізу інформації має загальний вигляд:

$$Sc = \bigcup_i \left\{ \begin{matrix} El_i, Ex_i, F^{El}\left(null \mid Ob_1, ...Ob_n\right) \mid \\ i = \overline{1,N}; j = \overline{1,N}; i \neq j \end{matrix} \right\}$$

де El_i - елемент сценарію аналізу інформації, що може бути дією, подією або зв'язком;

Ex_i - виконавець елементу сценарію, який може бути бізнес аналітиком, аналітиком даних, експертом, працівником організації та процесом;

$F^{El}\left(null \mid Ob_1, ...Ob_n\right)$ - функція к-ої задачі аналізу інформації, що може приймати як вхідні значення поняття предметної області або пусте значення.

Модель прецедентів відповідає за взаємодію користувача з графічним інтерфейсом при виконанні задач аналізу інформації. Модель прецедентів застосовується для виявлення вимог до поведінки системи, відомих також як функціональні вимоги:

$$\Pr^M = \bigcup \left\{ \begin{matrix} Ac_i, P_i^{Ac}, Rl_{ij}^{Ac}, I_k^{nU}, F_i^p \mid \\ i = \overline{1,N}, j = \overline{1,N} \end{matrix} \right\},$$

де Ac_i - актор, що відноситься до учасників сценарію певної предметної області: $Ac_i \leq Ex_j$;

P_i^{Ac} - прецедент використання, що відповідає за послідовність дій користувачів (акторів) СІУ та реалізується певним сценарієм виконання i-го завдання контрольно-управлінського аналізу інформації: $P_i^{Ac} \geq Sc_j$;

F_i^p - екранні форми користувачів для взаємодії з СІУ;

Rl_{ij}^{Ac} - відношення між прецедентами використання:

$$Rl_{ij}^{Ac} = \left\{ include, extend, generalize, normal \right\}$$

де *normal* – комунікаційний зв'язок між прецедентом та актором, який в ньому бере участь;

extend – включення додаткової поведінки в базовий прецедент, який про це не знає;

include – включення додаткової поведінки в базовий прецедент, що детально описує дане включення;

generalization – взаємозв'язок між загальним прецедентом та більш специфічними прецедентами, що наслідують та додають властивості до загального прецеденту;

Компонентна модель застосовується при визначені архітектури реалізації сценаріїв у розподіленому середовищі й побудові комп'ютерної моделі системи реалізації сценаріїв. Та визначається як сукупність вимог до компонентів УІС та правил взаємодії і побудови компонентних конфігурацій.

Компонентна модель застосовується при визначені архітектури реалізації сценаріїв у розподіленому середовищі й побудові комп'ютерної моделі системи реалізації сценаріїв.

Модель реалізації компонент Dv^M включає опис інформаційної взаємодії об'єктів та елементів системи та відображається у фабрику програмного забезпечення (software factory [18]):

$$Dv^M = \bigcup_i \left\{ St_k^{MIS}, I_k^{nU}, V_i^{In}, V_i^{Out} \mid i = \overline{1,N} \right\},$$

де St_k^{MIS} - структурні елементи СІУ;

I_k^{nU} - набір початкових налаштувань *к*-ої задачі аналізу інформації;

V_k^{In} - сценарії взаємодії компонентів СІУ;

V_k^{Out} - сценарії взаємодії СІУ з зовнішніми середовищами.

Відповідно до другого принципу модельної розробки необхідно передбачити та визначити інформаційну взаємодію між моделями.

Верифікація моделей відбувається в фабриці програмного забезпечення, що реалізує моделі в СІУ.

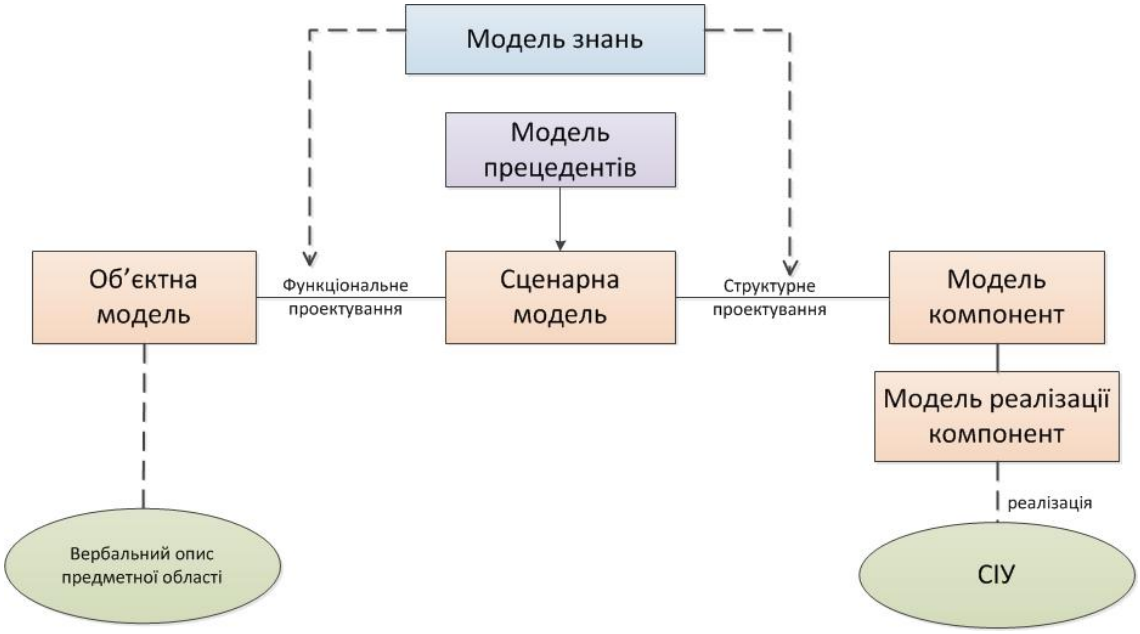


Рисунок 3. СІУ з дієвою аналітикою

Пропонується три критерії перевірки комп'ютерної моделі СІУ:

1. Цілісність – відносна незалежність системи від зовнішніх систем. Цілісність висловлює інтегрованість, самодостатність та автономність об'єктів системи, їх протипоставлення оточенню, що пов'язане з їх внутрішньої активністю. Цілісність характеризує якісну своєрідність об'єктів системи, що обумовлено притаманним їм специфічними закономірностями функціонування та розвитку.

2. Повнота – характеристика, що визначає кількість інформації, необхідної і достатньої для прийняття правильних рішень.

3. Несуперечність визначається як в класичній логіці та передбачає неможливість виведення з однієї сценарної моделі двох результатів, що виключають один одного.

Відповідно до критеріїв верифікації запропоновано наступні етапи перевірки сценарних моделей:

1. Перевірка синтаксису сценарію, а саме перевірка:

– повноти інформації по кожному елементу сценарію;

– наявність біль ніж одного елементу сценарію;

– взаємозв'язків між елементами;

– участі актору хоча б в одному елементі сценарію;

– того, що кожен елемент сценарію використовує ресурс (в якості ресурсу може виступати інформаційний ресурс або кількісне значення);

– наявності необхідних ресурсів для виконання елементу сценарію;

2. Перевірка семантики сценарію, а саме перевірка:

– відповідності сценарію загальній цілі та елементу сценарію підцілі;

– того, що виконані попередні умови для виконання елементу сценарію;

– того, що кожному елементу сценарію відповідає певна функція (модуль або сервіс);

3. Перевірка посилань на коректність.

Висновки

Розробка СІУ ДА вимагає її верифікації на етапі проектування. Описана аналітична модель висвітлює основні елементи СІУ ДА, на базі яких обрано три критерії верифікації: цілісність, повнота та несуперечність.

Наступним етапом необхідно розробити систему верифікацію комп'ютерної моделі СІУ ДА, враховуючи поставлені критерії верифікації.

Список литературы

1. Magic Quadrant for Business Intelligence and Analytics Platforms. — 2013, © Gartner Analytic
2. Barton D. et al. Making advanced analytics work for you //Harv. Bus. Rev. – 2012. – Т. 90. – С. 78-83.
3. Модель сценарно-целевого подхода при построении информационно-аналитической системы / Коваль А.В., Бойко Ю.Д., Зайцева Е.А.// Системный анализ и информационные технологии: материалы 16-й Международной научно-технической конференции SAIT 2014, Киев, 26-30 мая 2014 г. / УНК "ИПСА" НТУУ "КПИ". – К.: УНК "ИПСА" НТУУ "КПИ", 2014. – С. 105-107.
4. Thacker B. H. et al. Concepts of model verification and validation. – Los Alamos National Lab., Los Alamos, NM (US), 2004. – №. LA-14167.
5. France R., Rumpe B. Model-driven development of complex software: A research roadmap //2007 Future of Software Engineering. – IEEE Computer Society, 2007. – С. 37-54.
6. J.L.G. Dietz. Architecture – Building strategy into design. Academic Service, The Hague, The Netherlands, 2008.
7. Thomas Erl. Principles of Service Design. The Prentice Hall Service-Oriented Computing Series from Thomas Erl. Prentice Hall, Upper Saddle River, NJ, USA, 2007.
8. Stein S. et al. Evaluation of OrViA framework for model-driven SOA implementations: An industrial case study //Business Process Management. – Springer Berlin Heidelberg, 2008. – С. 310-325.
9. http://www.theenterprisearchitect.eu/blog/2009/06/03/a-framework-for-model-driven-soa/
10. Самойлов В.Д. Модельное конструирование компьютерных приложений. Киев. Наукова думка, 2007. 198 стр.
11. Писаренко, А. П. Компьютерные технологии моделирования для динамических тренажеров / А.П. Писаренко, В.Д. Самойлов, О.Я. Стеценко. - К. : Наукова думка, 1992. - 168 с.
12. Сценарные структуры ситуационных тренажеров / В.Д. Самойлов, О.П. Нетлюх // Збірник наукових праць Інституту проблем моделювання в енергетиці ім.Г.Є.Пухова НАН України. — К.: ІПМЕ ім. Г.Є.Пухова НАН України, 2009. — Вип. 52.
13. Додонов О. Г., Коваль О. В., Сенченко В. Р., Методологія побудови корпоративних інформаційно-аналітичних систем // Реєстрація, зберігання і обробка даних, Київ, - 2007. Том 9, № 4. 60–75 сс.
14. Юдицкий С. А. Сценарно-целевой подход к системному анализу. Автомат. и телемех., 2001, № 4, 163–175.
15. Формальные модели компонентного программирования /В.Н. Грищенко// Проблеми програмування. — 2003. — N 2. — С. 42—57.
16. Формування сценаріїв аналітичної діяльності / Коваль О.В., Зайцева К.А., Бойко Ю.Д. // Системні дослідження та інформаційні технології. — 2014. — № 1. — С. 20–25.
17. Yoder J.W. Architecture and Design of Adaptive Object-Models / J.W. Yoder // SIGPLAN Notices 12. – 2001. – P. 50-60.
18. Greenfield J., Short K. Software factories: assembling applications with patterns, models, frameworks and tools //Companion of the 18th annual ACM SIGPLAN conference on Object-oriented programming, systems, languages, and applications. – ACM, 2003. – С. 16-27.

УДК 616-072

БОБРОВА Н.Л.

ОБОСНОВАНИЕ ИСПОЛЬЗОВАНИЯ КОМПЛЕКСА ДИАГНОСТИЧЕСКИХ МЕТОДИК ДЛЯ ОЦЕНКИ ПСИХОФИЗИОЛОГИЧЕСКОГО СОСТОЯНИЯ ЧЕЛОВЕКА

В настоящей статье приведено математическое обоснование выбора нескольких методов для психофизиологической диагностики организма человека. Проведен анализ и отбор диагностических методик для программно-аппаратного комплекса исследования психофизиологического состояния человека. Дана краткая характеристика выбранных методик и описаны количественные критерии, позволяющие характеризовать с различных сторон текущее функциональное состояние центральной нервной системы человека: функциональный уровень системы, устойчивость реакции и уровень функциональных возможностей.

The subject of the article is proposed formal approach to the assessment of psychophysiological state of a person using probabilistic methods, in particular vector analysis, which allows to reduce the error level of the developed diagnostic system by increasing the number of diagnostic methods used in the hardware-software complex as the main criteria influencing the choice of test methodology, proposed the following: noninvasiveness, portability and the ability to work offline, time and place of the examination, the ability to survey a trainer, instructor or sports physician, and the presentation of the results in an understandable form for professionals with different qualifications.

The complex can be used for a wide range of scientific and practical problems in the field of General and vocational education, military service, physical culture and sports, social services.

Введение

Поведение человека в сложных экстремальных ситуациях определяется его психологическим состоянием и готовностью к принятию решения и адекватным действиям. Для снижения возможности проявления ошибочных действий человека необходимо организовывать обучение, тренировки, развивающие быстроту мышления, подсказывающие, как использовать прежний опыт для успешного принятия решения, для перевода действий работника на уровень стереотипов, а также формирующие способность к прогнозированию. Кроме этого, нужно проводить профессиональный отбор, а также соответствие психофизиологических возможностей человека условиям труда. Профессиональный психологический отбор работников любой сферы деятельности ставит задачу выявить людей, у которых процесс обучения дает максимальный эффект при минимальном времени обучения. Профессиональная пригодность определяется положительной мотивацией к данной специальности; высоким порогом ощущения опасности; быстротой реакции на экстремальные ситуации; хорошим глазомером; устойчивостью, концентрацией и распределением внимания; нормальным состоянием двигательного аппарата; высокой пропускной способностью анализаторов и т. д.

Для этой цели разработан компьютерный комплекс на основе формализованного подхода к оценке психофизиологического состояния человека с использованием вероятностных методов, предназначенный для комплексной оценки психофизиологических и психологических свойств и функций организма здоровых, а также имеющих заболевания людей по результатам выполнения тестовых заданий.

Обоснование использования комплекса диагностических методик для оценки психофизиологического состояния человека

Дадим формализацию понятия психофизиологического состояния организма человека. Рассмотрим различные психофизиологические состояния организма, которые имеют свои нозологические единицы («образы»). Каждое из состояний может быть описано многими способами, принадлежащими к разным областям человеческих знаний[1]. Конечное множество состояний обозначим $A = \{A_1, A_2, A_3,..., A_n\}$.

Предположим, что для определения некоторого психофизиологического состояния $A_i(i=1,2,3,...,n)$ используются m методов диагностики. Пусть $D = \{D_1,D_2,D_3,...,D_m\}$ - объединение этих методов в множество. Существует k диапазонов величин, используемых для измерений каждого из состояния A_i (i = 1,2,3,...,n). Все имеющиеся диапазоны объединим в множество $B = \{B_1,B_2,B_3,...,B_k\}$

Следовательно, каждое из психофизиологических состояний организма A_i (i = 1,2,3,...,n) можно охарактеризовать с помощью различных методов диагностики D_j (j= 1,2,3,...,m), каждый из которых использует определенные диапазоны параметров B_l (l= 1,2,3,...,k) .

Так как, множество методов диагностики $D = \{D_1, D_2, D_3, ..., Dm\}$ образует множество несовместных событий, то исходя из теории вероятностей, можно представить в следующем виде:

$$\sum_{i=1}^{k} p(D_i) = 1, D_i \cdot D_j = 0, i \neq j; i, j = 1,2,3,...m \qquad (0.1)$$

Психофизиологическое состояние A_i, получаемое с помощью конкретного метода диагностики D_j с использованием определенных диапазонов параметров B_l (l= 1,2,3,...,k) обозначим:

$$A_i = A_i \bullet D_j \bullet B_1 + A_i \bullet D_j \bullet B_2 + ... + A_i \qquad (0.2)$$

$$\bullet D_j \bullet B_k = \sum_{l=1}^{k} A_i \bullet D_j \bullet B_l$$

Пусть a_j – количественная составляющая оценки психофизиологического состояния A_i полученная при диагностике методом D_j , а d_j =1, если используется при диагностике метод D_j и d_j =0, если при диагностике не используется метод D_j. Тогда психофизиологическое состояние, получаемое с помощью всех используемых методов диагностики D_j (j = 1,2,3,..., m) с использованием определенных параметров B_l (l = 1,2,3,...,k) обозначим:

$$a_i^* = \frac{\sum_{j=1}^{m}\sum_{l=1}^{k} a_i d_j B_l}{\sum_{l=1}^{k} B_l} \qquad (0.3)$$

Для определенного метода диагностики D_j множество параметров B_l (l= 1,2,3,...,k) также является полной системой несовместимых событий:

$$\sum_{i=1}^{k} p(B_i) = 1, B_i \cdot B_j = 0, i \neq j; i, j = 1,2,3,..k \qquad (0.4)$$

Определим вероятность диагностики психофизиологического состояния A_i исходя из того, что выбор методов диагностики равновероятен и выбор параметров при каждом из них также равновероятен. Получаем условную вероятность, т.е. вероятность обнаружения образа состояния при конкретной диагностике используя равновероятность выбора метода диагностики, теоремы сложения и умножения вероятностей и формулу полной вероятности:

$$\begin{aligned} &\rho(A_i) = \rho(A_i \cdot D_j \cdot B_1 + A_i \cdot D_j \cdot B_2 + ... + A_i \cdot D_j \cdot B_k) = \\ &\rho(A_i \cdot D_j) \cdot \rho(\sum_{l=1}^{k} B_l) = \rho(A_i \cdot D_j) = \rho(D_j) \cdot \rho(A_i / D_j) = \\ &\rho(A_i / D_j) \end{aligned} \qquad (0.5)$$

Вероятность диагностики психофизиологического состояния человека с помощью используемых методик и диапазонов параметров (среднее арифметическое условных вероятностей) будет:

$$\begin{aligned} &\rho(A_i) = \rho(A_i \cdot D_1 \cdot B_1 + A_i \cdot D_1 \cdot B_2 + ... \\ &+ A_i \cdot D_1 \cdot B_k + \\ &A_i \cdot D_2 \cdot B_1 + A_i \cdot D_2 \cdot B_2 + ... + \\ &A_i \cdot D_2 \cdot B_k + ... + A_i \cdot D_m \cdot B_1 + \\ &A_i \cdot D_m \cdot B_2 + ... + A_1 \cdot D_m \cdot B_k) = \\ &= \rho(A_i \cdot D_1 + A_i \cdot D_2 + ... + A_i \cdot D_m) = \\ &\sum_{j=1}^{m} \rho(D_j) \cdot \rho(A_i / D_j) = \frac{\rho(A_i / D_j)}{m} \end{aligned} \qquad (0.6)$$

Описание психофизиологического состояния человека может не совпадать с состоянием, получаемым с помощью одного метода диагностики, что приводит к ошибкам в диагностике состояния. Используя несколько методов диагностики психофизиологического состояния человека, можно избежать этих ошибок.

Процесс принятия решения (диагностики психофизиологического состояния человека состояния) может быть представлен логической схемой (рисунок 1), на которой определено время выполнения измерений и определение состояния.

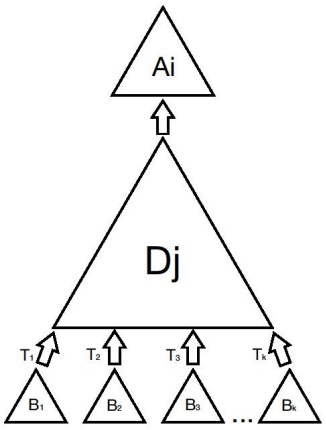


Рис. 1. Общая схема проведения диагностики психофизиологического состояния организма человека

Следовательно, для точной оценки ПФС человека необходимо использовать несколько методов диагностики, и совпадение результатов работы нескольких диагностических средств, функционирующих в одном программно-аппаратном комплексе, приводит к увели-

чению достоверности получаемых результатов по оценке состояния.

Выбор методик, которые были использованы для обследования работников сферы телекоммуникации, обусловлен диагностическими возможностями методов, их безопасностью и комфортностью операторов, а также малым временем, требуемым на проведение измерений. Для оценки функционального состояния центральной нервной системы были подобраны следующие методики:

Методика "Простая зрительно-моторная реакция". Простая зрительно-моторная реакция (ПЗМР) – это элементарный вид произвольной реакции человека на зрительный стимул. Простая зрительно-моторная реакция состоит из двух последовательных компонентов: сенсорного (латентного) периода и моторного периода.

Простая зрительно-моторная реакция лежит в основе других целенаправленных приспособительных реакций человека, поэтому на основании показателя скорости ПЗМР можно сделать вывод о временных параметрах более сложных составляющих поведения человека. Кроме того, скорость простой зрительно-моторной реакции позволяет оценить интегральные характеристики центральной нервной системы человека, т.к. при ее реализации задействованы как основные анализаторные системы человека (зрительная и кинестетическая), так и определенные отделы головного мозга и нисходящие нервные пути[2].

Оценка результатов по методике "Простая зрительно-моторная реакция" при наличии нормального распределения производится на основании среднего значения времени реакции и стандартного отклонения. Среднее значение отражает среднюю скорость ПЗМР, характерную для данного индивида: чем меньше среднее значение времени реакции, тем выше скорость реагирования. Стандартное отклонение является показателем стабильности сенсомоторного реагирования: чем меньше стандартное отклонение, тем более стабильной является скорость сенсомоторной реакции.

Для получения наиболее полной информации о свойствах и состоянии центральной нервной системы на основании результатов по данной методике можно использовать дополнительные показатели, в частности критерии Т.Д. Лоскутовой и коэффициент точности Уиппла. На основании соответствия нестандартных статистических показателей состоянию обследуемого А.М. Зимкина и Т.Д. Лоскутова определили три количественных критерия, позволяющие характеризовать с различных сторон текущее функциональное состояние центральной нервной системы: функциональный уровень системы, устойчивость реакции и уровень функциональных возможностей.

Для получения наиболее полной информации о свойствах и состоянии центральной нервной системы на основании результатов по данной методике используются три количественных критерия, позволяющие характеризовать с различных сторон текущее функциональное состояние центральной нервной системы: функциональный уровень системы, устойчивость реакции и уровень функциональных возможностей[3].

Первый критерий - функциональный уровень системы (ФУС). Его величина определяется, главным образом, абсолютным значением времени реакции и отражает текущее функциональное состояние ЦНС, степень развития утомления под влиянием факторов окружающей среды.

Второй критерий - устойчивость реакции (УР). Величина этого показателя тем больше, чем меньше вариабельность значений времени простой двигательной реакции. Поскольку разнообразие значений времени реакций является проявлениями непрерывных флуктуаций состояния ЦНС, показатель УР рассматривается как критерий устойчивости состояния ЦНС. Следовательно, чем выше показатель УР, тем устойчивее, стабильнее текущее функциональное состояние ЦНС.

Третий критерий - уровень функциональных возможностей (УФВ). Он наиболее полно характеризует состояние ЦНС и позволяет судить о ее способности сформировать и достаточно долго удерживать соответствующее функциональное состояние.

Оценка функционального уровня системы вычисляется по формулам:

$$ФУС = \ln 1 / T_{МОД} \cdot \Delta T_{0,5} \left[c^2\right] \qquad (0.7)$$

Оценка устойчивости реакции:

$$УР = \ln P_{МАКС} / \Delta T_{0,5} \left[c^{-1}\right] \qquad (0.8)$$

Оценка уровня функциональных возможностей:

$$УФВ = \ln P_{МАКС} / \Delta T_{0,5} \cdot T_{0,5} \left[c^{-2}\right] \quad (0.9)$$

где ТМОД – время реакции, соответствующее пику гистограммы(наиболее вероятному значению);

Рмакс – вероятность(частота) появления значения, равного моде(амплитуда моды);

ΔТ0,5 – диапазон времен реакций на уровне 0,5Рмакс .

На основании имеющегося экспериментального материала для аппаратной реализации метода получены [4] следующие граничные значения критериев функционального состояния ЦНС для здоровых испытуемых (таблица 1):

Функциональный уровень системы = 4,2-5,5.

Устойчивость реакций = 1,0-2,8.

Уровень функциональных возможностей = 2,7-4,8.

Табл. 1. - Граничные значения критериев функционального состояния центральной нервной системы

Состояние системы	ФУС	УР	УФВ
Высокий уровень	4,9-5,5	2,1-2,8	3,9-4,8
Средний уровень	4,5-4,8	1,5-2,0	3,1-3,8
Низкий уровень	4,2-4,4	1,0-1,4	2,7-3,0

Это исследование очень важно для интегральной оценки здоровья человека, особенно спортсменов и лиц экстремальных видов деятельности, подвергающихся воздействию чрезмерных и физических, и психоэмоциональных нагрузок. Для включения результатов теста зрительно-моторной реакции в итоговый протокол комплексного обследования вычисленные значения критериев пересчитываются в %.

Коэффициент точности Уиппла выявляет соотношение ошибок и правильных реакций и вычисляется по формуле:

$$KT = \frac{N - R}{N + P} \quad (0.10)$$

где N – число измерений (предъявленных сигналов);

R – количество правильных нажатий;

P – количество ошибок.

Чем меньше данный показатель, тем выше степень точности выполнения заданий. Показатель точности свидетельствует о степени свойства устойчивости внимания, обусловленного в свою очередь силой и уравновешенностью нервных процессов.

Методика "Реакция на движущийся объект". Реакция на движущийся объект (РДО) представляет собой разновидность сложной сенсомоторной реакции, т.е. такой реакции, которая помимо сенсорного и моторного периодов включает период относительно сложной обработки сенсорного сигнала центральной нервной системой.

Методика "Реакция различения". Реакция различения является разновидностью сложной сенсомоторной реакции. В отличие от простой реакции, реакция различения осуществляется на один определенный стимул из нескольких разнообразных стимулов. Поэтому процесс обработки сенсорной информации центральной нервной системой происходит не только по принципу наличия либо отсутствия сигнала, но и по принципу различения сигналов, отбора сигналов определенного цвета из общего их числа и формирования реакции на заданный вид сигнала. В связи с более сложным процессом обработки сенсорной информации центральной нервной системой скорость реакции различения меньше, чем скорость простой реакции, т.е. время, затраченное на осуществление реакции различения, больше, чем на осуществление простой реакции [5].

Заключение

Предложен формализованный подход к оценке психофизиологического состояния человека с использованием вероятностных методов, в частности векторного анализа, что позволяет снизить уровень ошибок разрабатываемой системы диагностики за счет увеличения количества методов диагностики, использующихся в программно-аппаратном комплексе. В качестве основных критериев, влияющих на выбор методики тестирования, предложены следующие: неинвазивность, портативность и возможность автономной работы, время и место обследования, возможность проведения обследования тренером, инструктором или врачом, а также представление результатов в доступном виде для специалиста различной квалификации.

Комплекс может использоваться для решения широкого спектра научных и практических задач в области общего и профессионального образования, военной службы, физкультуры и спорта, социального обслуживания.

Список литературы

1. Тьюки Дж. Анализ результатов наблюдений [пер. с англ] [Текст] / Дж.Тьюки. - М: Мир, 1981.-694 с.
2. Пшеничный Б.Н., Данилин Ю.М. Численные методы в экстремальных задачах. М.: Наука, 1975. - 319 с.
3. Короленко Ц.П. Психофизиология человека в экстремальных условиях / Ц.П. Короленко. – Л.: Медицина, 1978. – 272 с.
4. Одерышев Б.С. Методики измерения функционального состояния человека / Б.С. Одерышев // Экспериментальная и прикладная психология. Психические состояния / под ред. А.А.Крылова. – Л., 1981. – С. 30–38.
5. Tuckow A.P. Nocturnal growth hormone secretory dynamics are altered after resistance exercise: deconvolution analysis of 12-hour immunofunctional and immunoreactive isoforms / A.P. Tuckow // American Journal of Physiology. Regulatory, Integrative and Comparative Physiology. – 2006. – Vol. 291, N 6. – P. 1749–1755.

УДК 004.773.5

РОКОВОЙ А.П.,
ГАБИНЕТ А.В.,
ПЕТРУК В.А.

МЕТОД ЛОКАЛИЗАЦИИ МУЛЬТИМЕДИЙНОГО ТРАФИКА В ОДНОРАНГОВЫХ СЕТЯХ

Предложен метод локализации мультимедийного трафика в одноранговых сетях на основе прогнозирования количества внешних пиров. Метод позволяет повысить качество передачи мультимедийных данных для конечных пользователей за счет преждевременной перестройки логической топологии при изменении количества пиров. Доказана возможность прогнозирования временного ряда количества внешних пиров. Выбран метод прогнозирования, обеспечивающий требуемую точность прогноза.

The method of localization of multimedia traffic in peer-to-peer networks based on forecasting of outside peers quantity is offered in the article. The method allows to increase the quality of data transmission to end-users at the expense of premature changing of the logical topology while changing the quantity of peers. The possibility of forecasting the time series of the outside peers quantity is proved. The forecasting method that provides the required forecast accuracy is chosen.

Введение

Современный этап развития сети Интернет характеризуется увеличением объемов мультимедийного трафика, основная часть которого приходится на приложения, генерирующие потоковое видео. Это связано с тем, что сеть Интернет постепенно становится основным транспортным средством для распространения различного рода видеоданных, в том числе предоставления услуг телевидения.

Задача предоставления услуг телевидения в сети Интернет состоит в копировании одного и того же потока видео всем узлам-подписчикам в рамках требуемых временных ограничений. Для эффективного решения этой задачи можно использовать механизмы, встроенные в протокол IP [1]: групповая адресация и обеспечение качества обслуживания. Механизм групповой адресации выполняет копирование потока видео на уровне маршрутизаторов, тем самым снимая нагрузку с узла-источника видеоданных и уменьшая общий объем трафика, генерируемого приложением. Механизм обеспечения качества обслуживания должен гарантировать: полосу пропускания канала передачи данных не менее битовой скорости потока видеоряда, в среднем 500-1000 Кбит/с, задержку передачи данных от источника к приемнику не более 10 с, потери данных не более 1 %. [2]

Как групповая адресация, так и обеспечение качества обслуживания требуют поддержки со стороны маршрутизаторов. При этом все маршрутизаторы на пути следования должны реализовывать эти механизмы. Что достаточно сложно обеспечить для произвольной пары узлов сети Интернет. Поэтому указанные механизмы не могут быть использованы для предоставления услуг телевидения в сети Интернет. Единственным способом решения этой задачи до недавнего времени была классическая клиент-серверная архитектура. Где узел-источник видеоданных передавал поток видео для каждого клиента в отдельности. То есть, процедура копирования потока выполнялась на узле-источнике видеоданных.

Однако такой подход не может обеспечить передачу потокового видео для большого количества пользователей из-за ограниченной пропускной способности каналов передачи данных в определенной точке сети, а также высокой нагрузки на узел-источник. Решением проблемы может быть использование подхода наложенной поверх существующих каналов сети передачи данных. Где идея групповой доставки реализуется не на уровне маршрутизаторов, а на уровне приложений, работающих на конечных узлах сети.

Одноранговая сеть (P2P), где каждый узел одновременно выступает в роли клиента и в роли сервера, является эффективным решением для распространения потокового видео в сети Интернет. На сегодня существует огромное

количество коммерческих приложений, таких как SopCast[3], PPLive [4], PPStream [5], UUSee [6], которые успешно справляются с задачей доставки видеоданных для тысяч узлов (пиров).

В P2P сети пиры устанавливают соединения между собой, благодаря чему образуют наложенную сеть. В зависимости от географического местоположения пиров, наложенная сеть может охватывать большое количество интернет-провайдеров (ISP). Взаимодействие между пирами формирует поток мультимедийных данных через наложенную сеть и, следовательно, между провайдерами. Таким образом, наложенная сеть не учитывает топологию базовой сети и приводит к нежелательному трафику между ISP [7].

Естественный способ уменьшить количество мультимедийного трафика между ISP, который генерируется P2P сетью, это предпочтительная установка соединений внутри одного и того же ISP [8,9] и использование технологии выбора суперпиров [10]. При увеличении количества пиров возрастает нагрузка на суперпиры и возникает необходимость увеличения их количества. Это ведет за собой изменение логической топологии внутри сети ISP, появляются больше допустимых задержки и потери, что в результате ухудшает качество передачи видеоданных к конечным пирам. Поэтому возникает необходимость прогнозирования количества внешних пиров для преждевременного выбора новых суперпиров и изменения логической топологии.

Таким образом, целью работы является разработка эффективного метода локализации мультимедийного трафика на основе прогнозирования количества внешних пиров.

Проведение эксперимента

Среди множества приложений для передачи потокового видео в одноранговых сетях (P2PTV) самым популярным является SopCast[11]. SopCast – это бесплатное приложение для P2PTV, которое было разработано в университете Fundan в Китае. Скорость передачи видеоданных для SopCast, как правило, находится в диапазоне от 250 до 400 Кбит/с, в некоторых случаях может достигать 800 Кбит/с.

Приложение SopCast использует закрытый протокол, поэтому прямых способов выделения P2P трафика из общего объема не существует. В работе [12] были проведены исследования, на основании которых получены оценочные характеристики трафика, генерируемого приложением SopCast. Результаты исследований представлены в таблице 1. На основании этих данных можно построить фильтр для утилиты, осуществляющей захват сетевого трафика.

Таблица 1

Тип	Размер (байты)	Функциональность
Видео пакет	1320	Максимальный размер видео пакетов
	377, 497, 617, 1081, 1201	Видео фрагменты
Управляющий пакет	52	HELLO пакет для установления соединения
	80	Подтверждения на получение HELLO пакета
	28	Подтверждение
	42	Поддержка состояния между пирами
	46	Запрос видео пакета

Эксперимент проводился в локальной сети Национального технического университета Украины “КПИ” во время футбольного матча отборочного цикла Лиги Чемпионов. Упрощенная топология сети показана на рисунке 1. Клиенты, на которых запущено приложение SopCast, подключены к коммутатору SWR18h. Прямое соединение коммутатора с маршрутизатором R1 позволяет направлять IP трафик в сеть Интернет. Клиентов внутри автономной системы AS25500 будем называть локальными пирами, а остальных, которые установили соседство с ними и находятся снаружи – внешними пирами. Сбор P2P трафика осуществлялся на маршрутизаторе R1.

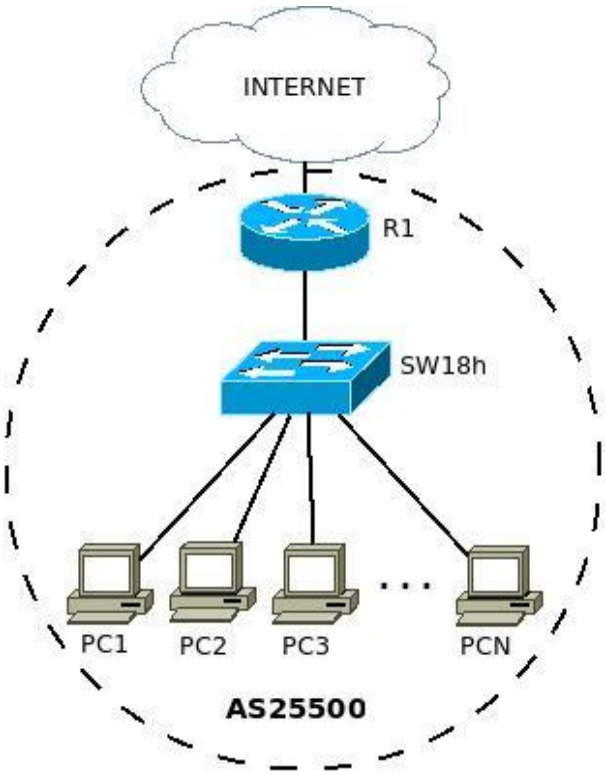


Рис.1. Топология для проведения эксперимента

В период времени с 21:15 по 21:55 для сбора P2PTV трафика на маршрутизаторе R1 была запущена утилита tcpdump [13] с фильтрами, построенными на основе данных из таблицы 1.

Обработка результатов

Для анализа захваченного трафика была разработано приложение p2p-analyzer, которое формирует временной ряд количества внешних пиров с заданным интервалом. В работе [12] было показано, что интервал обмена сообщениями в приложении SopCast между двумя пирами с вероятностью 0,95 не превышает порог в 2 с. Поэтому интервал для временного ряда был задан в 2 с. Временной ряд количества внешних пиров, полученный в результате анализа, представлен на рисунке 2.

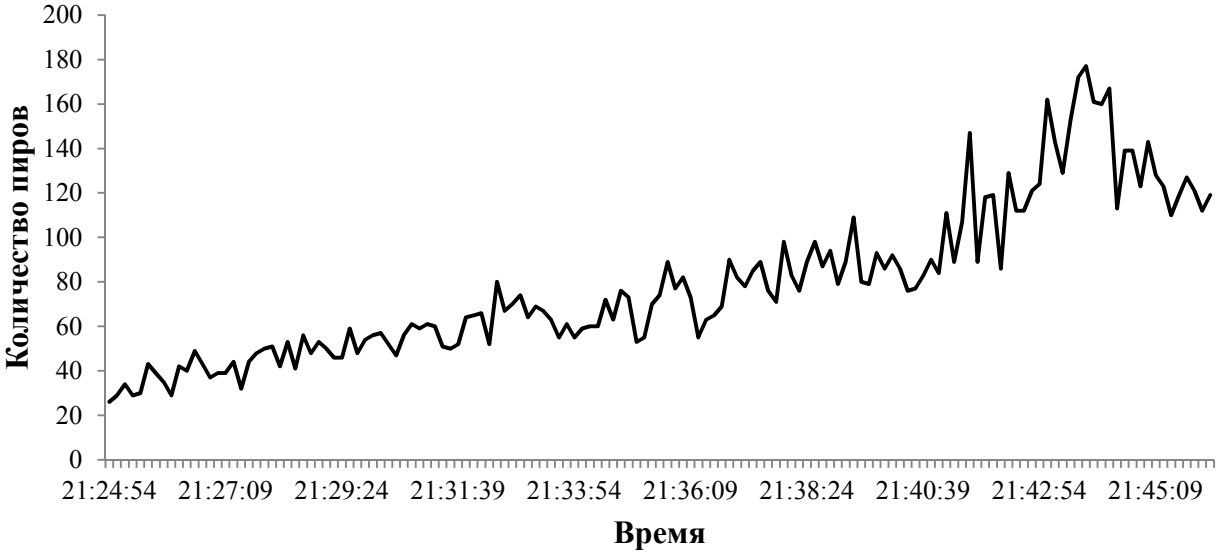


Рис.2. Временной ряд количества внешних пиров

Возможность прогнозирования временного ряда определяется показателем Херста, который рассчитывается следующим образом [14].

1. Среднее значение временного ряда $\bar{X}$ определяется по формуле 1.

$$\bar{X} = \frac{1}{N}\sum_{t=1}^{N} X_t \qquad (1)$$

где:

N – количество значений;

X_t – значение ряда.

2. Накопившееся отклонение ряда измерений $X_{t,N}$ определяется по формуле 2.

$$X_{t,N} = \sum_{i=1}^{t}(X_i - \bar{X}) \quad (2)$$

где:

N – количество элементов ряда, для которых определяется накопившееся отклонение.

3. Разность максимального и минимального накопившегося отклонений R определяется по формуле 3.

$$R(N) = max(X_{t,N}) - min(X_{t,N}), \quad 1 \le t \le N \quad (3)$$

4. Среднеквадратическое отклонение S определяется по формуле 4.

$$S = \sqrt{\frac{1}{N}\sum_{t=1}^{N}(X_t - \bar{X})^2} \quad (4)$$

5. Показатель Херста H рассчитывается по формуле 5.

$$H = \frac{log(\frac{R}{S})}{log(\frac{N}{2})} \quad (5)$$

В случае $0{,}5 < H < 1$ процесс называют персистентным, то есть процесс имеет длительную память. Иными словами, если в течение некоторого времени в прошлом наблюдались положительные приросты процесса, то и в дальнейшем в среднем будет происходить увеличение.

При $H = 0{,}5$ отклонения процесса от среднего есть действительно случайными и не зависят от предыдущих значений.

В случае $0 < H < 0{,}5$ процесс является переменчивым, то есть увеличение относительно среднего в прошлом, в будущем изменится в противоположном направлении.

Для определения показателя Херста временного ряда, представленного на рисунке 2, использовалось приложение The SELFIS Tool[15]. В результате получен параметра Херста $H = 0{,}632$. Следовательно, временной ряд имеет предпосылки к его прогнозированию.

Прогнозирование количества внешних пиров

Временной ряд количества внешних пиров, представленный на рисунке 2, имеет четко выраженную тенденция роста. Поэтому для его прогнозирования был выбран метод экспоненциального сглаживания с учетом тренда (метод Хольта [16]). В этом методе учитывается локальный линейный тренд, присутствующий во временных рядах.

Метод Хольта состоит из следующих действий.

1. Экспоненциально сглаженная величина ряда L_t определяется по формуле 6.

$$L_t = \alpha Y_t + (1-\alpha)(L_{t-1} - T_{t-1}) \quad (6)$$

где:

α – постоянная сглаживания для данных ($0 \le \alpha \le 1$);

Y_t – новое наблюдение или реальное значение ряда в период t;

T_t – оценка тренда.

2. Оценка тренда определяется по формуле 7.

$$T_t = \beta(L_t - L_{t-1}) + (1-\beta)T_{t-1} \quad (7)$$

где:

β — постоянна сглаживания для оценки тренда ($0 \le \beta \le 1$).

3. Прогноз на p точек вперед $\hat{Y}_{t+p}$ выполняется по формуле 8.

$$\hat{Y}_{t+p} = L_t + pT_t \quad (8)$$

Точность прогноза определяется следующими параметрами: постоянная сглаживания для данных (α), постоянная сглаживания для оценки тренда (β). Эмпирическим способом были определены значения этих параметров: $\alpha = 0{,}3$, $\beta = 0{,}1$.

С учетом особенностей работы приложения SopCast, временных параметров сети и объемов передаваемых мультимедийных данных, было определено время, необходимое для перестроения топологии внутри сети ISP при увеличении количества внешних пиров. Это время не превышает 10 с. Так как текущий временной ряд был построен для интервала в 2 с, достаточно выполнить прогнозирование на 5 точек вперед.

С помощью приложения MS Excel был построен прогноз методом Хольта для временного ряда, представленного на рисунке 2. Для исходного временного ряда с определенной точки строится прогноз на 5 точек вперед, эти значения заносятся во временной ряд-прогноз. Следующие 5 точек ряда-прогноза получаются с учетом реальных значений до текущей точки. Таким образом, на каждом шаге прогнозирования в качестве исходных данных берутся ре-

альные, а не спрогнозированные значения ряда. На рисунке 3 представлены исходный временной ряд и ряд-прогноз.

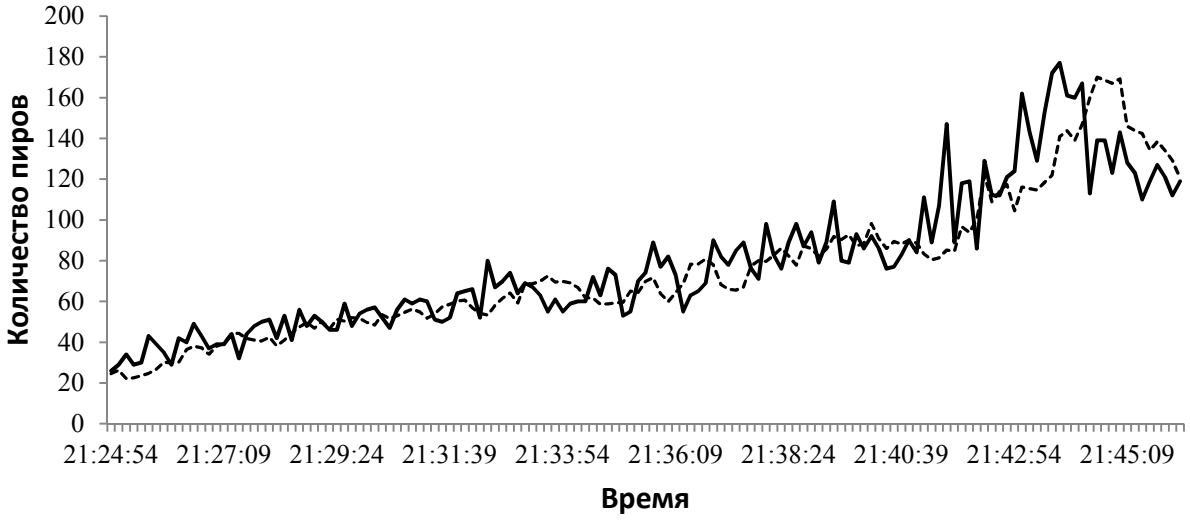


Рис.3. Прогнозирование методом Хольта

Выводы

В работе предложен метод локализации мультимедийного трафика в одноранговой сети на основе прогнозирования количества внешних пиров. Было доказано, что временной ряд количества внешних пиров можно прогнозировать на требуемый горизонт. Для прогнозирования временного ряда был выбран метод экспоненциального сглаживания с учетом тренда. Этот метод обеспечивает достаточную точность для выполнения преждевременной перестройки логической топологии одноранговой сети при изменении количества пиров в ней. Таким образом, использование предложенного метода позволяет повысить качество передачи мультимедийных данных конечным пользователям.

Список литературы

1. Postel J. RFC 791 – Internet Protocol / J. Postel // IETF. – Sep 1981.
2. End-user multimedia QoS categories // ITU-T Recommendation G.1010. – Nov 2001.
3. SopCast Software [Электронный ресурс] // – Режим доступа: http://www.sopcast.org. – Дата доступа: 09.04.2014.
4. PPTV Software [Электронный ресурс] // – Режим доступа: http://www.pplive.com – Дата доступа: 09.04.2014.
5. PPS Software [Электронный ресурс] // – Режим доступа: http://www.pps.tv – Дата доступа: 09.04.2014.
6. UUSee Software [Электронный ресурс] // – Режим доступа: http://www.uusee.com – Дата доступа: 09.04.2014.
7. Karagiannis T., Rodriguez P., Papagiannaki K. Should Internet Service Providers Fear Peer-Assisted Content Distribution? // The 5th ACM SIGCOMM conference on Internet Measurement, 2005. – P. 6.
8. Picconi F., Massoulie L. ISP-friend or foe? Making P2P live streaming ISP-aware // The 29th IEEE International Conference on Distributed Computer Systems (ICDCS), 2009. – P. 413-422.
9. Seedorf J., Niccolini S., Stiemerling M., Ferranti E., Winter R. Quantifying operational cost-savings through alto-guidance for P2P live streaming // 3rd Workshop on Economic Traffic Management (ETM 2010), 2010.
10. Jesi G.P., Montresor A., Babaoglu O. Proximity-aware superpeer overlay topologies // IEEE Transactions on Network and Service Management, 2007.
11. Петерс Э. Фрактальный анализ финансовых рынков: Применение теории Хаоса в инвестициях и экономике // М.: Интернет-Трейдинг, 2004 – 304 с.

12. Bermudez I., Mellia M., Meo M. Investigating overlay topologies and dynamics of p2p-tv systems: the case of sopcast // Selected Areas in Communications IEEE Journal on 29(9), 2011. – P. 1863-1871.
13. Tang S., Lu Y., Hernandez J.M., Kuipers F.A., P. Van Mieghem. Topology dynamics in a P2PTV network // Networking, 2009. – P. 326-337.
14. TCPDump Software [Электронный ресурс] // – Режим доступа: http://www.tcpdump.org . – Дата доступа: 09.04.2014.
15. Karagiannis, T., Faloutsos, M. SELFIS: A Tool For Self-Similarity and Long-Range Dependence Analysis // 1st Workshop on Fractals and Self-Similarity in Data Mining: Issues and Approaches (in KDD), Edmonton, Canada, 2002.
16. John E. Hanke, Dean W.Wichern. Business Forecasting / 8th edition. – Pearson, 2005.

УДК 519.854.2

ЗГУРОВСКИЙ М.З.,
ПАВЛОВ А.А.,
МИСЮРА Е.Б.,
МЕЛЬНИКОВ О.В.,
ЛИСЕЦКИЙ Т.Н.

МЕТОДОЛОГИЯ ПОСТРОЕНИЯ ЧЕТЫРЕХУРОВНЕВОЙ МОДЕЛИ ПЛАНИРОВАНИЯ, ПРИНЯТИЯ РЕШЕНИЙ И ОПЕРАТИВНОГО ПЛАНИРОВАНИЯ В СЕТЕВЫХ СИСТЕМАХ С ОГРАНИЧЕННЫМИ РЕСУРСАМИ

В статье методология построения четырехуровневой модели планирования, принятия решений и оперативного планирования в системах с сетевым представлением технологических процессов и ограниченными ресурсами иллюстрируется наиболее общим примером с использованием сети общего вида и наиболее сложного синтетического критерия планирования, состоящего из пяти базовых критериев.

In the article the construction methodology of the four-level planning, decision making and operational planning model in systems with a network representation of the technological processes and limited resources is illustrated by the most general example using the common view network and the most complex synthetic criterion of planning that consists of five basic criteria.

Введение

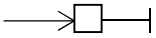
В организационно-технологических системах реальные процедуры планирования, управления и принятия решений методологически сочетают в себе как формальные модели, так и решения, принимаемые лицом, принимающим решения (ЛПР) и экспертами. Эта идеология реализована в данной работе в алгоритмическом обеспечении четырехуровневой модели планирования, принятия решений и оперативного планирования, представленном пятью блоками, где действия ЛПР и экспертов определяются в блоке принятия решений (блок 1), а формальные модели реализуются в блоках 2–5.

В [1] представлена четырехуровневая модель планирования, принятия решений и оперативного планирования, являющаяся обобщением модели планирования, изложенной в [2]. На первом уровне строится агрегированная модель планирования выполнения изделий, в которой все производство агрегируется в один прибор, на втором уровне модели – согласованные планы выполнения агрегированных работ в соответствии с одним из 31 критериев оптимальности. На третьем уровне модели строится и решается многоэтапная сетевая задача календарного планирования (выполняется точное планирование) и в блоке принятия решений выбирается один из альтернативных планов, полученных на третьем уровне, по критерию минимизации риска неполучения прибыли. Четвертый уровень предназначен для оперативного планирования – оперативной коррекции построенного на третьем уровне плана, в случае его частичного невыполнения.

В данной статье приведен расширенный, наиболее общий пример реализации четырехуровневой модели планирования, принятия решений и оперативного планирования для самого сложного из синтетических критериев планирования.

Сформулируем базовые элементы, на основе которых синтезируется исходная сетевая модель многоэтапной задачи календарного планирования [1].

Элемент типа 1.

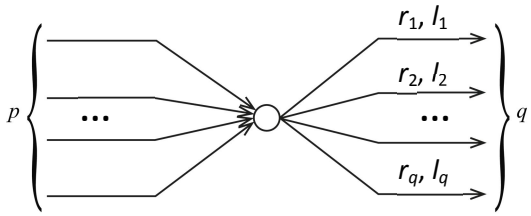


l – длительность выполнения работы, r – момент времени готовности работы к выполнению на приборе, C – момент окончания выполнения работы, d – директивный срок работы. В допустимом расписании $C \le d$. Символом ⊣ на сети обозначается окончание выполнения конечной работы (не имеющей преемников).

Стрелка, *входящая* в кружок ○, обозначает выполненную работу. Стрелка, *выходящая* из кружка – новая невыполненная работа. □ – обозначение прибора (ресурса), на котором выполняется работа. Приборы виртуальные, физически приборов реально меньше, т. к. на одном и том же физическом приборе может (в соответствии с технологией) выполняться работа (группа работ) после выполнения (по технологии) других работ (групп работ). Приборы и работы обозначаются уникальным номером, причем цифра в начале номера работы означает уровень входимости (на первом уровне входимости данной работе не предшествует ни одна другая работа). Значком ◌ обозначены фиктивные работы, соответствующие окончанию выполнения изделия. Под из-

делием понимается либо физически одно изделие, либо партия физически однотипных изделий.

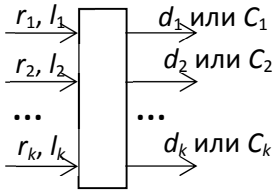
Расширим функциональность модели. Пусть в общем случае в элемент типа 1 может входить множество выполненных работ, а выходить из элемента типа 1 – множество невыполненных работ, т.е. элемент типа 1 отражает появление нового множества, состоящего из q невыполненных работ ($q \geq 1$) в результате окончания выполнения предыдущего множества из p работ ($p \geq 1$):



Множество выполненных работ каждого элемента типа 1 должно содержать работы по всем изделиям, представленным в множестве невыполненных работ.

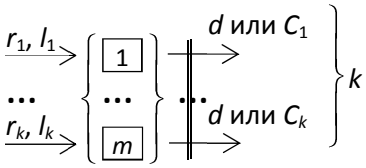
Элемент такого типа лишь описывает отношения предшествования между работами, но не является каким-либо объединением работ.

Элемент типа 2.



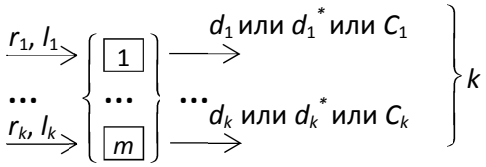
Прибор последовательно без прерываний выполняет k работ. r_i – моменты готовности выполнения работ на приборе, d_i – директивный срок выполнения i-й работы на приборе. В допустимом расписании $C_i \leq d_i$.

Элемент типа 3.



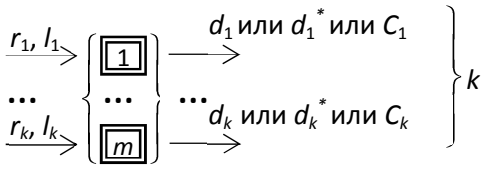
m независимых (идентичных) параллельных приборов без прерываний выполняют k работ, которые должны быть выполнены к одному директивному сроку d. В допустимом расписании $C_i \leq d$, $i = \overline{1,k}$. Моменты запуска приборов одинаковы, либо произвольны, либо заданы интервалы времени между моментами запуска приборов.

Элемент типа 4.



m независимых (идентичных) параллельных приборов *равной* производительности, работающих без прерываний, выполняют k работ, которые должны быть выполнены к различным директивным срокам d_i. $d_i^* = [d_i - \Delta_i, d_i]$, $i = \overline{1,k}$, определяет диапазон получения допустимого решения: $C_i \leq d_i$ либо $C_i \in d_i^*$, $\Delta_i > 0$. Моменты запуска приборов могут быть различны.

Элемент типа 5.



m независимых параллельных приборов *различной* производительности, работающих без прерываний, выполняют k работ, которые должны быть выполнены к различным директивным срокам d_i. $d_i^* = [d_i - \Delta_i, d_i]$, $i = \overline{1,k}$, определяет диапазон получения допустимого решения: $C_i \leq d_i$ либо $C_i \in d_i^*$, $\Delta_i > 0$. Моменты запуска приборов могут быть различны.

В данной статье показан пример, в котором, с целью упрощения, элементы типа 5 задаются как m пропорциональных (*proportional*) (в литературе также называются однотипными – *uniform*), а не произвольных (несвязанных – *unrelated*) приборов. Такой случай охватывает большинство практических приложений, однако это не является ограничением для общей методологии планирования.

Сетевая модель не является произвольной и точно отражает реальный технологический процесс. При этом номера работ, соответствующие индексам i их параметров l_i, d_i, r_i и т.д., для каждого элемента свои и определяются технологией выполнения работ.

Полное описание сетевой модели см. в [1].

Алгоритмическое обеспечение четырехуровневой модели

Алгоритмическое обеспечение модели состоит из пяти блоков, где действия ЛПР и экспертов определяются в блоке принятия решений (блок 1), а формальные модели реализуются в блоках 2–5:

Блок 1. Блок принятия решений.

Блок 2. Построение модели первого уровня.

Блок 3. Построение модели второго уровня.

Блок 4. Построение модели третьего уровня.

Блок 5. Оперативное планирование.

Рассмотрим их более подробно.

Блок 1. Блок принятия решений [2].

Функции блока принятия решений:

1) формирование портфеля заказов;

2) задание критерия качества планирования: выбор функционала; определение коэффициен-

тов, задающих функционал; задание минимальной ожидаемой прибыли;

3) задание технологии, реализующей портфель заказов: топология, связанная с технологией, задается в виде N отдельных подсетей, где N – число базовых критериев [1] в общем функционале качества; подсети могут иметь общее оборудование. Для синтетического критерия, таким образом, рассматривается частный случай, когда исходная сеть представляется в виде независимых подсетей с общими виртуальными приборами;

4) согласование с заказчиком при необходимости всех изменений плана (если не получено допустимое расписание);

5) формирование нового портфеля заказов (если полученная прибыль меньше минимально ожидаемой);

6) выбор наилучшего плана по критерию минимизации риска неполучения расчетной прибыли [2] из множества альтернативных планов, полученных на третьем уровне планирования;

7) организация взаимодействия между блоками 2–4;

8) осуществление оперативного планирования с использованием моделей блока 5;

9) при планировании по базовому критерию оптимальности, в случае необходимости, экспертным путем устанавливаются желаемые моменты окончания выполнения изделий, которые передаются на модель третьего уровня в качестве директивных сроков.

Так как модель второго уровня формализуется с учетом решения, полученного на модели первого уровня, а модель третьего уровня – с учетом решения, полученного на модели второго уровня, то формулировка модели и ее решение объединяются в единый блок.

Блок 2. Построение модели первого уровня.

В блоке 2 строится агрегированная модель планирования, на которой решаются оптимизационные задачи по критерию минимизации суммарного взвешенного момента окончания выполнения агрегированных работ (МВМ).

2.1. Постановка задачи: построение технологической сети, задающей технологию выполнения изделий (задание технологии гарантирует адекватность технологической сети реальному технологическому процессу). Задание одного из 31 критериев оптимальности (пяти базовых критериев и 26 синтетических критериев, являющихся всеми возможными линейными свертками базовых), задание минимальной ожидаемой прибыли предприятия. По каждой работе i задается удельная длительность выполнения l_i для одного изделия, а моменты начала r_i и окончания C_i выполнения работы i будут рассчитан при решении задачи. По каждому изделию i задаются число изделий в портфеле заказов N_i, вес ω_i, директивный срок d_i и диапазон получения допустимого решения Δ_i (для тех изделий, в технологии выполнения которых встречаются элементы типа 4 и 5). Для каждого ресурса указывается тип элемента и число параллельных приборов, для элемента типа 5 задаются производительности приборов s_i (длительность работы l_i задается для прибора с единичной производительностью).

Базовые критерии оптимизации [1]:

Критерий 1. Максимизация суммарной прибыли системы планирования в случае отсутствия директивных сроков:

$$\max\left\{\sum_{i=1}^{n}\omega_i(T)\cdot(T-C_i)\right\}+P-З,$$

где P – гарантированный минимальный доход от продажи (выполнения) всех n изделий (серий изделий), $З$ – все издержки, $\omega_i(T)$ – весовой коэффициент i-го изделия (i-й серии изделий), определенный экспериментальным путем; T – плановый период; $C_i \le T$ – момент окончания выполнения i-го изделия (i-й серии изделий), соответствующий моменту окончания выполнения его конечной вершины.

Критерий 2. Максимизация суммарной прибыли системы планирования при условии: для всех изделий $i \in I$ введены директивные сроки d_i, которые не могут быть нарушены (планирование «точно в срок»):

$$\max\left\{\sum_{i=1}^{n}\omega_i U_i\right\}-З, \text{ где } U_i=\begin{cases}1, C_i=d_i\\0, C_i\ne d_i\end{cases},$$

ω_i – прибыль от выполнения i-го изделия, если оно выполнено точно в срок, $З$ – все издержки.

Критерий 3. Максимизация суммарной прибыли системы планирования при условии: для всех изделий $i \in I$ введены директивные сроки d_i, необходимо минимизировать суммарное взвешенное запаздывание выполнения изделий относительно директивных сроков:

$$\max\left\{P-\sum_{i=1}^{n}\omega_i\max(0, C_i-d_i)\right\}-З,$$

где P – гарантированный минимальный доход от продажи (выполнения) всех n изделий, если все они выполнены без запаздывания, $З$ – все издержки; ω_i – штраф за запаздывание окончания выполнения i-го изделия относительно директивного срока на единицу времени.

Критерий 4. Для всех изделий $i \in I$ введены директивные сроки d_i. Для каждого изделия ука-

зана величина ω_i – абсолютная прибыль от выполнения изделия, не зависящая от момента окончания выполнения изделия в том случае, если изделие выполняется без запаздывания относительно директивного срока, иначе прибыль системы планирования по этому изделию равна нулю. Задача – максимизировать суммарную прибыль системы планирования:

$$\max\left\{\sum_{i=1}^{n}\omega_i U_i\right\} - З, \quad \text{где } U_i = \begin{cases}1, C_i \le d_i\\ 0, C_i > d_i\end{cases},$$

где ω_i – прибыль от выполнения i-го изделия, если оно выполнено без запаздывания относительно директивного срока; $З$ – риск уменьшения прибыли из-за срыва выполнения изделия в срок.

Критерий 5. Для всех изделий заданы директивные сроки d_i. Необходимо минимизировать суммарный штраф системы планирования, как за опережение, так и за запаздывание относительно директивных сроков:

$$\max\left\{P - \sum_{i=1}^{n}\omega_i\left|C_i - d_i\right|\right\} - З,$$

где P – гарантированный минимальный доход от продажи (выполнения) всех n изделий (изделий), если все они выполнены без опережения и запаздывания, $З$ – все издержки; ω_i – штраф за отклонение момента окончания выполнения i-го изделия от директивного срока на единицу времени.

Синтетические критерии 6–31 – это линейная свертка всех возможных комбинаций пяти базовых критериев. Пример синтетического критерия приведен на стр. 75, формула (2). Для синтетического критерия издержки $З$ определяются экспертом, поскольку изделия по разным базовым критериям выполняются на общем оборудовании.

На этапе планирования директивные сроки могут быть нарушены, а величины коэффициентов ω_i в базовых критериях определяют уменьшение расчетной прибыли.

Построенная топология должна однозначно определять (для построения критических путей изделий), к какому изделию относится та или иная работа из множества выполненных работ каждого элемента типа 1. Поэтому для элементов, где такое соответствие неоднозначно (т. е. работы слева от элемента типа 1 принадлежат разным изделиям), необходимо задать это соответствие. Для остальных работ их принадлежность конкретному изделию очевидным образом определяется по исходной топологии.

Изделия разбиваются на $N \in [1, 5]$ отдельных групп по числу базовых составляющих синтетического критерия, ни одно изделие не может входить в две разные группы. Топология, связанная с технологией, задается в виде N отдельных подсетей, соответствующих N отдельным группам изделий. Рассмотрим процедуру решения задачи по базовому критерию (количество подсетей $N = 1$), а затем отдельно опишем планирование по синтетическому критерию ($N \in [2, 5]$).

2.2. *Первый уровень агрегации* – построение агрегированного представления исходной сети посредством агрегации работ и ресурсов.

2.2.1. Построение агрегированных работ и ресурсов осуществляется таким образом. В соответствии с [2], агрегированной работой называется совокупность работ, выполняемых по одному изделию в рамках одного захода в агрегированный ресурс (агрегированные ресурсы на схеме отображаются элементами типа 1 и 2). Все элементы типа 3–5 становятся элементами типа 2. Если некоторая цепочка работ одного изделия выполняется последовательно на элементе с одним и тем же номером, то она объединяется в одну агрегированную работу. Если при этом на элементе типа 2 остается одна работа, то этот элемент преобразуется в элемент типа 1. Длительность выполнения агрегированной работы определяется: по элементам типа 2 – суммой длительностей работ, входящих в ее состав; по элементам типа 3, 4 – суммой длительностей работ, входящих в ее состав, деленной на количество приборов, входящих в состав элемента; по элементу типа 5 – суммой приведенных длительностей работ, входящих в ее состав. Приведенная длительность каждой работы определяется по формуле [3, с.104]:

$$l = 1\Big/\sum_{i=1}^{m}\left(1/l_i\right), \qquad (1)$$

где l_i – длительность работы на приборе i. Длительность работы j на приборе i элемента 5 равна l_j/s_i, где s_i – производительность прибора i (длительность работы l_j задается для прибора с единичной производительностью).

В результате агрегации некоторые конечные работы изделий могут быть объединены в одну конечную работу.

2.2.2. В связи с тем, что параллельные приборы (элементы типа 3–5) агрегируются в один прибор (элементы типа 1–2), меняется длительность работ, оставшихся на элементах и не объединенных с другими работами. Длительность этих работ рассчитывается по тем же правилам: по элементам типа 1–2 не меняется; по элементам типа 3–4 делится на количество приборов, входящих в состав элемента; по элементу типа 5 – рассчитывается по формуле (1).

В результате агрегированная сеть первого уровня агрегации может содержать как агрегированные работы, так и неагрегированные (с

исходной или измененной длительностью), если они не вошли в состав агрегированных работ.

2.3. *Поиск критических путей изделий.* Несмотря на меньшую размерность задачи в агрегированном представлении, она все же остается труднорешаемой. Поэтому для упрощения модели вводятся следующие ограничения:

- длительность выполнения каждого изделия определяется его критическим путем;
- общие агрегированные работы разных изделий лежат на их критических путях и выполняются в одном ресурсе (элементе).

Таким образом, для каждого изделия определяется его критический путь – маршрут максимальной длительности по агрегированным работам от начала изготовления изделия до его полного выполнения. Для реализации алгоритма поиска критических путей должно быть гарантировано, что сеть является ориентированным ацикличным графом, а также созданы две промежуточные структуры (табл. 1 и 2; стек – это список типа «последний вошел – первый вышел»).

Табл. 1. Список работ критического пути

Наименование поля	Обозначение
Номер работы-преемника по критическому пути	N_Z
Длина максимального пути до изделия	L_{max}

Табл. 2. Стек для рекурсивного просмотра связей надграфа

Наименование поля	Обозначение
Номер работы-предшественника	N_P
Номер работы-преемника	N_Z

В отличие от алгоритмов поиска кратчайшего пути, в этом случае невозможно избежать полного прохода по сети, но перебор должен быть направленным. Для древовидных сетей простая рекурсивная процедура поиска пути в наддереве (или поддереве, для исходящих деревьев) позволяет обойти все пути в сети и при этом попасть на каждую работу только один раз. Для сетей общего вида не исключены случаи повторного попадания на уже рассмотренную работу. Поэтому возникает необходимость рассмотреть возможности сокращения перебора при повторном попадании. Самый длинный путь от заданной работы до конечной работы изделия можно хранить в таблице по номеру работы. Если создать фиктивную работу с номером 0 как предшественника всех агрегированных работ первого уровня входимости, то критический путь изделия будет храниться в той же таблице в ее нулевой записи.

Если при повторном попадании на работу меняется длина максимального пути от нее до изделия, то очевидно, что на ту же величину меняется длина максимальных путей всех работ ее надграфа. Проблема была бы решена, если бы удалось без повторного перебора связей пересчитать весь надграф для любой работы. Но поиск по надграфу требует рекурсивного рассмотрения предшественников каждой работы, что аналогично полному повторному перебору. Существует способ сократить перебор: если работа, на которую пришлось повторное попадание, лежит на самом длинном до сих пор найденном пути, который хранится в нулевой записи таблицы, то достаточно увеличить все длины только на этом пути, а не во всем надграфе. Тогда повторное попадание на работы, не лежащие на этом пути, приведет просто к повторному перебору, и пересчет длины даст правильную характеристику вне зависимости от ее предыдущего значения.

Теоретически существует два алгоритма перебора работ: методы прямого и обратного прохода. В первом случае используется поиск по преемникам, во втором – по предшественникам.

Описание алгоритма поиска критического пути по одному изделию *(по методу обратного прохода):*

1. Занести в список конечную (фиктивную) работу изделия с $N_Z = 0$ и $L_{max} = 0$, записать в стек ее предшественников – конечные работы изделия.

2. Обработка вершины стека: если N_P отсутствует в списке, то записать ее в список с N_Z из стека и длиной $L_{max} = L_{max}(N_Z) + L(N_P)$, где $L(j)$ – длительность j-й агрегированной работы.

Иначе, если $L_{max}(N_Z) + L(N_P) > L_{max}(N_P)$, то заменить N_Z на рассматриваемую, L_{max} на $L_{max}(N_Z) + L(N_P)$; пересмотреть список (путь от нулевой фиктивной работы до изделия), т.е. найти N_P среди преемников: если N_P лежит на этом пути (повторное попадание на ту же работу), то пройти путь еще раз, увеличив на этом пути длину L_{max} на $\Delta = L_{max}(N_Z) + L(N_P) - L_{max}(N_P)$, иначе занести предшественников работы N_P в стек для повторного перебора.

3. Удалить вершину стека. Если N_P занесли в первый раз, то занести в стек всех ее предшественников.

4. Если стек пуст, то переход на шаг 5, иначе на шаг 2.

5. Критический путь найден – это последовательность агрегированных работ в списке, начиная с нулевой записи.

2.4. *Второй уровень агрегации* – построение сети на критических путях изделий с общими вершинами – общими агрегированными работами, принадлежащими разным изделиям одной

группы. Сеть при этом имеет меньшую размерность, т.к. она содержит только агрегированные работы, принадлежащие критическим путям изделий, а также некоторые агрегированные работы объединены в общие вершины.

2.4.1. *Поиск общих вершин на критических путях изделий.* Агрегированные работы, принадлежащие разным критическим путям и выполняемые в рамках одного захода в ресурс, объединяются в общие вершины, если такое объединение, определяемое технологией изготовления изделий, позволяет сократить длительность прохождения изделий в системе или иным способом улучшить производство – например, уменьшить время переналадок оборудования, повысить общее энергосбережение или уменьшить стоимость работ. Длительность выполнения общей вершины определяется суммой длительностей агрегированных работ, входящих в ее состав.

Разработаны несколько правил объединения в общие вершины:

1) если заданы времена переналадки приборов, то в общие вершины объединяются те агрегированные работы, выполняемые на одном элементе, моменты запуска которых по их критическим путям отличаются не более, чем на длительность переналадки прибора: $\Delta_{ОВ} = l_{нал}$ (считается разница суммарных длительностей по критическим путям: для групп изделий, оптимизируемых по базовому критерию 1 – от работ первого уровня входимости до объединяемых, а для групп изделий, по которым заданы другие базовые критерии – от директивных сроков до объединяемых работ);

2) если есть работы, близкие по технологии, для которых выгодно совместное использование оборудования, то вводится разница $\Delta_{ОВ}$, определяемая экспериментальным путем, обычно равная 5–10% от максимальной длины критического пути изделий, и в общие вершины объединяются те агрегированные работы, выполняемые на одном приборе, моменты запуска (для изделий, оптимизируемых по базовому критерию 1) или окончания (для остальных базовых критериев) которых по их критическим путям отличаются не более, чем на $\Delta_{ОВ}$;

3) если в системе присутствует уникальное оборудование (например, требующее непрерывной работы или сложной переналадки в случае остановки), то с целью его более эффективного использования может быть введено объединение в общие вершины только на этом оборудовании, при этом $\Delta_{ОВ}$ может быть увеличена до 20–30% от максимальной длительности критического пути.

Величина $\Delta_{ОВ}$ и правило объединения общих вершин задаются экспертом в блоке принятия решений либо выбираются из правил, описанных в [2, гл. 9].

Примечание. В общие вершины нельзя объединять конечные работы изделий (нагруженные весом), а также работы, принадлежащие множеству q невыполненных работ элемента типа 1 при $q > 1$, кроме тех работ, объединение которых не приводит к появлению связей, которых раньше в технологии не было (т.е. не появляется, например, предшествование новой общей вершине работы, которая раньше не предшествовала ни одной из объединяемых вершин).

Пример. В части агрегированной подсети, показанной на рис. 1, нельзя объединять работы 310 и 311, т.к. тогда работа 312 будет предшествовать этой общей вершине и появится предшествование ее работе 39, а такой связи в технологии не было. Однако, можно объединить работы 211 и 212, т.к. новых связей не появляется.

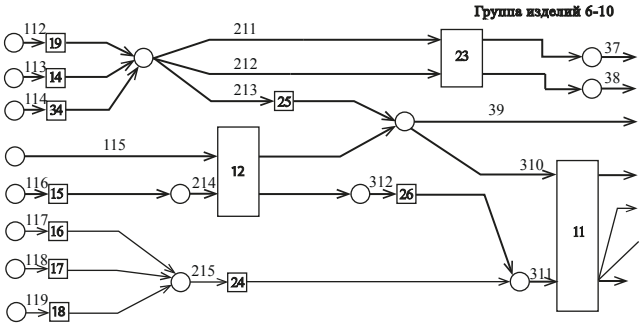


Рис. 1. Проверка на появление новых связей

2.4.2. *Построение сети на критических путях изделий* с общими вершинами: критические пути объединяются на основе общих вершин. При этом удаляются работы, не лежащие на критических путях, однако остаются их связи с оставшимися работами, чтобы не нарушились отношения предшествования, изначально имевшиеся в сети.

Пример. В части агрегированной сети, показанной на рис. 2 (см. ниже), удаляемая работа 238 и, следовательно, работа 141 (изделие 22) предшествует работе 332 (изделие 21), а работа 237 (изделие 21) предшествует удаляемой работе 334 и, следовательно, работе 439 (изделие 22). Поэтому в сети на критических путях следует оставить связи от 141 к 332 и от 237 к 439.

Так как не все конечные работы изделий лежат на их критических путях, то некоторые из них могут быть удалены из списка конечных работ изделий при построении сети на критических путях.

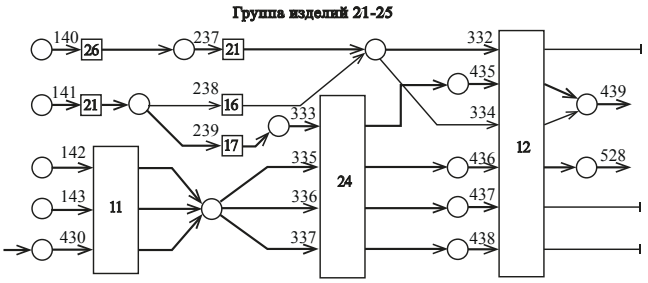


Рис. 2. Проверка удаляемых связей

2.5. Построение и решение оптимизационных задач по критерию минимизации суммарного взвешенного момента окончания выполнения агрегированных работ (МВМ) на сети второго уровня агрегации (п. 2.4).

2.5.1. Для каждой базовой составляющей синтетического критерия, входящей в заданный функционал, кроме базового критерия 1, строится и решается аппроксимирующая задача МВМ, для базового критерия 1 решается задача МВМ без изменения параметров [1]. Задача решается с помощью алгоритма, описанного в [2, гл. 9].

2.5.2. В результате решения задачи МВМ получена оптимальная (субоптимальная, если алгоритм приближенный) последовательность выполнения изделий. Эта последовательность служит для определения очередности назначения изделий при согласованном планировании.

При планировании *по синтетическому критерию* исходная сеть представляет собой совокупность N подсетей, не имеющих общих связей и связанных только общим оборудованием, каждая подсеть соответствует своему базовому критерию. В этом случае процедуры, описанные в п.2.2–2.5.2, осуществляются для каждой подсети отдельно, в результате получаем N оптимальных (субоптимальных) последовательность выполнения изделий, для каждой из N подсетей.

Блок 3. Построение модели второго уровня.

В блоке 3 осуществляется формализация и решение задачи согласованного планирования выполнения агрегированных работ на сети первого уровня агрегации с учетом заданного критерия оптимальности.

3.1. Постановка задачи согласованного планирования *по базовому критерию*: на сети первого уровня агрегации (п. 2.2) найти последовательность выполнения агрегированных работ, минимизирующую суммарную длительность прохождения работ (эквивалентно минимизации простоев оборудования) при ограничении: порядок назначения работ определяется приоритетно-упорядоченной последовательностью, полученной в результате решения задачи МВМ (информация передается из блока 2).

Постановка задачи согласованного планирования *по синтетическому критерию*: на совокупности N подсетей первого уровня агрегации (п. 2.2) найти последовательность выполнения агрегированных работ, минимизирующую суммарную длительность прохождения работ (эквивалентно минимизации простоев оборудования) при ограничениях: назначение агрегированных работ осуществляется группами; группа работ задается соответствующей задачей МВМ (информация передается из блока 2), а в рамках каждой группы изделий порядок определяется приоритетно-упорядоченной последовательностью, полученной в результате решения задачи МВМ, соответствующей этой группе. Простои оборудования, полученные в результате выполнения всех работ предыдущих групп изделий (в соответствии с очередностью их назначения на выполнение) используются для назначения работ из следующей распределяемой группы изделий (в соответствии с очередностью их назначения в этой группе).

Рассмотрим сначала методику согласованного планирования по базовому, а потом по синтетическому критерию оптимальности.

3.1.1. С целью отражения полной технологии изготовления изделий, для определения очередности назначения агрегированных работ внутри каждой группы изделий, последовательность, полученная при решении задачи МВМ, дополняется агрегированными работами, не лежащими на критических путях изделий, в соответствии с технологическими связями. При этом перед каждой агрегированной работой, лежащей на критических путях, в последовательность вставляется надграф ее предшественников, которые еще не включены в последовательность. Общие вершины разбиваются на их составляющие (в последовательность включаются в виде цепочки), при этом есть возможность начинать выполнение следующих по технологии агрегированных работ сразу после окончания каждой составляющей. Получаем дополненную оптимальную последовательность (последовательность σ^*). Список конечных работ изделий также возвращается к виду, полученному на первом уровне агрегации.

3.1.2. Построение согласованного плана выполнения агрегированных работ. Реализация согласованного планирования основана на следующих *принципах*:

1) Максимальное приближение длительности прохождения каждого изделия в системе к его критическому пути, что позволяет:

уменьшить объемы незавершенных работ;

сократить длительности выполнения изделий;

уменьшить простои и тем самым увеличить

производственную мощность;

повысить рентабельность системы планирования.

2) Изделия, для которых отсутствуют директивные сроки, назначаются на выполнение в очередности, определенной приоритетно-упорядоченной последовательностью, с начала в конец, по правилу: чем выше приоритет изделия, тем раньше оно должно быть назначено на выполнение.

3) Изделия с директивными сроками назначаются на выполнение в очередности, определенной приоритетно-упорядоченной последовательностью, максимально близко к директивному сроку, с конца в начало – от конечных до начальных работ изделия, по правилу: чем выше приоритет, тем меньше отклонение выполнения изделия от директивного срока.

Для реализации распределения работ отдельно по каждой группе изделий разработаны следующие *алгоритмы распределения*: построение компактных расписаний (алгоритм 1); построение незадерживающих расписаний (алгоритм 2); построение расписаний, обеспечивающих выполнение в заданные директивные сроки изделий с наивысшим приоритетом (алгоритм 3).

Определение 1. Календарный план называется компактным [3], если ни одна из его операций не допускает сдвига влево, т. е. такого смещения влево, которое ограничено только предыдущей операцией этой же детали (или допустимым моментом начала ее обработки) и не приводит к увеличению моментов начала выполнения других операций. Компактный календарный план получается путем допустимого сжатия как «скольжением», так и «переносом» операций.

Определение 2 [3]. Календарный план называется незадерживающим, если в любой момент в нем ни один станок не простаивает, если он свободен и доступен для выполнения операции.

В зависимости от заданного базового критерия, для группы изделий применяется один из разработанных алгоритмов. Алгоритмы 1 и 2 используются для распределения изделий, оптимизируемых по базовому критерию 1, алгоритм 3 применяется для изделий с директивными сроками (базовые критерии 2–5). При планировании по базовому критерию все изделия распределяются одним алгоритмом.

Алгоритмы 1 и 2 построены с использованием алгоритмов Гиффлера-Томпсона [3]. Можно показать, что всякое оптимальное решение календарного планирования с регулярным критерием является компактным календарным планом. Вместе с тем не всякий незадерживающий календарный план является оптимальным [3]. Однако незадерживающие календарные планы составляют очень важный подкласс, так как, во-первых, они просто строятся и, во-вторых, несмотря на возможность отсутствия среди них оптимального календарного плана, практические соображения указывают, что в большинстве случаев оптимальный календарный план является незадерживающим.

В компактных и незадерживающих календарных планах минимизируются простои между выполняемыми агрегированными работами и, следовательно, минимизируется длительность прохождения изделий в системе. Выбор алгоритма 1 или 2 осуществляется в зависимости от типа исходных данных: если длительности выполнения агрегированных работ, входящих в состав изделий, различаются несущественно (длительности одного порядка), выбирается алгоритм 2, в противном случае – алгоритм 1.

При согласованном планировании осуществляется распределение агрегированных работ последовательности σ^* по ресурсам. Структура последовательности σ^* следующая: первые позиции в этой последовательности занимает подпоследовательность максимального приоритета (ПМП) σ_1 (номер ПМП равен 1), готовая к выполнению в момент 0 и имеющая наивысший приоритет из всех возможных допустимых подпоследовательностей, организованных на множестве агрегированных работ J. За подпоследовательностью σ_1 следует подпоследовательность σ_2 (номер ПМП равен 2), построенная по аналогичным правилам, готовая к выполнению после выполнения σ_1 и имеющая наивысший приоритет из допустимых подпоследовательностей, организованных на множестве агрегированных работ $J \setminus \sigma_1$. Следующие позиции в последовательности σ занимает подпоследовательность σ_3 (номер ПМП равен 3), определяемая по правилам, изложенным выше. Процесс продолжается, пока не будет рассмотрено все множество агрегированных работ J.

Каждый из алгоритмов распределения, описанных ниже, построен на последовательном назначении агрегированных работ подпоследовательностей σ_i на резервы приборов (временные промежутки между уже назначенными ранее агрегированными работами). Под приборами понимаются элементы типа 1 и 2 агрегированной сети, полученной в п. 2.2. До начала распределения создается пустой список приборов без назна-

ченных работ, резервируется время под назначенные работы с момента 0 до конца планового периода *T*. Назначение агрегированных работ начинается либо с работ первого уровня входимости, т.е. не имеющих предшественников (в алгоритмах 1 и 2), либо с конечных работ изделий, т.е. не имеющих преемников по агрегированной сети (в алгоритме 3). Эти работы заносятся в список неназначенных работ. Далее, пока список неназначенных работ не пуст, в нем находится очередная агрегированная работа и по определенным правилам ищется такой резерв прибора для ее размещения, чтобы назначение работы не нарушало технологических ограничений. Если имеется два равноправных резерва достаточной для назначения работы длительности (например, если в системе предусмотрены параллельные однотипные приборы), выбирается резерв меньшей длительности. Если требуемый резерв найден, производится перенос работы из списка неназначенных в список назначенных агрегированных работ, т.е. назначение работы на найденный резерв прибора. Если требуемый резерв не найден, это значит, что выполнение данного изделия не укладывается в его директивный срок, при этом следует принять решение по данному изделию в блоке принятия решений. После назначения агрегированной работы в список неназначенных работ добавляются все ее преемники (для алгоритмов 1 и 2) или предшественники по агрегированной сети (для алгоритма 3) и по аналогичным правилам производится их назначение на выполнение. Согласованный план выполнения агрегированных работ представляет собой список назначенных агрегированных работ с информацией об их моментах запуска и окончания выполнения на приборах.

Ниже приводится алгоритм, с помощью которого строят компактные календарные планы, а затем его модификация для построения незадерживающих календарных планов.

Алгоритм распределения 1.

Построение компактных расписаний основано на следующей эвристике. На каждом шаге выбирается из последовательности σ^* очередная агрегированная работа с минимальным номером ПМП и назначается на прибор с минимальным временем освобождения после выполнения этой работы таким образом, чтобы не нарушались технологические ограничения. Минимальное время освобождения прибора определяется наиболее ранним временем начала выполнения агрегированной работы плюс длительность ее выполнения. Наиболее раннее время начала выполнения агрегированной работы равно максимальному из моментов окончания выполнения всех ее предшественников, определенных при назначении предшественников на выполнение. При назначении агрегированной работы ищут такой самый ранний свободный резерв в рамках фонда времени прибора (имеющихся простоев оборудования, полученных в результате выполнения всех предыдущих работ, в соответствии с очередностью их назначения), чтобы можно было разместить в нем агрегированную работу не раньше, чем наиболее раннее время ее начала. Это обеспечивает невозможность выполнения какой-либо агрегированной работы в более поздний момент без задержки момента начала выполнения какой-либо другой агрегированной работы.

Алгоритм распределения 2.

Следующая модификация алгоритма 1 позволяет получить незадерживающие расписания. Алгоритм 2 основан на такой эвристике: на каждом шаге назначается на выполнение из последовательности σ^* очередная агрегированная работа с минимальным временем начала выполнения и минимальным номером ПМП. Минимальное время начала выполнения работы определяется как максимальный из моментов окончания выполнения всех ее предшественников по сети.

Достоинством компактных и незадерживающих расписаний является минимизация простоев на приборах и, следовательно, минимизация времени прохождения изделий в системе, уменьшение объема незавершенных работ и повышение эффективности системы.

Алгоритм распределения 3.

Распределение агрегированных работ осуществляется на основании следующей эвристики. Последовательность σ^* рассматривается с начала, с изделий более высокого приоритета. На первом шаге выбираются из последовательности (в порядке их появления в σ^*) агрегированные работы из списка конечных работ и назначаются таким образом, чтобы момент окончания их выполнения соответствовал директивному сроку изделия (если задан диапазон допустимого решения Δ_i, то работа назначается в интервал $[d_i - \Delta_i, d_i]$). Если такое назначение невозможно, то есть требуемый ресурс занят другими работами, то следует назначить конечную работу максимально близко к директивному сроку (интервалу $[d_i - \Delta_i, d_i]$), причем для изделий, оптимизируемых по базовому критерию 2 или 3 – не позднее директивного срока (интервала $[d_i - \Delta_i, d_i]$). На каждом следующем шаге на ресурс назначается очередная аг-

регированная работа с максимальным временем запуска в ресурсе и минимальным номером ПМП. Максимальное время запуска агрегированной работы в ресурсе определяется наиболее поздним временем окончания выполнения агрегированной работы минус длительность ее выполнения. Наиболее пóзднее время окончания выполнения агрегированной работы определяется минимальным из времен запуска всех ее преемников по сети, определенных при назначении преемников на выполнение. При назначении агрегированной работы ищут такой наиболее поздний свободный резерв в рамках фонда времени прибора (имеющихся простоев оборудования, полученных в результате выполнения всех предыдущих работ, в соответствии с очередностью их назначения), чтобы можно было разместить в нем агрегированную работу не позже, чем наиболее позднее время ее окончания.

Если свободный резерв прибора требуемой длины найти не удается (изделие невозможно распределить на имеющиеся ресурсы), следует продолжить процедуру распределения по данному изделию, назначая агрегированные работы до начала планового периода (с отрицательными моментами запуска). В блоке принятия решений нужно определить, что делать с изделиями, по которым плановый период нарушен – например, сдвинуть начало выполнения группы изделий на более поздний период, уменьшить количество изделий в портфеле заказов или удалить их из портфеля заказов, изменить директивный срок одного из изделий, увеличить количество ресурсов и т. п. (все варианты требуют повторного решения задачи целиком, кроме сдвига начала выполнения и изменения директивного срока, которые требуют пересмотра общих вершин в сети на критических путях, повторного решения задачи МВМ и распределения).

При таком распределении в первую очередь на выполнение назначаются изделия с наивысшим приоритетом максимально близко к директивным срокам без их нарушения, а для запаздывающих изделий минимизируется величина суммарного штрафа за запаздывание относительно директивных сроков, что позволяет предприятию выполнить наибольшее количество изделий без нарушения директивных сроков и максимизировать прибыль предприятия.

Более подробно алгоритмы распределения приводятся в [4].

В реализациях всех алгоритмов распределения также осуществляется *проверка фактического объединения общих вершин* – составляющие общих вершин назначаются последовательно на один прибор, если разрыв между ними, т.е. разность между моментом окончания одной составляющей и моментом запуска другой составляющей, окажется меньше $\Delta_{ОВ}$ (определенным в соответствии с принятым при оптимизации правилом). Если окажется, что из-за занятости ресурсов общая вершина не может быть сформирована (разрыв между ее составляющими больше $\Delta_{ОВ}$), то следует изменить технологию (разбить такие общие вершины в сети второго уровня агрегации на составляющие с восстановлением связей первого уровня агрегации и исходных параметров агрегированных работ, полученных на первом уровне агрегации, п. 2.2). Если в общей вершине более, чем две составляющие, и разрыв больше $\Delta_{ОВ}$ имеется лишь между некоторыми из них (и невозможно ликвидировать его перестановкой составляющих между собой), то в составе общей вершины остаются лишь составляющие, имеющие разрыв, меньший $\Delta_{ОВ}$, при этом длительность общей вершины уменьшается до суммы их длительностей, и в агрегированную сеть второго уровня агрегации добавляются удаленные при агрегации связи с исключенными из общей вершины составляющими.

Все изменения, внесенные в агрегированную сеть второго уровня агрегации на этапе согласованного планирования, должны быть также отражены в агрегированной сети первого уровня агрегации и исходной сети (после утверждения согласованного плана, при передаче его на третий уровень). При разбивке общих вершин соответствующие агрегированные работы (в агрегированной сети) и соответствующие им исходные работы (в исходной сети), вместе с их отношениями предшествования, разносятся на элементы с тем же номером для выполнения в разное время (то есть, условие непрерывности выполнения работ теперь действует для отдельных групп работ, а не для одной группы, которую они представляли ранее, выполняясь на одном элементе). Пример такой разбивки см. на рис. 3 ниже (разносятся работы на элементе №12).

Если состав хотя бы одной общей вершины изменился (т.е. изменена агрегированная сеть второго уровня агрегации), то после окончания распределения работ и проверки всех общих вершин требуется заново решить задачу МВМ (п. 2.5) и выполнить повторное распределение агрегированных работ (п. 3.1). Следует дезагрегировать все общие вершины, для которых выполнены условия дезагрегации, и лишь после этого осуществлять повторное решение задачи и

согласованное планирование, а не после каждой дезагрегации общей вершины, так как это может привести к зацикливанию с постоянной сменой набора общих вершин на каждой итерации.

По причине относительной трудоемкости процедуры распределения, проверку фактического объединения общих вершин можно проводить распределением только агрегированных работ, лежащих на критических путях изделий, без их дополнения остальными агрегированными работами.

Особенности согласованного планирования по разным базовым критериям оптимальности [2].

Критерий 1. Приоритетно-упорядоченная последовательность σ^*, полученная в результате решения задачи МВМ, распределяется по алгоритмам 1 или 2, что позволяет минимизировать суммарное время выполнения каждой из ПМП и, таким образом, обеспечить минимальное время окончания выполнения наиболее выгодных, с точки зрения прибыли, изделий. При такой реализации решения задачи максимизируется суммарная прибыль предприятия по всем изделиям.

Критерий 2. Распределение агрегированных работ последовательности σ^* выполняется по алгоритму 3. При такой реализации алгоритма решения задачи изделия назначаются на выполнение в соответствии с приоритетом подпоследовательности, в состав которой они включены (под приоритетом подпоследовательности понимают отношение суммы весов агрегированных работ, включенных в нее, к сумме их длительностей). Чем выше приоритет подпоследовательности, тем раньше назначаются на выполнение работы, принадлежащие ей (в последовательности σ^* эти агрегированные работы занимают более ранние позиции). Это обеспечивает выполнение в заданный директивный срок в первую очередь изделий с максимальным приоритетом и, следовательно, позволяет максимизировать суммарную прибыль предприятия.

При наличии изделий, которые не могут быть выполнены в срок, расписание передается в блок принятия решений для корректировки. Эти изделия либо исключаются из выполнения, либо принимается решение о приобретении дополнительных ресурсов.

Критерий 3. Процедура распределения является смешанной. В первую очередь по алгоритму 3 распределяются изделия, имеюшие наивысший приоритет. Они назначаются на выполнение на интервалы планового периода, максимально близкие к директивным срокам, без их нарушения, освобождая при этом резервы на более ранних интервалах для остальных изделий. Если выполнение какого-либо изделия не укладывается в его директивный срок, то изделие распределяется по алгоритму 1 или 2 в очередности, определенной последовательностью σ^*.

При таком распределении изделия с наивысшим приоритетом, т. е. наиболее выгодные, выполняются без запаздывания, а для запаздывающих изделий минимизируется значение суммарного штрафа за запаздывание относительно директивных сроков. Таким образом, максимизируется суммарная прибыль предприятия по всему множеству изделий.

Критерий 4. Последовательность σ^* распределяется по алгоритму 3. Если какое-либо изделие не удается распределить без нарушения директивного срока, то расписание передается в блок принятия решений для корректировки.

При таком распределении в первую очередь назначаются на выполнение изделия с наивысшим приоритетом максимально близко к директивным срокам без их нарушения, что позволяет предприятию выполнить наибольшее количество изделий без запаздывания и, следовательно, получить максимальную суммарную прибыль по рассмотренному критерию оптимальности.

Критерий 5. Последовательность σ^* распределяется по алгоритму 3, встраивая изделия так, чтобы момент окончания их выполнения соответствовал директивному сроку d_i или был меньше d_i не более, чем на заданную величину Δ_i. Изделия, которые не удалось распределить без нарушения директивного срока, выносятся в отдельный список, который упорядочивается в соответствии с их директивными сроками и распределяется по алгоритму 1 или 2.

При таком распределении без нарушения директивных сроков (или максимально близко к ним) выполняются наиболее выгодные изделия. Для изделий, выполняемых с запаздыванием, минимизируется отклонение момента окончания выполнения от директивного срока. В результате минимизируется суммарный штраф предприятия за опережение или запаздывание относительно директивных сроков и максимизируется суммарная прибыль предприятия.

3.1.3. После распределения осуществляется проверка совместимости технологии: если некоторые работы (группы работ), которые в агрегированной сети первого уровня агрегации должны были выполняться непрерывно на одном приборе, в результате распределения выполняются в разное время, и разрыв по времени

между такими работами (группами работ) нельзя ликвидировать без перераспределения других агрегированных работ, то требуется изменить агрегированную сеть первого уровня агрегации (и, соответственно, исходную сеть) – определить эти работы на выполнение на этом же приборе, но в разное время. Это не требует повторного решения задачи, поскольку не меняются критические пути изделий, а состав общих вершин уже уточнен при распределении, но новую модель необходимо использовать на следующих уровнях планирования.

Пример. Если работы 332, 334, 435, 436, 437 и 438 выполнялись на одном элементе 12 в исходной и агрегированной сети, но при согласованном планировании выполняются непрерывно только две группы работ – 332 и 334, а также 435 и 436, то эта часть агрегированной сети и, соответственно, исходной сети, изменяется так, как показано на рис. 3.

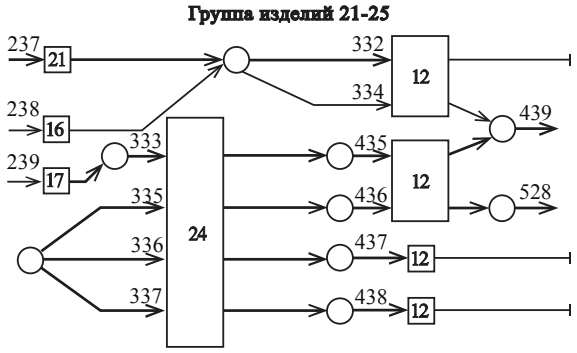


Рис. 3. Изменение агрегированной модели по результатам распределения

При планировании *по синтетическому критерию* согласованное планирование выполняется на подсетях первого уровня агрегации (п. 2.2). N подсетей первого уровня агрегации ($N \in [2, 5]$) для N групп изделий объединяются в одну общую сеть – совокупность подсетей, не имеющих общих связей и общих вершин между подсетями, связанных только общим оборудованием, что дает возможность решать задачу последовательным чередованием групп изделий, чтобы выбрать наилучшее решение (рассматриваются все возможные комбинации очередности последовательного назначения групп изделий). В этом случае процедуры, описанные в п.3.1.1–3.1.3, осуществляются для каждой подсети $N!$ раз в соответствии с различным порядком чередования групп изделий. Распределение следующей группы изделий не начинается, пока не распределена на выполнение предыдущая. В результате получаем $N!$ согласованных планов выполнения агрегированных работ, из которых на следующем этапе будет выбран наилучший.

Для синтетических критериев процедура распределения на выполнение агрегированных работ последовательности σ^*, полученной в результате решения задачи МВМ, является смешанной. В зависимости от заданной базовой составляющей синтетического критерия, для группы изделий применяется один из разработанных алгоритмов. Алгоритмы 1 и 2 используются для распределения групп изделий, оптимизируемых по базовому критерию 1 (выбор алгоритма 1 или 2 делается в соответствии с характеристиками исходных данных), алгоритм 3 применяется для групп изделий с директивными сроками (базовые критерии 2–5).

Резервы в рамках фонда времени прибора определяются имеющимися простоями оборудования, полученными в результате выполнения всех работ предыдущих групп изделий, в соответствии с очередностью их назначения на выполнение, и предыдущих назначенных работ из распределяемой группы изделий, в соответствии с очередностью их назначения в этой группе.

Примечание. Условие, что работы из разных групп изделий могут выполняться на одном элементе в одном временном отрезке, не учитывается при агрегации и согласованном планировании и уточняется лишь при точном планировании (например, если требуется переналадка оборудования, чтобы улучшить план, мы можем объединить работы разных групп изделий, попавшие в результате распределения на один элемент в один временной отрезок).

3.2. После построения всех согласованных планов для каждой возможной очередности назначения групп изделий рассчитывается значение исходного функционала по полученным в результате согласованного планирования моментам окончания выполнения изделий $\mathbf{C}_i$, которые по каждому изделию определяются максимальным из моментов окончания выполнения его конечных работ. Из этих планов выбирается наилучший по значению функционала и, если прибыль меньше ожидаемой минимальной, передается в блок принятия решений для формирования нового портфеля заказов. Если прибыль не меньше ожидаемой минимальной, он передается на следующий уровень планирования.

Блок 4. Построение модели третьего уровня.

В блоке 4 по информации, полученной в блоке 3, формируется задача третьего уровня четырехуровневой модели планирования – многоэтапная сетевая задача календарного планирования.

После утверждения в блоке 3 согласованного плана выполнения агрегированных работ производится дезагрегация агрегированных работ, т.е. возврат к исходной модели сети, с учетом изменений, внесенных при согласованном планировании. Если в принятом на реализацию на третьем уровне плана согласованного планирования

некоторые элементы типа 2–5 оказались разбитыми на части, то в исходной технологической сети каждый такой элемент заменяется аналогичными независимыми элементами того же типа, с соответствующим перераспределением входных и выходных связей между новыми элементами, причем каждая выходящая связь входит в ту же вершину, что и в исходной технологической модели.

Пример. На данном фрагменте исходной сети элемент №23 разбивается на два аналогичных:

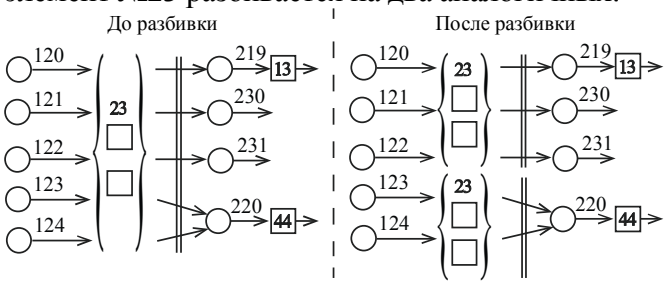


Рис. 4. Разбивка элемента исходной сети

Другой пример приведен выше на рис. 3.

Примечание. Реальное оборудование, реализующее новые виртуальные приборы сети, однозначно определяется только после получения пооперационного плана.

На третьем уровне планирования строится многоэтапная сетевая задача календарного планирования путем задания на измененной исходной сети директивных сроков конечных работ. Моменты окончания выполнения изделий $\mathbf{C}_i$ (моменты C_i окончания выполнения конечных работ каждого изделия), определенные при согласованном планировании, задают реальные директивные сроки конечных работ d_i для следующего уровня планирования (если изделие имеет несколько конечных работ, то из моментов окончания их выполнения выбирается максимальный, и он становится единым директивным сроком для всех конечных работ изделия). Для изделий, планируемых по критерию 5 (минимизация штрафа за опережение/запаздывание относительно директивного срока) выбирается конечная работа с самым поздним моментом окончания выполнения, определенным при согласованном планировании, по сравнению с моментами окончания выполнения остальных конечных работ этого изделия, и он устанавливается как директивный срок, относительно которого определяется штраф за опережение/запаздывание в функционале модели третьего уровня, а для остальных его конечных работ этот момент является директивным сроком, и эти работы могут быть выполнены ранее, без штрафа за опережение. Таким образом, формулируются задачи оптимизации для каждого из элементов типа 2–5: найти допустимое (не нарушающее директивные сроки) расписание выполнения работ на каждом приборе данного элемента, работающем непрерывно, оптимальное по критерию оптимальности (определение 4 ниже).

Определение 3 [1]. *Допустимым* является расписание, в котором моменты окончания выполнения всех конечных работ, не превышают их директивных сроков.

Для сетевой многоэтапной задачи календарного планирования, построенной для базового критерия оптимальности, сформулируем условия оптимальности допустимого решения.

Определение 4 [1]. Расписание является *оптимальным*, если оно отвечает следующим условиям:

1) расписание допустимое;

2) из всех допустимых расписаний оптимальным может быть только то, в котором достигается минимум суммарного опережения относительно конечных директивных сроков;

3) а) оптимальным является допустимое расписание, удовлетворяющее пунктам 1) и 2), в котором на множестве начальных работ достигается максимум:

$$\max\left(\min_j r_{1j}\right),\ j=\overline{1,p},$$

где $r_{1j} \geq 0$, где r_{1j} – моменты запуска начальных работ сети (первых работ на приборах $1j$), а p – число таких приборов;

б) оптимальным является допустимое расписание, удовлетворяющее пунктам 1) и 2), в котором на множестве начальных работ достигается максимум:

$$\max\left(\min_j r_{1j}\right)=r_{1j_1},\ j=\overline{1,p}$$

$$\max\left(\min_j r_{1j}\right)=r_{1j_l},\ j=\overline{1,p},\ l=\overline{2,p},\ j\neq j_k,\ k=\overline{1,l-1}.$$

Методология решения сформулированной многоэтапной сетевой задачи календарного планирования по *базовому критерию* оптимальности представлена в [1] и заключается в том, что решение ищется с конца сети к началу, начиная с элементов последнего уровня входимости. Исходная сеть при планировании по базовому критерию удовлетворяет свойствам, описанным в [1].

При планировании по *синтетическому критерию* оптимальности, в силу того, что исходная сеть представляет собой совокупность независимых подсетей, каждая из которых удовлетворяет свойствам, описанным в [1], и в разных подсетях может использоваться общее оборудование, метод решения задачи с конца сети к началу не может быть использован, т.к. расписание, полученное в одной подсети, может не поз-

волить получить расписание в другой подсети, если оборудование будет уже занято расписанием выполнения предыдущей группы изделий. Поэтому предлагается следующий альтернативный метод решения сетевой многоэтапной задачи календарного планирования и построения пооперационного плана. Критерием оптимальности для планирования по синтетическому критерию является получение пооперационного плана, максимально близкого к плану, полученному на этапе согласованного планирования. Поскольку метод универсален, его можно применять и для базового критерия оптимальности для любой сети виртуальных приборов.

В альтернативном методе директивные сроки работ на элементах исходной сети задаются моментами окончания агрегированных работ, в которые эти работы вошли в результате агрегации, в утвержденном согласованном плане второго уровня. Моменты запуска агрегированных работ в согласованном плане определяют минимальные моменты запуска работ на приборах. Благодаря согласованному планированию работы, выполняемые на элементах с одинаковым номером, разнесены по времени. Многоэтапная сетевая задача календарного планирования решается с начала в конец, начиная с элементов первого уровня входимости, в произвольном порядке рассмотрения элементов сети. В предположении, что для каждой задачи агрегированная длительность соответствует реальной (не меньше суммарной дезагрегированной), для получения пооперационного плана решаются одноэтапные задачи календарного планирования для каждого из элементов типа 2–5: найти допустимое (не нарушающее директивные сроки) расписание выполнения работ, оптимальное по следующему критерию оптимальности: для элементов типа 2, 3: $\max(\min r)$, где r – момент запуска приборов; для элементов типа 4, 5 – см. п. а), б) по определению 4.

Если из-за того, что агрегированная длительность оказывается меньше суммарной дезагрегированной, работы претендуют на один и тот же временной интервал, план должен быть пересчитан, начиная с работ первого уровня входимости, состыковкой задач на элементах по принципу разнесения по времени, с выполнением условия: максимальный момент окончания выполнения работ на элементе-предшественнике равен минимальному моменту запуска работ на выполнение на элементе-преемнике (или хотя бы не меньше его). Если это условие не выполняется, производится сдвиг интервала выполнения работ на элементе-преемнике в сторону увеличения минимального момента запуска и директивных сроков. В полученном пооперационном плане анализируются моменты окончания последних работ изделий на элементах последнего уровня входимости.

Если полученное решение этой задачи допустимое (директивные сроки, установленные заказчиком, не нарушены), то для данного портфеля заказов получен пооперационный план.

Если полученное решение оказалось недопустимым (моменты выполнения некоторых конечных работ больше директивных сроков, установленных заказчиками) либо становится таковым в силу наличия отрицательных моментов начала выполнения некоторых работ, то полученный пооперационный план пересчитывается путем смещения отрицательных моментов начала выполнения начальных работ технологической сети к началу планового периода. При этом порядок выполнения всех работ сохраняется, а моменты запуска элементов сети типа 2–5 сдвигаются минимально для выполнения условия непрерывности их работы. Для этого расписания заново рассчитывается прибыль по исходному функционалу. Если новая рассчитанная прибыль не меньше ожидаемой минимальной, то необходимо перейти на блок принятия решений для согласования новых директивных сроков с заказчиком. Если прибыль меньше минимальной ожидаемой, то в случае планирования по базовому критерию оптимальности реализуем п. 9) блока принятия решений. Если критерий синтетический, то непосредственно в блоке принятия решений изменяется портфель заказов. Если в результате реализации п. 9) блока принятия решений новая прибыль меньше минимальной ожидаемой, то в блоке принятия решений изменяется портфель заказов.

Реализация трехуровневой модели планирования статистически значима и практически всегда эффективна, если критерий планирования базовый (а для синтетического критерия – в предположении, что эвристика последовательного назначения групп изделий соответствует реальному производственному процессу). Тогда, благодаря решению задачи МВМ и согласованному планированию получение сроков выполнения изделий становится формализованным и близким к оптимальному, в отличие от неформализованной процедуры при реализации п. 9) в блоке принятия решений. И только в случае, когда эвристика назначения изделий группами не адекватна реальному производственному процессу, реализация п. 9) становится необходимой.

Обоснование эвристики согласованного планирования по базовому критерию. Согласованное планирование выполняется в соответствии с последовательностью агрегированных работ, полученной в результате решения аппроксимирующей задачи МВМ. Такая очередность назначений работ на выполнение максимизирует соответствующую составляющую синтетического функционала с точностью до адекватности его значения функционалу аппроксимирующей задачи МВМ.

Обоснование эвристики согласованного планирования по синтетическому критерию. Согласованное планирование реализуется через последовательное назначение на выполнение групп изделий (каждая из которых определяется соответствующей составляющей синтетического функционала), рассматриваются все возможные комбинации очередности назначения групп изделий. В каждой группе изделий очередность назначений изделий на выполнение максимизирует соответствующую составляющую синтетического функционала с точностью до адекватности его значения функционалу аппроксимирующей задачи МВМ. Эта эвристика позволила исходную задачу планирования по 31 критерию свести к решению одной многоэтапной сетевой задачи календарного планирования (каждому из решений по 31 критерию соответствуют свои директивные сроки выполнения конечных работ). Эта эвристика эффективна, если она адекватна реальному технологическому процессу.

На третьем уровне планирования могут быть реализованы альтернативные планы выполнения различных портфелей заказов. Каждый план имеет свою степень риска неполучения расчетной прибыли после его реализации. Поэтому в блоке принятия решений осуществляется выбор наилучшего плана по критерию минимизации риска неполучения расчетной прибыли [2] из множества альтернативных планов.

Блок 5. Оперативное планирование.

На четвертом уровне модели реализуется оперативное планирование в случае частичного невыполнения плана, полученного в блоке 4. Во всех моделях четвертого уровня минимизируются аддитивные функционалы, компонентами которых являются новые весовые коэффициенты в функционале качества – штрафы за запаздывание либо за опережение/запаздывание относительно директивных сроков (базовые критерии 3 и 5), – независимо от ранее используемых критериев оптимальности, какими бы они ни были.

Теперь покажем реализацию алгоритмического обеспечения более подробно на примере.

Пример реализации методологии построения модели

2.1. Постановка задачи. Пример сетевой модели строится на основе технологии реального производства мелкосерийного типа. В нашем примере будет синтетический критерий оптимальности, состоящий из $N = 5$ базовых критериев, поэтому имеется 5 независимых групп изделий, т.е. исходная сеть имеет 5 подсетей, не связанных технологическими связями. В каждой подсети выполняется по 5 изделий.

На рис. 5 и 6 показана исходная сеть с фиктивными работами (технологическая сеть, задающая технологию выполнения изделий). Для упрощения схемы вместо параметров невыполненных работ над стрелками будем писать номера работ и над приборами – их номера, а характеристики работ и элементов сведём, соответственно, в таблицы 3 и 4. Удельная длительность l_i выполнения работы на одно изделие задается в минутах.

Табл. 3. Характеристики работ

№(i)	l_i	№(i)	l_i	№(i)	l_i	№(i)	l_i
11	1	**29**	11	**321**	3	**435**	4
12	4	**210**	5	**322**	5	**436**	1
13	2	**211**	2	**323**	2	**437**	1
14	5	**212**	6	**324**	6	**438**	5
15	15	**213**	1	**325**	12	**439**	14
16	1	**214**	7	**326**	12	**440**	2
17	8	**215**	3	**327**	11	**51**	6
18	3	**216**	8	**328**	5	**52**	2
19	3	**217**	3	**329**	6	**53**	6
110	1	**218**	6	**330**	4	**54**	7
111	4	**219**	3	**331**	4	**55**	3
112	4	**220**	8	**332**	1	**56**	6
113	14	**221**	6	**333**	1	**57**	9
114	2	**222**	8	**334**	4	**58**	3
115	5	**223**	2	**335**	6	**59**	5
116	4	**224**	4	**336**	3	**510**	8
117	6	**225**	7	**337**	7	**511**	8
118	8	**226**	7	**338**	2	**512**	4
119	3	**227**	2	**41**	7	**513**	6
120	8	**228**	1	**42**	15	**514**	3
121	10	**229**	9	**43**	2	**515**	4
122	4	**230**	3	**44**	15	**516**	6
123	5	**231**	15	**45**	3	**517**	1
124	23	**232**	2	**46**	3	**518**	8
125	3	**233**	6	**47**	6	**519**	8
126	12	**234**	9	**48**	1	**520**	5
127	4	**235**	12	**49**	5	**521**	9
128	4	**236**	3	**410**	9	**522**	4
129	1	**237**	7	**411**	3	**523**	5
130	15	**238**	2	**412**	6	**524**	7
131	11	**239**	4	**413**	4	**525**	5
132	6	**30**	7	**414**	6	**526**	3
133	10	**31**	9	**415**	12	**527**	4
134	7	**32**	12	**416**	15	**528**	5
135	4	**33**	18	**417**	4	**61**	11
136	2	**34**	2	**418**	22	**62**	12

№(i)	l_i	№(i)	l_i	№(i)	l_i	№(i)	l_i
137	2	**35**	4	**419**	4	**63**	15
138	4	**36**	7	**420**	2	**64**	11
139	4	**37**	4	**421**	6	**65**	15
140	6	**38**	16	**422**	8	**66**	17
141	8	**39**	9	**423**	4	**67**	19
142	3	**310**	6	**424**	9	**68**	12
143	1	**311**	8	**425**	2	**69**	4
144	3	**312**	23	**426**	4	**610**	12
21	4	**313**	6	**427**	7	**611**	15
22	2	**314**	4	**428**	9	**612**	21
23	1	**315**	3	**429**	13	**613**	20
24	6	**316**	1	**430**	6	**614**	14
25	8	**317**	1	**431**	8	**615**	7
26	8	**318**	5	**432**	3	**616**	21
27	12	**319**	4	**433**	9	**71**	24
28	20	**320**	7	**434**	3	**72**	23

Табл. 4. Характеристики ресурсов

№	тип	m	№	тип	m	№	тип	m	№	тип	m
11	2	1	**18**	1	1	**25**	1	1	**41**	4	2
12	2	1	**19**	1	1	**26**	1	1	**42**	1	1
13	1	1	**21**	1	1	**31**	2	1	**43**	1	1
14	1	1	**22**	5	3	**32**	4	3	**44**	1	1
15	1	1	**23**	3	2	**33**	2	1	**51**	1	1
16	1	1	**24**	2	1	**34**	1	1	**52**	1	1
17	1	1									

Для элемента типа 5 (в данном случае это единственный элемент №22) зададим производительности приборов s_i ($i = \overline{1,m}$): 1; 0,8; 0,2.

Характеристики изделий зададим таблицей 5 (N_i – число изделий в портфеле заказов, ω_i – вес, d_i – директивный срок, Δ_i – диапазон получения допустимого решения – задается для тех изделий, в технологии которых встречаются элементы типа 4 и 5, j_{end} – номера конечных работ изделия).

Момент окончания выполнения изделия определяется максимальным из моментов окончания выполнения его конечных работ. Например, для 1-го изделия $\mathbf{C}_1 = \max\{C_{61}, C_{62}\}$.

Для однозначного определения критических путей изделий необходимо знать, к какому изделию относится та или иная работа из множества выполненных работ каждого элемента типа 1. Поэтому для элементов, где такое соответствие неоднозначно, зададим его таблицей 6. Для остальных работ их принадлежность конкретному изделию очевидным образом определяется по исходной сети.

Табл. 5. Характеристики изделий

№(i)	j_{end}	N_i	ω_i	d_i	Δ_i	№(i)	j_{end}	N_i	ω_i	d_i	Δ_i
1	61, 62	2	1			**14**	519, 521, 526	10	80	1200	5
2	53	4	2								
3	63, 64	3	3			**15**	426–428	15	100	1250	10
4	71	5	4			**16**	429, 613	5	200	1220	
5	57–59, 510	10	5			**17**	431	10	400	1110	
6	66	15	200	735	5	**18**	614	15	600	850	5
7	67	5	400	550	10	**19**	615	5	800	780	10
8	72	10	600	830	15	**20**	616	10	1000	810	15
9	413	15	800	435		**21**	332	15	20	1400	
10	69, 611	5	1000	920		**22**	439	5	40	1475	
11	612	10	20	1150	5	**23**	528	10	60	1350	
12	425, 518	15	40	1050	10	**24**	437	15	80	1275	
13	421	5	60	875	15	**25**	438	5	100	1250	

В качестве критерия оптимальности зададим самый сложный (31-й) из синтетических критериев, являющийся комбинацией пяти базовых критериев, в каждой группе из пяти изделий используется один базовый критерий, а именно, для группы изделий 1–5 – базовый критерий 1, для группы изделий 6–10 – базовый критерий 2, для группы изделий 11–15 – базовый критерий 3, для группы изделий 15–20 – базовый критерий 4, для группы изделий 21–25 – базовый критерий 5:

$$\max\left\{\sum_{i=1}^{5}\omega_i(T)\cdot(T-C_i)+\sum_{i=6}^{10}\omega_i U_i-\sum_{i=11}^{15}\omega_i'\max(0,\,C_i-d_i)+\sum_{i=16}^{20}\omega'' V_i-\sum_{i=21}^{25}\omega_i'''|C_i-d_i|\right\}, \qquad (2)$$

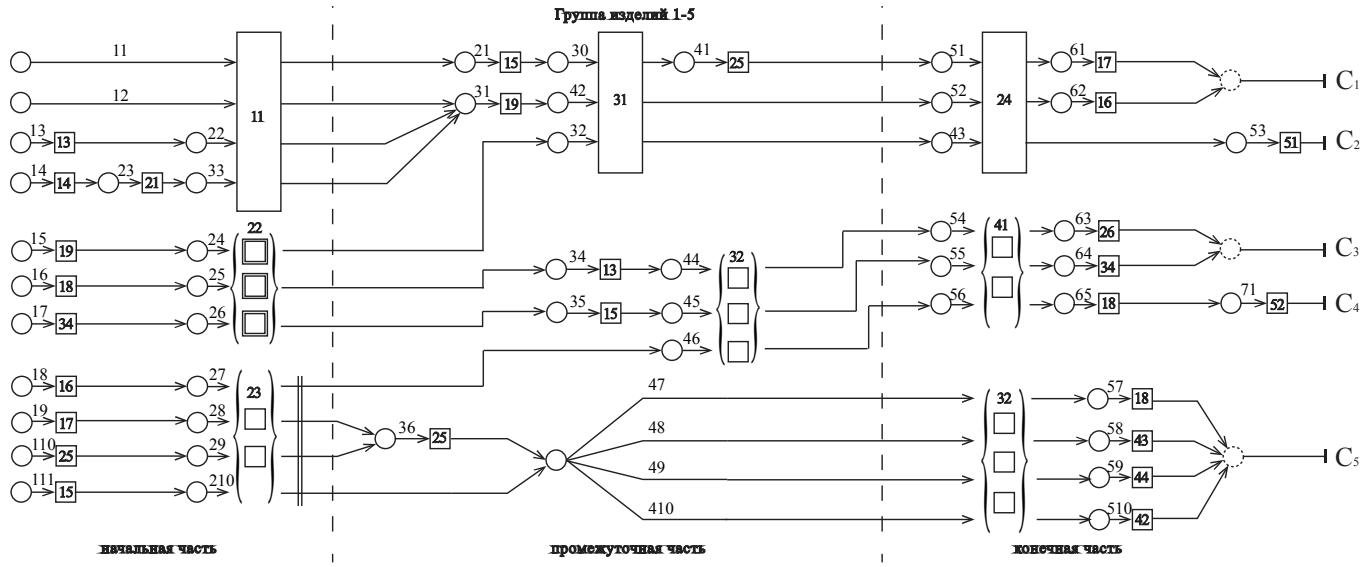


Рис. 5. Исходная сеть (изделия 1–5)

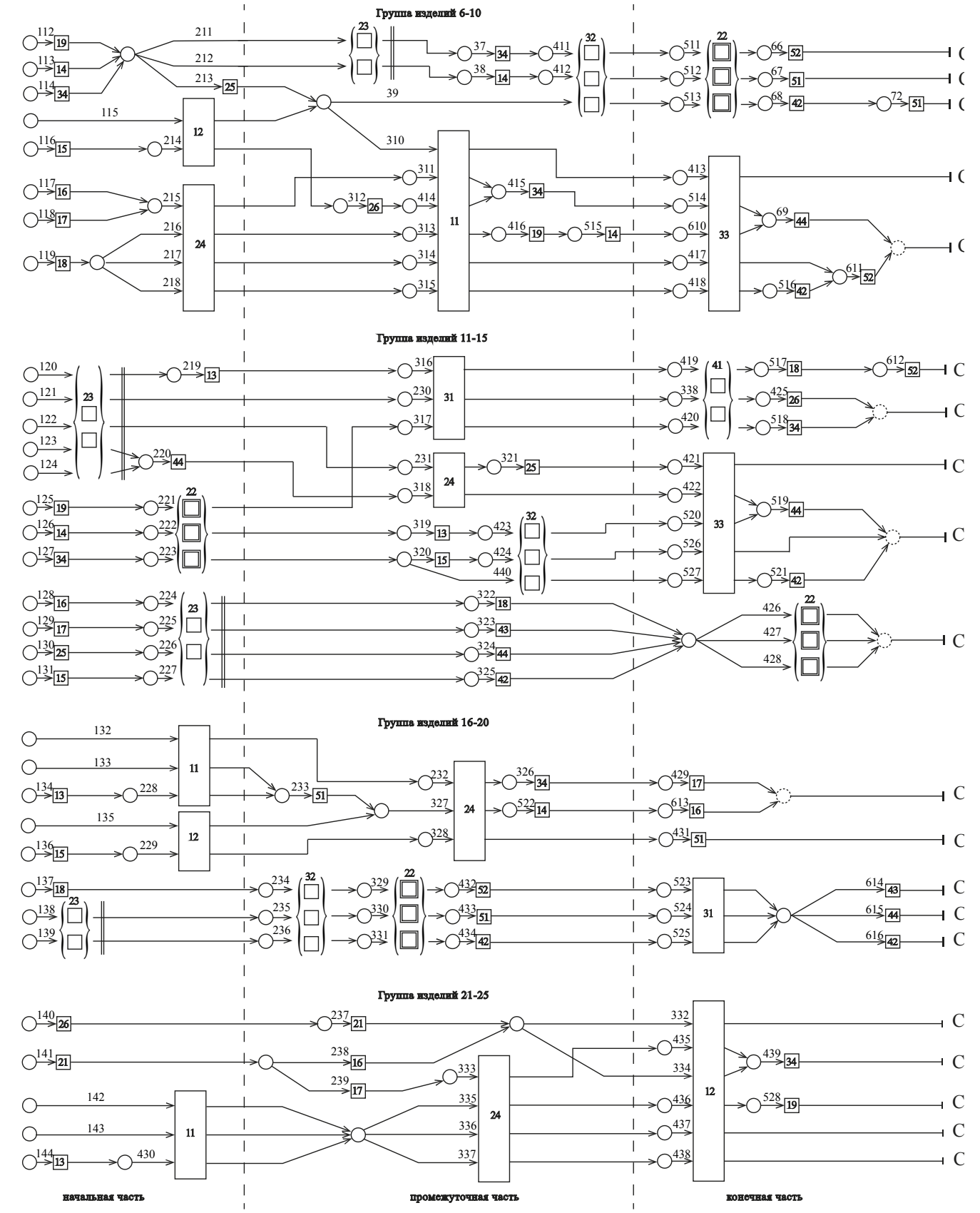


Рис. 6. Исходная сеть (изделия 10–25)

где $U_i = \begin{cases} 1, C_i = d_i \\ 0, C_i \neq d_i \end{cases}$, $V_i = \begin{cases} 1, C_i \leq d_i \\ 0, C_i > d_i \end{cases}$,

$\omega_i(T)$ – весовой коэффициент изделия i, определенный экспериментальным путем; T – плановый период; ω_i – прибыль от выполнения i-го изделия, если оно выполнено точно в срок, ω_i' – штраф за запаздывание окончания выполнения i-го изделия относительно директивного срока на единицу времени, ω_i'' – прибыль от выполнения i-го изделия, если оно выполнено без запаздывания, а ω_i''' – штраф за отклонение момента окончания выполнения i-го изделия от директивного срока на единицу времени.

Табл. 6. Соответствия работ изделиям по элементам типа 1

Номер работы	Номер изделия	Номер работы	Номер изделия	Номер работы	Номер изделия
112	6	**137**	18	**238**	22
113	7	**142**	23	**430**	25
114	8	**143**	24	**523**	18
115	9	**213**	8	**524**	19
136	17	**237**	21	**525**	20

Зададим минимальную ожидаемую прибыль предприятия (минимальное значение критерия (1), которое мы ожидаем получить при решении задачи) на уровне 20000 условных единиц (единица измерения прибыли – та же, что и для весовых коэффициентов $\omega_i(T)$, ω_i, ω_i', ω_i'', ω_i''', с учетом их умножения на единицу времени).

2.2. *Первый уровень агрегации* является агрегированным представлением исходной сети.

2.2.1. Агрегированная сеть первого уровня агрегации приведена на рис. 7. Для удобства, нумерация работ и приборов оставлена прежней (в качестве номера агрегированной работы используется первый из номеров объединяемых работ).

Удельные длительности агрегированных (объединенных) работ, рассчитанные по вышеприведенным правилам, сведём в таблицу 7. Например, удельные длительности работ 11, 12, 22, 33 (все работы – изделия 1), выполняемых на одном элементе типа 2, суммируются, в результате получается работа 11 с удельной длительностью 25.

Табл. 7. Агрегированные работы

№(i)	Объединены №	l_i	№(i)	Объединены №	l_i
11	11, 12, 22, 33	25	**222**	222, 223	5
25	25, 26	8	**224**	224, 225, 226, 227	10
28	28, 29, 210	18	**232**	232, 327	13
30	30, 42	22	**311**	311, 313, 314, 315, 414	27
44	44, 45	6			
47	47, 48, 49, 410	7	**338**	338, 420	2
51	51, 52	8	**417**	417, 418, 514, 610	41
54	54, 55	5	**422**	422, 520, 526, 527	20
123	123, 124	14	**423**	423, 424, 440	5
132	132, 133, 228	17	**426**	426, 427, 428	10
215	215, 216, 217, 218	20			

В результате агрегации некоторые конечные работы изделий объединены в одну конечную работу: по изделию 14 конечная работа 526 вошла в состав агрегированной работы 422, а по изделию 15 конечные работы 426–428 объединились в одну агрегированную работу 426.

2.2.2. Также в связи с тем, что элементы 3–5 агрегируются элементы 1–2, меняется длительность работ, оставшихся на элементах и не объединенных с другими работами (табл. 8). Например, у работы 46 (длительность 3), выполняемой на элементе 32 (тип 4, количество приборов $m = 3$), новая длительность равна 3 / 3 = 1.

Табл. 8. Измененные длительности

№(i)	l_i	№(i)	l_i	№(i)	l_i	№(i)	l_i	№(i)	l_i
24	3	**120**	4	**211**	1	**236**	1	**412**	2
27	6	**121**	5	**212**	3	**329**	3	**419**	2
39	3	**122**	2	**221**	3	**330**	2	**511**	4
46	1	**138**	2	**234**	3	**331**	2	**512**	2
56	3	**139**	2	**235**	4	**411**	1	**513**	3

2.3. *Поиск критических путей изделий.* Критические пути изделий показаны жирными линиями на рис. 7.

2.4. *Второй уровень агрегации* – построение сети на критических путях с общими вершинами.

2.4.1. *Поиск общих вершин на критических путях изделий.* В нашем примере введем $\Delta_{ОВ} = 70$, что составляет около 10% от максимальной длительности критического пути ($L_{кп} = 710$ у изделия 10), и будем использовать второе правило объединения в общие вершины. В результате все общие вершины, заданные технологией, объединились, кроме указанных в примечании на стр. 65.

Построенная сеть второго уровня агрегации показана на рис. 8, общие вершины отображены символом ◎. В табл. 9 представлены их длительности (уже учитывающие количество изделий в портфеле заказов N_i). Номером общей вершины (для упрощения) считаем номер первой из объединяемых работ.

Например, в общую вершину 24 объединены агрегированные работы 24 (изделие 2) и 25 (изделие 3). Число изделий 2 в портфеле заказов равно $N_2 = 4$, $N_3 = 3$. Удельные длительности агрегированных работ 24 и 25, соответственно, равны $l_{24} = 3$, $l_{25} = 8$. Поэтому длительность общей вершины 24 составляет $l_{24}N_2 + l_{25}N_3 = 3\cdot4 + 8\cdot3 = 36$.

Табл. 9. Общие вершины

№(i)	Объединены №	l_i	№(i)	Объединены №	l_i
24	24, 25	36	**230**	230, 316	55
27	27, 28	210	**232**	232, 328	115
30	30, 32	92	**234**	234, 235, 236	75
44	44, 46	23	**329**	329, 330, 331	75
51	51, 43	24	**333**	333, 335, 336, 337	145
54	54, 56	30	**338**	338, 419	50
120	120, 121, 122	125	**411**	411, 412	25
138	138, 139	30	**435**	435, 436	30
142	142, 143, 430	75	**511**	511, 512, 513	100
211	211, 212	30	**523**	523, 524, 525	160

2.4.2. При *построении сети на критических путях изделий* все критические пути объединяются в одну сеть.

Так как не все конечные работы изделий лежат на их критических путях, то некоторые из них удалены при построении сети на критических путях: по изделию 1 осталась работа 62 (работа 61 удалена из списка конечных вершин), по изделию 3 осталась работа 63 (работа 64 удалена), по

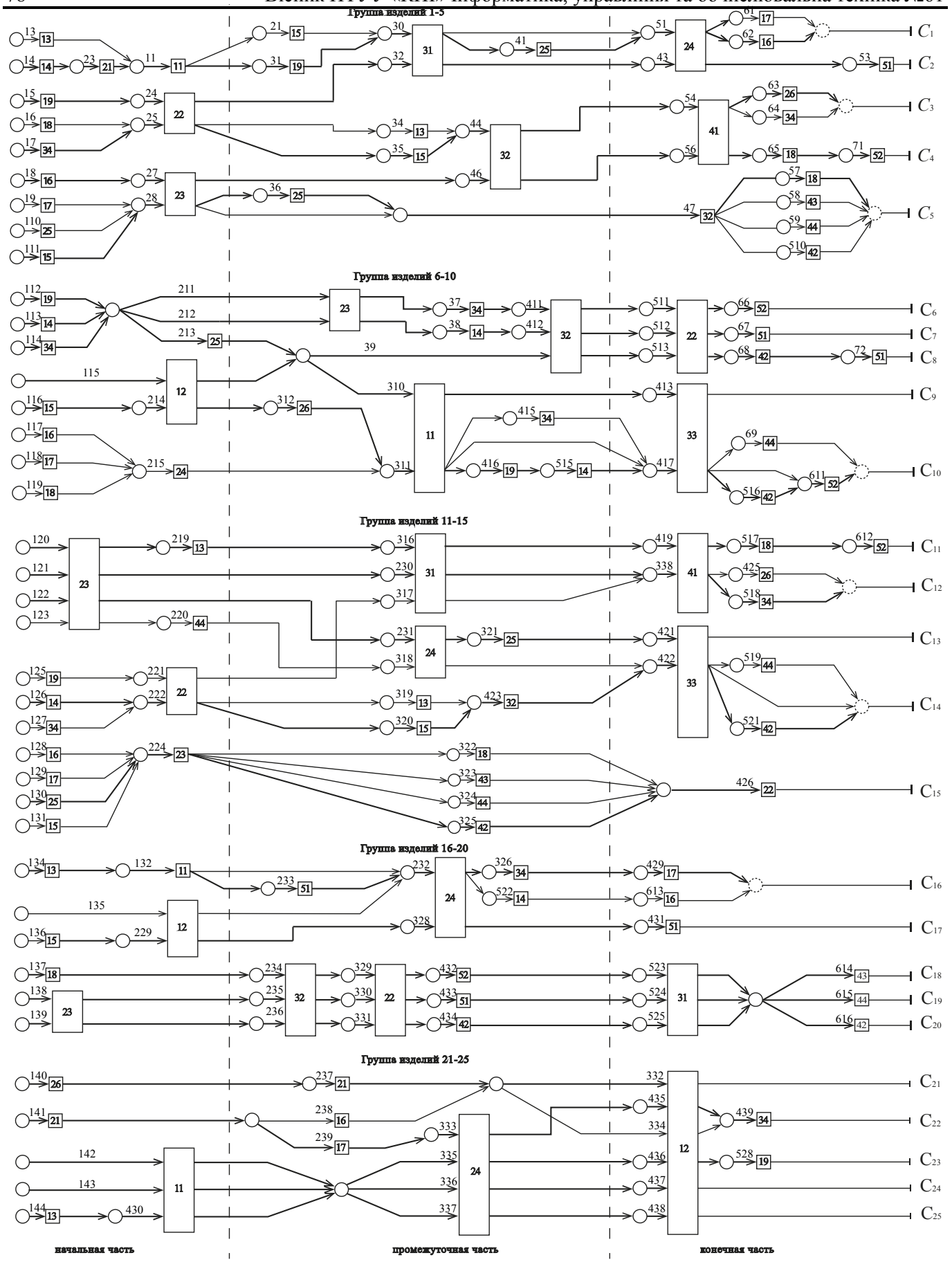


Рис. 7. Агрегированная сеть первого уровня агрегации с критическими путями (изделия 1–25)

изделию 5 осталась работа 57 (работы 58, 59, 510 удалены), по изделию 10 осталась работа 611 (работа 69 удалена), по изделию 12 осталась работа 518 (работа 425 удалена), по изделию 14 осталась работа 521 (работы 422, 519 удалены), по изделию 16 осталась работа 429 (работа 613 удалена).

2.5. Построение и решение оптимизационных задач по критерию МВМ на агрегированной сети второго уровня агрегации (рис. 8).

2.5.1. Для каждой базовой составляющей синтетического критерия, входящих в заданный функционал, кроме базового критерия 1, строится и решается аппроксимирующая задача МВМ, для базового критерия 1 решается задача МВМ без изменения параметров. Задача решается с помощью алгоритма, описанного в [2, гл. 9].

Для решения задачи МВМ по каждой группе изделий строится начальная допустимая последовательность, включающая работы сети на критических путях с учетом заданных отношений предшествования. Изделия в начальных последовательностях упорядочиваются по приоритетам, равным отношению веса изделия к длине его критического пути. Начальные последовательности сведем в табл. 10.

2.5.2. В результате решения задачи МВМ по каждой группе изделий получены последовательности выполнения агрегированных работ сети на критических путях изделий (табл. 11).

Блок 3. Построение модели второго уровня.

3.1. *Построение согласованного плана* выполнения агрегированных работ с учетом критерия оптимальности каждой группы изделий.

3.1.1. В табл. 12 сведены дополненные оптимальные последовательности σ^* по всем группам изделий, т.е. дается информация о номерах ПМП и длительностях агрегированных работ.

3.1.2. На этапе согласованного планирования рассматриваются все возможные комбинации очередности назначения групп изделий. Мы приведем пример распределения агрегированных работ по приборам для следующих упорядочений групп изделий:

	Группы изделий				
Порядок 1	1–5	6–10	11–15	16–20	21–25
Порядок 2	6–10	11–15	16–20	21–25	1–5

Порядок 1 соответствует случаю распределения сначала изделий, выполняемых без ограничений по директивному сроку. Порядок 2 применяется, если более важными считаются изделия, по которым должен быть соблюден директивный срок.

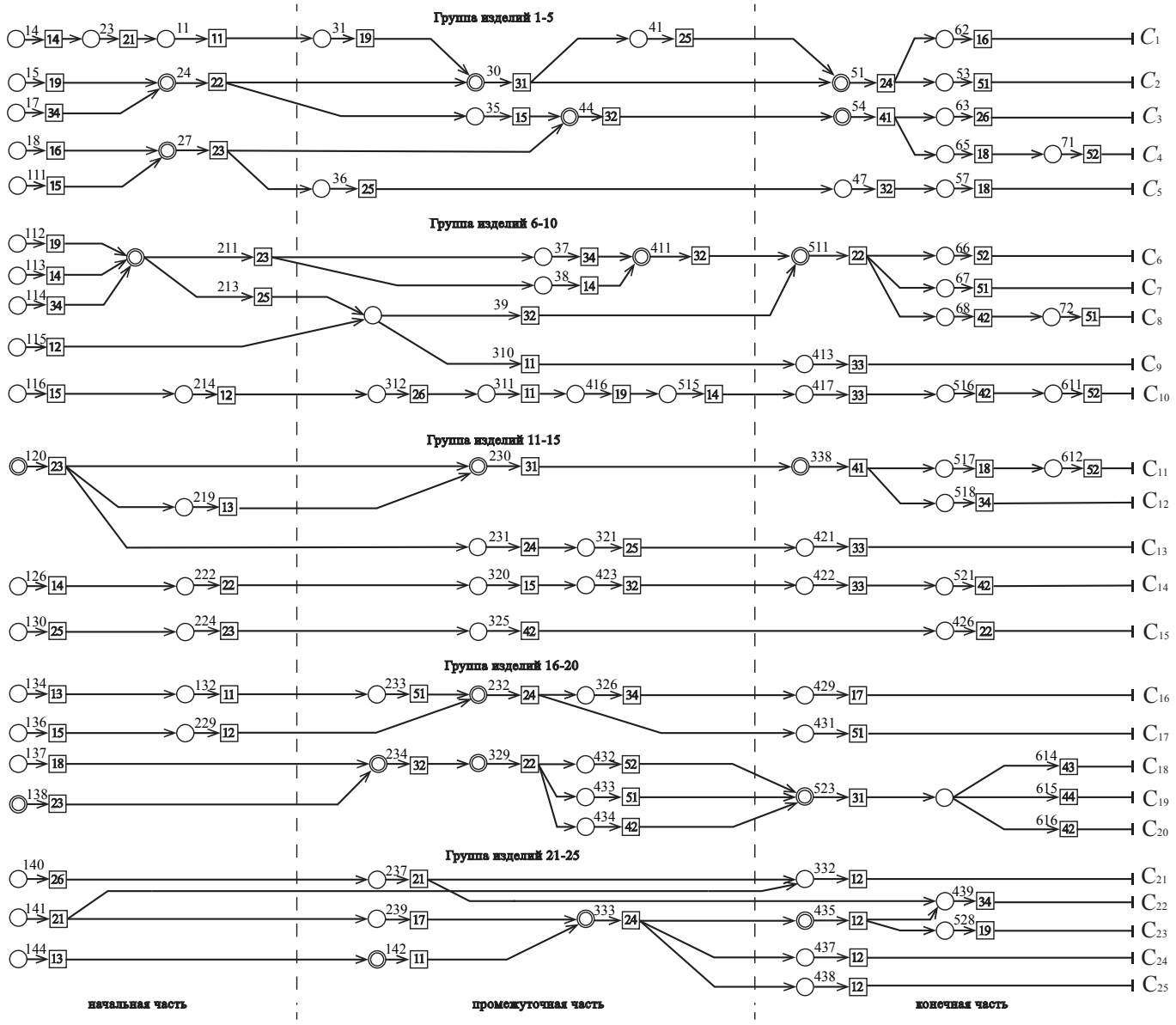


Рис. 8. Сеть на критических путях изделий с общими вершинами

Табл. 10. Начальные последовательности

Группа изделий 1–5:

№(i)	l_i	ω_i	C_i	f_i
15	60		60	
17	24		84	
24	36		120	
35	12		132	
18	15		147	
111	40		187	
27	210		397	
44	23		420	
54	30		450	
63	45	3	495	1485
65	75		570	
71	120	4	690	2760
14	10		700	
23	2		702	
11	50		752	
31	18		770	
30	92		862	
41	14		876	
51	24		900	
53	24	2	924	1848
36	70		994	
47	70		1064	
57	90	5	1154	5770
62	24	1	1178	1178
Функционал: 13041				

Группа изделий 6–10:

№(i)	l_i	ω_i^*	C_i	f_i
112	60		60	
113	70		130	
114	20		150	
213	10		160	
115	75		235	
310	90		325	
413	60	4	385	1540
211	30		415	
38	80		495	
37	60		555	
411	25		580	
39	30		610	
511	100		710	
67	95	2	805	1610
116	20		825	
214	35		860	
312	115		975	
311	135		1110	
416	75		1185	
515	20		1205	
417	205		1410	
516	30		1440	
611	75	5	1515	7575
68	120		1635	
72	230	3	1865	5595
66	255	1	2120	2120
Функционал: 18440				

Группа изделий 11–15:

№(i)	l_i	ω_i^*	C_i	f_i
120	125		125	
231	75		200	
321	15		215	
421	30	3	245	735
219	30		275	
230	55		330	
338	50		380	
518	120	2	500	1000
130	225		725	
224	150		875	
325	180		1055	
426	150	5	1205	6025
126	120		1325	
222	50		1375	
320	70		1445	
423	50		1495	
422	200		1695	
521	90	4	1785	7140
517	10		1795	
612	210	1	2005	2005
Функционал: 16905				

Группа изделий 16–20:

№(i)	l_i	ω_i^*	C_i	f_i
138	30		30	
137	30		60	
234	75		135	
329	75		210	
433	45		255	
434	30		285	
432	45		330	
523	160		490	
615	35	4	525	2100
616	210	5	735	3675
136	20		755	
229	90		845	
134	35		880	
132	85		965	
233	30		995	
232	115		1110	
431	80	2	1190	2380
614	210	3	1400	4200
326	60		1460	
429	65	1	1525	1525
Функционал: 13880				

Группа изделий 21–25:

№(i)	l_i	ω_i^*	C_i	f_i
141	40		40	
239	20		60	
144	15		75	
142	75		150	
333	145		295	
437	15	4	310	1240
438	25	5	335	1675

№(i)	l_i	ω_i^*	C_i	f_i
435	30		365	
528	50	3	415	1245
140	90		505	
237	105		610	
439	70	2	680	1360
332	15	1	695	695
Функционал: 6215				

Табл. 11. Оптимальные последовательности

Группа изделий 1–5:

i	ПМП	l_i	ω_i	C_i	f_i
18	1	15		15	
111	1	40		55	
27	1	210		265	
36	1	70		335	
47	1	70		405	
57	1	90	5	495	2475
15	1	60		555	
17	1	24		579	
24	1	36		615	
35	1	12		627	
44	1	23		650	
54	1	30		680	
63	1	45	3	725	2175
65	1	75		800	
71	1	120	4	920	3680
14	2	10		930	
23	2	2		932	
11	2	50		982	
31	2	18		1000	
30	2	92		1092	
41	2	14		1106	
51	2	24		1130	
53	2	24	2	1154	2308
62	2	24	1	1178	1178
Функционал: 11816					

Группа изделий 6–10:

i	ПМП	l_i	ω_i^*	C_i	f_i
112	1	60		60	
113	1	70		130	
114	1	20		150	
213	1	10		160	
115	1	75		235	
310	1	90		325	
413	1	60	4	385	1540
116	2	20		405	
214	2	35		440	
312	2	115		555	
311	2	135		690	
416	2	75		765	
515	2	20		785	
417	2	205		990	
516	2	30		1020	
611	2	75	5	1095	5475
211	3	30		1125	
38	3	80		1205	
37	3	60		1265	
411	3	25		1290	
39	3	30		1320	
511	3	100		1420	
67	3	95	2	1515	3030
68	3	120		1635	
72	3	230	3	1865	5595
66	4	255	1	2120	2120
Функционал: 17760					

Группа изделий 11–15:

i	ПМП	l_i	ω_i^*	C_i	f_i
120	1	125		125	
231	1	75		200	
321	1	15		215	
421	1	30	3	245	735
219	2	30		275	
230	2	55		330	
338	2	50		380	
518	2	120	2	500	1000
130	3	225		725	
224	3	150		875	
325	3	180		1055	
426	3	150	5	1205	6025
126	4	120		1325	
222	4	50		1375	
320	4	70		1445	
423	4	50		1495	
422	4	200		1695	
521	4	90	4	1785	7140
517	5	10		1795	
612	5	210	1	2005	2005
Функционал: 16905					

Группа изделий 16–20:

i	ПМП	l_i	ω_i^*	C_i	f_i
138	1	30		30	
137	1	30		60	
234	1	75		135	
329	1	75		210	
433	1	45		255	
434	1	30		285	
432	1	45		330	
523	1	160		490	
615	1	35	4	525	2100
616	1	210	5	735	3675
614	1	210	3	945	2835
136	2	20		965	
229	2	90		1055	
134	2	35		1090	
132	2	85		1175	
233	2	30		1205	
232	2	115		1320	
431	2	80	2	1400	2800
326	2	60		1460	
429	2	65	1	1525	1525
Функционал: 12935					

Группа изделий 21–25:

i	ПМП	l_i	ω_i^*	C_i	f_i
141	1	40		40	
239	1	20		60	
144	1	15		75	
142	1	75		150	
333	1	145		295	
437	1	15	4	310	1240
438	1	25	5	335	1675

i	ПМП	l_i	ω_i^*	C_i	f_i
435	1	30		365	
528	1	50	3	415	1245
140	2	90		505	
237	2	105		610	
439	2	70	2	680	1360
332	2	15	1	695	695
Функционал: 6215					

* Для упрощения расчетов при решении задачи МВМ, в группах изделий 6–10 и 16–20 веса изделий пропорционально уменьшены в 200 раз, а в группах изделий 11–15 и 21–25 – в 20 раз, что влияет только на значение функционала, но не влияет на оптимальную последовательность.

Табл. 12. Последовательности σ^* агрегированных работ для согласованного планирования

Изделия 1–5:

№(i)	ПМП	l_i
18	1	15
111	1	40
19	1	30
110	1	10
27	1	30
28	1	180
36	1	70
47	1	70
58	1	30
59	1	50
510	1	80
57	1	90
15	1	60
17	1	24
16	1	3
24	1	12
25	1	24
35	1	12
34	1	6
44	1	18
46	1	5
54	1	15
56	1	15
64	1	33
63	1	45
65	1	75
71	1	120
14	2	10
23	2	2
13	2	4

№(i)	ПМП	l_i
11	2	50
31	2	18
21	2	8
30	2	44
32	2	48
41	2	14
51	2	12
43	2	8
53	2	24
61	2	22
62	2	24

Изделия 6–10:

№(i)	ПМП	l_i
112	1	60
113	1	70
114	1	20
213	1	10
115	1	75
310	1	90
413	1	60
116	2	20
214	2	35
312	2	115
117	2	30
118	2	40
119	2	15
215	2	100
311	2	135
416	2	75
515	2	20

№(i)	ПМП	l_i
415	2	60
417	2	205
516	2	30
69	2	20
611	2	75
211	3	15
212	3	15
38	3	80
37	3	60
411	3	15
412	3	10
39	3	30
511	3	60
512	3	10
513	3	30
67	3	95
68	3	120
72	3	230
66	4	255

Изделия 11–15:

№(i)	ПМП	l_i
120	1	40
121	1	75
122	1	10
231	1	75
321	1	15
421	1	30
219	2	30
230	2	45
125	2	45

№(i)	ПМП	l_i
221	2	45
317	2	15
338	2	30
419	2	20
425	2	30
518	2	120
130	3	225
128	3	60
129	3	15
131	3	165
224	3	150
325	3	180
322	3	75
323	3	30
324	3	90
426	3	150
126	4	120
127	4	40
222	4	50
320	4	70
319	4	40
423	4	50
123	4	140
220	4	80
318	4	50
422	4	200
519	4	80
521	4	90
517	5	10
612	5	210

Изделия 16–20:

№(i)	ПМП	l_i
138	1	10
139	1	20
137	1	30
234	1	45
235	1	20
236	1	10
329	1	45
330	1	10
331	1	20
433	1	45
434	1	30
432	1	45
523	1	75
524	1	35
525	1	50
615	1	35
616	1	210
614	1	210
136	2	20
229	2	90
134	2	35
132	2	85
233	2	30
135	2	20
232	2	65
328	2	50
431	2	80
326	2	60
522	2	20

№(i)	ПМП	l_i
613	2	100
429	2	65

Изделия 21–25:

№(i)	ПМП	l_i
141	1	40
239	1	20
238	1	10
144	1	15
142	1	30
143	1	15
430	1	30
333	1	5
335	1	60
336	1	45
337	1	35
437	1	15
438	1	25
435	1	20
436	1	10
528	1	50
140	2	90
237	2	105
334	2	20
439	2	70
332	2	15

В нашем примере для согласованного планирования, в соответствии с принципами, указанными в общей методологии, применяем: для группы изделий 1–5 – алгоритм распределения 2, для остальных групп изделий – алгоритм 3. Алгоритмы распределения описаны на стр. 67.

Во время распределения агрегированных работ по ресурсам *проверяется фактическое объединение общих вершин* в соответствии с сетью на критических путях изделий. В нашем случае все общие вершины, определенные технологией, объединились при распределении, изменений в сети второго уровня агрегации не потребовалось.

Распределение групп изделий по приборам в соответствии с порядком 1 приведено в табл. 13 и в графическом виде (диаграмма Гантта) – на рис. 10, а в соответствии с порядком 2 – в табл. 14 (приведены только работы, положение которых изменилось в сравнении с Порядком 1) и на рис. 11. В диаграммах Гантта используются обозначения, приведенные ниже на рис. 9.

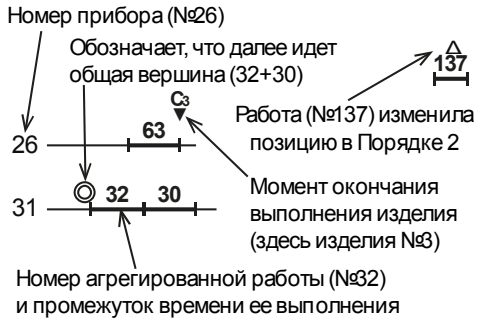


Рис. 9. Обозначения, принятые в диаграммах

Табл. 13. Распределение агрегированных работ по приборам (порядок 1)

прибор	работа	изделие	$t_{нач}$	$t_{кон}$	прибор	работа	изделие	$t_{нач}$	$t_{кон}$
11	11	1	12	62	23	120	11	830	870
11	310	9	285	375	24	43	2	170	178
11	311	10	380	515	24	51	1	178	194
11	132	16	915	1000	24	215	10	280	380
11	142	23	1075	1105	24	231	13	755	830
11	143	24	1105	1120	24	318	14	860	910
11	430	25	1120	1150	24	328	17	980	1030
12	115	9	155	230	24	232	16	1030	1095
12	214	10	230	265	24	337	25	1150	1185
12	229	17	890	980	24	336	24	1185	1230
12	135	16	1010	1030	24	335	23	1230	1290
12	438	25	1225	1250	24	333	22	1290	1295
12	437	24	1260	1275	25	110	5	0	10
12	436	23	1290	1300	25	41	1	164	178
12	435	22	1300	1320	25	213	8	215	225
12	334	22	1365	1385	25	36	5	225	295
12	332	21	1385	1400	25	130	15	370	595
13	13	1	0	4	25	321	13	830	845
13	34	3	60	66	26	63	3	105	150
13	134	16	785	820	26	312	10	265	380
13	319	14	820	860	26	425	12	1020	1050
13	219	11	870	900	26	140	21	1170	1260
13	144	25	1105	1120	31	32	2	72	120
14	14	1	0	10	31	30	1	120	164

при-бор	рабо-та	изде-лие	$t_{нач}$	$t_{кон}$	при-бор	рабо-та	изде-лие	$t_{нач}$	$t_{кон}$
14	113	7	145	215	31	523	18	440	515
14	38	7	300	380	31	525	20	515	565
14	515	10	590	610	31	524	19	565	600
14	126	14	620	740	31	317	12	840	855
14	522	16	1100	1120	31	230	12	855	900
15	111	5	0	40	31	316	11	900	910
15	35	3	60	72	32	46	4	67	72
15	21	1	72	80	32	44	3	72	90
15	116	10	210	230	32	234	18	220	265
15	131	15	430	595	32	235	19	265	285
15	320	14	790	860	32	236	20	285	295
15	136	17	870	890	32	47	5	295	365
16	18	4	0	15	32	411	6	365	380
16	62	1	194	218	32	412	7	380	390
16	117	10	250	280	32	39	8	420	450
16	128	15	535	595	32	423	14	860	910
16	613	16	1120	1220	33	413	9	375	435
16	238	22	1355	1365	33	417	10	610	815
17	19	5	0	30	33	421	13	845	875
17	61	1	194	216	33	422	14	910	1110
17	118	10	240	280	34	17	3	0	24
17	129	15	580	595	34	64	3	105	138
17	429	16	1155	1220	34	114	8	195	215
17	239	22	1270	1290	34	37	6	305	365
18	16	3	0	3	34	415	10	550	610
18	65	4	90	165	34	127	14	700	740
18	137	18	190	220	34	518	12	930	1050
18	119	10	265	280	34	326	16	1095	1155
18	57	5	365	455	34	439	22	1405	1475
18	517	11	930	940	41	56	4	75	90
18	322	15	1025	1100	41	54	3	90	105
19	15	2	0	60	41	338	12	900	930
19	31	1	62	80	41	419	11	930	950
19	112	6	155	215	42	510	5	365	445
19	416	10	515	590	42	434	20	450	480
19	125	12	750	795	42	68	8	480	600
19	528	23	1300	1350	42	616	20	600	810
21	23	1	10	12	42	516	10	815	845
21	141	22	1220	1260	42	325	15	920	1100
21	237	21	1260	1365	42	521	14	1110	1200
22	25	3	36	60	43	58	5	365	395
22	24	2	60	72	43	614	18	640	850
22	329	18	305	350	43	323	15	1070	1100
22	330	19	350	360	44	59	5	365	415
22	331	20	360	380	44	615	19	745	780
22	511	6	380	440	44	220	14	780	860
22	512	7	440	450	44	69	10	900	920
22	513	8	450	480	44	324	15	1010	1100
22	222	14	740	790	44	519	14	1120	1200
22	221	12	795	840	51	53	2	178	202
22	426	15	1100	1250	51	433	19	410	455
23	27	4	15	45	51	67	7	455	550
23	28	5	45	225	51	72	8	600	830
23	138	19	245	255	51	233	16	1000	1030
23	139	20	255	275	51	431	17	1030	1110
23	212	7	275	290	52	71	4	165	285
23	211	6	290	305	52	432	18	395	440
23	123	14	455	595	52	66	6	480	735
23	224	15	595	745	52	611	10	845	920
23	122	13	745	755	52	612	11	940	1150
23	121	12	755	830					

Табл. 14. Распределение агрегированных работ по приборам (порядок 2): измененные положения работ

при-бор	рабо-та	изде-лие	$t_{нач}$	$t_{кон}$	при-бор	рабо-та	изде-лие	$t_{нач}$	$t_{кон}$
18	137	18	235	265	32	236	20	350	360
23	138	19	320	330	32	47	5	450	520
23	139	20	330	350	34	114	8	255	275
25	213	8	275	285	19	112	6	215	275
25	36	5	285	355	18	57	5	520	610
14	113	7	205	275	43	58	5	520	550
32	234	18	285	330	44	59	5	520	570
32	235	19	330	350	42	510	5	1200	1280

3.1.3. Проверка совместимости технологии: в нашем случае работы лишь одной группы (316, 317 и 230) фактически распределились на прибор последовательно. В случае пары работ 135 и 229 существует разрыв 30 минут, который можно ликвидировать сдвигом работы 135 на 30 минут к началу: $t_{нач} = 1010 - 30 = 980$. Для остальных групп работ, а именно: 221 и 222; 231 и 318; 120, 121, 122 и 123; 332, 334, 435, 436, 437 и 438, – разрывы между выполняющимися не последовательно работами ликвидировать сложно (это требует перераспределения части агрегированных работ, что может повлечь за собой появление новых несоответствий полученного распределения исходной технологии). Поэтому вносятся изменения в агрегированную сеть первого уровня агрегации (рис. 7) и исходную сеть. Например, часть агрегированной сети первого уровня и исходной сети по группе изделий 21–25 на приборе 12 изменяется так, как показано на рис. 3. Это не требует повторного решения задачи, поскольку не меняются критические пути изделий.

3.2. По результатам распределения получены моменты окончания выполнения конечных работ изделий (табл. 15), используемые для расчета значения функционала (1).

Показатели согласованных планов в различных разрезах сведены в табл. 16. Сумма длительностей критических путей по всем изделиям равна 7523. В расчетах значения функционала по первой группе изделий принимаем величину планового периода $T = 1300$ минут.

Ухудшение показателей по второму порядку распределения произошло из-за занятости ресурсов, в результате чего некоторые работы пришлось выполнять намного позже, чем задано технологией.

В нашем случае была достигнута минимальная ожидаемая прибыль предприятия (20000 условных единиц) только при первом порядке распределения групп изделий – значение функционала (1) составило $21013 > 20000$ у.е., поэтому для передачи на следующий уровень планирования выбирается согласованный план выполнения работ по первому порядку назначения групп изделий.

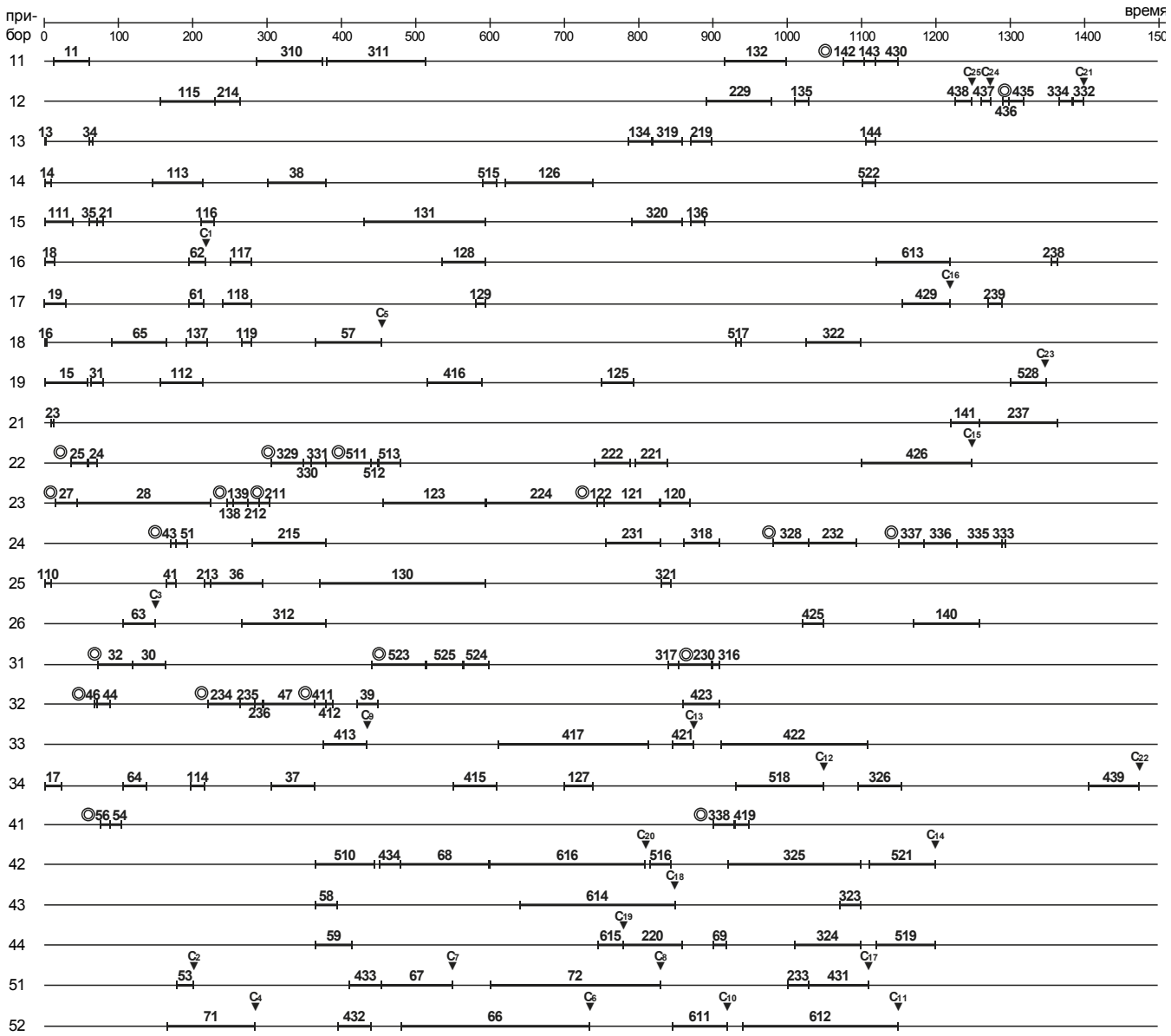


Рис. 10. Диаграмма Гантта: распределение агрегированных работ по приборам (Порядок 1)

Табл. 15. Моменты окончания изделий C_i

№(i)	j_{end}	C_i	d_i	Δ_i	T_i	№(i)	j_{end}	C_i	d_i	Δ_i	T_i
1	62	218			–	**14**	521	1200	1200	5	0
2	53	202			–	**15**	426	1250	1250	10	0
3	63	150			–	**16**	429	1220	1220		0
4	71	285			–	**17**	431	1110	1110		0
5	57*	455*			–	**18**	614	850	850	5	0
6	66	735	735	5	0	**19**	615	780	780	10	0
7	67	550	550	10	0	**20**	616	810	810	15	0
8	72	830	830	15	0	**21**	332	1400	1400		0
9	413	435	435		0	**22**	439	1475	1475		0
10	611	920	920		0	**23**	528	1350	1350		0
11	612	1150	1150	5	0	**24**	437	1275	1275		0
12	518	1050	1050	10	0	**25**	438	1250	1250		0
13	421	875	875	15	5						

j_{end} – номера конечных работ изделия.

* При втором порядке распределения конечная работа 510 выполнилась последней из списка конечных работ изделия 5, поэтому момент окончания выполнения изделия 5 равен $\mathbf{C}_5 = C_{510} = 1280$.

Табл. 16. Показатели согласованных планов

Показатель	Порядок 1	Порядок 2
Значение функционала (1), у.е.	21013	16888
Количество изделий, законченных точно в свой директивный срок	20*	20*
Суммарное время прохождения изделий в системе планирования	9780	10230
Суммарная задержка выполнения критических путей	2257	2707
в т.ч.: по группе изделий 1–5	132	957
по остальным изделиям	2125	1750
Суммарный простой приборов после начала выполнения работ	16831	16516

* По изделиям первой группы директивный срок не задавался

Блок 4. Построение модели третьего уровня.

В блоке 4 формируется многоэтапная сетевая задача календарного планирования; директивные сроки (или интервалы допустимого решения) для конечных работ изделий определяются моментами $\mathbf{C}_i$ окончания их конечных работ, полученными при согласованном планировании (табл. 15).

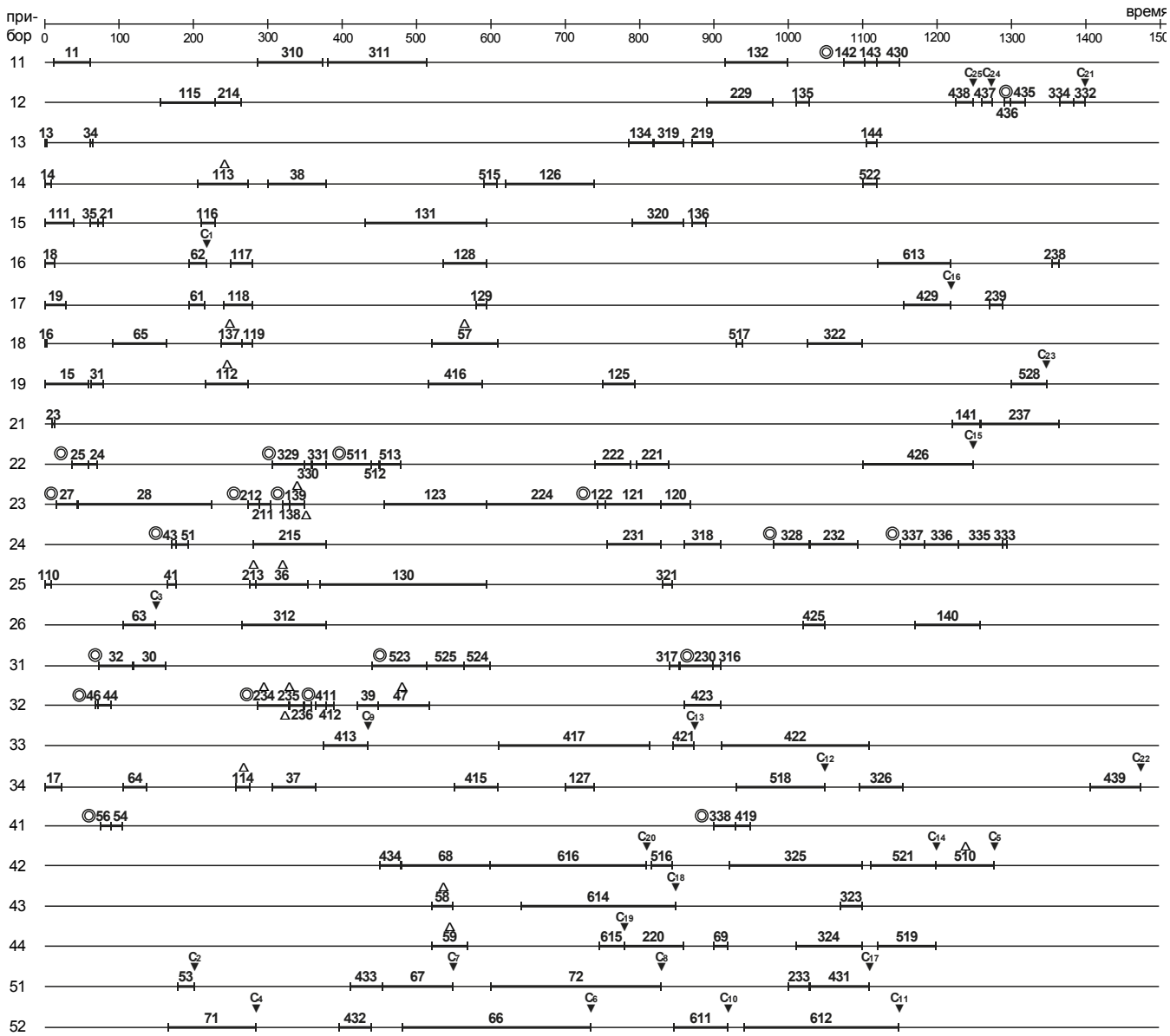


Рис. 11. Диаграмма Гантта: распределение агрегированных работ по приборам (Порядок 2)

Блок 5. Оперативное планирование.

В случае частичного невыполнения плана реализуется оперативное планирование [1].

Решение задач блоков 4 и 5 более подробно рассмотрено в общей методологии планирования.

Выводы

Приводится методология реализации четырехуровневой модели планирования, принятия решений и оперативного планирования в сетевых системах с ограниченными ресурсами в соответствии с общей схемой реализации алгоритмического обеспечения, представленной в [1]. На примере показана методология предварительного и согласованного планирования, выявлены и решены сопутствующие проблемы. Приведенная методология станет основой для создания программной системы планирования, принятия решений и оперативного планирования в сетевых системах с ограниченными ресурсами.

Список литературы

1. 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 –С.11–23.
2. Згуровский М.З., Павлов А.А. Принятие решений в сетевых системах с ограниченными ресурсами: Монография.– К.: Наукова думка. – 2010. – 573 с.
3. Конвей Р.В., Максвелл У.Л., Миллер Л.У. Теория расписаний.– М.: Наука, 1975.– 359 с.
4. Павлов А.А., Теленик С.Ф. Информационные технологии и алгоритмизация в управлении.– К.: Техника.– 2002.– 344 с.

УДК 004.4.277 : 004.056

ШИРОЧИН С.С.,
СУЛЕМА Є.С.

СПОСІБ СТЕГАНОГРАФІЧНОГО ЗАХИСТУ ДАНИХ В АУДІО-ФАЙЛАХ НА ОСНОВІ КОМПЛЕМЕНТАРНОГО ОБРАЗУ

Хмарні сховища є зручним засобом для збереження цифрових мультимедійних даних користувача (зображення, аудіо- та відеодані), й вони все більше набувають популярності серед широкого кола користувачів. Але при використанні хмарних сховищ, що не надають гарантії захисту даних, виникає небезпека доступу до особистих мультимедійних даних користувача сторонніх осіб. Використання існуючих способів захисту інформації для захисту мультимедійних даних користувача у хмарних сховищах не є доцільним, оскільки, по-перше, вони не враховують особливостей мультимедійних даних, а по-друге, не є орієнтованими на потреби звичайного користувача, зокрема, при розробленні засобів захисту звичайно не береться до уваги час обробки даних при їх шифруванні та дешифруванні, хоча для користувача швидкодія є важливою характеристикою програми. Тому розроблення нових способів захисту мультимедійних даних користувача є актуальною задачею.

В даній статті пропонується стеганографічний спосіб захисту даних в аудіо-файлах, який ґрунтується на збереженні комплементарного образу стегоданих замість самих стегоданих та використанні в ролі ключа модифікованого латинського квадрату. При побудові комплементарного образу використовуються особливості мультимедійних даних, а саме: багатоканальність аудіо. Уданому способі передбачається виконання паралельних обчислень, що дозволяє досягти кращих часових показників процедури кодування та декодування стегоданих порівняно з використанням алгоритму шифрування AES. Спосіб, що пропонується, може бути використаний для захисту особистих даних користувача в реальному часі при їх передаванні по мережі та зберіганні в хмарних сховищах.

Cloud storages give a convenient way for keeping user's digital multimedia data (images, audio and video data), and they are becoming more and more popular among a large number of users. However in case of using cloud storage that doesn't guarantee data protection a threat of unauthorized access to user's personal multimedia data appears. The application of existing protection methods to multimedia data in clouds isn't appropriate, because, firstly, they don't take into account the specific properties of multimedia data, and secondly, they aren't oriented to needs of a usual user. In particular, time of data processing during encryption and decryption isn't usually considered as a characteristic of data protection software, though it's important for a user. Therefore the development of specific methods of multimedia data protection is a topical task.

In this paper a new steganography method of data protection in audio-files is proposed, it's based on storing a complimentary image instead of stegodata as well as on the use of a modified Latin square as a key. Special properties of multimedia data are used while creating a complimentary image, in particular multi-channel nature of audio data is taken into account. Parallel computations are stipulated to be used in the proposed method. It lets to achieve better time characteristics of stegodata encryption and decryption in comparison with the use of AES encryption algorithm. The proposed method can be used for user's personal data protection in real time while transferring through network and keeping in cloud storages.

Вступ

Поширення використання хмарних сховищ даних призводить до того, що більшість користувачів Інтернету мають певну долю власних даних, що зберігаються в хмарних сховищах. Зокрема, користувачі зберігають в хмарних сховищах свої персональні мультимедійні дані – фотографії, аудіо- та відеозаписи. Дана інформація носить характер особистої і не має розповсюджуватися без згоди власника. Проте, зберігаючи дані поза локальним комп'ютером, користувач втрачає контроль за доступом до цих даних. Оскільки не кожен користувач є фахівцем із захисту інформації, задача розроблення простого та доступного кожному користувачу способу захисту персональних мультимедійних даних від несанкціонованого доступу в хмарних сховищах є актуальною.

Одним зі способів захисту даних є стеганографічний захист, який полягає у приховувані самого факту наявності деяких секретних даних. Позитивною якістю стеганографії для захисту мультимедійних даних є те, що стеганографічний спосіб дозволяє приховувати зображення у зображенні та аудіо-дані у аудіо-даних, що на практиці означає можливість збереження у відкритому доступі (наприклад, у хмарному

сховищі) замість мультимедійних даних користувача, які не підлягають поширенню (*стегоданих*), іншого зображення або аудіо-запису, що насправді є *контейнером*, який приховує стегодані. Таким чином, користувач може вільно користуватись відкритим сховищем, приховуючи сам факт наявності у нього деяких мультимедійних даних, які він хоче тримати у таємниці. Проте стеганографія має суттєвий недолік: якщо факт наявності стегоданих у контейнері стане відомим зловмиснику, несанкціонований доступ до них буде технічно нескладною задачею. Отже, доцільним є включення у процедуру захисту етапу попереднього шифрування стегоданих. Це шифрування може виконуватись за деяким відомим алгоритмом криптографічного захисту (наприклад, DES, AES, RSA тощо) або за спеціалізованим алгоритмом шифрування [1,2]. В даному досліджені пропонується спосіб стеганографії аудіо-даних, у якому як контейнер використовується аудіо-файл, а підвищення ступеня захисту стегоданих досягається за рахунок їх попереднього перетворення за допомогою *комплементарного перетворення*. Окремо слід підкреслити, що оскільки для користувача важливим є не лише захист даних, а й час їх шифрування та дешифрування, тому у даній статті питанню швидкодії алгоритму обробки стегоданих приділяється особлива увага.

Мета дослідження

Метою дослідження, результати якого наведено у дані статті, є створення стеганографічного способу захисту даних користувача в аудіо-файлах при їх зберіганні у хмарних сховищах зі забезпеченням підвищеної швидкодії обробки даних за рахунок виконання паралельних обчислень.

Опис способу

Спосіб, що пропонується, є модифікацією способу стеганографії зображень [2]. Однією із особливостей способу є використання природних властивостей аудіо-даних, а саме – *багатоканальності*. Однакова тривалість звукових каналів дає можливість їх комбінування для забезпечення захисту даних, що приховуються. В даному способі пропонується модифікація молодших бітів (LSB – *Least significant Bits*) [3] значень рівня звукового сигналу лише в одному з каналів, який будемо називати *контейнерним* каналом.

Іншою й основною особливістю способу є використання зашифрованого образу стегоданих, що утворюється за допомогою комплементарного перетворення, яке задає відображення множини стегоданих на множину даних одного зі звукових каналів контейнера, що не використовується в ролі контейнерного каналу. Будемо називати цей канал *комплементарним*.

Результатом цього відображення є множина зашифрованих даних, яку будемо називати *комплементарним образом (КО)*. КО вбудовується в контейнер замість стегоданих, проте тільки в один канал – контейнерний.

Комплементарне перетворення задається таблицею (*ключем*), що зберігає відповідність між байтами комплементарного каналу та байтами стегоданих. Комплементарне перетворення має задовольняти такі вимоги:

- для будь-якої пари значень даних у комплементарному каналі та стегоданих, що знаходяться в певному діапазоні значень, має існувати одне та лише одне значення ключа, що знаходиться у тому ж діапазоні;
- має існувати однозначне зворотне перетворення, що дозволяє отримати стегодані з відповідних даних КО, комплементарного каналу і ключа.

Ці вимоги задовольняє комплементарне перетворення, що ґрунтується на застосуванні латинського квадрату [4]. Створення ключа на основі латинського квадрату відбувається за алгоритмом, розробленим для захисту графічних даних [2]. Даний алгоритм передбачає, відповідно до наведених вище вимог, формування таблиці, яка заповнюється за наступними правилами:

- кожне значення може зустрічатись в рядку лише один раз;
- кожен рядок має містити всі варіанти значень у заданому діапазоні;
- послідовність значень в рядках має бути випадковою.

Таким чином, генерування ключа в способі, що пропонується, відбувається за наступним алгоритмом:

1. Створюється латинський квадрат розміром 256×256.
2. Для кожного рядка генерується псевдовипадкове число кількості перестановок у діапазоні від 128 до 255.
3. Для кожної перестановки генеруються два псевдовипадкових числа у діапазоні від 0 до 255.
4. Елементи з номерами двох отриманих чисел міняються місцями.

Отриманий ключ використовується для створення КО.

Перед застосуванням способу необхідно попередньо виконати аналіз місткості контейнера, враховуючи наступні міркування.

Для зберігання даних використовується лише один канал, при чому змінювати можна лише старший байт кожного елементу типу *double*. Через це обсяг даних, що вбудовуються у контейнер, збільшується в 4 рази, порівняно зі звичайним стеганографічним способом захисту на основі модифікації LSB. Розмір КО (тобто об'єм даних, що вбудовуються у контейнер) залежить від кількості бітів b, що модифікуються: оскільки значення b може обиратись у діапазоні від 1 до 8, це може допомогти компенсувати збільшення розміру КО.

Отже, розміщення стегоданих в контейнері є можливим за виконання наступної умови:

$$\frac{C \cdot b}{8} \geq 4 \cdot I,$$

що спрощується до умови:

$$C \geq \frac{32 \cdot I}{b},$$

де b – кількість біт, що модифікуються;

I – розмір стегоданих;

C – розмір контейнера;

Процес створення КО полягає у заміні пари значень $< \mathrm{S_i}, \mathrm{C_j} >$, де S_i – i-й байт стегоданих, $\mathrm{C_j}$ – j-й байт контейнера, значенням X_i, яке являє собою i-й байт КО, відповідно до ключа T. При цьому S_i та $\mathrm{C_j}$ використовуються в ролі координат комірки зі значенням X_i у ключі (таблиці) T.

Отриманий блок даних (тобто КО) вбудовується в контейнерний канал шляхом модифікації LSB. Зміна даних у комплементарному каналі не відбувається. Оскільки для збереження аудіо-даних в форматі *wave*, до якого в даному способі перетворюються всі вхідні аудіо-формати, використовується тип даних *double*, що займає 2 байти, то модифікація LSB відбувається, як було зазначено вище, лише у старшому байті кожного елементу типу *double*: байти в *double* розташовані в зворотному порядку і модифікація старшого байту менше впливає на значення *double*. В більшості випадків старший байт містить нулі в молодших бітах, тому для модифікації може бути використано більше, ніж 1 біт старшого байта. При реалізації даного способу програмним шляхом була забезпечена можливість модифікації від 1 до 8 біт старшого байта.

Після вбудовування даних у контейнер, аудіо-дані ущільнюються без втрат. Для збереження контейнеру можуть бути використані будь-які формати з ущільненням без втрат, наприклад, FLAC.

Ключ передається окремо від основних даних згідно зі загальноприйнятими правилами передачі секретного ключа.

Схема захисту даних в аудіо-файлі представлена на рис. 1.

Декодування стегоданих відбувається аналогічно декодуванню у способі стеганографії зображень на основі комплементарного образу [2] і складається з двох етапів. Спочатку відбувається зчитування комплементарного образу, що для біту n байту k комплементарного образу визначається наступним чином:

$$X_k = X_k | (C_i \& (255 \gg (8-b))) \ll (8-n-b),$$

де X_k – k-й байт комплементарного образу;

C_i – i-й байт контейнера;

b – кількість стегобіт в байті контейнера;

n – порядковий номер стегобіта в послідовності стегобіт, $n \leq b$.

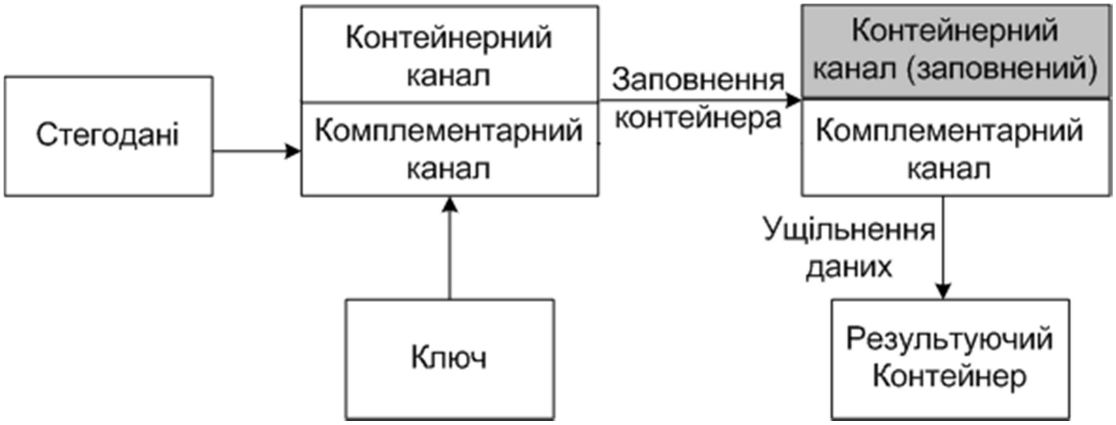


Рис. 1. Схема захисту даних в аудіо-файлі

Далі з КО отримується приховане зображення. З пари значень $< C_i, X_i >$, де C_i – i-й байт контейнерного каналу, X_i – i-й байт комплементарного каналу, за допомогою ключа T отримується значення S_i, що являє собою i-й байт прихованого зображення.

Оскільки канали розміщуються в аудіо-файлі паралельно, зі зсувом на 2 байти, адреси КО можна виразити через адреси контейнера, тоді значення S_i буде отримуватись з пари $< C_i, C_{i+2} >$, якщо правий канал є комплементарним, а лівий – контейнерним і з пари $< C_{i+2}, C_i >$ в протилежному випадку.

Розпаралелювання

Для покращення часових характеристик алгоритму реалізації запропонованого способу було застосоване розпаралелювання між ядрами багатоядерного процесора. Частина алгоритму, що має бути розпаралелена, обрана аналогічним чином до способу стеганографії зображень на основі комплементарного образу [2]. Розпаралелювання було застосовано до запису даних в контейнер та до зчитування їх з контейнеру.

Програмно алгоритм запису даних в контейнер (рис. 2) реалізується у вигляді двох вкладених циклів, де зовнішній цикл відповідає за побайтову обробку стегоданих, а внутрішній – за вбудовування кожного окремого біту. Кількість ітерацій зовнішнього циклу залежить від об'єму стегоданих, кількість ітерацій внутрішнього – від кількості стегобіт. Для обрання оптимального способу розпаралелювання було розглянуто декілька варіантів паралельних реалізації алгоритму:

1. Розпаралелювання лише внутрішнього циклу.
2. Розпаралелювання як зовнішнього, так і внутрішнього циклів.
3. Розпаралелювання лише зовнішнього циклу.

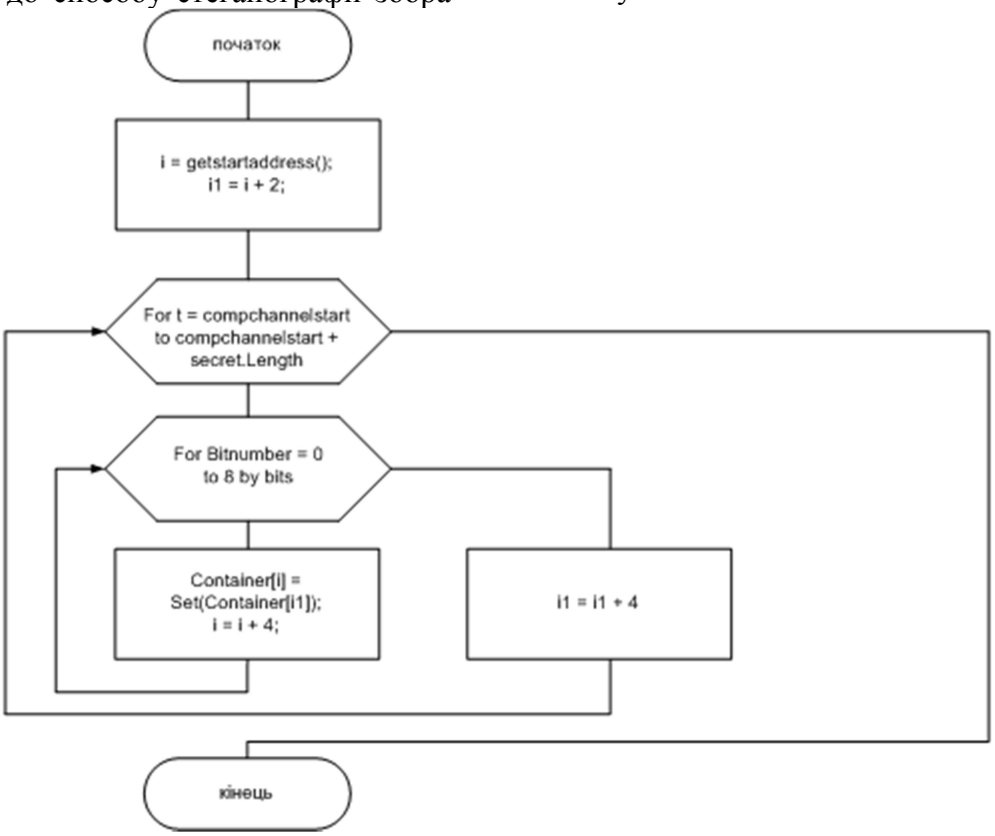


Рис. 2. Блок-схема алгоритму запису даних в контейнер без використання розпаралелювання

Оскільки внутрішній цикл має обмежену кількість ітерацій (від 1 до 8), то у першому та другому варіантах розпаралелювання є неефективним через те, що навіть у випадку 2 ядер та 8 стегобіт кожен паралельний потік буде виконаний лише 4 рази. В більшості випадків використовується мінімальна кількість стегобіт, тому розпаралелювання буде лише збільшувати обчислювальну складність через необхідність організації паралельних потоків, у яких фактично буде виконуватись лише 1 ітерація.

Найбільш раціональним є варіант розпаралелювання лише зовнішнього циклу, оскільки в такому випадку у кожному потоці буде виконуватись достатньо велика кількість ітерацій.

Схема роботи розпаралеленого алгоритму запису даних в контейнер зображена на рис. 3.

Розпаралелювання алгоритму зчитування даних з контейнеру було виконане аналогічним чином

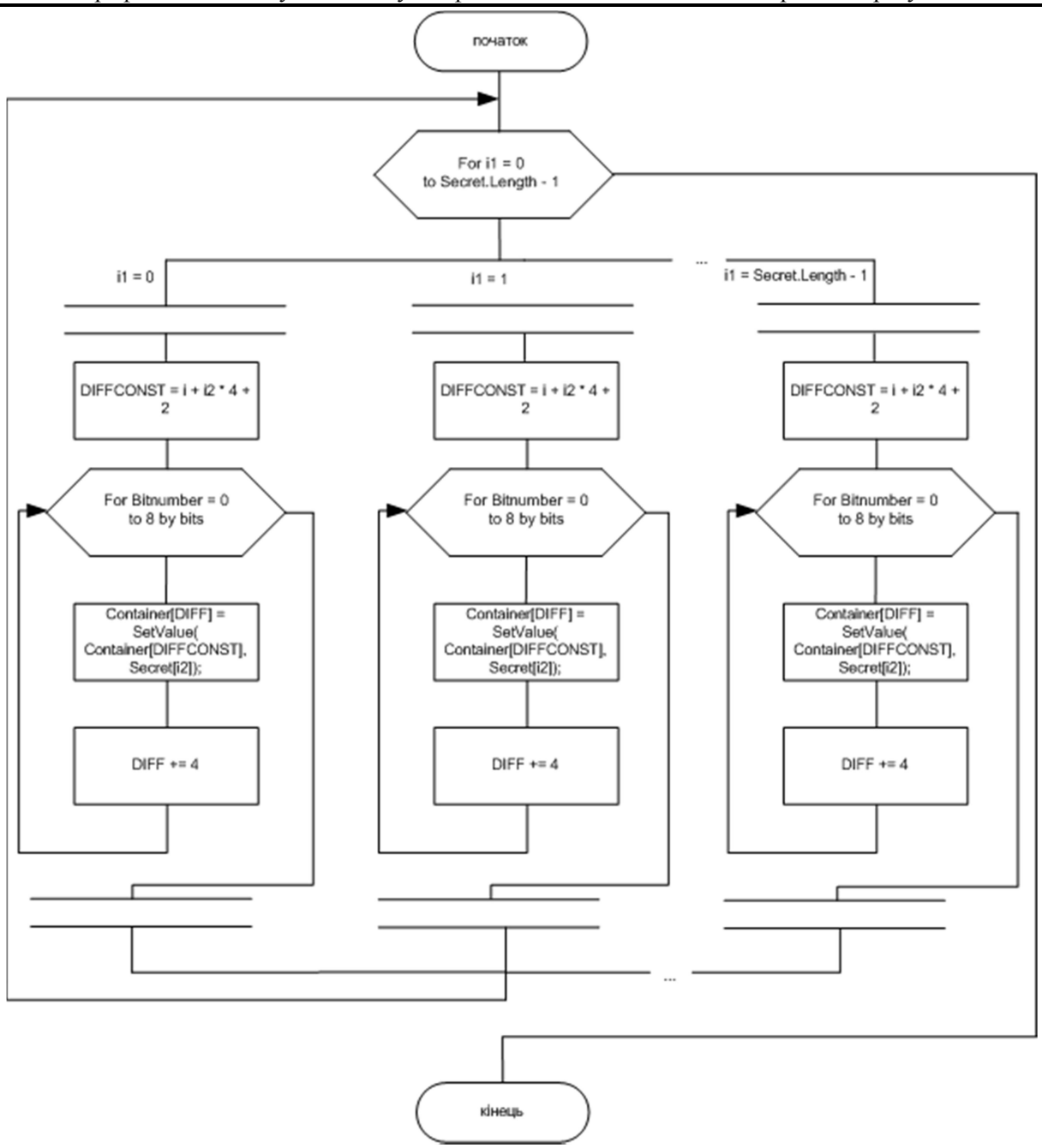


Рис. 3. Блок-схема алгоритму запису даних в контейнер з використанням розпаралелювання

Результати

Запропонований спосіб оцінювався за двома критеріями: ступінь захисту стегоданих та швидкодія алгоритму.

Ступінь захисту може бути оцінений наступним чином. Ключ має фіксований розмір 64 Кб, тобто 524288 біт, отже складність підбору ключа буде 2^{524288}. Для порівняння – максимальний розмір ключа AES дорівнює 256 бітам, отже складність його підбору складає 2^{256}. Оскільки у запропонованому способі передбачається використання ключа лише один раз, це гарантує захист від фальсифікації ключа.

При оцінці швидкодії програмної реалізації запропонованого способу часові характеристики алгоритму порівнювались з часовими характеристиками таких алгоритмів:

- алгоритм стеганографічного захисту на основі модифікації LSB (базовий спосіб захисту),
- алгоритм стеганографічного захисту на основі модифікації LSB з використанням шифрування AES (виконується попереднє шифрування стегоданих за алгоритмом AES перед виконанням модифікації LSB),
- алгоритм стеганографічного захисту з використанням фрагментації [5],
- алгоритм на основі КО в двох варіантах: звичайному та розпаралеленому.

Всі заміри проводились на ПК з двоядерним процесором Intel Centrino, під операційною сис-

темою Windows 7. В експериментах використовувався однаковий набір файлів. В ролі контейнера використовувався звуковий файл розміром 16 Мб, а в ролі стегоданих – файл розміром 350 Кб.

На рис. 4 наведені результати замірів часу запису стегоданих в контейнер. Аналіз отриманих експериментальних даних дозволяє зробити наступні висновки:

1. Включення у процедуру стеганографічного захисту будь-якого шифрування уповільнює час роботи алгоритму.
2. Збільшення кількості стегобіт прискорює запис при всіх способах шифрування.
3. Швидкодія алгоритму захисту на основі комплементарного образу залежить від кількості стегобіт: при 1 та 2 стегобітах він дає гірші результати, ніж інші способи; при 4 і 8 стегобітах він є швидшим, ніж алгоритм з використанням шифрування AES.
4. Розпаралелювання алгоритму захисту на основі комплементарного образу дозволяє досягнути збільшення швидкодії. Чим менша кількість стегобіт, тим більший ефект від розпаралелювання.

Також було заміряно час зчитування стегоданих (рис. 5). Аналіз отриманих даних дозволяє зробити такі висновки:

1. Алгоритм захисту на основі комплементарного образу показує кращі часові результати, ніж алгоритм з використанням шифрування AES.
2. Розпаралелювання алгоритму захисту на основі комплементарного образу при зчитуванні дає прискорення лише при 1 та 2 стегобітах. В інших випадках значення приблизно однакові (4 стегобіти) або паралельна реалізація вимагає більше часу (8 стегобіт).
3. Збільшення кількості стегобіт прискорює всі алгоритми, оскільки модифікується менша кількість байт контейнера.

Також було проаналізовано залежність швидкодії алгоритму від кількості стегобіт. Було виявлено, що при збільшенні кількості стегобіт в кожному байті контейнера швидкодія алгоритму зростає, оскільки для вбудовування тієї ж самої кількості стегоданих потрібно модифікувати меншу кількість байт контейнера, отже обробка даних виконується за меншу кількість ітерацій. Значення в табл. 1 показують, наскільки швидше алгоритм виконується у випадку 2, 4 та 8 стегобіт, ніж при 1 стегобіті за однакових інших умов.

Отримані результати дозволяють зробити висновок, що збільшення швидкодії при записі спостерігається для всіх способів, особливо для способу на основі комплементарного образу. При цьому паралельна реалізація має менші відносні показники приросту швидкодії, ніж послідовна. При збільшенні кількості стегобіт ефективність розпаралелювання зменшується, проте воно все ще є доцільним.

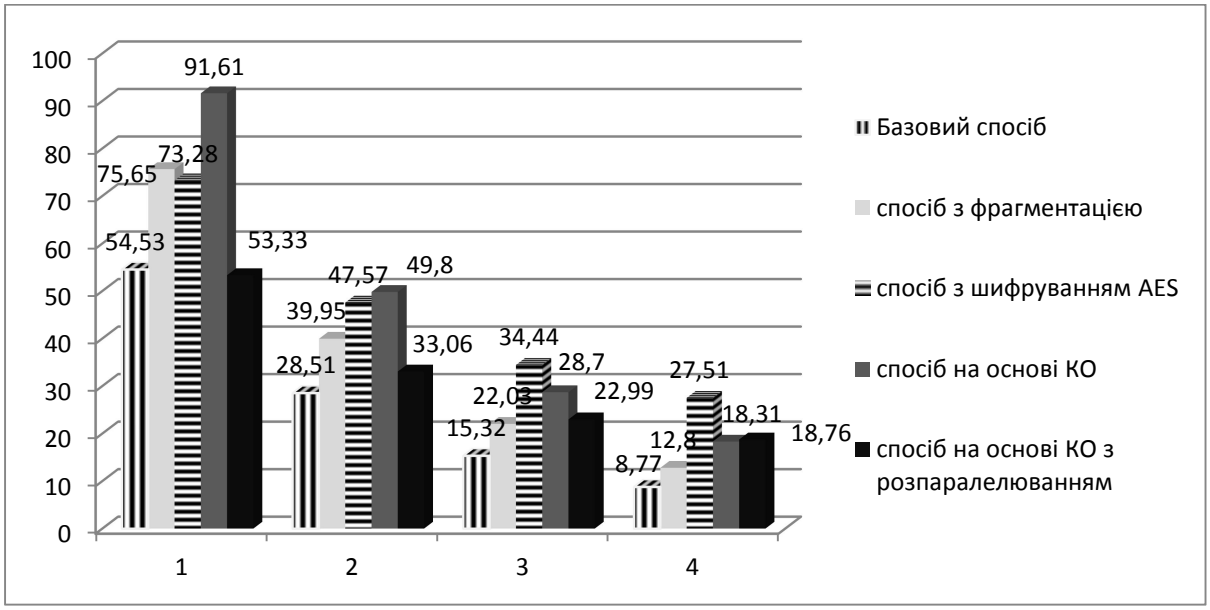


Рис. 4. Час запису стегоданих (мс): 1 – 1 біт, 2 – 2 біта, 3 – 4 біта, 4 – 8 біт

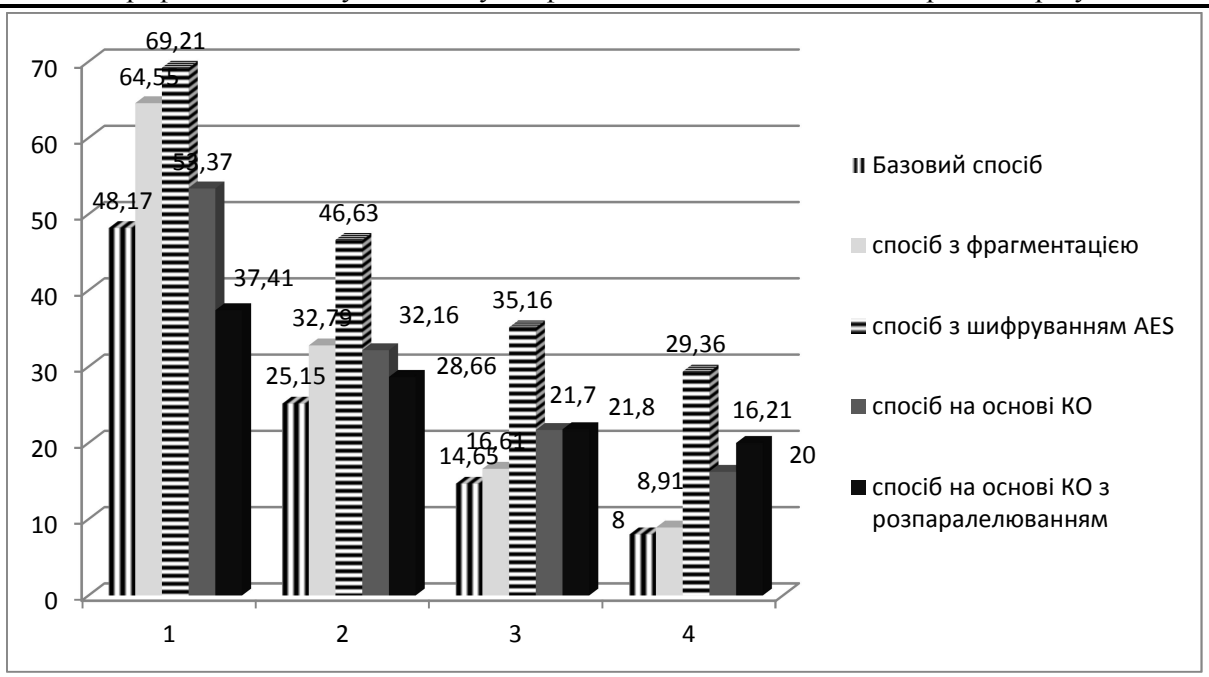


Рис. 5. Час зчитування стегоданих (мс): 1 – 1 біт, 2 – 2 біта, 3 – 4 біта, 4 – 8 біт.

Табл. 1. Ступінь прискорення при збільшенні кількості стегобіт

Алгоритм	**Запис**			**Зчитування**		
	2 біти	**4 біти**	**8 біт**	**2 біти**	**4 біти**	**8 біт**
Базовий спосіб	1,91	3,56	6,22	1,92	3,29	6,02
Спосіб з фрагментацією	1,89	3,43	5,91	1,97	3,89	7,24
Спосіб з шифруванням AES	1,54	2,13	2,66	1,48	1,97	2,36
Спосіб на основі КО	1,84	3,19	5,00	1,66	2,46	3,29
Спосіб на основі КО з розпаралелюванням	1,61	2,32	2,84	1,31	1,72	1,87

Аналіз приросту швидкодії при зчитуванні даних дає інші результати. Найбільший приріст швидкодії спостерігається для базового способу та для способу з фрагментацією, найменший – для способу на основі комплементарного образу з розпаралелюванням. При 1 та 2 стегобітах розпаралелювання є доцільним, а при 4 та 8 ефект втрачається і стає зворотним. Проте у всіх випадках абсолютні часові показники способу на основі комплементарного образу є кращими, ніж при використанні способу з шифруванням AES, особливо при паралельній реалізації алгоритму.

Висновки

Порівняння запропонованого способу зі способом стеганографічного захисту з використанням шифрування AES дозволяє зробити висновок про те, що оскільки розмір ключа на основі латинського квадрату є більшим, ніж розмір ключа при шифруванні AES, то підбір ключа ускладнюється, що підвищує стійкість способу, але водночас великий розмір ключа на основі латинського квадрату спричинює збільшення часу, що витрачається на його генерування.

Для покращення часових характеристик використовується розпаралелювання алгоритму для виконання на багатоядерних процесорах. Аналіз швидкодії алгоритму реалізації запропонованого способу показав його перевагу порівняно з алгоритмами реалізації інших способів, зокрема з алгоритмом реалізації способу з використанням шифрування AES. Час запису стегоданих у всіх випадках є меншим, ніж при шифруванні, незважаючи на те, що частина часу витрачається на генерування ключа. При зчитуванні стегоданих даний алгоритм є також більш швидким, ніж при застосуванні шифрування, проте його паралельна реалізація є доцільною лише при 1 та 2 стегобітах. При дослідженні виконувалось розпаралелювання на 2 ядра, отже при використанні даного способу на

процесорах з більшою кількістю ядер ефективність розпаралелювання буде більшою. Розпаралелювання обчислень на 2 ядра не дає двократного приросту швидкодії, тому що частина ресурсів витрачається на організацію взаємодії роботи паралельних потоків.

Отже, даний спосіб забезпечує достатній ступінь захисту даних та високі часові показники обробки даних і може бути застосований для захисту користувацьких даних в реальному часі. Зокрема запропонований спосіб може бути використаний для захисту персональних мультимедійних даних користувача при їх зберіганні в хмарних сховищах.

Список посилань

1. Сулема Є.С., Широчин С.С. Метод захисту зображень на основі шифрування палітри // Науковий журнал «Вісник Хмельницького національного університету». – Хмельницький : ХНУ. – 2014. – №3. – С. 114-119.
2. Сулема Є.С., Широчин С.С. Спосіб стеганографії зображень на основі комплементарного образу // Журнал «Захист інформації». – Київ : НТУУ «КПІ». – 2013. – Випуск 4. – С. 345-353.
3. Owens, M. A discussion of covert channels and steganography // SANS Institute, 2002.
4. Латинские квадраты [Електронний ресурс] // Режим доступу: http://www.e-olimp.com/en/problems/2105 – [02.03.2012].
5. Сулема Є.С., Широчин С.С. Спосіб стеганографії зображень з фрагментацією стегоданих та розділенням закритого ключа. // Науково-технічний збірник «Правове, нормативне та метрологічне забезпечення системи захисту інформації в Україні». – Київ : НТУУ «КПІ». – 2012. – Випуск 1 (22). – С. 64-68.

УДК 519.854.2

ПАВЛОВ О.А.,
ЖДАНОВА О.Г.,
СПЕРКАЧ М.О.

ЗАДАЧА СОСТАВЛЕНИЯ ДОПУСТИМОГО РАСПИСАНИЯ С МАКСИМАЛЬНО ПОЗДНИМ МОМЕНТОМ ЗАПУСКА ВЫПОЛНЕНИЯ ИДЕНТИЧНЫМИ ПАРАЛЛЕЛЬНЫМИ ПРИБОРАМИ РАБОТ С ОБЩИМ ДИРЕКТИВНЫМ СРОКОМ

В статье рассмотрены исследования свойств задачи составления допустимого расписания выполнения идентичными параллельными приборами работ с общим директивным сроком с максимально поздним моментом запуска. Применяя методологию построения ПДС-алгоритмов, разработаны признаки оптимальности расписаний, на основе которых определено множество перестановок, позволяющих последовательно улучшать значение критерия. Разработана полиномиальная составляющая ПДС-алгоритма решения задачи. Рассмотрено практическое применение разработанного алгоритма. Проведены экспериментальные исследования алгоритма.

The article describes the properties research of the problem of creating a feasible schedule of jobs with the common due date on identical parallel machines with a maximum late start. Applying the methodology of PDC-algorithms the signs of optimal schedules are developed, which are defined on the basis of the permutations set that allow to consistently improve the value of the criterion. The polynomial component of the PDC-algorithm for the problem solution is developed. A practical application of the developed algorithm is considered. The experimental studies of the algorithm is given.

1. Введение

Рассматриваемая в работе задача относится к задачам теории расписаний, методы которой используются для оптимизации оперативно-календарных планов. Одним из типов оперативно-календарного планирования является цеховое планирование, в котором встречаются задачи, где в систему с идентичными приборами одновременно поступает некоторое количество работ и имеются ограничения на сроки запуска и выпуска продукции. При этом есть потребность в разработке таких алгоритмов построения расписаний, которые могут обеспечить высокое качество получаемых результатов и не требуют значительных вычислительных ресурсов.

2. Постановка задачи

Задано множество работ J ($|J| = n$), количество приборов m, для работы $j \in J$ известна продолжительность выполнения p_j. Предполагается, что все работы поступают одновременно и имеют общий директивный срок d, процесс обслуживания каждой работы протекает без прерываний до завершения обслуживания работ. Процесс выполнения работ каждым из приборов является непрерывным: после выполнения первой по порядку сразу же начинает выполняться вторая и т.д.

Необходимо найти расписание, в котором все работы не запаздывают и которое имеет максимально поздний момент запуска приборов (под *моментом запуска* понимается момент начала выполнения множества работ или минимальный из моментов начала выполнения работ из множества J).

Рассматриваемая задача принадлежит к классу NP. Результаты, полученные ранее для этой задачи, приведены в работах [1, 2].

3. Исследование свойств задачи

Пусть $C^* = \left\lfloor \dfrac{\sum_{j=1}^{n} p_j}{m} \right\rfloor$ (здесь $\lfloor a \rfloor$ - наибольшее целое, для которого выполняется $\lfloor a \rfloor \le a$), $\delta = \sum_{j=1}^{n} p_j - C^* m$ (по определению $\delta \ge 0$). Тогда C^* ($C^* + 1$) - теоретически минимальное время, за которое все приборы могли бы выполнить все работы, если $\delta = 0$ (если $\delta > 0$).

Рассмотрим некоторое расписание σ. Введем обозначения:

$C_i(\sigma)$ – момент завершения выполнения всех работ прибором i, $i = \overline{1,m}$;

$\Delta_i(\sigma) = \max\{0; C_i(\sigma) - C^*\}$, $i = \overline{1,m}$;

$$R_i(\sigma) = \max\{0; C^* - C_i(\sigma)\}, \quad i = \overline{1,m}$$

(далее $R_i(\sigma)$ будем называть *резервом* прибора i, $\Delta_i(\sigma)$ *выступом* прибора i);

$I_\Delta(\sigma)$ - множество приборов, у которых $\Delta_i(\sigma) > 0$;

$I_R(\sigma)$ - множество приборов, у которых $R_i(\sigma) > 0$;

$I_0(\sigma)$ - множество приборов, у которых $\Delta_i(\sigma) = R_i(\sigma) = 0$;

$J_i(\sigma)$ - множество работ, которые в расписании σ выполняются прибором i.

С учетом обозначений целевая функция задачи имеет вид:

$$r(\sigma) = d - \left(C^* + \max_i \Delta_i(\sigma)\right) = d - C^* - \max_i \Delta_i(\sigma).$$

Так как величины d и C^* не зависят от упорядочения, максимизация момента запуска $r(\sigma)$ эквивалентна минимизации величины максимального из выступов $\max_i \Delta_i(\sigma)$.

Обозначим через Ψ класс расписаний, для которых выполняется:

$$\neg\exists h,\ j,s \mid h \in I_\Delta(\sigma),\ j \in J_h(\sigma),\ s \in I_R(\sigma),\ p_j \le R_s(\sigma)$$

и $\neg\exists h,\ j,s \mid h \in I_\Delta(\sigma),\ j \in J_h(\sigma),$

$$s \in I_R(\sigma),\ p_j \le \Delta_h(\sigma), R_s(\sigma) > 0.$$

Можно показать, что существует оптимальное расписание, принадлежащее классу расписании Ψ.

Рассмотрим некоторые свойства расписаний $\sigma \in \Psi$.

Утверждение. Для произвольного расписания σ выполняется [1]:

$$\sum_{i=1}^{m} \Delta_i(\sigma) = \sum_{i=1}^{m} R_i(\sigma) + \delta.$$

Из этого утверждения, учитывая неотрицательность δ, следует, что неравенство $\sum_{i=1}^{m} \Delta_i(\sigma) < \sum_{i=1}^{m} R_i(\sigma)$ невозможно.

Применяя методологию построения ПДС-алгоритмов [3], согласно которой полиномиальная составляющая алгоритма порождается логико-аналитическими условиями, выполнение которых гарантирует оптимальность полученного решения, разработаем признаки оптимальности расписаний.

4. Признаки оптимальности расписаний

Для расписания $\sigma \in \Psi$ возможны такие взаимоисключающие случаи.

Случай I: $\delta = 0$ и $C_i = C^*$, $i = \overline{1,m}$. В этом случае имеем расписание с равномерной загрузкой приборов. Очевидно, что это расписание является оптимальным.

Случай II: $\delta = 0$ и не для всех i выполняется $C_i = C^*$, при этом имеем:

$$\sum_{i=1}^{m} \Delta_i(\sigma) = \sum_{i=1}^{m} R_i(\sigma) > 0.$$

В этом случае построить расписание с равномерной загрузкой приборов возможно.

Случай III: $\delta > 0$, при этом имеем:

$$\sum_{i=1}^{m} \Delta_i(\sigma) > \sum_{i=1}^{m} R_i(\sigma).$$

В этом случае построить расписание с равномерной загрузкой приборов невозможно.

А теперь сформулируем признаки оптимальности расписаний, к выполнению которых нужно стремиться.

Признак оптимальности 1

Равномерное расписание: $\delta = 0$ и $C_i = C^*$, $i = \overline{1,m}$.

Признак оптимальности 2

Пусть, b - наибольший общий делитель (НОД) длительностей выполнения работ $p_j, j = \overline{1,n}$, тогда расписание, в котором для всех $i = \overline{1,m}$ выполняется: $\Delta_i(\sigma) \in \{0, b\}$ является оптимальным [2]. Если разделить величины $p_j, j = \overline{1,n}$ на b, то признак оптимальности в этом случае формулируется так: расписание, в котором для всех $i = \overline{1,m}$ выполняется условие $\Delta_i(\sigma) \in \{0,1\}$, является оптимальным (далее будем предполагать, что все p_j имеют НОД, равный 1).

На основе признаков оптимальности расписаний определим множество перестановок, которые позволят последовательно улучшать значение критерия.

5. Перестановки, улучшающие расписания

Для улучшения расписания необходимо направить усилия на уменьшение максимального из выступов $\max_{i \in I_\Delta} \Delta_i(\sigma)$. Для этого предлагается использовать перестановки работ между приборами: когда некоторое подмножество ра-

бот с прибора h (обозначим его $K_h(\sigma)$, по определению $K_h(\sigma) \subseteq J_h(\sigma)$) меняется местами с некоторым подмножеством работ с прибора s (обозначим это подмножество как $L_s(\sigma)$, $L_s(\sigma) \subseteq J_s(\sigma)$). Обозначим через θ разность между суммами длительностей работ, которые принимают участие в перестановке:

$$\theta = \sum_{j \in K_h(\sigma)} p_j - \sum_{j \in L_s(\sigma)} p_j .$$

Отметим, что в результате перестановок, примененных к текущему расписанию σ, в новом расписании σ^1 для величин $\sum_{i=1}^{m} \Delta_i(\sigma^1)$ и $\sum_{i=1}^{m} R_i(\sigma^1)$ выполняется:

$$\sum_{i=1}^{m} \Delta_i(\sigma) - \sum_{i=1}^{m} \Delta_i(\sigma^1) = \sum_{i=1}^{m} R_i(\sigma) - \sum_{i=1}^{m} R_i(\sigma^1).$$

Рисунок 1 иллюстрирует эту закономерность.

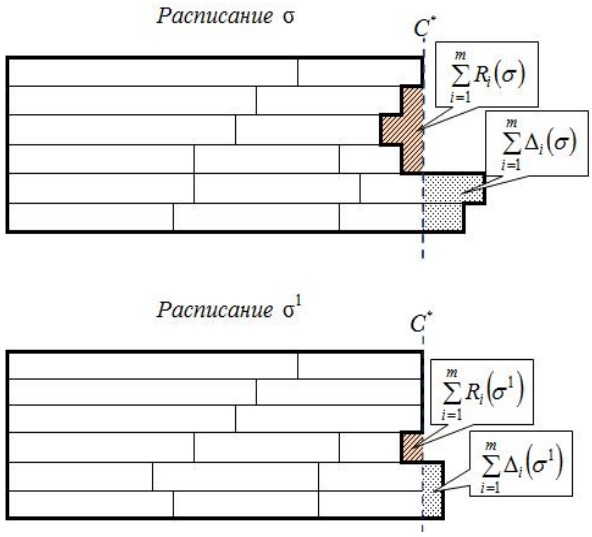


Рис. 1 Синхронное изменение суммы резервов и суммы выступов

Учитывая условия выполнения и последствия, все перестановки, которые приводят к улучшению расписания, можно разделить на четыре типа, которые условно обозначим как ***А***, ***Б***, ***В*** и ***Г***.

Цель перестановок типа ***А***: уменьшение максимального из выступов $\Delta_h(\sigma) = \max_{i \in I_\Delta(\sigma)} \Delta_i(\sigma)$ за счет уменьшения резерва одного из приборов множества I_R.

Условия выполнения перестановки типа ***А***:

$$\theta > 0 \,;\, \theta \le \Delta_h(\sigma);\ \theta \le R_s(\sigma).$$

В результате перестановки получаем расписание σ^1 в котором:

$$\Delta_h(\sigma^1) = \Delta_h(\sigma) - \theta,$$
$$R_s(\sigma^1) = R_s(\sigma) - \theta.$$

Выступ, который был максимальным в σ, уменьшился на θ:

$$\Delta_h(\sigma^1) = \max_{i \in I_\Delta(\sigma)} \Delta_i(\sigma) - \theta .$$

Таким образом, расписание σ^1 не хуже расписания σ, а при отсутствии альтернатив при выборе прибора h лучше, чем σ.

Перестановки типа ***А*** можно разделить на подтипы в зависимости от количества работ, которые участвуют в перестановке. В дальнейшем анализе количество подтипов перестановок ограничим величиной 2: $\max\{|K_h(\sigma)|, |L_s(\sigma)|\} = 2$ (увеличение количества используемых подтипов приводит к повышению эффективности алгоритма, но увеличивает время его работы).

Обобщенная характеристика и свойства перестановок типа ***А*** представлены в таблице 1.

Табл. 1. Свойства перестановок типа А

Тип пере-ста-новки	Приборы и работы, которые принимают участие в перестановке		θ ($\theta > 0$)	Условие, при котором выполняется перестановка	Характеристики расписания σ^1, полученного в результате перестановки
	h	s			
1-1А	$h \in I_\Delta(\sigma)$, $j_1 \in J_h$	$s \in I_R(\sigma)$, $j_2 \in J_s$	$p_{j_1} - p_{j_2}$	$\theta \le \Delta_h(\sigma)$, $\theta \le R_s(\sigma)$	$\Delta_h(\sigma^1) = \Delta_h(\sigma) - \theta$, $R_s(\sigma^1) = R_s(\sigma) - \theta$.
1-2А	$h \in I_\Delta(\sigma)$, $j_1 \in J_h$	$s \in I_R(\sigma)$, $j_2, j_3 \in J_s$	$p_{j_1} - (p_{j_2} + p_{j_3})$		
2-1А	$h \in I_\Delta(\sigma)$, $j_1, j_2 \in J_h$	$s \in I_R(\sigma)$, $j_3 \in J_s$	$(p_{j_1} + p_{j_2}) - p_{j_3}$		
2-2А	$h \in I_\Delta(\sigma)$ $j_1, j_2 \in J_h$	$s \in I_R(\sigma)$, $j_3, j_4 \in J_s$	$(p_{j_1} + p_{j_2}) - (p_{j_3} + p_{j_4})$		

В результате перестановок типа ***А*** мощности множеств I_Δ и I_R остаются неизменными или уменьшаются.

Цель перестановок типа ***Б***: уменьшение максимального из выступов за счет резерва прибора с множества I_R и с образованием выступа на этом же приборе. В результате перестановок этого типа может увеличиться мощность множества I_Δ.

Условия выполнения перестановки типа ***Б***:

$$\theta > 0\,;\ \theta \le \Delta_h(\sigma);\ \theta > R_s(\sigma).$$

В результате перестановки получаем расписание σ^1, в котором:

$$\Delta_h(\sigma^1) = \Delta_h(\sigma) - \theta;$$
$$\Delta_s(\sigma^1) = \theta - R_s(\sigma) > 0;$$
$$s \in I_\Delta(\sigma^1).$$

Выступ, который был максимальным в σ, уменьшился на величину θ:

$$\Delta_h(\sigma^1) = \max_{i \in I_\Delta(\sigma)} \Delta_i(\sigma) - \theta,$$

а для вновь образованного выступа $\Delta_s(\sigma^1)$ выполняется: $\Delta_s(\sigma^1) < \Delta_h(\sigma)$. Покажем это:

$$\Delta_h(\sigma) - \Delta_s(\sigma^1) = \Delta_h(\sigma) - (\theta - R_s(\sigma)) =$$
$$= \Delta_h(\sigma) + R_s(\sigma) - \theta > 0,$$

поскольку, $\Delta_h(\sigma) - \theta \ge 0$, $R_s(\sigma) > 0$.

Таким образом, расписание σ^1 не хуже расписания σ, а при отсутствии альтернатив при выборе прибора h лучше, чем σ.

Обобщенная характеристика и свойства перестановок типа ***Б*** представлены в табл. 2.

Табл. 2. Свойства перестановок типа Б

Тип перестановки	Приборы и работы, которые принимают участие в перестановке		θ ($\theta > 0$)	Условие, при котором выполняется перестановка	Характеристики расписания σ^1, полученного в результате перестановки
	h	s			
1-1Б	$h \in I_\Delta(\sigma)$, $j_1 \in J_h$	$s \in I_R(\sigma)$, $j_2 \in J_s$	$p_{j_1} - p_{j_2}$	$\theta \le \Delta_h(\sigma)$, $\theta > R_s(\sigma)$	$\Delta_h(\sigma^1) = \Delta_h(\sigma) - \theta$, $\Delta_s(\sigma^1) = \theta - R_s(\sigma)$
1-2Б	$h \in I_\Delta(\sigma)$, $j_1 \in J_h$	$s \in I_R(\sigma)$, $j_2, j_3 \in J_s$	$p_{j_1} - (p_{j_2} + p_{j_3})$		
2-1Б	$h \in I_\Delta(\sigma)$, $j_1, j_2 \in J_h$	$s \in I_R(\sigma)$, $j_3 \in J_s$	$(p_{j_1} + p_{j_2}) - p_{j_3}$		
2-2Б	$h \in I_\Delta(\sigma)$, $j_1, j_2 \in J_h$	$s \in I_R(\sigma)$, $j_3, j_4 \in J_s$	$(p_{j_1} + p_{j_2}) - (p_{j_3} + p_{j_4})$		

Множество перестановок типа ***Б*** приводит к тому, что мощность множества I_Δ может увеличиться, но при этом величина максимального из выступов уменьшается. Следующее множество перестановок по условиям и последствиям является симметричным к множеству типа ***Б***.

Цель перестановок типа ***В***: уменьшение максимального из выступов за счет резерва прибора из множества I_R с возможным перераспределением резервов и выступов. В результате перестановок этого типа изменяется состав множеств I_R и I_Δ. В отличии от перестановок типа ***А*** и ***Б***, в перестановке этого типа, множество $L_s(\sigma)$ может быть пустым.

Условия выполнения перестановки типа ***В***:

$$\theta > 0\,;\ \theta > \Delta_h(\sigma);\ \theta \ge R_s(\sigma);$$
$$\Delta_h(\sigma) + R_s(\sigma) > \theta.$$

В результате перестановки получаем расписание σ^1, в котором:

$$R_h(\sigma^1) = \theta - \Delta_h(\sigma) > 0,$$
$$\Delta_s(\sigma^1) = \theta - R_s(\sigma).$$

При этом в новом расписании $h \in I_R(\sigma^1)$, $s \notin I_R(\sigma^1)$, $\Delta_h(\sigma) > \Delta_s(\sigma^1)$ (то есть расписание σ^1 не хуже, чем σ).

Характеристики перестановок типа ***В*** представлены в таблице 3.

Табл. 3. Свойства перестановок типа В

Тип пе-реста-новки	Приборы и работы, которые принимают участие в перестановке		θ ($\theta > 0$)	Условие, при котором выполняется перестановка	Характеристики расписания σ^1, полученного в результате перестановки
	h	s			
1-0В	$h \in I_\Delta(\sigma)$, $j_1 \in J_h$		p_{j_1}	$\theta > \Delta_h(\sigma)$, $\theta > R_s(\sigma)$	$R_h(\sigma^1) = \theta - \Delta_h(\sigma)$, $\Delta_s(\sigma^1) = \theta - R_s(\sigma)$
1-1В	$h \in I_\Delta(\sigma)$, $j_1 \in J_h$	$s \in I_R(\sigma)$, $j_2 \in J_s$	$p_{j_1} - p_{j_2}$	$\theta > \Delta_h(\sigma)$, $\theta \geq R_s(\sigma)$	
1-2В	$h \in I_\Delta(\sigma)$, $j_1 \in J_h$	$s \in I_R(\sigma)$, $j_2, j_3 \in J_s$	$p_{j_1} - (p_{j_2} + p_{j_3})$		
2-1В	$h \in I_\Delta(\sigma)$, $j_1, j_2 \in J_h$	$s \in I_R(\sigma)$, $j_3 \in J_s$	$(p_{j_1} + p_{j_2}) - p_{j_3}$		
2-2В	$h \in I_\Delta(\sigma)$, $j_1, j_2 \in J_h$	$s \in I_R(\sigma)$, $j_3, j_4 \in J_s$	$(p_{j_1} + p_{j_2}) - (p_{j_3} + p_{j_4})$		

В таблице 4 рассмотрены все возможные соотношения между величинами θ, Δ_h и R_s, а также указаны соответствующие им значения изменения частичной целевой функции (максимального из выступов приборов h и s).

Табл. 4. Влияние перестановок на значение целевой функции

Условия	Тип пере-ста-новки	Уменьшение значения целевой функции по приборам *h* и *s*
$\theta < \Delta_h$, $\theta \leq R_s$	***А***	θ
$\theta = \Delta_h$, $\theta \leq R_s$	***А***	Δ_h
$\theta \leq \Delta_h$, $\theta = R_s$ $\theta \leq \left\lfloor \frac{\Delta_h + R_s}{2} \right\rfloor$	***А***	θ
$\theta \leq \Delta_h$, $\theta > R_s$, $\theta \leq \left\lfloor \frac{\Delta_h + R_s}{2} \right\rfloor$	***Б***	θ
$\theta \leq \Delta_h$, $\theta > R_s$, $\theta \geq \left\lfloor \frac{\Delta_h + R_s}{2} \right\rfloor + 1$	***Б***	$\Delta_h + R_s - \theta$
$\theta > \Delta_h$, $\theta \geq R_s$	***В***	$\Delta_h + R_s - \theta$

Обобщим эти результаты: для выбранной пары приборов $h - s$ перестановки ***А***, ***Б*** и ***В*** приводят к такому уменьшению значения максимального из выступов этих приборов:

$$\Delta_h(\sigma) - \max\{\Delta_h(\sigma^1), \Delta_s(\sigma^1)\} = \\ = \min\{\Delta_h(\sigma), \theta, \Delta_h(\sigma) + R_s(\sigma) - \theta\}.$$

Когда нет возможности улучшения расписания перестановками ***А***, ***Б*** и ***В*** предлагается использовать перестановки типа ***Г***, которые уменьшают значение $\max_i \Delta_i(\sigma)$ за счет перераспределения выступов между приборами.

Условия выполнения перестановки типа ***Г***:

$$\theta > 0;\ \theta < \Delta_h(\sigma);\ \theta < \Delta_h(\sigma) - \Delta_s(\sigma).$$

В результате перестановки получаем расписание σ^1, в котором:

$$\Delta_h(\sigma^1) = \Delta_h(\sigma) - \theta,$$
$$\Delta_s(\sigma^1) = \Delta_s(\sigma) + \theta.$$

При этом для расписания σ^1 выполняется:

$$\sum_{i=1}^{m} \Delta_i(\sigma^1) = \sum_{i=1}^{m} \Delta_i(\sigma),$$

но $\max\{\Delta_h(\sigma^1), \Delta_s(\sigma^1)\} < \max\{\Delta_h(\sigma), \Delta_s(\sigma)\}$,

то есть расписание σ^1 не хуже, чем σ.

Для выбранной пары приборов $h - s$ в результате перестановок типа ***Г*** получаем такое уменьшение значения максимального из выступов этих приборов в двух расписаниях:

$$\Delta_h(\sigma) - \max\{\Delta_h(\sigma) - \theta, \Delta_h(\sigma) + \theta\}.$$

Характеристики перестановок ***Г*** разных подтипов представлены в таблице 5.

Табл. 5. Свойства перестановок типа Г

Тип пере-ста-новки	Приборы и работы, которые принимают участие в перестановке		θ ($\theta > 0$)	Условие, при котором выполняется перестановка	Характеристики расписания σ^1, полученного в результате перестановки
	h	s			
1-1Г	$h \in I_\Delta(\sigma)$, $j_1 \in J_h$	$s \in I_\Delta(\sigma) \cup I_0(\sigma)$, $j_2 \in J_s$	$p_{j_1} - p_{j_2}$	$\theta < \Delta_h(\sigma)$, $\theta < \Delta_h(\sigma) - \Delta_s(\sigma)$	$\Delta_h(\sigma^1) = \Delta_h(\sigma) - \theta$, $\Delta_s(\sigma^1) = \Delta_s(\sigma) + \theta$
1-2Г	$h \in I_\Delta(\sigma)$, $j_1 \in J_h$	$s \in I_\Delta(\sigma) \cup I_0(\sigma)$, $j_2, j_3 \in J_s$	$p_{j_1} - (p_{j_2} + p_{j_3})$		
2-1Г	$h \in I_\Delta(\sigma)$, $j_1, j_2 \in J_h$	$s \in I_\Delta(\sigma) \cup I_0(\sigma)$, $j_3 \in J_s$	$(p_{j_1} + p_{j_2}) - p_{j_3}$		
2-2Г	$h \in I_\Delta(\sigma)$, $j_1, j_2 \in J_h$	$s \in I_\Delta(\sigma) \cup I_0(\sigma)$, $j_3, j_4 \in J_s$	$(p_{j_1} + p_{j_2}) - (p_{j_3} + p_{j_4})$		

Эти перестановки положены в основу разработанной полиномиальной составляющей ПДС-алгоритма решения задачи.

6. ПДС-алгоритм решения задачи

На основании признаков оптимальности и разработанного множества перестановок построен алгоритм определения максимально позднего момента начала выполнения работ параллельными приборами с общим директивным сроком в допустимом расписании.

В работе [1] приведен жадный алгоритм построения расписания σ^0, которое принадлежит классу расписаний Ψ. Это расписание может быть взято в качестве начального.

Описание алгоритма

ШАГ 1 Построить начальное расписание σ^0, $\sigma = \sigma^0$.

ШАГ 2 Определить множества $I_\Delta(\sigma)$ и $I_R(\sigma)$.

ШАГ 3 *Проверка выполнения признаков оптимальности*

ЕСЛИ выполняется одна из признаков оптимальности

ТО перейти на **ШАГ 6** (σ – оптимальное расписание).

ШАГ 4 Определить прибор h, которому соответствует максимальное значение выступа: $\Delta_h(\sigma) = \max_{i \in I_\Delta} \Delta_i(\sigma)$.

ШАГ 5 Для прибора h, перебирая приборы $s \in I_R(\sigma)$ выполнить перестановку типа ***А***, ***Б*** или ***В***.

ЕСЛИ таких перестановок не нашлось,

ТО

5.1 Для прибора h перебирая все приборы $s \in I_0(\sigma) \cup I_\Delta(\sigma)$ выполнить перестановку типа **Г.**

5.2 ЕСЛИ таких перестановок не нашлось,

ТО перейти на **ШАГ 6**,

ІНАЧЕ перейти на **ШАГ 2.**

ІНАЧЕ перейти на **ШАГ 2.**

ШАГ 6 Определить максимально поздний момент запуска работ на выполнение в текущем расписании σ: $r(\sigma) = d - \left(C^* + \max_i \Delta_i(\sigma)\right)$.

КОНЕЦ АЛГОРИТМА

Возможные варианты реализации ШАГА 5:

1) находим первую перестановку, которая улучшает расписание, и выполняем ее;
2) перебираем все допустимые перестановки, находим среди них самую эффективную и выполняем ее.

Сложность алгоритма составляет $O(n^4 W)$, где $W = \sum_{i=1}^{n} p_i$.

Если для результирующего расписания σ выполняется один из признаков оптимальности, то это расписание является оптимальным. В противном случае целевая функция расписания σ отличается от ее оптимального значения на

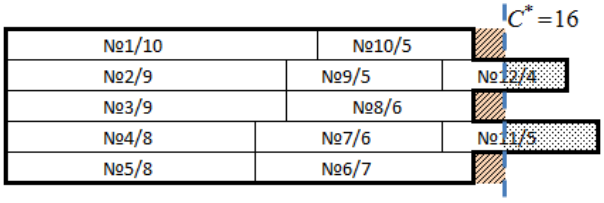
величину не более чем $\max_i \Delta_i(\sigma)$ (если $\delta = 0$), $\max_i \Delta_i(\sigma) - 1$ (если $\delta > 0$).

7. Пример применения алгоритма

Возьмем для примера задачу, в которой количество приборов $m = 5$, $n = 12$ работ, директивный срок $d = 80$.

Определим основные расчетные величины алгоритма: $C^* = 16$, $\delta = 2$. Так как $\delta > 0$, то в процессе применения алгоритма нашим ориентиром является признак оптимальности 2.

Начальное расписание σ^0, построенное по алгоритму А0 [1], представлено на рисунке 2.



Примечание: Запись №1/10 означает: №1 – номер работы / 10 – длительность выполнения этой работы.

Рис. 2. Начальное расписание σ^0

Для уменьшения количества возможных вариантов при решении данной задачи будем применять только перестановки подтипа 1-1.

Итерация №1

Для начального расписания σ^0 имеем: $C_1 = 15$, $C_2 = 18$, $C_3 = 15$, $C_4 = 19$, $C_5 = 15$;

$I_0(\sigma^0) = \{0\}$;

$I_\Delta(\sigma^0) = \{2;4\}$; $\Delta_2(\sigma^0) = 2$, $\Delta_4(\sigma^0) = 3$;

$I_R(\sigma^0) = \{1;3;5\}$; $R_1(\sigma^0) = 1$, $R_3(\sigma^0) = 1$, $R_5(\sigma^0) = 1$.

Определим приборы $h \in I_\Delta(\sigma^0)$ и $s \in I_R(\sigma^0)$: $h = 4$ (прибор с максимальным значением выступа), $s = 5$ (один из приборов с максимальным значением резерва).

Определим, каким парам работ приборов 4 и 5 соответствуют допустимые перестановки. Для этого сначала выделим те пары работ, для которых $\theta = p_{j_1} - p_{j_2} > 0$, к ним относится пара работ 4-6. А потом проанализируем, как соотносятся величины θ, Δ_4 и R_5, то есть к какому из типов ***А***, ***Б*** или ***В*** относятся потенциальные перестановки. В таблице 6 приведены перестановки, которые являются допустимыми для пары приборов 4-5.

Табл. 6. Параметры допустимых перестановок для пары приборов 4-5

Пара работ	Δ_4	R_5	θ	$\Delta_4 + R_5 - \theta$	Величина улучшения	Тип перестановки
4-6	3	1	1	3	1	***А***

На рисунке 3 изображено расписание σ^1, полученное в результате этой перестановки.

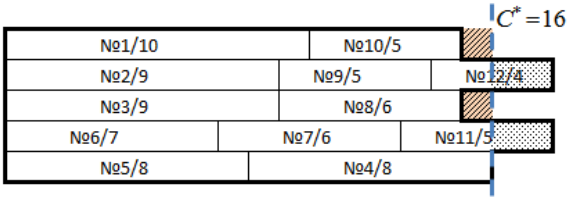


Рис. 3. Расписание σ^1

Для расписания σ^1 не выполняется признак оптимальности 2, продолжаем работу алгоритма.

Итерация №2

Для текущего расписания σ^1 имеем: $C_1 = 15$, $C_2 = 18$, $C_3 = 15$, $C_4 = 18$, $C_5 = 16$;

$I_0(\sigma^1) = \{5\}$;

$I_\Delta(\sigma^1) = \{2;4\}$; $\Delta_2(\sigma^1) = 2$, $\Delta_4(\sigma^1) = 2$;

$I_R(\sigma^1) = \{1;3\}$; $R_1(\sigma^1) = 1$, $R_3(\sigma^1) = 1$.

Определим приборы $h \in I_\Delta(\sigma^1)$ и $s \in I_R(\sigma^1)$: $h = 4$ (один из приборов с максимальным значением выступа), $s = 1$ (один из приборов с максимальным значением резерва). Для выбранных приборов допустимыми являются перестановки таких пар работ: 6-10, 7-10. В таблице 7 приведены параметры этих перестановок.

Табл. 7. Параметры допустимых перестановок для пары приборов 4-1

Пара работ	Δ_4	R_1	θ	$\Delta_4 + R_1 - \theta$	Величина улучшения	Тип перестановки
6-10	2	1	2	1	1	***Б***
7-10	2	1	1	2	1	***А***

Последствия всех перестановок одинаковы, выберем перестановку работ 6-10 (первую в списке) и выполним ее. На рисунке 4 изображено расписание σ^2, полученное в результате этой перестановки.

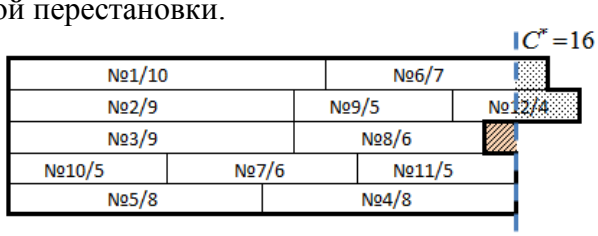


Рис. 4. Расписание σ^2

Для расписания σ^2 не выполняется признак оптимальности 2, продолжаем работу алгоритма.

Итерация №3

Для текущего расписания σ^2 имеем: $C_1 = 17$, $C_2 = 18$, $C_3 = 15$, $C_4 = 16$, $C_5 = 16$;

$I_0(\sigma^2) = \{4;5\}$;

$I_\Delta(\sigma^2) = \{1;2\}$; $\Delta_1(\sigma^2) = 1$, $\Delta_2(\sigma^2) = 2$;

$I_R(\sigma^2) = \{3\}$; $R_3(\sigma^2) = 1$.

Определим приборы $h \in I_\Delta(\sigma^2)$ и $s \in I_R(\sigma^2)$: $h = 2$ (прибор с максимальным значением выступа, $s = 3$ (единственный прибор с ненулевым значением резерва). Для выбранной пары приборов допустимых перестановок типов ***А***, ***Б*** и ***В*** нет. Переходим к применению перестановок типа ***Г***. Определим прибор $s \in I_0(\sigma^2) \cup I_\Delta(\sigma^2)$: $s = 5$ (прибор из множества $I_0(\sigma^2)$, отметим, что для пары приборов 2-4 допустимых перестановок нет).

Для выбранных приборов допустимыми являются перестановки таких пар работ: 2-5, 2-4. В таблице 8 приведены параметры этих перестановок.

Табл. 8. Параметры допустимых перестановок для пары приборов 2-5

Пара работ	Δ_2	Δ_5	θ	$\Delta_2 + \Delta_5 - \theta$	Величина улучшения
2-5	2	0	1	1	1
2-4	2	0	1	1	1

Последствия всех перестановок одинаковы, выберем перестановку работ 2-5 (первую в списке) и выполним ее. На рисунке 5 изображено расписание σ^3, полученное в результате перестановки.

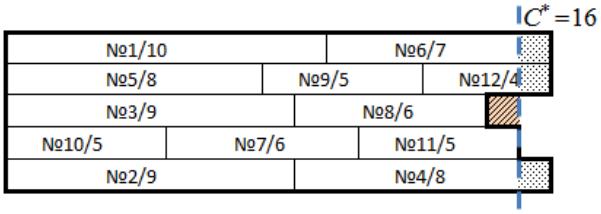


Рис. 5. Расписание σ^3

Полученное расписание является оптимальным по признаку оптимальности 2. Определим для σ^3 момент запуска, при котором все работы будут выполнены до директивного срока:

$$r(\sigma) = d - \left(C^* + \max_i \Delta_i(\sigma^3)\right) = 80 - (16+1) = 63.$$

7. Экспериментальное исследование алгоритма

Классификация задач

Качество результата алгоритма (степень близости к оптимуму) зависит от набора входных данных задачи. Интуитивно понятно, что для систем «на каждом из приборов выполняется достаточно большое количество работ, продолжительности которых не очень сильно отличаются друг от друга» и «на каждом из приборов выполняется небольшое количество работ, продолжительности которых сильно отличаются друг от друга» время работы и качество результатов алгоритма будут различными.

Для исследования того, как влияют начальные условия на получаемые результаты алгоритма, выделим следующие признаки, по которым задачи можно классифицировать:

– длительности работ;

– дисперсия длительностей работ.

Длительность работ

Пусть $\overline{p}$ – среднее значение длительностей работ. Величина $\frac{C^*}{\overline{p}}$ характеризует среднее количество работ, выполняющихся на каждом из приборов. Для разделения задач на классы введем величины G_L, G_M, G_S ($G_L < G_M < G_S$). В зависимости от соотношения величин $\frac{C^*}{\overline{p}}$ *индивидуальные задачи* (задачи с конкретно определенными значениями $m, n, d, p_i, i = \overline{1,n}$) будем относить к одной из категорий:

– категория S – задачи с "короткими" работами; к этой категории будем относить задачи, в которых:

$$G_M < \frac{C^*}{\overline{p}} \le G_S;$$

– категория M – системы с "средними" работами; к этой категории будем относить системы, в которых:

$$G_L < \frac{C^*}{\overline{p}} \le G_M;$$

– категория L – системы с "длинными" работами; к этой категории будем относить системы, в которых:

$$\frac{C^*}{\overline{p}} \le G_L.$$

Значения граничных величин G_L, G_M, G_S, характеризующих категории (далее *классифи-*

кационные параметры), устанавливаются экспериментально (их значения определяются таким образом, чтобы рассматриваемые классы задач статистически отличались).

Значение дисперсии длительностей работ

При моделировании множества работ будем предполагать, что длительности работ являются случайными величинами и имеют равномерное распределение. Это объясняется тем, что на практике для продолжительности работ обычно существуют верхняя и нижняя границы, в то время как случайные величины, имеющие другие распространенные распределения вероятностей (нормальный, экспоненциальный, Эрланга и т.д.) могут принимать отрицательные значения и/или значения, которые стремятся к бесконечности.

В зависимости от категории, к которой принадлежит задача по среднему значению длин работ, имеем значение $\overline{p}$. Но длительности работ в задаче могут быть как близкими к $\overline{p}$, так и значительно отличаться. Для моделирования этой ситуации введем параметр Q (определяется как доля от $\overline{p}$). Он характеризует пределы интервала, на котором равномерно распределена случайная величина P, реализациями которой являются длительности работ в системе. То есть, все возможные значения p длительностей работ будут лежать в промежутке:

$$p_{\min} = \overline{p}\cdot(1-Q) \le p \le \overline{p}\cdot(1+Q) = p_{\max}.$$

Зная пределы интервала $[a,b]$, где $a,b \in R,\ a < b$, на котором равномерно распределена случайная величина X, можно вычислить для нее значение дисперсии:

$$D[X] = \frac{(b-a)^2}{12}.$$

В нашем случае:

$$D[P] = \frac{(p_{\max} - p_{\min})^2}{12} = \frac{(2\overline{p}\cdot Q)^2}{12} = \frac{(\overline{p})^2\cdot Q^2}{3}.$$

В зависимости от величины Q и, соответственно, значения дисперсии, индивидуальные задачи будут относиться к одной из категорий:

– категория S – задачи с малым рассеиванием; к этой категории будем относить задачи, в которых:

$$Q \le V_S;$$

– категория M – задачи со средним рассеиванием; к этой категории будем относить задачи, в которых:

$$V_S < Q \le V_M;$$

– категория L – задачи с большим рассеиванием; к этой категории будем относить задачи, в которых:

$$V_M < Q.$$

Для классификации задач далее будем использовать запись T_P / T_Q:

– T_P характеризует продолжительность работ, $T_P \in \{S, M, L\}$;

– T_Q характеризует дисперсию длительностей работ, $T_Q \in \{S, M, L\}$.

Например, если $G_S = 200$, $G_M = 100$, $G_L = 10$, $V_S = 0{,}1$, $V_M = 0{,}5$, то к классу *M/S* относятся задачи, в которых при некотором заданном среднем значении длительностей работ $\overline{p}$ средняя продолжительность выполнения всех работ одним прибором при равномерной загрузке C^* находится в промежутке:

$$11\overline{p} \le C^* \le 100\overline{p},$$

(то есть на приборе в среднем выполняется от 11 до 100 работ), а длины всех работ равномерно распределены на интервале:

$$0{,}9\overline{p} \le p \le 1{,}1\overline{p}.$$

Результаты экспериментов

Для проведения экспериментов был разработан программный продукт, включающий в себя два модуля:

– модуль - генератор индивидуальных задач;

– модуль, в котором реализован ПДС-алгоритм.

Первый модуль позволяет генерировать индивидуальные задачи (заданных классов), для которых известно оптимальное расписание. Это позволяет для каждого результирующего расписания оценить степень достижения оптимума. Для классификации генерируемых задач были использованы такие параметры классификации: $G_S = 200$, $G_M = 100$, $G_L = 10$, $V_S = 0{,}1$, $V_M = 0{,}5$. Всего исследовано 9 классов задач (S/S, S/M, S/L ..., L/L). Для каждого класса генерировалось достаточно большое количество индивидуальных задач (это количество подбиралось таким, что его небольшое изменение в ту или иную сторону, не приводило к изменению статистических характеристик алгоритма). При исследовании алгоритма анализировались:

– точность алгоритма (процент задач, в которых было достигнуто оптимальное расписание);

– степень достижения оптимума (относительное отклонение значения целевой функции в полученном расписании от известного оптимального значения);

– время нахождения результирующего расписания.

При этом исследовались такие варианты алгоритма: полный (с использованием всех подтипов перестановок) и упрощенный (с использованием только подтипов **1-1**). Как и следовало ожидать, алгоритм проявил себя по-разному для разных классов задач. На рисунке 6 представлено влияние числа работ в системе на точность алгоритма.

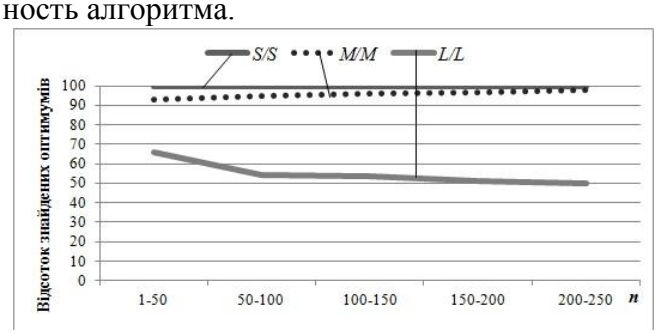


Рис. 6. Зависимость точности алгоритма от количества работ *n*

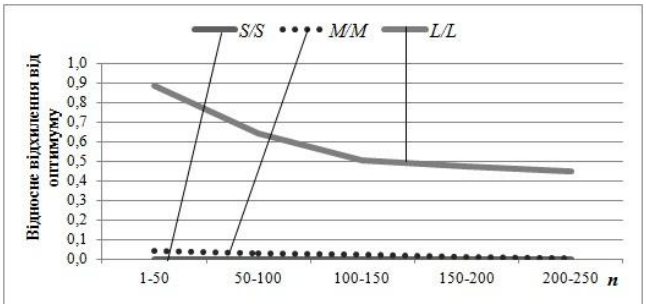


Рис. 7. Зависимость степени достижения оптимума от количества работ *n*

На рисунке 7 представлено влияние числа работ в системе на степень достижения оптимума (для упрощенного варианта алгоритма).

На рисунке 8 показано влияние числа работ в системе на время работы алгоритма, использующего только перестановки 1-1 (расчеты производились на процессоре с тактовой частотой 2.10 GHz).

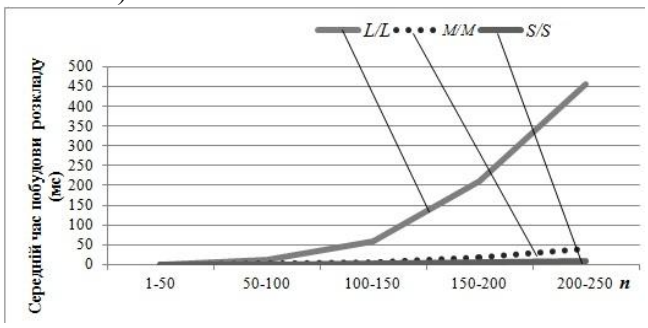


Рис. 8. Зависимость времени работы (мс) алгоритма от количества работ *n*

Использование других подтипов перестановок улучшает характеристики алгоритма (точность и степень достижения оптимума), но увеличивает время работы.

8. Заключение

Исследованы свойства задачи составления допустимого расписания с максимально поздним моментом запуска выполнения идентичными параллельными приборами работ с общим директивным сроком. Применяя методологию построения ПДС-алгоритмов, разработаны признаки оптимальности расписаний. На основе этих признаков определено множество перестановок, которые позволяют последовательно улучшать значение критерия. Разработана полиномиальная составляющая ПДС-алгоритма решения задачи. Приведен пример применения алгоритма. Проведены экспериментальные исследования алгоритма, которые показали его достаточно высокую точность.

Список литературы

1. Павлов, О.А. Поліноміальна складова ПДС-алгоритму розв’язання однієї задачі теорії розкладів / О. А. Павлов, О.Г. Жданова, О.Б. Місюра, М.О. Сперкач// Технологический аудит и резервы производства, 2013. — №6/3 (14). — С.47—52.
2. Павлов, А.А. Признаки оптимальности допустимых решений труднорешаемых задач комбинаторной оптимизации / А. А. Павлов// Вісник НТУУ “КПІ”. Серія «Інформатика, управління та обчислювальна техніка». — К.: “ВЕК+”, 2013. – №59. — С.4—12.
3. Згуровский, М. З. Принятие решений в сетевых системах с ограниченными ресурсами [Текст]: монография / М. З. Згуровский, А. А. Павлов. – К.: Наукова думка, 2010.– 573 с.

УДК 004.056.53

МАРКОВСЬКИЙ О.П.,
НЕВДАЩЕНКО М.В.,
БІЛАШЕВСЬКА А.М.

ЗАХИЩЕНА РЕАЛІЗАЦІЯ ФІЛЬТРАЦІЇ ЗОБРАЖЕНЬ В GRID-СИСТЕМАХ

В статті пропонуються методи захищеної реалізації медіанної та середньоарифметичної фільтрації на віддалених процесорних засобах GRID-систем. Методи забезпечують захист від доступу до зображень при їх передачі та під час їх обробки на віддалених комп'ютерних системах. Метод базується на використанні інтервального шифрування для медіанної фільтрації і використання адитивного маскування точок зображення для середньоарифметичної фільтрації. Детально описані процедури шифрування зображень і їх дешифрування після фільтрації. Наведено числові приклади шифрування та дешифрування зображень для обох типів фільтрації.

This paper proposes a method for protected implementation of image digital median and averaging filtering at remote processors of GRID-systems. Proposed methods ensures protection of image against unauthorized access while image network transmission and while image processing on remote computer systems. The methods for median filtering image protection is based on interval encryption and developed method for averaging filtering image protection is based on using of additional masking. The proposed procedure for image encryption and decryption are described in details. A numerical example for procedure image encryption and decryption for both type of image filtering are given.

Вступ

Спіралевидний характер розвитку комп'ютерних технологій призвів на початку нового тисячоліття до повернення на якісно новому рівні до колективного використання обчислювальних потужностей: віддаленої обробки задач користувачів на потужних багатопроцесорних комп'ютерних системах.

Динамічний прогрес способів телекомунікації та мережевих технологій дозволив широкому колу різнорідних користувачів прямо використовувати значні обчислювальні ресурси сучасних кластерних систем для вирішення наукових та прикладних задач. Це суттєво прискорило обробку прикладних задач, допомогло підвищити рівень їх розв'язування, а також, як наслідок, залучити до комп'ютерної обробки більш широке коло нових задач, які, в силу їх значної ресурсоємності, не могли бути вирішені на персональних комп'ютерах.

Технології віддаленого надання розподілених обчислювальних ресурсів активно розвиваються в рамках GRID-систем [1]. Особливістю таких систем є використання рознесених у просторі ресурсів. При цьому GRID-система бере на себе функцію забезпечення ефективного використання комп'ютерних систем шляхом динамічного розподілення потоку задач користувачів. Відповідно, звертаючись до GRID-системи, користувач в принципі не знає на якій з комп'ютерних систем буде розв'язуватися його задача. Відкритість GRID-систем зумовлює їх основний недолік – незахищеність від несанкціонованого доступу до їх задач користувачів. В сучасних умовах доступ до даних та задач користувачів в GRID-системах суттєво обмежує коло їх застосування і як результат помітно знижує ефективність віддаленого надання обчислювальних ресурсів [2].

Однією з найбільш поширених прикладних задач є обробка зображень. Висока ресурсоємність процедур обробки зображень, що містять мільйони пікселів диктує доцільність використання потужних багатопроцесорних комп'ютерних систем. Разом з цим, характер обмежень доступу до зображення не дозволяє, в більшості випадків, їх обробку у відкритих системах, в тому числі GRID-системах. Відповідно, виникає практична потреба в організації захищеної обробки зображень в сучасних відкритих розподілених комп'ютерних системах великої потужності [3].

Таким чином, наукова задача створення методів і засобів захищеної обробки зображень і, зокрема їх фільтрації, в GRID-системах є актуальною та важливою для практики.

Аналіз існуючих методів фільтрації зображень та її захищеної реалізації

Одним з основних процедур обробки зображень є їх фільтрація, яка дозволяє позбутися імпульсних завад [3]. Будь-яка фільтрація виконується над зображенням, яке можна подати

в наступному вигляді: $I= \{f(x,y);\ x \in [1;l],\ y \in [1;h]\}$ де I– зображення; $f(x,y)$– функція значення інтенсивності кольору в точці з координатами(x,y); l та h– відповідно ширина та висота зображення.

Якщо вважати, що будь-яке зображення має вади, то функцію $f(x,y)$ буде сумою значень інтенсивностей оригінального зображення $g(x,y)$ та вад $v(x,y)$, які виникли під час формування зображення: $f(x,y)= g(x,y)+ v(x,y)$.

Отже, метою фільтрації є зменшення значення інтенсивності вад $v(x,y)$.

Найбільшого поширення вважаються медіанна та середньоарифметична фільтрації [3]. Операція полягає в тому, що зображення сканується апертурою, що містить непарну кількість рядків та стовпців. В процесі сканування центральний елемент замінюється медіаною або середнім арифметичним точок апертури. Операція починається з верхнього лівого кутка і , поступово зміщуючись, охоплює весь об'єм зображення.

Операція медіанної фільтрації полягає в тому, що зображення сканується квадратною апертурою з непарною кількістю рядків та стовбців. Під час сканування значення інтенсивності точок апертури сортуються та серед ним вибирається елемент, що стоїть посередині, тобто являється медіаною. Значення інтенсивності замінюється на значення медіани і апертура переходить до обробки наступної точки. Обробка розпочинається з лівого верхнього кута, далі апертура переміщується по рядку, а в кінці рядка переходить на першу точку наступного рядка.

Приклад на рисунку 1 ілюструє принцип медіанної фільтрації. Для зручності взято за приклад зображення 4x4 із застосуванням апертури 3x3. Зліва показано початкове зображення. Далі зліва направо приведена послідовність зміни зображення в процесі його медіанної фільтрації. На малюнку виділено апертури, які використовуються в процесі фільтрації.

5	3	5	3
2	4	1	3
5	3	4	2
1	2	2	1

1 2 3 3 **4** 4 5 5 5

5	3	5	3
2	4	1	3
5	3	4	2
1	2	2	1

1 2 3 3 **3** 3 4 4 5

5	3	5	3
2	4	3	3
5	3	4	2
1	2	2	1

1 2 2 2 **3** 3 4 4 5

5	3	5	3
2	4	3	3
5	3	4	2
1	2	2	1

1 2 2 2 **3** 3 3 4 4

5	3	5	3
2	4	3	3
5	3	3	2
1	2	2	1

Рис.1. Приклад медіанної фільтрації

Операція середньоарифметичної фільтрації полягає в тому, що зображення сканується квадратною апертурою з непарною кількістю рядків та стовбців. Під час сканування знаходиться середнє значення інтенсивності точок апертури, яке замінює значення інтенсивності точки. Обробка розпочинається з лівого верхнього кута, далі апертура переміщується по рядку, а в кінці рядка переходить на першу точку наступного рядка.

Приклад на рисунку 2 ілюструє принцип середньоарифметичної фільтрації. За приклад взято зображення 4x4 із застосуванням апертури 3x3. Зліва показано початкове зображення. Далі, зліва направо приведена послідовність зміни зображення в процесі його середньо арифметичної фільтрації. На малюнку виділено апертури, які використовуються в процесі фільтрації.

5	3	5	3
2	4	1	3
5	3	4	2
1	2	2	1

5+3+5+2+4+1+5+3+4)/9=3,56

5	3	5	3
2	4	1	3
5	3	4	2
1	2	2	1

(3+5+3+4+1+3+3+4+2)/9=3,1

5	3	5	3
2	4	3	3
5	3	4	2
1	2	2	1

(2+4+3+5+3+4+1+2+3)/9=3

5	3	5	3
2	4	1	3
5	3	4	2
1	2	2	1

(4+3+3+3+4+2+2+2+1)/9=2,7

5	3	5	3
2	4	1	3
5	3	3	2
1	2	2	1

Рис.2. Приклад середньоарифметичної фільтрації

Описані вище методи фільтрації зображень не орієнтовані на захищену реалізацію на відкритих віддалених комп'ютерних системах. Разом з тим, як зазначалося вище, така задача з появою і початком широкого застосування GRID-систем набуває дедалі більшої актуальності.

До теперішнього часу, на відміну від захисту інформації при її передачі та зберіганні, проблема захисту даних в процесі їх безпосередньої обробки в загальному вигляді не вирішена [4]. Якщо в процесі обробки над даними а виконується функціональне перетворення $s =f(a)$ то захищена реалізація передбачає попереднє

функціональне перетворення $b=\varphi(a,m)$, де m-секретний ключ і передачу для безпосередньої обробки коду b. Головною проблемою захищеної реалізації є одержання функції відновлення $\psi(x,m)$, $s=\psi(f(\varphi(a,m)),m)=f(a)$. При цьому обчислювальна складність функціональних перетворень φ та ψ має бути суттєво меншою в порівнянні з f.

Практично, до теперішнього часу для кожної процедури (функції f) віддаленої обробки даних вибирається свої перетворення φ та ψ, які забезпечують вказані вище умови ефективної захищеної реалізації.

Ціллю досліджень є розробка ефективних методів захисту даних зображень безпосередньо під час їх віддаленої фільтрації шляхом отримання процедур шифрування точок зображень та дешифрування результатів фільтрації.

Захищена реалізація медіанної фільтрації

Оскільки в основі медіанної фільтрації лежить операція порівняння, то при шифруванні має зберігатись відносний порядок між точками зображення : якщо для точок зображення a_i і a_j виконується $a_i > a_j$, то і для зашифрованих представлень відповідних точок b_i і b_j має так само виконуватись $b_i > b_j$. Для того, щоб ця умова виконувалась, пропонується використати інтервальне шифрування. Основою такого методу є те, що розрядність d представлення кодів точок зображення менша за розрядність u зашифрованих представлень цих точок. При цьому, якщо кількість можливих значень точок становить r, то діапазон 0 до 2^u-1 можливих значень зашифрованих точок випадковим чином розділяється на r інтервалів з границями : $g_1, g_2, \ldots, g_r$, причому $g_1 < g_2 < \ldots < g_r$. Якщо позначити точки зображення через $a_1, a_2, \ldots, a_N$, де $N=h \cdot l$- кількість точок зображення, то запропонований спосіб захищеної реалізації медіанної фільтрації на відкритих GRID-системах полягає в наступному.

1) Створюється впорядкований масив, присутніх в зображенні значень точок, що a'_1, a'_2, a'_3, a'_4, ... a'_k , де k – загальна кількість можливих інтенсивностей зображення(для прикладу, показаного на рис. 1, k =4).

2) Обирається інтервал X=(0, 2^u-1), d'-розрядність представлення точок зображення у шифрованому вигляді.

3) Інтервал X розбивається випадковим чином на k проміжків: Xj, де j=1,k . Кожному значенню a'_j відповідає інтервал X_j .

4) Для кожної точкою виконується операція її шифрування: $\forall\ i \in \{1 \ldots N\}\, b_i = r_i$. Де r_i – випадкове число, що належить інтервалу Xj : $a_i = a'_j$.

5) Масив чисел b_1, b_2, b_3, b_4, ... b_N передається на обробку системі, яка потім повертає масив значення точок відфільтрованого шифрованого зображення q_1, q_2, q_3, q_4, ... q_N .

6) Користувач виконує дешифрацію точок $q_1, q_2, q_3, q_4, q_5, \ldots, q_N$ отриманого зображення: $\forall i \in \{1,2,\ldots,N\}$: $w_i = a'_j$, який відповідає інтервалу Xj так, що $q_i \in Xj$,де w_i - реальні відфільтровані значення .

Запропонований спосіб шифрування цілком відповідає зазначеним вище теоретичним умовам. Впорядкувавши вибірку кольорів і поставивши їм у відповідність проміжки поступово розбитого інтервалу, виконується головна вимога – збереження порядку числа, що робить можливим порівняння шифрованих аналогічне порівнянню реальних значень. Це забезпечує коректність дешифрування.

Запропонований спосіб захищеної реалізації медіанної фільтрації може бути ілюстровано наступним прикладом фільтрації для прикладу, зображеного на рис. 1. Спершу впорядкується вибірка можливих кольорів:

$a'_1 = 1$, $a'_2 = 2$, $a'_3 = 3$, $a'_4 = 4$, $a'_4 = 5$.

Оскільки розрядність реальних чисел – 3, то для шифрованих можна обрати будь-яке більше, з розрахунком на те, що більший проміжок для позначення числа дає більший захист за рахунок збільшення діапазону випадкових варіантів. Для розглянутого прикладу було обрано розрядність u = 7. Тоді основний інтервал – X=(0, 127).

Співвідношення між можливими значеннями та випадковими інтервалами ілюструє таблиця 1, наведена нижче:

Табл1. Розподіл можливих значень за інтервалами

a'_j	X_j
1	0...26
2	27...53
3	54...84
4	85...111
5	112...127

Приклад на рисунку 3 ілюструє принцип фільтрації зашифрованих значень. Зліва показано початкове зображення, що утворюється за допомогою обраного методу шифрування. Далі зліва направо приведена послідовність зміни зображення в процесі його медіанної фільтрації. На малюнку виділено апертури, які використовуються в процесі фільтрації.

115	55	113	82
46	86	0	57
126	64	94	34
13	30	43	23

0 46 55 64 **86** 94 113 115 126

115	55	113	82
46	86	0	57
126	64	94	34
13	30	43	23

0 34 55 57 **64** 82 86 94 113

115	55	113	82
46	86	64	57
126	64	94	34
13	30	43	23

13 30 43 46 **64** 64 86 94 126

115	55	113	82
46	86	64	57
126	64	94	34
13	30	43	23

23 30 34 43 **57** 64 64 86 94

115	55	113	82
46	86	64	57
126	64	43	34
13	30	43	23

Рис.3. Приклад захищеної медіанної фільтрації

Дешифровані медіани:

1) 86-4
2) 64-3
3) 64-3
4) 57-3

Одержані результати співпадають з представленими рис.1 результатами звичайної реалізації медіанної фільтрації.

Метод захищеної реалізації середньоарифметичної фільтрації.

В основі середньоарифметичної фільтрації лежить операція обчислення середнього арифметичного точок поточної апертури. Ця операція включає арифметичне додавання та ділення. Найпростішим способом шифрування операндів арифметичного додавання є використання адитивного маскування, тобто додавання до кожного із операнду випадкового цілого, кратного певному модулю m. Зняття такої маски може бути доволі просто виконане шляхом знаходження залишку від ділення суми на модуль m, за умови, що модуль m більший за суму. Зняття адитивної захисної маски після операції ділення можливе за умови, що ділене буде кратним як подільнику, так і модулю m.

З урахуванням наведеного, спосіб шифрування, що пропонується зводиться до накладання на точки реального зображення адитивної складової кратної простому числу m, яке за значенням більше будь-якої точки зображення. Якщо позначити значення n реальних точок апертури як $a_1, a_2, a_3, a_4, \ldots a_n$, а через $b_1, b_2, b_3, \ldots b_n$ - відповідно захищені від читання точки цієї апертури, то формально процес шифрування точок апертури можна представити у вигляді: $\forall j \in \{1,..,n\}$: $b_j = a_j + r_j \cdot m$, де r_j – випадкове ціле число. Для того, щоб результат фільтрації q, що формується як середнє арифметичне значень $b_1, b_2, b_3, \ldots b_n$, міг бути відтворений користувачем, його значення також має бути сумою середнього арифметичного s точок $a_1, a_2, a_3, a_4, \ldots a_n$ та адитивної компоненти кратної m: $q = s + d \cdot m$, де d - ціле число. Таким чином, значення q фактично обчислюється у вигляді:

$$q = \frac{1}{n} \cdot \sum_{j=1}^{n} b_j = \frac{1}{n} \cdot \sum_{j=1}^{n} a_j + \frac{m}{n} \cdot \sum_{j=1}^{n} r_j = s + d \cdot m \qquad (1)$$

Відновлення фільтрованої точки s початкового зображення по зашифрованому коду q, як слідує з (1) реалізується через обчислення залишку від ділення q на m: $s = q \bmod m$. Коректність такого перетворення базується на тому, що $s<m$ (в силу того, що середнє арифметичне менших за m також менше за m), та на тому, що d - ціле.

Так як d має бути цілим, то з (1) слідує, що сума $r_1+r_2+\ldots+r_n$ має бути кратною числу n точок апертури. В процесі фільтрації обчислене значення q замінює центральну $(n+1)/2$-ту точку $b_{(n+1)/2}$ апертури. Так як випадкові коефіцієнти генеруються користувачем до передачі зображення в GRID-систему з урахуванням зазначеної вище умови кратності суми коефіцієнтів апертури її розміру, то для того, щоб при заміні центральної точки апертури середнім значенням q ця умова не порушувалась, коефіцієнт $r_{(n+1)/2}$ центральної. $(n+1)/2$-ї точки має дорівнювати середньому арифметичному коефіцієнтів точок апертури:

$$r_{(n+1)/2} = \frac{1}{n} \cdot \sum_{j=1}^{n} r_j \qquad (2)$$

Таким чином, для захищеної реалізації середньоарифметичної фільтрації на віддалених комп'ютерних системах зображення, що містить N точок: $a_1, a_2, \ldots, a_N$, пропонується спосіб, який зводиться до наступної послідовності дій користувача.

1) Генеруються випадковим чином цілі числа $r_1, r_2, \ldots, r_N$, по кількості N точок зображення

таким чином, щоб для кожної апертури з n точок їх сума була кратна n і виконувалася умова (2). Вказані дії не прив'язані до конкретного зображення і можуть виконуватися заздалегідь.

2) Для зображення обирається модуль m, що є простим числом, більшим, за значення будь-якої точки зображення: $\forall i \in \{1,2,...,N\}$: $m > a_i$.

3) Для кожної з точок зображення виконується її шифрування накладанням адитивної маски: $\forall i \in \{1,2,...,N\}$: $b_i = a_i + r_i \cdot m$.

4) Зашифровані точки $b_1, b_2, ..., b_N$ зображення передаються на GRID-систему для віддаленої фільтрації.

5) В GRID-системі виконується віддалена фільтрація на вільних обчислювальних потужностях і повертається користувачеві в вигляді кодів $s_1, s_2, ..., s_N$ точок відфільтрованого зображення.

6) Користувач виконує дешифрацію точок $s_1, s_2, ..., s_N$ отриманого зображення шляхом обчислення залишку від ділення: $\forall i \in \{1,2,...,N\}$: $v_i = s_i \bmod m$.

Запропонований метод захищеної реалізації середньоарифметичної фільтрації в GRID-системах може бути ілюстровано прикладом його застосування для фільтрації зображення на рис.1.

Згідно в викладеною вище методикою, випадковим чином формується N=16 цілих коефіцієнтів $r_1, r_2, ..., r_{16}$ таких, що їх сума в рамках будь-якої з 4-х апертур розміром 3 х 3 ділиться на 9 і при цьому частка від ділення дорівнює коефіцієнту центрального елементу апертури. Сформовані випадкові коефіцієнти для прикладу наведені в таблиці 1. Легко пересвідчитись, що, наприклад, для правої нижньої апертури, що включає точки 6,7,8, 10,11,12,14,15 і 16, $r_6 + r_7 + r_8 + r_{10} + r_{11} + r_{12} + r_{14} + r_{15} + r_{16} =$ 5+6+5+8+9+10+5+5+28= 81; тобто сума ділиться на 9 і коефіцієнт r_{11} центрального елементу апертури дорівнює середньому значенню її коефіцієнтів: $r_{11} = 9 = 81/9$.

Відповідно до п.2 запропонованої методики для заданого зображення випадковим чином обирається модуль m: більше за будь-яке значення просте число , за значенням більше ніж будь-яка точка зображення. Для прикладу, що розглядається m=7. Таблиця 2 відображає шифрування значень точок зображення.

Приклад на рисунку 4 ілюструє принцип фільтрації зашифрованих значень. Зліва показано початкове зображення, що утворюється за допомогою обраного методу шифрування, та поступове обчислення середніх значень. Далі зліва направо приведена послідовність зміни зображення в процесі його середньоарифметичної фільтрації. На малюнку виділено апертури, які використовуються в процесі фільтрації.

Номер точки	a_i	r_i	b_i
1	5	1	12
2	3	2	17
3	5	3	26
4	3	6	45
5	2	4	30
6	4	5	39
7	1	6	43
8	3	5	38
9	5	7	54
10	3	8	59
11	4	9	67
12	2	10	72
13	1	23	162
14	2	5	37
15	2	5	37
16	1	28	197

Табл2. Шифрування точок

12	17	26	45
30	39	43	38
54	59	67	72
162	37	37	197

12	17	26	45
30	38,5	43	38
54	59	67	72
162	37	37	197

12	17	26	45
30	38,5	45	38
54	59	67	72
162	37	37	197

12	17	26	45
30	38,5	45	38
54	58,8	67	72
162	37	37	197

115	55	113	82
46	86	0	52
126	64	43	34
13	30	43	23

(12+17+26+30+39++43+54+59+67)/9=38,56
(17+26+45+38,5+43+38+59+67+72)/9=45
(30+38,5+45+54+59+67+162+37+37)/9=58,8
(38,5+45+38+58,8+67+72+37+37+197)/9=65,59

Рис.4. Приклад захищеної середньоарифметичної фільтрації

В результаті фільтрації, від системи користувач отримує наступні значення відфільтрованих точок 4-х апертур: s_6=38,5; s_7=45; s_{10}=58,8; s_{11}=65,59. Інші точки зображення повертаються системою в незмінному вигляді. Дешифрування змінений точок, згідно з викладеним вище, виконується користувачем наступним чином.

$v_6 = s_6 \, mod \, 7 = 38.56 \, mod \, 7 = 3.56 = 4$;

$v_7=s_7$ *mod* 7=45 *mod* 7=3;

$v_{10}=s_{10}$ *mod* 7=58.8 *mod* 7=2.8=3;

$v_{11}=s_{11}$ *mod* 7=65.59 *mod* 7=2.59=3;

Порівняння наведених значень з прикладом на рис.2 свідчить проте, що результат захищеної за запропонованім методом фільтрації повністю співпадає з результатами незахищеної середньоарифметичної фільтрації.

Аналіз ефективності

Основною перевагою запропонованих методів є можливість виконання масових операцій обробки зображень на відкритих GRID-системах з захистом самого зображення від несанкціонованого читання під час передачі та обробки. Відповідно, ефективність запропонованих методів оцінюється двома критеріями: рівнем захищеності, що досягається при їх застосуванні та часом, потрібний на їх реалізацію.

Задачею порушення захисту є відновлення істинного зображення, тобто встановлення за кодами $b_1,b_2,...,b_N$ значення невідомих кодів $a_1,a_2,...,a_N$ точок істинного зображення. При застосуванні запропонованого методу для захисту віддаленої реалізації медіанної фільтрації для цього зловмисник має виявити границі k інтервалів, на які розділяють діапазон представлення 2^u чисел закодованого зображення. Очевидно, що число ν варіантів виділення k інтервалів в комбінаторному плані дорівнює числу вибору k предметів з 2^u, яке згідно [5] визначається як $\upsilon = C_{2^u}^k = \frac{2^u!}{(2^u-k)!\cdot k!}$. При умові $k<<2^u$, що найчастіше має місце на практиці, значення $\upsilon \approx 2^{u\cdot k}/k! \approx (2^u \cdot e/k)^k \cdot \sqrt{2\cdot\pi\cdot k}$. Очевидно, що для типових для практики значень u та k число ν є доволі великим, для того, щоб гарантувати неможливість порушення захисту. Наприклад, при використанні 32-розрядних чисел (u=32) для кодування точок зображень, інтенсивність яких представляється 16-розрядним кодом ($k=2^{16}$) значення числа ν близьке до 10^{314572}, що практично повністю виключає порушення захисту шляхом перебору.

Час виконання операцій шифрування та дешифрування точок зображення значною мірою залежить від організації виконання цих операцій користувачем.

При обробці серії однотипних зображень, що найчастіше зустрічається на практиці, можна попередньо виконати допоміжні обчислення, пов'язані з розділенням на інтервали і генерацією випадкових чисел в цих інтервалах. За цих умов можна використовувати технології безколізійного хешування (perfect hashing) [5], які дозволяють уникнути використання бінарного інтервального пошуку. Тоді, запропонований спосіб шифрування та дешифрування зображень при віддаленій медіанній фільтрації зводиться до звертання до пам'яті при обробці кожної із точок зображення, тобто обчислювальна складність становить O(N). Обчислювальна складність фільтрації становить O(N·n^2) [3]. Таким чином, обчислювальна складність шифрування в n^2 раз менша за складність операції фільтрації. Наприклад, для розміру апертури 7 x 7 (n=49), складність операції шифрування складає менше одного відсотка від складності операцій медіанної фільтрації.

При виконанні середньоарифметичної фільтрації порушення захисту зображення, що передається для фільтрації в віддалену обчислювальну систему полягає в підборі модуля m, що лежить в інтервалі від k від 2^u. Технологія підбору полягає, що для вибраної точки перебираються всі k можливі її значення. Для кожного з цих значень перебираються можливі значення модуля m з вказаного вище інтервалу. Таким чином, об'єм перебору становить $k\cdot(2^u-k)$. Наприклад, при k=2^{16} і u=32, об'єм вказаного перебору становить $10^{14.4}$, що свідчить, що рівень захищеності запропонованого методу шифрування приблизно відповідає рівню захищеності відомого алгоритму DES [4].

Складність виконання операції фільтрації визначається тим, що для кожної з N точок зображення потрібно виконати n операцій додавання і одну операцію ділення. Шифрування точок може бути зведено до однієї операції додавання за умови, що масив добутків випадкових чисел на модуль сформовано заздалегідь. Дешифрування кожної точки зводиться до однієї операції ділення. Якщо вважати, що операція ділення по часу виконання відповідає приблизно 44-м операціям додавання [6], то можна вважати, що складність дешифрування в 1+n/44 раз менша складності фільтрації.

Висновки

В результаті проведених досліджень запропоновано методи захищеної реалізації масових операцій обробки зображень - медіанної та середньоарифметичної фільтрації. На основі проведеного аналізу операцій фільтрації запропо-

новано інтервальний метод шифрування точок зображення перед передачею його для обробки в GRID-системи. Метод забезпечує виконання медіанної фільтрації на віддалених відкритих комп'ютерних системах, закриваючи при цьому доступ до справжнього зображення.

Доведено, що за рівнем захищеності та часом реалізації запропонований метод захищеної реалізації медіанної фільтрації значно ефективніший в порівнянні з розробленим методом середньоарифметичної фільтрації. Сама медіанна фільтрація виконується швидше ніж середньоарифметична фільтрація.

З проведеного аналізу можна зробити висновок, що ефективність запропонованих методів захищеної реалізації фільтрації збільшується при зменшенні кількості градацій точок зображень. Найбільша ефективність захисту досягається при віддаленій обробці бінарних зображень.

Список посилань

1. Петренко А. Національна GRID інфраструктура для забезпечення наукових досліджень і освіти / А.Петренко Системні дослідження та інформаційні технології.- 2008 - № 4. - С.79-92.
2. Hennessy, John L. Computer architecture: a quantitative approach/ John L Hennessy, David A Patterson. – Amsterdam, Noord-Hooland, Netherlands : Elsevier, 2011. – 824 p.
3. Гонсалес Р. Цифровая обработка изображений / Р. Гонсалес, Р. Вудс; пер. с англ. В.В. Харитонова.- М.:Техносфера, 2005. - 1072 с.
4. Шнаер Б. Прикладная криптография / Б. Шнаер.; пер.с англ. П.В. Семьянов.- М.:Издательство Триумф, 2003.- 816 с.
5. Кнут Д. Искусство программирования для ЭВМ / Д. Кнут.; пер.с англ. Н.И.Вюковой, В.А. Галатенко, А.Б.Ходулева.- М.:Мир, 1978.- 840 с.
6. Брэй Б. Микропроцессоры Intel. Архитектура, программирование и интерфейсы. Шестое издание / Б. Брэй; перю с анг. А.В.Жукова.- Санк-Петербург: БХВ-Петербург, 2005.-1328 с.

УДК 683.519

ДЕМЧИНСКИЙ В.В.

ОСОБЕННОСТИ ПРИМЕНЕНИЯ МЕХАНИЗМОВ ФОРМИРОВАНИЯ И ОГРАНИЧЕНИЯ ТРАФИКА

В статье рассмотрены некоторые практические аспекты применения формирования и ограничения трафика – технологии качества обслуживания в телекоммуникационных сетях.

The article discusses some practical aspects of the traffic shaping and policing - Quality of Service technology in telecommunication networks.

Направление и цель работы

Поводом для данной статьи послужило проведение тренингов по управлению трафиком в конвергентных сетях, где возникала необходимость проиллюстрировать действие тех или иных механизмов QoS (Quality of Service, качество обслуживания) статистическими характеристиками трафика. Суть качества обслуживания – в обеспечении требуемых характеристик сервиса (как правило, задержки, джиттера и потерь) для определенных классов трафика. Стратегии обеспечения QoS направлены на управление перегрузками (очередями) и предотвращение перегрузок (контроль за установлением соединений, профилирование трафика и управление скоростью передачи). Фундаментальные исследования на данную тему приведены, например, в [1,2].

В ранее опубликованной статье [3] были исследованы механизмы управления очередями. В этой статье исследуем механизмы управления интенсивностью трафика. В данном контексте под управлением интенсивностью трафика будем понимать принудительное воздействие на трафик с целью соответствия потока пакетов заданному скоростному профилю. Различают два механизма управления интенсивностью трафика: формирование трафика (traffic shaping, TS) и ограничение трафика (traffic policing, TP). Мотив использования этих механизмов − создание предсказуемого потока трафика или ограничение скорости передачи.

Цель данной работы состоит в исследовании влияния механизмов управления трафиком (формирования и ограничения) и параметров их настройки на характеристики обслуживания.

Формирование трафика

Формирование трафика ограничивает скорость передаваемых пакетов путем временного удержания некоторых пакетов в очереди. Т.е. пакеты, передача которых отложена, задерживаются в очереди. Как результат, скорость передачи пакетов не будет превосходить заданную величину.

Реализация механизмов управления интенсивностью трафика основана на алгоритме маркерного ведра (token bucket). Легенда гласит, что есть ведро заданной емкости, которое периодически пополняется определенным количеством маркеров, а при передаче пакета некоторого размера соответствующее количество маркеров удаляется из ведра. При этом ведро не может вместить больше заданного количества маркеров, т.е. избыточные маркеры попросту «сливаются».

Другими словами, если в ведре достаточно маркеров для передачи пакета — он передается, а соответствующее количество маркеров удаляется из ведра. В ином случае пакет задерживается в очереди до тех пор, пока в ведре не будет достаточно маркеров для его передачи (или же очередь достигнет предела и излишние пакеты будут сброшены).

При настройке формирования трафика, как правило, задаются три величины: согласованная скорость передачи (CIR, Committed Information Rate), согласованный выброс (Bc, Burst Committed) и избыточный выброс (Be, Burst Excess). Время, в течение которого может быть передано количество трафика, соответствующее Bc, равно Bc/CIR и обозначается как Tc. Tc это и есть временной интервал, через который ведро маркеров пополняется на Bc маркеров. Как правило, в оборудовании используются значения Tc от 10 до 125 мс.

Если задано значение избыточного выброса

(Be), то максимальное количество маркеров в ведре может достигать Bc+Be. Когда в конце предыдущего Tc интервала осталось некоторое количество маркеров, то к ним в начале следующего интервала добавляются Bc маркеров. В частности, после Tc интервала не активности за следующий интервал может быть передано Bc+Be трафика. Следует добавить, что формирование трафика может быть применено только к исходящему трафику интерфейса или субинтерфейса.

Ограничение трафика

Ограничение трафика, в отличие от формирования, не откладывает передачу пакета. Реализация механизма ограничения трафика также основывается на алгоритме маркерного ведра. Механизм ограничения трафика отличается от формирования трафика поведением при исчерпании маркеров. В одном варианте настройки, если для передачи пакета достаточно маркеров — он передается. Если используются избыточные маркеры (Be), то к пакету применяется действие1. В ином случае (отсутствие маркеров) — применяется действие2. «Действием» может быть передача пакета, сброс пакета или изменение метки пакета. Заметим, что формирование трафика не предполагает перемаркировку пакетов.

Другой вариант настройки ограничения трафика предполагает использование двукратного алгоритма маркерного ведра. Первое «ведро», как и в предыдущих случаях, контролирует среднюю скорость (CIR), а второе — пиковую скорость (PIR, Peak Information Rate). Первое ведро, как и ранее, пополняется на Bc маркеров в начале каждого интервала Tc, а второе, соответственно, на Be. Если для передачи пакетов хватает маркеров из ведра CIR – они используются и пакет передается далее. Иначе, если достаточно маркеров в ведре PIR, маркеры используются и к пакету применяется действие1. В ином случае (отсутствует достаточное количество маркеров) – к пакету применяется действие2. «Действием», аналогично, может быть передача пакета, сброс пакета или изменение метки пакета.

Таким образом, механизм ограничения трафика позволяет ограничивать скорость пакетов и удалять избыточное количество пакетов или изменять их принадлежность к некоторому классу трафика (деклассируя их, путем установления метки принадлежности к менее приоритетному классу).

Ограничение трафика может применяться как к исходящему, так и входящему трафику интерфейса или субинтерфейса. Обычно формирование или ограничение трафика применяется к классовым очередям, т.е. классам трафика. Формирование или ограничение трафика может также применяться ко всему трафику интерфейса.

В оборудовании многих производителей существуют опции настройки, позволяющие настраивать параметры формирования или ограничения трафика как в абсолютных величинах (обычно, скорость в битах за секунду), так и в процентах к скорости интерфейса. Второй случай более подходит для определения универсальных профилей QoS, применимым к интерфейсам с разной скоростью.

Эксперименты и результаты

Для демонстрации свойств механизмов формирования и ограничения трафика покажем результаты экспериментов. Возьмем следующую топологию:

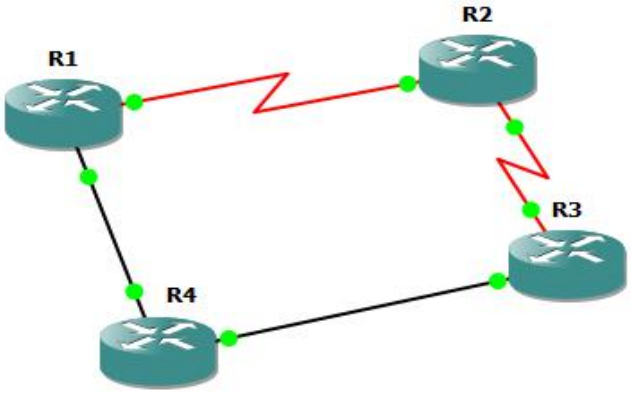


Рис. 1. Топология сети

Механизмы TS и TP будут применяться к выходному трафику маршрутизатора R1 (канал R1-R2). Скорости последовательных каналов (R1-R2 и R2-R3) искусственно ограничим 800 Кбит/с, а скорости остальных каналов (Ethernet) возьмем равными 10 Мбит/с. На R4 запустим генератор трафика TGN [4] и создадим 4 потока трафика с равной интенсивностью (средняя интенсивность каждого потока – 44 пакета/сек., длина пакетов равномерно распределена в интервале от 200 до 1000 байт.), которая подобрана таким образом, чтобы канал R1-R2 постоянно был в состоянии насыщения с минимально возможными потерями. Поток

трафика передается по маршруту R1-R2-R3-R4 и принимается на другом порту R4. На всех очередях установлена дисциплина обслуживания FIFO.

В 1-м эксперименте механизмы TS и TP не используются. Во 2-м эксперименте применим формирование трафика и каждому классу трафика выделим равную ПС (200000 бит/с). В 3-м установим ограничение скорости с аналогичными параметрами. А в 4-м эксперименте трафику 1 (условно, наиболее приоритетному) позволим использовать скорость до PIR=250000 бит/с. Трафику 2 (менее приоритетному) назначим Be=2*Bc=25000 бит. В настройках ограничения скорости трафика 3 (наименее приоритетного) установим только Bc=25000 бит. Весь трафик, превосходящий установленные ограничения скорости, будет сброшен. К трафику 4 механизмы формирования/ограничения скорости не применяются. Все параметры экспериментов приведены в Табл. 1.

Табл. 1. Параметры экспериментов.

Сценарий	**2, 3**		**4**			
Трафик	**CIR (byte/s)**	**Bc (byte)**	**CIR (byte/s)**	**Bc (byte)**	**PIR (byte/s)**	**Be (byte)**
1	200000	6250	200000	6250	250000	7812
2	200000	6250	200000	25000	-	50000
3	200000	6250	200000	25000	-	-

Результаты моделирования (табл. 2) подтверждают, что формирование трафика по сравнению с механизмом ограничения увеличивает время ожидания в очереди. В свою очередь, применение механизма ограничения скорости отличается бóльшими потерями пакетов. Результаты 4-го эксперимента демонстрируют, что при установке параметра Be (трафик 2) потери пакетов значительно меньше, чем для трафика 3. В то же время, увеличение максимальной скорости (трафик 1), естественно, ликвидировало сброс пакетов для данного трафика. А уменьшение среднего времени задержки объясняется увеличением количества сброшенных пакетов.

Табл. 2. Результаты моделирования.

Сценарий	**1**			**2**			**3**			**4**		
Трафик	**Delay (ms)**	**Jitter (ms)**	**Loss**	**Delay (ms)**	**Jitter (ms)**	**Loss**	**Delay (ms)**	**Jitter (ms)**	**Loss**	**Delay (ms)**	**Jitter (ms)**	**Loss**
1	42,7	20,9	4	163	61	182	27,4	10,2	26	14,9	6,9	0
2	42,9	20,7	6	167	58,7	192	26,7	10,2	24	15,4	6,7	30
3	42,1	20,8	9	160	59,2	171	27,1	10,1	29	14,7	6,9	86
4	42,5	20,8	2	161	55,2	164	27,5	10,1	25	15,3	6,9	0

Однако, экспериментально не удалось показать влияние параметра Tc на время задержки и потери пакетов. Это можно объяснить пачечностью трафика или особенностями реализации механизмов TS и TP.

Сравнивая механизмы формирования и ограничения трафика в целом, отметим, что ограничение скорости «срезает» выбросы интенсивности, а формирование трафика выравнивает скорость передачи на уровне установленной интенсивности (что естественным образом делает трафик более предсказуемым).

Можно заметить, что эффективность утилизации пропускной способности выше при использовании формирования трафика (передача пакета может быть отложена, в то время, как при механизме ограничения скорости пакет будет сброшен). Причем, формирование трафика достигает наибольшей эффективности, если размеры выбросов трафика меньше значения Bc.

Кроме того, из-за эффекта глобальной синхронизации, ограничение скорости ухудшает эффективность передачи TCP

трафика, в то время, как для UDP-трафика или в случае использования формирования трафика этого не наблюдается. Это объясняется тем, что при отбрасывании пакетов протокол TCP сужает окно потока, что может быть причиной падения интенсивности. Подобрав параметры всплеска трафика в соответствии с реализацией TCP теоретически возможно повысить эффективность передачи TCP. На трафики, не использующие адаптивное регулирование скорости передачи (IP, UDP, ICMP), параметр всплеска не оказывает заметного влияния.

Вариация параметров настройки ограничения трафика показала, что для потоков с бóльшими колебаниями интенсивности эффективней бóльшие значения Bc. В некоторых случаях утилизацию канала можно повысить, удвоив значение Bc (в рекомендациях производителя оборудования встречается рекомендованная цифра Bc=1.5*CIR).

Выводы

Знание особенностей управления интенсивностью трафика (формирования и ограничения) позволяет более конструктивно применять данные механизмы с целью профилирования трафика на входе в сеть или на границе областей ответственности. Оперируя трафиками с предсказуемым профилем, возможно более точно организовать обработку таких потоков с целью обеспечить требуемые характеристики качества обслуживания.

Не смотря на то, что применяемые в настоящее время модели и механизмы управления трафиком сформировались примерно десяток лет тому назад, их эволюцию нельзя считать завершенной. За это время появились такие понятия, как конвергентные сети, triple play, wireless QoS, которые безусловно повлияли на представление об архитектуре качества обслуживания. Помимо этого, появляются новые реализации ранее известных механизмов качества обслуживания, а также категории трафика с не встречавшимися ранее статистическими паттернами, что требует некоторого переосмысления поведения известных механизмов качества обслуживания.

Список литературы

1. Вегешна Ш. Качество обслуживания в сетях IP. :Пер. с англ. - М.: Изд. дом «Вильямс», 2003. - 368 с.
2. Кучерявый Е.А. Управление трафиком и качество обслуживания в сети Интернет. - Спб.: Наука и Техника, 2004. - 336 с.: ил.
3. Демчинский В.В., Дорогой Я.Ю., Дорошенко К.С. Анализ и практика внедрения механизмов качества обслуживания. // Вісник НТУУ “КПІ”. Інформатика, управління та обчислювальна техніка: Зб. наук. пр. - К.: Век+,-2012. - №57. - 171 с.
4. TGN Traffic Generation Using Cisco Pagent IOS. - [Електронний ресурс]. - URL: http://www.ciscoconsole.com/nms/traffic-generation-using-cisco-pagent-ios.html/

УДК 004.82

ДЕНИСЮК А.В.,
КАВИЦКАЯ В.С.,
ЛЮБЧЕНКО В.В.

ОПРЕДЕЛЕНИЕ АДЕКВАТНОЙ МОДЕЛИ ДЛЯ ПРЕДСТАВЛЕНИЯ ЗНАНИЙ В СОВРЕМЕННЫХ ИНФОРМАЦИОННЫХ СИСТЕМАХ

Обоснована ограниченность реляционной модели данных для использования современными информационными системами. Определены основные особенности знаний, которые присущи современным информационным системам. Проанализированы нереляционные модели данных, такие как фреймы, объектные модели, онтологические модели с точки зрения отображения особенностей знаний. Определена адекватная модель данных для задач, решаемых информационными системами.

The article summarizes limitations of relational database approach in modern informational systems. Analysis is made based on main distinctive features of knowledge that is processed by modern informational systems. Non-relational data models such as frames, object model and ontology are considered as alternatives to relational approach in regard to defined particularities. As result, rules are defined for choosing the adequate data model to be used in modern informational systems.

Введение

Любая информационная система (ИС) обязательно оперирует данными. Данные представляют собой совокупность сведений, зафиксированных на определенном носителе в форме, пригодной для постоянного хранения, передачи и обработки. Преобразование и обработка данных позволяет получить информацию. Например, в базах данных хранятся различные данные, а по определенному запросу система управления базой данных выдает требуемую информацию.

Усложнение ИС, а также решаемых ими задач и необходимой для этого информации, приводит к необходимости оперировать не данными, а знаниями. При этом знания рассматриваются как осознание и толкование определенной информации с учетом путей наилучшего ее использования для достижения конкретных целей. Очевидно, что переход к использованию знаний требует обратить внимание на модель, которой представляются знания в ИС, для простоты их понимания разработчиком и эффективного их использования в процессе работы ИС. В связи с этим актуальной является задача выбора адекватной модели данных для задач, решаемых ИС.

1. Анализ литературных данных и постановка проблемы

ИС оперирует различными данными в своей деятельности. Некоторые данные могут быть использованы непосредственно, без дополнительной обработки. Тем не менее, большинство данных требуют дополнительной обработки в ИС, а именно, подготовки и форматирования данных, анализа данных и извлечения знаний из этих данных [1]. С ростом необходимости использования знаний в современных ИС, возросла сложность решаемых задач. Реляционные модели данных (РМД) уже не могли эффективно решать эти задачи [2]. Остро стала проблема выбора модели данных, которая бы позволила справляться с возросшей сложностью задач.

2. Цель и задачи исследования

Целью данной работы является обобщение результатов анализа существующих моделей представления данных для того, чтобы сформулировать правила выбора адекватной модели данных для задач, решаемых ИС.

Для достижения поставленной цели решались такие задачи:

– определить особенности знаний, используемых в современных ИС;

– проанализировать способности реляционной модели отображать особенности сложных знаний;

– проанализировать способности нереляционных моделей отображать особенности сложных знаний;

– определить адекватную модель для представления знаний в современных ИС.

3. Материалы и методы исследования

3.1. Анализ способности реляционной модели данных представлять знания ИС

Определение состава и структуры знаний, необходимых для функционирования ИС, осу-

ществляется на основе модели предметной области (ПрО). Модель ПрО – это совокупность структурированных моделей сущностей ПрО, их свойств и отношений между ними, а также ограничений, определяющих контекст ПрО [3].

В настоящее время наиболее распространенной формой представления модели ПрО в ИС являются данные – отдельные факты, характеризующие объекты, процессы и явления ПрО, а также их свойства [4].

Ключевым свойством данных является то, что почти всегда они являются результатами измерений и наблюдений, преобразованными для хранения и представления в ИС. Но стоит отметить, что данные, в том числе, могут быть высказаны человеком на основании его опыта.

Наиболее распространенной моделью представления данных в настоящее время является реляционная модель данных (РМД) – логическая модель данных, в которой данные представляют собой набор отношений (таблиц), отвечающих определенным условиям целостности, а также существуют операторы манипулирования отношениями. Кроме того, в состав РМД обычно включают теорию нормализации. На основе РМД строятся реляционные базы данных (РБД).

РМД является простой моделью ПрО, в которой не предполагается или предполагается небольшая изменчивость, а РБД хранит данные, собираемые в соответствии с моделью ПрО, зафиксированной на этапе разработки. РМД имеет ряд неоспоримых достоинства, среди которых следует отметить [5]:

1. Предсказуемость результатов работы с данными. В основе реляционной модели лежит математическая модель – любой корректный запрос к БД выдаст ответ, однозначно определяемый схемой БД и конкретными данными.
2. ПрО часто достаточно естественно описывается в терминах таблиц, кроме представлений иерархических структур.

Эволюция ИС, существенное усложнение решаемых ими задач, появление новых технологий реализации ИС привели к тому, что данные и РМД стали представлять «узкое место» в работе систем. В результате этого возрос интерес к использованию знаний в ИС.

Знания – это закономерности ПрО (принципы, связи, законы). Их основным свойством является то, что почти всегда они – результат интеллектуальной деятельности, представляющей либо обработку данных, либо обработку уже существующих знаний.

Для хранения знаний разрабатывают базы знаний (БЗ), которые содержат структурированную информацию, покрывающую некоторую ПрО, для использования ИС с конкретной целью. Полноценные БЗ содержат в себе не только фактическую информацию, но и правила вывода, допускающие автоматические умозаключения о вновь вводимых фактах и, как следствие, осмысленную обработку информации.

Сравнение РМД и модели знаний позволяет сделать вывод, что модель знаний – это упрощенная модель, описывающая часть понятий ПрО для работы со знаниями данной ПрО, в то время как РМД – это упрощенная модель, описывающая только те понятия ПрО, которые требуется сохранить.

Кроме этого можно утверждать, что РМД является простой моделью знаний, в которой предполагается нулевая или небольшая изменчивость, а РБД – это БЗ, построенная с использованием простой модели знаний, которая зафиксирована на этапе разработки.

Очевидно, что если знания можно описать простой моделью знаний, то для работы ИС с ними можно использовать РМД и РБД. Однако, применимость таких знаний существенно ограничена. В современных ИС все чаще возникает необходимость учитывать такие особенности знаний, как

– наличие исключений из правил;
– изменения в наборе атрибутов сущности, возможность появления новых либо отсутствия уже известных;
– изменения типа атрибутов в зависимости от контекста;
– необходимость отображения отношений;
– изменчивость структуры ПрО;
– начальная неопределенность в модели, так как при проектировании новой БЗ почти всегда не имеется полного объема входящей информации, из-за чего ее структура почти всегда некорректна;
– постоянные изменение и уточнения, как данных о сущностях ПрО, так и структуры и связей между ними;
– неоднородность данных;
– разное количество данных, так как разные классы сущностей могут быть представлены разным количеством данных, при этом количество может отличаться на несколько порядков.

В связи с этим возникает проблема выбора адекватной модели данных для представления знаний ИС.

3.2. Анализ способности нереляционных моделей данных представлять знания ИС

Известно, что модель данных следует рассматривать, как комбинацию трех компонент: коллекции типов объектов данных, коллекции общих правил целостности и коллекции применимых операций [6]. Фактически, способ определения каждой из компонент зависит от цели моделирования данных, то есть для нас представляет интерес адекватность модели данных – совпадение ее свойств и свойств моделируемых данных. Дадим определение.

Адекватная модель данных – это абстрактное, логическое определение объектов, моделирующих структуру данных, операторов, моделирующих поведение данных, и прочих элементов, в совокупности составляющих абстрактную машину доступа к данным, свойства которой соответствуют закономерностям ПрО.

Как было отмечено ранее, в случае достаточности простой модели знаний РМД является адекватной моделью. Рассмотрим свойства известных нереляционных моделей данных – фреймов, объектных моделей и онтологий – с точки зрения моделирования сложных знаний.

Фрейм – это структура данных, представляющая стереотипную ситуацию. Он состоит из нескольких слотов, каждый из которых имеет свое назначение и связанное с ним значение. Кроме того, со слотами могут быть связаны специальные процедуры, предназначенные для специфической обработки значений в слотах. Это единица представления знаний, детали которой при необходимости могут быть изменены согласно ситуации. Фреймовая модель является достаточно универсальной, так как обеспечивает не только фреймы для обозначения объектов и понятий, но и фреймы-роли, фреймы-ситуации и др. [7].

Проанализируем способность фреймовой модели обеспечивать отображение особенностей сложных знаний.

1. Фреймовая модель плохо учитывает наличие исключений из правил, из-за сложности реализации самой структуры фрейма, что на порядок увеличивает трудоемкость создания модели.
2. Фреймовая модель плохо учитывает изменения в наборе атрибутов сущности, так как возникает проблема хрупкости базового класса — изменения в наборе атрибутов в корневом узле могут быть небезопасными для наследуемых классов (при этом критично удаление атрибутов в базовом классе).
3. Фреймовая модель плохо учитывает изменения типа атрибутов в зависимости от контекста, так как возникает проблема — изменения типа атрибутов в корневом узле могут быть критичными для наследуемых классов.
4. Фреймовая модель позволяет отображать отношения с помощью отношений «абстрактное – конкретное» и «целое – часть». Иерархическая структура фреймов основывается на отношениях «абстрактное – конкретное». Отношение «целое – часть» касается структурированных объектов и показывает, что объект нижнего уровня является частью объекта верхнего уровня.
5. Изменчивость структуры ПрО при описании фреймовой моделью приводит к снижению скорости работы механизма вывода и увеличивает трудоемкость процедур внесения изменений в родовидовую иерархию модели ПрО.
6. Фреймовая модель плохо учитывает начальную неопределенность в модели, потому что обладает четко выраженной иерархией и процедурой наследования значений, что требует определения набора фреймов на начальном этапе проектирования.
7. Фреймовая модель плохо учитывает постоянные изменения и уточнения о данных и сущностях ПрО, так как фреймам присуща иерархическая структура, что ведет за собой проблему хрупкости базового класса.
8. Неоднородность данных описывается фреймовой моделью, так как модель позволяет отобразить все разнообразие знаний через фреймы-структуры, фреймы-роли, фреймы-сценарии, фреймы-ситуации.
9. Фреймовая модель позволяет учитывать разное количество данных.

Объектная модель описывает структуру объектов, составляющих ПрО, их атрибуты, операции, взаимосвязи с другими объектами. В объектной модели отображаются только те понятия и объекты реального мира, которые важны для проектируемой ПрО. Соответственно, в объектно-ориентированных БД хранятся не записи, а объекты. Цель разработки объектной модели — описать объекты, составляющие в совокупности проектируемую ПрО, а также выявить и ука-

зать различные зависимости между объектами [8].

Проанализируем способность объектной модели обеспечивать отображение особенностей сложных знаний.

1. Объектная модель позволяет описывать исключения из правил, так как обладает возможностью выразить сложные зависимости.

2. Объектная модель плохо учитывает изменения в наборе атрибутов сущности, а также возможность появления новых либо отсутствия уже известных (удаление является критичным), так как изменения в наборе атрибутов базового класса могут оказаться разрушительными для производных классов.

3. Изменения типа атрибутов в зависимости от контекста объектной моделью описываются на низком уровне, так как изменения типа атрибутов базового класса могут быть небезопасными для производных классов.

4. Объектная модель на высоком уровне отображает отношения в БЗ, так как она реализует механизм ассоциативных связей и отношений наследования.

5. Объектная модель на среднем уровне учитывает изменчивость структуры ПрО, так как обладает проблемой уязвимости базового класса.

6. Объектная модель позволяет учитывать начальную неопределенность в модели, однако она обладает сложностью реализации. Использование объектно-ориентированной модели для проектирования новой БЗ, которая почти всегда не имеется полного объема входящей информации, требует больших временных затрат и приводит к построению сложной и требовательной к ресурсам модели.

7. Объектная модель позволяет учитывать постоянные изменение и уточнения, как данных о сущностях ПрО, так и структуры и связей между ними, однако усложняется методология проектирования — объектная модель требует введения дополнительных способов представления информации о ПрО и методов ее анализа.

8. Неоднородность данных при использовании объектной модели приводит к построению сложной и требовательной к ресурсам модели.

9. Объектная модель на высоком уровне позволяет описывать разное количество данных.

Онтология — формальная спецификация разделяемой концептуализации, которая имеет место в некотором контексте ПрО. При этом под концептуализацией понимают, кроме сбора понятий, также всю информацию, касающуюся понятий — свойства, отношения, ограничения, аксиомы и утверждения о понятиях, необходимые для описания и решения задач в избранной ПрО [9].

Онтология ПрО определяет формальное приближение концептуализации. В онтологии зафиксирована та часть концептуализации, которая зависит от взгляда на мир применительно к конкретной области интересов. На формальном уровне, онтология – это система, состоящая из набора понятий и набора утверждений об этих понятиях, на основе которых можно строить классы, объекты, отношения, функции и теории [10].

Проанализируем способность онтологической модели обеспечивать отображение особенностей сложных знаний.

1. Онтологическая модель позволяет учитывать исключения из правил, так как обладает возможностью описывать сложные знания, начиная с простых словарей до формальных логических структур, которые позволяют выполнять сложные логические выводы.

2. Онтологическая модель позволяет учитывать изменения в наборе атрибутов сущности, возможность появления новых либо отсутствия уже известных за счет использования свойств общего типа, с реализованной для них возможностью переопределения типа.

3. Онтологическая модель позволяет учитывать изменения типа атрибутов в зависимости от контекста за счет использования соответствующего математического аппарата (например, теории категорий), позволяющего учитывать эволюционные изменения.

4. Онтологическая модель позволяет отображать отношения за счет отношений таксономии («kind-of», «is-a»), партономии («has part»), генеалогии («parent-child»). Полноценные онтологии имеют больше возможностей для описания отношений, чем простые таксономические иерархии. Отноше-

ния (связи) между понятиями значительно увеличивают количество способов осуществлять навигацию и поиск по ПрО, выполнять анализ, классификацию и визуализацию знаний.

5. Изменчивость структуры ПрО описывается онтологией на высоком уровне. Онтологическая модель декларативным образом описывает ПрО, соответствующие словари типов, ограничения использования этих данных, в предположении понимания того, что может быть выведено из этого описания, а также добавлено в процессе проектирования ПрО.

6. Онтологическая модель позволяет учитывать начальную неопределенность в модели, так как предоставляет совместно используемые понятия, которые могут использоваться для формулировки и записи утверждений об экземплярах ПрО.

7. Онтологическая модель позволяет учитывать постоянные изменения и уточнения, как данных о сущностях ПрО, так и структуры и связей между ними, так как онтологии являются более чем простыми словарями точно определенных понятий: их реальная сила заключается в описании связей между сущностями ПрО.

8. Онтологическая модель позволяет учитывать неоднородность данных, так как в каждой ПрО могут существовать различные понимания одних и тех же терминов. Онтология определяет соглашение о значении терминов и является посредником между человеком и машинно-ориентированным уровнем представления информации.

9. Онтологическая модель позволяет учитывать разное количество данных, так как разные классы сущностей могут быть представлены разным количеством данных, при этом количество может отличаться на несколько порядков.

4. Результаты исследований

Таким образом, можно утверждать, что для сложных знаний адекватной моделью следует считать онтологическую модель, так как она способна наследовать некоторые характеристики описываемых понятий реального мира. К таким характеристикам можно отнести изменчивость структуры ПрО, изменения в наборах атрибутов сущностей и связях между ними, наличие исключений из общих правил и другие. Какие конкретно характеристики унаследует онтология, зависит от того, какую часть реального мира она описывает.

Выполненный анализ позволяет сделать следующее обобщение. Онтология по сути – это упрощенная модель, описывающая часть понятий реального мира для работы со знаниями данной области, в то время как РМД – это упрощенная модель, описывающая только те понятия реального мира, которые требуется сохранить.

Онтологии можно классифицировать следующим образом:

1. Простые – онтологии с постоянной структурой, однородностью, небольшой изменчивостью и связностью. Такие онтологии обычно описывают либо небольшую часть понятий реального мира, либо описывают понятия ПрО, которая была создана человеком (а значит, сама является моделью – упрощенным представлением).

2. Сложные – онтологии с изменчивой структурой, обладающие большой степенью связности, изменчивости, либо неоднородности. Такие онтологии обычно описывают большую часть понятий реального мира, понятия на грани нескольких областей и др.

Выводы

Очевидно, РМД является простой онтологией, в которой предполагается нулевая или небольшая изменчивость, а РБД – это БЗ, построенная с использованием простой онтологии зафиксированной на этапе разработки.

Из этого следует простой вывод: знания, которые можно описать простой онтологией, возможно также представить в виде данных. Можно также предположить, что при достаточно мощном инструменте для работы с РБД возможно также сохранение знаний обладающих сложной онтологией.

Список литературы

1. Géczy P. Big data characteristics / P. Géczy. – A multidisciplinary journal of global macro trends, 2014. — V. 3 (6). — P. 94-104.
2. Bhat U. Moving Towards Non-Relational Databases / U. Bhat, S. Jadhav. — International Journal of Computer Applications, 2010 — V. 1 (13).

3. Любченко В. В. Методологічні основи розробки моделей предметних областей з метою, що інкапсульована : автореф. дис. … д-а техн. наук : 05.13.06 / В. В. Любченко. — Одеський національний політехнічний університет, 2013. — 43 с.
4. Гаврилова Т. А. Базы знаний интеллектуальных систем / Т. А. Гаврилова, В. Ф. Хорошевский. — СПб.: Питер, 2001. – 384 с.
5. Когаловский М. Р. Энциклопедия технологий баз данных [Текст] / М. Р. Когаловский. — М.: Финансы и статистика, 2002. — 800 с.
6. Akerkar R. Knowledge-Based Systems / R. Akerkar, P. Sajja. — Jones & Bartlett Learning, 2009. — 354 p.
7. Minsky M. A framework for representing knowledge / M. Minsky. — B.; N. Y.: Gruyter, 1980. — 153 p.
8. Booch G. Object-oriented analysis and desingn with applications [Електронний ресурс] / G. Booch. — Santa Clara, California. — 1998. – Режим доступа: www / URL:http://www.cvauni.edu.vn/imgupload_dinhkem/file/pttkht/object-oriented-analysis-and-design-with-applications-2nd-edition.pdf. – Загл. с экрана.
9. Brunetti G. Feature ontologies for the explicit representation of shape semantics / G. Brunetti, G. Stephan. — International Journal of Computer Applications in Technology (IJCAT), 2005. — V. 25. — #2/3/4 — P. 192-202.
10. Grimm S. Knowledge Representation and Ontologies / S. Grimm, P. Hitzler, A. Abecker. — Semantic Web Services: Concepts, Technology and Applications. — Berlin: Springer, 2007. — P. 51-106.

УДК 004

БОЙКО Ю.В.,
ГЛИБОВЕЦЬ М.М.,
ЄРШОВ С.В.,
КРИВИЙ С.Л.,
ПОГОРІЛИЙ С.Д.,
РОЛІК О.І.,
ТЕЛЕНИК С.Ф.,
ЯСОЧКА М.В.

УПРАВЛІННЯ ВИСОКОПРОДУКТИВНИМИ ІТ-ІНФРАСТРУКТУРАМИ

Наведений аналітичний огляд сучасних моделей і методів управління ІТ-інфраструктурою. Моделі і методи згруповані у відповідності з визнаними ІТ-галуззю 15 групами функцій управління. Основна увага приділяється концептуальним основам управління такими складними ієрархічними розподіленими об'єктами управління, які сьогодні становлять глобальні, національні і корпоративні ІТ-інфраструктури. Найбільш широко представлені моделі розподілу ресурсів і навантаження та управління рівнем обслуговування користувачів.

An analytical survey of the modern IT-infrastructure management models and methods is presented. This models and methods are grouped in accordance with approved by IT-community15 groups of management functions. The main attention is given to management basics of global, national and corporative IT-infrastructures — complex hierarchical distributed object of management. The models of resources and load allocation and users services level management are widely presented.

Введение

Глобалізація світової економіки, формування соціально-орієнтованого середовища обумовлює новий рівень вимог до інформаційних технологій (ІТ). Бізнес, наукові дослідження і розроблення нових товарів і технологій, підтримка державного управління і комунального господарства вимагають вирішення все складніших проблем. Потрібні значні обчислювальні і телекомунікаційні ресурси, системи накопичення і зберігання великих обсягів даних [1, 2, 3, 4].

Вирішення проблем підтримки власних ІТ-інфраструктур відволікало підприємства, компанії, органи управління від основної діяльності. ІТ-інфраструктури створювалися в розрахунку на пікові навантаження і загалом використовувалися на рівні 15–20% від їх можливостей.

Реакція ІТ-галузі на цю ситуацію полягала у концентрації ресурсів у центрах оброблення даних (ЦОД) і їх наданні підприємствам, компаніям, органам управління у вигляді сервісу на потребу. Організаційно-технічно ці тенденції реалізовані за допомогою хмарних або Grid-технологій. Хмарні сервіси — Infrastructure as a Service (IaaS), Platform as a Service (PaaS) і Software as a Service (SaaS) — поступово змінюють уявлення про роль ІТ у бізнесі і власне ІТ як бізнес. Провайдери хмарних сервісів, насамперед Salesforte.com, Amazon, Microsoft, Oracle, SAP, Softlayer, Terremark, Rackspace, NetSuit, Rackspace постійно збільшують дохід від надання сервісів, знижуючи ціни.

Створення, підтримка і використання ІТ-інфраструктури для розв'язання наукових задач, базування автоматизованих інформаційних систем (АІС) породжує складні науково-практичні проблеми. Колектив авторів плідно працює над вирішенням зазначених проблем. У статті розглядаються проблеми управління високопродуктивною ІТ-інфраструктурою.

Сучасні тенденції розвитку ІТ-галузі

Шукаючи шляхи вирішення проблем, провайдери інформаційно-комунікаційних сервісів (ІКС) удосконалюють діяльність з метою збереження достатнього рівня її рентабельності в умовах зростання конкуренції і жорстких вимог замовників [5]. До основних тенденцій розвитку ІТ-галузі, пов'язаних з темою статті, належать:

1. *Втілення ідеї глобалізації у розвитку корпоративних, національних і глобальних ІТ-інфраструктур* — організованих сукупностей взаємопов'язаних мереж, ІТ, інформаційних ресурсів і обладнання кінцевих користувачів та їх оточення, як організованої сукупності інформаційних застосувань, застосувань користувачів та інформаційних послуг, взаємодія яких

забезпечує підтримку процесів збору, оброблення і збереження інформації користувачів.

2. *Поява нових видів провайдерів* (Інтернет, контенту, сервісів за моделями IaaS, PaaS, SaaS) і формування класу конвергентних провайдерів.

3. *Зосередження провайдерів на проблемах користувачів і інтеграція ресурсів ЦОД для* надання їм сервісів для підтримки діяльності.

4. *Поява хмарних обчислень — Cloud Cotriting*, які забезпечують оптимізацію ресурсів ЦОД і ІТ-середовища в цілому.

5. *Віртуалізація платформ і ресурсів ІТ-інфраструктури*, як засіб для спрощення управління групами ресурсів і доступу до них.

6. *Поява мережоцентричних архітектур* як бази електронної комерції і п*оширення сервіс-орінтованої і WEB-орієнтованої архітектур*.

7. *Поширення компонентно-базованого підходу до створення АІС* шляхом комплексування готових компонентів єдиного ІТ-середовища.

8. *Розвиток розподілених систем збереження даних* (*СЗД*) об'єднанням сховищ різної архітектури з IP-взаємодією з застосуваннями.

9. *Широке впровадження автоматизації і поява систем управління ІТ-інфраструктурою (СУІ)* надання *сервісів внаслідок її ускладнення*.

Конвергентними стають сервіси і провайдери. Мережі провайдерів, корпорацій, комунальні ІТ-системи інтегрувалися, утворивши єдине ІТ-середовище. ІТ-інфраструктура, як його основа, перетворилася у важливий об'єкт управління. Використання ІКС від провайдерів стимулюється економічними чинниками. Відносини користувача і провайдера регламентує SLA. Гарантований рівень обслуговування убезпечує провайдерів від збитків і сприяє залученню користувачів. Провайдерів ІКС приділяють велику увагу його забезпеченню, намагаючись скоротити витрати на надання послуг.

Системи управління провайдерів будуються на основі процесно-функціональної, структурно-організаційної і ресурсної концепцій управління. В них виділяють business (BSS) і operations (OSS) support systems, які формують уніфіковані електронні образи бізнесу і ІТ-інфраструктури провайдера, забезпечуючи аналіз стану, генерування, оцінювання і вибір рішень. Їх будова і функціонування базуються на виділенні ІТ-інфраструктури як об'єкта управління, процесів, які задіюють її і підтримують діяльність як основу бізнесу, і організації, яка перетворює потенціал в реальні результати[6,7].

Розраховане на пікове навантаження традиційне горизонтальне структурування ІТ-середовищ підприємств виявилося не гнучким і не адаптованим до змін у діяльності, мало високу вартість створення, підтримки і управління. Консолідована ІТ-інфраструктура об'єднує всі ресурси в єдиний пул для підтримки діяльності всіх бізнес-підрозділів [8]. Консолідація базується на технологіях віртуалізації платформ шляхом створення віртуальних машин (ВМ), ідеї якої переносять на мережі, СЗД та ін. [9].

Впровадження процесного підходу на основі стандартів передбачає реалізацію трьох груп функцій управління — підприємством, розвитком і операційною діяльністю. Перша група традиційна для всіх підприємств. Специфіку діяльності провайдерів ІКС визначають друга і третя групи, спрямовані на клієнтів, постачальників і партнерів. Процеси другої групи спрямовані на створення середовища функціонування, яке забезпечує підтримку і автоматизацію операційних процесів, визначаючи напрями розвитку ІТ-інфраструктури і продуктів. Їх підтримують підгрупи процесів розроблення стратегії розвитку провайдера; управління життєвим циклом (ЖЦ) інфраструктури і продукту. Операційні процеси спрямовані на роботу з клієнтами, їх потребами, вимогами, оцінками і історією співпраці, утворюючи вертикальні наскрізні угрупування процесів підтримки операцій і забезпечення готовності, виконання, управління якістю обслуговування і білінгу.

Аналіз існуючих рішень

Основні постачальники ІТ вийшли на ринок з рішеннями, призначеними для управління ІТ-інфраструктурою. Цей сегмент ринку насичений продуктами, створеними на основі IT Infrastructure Library (ITIL) і концепції IT Service Management (ITSM). У процесній моделі ITIL виділяють 3 групи процесів управління: *управління інфраструктурою* (проектування і планування, розширення, супроводження і технічна підтримка); *підтримка обслуговування* (управління інцидентами і проблемами, конфігурацією і змінами, релізами); управління наданням послуг (рівнем обслуговування, фінансами, ІТ-послугами, готовністю, неперервністю обслуговування, потужностями).

Повнофункціональні рішення Microsoft, IBM, HP, EMC Symantec і Borland надають адміністраторам широкі можливості з управління ІТ-інфраструктурою ЦОД. Забезпечуючи створення єдиного інформаційного середовища, централізацію процесів розгортання, експлуатації і моніторингу ІТ-інфраструктури, вони

оцінюють вплив змін в застосуваннях, конфігураціях і топології на бізнес-процеси, підтримують роботи поширених платформ.

Продукти інших виробників реалізують базову функціональність. Крім того, деякі розробники систем ERP, наприклад, PeopleSoft і SAP включили до складу своїх основних продуктів компоненти управління IT-послугами.

Загалом рішення орієнтовані на підвищення гнучкості IT-інфраструктури і оптимізацію використання ресурсів. Але їх ефективність суттєво залежить від моделей розподілу ресурсів, особливо в умовах ресурсних обмежень. На жаль, у відкритих джерелах мало інформації про моделі і методи. покладені в основу реалізації наведених вище програмних продуктів. Окрім того, розглянуті рішення переважно орієнтуються на корпоративні комерційні структури і не враховують оперативні вимоги. Розглянуті компоненти не надають засобів оцінки показників оперативної діяльності та планування, оптимізації та управління АІС користувачів з урахуванням відповідних критеріїв і обмежень. Суттєвими недоліками згаданих продуктів є використання закритих фірмових технологій, моделей і алгоритмів управління. Висока вартість цих продуктів та орієнтованість на крупні підприємства з великим парком обчислювальної техніки не дозволяє розглядати їх в якості СУІ для невеликих підприємств.

Отже, обґрунтоване вирішення всього комплексу задач управління IT-інфраструктурою відсутнє і виникає потреба у недорогій СУІ, яка відрізняється простотою адаптації та використанням ефективних моделей і методів.

Проблеми управління високопродуктивною IT-інфраструктурою

Втілення потенційних переваг вимагає реалізації всіх груп функцій процесної моделі ITIL. Це робить необхідним вирішення ряду проблем ефективного планування і управління IT-інфраструктурою і діяльністю провайдерів ІКС.

Загальні проблеми, викликані консолідацією: визначення напрямів розвитку IT-інфраструктури; забезпечення відмовостійкості її компонентів; побудова високопродуктивної IT-інфраструктури; безпека; взаємодія компонентів різних виробників; підтримка діяльності IT-служб.

Реалізація сервіс-орієнтованого підходу вимагає вирішення таких проблем: відстеження галузевих тенденцій; аналіз ринку; оцінювання перспективності і конкурентоздатності діяльності провайдера; вчасне і виважене введення нових продуктів і сервісів; адресування серверів; взаємодія сервісів; маршрутизація доступу до сервісів; реєстрація сервісів та їх представлення користувачам; визначення раціональних параметрів SLA; реєстрація ресурсів і їх представлення користувачам; забезпечення визначеного QoS, безпеки і довіри; надання здатності до самоуправління і самоорганізації.

Орієнтація на проблеми користувачів вимагає вирішення проблем реалізації компонентно-базованого підходу до створення АІС компаній-користувачів на основі комплексування сервісів, управління ЖЦ сервісів, реалізації міжсервісної взаємодії з урахуванням SLA на основі політик, забезпечення рівня обслуговування, визначеного з потреб бізнесу, забезпечення необхідних для надання ІКС ресурсів і управління ними та ін. [10, 11].

Забезпечення надання сервісів незалежно від технологій транспортного і ресурсного рівнів вимагає вирішення таких проблем: управління ресурсами; оптимізація мережного трафіку; інформованість користувачів про ІКС і їх зв'язок з бізнес-процесами; вибір сервісів і їх комплексів для підтримки діяльності користувачами; управління безпекою на основі динамічно змінюваних політик; надання системам управління провайдерів і користувачів здатності до самоуправління і самоорганізації.

Розглянемо найважливіші проблеми управління IT-інфраструктурою, виявляючи чинники впливу, критерії, обмеження та інші аспекти.

Проблема ресурсів. Ресурси IT-інфраструктури вимагають обліку і аналізу відповідності вимогам. Гнучкі рішення на основі на балансування навантаження і розподілу ресурсів, є важелем підтримки рівня надання ІКС.

Проблема управління рівнем обслуговування користувачів. Для її вирішення необхідно об'єктивно і своєчасно визначати причину погіршення параметрів сигналу, аналізувати зміни параметрів якості, передбачати завантаженість мережі, популярність того чи іншого контенту, і забезпечувати, за потреби, виділення додаткових ресурсів на надання сервісу. Визначальним для управління IT-інфраструктурою є забезпечення рівня обслуговування користувачів на основі раціонального співвідношення ціна/якість. Базою такого управління є моніторинг, оцінювання і контроль параметрів якості на всьому шляху контенту до користувача в розрізі мережі, медіапотоків, контенту та управління. Ядром такого управління є імпакт-аналіз.

Проблема управління потоками даних. Для управління мережними ресурсами ІТ-інфраструктури можна використовувати пріоритезацію пакетів, оптимізацію джитера і затримок, зменшення втрат пакетів і виділення смуги. Але управління потоками мережі з огляду на інтеграцію сервісного і процесного підходів та тенденції розвитку ІТ-галузі видається найбільш перспективним. Для забезпечення потрібної градації якості послуг проблема декомпозується на підпроблеми для окремих мереж, технологій і обладнання з врахуванням методів стиснення, роздільних здатностей та інших параметрів.

Проблема опрацювання метрик. Багаторівневість представлень ІТ-інфраструктури, наявність широкої номенклатури технологій, обладнання, ресурсів і компонентів обумовили наявність величезної кількості різноманітних метрик їх оцінювання. Тому виникає проблема впорядкування метрик, їх систематизації, зведення часткових метрик в інтегральні.

Вирішення цих проблем ускладнюють велика кількість користувачів, складність ІТ-інфраструктури, велика кількість типів і одиниць обладнання та інші чинники. Потрібні ефективні математичні моделі і методи.

Концептуальні основи управління ІТ-інфраструктурою

Орієнтована на створення єдиного універсального середовища інтегрованого управління різноманітними ІТ, розподіленими застосуваннями, ресурсами і мережним обладнанням концепція управління ІТ-інфраструктурою враховує усі аспекти цього надскладного об'єкта управління. Її принципові положення [12]:

1. Включення контуру оперативного управління ІТ-інфраструктурою в зовнішній контур управління бізнесом і внутрішній контур управління операційною діяльністю.

2. Виділення в ІТ-інфраструктурі ієрархічних рівнів (сервісів, застосувань, обчислювальних ресурсів, мережної взаємодії) і розподіл функцій, команд і методів управління між ними.

3. Виділення категорій управління (наданням і продуктивністю сервісів, сервісами і процесами, підтримка сервісів і управління ресурсами) і загальних процесів (моніторинг, аналіз, управління, оптимізація, планування).

4. Використання декомпозиційно-компенсаційного способу організації управління рівнем послуг, заснованого на декомпозиції задач управління і компенсації негативного впливу окремих чинників за рахунок виділення додаткових ресурсів критичним застосуванням [13].

5. Комплексне управління ІТ-інфраструктурою, інтеграція процесного і оперативного управління з організацією міжрівневої взаємодії на основі базової моделі дворівневої системи управління з координатором.

6. Врахування вимог бізнес-процесів.

7. Використання сервісно-ресурсних моделей (СРМ).

8. Впровадження процесно-оперативного підходу до управління рівнем ІТ-послуг.

9. Визначення універсального об'єкта моніторингу та управління (ОМУ) і робота СУІ не з реальними (РОУ), а логічними (ЛОУ) ОМУ.

10. Використання агентського підходу.

11. Застосування єдиного механізму взаємодії СУІ з агентами і ОМУ.

12. Використання універсальної схеми кодування стану ОМУ.

Декомпозиційно-компенсаційний підхід до управління рівнем ІТ-послуг. Розвиток бізнесу за рахунок множини $S = \{s_i\}$, $i = \overline{1, K}$ необхідних ІТ-послуг вимагає обґрунтованого співвідношення їх рівня якості Q і витрат C. Необхідність розкладання сукупної поведінкової моделі складної системи обумовлює декомпозицію управління [13]. Загальна схема підходу:

Етап 1. Комплексний аналіз співвідношення SLA/вартість (ціна рівня).

Етап 2. Визначення і обґрунтування інтегрального рівня SLA на основі можливих втрат провайдера внаслідок незадовільного рівня обслуговування і його витрат на надання сервісів з визначеним рівнем з урахуванням ризиків.

Етап 3. Аналіз залежності інтегрального показника SLA від базових.

Етапи 4 — 7. Декомпозиція проблема визначення мінімальної вартості досягнення рівня обслуговування на підпроблеми компенсаційного визначення цієї вартості на підмережах, сервісах, ресурсах і технологіях.

Етап 8. Облік і контроль дотримання визначених параметрів.

Управління рівнем послуг вимагає інтегрованої взаємодії процесів погодження та управління рівнем послуг і планування ресурсів (рис.1).

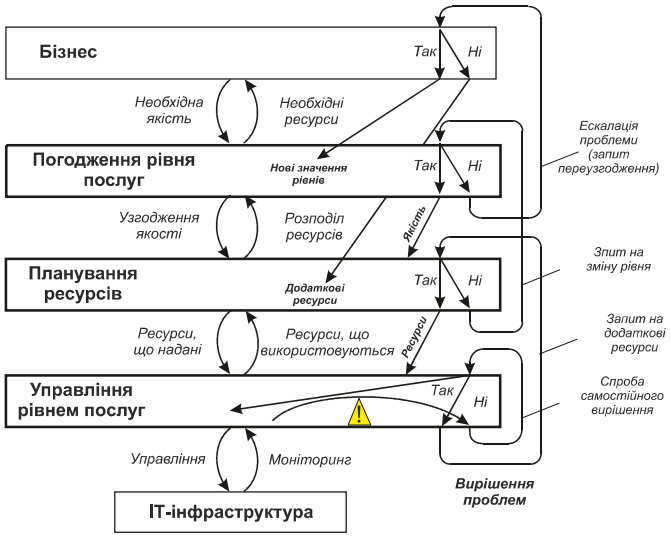


Рис. 1. Взаємодія процесів при управлінні рівнем послуг

Процес погодження рівня послуг формує множину S і матрицю $Q=\|q_{ki}\|$, де q_{ki}, $k=\overline{1,M_i}$, $i=\overline{1,K}$ — значення k-го показника якості i-ї послуги. Пара $<Q,r>$ визначається системою $<\hat{D},\hat{C},\hat{L},Q,M_o,N_0>$, де $\hat{D}$ — доходи бізнесу від ІТ, $\hat{C}$ — витрати на ІТ-інфраструктуру, $\hat{L}$— втрати бізнесу від неякісних сервісів, $Q=f_1(c)$ — витрати на підтримку якості сервісів, M_0 — ризики і N_0 —невизначеність. На основі пари $<Q,r>$, де r — виділений послугам S ресурс, розв'язуються задачі на наступному рівні. Авторами запропоновані моделі, спрямовані на досягнення $\max(\hat{D}-\hat{C})$.

Процес планування формує матрицю $\mathbf{P}=\|\rho_{ij}\|$, де ρ_{ij} — кількість виділеного послузі s_i, $i=\overline{1,K}$ ресурсу R_j, j=1,…,m. За дефіциту ресурсів підтримуються найважливіші застосування з врахуванням стану ІТ-інфраструктури і зміни інтенсивності запитів. Задача погодження рівня послуг зводиться до визначення матриці $Q=\mathcal{F}_1(S\ ,r,W_p,Z_s)$, де $W_p=\{w_i\,|\,i=\overline{1,K}\}$; w_i — важливість послуги s_i; $Z_s=\{z_i\,|\,i=\overline{1,K}\}$; z_i — запланований ступінь підтримки i-ї послуги, r — сумарна кількість ресурсів.

За надлишку ресурсів забезпечують обумовлені значення показників якості q_{ki} застосувань із множини $A=\{A_l\}$, $l=\overline{1,I}$, що підтримують послуги, за критерієм максимального вивільнення ресурсів з врахуванням їх надійності. Визначається якість $Q=\mathcal{F}_2(S\ ,\hat{r},r_r,C)$ при відомих ресурсах (задача першого роду), або ресурси $r=\mathcal{F}_3(S\ ,Q,C)$, потрібні для забезпечення заданої якості (задача другого роду). Тут r_r — резервний обсяг ресурсів, $\hat{r}=r+r_r$. Авторами запропоновані відповідні моделі і методи.

Процес управління рівнем послуг підтримує відповідність фактичних значень q^*_{ki}, $k=\overline{1,M_i}$, $i=\overline{1,K}$ показників якості послуг погодженим

$$q_{ki}-q^*_{ki}=0,\quad \forall k=\overline{1,M_i}, i=\overline{1,K}. \qquad (1)$$

Визначаються операції (зміна параметрів функціонування ІТ-інфраструктури, виділення додаткових чи вимикання незадіяних ресурсів, перерозподіл ресурсів між користувачами, ескалація проблеми на рівень планування), які забезпечують $\min(q_{ki}-q^*_{ki})$, $k=\overline{1,M_i}$, $i=\overline{1,K}$, при $q^*_{ki}<q_{ki}$, або $\min\hat{C}$, при $q^*_{ki}>q_{ki}$, $k=\overline{1,M_i}$, $i=\overline{1,K}$. Відповідні моделі наведені нижче.

Нехай управління $u^+\in U$, де U — множина управляючих дій, полягає у виділенні додаткових ресурсів застосуванню $A^*_l\in A$, для якого фактична якість $q_ф$ гірша цільової $q_ц$, $q_ф<q_ц$, а $u^-\in U$ управління, яке вилучає ресурси у застосування $A^*_l\in A$, якщо $q_ф>q_ц$. Для монотонних функцій f_{qr}, $q=f_{qa}(\hat{a})$ і $q=f_q(r,\hat{a})$, враховуючи монотонний характер залежності q_{ki}, $\forall k=\overline{1,M_i}, i=\overline{1,K}$ від r, доведено існування бажаного управління [13].

Теорема 1. Для заданих значень q_{ki}, $\forall k=\overline{1,M_i}, i=\overline{1,K}$, якщо $q^*_{ki}<q_{ki}$, існує управління $u^+\in U$, яке дозволяє забезпечити $q^*_{ki}=q_{ki}$ при $\min r$ на підтримку рівня послуг.

Обґрунтовано, що управління $u^+\in U$ знаходиться ітеративно.

Теорема 2. Якщо $\hat{r}=r+r_r$ і $q^*_{ki}<q_{ki}$ і відомі значення $\hat{a}$, то управління $u^+\in U$, яке дозволяє відновити умову $q^*_{ki}=q_{ki}$, $\forall k=\overline{1,M_i}, i=\overline{1,K}$ може бути знайдено без ітеративних процедур.

Наслідок. Якщо $q^*_{ki}<q_{ki}$ і i-а послуга підтримується застосуванням $A^*_l\in A$, а за рахунок застосувань із A, для яких $q_ф>q_ц$, можна вивільнити обсяг ресурсів $\Delta\hat{\rho}_в$, причому $\Delta\hat{\rho}_в\geq\Delta\hat{\rho}_{lj}$, де $\Delta\hat{\rho}_{lj}$ — додатковий обсяг ресурсів, який потрібно виділити l-му застосуванню,

щоб забезпечити виконання $q_{ki}^{*} = q_{ki}$, то управління $u^{+} \in U$ в умовах дефіциту ресурсів дозволяє відновити рівень *i*-ї послуги, що підтримується *l*-м застосуванням, за один прохід.

Процес погодження рівня послуг переформовує матрицю $Q' = \|q'_{ki}\|$. На її основі нижчий рівень вивільняє і виділяє ресурси послугам, для яких $q_{ki}^{*} \neq q_{ki}$, $k = \overline{1, M_i}$, $i = \overline{1, K}$. У випадку невдачі здійснюється ескалація проблеми на рівень бізнесу, який генерує матрицю $Q' = \|q'_{ki}\|$, або збільшує $r_1,...,r_m$.

Багаторівневі СУІ доцільно будувати з використанням універсальної компоненти — дворівневої системи управління з координатором.

Базова дворівнева модель. Багаторівневі ієрархічні СУІ будуються із дворівневих систем управління з координатором (рис. 2) [14].

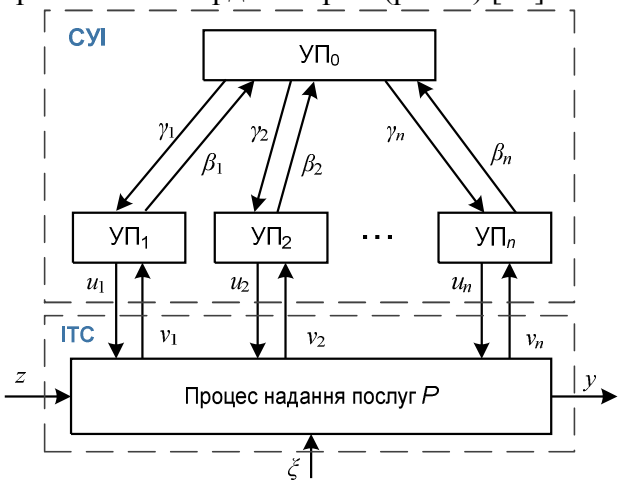


Рис. 2. Базова дворівнева модель представлення ієрархічних СУІ

Ці системи забезпечують не оптимальне функціонування ІТ-інфраструктури, а поліпшення якісних характеристик її використання. Координатор погоджує рішення підсистем з точки зору якості послуг. Процес *P* надання послуг задамо відображенням $P: U \times Z \times \Xi \to Y$, де $U = U_1 \times U_2 \times ... \times U_n$. — множина управляючих сигналів (компоненту $u_g \in U_g$ управляючого впливу *u* формує підсистема УП$_g$, $g = \overline{1, n}$); Z — множина вхідних сигналів *z* (запитів користувачів); Ξ — множина збурень ξ; *Y* — множина виходів (результатів виконання запитів). На вхід УП$_g$, $g = \overline{1, n}$ поступають сигнали: координуючий $\gamma_g \in \Gamma$ — від управляючої підсистеми верхнього рівня УП$_0$; інформаційний зворотного зв'язку $v_g \in V_g$ — від системи моніторингу.

Підсистема УП$_g$, $g = \overline{1, n}$ реалізує відображення $C_g : \Gamma \times V_g \to U_g$. Сигнал v_g є функцією $f_g : U \times Z \times \Xi \times Y \to V_g$ сигналів *u*, входів *z*, збурень ξ і виходів *y*. Управляюча підсистема УП$_0$ як координатор виробляє сигнали $\gamma_g \in \Gamma$, $g = \overline{1, n}$, реалізуючи відображення $C_0 : \mathbf{B} \to \Gamma$, де **B** — множина інформаційних сигналів $\beta = (\beta_1, ..., \beta_n)$ зворотного зв'язку. Сигнал β_g, $g = \overline{1, n}$ від УП$_g$, визначається відображенням $f_0 : \Gamma \times V \times U \to \mathbf{B}$, де $V = V_1 \times ... \times V_n$.

Координація погоджує дії підсистем для досягнення глобальної цілі системи — забезпечення максимальної якості Q ІТ-сервісів

$$\max Q \Leftrightarrow \max Q_i, \forall i = \overline{1, K} \Leftrightarrow \\ \Leftrightarrow \max q_{ki}, \forall i = \overline{1, K}, \forall k = \overline{1, M_i}, \quad (2)$$

з мінімальною вартістю C (тут $Q_i, i = \overline{1, K}$ — якість *i*-ї послуги; $q_{ki}, k = \overline{1, M_i}$ — значення *k*-го показника якості *i*-ї послуги);

Ціллю координатора є підтримка якості Q послуг на погодженому рівні з мінімальними затратами C на задіяні ресурси $Q = const\big|_{\min C}$, *локальні цілі підсистем* полягають у підтримці заданих значень параметрів функціонування ІТ-інфраструктури з мінімальними витратами.

Координація на основі постулатів сумісності не можлива, оскільки точний прогноз значень параметрів процесу *P* і аналітичні вирази для вирішення задачі координації не відомі. Тому розроблена методика синтезу координатора на основі декомпозиції глобальної задачі.

Якщо Q_i і Q_i^{*}, q_{ki} і q_{ki}^{*} — цільове і фактичне значення якості *i*-ї послуги і її *k*-го показника, ціль координатора набуде вигляду $\min \Delta Q_i = \min(Q_i - Q_i^{*}), \forall i = \overline{1, K} \Leftrightarrow \min \Delta q_{ki} = \\ = \min(q_{ki} - q_{ki}^{*}), \forall k = \overline{1, M_i}, \forall i = \overline{1, K}$,

а задача СУІ зводиться до вибору управління, яке протидіє збуренню. Координатор виробляє координуючі сигнали, щоб мінімізувати відхилення, і природно використовувати принцип управління по відхиленню. Якщо можна виміряти збурення, вплив несправностей на якість сервісів, завантаженість каналів зв'язку, обґрунтовано перехід до комбінованого управління і виконана декомпозиція координатора.

Зміни глобальної цілі управління ІТ-інфраструктурою призводять до змін пріоритетів Pr застосувань $\{A_l\}$ і цільових значень $Q_i, i = \overline{1, K}$

рівня послуг, а відображення C_0 набуває вигляду $C_0 : Pr \times Q_i \times \mathbf{B} \times V \times \Xi \to \Gamma, \quad i = \overline{1, K}$.

Вибір координуючого впливу визначається системою $< Pr, \Delta\hat{q}, \mathbf{B}, V, \Xi, \mathcal{N} >$, де $\Delta\hat{q}$ — вектор відхилень; $\mathcal{N}$ — найбільш характерна для ІТ-інфраструктури ситуаційна невизначеність. Пріоритети Pr застосувань набувають значень із множини $\{1, 2, ..., Pr_m\}$. Ступінь відхилення значень елементів вектора $\Delta\hat{q} = (\Delta q_{1,1}, ..., \Delta q_{k,i}, ..., \Delta q_{M_K,K})$ від нульових значень характеризує функція $w(\Delta q_{kl}) = (q_{ki} - q_{ki}^*) / q_{кр_{ki}}, \quad \forall k = \overline{1, M_i}, \forall i = \overline{1, K}$, де $q_{кр_{k,l}}$ — критичне значення показника якості i-ї послуги. Оскільки в аналітичному вигляді визначити відповідне відображення не видається можливим, розроблена ітеративна процедура координації [14], що передбачає участь усіх процесів, наведених на рис. 1. Рішення координатора впливають на вибір не управляючих, координуючих впливів [14], що визначають УП$_g$, $g = \overline{1, n}$, якому віддається перевага при відновленні якості послуг і які методи доцільно використовувати. Авторами розроблений комплекс відповідних методів і продукційних систем для організації такої координації.

Отже, координація базується на ітеративних процедурах поліпшення координуючих сигналів на основі аналізу результатів координації або використання зворотного зв'язку для корегування координуючого сигналу [14]. В обох випадках для визначення сигналу похибки при оцінюванні впливу запропоновані методи зведення метрик, що вимірюються на рівні процесу P, до метрик, з якими оперує координатор.

Реалізація декомпозиційно-компенсаційного підходу на основі СРМ. Авторами розроблена методологія управління рівнем ІТ-послуг, яка визначає основи розроблення, впровадження і застосування СРМ [15]. Ці моделі зображуються ієрархічними графами, вершини яких відповідають ЛОУ у вигляді ресурсів, активів та інших сервісів, а ребра — зв'язкам між ними. Типи ЛОУ і зв'язки, що використовуються для створення СРМ, визначаються будовою сервісу і цілями створення моделі.

Методика створення СРМ розроблена на основі системної методології в рамках визначеного процесу управління ІТ-інфраструктурою. Вигляд моделі залежить від цілей процесу, для якого створюється модель. Одна із основних задач СРМ полягає у організації моніторингу параметрів якості ІТ-послуг [15]. При цьому на практиці якість ІТ-послуг визначається комплексом вимірюваних характеристик, що відповідають вимогам користувачів.

Методологія управління рівнем послуг на основі СРМ надає розроблені авторами моделі, методи, методики і алгоритми для виконання таких комплексів робіт з управління рівнем послуг: формування каталогу ІТ-сервісів; визначення параметрів якості ІТ-послуг; проектування СРМ; визначення методів отримання значень показників якості сервісів; моніторинг і аналіз значень параметрів якості; управління ІТ-інфраструктурою з підтримки рівня послуг.

Реалізація декомпозиційно-компенсаційного підходу на основі СРМ передбачає комплексне управління рівнем ІТ-послуг, тісний взаємозв'язок процесного і оперативного управління з закріпленням за СУІ ролі зв'язуючої ланки для інтеграції управління [16–21]. Процесне управління корегує цільові значення $q_{b,ki}$, $\forall b, k, i$ показників якості сервісів і стимулює вирішення проблем в ІТ-інфраструктурі. У контурі оперативного управління аналізуються ризики, планується зростання/скорочення кількості користувачів, обсягів даних збереження, компенсуються піки завантаженості з врахуванням змін умов надання послуг. Його Інструментом є перерозподіл ресурсів ІТ-інфраструктури з урахуванням пріоритетів бізнесу, критичності послуг і прогнозу вимог до їх доступності з забезпеченням прийнятого рівня якості.

Авторами розроблені математичні моделі, методи і відповідні алгоритми для реалізації усіх функцій управління ІТ-інфраструктурою. У статті розглядаються лише моделі і методи для: розподілу ресурсів і навантаження; імпакт-аналізу; управління потоками в мережах; зведення метрик ІТ-інфраструктури і її складових.

Математичні моделі і методи управління ІТ-інфраструктурою

Управління ресурсами і навантаженням. Класифікація моделей управління ресурсами і навантаженням ІТ-інфраструктури за ознаками рівень абстракції ресурсів (абстрактні, конкретні), етапом ЖЦ (узгодження, планування, впровадження і управління ІТ-сервісами), призна-

чення (управління власною ІТ-інфраструктурою, надання ІТ-послуг), технологія (виділені сервери, серверна віртуалізація, віртуальний хостинг, хмарні обчислення), забезпечення процесу ресурсами (цілком або ніяк, допускається часткове забезпечення) визначає потребу в $n \cdot 100$ моделях управління ресурсами і навантаженням. Авторами розроблені моделі для всіх важливих класів. Розглянемо деякі з них.

Розподіл абстрактних ресурсів. Базова модель розподілу обмежених ресурсів враховує бізнес-процеси $Z_1, ..., Z_n$ ($W = (w_1, ..., w_n)$ – вектор коефіцієнтів їх важливості), що використовують ресурси $R_1,...,R_m$ ІТ-інфраструктури. Нехай $\|p_{ij}\|$ – матриця потреб бізнес-процесів у ресурсах, де p_{ij} відповідає приведеній кількості потрібного для процесу Z_i ресурсу R_j, $R = (r_1,..., r_m)$ – вектор обмежень на ресурси [22].

Якщо процес забезпечується ресурсами цілком або ніяк, вводиться булева змінна $X = (x_1,..., x_n)$, яка визначає підтримку бізнес-процесів. Тоді критерій набуває вигляду:

$$\max \sum_{i=1}^{n} x_i \cdot w_i \,, \qquad (3)$$

і мають виконуватися ресурсні обмеження:

$$\sum_{i=1}^{n} x_i \cdot p_{ij} \le r_j \,, j = 1, ..., m. \qquad (4)$$

Якщо допускається часткове забезпечення ресурсами, вводиться неперервна змінна x_i, яка визначає рівень підтримки бізнес-процесу Z_i. Тоді критерій (3) і ресурсні обмеження (4) залишаються і додається обмеження

$$0 \le x_i \le 1, \, i = 1,\ldots, n, \qquad (5)$$

Також розглянуто ряд інших корисних випадків, наприклад з: стохастичними оцінками наявних ресурсів, для якого визначено умови зведення до еквівалентної детермінованої задачі лінійного програмування; врахуванням надійності ресурсів, зведений до лінійної задача булевого програмування [23]: максимізувати (3) при виконанні обмежень (4) і вимоги до надійності [24]

$$\sum_{j=1}^{m} x_i \cdot d_{ij} \cdot \ln q_j \ge \ln s_i \,, \text{ для } i = 1,..., n, \qquad (6)$$

де $s_1, ..., s_n$ – вимоги до показників надійності бізнес-процесів, $i = 1,..., n$, $q_1,..., q_m$ – показники надійності ресурсів, $j=1, ..., m$; $D = \|d_{ij}\|$ – булева матриця потреб бізнес-процесів у ресурсах.

Розподіл конкретних ресурсів. При плануванні і управлінні ресурсами і навантаженням для підтримки бізнес-процесів реальними ресурсами з врахуванням технічних і географічних обмежень інтегрований ресурс R_j розглядається як група конкретних ресурсів одного типу але різного об'єму та надійності. У цьому класі запропоновано ряд лінійних і нелінійних булевих і неперервних задач розподілу ресурсів і навантаження з урахуванням ресурсних, надійнісних, технічних і географічних обмежень [25, 26].

Розподіл ресурсів і навантаження ЦОД. Розроблено підхід [27] до управління розподілом ресурсів ЦОД при віртуальному хостингу, серверній віртуалізації та хмарних обчисленнях. Розв'язано низку задач, постановки яких враховують моделі хостингу (віртуальний; віртуальні приватні сервери; виділені сервери; колокейшн), архітектуру побудови систем, технології побудови і організації доступу до прикладного програмного забезпечення, технології віртуалізації і кластеризації.

Моделі і методи управління навантаженням і ресурсами для виділених серверів. Сервери $N_1,\ldots,N_n$ обслуговують клієнтів $K_1,\ldots,K_m$, надаючи їм доступ до застосувань. Перші характеризуються набором технічних параметрів, а другі – набором вимог. Сервер N_i, i=1,…,n, має чотири параметри, які разом характеризують його потужність: Ω_i – процесорна місткість; місткість Γ_i оперативної пам'яті і Φ_i жорстких дисків; Λ_i – місткість каналів. Клієнт K_j, j=1,…,m, має вимоги до процесорної місткості серверів (ω_j), оперативної пам'яті (γ_j), пам'яті жорстких дисків (φ_j) і місткості каналів (λ_j) серверів, на яких розташовані застосування, які обслуговують користувачів клієнта K_j [28].

Характеристики серверів і вимоги застосувань поділяються на залежні (Ω_i, ω_j, Λ_i, λ_j) і не залежні від навантаження (Γ_i, γ_j, Φ_i, φ_j). Вимоги до залежних від навантаження параметрів серверів визначаються сумарними вимогами користувачів клієнта, а вимоги до незалежних від завантаження параметрів серверів – сумарними вимогами усіх застосувань клієнта. Можливості сервера N_i розмістити застосування j визначає відповідний елемент булевої матриці $R=|R_{ji}|_{m \times n}$, w_j – важливість застосування j. Запропоновано задачі планування розміщення застосувань на серверах і диспетчеризації навантаження.

Задача планування розподілу ресурсів полягає у визначенні матриці $\mathbf{X}=\|x_{ji}\|$ прив'язки клієнтів до серверів з врахуванням обмежень:

1) сумарні незалежні від навантаження вимоги застосувань сервера не перевищують його незалежні від навантаження параметри;

2) залежні від завантаження вимоги застосувань повинні бути забезпечені сумарними можливостям залежних від завантаження параметрів серверів, на яких вони розміщуються;

3) обмеження, визначені матрицею R і вектором технологічних вимог C.

Авторами розроблено декілька моделей із відмінностями у забезпеченості серверами за різними критеріями (максимальна сумарна важливість застосувань, що отримали ресурси; рівномірне завантаження серверів; максимальні сумарні експлуатаційні витрати на вивільнені сервери; мінімальна сумарна вартість реалізації операцій переходу від попереднього розміщення). Наприклад, задача вибору мінімальної за витратами на їх підтримку підмножини серверів, які зможуть задовольнити вимоги існуючих користувачів застосувань (тут s_i – експлуатаційні витрати на підтримку сервера N_i):

$$\min \sum_{j=1}^{m} \sum_{i=1}^{n} s_i \cdot x_{ji} . \qquad (7)$$

за обмежень:

$$\begin{gathered} \omega_j x_{ji} \le \mathbf{\Omega}_i,\ i = 1,\ldots,n, j = 1,\ldots,k;\ \lambda_j x_{ji} \le \mathbf{\Lambda}_i, \\ i = 1,\ldots,n, j = 1,\ldots,k;\ \gamma_j x_{ji} \le \mathbf{\Gamma}_i,\ i = 1,\ldots, n, \\ j = 1,\ldots,k;\ \phi_j x_{ji} \le \mathbf{\Phi}_i,\ i = 1,\ldots,n, j = 1,\ldots,k; \\ \sum_{j=1}^{m} x_{ji} \le 1,\ i = 1,\ldots, n;\ \sum_{i=1}^{n} x_{ji} = 1,\ j = 1,\ldots, m; \\ \delta_j \le \mathbf{\Delta}_i,\ i = 1,\ldots, n;\ x_{ji} \le R_{ji},\ i = 1,\ldots, n, j \\ =1,\ldots,k. \end{gathered} \qquad (8)$$

Ці задачі є лінійними та нелінійними варіантами задач булевого програмування, що виправдовує використання евристик або методів «м'яких» обчислень, насамперед генетичних алгоритмів, модифікованих для отримання контрольованого наближення до результату.

<u>*Модель серверної віртуалізації.*</u> На фізичному сервері функціонує декілька віртуальних серверів S_i, i=1,...,n, у кожному з яких встановлене одне чи декілька застосувань A_{ij}, j=2,...,m_i; m_i >1. Нехай r_k – кількість ресурсів типу R_k, встановлена на фізичному сервері, r_{ki} – кількість ресурсів типу R_k, призначена віртуальному серверу S_i, r_{kij} – кількість ресурсів типу R_k, що споживається застосуванням A_{ij}. Вочевидь, має виконуватись така умова:

$$\sum_{i=1}^{n} \sum_{j=1}^{m_i} r_{kij} \le r_k,\ k = 1,..,l . \qquad (9)$$

Для множини K_o типів ресурсів, що не можуть динамічно перерозподілятись між віртуальними серверами, мають виконуватись умови:

$$\sum_{j=1}^{m_i} r_{kij} \le r_{ki},\ i = 1,...,n,\ \ k \notin K_0, \qquad (10)$$

$$\sum_{i=1}^{n} r_{ki} \le r_k\ ,\ k \notin K_0 . \qquad (11)$$

Якщо булева або неперервна змінна x_p, визначає ступінь підтримки, а w_p – важливість сервісу Z_p, p=1,...,z, то критерій набуде вигляду

$$\max \sum_{p=1}^{z} x_p \cdot w_p . \qquad (12)$$

Авторами розроблено ряд задач булевого і лінійного програмування для управління розподілом ресурсів та навантаження на етапах планування, управління і диспетчеризації [29–31].

Група задач управління для таких можливостей зупиняти застосування і віртуальні сервери для вивільнення ресурсів: не можна зупиняти ні перші, ні другі; можна зупиняти перші, але не другі; можна зупиняти і перші, і другі; можна зупиняти і перші, і другі (змінні неперервні).

Група задач планування: потрібно вибрати конфігурацію фізичного сервера і розподілити ресурси між віртуальними серверами та застосуваннями; при визначеній конфігурації сервера потрібно розподілити ресурси між віртуальними серверами та застосуваннями так, щоб найбільш ефективно підтримати сервіси у випадку дефіциту і надлишку ресурсів.

Задача диспетчеризації полягає у розподілі клієнтських запитів між віртуальними серверами пропорційно до значень змінних x_p, які були визначені на етапах планування та управління і розв'язується на основі системи правил.

<u>*Моделі і методи управління ресурсами і навантаженням для випадку віртуалізації серверів з можливістю міграції.*</u> Нехай r_{ki} – кількість ресурсу типу R_k сервера S_i; p_{kj} – потреби ВМ V_j у ресурсах типу R_k для забезпечення заданого рівня SLA; x_{ij} – булева змінна, яка визначає, чи встановлена ВМ V_j на сервері S_i. Оскільки кожна ВМ одночасно розташована не більше, ніж на одному сервері, має виконуватись умова:

$$\sum_{i=1}^{n} x_{ij} \le 1,\ j = 1,...,m . \qquad (13)$$

Для нормального функціонування ВМ повинні бути забезпечені достатнім обсягом ресурсів серверів, на яких вони розташовані:

$$\sum_{j=1}^{m} x_{ij} p_{kj} \le r_{ki}, k = 1,...,l, i = 1,...,n. \quad (14)$$

Авторами розроблені моделі для розташування ВМ у випадку, якщо:

неможливо забезпечити усі ВМ ресурсами і визначається їх розташування, що підтримує сервіси з найбільшою сумарною важливістю:

$$\max \sum_{j=1}^{m} \sum_{i=1}^{n} x_{ij} w_j . \quad (15)$$

при обмеженнях (13), (14);

кількість наявних ресурсів суттєво перевищує потреби ВМ і визначається їх розташування ВМ, що забезпечує найменше енергоспоживання за рахунок вимкнення живлення серверів, які не використовуються:

$$\max \sum_{i=1}^{n} d_i e_i . \quad (16)$$

при обмеженнях (14), (17), (18):

$$\sum_{i=1}^{n} x_{ij} = 1, j = 1,...,m, \quad (17)$$

$$d_i = \prod_{j=1}^{m} \overline{x_{ij}}, j = 1,...,m . \quad (18)$$

Тут e_i позначає енергоспоживання сервера S_i коли на ньому не працює жодна ВМ.

Для цих задач застосовувалися методи булевого програмування – послідовного аналізу варіантів та ін. Врахування специфіки задачі дозволило суттєво зменшити перебір за рахунок симетричних рішень, однакових конфігурацій серверів тощо. Використання для планування наступних кроків наближених швидких алгоритмів теж дозволило зменшити перебір.

Моделі управління ресурсами і навантаженням для випадку надання сервісів IaaS провайдерами хмарних обчислень. Якщо сервіси IaaS планують у нодах, при укладанні договору фіксуються гарантовано (знаходитись у розпорядженні замовника в будь який момент часу) і додаткова (виділяється при потребі за наявності зайвих ресурсів) кількість нод [32, 33]. При цьому гарантовані ноди оплачуються за весь період, а додаткові – лише по факту їх використання. Оскільки додаткові ноди можуть виділятися замовнику з того ж фізичного сервера, де базується його ВМ, інструментом розподілу ресурсів стає міграція ВМ між серверами і виникає потреба у їх ущільненні. Нехай r_i – ресурси сервера S_i, у нодах, p_{0j} і p_j – кількість гарантованих і додаткових нод для ВМ V_j, x_{ij} – булева змінна, яка визначає, чи встановлена ВМ V_j на сервері S_i. Якщо можливості замовляти додаткові ресурси не надається, тобто $p_j = p_{oj}$, для $j = 1, ..., m$, проблема зводиться до описаних вище задач. Також мають виконуватися такі умови:

$$\sum_{i=1}^{n} x_{ij} = 1, j = 1,...,m \quad (19)$$

$$\sum_{j=1}^{m} x_{ij} p_{oj} \le r_i, i = 1,...,n . \quad (20)$$

За наявності вільних ресурсів провайдер:

– у випадку суттєвого надлишку ресурсів визначає найщільніший розподіл, щоб вивільнити частину серверів з метою економії електроенергії. При цьому мають бути задоволені усі додаткові потреби замовників у ресурсах

$$\sum_{j=1}^{m} x_{ij} p_j \le r_i, i = 1,...,n . \quad (21)$$

Тоді задача формулюється: максимізувати (16) при обмеженнях (19), (20);

– у випадку несуттєвого перевищення наявними ресурсами вимог користувачів визначає найщільніший розподіл з менш суворими вимогами (20) замість (21). За критерій приймаємо $\min \sum_{i=1}^{n} \left| r_i - \sum_{j=1}^{m} x_{ij} p_j \right|$, а обмеження – (19), (20).

Керовані і адаптивні генетичні алгоритми для задач управління ресурсами і навантаженням ІТ-інфраструктури

Використання традиційних методів розв’язання сформульованих задач розподілу ресурсів і навантаження виявилося неефективним через їх велику вимірність. Виправданою виявилася комбінована стратегія розв’язання зазначених задач – використання традиційних методів для задач невеликої вимірності і евристичних та генетичних алгоритмів для задач великої вимірності. Але експериментальні дослідження показали, що використання класичного генетичного алгоритму не забезпечує достатньої швидкості зміни значення функції пристосованості. Так, при розподілі обмежених ресурсів початкова популяція створювалась за допомогою рандомізованого емпіричного «жадібного» алгоритму, використовувався одноточковий кросинговер, імовірність мутації була фіксована і в різних експериментах набувала значення від 0,01 до 0,05. Значення функції пристосованості були досить далекими від оптимальних і

повільно зростали. Заміна одноточкового кросинговеру на двоточковий, підбір розміру популяції та інші технічні прийоми суттєво не покращили ситуацію. Позитивний вплив на якість знайденого рішення за рахунок наявності більшого об'єму генетичного матеріалу нівелювався збільшенням часу пошуку цього рішення, оскільки, по-перше, збільшується кількість епох, потрібна для випадкового використання в операторі кросинговеру підходящих одна до одної особин, а по-друге, час роботи оператора селекції, який використовується на початку кожної епохи, знаходиться у прямій залежності від розміру популяції. Було встановлено, генетичні алгоритми здатні конкурувати з іншими методами, забезпечуючи достатньо високу ефективність, однак їх ефективність суттєво залежить від кодування рішень, налаштування параметрів, реалізації операторів. Не існує строгих механізмів чіткого передбачення результатів для проблем, які розв'язуються за допомогою генетичних алгоритмів.

Тому виникла необхідність розроблення методології управління збіжністю генетичних алгоритмів для надання їм здатності ефективно розв'язувати різноманітні задачі. Авторами було розроблено керований і адаптивний варіанти генетичного алгоритму [34–36].

Керований генетичний алгоритм. Концептуально він відрізняється від класичного генетичного алгоритму механізмом, який, виходячи із розмірності задачі і заданої точності рішень, знаходить потрібний компроміс між кількістю кроків алгоритму і точністю рішень. Керований генетичний алгоритм враховує особливості задачі, характеристики попередньої популяції і послідовності найкращих рішень з метою визначення такої послідовності операторів кросинговеру, мутації та вибору, яка забезпечує найшвидшу збіжність. Основна ідея підходу полягає в застосуванні системи правил управління операторами вибору, кросинговеру і мутації в процесі розв'язання конкретної задачі на основі поточних результатів, історії і обчислення додаткових параметрів. Система правил підтримує баланс між розширенням дослідження простору пошуку і ефективним використанням усіх корисних для одержання найкращого рішення шим. Правила працюють таким чином, щоб наближення до найкращого рішення було контрольованим. Якщо кроки наближення досить великі, то необхідно прискорювати пошук, а якщо вони сповільнюються, то необхідно розширити використання корисних шим. Одночасно правила повинні утворити бар'єр на шляху збіжності до неоптимальних рішень.

У базовому варіанті генетичного алгоритму отримання популяції нової епохи відбувається шляхом послідовного виконання операторів кросинговеру (C), мутації (M) та селекції (S). Певного роду селекція відбувалась також перед кросинговером при відборі за певним принципом особин, що братимуть у кросинговері участь. Подібна організація процесу, хоча і давала збіжність пошуку, але не гарантувала оптимальності розв'язку. Це пов'язано із наявністю проблеми балансу «дослідження» і «використання», яка полягає у балансуванні між використанням вже існуючого матеріалу (закріплення оптимальних схем розв'язку за допомогою кросинговеру) та дослідженням нових оптимумів, що не входять до вивченого простору (отримання якісно нових схем за допомогою використання мутації). Велика кількість мутацій може призвести до руйнування схеми кращого представника поточної популяції, в той час як використання кросинговеру призводить до підвищення швидкості збіжності процесів пошуку до оптимуму (може бути як «локальним» так і «глобальним»).

На відміну від базового варіанту керований генетичний алгоритм забезпечує балансування між «дослідженням» і «використанням» за рахунок системи правил, яка визначає вибір наступного оператора. Кожна окрема епоха процесу пошуку присвячена, або «використанню» існуючого матеріалу (закріпленням оптимальних розв'язків), або «дослідженню» нових областей простору розв'язків. Селекція у кожному з цих випадків залишає у складі популяції нової епохи лише кращих представників із сукупності представників минулої епохи та отриманих на їх базі нащадків. У випадку, коли кількість отриманих нащадків буде занадто мала, сукупність, з якої робитиметься відбір, поповнюватиметься додатковим вливанням випадково згенерованих представників. Система правил формується на основі таких параметрів:

коефіцієнт приросту популяції $k_{п.п.} = \frac{l}{n}$, що характеризує її вироджуваність (тут l – кількість отриманих нащадків, n – розмір популяції);

перспективність популяції $\rho = \frac{f_{\max}(x)}{f_{cp}(x)}$, де $f(x)$ – цільова функція або функція пристосованості, а $f_{cp}(x)$ – її середнє значення для поточної популяції;

швидкість збіжності $\Delta_{i-1,i} = \rho_{i-1} - \rho_i$, , що характеризує тенденції зміни збіжності при переході від епохи до епохи.

Авторами сформульований набір правил за якими обираються операції для отримання популяції наступної епохи:

1) оператор G для «вливання» нового генетичного матеріалу

IF ($k_{n.n.} \le k_{гр}$) AND ($n \le n_{гр}$) THEN G;

2) оператор кросинговеру (C) для формування популяції наступної епохи

IF ($\Delta_{i-1,i} < 0$) THEN C;

або

IF (($\rho_i \ge \rho_{гр}$) AND ($\Delta_{i-1,i} \ge \Delta_{гр}$)) THEN C;

3) оператор мутації для отримання популяції наступної епохи і переходу від стратегії «використання» до стратегії «дослідження»

IF (($\rho_i < \rho_{гр}$) OR ($\Delta_{i-1,i} < \Delta_{гр}$)) THEN M.

Для організації управління пошуку розв'язку необхідно задатись граничними значеннями параметрів введених коефіцієнтів $k_{гр}, n_{гр}$ $\Delta_{гр}$, $\rho_{гр}$. Налаштуванням цих коефіцієнтів можна регулювати швидкість збіжності пошуку генетичного алгоритму та характер процесів що в ньому відбуваються на етапі модифікації системи правил, що дозволяє влаштувати свого роду зворотній зв'язок по оптимальності отримуваного результату.

Авторами виконані об'ємні дослідження ефективності керованого генетичного алгоритму для усіх видів задач управління ресурсами і навантаження ІТ-інфраструктури. Результати досліджень продемонстрували переваги керованого ГА, як з точки зору якості рішення, так і часу, потрібного для його отримання, для всіх видів задач.

Наприклад, дослідження ефективності запропонованих алгоритмів для задачі виділення ресурсів для ВМ нодами показало, що керований генетичний алгоритм дає кращі результати. При цьому перевага його збільшується зі зростанням розмірів ВМ відносно до розміру фізичних серверів. Порівняльне дослідження запропонованого авторами керованого генетичного алгоритму і жадібного емпіричного алгоритму було виконане для віртуалізації серверів з можливістю міграції. Ефективність варіанту генетичного алгоритму суттєво зростає при збільшенні розміру кластеру, вимог ВМ до ресурсів, та розкиду параметрів ВМ. Ефект відчувається значно слабший для невеликих ВМ та зовсім відсутній для випадку використання ВМ з однаковими вимогами до ресурсів. Керований генетичний алгоритм забезпечував підвищення ефективності використання ІТ-інфраструктури в середньому від 2% до 7 %.

Моделі й методи імпакт-аналізу в управлінні ІТ-інфраструктурою

При визначенні ступеня впливу несправностей на рівень обслуговування, кожному пристрою ІТ-інфраструктури приписують рівень відповідності нормам функціонування — severity level. Враховуючи кількість і складний характер чинників впливу несправностей на рівень обслуговування, розміри ІТ-інфраструктури і кількість користувачів, зробити це в умовах жорстких часових обмежень дуже складно.

Традиційно алгоритм розрахунку рівня обслуговування для кожного абонента будують на основі шляху доступу для нього і severity level для кожного елемента ІТ-інфраструктури. Але змінність ІТ-інфраструктури, мінливість політики провайдерів щодо оцінювання впливу роблять не ефективним чисто алгоритмічний підхід. З іншого боку, використання продукційних систем також дозволяє на основі значення severity level для кожного елемента ІТ-інфраструктури, її структури, параметрів технологій, сервісів і обладнання, визначити рівень обслуговування для кожного абонента. Тут потреби у перепрограмуванні при змінах не виникає, але застосування побудованих на символьних перетвореннях продукційних систем не задовольняє жорстких часових обмежень.

Тому авторами запропонований оригінальний комбінований продукційно-алгоритмічний підхід. Його основна ідея полягає у використанні спрощеного представлення ІТ-інфраструктури, яке містить результати опрацювання правил продукційної системи у вигляді, готовому для алгоритмічних розрахунків. Тобто продукції застосовуються заздалегідь і на відповідному шляху у графі зберігаються потрібні для розрахунків значення коефіцієнтів поширення впливу несправностей. Коли ж виникають несправності, для визначення їх впливу потрібно лише обійти граф від вершин, де виникли несправності, до вершин-користувачів

і виконати арифметичні операції над попередньо визначеними коефіцієнтами.

Аналіз впливу несправностей на рівень обслуговування поділяється на дві фази. На першій правила прив'язуються до структури мережі у спрощеному представленні механізмом виведення. На другій фазі виконується перерахунок оцінок рівня обслуговування. Підготовлене заздалегідь спрощене представлення опрацьовується алгоритмічно після надходження повідомлення про несправність з використанням швидких алгоритмів обходу графів.

За параметр якості послуг вибрана швидкість доступу до мережі. Тоді імпакт-аналіз зводиться до визначення міри зниження швидкості доступу відносно її номінального значення, викликаного несправностями. Для визначення впливу несправностей на бізнес-процеси компаній і ефективного відновлення мережі необхідні її структура і схеми підтримки сервісів ресурсами і бізнес-процесів компаній-клієнтів сервісами провайдера. Спрощене представлення включає лише важливі для аналізу аспекти ІТ-інфраструктури:

1) топологічну схему мережі, задану графом, вершини якого відповідають вузлам мережі, а ребра — лініям зв'язку між вузлами;

2) модель вузла, яка описує проходження інформаційних потоків через нього і враховує реакцію обладнання і технологій на несправності;

3) систему правил, які описують особливості поширення впливу несправності, враховуючи особливості технологій, обладнання і функціонування мережевого вузла і його взаємодію з іншими вузлами.

Авторами розроблено базовий, швидкий і перспективний методи імпакт-аналізу на основі запропонованого продукційно-алгоритмічного підходу [37].

Основу ***базового методу*** складають: структурно-функціональна схема оцінювання впливу несправностей на якість сервісів і рівень обслуговування; моделі визначення послідовності дій і потрібних для їх виконання даних; механізм підключення моделей; моделі уточнення чинників, що визначають поширення впливу несправностей; система правил поширення впливу несправностей в залежності від її структури, сервісів, технологій, тощо [38].

Різноманіття сервісів і технологій вимагає побудови моделі для кожного сервісу чи їх групи. Для деталізації запропонованого підходу були вибрані мережі доступу в ІТ-інфраструктурах провайдерів, високу трудомісткість і вимогливість до ресурсів задачі імпакт-аналізу в яких обумовлюють: велика кількість абонентів (до $n{\cdot}10^6$ юридичних і фізичних осіб); громіздкість мереж (до десяти і більше ієрархічних рівнів, $n{\cdot}10^4$ кількість типів і $n{\cdot}10^7$ одиниць обладнання і технологій); складний характер залежностей впливу несправностей на якість ІКС; складна інтерференційна картина впливу сукупності несправностей.

Для формального подання методу імпакт-аналізу введемо:

1) коефіцієнт зменшення швидкості доступу $k_{i,j(n)}$, $i = 1, \ldots, I$, $j = 1, \ldots, J_i$, $n = 1, \ldots, N_{i,j}$, де j — номер вузла на i-му рівні ієрархії мережі доступу, n — номер вихідної для i,j-го вузла $U_{i,j}$ (j-й вузол мережі на i-му рівні її ієрархії) мережі лінії зв'язку, I, J_i і $N_{i,j}$ — кількості рівнів ієрархії, вузлів на i-му рівні, ліній зв'язку, що виходять з вузла $U_{i,j}$. Коефіцієнт $k_{i,j(n)}$ набуває значення в діапазоні [0,1] і визначає, у скільки разів зменшується швидкість доступу в n-й лінії зв'язку, що виходять з вузла $U_{i,j}$, відносно нормативного показника;

2) коефіцієнт $K_{i,j}$, який залежить від стану j-го вузла, $j = 1, \ldots, J_i$, i-го, $i = 1, \ldots, I$, рівня ієрархії мережі доступу $U_{i,j}$, обумовленого несправностями у мережі доступу, і визначає ступінь падіння швидкості передавання на виходах вузла $U_{i,j}$ відносно нормативної (J_i — кількість вузлів U на рівні i).

Якщо дію усіх несправностей в вузлах і лініях зв'язку можна визначити зміною коефіцієнтів $k_{i,j(n)}$, то фактична швидкість доступу в n-й лінії зв'язку, що виходять з i,j-го вузла, $Vp_{i,j(n)} = k_{i,j(n)} \times Vн_{i,j(n)}$, де $Vн_{i,j(n)}$ — нормативна швидкість доступу в n-й лінії на виході i,j-го вузла. Результат порівняння фактичної швидкості Vp_m для абонента A_m, $m = 1,\ldots,M$, з визначеним в SLA пороговим значенням допустимого зменшення швидкості для m-го абонента разом зі значимістю абонента є основою прийняття рішення щодо дій провайдера.

Коефіцієнт $K_{i,j}$ можна визначати на основі правил для типів вузлів, які враховують структурно-технологічні особливості мереж, сервісів, принципи їх функціонування, налаштування обладнання, резервування та інші чинники. Правила згруповані за наведеними на рис. 3 способами з'єднання вузлів [39].

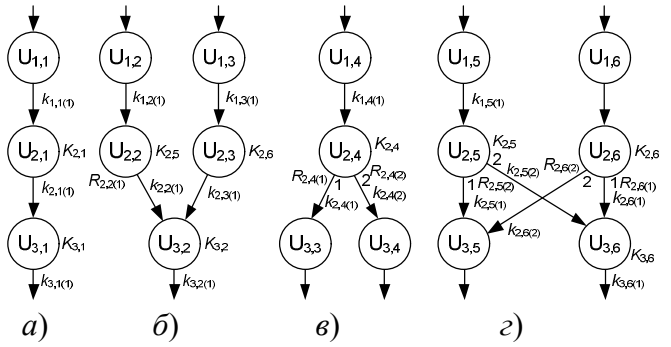


Рис. 3.Можливі способи з'єднання вузлів мереж доступу

Розглянемо, наприклад, спосіб а). При справному вузлі $U_{2,1}$ швидкість доступу у вихідній лінії визначається вхідною швидкістю і обмеженнями на швидкість передавання і приймання, яка встановлена для портів вузла $U_{2,1}$ при його конфігуруванні. Якщо вузол $U_{2,1}$ не справний, то можливі такі випадки:

1) якщо $K_{2,1} = 0$, $Vp_{2,1(1)} = 0$, незалежно від швидкості на вході;

2) якщо $K_{2,1} \neq 0$ і несправність у вузлі $U_{2,1}$ призводить до падіння швидкості у вихідній лінії на коефіцієнт $K_{2,1}$, то коефіцієнт падіння швидкості відносно номінальної у вихідній лінії зв'язку вузла $U_{2,1}$ буде становити $k_{2,1(1)} = K_{2,1} \times k_{1,1(1)}$. Падіння швидкості на виході вузла $U_{3,1}$ буде визначатися за допомогою коефіцієнта $k_{3,1(1)} = K_{3,1} \times k_{2,1(1)} = K_{3,1} \times K_{2,1} \times k_{1,1(1)}$.

Авторами розглянуті всі способи з'єднань. На найнижньому рівні ієрархії мережі доступу коефіцієнт $k_{i,j(n)}$ визначає падіння швидкість передачі на вході чи виході абонентського обладнання відносно швидкості, заданої в SLA. Тепер фактичну швидкість доступу абонента A_m, $m = 1, \ldots, M$, можна визначити як

$$Vp_m = k_m \times Vн_m, \qquad (22)$$

де k_m — коефіцієнт, відповідний $k_{i,j(n)}$ на виході абонентського обладнання, $Vн_m$ — нормативна швидкість доступу m-го абонента.

Швидкий метод імпакт-аналізу для підвищення ефективності механізму виведення використовує дерева виведення, а для обмеження пошукових просторів — методологію автоматичного доведення теорем. Ефективний перерахунок показників впливу несправностей елементів мережі доступу на інші елементи мережі забезпечується комбінованим застосуванням таких засобів: індексація елементів робочої області; структурування системи правил шляхом побудови дерев описання простих умов; прив'язування правил до вузлів спрощеного представлення мережі доступу; використання швидких алгоритмів обходу графів. Для ефективної реалізації швидкого методу, побудованого на збереженні і опрацюванні спрощеного представлення у оперативній пам'яті [40], авторами розроблені ефективні способи врахування резервних вузлів і вирішена проблема розрахунку петель за рахунок використання еквівалентних перетворень. Результати експериментального дослідження швидкого методу для різних варіантів структур мереж, сервісів, кількостей користувачів, різних характеристик обладнання і технологій показали, що для мереж з 0,5 млн вузлів час створення спрощеного представлення склав трішечки більше хвилини, а час перерахунку коефіцієнтів поширення несправностей — $n \cdot 100$ mc.

Перспективний метод імпакт-аналізу використовує комплекс прийомів пришвидчення розрахунків: структурування представлень мережі, стану і правил виконання розрахунків; обмеження сегменту мережі, на який вплинули несправності; оцінювання впливу несправностей на класи користувачів і перенесення його на користувачів за належністю до класу; оцінювання у порядку спадання важливості користувачів; використання попередньо підготовлених варіантів розрахунків, комбінуючи які можна охопити усі класи, на які вплинули несправності; використання представлення мережі, яке дозволяло б зберігати усі необхідні для проведення розрахунків дані у оперативній пам'яті і дозволяло б паралельне виконання розрахунків для визначення впливу несправностей.

ІТ-інфраструктура подається як система точок впливу (елементів розгалуження мережі, які впливають на функціонування пов'язаних з ними елементів, сегментів або сервісів) і відповідних їм сегментів впливу та коефіцієнтів розрахунку, що дозволяє задати напрям і об'єм необхідних обчислень для кожної точки впливу.

На основі наведених концепцій для розв'язання задачі імпакт-аналізу були розроблені алгоритми і відповідні реалізації типів *A* та *B* [39]. *Алгоритм типу A* комбінує «пошук в глибину» з «пошуком в ширину», що може давати складність $O(an)$, де n — кількість вершин, $a > 1$. Загальна складність алгоритму становить $O(n+m)$, де n — кількість вершин, m — кількість ребер, а на створення і сортування масивів потрібен час $O(m \cdot \ln(m))$. Оцінка пам'яті, необхідної для виконання становить $O(d)$, де d — обсяг буфера в оперативній пам'яті, необхідного для збереження чергової порції даних із бази даних. *Алгоритм типу B* працює з повним представленням, яке зберігається на дисках,

системою правил і спрощеним представленням, які зберігаються в оперативній пам'яті. Формування графа у найгіршому випадку виконується за час $O(n+m)$. Час виконання не перевищує $O(n)$. Оцінка пам'яті, необхідної для виконання: $O(n^2)$. Використовуючи ідею порівневого обходу дерева розповсюдження, можна пришвидшити час виконання. Для вирішення проблеми циклів запропонована спеціальна система позначок необхідності розрахунку чи використання обчислених раніше значення показника впливу для вузлів. Для швидкого встановлення у вузлах і їх прямих і непрямих спадкоємцях значень позначок встановлено корисну властивість цього процесу, яка дозволяє визначати значення позначок для великих підграфів. Дійсно, зміна значення показника впливу у вузлі u впливає лише на всіх прямих і непрямих спадкоємців і може вплинути на його батьківський вузол v, якщо u «живиться» тільки від резервних ліній і v є саме тим батьківським вузлом, від якого резервною лінією до u надходило живлення. Тоді час виконання можна оцінити складністю алгоритму позначення вершини u і її прямих і непрямих спадкоємців. Вона у середньому становить $O(nu(t))$, де $nu(t)$ — кількість спадкоємців u, які позначені як t, а в найгіршому випадку — $O(mu(t))$, де $mu(t)$ — кількість ребер, які виходять із спадкоємців u). Оцінка необхідної пам'яті: $O(n+m)$.

Виконано експериментальне дослідження запропонованих алгоритмів типу B, типу A, базового методу за об'ємом використаної оперативної пам'яті та часом обчислень. Різноманітні випадки виходу з ладу (частково чи повністю) окремих елементів ІТ-інфраструктури генерувалися випадковим чином. Розрахунки виконувалися з різними варіантами структур мереж доступу, сервісів, кількостей клієнтів, характеристиками обладнання і технології. Вхідні дані генерувалися випадковим чином. Алгоритми типу B значно швидші ($n\cdot 10$ мс на 0,5 млн вузлів), а їх потреби в оперативній пам'яті ($n\cdot 10$ Мб на 0,5 млн вузлів) не надто великі з огляду на можливості сучасних серверів.

Моделі і методи управління інформаційними потоками в ІТ-інфраструктурі

Управління ресурсами мережної складової ІТ-інфраструктури з метою зниження перевантажень, які призводять до погіршення якості послуг, доцільно зводити до управління потоками. За умови низької динаміки характеристик трафіка можна визначити дискретні часові інтервали (інтервали управління T_y), протягом яких параметри трафіка змінюються незначно, а стан мережі можна вважати стаціонарним. У такому випадку вирішити проблему перевантажень з періодом T_y можна як задачу математичного програмування, яка не містить час в явному вигляді [41].

Задача полягає в тому, щоб підтримувати значення фактичної завантаженості $R_C^{'}$ максимально близьким до максимально досяжної завантаженості R_C, не допускаючи перевантаження мережі, тобто управляти навантаженням так, щоб $K_c \to 1$. Якщо необхідно врахувати вимоги бізнес-процесів, критерієм ефективності може бути мінімальна затримка τ_{cp} передавання даних. Тому реалізовані такі політики управління потоками: надання доступу до мережних ресурсів у повному обсязі; обмеження доступу до мережних ресурсів; відмова у доступі бізнес-процесу до мережних ресурсів.

Виникає потреба у методі управління інформаційними потоками в IP-мережі корпоративної ІТ-інфраструктури, який би враховував вимоги бізнес-процесів, значимість інформації, що передається, для підтримки бізнес-процесів, і був здатен адаптуватися до мінливості умов передавання, пріоритетів бізнес-процесів і поточного стану мережі (завантаженості каналів зв'язку і відмов мережного обладнання).

Авторами розроблений такий метод для управління інформаційними потоками в мережах з комутацією пакетів. Концептуально управління здійснюється таким чином, щоб у випадку загрози виникнення мережних перевантажень обмежувати або забороняти передавання потоків, які генеруються низькопріоритетними застосуваннями.

Нехай кожний бізнес-процес b_k із їх множини $B=\{b_k, k=\overline{1,K}\}$ породжує множину інформаційних потоків $\lambda_k^P=\{\lambda_{kl}^P\}$, $k=\overline{1,K}$, $l=\overline{1,L_k}$. Для кожного потоку λ_{kl} задається пріоритет Pr_{kl}, $k=\overline{1,K}$, $l=\overline{1,L_k}$ із множини $\Pi=\{1,...,Pr_{\max}\}$. Легко визначити сумарний трафік Λ^P усіх потоків і Λ^U потоків, не пов'язаних безпосередньо з підтримкою бізнес-процесів. Трафік Λ^U обмежується або блокується СУІ у першу чергу, якщо він не обмежений або не заборонений корпоративною політикою використання мережних ресурсів. Кожному його потоку або типу можуть бути приписані

пріоритети із множини Π. Трафік Λ^T, що залишився, є технологічним. Його потокам або застосуванням, що їх генерують, можуть бути приписані пріоритети із множини. Сумарний трафік Λ^{Σ} корпоративної ІТ-інфраструктури становить $\Lambda^{\Sigma} = \Lambda^P + \Lambda^U + \Lambda^T$, причому він не має перевищувати сумарної пропускної здатності C^{Σ} мережі $\Lambda^{\Sigma} \le C^{\Sigma}$. Оскільки виконання цієї умови не виключає перевантажень окремих каналів, доцільно управляти потоками так, щоб виконувалася умова $\Lambda_i^{\Sigma} \le C_i$ для всіх $i = \overline{1, I}$, де I — максимальна кількість каналів зв'язку, Λ_i^{Σ} — сумарний трафік в i-му каналі, а C_i — його пропускна здатність, причому $C^{\Sigma} = \sum_{i=1}^{I} C_i$.

Вхідними даними для управління інформаційними потоками є дані щодо завантаженості каналів зв'язку, вихідними — команди управління мережним обладнанням, що обмежують або забороняють проходження окремих потоків. Управління потоком здійснюється на вході кожного із вузлів мережі. При цьому СУІ намагається пропустити через кожний вузол мережі максимальну кількість потоків з обумовленими параметрами якості обслуговування. Задаються значення пріоритетів інформаційних потоків Λ^P і Λ^T, а також вимоги до якості обслуговування цих потоків. Обмеження потоків можливе як для окремого користувача, так і для окремого мережного сервісу.

Управління здійснюється протоколами динамічної маршрутизації і оптимізацією мережних потоків. Критерієм оптимального управління потоками є підтримка максимально можливої швидкості передавання пакетів при відсутності втрат внаслідок переповнення буферів:

$$S_q - S_q(t) \ge 0, \text{ при } \tau(t) + u(t) \to \min, \quad (23)$$

де S_q — обсяг вхідного буфера, $S_q(t)$ — поточне значення черги, $\tau(t)$ — міжкадровий інтервал, $u(t)$ — змінна управління.

Основні функції управління інформаційними потоками:

1) аналіз стану завантаженості окремих каналів і мережі в цілому, полос пропускання усіх потоків кожного інтерфейсу вузлів комутації;

2) завантаженість каналу зв'язку більш ніж на 90% викликає попередження адміністратору щодо можливого перевантаження мережі чи окремих трактів;

3) завантаженість каналу зв'язку більш ніж на 95% викликає автоматичне обмеження вхідного навантаження. Для цього із списку потоків, що проходять через інтерфейс, вибираються потоки, які попали в таблицю пріоритетності, і ті, які не попали. Полоса пропускання перших обмежується до вимог, указаних в таблиці, а другі автоматично послідовно забороняються. При цьому є можливість вводити обмеження для окремих користувачів, всього потоку чи групи потоків і користувачів. Відповідні управляючі впливи (у вигляді команд мережного обладнання) передаються на вузли комутації. Цей крок виконується доки завантаженість каналу не впаде нижче заданого порогового значення. Якщо цього недостатньо обмежуються найменш пріоритетні потоки Λ^P і забороняються потоки Λ^U.

Основною перевагою запропонованого авторами рішення є здатність побудованої на його основі системи обмежувати або блокувати тільки конкретні потоки окремих користувачів. Запропонований метод управління інформаційними потоками дозволяє: підтримувати погоджений рівень найважливіших послуг; підвищити ефективність використання мережних ресурсів; оптимізувати розподіл мережних ресурсів між бізнес-процесами, розподіленими застосуваннями і користувачами; балансувати вхідне навантаження і продуктивність мережі з метою попередження падіння ефективності мережі і можливості втрати працездатності в результаті перевантажень.

Для ефективної роботи підсистеми потрібно визначити величину и навантаження кожного потоку на мережу. Для цього було виконане моделювання різних підходів до управління потоками даних в корпоративній ІТ-інфраструктурі, що використовує IP-протокол, порівняння їх ефективності у вирішенні проблеми уникнення перевантажень мережі.

Авторами побудовані моделі таких регуляторів для реалізації СУІ:

1. Апаратний регулятор виконаний по аналогії з методом управління трафіком в маршрутизаторах policing, який для обмеження навантаження на канали зв'язку затримує або відкидає пакети при перевищенні ним потоком заданого рівня. Цей регулятор відкидає трафік, який перевищує цільове значення. Його перевагами є простота реалізації та універсальність, а недоліком — здійснення управління за рахунок не регулювання, а обмеження у випадках перевантажень. Управляючі дії, які б синтезували сигнали для генераторів трафіка щодо необхідноc-

ті з метою запобігання перевантажень зменшення обсягів даних, що передаються, не виробляються.

2. Програмний регулятор у випадку необхідності обмеження потоку після порівняння цільової і фактичної його величин поступово зменшує коефіцієнт передавання, доки потік не буде достатньо обмеженим. Перевагою такого регулятора є універсальність, а недоліком — суттєвий вплив на якість управління затримок збору та оброблення інформації і поширення управляючих дій. Великі затримки викликають невідповідні управляючі дії. При цьому лінійне зменшення коефіцієнта може супроводжуватися стрибкоподібним збільшенням, коли потік уже достатньо обмежений, але через відсутність актуальної інформації продовжується його подальше обмеження.

3. Оптимальний за швидкодією регулятор, який на відміну від апаратного і програмного регуляторів, що функціонують у відповідності з алгоритмами, використовуваними мережним обладнанням, для управління інформаційними потоками використовує методи теорії автоматичного управління. Об'єктом управління є станція, підмережа або сукупність елементів, що генерують потоки інформації. Вихідною (регульованою) величиною цієї системи є завантаженість каналу зв'язку, вхідним впливом — пропускна здатність каналу зв'язку, помилкою — величина перевантаження мережі, а ціллю управління — мінімізація помилки за максимально короткий проміжок часу. Цю систему можна класифікувати як різновид дискретної системи стабілізації, оскільки при відсутності перевантажень управляти потоками не потрібно, а в супротивному випадку регулятор намагається підтримувати стабільне значення величини потоку не вище заданого вхідного впливу.

Авторами запропонована і обґрунтована методика синтезу регулятора, який забезпечує скінченій системі необхідні показники якості — відсутність статичної помилки і максимальну швидкодію — на основі передавальної функції і вибору бажаних характеристик замкненої системи.

Результати моделювання з регулятором, розрахованим в середовищі Matlab, підтвердили очікування. Порівняння перехідних характеристик систем управління потоками трафіка без регулятора і з оптимальним за швидкодією регулятором показало, що застосування запропонованих методів дозволяє отримати швидкий процес регулювання. Однак, використання цих методів обмежене чітко описаними об'єктами управління, що складно зробити в реальних мережах, а також чітко заданим видом трафіка. У інших випадках можна використовувати розроблені авторами нечіткі моделі управління трафіком.

Для підвищення якості передавання даних високопріоритетних застосувань обґрунтована можливість ефективного управління потоками інформації в мережі з врахуванням важливості даних, що передається [40]. Авторами розроблений адаптивний регулятор, який дозволяє управляти потоками інформації для уникнення перевантаженості каналів зв'язку шляхом контролю швидкості передавання даних застосувань $A_i,\ l=\overline{1,m}$ з врахуванням важливості $w_l,\ l=\overline{1,m}$ даних, що передається. Для цього розроблені модель адаптивного регулятора, алгоритми управління, обґрунтована можливість застосування існуючих протоколів управління мережними елементами і розв'язано ряд інших задач. Напрацювання управляючих дій, спрямованих на підтримку потоків високопріоритетних застосувань за рахунок обмеження потоків даних, що генеруються низькопріоритетними застосуваннями, базується на інформації щодо завантаженості мережі з врахуванням затримок на передавання контрольної інформації і управляючих дій, а також затримок на визначення і реалізацію сигналів управління.

Об'єктом регулювання є елементи ІТ-інфраструктури, що генерують потоки даних з інтенсивністю $\lambda(t)$. Задаючий вплив або вхідна величина відповідає пропускній здатності каналу передавання даних і в реальних СУІ встановлюється на рівні 95—98% пропускної здатності. Вихідна величина — інтенсивність потоку $\lambda_p(t)$ на виході об'єкта регулювання після реалізації управляючого впливу. Помилкою регулювання є величина, що відповідає перевантаженням у каналі зв'язку.

Авторами розроблена методика синтезу регулятора, який на основі аналізу вихідної величини визначає управляючу дію, що мінімізує помилку регулювання. При цьому управляюча дію не виробляється, коли генерований трафік $\lambda(t)$ не перевищує задаючий вплив. У супротивному випадку СУІ виробляє управляючі дії як система стабілізації з принципом управління по відхиленню.

Стохастична природа потоків інформації в мережах ІТ-інфраструктур унеможливлює застосування систем диференційних рівнянь. Од-

нак управління здійснюється на інтервалі T_y, значення якого вибирається із умови:

$$T_y \geq \sum (T_{изм} + T_{п1} + T_{ув} + T_{п2} + T_p), \qquad (24)$$

де $T_{изм}$ — час вимірювання завантаженості мережі; $T_{п1}$ — час доставки результатів вимірювання в СУІ; $T_{ув}$ — час визначення управляючої дії; $T_{п2}$ — час передавання даних щодо характеру і величини управляючої дії в об'єкт регулювання; T_p — час реалізації управляючої дії в об'єкті регулювання. І при виконанні умови $T_y \leq T_{вар}$, де $T_{вар}$ — інтервал між зафіксованими змінами характеристик трафіка, при визначенні передавальних функцій регулятора і управляючих дій можна виключити параметр часу.

Для об'єкта регулювання, що постійно змінюється, потрібен адаптивний регулятор. Модель трафіка можна розглядати як об'єкт регулювання зі змінними параметрами, коли у проміжках часу $T_{вар}$, що відповідають окремим стрибкам, величина трафіка не змінюється у часі. Передавальну функцію об'єкта регулювання $W_{OP}(z)$ на цих проміжках можна представити пропорційною ланкою з коефіцієнтом $K_П$, рівним величині потоку, тобто:

$$W_{OP}(z) = K_П . \qquad (25)$$

Зміна інтенсивності потоку викликає відповідну зміну значення параметра об'єкта регулювання. Для його передавальної функції змінюється не структура регулятора, а тільки значення параметрів. У цьому випадку регулятор можна розглядати як аналітична система з самоналаштуванням по характеристикам об'єкта управління. Такі зміни значень параметрів регулятора виконуються пристроєм налаштування параметрів регулятора. Авторами запропонована методика синтезу адаптивного регулятора, який працює тільки в той час, коли канал перевантажений. При допустимій завантаженості каналу зв'язку контур регулювання не задіяний, інформаційний потік $\lambda(t)$ передається без обмежень. При появі перевантажень адаптивний регулятор переходить в активний стан і управляє потоком. Елементи затримки дозволяють врахувати потрібні інтервали часу.

Тестування роботи адаптивного регулятора виконувалося на трьох типах трафіка: стохастичний трафік із ступінчатими змінами інтенсивності потоку; трафік, що генерується IRC-клієнтами; реальний e-mail трафік, що генерується пакетом OPNET на основі визначених за результатами досліджень параметрів.

Результати показали, що синтезований адаптивний регулятор дозволяє управляти інтенсивністю вхідних інформаційних потоків, виключаючи перевантаження каналів зв'язку. Регулятор обмежує інформаційні потоки за відсутності апріорної інформації про об'єкт управління, при помилці регулювання, що не перевищує 10^{-3}. Застосування запропонованих моделей і методів в СУІ дозволить управляти мережними ресурсами ІТ-інфраструктури в залежності від значимості застосувань, що генерують інформаційні потоки, обмежуючи при мережних перевантаженнях інтенсивність потоків малозначимих застосувань.

Моделі й методи зведення метрик ІТ-інфраструктури і її складових

Для реалізації процесів збору, накопичення і оброблення інформації в СУІ на основі системи метрик розроблено засоби визначення і оперування метриками, що дозволяють швидко зберігати, вибирати потрібні метрики і агрегувати існуючі метрики з метою формування і аналізу нових корисних для різних служб СУІ метрик.

Для цього запропоновано використовувати опис всіх метрик у вигляді цілісної системи, в якій кожна метрика одержує системний код, складові якого визначають її сутність, зв'язки з іншими метриками системи і алгоритми оброблення [42]. Досвід розроблення СУ для провайдерів ІКС і аналіз відомих метрик вимірювання і оцінювання мереж дозволив визначити потрібний для цього набір системних параметрів описання метрик в рамках узагальненої структури: ⟨значення⟩ ⟨тип_метрики⟩ ⟨об'єкт_ рахунку⟩ ⟨рівень_ІТ-інфраструктури⟩ ⟨компонент_ІТ-інфраструктури⟩ ⟨одиниця_вимірювання⟩. Її складовими можуть бути: значеннями — безпосередні значення метрики; типами метрики — абсолютна кількісна, відносна часткова та ін.; об'єктами рахунку — реальні елементи мережі, події, час, стани та ін.; рівнями ІТ-інфраструктури — користувачі, технології, ресурси, сервіси, продукти; компонентами — підмережі та інші важливі складові; одиницями вимірювання: всі відомі одиниці.

Системний характер запропонованого підходу підкреслює комплекс взаємозв'язків значень параметрів, який включає ієрархічні, структурні та інші відношення.

Різноманіття суб'єктивних оцінок рівні обслуговування і потреба в моделях його

об'єктивного оцінювання за наявності багатьох чинників впливу в ІТ-інфраструктурі провайдерів визначило потребу у методі агрегування показників оцінювання якості, заданих у вигляді нечітких множин [42, 43]. Нехай n метрик ІТ-інфраструктури, мереж, сервісів, технологій $K_1,K_2,\ldots,K_n$ потрібно звести у одну метрику Q. Метрики, насамперед суб'єктивні, часто є не числами, а лінгвістичними змінними, які характеризують функціями належності (ФН). Кожному значенню лінгвістичної змінної відповідає певна нечітка множина зі своєю ФН. Наприклад, якість сервісу можна визначити за допомогою змінної «*Якість*», значення якої можуть належати множині S = {*надзичайно добра* (S_1); *дуже добра* (S_2); *добра* (S_3); *середня* (S_4); *нижча середньої* (S_5); *погана* (S_6) і *дуже погана* (S_7)}. У нечіткій логіці з n входами і одним виходом міркування здійснюються за нечіткими правилами R_j вигляду:

R_j: Якщо $K_1 \in A_{1,j} \wedge \ldots \wedge K_n \in A_{n,j}$, то $Q \in S_j$, (26)

де K_i, $i = 1,\ldots,n$, Q — вхідні і вихідна лінгвістичні змінні; S_j, $A_{i,j}$ — нечіткі вихідна множина і множини вхідної лінгвістичної змінної K_i.

Оскільки в СУІ накопичується багато даних моніторингу ІТ-інфраструктури і її компонентів, будувати ФН вхідних і вихідних нечітких змінних та нечіткі правила зведення показників оцінювання якості сервісів доцільно на основі навчальних вибірок даних. Потрібен відповідний алгоритм навчання.

Авторами розроблено підхід до побудови ФН для нечіткої системи з n входами і одним виходом і правилами вигляду (26) на основі множини числових навчальних вибірок $P = \{(k_{1,j},\ldots,k_{n,j},q_j) \mid j = 1,\ldots,m\}$ з n входами і одним виходом, де $k_{i,j}$ — значення i-ї вхідної лінгвістичної змінної K_i, які складають множину $(k_{1,j},\ldots,k_{n,j})$; q_j — відповідне значення вихідної лінгвістичної змінної Q, $1 \le i \le n$ і $1 \le j \le m$ [42–45].

Система управління ІТ-інфраструктурою

Розроблені авторами методологічні основи проектування, впровадження та експлуатації СУІ реалізовано в багатьох реальних об'єктах управління [4, 46–48]. Створення СУІ базується на ІТ управління ІТ-інфраструктурою. Архітектура інструментальних засобів ІТ побудована на принципах трирівневої клієнт-серверної технології, а реалізована вона на платформі швидкого розроблення і експлуатації застосувань SmartBase [49–53]. СУІ складається з трьох програмних модулів — серверного, клієнтського та агентського. ІТ управління ІТ-інфраструктурою реалізує п'ять процесів — моніторингу, аналізу, управління оптимізації та планування. Кожний з процесів реалізується програмним модулем, що дозволяє нарощувати їх функціонал без внесення істотних змін в інші модулі, створювати версії СУІ зі скороченим функціоналом [4].

Модуль моніторингу визначає стан ІТ-інфраструктури та її елементів, використовуючи статистичний, реактивний і проактивний методи. Модуль аналізу визначає стан елементів ІТ-інфраструктури, забезпечує потрібною інформацією модулі управління, оптимізації і планування. Модуль управління вибирає управляючі дії на основі політик з урахуванням існуючих обмежень, реалізує функції координатора. Модуль оптимізації розподіляє та перерозподіляє ресурси ІТ-інфраструктури, а також вирішує завдання системної оптимізації у автоматизованому або автоматичному режимі роботи. Реалізація рішення виробляється зміною конфігурації ІТ-інфраструктури, в модуль управління надходить новий план розподілу ресурсів. Модуль планування вирішує завдання розвитку ІТ-інфраструктури. Кожен з модулів реалізує свої функції для всіх рівнів ієрархії ІТ-інфраструктури.

Впровадження результатів

Тематика робіт, спрямованих на розроблення моделей, методів і технологій управління ІТ-інфраструктурою включена в науково-технічні плани Наукового парку «Київська політехніка», навчально-наукового центру «НТУУ «КПІ»–Неткрекер», КНУ імені Тараса Шевченка, НаУКМА, ІК ім. В.М.Глушкова. Робота виконувалась в рамках ряду держбюджетних та договірних науково-дослідних та дослідно-конструкторських робіт.

Результати роботи були використані при створенні ряду систем спеціального призначення. ІТ управління ІТ-інфраструктурою та інші результати використані при проектуванні та реалізації підсистеми управління інфраструктурою ТОВ «БМС Консалтінг», при розробленні системи управління ІТ-інфраструктурою ТОВ «ІНЛАЙН ГРУП ЗАХІД», в системі управління процесами діяльності ТОВ «САП Україна», при розробленні підсистеми управління експлуатацією системи управління інформаційно-телекомунікаційною мережею ТОВ «Система мобільних платежів України», при розробленні

системи управління функціонуванням інформаційно-телекомунікаційної інфраструктури ТОВ «Спеціалізовані інформаційні послуги». Практичні результати впровадження:

1. Результати досліджень можуть бути використані при розробленні як СУІ, так і окремих підсистем і модулів.

2. Використання положень і результатів роботи дозволило на 15–30% скоротити час на розробку на проектування і реалізацію СУІ.

3. Застосування декомпозіційно-компенсаційного підходу до управління рівнем послуг і методів імпакт-аналізу дозволило скоротити час аналізу до 1 хв. і скарги користувачів на якість послуг на 30%, а витрати на експлуатацію серверів зменшити на 8–10%.

4. Впровадження моделей і методів управління ресурсами дозволило підвищити на 8–15% ефективність використання ресурсів, що виділяються для надання послуг користувачам, або обслуговувати на 15–20% більше користувачів наявними парком серверів із збереженням рівня обслуговування, скоротити витрати на експлуатацію серверного парку на 10–15%.

5. Використання методів, моделей та інструментів управління рівнем якості ІТ-послуг дозволяє збільшити на 12% дохід від надання послуг завдяки зменшенню відтоку клієнтів і втрат від штрафів за неякісне надання сервісів.

6. Методи аналізу дозволяють на 20% зменшити кількість тестових перевірок при виявленні несправних елементів ІТ-інфраструктури, на 10% знизити витрати на обслуговування, а також підвищити значення коефіцієнта готовності сервісу.

Висновки

Розроблена і обґрунтована цілісна методологія управління складним об’єктом управління – сучасними глобальними, національними і корпоративними ІТ-інфраструктурами.

Запропоновано ряд нових методів імпакт-аналізу, управління потоками в мережах, нечіткого зведення метрик та ін.

Запропоновано керований генетичний алгоритм для розв’язання широкого класу задач управління ІТ-інфраструктурою.

На основі отриманих теоретичних результатів розроблено сучасну ІТ управління ІТ-інфраструктурами.

Впровадження розроблених на основі ІТ управління ІТ-інфраструктурами СУІ на багатьох об’єктах управління продемонструвало їх переваги як в процесі розроблення (скорочення витрат і термінів), так і в процесі експлуатації (підвищення якості послуг, зменшення витрат, збільшення доходів тощо).

Список літератури

1. Крывый С.Л. Верификация программных систем. 1/ С.Л. Крывый, А.Н. Максимец // Кибернетика и системный анализ. – 2013. – № 6. – С. 3–12.
2. Крывый С.Л. Верификация программных систем. 2/ С.Л. Крывый, А.Н. Максимец // Кибернетика и системный анализ. – 2014. – № 1. – С. 14–21.
3. Ершов С.В. Мультиагентные модели на основе нечеткой логики высшего типа для высокопроизводительной среды / И.Н. Парасюк, С.В. Ершов // Проблеми програмування. – 2012. – № 2–3. – С. 260–269.
4. Ролик А.И. Тенденции и перспективы развития управления информационными технологиями / А.И. Ролик // Вісник НТУУ «КПІ»: Інформатика, управління та обчислювальна техніка. – К.: «ВЕК+», 2012. – № 55. – С. 81–109.
5. Боданюк М.Є. Управління системами збереження даних / М.Є. Боданюк, О.К. Карнаухов, О.І.Ролік, С.Ф.Теленик // Електроніка та зв’язок. – №5 (76). –2013. – С.81-90.
6. Павлов А.А. Информационные технологии и алгоритмизация в управлении / А.А. Павлов, С.Ф. Теленик // Киев. – Техника, 2002. – 344 с.
7. Теленик С.Ф. Воєнно-економічна безпека: управлінські та технічні аспекти / С.Ф. Теленик, Л.П. Артеменко, А.Є. Довгань, П.І. Терещенко// Київ: НТУУ «КПІ», 2008.– 476 с.
8. Погорілий С.Д. Методи кластерних обчислень // С.Д. Погорілий, Ю.В. Бойко, Р.І. Левченко, В.А. Мар’яновський/ За ред С. Д. Погорілого. – К: ВПЦ «Київський університет», 2013. – 416 с.
9. Погорілий С.Д. Технологія віртуалізації. Динамічна реконфігурація ресурсів обчислювального кластера / С.Д. Погорілий, І.В. Білоконь, Ю.В. Бойко// Математичні машини і системи. – №3. – 2012. – С. 3–18.
10. Крывый С.Л. Вступ до методів створення програмних продуктів/ С.Л. Крывый // Чернівці: Букрек. – 2012. – 423 с.
11. Ершов С.В. Трансформационный поход к разработке программных архитектур на основе нечетких графовых моделей / И.Н. Парасюк, С.В. Ершов // Кибернетика и системный анализ. – 2008. – № 5. – С. 139–150.

12. Ролик А.И. Концепция управления корпоративной ИТ-инфраструктурой / А.И. Ролик // Вісник НТУУ «КПІ»: Інформатика, управління та обчислювальна техніка. – К.: «ВЕК+», 2012. – № 56. – С. 31–55.
13. 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.
14. Ролик А.И. Управление уровнем услуг корпоративной ИТ-инфраструктуры на основе координатора / Вісник НТУУ «КПІ». Інформатика, управління та обчислювальна техніка. – К.: Век+. – 2013. – № 59. – С. 98–105.
15. Ролик А.И. Методика оперативного управління рівнем ІТ-послуг на основі сервісно-ресурсних моделей// О.І. Ролік // Вісник НТУУ «КПІ», Інформатика, управління та обчислювальна техніка». – №60. –2014. – С.119–129.
16. Теленик С.Ф. Логічний підхід до інтеграції програмних застосувань підтримки міждисциплінарних наукових досліджень / С.Ф.Теленик, О.А.Амонс, К.В.Єфремов, В.Т.Лиско // Наукові вісті НТУУ «КПІ». – №5 (91). – 2013. – С.53–72.
17. Теленик С.Ф. Створення гетеросистемного комплексу для поєднання різноманітних програмних систем на різних платформах / С.Ф. Теленик, Л.С. Глоба, О.В. Турчинов, Т.І. Кривець // Електроніка і зв'язок. – №20. – 2003.
18. Теленик С.Ф. Логика действий для интеллектуальных систем // Математические машины и системы. – 1999. – №2. – С.12–24.
19. Теленик С.Ф. Логики организации взаимодействия компонентов в адаптивной технологии // Системные технологии. Системы и процессы обработки информации и управления: Сб. науч. тр. – Вып.7.– Днепропетровск, 1999. – С.119–129.
20. Теленик С.Ф. Логика представления вычислительных процессов в интеллектуальной системе SmartBase // Системные технологии. Системное моделирование технологических процессов: Сб. науч. тр.– Вып.6.– Днепропетровск, 1999. – С.131–139.
21. Теленик С.Ф. Интеллектуальные системы и логики высших порядков // Радиоэлектроника, информатика, управление.– 1999.– №1. – С.96–105.
22. Теленик С.Ф. Технологія управління ІТ-інфраструктурою на основі ресурсного підходу/ С.Ф. Теленик, О.І. Ролік, М.М. Букасов // Вісник ЖДТУ. – 2008.– № 4 (47). – С. 180–189.
23. Telenyk S.F. Resource allocation and load management considering the assessed quality of the provided service and the use of agent technologies / S.F. Telenyk, A.I. Rolik, A.A. Pokotylo // 23rd International Crimean Conference "Microwave & Telecommunication Technology" (CriMiCo'2013). 9–13 September, Sevastopol, Crimea, Ukraine. – 2013. – p. 535–536.
24. Теленик С.Ф. Забезпечення процесів діяльності ресурсами з визначеним рівнем надійності в інформаційно-телекомунікаційній системі спеціального призначення / С.Ф. Теленик, О.І. Ролік, П.І. Терещенко, М.М. Букасов // Зб. наук. праць ВІТІ НТУУ «КПІ».–№ 3.–Київ:ВІТІ НТУУ «КПІ», 2006. – С. 134–138.
25. Теленик С.Ф. Управління доступом до обмежених ресурсів інформаційно-телекомунікаційної мережі АСУ військового призначення / С.Ф. Теленик, О.І. Ролік, М.М. Букасов, П.І. Терещенко // Зб. наук. праць Центрального НДІ Збройних Сил України. – 2006.– №3 (37). – С. 33–43.
26. Теленик С.Ф. Моделі управління розподілом обмежених ресурсів в інформаційно-телекомунікаційній мережі АСУ/ С.Ф. Теленик, О.І. Ролік, М.М Букасов. // Вісник НТУУ «КПІ». Інформатика, управління та обчислювальна техніка. – К.: Єкотех, 2006. – № 44.– С. 234–239.
27. Теленик С. Управління ресурсами центрів оброблення даних / С. Теленик, О. Ролік, М. Букасов, К. Крижова // Вісник Львів. ун-ту. Серія прик. матем. інформ. – Вип. 15. – 2009. – С. 325–340.
28. Теленик С.Ф. Управління навантаженням і ресурсами центрів оброблення даних при виділених серверах / С.Ф. Теленик, О.І. Ролік, М.М. Букасов, Р.В. Римар, К.О. Ролік // Автоматика. Автоматизація. Електротехнічні комплекси та системи. – 2009. – №2 (24). – С. 122–136.
29. Теленик С. Управління навантаженням і ресурсами центрів оброблення даних при віртуальному хостингу / С. Теленик, О. Ролік, М. Букасов, С. Андросов, Р. Римар // Вісн. Тернопільского держ. техн. ун-ту. – 2009. – Том 14. – № 4. – С. 198–210.
30. Теленик С.Ф. Моделі і методи розподілу ресурсів в системах з серверною віртуалізацією / С.Ф. Теленик, О.І. Ролік, М.М. Букасов, О.А Косован., О.І. Кобец // Зб. наук. праць ВІТІ НТУУ «КПІ». – Випуск № 3. – Київ: ВІТІ НТУУ «КПІ», 2009. – С. 100–109.
31. Теленик С.Ф. Моделі управління віртуальними машинами при серверній віртуалізації / С.Ф. Теленик, О.І. Ролік, М.М. Букасов, А.Ю. Лабунський // Вісник НТУУ «КПІ»: Інформатика, управління та обчислювальна техніка. – К.: «ВЕК+», 2009. – № 51. – С. 147–152.
32. Telenyk S. Qualitative evaluation method of IT-infrastructure elements functionin / S. Telenyk, Y. Dorogiy, O. Rolick, D. Halushko, M. Bukasov, A. Pysarenko // Proc. of 2014 IEEE International Black Sea Conference on Communications and Networking (BlackSeaCom 2014), Chisinau, Moldova, May 27–30, 2014. – pp.165–169.

33. Telenyk S. Models and methods of resource management for VPS hosting / S. Telenyk, O. Rolik, M. Bukasov, D. Halushko//Technical transaction. Automatic control.–Politechnica Krakowska, 2013.–vol. 4-AC. – pp. 41–52.
34. Теленик С.Ф. Генетичні алгоритми вирішення задач управління ресурсами і навантаженням центрів оброблення даних / С.Ф. Теленик, О.І. Ролік, М.М. Букасов, С.А. Андросов // Автоматика. Автоматизація. Електротехнічні комплекси та системи. – 2010. – №1 (25). – С. 106–120.
35. Теленик С. Ф. Управляемый генетический алгоритм в задачах распределения виртуальных машин в ЦОД / С. Ф. Теленик, О. І. Ролік, П. С. Савченко, М. Е. Боданюк // Вісник ЧДТУ. – 2011. – №2. – С.104–113.
36. Теленик С.Ф. Адаптивный генетический алгоритм для решения класса задач распределения ресурсов ЦОД / С.Ф. Теленик, А.И. Ролик, П.С. Савченко // Вісник НТУУ «КПІ»: Інформатика, управління та обчислювальна техніка. – К.: «ВЕК+», 2011. – № 54. – С. 164–174.
37. Теленик С.Ф. Метод определения влияния неполадок на качество сервиса в IP-сетях с протоколами RIP и OSPF/ С.Ф. Теленик, А.И. Ролик, М.В. Ясочка, М.Е. Бутенко // Проблеми інформатизації та управління: Зб. наук. праць: Вип. 1 (37). – К.: НАУ, 2012. – С. 118–126.
38. Теленик С.Ф. Аналіз підходів до визначення впливу неполадок елементів мережі доступу на якість обслуговування клієнтів/ С.Ф. Теленик, О.І. Ролік, М.В. Ясочка, К.С. Голоднов // Вісник Вінницького політехнічного інституту. – 2011. – №3. – С. 69–83.
39. Теленик С.Ф. Определение распространения влияния неисправностей в сети доступа на качество предоставляемых сервисов / С.Ф. Теленик, А.И. Ролик, М.М. Букасов, М.В. Ясочка // Вісник НТУУ «КПІ»: Інформатика, управління та обчислювальна техніка. – К.: «ВЕК+», 2009. – № 50. – С. 164–173.
40. Теленик С.Ф. Швидкий алгоритм визначення впливу неполадок елементів мережі доступу на якість обслуговування клієнтів / С.Ф. Теленик, О.І. Ролік, М.В. Ясочка, Д.В. Малюгін // Інформаційні технології та комп'ютерна інженерія. — 2010.— №2 (18).— С. 51—62.
41. Ролик А.И. Система управления информационными потоками в корпоративной IP-сети / А.И. Ролик, В.А. Иосифов // Адаптивні системи автоматичного управління. Міжвідомчий науково-технічний збірник. – Дніпропетровськ: Системні технології, 2009.– Вип. 14 (34).– С. 73–85.
42. Теленик С.Ф. Нечітке оцінювання в задачах управління рівнем обслуговування / С.Ф. Теленик, О.І. Ролік, М.В. Ясочка, О.М. Моргаль // Наукові записки УНДІЗ. – 2011. – № 2 (18). – С. 29–42.
43. 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.
44. Теленик С.Ф. Метод розподілу ресурсів між проектами / С.Ф. Теленик, П.І. Бідюк, О.А. Амонс, К.О. Крижова // І Вісник НТУУ «КПІ». Інформатика, управління та обчислювальна техніка. – К.: «ВЕК+» – 2008. – № 48. – С. 33–41.
45. Теленик С.Ф. Метод формування логічного висновку із залученням експертного комітету / С.Ф. Теленик, П.І. Бідюк, Л.О. Коршевнюк, В.С. Хмелюк // Проблеми програмування. – 2008.– № 4. – С. 73 – 83.
46. Теленик С.Ф. Система управління інформаційно-телекомунікаційною системою корпоративної АСУ / С.Ф. Теленик, О.І. Ролік, М.М. Букасов, Р.Л. Соколовський // Вісник НТУУ «КПІ». Інформатика, управління та обчислювальна техніка. – К.: «ВЕК+», – 2006. – № 45.– С. 112–126.
47. Теленик С.Ф. Модель управління розподілом ресурсів інформаційно-телекомунікаційної системи збройних сил України / С.Ф. Теленик, О.І. Ролік, П.І. Терещенко, О.В. Літвінцов // Наук.-техн. зб. ННДЦ оборонних технологій і воєнної безпеки України. – 2006. – №5 (34). – С. 117–124.
48. 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.
49. Теленик С.Ф. Семантична інтеграція різнорідних інформаційних ресурсів / С.Ф.Теленик, О.А.Амонс, К.В.Ефремов, С.В.Жук // Вісник НТУУ «КПІ», Інформатика, управління та обчислювальна техніка». – №58. –2013. – С.29-45.
50. Теленик С.Ф. Проблеми розробки адаптивних технологій створення інформаційно-керуючих систем Наукові вісті НТУУ "КПІ". – 2000. – № 1. – С.47–57.
51. Теленик С.Ф. Алгебро-логическая формализация в объектно-ориентированных технологіях/ С.Ф. Теленик, Г.Е. Цейтлин, А.А. Амонс//Проблемы программирования.–2002.–№1–2. Спец. Выпуск. – С. 136–146.
52. Матрична резолюція для клаузальних логік / С.Ф. Теленик, О.А. Амонс, Р.В. Смічик, В.С. Хмелюк // Вестник ХНАДУ. Сб. науч. тр. Вып.28, 2003. – С. 243–46.
53. Теленик С.Ф. Трансформационные преобразования и синтез классов алгоритмов / С.Ф. Теленик, Г.Е. Цейтлин, А.А. Амонс // Вісник НТУУ «КПІ»: Інформатика, управління та обчислювальна техніка, К., 2004, № 42.– С. 20–27.

УДК 621.314

МИХАЙЛЕНКО В. В.,
ВИНОГРАДОВ Ю. М.,
БАБЕНКО А. А.

КОМП'ЮТЕРНИЙ АНАЛІЗ НАПІВПРОВІДНИКОВИХ ПЕРЕТВОРЮВАЧІВ ЕЛЕКТРОЕНЕРГІЇ З БАГАТОРОЗГАЛУЖЕНИМИ СТРУКТУРАМИ

У статті проведено аналіз електромагнітних процесів в електричних колах з напівпровідниковими комутаторами. Створено математичну модель для аналізу електромагнітних процесів у напівпровідникових перетворювачах з широтно-імпульсним регулюванням вихідної напруги. Наведено графіки, що відображають електромагнітні процеси у електричних колах.

Analysis of the electromagnetic processes is organized in article in electric circuit with semiconductor commutator. Mathematical model is created for analysis of the electromagnetic processes in semiconductor converter with width pulsed regulation of the output voltage. It is shown graphs that show the electromagnetic processes in electric circuit.

Вступ

Одним з найбільш потужних інструментів для розрахунків, аналізу та оптимізації напівпровідникових перетворювачів (НПП) електроенергії і пристроїв силової та інформаційної електроніки є сучасні пакети програмного забезпечення персональних комп'ютерів (ПК). Практичне використання такого програмного забезпечення особливо актуальне і виправдане зараз, коли в умовах розширеної інформаційності потужність та пакети програм сучасних ПК стали такими, що їхнім користувачам тільки потрібно зосередити свої зусилля на вирішенні науково-технічних задач, притаманних лише певним профілям їхньої спеціалізації. Це стало можливим сьогодні завдяки існуванню цілої низки інтегрованих систем проектування, автоматизації розрахунків тощо. До них відносяться: MatLab, MathCAD, EWB 8, OrCAD 9.2, MC 6, CAD-системи, SPICE, XSPICE, MultiSim 2014 тощо, з допомогою яких розроблено цілу низку моделей елементної бази силової та інформаційної електроніки, окремих вузлів, блоків і систем НПП електроенергії та технологічних установок на їхній основі [1 – 14]. Неперервне удосконалювання математичного забезпечення програм такого класу дозволяє проектувальникам НПП адекватно і швидко отримувати і обробляти результати моделювання.

Проте використання традиційного підходу до моделювання всього лише вузлів НПП за схемою електричною принциповою, кожен силовий чи будь-який елемент якої подається комп'ютерною моделлю, що враховує найменшу сталу часу при його функціонуванні, призводить до збільшення розмірності та жорсткості диференціальних рівнянь математичного забезпечення повної моделі, внаслідок чого зростає час моделювання, а то й зривається його стійкість. Особливо цей аспект проявляється у разі моделювання електромагнітних процесів у НПП електроенергії з багаторозгалуженими структурами силових частин та систем керування.

Одним з напрямків вирішення проблеми комп'ютерного аналізу НПП з багаторозгалуженими структурами став перехід від властивостей та фізичних процесів у окремих елементах на більш високі ієрархічні рівні, – функціональний та структурний при умові їхньої сумісності з моделями окремих елементів [1, 2, 4, 5]. Іншим, не менш перспективним напрямком вирішення згаданої проблеми є використання системних методів моделювання електромагнітних процесів у НПП [6 – 9] з застосуванням математичного процесора MathCAD.

Метою даної роботи є порівняльний аналіз деяких аспектів названих вище напрямків комп'ютерного аналізу НПП для визначення доцільності їхнього використання на різних етапах дослідження електромагнітних процесів.

Аналіз електромагнітних процесів

Перший з напрямків до початку безпосереднього моделювання потребує досить трудомісткої попередньої роботи, яка, в залежності від складності конкретного схемотехнічного рішення НПП, інколи може зайняти більшу частину часу, відведеного на процес наукових досліджень. При наявності бібліотек функціональних блоків така робота передбачає, крім їхнього

тестування у необхідних ситуаційних режимах, подання схем НПП на функціональному та структурному рівнях, їхній опис з представленням множин необхідних завдань у вигляді сталих величин чи програм, а також установку та тестування, зокрема в пакеті MatLab, індикаційного обладнання. В подальшому проявляються власні достоїнства створених моделей у можливості оперативної зміни завдань за їхньою кількістю та числовими значеннями.

Другий напрямок базується на методі багатопараметричних модулюючих функцій (БМФ) [6, 7], який забезпечує можливість аналізу електромагнітних процесів шляхом їхнього моделювання за формалізовано складеними алгоритмічними рівняннями роботи НПП, як систем перетворення електроенергії. У такому разі алгоритмічні рівняння подаються функціями багатопараметричного виду, аргументами яких є структурні параметри систем, що віддзеркалюють їхній склад за множинами пісистем та елементів, котрі утворюються в результаті декомпозиції системи в межах цілого. По суті, складене таким чином рівняння описує за наперед заданим законом модуляції зовнішній вплив НПП на іншу систему, якою є навантаження разом зі згладжувальним фільтром. Повний цикл моделювання проводиться від визначення алгоритмічного рівняння, яке дає можливість оперативно отримати кінцевий результат, що подає поведінку вихідних координат системи, як цілого (причому як у сталих, так і в перехідних режимах роботи без зміни виду алгоритмічного рівняння), до аналізу процесів у найпростіших елементах підсистем.

Системний підхід дозволяє проведення досліджень з певним числом ітерацій, кожна наступна з яких має нові, більш жорсткі відносно попередньої, припущення. Це дає можливість оперативно оцінювати взаємозалежні електромагнітні процеси з урахуванням як внутрішніх міжпідсистемних, так і зовнішніх міжсистемних впливів з одночасним підвищенням достовірності результатів моделювання. Власним достоїнством системних досліджень у середовищі MathCAD є те, що вони потребують значно менших за трудомісткістю підготовчих робіт. Підготовка до моделювання передбачає окреслення меж систем, їхню декомпозиція за підсистемними і елементними складовими та визначення алгоритмічних рівнянь роботи НПП, як систем, з урахуванням зовнішніх та внутрішніх впливів на них. Процес моделювання проводиться поетапно з представленням множин необхідних завдань до відповідних алгоритмічних рівнянь у вигляді сталих чи змінних величин, що дозволяє, з урахуванням показань часових діаграм, які реалізуються наявним у середовищі MathCAD потужним графічним редактором, оперативне коригування та виключення помилок у процесі досліджень.

Для порівняння згаданих вище напрямків аналізу НПП з багаторозгалуженими структурами розглянемо приклад моделювання електромагнітних процесів у НПП трифазної напруги у постійну з можливістю двозонного регулювання вихідної напруги, побудованого за схемою, що містить ланку високої частоти. До складу НПП входять: випрямляч (В) за схемою Ларіонова, з'єднаний вхідними шинами з трифазним джерелом напруги, а виходом підключений до силових входів високочастотних інверторів випрямленої напруги (ІВН). Узгоджувальні трансформатори ІВН з'єднаними послідовно вторинними обмотками створюють контур для підключення діагоналі постійного струму високочастотного випрямляча (ВВ), вихід якого утворює вихід НПП. За базовий приймемо один з найбільш популярний сьогодні програмний пакет MatLab, зокрема одну з його версій (MATLAB 6.5, release13), яка постачається з пакетом візуального моделювання Simulink 5.0.

На рис.1 представлена модель згаданого НПП, в якій блок Universal Bridge подає випрямляч за схемою Ларіонова. Крім цього, при побудові моделі використано наступні функціональні блоки: трифазне джерело напруги, з можливістю завдання його внутрішніх параметрів за величинами еквівалентних опорів та індуктивностей впродовж фазних проводів; ІВН1, ІВН2 – інвертори випрямленої напруги першої та другої зон регулювання вихідної напруги; Т1, Т2 – узгоджувальні трансформатори ІВН1 та ІВН2 відповідно; ВВ – високочастотний випрямляч; СК – схема керування; $i_d(t)$ (Current Measurement), $u_d(t)$ (Voltage asurement) – вимірювальні прилади,

що фіксують струм навантаження та напругу на ньому відповідно; $u_1(1,t)$, $u_1(2,t)$, $u_1(3,t)$ та $i_A(t)$, $i_B(t)$, $i_C(t)$ – блоки вимірювання напруги та струмів вхідної трифазного джерела напруги; Scope1, Scope2 – блоки візуального спостереження електромагнітних процесів.

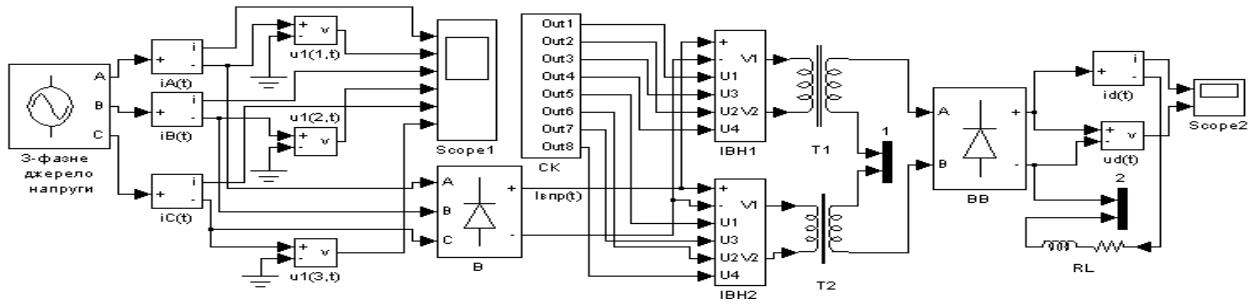


Рис.1. Структурна схема перетворювача

Кожен з ІВН (рис.2) складається з 4-х транзисторів IGBT та 4-х зворотних діодів D, увімкнених у відповідних стійках, з'єднаних за мостовими схемами. Крім того до складу ІВН входять енергообміні конденсатори С.

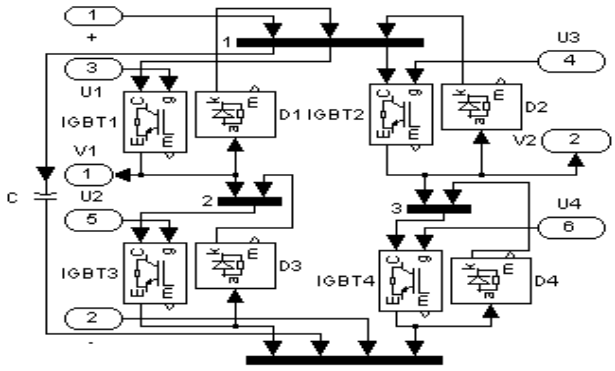


Рис. 2. Схема ІВН

СК (рис.3) містить: ГПН1, …, ГПН3 та Г1, Г2 – генератори пилкоподібних напруг; та напруг з лінійними законами зміни відповідно; Sum1 та Sum2 – аналогові суматори, які дозволяють алгебраїчне підсумовування будь-якої кількості вхідних сигналів; sin та sign1, …, sign4 – блоки формування тригонометричних та знакових функцій від завдань на входах; Product1, …, Product4 – аналогові помножувачі вхідних напруг.

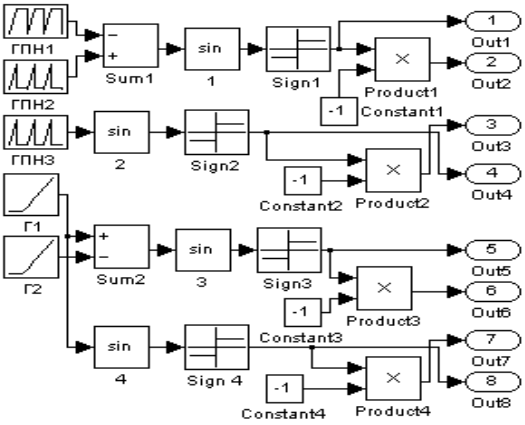


Рис. 3. Схема СК

Моделювання електромагнітних процесів проведено для випадку симетричної трифазної напруги живлення та активно-індуктивного навантаження.

З метою порівняння наведеної моделі з моделлю, отриманою за методом БМФ, процес моделювання проведемо у два етапи. На першому з них приймемо вхідну мережу такою, що має нульові значеннями внутрішніх параметрів за величинами еквівалентних опорів та індуктивностей, а силові транзистори та діоди НПП подамо ідеальними ключами. Другий етап проведемо за умови установки нових завдань, які дозволяють ураховувати вплив еквівалентних параметрів мережі живлення та внутрішніх опорів ключових елементів на протікання електромагнітних процесів в НПП. Крім того, на вхідних силових шинах В відносно нульового проводу мережі установимо додаткові конденсатори для компенсації реакції еквівалентних індуктивностей на дію перервних струмів вхідних контурів НПП.

Результати першого етапу моделювання процесів відносно вихідної напруги $u_d(t)$ НПП у процесі його "м'якого" пуску, струму $i_d(t)$ навантаження та струмів $i_A(t)$, $i_B(t)$, $i_C(t)$ у вхідних контурах трифазного джерела напруги в координатах його фазних напруг $u_1(1,t)$, $u_1(2,t)$, $u_1(3,t)$ подані діаграмами на рис. 4.

Якщо силову частину розглянутого НПП подати у вигляді системи перетворення електроенергії, то за методом БМФ, подібно дослідженням, проведеним в [8], алгоритмічне рівняння роботи НПП запишемо у вигляді

$$u_d(t) = \frac{\sum_{p=1}^{N} u'_d(t)\psi\left(\alpha_p(t)\right)}{k_{\text{T}}} \qquad (1)$$

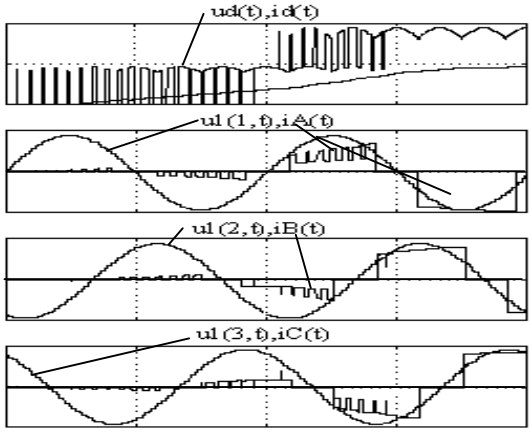


Рис. 4. Діаграми напруги та струму навантаження, вхідних струмів та напруг на першому етапі моделювання

де $u_d(t)$ – миттєві значення вихідної напруги НПП, що містить N ІВН ($N = 2$); p = 1, 2, 3, …, $N = L$ – номер зони регулювання вихідної напруги; $u'_d(t)$ – миттєві значення вихідної напруги В; $\psi(\alpha_p(t))$ – еквівалентний модулюючий вплив (ЕМВ), миттєві значення якого залежать від величини кутів управління $\alpha_p(t)$, за рахунок зміни яких забезпечується широтно-імпульсне регулювання вихідної напруги НПП. $k_{\text{Т}}$ – коефіцієнт трансформації узгоджувального трансформатора.

Миттєві значення вихідної напруги $u'_d(t)$ в (1) подамо виразом виду

$$u'_d(t) = \frac{1}{2}\sum_{i=1}^{3} E_m \sin\left[\omega_1 t + \frac{\pi\left(1-4\left(i-1\right)\right)}{6}\right], \quad (2)$$

де E_m, $\omega_1 = 2\pi f_1$ та $i = 1,\ 2,\ 3$ – відповідно амплітудне значення, кругова частота та номер фазної напруги мережі живлення.

З (2) випливає, що миттєві значення фазної напруги визначатимуться, як

$$u_1(i,t) = E_m \sin\left[\omega_1 t + \pi\left[1-4\left(i-1\right)\right]/6\right]. \quad (3)$$

ЕМВ $\psi(\alpha_p(t))$ запишемо співвідношенням виду

$$\psi\left(\alpha_p(t)\right) = \sum_{2} \text{sign}\left[\sin\left(\omega_2 t \pm \alpha_p(t) - \varphi\right)\right]/2, \quad (4)$$

де $\omega_2 = 2\pi f_2$ та φ – відповідно кругова частота та початкова фаза ЕМВ.

Струм навантаження знайдемо з урахуванням (1) з рішення диференціального рівняння виду

$$D(t,y) = \frac{u_d(t) - Ry_0}{L}, \quad (5)$$

де y_0 – визначається з початкових умов; R і L – відповідно активний опір і індуктивність навантаження.

Рішення (5) відносно струму навантаження визначимо у вигляді матриці

$$i_d(t) = \text{rkfixed}\left(y, 0, k, s, D\right), \quad (6)$$

де: y – вектор початкових умов; 0, k – часовий інтервал рішень; s – кількість точок на часовому інтервалі рішень; D – символьний вектор струму навантаження.

З урахуванням анодних струмів анодної та катодної груп вентилів В, за відомою з [6] методикою струми його вхідних контурів подамо, як

$$i(i,t) = \left[f(n,t) - f(m,t)\right] \times \\ \times \frac{1}{k_{\text{Т}}} \sum_{p=1}^{N} i_d(t) \left|\psi\left(\alpha_p(t)\right)\right|, \quad (7)$$

де $f(n,t)$ та $f(m,t)$ – комутаційні функції, що збігаються за часом з інтервалами провідних станів діодів В; n = 1, 2, 3 та m = 4, 5, 6 – відповідно номери діодів анодної та катодної груп; причому, згідно алгоритму роботи В, для відповідних фаз мережі живлення обов'язковими є умови: якщо i = 1, то n = 1, а m = 4, якщо i = 2, то n = 2, а m = 5, якщо i = 3, то n = 3, а m = 6.

Вирази (1) – (7) подають математичну модель електромагнітних процесів у НПП з багатозонним регулюванням вихідної напруги і за рівних умов за завданнями відображаються діаграмами, які повністю повторюють діаграми, наведені на рис. 4. Проте найважливішим у цьому разі є те, що для реалізації математичної моделі за методом БМФ необхідно значно менших, порівняно з попередньою моделлю, трудових затрат, оскільки вона не потребує схемотехнічного супроводження і попереднього тестування.

На рис. 5 наведено фрагменти другого зі згаданих вище етапів моделювання відносно вихідної напруги $u_d(t)$ НПП при фіксованому значенні кута керування, струму $i_d(t)$ навантаження та струмів $i_{1A}(t)$, $i_{1B}(t)$, $i_{1C}(t)$ у фазних проводах мережі в координатах напруг $u_{C0}(i,t)$ на енергообмінних конденсаторах

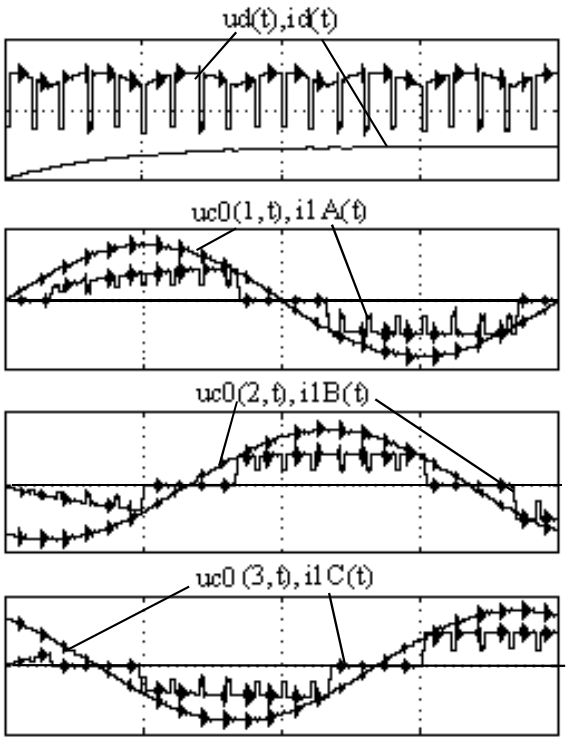


Рис. 5. Діаграми напруги та струму навантаження, вхідних струмів та напруг на другому етапі моделювання

У цьому разі потрібно відмітити позитивний чинник, який показує на те, що, для отримання наведених на рис. 5 діаграм, схемотехнічне супроводження моделі, розроблене на першому етапі моделювання, майже не змінюється

У середовищі MathCAD подібні дослідження проводяться при комплексному використанні методів БМФ, визначення еквівалентних опорів сталих структур НПП [7] та накладання в методиці моделювання електромагнітних процесів у системі "мережа живлення – НПП електроенергії" [9]. У такому разі, як і у даному прикладі, розглядаються НПП, що формують вихідну енергію в режимі джерела напруги, з припущеннями, що струми їхніх вхідних контурів залежать від характеру струмів навантаження і визначені в результаті попереднього моделювання за методом БМФ з урахуванням внутрішніх опорів ключових елементів [7]. В подальшому за методом накладання визначаються струми $i_1(i,t)$ у фазних проводах мережі та напруги $u_{C0}(i,t)$ на конденсаторах, з використанням схеми заміщення, наведеної на рис. 6 [9].

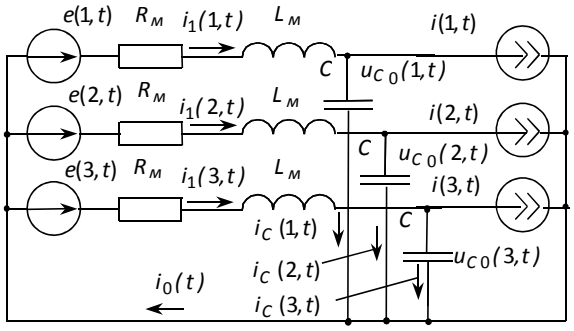


Рис. 6. Схема заміщення

Висновки

Порівняльний аналіз результатів моделювання у двох згаданих вище середовищах, проведений на другому етапі для НПП, який має вихідну потужність 15 кВТ, показав розбіжність, яка не перевищує 10%. Основною причиною цього є те, що у процесі моделювання за методом БМФ не враховані втрати, визнані комутацією струмів діодами випрямлячів та силових ключів ІВН.

Таким чином, можемо зауважити, що розглянуті напрямки комп'ютерного аналізу НПП не суперечать один одному у процесі досліджень, а, навпаки, можуть вдало доповнювати один одного на різних етапах автоматизованого проектування НПП електроенергії. Так, розглянуті аспекти комп'ютерного аналізу НПП показують, що для оперативного вибору технічних рішень і попередньої оцінки їхніх функціональних можливостей у процесі проектуванні НПП електроенергії з багаторозгалуженими структурами перевагу має другий з розглянутих напрямків, що передбачає системні методи досліджень, які дозволяють проведення моделювання електромагнітних процесів без схемотехнічного супроводження, а, отже, з меншими трудовими затратами. При більш детальних дослідженнях НПП перевагу має перший напрямок, як такий що дозволяє уточнювати результати моделювання без зміни початкової моделі.

Список посилань

1. Жемеров Г. Г. Компьютерное моделирование преобразовательной системы с компенсационным управляемым выпрямителем / Г. Г. Жемеров, Д. С. Крылов // Технічна електродинаміка. Проблеми сучасної електротехніки. Ч. 4. – Тематичний випуск. –2002. – С. 59 – 62.

2. Халиков В. А. Исследование энергоэффективности источников сварочной дуги современными программными пакетами моделирования / В. А. Халиков, К. А. Липковский, В. М. Паханьян, А. Ф. Шатан // Технічна електродинаміка. -№ 3. –2004.- –С. 17-22.
3. Перевєрзєв А. В. Моделювання пристроїв силової електроніки / А. В. Перевєрзєв, А. К. Тімовський, О. В. Василенко – Запоріжжя: ЗДІА, 1988. – 98 с.
4. Перевєрзев А. В. Макромоделирование силовых элементов преобразователей / А. В. Перевєрзев, С. Л. Шмалий // Технічна електродинаміка. Проблеми сучасної електротехныки. – Ч.3. – Тематичний випуск. – 2004. – С. 87-90.
5. Василенко О. В. Макромоделі на основі В-елементів для автоматизованого схемотехнічного проектування пристроїв силової електроніки: Автореф. дис. канд. техн.наук: 05.09.12 / Національний технічний університет України "Київський політехнічний інститут". - К., 2002. - 20 с.
6. Макаренко М. П. Моделювання мережних перетворювачів електроенергії модуляційного типу / М. П. Макаренко, В.І . Сенько, М. М. Юрченко – К.: ІЕД НАН України, 2002. - 140 с.
7. Макаренко М. П. Метод системного макромоделювання процесів у напівпровідникових перетворювачах модуляційного типу / М.П. Макаренко // Праці ІЕД НАНУ, 2003. - № 1 (4): Зб. наук. пр. – 2003. - с. 54-61.
8. Макаренко М. П. Математична модель перетворювача трифазної напруги в постійну напругу / М. П. Макаренко, В. В. Михайленко // Электроника и связь. – 2002. – № 14. – С. 73–75.
9. Макаренко М. П. Моделювання електромагнітних процесів у системі "мережа живлення – напівпровідниковий перетворювач електроенергії" / М. П. Макаренко, В. І. Сенько, М. М. Юрченко // Вісник Кременчуцького державного політехнічного університету. – Випуск 3/2004 (26). – С. 147-149.
10. Макаренко М. П. Системний аналіз електромагнітних процесів у напівпровідникових перетворювачах електроенергії модуляційного типу / М. П Макаренко, В. І. Сенько, М. М. Юрченко – К. : НАН України, ІЕД, 2005. – 241 с.
11. Макаренко М. П. Аналіз електромагнітних процесів у перетворювачах з багатозонним регулюванням вихідної напруги функціями багатопараметричного виду / М. П. Макаренко, В. В. Михайленко // Техн. електродинаміка. Тем. вип. "Силова електроніка та енергоефективність". – 2002. – Ч. 1. – С. 19–22.
12. Макаренко Н. П. Анализ электромагнитных процессов в двенадцатипульсном преобразователе с зонным регулированием выходного напряжения / Н. П. Макаренко, В. В. Михайленко, Н. Н. Юрченко // Вестник НТУ "Харьковский политехнический институт". "Проблемы автоматизированного электропривода. Теория и практика". – 2002. – Т. 1. – С. 233–234.
13. Макаренко М. П. Деякі аспекти комп'ютерного аналізу напівпровідникових перетворювачів електроенергії з багаторозгалуженими структурами / М. П. Макаренко, В. В. Михайленко // Техн. електродинаміка. Тем. вип. "Силова електроніка та енергоефективність". – 2004. – Ч. 1. – С. 112–115.
14. Макаренко М. П. Алгоритмічно-структурні аспекти аналізу імпульсних перетворювачів змінної напруги / М. П. Макаренко, В. В. Михайленко, В. В. Пілінський, В. І. Сенько, М.М. Юрченко // Техн. електродинаміка. Тем. вип. "Силова електроніка та енергоефективність". – 2007. – Ч. 4. – С. 66–69.

УДК 681.518

РОЛИК А.И.

КОМПЛЕКСНЫЙ ПОДХОД К УПРАВЛЕНИЮ УРОВНЕМ ИТ-УСЛУГ В КОРПОРАТИВНЫХ ИТ-ИНФРАСТРУКТУРАХ

Предложен комплексный подход к управлению уровнем услуг в корпоративных ИТ-инфраструктурах. Подход основан на интеграции процессного и оперативного управления качеством ИТ-услуг.

The complex method of service level management was offered. The method was based on the integration of process management and operational management of quality IT-services.

Введение

Конкурентоспособность компаний, особенно тех, которые работают на рынке предоставления информационно-коммуникационных услуг, существенно зависит от эффективности использования информационно-коммуникационных ресурсов. Если компании для организации и предоставления ИТ-услуг не используют облачные технологии, то они вынуждены делать существенные финансовые вложения на приобретение и нести постоянные затраты на обслуживание ИТ-инфраструктуры (ИТИ). Когда ресурсы ИТ-инфраструктуры используются неэффективно, то себестоимость предоставления ИТ-услуг компанией клиентам или абонентам становится выше, чем у конкурентов, и это не позволит ей долго занимать свою нишу на рынке. Поэтому компании заинтересованы так организовать управление предоставлением ИТ-услуг, чтобы высокое качество достигалось с наименьшими затратами, т. е. с минимальным количеством задействуемых ресурсов и с минимальными затратами на обслуживание ИТ-инфраструктуры. Для этих целей компании закупают или разрабатывают различные системы управления ИТ-инфраструктурой (СУИ), из которых наибольший интерес представляют СУИ, использующие соответствующие ИТ с функционалом управления уровнем ИТ-услуг и эффективного использования ресурсов. Для создания ИТ управления ИТ-инфраструктурой (ИТ УИТИ), применяемых в СУИ, необходимо разработать концепцию, модели и методы управления. Поэтому данная работа, в которой предлагается комплексный подход к управлению уровнем ИТ-услуг, является актуальной.

Анализ проблем создания системы управления ИТ-инфраструктурой

При управлении уровнем ИТ-услуг следует стремиться не к достижению максимально возможного уровня услуг, а к непрерывному поддержанию качества услуг на согласованном с бизнес-подразделениями или клиентами уровне. Причем для повышения конкурентоспособности компании следует делать это с минимальными затратами, т. е. минимальным количеством задействуемых для предоставления ИТ-услуг ресурсов ИТ-инфраструктуры [1].

Управление ИТ-инфраструктурой с целью поддержания согласованного с бизнесом уровня ИТ-услуг является сравнительно новым разделом управления, но оно опирается на множество классических направлений организационно-технического управления, а используемые механизмы, модели и методы управления ИТ-инфраструктурой соответствуют механизмам, моделям и методам, применяемым при решении широкого круга задач управления в технических и организационных системах — от управления технологическими процессами до принятия решений при управлении предприятиями.

СУИ являются новыми для теории управления техническими системами и характеризуются такими свойствами, как многообъектность, рассредоточенность объектов и распределенность средств управления, большое количество параметров и переменных, наличие факторов неопределенности и риска. Существенная стоимость ИТ-инфраструктуры и ее важность для бизнеса, с одной стороны, низкая эффективность ИТ и медленный возврат вложений в ИТ-инфраструктуру — с другой, определяют необходимость разработки теоретических и методических основ автоматизированного управления ИТ-инфраструктурой. Характер задач, решаемых СУИ, существенно отличается от задач, решаемых традиционными системами автоматического управления и регулирования. Необходимость согласования действий между подсистемами СУИ, цели функционирования которых могут существенно различаться, большое

количество рисков и неопределенностей, возникающих при выборе управляющего воздействия, существенный диапазон и динамика изменения внешних воздействий, а также множество режимов функционирования СУИ привели к появлению новых функциональных задач управления: рационального распределения и оперативного перераспределения вычислительных и коммуникационных ресурсов, координации действий подсистем, оперативного управления уровнем услуг и функционирования ИТ-инфраструктуры. Несмотря на практические успехи в области автоматизированного управления многообъектными распределенными системами управления, к которым относится СУИ, работы по созданию теоретических основ управления такими системами далеки от завершения [2]. В настоящее время отсутствует теория управления ИТ-инфраструктурами, и ввиду большого количества задач, которые необходимо решить для автоматизации и эффективного управления ИТ-инфраструктурой, создание такой теории потребует много сил и времени. Поэтому сейчас при управлении ИТ-инфраструктурой превалирует подход, основанный на постоянном наращивании функционала управления в ИТ УИТИ.

Наиболее масштабные исследования в области управления ИТ-инфраструктурой проводятся международными организациями (TM Forum, Office of Government Commerce, IEEE, ISO и др.), аналитическими агентствами (Forrester Research, Gartner и пр.) и корпорациями (IBM, Cisco, Microsoft, Hewlett-Packard и др.). Однако эти исследования не охватывают всех проблем, связанных с разработкой ИТ УИТИ, поскольку в основном сосредоточены на процессном управлении ИТ. Кроме того, недостаточно детально проработаны такие аспекты управления ИТ-инфраструктурой, как оперативное управление качеством ИТ-услуг, рациональное использование ресурсов ИТ-инфраструктуры, анализ функционирования элементов ИТ-инфраструктуры и др. Это объясняется тем, что первоначально ИТ УИТИ создавались для управления сетевым оборудованием. Большая часть фирменных СУИ предназначается для работы с оборудованием собственного производства, а модернизация и применение этих систем для управления оборудованием других производителей сильно затруднены или невозможны. Фирменные системы управления сетевым оборудованием основаны на ноу-хау, носят закрытый характер, часто используют собственные языки программирования и содержат примитивные средства сопряжения с другими системами управления. Такая же картина имеет место и в области управления качеством услуг, предоставляемых ИТ-инфраструктурой, когда используется большое количество информационно-коммуникационных технологий и оборудования разных производителей. Каждая технология может иметь собственную систему управления, работающую независимо от других систем управления. Разрозненные системы управления решают частные задачи без учета задач, решаемых другими системами управления, значимости бизнес-процессов, приоритетов приложений и режимов работы ИТ-инфраструктуры. Создание ИТ УИТИ сопряжено с необходимостью решения множества разнообразных взаимосвязанных задач эффективного управления ИТ-инфраструктурой в условиях динамики доступности и производительности ресурсов.

В корпоративных ИТ-инфраструктурах создаются и развиваются масштабные решения вспомогательных ИТ-систем, обеспечивающих эффективную работу ИТ-инфраструктуры, ИТ-подразделения и осуществляющих наряду с множеством функций управления и такие функции, как хранение данных, защита информации и пр. При традиционной организации ИТ-подразделения создается несколько ИТ-служб, отвечающих за работоспособность отдельных приложений, ИТ-сервисов или ресурсов ИТ-инфраструктуры. При этом отсутствие единой СУИ вызывает необходимость использования большого количества систем управления, знания множества ИТ, принципов управления разными ресурсами, информационно-коммуникационными сервисами и пр., что приводит к увеличению количества администраторов. Отсутствие единого центра управления, что характерно для больших корпораций, приводит к несогласованным действиям администраторов, нарушающим функционирование ИТ-инфраструктуры, неэффективной работе приложений и нерациональному использованию ресурсов.

Учитывая, что корпоративные ИТ-инфраструктуры отличаются размерами, характером решаемых задач, определяемых бизнесом, создание единой ИТ УИТИ с обширным функционалом, полностью решающей пробле-

мы управления ИТ-инфраструктурой, представляет собой чрезвычайно сложную, трудоемкую, дорогостоящую, но решаемую задачу. При этом усилия разработчиков должны быть сосредоточены на разработке ИТ УИТИ, обеспечивающей поддержание качества ИТ-услуг на заданном, согласованном с бизнес-подразделением, уровне путем разработки и внедрения моделей, методов и инструментария управления ИТ-инфраструктурой с осознанием необходимости рационального использования ее ресурсов.

Базовая модель управления

С точки зрения управления ИТ-инфраструктурой можно выделить три контура управления: внешний, внутренний и оперативный. Эти контуры приведены на рис. 1, где изображена базовая модель управления ИТ-инфраструктурой.

Рис. 1. Базовая модель управления ИТ-инфраструктурой

Внешний контур осуществляет проектирование и внедрение бизнес-процессов с последующим контролем эффективности их выполнения. При этом информационно-управляющая система (ИУС) является основой для эффективного ведения бизнеса. Внутренний контур обеспечивает отображение бизнес-процессов на ИТ-услуги с определением целевых значений, фиксируемых в SLA, и контролем качества ИТ-услуг. Декомпозиция ИТ-сервисов определяет задачи оперативного контура, которые сводятся к непрерывному поддержанию уровня услуг на согласованном с персоналом управления бизнесом уровне с минимальными затратами.

Бизнес-процессы (управляющие, операционные и поддерживающие) или производственные процессы при принятии решений используют критерии эффективности бизнеса $\mathcal{P}$.

ИУС — это система, в которой объединяются персонал управления бизнесом (основной и вспомогательной деятельностью) и ИТ-инфраструктура. Посредством ИУС принимаются решения, обеспечивающие выполнение критерия бизнеса $\mathcal{P}$.

В то же время информационно-телекоммуникационная система (ИТС) — совокупность ИТ-инфраструктуры, персонала поддержки ИТ-инфраструктуры (ИТ-персонала), а также технических, программных и информационных средств такой поддержки в виде СУИ — обеспечивает предоставление бизнес-подразделениям ИТ-услуг с согласованным уровнем качества при рациональном использовании ресурсов ИТ-инфраструктуры.

На вход ИТ-инфраструктуры поступает поток запросов $\tilde{\mathcal{R}}$, на который ИТ-инфраструктура реагирует потоком результатов $\tilde{\mathcal{A}}$. На ИТ-инфраструктуру воздействуют возмущающие воздействия $\tilde{\Xi}$. В СУИ анализируется текущее состояние $S \in \mathcal{S}$ ИТ-инфраструктуры и выбирается управление $U \in \mathcal{U}$, при котором достигается максимальная эффективность управления.

В соответствии с рис. 1. ИТ-инфраструктура является общей составляющей взаимодействующих компонентов: ИУС и ИТС. Соответственно, можно выделить два масштабных объекта управления: бизнес-процессы (производственные процессы) и ИТ-инфраструктуру, и две системы управления: ИУС и СУИ, в контурах управления которых занят персонал управления бизнесом и ИТ-инфраструктурой соответственно.

Для эффективного ведения бизнеса должна быть сформирована и скоординирована целостная система процессов всех трех контуров управления с выделением в контурах следую-

щих процессов. Во внешнем: прогнозирование → планирование развития → учет → оценивание по критерию доходность/качество → контроль → корректирование планов развития. Во внутреннем, как правило, используется цикл Деминга: планирование → выполнение → проверка → действия по выполнению плана. В оперативном: мониторинг → анализ состояния → управление → системная оптимизация → планирование развития [3]. Своевременность и точность выполнения интегрированного комплекса процессов трех контуров управления обеспечивает достижение целей бизнеса.

Для поддержки замкнутых контуров управления в ИТ УИТИ аспект оперативного управления ИТ-инфраструктурой интегрируется с аспектом скоординированного взаимодействия компонентов системы управления компании [4]. Координация содействует достижению цели функционирования ИТ УИТИ как составляющей системы управления компании. При этом должно обеспечиваться координированное взаимодействие и обмен информацией между контурами управления с соответствующим обобщением информации в каждом из них для принятия решений. В этом случае существенно усиливается роль ИТ УИТИ, в которой произошло смещение акцентов управления ИТ-инфраструктурой от поддержки информационно-коммуникационных технологий и оборудования в сторону поддержки уровня ИТ-услуг.

Таким образом, управление ИТ-инфраструктурой нельзя рассматривать в отрыве от управления бизнесом, а при повышении эффективности функционирования ИТС с ориентацией на поддержание согласованного уровня ИТ-услуг главная роль отводится ИТ УИТИ.

В оперативном контуре появляются задачи измерения, анализа, оценивания, учета, контроля, прогнозирования, планирования и управления, что порождает необходимость разработки соответствующих моделей и методов решения этих задач. Прежде, чем разрабатывать эти модели и методы, следует определиться с общим подходом к решению задач оперативного контура с учетом его комплексного характера и необходимости взаимодействия и координации с внешним и внутренним контурами управления.

Управление ИТ-подразделением, в котором сосредоточен ИТ-персонал, подразумевает прежде всего организационные формы управления, которые хорошо проработаны в ITSM [5] по ISO/IEC 20000, при этом ИТ УИТИ предоставляет средства для автоматизации управления ИТ-подразделением.

Механизмы управления СУИ закладываются на этапе проектирования и в основном включают в себя изменение конфигурации и самодиагностику.

Управление ИТ-инфраструктурой осуществляет ИТ-подразделение с использованием методов автоматического, автоматизированного и ручного управления.

Эволюция управления ИТ-инфраструктурой

Анализ многочисленных публикаций, существующих СУИ, стандартов, практик и протоколов управления позволяет выделить три уровня зрелости управления ИТ-инфраструктурой (рис. 2): управление сетями и вычислительной техникой, управление информационными системами и управление уровнем ИТ-услуг.

Эволюция развития средств обработки информации [6] позволяет понять, что необходимость управления сетями и вычислительной техникой была осознана давно. Это привело к тому, что исторически первыми появились системы управления на I и II-м уровнях ИТ-инфраструктуры. Вопросы сетевого управления и управления вычислительной техникой очень хорошо проработаны вплоть до создания систем автоматического управления некоторыми сетями и серверами. Несмотря на это, недостаточно проработаны вопросы управления ограниченными ресурсами ИТ-инфраструктуры с учетом важности бизнес-процессов, а также эффективного управления ресурсами центров обработки данных (ЦОД).

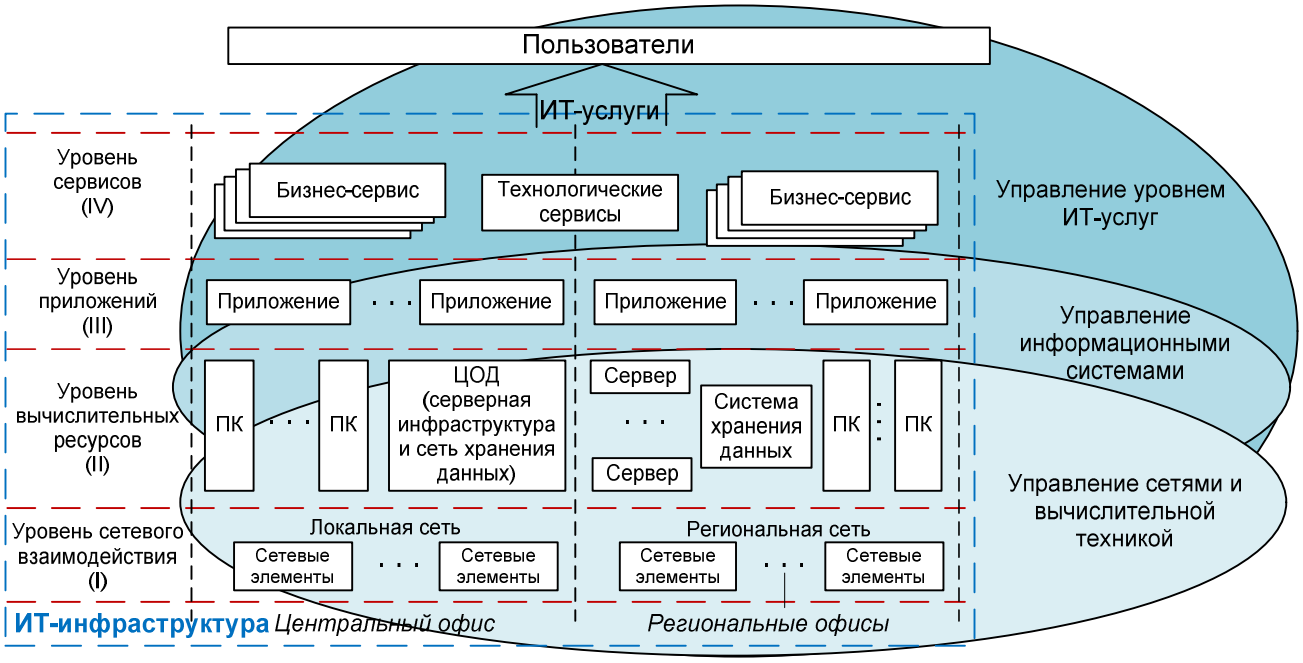


Рис. 2. Этапы эволюции управления ИТ-инфраструктурой

Следующим этапом эволюции управления ИТ-инфраструктурой стало управление информационными системами, которое в основном сводилось к управлению распределенными приложениями. На этом этапе разработчики ИТ УИТИ столкнулись с отсутствием принятого ИТ-сообществом стандарта управления приложениями.

Особенность управления на этих этапах эволюции заключается в поддержании параметров функционирования элементов и приложений на определенном соответствующим регламентом уровне. Считалось, что если поддерживать на заданном уровне значения показателей качества функционирования ИТ-инфраструктуры, то качество предоставляемых ИТ-услуг будет соответствовать целевым значениям.

Третий этап эволюции связан с осознанием необходимости и возможности управления уровнем ИТ-услуг, что произошло в последние годы. Поэтому вопросы управления уровнем ИТ-услуг, в силу новизны решаемых задач, исследованы и проработаны недостаточно хорошо.

Четко просматривается и очередной — четвертый этап эволюции управления ИТ-инфраструктурой, которым должно стать управление ИТ-инфраструктурой персоналом управления бизнесом. Это станет возможным, когда степень автоматизации управления ИТ-инфраструктурой достигнет такого уровня, что осуществлять управление ею окажется не сложнее, чем управлять персональным компьютером. Это ознаменует появление новой бизнес-среды с интегрированной в нее ИТ-инфраструктурой, управление которой не будет требовать наличия специального ИТ-персонала, включенного в контур оперативного управления. Движение в этом направлении подтверждается появлением публикаций, утверждающих, что при наличии средств автоматизации управление ЦОД будет осуществлять так же просто, как сейчас квалифицированный пользователь управляет собственным компьютером[7]. При этом высококвалифицированные специалисты будут привлекаться только время от времени для разрешения сложных ситуаций, не описанных в базе знаний СУИ.

Выделение функциональных областей управления корпоративной ИТ-инфраструктурой

ИТ УИТИ должна решать большое количество разнообразных задач и выполнять множество сильно различающихся функций. Все задачи и функции ИТ УИТИ можно отнести к одной из трех обобщенных функциональных областей управления ИТ-инфраструктурой, полностью покрывающих весь функционал управления (рис. 3):

1) поддержание качества ИТ-сервисов на согласованном уровне;

2) рациональное использование ресурсов ИТ-инфраструктуры;

3) автоматизация оперативной деятельности ИТ-подразделения.

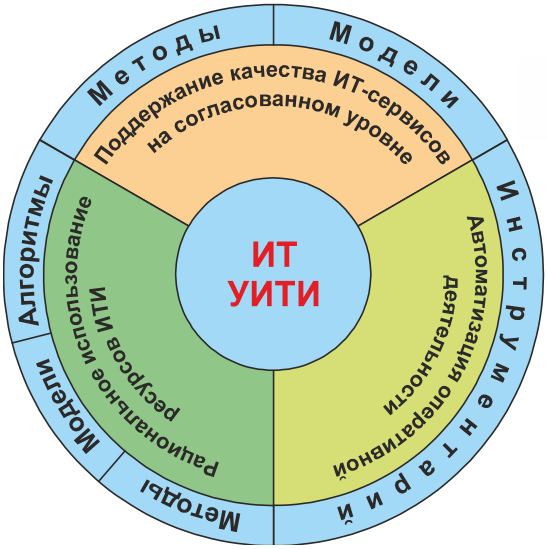


Рис. 3. Функциональные области управления ИТ-инфраструктурой

При поддержании качества ИТ-сервисов на согласованном уровне для повышения удовлетворенности пользователей решаются следующие задачи: поддержание целевых значений параметров функционирования приложений и поддержание согласованного уровня услуг.

Функции и задачи рационального использования ресурсов ИТ-инфраструктуры: обеспечение эффективного использования ИТ-ресурсов при работоспособном состоянии ИТ-инфраструктуры и осуществление адаптации к возмущающим воздействиям, вызывающим перегрузки вследствие нестационарности процессов в ИУС или в результате проявления неисправностей; повышение надежности и доступности критически важных ИТ-ресурсов; распределение и перераспределение ресурсов ИТ-инфраструктуры.

При автоматизации оперативной деятельности ИТ-подразделения решаются следующие задачи и выполняются такие функции: повышение эффективности работы предприятия за счет нацеленности ИУС на потребности бизнеса и улучшения взаимодействия между бизнес-структурами и ИТ-подразделением; обеспечение функционирования ИТ-инфраструктуры в соответствии с заданным регламентом; повышение производительности и эффективности управления ИТ-инфраструктурой; повышение надежности функционирования ИТ-инфраструктуры; улучшение функциональности и оперативности управления ИТ-инфраструктурой; повышение производительности труда администраторов ИТ-инфраструктуры; улучшение взаимодействия между администраторами.

Обеспечение стабильно высокого качества ИТ-услуг попадает в сферу ответственности как процессного управления в соответствии с ITSM, так и оперативного управления с помощью ИТ УИТИ (рис. 4). Поэтому управление качеством услуг обеспечивается процессами ITSM и оперативным управлением ИТ-инфраструктурой посредством СУИ.

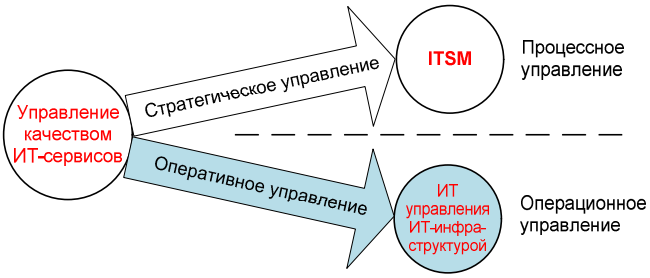


Рис. 4. Распределение сфер ответственности управления ИТ-сервисами между процессным и оперативным управлением

ITSM и ИТ УИТИ имеют различные объекты управления, общим из которых являются ИТ-сервисы (рис. 5).

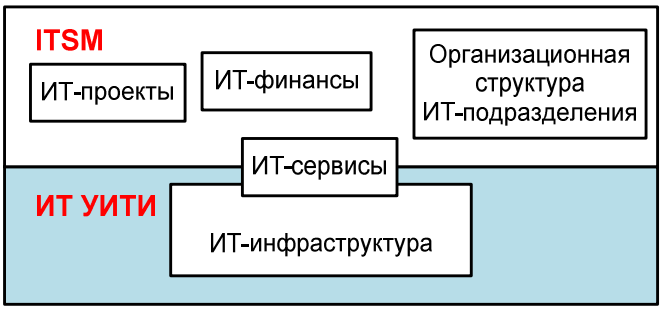


Рис. 5. Объекты управления ITSM и ИТ УИТИ

ITSM и ИТ УИТИ преследуют различные цели управления. ITSM нацеливается на постоянное повышение качества ИТ-услуг, ИТ УИТИ — на текущее поддержание показателей качества ИТ-услуг на согласованном уровне. ITSM не рассматривает ИТ-инфраструктуру как объект управления для обеспечения высокого уровня ИТ-услуг.

В таблице 1 приведены результаты оценки наличия ПО управления ИТ-инфраструктурой, имеющего отношение к управлению уровнем услуг.

Таблица 1 — Наличие ПО управления ИТ-инфраструктурой, имеющего отношение к управлению качеством услуг

Основные функции и задачи по управлению качеством сервисов	Наличие ПО автоматического/автоматизированного управления	
	Процессное управление	Оперативное управление
Организация работы ИТ-подразделения	+	Не рассматривается
Централизованное управление ИТИ	-	-
Интегрированное управление всеми уровнями ИТИ	Не рассматривается	-
Управление предоставлением услуг	+	Не рассматривается
Управление качеством ИТ-услуг	+	-
Организация сквозного процесса управления ИТИ от сбора информации от объектов управления с трансформацией в качество услуг	-	-
Анализ влияния сбоев в ИТИ на качество ИТ-услуг	+/-	-
Прогноз функционирования ИТИ и качества ИТ-услуг	-	-
Мониторинг качества предоставляемых сервисов	+	+/-
Сведение метрик	-	-
Поддержание качества услуг на заданном уровне	Не рассматривается	-
Управление приложениями	Не рассматривается	+/-
Взаимодействие между приложениями	Не рассматривается	+/-
Интегрированное управление приложениями, телекоммуникационной сетью с учетом важности бизнес-процессов	Не рассматривается	-
Управление устранением неисправностей	+	-
Распространение влияния неисправностей	-	-
Управление доступом к ограниченными ресурсам	Не рассматривается	-
Распределение ресурсов ЦОД	-	-
Управление производительностью приложений	+	-
Управление потоками приложений в сети	Не рассматривается	-
Мониторинг использования приложениями ресурсов	Не рассматривается	+
Управление серверами	Не рассматривается	+
Управление сетями	Не рассматривается	+
Интегрированное управление серверами и сетями	Не рассматривается	Только мониторинг
Управление доступом пользователей к приложениям	Не рассматривается	+
Генерация отчетов о функционировании ИТИ	+	+

Вопросы процессного управления уровнем услуг достаточно хорошо проработаны в литературе и даже стандартизованы. В то же время стандарты, практики и ПО управления уровнем услуг в соответствии с ITSM сосредотачиваются на вопросах организационного управления ИТ-подразделением и не рассматривают оперативное управление ИТ-инфраструктурой, не предлагают модели, методы и алгоритмы рационального распределения ресурсов ИТ-инфраструктуры. Ведущие поставщики систем управления не раскрывают внутренних механизмов работы своего ПО, а ITIL и решения ITSM содержат только формализованное описание организации работы ИТ-подразделения и управление ИТ-услугами.

Вопросам обеспечения должного качества уровня услуг посредством оперативного управления ИТ-инфраструктурой только в последнее время начинают уделять должное внимание. Кроме того, практически не рассматривается интегрированное взаимодействие процессного и оперативного управления при поддержании высокого качества ИТ-услуг.

В настоящее время отсутствуют международные стандарты, специфицирующие комплексное управление ИТ-инфраструктурой на всех иерархических уровнях. Задачи управления на нижних уровнях ИТ-инфраструктуры проработаны стандартами лишь по отдельности для сетей, серверов, хранилищ данных, отдельных информационно-коммуникационных технологий и пр. и не рассматривают интегральный подход к управлению ИТ-инфраструктурой. Не известны интегрированные СУИ, позволяющие централизованно управлять всеми компонентами ИТ-инфраструктуры и представляющие инструментарий ИТ-подразделению для эффективного автоматизированного управления всеми ресурсами и приложениями. Поэтому в работе, учитывая важность для бизнеса получения ИТ-услуг со стабильно высоким качеством, предлагается комплексный процессно-оперативный

подход к управлению уровнем ИТ-услуг. При этом рассмотрение ИТ-инфраструктуры как объекта управления для поддержания приемлемого уровня качества предоставления ИТ-услуг не только возможно, но и необходимо.

Общие процессы и иерархические уровни управления

В [6] при управлении ИТ-инфраструктурой предлагается выделять пять универсальных процессов: мониторинга, анализа, управления, оптимизации и планирования (МАУОП) (рис. 6). Некоторые из процессов для ряда базовых функций управления ИТ-инфраструктурой могут иметь нулевую функциональность. Отобрав для каждого из процессов МАУОП набор методов и реализовав его в соответствующем универсальном инструментарии ИТ УИТИ, можно автоматизировать выполнение процессов в различных категориях управления, используя одни и те же методы и инструменты.

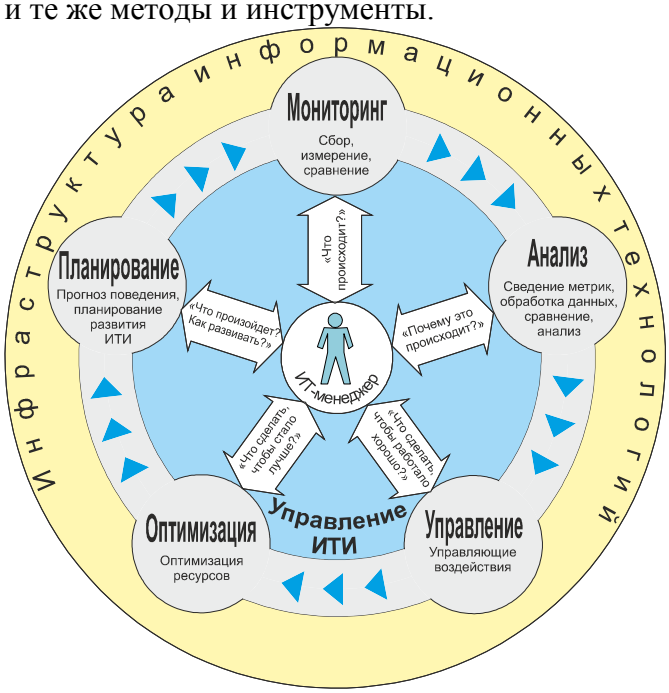


Рис. 6. Последовательность выполнения процессов МАУОП

Тогда единый инструментарий мониторинга для всех категорий, например агент, может сразу следить за функционированием информационно-коммуникационных технологий, элементами ИТ-инфраструктуры, работой приложений, сервисов и пр., находящихся в зоне ответственности агента, и осуществлять управление. Набор методов анализа, реализованный в универсальном инструментарии, позволит выявлять тенденции поведения компонентов ИТ-инфраструктуры, упреждать неисправности, реализовать проактивное управление. Инструментарий планирования позволит, например, при планировании расширения предприятия, определить масштабы расширения сети, объемы наращиваемых ресурсов, количество дополнительно привлекаемого ИТ-персонала и пр. Процессы МАУОП применяются во всех трех функциональных областях управления ИТ-инфраструктурой (рис. 7).

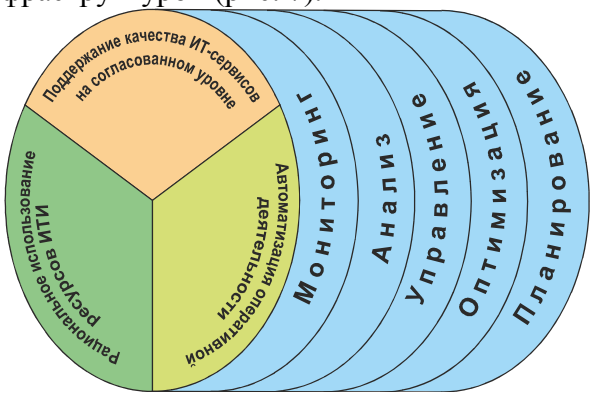


Рис. 7. Место процессов МАУОП в функциональных областях управления ИТ-инфраструктурой

Поскольку ИТ-инфраструктура имеет иерархическую структуру (см. рис. 2) с выделением четырех иерархических уровней сервисов, приложений, вычислительных ресурсов и сетевого взаимодействия [6], средства управления ИТ-инфраструктурой целесообразно также строить по иерархическому принципу. Для этого необходимо выделить иерархические уровни управления, рационально распределить функции управления, осуществить дифференциацию команд и методов управления. Уровней управления в ИТ УИТИ может быть больше или меньше, чем иерархических уровней ИТ-инфраструктуры, кроме того, уровни управления могут не совпадать с уровнями ИТ-инфраструктуры.

Компания Cisco использует модель в виде трех уровней управления — сервисами, сетью и сетевыми элементами [8]. Концепция TMN специфицирует управление: сетевыми элементами; сетью; услугами; бизнес-процессами операторов телекоммуникационных сервисов [9]. В [10] выделяется три слоя управления ИТ-инфраструктурой: сетями, системами и сервисами. Парадигма ПО управления ИТ — IT Management Software 2.0 (ITMS2) [11] предполагает создание единого инструментария для управления элементами уровней вычислительных ресурсов и сетевого взаимодействия. Это означает, что команды управления сетевым оборудованием и серверами будут относиться к

одной категории, несмотря на то, что эти компоненты относятся к разным уровням иерархии ИТ-инфраструктурой. Подобным образом, ввиду родственности решаемых задач, следует объединить управление сервисами и приложениями. Современный взгляд на комбинирование функций управления подытожен в [3], где предложена перспективная концепция управления ИТ-инфраструктурой, а стратификация функций, методов и команд управления ИТ-инфраструктурой предусматривает выделение трех иерархических уровней: управление ИТ-инфраструктурой, управление приложениями, управление ресурсами.

Соотношение между уровнями управления, уровнями иерархии ИТ-инфраструктуры и управляемыми компонентами ИТС представлено на рис. 8.

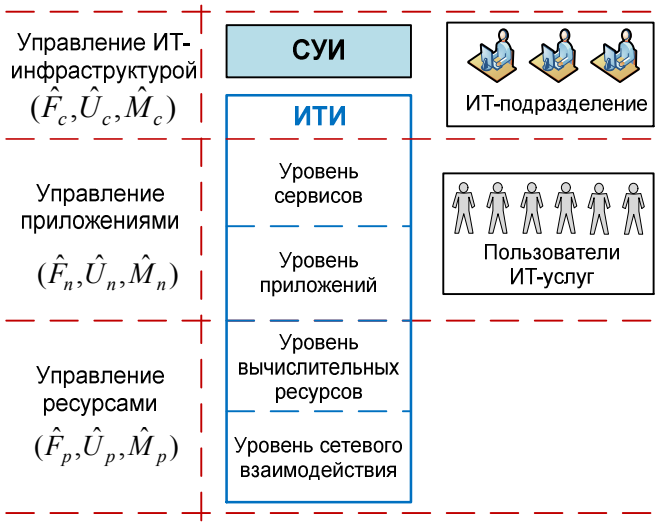


Рис. 8. Соотношение между уровнями управления ИТ-инфраструктурой

Функции множества $\hat{F}_c \in \hat{\mathcal{F}}$ и команды управления множества $\hat{U}_c$ уровня управления ИТ-инфраструктурой обеспечивают поддержание функционирования ИТ-инфраструктуры в соответствии с заданным регламентом, управление функциональными подсистемами ИУС, бизнес-процессами ИТ-подразделения и организационное управление ИТ-инфраструктурой, координацию, т. е. согласованное управление работой нижних уровней управления для достижений общей цели функционирования ИУС. Этот уровень содержит лицо, принимающее решение. На этом уровне общие цели и задачи, стоящие перед ИУС, преобразуются в задачи и целевые значения для нижних уровней, перераспределяются ресурсы ИТ-инфраструктуры и принимаются решения в нештатных ситуациях.

Множество $\hat{U}_n$ составляют команды управления: приложениями и взаимодействием между распределенными приложениями; услугами и поддержанием согласованного уровня предоставления услуг; отдельными типами телекоммуникационных сетей (ТКС) и пользователями. Здесь используется программное управление в соответствии с целевыми значениями, задаваемыми уровнем управления ИТ-инфраструктурой.

Команды из множества $\hat{U}_p$ управляют: вычислительными ресурсами и сетевым взаимодействием; компонентами приложений; доступом пользователей к информационным и телекоммуникационных ресурсам; службами, серверами, ПК, оборудованием ТКС, вспомогательным оборудованием, ОС и др. Управление сводится к программному изменению параметров объектов мониторинга и управления ИТ-инфраструктуры в соответствии со значениями, задаваемыми на вышерасположенных уровнях управления.

Реализация команд высшего уровня, как правило, осуществляется путем формирования наборов команд нижних уровней

$$\hat{U}_c \to \hat{U}_n \to \hat{U}_p .$$

Например, функции управления, определенные для ТКС стандартом X.700 и включающие управление: устранением неисправностей, учетом, конфигурациями, производительностью и безопасностью, реализовываются командами управления всех трех уровней.

На рис. 9 показано взаимоотношение между функциональными областями управления, уровнями и процессами управления.

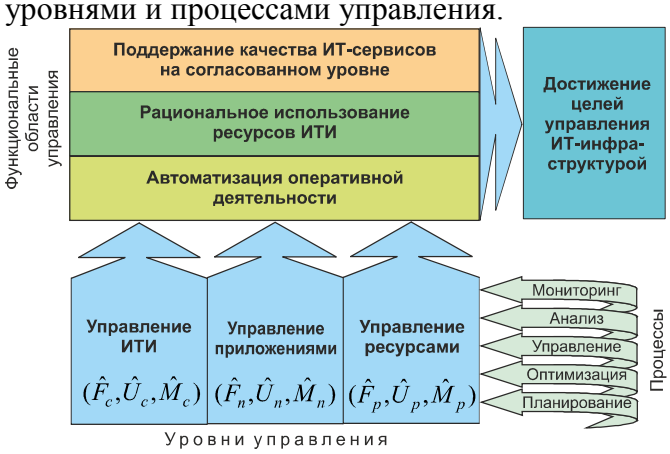


Рис. 9. Взаимосвязь между функциональными областями, уровнями и процессами управления для достижения цели создания ИТ УИТИ

Определение задач инструментария управления ИТ

ИТ-инфраструктура включает в себя разнородные составляющие — информационные активы, приложения, средства взаимодействия и техническую инфраструктуру. Управление каждой из них осуществляется с учетом индивидуальных особенностей: информация — рост объемов данных; приложения — потребность в интеграции; средства взаимодействия — необходимость увеличения полосы пропускания и поддержки новых сервисов; техническая инфраструктура — возрастающая сложность.

Сложность оценивания качества функционирования ИТ-инфраструктуры обусловлена тем, что она может содержать до n•10000 единиц оборудования, каждая единица оборудования может состоять из n•100 компонентов, ИТ-инфраструктура предоставляет до n•100 сервисов, каждый из которых может быть композицией n•10 услуг, которыми пользуются n•1000 пользователей, для удовлетворения потребностей которых выделяется до n•1000 ресурсов. Причем все перечисленные компоненты обычно характеризуются большим набором значений параметров (до n•1000 параметров).

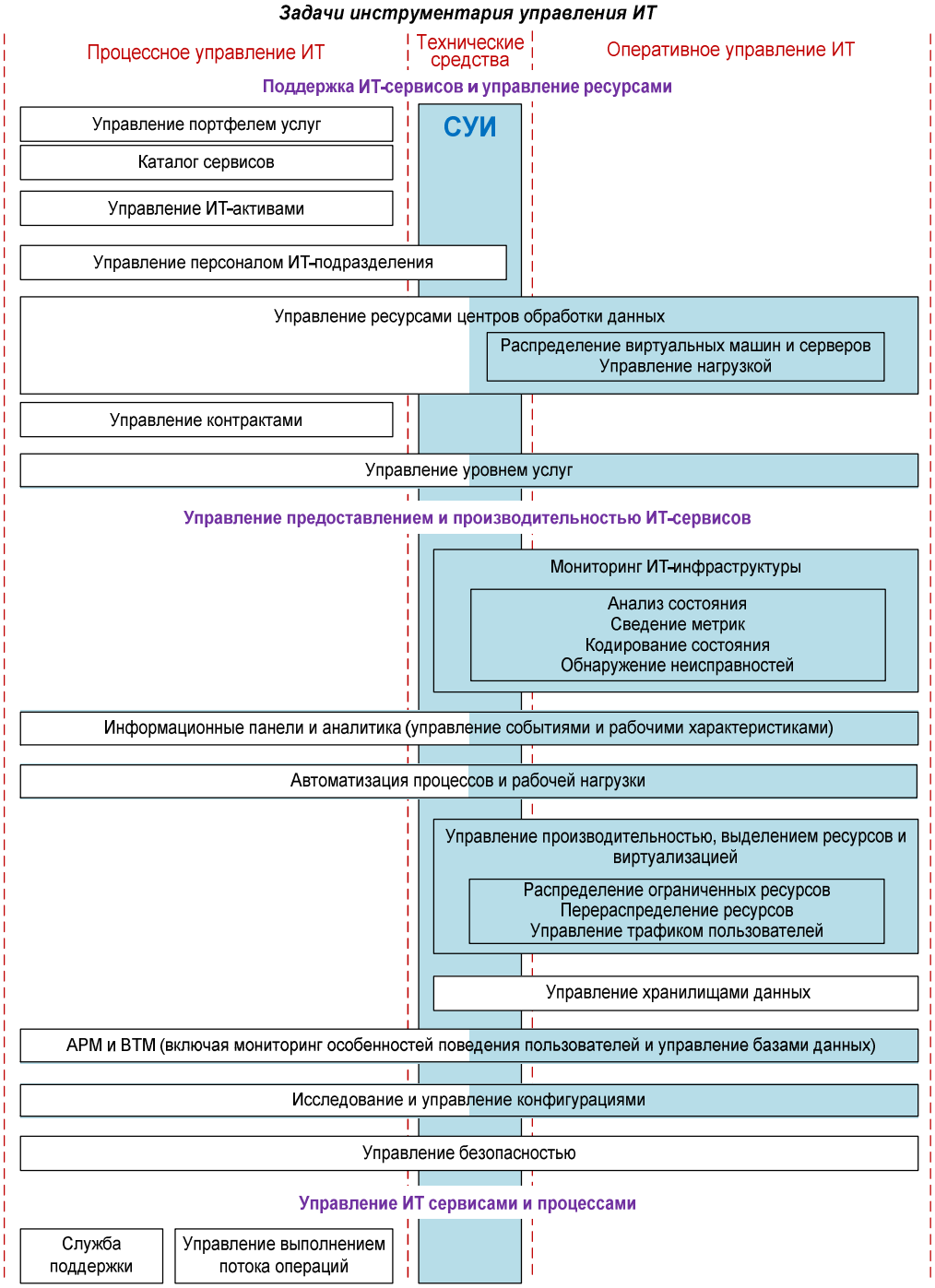


Рис. 10. Задачи инструментария управления ИТ-инфраструктурой

Управление ИТ-инфраструктурой нельзя рассматривать в отрыве от управления ИТ и управления качеством ИТ-услуг, поскольку основная задача ИТ-инфраструктуры заключается в предоставлении пользователям ИТ-услуг с согласованным уровнем качества. На рис. 10 приведены задачи инструментария управления ИТ в соответствии с ITMS2 для процессного и оперативного управления, а фоном выделены первоочередные задачи, требующие решения при автоматизации оперативного управления.

Из обобщения материала, изложенного в [1, 3, 6] и в данной работе, можно сделать вывод о важности и актуальности научно-прикладной проблемы создания проблемно-ориентирован-ной ИТ УИТИ, нацеленной на поддержание согласованного уровня ИТ-услуг при использовании минимальных объемов ресурсов ИТ-инфраструктуры в условиях виртуализации, кластеризации и консолидации вычислительных ресурсов при существенной динамике запросов пользователей. Для создания такой ИТ УИТИ необходимо решить следующие задачи (рис. 11):

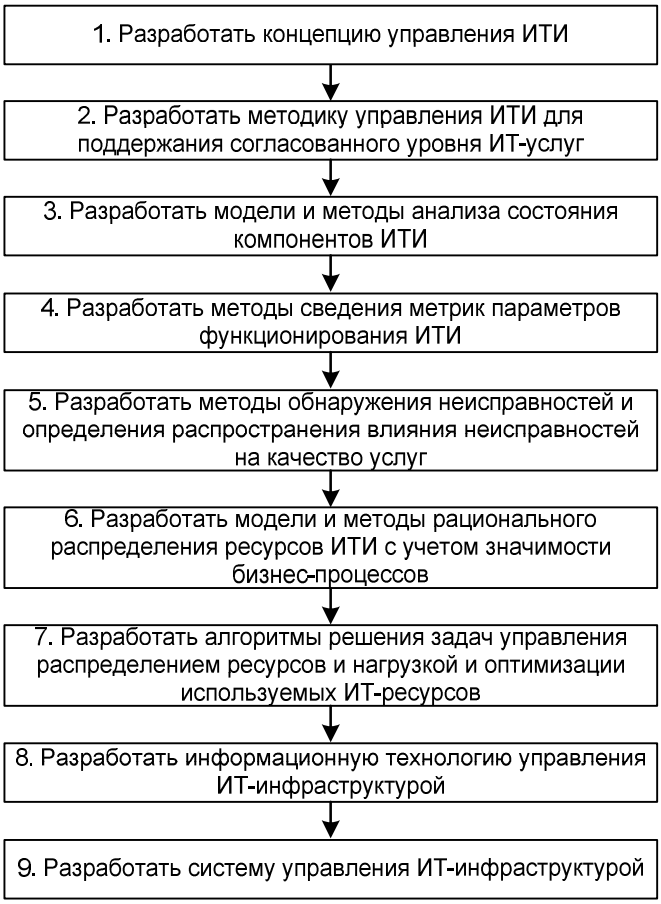


Рис. 11. Задачи, решаемые при создании ИТ УИТИ

Суть задач сводится к следующему:

1. Важнейшей задачей, которую нужно решить при создании ИТ УИТИ, является разработка концепции управления такой сложной системой, как корпоративная ИТ-инфраструктура. Учитывая то, что эффективность бизнес-процессов непосредственно зависит от эффективности управления ИТ-услугами и функционирования ИТ-инфраструктуры, разработка концепции управления ИТ-инфраструктурой является первоочередной задачей. В настоящее время известна только одна ориентированная на предоставление услуг концепция управления ИТ-инфраструктурой [3] и работа в этом направлении должна быть продолжена.

2. Современный этап развития ИТ характеризуется повышенными требованиями бизнеса к качеству ИТ-услуг. Поэтому созданию методик оперативного управления ИТ-инфраструктурой, нацеленных на поддержание согласованного уровня ИТ-услуг наименее ресурсоемким способом, должно быть уделено особое внимание.

3. Решение задач управления, нацеленных на обеспечение функционирования объектов мониторинга и управления ИТ-инфраструктурой в соответствии с заданным регламентом или связанных с поддержанием производительности ИТ-инфраструктуры на согласованном уровне, требует знания состояния, производительности и других характеристик ее структурных компонентов. Поэтому для оценки качества функционирования ИТ-инфраструктуры необходимо выбрать или разработать методы и средства определения и анализа состояния объектов мониторинга и управления.

4. Наличие большого количества элементов, сложность связей и многоуровневость ИТ-инфраструктуры приводит к необходимости учета анализа n•1000 различных параметров и метрик для измерения и оценки состояния элементов ИТ-инфраструктуры и качества предоставляемых услуг. Метрики верхнего уровня часто имеют качественный характер, характеризуются неопределенностью, нечеткостью и субъективностью. Для формирования сводных метрик уровня услуг необходимо иметь или разработать соответствующие модели и методы.

5. При ориентации управления ИТ-инфраструктурой на поддержание согласованного уровня услуг наиболее дешевым способом становится важной задача обнаружения неис-

правностей и оценка их влияния на качество предоставляемых ИТ-услуг. Решение этой комплексной задачи позволит сократить потери, связанные с неисправностями, за счет сокращения времени на локализацию и устранение неисправностей.

7. Переход от рассредоточенной обработки данных к централизованной обработке в ЦОД требует решения ряда задач, и прежде всего разработки методов и алгоритмов эффективного использования дорогостоящих вычислительных ресурсов ЦОД, что особенно актуально в условиях повсеместной виртуализации [12] и облачных вычислений [13].

8. Решения предшествующих задач сформируют теоретико-методологические основы создания ИТ УИТИ, обеспечивающей оперативное управление во всех трех функциональных областях с упором на управление уровнем услуг и рациональное использование ИТ-ресурсов. Однако разработка собственно ИТ УИТИ представляет собой сложную в программно-техническом и программно-технологическом аспектах задачу.

9. Разработка эффективной ИТ УИТИ составит основу для создания, внедрения и эксплуатации СУИ. Однако здесь также возникает сложная задача поддержки проектных и реализационных работ, интеграции с другими системами и развития.

Процессно-оперативный подход к управлению уровнем ИТ-услуг

Управление уровнем услуг осуществляется как процессным управлением ITSM, так и оперативным — посредством СУИ (см. рис. 4 и 5). В настоящее время в ИТ-подразделениях эти виды управления рассматриваются и существуют независимо друг от друга. Интерес представляет комплексный — процессно-оперативный подход, предполагающий тесную взаимосвязь процессного и оперативного управления уровнем услуг. В этом случае связующим звеном для интеграции управления является СУИ (рис. 12), которая, кроме автоматизации управления ИТ-инфраструктурой и оперативного управления по поддержанию согласованного уровня услуг, выполняет ряд функций процессного управления ITSM наряду с обеспечением взаимодействия администраторов — управления уровнем обслуживания (Service-level management — SLM) и ИТ-инфраструктуры для управления уровнем услуг и решения проблем в ИТ-инфраструктуре. На рис. 12 управление уровнем ИТ-услуг осуществляется с использованием сервисно-ресурсных моделей [14], отображающих взаимосвязи ИТ-услуг с ИТ-ресурсами, которые их обеспечивают.

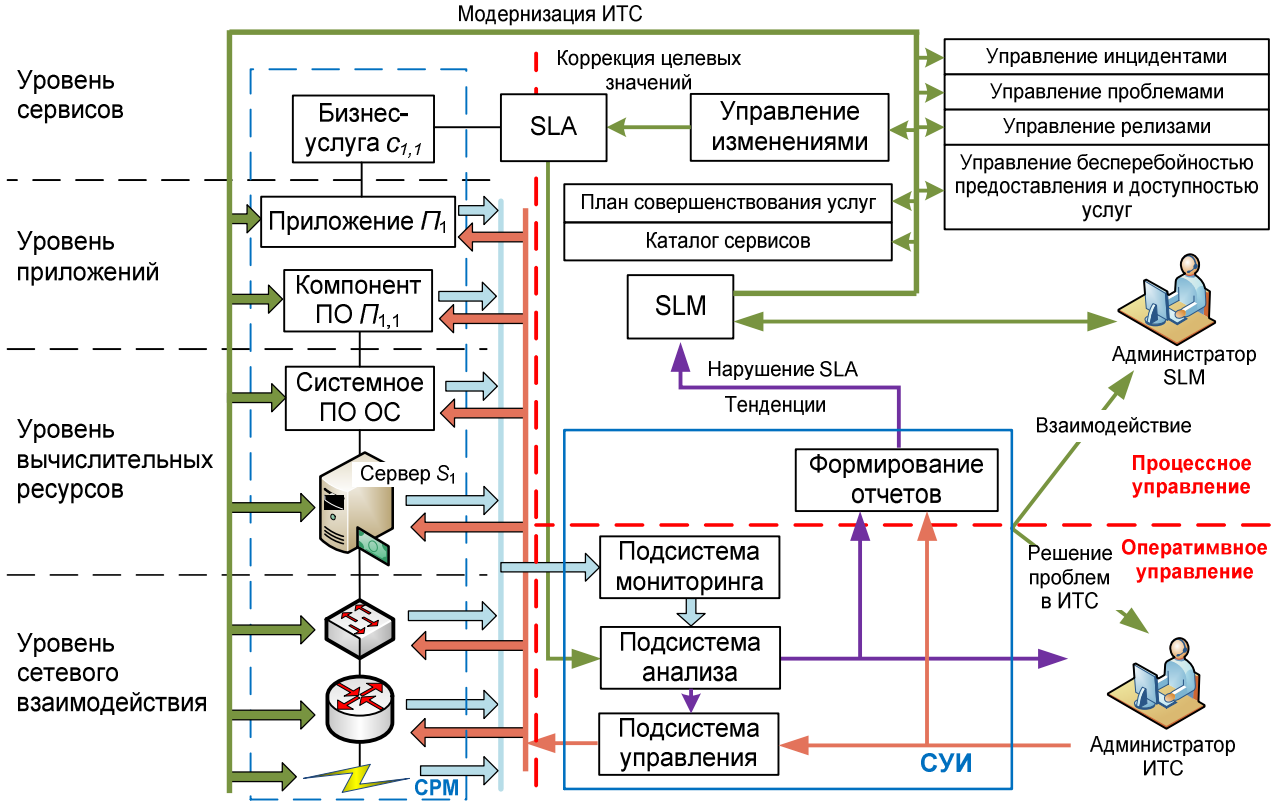


Рис. 12. Схема реализации комплексного подхода к управлению уровнем услуг

Оперативное управление нацелено на поддержание наиболее дешевым способом качества услуг на согласованном уровне, при этом управление должно быть таким, чтобы обеспечивалось

$$q_{b,ki} - q^*_{b,ki} \to 0, \forall b,k,i,$$

где $q_{b,ki}$ и $q^*_{b,ki}$ — соответственно, целевое и фактическое значения k-го показателя качества j-й услуги, b-го сервиса.

Процессное управление по результатам аналитического контроля и с учетом финансовых затрат в рамках процесса SLM производит коррекцию целевых значений $q_{b,ki}$, $\forall b,k,i$ показателей качества услуг и стимулирует решение проблем в ИТ-инфраструктуре, влияющих на снижение качества. Целью SLM является повышение уровня услуг. Для достижения этой цели выполняется цикл, включающий в себя этапы определения, согласования, утверждения, контроля и оценки уровня предоставляемых ИТ-услуг, а также активации действий по повышению качества услуг. Для этого определяются точные и измеримые показатели $q_{b,ki}$, $\forall b,k,i$ качества всех услуг. После чего фактический уровень услуг $q^*_{b,ki}$, $\forall b,k,i$, определяемый СУИ, контролируется, а нарушения SLA фиксируются в отчетах, наряду со сведениями о тенденциях изменения уровня услуг и причинах несоответствия. Все действия по улучшению услуг вносятся в план совершенствования услуг [15]. SLM должен содействовать реализации превентивного подхода к управлению уровнем услуг. Качество услуг зависит от того, как организована деятельность по их управлению. При процессном управлении используется простая модель управления качеством в виде цикла Деминга [16].

Поставщик и заказчик услуг регулярно корректируют уровень услуг $q_{b,ki}$, $\forall b,k,i$ путем запуска процесса SLM. Среди параметров, характеризующих качество ИТ-услуг, имеется показатель удовлетворенности пользователей (QoE), который рассматривается как субъективная составляющая качества услуги, а целевые показатели уровня услуг — как объективные параметры. Объективные параметры, значения которых зависят от состояния и функционирования ИТ-инфраструктуры, являются предметом контроля со стороны СУИ при оперативном управлении ИТ-инфраструктурой. Для этого СУИ осуществляет непрерывный контроль выполнения целевых показателей уровня услуги. Кроме этого производится анализ рисков возникновения сбоев, планирование роста/сокращения численности пользователей или объема подлежащих хранению данных, компенсация пиков загруженности ИТ-инфраструктуры и всех изменений условий предоставления услуг. В этих случаях оперативное управление реализуется перераспределением ресурсов с учетом приоритетов бизнеса, критичности услуги и прогноза будущих требований к доступности с обеспечением приемлемого уровня услуг.

Комплексное управление уровнем услуг позволяет снизить число нарушений SLA и степень влияния ухудшения качества предоставления ИТ-услуг на успешность ведения бизнеса, обеспечивая поддержание согласованного уровня услуг, достаточного для удовлетворения потребностей бизнеса.

Выводы

Основные требования к ИТ со стороны бизнеса заключаются в необходимости предоставления скоординированных услуг с высоким уровнем, повышения экономической отдачи от ИТ и интеграции процессов управления ИТ.

Выделено три функциональные области управления, покрывающие весь функционал управления ИТ-инфраструктурой: поддержание качества ИТ-сервисов на согласованном уровне; рациональное использование ресурсов ИТ-инфраструктуры; автоматизация оперативной деятельности ИТ-подразделения. Показано, что управление ИТ-инфраструктурой, направленное на поддержание согласованного уровня ИТ-сервисов, является критически важным для обеспечения эффективного выполнения бизнес-процессов.

Предложен комплексный подход к управлению уровнем услуг на основе сервисно-ресурсных моделей, отличающийся интеграцией процессного и оперативного управления. Подход позволяет обеспечивать не только оперативное поддержание согласованного уровня услуг, но и стратегическое обоснование согласованного уровня услуг в зависимости от имеющихся ресурсов ИТ-инфраструктуры.

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. Методы робастного, нейро-нечеткого и адаптивного управления: Учебник / Под ред. Н.Д. Егупова. – М.: Изд-во МГТУ им. Н.Э. Баумана, 2001. – 744 с.
3. Ролик А.И. Концепция управления корпоративной ИТ-инфраструктурой / А.И. Ролик // Вісник НТУУ «КПІ»: Інформатика, управління та обчислювальна техніка. – К.: «ВЕК+», 2012. – № 56. – С. 31–55.
4. Теленик С.Ф. Системи управління хмарними ІТ-структурами / С.Ф. Теленик, О.І. Ролік, М.В. Ясочка, О.С. Квітко // Інтелектуальні системи прийняття рішень і проблеми обчислюваного інтелекту: Матеріали між нар. наук. конф. (16–20 травня) 2011 р. м. Євпаторія. – Том 1. – Херсон: ХНТУ, 2011. – С. 124–127.
5. IT Service Management: An Introduction// J.V. Bon, G. Kemmerling, D. Pondman, Publisher: Van Haren Publishing. – 2002. – 217 p.
6. Ролик А.И. Тенденции и перспективы развития управления информационными технологиями / А.И. Ролик // Вісник НТУУ «КПІ»: Інформатика, управління та обчислювальна техніка. – К.: «ВЕК+», 2012. – № 55. – С. 81–109.
7. Barroso L.A. The Datacenter as a Computer: An Introduction to the Design of Warehouse-Scale Machines Second Edition / L.A. Barroso, J. Clidaras, U. Hölzle// Morgan & Claypool Publishers. – 2013. – 156 p.
8. Cisco Advanced Services Network Management Systems Architectural Leading Practice. White Paper. – Cisco Systems, Inc. – 2007. – 11 p.
9. Principles for a telecommunications management network: ITU-T Rec. M.3010. – Geneva. – 2000. – 36 p.
10. Катышев С. Об одной концепции управления распределенным ресурсами // С. Катышев / Открытые системы. – 1998. – №3.
11. Garbani J.-P. It's Time For IT Management Software 2.0 / J.-P. Garbani, T. Mendel, E. Hubbert, E. Radcliffe. – Forrester Research, Inc. – 2009. – Dec. 8. – 13 p.
12. Marshall D. Advanced Server Virtualization: VMware and Microsoft Platforms in the Virtual Data Center / D. Marshall, W. A. Reynolds, D. McCrory. – Auerbach Publications, 2006. – 742 p.
13. Garbani J.-P. IT Infrastructure And Operations: The Next Five Years / J.P. Garbani, M. Cecere. – Forrester Research, Inc. – 2011. – May 3. – 20 p.
14. Cannon D. ITIL Service Strategy. 2011 Edition / D. Cannon.– UK: TSO, Norwich.– 2011.– 496 c.
15. Information technology. Service management. Part 1: Specification: ISO/IEC 20000-1:2005. – ISO/IEC, 2005. – 16 p.
16. Будкова Л. Методическое руководство для подготовки к профессиональным экзаменам ISO 20000 Foundation и ISO 20000 Foundation Bridge» / Л. Будкова, Р. Журавлёв – М.: Клеверикс, 2010. – 123 с.

УДК 004.724.4(045)

КУЛАКОВ Ю.А.,
ДИБРОВА М.А.

СПОСОБ ФОРМИРОВАНИЯ ВИРТУАЛЬНОЙ СТРУКТУРЫ GRID, ОРИЕНТИРОВАННОЙ НА МНОГОПУТЕВУЮ МАРШРУТИЗАЦИЮ

В статье рассмотрены вопросы формирования виртуальных сетей для Grid. Приведена временная сложность формирования виртуальных каналов. Разработан алгоритм формирования с минимальной временной сложностью. Приведен пример формирования множества путей от одного узла к нескольким узлам Grid.

The article discusses the formation of virtual networks for Grid. Shows the time complexity of the formation of virtual channels. An algorithm for the formation of the minimum time complexity. An example of forming a plurality of paths from one node to multiple nodes Grid.

1. Введение

В настоящее время при организации Grid систем широко используются виртуальные частные сети (VPN) [1,2,3]. Узлы VPN объединяются между собой с помощью виртуальных каналов или туннелей, представляющих собой последовательность каналов передачи данных сети передачи. Как правило, топология виртуальной сети формируется с учетом информационных потоков между ее узлами. Например, при использовании технологии « клиент-сервер» VPN характеризуется звездообразной топологией. Узлы одноранговых VPN связываются по технологии P2P. При организации многоуровневых VPN виртуальные каналы формируют древовидную структуру сети.

Минимальным количеством виртуальных каналов $k= N-1$, где N – количество вершин, характеризуется VPN с топологией минимального покрывающего дерева. Наиболее универсальной является полносвязная топология VPN с количеством каналов $k= N(N-1)$.

В настоящее время при организации передачи информации в распределенных компьютерных системах широко используется многопутевая маршрутизация [4] при которой несколько физических каналов объединяются в один многоканальный виртуальный путь. Это позволяет повысить эффективность процедуры конструирования трафика (TE), обеспечить заданные параметры QoS при изменении параметров системы передачи информации, в частности при изменении метрики каналов, путем замены одного физического канала другим в рамках одного многоканального виртуального пути.

Временная сложность формирования многоканального виртуального пути состоящего из q физических каналов с помощью комбинаторных алгоритмов, например, алгоритма Дейкстры равна $O(qN^2)$. В связи с этим актуальным является разработка и использования потоковых алгоритмов многопутевой маршрутизации, характеризующихся минимальной временной сложностью [5,6]. Уменьшить временную сложность можно за счет исключения операции перебора вариантов на основе метода « ветвей и границ».

2. Способ формирования множества MVP в Grid системах

Как правило, в Grid системах используется многоточечное соединение, при котором одна вершина соединяется с несколькими вершинами, образуя звездообразную, древовидную или полносвязную топологию. С учетом этого предлагается использовать модифицированный алгоритм « ветвей и границ» [7] формирования множества путей между двумя вершинами. С целью дальнейшего уменьшения временной сложности формирования виртуальных каналов в данной работе предлагается формировать одновременно виртуальные каналы от одного узла v_i виртуальной сети к множеству смежных узлов $V_i =\{v_j|\forall e_{i,j}\}$ виртуальной сети, представленной в виде графа $G_z(V,E)$. В рамках каждого виртуального канала e_i, между вершинами v_i и v_j формируется множество $W_{i,j} = \{L_r \,|r =1,2,..m\}$ не пересекающихся физических каналов.

Вершины множества $B^i_1= \{в_i|\ i=1,2,\ldots,n\}$ подграфа $G_1(B_1,E_1)$ не смежные с граничными вершинами множества B_s будем относить к внутренним вершинам. Для множества внутренних вершин $B^i_1 \subset B_1$ выполняется условие $E^i=\{ e^i_{k,j}|\ в_k \in B_1,\ в_j \in B_1\}$. Соответственно ребро $e^i_{k,j}$ является внутренним ребром.

Вершины множества $B^o_1= \{в^o_i|\ i=1,2,\ldots,n\}$, смежные с вершинами множества B_s , будем называть граничными вершинами подграфа $G_1(B_1,E_1)$. Удаление граничных вершин приводит к тому, что граф становиться несвязным. Соответственно ребро $e^o_{k,j}$ будем называть граничным ребром. Для множества граничных вершин $B^o_1 \subset B_1$ выполняется условие $E^o=\{ e^o_{k,j}|\ в_k \in B_1,\ в_j \in B_2\}$.

Будем различать внутреннюю S^i_k и внешнюю S^o_k степени вершины.

Количество внутренних ребер вершины $в_k$ определяет ее внутреннюю степень S^i_k. В свою очередь ребро $e^o_{k,j}=\{ в_k \in B_1,\ в_j \notin B_1\}$ является внешним для подграфа $G_1(B_1,E_1)$, при этом вершина $в_k \in B_c$. Количество внешних ребер вершины $в_k$ определяет ее внешнюю степень S^o_k.

Для подграфа $G_1(B_1,E_1)$, представленного на рис.1, множество вершин которого $B_1=\{ в_0, в_1, в_2, в_3 \}$ граничными являются вершины $B_s=\{в_1, в_2, в_3\}$.

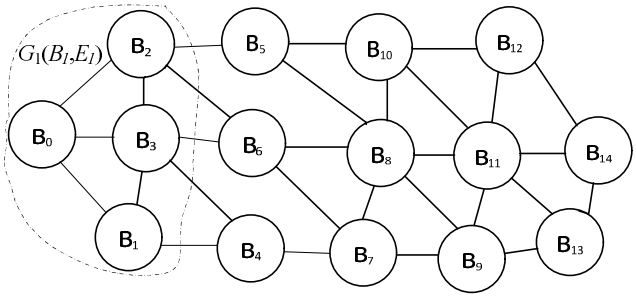


Рис.1. Граф сети

Минимальное множество граничных вершин $B_{s\ min}$ графа $G_0(B_0,E_0)$ составляют вершины $в_6$ и $в_7$, то есть $B_{s\ min}=\{в_6, в_7,\}$. Множество внутренних вершин подграфа $G_1(B_1,E_1)$ равно $B^i_1= \{в_0, в_1, в_2, \}$, а множество граничных вершин данного подграфа равно $B^0_1= \{в_3, в_4, в_5,\}$.

Процесс формирования множества осуществляется следующим образом. На начальном этапе предлагается формировать пути от исходной вершины до смежных с ней вершин $в_i \in B_s$, которые в данном случае являются граничными для множества, состоящего из одной исходной вершины. Затем граничные вершины включаются в множество внутренних вершин подграфа $G_1(B_1,E_1)$ и формируется новое множество B_s. Таким образом строится дерево решений с корнем в вершине $в_0$ до тех пор пока не сформируются все непересекающиеся пути ко всем заданным вершинам.

Временная сложность данного алгоритма равна $O(S_mD)$, где S_m – средняя степень вершин и D- диаметр графа.

С учетом введенных обозначений алгоритм формирования множества путей от одной вершины к заданному множеству вершин заключается в следующем.

1. Формируем подграф $G_1(B_1,E_1)$ в котором $B_1=\{ в_0, B_c \}$, где: $в_0$- исходная вершина; $B_s= \{в_i|\ i=1,2,\ldots,p\}$ – в данном случае представляет множество вершин, смежных с вершиной $в_0$.
2. Для подграфа $G_1(B_1,E_1)$ определяем множество путей $W=\{L_{o,r}|,r=1,2,\ldots,p\}$, где $L_{o,j}=\{ в_0, в_j \}$.
3. Формируем новое множество $B_1= B_1 \cup B_s$
4. Формируем подграф $G_1(B_1,E_1)$ в котором $B_1= B_1 \cup B_s$
5. Формируем подграф $G_2(B_2,E_2)$ в котором $B_2= B_2 \setminus B_s$
6. Для подграфа $G_1(B_1,E_1)$ формируем множество граничных вершин B_s.
7. Среди множества вершин $B_s= \{в_i|\ i=1,2,\ldots,p\}$ выбираем вершину $в_i$ с минимальной внешней степенью $S^o_k= \min \{S^o_i|\ i=1,2,\ldots,p\}$.
8. Для вершины $в_i$ находим внутреннюю вершину $в_j \in B^i_2$ подграфа $G_2(B_2,E_2)$ с минимальной степенью.
9. Продлеваем путь $L_{o,i}$ до вершины $в_j \in B^i_2$
10. Если $в_j \notin B^V$ то ($в_j \subset B_c$, $в_i \subset B^i_1$,) иначе (путь $L_{o,j}$ сформирован)
11. Если множество $B_1 \neq \varnothing$ то (переход к п. 7)
12. Конец.

Рассмотрим пример формирования множества путей от вершины от вершины $в_0$ к вершине $в_{14}$.

На рис. 2 представлен пример формирования первого пути от вершины $в_0$ к вершине $в_{14}$.

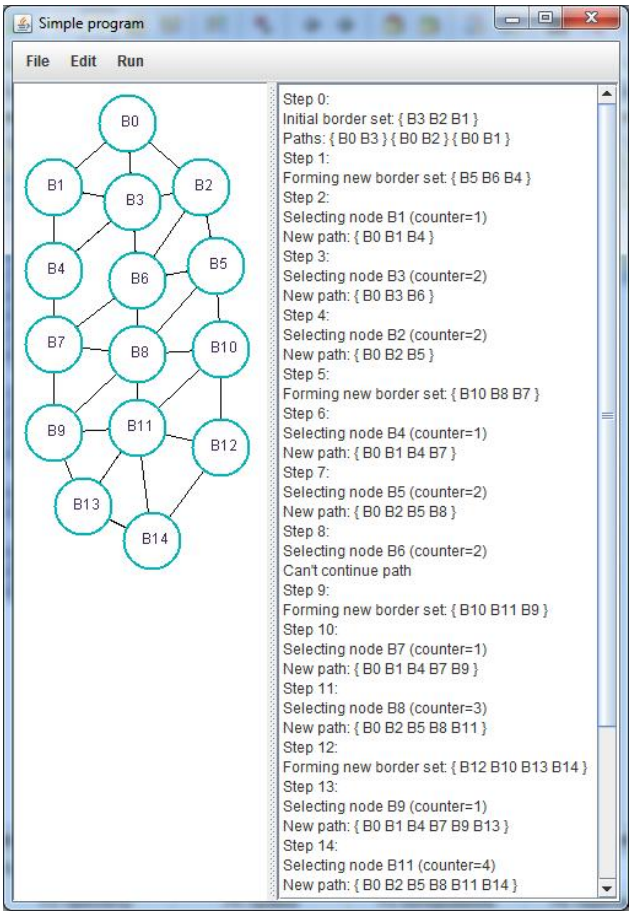


Рис. 2. Формирование первого пути от вершины $в_0$ к вершине $в_{14}$.

Формирование всего множества путей между вершинами $в_0$ и $в_{14}$ заключается в следующем:

Step 0:

Initial border set: { B1 B3 B2 } /*формирование множества граничных узлов для вершины $в_0$ */

Paths: { B0 B1 } { B0 B3 } { B0 B2 } /* формирование путей от вершины $в_0$ к вершинам $в_3$,$в_2$,$в_1$. */

Step 1:

Forming new border set: { B4 B6 B5 }/* новое множество граничных узлов */

Step 2:

Selecting node B1 (counter=1) /* выбор граничной вершины с минимальной значением внешней степени */

New path: { B0 B1 B4 } /* продолжение пути от вершины $в_0$ к вершине $в_4$ */

Step 3:

Selecting node B3 (counter=2) /* выбор граничной вершины с минимальной значением внешней степени */

New path: { B0 B3 B6 } /* продолжение пути от вершины $в_0$ к вершине $в_6$ */

Step 4:

Selecting node B2 (counter=2) /* выбор граничной вершины с минимальной значением внешней степени */

New path: { B0 B2 B5 } /* продолжение пути от вершины $в_0$ к вершине $в_5$ */

Step 5:

Forming new border set: { B8 B7 B10 } /* новое множество граничных узлов */

Step 6:

Selecting node B4 (counter=1) /* выбор граничной вершины с минимальной значением внешней степени */

New path: { B0 B1 B4 B7 } /* продолжение пути от вершины $в_0$ к вершине $в_7$ */

Step 7:

Selecting node B6 (counter=2) /* выбор граничной вершины с минимальной значением внешней степени */

New path: { B0 B3 B6 B8 } /* продолжение пути от вершины $в_0$ к вершине $в_8$ */

Step 8:

Selecting node B5 (counter=2) /* выбор граничной вершины с минимальной значением внешней степени */

New path: { B0 B2 B5 B10 } /* продолжение пути от вершины $в_0$ к вершине $в_{10}$ */

Step 9:

Forming new border set: { B11 B9 B12 } /* новое множество граничных узлов */

Step 10:

Selecting node B7 (counter=1) /* выбор граничной вершины с минимальной значением внешней степени */

New path: { B0 B1 B4 B7 B9 } /* продолжение пути от вершины $в_0$ к вершине $в_9$ */

Step 11:

Selecting node B8 (counter=2) /* выбор граничной вершины с минимальной значением внешней степени */

New path: { B0 B3 B6 B8 B11 } /* продолжение пути от вершины $в_0$ к вершине $в_{11}$ */

Step 12:

Selecting node B10 (counter=2) /* выбор граничной вершины с минимальной значением внешней степени */

New path: { B0 B2 B5 B10 B12 } /* продолжение пути от вершины $в_0$ к вершине $в_{12}$ */

Step 13:

Forming new border set: { B13 B14 } /* новое множество граничных узлов */

Step 14:

Selecting node B9 (counter=1) /* выбор граничной вершины с минимальной значением внешней степени */

New path: { B0 B1 B4 B7 B9 B13 } /* продолжение пути от вершины $в_0$ к вершине $в_{13}$ */

Step 15:

Selecting node B12 (counter=1) /* выбор граничной вершины с минимальной значением внешней степени */

New path: { B0 B2 B5 B10 B12 B14 } /* формирование первого пути от вершины $в_0$ к вершине $в_{14}$ */

Step 16:

Selecting node B11 (counter=2) /* выбор граничной вершины с минимальной значением внешней степени */

New path: { B0 B3 B6 B8 B11 B14 } /* формирование второго пути от вершины $в_0$ к вершине $в_{14}$ */

Step 17:

Forming new border set: { B14 } /* новое множество граничных узлов */

Step 18:

Selecting node B13 (counter=1) /* выбор граничной вершины с минимальной значением внешней степени */

New path: { B0 B1 B4 B7 B9 B13 B14 }/* формирование третьего пути к вершине $в_{14}$ */

Done

В результате формируются следующая структура графа путей (рис.3).

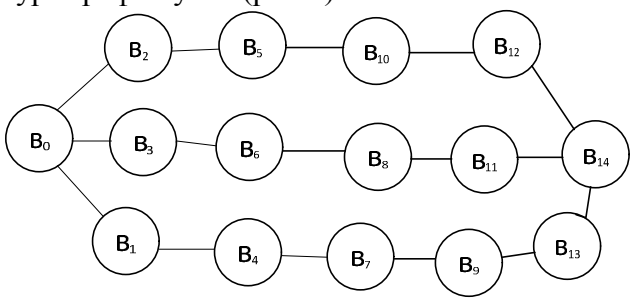


Рис.3. Множество путей между вершинами $в_2$ и $в_{14}$

Формирование первого пути { B0 B2 B5 B10 B12 B14 } к вершине $в_{14}$ происходит на 15 шаге из 18 шагов, второго пути { B0 B3 B6 B8 B11 B14 } происходит на 16 и третьего пути происходит на последнем 18. Таким образом, все множество путей формируется за один проход алгоритма. На формирование первого пути затрачивается 30 операций, на второго пути еще 2 операции и на третьего пути еще 3 операции, всего 35 операций. В среднем количество шагов равно числу пройденных вершин, умноженному на их среднюю степень.

Данный алгоритм позволяет сформировать множества путей от одной вершины к нескольким вершинам. Рассмотрим случай формирования путей от вершины $в_0$ множества вершин { $в_7$, $в_{10}$, $в_{14}$}.

На рис. 4 представлен пример формирования первого пути от вершины $в_0$ к вершине $в_7$

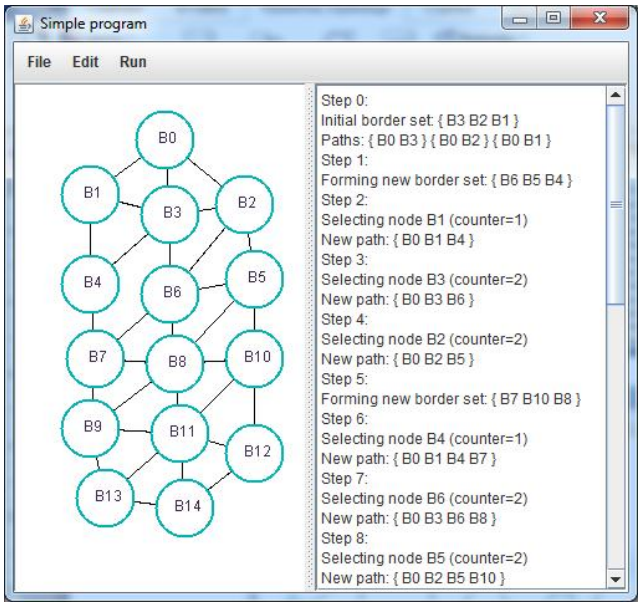


Рис.4. Формирование первого пути от вершины $в_0$ к вершине $в_7$.

В результате формируется дерево путей, представленное на рис. 5.

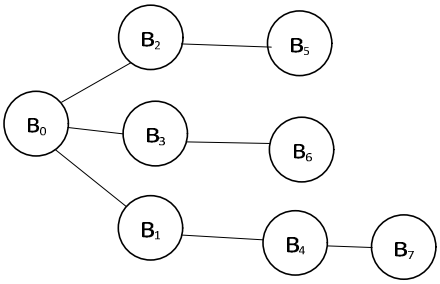


Рис. 5. Дерево путей

Следует заметить, что формирование первого пути к вершине $в_7$ (рис.4) совпадает с началом формирования пути к вершине $в_{14}$ (рис.2).

Формирование всего множества путей между вершинами $в_0$ и $в_7$ заключается в следующем:

Step 0:

Initial border set: { B3 B2 B1 } /*формирование множества граничных узлов для вершины $в_0$ */

Paths: { B0 B3 } { B0 B2 } { B0 B1 } /* формирование путей от вершины $в_0$ к вершинам $в_3$,$в_2$,$в_1$. */

Step 1:

Forming new border set: { B4 B6 B5 } /* новое множество граничных узлов */

Step 2:

Selecting node B1 (counter=1) /* выбор граничной вершины с минимальной значением внешней степени */

New path: { B0 B1 B4 } /* продолжение пути от вершины $в_0$ к вершине $в_4$ */

Step 3:

Selecting node B3 (counter=2) /* выбор граничной вершины с минимальным значением внешней степени */

New path: { B0 B3 B6 } /* продолжение пути от вершины $в_0$ к вершине $в_6$ */
Step 4:
Selecting node B2 (counter=2) /* выбор граничной вершины с минимальным значением внешней степени */
New path: { B0 B2 B5 } /* продолжение пути от вершины $в_0$ к вершине $в_5$ */
Step 5:
Forming new border set: { B10 B7 B8 } /* новое множество граничных узлов */
Step 6:
Selecting node B4 (counter=1) /* выбор граничной вершины с минимальным значением внешней степени */
New path: { B0 B1 B4 B7 } /* сформирован первый путь к вершине $в_7$ */
Step 7:
Selecting node B6 (counter=2) /* выбор граничной вершины с минимальным значением внешней степени */
New path: { B0 B3 B6 B7 } /* сформирован второй путь к вершине $в_7$ */
Step 8:
Selecting node B5 (counter=2) /* выбор граничной вершины с минимальным значением внешней степени */
New path: { B0 B2 B5 B8 } /* продолжение пути от вершины $в_0$ к вершине $в_8$ */
Step 9:
Forming new border set: { B10 B7 B11 B9 } /* новое множество граничных узлов */
Step 10:
Selecting node B8 (counter=4) /* выбор граничной вершины с минимальным значением внешней степени */
New path: { B0 B2 B5 B8 B7 } /* сформирован третий путь к вершине $в_7$ */
Finish riched
Done

В результате работы алгоритма формируются виртуальный канал $c_{0,7}$, состоящий из трех физических каналов (рис. 6):
$c_{0,7}$ ={ L_1={ $в_0$, $в_1$, $в_4$, $в_7$}; L_2={ $в_0$, $в_3$, $в_6$, $в_7$} L_3={$в_0$, $в_2$, $в_5$, $в_8$, $в_7$ } }.

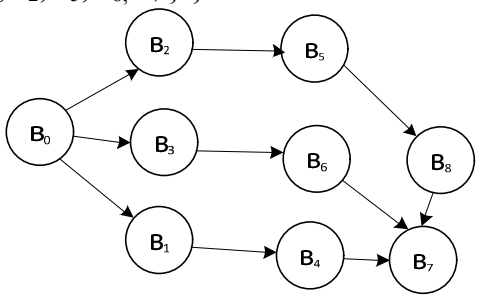


Рис.6. Множество путей от вершины $в_0$ до вершины $в_7$

После формирования пути до вершины $в_7$ (рис.4) продолжается формирование множества путей до следующей ближайшей вершины, в данном случае это вершина $в_{10}$.
Step 8:
Selecting node B5 (counter=2) /* граничная вершины с минимальной внешней степенью */
New path: { B0 B2 B5 B10 } /* сформирован первый путь к вершине $в_{10}$ */
Step 9
New path: { B0 B2 B5 B8 } /* продолжение пути от вершины $в_0$ к вершине $в_8$ */
Step 10:
Forming new border set: { B10 B7 B11 B9 } /* новое множество граничных узлов */
Step 11:
Selecting node B8 (counter=4) /* граничная вершины с минимальной внешней степенью */
New path: { B0 B2 B5 B8 B7 } /* сформирован третий путь к вершине $в_7$ */
Step 12:
Forming new border set: { B11 B9 B10 } /* новое множество граничных узлов */
Step 13:
Selecting node B7 (counter=1) /* граничная вершина с минимальной внешней степенью */
New path: { B0 B1 B4 B7 B9 } /* сформирован второй путь к вершине $в_9$ */
Step 14:
Selecting node B8 (counter=3) /* граничная вершина с минимальной внешней степенью */
New path: { B0 B3 B6 B8 B10 } /* сформирован второй путь к вершине $в_{10}$ */
Step 15:
Forming new border set: { B11 B13 } /* новое множество граничных узлов */
Step 16:
Selecting node B9 (counter=2) /* граничная вершина с минимальной внешней степенью */
New path: { B0 B1 B4 B7 B9 B11 } /* новое множество граничных узлов */
Step 17:
Selecting node B11 (counter=2) /* новое множество граничных узлов */
New path: { B0 B1 B4 B7 B9 B11 B10 } /* сформирован третий путь к вершине $в_{10}$ */
Done

В результате работы алгоритма в дополнение к виртуальному каналу $c_{0,7}$ формируются виртуальный канал $c_{0,10}$, состоящий из трех физических каналов (рис7):
$c_{0,10}$ ={L_4={ $в_0$, $в_1$, $в_4$, $в_7$, $в_9$, $в_{11}$, $в_{10}$ }; L_5={ $в_0$, $в_3$, $в_6$, $в_8$, $в_{10}$} L_6={ $в_0$, $в_2$, $в_5$, $в_{10}$ } }.

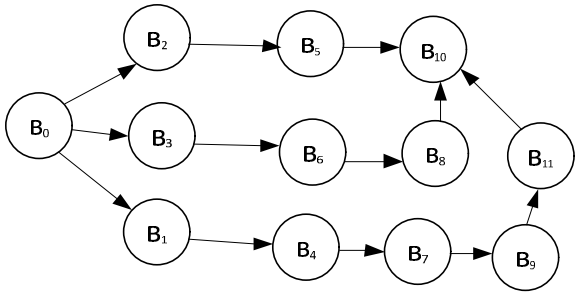


Рис.7. Формирование путей к вершине в$_{10}$

После формирования пути до вершины в$_{10}$ (рис.5) на следующем 18 шаге продолжается формирование множества путей до последней вершины в$_{14}$.

Step 18:
Forming new border set: { B11 B9 B12 } /* новое множество граничных узлов */
Step 19:
Selecting node B8 (counter=2) /* граничная вершина с минимальной внешней степенью */
New path: { B0 B3 B6 B8 B11 } /* сформирован новый путь к вершине в$_{11}$ */
Step 20:
Selecting node B10 (counter=2) /* граничная вершина с минимальной внешней степенью */
New path: { B0 B2 B5 B10 B12 } /* сформирован путь к вершине в$_{12}$ */
Step 21:
Forming new border set: { B13 B14 } /* новое множество граничных узлов */
Step 22:
Selecting node B9 (counter=1) /* граничная вершина с минимальной внешней степенью */
New path: { B0 B1 B4 B7 B9 B13 } /* сформирован путь к вершине в$_{13}$ */
Step 23:
Selecting node B12 (counter=1) /* граничная вершина с минимальной внешней степенью */
New path: { B0 B2 B5 B10 B12 B14 } /* <u>сформирован первый путь к вершине в$_{14}$</u> */
Step 24:
Selecting node B11 (counter=2) /* граничная вершина с минимальной внешней степенью */
New path: { B0 B3 B6 B8 B11 B14 } /* <u>сформирован второй путь к вершине в$_{14}$</u> */
Step 25:
Selecting node B13 (counter=1) /* граничная вершина с минимальной внешней степенью */
New path: { B0 B1 B4 B7 B9 B13 B14 } /* <u>сформирован третий путь к вершине в$_{14}$</u> */

В результате работы алгоритма в дополнение к виртуальным каналам $c_{0,7}$ и $c_{0,10}$ формируются виртуальный канал $c_{0,14}$, состоящий из трех физических каналов (рис. 8):

$c_{0,14}$ ={ L_7={ в$_0$, в$_1$, в$_4$, в$_7$, в$_9$, в$_{13}$, в$_{14}$ }; L_8={ в$_0$, в$_3$, в$_6$, в$_8$, в$_{11}$, в$_{14}$} L_9={ в$_0$, в$_2$, в$_5$, в$_{10}$, в$_{12}$, в$_{14}$ } }.

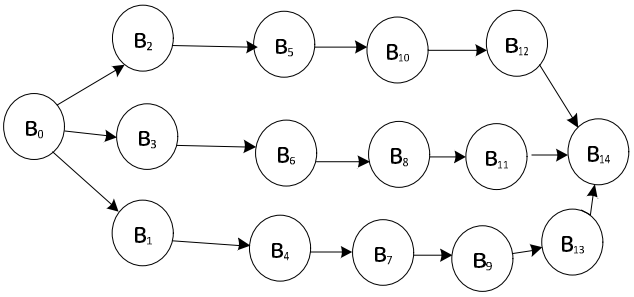


Рис.8. Формирование путей к вершине в$_{14}$

Таким образом, формируется виртуальная звездообразная топологи Grid поверх физической структуры сети (рис.9).

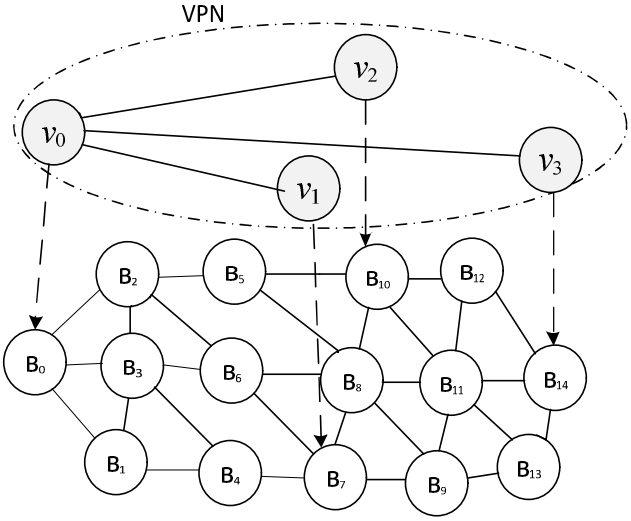


Рис.9. Виртуальная топология Grid системы

Для формирования полносвязной топологии необходимо дополнительно сформировать виртуальные пути от вершины в$_7$ к вершинам в$_8$, в$_{12}$ и от вершины в$_8$ к вершине в$_{12}$.

3. Заключение

Организация виртуальной структуры с многоканальными связями позволяет повысить эффективность функционирования Grid систем и обеспечить заданные параметры QoS. Предложенный в работе модифицированный алгоритм « ветвей и границ» за счет одновременного формирования множества путей от одного узла ко всем остальным узлам существенным образом уменьшает временную сложность формирования виртуальной структуры Grid систем.

В качестве одного из дальнейших направлений исследования является организации параллельной передачи частей сообщения по многоканальным виртуальным соединениям.

Список литературы

1. N. Mahesh Kumar Proposed Architecture For Implementing Privacy In Cloud Computing Using Grids And Virtual Private Network / N. Mahesh Kumar, K. Senthilkumar // International Journal Of Technology Enhancements And Emerging Engineering Research, Vol 1, ISSUE 3 ISSN 2347-4289 2013. P. 12-15.
2. Francesco Palmieri Introducing Virtual Private Overlay Network services in large scale Grid infrastructures/ Journal Of Computers, Academy Publisher VOL. 2, NO. 2, APRIL 2007. P. 61-72.
3. A. Di Costanzo "Harnessing cloud technologies for a virtualized distributed computing infrastructure,"/ A. Di Costanzo, M. D. De Assuncao, and R. Buyya // internet Computing, IEEE, vol. 13, pp. 24-33, 2009.
4. Eun-Sung Jung Distributed Multipath Routing Algorithm for Data Center Networks/ Eun-Sung Jung, Venkatram Vishwanath, Rajkumar Kettimuthu // 2014 International Workshop on Data Intensive Scalable Computing Systems
5. Lemeshko A.V. Research on Tensor Model of Multipath Routing in Telecommunication Network with Support of Service Quality by Greate Number of Indices / Lemeshko A.V., Evseeva O.Yu., Garkusha S.V // Telecommunications and RadioEngineering, Vol.73, No 15. - P. 1339-1360
6. Mahalakshmi.C Multipath Data Transfer Scheme for Virtual Private Networks/ Mahalakshmi.C, Ramaswamy. M// International Journal of Computer Applications (0975 – 8887) Volume 44– No.8, April 2012 P.27-31
7. Kulakov Y. The method of plurality generation of disjoint paths using horizontal exclusive scheduling / Y. Kulakov, A. Kogan // The science advanced. – 2014. – issue 9. – pp.16-18.

УДК 621.391

ДОРОГИЙ Я.Ю.

АЛГОРИТМ СТРУКТУРНОЇ ОПТИМІЗАЦІЇ НЕЙРОННОЇ МЕРЕЖІ

В статті розглянуте питання архітектурної оптимізації нейронної мережі. Запропонований алгоритм структурної оптимізації в ході навчання нейронної мережі.

In the article the question of neural network architecture optimization was considered. The algorithm of structure optimization on learning stage was offered

Введення в проблему

На сьогоднішній день апарат нейронних мереж широко використовується для розв'язання різноманітних задач. В зв'язку з цим, питання розробки алгоритму навчання, що мав би смогу динамічно оптимізувати структуру нейронної мережі, є дуже актуальним. Наявність такого методу дала б можливість досліднику швидше отримати структуру нейронної мережі, яка б найкраще відповідали предметній області та наявним вхідним даним.

Аналіз існуючих рішень

Для побудови оптимальної структури нейронної мережі використовується досить широке коло алгоритмів. Першим з таких алгоритмів є алгоритм черепичної побудови [1]. Ідея алгоритму полягає в додаванні нових шарів нейронів таким чином, щоб вхідні навчальні вектори, які мають різні відповідні вихідні значення, мали в ньому різне внутрішнє представлення. Ще одним яскравим представником є алгоритм швидкої надбудови [2]. Нові нейрони за цим алгоритмом додаються між вихідними шарами. Роль цих нейронів – корегування помилки вихідних нейронів. В загальному вигляді нейронна мережа, що побудована за таким алгоритмом, має форму бінарного дерева. Широко відомими є алгоритми Monoplan, NetLines та NetSphere [3], метод редукції [4]. Але всі ці алгоритми мають досить широкий перелік недоліків, тому було вирішено розробити свій алгоритм.

Мета роботи

Метою даної роботи є створення алгоритму структурної оптимізації нейронної мережі в ході її навчання.

Графова модель представлення нейронної мережі

Будь-яку нейронну мережу можна представити у вигляді графу. Багатошарова архітектура нейронної мережі представляє собою ациклічний граф G, представлений як набір вузлів L_i^j, згрупованих у кластери за шарами та зв'язків між ними. Приклад такого представлення нейронної мережі показано на рис. 1.

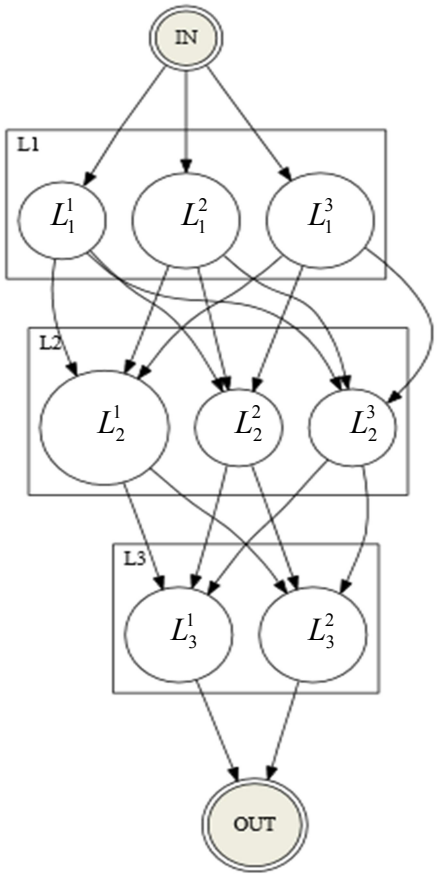


Рис.1. Графова модель НМ

Кожний вузол L_i^j представляє собою або елементарний нейрон, або елемент більш складної архітектури, наприклад, невелику нейронну мережу (рис. 2).

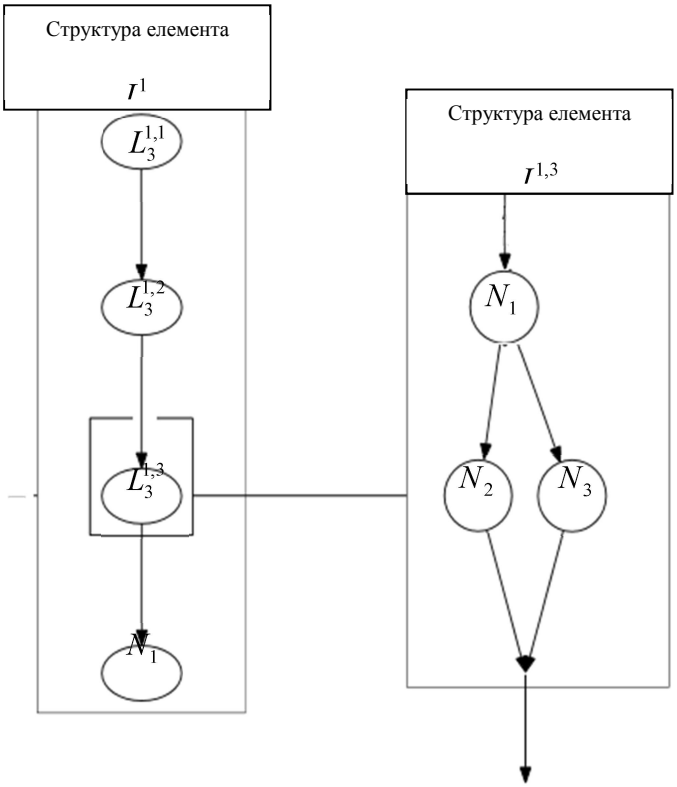


Рис.2. Структура вузла

Елементарні структурні операції над нейронною мережею

Введемо наступні елементарні структурні операції над мережею:

- додавання синапсу між двома випадково вибраними незв'язаними вузлами або нейронами мережі – операція Syn_{ADD};
- видалення синапсу між двома випадково вибраними незв'язаними вузлами або нейронами мережі – операція Syn_{DEL};
- переміщення синапсу між двома випадково вибраними незв'язаними вузлами або нейронами мережі – операція Syn_{MOD};
- зміна функції активації нейрона для випадково вибраного нейрона – операція A_{MOD};
- серіалізація вузла або нейрона – операції - Ser_{NODE} і Ser_{NR};
- паралелізація вузла або нейрона – операції Par_{NODE} і Par_{NR};
- додавання вузла або нейрона – операції Add_{NODE} і Add_{NR};
- створення нового шару – операція L_{ADD};
- видалення шару НМ – операція L_{DEL}.

Розглянемо операції більш детально.

Нехай нашу мережу задано наступним графом:

G = {V, E}, де

V – множина вузлів та нейронів;

E – множина синапсів, що їх зв'язуються;

V_A – підмножина вузлів, що належить попередньому підрівню (тобто такі, з яких виходять синапси);

V_B – підмножина вузлів, що належить наступному підрівню (тобто такі, в які входять синапси).

Операція додавання синапсу між двома випадково вибраними незв'язаними вузлами або нейронами мережі проілюстрована на рис. 3.

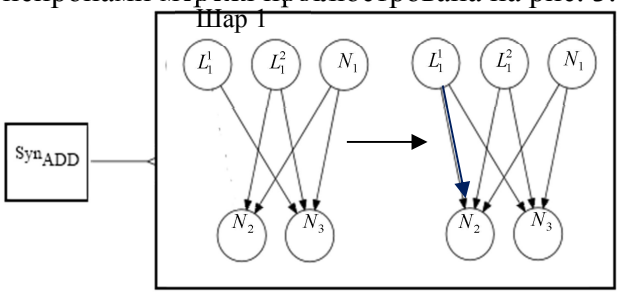


Рис.3. Операція Syn_{ADD}

Тобто,

$E(G) = E(G) \cup (l, n)$

$V_A = \{L_1^1, L_1^2, L_1^3\}, V_B = \{N_1, N_2\},$

$l \in V_A, n \in V_B.$

Операція видалення синапсу між двома випадково вибраними зв'язаними вузлами або нейронами мережі представлена на рис. 4.

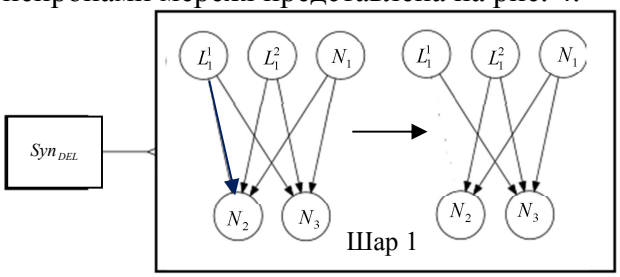


Рис.4. Операція Syn_{DEL}

Тобто,

$E(G) = E(G) \setminus (l, n)$

$V_A = \{L_1^1, L_1^2, L_1^3\}, V_B = \{N_1, N_2\},.$

$l \in V_A, n \in V_B.$

Операція переміщення синапсу між двома випадково вибраними вузлами або нейронами мережі представлена на рис. 5.

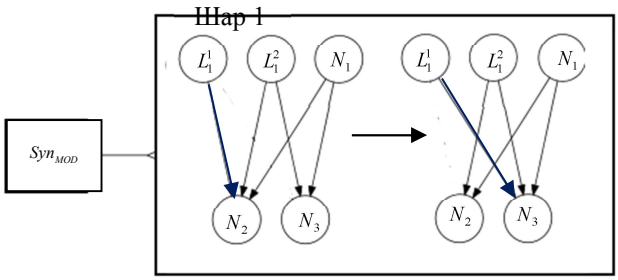


Рис.5. Операція Syn_{MOD}

Тобто,

$E(G) = E(G) \setminus (l,n);$

$E(G) = E(G) \cup (l,m);$

$V_A = \{L_1^1, L_1^2, L_1^3\}, V_B = \{N_1, N_2\},$

$l \in V_A, n, m \in V_B.$

Операція серіалізації Ser_{NODE} і Ser_{NR} випадково вибраного вузла або нейрона мережі представлена на рис. 6.

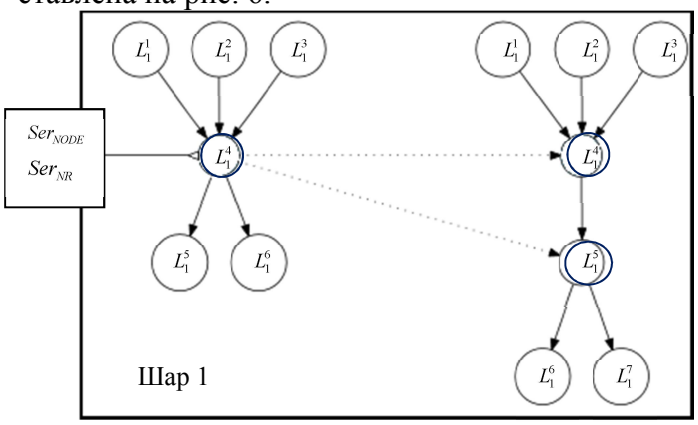


Рис.6. Операція серіалізації

Тобто,

$E(G) = E(G) \setminus \forall (l,n);$

$V_A = \{L_1^4\}, V_B = \{L_1^5, L_1^6\}, l \in V_A, n \in V_B.$

$V(G) = V(G) \cup L_1^5$ (переіндексація)

$V_B' = \{L_1^6, L_1^7\}, V_A' = \{L_1^5\}$

$E(G) = E(G) \cup (l,m); l \in V_A', m \in V_B';$

$E(G) = E(G) \cup (k,p); k \in V_A, p \in V_A'.$

Операція паралелізація Par_{NODE} і Par_{NR} випадково вибраного вузла або нейрона мережі представлена на рис. 7.

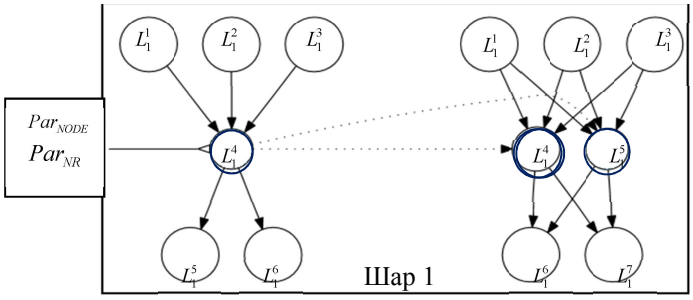


Рис.7. Операція паралелізації

Тобто,

$V(G) = V(G) \cup L_1^5$ (переіндексація)

$V_B' = \{L_1^6, L_1^7\}, V_A = \{L_1^1, L_1^2, L_1^3\}, V_A' = \{L_1^5\}$

$E(G) = E(G) \cup (l,m); l \in V_A, m \in V_A';$

$E(G) = E(G) \cup (k,p); k \in V_A', p \in V_B'.$

Операція додавання вузла або нейрона Add_{NODE} і Add_{NR} проілюстрована на рис. 8.

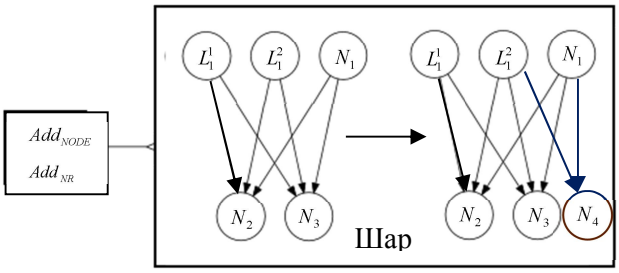


Рис.8. Операція Add_{NODE} ***і*** Add_{NR}

При додаванні нового вузла чи нейрона випадковим чином додаються декілька синапсів від створеного нейрона та до нього.

При видаленні вузла чи нейрона видаляються самі нейрони та вузли, а також всі синапси, що входять в даний елемент. Всі зв'язки, що виходять з даного елементу випадковим чином з'єднуються з елементами, що йдуть за видаленим.

Операція створення нового шару L_{ADD} є модифікатором для розглянутих вище операцій додавання вузла або нейрона та операцій серіалізації і паралелізації і вказує на те, що результат операції додається в новий шар мережі.

Алгоритм структурної оптимізації при навчанні

Розглянемо алгоритм структурної оптимізації при навчанні. В якості базового алгоритму навчання візьмемо алгоритм зворотного поширення помилки, запропонований в [5].

НМ, що розглядається, має сигмоїдальну функцію активації:

$$f(u) = \frac{1}{1+e^{-u}}, \qquad (1)$$

де

$$u = w_0 + w_1 x_1 + \ldots + w_n x_n. \qquad (2)$$

Введемо наступні позначення: множина навчальних даних $X_s = \{x_i \mid i = 1,\ldots,N\}$ та відповідні їм очікувані значення $Y_s = \{y_i \mid i = 1,\ldots,M\}$, де N – число входів НМ, M – число виходів НМ, $s = 1,..,S$, а S – вимірність навчальної вибірки. Позначимо множину значень виходів нейронів

НМ через $F_i = \{f_j^i\}$, де і – номер шару, а j – номер нейрона в шарі. Для вхідного шару $j = 1,2,...,N$, для першого прихованого шару – $j = 1,2,...,C_1$, другого – $j = 1,2,...,C_2$ і т.д., для останнього прихованого шару – $j = 1,2,...,M$. Загальна кількість шарів НМ дорівнює К та кількість нейронів в усіх прихованих шарах буде рівна $C = \{C_0, C_1,...C_K\}$, де $C_0 = N, C_K = M$.

Алгоритм зворотнього поширення помилки можна представити наступним чином:

1) Для вхідного шару встановлюємо значення кожного елементу відповідно з вхідним вектором. Значення виходу кожного елемента встановимо рівним входу.

$$F_0 = X_1. \quad (3)$$

2) Для нейронів першого шару обчислюємо загальний вхід та вихід:

$$u_j^1 = w_{j,0}^1 + \sum_{i=1}^{N} f_i^0 w_{j,i}^1, \quad (4)$$

де

$$f_j^1 = \frac{1}{1+e^{-u_j^1}}, \quad (5)$$

де $j = 1,2,..., N_1$ – номер нейрона першого прихованого шару, $w_{j,i}^l$ – і-й ваговий коефіцієнт j-го нейрона l–шару НМ, f_i^0 – значення і-го нейрона вхідного шару.

3) Повторюємо перший крок для всіх прихованих шарів НМ, включаючи вихідний шар нейронів, причому формули (4) та (5) трохи зміняться, та будуть мати вигляд:

$$u_j^k = w_{j,0}^k + \sum_{i=1}^{C_k} f_i^{c-1} w_{j,i}^c, \quad (6)$$

де

$$f_j^c = \frac{1}{1+e^{-u_j^c}}, \quad (7)$$

де $c = 2,..., K$.

4) Порівнюємо значення векторів Y_0 та F_K, якщо різниця між вектором – зразком Y_0 та реальним виходом НМ F_K знаходиться в допустимому діапазоні, то переходимо до кроку 5, якщо ні, то переходимо до кроку 6.

5) Для вхідного вектора встановлюємо значення кожного елемента, що дорівнює відповідному елементу $X_2 (F_0 = X_2)$ та знову переходимо до кроку 2. Якщо ж на вхід НМ подано останній вектор X_S, то в залежності від того, чи є необхідним навчання на цій ітерації, або знову подаються на вхід усі навчальні вектори, починаючи з першого, або вважається, що НМ навчена і навчання закінчується.

6) Для кожного нейрона вихідного шару обчислюється помилка. Оскільки розглядається НМ з сигмоїдальною функцією активації, то значення помилки дорівнює

$$\delta_i = \begin{cases} f_i^{'1}(1-f_i^{'1})(y_i - f_i^{'1}) \, при \, \mathrm{f_i} < f_i^{'1} \\ \mathrm{f_i}(1-\mathrm{f_i})(y_i - \mathrm{f_i}) \, при \, f_i^{'2} < \mathrm{f_i} < f_i^{'1} \\ f_i^{'2}(1-f_i^{'2})(y_i - f_i^{'2}) \, при \, \mathrm{f_i} > f_i^{'2} \end{cases}, \quad (8)$$

де f_i – отримане значення j-го елемента вихідного шару; $f_i^{'1}, f_i^{'2}$ – точки локальних екстремумів для очікуваного значення y_i в діапазоні [0,1].

7) Розраховується коефіцієнт мутації мережі:

$$M_{NET} = \frac{1}{\delta(t+1)}, \quad (9)$$

В залежності від розрахованого коефіцієнта виконується модифікація структури нейронної мережі за допомогою використання елементарних структурних операцій, розглянутих раніше (див. табл. 1). Тренд розрахованого коефіцієнта вказую на динаміку навчання. Якщо з кожною епохою навчання відбувається його зростання, тобто помилка зменшується, не має сенсу щось міняти в структурі мережі. І, навпаки, якщо тренд падає, потрібно модифікувати структуру мережі.

Табл. 1. Використання елементарних структурних операцій

Значення M_{NET}	Тренд M_{NET}	Операція
>1	Зростає	-
>1	Падає	L_{ADD}
>100	Зростає	-
>100	Падає	Ser_{NODE}, Ser_{NR}, Par_{NODE}, Par_{NR}, Add_{NODE}, Add_{NR}
>500	Зростає	-
>500	Падає	Syn_{ADD}, Syn_{MOD}, A_{MOD}

7) Для передостаннього шару обчислюється помилка кожного нейрона

$$\delta_i^{K-1} = f_i^{K-1}(1-f_i^{K-1}) \sum_{j=1}^{C_K} \delta_j^K w_{i,j}^{K-1}, \quad (10)$$

де δ_j^K – значення помилки j-го елементу К-го шару, $w_{i,j}^{K-1}$ – вага зв'язку між і-м елементом К-1 шару та j-м елементом К-го шару, f_i^{K-1} – значення і-го елемента К-1-го шару.

8) Для всіх інших прихованих шарів помилка обчислюється за формулою (10).

9) Далі, для всіх шарів поновлюються значення вагових коефіцієнтів кожного нейрона

$$\Delta w_{i,j}^c(t+1) = \eta(\delta_i^c f_j^{c-1}) + \alpha \Delta w_{i,j}^c(t), \qquad (11)$$

де η – швидкість навчання, α – інерція, або вплив значення попередньої зміни, зазвичай береться значення $\alpha < 1$, t – номер ітерації. І тоді встановлюємо нові значення вагових коефіцієнтів, рівними

$$w_{i,j}^c = w_{i,j}^c + \Delta w_{i,j}^c(t) \qquad (12)$$

10) Для вхідного вектора встановлюємо значення кожного нейрона, що дорівнює відповідному елементу X_2 $(F_0 = X_2)$ та знову виконуємо перехід до кроку 2. Якщо вже поданий останній вектор X_S на вхід НМ, то знову подається перший навчальний вектор.

Висновки

В ході дослідження побудований алгоритм структурної оптимізації нейронної мережі. Також, проведений ряд дослідів побудованого алгоритму, які доводять можливість використання його в задачах розпізнавання та класифікації даних.

Список використаної літератури

1. Mezard M., and Nadal J.P. Learning in feedforward layered networks: The Tiling algorithm // Journal of Physics. – 1989. – V. A22. – P. 2191 – 2203,
2. Frean M. The Upstart Algorithm: A Method for Constructing and Training Feed-Forward Neural Networks // Tech. Rep. 89/469, Edinburgh Univ., 1989.
3. Ash T. Dynamic Node Creation in Back-Propagation Networks // Connection Science. – 1989. – V. 1.
4. Mozer M.C., Smolensky P. Skeletonization: a technique for trimming the fat from a network via relevance assessment // Advances in Neural Information Processing Systems. – 1989. – V. 1. – P. 107 – 115.
5. Дорогий Я.Ю. Ускоренный алгоритм обучения сверточных нейронных сетей / Я.Ю.Дорогий // Вісник НТУУ «КПІ», «Інформатика, управління та обчислювальна техніка», №57. – 2012. – С. 150-154.

УДК 004.415.2.043

САЛАПАТОВ В.І.

МОДЕЛЮВАННЯ, ВЕРИФІКАЦІЯ ТА РОЗРОБКА ПРОГРАМ

У статті розглядається удосконалення методу MODEL CHECKING при створенні програм. При цьому технічне завдання у вигляді низки вимог, які записуються за допомогою рівнянь темпоральної логіки, використовуються безпосередньо для побудови автоматної моделі майбутньої програми. Таким чином, при створенні моделі одразу здійснюється її верифікація, оскільки вимоги будуть виконуватись у процесі побудови моделі. В подальшому автоматна модель, яка являє собою логічну структуру програми, використовується для створення програми на будь-якій процедурній мові програмування.

In the paper the improvement of the method MODEL CHECKING in programming is considered. This technical problem consist of a number of requirements that are represented using temporal logic equations and used directly to build the automaton model for future programs. Therefore, when creating models at once by its verification, since the requirements will be implemented in the model creating. Further automaton model that represents the logical structure of the program used to create applications on any procedural programming language.

Вступ

Коректність роботи програми, яка підлягає розробці, формально може бути доведена тільки по відношенню до її моделі. При цьому модель алгоритму майбутньої програми або її семантика являють собою спрощену абстракцію, яка може бути визначена за допомогою певних специфікацій у вигляді логічних обмежень або формул темпоральної логіки [1]. Перехід з одного стану автоматної моделі визначається не тільки змінами внутрішніх змінних, але і зовнішніх змінних (типа volatile). В такому разі стани моделі програми умовно розділені на такі, що залежать від зміни власних змінних, та ті, які змінюються ззовні. Час зміни таких змінних є не передбачуваним і повязаний із зовнішніми перериваннями. Тому перехід у стани, які викликані такими змінами, складають окремі випадки і пов'язані із синхронізацією паралельних процесів. Переходи у такі стани та порядок їх обробки мають описуватись окремо. Коректність роботи будь-якої паралельної програми залежить від коректності опису таких станів моделі. Тому для перевірки роботи паралельних програм особливо важливим є контроль поведінки моделі цих програм саме в таких станах, що викликаються зовнішніми подіями.

Суперпозиція графів моделі програм

Для створення комплексної моделі складних програм доцільно створити моделі усіх незалежних програмних модулів або гілок. Кожна з таких гілок має бути визначена сукупністю обмежень та специфікацій у вигляді рівнянь темпоральної логіки S. Після побудови графів автоматних моделей та їхньої верифікації треба виконати суперпозицію цих моделей і створити спільну модель. Під час такої сборки моделі - $Sc = \bigcup_{i=1}^{N} Si$ можуть з'явитись додаткові специфікації, тобто модель буде уточнюватись. Тут Sc об'єднання специфікацій від кожної незалежної гілки паралельної програми. Крім того, така сборка може виявити однакові специфікації у вигляді рівнянь в різних гілках паралельної програми. А це призводить до спрощення моделі програми - спільні специфікації можуть зменшити кількість станів моделі паралельної програми та породження нових станів.

Такий процес нормалізації графової моделі програми дозволяє оптимізувати її за кількістю станів та переходів між цими станами. Схожі вимоги специфікацій, якім мають відповідати окремі програмні фрагменти, можуть бути визначені як частина дій у кожному фрагменті моделі, або як спільна частина її нового стану. В першому випадку спільній частині видповідатиме певна підпрограма, а в другому - нова об'єднана вершина на графовій моделі. В такому випадку процес верифікації розбивається на кілька етапів і значно спрощується. В разі оформлення підпрограм специфікації, які визначають відповідні частини програмної моделі, мають повністю співпадати і знаходитись в різних частинах графа. Якщо вони сходяться до одного стану, то тоді вони створюють новий спільний стан моделі програми. При цьому повне співпадіння специфікацій не обов'язкове, їхні відмінності можуть породжувати певні розгалуження. Таким чином, процес верифікації мо-

делі програми на відміну від класичної технології MODEL CHECKING спрощується і не вимагає наявності ліцензійних і коштовних програм-верифікаторів. Такий процес удосконалення і спрощення програмних моделей нагадує нормалізацію баз даних, коли данні, з одного боку стиснюються, а з іншого - зберігається їхня цілісність і несуперечливість. В процесі такого спрощення спочатку визначаються схожі специфікації моделі, а потім здійснюється спрощення моделі.

Побудова програми за її моделлю

Після того, як в результаті побудови та нормалізації автоматної моделі програми виконана її перевірка на відповідність заданим специфікаціям можна перейти безпосередньо до створення програми. Обрання мови програмування в якості процедурної, інструментальної мови залежить від вимог замовника та міркувань виконавця. Власно процес програмування полягає у послідовному обході графу автоматної моделі, починаючи з її початкового стану до кінцевого стану по всіх можливих шляхах. Лінійні дії в кожному стані моделі автомату реалізуються відповідними мовними конструкціями, а функції переходу - за допомогою розгалужень. Такий підхід створення програм дозволяє перейти від інтуітивної побудови блок-схем алгоритмів програм до формалізації побудови їх автоматних моделей для наступного безпосереднього кодування. Логічна структура програми повинна створюватись згідно вимог технічного завдання на розробку. Треба також зпзначити, під час моделювання можуть виявитись недоліки і навіть помилки в специфікаціях технічного завдання, які легше і простіше виявити саме на цьому етапі. Крім того, на цьому етапі простіше вносити доповнення і можливі зміни в технічне завдання у вигляді нових специфікацій у модель.

Автоматна модель програми

Кінцеві автомати є дуже зручним формалізмом, який кінцевим чином задає звичайні формальні мови - нескінченні множини ланцюжків кінцевої довжини. Основою для побудови автоматної моделі програми є структура Крипке [1] Модель програми у вигляді кінцевого автомату створюється згідно її логічного опису та математичної моделі. Особливістю побудови такої моделі є наявність у кожному її стані власної функції збудження, що ускладнює подання автоматної моделі у табличному вигляді. Функції переходу з одного стану в інший також визначаються в кожному стані по-різному. Побудову автоматної моделі покажемо на прикладі створення програми обробки виразу за стековим алгоритмом [2]. Для опису математичної моделі будемо користуватись апаратом темпоральної логіки [1]. Тоді для обробки виразу його синтаксис та семантику можна описати наступним чином.

get (1); t U r (2);
$(p(t_i) > (t_{i-1}))$ U tostack (3);
$(p(t_i) \geq p(t_{i-1}))$ U code, fromstack (4);
(stack = Ø V (r = fin) U END (5);
(stack = Ø V (r ≠ fin) U get) (6);
(t_b = '-' V r = Ø) U (r = 0) (7);
(t = '(') V r ≠ Ø) U funct (8);
(t = '(') V r = Ø) U tostack (9);
(t = '(') V r = Ø) U Ee t = ')' (10);
(t_{i+1} = ')') V r ≠ Ø V t_i ≠'(') U goto 4 (11);
(t_i = '(' V t_{i+1} = ')') U lost, razd (12).

Робота алгоритму починається з пошуку пар лексем t та r – лексеми даних та лексеми дії (1). Формула (2) вказує на те, що елементи виразу складаються з пар лексем: лексеми даних - r та лексеми дії - t. Друга формула (3) вказує, якщо пріоритет лексеми дії у поточній парі лексем більше за такий самий пріоритет у попередній парі, то поточна пара лексем зберігається у стеку. Наступна формула вказує (4), що у звортному випадку породжується проміжний код та перевіряється стек. Формула 5 вказує, що програма завершується, коли стек пустий та лексема дії вказує на кінець виразу. Якщо лексема дії не вказує на кінець виразу (6) то продовжується пошук чергової пари лексем. Формула 7 описує варіант наявності унарного мінусу на початку виразу. У формулах 8-12 враховуються особливості обробки дужок. Так, формула 8 визначає обробку функції (після ідентифікатора – ліва дужка). Формула 9 визначає обробку лівої дужки без лексеми даних як високо пріоритетну операцію. Формула 10 вказує, що коли є права дужка, то до кінця виразу (Ee) обов'язково повинна з'явитись права дужка. У формулі 11 визначається порядок дій при обробці правої дужки, коли попередня лексема дії не є правою дужкою, тобто перехід до формули 4. Якщо стек виявиться пустим, то маємо помилку порушення парності дужок. І, нарешті, остання формула (12) вказує на умову знищення дужок (lost) та пошук після правої дужки лексеми дії (razd). Треба зазначити, що правила включають використання певних функцій. Функція get означає пошук пари лексем даних та дії. Функція tostack передбачає запис у стек пари лексем

даних та дії. При виконанні функції fromstack аналізується стек та якщо він не пустий, то виконуються формула 4, інакше - формули 5 та 6. Функція code вказує на породження коду, а функція lost на знищення парних дужок. Функція END вказує на умову закінчення роботи алгоритму обробки виразу. Для того, щоб побудувати автоматну модель алгоритму, треба визначити порядок застосування формул. Частково це вже було зроблено стосовно функції fromstack

Оскільки модель програми створюється у процесі її розробки і при цьому ще й перевіряється, то верифікація її моделі вже не потрібна. Коректність створеної програми визначається коректністю її автоматної моделі згідно специфікацій.

Побудова автоматної моделі програми та її верифікація

Формули темпоральної логіки більш конкретно дозволяють і точно описати модель програми ніж більш абстрактні та менш інформативні форми Бекуса-Наура. Для створення автоматної моделі треба пов'язати порядок застосування цих формул. Коли це буде зроблено, фактично отримаємо автоматну модель програми. Початковий та кінцевий стани автоматної моделі визначені у формулах 1 та 5. Отже, в разі наявності пари лексем (формули 1 та 2) наступним є застосування формул 3 та 4. Якщо лексема даних відсутня, то застосовуються формули 7, 9, 10. Формула 8 передбачає обробку функцій, які сприймаються як лексеми даних та обробляються окремою процедурою. Формули 3 та 4 застосовуються за наявності двох пар лексем, причому формула 5 визначає кінцевий стан автоматної моделі.

Формула 7 може бути застосована спільно з формулами 7, 9, 10, тобто функція переходу визначається умовами в цих формулах і програмно реалізується у вигляді розгалуження. Графічно модель програми представлена на малюнку нижче.

Стан 4 має додаткові стани, які ілюструють виконання функцій формування коду та аналізу стеку - code та fromstack. Стан 12 також розбитий на два стани, щоб показати наочно процедури lost та razd. Виділення станів 4_1 та 4_2 дозволило оптимізувати граф і виконати перехід із стану 11 у стан 4_1, а спільне використання процедури tostack у станах 2 та 9 – використати однакові дії у стані 3.

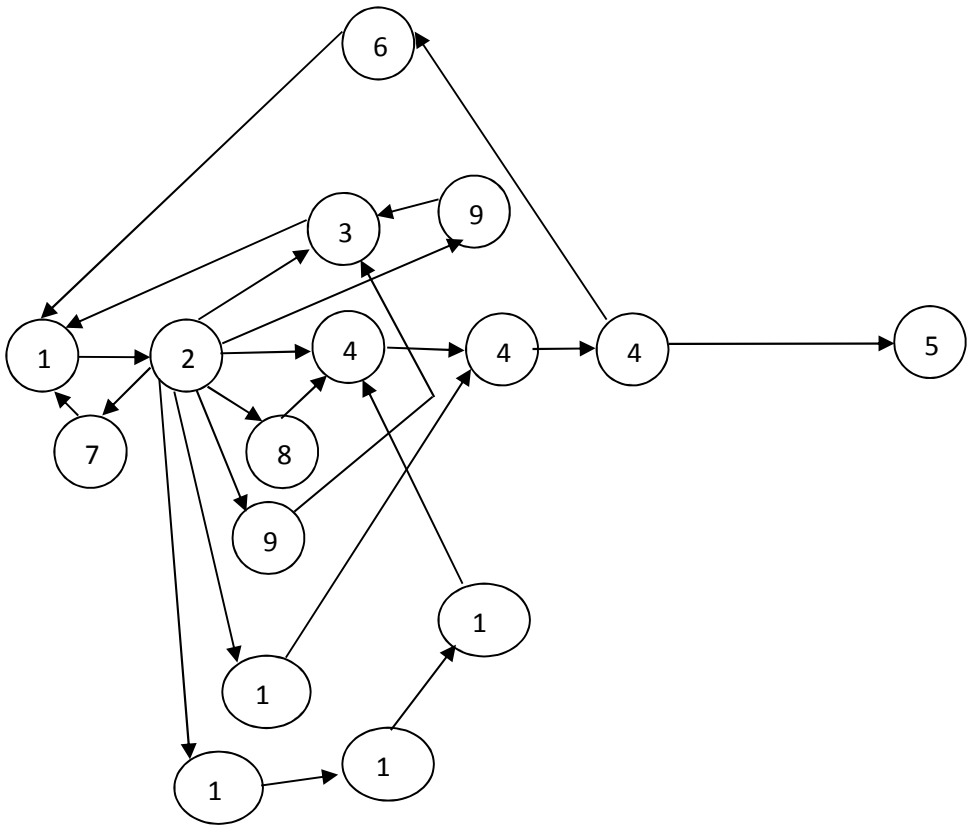


Рис.1. Граф моделі програми обробки виразу

Висновки

На даному прикладі показана формалізація опису програм в термінах темпоральної логіки, частково застосована суперпозиція окремих станів. При побудові графічної моделі це дозволило уточнити точки переходу з одного стану в інший і таким чином оптимізувати графову модель. Під час опису моделі перевірялось виконання формул темпоральної логіки, а також здійснювались зв'язки при переході від одних формул до інших. Тому процес верифікації моделі програм здійснювався безпосередньо при її створенні як обов'язковий етап технологічного процесу. При створенні більш складних програмно-апаратних систем доцільно виконати розбиття їх на окремі складові, побудувати їхні автоматні моделі, а потім виконати комплексну верифікацію цих моделей та здійснити остаточну перевірку. Такий підхід усуває застосування програм-верифікаторів, дозволяє перевірити виконання формул темпоральної логіки і спрощує процес створення складних систем. А розширення темпоральної логіки додатковими вказівками щодо переходу з одного стану в інший додатково полегшують побудову моделі.

Список посилань

1. Карпов Ю.Г. MODEL CHECKING. Верификация параллельных и распределенных программных систем. /Ю.Г. Карпов – СПб.; БХВ-Петербург, 2010.- 560 с.
2. Салапатов В.І. Синтаксичний аналіз із розподілом лексем на групи / В. Салапатов// Вісник національного технічного університету України «КПІ», Інформатика, управління та обчислювальна техніка. Київ. – 2008. - № 49. - С. 29-33.

Відомості про авторів

Бабенко А.А.	студенка кафедри відновлювальних джерел енергії НТУУ «КПІ»
Білашевська А. В.	студентка кафедри обчислювальної техніки
Боброва Н.Л.	старший викладач кафедри мікропроцесорних систем та мереж, Білоруський державний університет інформатики та радіоелектроніки
Бойко Ю.В.	к.ф.-м.н., директор ІОЦ КНУ ім. Тараса Шевченка
Виноградов Ю.М.	старший викладач обчислювальної техніки НТУУ «КПІ»
Габінет А. В.	аспірант кафедри обчислювальної техніки НТУУ «КПІ»
Глибовець М.М.	д.ф.-м.н., професор, декан факультету інформатики, завідувач кафедри інформатики національного університету «Києво-Могилянська академія»
Губська А. О.	студентка кафедри обчислювальної техніки НТУУ «КПІ»
Гусев Е. И.	здобувач кафедри обчислювальної техніки НТУУ «КПІ»
Демчинський В. В.	к.т.н., доц. кафедри інформаційної безпеки НТУУ «КПІ»
Денісюк О.В.	аспірант кафедри системного програмного забезпечення Одеського національного політехнічного університету
Діброва М. А.	аспірант кафедри обчислювальної техніки НТУУ «КПІ»
Долголенко О. М.	доцент кафедри обчислювальної техніки НТУУ «КПІ»
Дорогий Я.Ю.	к.т.н, асистент кафедри автоматики та управління в технічних системах НТУУ «КПІ»
Єршов С.В.	д.ф.-м.н., с.н.с., завідувач відділу методів та технологічних засобів побудови інтелектуальних програмних систем, Інститут кібернетики імені В.М. Глушкова НАН Українис
Жданова О.Г.	к.т.н., доцент кафедри АСОІУ НТУУ «КПІ»
Зайцева К. А.	аспірант кафедри АПЕПС факультету ТЕФ
Згуровський М.З.	д.т.н., професор, академік НАНУ, ректор НТУУ «КПІ»
Кавіцька В.С.	старший викладач кафедри системного програмного забезпечення Одеського національного політехнічного університету
Клименко І.А.	к.т.н., доцент, докторант НТУУ «КПІ»
Коваль О. В.	к.т.н., доцент кафедри АПЕПС факультету ТЕФ, заступник проректора з наукової роботи
Кривий С.Л.	д.ф.-м.н., професор кафедри інформаційних систем факультету кібернетики КНУ ім. Тараса Шевченка
Кулаков Ю.О.	д.т.н., професор кафедри обчислювальної техніки НТУУ «КПІ»
Лисецький Т.М.	пошукач каф. автоматизованих систем обробки інформації та управління НТУУ «КПІ»
Любченко В.В.	д.т.н., доцент кафедри системного програмного забезпечення Одеського національного політехнічного університету
Марковський О. П.	кандидат технічних наук, доцент кафедри обчислювальної техніки
Мельников О.В.	к.т.н., н.с. НДІ Інформаційних процесів НТУУ «КПІ»
Михайленко В.В.	к.т.н., ст. викладач кафедри теоретичної електротехніки НТУУ «КПІ»
Місюра О.Б.	к.т.н., с.н.с. НДІ Інформаційних процесів НТУУ «КПІ»
Невдащенко М. В.	старший викладач обчислювальної техніки НТУУ «КПІ»
Павлов О.А.	д.т.н., проф., зав. кафедри автоматизованих систем обробки інформації та управління НТУУ «КПІ»
Петрук В. А.	студент кафедри обчислювальної техніки НТУУ «КПІ»
Погорілий С.Д.	д.т.н., професор, завідувач кафедри комп'ютерної інженерії факультету радіофізики, електроніки та комп'ютерних систем КНУ ім. Тараса Шевченкас
Роковий О. П.	старший викладач обчислювальної техніки НТУУ «КПІ»
Ролік О.І.	д.т.н., професор кафедри автоматики та управління в технічних системах НТУУ «КПІ»
Салапатов В.І.	к. т. н., доцент, декан факультету обчислювальної техніки, інтелектуальних та управляючих систем Черкаського національного університету імені Богдана Хмельницького
Сперкач М.О.	асистент кафедри АСОІУ НТУУ «КПІ»
Стіренко С. Г.	к.т.н., доцент кафедри обчислювальної техніки НТУУ «КПІ»
Сулема Е.С.	к.т.н., доцент кафедри програмного забезпечення комп'ютерних систем НТУУ «КПІ»
Теленик С.Ф.	д.т.н., професор, завідувач кафедри автоматики та управління в технічних системах НТУУ «КПІ»
Халус О. А.	асистент автоматизованих систем обробки інформації та управління НТУУ «КПІ»
Широчин С. С.	аспірант кафедри програмного забезпечення комп'ютерних систем
Ясочка М.В.	к.т.н., начальник відділу наукового супроводження розробок ТОВ «Неткрекер»

SUMMARY

Zgurovskiy M.Z., Pavlov A.A., Misura E.B. The lexicographical criterion minimization for a feasible schedule on parallel machines with arbitrary due dates ……..... 4

Kulakov Y.O., Klymenko I.A. The multilevel memory in the reconfigurable computing system …….. 18

Pavlov O.A, Khalus O.A. Creation of feasible schedule of performing an operations at a single device optimal by criterion of minimizing the total earliness……...……. 27

Stirenko S.G. Implementation of the method for determining transmission delay time the computer cluster systems……...…….…… 35

Koval O.V., Zaitseva K.A. Verification of computer model of information management system ……... 43

Bobrova N.L. Justify the use of a complex of diagnostic techniques for assessing human psycho-phyiological state ..…….……... 49

Rokovoy A. P., Gabinet A. V.,Petruk V.A. Localization method of the multimedia traffic in the peer-to-peer networks ……. 54

Zgurovsky M.Z., Pavlov A.A., Misura E.B., Melnikov O.V., Lisetsky T.N. Implementation of the methodology of the four-level model of planning, decision-making and operational management in networked systems with limited resources ……...…….…… 60

Shyrochyn S.S., Sulema Y.S. Method of steganographic data protection in audio-files based on complimentary image ……..…… 80

Pavlov A.A., Zhdanova O.G., Sperkach M.O. The problem of creating a feasible schedule with maximum late moment of starting the jobs with common due date on identical parallel machines …… 88

Markovskiy O.P., Nevdashenko M.V., Bilashevska A.V. Protected implementation of image filtering in grid-systems ……..... 98

Demchinskyy V.V. Practical aspects of the traffic shaping and policing……... 105

Denisiuk O.V., Kavitska V.S., Liubchenko V.V. Definition of adequate model for knowledge representation in modern information systems …….…….…… 109

Boyko Y.V., Glybovets N.N., Yershov S.V., Kryvyi S.L., Pogorilyy S.D., Rolik O.I., Telenyk S.F., Yasochka M.V. High performance IT-infrastructure management ……...…… 120

Mihylenko V.V. Vinogradov Y.M., Babenko A.A. Computer analysis semiconductor transformers of electric power with the multiramified structures ...…… 143

Rolik O.I. The complex method of service level management in the corporate IT-infrastructure.....…… 149

Kulakov Y. A., Dibrova M.A. Method of GRID virtual structure forming, oriented on multipath routing 162

Dorohyi Y.Y. Algorithm of neural network architecture optimization …….. 169

Salapatov V.I. Modelling, verification and development program ……... 174

Information about authors ……..……..…….. 178

Summary……. 179