ВІСНИК

НАЦІОНАЛЬНОГО ТЕХНІЧНОГО УНІВЕРСИТЕТУ УКРАЇНИ
"Київський політехнічний інститут"

Інформатика, управління та обчислювальна техніка

53

2011

Міністерство освіти і науки України
Національний технічний університет України “КПІ”

ВІСНИК

НАЦІОНАЛЬНОГО ТЕХНІЧНОГО УНІВЕРСИТЕТУ УКРАЇНИ “КПІ”

Інформатика, управління та обчислювальна техніка

Заснований у 1964 р.
Випуск 53

Київ “ВЕК+” 2011

УДК 004

Вісник НТУУ «КПІ». Інформатика, управління та обчислювальна техніка: Зб. наук. пр. – К.: Век+, – 2011. – № 53. – 211 с.

Рекомендований до друку Вченою радою факультету інформатики та обчислювальної техніки, протокол № 4 від 23.12.2010

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

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

ISSN 0201-744X
ISSN 0135-1729

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 13.07.2011. Формат 60×84 1/16. Гарнітура Times. Папір офсетний №1. Наклад 150 прим.
Надруковано в ЗАТ "ВІПОЛ", 03151, г. Київ, вул. Волинська, 60.

ЗМІСТ

Павлов А.А., Мисюра Е.Б., Костик Д.Ю. Минимизация суммарного запаздывания при налички заданий с отрицательными значениями директивных сроков……..……..…..3
Єфремов К.В., Мазур Р.Ф. Системний та структурний рівні організації мультиагентних систем з можливістю адаптації процесу обробки даних …….....6
Симоненко В.П., Куренёв А.С. Алгоритм статического планирования для GRID систем..……...10
Марковский А.П., Абу Усбах А.Н., Альмуради В.М. Организация идентификации абонентов многопользовательских систем с использованием аппаратно защищенной памяти……...……17
Марковский А.П., Шаршаков А.С. Способ ускоренной генерации псевдослучайных перестановок элементов в системах защиты информации…….…...24
Лапчук А.С., Юрлов В.І., Шило С.О., Шиховця О.В. Інтерференційні ефекти при відтворенні інформації частково когерентним випромінюванням……...28
Симоненко В.П. Математическая постановка задачи динамического распределения работ в GRID системах и оценки качества решения……...37
Симоненко В.П., Симоненко А.В. Динамическое распределение работ по ресурсам неоднородной системе с ограничениями реального времени……..……..42
Симоненко А.В. Элементы теории повышения эффективности решения задачи динамического планирования в GRID системах……...…48
Томашевський В.М., Яцишин А.Ю. Математична модель задачі проектування гібридних сховищ даних з врахуванням структур джерел даних…….54
Демчинський В.В., Дорогий Я.Ю., Дорошенко К.С. Аспекти асимметричної маршрутизації в internet..……..…62
Болдак А.О., Пустовит М.А. Архитектура информационной системы интеллектуальной обработки данных…….68
Стеценко И.В. Формальное описание систем средствами Петри-объектных моделей……74
Полторак В.П., Вітіщенко Н.С. Калькулятор GF(q) для циклічних кодів…….82
Томашевський В.М., Грегуль В.В., Назарук О.В., Тимошенко Я.Я. Оцінка ефективності впливу на фактори ризику системи охорони здоров'я засобами імітаційного моделювання…….90
Янчевський С.Л. Многокритериальная оптимизация планирования космической съемки на основе геопространственной експертной информации……...96
Молчановський О.І., Мистецкий В. А., Нгієм Ле Куан, Буй Хиу Дат Задача пошуку файлів зображень…….106
Зайченко Ю.П., Ови Нафас Агаи Аг Гамиш Исследование зависимости «доходность-риск» в задаче оптимизации нечеткого портфеля…….114
Болдак А.А., Сухарев Д.Л. Определение количества кластеров в статистических даннях……..118
Мурга М.О. Модифікація методу навчання радіальної базисної мережі та її порівняння з нечіткою нейронною мережею МАМДАНІ та ННМАННКП…….....123
Федоречко О.И.,Виноградов Ю.Н.,Иванов А.Н. Метод исправления "пачки" ошибок в каналах с кодово-импульсной модуляцией на основе умножения без межразрядных переносов……..134
Пустоваров В.І. Механізми включення засобів формальних специфікацій до баз знань ком'пютерних мов…….…...142
Грибенко Д.В.,Стіренко С.Г. Реалізація розв'язання задачі знаходження власних чисел та векторів методом ланцоша на кластерній системі……...150
Ролік О.І., Можаровский П.Ф.,Вовк В.М., Захаров Д.С. Метод зведення метрик якості функціонування компонентів IT-інфраструктури за допомогою апарату непараметричної статистики……160
Амонс А.А.,Зайцев С.Ю.,Киричек А.А. Выявление плагиата в программном коде C#.....170
Ульяницкая К.А., Кунщиков Е.О., Островский С.М. подход к использованию матричного метода для автоматизации бизнес процессов……180
Ващук Ф.Г., Павлов О. А., Місюра О. Б., Мельник О.О. Складання розкладів сумарного випередження і запізнення із налагодженнями, що залежать від послідовності……192
Игнатов В.А., Гузий Н.Н.,Сорая М.А. Оптимальное обеспечение гарантоспособности телекоммуникационных и компьютерных сетей……..……195
Петренко О.О. Метод обчислення власних векторів матриці, що базується на зшироному методі діагональної модифікації……203
Павлов А.А., Мисюра Е.Б., Лисецкий Т.Н. Объединение работ в группы с учетом их приоритетов, готовности к выполнению и директивных сроков……...209

УДК 519.854.2

ПАВЛОВ А. А.,
МИСЮРА Е. Б.,
КОСТИК Д.Ю.

МИНИМИЗАЦИЯ СУММАРНОГО ЗАПАЗДЫВАНИЯ ПРИ НАЛИЧИИ ЗАДАНИЙ С ОТРИЦАТЕЛЬНЫМИ ЗНАЧЕНИЯМИ ДИРЕКТИВНЫХ СРОКОВ

Рассматриваются новые правила отсечений бесперспективных перестановок в задаче минимизации суммарного запаздывания при выполнении независимых заданий одним прибором для случая, когда директивные сроки заданий могут принимать как положительные, так и отрицательные значения.

New rules for truncation unpromising permutations are considered for the problem of minimizing the total tardiness of independent tasks execution on one machine for the case when the deadlines of tasks can take both positive and negative values.

Введение

В книге [1] приведен эффективный точный ПДС-алгоритм решения труднорешаемой задачи комбинаторной оптимизации «Минимизация суммарного запаздывания при выполнении независимых заданий на одном приборе» (МСЗ).

Постановка задачи. Предположим, что задано множество независимых заданий $J = \{ j_1, j_2, ..., j_n \}$, каждое из которых состоит из одной операции. Для каждого задания j известны длительность выполнения l_j и директивный срок выполнения d_j. Задания поступают в систему одновременно в момент времени $a_j = 0, j = \overline{1,n}$. Прерывания не допускаются. Необходимо построить расписание выполнения заданий для одного прибора, минимизирующее суммарное запаздывание при выполнении заданий:

$$f = \sum_{j=1}^{n} \max(0, C_j - d_j),$$

где C_j – момент завершения выполнения задания j.

Предложенный в [1] ПДС-алгоритм обладает следующими свойствами. В процессе исследования теоретических свойств задачи МСЗ:

а) найдены логико-аналитические условия (p-условия), выполнение которых в процессе решения ПДС-алгоритмом произвольной индивидуальной задачи данного класса приводит к получению оптимального решения полиномиальной вычислительной процедурой заранее известной сложности;

б) если в процессе решения индивидуальной задачи выполняются теоретически обоснованные логико-аналитические условия (d-условия), то она разбивается (декомпозируется) на подзадачи меньшей размерности.

В данной статье рассматриваются новые правила отсечений для случая, когда директивные сроки d_j могут принимать как положительные, так и отрицательные значения.

Новые правила отсечений

Введем необходимые для изложения материала определения. Пусть j и j_i обозначает номер работы в соответствии с индексацией, заданной функционалом; $j_{[g]}$ – номер работы, стоящей в допустимом расписании на позиции g.

Определение 1. Резервом времени $r_{j_{[g]}}$ задания $j_{[g]}$ называется величина $r_{j_{[g]}} = d_{j_{[g]}} - C_{j_{[g]}} > 0$.

Определение 2. Перестановкой называется процедура переноса задания $j_{[g]}$ на позицию k ($k > g$) и, одновременно, заданий, занимающих позиции $g + 1$, $g + 2$,..., $k - 1$, k на позиции g, $g + 1$, ..., $k - 2$, $k - 1$, соответственно.

Определение 3. Интервалом перестановки задания $j_{[g]}$ на позицию k в последовательности σ называется интервал, определяемый суммой длительностей заданий, занимающих в этой последовательности позиции g+1, g+2, ..., $k - 1$, k.

Определение 4. Встраиванием называется процедура переноса задания $j_{[g]}$ на позицию p ($g > p$) и, одновременно, заданий p, $p + 1$, ..., $g - 2$, $g - 1$ на позиции $p + 1$, p + 2, ..., $g - 1$, g, соответственно.

Определение 5. Интервалом встраивания задания $j_{[g]}$ на позицию p ($g > p$) в последовательности σ называется интервал, определяемый суммой длительностей заданий, занимаю-

щих в этой последовательности позиции p, p+1,...g–1, где p определяется из условия

$$\sum_{i=p-1}^{g-1} l_{j_{[i]}} < C_{j_{[g]}} - d_{j_{[g]}} \le \sum_{i=p}^{g-1} l_{j_{[i]}} \quad (1)$$

Если же условие (1) не выполняется ни для одной позиции, то p = 1. Таким образом, запаздывание по заданию j на позиции p должно быть равно нулю или минимально.

Определение 6. Задание $j_{[g]}$ называется запаздывающим в последовательности σ, если для него выполняется условие $d_{j_{[g]}} < C_{j_{[g]}}$.

Определение 7. Последовательностью $\sigma^{\text{уп}}$ (сигма упорядоченная) называется последовательность заданий множества J, $j = \overline{1,n}$, в которой задания упорядочены по неубыванию длительностей l_j, т. е. $\forall j, i, j < i : l_j < l_i$, а при $l_j = l_i, d_j \le d_i$.

Определение 8. Процедурой свободной перестановки называется процедура перестановки задания $j_{[k]}$ на позицию q ($k < q$) такую, что $d_{j_{[k]}} \ge C_{j_{[q]}}$, $d_{j_{[k]}} < C_{j_{[q+1]}}$, если хотя бы для одного задания на интервале $\overline{k+1,q}$ выполняется:

$$d_{j_{[i]}} < C_{j_{[i]}}, \quad i = \overline{k+1,q}.$$

Определение 9. Последовательность заданий, полученную в результате выполнения всех свободных перестановок в последовательности $\sigma^{\text{уп}}$, назовем $\sigma^{\text{сп}}$.

Разработанный ПДС-алгоритм построен на перестановках и встраиваниях, направленных на оптимальное использование запаздывающими заданиями резервов времени незапаздывающих заданий.

В следующих утверждениях обосновываются новые свойства последовательности $\sigma^{\text{сп}}$, позволяющие сформулировать новые правила отсечения бесперспективных перестановок.

Утверждение 1. В последовательности $\sigma^{\text{сп}}$ для заданий с отрицательными директивными сроками справедливо: при $i < j : l_i \le l_j$.

Доказательство основано на правилах построения последовательности $\sigma^{\text{сп}}$: задания с отрицательными директивными сроками в перестановках не участвуют.

Пусть уже построена оптимальная последовательность для заданий на интервале [1, $g-1$] и выполняется оптимизация для задания $j_{[g]}$.

Утверждение 2. Если на интервале встраивания задания $j_{[g]}$ с отрицательным знаением директивного срока есть другое задание $j_{[k]}$ с отрицательным значением директивного срока, то при выполнении процедуры встраивания задание $j_{[g]}$ займет позицию k + 1.

Доказательство. Из утверждения 1 следует $l_{j_{[k]}} \le l_{j_{[g]}}$. Возможные моменты начала выполнения заданий $j_{[k]}$ и $j_{[g]}$ равны нулю, и так как задание $j_{[k]}$ на предыдущих шагах алгоритма не заняло более раннюю позицию, чем k, то задание $j_{[g]}$ с большей длительностью не сможет занять более раннюю позицию, чем k+1.

Утверждение 3. Если в последовательности $\sigma^{\text{сп}}$ первые позиции i = [1, k] занимают задания, для которых $r_{j_{[i]}} \le 0$, то последовательность $\sigma^{\text{сп}}$ декомпозируется на две подпоследовательности: σ^1 включает задания от $j_{[1]}$ до $j_{[k]}$, σ^2 – от $j_{[k+1]}$ до $j_{[n]}$. Подпоследовательность σ^1 оптимальна, и задания, принадлежащие этой подпоследовательности, в перестановках не участвуют. Оптимизация выполняется только для заданий подпоследовательности σ^2.

Доказательство. При построении последовательности $\sigma^{\text{сп}}$ задания, которые в результате свободных перестановок заняли более поздние позиции, в соотвествии с утверждением 1.2 из [1] не претендуют на резервы на интервале [1, k]. Для остальных заданий выполняется:

$$l_{j_{[i]}} \le l_{j_{[r]}}, j_{[i]} \in \sigma^1, j_{[r]} \in \sigma^2,$$

и любые перестановки из σ^2 в σ^1 приведут к увеличению суммарного запаздывания (утверждение 1.1 из [1]).

Иллюстративный пример

В последующих таблицах f_j = max(0; $C_j - d_j$). Пусть исходные данные задаются таблицей 1.

Табл. 1. Последовательность $\sigma^{\text{уп}}$

Позиция	№ задания	l_j	d_j	C_j	f_j
1	1	60	1200	60	–
2	2	72	-100	132	232
3	3	98	-300	230	530
4	4	101	484	331	–
5	5	104	200	435	235
6	6	109	383	544	161
7	7	115	590	659	69
8	8	120	720	779	59
9	9	125	-400	904	1304
10	10	128	970	1032	62
11	11	130	-100	1162	1262
					3914

Табл. 2. Последовательность $\sigma^{сп}$

Позиция	№ задания	l_i	d_i	C_i	f_i
1	2	72	-100	72	172
2	3	98	-300	170	470
3	5	104	200	274	74
4	6	109	383	383	0
5	4	101	484	484	0
6	7	115	590	599	9
7	8	120	720	719	0
8	9	125	-400	844	1244
9	10	128	970	972	2
10	11	130	-100	1102	1202
11	1	60	1200	1162	0
					3173

На интервале [1, 5] последовательность $\sigma^{сп}$ оптимальна на основании утверждения 3. Для запаздывающего задания 9 перестановка на более раннюю позицию не реализуется, т.к. существующих резервов заданий 7 и 8 недостаточно. Для задания 10 выполняется: $l_9 < l_{10}$, $d_9 < d_{10}$. Следовательно, задание 10 остается на занимаемой позиции. На интервале встраивания задания 11 находится задание 9 с отрицательным значением директивного срока. Следовательно, согласно утверждению 2, позиция встраивания задания 11 равна 9, но эту позицию занимает запаздывающее задание высшего приоритета, следовательно, задание 11 остается на занимаемой позиции (утверждение 1.1 из [1]). Таким образом, последовательность $\sigma^{сп}$ – оптимальная последовательность.

Исследования эффективности новых отсечений

Для исследования эффективности ПДС-алгоритма с новыми отсечениями была создана система моделирования, написанная на языке C# в среде разработки Visual Studio 2010 под библиотеку Microsoft .NET 4.0. Испытания проводились на персональном компьютере с процессором Pentium CORE 2 Duo 2.0 ГГц с оперативной памятью 2 Гбайта под управлением ОС Microsoft Windows Vista. Исследовались задачи размерности до 500 заданий.

Для определения эффективности новых отсечений были проведены исследования зависимости среднего времени решения задачи в зависимости от размерности. В тестовые задачи были сгенерированы по 100 примеров каждой размерности от 50 до 500 с шагом 50 при значениях фактора запаздывания T и диапазона директивных сроков R [2], соответственно равных 0,4 и 0,8. Проведенное моделирование для исследования эффективности ПДС-алгоритма показало, что добавление новых отсечений значительно уменьшает среднее время решения задач, в зависимости от числа заданий с отрицательными директивными сроками. Результаты моделирования приведены в табл. 3. Проведенные исследования дали возможность показать полиномиальную зависимость среднего времени решения от размерности задач.

Табл. 3. Среднее время решения задач (мс)*

Размерность	Среднее время решения без дополнительных отсечений	Среднее время решения с дополнительными отсечениями
50	50	32
100	321	180
150	440	232
200	218	188
250	371	265
300	409	327
350	568	434
400	584	461
450	758	524
500	408	321

* Число заданий с отрицательными директивными сроками составляет 20% от общего числа заданий.

Выводы

Исследованы дополнительные свойства последовательности $\sigma^{сп}$, сформулированы новые правила отсечений в случае наличия заданий с отрицательными директивными сроками и правило декомпозиции последовательности $\sigma^{сп}$. Введение в алгоритм новых правил позволяет существенно сократить число выполняемых перестановок, что подтверждается результатами статистических исследований.

Перечень литературы

1. Згуровский М.З., Павлов А.А. Принятие решений в сетевых системах с ограниченными ресурсами: Монография.– К.: Наукова думка, – 2010.– 573 с.
2. Fisher, M.L. (1976) A dual algorithm for the one machine scheduling problem. Mathematical Programming, 11, 229-251.

УДК 004.031.2

ЄФРЕМОВ К.В.,
МАЗУР Р.Ф.

СИСТЕМНИЙ ТА СТРУКТУРНИЙ РІВНІ ОРГАНІЗАЦІЇ МУЛЬТИАГЕНТНИХ СИСТЕМ З МОЖЛИВІСТЮ АДАПТАЦІЇ ПРОЦЕСУ ОБРОБКИ ДАНИХ

У статті розглядаються системний рівень подання мультиагентної системи та структурна організація її агентів. Агент представляється у вигляді набору елементів, що будують граф із можливими шляхами обробки вхідних заявок. Запропоновано механізм обробки заявок агентом, який надає можливість адаптації системи до класу задач, що розв'язуються, та динамічної зміни цілей агентів.

The paper deals with system level representation of multi-agent systems and structural organization of its agents. The agent is presented as a set of elements that construct a graph of possible routes of processing incoming requests. What is offered is the way an agent handles requests that allows the system to adapt to the classes of tasks solved and dynamic changes of agent's goals.

Вступ

Прогрес у розвитку засобів обчислювальної техніки забезпечує можливість розв'язувати все більш широкий клас задач за допомогою програмних систем. Це обумовлює необхідність застосування нових підходів до їх проектування та розробки. Один із таких підходів оснований на застосуванні ідеї колективної діяльності сукупності агентів – мультиагентних систем (МАС).

Одним із успішних застосувань МАС стало агентно-орієнтоване моделювання, яке надало можливості створення адекватних моделей у таких сферах, як: біологія, соціологія, машинобудування та ін. Зокрема, представивши у вигляді агентів вірус імунодефіциту людини та лімфоцити, за допомогою МАС у роботі [7] моделювався перебіг захворювання ВІЛ-інфікованих з метою визначення причин значної диференціації проміжків часу між зараженням та проявом СНІД у різних хворих. У роботі [10] за допомогою МАС була змодельована система контролю за рухом повітряного транспорту. Варто відзначити, що в подібних системах поведінка кожного з агентів точно визначається відповідно до об'єктів, які за допомогою них моделюються.

Іншим напрямком розвитку МАС є вироблення концепцій агентно-орієнтованого проектування (АОП) програмних систем може використовуватися у сферах електронної комерції, управління та моніторингу телекомунікаційних систем, оптимізації транспортних систем чи промислового виробництва, аналізу інформації та інформаційних середовищ (наприклад, Інтернет), автоматичного планування зустрічей, електронних розваг чи інтерактивних ігор [9].

За останні роки було запропоновано ряд методологій для розробки мультиагентних систем, таких як TROPOS [1], Prometheus [6], ADELFE [3], Gaia [11], у яких передбачається розгляд нотацій агентів, починаючи з ранніх стадій проектування. Для реалізації подібних проектних рішень можуть використовуватися програмні каркаси, такі як JADE [2]. Варто відзначити, що втілення даних підходів має досить багато спільного із сервісно-орієнтованою архітектурою [5] та багатошаровим програмним забезпеченням.

Постановка задачі

Не зважаючи на прогрес в області розробки методологій АОП, на сьогоднішній день, на жаль, ще не вироблено достатньо підходів для реалізації таких властивостей агентів, які забезпечують адаптивність системи. Тому метою даної роботи стала розробка моделі організації МАС з адаптивними агентами, використання якої дає можливість оптимізувати процес обробки даних в системі при динамічній зміні класу задач, що розв'язуються, та цілей агента.

Визначення мультиагентної системи

Наразі дослідники не дійшли згоди стосовно однозначних визначень агента та мультиагентної системи, однак більшість із них погоджуються зі ствердженнями, наведеними в [12]. Зокрема, агентом є комп'ютерна система, яка розміщена в певному середовищі та здатна виконувати автономні дії з метою досягнення поставлених під час проектування цілей.

Система називається мультиагентною, якщо вона складається з набору автономних агентів, що можуть взаємодіяти між собою [4]. Такі системи використовуються для розв'язання задач, рішення яких є неможливим або занадто складним при використанні одного агента чи монолітної системи. Важливою особливістю МАС є те, що її глобальна функція не задається чи проектується, вона виникає на основі взаємодії агентів між собою в наслідок синергетичного ефекту.

У роботі [12] було виділено набір таких властивостей: автономність, соціальність, реактивність та проактивність, якими в тій чи іншій мірі повинен володіти агент. Останні дві властивості тісно пов'язані з адаптивністю, тобто такою властивістю системи, яка полягає в тому, що вона може виконувати поставлені задачі у кілька способів та має достатні знання про свою будову для здійснення змін у процесі роботи.

З метою оптимізації витрат на досягнення цілей агенти мають включати в себе засоби для оцінки власної поведінки та продуктивності разом із здатністю до перепланування власних дій з метою підвищення ефективності своїх операцій [8].

Системний рівень організації МАС

Як уже було сказано, складовими частинами мультиагентної системи є множина агентів та їхнє середовище. Оскільки така система не має інших складових, необхідний зв'язок з іншими системами здійснюється через агентів. Кожен із агентів здатен аналізувати стан середовища, у якому знаходиться, та генерувати певний вплив на це середовище. Ці вхідні та вихідні дані агента *A* мають свої алфавіти (*X* та *Y* відповідно), а сам агент може розглядатися як перетворення над даними в цих алфавітах $A: X \to Y$.

Дані на вході агента можуть містити інформацію про вхідні аргументи його перетворення з алфавітом X_D та про директиви контролю цього перетворення з боку середовища з алфавітом X_C. Тобто, $X = X_D \times X_C$.

Насправді, вхідний алфавіт агента характеризує дані, у яких він може бути зацікавлений. Вихідний – дані, які він може надати системі. Це означає, що факт появи даних на вході агента ніяким чином не гарантує появу даних на його виході. Однак, якщо агент має вихідний алфавіт, то вважається що в процесі роботи системи він обов'язково спродукує дані (варто відзначити, що тривалість роботи системи ніяк не обмежується), і в контексті всієї тривалості роботи системи агент здійснюватиме перетворення. Справедливе і протилежне твердження: якщо агент не надає відомостей про свій вихідний алфавіт, то він здатен лише отримувати вхідні дані. Для таких агентів вихідний алфавіт рівний $O = \{\theta\}$, тобто пустим даним. Таким чином, можливі три типи агентів:

1. агенти-перетворення (з повними вхідним та вихідним алфавітом), $A: X \to Y$;
2. агенти-споживачі (з пустим алфавітом вихідних даних) , $A: X \to O$;
3. агенти-генератори (з пустим алфавітом вхідних аргументів), $A: O \times X_C \to Y$.

Існування агентів останніх двох типів обґрунтовується необхідністю зв'язку з іншими зовнішніми системами. Далі розглядатимуться агенти першого типу.

Поставимо умову, що сенсори агентів отримують від середовища дані, представлені кортежем $R = \langle X', Y', x \rangle$, який несе інформацію про заявку на виконання перетворення $X' \to Y'$ над аргументами *x*. При цьому, оскільки агент є автономним, він здатен самостійно зробити запит типу *R* до середовища.

Щоб називатися адаптивною та автономною, система повинна володіти механізмом самостійної оцінки власного стану та винесення рішення про необхідність змін у роботі. Для здійснення такої оцінки у системі існує *p* параметрів роботи агента. У такому випадку стан агента може описуватися вектором $\bar{s} \in \mathrm{R}^p$, який містить значення кожного з параметрів.

Стуктурний рівень організації агента

Нехай агент *A* визначається набором елементів $a_i, i = \overline{1,n}$, кожен з яких може виконувати перетворення f_{a_i} над підміножиною вхідного алфавіту агента X^i. При цьому результатом такого перетворення є підмножина вихідного алфавіту агента Y^i. Тобто,

$$A = \left\{ a_i \middle| f_{a_i} : X^i \to Y^i \right\}, i = \overline{1,n},$$
$$X = \bigcup_i X^i, Y = \bigcup_i Y^i.$$

Семантика кожного такого перетворення визначається парою вхідного та вихідного алфавітів та символом перетворення $\left(X^i, Y^i, f_{a_i}\right)$. Дані

про елементи, а саме про їхній вхідний та вихідний алфавіт, заносяться до внутрішнього реєстру агента.

Вважатимемо, що набір елементів в агенті може змінюватися завдяки реконфігурації. При цьому поява нового елемента може означати для агента або розширення його функціональності, або появу альтернативних засобів обробки даних.

Знання про внутрішню структуру агента можуть бути представлені у вигляді орієнтованого графа G з множиною вершин V та множиною дуг E. Множина перетворень, які здійснюють елементи агента, є підмножиною прямого добутку його вхідного та вихідного алфавітів:

$$A \subseteq X \times Y = \bigcup_i X^i \times \bigcup_i Y^i .$$

Дана множина елементів агента складає множину вершин графа G, тобто $A=V$. Множина дуг є підмножиною усіх відношень на множині A та визначається як

$$E = \left\{(j,k) : Y^j \subseteq X^k\right\};$$
$$j = \overline{1,n};\ k = \overline{1,n}.$$

При появі на вході агента заявки $R = \langle X', Y', x \rangle$, він може відреагувати на неї, виконавши відповідне перетворення. При цьому необхідною умовою обробки є

$$\begin{cases} X' \subseteq X, \\ Y' \subseteq Y; \end{cases} \Rightarrow \exists \left\{j \middle| j \in [1,n]\right\}, \left\{k \middle| k \in [1,n]\right\}:$$
$$X' = \bigcup_{\{j\}} X^j \cap Y' = \bigcup_{\{k\}} Y^k.$$

Виконання даної умови означає, що існують такі елементи агента, вхідний та вихідний алфавіти яких відповідають заявці, що надійшла.

Достатньою умовою виконання заявки агентом є існування непустої множини шляхів $\{P_{j,k}\}$ між відповідними вершинами j та k:

$$P_{j,k} = \{p_{j.k}\} = \left\{\langle a_j, ..., a_k \rangle\right\}.$$

При цьому вважатимемо, що передача даних по одній із дуг графа G пов'язана зі зміною стану агента ($\Delta\bar{s}$). Тобто існує множина функцій $\{\varphi : E \to \mathrm{R}^p\}$, які обчислюють вартість кожного з переходів у графі для агента.

Таким чином, структуру S агента можна описати через множину його елементів-перетворень A, вектор стану $\bar{s}$ та множину функцій, які визначають вартість кожного з переходів по графу G:

$$S = \left\langle A, \bar{s}, \{\varphi\} \right\rangle.$$

Граф G може мати декілька маршрутів між будь-якими парами вершин j та k, що дає можливість зміни шляху обробки заявки в залежності від вхідних даних x, поточного набору елементів $A = \{a_i\}$, та значень параметрів стану агента $\bar{s}$. При цьому потрібно розв'язувати задачу оптимізації.

Процес обробки заявки в агенті

Із появою заявки R на вході агента у випадку, коли вона може бути оброблена, в агенті починається просування пакетів з даними. Даний пакет C у момент часу τ є четвіркою, яка складається з унікального ідентифікатору поточного запиту $u \in \mathrm{N}$, алфавіту даних, які обробляються M, поточних даних d для обробки елементом агента та шляху P, який даний пакет пройшов в агенті (упорядкована підмножина елементів останнього):

$$C^\tau = \left\langle u, M^\tau, d^\tau, P^\tau \right\rangle,\ P^\tau \subseteq A.$$

При $\tau = 0$ маємо $M^\tau = X', d^\tau = x, P^\tau = \{\ \}$.

Після обробки пакета елементом a_t $(t \in [1,n])$ агента, частини M та d змінюються відповідно до вихідного алфавіту поточного елемента та результатів здійсненого перетворення:

$$\begin{cases} M^{\tau+1} = Y^t, \\ d^{\tau+1} = f_{a_t}(d^\tau), \\ P^{\tau+1} = P^\tau \cup a_t. \end{cases}$$

Далі відбувається передача пакета $C^{\tau+1}$ наступним елементам. На агента покладається задача визначення, кому з елементів необхідно передати заданий пакет. Останній може бути доставлений лише тому елементу, який має відповідний вхідний алфавіт. Отже, при прийнятті рішення про пункт призначення пакета агент формує вибірку елементів із множини доступних (тих, які можуть обробити поточні дані), керуючись результатами аналізу поточного стану системи. Така вибірка розміром з m елементів може бути описана наступним чином:

$$\left\{a_i \middle| M^\tau \subseteq X^i\right\} \subseteq A, i = \overline{1,m}.$$

Слід вказати, що ситуація з декількома маршрутами можлива, коли агент має в своєму складі елементи, чиї вхідні алфавіти перетинаються:

$$m>1 \Rightarrow \exists\, j,k \in [1,n]: X^j \cap X^k \neq \{\ \}.$$

Зміни у роботі системи можуть бути здійснені шляхом модифікації рішень стосовно доставки пакета через граф *G* агента.

Агент повинен вибрати маршрут з найменшою вартістю для просування пакету. Тоді, коли такий маршрут невідомий, виконується широкомовна передача пакета елементам, які можуть обробити дані, що передаються всередині. Із усіх можливих маршрутів, утворених таким чином, вибирають ті, які дають мінімальний час проходження пакета. Такі маршрути запам'ятовуються та використовуються для подальших обробок відповідних заявок.

Виходячи зі свого поточного стану та набору функцій φ, агент може прийняти рішення про «небажаність» проходження пакету по певній дузі графа *G* або маршруту. У такому випадку агент може уповільнювати проходження даних по вибраній дузі і корегувати таким чином вибір маршруту.

Отже, агент отримує можливість зміни способів обробки вхідних заявок у залежності від власних цілей, які визначаються функцією вартості φ.

Така реконфігурація маршруту здійнюється у випадку появи в структурі агента елемента, який може входити до альтернативного маршруту, або зміни вартості маршрутів обробки заявки.

Висновки

У роботі на системному рівні подання МАС виділені три типи агентів, запропоновані їхня структурна організація та процес обробки заявок в них, які створюють можливість динамічної структурної реорганізації агентів з метою оптимізації процесу роботи МАС.

Перелік посилань

1. Bresciani P. Tropos: An Agent-Oriented Software Develpment Methodology / Bresciani P., Giorgini P., Giunchiglia F. – Trento : Kluwer Academic Publishers, 2003.
2. Caire F. JADE: A software framework for developing multi-agent applications. Lessons learned / Caire F., Poggi A., Rimassa G. // Information and Software Technology. – 2008. – 50. – pp. 10-21.
3. Carole B. ADELFE, a Methodology for Adaptive Multi-Agent Systems Engineering : ESAW'02 Proceedings of the 3rd international conference on Engineering societies in the agents world III / Carole B., Sylvain P., Gauthier P. - Berlin : Springer-Verlag, 2003.
4. D'Inverno M. Understanding agent systems / D'Inverno M., Luck M., Luck M. M. – Springer, 2004. – 2 : p. 240.
5. Erl T. Service-oriented architecture: concepts, technology, and design. – Prentice Hall Professional Technical Reference, 2005.
6. Padgham L. The Prometheus Methodology / Padgham L., Winikoff M. // Engineering Applications of Artificial Intelligence. –2004. – 18(2).
7. Perrin D. An agent-based approach to immune modelling / Perrin D., Ruskin H. J., Burns J., Crane M. // Lecture Notes in Computer Science. – Springer, 2006. – 3980. – pp. 612-621.
8. Robertson R. Self-adaptive software: applications: Second International Workshop, IWSAS 2001 (Balatonfűred, Hungary, May 17-19, 2001) / Robertson R., Shrobe H. – Springer, 2003. – Vol. 2.
9. Russell S. J. Artificial Intelligence: A Modern Approach / Russell S. J., Norvig P. – Prentice Hall, 2009. – 3.
10. Wolfe S. Comparing Route Selection Strategies in Collaborative Traffic Flow Management: IEEE/WIC/ACM International Conference on IAT / Wolfe S., Jarvis P., Enomoto F., Sierhuis M. – 2007.
11. Wooldbridge M. The Gaia Methodology for Agent-Oriented Analysis and Design / Wooldbridge M., Jennings N. R., Kinny D. // Autonomous Agents and Multi-Agent Systems. – Kluwer Academic Publishers, 2000. – 3. – pp. 285-312.
12. Wooldridge M. Intelligent agents: theory and practice / Wooldridge M., Jennings N. R. // The Knowledge Engineering Review. – 1995. – 10(2). – pp. 115-152.

УДК 683.519

СИМОНЕНКО В.П.,
КУРЕНЁВ А.С.

АЛГОРИТМ СТАТИЧЕСКОГО ПЛАНИРОВАНИЯ ДЛЯ GRID СИСТЕМ

В данной статье описан алгоритм статического планирования для многопроцессорных и Grid систем. Задача планирования сводится к поиску плана распределения задач по ресурсам с минимальным количеством вычислительных узлов с минимизацией времени решения. В этой статье предложен алгоритм пошагового конструирования, позволяющий уменьшить время планирования, за счет применения принципа одновременного прохода планировщика «сверху» и «снизу».

This article describes new static scheduling algorithm for multiprocessor and Grid systems. The main goal of scheduling is to find optimal distribution of tasks between resources with a minimum number of computing nodes while reducing processing time. In this article we propose step-by-step algorithm, which allows reducing scheduling time, by using the principle of simultaneous “forward” and “backward” scheduler’spassage.

1. Введение

В настоящее время ведутся активные исследования в создания инфраструктуры, которая обеспечивает надежный, устойчивый и недорогой доступ к высокопроизводительным вычислительным ресурсам. Такой инфраструктурой являются Grid-системы. Так же в данный момент в Украине эффективно развивается Украинский Академический Grid. Одной из актуальных задач в подобных системах является такое планирование заданий, которое позволяет наиболее эффективно использовать ресурсы вычислительной среды. В данном случае под планированием понимается распределение поступающих на вход заданий, подготовленных для параллельного выполнения на имеющиеся ресурсы. Целью планирования является определение эффективности распределения заданий при использовании различных критериев.

2. Обзор существующих решений

На данный момент разработано множество различных алгоритмов статического планирования [1], однако всем им присущ существенный недостаток, они требуют выполнять полное сканирование графа, причем зачастую несколько раз, что существенно замедляет работу самого планировщика. Большинство планировщиков для Grid систем, таких как, FCFS, SJF [2] просто выделяют определённое количество ресурсов под задачи, не выполняя оптимизацию внутри самих задач, что бы дало не только предпосылки к уменьшению времени самих выполнения задач, но и к более эффективному использованию ресурсов.[3]

3. Цель

Целью данной статьи является разработка нового алгоритма пошагового конструирования, позволяющего уменьшить время планирования, за счет применения принципа одновременного прохода планировщика «сверху» и «снизу».

4. Постановка задачи

Исходная задача поступает в систему в виде ориентированного ациклического графа (DAG) [4] задачи в ярусно-параллельной форме. Граф задается в виде множества:

$$G = \{N; E; W; C\},$$

где N – множество вершин графа; E – множество дуг (переходов между вершинами); W – веса (вычислительная сложность) вершин; C – веса (время коммуникации) дуг.

Для определения порядка запуска задач используется представление графа в виде матрицы связности: значение элемента матрицы указывает вес пути из одной вершины в другую, если такой существует. Эти значения также указывают зависимость одной задачи от другой.

При планировании необходимо учитывать условие предшествования. Задача готова к выполнению, когда все зависимости разрешены.

Для первой разрешённой задачи время запуска считается 0. Для остальных выбирается максимальное из всех определённых при разрешении зависимостей, таким образом, ко времени запуска задачи все её предшественники

будут выполнены, и время на пересылку данных учтено.

Таким образом, основной задачей планирования [5] является построение распределение (план) задач по доступным ресурсам так, чтобы минимизировать время выполнения приложения и использование ресурсов.

5. Описание алгоритма

На первом шаге алгоритма строится базовое решение. Под базовым решением понимается выделение отдельного процессора для каждой задачи. Такое решение используется в качестве основного, от которого производится поиск лучшего решения.

Основными атрибутами, по которым осуществляется планирование для DAG, являются t-level и b-level [1,6].

t-level – это самый длинный путь от начальной вершины до данной вершины n_i , без учёта веса n_i. Путь считается путём сложения всех весов вершин и пересылок, через которые он проходит. Этот параметр так же отражает наименьшее время начала вершины.T-level для i-й вершины вычисляется по следующей формуле:

$$tlevel\ (n_i) = max\ (tlevel(n_m) + w_m + c_{m,i}),$$

где$n_m = pred(n_i)$ – предшествующие вершины, w_m вычислительная сложность, $c_{m,i}$ – временные затраты на коммуникацию.

b-level- это самый длинный путь от вершины n_iдо исходной. Так же выделяют статический b-level, в котором не учитываются пересылки. B-level для i-го узла вычисляется по формуле:

$$blevel(n_i) = w_i + max\ (blevel(n_m) + c_{m,i}),$$

где $n_m = succ(n_i)$ – последующие вершины, w_i- вычислительная сложность, $c_{m,i}$ – временные затраты на коммуникацию. А статический b-level вычисляется по формуле:

$$sblevel(n_i) = w_i + max(sblevel(n_m)),$$

где $n_m = succ(n_i)$ – последующие вершины, w_i- вычислительная сложность.

Стоит отметить, что данные необходимо вычислять на каждом шаге алгоритма только для тех вершин, которые находятся на последующем уровне, при нисходящем планировании, и тех, которые находятся уровнем выше при восходящем. Это позволяет исключить повторное полное сканирование графа, что значительно ускоряет процесс планирования.

Обозначим множество вершин, принадлежащих k-му уровню:

$$n^k = \{n_i^k\}, i = \overline{1, m_k},$$

где n_i^k – i-я вершина k-го уровня, m_k – количество вершин на k-ом уровне.

Если вершина n_i^k имеет единственную инцидентную ей дугу e, и вершина n_j^{k-1} так же имеет единственную дугу e инцидентную ей, то вершины n_i^k и n_j^{k-1} необходимо кластеризовать.

При нисходящем планировании на каждом шаге алгоритма выполняются следующие действия:

- Вычисляется t-level для каждой вершины n_j^{k+1}, находящейся на следующем уровне.
- Рассматривается множество задач n^k находящихся на одном (k) уровне. Для каждой задачи k-го уровня выбирается множество задач $n_j^{k+1}, j = \overline{1, m_{k+1}}$, находящихся k+1 уровне, имеющих дугу, инцидентную рассматриваемой вершине n_i^k, т.е. вершина n_j^{k+1} имеет информационную зависимость от вершины N_i^k.
- Для кластеризации с N_i^k выбирается та вершина n_j^{k+1}, которая имеет наибольший t-level:max($tlevel\ (n_j^{k+1})$). Остальные вершины, имеющие информационную зависимость отрассматриваемой остаются без изменений, однако в последствии могут быть кластеризованы с другими вершинами k-го уровня.

При восходящем планировании на каждом шаге алгоритма выполняются следующие действия:

- Вычисляется b-level для каждой вершины n_j^{k-1}, находящейся на предыдущем уровне.
- Рассматривается множество задач n^k находящихся на одном (k) уровне. Для каждой задачи k-го уровня выбирается множество задач $n_j^{k-1}, j = \overline{1, m_{k+1}}$, находящихся k-1 уровне, имеющих дугу, инцидентную рассматриваемой вершине n_i^k, т.е. вершина N_i^k имеет информационную зависимость от вершины n_j^{k-1}.
- Для кластеризации с N_i^k выбирается та вершина n_j^{k+1}, которая имеет наибольший b-level: max($blevel\ (n_j^{k-1})$).

Для кластеризованных вершин, находящихся в одном узле, время затрачиваемое на коммуникацию приравниваем к 0.

Так же при планировании стоит учитывать вершины, обладающие свойствами неявной транзитности, т. е.для n_i^k вершины выполняется неравенство:

$$tlevel\left(n_i^k\right) \geq \min(tlevel\left(n_j^{k+1}\right)).$$

Такие вершины при планировании могут быть кластеризованы с вершинами, находящимися на более низком уровне, чем k+1. Зачастую это может привести к уменьшению ресурсов, требуемых для текущей задачи. Такую вершину целесообразно перенести на другой уровень для кластеризации, если выполняется соотношение:

$$ntlevel\left(n_i^k\right) \leq \max(tlevel\left(n_j^k\right)),$$

где $ntlevel$ –$tlevel$ для нового уровня.

Выполним планирование при помощи данного алгоритма для графа, представленного на рис. 1.

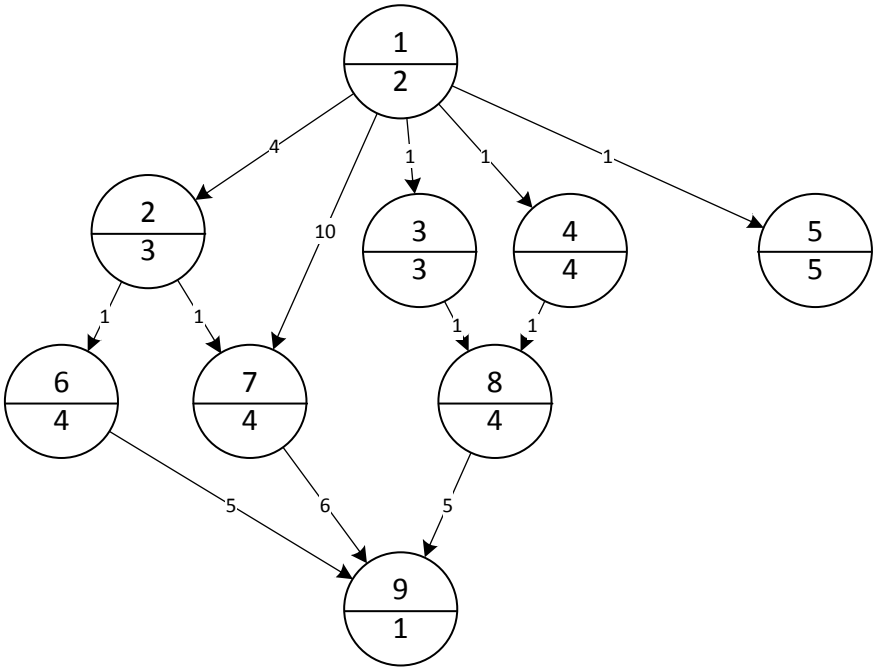


Рис.1. Исходный граф

Выполним построение базового решения, выделив каждой задаче по отдельному ресурсу R (рис. 2).

Распределим вершины по уровням:

$$n^1 = \{1\}$$
$$n^2 = \{2,3,4,5\}$$
$$n^3 = \{6,7,8\}$$
$$n^4 = \{9\}$$

Стоит отметить, что вершина 5 обладает свойством транзитности, поэтому она может находиться как на втором уровне, так и на третьем и четвертом.

Рассмотрим для вершины 1 (рис.3):

Из всех вершин второго уровня наибольший t-level имеет вершина 2:

$$tlevel(n_2) = 1 + 4 = 5$$

Поэтому вершину 2 кластеризуем с вершиной 1.

Теперь рассмотрим шаг для восходящего планирования (рис. 4).

Вершина 8, находящаяся на третьем уровне имеет наибольший b-level, поэтому она кластеризуется с 9 вершиной. Для вершин, которые находятся на одном ресурсе время, затрачиваемое на коммуникацию, приравниваем к нулю. В результате в начале следующего шага диаграмма будет выглядеть, как представлено на рис. 5.

Переходим на следующий уровень и для каждой вершины выполняем аналогичные действия. После выполнения шага 2 при нисходящем планировании получим диаграмму, представленную на рис. 6., в начале следующего шага диаграмма будет выглядеть, как представлено на рис. 5.

Переходим на следующий уровень и для каждой вершины выполняем аналогичные действия. После выполнения шага 2 при нисходящем планировании получим диаграмму, представленную на рис. 6.

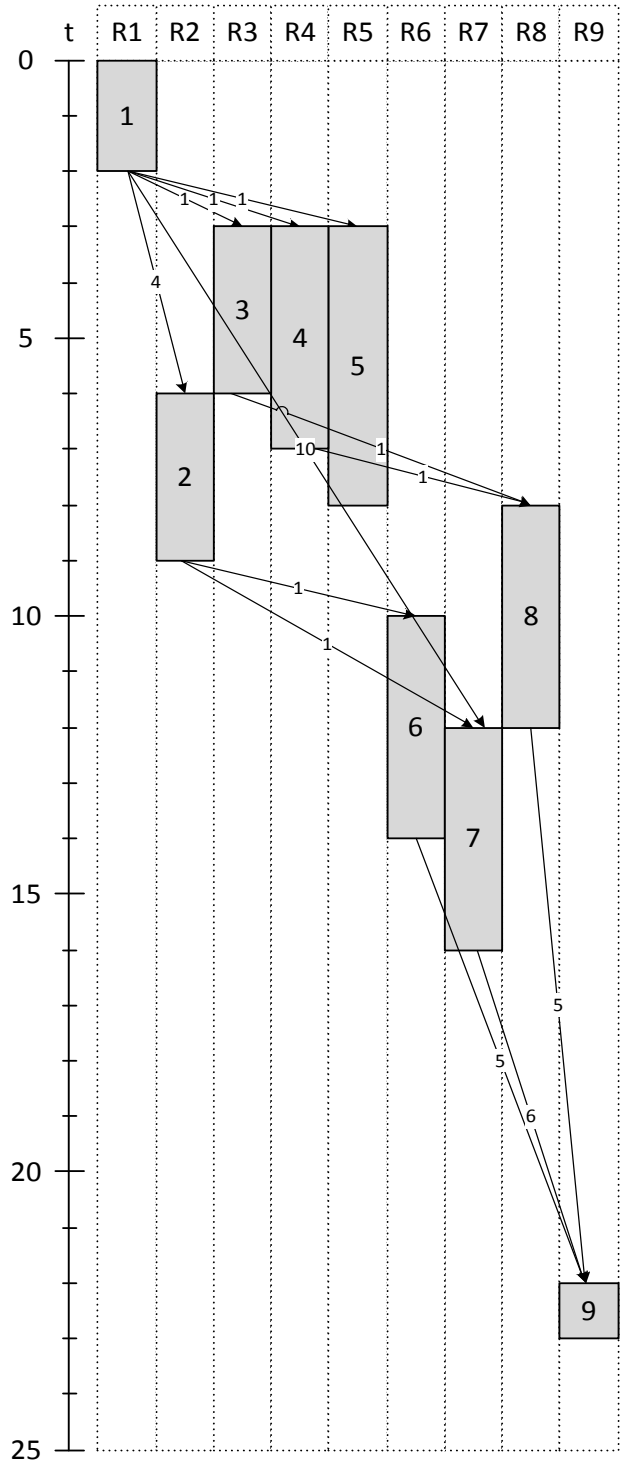


Рис. 2. Базовое решение

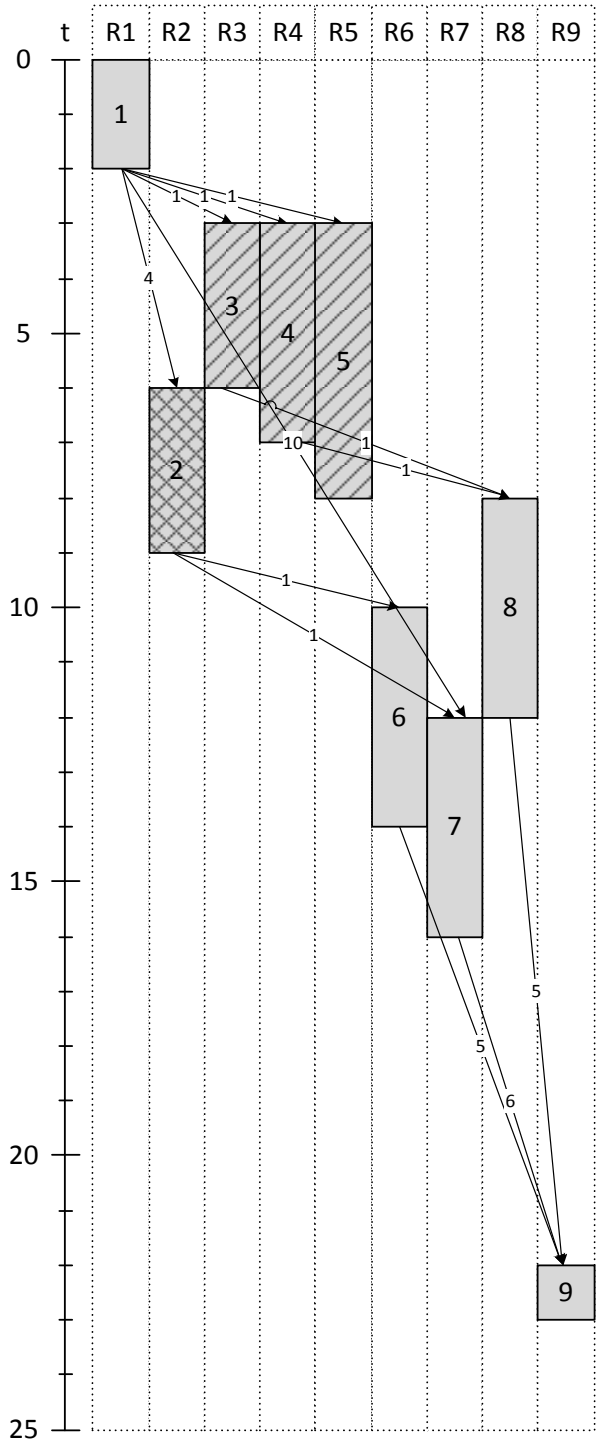


Рис. 3. Диаграмма Ганта. Шаг 1 (нисходящее планирование)

При выполнении шага для восходящего планирования берем во внимание свойство транзитности вершины 5. Т.к. Её можно кластеризовать с вершиной 1, то проверим, выполняется ли условие целесообразности данного действия:

$$ntlevel\ (n_5^4) \le tlevel\ (n_9^4),$$

$$ntlevel\ (n_5^4) = 9;\ tlevel\ (n_9^4) = 15,$$

Т.к. $9 < 15$, то вершину 5 стоит кластеризовать с вершиной 1.

Результирующая диаграмма Ганта представлена на рис. 7.

6. Результаты

В результате время выполнения, по сравнению с базовым решением сократилось на 30%. Для данного примера, описанный в данной статье алгоритм, показывает очень близкий к наилучшему результат, в сравнении с другими су-

ществующими алгоритмами [3], не только по критерию минимизации процессорного времени, затраченного навыполнения задачи, но и по количеству необходимых шагов (итераций) для осуществления планирования.

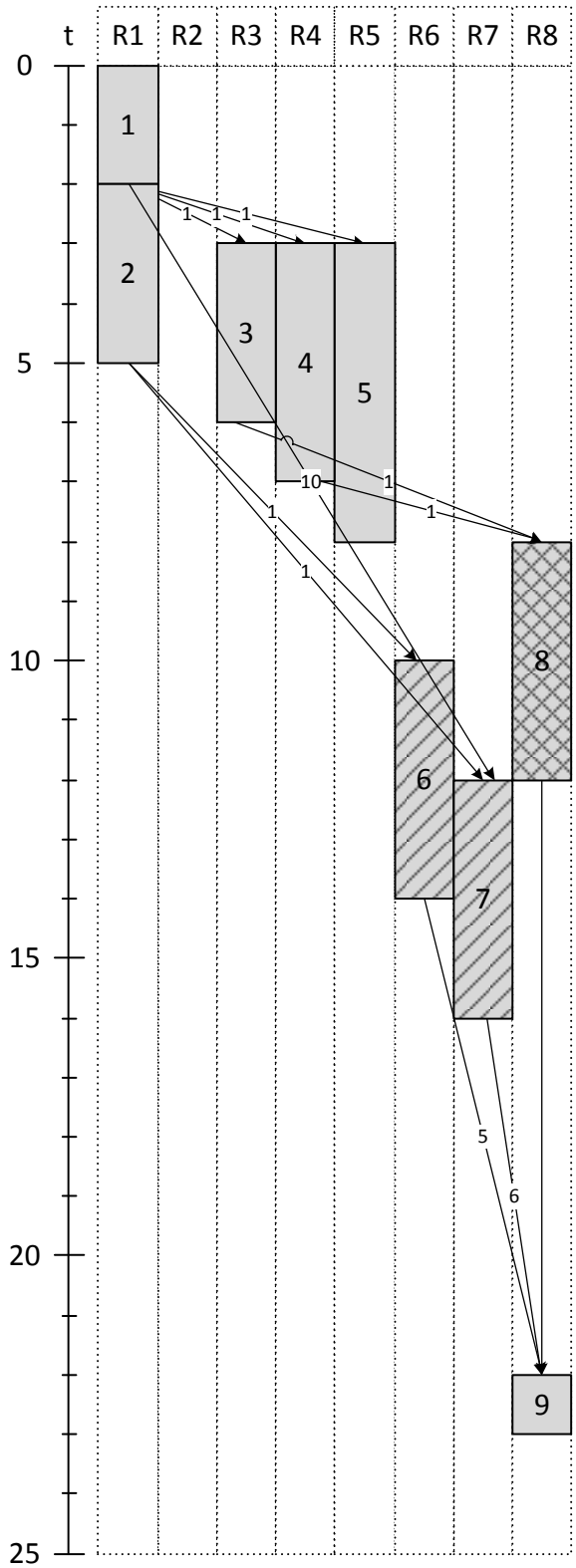


Рис. 4. Диаграмма Ганта. Шаг 1 с (восходящее планирование)

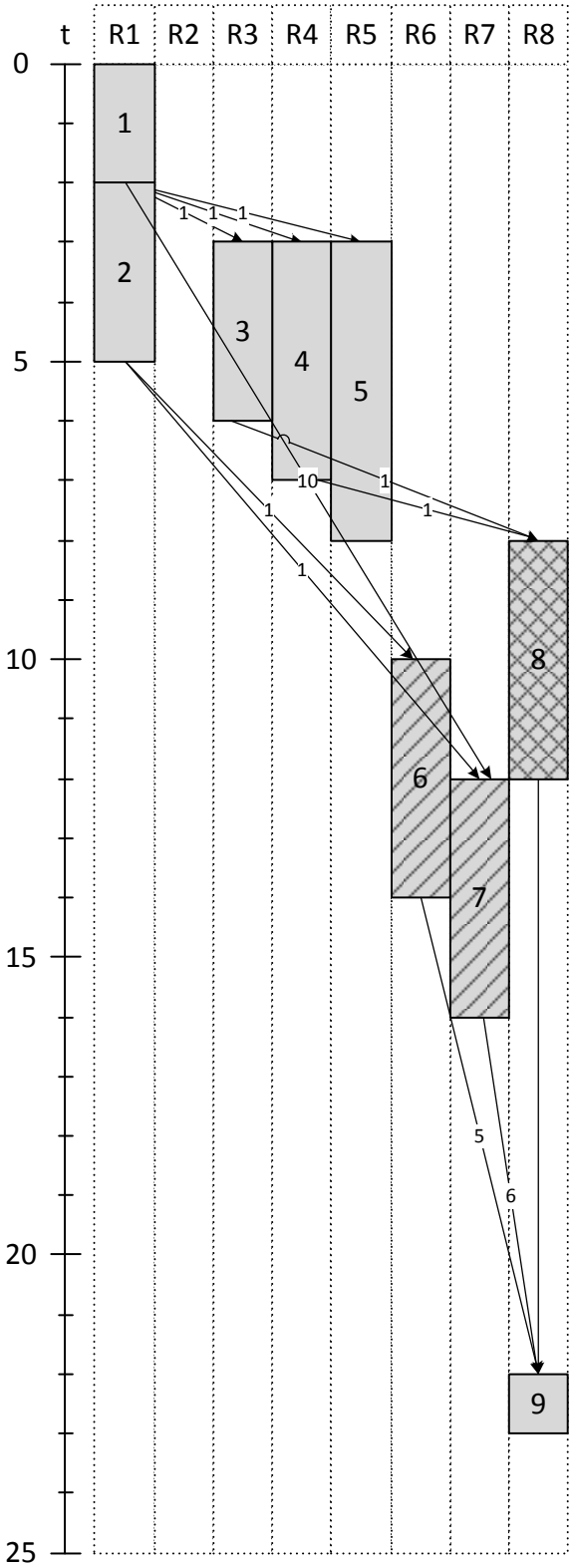


Рис. 5. Диаграмма Ганта. Конец шага 1

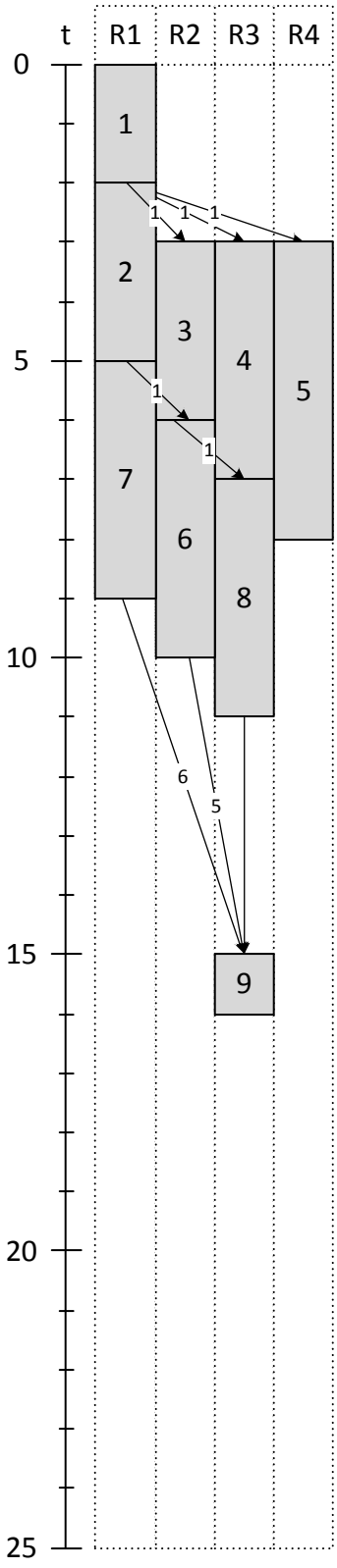


Рис. 6. Диаграмма Ганта. Шаг 2 (нисходящее планирование)

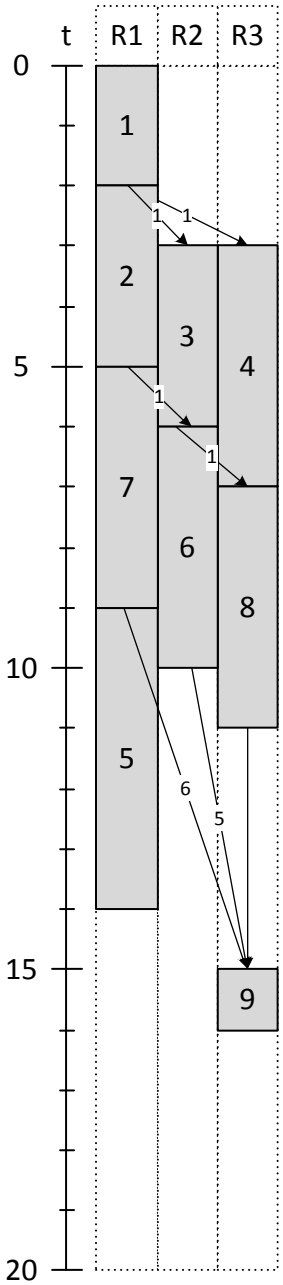


Рис. 7. Диаграмма Ганта. Результат

7. Дальнейшие перспективы

Для достижения лучших результатов планируется использовать в данном алгоритме методы, позволяющие оптимизировать планирование за счет дублирования вычислений на различных ресурсах, что даст предпосылки к уменьшению временных затрат на коммуникации, что в свою очередь может привести к сокращению времени выполнения всей задачи.

Список литературы

1. Y.-K. Kwok, I. Ahmad Static Scheduling Algorithms for Allocating Directed Task Graphs //ACM Computing Surveys. – 1999. – №4. – С. 406-471.
2. Коваленко В.Н. Управление параллельными заданиями в гриде с неотчуждаемыми ресурсами / В.Н. Коваленко, Е.И. Коваленко, Д.А. Корягин, Д.А Семячкин // Препринт No63. – М.: ИПМ РАН. – 2007. – С. 1-28.
3. М.А. Волк, Т.В. Филимончук, Р.Н. Гридель Методы распределения ресурсов для GRID-систем // ЗбірникнауковихпрацьХарківськогоуніверситетуПовітряних Сил. – 2009. – №19. – С. 100-104.
4. Ishfaq Ahmad and Min-You Wu, “Performance Comparison of Algorithms for Stat icScheduling of DAG to Multiprocessors”, [Електроннийресурс], режим доступу до журналу: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.42.8979&rep=rep1&type=pdf.
5. Shiyuan Jin, Guy Schiavone, DamlaTurgut , “A Performance Study of Multiprocessor Task Scheduling Algorithms” 43, C. 77-97, Jan 2008.
6. ParneetKaur, Dheerendra Singh, Gurvinder Singh &Navneet Singh, “Analysis, comparison and performance evaluation of BNP scheduling algorithms in parallel processing”. – International Journal of Information Technology and Knowledge Management // January-June 2011, Volume 4, No. 1, C. 279-284.

УДК 004.052.42

МАРКОВСКИЙ А.П.,
АБУ УСБАХ А.Н.,
АЛЬМУРАДИ В.М.

ОРГАНИЗАЦИЯ ИДЕНТИФИКАЦИИ АБОНЕНТОВ МНОГОПОЛЬЗОВАТЕЛЬСКИХ СИСТЕМ С ИСПОЛЬЗОВАНИЕМ АППАРАТНО ЗАЩИЩЕННОЙ ПАМЯТИ

Разработана новая двухуровневая организация идентификации удаленных пользователей, которая использует один цикл передачи между пользователями и системой. Предложенная схема не использует списка паролей зарегистрированных пользователей и операций поиска и соответственно не накладывает ограничений на число пользователей. Объем информации, используемой при идентификации в предложенной схеме существенно меньше в сравнении с известными схемами. Это позволяет использовать аппаратно защищенную энергонезависимую память для хранения идентификационной информации и таким путем уменьшить риск незаконного доступа к ресурсам системы.

The new two-level organization of identification of remote abonent, which requires one communication between users and verifiers in system, has been proposed. Presented organization does not imply the password list of the legal users and searching operations and correspondingly does not impose restrictions on the number of users. The information capacity to be stored in secret is much less in the proposed identification scheme as compared to the known ones. This allows using hardware-based secure active low-capacity nonvolative memory for storing identification information and such way impair the risk of illegal access to system resources.

Введение

Развитие интегрированных систем обработки информации в значительной степени зависит от эффективности реализации в них функций защиты информации и разделения прав доступа к данным. Важное место в арсенале средств защиты интегрированных информационных и вычислительных ресурсов от несанкционированного доступа играет идентификация абонентов многопользовательских систем.

Расширение использования интегрированных систем хранения и обработки информации сопряжено с увеличением риска несанкционированного доступа к их ресурсам. Это обусловлено, с одной стороны, ростом технических возможностей для реализации несанкционированного доступа, а с другой – увеличением потенциальных выгод от такого доступа.

В этих условиях необходимо адекватное совершенствование всего арсенала средств, исключающих несанкционированный доступ к информационным и вычислительным ресурсам, в том числе и средств идентификации абонентов многопользовательских систем. Особую остроту в современных условиях приобретает проблема защиты от несанкционированного доступа к интегрированным системам компьютеризированного управления сложными техническими объектами.

Таким образом, проблема повышения эффективности идентификации удаленных абонентов многопользовательских систем является актуальной и важной для современного этапа развития информационных технологий.

Анализ проблемы эффективности идентификации абонентов

Эффективности идентификации удаленных абонентов пользователей определяется двумя факторами: устойчивостью к попыткам незаконного доступа (измеряется объемом затрат ресурсов, требующимися для такого доступа) и объем ресурсов, затрачиваемых для идентификации. Очевидно, что указанные факторы являются взаимосвязанными: чем выше уровень надежности идентификации абонента, тем сложнее ее процедура и тем больше ресурсов требуется для реализации процесса идентификации. С другой стороны, многопользовательские системы являются системами массового обслуживания и, соответственно, должны обладать производительностью, обеспечивающей возможность обработки запросов большого числа абонентов без существенных задержек. Фактически зависимость между рассматриваемыми факторами эффективности носит более сложный характер, поскольку результативность ряда способов незаконного доступа к ресурсам прямо зависит от времени

идентификации абонентов [2]. Следовательно, высокая эффективность идентификации абонентов может быть достигнута только в рамках разрешения компромисса между надежностью и скоростью идентификации.

В основе большинства существующих систем идентификации удаленного абонента лежит концепция доказательства знания им какой-то информации. Соответственно, схема идентификации включает два этапа: регистрацию абонента, во время которой абонент обретает идентифицирующую информацию и собственно идентификацию, в рамках которой осуществляется проверка легальности доступа.

К настоящему времени созданы и активно используется большое число протоколов идентификации удаленных абонентов [1,2]. Обычно выделяют два базовых подхода к идентификации: с использованием паролей и на основе концепции нулевого знания. Идентификация на основе концепции нулевого знания считается строгой, однако существующие методы ее реализации требуют больших вычислительных ресурсов [2]. Именно поэтому, идентификация на основе паролей широко используется во многих системах.

Для получения несанкционированного доступа, нарушитель может выполнить чтение, перехват и подмену информации, используемой в процессе идентификации при ее передаче по открытых линиях. Кроме того, потенциальную опасность представляет возможность доступа к информации, используемой для предоставления прав доступа со стороны самой системы [2].

Одним из наиболее потенциально опасных факторов риска для многопользовательских систем является возможность получения несанкционированного доступа к их ресурсам посредством чтения или изменения информации, используемой при идентификации легальных абонентов. Доступ к такой информации может быть осуществлен различными способами: с использованием специальных компьютерных вирусов, при “удачном” случайном входе в систему нелегального пользователя, а также легальным абонентом, пытающимся расширить свои права или предоставить возможность доступа к ресурсам системы нелегальным пользователям, и, наконец, недобросовестным лицом из персонала системы. Во всех перечисленных случаях доступ к информации, используемой при идентификации, осуществляется программным путем.

Существенным недостатком большинства известных схем идентификации абонентов является малая производительность, связанная с выполнением нескольких сеансов обмена информацией, а также необходимостью операций поиска по ключу. Невысокая скорость идентификации существенно ограничивает количество абонентов и не позволяет реализовать повторные сеансы идентификации непосредственно в процессе информационного обмена с тем, чтобы воспрепятствовать технологии доступа к ресурсам путем ”подмены легального абонента”.

Наиболее надежным способом сведения к минимуму или даже исключения возможности доступа сторонних лиц к информации, используемой для идентификации, является применение специальной энергонезависимой памяти, реализующей защиту хранящихся в ней данных на аппаратном уровне. СБИС такой памяти серийно выпускаются некоторыми фирмами. Микросхемы защищенной памяти способны не только хранить информацию, но и выполнять с ней операции, предусмотренные протоколами идентификации, чтобы исключить временное хранение и использование на программно-доступных узлах компьютерных систем.

Основным недостатком существующих СБИС защищенной памяти является относительно малый объем информации, который может в ней храниться, в то время, как с ростом числа абонентов объемы информации, используемые для их идентификации быстро растут. В работе [3] предложена схема идентификации, требующая для идентификации всех абонентов только один код. Однако использование такой схемы не защищает от доступа легальных абонентов к недозволенным им ресурсам системы. Достоинством этой схемы является изменяемость идентифицирующей посылки при каждом обращении к системы.

Исходя из этого, для эффективного использования аппаратных средств защиты идентификационной информации в многопользовательских системах необходимо разработать способы идентификации, не требующие хранения больших объемов идентификационной информации.

Структура аппаратно-защищенной памяти

Основным назначением аппаратно реализуемой защищенной памяти (ЗП) является хранение идентификационной информации и осуществление контроля доступа к ней. Память должна быть энергонезависимой, чтобы исклю-

чить риск доступа к хранящейся в ней информации в процессе ее записи.

В основу построения защищенной памяти положены следующие принципы:

1. Аппаратно реализуемая ЗП должна иметь открытый интерфейс подключения к стандартным шинам компьютеров, в частности к шине PCI. Конфигурирование ЗП и выделение ей адресного пространства должно выполняться в соответствии с технологией Plug and Play.

2. В ЗП хранится информация, которая является объектом защиты. Эта информация не должна записываться в ЗП и считываться из нее в явном виде.

3. ЗП кроме функций хранения информации должна реализовать операции криптографической обработки, в которой используется указанная секретная информация. Отсюда следует, что эффективность ЗП напрямую зависит от ее специализации: тем больше функций первичной обработки, связанной с секретной информацией будет выполнять ЗП, тем ее использование будет эффективнее в плане реализации функций защиты.

Структура защищенной памяти, ориентированной для использования в системах идентификации абонентов показана на рис.1.

При записи информации в накопитель, адрес поступает на входной интерфейс и фиксируется на регистре адреса. Сами данные, которые записываются в ЗП, маскируются кодом, который известен только главному администратору системы. Для маскирования используется операция побитового суммирования по модулю 2. Все коды масок, которые секретны и известны только администратору системы хранятся в постоянной памяти ЗП. При записи секретной информации необходимо задать адрес используемой маски. этот адрес фиксируется на регистре адреса маски. Соответственно, из постоянной памяти масок считывается секретный код маски. На блоке логических элементов XOR маска снимается с записываемого кода и последний записывается в память. Такой порядок записи исключает возможность перехвата секретной информации во время записи в защищенную память.

Аналогичный порядок маскирования используется и при задании данных, которые сравниваются в процессе идентификации с кодами, хранящимися в накопителе ЗП. Код, который необходимо сравнить с секретным идентифицирующим кодом, хранящимся в ЗП, маскируется администратором и подается на вход ЗП.

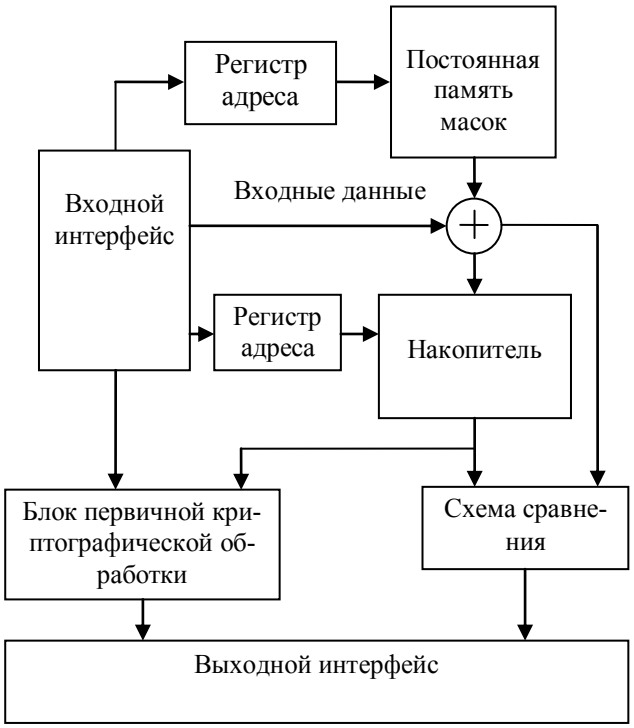


Рис.1. Структурная схема защищенной памяти, ориентированная для использования в системах идентификации абонентов

Внутри ЗП маска снимается. Одновременно на входной интерфейс подается адрес секретного кода, с которым производится сравнение. Этот секретный код считывается из накопителя и сравнивается с заданным кодом. Само сравнение осуществляется на аппаратном уровне внутри ЗП. Результат сравнения выдается в компьютер через выходной интерфейс.

Кроме операции сравнения аппаратные средства ЗП позволяют производить первичную обработку секретной информации при реалиизации криптографических алгоритмов защиты информации.

Двухуровневая схема идентификации абонентов многопользовательских систем

Высокая эффективность идентификации удаленных абонентов достигается как результат компромисса между противоречивыми требованиями. Сложность проблемы обусловлена невозможностью построения адекватной модели действий стороны, производящей попытку несанкционированного доступа. В первом приближении, процедура идентификации должна удовлетворять следующим требованиям:

1. Хранение идентифицирующей информации должна быть таким, что ее часть находится у абонента, а другая – в системе и каждая из час-

тей не была бы самодостаточной для доступа к ресурсам системы.

2. Пароль должен выбираться абонентом, не храниться в памяти, а вводиться при каждом сеансе и не быть достаточным для предоставления доступа к ресурсам.

3. Минимальное использование линий передачи данных – наиболее уязвимого места, с точки зрения незаконного проникновения к ресурсам системы;

4. Изменение идентифицирующей посылки абонента информации при каждом сеансе обращения к системе.

5. Объем сохраняемой в системе закрытой информации, которая используется для идентификации абонентов должен быть возможно меньшим.

6. Идентификационная информация при передаче по линии передачи данных должна шифроваться с использованием ключей, одинаковых для всех абонентов.

7. Распознавание легальных абонентов и сопоставление предоставляемых им прав доступа должно реализоваться разными механизмами защиты.

Приведенные требования сложно удовлетворить в рамках одноуровневой схемы. Поэтому целесообразным представляется разнесение этапа установления легальности абонента и установления прав доступа его к ресурсам системы в рамках двух разных уровнях идентификации. Такое разнесение реализовано в рамках разработанной организации идентификации абонентов многопользовательских систем.

Сущность предложенной двухуровневой организации идентификации абонентов состоит в том, что на первом уровне производится установление легальности абонента без использования системной информации, привязанной к конкретному пользователю. На втором уровне для идентификации абонента используются хранящиеся в системе данные, относящиеся к конкретному абоненту. Такой принцип использования идентифицирующей информации позволяет ускорить фильтрацию обращений к системе, связанных с попытками незаконного проникновения к ее ресурсам со стороны нелегальных пользователей.

В предлагаемой организации идентификации абонентов используются симметричное:

$$R = SCT(D,K)$$

и несимметричное (с открытым ключом):

$$R = NSCT(D,K_D)$$

криптографические преобразования (в качестве первого может, например, использоваться алгоритм Rijndael, а в качестве второго–RSA).

Через D обозначен блок данных до шифрования, а через R-после шифрования, K–ключ преобразования. Обратные преобразования обозначены как:

$$D = SCT^{-1}(R,K) \text{ и } D = NSCT^{-1}(R,K_R), K_D \neq K_R$$

Кроме упомянутых преобразований предлагаемая организация предполагает использование функции P(X) перестановки битов кода X ,через P^{-1} обозначая обратную перестановку, так, что:

$$X = P^{-1}(P(X))$$

При идентификации используется хеш-преобразование H(X), формирующая хеш-сигнатуру X. В качестве такой функции может быть использован один из хеш-алгоритмов, например, SHA.

Организация регистрации абонента схематично показана на рис.2.

При регистрации абонента A, им произвольно выбирается мнемонический пароль P_A, который вводится абонентом при каждом сеансе обращения к системе. При регистрации этот пароль P_A передается системе, где перемешивается с секретным постоянным кодом W:

$$K_A = P(P_A, W)$$

Полученный код K_A используется в качестве части ключа для преобразования универсального для всех абонентов системы секретного кода U во вторую часть пароля D_A абонента:

$$D_A = SCT(U, K_A)$$

Вычисленная описанным способом часть пароля D_A перемешивается:

$$D_{A'} = P(D_A)$$

возвращается системой абоненту вместе с его номером N_A. В памяти системы выделяется область памяти, адресуемая кодом N_A. В этой области записываются ключи доступа абонента А к ресурсам системы. Генерируется случайная строка S, которая сохраняется в области памяти абонента А. Эта строка вместе с D_A и N_A возвращаются абоненту А.

Обмен регистрационной информации производится в зашифрованном виде. Для этого абонент с использованием открытого ключа K_D системы шифрует мнемонический пароль P_A и случайно выбранный абонентом сеансовый ключ K_C : $T_1 = NSCT((P_A, K_C), K_D)$.

Система с использованием закрытого открывающего ключа K_R восстанавливает коды P_A и K_c. С использованием полученного сеансового

ключа K_C система шифрует сгенерированную часть D_A пароля абонента, его номер N_A и сохраненную строку S:

$$T_2 = SCT((D_A, N_A, S), K_C)$$

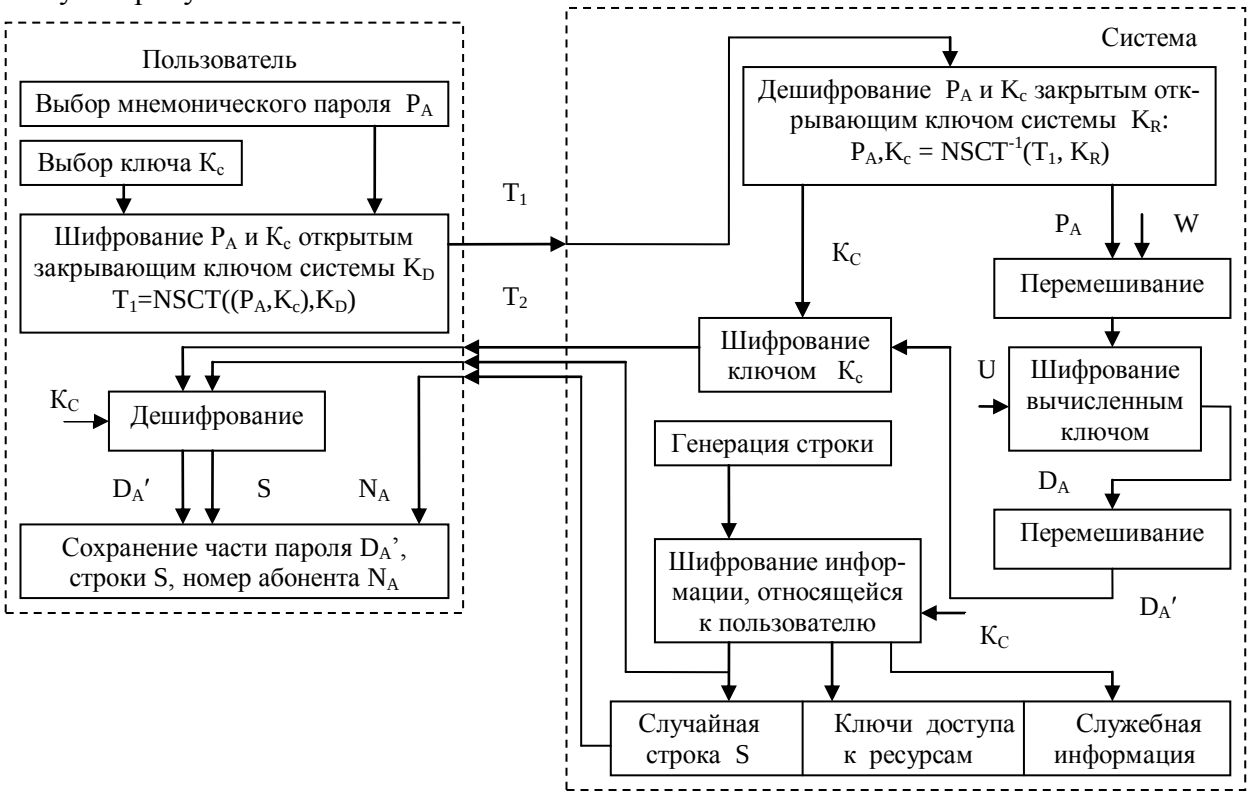


Рис.2. Структура операций, выполняемых при регистрации абонента

Таким образом, после регистрации в закрытой памяти системы сохраняются общие для всех абонентов коды W и U. Кроме того, в общей памяти системы в области, адресуемой N_A, сохраняется строка S и коды доступа к ресурсам.

Абонент сохраняет после регистрации в мнемонической памяти пароль P_A, в компьютере хранится часть пароля D_A и принятая от системы строка S.

Организация идентификации абонента при его обращении к системе схематично показана на рис.3.

При обращении абонента к системе, выполняется цикл его идентификации, включающий следующую последовательность действий:

1.Абонент А вводит строку мнемонического пароля P_A, который конкатенируется со строкой S. Над результатом конкатенации выполняется хеш-преобразование H с получением новой строки: $S' = H\langle P_A | S \rangle$

Строка S' замещает в памяти ранее хранившуюся строку S.

2.Абонентом выполняется конкатенация мнемонического пароля P_A, второй части пароля – D_A', номера N_A и строки S'. Результат конкатенации шифруется открытым закрывающим ключом K_D системы:

$$T = NSCT(P_A, D_{A'}, N_A, S')$$

Полученный код T отсылается в систему.

3.Многопользовательская система принимает идентифицирующий код T, посланный абонентом А и, используя свой секретный открывающий ключ K_R, выполняет дешифрацию компонент принятого кода:

$$P_A, D_{A'}, N_A, S' = NSCT^{-1}(T)$$

4.Обратной перестановкой битов система восстанавливает исходный код:

$$D_A = P^{-1}(D_{A'})$$

и вычисляет код ключа K_A путем перемешивания с секретным постоянным кодом W:

$$K_A = P(P_A, W)$$

5.С использованием полученного ключа K_A выполняется обратное криптографическое преобразование над кодом D_A:

$$R = SCT^{-1}(D_A, K_A)$$

Если результат равен коду U, то есть если R=U, то принимается решение о легальности абонента А. В противном случае, в доступе отказано.

6.Если установлен факт легальности абонента А, то выполняется идентификация прав

доступа абонента А к оговоренным ресурсам системы.

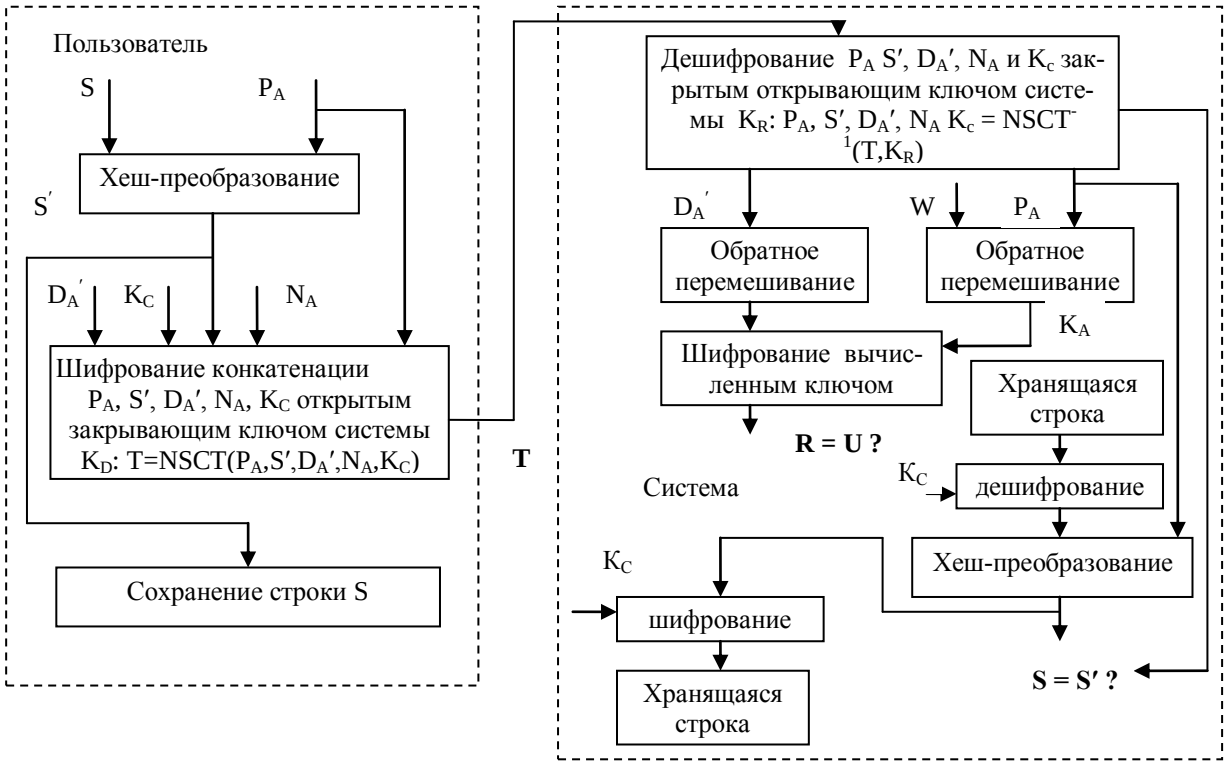


Рис.3. Структура цикла идентификации

Для этого из области памяти системы, адресуемой кодом N_A, считывается строка S_A.

Выполняется конкатенация полученного от абонента мнемонического пароля P_A со считанной из памяти строкой S_A. Над результатом конкатенации выполняется хеш-преобразование H с получением новой строки:

$$S_{A'} = H\langle P_A | S_A \rangle$$

Полученная в результате хеш-преобразования строка S_A' сравнивается со строкой S', полученной от абонента. Если эти строки совпадают, то есть если: $S_{A'} = S'$, то абоненту предоставляется право использовать ресурсы системы, обозначенные в области памяти N_A. В этом случае строка S' замещает в памяти ранее хранившуюся строку S_A в области памяти, адресуемой N_A. Если: $S_{A'} \neq S'$, то возникшая ситуация классифицируется как попытка доступа легального пользователя к ресурсам, доступ к которым не оговорен при его регистрации. Соответственно, в доступе отказано и замещения кода строки в области, адресуемой N_A, не производится.

Таким образом, предложенная организация реализует двухуровневую схему идентификации абонентов многопользовательских систем. На первом уровне со стороны системы используются только три секретных кода: открывающий ключ K_R, и произвольно выбираемые при инициализации системы коды W и U, одинаковые для всех абонентов. На первом уровне признаком того, что абонент легальный является совпадение результата описанного в пп.4-5 преобразования с секретным, единым для всех пользователей, кодом U, а не результатом совпадения с элементами списка идентификационной информации, как это реализовано в известных схемах идентификации удаленных пользователей [1,2]. Это обусловливает высокую скорость идентификации удаленных пользователей, причем, время идентификации не зависит от их количества. При этом многократно уменьшается объем хранящейся в системе секретной информации, что упрощает техническую реализацию закрытой памяти.

На втором уровне идентификация осуществляется сравнением строк, сгенерированных абонентом и системой. Это обеспечивает изменчивость идентифицирующей посылки для каждого из сеансов обращения к системе. Анализ отказов в доступе при реализации второго уровня идентификации позволяет эффективно выявлять попытки незаконного доступа к ресурсам системы и осуществлять мониторинг ее безопасности.

Предложенная организация идентификации использует только один цикл передачи идентифицирующей информации от абонента к системе. Это уменьшает риск незаконного проник-

новения в систему путем внедрения в процессе передачи. Кроме того, снижается использование важного для систем коллективного доступа ресурса – линии передачи.

Важным достоинством предложенной организации является то, что в ее рамках достигается разнесение информации, используемой при идентификации: выбранный абонентом мнемонический пароль P_A не сохраняется в компьютерной памяти, что исключает несанкционированный доступ к нему; однако подбор этого пароля неэффективен в силу того, что сам по себе он позволяет получить доступ к ресурсам системы. Дополнительная часть пароля D_A хранится в преобразованном виде D_A' только в компьютерной памяти абонента. Общая для всех абонентов часть пароля W хранится в закрытой памяти системы. Это не позволяет абоненту, которому не известны коды D_A и W, равно как и функция P перестановки, установить код U.

Для проникновения в систему нелегального пользователя, последний должен знать пароль P_A, соответствующий ему код D_A, номер N_A и код строки S. Разрядность D_A и P_A равна 256 (при использовании в качестве симметричного преобразования Rijndael), длина S больше 256. Очевидно, что успешный подбор указанных компонент маловероятен. Для получения доступа легального пользователя к недоступным для него ресурсам необходимо также подобрать связанные между собой необратимыми преобразованиями коды P_A, D_A, N_A и S. В случае проникновения к общей памяти системы можно получить коды строк S и номер области памяти, что не позволяет реализовать доступ извне: для этого надо подбирать коды P_A и D_A.

Выводы

Одним из наиболее эффективных подходов к снижению риска несанкционированного доступа к идентификационной информации со стороны системы целесообразно организовать хранения всех связанных с идентификацией данных в специальной энергонезависимой памяти, реализующей защиту на аппаратном уровне и, кроме того, реализующие все операции обработки таких данных.

Сформулированы принципы работы аппаратно защищенной памяти, ориентированной для использования в системах идентификации удаленных абонентов, обоснована и разработана структура такой памяти

Предложена двухуровневая организация идентификации абонентов многопользовательских систем, отличающаяся тем, что для снижения риска несанкционированного расширения прав доступа со стороны легальных абонентов, реализованы два уровня идентификации, на первом из которых выявляются легальные абоненты системы, а на втором – производится верификация их прав доступа. В отличии от известных схем идентификации, определение легальности абонента производится без обращения к области памяти, связанной с абонентом. Это позволяет ускорить идентификацию и сократить объем секретной информации до трех кодов: K_R, W и U, общих для всех абонентов, что упрощает реализацию специальной защищенной памяти.

Список литературы

1. Bengio S., Brassard G., Desmedt Y.G. Goutier C.,Quisquater J.J. “Secure implementation of identification system”, Jornal of Cryptology, v.4, n.3, 1991, pp.186-192.
2. Menezer A.J., Van Oorschot P.C., Vanstone S.A. Handbook of Applied Cryptography. CRC-Press, – 1997. – 780 p.
3. Bardis N.G., Polymenopoulos A., Bardis E.G., Markovskyy A.P. , “Methods for Increasing the Efficiency of the Remote User Authentication in Integrated Systems”, TRENDS IN COMPUTER SCIENCE, Volume 12. – No.1,– 2003. – pp.99-107.

УДК 004.056.5

МАРКОВСКИЙ А.П.,
ШАРШАКОВ А.С.

СПОСОБ УСКОРЕННОЙ ГЕНЕРАЦИИ ПСЕВДОСЛУЧАЙНЫХ ПЕРЕСТАНОВОК ЭЛЕМЕНТОВ В СИСТЕМАХ ЗАЩИТЫ ИНФОРМАЦИИ

В статье предложен новый способ генерации псевдослучайных перестановок элементов конечного множества, базирующийся на идее конгруэнтных генераторов. Предложенный способ позволяет ускорить формирование случайных перестановок без значительных затрат памяти. Показано, предложенный подход позволяет увеличить скорость программной реализации комбинаторной задачи формирования перестановок элементов в n раз по сравнению с известными методами. Предлагаемый способ может найти широкое применение в современных высокопроизводительных системах защиты информации.

In paper we propose the new approach to generation of pseudorandom finite set of n elements permutation based on the idea of congruent generators usage. The proposed approach allows to speed up the pseudorandom permutations generation without extra memory amounts. It has been shown that proposed approach allows to speed up software implementation of combinatory task permutations generation by n times in compare to known methods. The proposed approach could be used in modern high-performance data security systems.

Введение

Генерация случайных чисел, и их последовательностей играют важную роль в информационных технологиях. Они широко используются при статистическом моделировании, тестировании аппаратных и программных средств, в технологиях искусственного интеллекта. В последнее десятилетие быстрый прогресс информационной интеграции стимулирует развитие систем защиты данных, в которых широко используются случайные и псевдослучайные элементы [1].

Одной из важных для современных технологий защиты информации задач является генерация случайных перестановок элементов конечного множества (ГСПЭ). Эта задача лежит в основе перестановочных криптографических алгоритмов, процедур модификаций преобразований алгоритмов блокового шифрования, полиморфной реализации программных кодов для защиты данных от их реконструкции анализом динамики потребляемой мощности (АДПМ). При работе же с полиморфными программами реализация незаконного доступа к данным с использованием АДПМ значительно усложняется [2].

При всех указанных применениях ГСПЭ важным фактором эффективности является время решения этой задачи. Поскольку защита данных является, по своей сути, вспомогательной функцией, ее реализация не должна существенным образом сказываться на скорости обработки информации. Поэтому комбинаторная задача ГСПЭ должна решаться ДОСтаточно быстро и в качестве критерия здесь используется зависимость числа выполняемых операций от количества элементов. Предпочтение отдается тем методам, которые реализуют задачу не за счет дополнительных ресурсов, а за счет математического аппарата.

Другим важным критерием эффективности при решении задачи ГПСЭ количества вариантов перестановки, которые могут быть сгенерированы. Для некоторых применений важным является, чтобы это число вариантов было достаточно большим, для других играет роль, чтобы генерировались принципиально все возможные варианты перестановок.

Таким образом, научная задача создания эффективных способов и средств генерации случайных и псевдослучайных перестановок элементов конечного множества является актуальной и практически важной для современного этапа развития информационных технологий в целом и систем защиты информации в частности.

Анализ известных технологий ГСПЭ

Задача ГСПЭ может быть сведена к выбору последовательности номеров следования элементов конечного множества $\Omega=\{X_1,X_2,,X_n\}$. В таком ракурсе можно рассматривать элементы множества Ω как натуральные числа от 1 до n, т.е. $X_1=1$, $X_2=2,\ldots$, $X_n=n$.

Для задачи ГСПЭ основными критериями эффективности механизма ГСПЭ являются [3]:

– длительность цикла n повторения чисел в выборке;

– число вариантов h перестановок элементов множества Ω;

– время генерации t очередного элемента множества Ω;

– среднее время генерации T множества Ω;

Существует ряд известных технологий решения задачи ГСПЭ [1]. Их можно разделить на два класса, в зависимости от механизма, обеспечения неповторяемость элементов:

– методы, которые обеспечивают неповторяемость элементов Ω в генерируемой последовательности за счет обработки указанного множества (т.е. выполняя сравнение с ранее сформированными элементами указанного множества и исключение из множества);

– методы, в которых неповторяемость элементов множества Ω обеспечивается математическими процедурами генерации Ω (т.е. используется технология генерации последовательности, не требующей обработки).

Ниже будут проанализированы технологии обоих классов и даны их характеристики.

Тривиальная технология, относящаяся к первому классу, состоит в том, что вначале множество Ω полагается пустым, а затем выполняется n циклов, в i-том (i=1,…,n) из которых генерируется случайное число $r\in\{1,\ldots,n\}$, которое сравнивается с ранее сформированными элементами множества Ω: если $r\notin\Omega$, то элемент r добавляется к множеству Ω: $\Omega = \Omega\cup r$; иначе генерируется новое число $r\in\{1,\ldots,n\}$ и снова осуществляется проверка, пока не будет сгенерировано такое r, которое можно добавить ко множеству Ω (т.е. $r\notin\Omega$). Пример процесса ГСПЭ для n = 8 представлен в таблице 1. Среднее число g_i циклов генерации i-го элемента множества Ω определяется формулой:

$$g_i = \sum_{j=1}^{\infty} j\cdot\frac{n-i+1}{n}\cdot\left(\frac{i-1}{n}\right)^{j-1} = \quad (1)$$

$$= \frac{n-i+1}{n}\cdot\sum_{j=1}^{\infty} j\cdot\left(\frac{i-1}{n}\right)^{j-1} =$$

$$= \frac{n-i+1}{n}\cdot\left(\frac{n}{n-i+1}\right)^{2} = \frac{n}{n-i+1}$$

Суммарное число g_Σ циклов генерации всех элементов множества Ω определяется формулой:

$$g_\Sigma = \sum_{i=1}^{n} g_i = n\cdot\sum_{i=1}^{n}\frac{1}{i} = n\cdot\ln(n) \quad (2)$$

Среднее время T_1 генерации всех n элементов множества Ω определяется выражением: $T_1 = g_\Sigma\cdot(\tau + 0.25\cdot t\cdot n) = n\cdot\ln(n)\cdot(\tau + 0.25\cdot t\cdot n)$, где τ – время генерации r, t –время выполнения команды процессора. Вычислительная сложность этого способа составляет $O(n^2\cdot\ln(n))$.

Табл. 1. Пример ГСПЭ базовым способом первого класса

№ цикла	*r*	Ω
		{}
0	2	{2}
1	7	{2,7}
2	2 5	{2,7} {2,7,5}
3	4	{2,7,5,4}
4	5 7 1	{2,7,5,4} {2,7,5,4} {2,7,5,4,1}
5	3	{2,7,5,4,1,3}
6	4 6	{2,7,5,4,1,3} {2,7,5,4,1,3,6}
7	3 1 0	{2,7,5,4,1,3,6} {2,7,5,4,1,3,6} {2,7,5,4,1,3,6,0}

Далее приведен анализ другой, наиболее известной из относящихся к первому классу, технологии ГСПЭ. Она сводится к выполнению n циклов, в i-том из которых (i=1,…,n) генерируется случайное число $r\in\{1,\ldots,n+1-i\}$ с исключением r-го элемента из копии S множества Ω. Реализация исключения выбранного элемента r решается путем сдвига части множества S. Среднее время T_2 ФСПЭ составляет $T_2=n\cdot(\tau+0.25\cdot t\cdot n)$, где τ – время генерации случайного числа r, t – время выполнения команды процессора.

Еще одна технология, основанная на идее предвычислений, состоит в предварительном формировании $G \le n!$ различных перестановок элементов Ω с сохранением их в памяти. Соответственно, при ГСПЭ генерируется случайное число $q\in\{1,..,G\}$ которое и определяет номер считываемой из памяти последовательности S. Время T_3 ГСПЭ состав-ляет $T_3 = \tau + t\cdot n$, а требуемый для хранения предвычисленных перестановок объем V_3 памяти: $V_3 = G\cdot n$. При очевидном выигрыше по времени ГСПЭ на устройстве, требуются значительные затраты па-

мяти и дополни-тельные меры по защите от утечки информации о таблице предвычислений.

Общим недостатком способов, относящихся к первому классу, является то, что время получения множества Ω имеет квадратичную зависимость от числа n элементов формируе-мого множества, то есть вычислительная сложность процедур генерации составляет $O(n^2)$. Достоинством технологий рассмат-риваемого класса является то, что они позволяют реализовать близкое к макси-мальному (т.е. $n!$) число вариантов h пере-становок элементов множества Ω.

Способ ГСПЭ конгруэнтного типа

Для реализации процедуры генерируется два случайных числа $1 \le \alpha$ и $1 \le \beta < n$, причем β и n должны быть взаимно простыми, то есть не иметь общих делителей. Каждое i-тое число X_i множества Ω формируется в соответствии со следующим выражением:

$$X_i = (\alpha + i \cdot \beta) \bmod n \qquad (3)$$

По сути, предлагаемый подход к генерации неповторяющейся последовательности из n натуральных чисел является модификацией конгруэнтных генераторов [1]. Достаточно просто показать, что ни одно из n генерируемых с использованием выражения (3) чисел не повторяется. Действительно, если пред-положить, что в процессе генерации формируются два равных между собой числа $X_i = X_j$ с порядковыми номерами i и j (причем $i, j \in \{1,..,n\}: j > i$), то это означает, что $i \cdot \beta \bmod n = = j \cdot \beta \bmod n$ или, что есть то же самое, $(j-i) \cdot \beta \bmod n = 0$. Из последнего следует, что суще-ствует целое d такое, что $(j-i) \cdot \beta = d \cdot n$, или:

$$\frac{(j-i) \cdot \beta}{n} = d \qquad (4)$$

Если $j-i$ и n не имеют общих делителей, то есть являются взаимно простыми, то левая часть выражения (4) не может быть целым числом. Это означает, что выражение (4) не выполняется при условии существования целого d. Пусть $j-i$ и n имеют наибольший общий целый делитель a, то есть $n = a \cdot s$ и $j-i = a \cdot c$. Тогда, в силу того, что $j-i < n$, то $c < s$, левая часть (4) имеет вид: $c \cdot \beta / s$. Поскольку ни c, ни β не имеют общих делителей с s, то выражение $c \cdot \beta / s$ не может быть целым числом. А это значит, что равенство (4) не может иметь места. Таким образом, доказано, что генерация элементов множества Ω в соответствии с выражением (3) обеспечивает отсутствие повторений.

Функционирование предложенного способа ГСПЭ иллюстрируется следующим примером. Например, при $\alpha = 3$, $\beta = 5$ и $n = 8$ генерируется следующая последовательность значений, составляющих множество Ω: {3,0,5,2,7,4,1,6}, как показано в таблице 2.

Табл. 2. Пример ГСПЭ предлагаемым способом

№ цикла i	Выражение для X_i	Ω
		{}
0	(3+0·5) mod 8	{3}
1	(3+1·5) mod 8	{3,0}
2	(3+2·5) mod 8	{3,0,5}
3	(3+3·5) mod 8	{3,0,5,2}
4	(3+4·5) mod 8	{3,0,5,2,7}
5	(3+5·5) mod 8	{3,0,5,2,7,4}
6	(3+6·5) mod 8	{3,0,5,2,7,4,1}
7	(3+7·5) mod 8	{3,0,5,2,7,4,1,6}

Очевидно, что порядок следования чисел в генерируемой последовательности не зависит от значения α: при изменении α меняется начальное значение. Например, при $\alpha = 1$, $\beta = 5$ и $n = 8$ генерируется следующая последовательность значений, составляющих множество Ω: {1,6,3,0,5,2,7,4}.

Ясно, что количество $N(\alpha)$ возможных значений α не превышает n: $N(\alpha) \le n$. В то же время количество $N(\beta)$ возможных значений β меньше n: $N(\beta) < n$. Общее количество возможных последовательностей чисел длиной n определяется произведением $N(\alpha) \cdot N(\beta) < n^2$.

Важно отметить, что на начальное значение генерируемой последовательности влияют как α, так и i. Соответственно, можно получать начальное значение α либо i, используя ГПСЧ любой разрядности, поскольку α и i не обязательно меньше n. Для получения следующих элементов множества Ω используется инкремент либо декремент i.

Однако стоит отметить следующую особенность: если есть такие $\beta_1 \ne \beta_2$, что $\beta_1 + \beta_2 = n$, то β_1 и β_2 генерируют обратные последовательности. Например, для n=8, $\alpha = 0$:

$\beta_1 = 1$, $\Omega = [0, 1, 2, 3, 4, 5, 6, 7]$.
$\beta_2 = 7$: $\Omega = [0, 7, 6, 5, 4, 3, 2, 1]$.
$\beta_3 = 3$, $\Omega = [0, 3, 6, 1, 4, 7, 2, 5]$.
$\beta_4 = 5$: $\Omega = [0, 5, 2, 7, 4, 1, 6, 3]$.

Исходя из этого, следует учесть закономерность, по которой определяется число вариантов h перестановок элементов множества Ω. При $n = 8$ есть 2 пары $\beta_1 = 1$, $\beta_2 = 7$ и $\beta_3 = 3$, $\beta_4 = 5$. При n вариантах выбора начального значения последовательности получим $h = 4 \cdot 8 = 32$.

При $n = 16$ есть 4 пары $\beta_1 = 1$, $\beta_2 = 15$, $\beta_3 = 3$, $\beta_4 = 13$, $\beta_5 = 5$, $\beta_6 = 11$ и $\beta_7 = 7$, $\beta_8 = 9$. При n ва-

риантах выбора начального значения последовательности получим: $h = 8 \cdot 16 = 128$.

То есть количество вариантов h перестановок элементов множества Ω пропорционально n и количеству таких чисел β ($1 \le \beta < n$), которые являются взаимно простыми с n. С учетом этого сложно оценить число вариантов h перестановок элементов множества Ω для данного способа в общем случае. Экспериментальные данные о числе вариантов h перестановок для $10 \le n \le 200$ представлены графически на рисунке 1. При составлении графика для демонстрации тенденции увеличения числа вариантов h перестановок с увеличением n была выполнена интраполяция. Фактически зависимость $h(n)$ связана с распределением взаимно простых с n чисел и сложно предсказуема.

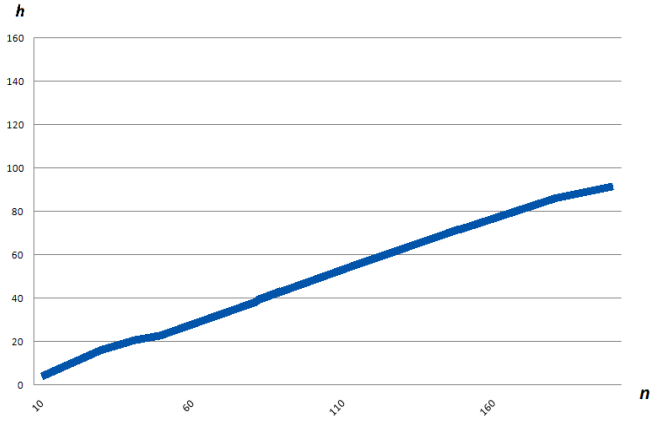


Рис. 1. График зависимости количества перестановок h от количества элементов n

Время T_4 генерации последовательности чисел, составляющих множество Ω определяется как: $T_4 = 2 \cdot \tau + 3 \cdot n \cdot t$, причем для генерации очередного элемента требуется только 3 команды: умножения, сложения и целочисленного деления для нахождения остатка, то есть минимум меньше по сравнению с рассмотренными выше способами. Затраты памяти при реализации предлагаемого способа также незначительны, что соответствует поставленной задаче.

Сравнительные характеристики предложенного способа ГППЭ представлены в таблице 3.

Табл. 3. Сравнительная характеристика параметров описанных механизмов ГСПЭ

Класс	Усл. обозн.	O	V	h
1	1Б	$O(n^2 \cdot \ln(n))$	$\approx n$	$\le n!$
	1И	$O(n^2)$	$\approx 2 \cdot n$	$\le n!$
	1П	$O(n^2)$	$\le n \cdot n!$	$\le n!$
2	2К	$O(n)$	$\approx n$	н/о

Условные обозначения, использованные в таблице 3: 1Б, 1И и 1П – базовый вариант, метод исключения и метод предвычислений соответственно как реализации методов первого класса, 2К – предложенный метод на основе конгруэнтного генератора.

Выводы

Разработан конгруэнтный способ ГСПЭ, обеспечивающих повышенную производительность. Доказано, что способ позволяет в n раз ускорить процедуру ГСПЭ при программной реализации за счет того, что для генерации очередного элемента последовательности требуется только 3 команды: умножения, сложения и целочисленного деления.

Предлагаемый способ ГСПЭ ориентирован на применение в современных высокопроизводительных системах защиты информации.

Список литературы

1. Иванов М.А., Чугунков И.В. Теория, применение и оценка качества генераторов псевдослучайных последовательностей. М.: КУДИЦ-ОБРАЗ. – 2003 – 260 с.
2. Марковский А.П., Зюзя А.А. Эффективная реализация псевдослучайного выбора элементов множества // Матеріали XII Міжнародної науково-технічної конференції ”Системний аналіз та інформаційні технології”. – К.: НТУУ ”КПІ”. ІПСА. – 2010. – с.458.
3. Самофалов К.Г., Марковский А.П., Зюзя А.А., Лёзин А.С. Стохастически полиморфная реализация алгоритма Rijndael на микроконтроллерах и смарт-картах // Проблеми інформатизації та управління. Збірник наукових праць. К.:НАУ. – 2010.- Вип. 1(29). – с.150-158.

УДК 004353

ЛАПЧУК А.С.,
ЮРЛОВ В.І.,
ШИЛО С.О.,
ШИХОВЦЯ О. В.

ІНТЕРФЕРЕНЦІЙНІ ЕФЕКТИ ПРИ ВІДТВОРЕННІ ІНФОРМАЦІЇ ЧАСТКОВО КОГЕРЕНТНИМ ВИПРОМІНЮВАННЯМ

Запропоновані оптичні схеми лазерного проектора для зменшення спеклів з використанням двох і трьох частково когерентних світлових пучків. Розроблений алгоритм розрахунку спеклів для запропонованого методу. Показано, що при малій довжині декорреляції і для тих же товщини прозорих пластин, два частково когерентних пучків забезпечують менший контраст спеклів ніж три пучки від різних дифракційних порядків. Однак, для великої довжини декорреляції, конструкція з трьох пучків забезпечує менший контравт спеклів для всіх трьох лазерів різних кольорів і отримані значення є близькі до мінімально можливих значень. Експеримент підтвердив отримані теоретичні результати.

An optical scheme for speckle suppression using two and three partially coherent beams in a laser projection system is proposed. An algorithm is developed for calculation of speckle contrast ratio for the proposed method. It is shown that for a small decorrelation length with using the same thickness of the transparent plates two partially coherent beams would provide better speckle suppression than three beams optical scheme. However, for a large decorrelation length, the three beam setup provides better speckle suppression for all three laser of different colors with a suppression coefficient close to theoretical limits. The theoretical results are confirmed by experiment data.

Вступ

Розвиток портативних проекторів на даний час є одним з основних напрямків розвитку проектційних дисплеїв. Лазерні діоди мають вузький промінь і високу оптичну ефективність, а також забезпечують чистоту кольорів і великий кольоровий охват. Всі ці фактори дозволяють отримати малогабаритні оптичні системи проекторів з малим споживанням енергії і які створюють зображення високої якості з високою насиченістю кольорів. Одним з перспективних технічних рішень проектора на основі лазерних діодів є 1D лазерний проектор. В даний час є декілька добре розроблених технічних рішень для конструкції 1D лазерного проектора [1-4]. Основна відмінністю цих пристроїв є використання дифракційної елементів, що складаються із стрічкових мікро-дзеркал для утворення лінійки пікселів. Дифракційні елементи мають рухомі дзеркала і тому можуть змінити різницю фаз між світлом відбитим від рухомих і нерухомих частин дифракційних елементів при прикладенні напруги до рухомих частин. Відбите від лінійки дифракційних елементів світло проектується на екран за допомогою оптичної системи і при цьому сканується вздовж екрану за допомогою дзеркала, що осцилює навколо вертикальної осі.

Діафрагма з отвором розташована у фокальній площині (Фуріє площині) об’єктива і зупиняє світло від всіх окрім одного (робочого) дифракційних порядків. Інтенсивність світла робочого дифракційного порядку змінюється в залежності від позиції рухомих дзеркал дифракційних елементів (від прикладеної до них напруги). Модулюванням інтенсивність світла робочого дифракційного порядку дифракційних елементів у відповідності з зображенням на позиції променя на екрані і скануванням променя вздовж екрану створюється 2D образ.

Не дивлячись на великі переваги використання лазерних діодів в проекторах, лазерні проектори до цих пір не впроваджені в масове виробництво. Основним стримуючим фактором є спекли – ефект виникнення модуляції інтенсивності світла в образі зображення (образу в людському оці) при спостереженні за світловою плямою, образ якої створює розсіяне на шорсткому екрані когерентне світло (суб'єктивні спекли) [5,6,7,8]. Спекли значно погіршують якість зображення. Можна зменшити величину модуляції за допомогою методів усереднення декількох незалежних спеклів за рахунок: зменшення часової когерентності випромінювання (лазерні діоди з широким спектром) [9,10] ; зменшення просторової когерентності випромінювання використовуючи декілька декорельованих лазерних пучків що падають на екран під різними кутами [11-14] (або шляхом вібрації екрану [15]) ; шляхом використання двох деко-

рельованих променів різної поляризації [16]. Основний параметр, що визначає величину модуляції зображення спеклами є контраст спеклів (CR). Він визначається при рівномірно освітленому екрані як відношення стандартного середньоквадратичного відхилення до середньої інтенсивності в образі екрану [17] :

$$CR = \sigma_I / \langle I \rangle = \sqrt{\langle I^2 \rangle - \langle I \rangle^2} \qquad (1)$$

В 1D лазерному проектору величина контрасту спеклів залежить від місця і параметрів оптичної системи, шо сторює зображення. Тому при дослідженні спеклів в проекторах вимірювання повинні проводитися оптичною системою, що має оптичні параметри аналогічні оптичним параметрам людського ока. Для того щоб око не відчувало перепади інтенсивності, обумовленого спеклами, необхідно щоб контраст спеклів не перевищував 0,04. Такого рівня спеклів можна досягти тільки при використанні усіх можливих методів зменшення спеклів. Тут буде розглянуто пониження спеклів за рахунок частково декорельованих променів, що падають на екран під різними кутами.

Нами було запропоновано більш просте вирішення проблеми створення декількох некогерентних пучків світла з використанням тільки одного лазера для кожного кольору та дифракційного елемента, вставленого в проміжну площину зображення, яке показано на рис. 1. Дифракційні елементи в площині зображення (рис. 1 b) розділяють промінь, що несе зображення, на декілька променів (дифракційних порядків), кожен із яких також несе зображення. Прозорі пластинки різної товщини вставлені в Фур'є-площині об'єктива проектора. Різниця в товщині пластинок повинна бути достатньою для декореляції променів. Разом з тим різниця ходу променів, що проходять крізь різні пластинки, не повинна перевищити глибину фокуса об'єктива, щоб за причини дефокусування окремих променів не відбулося розмивання зображення. Тільки регулярні (періодичні) дифракційні елементи можуть бути застосовані в цьому випадку, з тим щоб дифракційні порядки розділилися в Фур'є-площині і була можливість вносити фазовий пластинки в окремі дифракційні порядки без часткового їх перекриття, і тим самим не спотворити зображення.

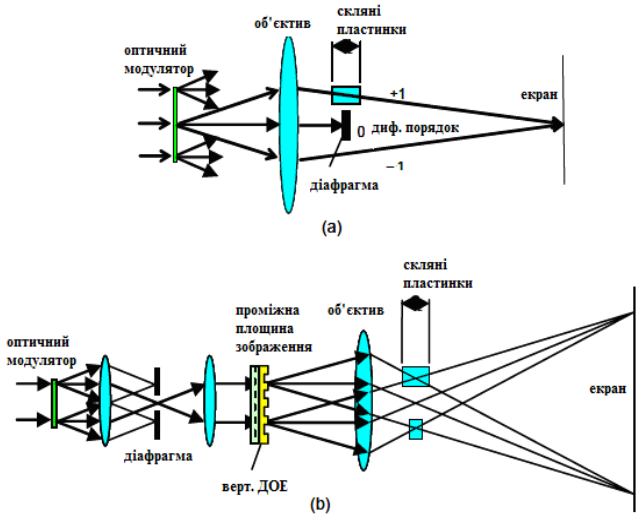


Рис.1. Оптична схема для лазерних проекторів, що використовують два a) і три b) пучка світла дифракційних порядків для створення образу на екрані.

Метод розрахунку контрасту спеклів

Сучасні лазерні діоди мають досить вузький спектр частот, а значить, промінь має досить велику довжину повздовжньої просторової (часової) когерентності. Тому дуже важко отримати повну некогерентність лазерних променів таким чином, особливо для маленьких проекторів. За цієї причини, для обчислення контрасту спеклів для такої оптичної системи необхідно мати коефіцієнти кореляції між різними пучками світла. Тобто, нам потрібно обчислити коефіцієнти кореляції між окремими пучками, отриманими шляхом розділу одного пучка світла на частини, і які мають різницю оптичного ходу між ними, що дорівнює різниці оптичого ходу між пучками світла, що розповсюджуються через прозорі пластинки різної товщини s_i.

$$I(k) = I_0 \exp -\left[\left(k - k_0\right)^2 / \left(0.5\delta k\right)^2 \right]$$

Для обчислення кореляційного коефіцієнта ми представимо випромінювання лазера як стохастичний стаціонарний процес з набором гармонік, що мають Гаусів розподіл інтенсивності по частоті де k *і* k_0 – хвильовий вектор і центральний хвильовий вектор смуги випромінювання лазерного діода відповідно, і δk – ширина смуги, виміряна на *1/e* рівні інтенсивності поля. Кореляційний коефіцієнт для світлових пучків різних дифракційних порядків може бути записаний як автокореляційна функція з урахуванням різниці шляху двох променів, що проходять крізь різні скляні пластинки різної

товщини s_i. Застосувавши теорему Вінера-Хінчина, можна записати нормовану автокореляційну функцію як Фур'є перетворення спектру інтенсивності лазерного пучка

$$\mu(c\tau) = A\int_{-\infty}^{\infty} I(k)\exp(ikc\tau)\,dk = \\ = \exp\left(-(\delta kc\tau)^2/16\right) \quad (2)$$

де c – швидкість світла, τ – різниця в часі проходження двох променів і A – це коефіцієнт нормування, який визначається з умови $\mu(0)=1$. Враховуючи, що різниця в оптичній довжині шляху між i та j променем обчислюється за формулою $c\tau_{ij} = s_i(n_i-1) - s_j(n_j-1)$ (де $-n_i$ це коефіцієнт заломлення i-ої скляної пластинки) ми можемо переписати рівняння для кореляційного коефіцієнта двох променів як:

$$\mu_{ij} = \exp\left\{-(\delta k/4)^2\left\{s_i(n_i-1) - s_j(n_j-1)\right\}^2\right\} = \\ \exp\left\{-\left(\pi\delta\lambda/(2\lambda^2)\right)^2\left\{s_i(n_i-1) - s_j(n_j-1)\right\}^2\right\} \quad (3)$$

і для випадку $n_i = n_j = n$:

$$\mu_{ij} = \exp\left\{-\left(\pi\delta\lambda/(2\lambda^2)\right)^2\left\{(s_i - s_j)(n-1)\right\}^2\right\} \quad (4)$$

З рівняння (4) легко отримати товщину скла, що забезпечує кореляційний коефіцієнт менше ніж 1/e (декореляційну товщину скла). Для випадку, коли пластинки зроблені з одного матеріалу декореляційні товщини можна обчислити з формули:

$$|s_i - s_j| \geq 2\lambda^2/\left[\pi\delta\lambda(n-1)\right] \quad (5)$$

Як ми вже говорили вище, далеко не завжди можна отримати повну декореляцію променів різних дифракційних порядків за причини обмежень, що накладають на товщину пластинок глибина різкості зображення і обмежена глибина області навколо Фур'є площини, в якій дифракційні порядки не перекриваються. Тому, для оптимального зменшення спеклів потрібно знати, який же буде ефект на контраст спеклів, якщо застосовувати для створення зображення тільки частково деколерьовані промені. Нижче ми порівняємо контраст спеклів для двох випадків (рис. 1): оптична схема з двома пучками світла (рис. 1a) оптична схема з трьома пучками світла (рис. 1b) В першому випадку ми розраховуємо на два пучка світла однакової інтенсивності для лазерних пучків всіх кольорів в один із яких вставлено скло товщиною s_2. Таку схему можна отримати в оптичній системі без проміжної області зображення, використовуючи +1 і –1 дифракційні порядки оптичного модулятора для створення образу (вони мають однакову залежність інтенсивності від поданої напруги і тому мають однакову інтенсивність). В другому випадку, з використанням трьох променів, потрібно створювати проміжний образ в оптичній системі і, оскільки, в цьому випадку дифракційний елемент є пасивним дифракційним елементом з фіксованою глибиною рельєфу, то ми отримаємо різну інтенсивність дифракційних порядків для лазерів різних кольорів (червоного, зеленого, фіолетового). Проведене нами чисельне моделювання показало, що прямокутної форми дифракційний елемент з однаковою шириною верхніх і нижніх ділянок, (рис. 2) з глибиною рельєфу, яка забезпечує рівну інтенсивність 0 і ±1 дифракційних порядків для зеленого лазеру, дає мінімальну різницю в інтенсивності дифракційних порядків для всіх трьох променів за умови неглибокої модуляції глибини рельєфу $h \leq (n-1)\lambda/2$. На рис. 3 показано залежність інтенсивності дифракційних порядків від довжини хвилі пучка світла для прямокутної форми дифракційного елемента з оптимальною глибиною рельєфу і однаковою шириною верхніх і нижніх поверхонь дифракційного елементу. При чисельному моделюванні ми вважали, що червоний, зелений і синій лазери мають довжину хвилі 640, 532 і 440 нм, відповідно. В такому випадку ми маємо наступний розподіл інтенсивності за дифракційними порядками $I_{R\pm1} = 0{,}5\cdot I_{R0}$ – для червоного лазеру і $I_{B\pm1} = 2{,}9\, I_{B0}$ – для синього.

Тепер ми порівняємо оптичні схеми з двома і трьома лазерними пучками, за умови, що максимальна товщина скляних пластинок s_2 у них однакова (оскільки максимальна товщина визначається фокусною відстанню об'єктива і ми використовували один і той же об'єктив у двох схемах). Для випадку двох частково декорельованих однакової інтенсивності променів формула для контрасту спеклів добре відома [18]:

$$CR = CR_0\sqrt{\left(1+\mu^2\right)/2} \qquad (6)$$

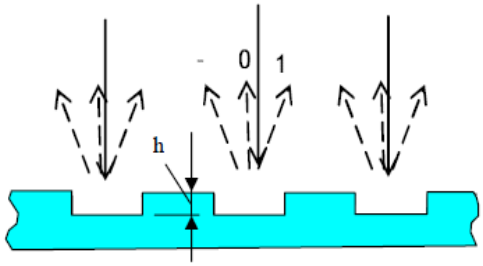


Рис. 2. Поперечний переріз дифракційного елемента, що використовувався для зменшення контрасту спеклів і дифракційні порядки, задіяні в схемі зменшення спеклів.

де CR_0 – контраст спеклів одного пучка світла, а μ – коефіцієнт кореляції між ними.

Але для випадку трьох частково корельованих променів не існує простої формули для обрахунку контрасту спеклів. Тому для цього випадку ми використаємо більш складний алгоритм, заснований на матриці когерентності, розроблений Гудманом [18] для обрахунку контрасту спеклів для оптичної схеми з N частково корельованих променів. В цьому алгоритмі використано ортогональне лінійне перетворення поля трьох частково корельованих променів з тим, щоб представити його як суперпозицію полів трьох декорельованих променів. Для нового представлення поля, як суперпозиції декорельованих полів, контраст спеклів може бути обрахований з вико ристанням формули (1). Інтенсивності декорельованих променів находиться як власні числа матриці когерентності. Для нашого випадку, коли ми маємо однакової інтенсивності 1 і -1 дифракційні порядки, матриця когерентності може бути записана як:

$$\begin{pmatrix} I_0 & \sqrt{I_0 I_1}\mu_{01}{}^* & \sqrt{I_0 I_1}\mu_{0-1}{}^* \\ \sqrt{I_0 I_1}\mu_{01} & I_1 & I_1\mu_{1-1}{}^* \\ \sqrt{I_0 I_1}\mu_{0-1} & I_1\mu_{1-1} & I_1 \end{pmatrix} \qquad (7)$$

де I_0 та I_1 – інтенсивності 0 і ±1 дифракційних порядків, μ_{ij} – коефіцієнти кореляції між ними, а * означає комплексне спряження. Для нахождження власних значень матриці (7) ми повинні розв'язати рівняння:

$$\det\begin{pmatrix} I_0-\lambda & \sqrt{I_0 I_1}\mu_{01}{}^* & \sqrt{I_0 I_1}\mu_{0-1}{}^* \\ \sqrt{I_0 I_1}\mu_{01} & I_1-\lambda & I_1\mu_{1-1}{}^* \\ \sqrt{I_0 I_1}\mu_{0-1} & I_1\mu_{1-1} & I_1-\lambda \end{pmatrix} \qquad (8)$$

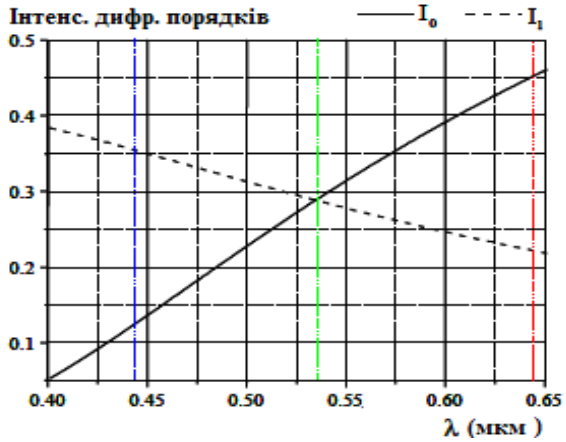


Рис. 3. Інтенсивність дифракційних порядків для прямокутної форми дифракційного елементу з глибиною рельєфу, що відповідає рівній інтенсивності 0 і ±1 дифракційних порядків випромінювання зеленого лазеру (λ=532 нм).

Після простих математичних перетворень рівняння (8) перепишеться як:

$$\lambda^3 - \lambda^2\left(I_0+2I_1\right) + \lambda\begin{bmatrix} 2I_0I_1 + I_1{}^2 - I_1{}^2\left|\mu_{1,-1}\right|^2 \\ -I_0I_1\left(\left|\mu_{0,1}\right|^2 + \left|\mu_{0,-1}\right|^2\right) \end{bmatrix} \qquad (9)$$
$$+ I_0I_1{}^2\begin{pmatrix} \left|\mu_{1,-1}\right|^2 + \left|\mu_{0,1}\right|^2 + \left|\mu_{0,-1}\right|^2 - 1 - \\ \mu^*{}_{0,1}\mu_{0,-1}\mu^*{}_{1,-1} - \mu_{0,1}\mu^*{}_{0,-1}\mu_{1,-1} \end{pmatrix} = 0$$

Ми будемо проводити розрахунок для випадку, коли друга пластинка має вдвічі більшу товщину ніж перша (рис. 1b) внаслідок чого декореляційні коефіцієнти між 0 і -1 і між 1 і -1 є рівними, тобто $\mu_{0,1} = \mu_{1,-1}$. З (4) випливає також, що для цього випадку $\mu_{0,-1} = \left(\mu_{0,1}{}^4\right)$. Рівняння (9) розв'язувалось чисельно на комп'ютері. Для розрахунку контрасту спеклів була використана формула для трьох когерентних декорельованих променів:

$$CR = \sqrt{\lambda_1{}^2 + \lambda_2{}^2 + \lambda_3{}^2}\,/\left(\lambda_1+\lambda_2+\lambda_3\right) \qquad (10)$$

де λ_1, λ_2 і λ_3 це корні рівняння (9). На рис. 4 показано пониження контрасту спеклів для зеленого лазеру від товщини найтовщої пластини s_2 для зеленого лазера з різною шириною спектру випромінювання. З наведених графіків видно, що зміна ширини спектру випромінювання лазеру призводить до простого стискання або розширення кривої залежності контрасту спеклів від товщини пластини. Розширення спектру призводить до пропорційного стискання, а звуження – до пропорційного розширення кривої залежності контрасту спеклів. З рис. 4 випливає

що для тонких пластин, що мають різницю оптичного шляху недостатню для повної декореляції ($\mu_{0,1} = \mu_{1,-1} > 0.621; \mu_{0,-1} > 0.15$) трьох променів, варіант з двома дифракційними порядками призводить до більшого послаблення контрасту спеклів. Більше послаблення спеклів для системи з використанням двох променів отримано тому, що мала товщина s_2 призводить до малої декореляції між 0 і 1 та між 1 і -1 дифракційними порядками, і в той же час до відносно великої декореляцією між 0 і -1 дифракційними порядками для оптичної схеми з трьома пучками світла. В той же час для схеми з двома дифракційними порядками ми будемо мати для двох її дифракційними порядками відносно велику деко реляцію, таку ж як і між 0 і -1дифракційними порядками для оптичної схеми з трьома дифракційними порядками. Це забезпечує для схеми з двома пучка світла пониження контрасту спеклів близького до максимально можливого для цієї схеми (приблизно у *sqrt(2)*=1,4 рази). Тому, якщо оптична схема не дозволяє отримати значної декореляції для трьох променів, краще використовувати оптичну схему з двома пучками світла.

На рис. 5 показано залежність контрасту спеклів від товщини s_2 для червоного лазера при використанні того ж дифракційного елемента. Криві на графіку відрізняються відносно таких же графіків для зеленого лазеру тим, що вони більше витягнуті вздовж осі абсцис. Цей факт пояснюється тим, що за причини більшої довжини хвилі червоного лазера, для досягнення такої ж різниці оптичного ходу в довжинах хвилі (такої ж декореляції) потрібні більш товсті пластини. Не дивлячись на велику різницю в інтенсивності дифракційних порядків для червоного лазера ($I_{\pm1}=0,5*I_0$), контраст спеклів змінюється схожим чином зі зміною s_2 як і для зеленого лазера (рис. 4). Але за причини різної інтенсивності дифракційних порядків, при повній декореляції трьох променів, для червоного лазера кінцевий контраст спеклів є дещо більшим ніж для зеленого лазера. Для фіолетового лазера співвідношення інтенсивності між 0-им і ±1-ими дифракційними порядками є зовсім іншим ніж для червоного і зеленого лазерів. Але, не дивлячись на це, залежність контрасту спеклів від товщини декореляційних пластинок s_2 є приблизно ж такою, як і для зеленого і червоного лазерів (після масштабування відносно осі абсцис відповідно до довжини хвилі (рис. 6). За тієї ж причини, що і для червоного лазера, за повної

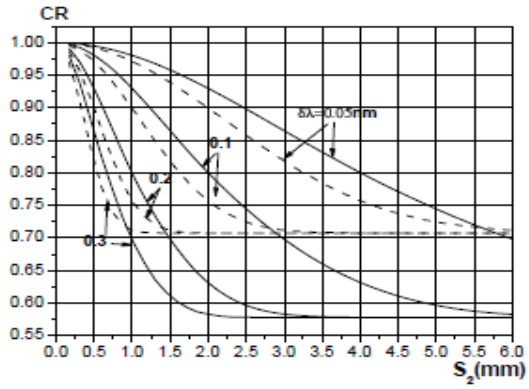


Рис. 4. Залежність контрасту спеклів від товщини пластинки скла (n=1,882) s_2 для зеленого лазера (λ= 0,532 мкм, $I_{\pm1}=I_0$): суцільні лінії - для випадку трьох променів; пунктирні лінії - випадок двох променів. Розглянуто лазер з різною шириною спектру: δλ = 0,05 нм, δλ = 0,1 нм, δλ = 0,2 нм і δλ = 0,3 нм.

декореляції трьох променів, для фіолетового лазера контраст спеклів є більшим ніж для зеленого лазера.

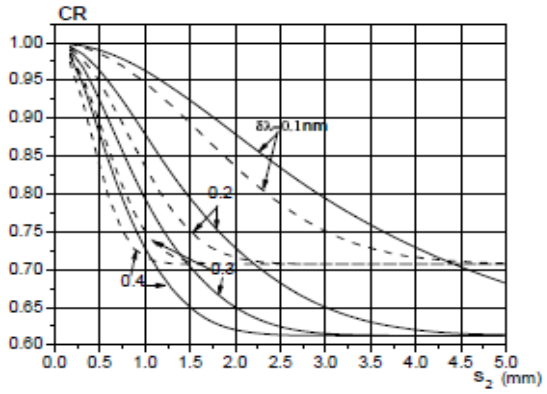


Рис. 5. Залежність контрасту спеклів СК від товщини пластинки скла (n=1.882) s_2 для червоного лазеру (λ=0,640нм; $I_{\pm1}=0.5I_0$): суцільні лінії - для випадку трьох променів; пунктирні лінії - випадок двох променів. Розглядуються чотири варіанти з різною шириною спектру випромінювання: δλ=0,1нм; δλ=0,2нм; δλ=0,3нм; δλ=0,4нм.

Експеримент і обговорення отриманих даних

Для експериментальних досліджень контрасту спеклів для оптичних схем проектора з двома і трьома лазерами були взяті скляні пластинки з товщинами s_1=1,25 мм і s_2=2,5мм відповідно. Товщина 2,5 мм є максимальною товщиною скляної пластинки, яку ще можна

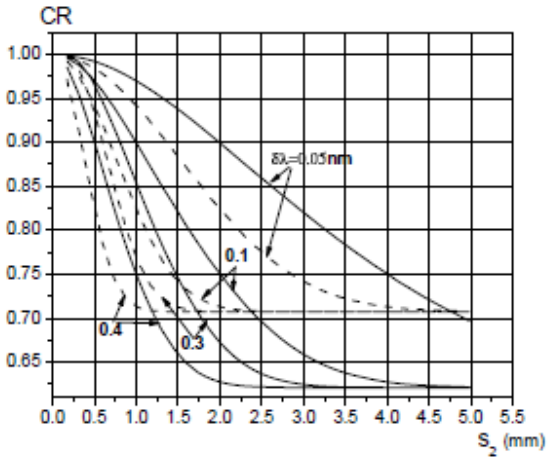
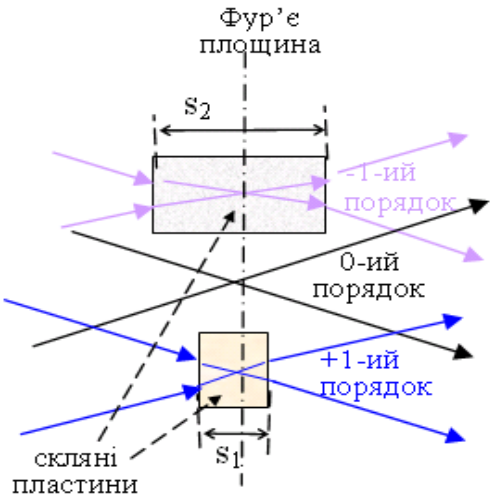


Рис. 6. Залежність контрасту спеклів від товщини пластинки скла (n=1,882) s_2 для фіолетового лазера (λ=0,440нм; $I_{\pm 1}=2.9I_0$): суцільні лінії - для випадку трьох променів; пунктирні лінії - випадок двох променів.

розмістити в Фур'є площині проектора (області без перекриття дифракційних порядків) і яка ще не призводить до погіршення якості зображення за причини дефокусування окремих дифракційних променів (рис.7). Для досягнення максимальної різниці ходу різних дифракційних порядків було виготовлено пластинки зі скла з максимальним коефіцієнтом заломлення n=1,88 (такий самий, як ми використовували при обрахуванні контрасту спеклів для графіків представлених на рис. 4-6. В експерименті використовували по одному лазеру для кожного кольору (червоний, фіолетовий лазерні діоди і зелений лазер).

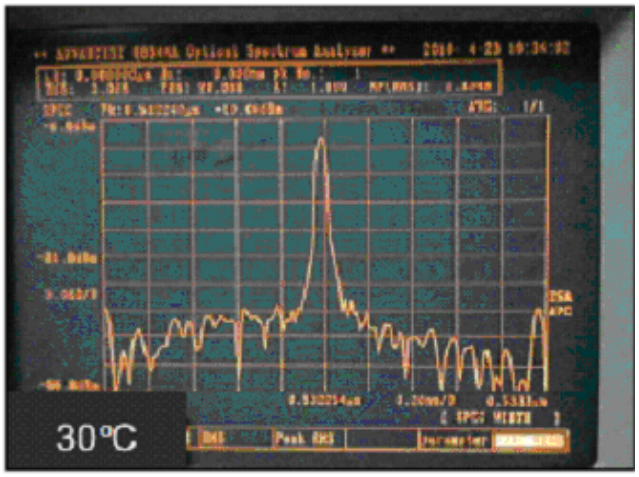
Добре відомо, що при застосуванні для зменшення спеклів методів, що базуються на різних фізичних властивостях світла, результат зменшення контрасту спеклів є мультиплікативним. В 1D скануючому лазерному проекторі існує декілька механізмів зменшення контрасту спеклів, які не можна виключити з експерименту і які описано вище. Тому для знаходження ефекту послаблення спеклів методом застосування декількох частково декорельованих променів, ми використали не величину контрасту спеклів, а коефіцієнт зменшення контрасту спеклів, який було визначено як відношення контрасту спеклів до і після застосування методу частково декорельованих променів. В нашому випадку метод декількох частково декорельованих променів є незалежним від інших механізмів зменшення контрасту спеклів (механізм сканування і поляризаційний). Тому слід очікувати, що коефіцієнт зменшення спеклів повинен мати таке ж значення, як контраст спеклів, який ми отримали при чисельному моделюванні (представлені по тексту зверху) для



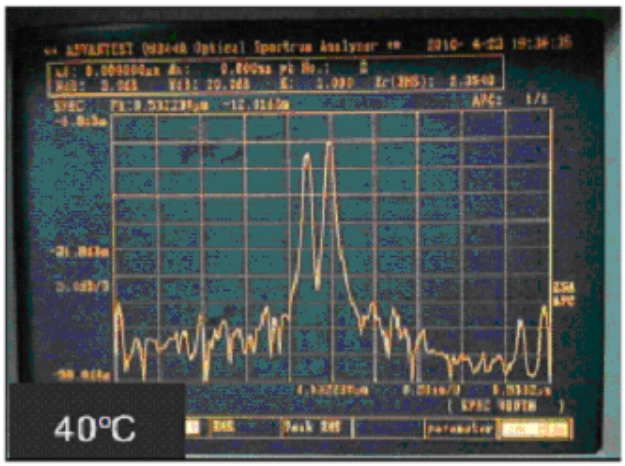
контрасту спеклів когерентних променів за відсутності інших механізмів зменшення спеклів.

Рис. 7. Принципова схема розповсюдження променів різних дифракційних порядків через скляні пластинки в Фур'є-площині об'єктива лазерного проектора.

Вимірювання коефіцієнта зменшення контрасту спеклів для зеленого лазера для оптичної схеми з трьома дифракційними порядками (на рис. 1-b) дало різний рівень зменшення спеклів при вимірюванні при різній температурі, і для різних зразків лазерів. Вимірювання ширини спектра випромінювання зеленого лазеру показало, що він змінюється з температурою і від зміни одного зразка лазера до іншого, від найвужчого ($\delta\lambda$~0,05нм) до найширшого ($\delta\lambda$~~0,14 нм), як показано на рис. 8. Ми отримали коефіцієнт послаблення контрасту спеклів $(1–C/C_0)$ 15% для випадку ширини спектра зеленого лазера 0,05 нм, і 34% для ширини 1,4 нм (де C_0, C – контраст спеклів без і з застосуванням схеми частково корельованих променів, відповідно). Співставлення експериментальних і теоретичних даних показує чітку кореляцію між ними.



a)



b)

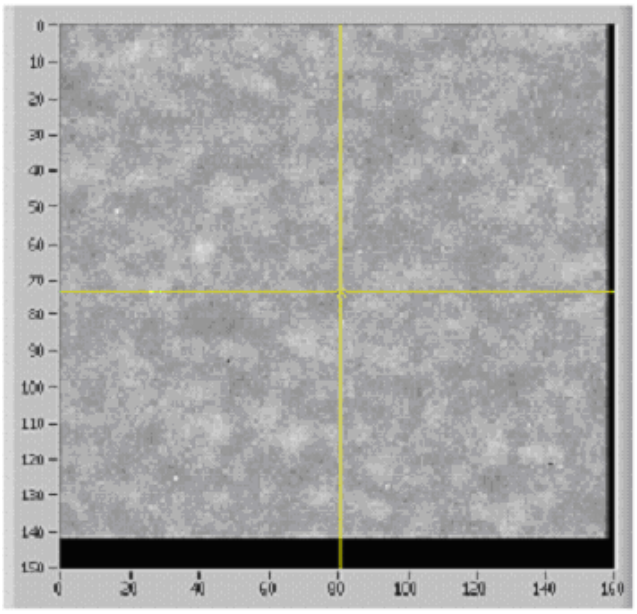
Рис. 8. Спектр випромінювання зеленого лазера при різній температурі навколишнього середовища: a) δλ=0,05 при T= 30°C; b) δλ=0,14 нм при 40°C. Ширина спектра вимірювалась за половинним рівнем інтенсивності від максимуму.

Для червоного лазера для методу з трьома дифракційними порядками в експерименті було отримано зменшення контрасту спеклів ≤20%, що згідно теорії відповідає ширині спектра випромінювання лазера $\delta\lambda$=0,08нм. Вимірювання ширини спектру і технічна документація на лазер підтвердили цю оцінку для ширини спектра червоного лазера. Для збільшення ширини спектра випромінювання червоного лазера для отримання більшого ефекту по зменшенню спекла для живлення лазера було використано напругу модульовану прямокутними імпульсами з частотою в декілька сотень кГц (аналогічно до [19, 20]). Після застосування високочастотної модуляції джерела живлення лазера його спектр розширився до 1нм, а контраст спеклів зменшився на 36% (по відношенню до випадку одного пучка світла). Для цієї оптичної схеми 36% є близьким до теоретично максимально можливого пониження контрасту спеклів для червоного лазера. За теорією таке зменшення в контрасті спеклів відповідає ширині спектру в 0,24 нм (рис. 5). Різниця з теорією в оцінці ширини спектру пояснюється тим, що за ширини спектру 0,24 нм фактично повністю здійснюється декореляція трьох променів для нашої конструкції проектора, і тому подальше розширення спектра лазера не буде призводити до подальшого зменшення контрасту спеклів. Так для $\delta\lambda$=1,0нм теорія дає 38% зменшення контрасту спеклів (див. Табл. 1).

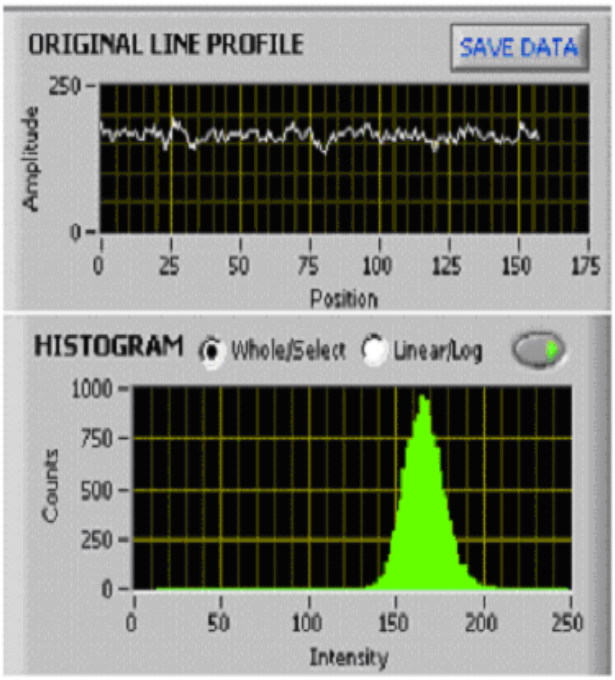
Табл. 1. Зменшення контрасту спеклів: порівняння теорії з експериментом

лазер (довж. хвилі, нм)	шир. спектру (нм, на рівні $0,5*I_0$)	зменш. контр. спекл. (%), теорія	змменш. контр. спекл. (%), експер.
532	0,05 0,14	14% 38%	15% 34%
640	0,1 1,0	20% 38%	20% 35%
440	0,8	38%	38,4%

За причини великої нерівномірності розподілу інтенсивності фіолетового лазера за дифракційними порядками при повній деполяризації трьох променів ми будемо мати лише 38% зменшення контрасту спеклів (див. рис. 6), що менше від максимально можливо значення для трьох повністю декорельованих променів (42%) однакової інтенсивності. Ми вико ристовували фіолетовий лазерний діод з шириною спектру 0,8нм, і це є більш ніж достатньо для повної декореляції трьох променів. В табл. 1 наведені підсумкові дані обрахованого і виміряного зменшення контрасту спеклів для схеми з трьома лазерами. З наведених даних видно хороше співпадіння теоретичних і експериментальних результатів. Але, разом з тим, теоретичні обрахунки дають дещо більший ефект пониження контрасту спеклів



a)



b)

Рис. 9. Результати застосування всіх методів зменшення спеклів з досягненням приблизно 5% контрасту в білому кольорі: a) фотографія білого екрану за допомогою CCD фотоапарата; b) зразок інтенсивності світла вздовж екрану в образі фотокамери і спектр розподіл інтенсивності.

ніж отримані в експерименті результати. Існує декілька причин, які можуть спричинити таку різницю: 1) спектр лазера не має гаусового розподілу за довжинами хвилі (як видно з наведених зразків спектрів); 2) існує слабка кореляція (зв'язок) між різними методами пониження спеклів; 3) системні похибки в вимірюванні контрасту спеклів. Мала різниця між експериментальними і теоретичними даними по контрасту спеклів підтверджує достовірність теорії. Значне зменшення контрасту спеклів показує ефективність такого методу зменшення контрасту.

На рис. 9 показано білий рівномірно засвічений екран, який демонструє результати зниження контрасту спеклів після застосування всіх наявних вище розглянутих методів: сканування, зміни поляризації і використання трьох частково декорельованих променів. Застосування цих трьох методів призвело до зменшення контрасту спеклів до 5% в білому світлі.

Висновки

Розроблено математичний алгоритм для розрахунку контрасту спеклів для лазерного проектора в якому для зменшення контрасту спеклів для створення зображення застосовується декілька частково когерентних пучків. Показано, що оптична схема з двома лазерними пучками забезпечує менші спекли ніж оптична схема з трьома пучками, для випадку, коли скляні пластинки не забезпечують повну декореляцію трьох лазерних пучків. Однак, якщо оптична різниця ходу дифракційних порядків є достатньо великою, щоб отримати повну декореляцію для трьох пучків, оптична схема з трьома променями забезпечує менший контраст спеклів. В цьому випадку, при використанні лазерних пучків 0 і ± 1 дифракційних порядків дифракційних елементів прямокутної форми, метод меншує контраст спеклів до величин близьких до теоретичної межі в (1/sqrt (3)) раз для всіх трьох лазерів (червоний, синій і зелений). При використанні усіх розроблених нами методів зменшення спеклів нам вдалося досягнути 5% контрасту спеклів в білому кольору.

Список використаних джерел

1. Trisnadi J. I., Carlisle C. B., Monteverde R. Overview and applications of Grating Light ValveTM based optical write engines for high-speed digital imaging// Proc. SPIE. –2004. – V. 5348. . –P. 52-64.
2. Kowarz M. W., Brazas J. C., Phalen J. G. Conformal Grating Electromechanical system GEMS) for High-Speed Digital Light Modulation// IEEE, 15th Int. MEMS Conf. Digest. –2002. . – P. 568-573.
3. Yun S. K., Song J., Lee T.-W. [at al.]. Spatial Optical Modulator (SOM): Samsung's Light Modulator for the Next Generation Laser Display// Proc. of SID. – 2006. –V. 29-1. – P. 551-555.
4. Yun S. K. Open hole-based diffractive light modulator// US Patent №er US7206118. –2007.
5. Goodman J. W. Some fundamental properties of speckle// J. Opt. Soc. Am. –1976. – V. 66. – P. 1145-1150.
6. Marom E., Kresic-Juric S., Bergstein L. Analysis of speckle noise in bar-code scanning Systems// J. Opt. Soc. Am. A. –2001. – V. 18. – P. 888-901.
7. Lencina A., Vaveliuk P., Tebaldi M., Bolognini N. Modulated speckle simulations based on the random-walk model// Opt. Lett. –2003. –V. 28 . –P. 1748-1750.
8. Goodman J. W. Speckle with a finite number of steps// Appl. Opt. –2008.– 47. – P. A111-A118.
9. Rodrigues C. M. P. , Pinto J. L. Contrast of polychromatic speckle patterns and its dependence to surface heights distribution// Opt. Eng. –2003. – V. 42. –P. 1699-1703.
10. Furukawa A., Ohse N., Sato Y. [at al.] Effective speckle reduction in laser projection displays// Proc. of SPIE. –2008. –V. 6911. –P. 69110T.
11. Wang L., Tschudi T., Boeddinghaus M. [at al.]. Speckle reduction in laser projections with ultrasonic waves// Opt. Eng. –2000. –V. 39. – P. 1659-1664.
12. Wang L., Tschudi T., Halldorsson T. [at al.]. Speckle reduction in laser projection systems by diffractive optical elements// Appl. Opt. –1998. –V. 37. – P. 1770-1775.
13. Trisnadi J. I. Hadamard speckle contrast reduction// Opt. Lett. –2004. – V. 29, . – P. 11-13.
14. Yurlov V., Lapchuk A., Yun S. K. [at al.]. Speckle suppression in scanning laser display// Appl. Opt. –2008 –V. 47 . –P. 179-187.
15. Rawson E. G., Nafarrate A. B., Norton R. E., [at al.]. Speckle-free rear-projection screen using two close screens in slow relative motion// J. Opt. Soc. Am. –1976. –V. 66. – P. 1290-1294.
16. Trisnadi J. I. Method and apparatus for reducing laser speckle using polarization average// US Patent № US6956878. – 2005.
17. Dainty J. C., Ennos A. E., Francon M. [at al.]. Laser speckle and related phenomena // New York: Springler-Verlag. –1975. –276 P.
18. Goodman J. W. Speckle Phenomena in Optics: Theory and application// New York:Roberts and Company Publishers. – 2007. –Chap. 5. – P. 132-157.
19. Chovina A., Garriguea P., Pecastaingsb G. Microarrays of near-field optical probes with adjustable dimensions// Ultramicroscopy. – 2006. – V. 106. – P. 57–65.
20. Riechert G., Verschaffelt F., Peeters M. Speckle characteristics of a broad-area VCSEL in the incoherent emission regime // Opt. Communications. – 2008. – V. 281, N17. – P. 4424 – 4431.

УДК 681.324

СИМОНЕНКО В.П.

МАТЕМАТИЧЕСКАЯ ПОСТАНОВКА ЗАДАЧИ ДИНАМИЧЕСКОГО РАСПРЕДЕЛЕНИЯ РАБОТ В GRID СИСТЕМАХ И ОЦЕНКИ КАЧЕСТВА РЕШЕНИЯ

В статье рассматривается общая математическая модель динамического планирования в рас пределенной, неоднородной GRID системе. Показано, что назначение задачи на вычислительный ресурс сводится к проблеме поиска максимального паросочетания в двудольном графе.

This paper presents a general mathematical model of dynamic scheduler for distributed heterogeneous GRID system. It is shown that searching of computational resource for a task can be solved as maximum matching problem for bipartite graph.

Введение

Распределения задач по ресурсам в GRID [1] системе является одной из наиболее сложных задач организации распределенных вычислений. Сложность задачи распределения или динамического планирования обусловлено неоднородностью как объекта распределения, так и неоднородностью распределяемых задач. Наиболее известные планировщики или диспетчеры задач (заданий) для GRID систем Platform LSF, Windows HPC Server 2008, PBS, Condor, SGE , LoadLever , MOSIX и внешний планировщик MAUI [2-11] предназначены для оптимизации распределения потока задач (зада-ний) на ресурсы системы. Следует отметить, что если учитывать свойство неоднородности GRID системы, то такое рас пределение не всегда приводит к равномерной загрузке ресурсов [12] и требует применения нового класса пространственных планировщиков, учитывающих и приоритетность задач и неоднородность вычислительной системы.

Постановка задачи

В неоднородной системе распределенной обработки данных (GRID), состоящей из N ресурсов, на момент времени распределения τ имеются N_τ свободных ресурсов и M независимых, готовых к выполнению заданий [1].

– *Система ресурсов* задана графом системы $G_R=(V_R,E_R,W_{VR},W_{ER})$, где:

- Множество вершин $V_R=\{R_1, R_2,..., R_N\}$, каждый элемент которого представляет один из N ресурсов системы и $R_i \in N|$ (множество натуральных чисел), i=1..N.
- Множество дуг $E_R=\{E_1, E_2,.., E_d\}$, каждый элемент которого определяют связи между двумя ресурсами $E_i=\{R_i,R_j\}$, где $R_i,R_j \in V_R$ и $0 \le d \le N^2$.
- Множество весов вершин $W_{VR}=\{WVR_1,WVR_2,...,WVR_N\}$, где $WVR_i=\{RE_i, RT_i\}$. Для $\forall i=1..N$, $RE_i \in \Re^+$ (множество положительных действительных чисел) есть характеристика ресурса R_i, $RT_i \in \{0$ и $\Re^+\}$ – состояния ресурсов.
- Множество весов дуг $W_{ER}=\{WER_1, WER_2,.., WER_p\}$. Это множество можно представить в виде некоторой матрицы $RC=RC[i,j] \in \Re^+$, где i=1..N и j=1..N.

– *Поток M заданий*, задан множеством $V_J=\{Job_1, Job_2,..., Job_M\}$, каждый элемент которого представляет одно из M заданий и $Job_i=\{JN_i, JE_i, JL_i, JM_i, JP_i\}$, $\forall i=1..N$:

- $JN_i \in N|$ – номер задания;
- $JE_i \in \Re^+$ – объем работы задания i;
- $JL_i=\{(R^1, \varphi_1),.., (R^q, \varphi_q)\}$, где $R^l \in V_R$ – ресурс, с которым данное задание требует обмена данными, $\varphi_1 \in \Re^+$ – объем передачи, l=1..q, $q \in N|$;
- $JM_i=\{0$ или $R^i\}$ – маска задания, где $R^i \in V_R$ – номер ресурса, на котором возможно или желательно выполнять данное задание;
- $JP_i \in \Re^+$ – приоритет данного задания.

Определение 1: Г есть отображение множества заданий $V_J=\{Job_1, Job_2,.., Job_M\}$ на множество ресурсов $V_R=\{R_1, R_2,.., R_N\}$ графа системы $G_R=(V_R,E_R,W_{VR},W_{ER})$, если результат отображения $\Gamma(V_J,V_R)$ есть некоторое множество A: $A=\{a_1, a_2,.., a_n\}$, где $a_i=(R^i, J^i)$, $R^i \in V_R$, $J^i \in V_J$, i=1..n, $n \in N|$.

Обозначим $AR=\{R^1, R^2,.., R^n\}$, $AJ=\{J^1, J^2,..,$

$J^n\}$. Таким образом, $|A|=|AR|\cap|AJ|$, $AR\subseteq V_R$, $AJ\subseteq V_J$.

Определение 2: отображение Г есть ***распределение*** заданий V_J на ресурсы V_R, если его результат $\Gamma(V_J,V_R)=A$, где $A=\{(R^1, J^1), (R^2, J^2),.., (R^n,J^n)\}$ удовлетворяет следующему условию: для $\forall i=1..n$, $R^i\notin AR\backslash R^i$, $J^i\in AJ\backslash J^i$, где $AR=\{R^1, R^2,.., R^n\}$, $AJ=\{J^1, J^2,.., J^n\}$. ***Размером*** данного распределения $\Gamma(V_J,V_R)$ является число n. Тогда $\Gamma(V_J,V_R)\rightarrow A$, $n=|A|$.

Определение 3: результат распределения заданий на ресурсы $A=\Gamma(V_J,V_R)$ называем ***расписанием*** для данного распределения Г. Пара $a_i=(R^i, J^i)$, i=1..n, называется ***назначением*** задания $J^i\in V_J$ на ресурс $R^i\in V_R$.

Определение 4: пусть $X=\{A^1, A^2,.., A^z\}$, $z\in N|$ — множество результатов всех возможных распределений для множества заданий V_J и для множества ресурсов V_R. Тогда $\Gamma(V_J,V_R)\equiv X$. Распределение заданий на ресурсы $\Gamma(V_J,V_R)\rightarrow A^*$ есть ***максимальное распределение*** для данных множества заданий V_J и множества ресурсов V_R если:

1) $n^*=|A^*|$;
2) $n^*=\max\{|A^1|, |A^2|,.., |A^z|\}$.

Определение 5: пусть Δ есть некоторая функция от назначения $a_s=(R^s, J^s)$ (то есть назначения задания J^s на ресурс R^s, $R^s\in V_R$ и $J^s\in V_J$). Тогда $\Delta(a_s)=\Phi$ или $\Phi=\Delta(R^s, J^s)$ и $\Phi_i=\Delta(a_i)=\Delta(R^i, J^i)$, где i=1..n, назовем ***весом назначения*** $a_i=(R^i, J^i)$ по Δ.

Определение 6: сумму весов всех назначений $\{a_1, a_2,.., a_n\}$ назовем ***весом D(A) расписания*** A.

То есть: $D(A)=\sum_{i=1}^{n}\Delta(a_i)$.

Определение 7: пусть $Xm=\{A_1, A_2,.., A_m\}$, $m\in N|$, есть множество всех максимальных распределений для множества заданий V_J и для множества ресурсов V_R. Тогда расписание $A^*=\Gamma(V_J,V_R)$ – ***оптимальное расписание*** распределения (заданий V_J на ресурсы V_R) Г по измерению Δ, если A* удовлетворяет следующим условиям:

1) $A^*=\{(R^1, J^1), (R^2, J^2),.., (R^n, J^n)\}$ является результатом максимального распределения для данных множества заданий V_J и множества ресурсов V_R , то есть $|A^*|=\max\{|A^1|, |A^2|,.., |A^z|\}$ (определение 5);

2) Вес расписания $A^*=\{a_1, a_2,.., a_n\}$ максимален из $Xm=\{A_1, A_2,.., A_m\}$, то есть:

$$D(A^*)=\sum_{i=1}^{n}\Delta(a_i^*)=\max\{D(A_1),$$

$$D(A_2),...,D(A_m)\}=\max_{j=1}^{m}\{D(A_j)\}$$

Требование: нужно найти ***оптимальное*** (максимальное по весу заданной функции Δ) ***расписание*** $A=\{(R^1, J^1), (R^2, J^2),.., (R^n, J^n)\}$, $n\in N|$ максимального распределения Г (по определению 7) для N_τ свободных ресурсов (V_R) и M готовых к выполнению заданий (V_J).

– Общая схема решения

Определим *модель* решения для задачи оптимизации и распределения (математическая постановка которой представлена в [2,3]) на основе модели оптимизации и распределения, представленной в [4,5].

Решение данной задачи для N_τ ресурсов $V_R=\{R_1, R_2,.., R_N\}$ и M заданий $V_J=\{J_1, J_2,.., J_M\}$ состоит из следующих этапов:

1 Определение функции Δ весов назначения. Определяются веса $\delta_{i,j}$ ($i=1..N_\tau$, $j=1..M$) всех возможных назначений по функции Δ.

2 Поиск оптимального расписания $A=\{a_1, a_2,.., a_n\}$, где $a_i=(R^i, J^i)$, $R^i\in V_R$, $J^i\in V_J$, $i=1..n$, $n\in N|$, которое удовлетворяет условиям определения 7 и весовым значениям, определенным на первом этапе.

– Определение функции измерения качества решения

При оптимизации и распределении, функцию Δ для измерения веса назначения задания J_j на ресурс R_i ($R_i\in V_R$ и $J_j\in V_J$), можно определить следующим образом:

$$\Delta(R_i,J_j)=\delta_{i,j}=$$

$$\prod_{k=1}^{K}P_k^{i,j}\times\prod_{x=1}^{H}C_x^{i,j}\times\sum_{y=1}^{G}L_yO_y^{i,j} \quad (1)$$

Где,

- $\prod_{k=1}^{K}P_k^{i,j}$ – величина приоритета назначения (R_i, J_j). Она вычисляется путем умножения величин всех K приоритетов $P_k^{i,j}\in\Re^+$ не только заданий, но и ресурсов. В приоритете учитываются разные факторы: время ожидания заданий, работоспособность ресурсов и т.д.).
- $\prod_{x=1}^{H}C_x^{i,j}$ – результат анализа H обязательных требований, $C_x^{i,j}$ определяет степень выполнения обязательного требования x для назна-

чения задания J_j на ресурс R_i, $C_x^{i,j} \in \{0,1\}$. Например, требования наличия каналов передачи, объема требуемой памяти, наличия программ, данных и т.д. $C_x^{i,j}=1$, если ресурс R_i полностью удовлетворяет требованиям задания J_j, $C_x^{i,j}=0$ в противном случае.

- $\sum_{y=1}^{G} L_y O_y^{i,j}$ – результат анализа G оптимизирующих требований, где $O_y^{i,j} \in \Re^+$ и $\mathbf{0} \le O_y^{i,j} \le \mathbf{1}$ – степень выполнения оптимизирующего требования **y** назначения задания J_j на ресурс R_i; $L_y \in \Re^+$ и $L^d \le L_y \le L^u$ – весовой коэффициент оптимизирующего требования y.

В предложенной системе представлений исходной информации имеем :

- $\prod_{k=1}^{K} P_k^{i,j}$ вычисляется с помощью приоритета JP_j задания J_j из выражения:

$$\prod_{k=1}^{K} P_k^{i,j} = \mu_i \times \rho_j,$$

где $\rho_j = JP_j = 1/Tw_j$ (Tw_j – время ожидания задания J_j в системе),

$$\mu_i = \begin{cases} M^o, если R_i = J, M_j = R^* \\ 1, если R_i = J, M_j \in \{0, R^*\} \end{cases}$$

(μ_i – маска задания).

- $\prod_{x=1}^{H} C_x^{i,j}$ вычисляется с помощью сравнения требований по коммуникациям задания $JL_j=\{(R^1, \varphi_1),..,(R^q,\varphi_q)\}$ с множеством дуг графа системы ресурсов $E_R=\{E_1, E_2,.., E_d\}$:
для $\forall l=1..q$: $CC_l^{i,j} = 1$, если $(R_i, R^l) \in E_R$;

$CC_l^{i,j} = 0$, если $(R_i, R^l) \notin E_R$;

T.e. $\prod_{x=1}^{H} C_x^{i,j} = C^{i,j} = \prod_{l=1}^{q} CC_l^{i,j}$.

$\sum_{y=1}^{G} L_y O_y^{i,j}$ вычисляется как сумма обратных величин времени выполнения $Te_{i,j}$ и времени, затрачиваемом на коммуникации $Tc_{i,j}$.

Коэффициент производительности ресурса $RE_i = k_i$ определяется из WVR_i, объем работы задания $JE_j=\varepsilon_j$ – из матрицы весов дуг графа системы ресурсов $RC[k,l]=\beta_{k,l}$, где k=1..N и l=1..N, объемы требований заданий по коммутациям из $JL_i=\{(R^1, \varphi_1),..,(R^q, \varphi_q)\}$.

Тогда $Te_{i,j}$ и $Tc_{i,j}$. вычисляются из следующих выражений:

$$Te_{i,j} = \varepsilon_j * k_i \; ; \; Tc_{i,j} = \sum_{l=1}^{q} (\varphi_l * \beta_{i,l}).$$

Таким образом, имеем:

$$\sum_{y=1}^{G} L_y O_y^{i,j} = 1/Te_{i,j} + 1/Tc_{i,j} =$$

$$1 / (\varepsilon_j * k_i) + 1 / \sum_{l=1}^{q} (\varphi_l * \beta_{i,l})$$

Из выражения (1) имеем:

$$\Delta(R_i,J_j)=\delta_{i,j}=(\mu_i \times \rho_j)*C^{i,j} *$$

$$(1 / (\varepsilon_j * k_i) + 1 / \sum_{l=1}^{q} (\varphi_l * \beta_{i,l})).$$

Очевидно, что $\delta_{i,j} \ge 0$ для $\forall i=1..N_\tau$, $j=1..M_\tau$. Поэтому $\inf(\Delta(R_i, J_j)) = 0$.

В случае отсутствия связи между ресурсами i и l, RC[i,l] получает такое значение $\beta_{i,l}$, что $Tc_{i,,j} = \sum_{l=p}^{q} (\varphi_l * \beta_{i,l}) > \lambda_0$, где λ_0 некоторое заданное число. Число λ_0 есть порог для определения существования связи между двумя ресурсами. Время выполнения $Te_{i,j}$ имеет некоторую нижнюю границу $\mathbf{T^0}$.

Верхняя граница диапазона изменения $\delta_{i,j}$ определяется следующим образом:

$$\sup(\Delta(R_i,J_j)) =$$

$$(\mu_{Oj} \times \rho_{Oj}) \times [1/T^O + 1/\lambda_O] = \delta_{max}.$$

Определенные значения $\delta_{i,j}$ для $i=1..N_\tau$, $j=1..M_\tau$ хранятся в матрице $JR[1..N_\tau, 1..M_\tau]$.

– Определение оптимального распределения

Множество N_τ ресурсов $V_R=\{R_1, R_2,.., R_{N\tau}\}$ и M заданий $V_J=\{J_1, J_2,..,J_M\}$ можно представлять как множество вершин некоторого графа G. Тогда множество неориентированных дуг $E =\{E_1, E_2,.., E_d\}$ между вершинами графа G соответствует множеству возможных назначений заданий J* на ресурсы R*. Исходная информация при такой постановке представляется в виде матрицы связности или двудольного графа (пример графа для 6-и ресурсов и 6-и заданий приведен на рис. 1). Дуга $E_k=\{R_i, J_j\}$, где $R_i \in V_R$ и $J_j \in V_J$, k=1..d, $0 \le d \le N_\tau \times M$, между вершинами J_j и R_i отсутствует только тогда, когда назначение задания J_j на ресурсе R_i является *"невозможным"*, то есть когда $\delta_{i,j} \le \delta_0$, где δ_0 есть некоторое заданное число (в данном примере δ_0=1).

ЗАДАНИЯ

РЕСУРСЫ	№	1	2	3	4	5	6
	1	3.5	0.2	7.1	0.6	6.1	0.1
	2	0	4.1	2.1	5.3	8.7	3.3
	3	2.1	5.9	0.3	0	7.2	0.7
	4	4.5	3.6	5.1	0.8	3.1	0
	5	8.1	7.1	0	0	4.9	0.4
	6	0.9	0	2.1	6.8	0	8.1

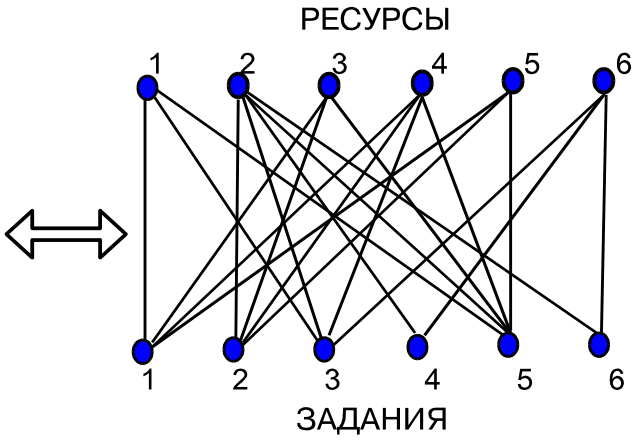


Рис.1.

Выполнение второго этапа распределения представляет собой *задачу назначения*. Существует несколько методов для решения задачи назначения для взвешенного двудольного графа

$G=(V_R,V_J, E, WE)$:

Где: $V_R=\{R_1, R_2,.., R_{N\tau}\}$ и $V_J=\{J_1, J_2,.., J_M\}$,

$E =\{E_1, E_2,.., E_d\}$, $E_k=\{R^*,J^*\}$, где $R^*\in V_R$ и $J^*\in V_J$,

$k=1..d$, $0\le d\le N_\tau\times M$.

$WE=\{WE_1, WE_2,.., WE_d\}$, $WE_k=\Delta(E_k)$,

Где: $k=1..d$, $0\le d\le N_\tau\times M$.

Решение задачи назначения для графа размером $N_\tau\times M$, где $N_\tau\ne M$ приводится к решению задачи назначения для графа размером $N\times N$, где: $N=\max\{N_\tau; M\}$

В случае планирования для однородной GRID решение задачи назначения для взвешенного графа G приводится к решению задачи назначения для невзвешенного графа G', полученного из графа G снятием весов всех дуг.

Задача назначения в такой постановке решается во многих приложениях [4,5,8,10,11]. На выбор метода и алгоритма решения влияет временная сложность, т.к. время решения задач планирования, особенно при динамическом планировании, является основным критерием.

Выделим два наиболее часто используемых подхода к решению данной задачи.

- поиск максимального потока в сети[6,7];
- поиск максимального паросочетания методом увеличивающего, чередующегося пути [7,9].

При динамическом планировании в GRID системах задача планирования сводится к поиску максимального паросочетания во взвешенном двудольном графе. Анализ методов ее решения показывает, что при решении задачи поиска максимального паросочетания в взвешенном двудольном графе используется поиск максимального паросочетания в невзвешенном двудольном графе. Поэтому задача поиска максимального паросочетания в невзвешенном двудольном графе. является ключевой и требует специального изучения, т.к. наиболее известные алгоритмы имеют временную сложность, ограничивающую их практическое использование.

Литература

1. Метод опережающего планирования для грид [Электронный ресурс] / В. Н. Коваленко, Е. И. Коваленко, Д. А. Корягин, Э. З. Любимский // Препринт ИПМ.2005. – № 112. – Режим доступа: http://www.keldysh.ru/ papers/2005/prep112/prep2005_112.html.
2. 5. Platform LSF 7 Update 6. An Overview of New Features for Platform LSF Administrators [Электронный ресурс] / Официальный сайт компании Platform Computing Corporation – 2009. – Режим доступа: http://www.platform.com/workload-management/whatsnew_lsf7u6.pdf.
3. Microsoft Windows Compute Cluster Server 2003 [Электронный ресурс] // Руководство пользователя – 2006. Режим доступа: https://msdb.ru/Downloads/ WindowsServer2003/CCS/CCS2003 Guide_Rus.pdf.
4. TORQUE Resource Manager Guide [Электронный ресурс] // Официальный сайт компании Cluster Resources Inc. – 2009 Режим доступа: http://www.clusterresources.com/products/torque-resource-manager.php.

5. PBS Works [Электронный ресурс] // Официальный сайт компании Altair Engineering, Inc. – Режим доступа: – 2006 http://www.pbsworks.com/.
6. Commercial-grade HPC workload and resource management [Электронный ресурс] // Официальный сайт компании Altair Engineering, Inc. – Режим доступа: – 2008 http:/ ww.pbsgridworks. com/Product.aspx? id=1.
7. Resource Library [Электронный ресурс] // Официальный сайт компании Altair Engineering, Inc. – Режим доступа: – 2007 http://www.pbsgridworks.com/ResLibSearchResult.aspx? Keywords = openpbs&industry=All&product_service=All&category=Free%20Software%20Downloads&order_by=date_ submitted.
8. What is Condor? [Электронный ресурс] // Официальный сайт продукта Condor - Режим доступа– 2006: http://www.cs.wisc.edu/condor/description.html.
9. Sun Grid Engine 6.2 Update 5 [Электронный ресурс] //Официальный сайт компании Oracle Corp. – Режим доступа: – 2009 http://www.sun.com/software/sge/index.xml.
10. IBM Tivoli Workload Scheduler LoadLeveler [Электронный ресурс] // Официальный сайт компании «Интерфейс» – 2007. – Режим доступа: http://www.interface.ru/home.asp?artId=6283. Maui Scheduler Administrator's Guide [Электронный ресурс] // Официальный сайт компании Cluster ResourcesInc. – Режим доступа: – 2008 http:// www. clusterresources. com/products /maui/ docs/index.shtml. – Загл. с экрана.
11. Moab Workload Manager [Электронный ресурс] // Официальный сайт компании Cluster Resources Inc. – Режим доступа: – 2008 http://www.clusterresources.com/products/moab-cluster-suite/ workload-manager.php.
12 Симоненко А.В. Выбор стратегии пространственного планирования в параллельных вычислительных системах. Вісник НТУУ «КПІ» Інформатика, управління та обчислювальна техника, Київ 2001 р. № 35.- с. 104-108.
13 Kaufmann A., Introduction a la combinatorique en vue des aplications, Dunod, Paris.(1968) .
14 Papadimitry X., Stayglitsh K., Combinatory optimization, algorithm and complexity, Moscow: Mir (1985).
15 Berge C. , Theorie des graphes et ses application. Dunod, Paris (1958).

УДК 681.324

СИМНЕНКО В.П.,
СИМОНЕНКО А.В.

ДИНАМИЧЕСКОЕ РАСПРЕДЕЛЕНИЕ РАБОТ ПО РЕСУРСАМ НЕОДНОРОДНОЙ СИСТЕМЕ С ОГРАНИЧЕНИЯМИ РЕАЛЬНОГО ВРЕМЕНИ

В работе предложен алгоритм динамического распределения задач в неоднородной системе в реальном времени. Представлен метод преобразования исходной информации, позволяющая решать задачу планирования известными методами.

This paper presents the dynamic tasks scheduling algorithm for heterogeneous systems in real time. The method for initial information transformation is proposed which allows solving scheduling problem by known methods.

Введение

Планирование в *реальном* времени характеризуется решением следующей задачи – определением плана решения совокупности задач с заданным временем исполнения и ограничениями по времени выхода задач из системы. Система планирования должна обеспечить выполнение требований по минимизации суммарного времени отклонения реального выхода задач из системы (выполнения сроков решения) от сходных временных ограничений при полном соблюдении порядка следования работ. В общем случае эта задача относится к классу NP-полных [5,6].

В большинстве случаях разработчики систем планирования реального времени используют статические алгоритмы и заранее определяют максимальный список заданий, допустив наихудший случай для получения статической управляющей таблицы (плана). Этот план фиксируется и используется для безусловного исполнения в динамическом режиме со слідующими допущениями:

- все временные ограничения остаются неизменными на время выполнения плана;
- все задачи вкладываются в свое критическое время.

В других случаях при помощи приемов стати-ческого планирования создается статический список приоритетов для использования в динамическом режиме во время диспетчеризации самих работ.

Если система реального времени работает только в динамическом режиме, то использование соглашений статического планирования (где все известно априори) недопустимо. В этом случае выбирается один из возможных алгоритмов составления расписания и тщательно анализируется на применимость его в ожидаемом динамическом окружении. Как правило, применяются алгоритмы, использующие планирование по спискам и приоритетное обслуживание [1-4]. В данной работе предлагается использовать модифицированный Венгерский алгоритм для решения задачи динамического распределения заявок по ресурсам неоднородной GRID системы с ограничениями реального времени, имеющий меньшую временную сложность чем известные. В работе предлагается метод преобразования исходной информации позволивший применить математический аппарат поиска максимального паросочетания для задачи составления плана распределения задач по ресурсам с ограничениями реального времени, выполняемый планировщиком нижнего уровня.

Постановка задачи

Рассмотрим общую постановку задачи в реальном времени.

На вход СРОД поступает множество работ. Каждая работа характеризуется тремя временными параметрами:

$T_{вх}^{i}$ – время поступления i-ой входной работы в ВС

$T_{вых}^{i}$ – время выхода i-ой работы из ВС

$T_{раб}^{i}$ – время выполнения i-ой работы в относительных единицах на вычислительном узле GRID, имеющем максимальную производительность GRID имеет множество

ресурсов. Каждый ресурс характеризуется производительностью R^j. Для каждой i-ой работы можно определить время решения i-ой задачи на j-ом ресурсе в относительных единицах.

Для этого определим R_{max} как $R_{max} = \max\{R^1,..,R^n\}$.

Определим относительную призводительность каждого вычислительного узла R^j по отношению к R_{max}. Для этого вычисляем

$$Z^j = \frac{R^j}{R_{мах}},\ j=1..n.$$

Имея относительную производительность каждого узла, можно определить отношение работа-ресурс с учетом времени выполнения работы и производительности каждого узла GRID системы.

Для этого сформируем матрицу связности (МС), где каждый элемент определяет относительное время выполнения каждой i-ой работы на j-ом ресурсе с учетом его призводительности $MC[i,j] = \frac{T^i_{раб}}{Z^j}$.

Ввиду того, что $1 \geq Z^j > 0$, то относительное время выполнения каждой работы на ресурсах будет определяться значением MC[i,j]. В связи с тем, что задача, выполняемая в реальном времени, должна завершиться до заданного $T^i_{вых}$, то необходимо вычислить $\Delta_t^{i,r}$ из выражения:

$$\Delta_t^{i,j} = T^i_{вых} - T^i_{вх} - MC[i,j].$$

Так как $(T^i_{вых} - T^i_{вх}) > 0$,

а $(T^i_{вых} - T^i_{вх}) > MC[i,j] > (T^i_{вых} - T^i_{вх})$, то $\Delta_t^{i,j}$ может принимать как положительные так и отрицательные значения. Положительные значения $\Delta_t^{i,j}$ означают, что работа i, назначенная на ресурс j, будет завершена на время $\Delta_t^{i,j}$ до назначенного срока, а отрицательные значения означают, что выход заявки из системы произойдет позже назначенного срока на время $\Delta_t^{i,j}$ (рис. 1).

При вычислении $\Delta_t^{i,j}$ следует учитывать, что на момент планирования ($T_{план}$) все заявки имеют одно и то же базовое время начала планирования независимо от времени прихода заявок в систему. Кроме этого следует учитывать и время работы планидовщика, т.е. время планирования ε.

Поэтому можно принять $T^i_{вх} = T_{план} + \varepsilon$.

	РЕСУРСЫ						
		1	2	3	4	5	6
	1	1	3	-2	-3	4	5
А	2	-2	0	-5	-6	-4	-2
Б	3	3	6	1	7	9	1
О	4	1	-3	-4	-5	-1	-1
Т	5	-1	4	-4	1	-5	-1
Ы	6	8	1	2	4	3	2

Рис. 1. Исходная матрица

В соответствии с требованиями режима реального времени значение $\sum_{z=1}^{n} \left|\Delta_t^{i,j}\right|$ должно быть больше или равно 0. При невозможности получения варианта распределения с выполнением требования по времени выхода для всех заявок и при включении в распределение значений с $\Delta_t^{i,j} < 0$ их сумма должна быть минимальной.

В такой постановке решение задачи назначения i-ой работы на j-ый ресурс сводится к задаче поиска максимального паросочетания в взвешенном двудольном графе и решатся Венгерским алгоритмом или алгоритмом направленного поиска. Следует отметить, что выполнение поиска варианта рас пределения выполняется для неоднородной вычислительной системы (степень неоднородности не ниже 0.7) и поиск решения выполняется в разреженной бинарной матрице связности. При решении задачи поиска максимального паросочетания можно применить адаптивный алгоритм, временная сложность которого зависит от коэффициента заполнения матрицы связности двудольного графа, а при коэффициенте заполнения менее 0.7 вре-

менная сложность не превышает $O(n^{1.5}\log n)$ [41].

Для поставленной задачи требуется модификАция формирования исходных данных и базовой (начальной) области поиска. Рассмотрим процедуры формирования исходных данных и области поиска на примере решения задачи назначения для 6-ти задач на 6-ти процессорах. После обработки исходных временных ограничений и учета производительности каждого процессора получим исходную матрицу $\Delta_t^{i,j}$ (рис. 3.12). Информация в матрице $\Delta_t^{i,j}$ учитывает только объем работы и относительное время ее выполнения с учетом производительности вычислительных узлов. Однако неоднородность СРОД обуславливает не обходимость учета индивидуальных характеристик каждого вычислительного узла (наличия достаточной оперативной памяти, необходимых данных в узле, необходимых программ для выполнения задания и т.д.), связанных с возможностью выполнения задания. С учетом этих критериев необходима оценка принципиальной возможности выполнения каждой работы в каждом узле. Для этого формируется матрица проверки конфликтности назначений каждой i-ой работы в j-ый узел:

$$Q_{i,j} = \prod_{x=1}^{p} C_x^{i,j} \text{ (рис. 2),}$$

где: $C_x^{i,j}$ – степень выполнения x обязательного требования для назначения i-ой заявки на j-ый ресурс;

На следующем этапе формирования исходной информации необходимо выполнить фильтрацию элементов матрицы $\Delta_t^{i,j}$ в соответствии со значениями элементов матрицы $Q_{i,j}$. В результате получаем новую матрицу $\underline{\Delta_t^{i,j}}$, где символом ∞ обозначены назначения, которые не могут рассматриваться в качестве возможных (рис. 3).

	1	2	3	4	5	6
1	1	1	0	0	1	0
2	1	1	1	0	1	1
3	0	1	0	1	1	0
4	0	1	1	1	1	0
5	1	0	0	1	1	1
6	1	1	0	1	1	1

Рис. 2 Матрица проверки конфликтных назначений $\Delta_t^{i,j}$

	1	2	3	4	5	6
1	1	3	∞	∞	4	∞
2	-2	0	-5	∞	-4	-2
3	∞	6	∞	7	9	∞
4	∞	-3	-4	-5	-1	∞
5	-1	∞	∞	1	-5	-1
6	8	1	∞	4	3	2

Рис. 3 Матрица $\underline{\Delta_t^{i,j}}$ ***после фильтрации назначений в матрице*** $\Delta_t^{i,j}$

Для формирования плана распределения задач по процессорам, в соответствии с требованиями Венгерского алгоритма и ограничениям накладываемыми нашей постановкой задачи, требуется сформировать начальную (исходную) зону поиска. Для этого выполним следующие действия. Ввиду того, что положительные значения элементов матрицы $\underline{\Delta_t^{i,j}}$ соответствуют назначениям, которые могут быть, безусловно, включены в решение, т.к. любое назначение, соответствующее координате i, j, не противоречит условиям временных ограничений, то всем положительным элементам присваиваем значения 0, а отрицательным элементам поменяем знак на положительный. В результате получаем новую матрицу $\underline{\Delta_t^{i,j}}$ (рис. 4). Дальнейшие действия по формированию исходной области поиска выполняем в точном соответствии с Венгерским алгоритмом. Из каждого элемента столбца матрицы $\underline{\Delta_t^{i,j}}$ вычитается наименьший элемент этого столбца. $\underline{\Delta_t^{i,j}}(1) = \underline{\Delta_t^{i,j}}(0) - \min_i \Delta_t^{i,j}$).

Из каждого элемента строки полученной матрицы $\underline{\Delta_t^{i,j}}$ вычитается наименьший элемент этой строки. $\underline{\Delta_t^{i,j}}(2) = \underline{\Delta_t^{i,j}}(1) - \min_j \Delta_t^{i,j}$;

В результате выполненных действий в каждой строке и каждом столбце имеем нулевой элемент (рис. 5).

	1	2	3	4	5	6
1	0	0	∞	∞	0	∞
2	2	0	5	∞	4	2
3	∞	0	∞	0	0	∞
4	∞	3	4	5	1	∞
5	1	∞	∞	0	5	1
6	0	0	∞	0	0	0

Рис. 4. Промежуточна матрица

	1	2	3	4	5	6
1	0	0	∞	∞	0	∞
2	2	0	2	∞	4	2
3	∞	0	∞	0	0	∞
4	∞	2	0	4	0	∞
5	1	∞	∞	0	5	1
6	0	0	∞	0	0	0

Рис. 5. Исходная матрица поиска

Для поиска решения из исходной матрицы поиска $\underline{\Delta_t^{i,j}}$ (2) выделяем нулевые элементы и формируем исходную матрицу для поиска максимального паросочетания (OPR) (рис.6). Как уже упоминалось, для получения решения используется один из алгоритмов поиска максимального паросочетания в невзвешенном двудольном графе. Для него необходимо инвертировать значения элементов матрицы OPR. Т.е. заменить в ней все 0-е элементы на 1 и наоборот. В результате получаем матрицу поиска решения PR (рис. 7).

	1	2	3	4	5	6
1	0	0	∞	∞	0	∞
2	2	0	5	∞	4	2
3	∞	0	∞	0	0	∞
4	∞	3	4	5	1	∞
5	1	∞	∞	0	5	1
6	0	0	∞	0	0	0

Рис. 6. Матрица области поиска решения (OPR)

	1	2	3	4	5	6
1	0	0	∞	∞	0	∞
2	2	0	2	∞	4	2
3	∞	0	∞	0	0	∞
4	∞	2	0	4	0	∞
5	1	∞	∞	0	5	1
6	0	0	∞	0	0	0

Рис. 7. Матрица поиска решения (PR)

Для рассматриваемого примера найдено максимальное паросочетание (рис. 8), которое является окончательным планом размещения заявок по процессорам, т.к. мощность полученного распределения (мощность паросочетания) равна размерности решения задачи, т.е. все заявки распределены.

	1	2	3	4	5	6
1	1	1	0	0	1	**1**
2	0	**1**	0	0	0	0
3	0	1	0	1	**1**	0
4	0	0	**1**	0	1	0
5	0	0	0	**1**	0	0
6	**1**	1	0	1	1	1

Рис. 8. План распределения

В том случае, если совершенное паросочетание не получено (мощность полученного паросочетания на равна размерности матрицы PR), необходимо выполнить действия для получения новой области поиска.

Дальнейшие действия определены процедурами Венгерского алгоритма, в результате выполнения которых определяются элементы $\underline{\Delta_t^{i,j}}$, которые могут быть дополнительно включены в зону поиска.

Рассмотрим пример, иллюстрирующий формирование новой зоны поиска.

В качестве исходной примем матрицу, представленную на рис. 9. Поиск максимального паросочетания по отмеченным “0” не дал решения.

	Y_1	Y_2	Y_3	Y_4	Y_5	Y_6
X_1	3	**0**	∞	2	0	**2**
X_2	5	**2**	8	**0**	∞	4
X_3	2	∞	**0**	3	**4**	3
X_4	8	0	**7**	∞	3	1
X_5	**0**	1	0	∞	0	0
X_6	∞	∞	1	3	**0**	6

Рис. 9.

Для данного примера имеем предварительное распределение, отмеченное выделенными нулями. Для дальнейшего поиска решения определяется минимальная опора. (Минимальная опора – минимальное множество линий, содержащих все “0” матрицы). Для этого действуем последовательно:

Отметим в каждой строке, в которой есть решение все нули;

Помечаем знаком “+” каждую строку, не содержащую отмеченных нулей.

Помечаем знаком “+” каждый столбец, содержащий отмеченные нули, какой-либо их помеченных “+” строк.

Помечаем знаком “+” каждую строку, содержащую отмеченный нуль в каком-нибудь столбце, помеченном “+”.

Действия повторяются до тех пор, пока возможно помечать “+” новые строки и столбцы.

Выделим минимальную опору. Для этого отметим все непомеченные “+” строки (в примере X_2, X_3, X_5) и все помеченные “+” столбцы (в примере Y_2, Y_5). В результате получаем помеченную матрицу (рис. 9).

	Y_1	Y_2	Y_3	Y_4	Y_5	Y_6	
X_1	3	**0**	∞	2	0	**2**	+
X_2	5	**2**	8	**0**	∞	4	
X_3	2	∞	**0**	3	**4**	3	
X_4	8	0	**7**	∞	3	1	+
X_5	**0**	1	0	∞	0	0	
X_6	∞	∞	1	3	**0**	6	+
		+			+		

Рис. 10. Выделенная минимальная опора

Затем формируется новая зона поиска. Для этого рассматривается подматрица, образованная плементами, через которую не проходят отмеченные на рис. 9 линии, и возьмем наименьший элемент этой подматрицы. Вычтем это число из элементов всех тех столбцов, через которые не проходят отмеченные линии, и затем прибавим его к элементам всех тех строк, через которые пунктирные линии проходят. В данном примере единица вычитается из элементов столбцов Y_1, Y_3, Y_4, Y_5 и прибавляется затем к єлементам строк X_2, X_3, X_5. В результате выполненных действий изменяется количество 0, определяющих зону поиска решения, и ищется максимальное паросочетание. Вышеописанные действия повторяются до тех пор, пока не будет получено максимальное паросочетание.

Результат этих действий показан на рис. 10.

	Y_1	Y_2	Y_3	Y_4	Y_5	Y_6
X_1	2	**0**	∞	1	0	**1**
X_2	5	**3**	8	**0**	∞	4
X_3	2	∞	**0**	3	**5**	3
X_4	7	0	**6**	∞	3	**0**
X_5	**0**	2	0	∞	1	0
X_6	∞	∞	0	3	**0**	5

Рис. 11. Окончательное решение

Заключение

Предложенный в статье метод преобразования исходной информации позволил привести решение задачи распределения заявок по вычислительным ресурсам к решению классической задачи коминаторной математики – поиска максимального паросочетания во взвешенном двудольном графе. А использование метода и алгоритма адаптивного планирования [7] позволяет значительно уменьшить временную сложность классического ‘венгерского алгоритма’, используемого для поиска максимального пароосочетания во взвешенном дудольном графе.

Литература

1. Зыль С. Операционная система реального времени QNX: от теории к практике. 2-издание. – СПб.: БХВ-Петербург, 2004. – 192 с.: ил. ISBN 5-94157-486-X
2. Зыль С. QNX Momentics. Основы применения. – СПб.: БХВ-Петербург, 2004. – 256 с.: ил. ISBN 5-94157-430-4
3. Кёртен Р. Введение в QNX/Neutrino 2. – СПб.: Петрополис, 2001. – 512 с. ISBN 5-94656-025-9
4. Ослэндер Д. М., Риджли Дж. Р., Рингенберг Дж. Д. Управляющие программы для механических систем: Объектно-ориентированное проектирование систем реального времени. – М.: Бином. Лаборатория знаний, 2004. – 416 с. ISBN 5-94774-097-4
5. Papadimitry X., Stayglitsh K., Combinatory optimization, algorithm and complexity, Moscow: Mir (1985).
6. Berge C. , Theorie des graphes et ses application,. Dunod, Paris (1958).
7. Симоненко А.В. Выбор стратегии пространственного планирования в параллельных вычислительных системах. Вісник НТУУ «КПІ» Інформатика, управління та обчислювальна техника, Київ 2001 р. № 35. – с. 104-108.
8. Симоненко А.В. Сравнительная характеристика методов адаптивного распределения данных в неоднородной вычислительной системе, Вісник НТУУ «КПІ» Інформатика, управління та обчислювальна техника, Київ 2006 р. № 45. – с. 54-60.

УДК 681.324

СИМОНЕНКО А.В.

ЭЛЕМЕНТЫ ТЕОРИИ ПОВЫШЕНИЯ ЭФФЕКТИВНОСТИ РЕШЕНИЯ ЗАДАЧИ ДИНАМИЧЕСКОГО ПЛАНИРОВАНИЯ В GRID СИСТЕМАХ

В статье выполнен анализ влияния количества рёбер в графе на выбор стратегии составления расписания. Приведены определения, теоремы и следствия, являющиеся основой эффективных алгоритмов для динамических планировщиков неоднородных GRID систем.

This paper presents the analysis of influence of number of edges in a graph on time scheduling strategy selection. Definitions, theorems and consequences are given that form a background of effective algorithms for dynamic scheduler for heterogeneous GRID systems.

Введение

Идея объединения компьютеров имеет давнюю историю. Начало положили стандарты и протоколы, позволяющие создавать локальные сети из нескольких компьютеров. Через некоторое время появилась возможность связать множество локальных сетей в одну глобальную – интернет. В рамках локальных сетей были разработаны программные средства, позволяющие использовать суммарные вычислительные мощности машин, принадлежащих одному административному домену. Естественным продолжением развития информационных систем является перенос возможности утилизации вычислительных мощностей удаленных компьютеров с локального уровня на глобальный.

Так зародилась идея новой формы организации вычислительных средств, впослідствии получившая название GRID [1-5], которая позволяет унифицированным образом обводинять различные виды ресурсов в рамках динамически организующейся глобальной среды.

Для обеспечения практической применимости грид должна быть решена проблема обеспечения качества обслуживания (QoS – quality ofservice) пользователей [6]. Качество обслуживания – многоаспектное понятие, включающее: безопасность участвующих в GRID ресурсов и безопасность выполняющихся заданий, надежность и постоянную доступность ресурсов. В конечном счете качество обслуживания должно обеспечивать приемлемое и предсказуемое время выполнения заданий.

В GRID одновременно и независимо друг от друга работают множество пользователей, в то время как базовый слой грид **[3**] образуют протоколы удаленного доступа к ресурсам, которые поддерживают лишь индивидуальные операции отдельных пользователей. Результатом независимой работы пользователей будут локальные перегрузки одних ресурсов, при простое других.

В условиях коллективного характера функционирования GRID необходимым механизмом обеспечения качества обслуживания является планирование, которое, выполняясь в контексте диспетчеризации, то есть автоматического распределения ресурсов при обслуживании запросов, осуществляет координацию разделения ресурсов между заданиями пользователей.

Ключевая идея рассматриваемого подхода заключается в разделении процесса распределения или составления расписания на предварительный анализ исходной информации, определения стратегии поиска решения и поиска варианта решения с использованием результатов анализа. Этап предварительного анализа исходной информации существенно уменьшает общее время решения по сравнению с классическими методами решения.

Постановка задачи

В общем виде требования заявок на захват ресурсов вычислительной системы можно разделить на обязательные Cx, x=1..p, и оптимизирующие Oy, y=1..к. С помощью обязательных $\forall C_x^{i,j} \in \{0,1\}$ требований анализируется принципиальная возможность предоставления i-той заявки j-того ресурса [1]. Оптимизирующие требования $\forall O_y^{i,j} \in [0,1]$ определяют степень предпочтения (приоритет) j-того ресурса

для назначения на него i-той заявки по $O_y^{i,j}$ требованиям. Для определения степени претендования (приоритета) j-того ресурса для назначения на него i-той заявки по всем требованиям можно использовать выражение:

$$Q_{i,j} = \prod_{x=1}^{p} C_x^{i,j} \times \sum_{y=1}^{k} R_y O_y^{i,j} \qquad (1)$$

Где.

$C_x^{i,j}$ – степень выполнения x обязательного требования для назначения i заявки на j ресурс;

$O_y^{i,j}$ – степень выполнения оптимизирующего требования y для назначения i заявки на j ресурс;

R_y – весовой коэффициент оптимизирующего требования y.

Если система планирования учитывает обязательные и оптимизирующие требования, то значение коэффициента претендования в матрице связности вычисляется из выражения (1) и находится в диапазоне $Q_{i,j} \in [0,1]$, а при диспетчеризации, выполняемой с учетом только обязательных требований $Q_{i,j} = \prod_{x=1}^{p} C_x^{i,j}$ и $Q_{i,j} \in \{0,1\}$. В этом случае матрица связности, отображающая претендование заявок на ресурсы, примет булевый вид. Значение "1" означает, что ресурс принципиально подходит для размещения заявки, (имеется достаточный объем памяти, процессор имеет необходимые характеристики, вычислительный узел имеет необходимые программы и данные и т.д.).

Для системы из N ресурсов в какой-то момент времени имеется M работ (M=N). Требования работ на захват ресурсов представлены булевой матрицей связности MC[i,j], i=1..N, j=1..N. Необходимо определить отношение работа-ресурс A={(Vi,Wj)}, Vi∈V={1,2,..N}, Wj∈W={1,2,..N} так, чтобы ∀Vi∉V\Vi è ∀Wj∉W\Wj: {MC[Vi,Wj]=1| Vi≠Wj}. Введем дополнительные условия: ресурс может обслужить только одну заявку; процесс обслуживания заявки не может быть прерван; каждая работа имеет индивидуальные характеристики и может претендовать на захват только некоторых системных ресурсов; нет очереди к ресурсам (отсутствие очередей определяется тем, что на данном уровне планирования планировщик более высокого уровня выбрал из общего потока заявок такое количество заявок, которое соответствует количеству свободных ресурсов); одна работа может быть обслужена только одним ресурсом. "1" в матрице связности MC соответствует паре Работа(i) и Ресурс(j) и соответствует выполнению всех K обязательных требований, предъявляемых к системе обработки соответствующей заявки на этом ресурсе. "0" свидетельствует о невозможности обслуживания. При такой постановке решение задачи распределения заявок по ресурсам сводится к задаче поиска максимального паросочетания в невзвешенном двудольном графе.

Элементы теории ускорения распределения работ по ресурсам в неоднородных GRID системах

В основу наиболее известных алгоритмов [8-11] поиска максимального паросочетания в произвольном графе положены два основных подхода: сведение задачи к поиску максимального потока в сети [8] и поиска увеличивающего пути от свободных вершин [8]. В основу поиска увеличивающего чередующегося пути положена схема Диница [9] и разработанный на его основе алгоритм Хопкрофта-Карпа [11]. Наилучшие известные алгоритмы, реализующие этот подход и алгоритм Хопкрофта-Карпа, или его модификации, имеют полиномиальную теоретическую временную сложность Однако эти алгоритмы и программы их реализующие имеют сложную структуру или предназначены для частных случаев и не обеспечивают приемлемых временных показателей, что существенно ограничивает применение их в системах оперативного диспетчирования в GRID системах. Для уменьшения временной сложности алгоритмов поиска максимального паросочетания некоторые исследователи предлагают распараллеленные алгоритмы. Для алгоритмов, использующих поиск увеличивающего чередующегося пути, правильность выбора начального варианта паросочетания в значительной степени влияет на количество шагов при поиске увеличивающего пути. Многие авторы [1,2,3,] подчеркивают, что уникальные свойства дудольного графа позволяют уменьшить временную сложность алгоритмов. Для выявления особенностей двудольного графа, влияющих на временную сложность, выполнено статистическое исследование программной

модели базового алгоритма Хопкрофта-Карпа с временной сложностью $O(N^{2.5})$. Результаты моделирования приведены на рис.1 для графов размерности 10

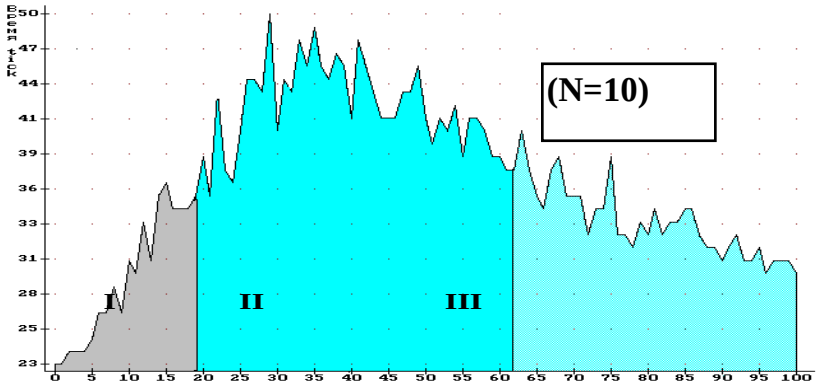


Рис. 1. График зависимости времени решения задачи поиска максимального паросочетания от коэффициента заполнения МС при N=10

Примечание: время вычисления максимального паросочетания вычислялось в относительных временных единицах (тиках), что соответствует нескольким тактам работы процессора.

Как видно из графика на рис. 1,2 время решения задачи зависит от коэффициента заполнения матрицы связности (МС) и размерности (на графиках приведены усредненные временные затраты испытаний с МС с заполненностью от 1% до 100% и 100-а испытаниях в каждой точке при шаге изменения процента заполнения МС равного 1%). Как видно из рис. 1, можно выделить три зоны. Исследования показали зависимость размера этих зон от размерности решения задачи (рис. 2). Значительные различия временных затрат в выделенных зонах обусловило необходимость исследования в них свойств двудольного графа.

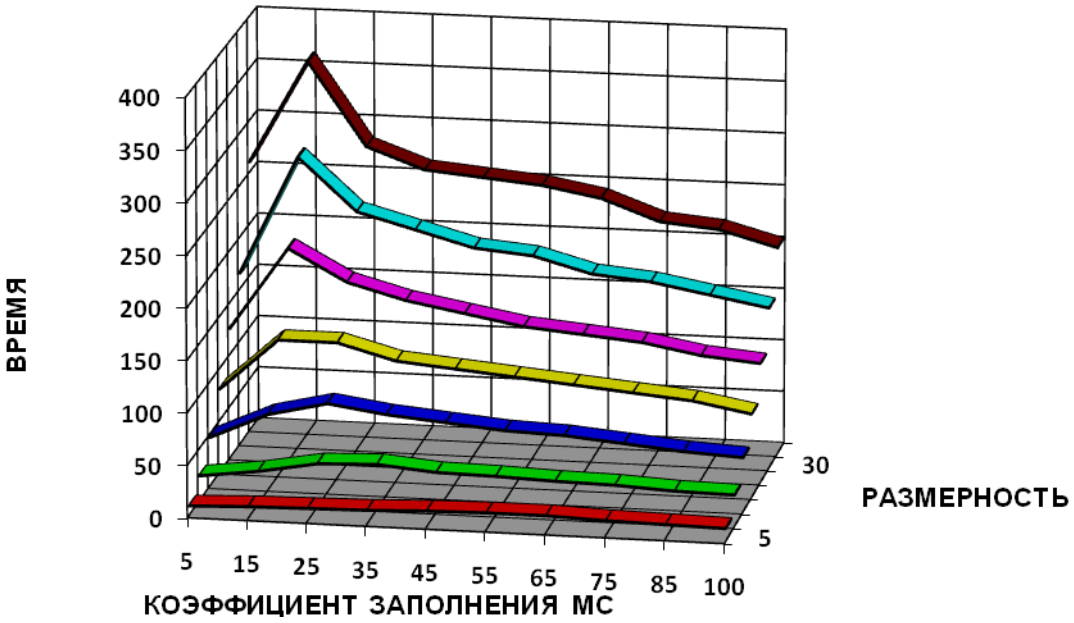


Рис. 2.

Кроме этого, анализ базового алгоритма показал зависимость времени решения задачи от правильности выбора исходного (базового) варианта решения.

Анализ алгоритмов поиска максимального паросочетания, а также анализ процесса поиска решения в выделенных зонах показывает, что наибольшие трудности, виляющие на

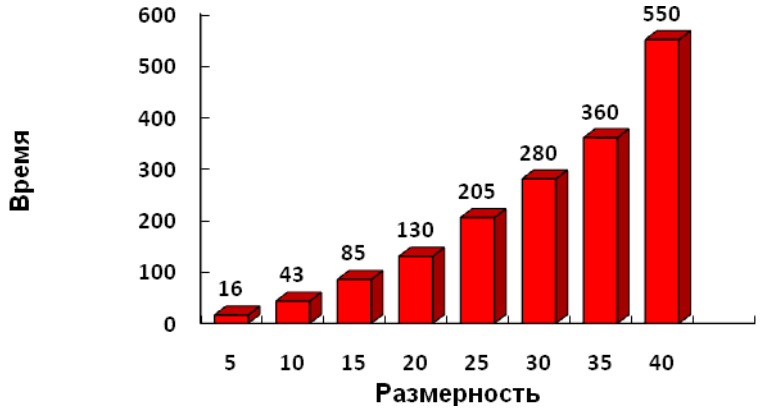


Рис. 3. Временная сложность решения задачи планирования с помощью базового алгоритма Карпа –Хопкрофта ($O(n^{2.5})$)

количество шагов, а отсюда и на время поиска максимального паросочетания, возникают в двудольных невзвешенных графах, в которых перманент МС близок к "1" или равен "0". Эти трудности вызваны тем, что поиск максимального паросочетания основан на центральной теореме Кенига-Холла о существовании паросочетания [8,9] и тереме Бержа [10]. По теоремой Бержа – "*паросочетание М в графе G максимально тогда и только тогда, когда в G не существует увеличивающего пути относительно М"*. Поэтому, все известные алгоритмы предусматривают выполнение попыток поиска увеличивающего пути от свободных вершин после генерирования базового варианта даже в том случае, если этого пути нет, что существенно увеличивает время поиска.

Элементы технологии предварительного анализа структуры двудольного графа

Чтобы упростить решение задачи поиска максимального паросочетания предлагается разделить его на несколько этапов, когда собственно решению предшествует быстрый анализ исходной информации и выработка рекомендаций для ее дальнейшего решения. Добавление дополнительных шагов значительно меньшей временной сложности, чем сам алгоритм, не влияет на теоретическую оценку временной сложности алгоритма в целом, однако, позволяют: уменьшить размерность решения задачи за счет выделения назначений, которые обязательно нужно сделать и выделить назначения, которые делать нельзя и за счет этого избежать лишние проверки на возможность включения их в решение.

Кроме этого, на этапе подготовки исходной информации возможно вычисление мощности максимального паросочетания. Имея численное значение мощности можно избежать поиск увеличивающего пути от свободных вершин при достижении расчетной мощности паросочетания на очередном шаге поиска решения.

Задача назначения требует определения условий возможности ее решения, т.е. возможности полного распределения всех заявок по ресурсам.

Необходимые условия существования пол.-ного решения можно сформулировать слідующим образом

$$\sum_{j=1}^{N} MC[i, j] \neq 0,\ i=1..M \qquad (2),$$

$$\sum_{i=1}^{M} MC[i, j] \neq 0,\ j=1..N \qquad (3).$$

Анализ появления конфликтных ситуаций и условий их возникновения показывает, что выполнение условий 2, 3 является необходимым для получения полного варианта размещения, но недостаточным, так как условия не оценивают взаимосвязи возможных мест размещения заявок и влияния возможного назначения на последующие.

При выполнении дальнейших исследований используются следующие определения:

Определение 1: задан невзвешенный двудольный граф G=(V,E), где $V=\{V_R,V_J\}$, $V_R=\{R_1, R_2, ..., R_N\}$, $V_J=\{J_1, J_2, ..., J_N\}$ — вершины графа, $E=\{E_1, E_2, ..., E_d\}$ — дуги графа, $E_k=\{R^*,J^*\}$, где $R^*\in V_R$ и $J^*\in V_J$, где $k=1..d$, $0\le d\le N^2$. Булева матрица RJ[1..N,1..N] называется матрицей

связности графа G, если для ∀ i=1.. N, j=1..N

Где $RJ[i,j]=\begin{cases}1 \text{ если } (R_i, Job_j)\in E\\ 0 \text{ если} (R_i, Job_j)\notin E\end{cases}$.

Определение 2: подмножество A: A={a_1, a_2, ..., a_n}, где $a_k=(R^k, J^k)$, $R^k\in V_R$, $J^k\in V_J$, k=1..n, n∈N| называется паросочетанием графа G или ее матрицы RJ если A={(R^1, J^1), (R^2, J^2),..., (R^n, J^n)} удовлетворяет следующим условиям: для ∀k=1..n:

- $R^k\notin AR\backslash R^k$,
- $J^k\notin AJ\backslash J^k$,

где AR={R^1, R^2, ..., R^n}, AJ={J^1, J^2, ...,J^n}.

Таким образом, $AR\subseteq V_R$, $AJ\subseteq V_J$.

Или подмножество A ребер графа G=(V,E) называется паросочетанием, если никакие два ребра из A не имеют общей вершины.

Определение 3: пара $a_k=(R^k, J^k)$, k=1..n называется *назначением* для ресурса $R^k\in V_R$ и для задания $J^k\in V_J$.

Определение 4: пусть X={A^1, A^2, ..., A^z}, z∈N| есть множество результатов всех возможных паросочетаний для графа G=(V,E) или ее матрицы RJ. Паросочетание A* называется максимальным паросочетанием или *решением* для данного графа G если:

1) |A*|=n*.
2) n*=max{|A^1|, |A^2|, ..., |A^z|}.

Теоретическое обоснование выделения обязательных назначений

Анализ свойств не взвешенного двудольного графа при решении задачи поиска максимального паросочетания, а также анализ известных алгоритмов позволяют сформулировать следующую теорему.

Теорема 1:

Если в матрице RJ[i,j], i=1..N, j=1..N графа $G=(V_R,V_J,E)$, где V_R={1,2,...N}, V_J={1,2,...N}, существует решение A мощностью n=N и существуют такие вершины (p,q) что

RJ[p,q]=1,

RJ[p,j]=0 ∀ j∈{1,..,N}\q (5) и/или

RJ[i,q]=0 ∀ i∈{1,..,N}\p. (6)

Тогда эта пара (p,q) всегда участвует в решении A, (p,q)∈A.

Определение 5: назначения (p,q) по теореме 1 называются “обязательными”.

Примечание 2. Ввиду редукции графа и соответствующей коррекции МС следствие 1 может применяться рекуррентно.

Теорема 1*

Любая из вершин не взвешенного двудольного графа, имеющая степень равную единице, всегда участвует в одном из вариантов максимального паросочетания.

Теорема 1* справедлива как для вершин-заявок, так и для вершин-ресурсов. В том случае, если вершины имеющие степень 1, образуют веер, то Теорема 1* справедлива для любой вершины входящей в веер и каждая из них может быть взята в паросочетание, а проверку остальных вершин на возможность получения увеличивающего пути выполнять не следует.

Теорема 2.

Если в матрице RJ[i,j], i=1..N, j=1..N, ∃FA (веер):

- FA={(R^1,q), (R^2,q),..., (R^f,q)}, $R^k\in${1,..,N}, 2≤ f ≤N, где RJ[R^k,q]=1 для ∀k=1..f и RJ[R^k,J^k]=0, ∀$J^k\in${1,..,N}\q (2.7) или
- FA={(p,J^1), (p,J^2),..., (p,J^f)}, $J^k\in${1,..,N}, 2≤ f ≤N, где RJ[p,J^k]=1 для ∀k=1..f и RJ[R^k,J^k]=0, ∀ $R^k\in${1,..,N}\ p ,

тогда любая из вершин FA входит в один из вариантов максимального паросочетания, задача назначения не имеет полного решения и мощность максимального паросочетания определяется из выражения M<N-f+1.

Следствие 2

Размерность решения задачи поиска максимального паросочетания может быть жменьшена на количество пар вершин, определенных по Теоремам 1 и 1* и поиск паросочетания должен вестись из нового суграфа.

Следствие 2*

Размерность решения задачи поиска максимального паросочетания должна быть жменьшена на количество вершин, входящих в веер, а поиск паросочетания должен вестись из нового суграфа.

Следствие 3

Смежные ребра, инцидентные вершинам, определенным по Теореме 1, должны быть удалены из дальнейшего рассмотрения, а исходный граф редуцирован и преобразован в новый суграф.

Следствие 4

Смежные ребра, инцидентные вершинам, определенным по Теореме 2, должны быть удалены из дальнейшего рассмотрения, а исходный граф редуцирован и преобразован в новый суграф.

Временная сложность алгоритм поиска максимального паросочетания на основе предло-

женного подхода равен для взвешенного дудольного графа равна $O(n^{1.5}\log n)$

Выводы

Использование предложенного подхода позволяет на основании анализа исходной информации выделить обязательные назначения и на ранней стадии планирования применить принцип исключающего планирования, что в свою очередь снижает среднее время ожидания заявок, понижает размерность решения задачи планирования, а соответственно и время работы планировщика, увеличивает пропускную способность вычислительной системы. Кроме этого применение описанной теоретической базы позволяет уменьшить временную сложность Венгерского алгоритма для поиска максимального паросочетания для взвешенного двудольного графа и применить ее для планирования в неоднородных системах.

Литература

1. Foster, C. Kesselman, and S. Tuecke, “The Anatomy of the GRID: Enabling Scalable Virtual Organizations,” International Journal of High Performance Computing Applications, 15 (3), 200–222, 2001
2. Myer, “GRID Computing: Conceptual Flyover for Developers,” IBM Corporation, 1133 Westchester Avenue, White Plains, New York 10604, May 2003
3. L.-J. Zhang, J.-Y. Chung, and Q. Zhou, “Developing GRID Computing Applications, Part 1: Introduction of a GRID Architecture and Toolkit for Building GRID Solutions,” Updated November 20, 2002, IBM Corporation, 1133 Westchester Avenue, White Plains, New York 10604, October 1, 2002
4. F. Berman, G. Fox, and A. J. Hey (Eds.), GRID Computing: Making the Global Infrastructure a Reality, Wiley, 2003
5. M. Chetty and R. Buyya, “Weaving Computational GRID: How Analogous Are They With Electrical GRID?” IEEE Computing in Science and Engineering, July/August 2002
6. GRID Computing Info Centre (GRID Infoware), “GRID Computing, Answers to the Enterprise Architect Magazine Query,” Enterprise Architect Magazine, http://www.cs.mu.oz.au/~raj/GRIDInfoware/GRIDfaq.html
7. R. Buyya, Economic-Based Distributed Resource Management and Scheduling for GRID Computing, Ph.D Thesis, Monash University, Melbourne, Australia, April 12, 2002
8. Elsadek A. A. and Wells B.E., “Heuristic model for task allocation in a heterogeneous distributed systems”, *proc. of PDPTA’96,* California USA, Vol.2, (1996), pp659-671.
9. Papadimitry X., Stayglitsh K., *Combinatory optimization, algorithm and complexity*, Moscow: Mir (1985).
10. Berge C. , *Theorie des graphes et ses application,*. Dunod, Paris (1958).
11. Hopcroft J.E. and Karp R.M., An $n^{2.5}$ Algorithm for Maxi-mum Matchings in Bipartite Graphs, SIAM J. Comput. 2(4) (1973) 225-231.

УДК 004.6

ТОМАШЕВСЬКИЙ В.М.,
ЯЦИШИН А.Ю.

МАТЕМАТИЧНА МОДЕЛЬ ЗАДАЧІ ПРОЕКТУВАННЯ ГІБРИДНИХ СХОВИЩ ДАНИХ З ВРАХУВАННЯМ СТРУКТУР ДЖЕРЕЛ ДАНИХ

У статті розглядається питання формулювання математичної моделі задачі проектування гібридних сховищ даних (ГСД). Приводиться опис існуючих рішень та описується придатність їх застосування. Пропонується математична модель, яка включає опис джерел даних і сховища даних, а також оптимізаційних параметрів та рівнянь.

In paper mathematical model formulation problem for hybrid data warehouse (HDW) building is discussed. An overview of existing solutions is described and their applicability is stated. Author introduces a mathematical model which includes data sources and warehouses description. Also, this model includes optimization parameters and equations.

Вступ

При проектуванні сховищ даних постає питання коректної побудови структур даних для ефективного його функціонування.

В різних наукових публікаціях були запропоновані різні математичні моделі для формалізації задачі проектування сховищ даних. Однак модель, яка б враховувала особливості задачі проектування гібридних сховищ з врахуванням структур джерел даних, запропонована не була.

При цьому, актуальним та невирішеним на сьогодні є питання автоматизованого чи напівавтоматизованого проектування гібридних сховищ даних у цілому. Ця проблема випливає з практичної необхідністі автоматизованої побудови сховищ даних з оптимальною швидкодією за вибраним критерієм оптимальності сховища.

Ціллю статті є запропонувати математичну модель для задачі автоматизованого проектування гібридних сховищ даних, б враховувала і розміщення даних у сховищі, і оптимізаційні фактори (матеріалізовані подання, індекси, схеми розбиття), частоту виконання запитів до сховища та структури джерел даних.

Питаннями застосування генетичних алгоритмів до побудови сховищ даних займалися та оптимізації сховищ даних Wen-Yang Lin,I-Chung Kuo [1], Chuan Zhang, Xin Yao, Jian Yang [2,5], Ladjel Bellatreche,Kamel Boukhalfa [3], J.-T. Horng, Y.-J. Chang, B.-J. Liu [4], Goran Velinov, Danilo Gligoroski, Margita Kon Popovska [6], Michael Lawrence [7], Бакуліна М.А. [8]

У праці [9] розглянуто основні принципи побудови та функціонування сховищ даних. У статтях [10] і [11] розглядаються питання проектування гібридних сховищ даних.

Виклад основного матеріалу

У дисертації Бакуліної М.А. [8] розглядаються методи та алгоритми автоматизації проектування структур сховищ даних для аналітичної обробки числових показників. У роботі ставляться цілі прискорення процесу проектування сховища даних та підвищення швидкості аналітичної обробки даних сховища даних.

Розглядаються наступні задачі :

1) Розробка єдиної математичної моделі бази даних і сховища даних
2) Розробка математичної моделі багатомірного сховища даних
3) Розробка математичної моделі операцій над багатомірним сховищем даних
4) Розробка математичної моделі структури даних в сховищі даних, що відповідає вимогам OLAP-систем по швидкодії
5) Розробка алгоритмів, що автоматизукть процес побудови сховищ даних на основі запропонованих моделей
6) Розробка алгоритмів OLAP на основі запропонованої структури
7) Розробка програмної системи, що здійснює автоматизацію проектування сховища даних і оперативний аналіз реляційного сховища даних на основі запропонованих алгоритмів.

Для розв'язання вишенаведених задач застосовуються методи тензорної алгебри, кратномасштабного аналізу, вейвлет-перетворень і сигнатуроного пошуку.

Математична модель

Математична модель сховища базується на тензорній моделі.

Відношення формалізуються наступним правилом:

1) відношенню $R(A_1, A_2, ..., A_n, B_1, B_2, ..., B_K, C_1, C_2, ..., C_m)$ R(Ax,A2 присвоється довільна велика літера, наприклад R;

2) n ключовим атрибутам $A_1, A_2, ..., A_n$ присвоється n різних довільних индексів, например $a_1, a_2, ..., a_n$;

3) атрибутам $B_1, B_2, ..., B_K$ и $C_1, C_2, ..., C_m$ присвоється $(k+m)$ різних індексів, що відмінних від індексов п.2, наприклад $b_1, b_2, ..., b_k, c_1, c_2, ..., c_m$;

4) $a_1, a_2, ..., a_n$ будуть відповідати коваріантним (нижнім) індексам тензору, що визначають вхідний потік даних, $b_1, b_2, ..., b_k, c_1, c_2, ..., c_m$ будуть відповідати контраваріантним (верхнім) індексам тензору, що визначають вихідний потік даних (див. рисунок 1); тензор буде мати вигляд $R^{b_1 b_2 ... b_k c_1 c_2 ... c_m}_{a_1 a_2 ... a_n}$

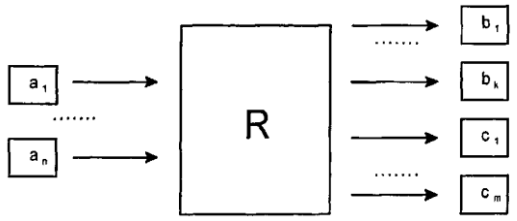


Рис. 1. Графічне представлення тензору

Запити до сховища даних формалізуються наступним правилом.

Поток запитів до сутності сутності бази даних позначається e^, де під «^» розуміється індекс (індекси), що позначають вхідні дані запиту. В загальному випадку e можно описати матрицею, в якій одиничні елементи розміщені на позиціях, що дорівнюють значенню «^».

Тензорна модель запиту описується виразом:

$$e^{jl} * P^k_{jl} = e^k, \quad (1)$$

яка має наступне матричне відображення:

$$\left\| e_{jl} \right\|_{n \times n} \times \left\| k_{jl} \right\|_{n \times n} = \begin{Vmatrix} 0 & 0 & . & 0 \\ 0 & 0 & . & 0 \\ . & . & 1_{jl} & . \\ 0 & 0 & . & 0 \end{Vmatrix} \times \begin{Vmatrix} k_{11} & k_{12} & . & k_{1n} \\ k_{21} & k_{22} & . & k_{2n} \\ . & . & k_{jl} & . \\ k_{n1} & k_{n2} & . & k_{nn} \end{Vmatrix} = k_{jl} \quad (2)$$

Якщо два тензори $R^{b_1 b_2 ... b_k c_1 c_2 ... c_m}_{a_1 a_2 ... a_n}$ та $U^{d_1 d_2 ... d_q}_{b_1 b_2 ... b_k}$ містять однойменні індекси $b_1, b_2, \ldots, b_k$, причому у тензору $R^{b_1 b_2 ... b_k c_1 c_2 ... c_m}_{a_1 a_2 ... a_n}$ індекси $b_1, b_2, ..., b_k$ – контраваріантні, а у тензора $U^{d_1 d_2 ... d_q}_{b_1 b_2 .. b_k}$ – коваріантні, то відношення $U(B_1, B_2, ..., B_k; D_1, D_2, ..., D_q)$ знаходиться в зв'язку один до одного з відношенням $R(A_1, A_2, ..., A_n; B_1, B_2, ..., B_k; C_1, C_2, ..., C_m)$ та індекси $b_1, b_2, ..., b_k$ в тензорі $R^{b_1 b_2 ... b_k c_1 c_2 ... c_m}_{a_1 a_2 ... a_n}$ позначають зовнішні ключі.

У моделі також визначені наступні операції над багатомірним сховищем даних:

1) зріз:

$$s_a(Z^k_{adx}) = Z^k_{a_m dx}, \quad 3)$$

де a_m – елемент деякої множини $a=\{a_1, a_2, a_3, ..., a_m, ..., a_n\}$.

2) обертання:

$$s_a(Z^k_{adx}) = Z^k_{a_m dx}, \quad 4)$$

$$Rot(Z^k_{adx}) = Z^k_{dxa} = Z^k_{xad} = Z^k_{ndx} \quad 5)$$

3) агрегація:

$$DU(Z^k_{adx}) = Z^K_{ADX}, \quad 6)$$

де $A=\{a_1, a_2, a_3, ..., a_n\}$, $D=\{d_1, d_2, d_3, ..., d_m\}$, $X=\{x_1, x_2, x_3, ..., x_m\}$, $K=\sum_{a=1}^{n}\sum_{d=1}^{m}\sum_{x=1}^{t} k_{adx}$

4) деталізація:

$$DD(Z^K_{ADX}) = Z^{K_1}_{ADx_1}, \quad 7)$$

де $A=\{a_1, a_2, a_3, ..., a_n\}$, $D=\{d_1, d_2, d_3, ..., d_m\}$, x_1=const, $K_1=\sum_{a=1}^{n}\sum_{d=1}^{m} k_{adx_1}$

Ця стаття пропонує непогану математичну модель сховища даних та алгоритм його проектування, однак не розглядає проектування сховища даних за певним критерієм, що не дозволяє говорити про оптимальність сховища даних.

Стаття [1] описує генетично-жадібний алгоритм для вибору куба даних OLAP. Задачею є вибрати куби даних з N вимірами з 2N можливих. Хромосома складена з 1 для вибраних кубів, 0 – для не вибраних. За функцію придатності взята функція вартості, яка дорівнює

$$\sum_{i=1}^{n} f_{c_i} E(c_i.M) + g_u \sum_{c \in M} U(c, M) \quad (8)$$

де f_{c_i} – частота звернення до куба і, $E(c_i.M)$ – вартість оцінки куба і при поточній матеріалізації М, g_u – частота вставки в базове відношення, $U(c,M)$ – вартість обслуговування куба і при поточній матеріалізації М. Селекція вибирається шляхом генерації випадкового числа від 0 до 1 і вибору більш придатної хромосоми, якщо це число перевищує 0,75. Схрещування відбувається з імовірністю (точкою схрещування) 0,7. Мутація відбувається з імовірністю pm. Крім генетичного алгоритму, використовується жадний алгоритм для тих рішень, які не є допустимими. Обробка недопустимих рішень відбувається так: якщо пара батьконащадок мають тих самих предків і вони матеріалізовані, тоді треба роз матеріалізувати вузол-нащадок, перерахувати загальну вартість і замінити старе рішення новим.

Дане рішення може бути використано як елемент МАР для вибору кубів у багатовимірній базі даних, однак воно не вирішує питання побудови цього сховища.

В статті [2] виріщується питання вибору представлень для матеріалізації шляхом «об'єднання» планів виконання запиту. Хромосома складена так: 1 – якщо представлення матеріалізується, 0 – якщо ні. Функція придатності дорівнює $C_{\max} - c(x)$, де $c(x)$ – функція вартості, а $C_{\max}$ – максимальне значення $c(x)$ в популяції або в останніх k поколіннях. Використовується одно точкове схрещування з випадково вибраною точкою схрещування від 1 до n, де n – довжина хромосоми. Мутація відбувається за допомогою інвертування біта від 1 до n, що випадково вибирається в хромосомі.

Дане рішення може бути використано для вибору матеріалізованих представлень.

У статті [3] розглядаються проблема вибору схеми розбиття сховища даних, вико ристовується горизонтальне розбиття. Кожен атрибут фрагментації бути предсттавлений масивом з n елементами, де n відповідає числу його піддоменів. Значення цих елементів знходться між 1 і n. Якщо два елементи мають ті ж значення, то вони будуть об'єднані для формування одного. Це кодування може бути використане для представлення фрагментів таблиць вимірів і таблиці фактів. Для селекції використовується метод колеса рулетки (він виділяє сектор колеса, що відповідає і-й хромосомі і створює нащадка, якщо згенероване число знаходиться в околі 0 зупиняється усередині призначеного сектора рядка). Використовується двоточковий механізм схрещування. Значення придатності визначається за допомогою призначення балів по двох показниках : поріг і виконання запитів. Поріг визначає, чи перевищує кількість отриманих фрагментів дорівнює ± 5 відсотків порогу, то будуть призначені 55 балів, інакше менше балів. Виконання запитів визнчає, чи перевищує вартість запиту встановлене значення, що обчислюється за допомогою функції вартості, якщо це так, то призначається менше 3 балів за запит. Функцією вартості є

$$Cost(Q) = \sum_{j=1}^{N} valid(Q, S_i) \prod_{i=1}^{M_j} \left(\frac{Sel_F^{p_i} \times \|F\| \times L}{PS} \right), \quad (9)$$

де M_j, F, L і PS позначають відповідно кількість предикатів вибірки, що визначають фрагмент факту підсхеми типу «зірка» $Sel_F^{p_i}$, кількість кортежів, присутніх в таблиці фактів F, розмір у байтах кортежу таблиці F та розмір сторінки файлової системи (у байтах).

Мутація відбувається з нормою 6-30%.

Дане рішення може бути використано як елемент МАР для вибору схеми розбивки бази даних (як реляційної, так і багатовимірної) , але тим самим вирішується лише питання оптимізації сховища даних.

У статті [4] розглядається застосування генетичного алгоритму до вибору матеріалізованих представлень. Хромосома складається з двох частин – вибраних планів виконання для кожного з запитів, та вибраних представлень для матеріалізації. Селекція відбувається за допомогою вибору числа від 1 до $\sqrt{n}$, потім це число підноситься до квадрату і вибирається індивід з отриманим порядковим номером, з упорядкованих індивідів по вартості у порядку її збільшення. Використовується одно точкове схрещування шляхом схрещування обох частин хромосоми: рядка планів виконання запитів та рядка представлень для матеріалізації, кожна схрещується окремо. Мутація відбувається шляхом інвертування бітів в рядку представлень, а також змінює числа в рядку планів виконання, підтримуючи допустимість рішення.

Дане рішення також може бути використано для вибору матеріалізованих представлень .

У статті [5] також розглядається проблема вибору матеріалізованих представлень. В якості хромосоми взято номери планів виконання запитів, об'єднані в бінарний рядок. Функція придатності дорівнює $C_{\max} - c(x)$, де $c(x)$ – функція вартості, а $C_{\max}$ – максимальне значення

c(x) в популяції або в останніх k поколіннях. Використовується одно точкове схрещування. Мутація відбувається шляхом випадкового вибору плану виконання запиту, з усіх доступних для вибраного для мутації запиту. Використовується турнірна селекція, де порівнюються індивіди, і кращий з них проходить до наступної популяції. Розмір турніру від 4 до 7 індивідів.

Дане рішення є більш раннім варіантом рішення [2] і також може бути використано для вибору матеріалізованих представлень.

Стаття [6] описує вибір індексів, матеріалізованих представлень для покращення реляційних баз даних. Ця стаття описує вибір індексів, матеріалізованих представлень для покращення реляційних баз даних. Використовується генетично-жадний алгоритм. Хромосома формується слідуючим чином. Об'єкт (просторові відношення, агрегатні представлення та індекси) представляється масивом бітів, тобто по частинах, які будуть об'єднані в хромосому. Просторові представлення представляються (як особливий тип представлень для матеріалізації) з їх різними варіантами . Кількітсь бітів, необхідних для представлення кожного просторового відношення з всіма його варіантами дорівнює k +1 , де k – кількість елементів додаткового набору атрибутів. Тому, перший біт використовується для представлення просторового відношення а інші n біт для представленнядодаткових атрибутів. Всі просторові відношення мають бути матеріалізовані,тому кожне з них представляється 1. Атрибут в додатковому наборі , при додаванні до його просторового відношення представляється 1,інакше 0. Агреговані представлення представляються схожим чином, тобто для кожного представлення, один біт вико ристовується для його представлення (1 – якщо вибране, 0 – не вибране для матеріалізації) і k бітів використовуються для представлення його додаткових атрибутів , тобто варіантів пред-ставлень. Атрибути додаткового набору агрегованих представляються у схожий спосіб з атрибутами додаткових наборів просторових відношень. Для кожного агрегованого представлення атрибути міри представляєься n бітами. Атрибут міри, якщо він додається до відповідного представлення, представляється 1, інакше 0.Для кожного представлення має бути доданий по якнайменше один атрибут міри. Після представлення кожного подання відбувається представлення їх можливих індексів. Кожний індекс представляється одними бітом , тобто 1 – якщо індекс вибраний, 0 – не вибраний. Кількість і порядок індексів визначаються попередньо.

Цільова функція задається наступним чином. Нехай SCM є станом схеми кубу даних SC з набором AVM AV ⊆ представлень-кандидатів на матеріалізацію, де кожний з них представляється його варіантом і набором SIM SI ⊆ індексів-кандидатів. Нехай також всі просторові відношення представляються їх відповідними варіантами. Тоді проблема оптимізації, обмежена витратами на обслуговування є наступною: вибрати стан SCM схеми кубу даних SC, яка мінімізує

$$\tau(SC, SCM, SQ) = \sum_{Q \in SQ} FQ \cdot P(Q, SCM), \qquad (10)$$

при обмеженні U (SC , SCM) ≤ S , де SQ – набір попередньо визначених запитів, Q F – частота запитів, P(Q,SCM) позначає мінімальну вартість обробки запиту Q в стані SCM схеми SC . Нехай U(SC, SCM) – загальна вартість обслуговування , визначена як:

$$U(SC, SCM) = \sum_{R \in R_{SC}} GR \cdot m(R, SCM) + \\ + \sum_{V \in AV_M} GV \cdot m(V, SCM) + \sum_{I \in I_M} GI \cdot m(I, SCM) \qquad (11)$$

де GR, GV and GI – частота оновлення відповідно відношень, представлень та індексів. Нехай m(R, SCM), m(V, SCM) та m(I, SCM) – мінімальна вартість обслуговування відношень, представлень і індексів відповідно в присутності стану SCM. P(Q, SCM) – цільова функція проблеми.

Розглянуті теоретичні рішення дозволяють оптимізувати сховище даних за допомогою індексів, матеріалізованих представлень та фрагментації. Оптимізацією є знаходження оптимальних рішень по заданих цільових функціях, керуючись заданими функціями придатності. Однак ці методи не вико ристовуються в комплексі, а методи не враховують поєднання реляційної та багатовимірної бази даних, а також структури джерел даних.

Тому необхідно запропонувати таку математичну модель, яка б враховувала і розміщення даних у сховищі, і оптимізаційні фактори (матеріалізовані подання, індекси, схеми роз-

биття), частоту виконання запитів до сховища та структури джерел даних.

Для формулювання математичної моделі задачі проектування гібридних сховищ даних з врахуванням структури джерел сховищ даних потрібно зробити формальний опис джерел даних, сховища даних, яке отримуємо в результаті проектування, а також оптимізаційних параметрів та рівнянь оптимізації.

Джерела даних. Джерела даних можуть бути структурованими файлами (розділеними чи XML) та базами даних двох типів: OLAP і OLTP.

У випадку розділених структурованих файлів метадані не задаються. У таких файлах може бути задана лише назви полів даних, що входять у файл. Типи даних не задаються. Формалізуються вони так: $S=\{s_i\}$ – множина атрибутів даних у розподіленому файлі.

У випадку структурованих файлів XML можуть бути задані метадані. У якості метаданих можу бути задані і метадані: назва поля, тип та інші необхідні характеристики. Формалізуються вони аналогічно до розподілених файлів : $X=\{x_i\}$ – множина атрибутів даних у файлі XML.

У випадку реляційних баз даних структура даних задається відношеннями. Відношення позначаються як $R=(r_1,r_2,...,r_1)$, де R – відношення, а r_i – атрибути відношення. Якщо два відношення $R=(r_1,r_2,...,r_1,r_{l+1},...,r_{l+p})$ та $R=(r_l,r_{l+1},...,r_{l+p},...,r_{l+k})$ мають спільні атрибути $r_1,r_{l+1},...,r_{l+p}$,то в одному з відношень цей набір атрибутів приймає унікальні значення, а в іншому – є зовнішнім ключем.

У випадку багатовимірних баз даних структура даних задається набором вимірів $D=\{d_1,d_2,\ldots,d_q\}$ та мір (можуть формулюватися також як вимір) $M=\{m_1,m_2,\ldots,m_u\}$.

Гібридне сховище даних. Відповідно до [11] гібридне сховище даних має наступні характерні особливості. Основними рисами гібридного сховища даних є наявність своєї окремої системи керування та автоматизоване проектування сховища під популяції запитів.

Узагальнене гібридне сховище даних передбачає наступне:

1) Гібридне сховище даних має свою систему керування гібридним сховищем даних (СКГСД), до за допомогою якої здійснюється робота з сховищем (виконання запитів до сховища).

2) В гібридному сховищі присутня реляційна БД.

3) В гібридному сховищі присутня багатовимірна БД, яка може містити як атомарні, так і узагальнені дані

4) Поєднуються підходи проектування «зверху вниз» і «знизу вгору». Реляційна і багатовимірна бази даних проектуються в комплексі, для забезпечення оптимального функціонування сховища даних у цілому.

У математичній моделі гібридне сховище даних представляється наступним чином. Воно містить :

1) Атрибути даних: $B=\{b_i\}=$ $=S\cup X\cup R\cup D\cup M$, тобто атрибути джерел повинні бути присутні в сховищі даних, що проектується.

2) Таблиці даних, що складаються з атрибутів даних: $T=\{t_i \mid t_i=\{b_j\}\}$)

3) Області сховища, що містять таблиці даних:

$$A=\{a_i\} \mid a_i=\{t_j\},\ j\in\overline{1,w},\exists b_k\in t_{j_1}, b_k\in t_{j_2})\}$$

Оптимізаційні параметри та рівняння оптимізації

З огляду на необхідність оптимізації гібридного сховища даних я пропоную використати два механізми покращення роботи сховища – застосування індексів і матеріалізованих представлень.

Індексом будемо вважати об’єкт бази даних, який дозволяє отримувати дані з таблиць швидше за допомогою збереження словників даних, які будуються за одним чи декількома стовпцями таблиці, на якій визначається індекс. У реляційних БД поля індексу доцільно визначати за допомогою генетичного алгориму : у хромосому включаються гени, які відображають включення поля в індекс таблиці. В багатовимірних базах даних побудова індексів найчастіше інкапсульована, а тому створення індексів є або неможливим, або зайвим, так як при наявності двох наборів індексів продуктивність бази даних може бути знижена. Однак принцип індексування в алгоритмі може поширюватися і на багатовимірну базу даних. Індексуються ті поля, при індексуванні яких ЦФ на цій таблиці є мінімальною.

Матеріалізованим представленням будемо вважати збережену окрему від баз даних таб-

лицю або таблиці даних, який включає в себе деякі поля цих таблиць, вибір яких здійснюється генетичним алгоритмом. У хромосому включаються гени, які від ображають включення поля в матеріалізоване представлення. Даний підхід повністю застосовний для реляційних баз даних, однак не застосовний для багатовимірних, оскільки розривання полів таблиці, що відповідає зрізу куба, може привести до порушення цілісності даних. Матеріалізуються тільки ті поля, при включенні яких у матеріалізоване представлення ЦФ на цій таблиці є мінімальною.

Для формалізації оптимізаційних параметрів та рівнянь оптимізації звернемося до викладених у роботі [10] результатів. У цій роботі Яцишином А.Ю. було запропоновано алгоритм проектування гібридних сховищ даних. Опишемо математичну модель, запропоновану в цій статті.

Відповідно до постановки задачі критерієм оптимальності сховища є кількість доступів до даних, тобто операцій читання даних, які необхідно провести для виконання запиту до сховища даних. Оскільки спроектувати базу даних під всі запити однаково ефективно неможливо, доцільно проектувати її під конкретні популяції запитів. Тому цільова функція задачі буде розраховуватися під перший запит популяції даних.

Цільова функція задачі має вигляд:

$$z = \sum_{i=1}^{n} A_i + \sum_{j=1}^{n-1} T_j , \qquad (11)$$

де n – кількість таблиць запиту, A_i – кількість доступів до *i*-ї таблиці реляційної бази даних або зрізу багатовимірної, T_j – час виконання операції з'єднання над таблицями $i_1=j$ та $i_2=j+1$

Але в практичній діяльності, враховуючи особливості використання різних баз даних, і те, що той самий запит може виконуватися порізному в залежності від виконаної оптимізації баз даних, а також неможливість отримання інформації про кількість доступів з ядра бази даних, вважаю, що дану ЦФ доцільно замінити на

$$z = \sum_{i=1}^{n} t_i + \sum_{j=1}^{n-1} T_j , \qquad (12)$$

де n – кількість таблиць запиту, t_i – час доступу до *i*-ї таблиці реляційної бази даних або зрізу багатовимірної, T_j – час виконання операції з'єднання над таблицями $i_1=j$ та $i_2=j+1$

Крім того, треба зауважити що t_i та T_j є функціями від розміщення таблиць даних:

$$t_i = f(L_1, L_2, ..., L_n) , \qquad (13)$$

де L_i – ознака розміщення таблиці даних (0 – в реляційній БД , 1 – в багатовимірній БД).

Тому ЦФ має такий остаточний вигляд:

$$z = \sum_{i=1}^{n} t_i(L_1, L_2, ..., L_n) + \sum_{j=1}^{n-1} T_j(L_1, L_2, ..., L_n) , \qquad (14)$$

де n – кількість таблиць запиту, L_i – ознака розміщення таблиці даних (0 – в реляційній БД , 1 – в багатовимірній БД).

Врахуючи змінні, запишемо наступний вираз:

$$z = \sum_{i=1}^{n} t_i(L_a, \{I_c\}, \{M_c\}) + \sum_{j=1}^{n-1} T_j , \qquad (15)$$

де n – кількість таблиць запиту,a – область, що містить таблицю i, $\{I_c\}$ та $\{M_c\}$ – ознаки індексування та матеріалізації полів c таблиці i. L_a – ознака розміщення областей сховища. $L_a = 1$, якщо всі таблиці області даних a розміщені в багатовимірній БД, та 0, якщо всі таблиці розміщені в реляційній БД.

Індексування полів таблиць представляється змінними I_c. $I_c = 1$, якщо індекс містить поле c, в протилежному випадку – 0.

Матеріалізованість полів таблиць представляється змінними $M_c . M_c = 1$, якщо поле матеріалізується, в протилежному випадку – 0.

На ОДР у роботі [10] накладаються такі обмеження:

1) Забороняється розміщення таблиць, що містять зовнішні ключі та нечислові поля, в багатомірних базах даних, оскільки такі таблиці фактів не підтримуються.

2) Забороняється створення індексів в багатовимірній базі даних, що випливає з природи багатовимірних баз даних, оскільки ця операція найчастіше інкапсульована в двигуні бази даних і виконується автоматично.

3) Забороняється включення зовнішніх ключів в поля, що входять до матеріалізованих представлень таблиць, які містяться в багатовимірній базі даних, оскільки це робить неможливим коректне виконання запитів до багатовимірної з бази даних.

4) Однак таке рішення не дозволяє врахувати такі параметри, як частоти використання відношень у сховищі, а також джерел даних.

Тому необхідно використовувати наступну модель:

$$z = \sum_{i=1}^{n} t_i(L_a,\{I_c\},\{M_c\}) + \sum_{j=1}^{n-1} T_j + (\frac{1}{F_a^{'}} - \frac{1}{\hat{F}_a})L_a \quad (16)$$

$$z = \sum_{i=1}^{n} t_i(L_a,\{I_c\},\{M_c\}) + \sum_{j=1}^{n-1} T_j + (T_a^{'} - \hat{T}_a)L_a, \quad (17)$$

де n – кількість таблиць запиту,a – область, що містить таблицю i, $\{I_c\}$ та $\{M_c\}$ – ознаки індексування та матеріалізації полів cтаблиці i, $F_a = \sum_{i \in a} f_i$ – частота доступу до області а. $F_a^{'}$– встановлене порогове значення частоти доступу до даних області a, $\hat{T}_a$ – середній час виконання запитів, $T_a^{'}$– порогове значення часу виконання запиту у до даних області a .

Показник частоти задається тому, що при високій частоті доступу до даних доцільно розміщувати у реляційній базі даних. Для цього додатково задається порогове значення частоти $F_a^{'}$, яке впливає наступним чином :

- Якщо сумарна частота доступів до області а більше порогового значення, цільова функція збільшується на різницю $(\frac{1}{F_a^{'}} - \frac{1}{F_a})L_a$, у випадку L_a=1 дане значення стає «гіршим» з точки зору оптимізації, так як існують рішення, на яких цільова функція приймає менше значення.
- Якщо сумарна частота доступів до області а менше порогового значення, цільова функція зменшується на різницю $(\frac{1}{F_a^{'}} - \frac{1}{F_a})L_a$, у випадку L_a=1 дане значення стає «краще» з точки зору оптимізації, ніж без врахування фактору частоти.

Запишемо остаточне формулювання задачі проектування гібридних сховищ даних.

Задані множини атрибутів розділених файлів S, файлів XML X, відношення у реляційній базі даних R, виміри багатовимірної бази даних D та міри M .. Спроектувати гібридне сховище даних, визначивши області сховища даних A, таблиці T та атрибути B та задавши порогові значення частот доступу до даних областей $F_a^{'}$.Знайти такі значення ознак розміщення областей L_a, індексування I_c та матеріалізації M_c, на яких значення цільової функції (17) буде мінімальним серед всіх можливих наборів значень цих змінних.

Висновки

У даній статті було проаналізовано існуючі роботи, що стосуються математичних моделей проектування сховищ даних . Розглянуті теоретичні рішення пропонують моделі проектування сховища даних, оптимізації за допомогою індексів,матеріалізованих представлень та фрагментації. Однак ці методи не використовуються в комплексі, а методи не враховують поєднання реляційної та багатовимірної бази даних.

Тому запропоновано математичну модель, що враховує структуру джерел даних та гібридного сховища даних, оптимізаційні параметри та рівняння.

В подальших дослідженнях слід вивчити особливості функції часу виконання запиту в залежності від її параметрів та запропонувати методи розв’язання задачі проектування гібридних сховищ даних з врахуванням структур сховищ даних.

Перелік посилань

1. Wen-Yang Lin. A Genetic Selection Algorithm for OLAP Data Cube [Текст] / Wen-Yang Lin,I-Chung Kuo – Knowledge and information systems 2004, vol. 6
2. Chuan Zhang.An Evolutionary Approach to Materialized Views Selection in a Data Warehouse Environment. Systems [Текст]/ . Chuan Zhang, Xin Yao, Jian Yang – Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on Volume 31, Issue 3, Aug 2001
3. Ladjel Bellatreche. An Evolutionary Approach to Schema Partitioning Selection in a Data Warehouse [Текст] / Ladjel Bellatreche,Kamel Boukhalfa – Lecture notes in computer science, Congrès DaWaK 2005 : data warehousing and knowledge discovery: International conference on data warehousing and knowledge discovery No7, Copenhagen , DANEMARK 2005 , vol. 3589
4. J.-T. Horng. Applying evolutionary algorithms to materialized view selection in a data warehouse [Текст]/ J.-T. Horng, Y.-J. Chang, B.-J. Liu – Soft Computing – A Fusion of Foundations, Methodologies and Applications, Volume 7, Number 8 / August, 2003

5. Chuan Zhang. Evolving Materialized views in a Data Warehouse [Текст] / Chuan Zhang, Xin Yao, Jian Yang – Evolutionary Computation, 1999. CEC 99. Proceedings of the 1999 Congress on Volume 2
6. Goran Velinov. Framework for Generalization and Improvement of Relational Data [Текст] / Goran Velinov , Danilo Gligoroski, Margita Kon Popovska – IJCSNS International Journal of Computer Science and Network Security, VOL.8 No.3, March 2008
7. Michael Lawrence. Multiobjective genetic algorithms for materialized view selection in OLAP data [Текст] / Michael Lawrence – Proceedings of the 8th annual conference on Genetic and evolutionary computation,2006
8. Бакулева Марина Алексеевна. Модели и алгоритмы автоматизации проектирования структур хранилищ данных для аналитической обработки числовых показателей [Текст] : диссертация кандидата технических наук : 05.13.12 Рязань, 2007 147 с., Библиогр.: с. 124-131 РГБ ОД, 61:07-5/4771
9. Шаховська Н.Б. Сховища та простори даних: Монографія [Текст]. / Шаховська Н.Б, Пасічник В.В. – Львів: Видавництво Національного університету «Львівська політехніка» , 2009. – 244 с.
10. Яцишин А.Ю. Застосування генетичного алгоритму для проектування гібридних сховищ даних [Текст]. Вісник Національного університету „Львівська політехніка”, секція "Інформаційні системи та мережі",/ Яцишин А.Ю. - м.Львів - 2011
11. Яцишин А.Ю. Підходи та алгоритми проектування гібридних сховищ даних [Текст]. Вісник Національного університету „Львівська політехніка”, секція "Інформаційні системи та мережі" / Яцишин А.Ю – м. Львів - 2010

УДК 004.738

ДЕМЧИНСКИЙ В.В.,
ДОРОГОЙ Я.Ю.,
ДОРОШЕНКО Е.С.

АСПЕКТЫ АСИММЕТРИЧНОЙ МАРШРУТИЗАЦИИ В INTERNET

В роботі розглядаються питання дослідження і класифікації причин виникнення асиметрії маршрутів передачі даних в Internet. Приведено аналіз впливу асиметрії на продуктивність мережних служб і ефективність використання ресурсів.

Objective of this paper – research and classification of Internet routing asymmetry causes and analysis of the such asymmetry impact on the network services performance and resource use efficiency.

Введение

Зачастую при классификации, анализе и управлении потоками данных в сетях неявно используется предположение о симметрии потоков. Как результат, использующие такое предположение методики хорошо работают в конечных сетях, по сути симметричных, но могут давать ошибки в магистралях и муниципальных сетях, менее предсказуемых по своей природе. Поэтому, используя неявное предположение о симметрии путей, следует быть уверенным в его оправданности.

При диагностике нарушений маршрутизации в сетях процесс начинается с проверки параметров прямого и обратного путей. Наличие асимметрии затруднит процесс диагностики, поскольку отсутствует уверенность в локализации неполадки. С другой стороны, статистическая классификация трафика, используемая при решении задач управления сетями, обеспечения безопасности и качества обслуживания, может давать не точные результаты в условиях асимметрии маршрутов.

Поскольку различные прикладные протоколы и приложения в процессе функционирования показывают слабую степень корреляции между прямыми и обратными потоками, то измерения, лишь частично отслеживающие такие потоки, приведут к искаженным оценкам.

Этими фактами обуславливается актуальность исследования асимметрии в сетях и ее влияния на производительность сетевых приложений и эффективность использования ресурсов.

Цель данной статьи – исследование и классификация причин возникновения асимметрии маршрутов в Internet, а так же анализ влияния такой асимметрии на производительность сетевых служб и эффективность использования ресурсов.

Обзор предметной области

Под асимметрией маршрутов будем понимать отличие характеристик прямого и обратного пути между двумя конечными хостами. В зависимости от первопричины возникновения, различают пространственную асимметрию (асимметрию топологии) и асимметрию характеристик каналов. Асимметрия может проявляться в: пропускной способности, задержках передачи, величине потерь в каналах. Например, среди существующих физических технологий передачи данных, спутниковые каналы характеризуются наибольшими задержками, а каналы беспроводных локальных сетей – наибольшим процентом потерь. Асимметрия пропускной способности будет увеличивать значение задержки в узком канале и, как следствие, асимметрию задержки. Ситуация усложняется временными колебаниями средней задержки (даже при стабильности маршрутов), которая еще более усиливает асимметрию.

Впервые аспекты появления и влияния асимметрии маршрутов в Internet были исследованы в работе [1]. Существуют методы и инструменты численной оценки асимметрии маршрутов передачи в сетях [1,2] и изучается влияние таких оценок на производительность сетевых служб.

Предпосылки возникновения асимметричной маршрутизации

Причинами возникновения явления асимметрии маршрутов могут быть:

- каналы связи с различной пропускной способностью или задержкой в разных направлениях;
- отличия в загрузке канала в разных направлениях;
- симметричная и асимметричная балансировка нагрузки (в том числе при использовании протокола HSRP);
- статическая маршрутизация или иная политика маршрутизации, вызывающая асимметрию (в том числе редистрибьюция маршрутов между различными протоколами или областями маршрутизации);
- различающиеся метрики или параметры метрик прямого и обратного пути (например, по причине несогласованности политик маршрутизации в разных областях администрирования или за счет динамической природы самих метрик протоколов маршрутизации);
- особенности функционирования протокола маршрутизации OSPF на границе между областями;
- переходные процессы при изменении маршрутов (временная асимметрия).

Другими словами, асимметрия характеристик маршрутов обуславливается асимметрией характеристик каналов или асимметрией самих путей. В свою очередь, асимметрия путей может возникать из-за субоптимальной маршрутизации, балансировки нагрузки или иных особенностей политики маршрутизации. В целом, согласно идеологии работы сетевого уровня как службы «с максимальным усилием», постоянство маршрутов в Internet не гарантируется.

Рассмотрим подробнее два случая, когда проявление асимметрии представляется наименее очевидным: асимметрия маршрутов между областями OSPF и между автономными системами Internet.

Как известно, преимущество OSPF маршрутизации – в иерархичности построения системы маршрутизации. В целом, вся OSPF система разбивается на ряд областей, связанных через магистраль (нулевую область). Данный подход, во-первых, позволяет скрыть внутренние изменения, происходящие в пределах некоторой области от других областей и, во-вторых, уменьшает вычислительную нагрузку на маршрутизаторы. Таким образом появляется возможность использовать преимущества иерархического разбиения адресного пространства.

Рассмотрим следующий пример поведения OSPF системы при взаимодействии между областями (Рис. 1):

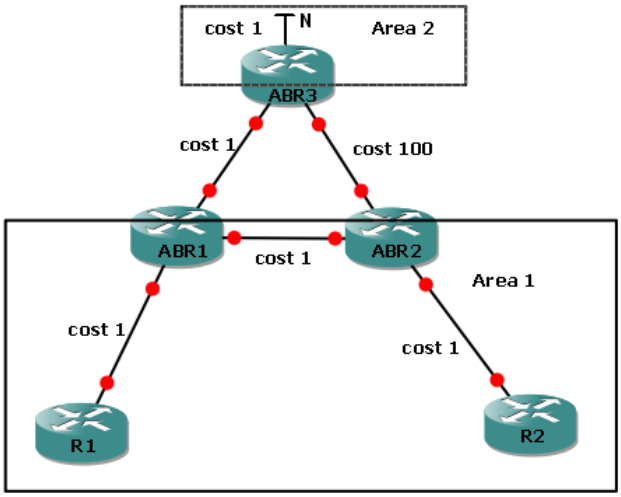


Рис. 1. Маршрутизация между областями OSPF системы

В примере две области (1 и 2) связаны через магистраль (область 0). Канал ABR1-ABR2 внутренний по отношению к области 1. Пограничные маршрутизаторы областей (ABR – Area Border Router) скрывают внутреннюю топологию одних областей от других, передавая за пределы области только обобщенную маршрутную информацию. При наличии нескольких альтернативных маршрутов маршрутизатор предпочитает внутренний маршрут области маршруту, проходящему через другую область. Технически данное правило обусловлено тем, что во избежание маршрутных петель, пограничные маршрутизаторы областей игнорируют маршрутную информацию других пограничных маршрутизаторов, поступающую не через магистраль. Поэтому выбираемый OSPF “кратчайший” маршрут к назначению, принадлежащему данной области, не может выходить за пределы этой области. Пусть каналы упомянутой выше топологии имеют стоимости, указанные на рисунке. При данных условиях ABR2 игнорирует маршрут к сети N с действительно минимальной метрикой 3 и выбирает маршрут

напрямую через ABR3, хотя и имеющий большую метрику, зато являющийся внутренним по отношению к магистрали (области 0).

Однако маршрутизатор R2 определяет маршрут по-другому. Для него кратчайший маршрут к N будет маршрут с минимальной метрикой 4, проходящий через ABR2-ABR1-ABR3. R2 посылает пакеты, адресованные к N, через ABR2, который, пользуясь правилом, описанным выше, выбирает субоптимальный маршрут. Выбирая обратный маршрут к R2, ABR3 отправляет пакеты через ABR1, который передает их ABR2 и тот, в свою очередь, R2. В результате, между R2 и N возникает асимметричный маршрут. Таким образом, данная особенность поведения протокола OSPF может проявляться в случаях наличия между двумя ABR альтернативных путей.

В целом же, возникновение асимметричных маршрутов более вероятно в магистральных, нежели в конечных сетях. т.е. асимметричная маршрутизация проявляется скорее в глобальных (региональных) сетях (асимметрия на уровне автономных систем Интернет). Например, в каналах ISP часто используется тактика «горячей картошки», согласно которой пакет должен покинуть сеть быстро, насколько это возможно, что означает минимизацию потребляемых ресурсов. Однако, реже используемая тактика «холодной картошки» предполагает, что ISP явно указывает другому ISP точку входа в свою сеть для трафика, адресованного по некоторому назначению. Вторая тактика ничуть не меньше первой может вызывать асимметрию маршрутов.

Помимо этого, может использоваться неоптимальная стратегия маршрутизации при кольцевой топологии магистрали. Под «неоптимальной» здесь понимается директивная политика маршрутизации, которая в отличие от оптимальной маршрутизации, не использует метрики протоколов как критерии выбора лучших маршрутов.

Если сравнивать поведение маршрутов в пределах автономных систем Интернет (AS – Autonomous System) и между автономными системами, то существуют предпосылки [2,3] считать, что изменения маршрутов между AS (автономными системами) происходит реже (асимметрия более стабильна), но больше влияют на сквозную задержку передачи, чем изменения маршрутов в пределах AS, которые меньше оказывают влияния на общую задержку. В целом же, маршруты между AS (как правило, вычисляемые по протоколу BGP) имеют большую асимметрию, чем внутренние маршруты AS (т. е. BGP маршрутизация имеет существенно ассиметричный характер).

Асимметрия маршрутизации между автономными системами носит стабильный характер в связи с особенностями реализации алгоритма BGP. В первую очередь маршрутизация зависит от принятой в каждой из автономных систем политики. Протокол BGP в своей реализации использует множество атрибутов, влияющих на маршрутизацию между автономными системами. Решение о выборе конкретного маршрута принимается внутри каждой автономной системы *независимо*. При использовании атрибутов со значениями по умолчанию решение о выборе маршрута принимается на основании атрибута AS-PATH. Данный атрибут представляет собой линейный вектор-список автономных систем, через которые проходит информация от источника до назначения. Более короткий вектор путей является предпочтительным. Имеется также возможность управлять исходящим и входящим маршрутом для определенного трафика. Таким образом, асимметричность маршрутизации для каждой автономной системы уникальна и является следствием решения администратора этой системы.

Для конечных сетей, имеющих несколько альтернативных каналов к магистралям (multhomed networks), с целью повышения эффективности использования каналов используется балансировка нагрузки между альтернативными каналами, в результате которой может появиться асимметрия маршрутов. И поскольку в основном альтернативные каналы проходят через разных ISP, то контроль со стороны абонента потоков по этим каналам затруднен.

Влияние маршрутной асимметрии на работу сетевых служб

Асимметрия путей может сказываться на работе сетевых служб, учитывающих состояния потоков (измерение и учет трафика, трансляция сетевых адресов, контекстный и рефлексивный контроль доступа, распределение нагрузки между сетевыми экранами). Так же, отличие пря-

мого и обратного маршрута может затруднять многоадресную передачу пакетов и применение функции RPF (Reverse Path Forwarding – проверка обратного маршрута при принятии решения о продвижении пакета). Т.е. обработка односторонних потоков перечисленными службами будет затруднена или вовсе невозможна.

Балансировка трафика по нескольким альтернативным каналам при избыточности каналов увеличивает разброс задержки передачи, усложняет ее оценку и тем самым пагубно сказывается на качестве обслуживания, (например с случае передачи голосового или другого трафика реального времени). Кроме того, многие прикладные службы, явно оценивающие величину круговой задержки и разброс величин задержек, показывают снижение производительности. Например, служба синхронизации времени NTP использует предположение, что сквозное время доставки пакета равно половине круговой задержки (неявное предположение о симметрии маршрутов), поэтому с увеличением асимметрии задержки будет нарастать ошибка синхронизации времени.

Известная оценка максимальной призводительности TCP-соединения [4] обратно пропорциональна величине круговой задержки (RTT – Round Trip Time) и вследствие этого увеличение вариации RTT снижает эффективность TCP-соединения. Собственно говоря, при наличии асимметричных маршрутов эвристическая оценка доступной пропускной способности, используемая TCP оказывается менее точной, чем при симметрии маршрутов.

Некоторые сетевые службы (например, мультикастовые службы трансляции видео в реальном времени или сети распространения контента) используют оценку величины круговой задержки как критерий выбора сервера (оценку удаленности сервера). В данном случае полезней будет использовать оценку односторонней задержки (OWD – One-way delay) (RFC 2679) в направлении основного потока передаваемых данных. Однако, сложность такой оценки вызвана отсутствием стандартных средств и проблемой точного измерения времени и синхронизации часов [5]. Исследования поведения односторонней задержки в Интернет показали слабую корреляцию колебаний OWD и RTT, однако более сильную зависимость колебаний OWD от изменения маршрутов.

При наблюдении за трафиком сетевых приложений практически невозможно оценить асимметрию задержки, поскольку в протоколах TCP и UDP отсутствуют временные метки. Популярные диагностические приложения (ping, traceroute...) дают оценку круговой задержки. То же касается оценки круговой задержки в протоколе TCP (используемой при установке таймера ожидания подтверждения). Более того, эта оценка не учитывает стратегию отложенного подтверждения в TCP (задержка подтверждения до 200 мс в ожидании попутных сегментов данных).

Стоит отметить, что службы, использующие протоколы UDP и ICMP, часто по природе своей асимметричны, т.е. не предполагают ответных пакетов на каждый единичный пакет. Диагностика маршрутной асимметрии посредством анализа трафика таких приложений представляет собой гораздо более сложную задачу, нежели для приложений симметричных. Кроме того, в процессе анализа сетевого трафика обнаруживается существование некоторого количества необусловленного (unsolicited) трафика, например трафика сканирования, на который отсутствует ответный трафик.

При балансировке между альтернативными маршрутами возможно переупорядочивание пакетов. Реакцией TCP – протокола на переупорядоченные пакеты будет дублирование подтверждений, запускающее после получения трех одинаковых подтверждений механизм быстрой повторной передачи (RFC 2581). Такое поведение протокола снижает эффективность соединения, а повторные передачи – эффективность утилизации пропускной способности. Другими причинами переупорядочивания пакетов могут быть особенности архитектуры маршрутизаторов или трафик-инжиниринг.

В целом, трудно оценить асимметрию маршрутов на качественном уровне. Например, утилита «traceroute» не всегда однозначно отображает адрес маршрутизатора, что делает невозможным сопоставление прямого и обратного маршрута. Однако, немалый процент асимметричных маршрутов имеют отличие лишь в одном звене (как правило, по причине балансировки трафика). Такая минимальная асимметрия

минимально влияет на отличие прямой и обратной задержек, однако делает возможным переупорядочивание пакетов. Количественная оценка асимметрии представляет собой более сложную задачу. Так, без синхронизации часов проблематично вычислить разницу задержки прямого и обратного пути.

Моделирование асимметрии маршрутов

Для иллюстрации влияния маршрутной асимметрии на производительность сетевых приложений рассмотрим простой эксперимент в следующей топологии с минимально возможной пространственной асимметрией на одном участке (Рис.2):

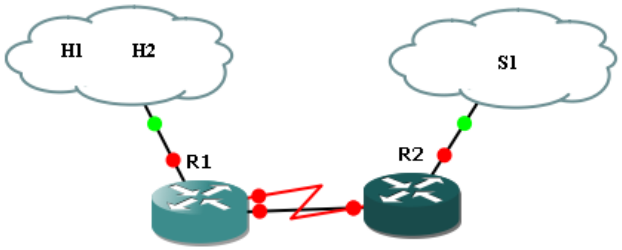


Рис. 2. Топология эксперимента

Будем сравнивать колебания круговой задержки пакетов между хостами H1 и S1 в условиях свободных каналов и в условиях загруженных каналов (передача некоторого большого файла между H2 и S1). В зависимости от настройки маршрутизаторов, получим несколько различных тестов:

- один симметричный маршрут между конечными хостами (асимметрия отсутствует);
- различные прямой и обратный маршрут между R1-R2, каналы R1-R2 с разной пропускной способностью (ПС);
- симметричная балансировка трафика на участке R1-R2; каналы R1-R2 с равной ПС;
- симметричная балансировка трафика на участке R1-R2; каналы R1-R2 с разной ПС;
- асимметричная балансировка трафика на участке R1-R2; каналы R1-R2 с разной ПС.

Другой трафик отсутствует (не берем в расчет служебный трафик протоколов OSPF, ARP, DNS, и т. д. в сумме потребляющий менее 1% пропускной способности каналов). Результаты моделирования в приведенной топологии свидетельствуют, что при наличии альтернативных маршрутов оценка RTT может варьироваться в зависимости от доли пакетов, прошедших по тому или иному маршруту. Если альтернативные каналы к тому же имеют разные скорости, то это увеличивает разброс значений RTT. Увеличение круговой задержки в свою очередь пропорционально снижает производительность TCP – соединения. Использование балансировки трафика пропорционально доступной ПС, как и следовало ожидать, позволяет снизить колебания круговой задержки и оптимизировать загрузку каналов.

Что касается пропускной способности, то в случае множества альтернативных маршрутов доступная пропускная способность увеличивается, а оценка доступной пропускной способности соединения при асимметрии путей будет равна минимуму пропускной способности прямого и обратного маршрута. В частности, для асимметричных приложений, передающих большие объемы данных по запросам (HTTP, FTP, …) и работающих через протокол TCP, асимметрия пропускной способности не снижает производительности данных приложений, если соответствует асимметрии объемов передаваемых потоков данных.

При разработке сценария моделирования и настройке маршрутизаторов использовался эмулятор Dynamips [6]. Эксперименты проводились на маршрутизаторах Cisco 3620 с IOS 12.4. И хотя виртуализированные маршрутизаторы не могу служить платформой для исследования производительности, качественное соотношение результатов тестов было одинаковым и на виртуальной и на реальной платформе.

Выводы

В статье впервые были систематизированы причины и последствия асимметричной маршрутизации в Internet. Причем последствия асимметрии могут проявляться как в сетях со сложной топологией, так и в небольших оконечных сетях. Влияние асимметрии на характеристики трафика наблюдаются даже в сетях с простой топологией, что подтверждается путем моделирования как на реальном оборудовании, так и на виртуализированной платформе.

В дальнейшем планируется исследование особенностей численной оценки асимметрии. Учитывая описанное влияние асимметрии маар-

шрутов, желательно минимизировать возможность возникновения асимметрии еще на этапах проектирования и развертывания сетей. Частично сгладить последствия существования асимметрии маршрутов можно путем правильного выбора стратегии планирования при балансировке трафика (по-пакетно, по адресу отправителя, по адресу получателя, по паре адрес отправителя + адрес получателя). Этим можно уменьшить количество переупорядоченных пакетов и снизить колебания задержки. Однако, полностью исключить асимметричную маршрутизацию практически невозможно, а искусственная симметрия при ручной подстройке будет оказывать негативное влияние и расплатой будет уменьшение избыточности топологии, повышение сложности администрирования и диагностики неисправностей.

Таким образом, асимметрия маршрутов является фактором неопределенности в работе сетей, важным аспектом измерения характеристик трафика, моделирования и управления сетями. Исходя из этого, с явлением асимметрии придется считаться и в Интернете и в больших объединенных сетях с избыточно структурой. Так, по некоторым оценкам [5,7], в будущем маршрутная асимметрия будет усиливаться, и приближаться к периферийным сетям.

Список литературы

1. V. Paxson. Measurements and Analysis of End-to-End Internet Dynamics. Technical report, U.C. Berkeley, 1997. Ph.D. Thesis.
2. Y. He, M. Faloutsos, and S. Krishnamurthy. Quantifying routing asymmetry in the internet at the AS level //In *Proceedings of the GLOBECOM 2004 Conference*, Dallas, Texas, USA, 2004. IEEE Computer Society Press.
3. Z. Morley Mao, L. Qiu, J. Wang, and Y. Zhang. On AS-level path inference //*Proceedings of the 2005 ACM SIGMETRICS international conference on Measurement and modelling of computer systems*, pages 339–349, New York, NY, USA, 2005. ACM.
4. Столлингз В. Современные компьютерные сети.: Пер. с англ.– Спб.: Питер, 2003.– 783с.
5. M. Crotti, F. Gringoli, and L. Salgarelli, Impact of Asymmetric Routing on Statistical Traffic Classification // IEEE GLOBECOM, Honololou, Hawaii, USA, 2009.
6. Демчинский В.В., Дорогой Я.Ю., Дорошенко Е.С. Виртуализация сетей передачи данных в dynamips //Вісник НТУУ “КПІ”. Інформатика, управління та обчислювальна техніка: Зб. наук. пр. – К.: Век+, - 2009. - №51. – 144 -146 с.
7. W. John, S. Tafvelin, and T. Olovsson, Passive Internet Measurement: Overview and Guidelines based on Experiences //Computer Communications, vol. 33, no. 5, pp. 533..550, 2010.

УДК 004.031.42

БОЛДАК А.О.,
ПУСТОВИТ М.А.

АРХИТЕКТУРА ИНФОРМАЦИОННОЙ СИСТЕМЫ ИНТЕЛЛЕКТУАЛЬНОЙ ОБРАБОТКИ ДАННЫХ

В работе сформулированы требования, которым должна удовлетворять информационная система коллективного пользования для интеллектуальной обработки данных. На основе этих требований определены архитектурные принципы и программные механизмы, составляющие основу программного каркаса для построения таких систем. Использование предложенной архитектуры обеспечивает повышение эффективности разработки и технического сопровождения информационных систем коллективного пользования, предназначенных для обработки больших информационных массивов.

In the article formulated the requirements which the information system of collective use for data mining has to satisfy. On the basis of these requirements are defined architectural principles and software mechanisms that form the basis of software framework for building such systems. Using the proposed architecture enhances the efficiency of development and technical support information systems for collective use, designed for processing large data arrays.

Введение

В последние десятилетия в научной среде все большее значение приобретает компьютерное моделирования и эксперименты, поставленные с его помощью. Область вычислительной науки (computational science)[1], занимающаяся проблемами создания инструментов для проведения подобных экспериментов так же бурно развивается. В некоторых случаях компьютерное моделирование выигрывает у натурного эксперимента по стоимости и по времени проведения эксперимента, дает более точные результаты.

Для проведения компьютерного моделирования сложных систем и анализа полученных результатов требуются значительные вычислительные мощности и объемы памяти для хранения данных. Актуальным становится решение задачи создания общедоступных информационных систем, которые могли бы использоваться большим количеством пользователей для проведения интеллектуальной обработки [2] больших информационных массивов.

Существующие сегодня решения подобного назначения, такие как Google Public Data Explorer [3] и Gapminder [4] не решают поставленную задачу в полной мере. Они являются закрытыми коммерческими системами, которые специализируются на визуализации данных. Расширение функциональности этих систем, использование непредусмотренных разработчиками источников данных и добавление новой функтиональности не представляется возможным.

Таким образом, цель настоящей работы состоит в повышении эффективности разкаботки и технического сопровождения информационных систем коллективного пользования для интеллектуальной обработки данных за счет определения архитектурных принципов, программных механизмов и информационных технологий, определяющих программные каркасы для построения таких систем.

1. Постановка задачи

Система статистической обработки данных должна удовлетворять требованиям, которые условно можно разделить на две категории. Одна из них связана с обеспечением требований, предъявляемых пользователем. Вторая же определят возможность быстрой разработки и компоновки как системы в целом, так и её компонентов.

С точки зрения пользователя такая система должна удовлетворять следующим требованиям:

- быть способной обрабатывать большие объемы информации и обрабатывать данные в «сыром» (raw data) необработанном формате;
- система должна быть доступна широкому кругу пользователей;
- система должна быть простой в использовании даже для пользователей, не знакомых с программированием;

- система должна позволять пользователю обрабатывать данные такого объема, который превышает объем хранилищ данных его оборудования.

Системные требования состоят в следующем:

- система должна быть масштабируемой (обеспечивать увеличение вычислительных ресурсов и объёмов хранения данных в соответствии с нагрузкой);
- внутренние механизмы обработки должны быть скрыты от пользователя. Они могут содержать алгоритмы являющиеся ноу-хау автора или использовать информацию, которая не находится в свободном доступе;
- архитектура и программные механизмы системы должны предусматривать возможность добавления новой функциональности и (или) реконфигурации системы.

2. Анализ современных программных технологий и средств разработки систем обработки даннях

Возможное превышение ресурсов, необходимое для обработки данных, над возможностями пользовательского оборудования заставляет использовать в качестве одного из архитектурных принципов организации системы её разделение на клиентские и серверные компоненты. Такая организация называется *клиент-серверной* архитектурой, при применении которой обработка данных производится на некоторых, возможно, удаленных, серверах [5]. В этом случае клиент только совершает запросы и получает результаты вычислений, т.е. оборудование пользователя играет роль тонкого клиента.

При этом могут использоваться сетевые протоколы уровня приложений и сеть Internet как коммуникационная среда. Это обеспечивает возможность использования разрабатываемой системы широким кругом пользователей.

Облачные вычисления. Использование клиент-серверной архитектуры с комуникационной средой на основе сети Internet позволяет определить разрабатываемую систему в терминах облачных вычислений, то есть парадигмы, в рамках которой информация постоянно хранится и обрабатывается на удаленных серверах в интернете и только временно кэшируется на клиентской стороне [6].

Существуют множество взглядов пользователя на системы облачных вычислений. Ему может предоставляться аренда аппаратных средств, виртуальная вычислительная среда или доступ к функционально законченному программному продукту.

Система статистической обработки данных с точки зрения пользователя должна выглядеть как *Software as a Service* (SaaS), которая предоставляет неподготовленному пользователю систему с простым и понятным интерфейсом [7]. При этом пользователь не должен заботиться, о поддержке работоспособности службы и развертывании ее на собственном оборудовании.

Выбор концепции облачных вычислений скрывает механизмы работы инструментов обработки данных, что является одним из требований к разрабатываемой системе.

Сервис-ориентированная архитектура. Принципы облачных вычислений могут быть реализованы в системах с различными архитектурами [8]. Одной из таких архитектур является *Сервис-ориентированная архитектура* (англ. SOA, service-oriented architecture), суть которой состоит в модульном подходе к разработке программного обеспечения, основанный на использовании сервисов (служб) со стандартизированными интерфейсами. SOA подразумевает множество слабосвязанных (либо вообще не связанных) служб, которые объединены общим коммуникационным механизмом.

В такой системе передача запросов, данных и служебной информации, происходит посредством коммуникатора. Каждая из служб занимается решением только одной задачи. Это позволяет многократно использовать службы для решения различных прикладных задач.

Сервисная шина предприятия. Одним из путей реализации SOA является использование механизма, называемого Enterprise Service Bus (ESB). Под данным понятием подразумевают, в некоторых случаях архитектурное решение, в других – набор программного обеспечения (ПО), решающего задачи взаимодействия компонентов системы. С точки зрения разработчика системы удобнее рассматривать ESB как набор ПО [9].

Для пользователя системная шина реализует концепцию единой точки доступа к приложению. Это подразумевает, что создается диспетчер, который несет ответственность за проведение запросов и выдачи результатов [11].

Таким образом, в архитектуру системы вводится промежуточное звено (шина), которое организует взаимодействие между подключаемыми к нему службами. При этом способы подключения служб к шине стандартизированы. Такая реализация хорошо масштабируется и расширяется.

Сегодня существует достаточно большое количество программных комплексов реализующих ESB (к примеру, IBM WebSphere ESB [12], JBoss Enterprise SOA Platform [13] и др.), что свидетельствует о целесообразности и эффективности описанного подхода к организации взаимодействия в слабосвязанных программных системах.

Одним из способов реализации подсистем для ESB являются веб-службы (англ. web service), которые представляют собой программные системы, идентифицируемые строкой URI, чьи общедоступные интерфейсы определены на языке XML [10]. Описание такой веб-службы может быть найдено другими подсистемами, которые могут взаимодействовать с ней, используя механизм передачи XML-сообщений. Эти сообщения передаются при помощи интернет-протоколов уровня приложений. Подобные протоколы, которые можно разделить на две категории: протоколы, основанные на использовании доступа к информационным ресурсам и протоколы, основанные на удаленном вызове процедур (RMI – remote method invocation).

Первую категорию называют REST (Representational State Transfer) протоколами. В случае использования этой архитектуры агенты пользователей взаимодействуют с ресурсами, которыми может быть всё, что можно поименовать и представить. Взаимодействие осуществляется с помощью единого интерфейса стандартных команд HTTP (GET, POST, PUT и DELETE). Для взаимодействия важно также объявить тип мультимедийного ресурса, который указывается с помощью заголовка типа содержимого HTTP. В этом случае вся информация, необходимая для обработки запроса ресурса, содержится в нем самом [14].

Протоколы второй категории используют механизм удаленного вызова процедур RMI (Remote Method Invocation), среди которых наиболее распространенным является протокол SOAP (Simple Object Access Protocol). Он предполагает передачу структурированных сообщений на основе XML. В отличие от REST, SOAP – это защищенный протокол с гарантированной передачей сообщений [15].

Каждый из описанных протоколов имеет свои достоинства и недостатки. Для обеспечения возможности интеграции системы статистической обработки данных с другими системами, необходимо реализовывать не один, а несколько протоколов, с использованием которых осуществляется передача сообщений.

3. Использование шаблонов проектирования для разработки программных механизмов системы статистической обработки данных

При реализации архитектурного решения ESB для системы статистической обработки данных ключевым аспектом является эффективная реализация программных механизмов, которые обеспечивают унификацию интерфейсов подсистем, методов доступа к данным, а так же их кеширование и визуализацию. При реализации этих программных механизмов могут быть использованы типовые решения уровня структурного проекттирования – шаблоны.

Унификация интерфейса подсистем. Общий интерфейс (API - application programming interface), который предоставляет система, не должен содержать всех функций, предоставляемых подсистемами, поскольку среди них могут быть утилитарные системные функции. Для скрытия внутренней структуры и создания общего интерфейса применяют шаблон Фасад [16, с.183], который можно использовать на двух уровнях: на уровне каждой подсистемы, в случае если она состоит из нескольких блоков и на уровне системы в целом. Во втором случае фасад представляет API системы.

Общий вид механизма интеграции подсистемы приведен на **Ошибка! Источник ссылки не найден.**.

ESB часто внедряется в системы, в которых уже созданы решения для многих задач. Но части системы, в которых реализованы решения, могут быть не выделены в отдельные структурные блоки. Даже если разделение на блоки уже произведено, их необходимо оформить в виде веб-служб со стандартизированными интерфейсами. В этом случае целесообразно использовать шаблон Адаптер [16, с. 141]. В этом случае в роли клиента выступает Диспетчер (ESB), в роли адаптируемого объекта – реализация службы.

Визуализация данных. Визуализация объектов это дополнительная функциональность,

которую удобно выносить в отдельные специализированные подсистемы. В таких случаях применяют шаблон Декоратор [16, с.173], цель применения которого состоит в добавлении функциональности объекту без порождения новых классов. Местоположение декоратора в общей структуре системы показано на **Ошибка! Источник ссылки не найден.**.

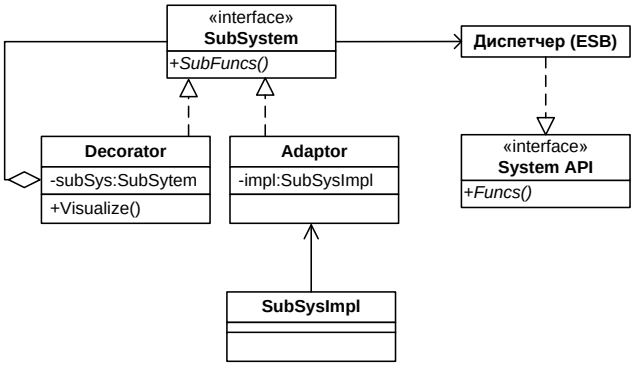


Рис. 1. Адаптер для интеграции подсистемы и декоратор для визуализации данных

Унификация методов доступа к данным. Источники данных, с которыми должна работать система, могут значительно различаться по устройству. Для обеспечения единообразного способа доступа к хранилищам данных используется шаблон Мост [16, с. 152] **Ошибка! Источник ссылки не найден.**, который

предусматривает разделение абстрактного интерфейса и реализации методов доступа. Следствием такого разделения является возможность создания нескольких реализаций, которые с точки зрения системы будут неотличимы в смысле протокола взаимодействия с источником данных.

Кеширование данных. Необходимость передачи и обработки больших массивов информации заставляет использовать специальные механизмы, которые обеспечивают минимизацию этих затрат. К таким механизмам относится кеширование данных и результатов запросов, реализация которых возможна с использование шаблона Заместитель (Proxy) [16, с. 203], (см. **Ошибка! Источник ссылки не найден.**). При использовании данного паттерна, тяжеловесный объект замещается объектом-заместителем. В данном случае в роли тяжеловесного объекта может выступать либо объект, хранящийся в удаленном хранилище данных, либо объект, выполнение функций которого может занимать значительное время. Заместитель ведет учет всех запрошенных данных и выполненных запросов на обработку, если происходит повторный запрос, то запрашивающей стороне отдается локальная копия. Это экономит вычислительные ресурсы и ресурсы сети.

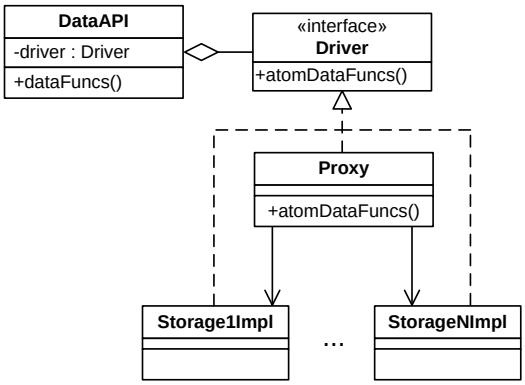


Рис. 2. Мост для создания API доступа к данным и заместитель для кеширования данных и запросов

4. Программный механизм обработки запросив

Координация работы подсистем реализуется диспетчером при условии, что эти подсистемы проектируются независимо друг от друга как самостоятельные функциональные блоки. При реализации такого диспетчера целесообразно использовать шаблон Посредник (Mediator) [16, с. 263].

Как видно из Рис. 3, на котором приведен пример последовательности действии при обработке запроса, вся ответственность за выполнение запроса возложена на Диспетчер

Концепция шаблона заключается во включении в систему объекта кординирующего работу ее частей. В этой системе обязанности посредника возлагаются на Диспетчер. При применении такого подхода, все подсистемы проектируются независимо друг от друга, как самостоятельные функциональные блоки.

Следует отметить, что в роли Клиента, может выступать не только пользователь, но и любая из подсистем.

Регистрация подсистем. В предложенной архитектуре диспетчер, играющий интеграционную роль, должен обладать функциональностью, обеспечивающей возможность расширения функциональности системы и (или) её реконфигурации. В этом случае элементами являются подсистемы в виде веб-сервисов.

Таким образом, диспетчер должен сохранять информацию о конфигурации системы. Реконфигурация системы или её функциональное расширение связанно с изменением этой информации.

Последовательность действий при регистрации подсистемы представлена на *Рис. 2.*

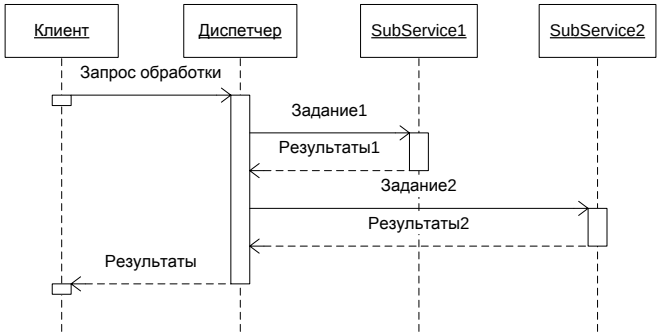


Рис. 3. Диаграмма последовательности выполнения запроса к системе

Для регистрации новой службы в системе пользователь должен направить соответструющий запрос диспетчеру, в котором указать URL (Uniform Resource Link) добавляемой системы.

Производится обращение к подсистеме с целью получения метаданных, описывающих подключаемую службу.

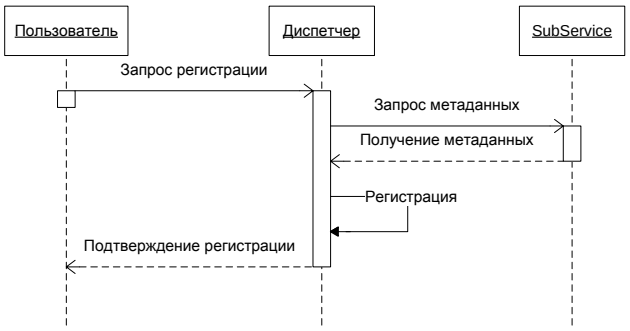


Рис. 2. Диаграмма последовательности регистрации новой службы в системе

После получения информации, диспетчер верифицирует её. Каждая служба может обрабатывать запросы нескольких типов, поэтому в метаданных системы должны содержаться сведения обо всех обрабатываемых типах запросов. Каждому типу должен быть присвоен уникальный URI (Universal Resource Identification) [17], состоящий из пути URL (совпадающий с URL добавляемой системы) и имени типа URN (Universal Resource Name). Диспетчер формирует таблицу маршрутизации запросов, которая используется им в процессе обработки запросов (см. Рис. 3).

Необходимым условием реализации описанной схемы является то, что API подсистемы обязательно должно содержать функцию, отдающую по запросу информацию, описывающую подсистему.

Заключение

В работе сформулированы требования, которые предъявляются различными категориями заинтересованных лиц, к информационной системе коллективного пользования для интеллектуальной обработки данных. Исходя из этих требований, определены принципы архитектурной организации и программные механизмы, составляющие программный каркас для построения такой системы. Эти принципы состоят в следующем.

Установлено, что выбор модели облачных вычислений удовлетворяет требованиям доступности системы и сокрытия механизмов обработки данных, что является необходимыми требованиями для обеспечения возможности использования системы широким кругом пользователей, не знакомых с программированием.

Использование принципов организации сервис-ориентированной архитектуры при реализации программного каркаса такой системы обеспечивает возможность ее масштабирования, а также расширения ее функциональности.

Основой программного механизма является диспетчер системы, реализующий функциональность системной шины предприятия (enterprise service bus), которая используется для организации взаимодействия всех компонентов системы. Это предполагает необходимость унификации интерфейсов подсистем, механизмов доступа к данным, кеширования и визуализации результатов их обработки.

Использование предложенной архитектуры информационной системы коллективного пользования для интеллектуальной обработки данных позволяет повысить эффективность разработки и технического сопровождения систем подобного назначения.

Список литературы

1. Syamlal, Madhava computational science: Enabling Technology Development [Электронный ресурс]// Syamlal, Madhava, Guenther, Chris, Cugini, Anthony, Ge, Wei, Wang, Wei, Yang, Ning, Li, Jinghai – Pe-

жим доступа: http://findarticles.com/p/articles/mi_qa5350/is_201101/ ai_n56829874/. Дата обращения: 28.04.2011
2. Christopher Clifton Data mining [Электронный ресурс] // C. Clifton – Режим доступа: http://www.britannica.com/EBchecked/topic/1056150/data-mining Дата обращения: 26.04.2011
3. Google Public Data Explorer [Электронный ресурс]// Режим доступа: http://www.google.com/publicdata/home. Дата обращения: 14.04.2011
4. Gapminder [Электронный ресурс]// Режим доступа: http://www.gapminder.org/. Дата обращения: 14.04.2011
5. Коржов В. Многоуровневые системы клиент-сервер [Электронный ресурс]// В. Коржов – URL: http://www.osp.ru/nets/1997/06/142618/#part_1. Дата обращения: 14.04.2011
6. Peter Mell, Tim Grance The NIST Definition of Cloud Computing // National Institute of Standards and Technology, Information Technology Laboratory - Version 15, 10-7-09
7. Колесов А. Модель SaaS – в мире и в России / А. Колесов // Журнал Byte Россия [Электронный ресурс]. - №10 (119), октябрь 2008 – Режим доступа: http://ej.kubagro.ru/plinks.asp
8. Сервис-ориентированная архитектура [Электронный ресурс] : Материал из Википедии – свободной энциклопедии : Версия 32077064, сохранённая в 14:44 UTC 19 февраля 2011 / Авторы Википедии // Википедия, свободная энциклопедия. – Электрон. дан. – Сан-Франциско: Фонд Викимедиа, 2011. – Режим доступа: http://ru.wikipedia.org/?oldid =32077064
9. Сервисная шина предприятия // Википедия. [2011–2011]. Дата обновления: 04.04.2011. Режим доступа: http://ru.wikipedia.org/?oldid=33340781 (дата обращения: 04.04.2011) http://www.w3.org/TR/ws-gloss/
10. Hugo Haas, Allen Brown Web Services Glossary [Электронный ресурс]/ Hugo Haas, Allen Brown. – Режим доступа: http://www.w3.org/TR/ws-gloss/ Дата обращения: 11.04.2011
11. Дональд Фергюсон, Марша Стоктон Модель программирования SOA для реализации Web-сервисов, Часть 1: Введение в модель программирования SOA [Электронный ресурс] // Фергюсон, Стоктон Режим доступа: http://www.ibm.com/developerworks/ru/ library/ws-soa-progmodel/
12. IBM WebSphere [Электронный ресурс]// Режим доступа: http://www-01.ibm.com/software/integration/wsesb/about/. Дата обращения: 14.04.2011
13. JBoss Enterprise SOA Platform [Электронный ресурс]// Режим доступа: www.jboss.com/pdf/SOA_infosheet.pdf. Дата обращения: 14.04.2011
14. Фландерс Д. Введение в службы RESTful с использованием WCF / Д. Фландерс // Журнал MSDN [Электронный ресурс]. – январь 2009 – Режим доступа: http://msdn.microsoft.com/ru-ru/magazine/dd315413.aspx
15. Маквитти Л. REST как альтернатива SOAP / Л. Маквитти // Сети и системы связи [Электронный ресурс]. Режим доступа: http://www.ccc.ru/magazine/depot/07_01/read.html?0502.htm
16. Гамма Э., Хелм Р., Джонсон Р., Влиссидес Дж. Приемы объектно-ориентированного проектирования. Паттерны проектирования. // СПб: Питер, 2001. – 368 с.
17. URI [Электронный ресурс] : Материал из Википедии – свободной энциклопедии : Версия 33228273, сохранённая в 21:27 UTC 30 марта 2011 / Авторы Википедии // Википедия, свободная энциклопедия. – Электрон. дан. – Сан-Франциско: Фонд Викимедиа, 2011. – Режим доступа: http://ru.wikipedia.org/?oldid=33228273

УДК 519.876.5:004.94

СТЕЦЕНКО И.В.

ФОРМАЛЬНОЕ ОПИСАНИЕ СИСТЕМ СРЕДСТВАМИ ПЕТРИ-ОБЪЕКТНЫХ МОДЕЛЕЙ

В статье рассматривается новый способ формального описания систем, основывающийся на объектно-ориентированной технологии и стохастической временной сети Петри. Получены уравнения состояний стохастической временной сети Петри с конфликтными и многоканальными переходами. Предложено понятие Петри-объекта и разработана технология конструирования имитационной модели системы с использованием Петри-объектов.

The article considers the new formal methods for system's description based on object-oriented methodology and stochastic timed Petri net. The state equation of stochastic timed Petri net with conflict and multi-transitions is received. The concept of Petri-object is proposed and construction technology of simulation model with Petri-objects using is developed.

Введение

Модели сетей Петри являются универсальным средством формализации процессов функционирования дискретно-событийных систем. Однако их использование для целей имитационного моделирования ограничено, во-первых, тем, что приходится использовать большое количество элементов даже для простых систем, во-вторых, тем, что отсутствие математической теории стохастических временных сетей Петри приводит к разнообразию подходов к построению алгоритмов имитации. В [1], [2] предлагается использовать блочную структуру построения моделей сетей Петри, что позволяет создавать подобные фрагменты сетей Петри простым копированием или вставкой соответствующего блока. Это в определенной степени облегчает составление моделей, но все же не решает проблему в случае, когда система состоит из сотен подобных элементов, взаимосвязанных между собой. В [3] предлагается разбивать большую сеть Петри на функциональные подсети, что позволяет исследовать вместо свойств сети Петри свойства ее подсетей.

В последнее время появляются научные работы, в которых рассматривается объединение объектно-ориентированного подхода и моделирования сетями Петри тем или иным способом в зависимости от задачи, которая решается ([4],[5],[6],[12]). Так, термин „объектно-ориентированные сети Петри (object oriented Petri net, OPN)" закрепился за расширенным понятием сети Петри, в котором существуют специфические элементы сети Петри (позиции, переходи или маркеры), выполняющие функции объединения составляющих частей (которые и являются объектами) сети Петри [4]. Одной из реализаций этого подхода является язык LOOPN Чарльза Лакоса, в котором термин объектно-ориентированная сеть Петри означает, что маркеры сети Петри являются объектами в терминах объектно-ориентированного программирования, а также отдельные фрагменты сети Петри могут служить объектами [5]. Термин „иерархическая объектно-ориентированная сеть Петри с временными задержками (timed hierarchical object-oriented Petri net)" введен в работе [6] . Есть также публикации, в которых объекты и методы представляют сетями Петри с целью перевести объектно-ориентированный подход в язык сетей Петри [7].

Во всех разработках сеть Петри модифицируется тем или другим способом, чтобы приспособить ее к требованиям объектно-ориентированной парадигмы. Эти модификации приводят к значительному усложнению инструмента сетей Петри. Кроме этого, указанные подходы позволяют реализовывать большие сети Петри только теоретически.

В настоящей работе разработана технология моделирования систем, которая, в отличие от существующих технологий, позволяет создавать модели больших систем средствами стохастических временных сетей Петри с конфликтными и многоканальными переходами.

Уравнения состояний стохастической временной сети Петри с конфликтными и многоканальными переходами

Рассмотрим сеть Петри $\mathrm{PetriNet} = (\mathbf{P}, \mathbf{T}, \mathbf{A}, \mathbf{W}, \mathbf{K}, \mathbf{I}, \mathbf{R})$, заданную множеством позиций $\mathbf{P} = \{P\}$; множеством переходов $\mathbf{T} = \{T\}$, $\mathbf{P} \cap \mathbf{T} = \varnothing$; множеством дуг

$\mathbf{A} \subseteq (\mathbf{P} \times \mathbf{T} \cup \mathbf{T} \times \mathbf{P})$; множеством натуральных чисел $\mathbf{W}: \mathbf{A} \cup \mathbf{I} \to \mathrm{N}$, задающих кратности дуг (количество связей); множеством пар значений $\mathbf{K} = \{(c_T, b_T) \mid T \in \mathbf{T}, c_T \in N, b_T \in [0;1]\}$, задающих приоритет и вероятность запуска переходов; множеством неотрицательных действительных значений $\mathbf{R}: \mathbf{T} \to \Re_+$, характеризующих временные задержки в переходах.

Для обозначения множества входных и множества выходных позиций перехода T будем пользоваться обозначениями, предложенными в [8], – ${}^\bullet T$ и $T^\bullet$ соответственно, множества входных переходов и множества выходных переходов позиции P – ${}^\bullet P$ и $P^\bullet$ соответственно.

Формальное описание стохастической временной сети Петри получено на основе уравнений состояний базовой сети Петри [8] и формального описания временной сети Петри с детерминированными временными задержками, которое содержится в работе [9].

Состояние сети Петри в момент времени t описывается состоянием ее позиций $\mathbf{M}(t)$ и состоянием ее переходов $\mathbf{E}(t)$: $(\mathbf{M}(t), \mathbf{E}(t)) = \mathbf{S}(t)$. Состояние позиций однозначно определяется маркировкой сети Петри в момент времени t: $\mathbf{M}(t) = \{M_P(t) \mid M_P(t) \in Z_+, P \in \mathbf{P}\}$, где Z_+ – множество целых неотрицательных чисел. Состояние переходов определяется множеством $\mathbf{E}(t) = \{E_T(t) \mid T \in \mathbf{T}\}$, где состояние каждого перехода $E_T(t)$ определяется множеством моментов выхода из переходов маркеров, которые на момент времени t находятся в переходе:

$$E_T(t) = \{\ [E_T(t)]_q \mid [E_T(t)]_q \in \Re_+, q \in \mathrm{N} \tag{1}$$

где q – номер канала перехода, $q = 1,2..|E_T(t)|$, $|E_T(t)|$ – количество занятых (активных) каналов в момент времени t.

Если выход маркеров из перехода не ожидается, что соответствует пустому (не активному) переходу, то множество $E_T(t)$ этого перехода содержит только одно значение ∞, означающее, что «в ближайшее время не ожидается выход маркеров из перехода»:

$$E_T(t) = \{\infty\} \tag{2}$$

Функционирование временной сети Петри заключается в выполнении упорядоченной во времени последовательности событий, соответствующих ее переходам. Рассмотрим моменты времени $t_0, t_1, ...t_n...$ такие, что: $\forall t \in (t_{n-1}, t_n)$ $\mathbf{S}(t) = \mathbf{S}(t_{n-1})$ т.е. изменение состояния сети Петри в интервале времени (t_{n-1}, t_n) не происходит.

В каждый момент времени t_{n-1} определяется момент возникновения ближайшего события, и время продвигается в этот момент времени t_n:

$$t_n = \min_T \left(\min_q [E_T(t_{n-1})]_q \right), t_n \geq t_{n-1} \tag{3}$$

Преобразование $D^+: \mathbf{S}(t_{n-1}) \to \mathbf{S}(t_n)$ сети Петри, соответствующее выходу маркеров из переходов имеет вид:

$$\forall P \in \mathbf{P} \quad M_P^+(t_n) = \\ = M_P(t_{n-1}) + \sum_{T \in {}^\bullet P} Y(T, t_n) \cdot W_{T,P} \mid s_T(t_{n-1}) \mid \tag{4}$$

$$\forall T \in \mathbf{T} \mid Y(T, t_n) = 1 \\ E_T^+(t_n) = \\ = \begin{cases} \{\infty\} \ if \ \ |s_T(t_{n-1})| = |E_T(t_{n-1})|, \\ E_T(t_{n-1}) \setminus \{[E_T(t_{n-1})]_q \mid q \in s_T(t_{n-1})\} \ else. \end{cases} \tag{5}$$

где $s_T(t)$ – множество каналов перехода, которым соответствует наименьший из всех моментов выхода маркеров из перехода, $s_T(t) = \{q \in \mathrm{N} \mid [E_T(t)]_q = \min_q [E_T(t)]_q\}$, $Y(T, t_n)$ – предикат, определяющий множество переходов $T \in \mathbf{T}$, для которых осуществляется выход маркеров в момент времени $t_n \in \Re$:

$$\left(\min_q [E_T(t_{n-1})]_q = t_n \right) \Rightarrow Y(T, t_n) = 1 \\ \left(\min_q [E_T(t_{n-1})]_q \neq t_n \right) \Rightarrow Y(T, t_n) = 0 \tag{6}$$

Преобразование $D^-: \mathbf{S}(t_n) \to \mathbf{S}(t_n)$ сети Петри, соответствующее входу маркеров в переходы сети Петри имеет вид:

$$\forall P \in \mathbf{P} \quad M_P(t_n) = \\ = M_P^+(t_n) - \sum_{T \in P^\bullet} W_{P,T} \cdot X(T, t_n) \tag{7}$$

$$\forall T \in \mathbf{T} \mid X(T, t_n) = 1 \\ E_T(t_n) = \\ = \begin{cases} \{t_n + R_T\} \ if \ \ \min_q [E_T(t_{n-1})]_q = \infty, \\ E_T^+(t_n) \cup \{t_n + R_T\} \ else. \end{cases} \tag{8}$$

где $X(T, t_n)$ – предикат, определяющий множество переходов $T \in \mathbf{T}$, для которых осуществляется вход маркеров в момент времени $t_n \in \Re$:

$$T \in \Psi'(t_n) \Rightarrow X(T,t_n) = 1,$$
$$T \notin \Psi'(t_n) \Rightarrow X(T,t_n) = 0. \quad (9)$$

Множество $\Psi'(t_n)$ представляет подмножество множества переходов с выполненным условием запуска, которое формируется в результате выбора из конфликтных переходов, основывающегося на значениях приоритетов и вероятностей запуска переходов.

Преобразование $D^-(\mathbf{S}(t_n))$ представляет результат одного входа маркеров в переходы сети Петри. Количество маркеров во входных позициях многоканального перехода может позволять запуск не одного, а нескольких каналов этого перехода. Поэтому вход маркеров в переходы должен осуществляться многократно до тех пор, пока еще есть хоть один переход, который запускается. В противном случае может возникнуть ситуация, когда условие запуска перехода выполнено, но он не запущен в момент t_n и в результате продвижения времени сможет быть запущен только в момент времени $t > t_n$, что противоречит правилам функционирования временных сетей Петри.

Максимальное количество запусков перехода в момент времени t_n определяется величиной $\min\limits_{P \in {}^\bullet T}\left\{\dfrac{M_P^+(t_n)}{W_{P,T}}\right\}$, где операция деления является операцией деления целых чисел. Фактическое количество m входов маркеров в переходы обусловлено требованием, что в результате достигается маркировка сети Петри, в которой ни один из переходов не запускается:

$$m : \left(D^-\right)^m(\mathbf{S}(t_n)) : \bigvee_{\mathrm{T}} Z(T,t_n) = 0 \quad (10)$$

где $\left(D^-\right)^m = D^- \circ D^- \circ D^- \ldots \circ D^-$ - результат m - кратного применения преобразования D^-, $Z(T,t_n)$ – предикат, определяющий множество переходов $T \in \mathbf{T}$ с выполненным условием запуска в момент времени $t_n \in \Re$:

$$\begin{aligned} &\left(\forall P \in {}^\bullet T \quad M_P^+(t_n) \geq W_{P,T}\right) \Rightarrow Z(T,t_n) = 1 \\ &\left(\exists P \in {}^\bullet T \quad M_P^+(t_n) < W_{P,T}\right) \Rightarrow Z(T,t_n) = 0 \end{aligned} \quad (11)$$

Пусть вектор $u_T(t_n) = \sum\limits_{i=1}^{m} X(T,t_n)_i$ представляет количество входов маркеров в переход T в серии входов маркеров в переходы $\left(D^-\right)^m$, соответствующей моменту времени t_n. Тогда преобразование $\left(D^-\right)^m$ описывается следующими уравнениями изменения состояния сети Петри:

$$\forall P \in \mathbf{P}$$
$$M_P(t_n) = M_P^+(t_n) - \sum_{T \in {}^\bullet P} W_{P,T} \cdot u_T(t_n), \quad (12)$$

$$\forall T \in \mathbf{T}$$
$$E_T(t_n) = $$
$$= \begin{cases} \underbrace{\{t_n + R_T\} \cup \ldots \cup \{t_n + R_T\}}_{u_T(t_n)} \ if \ \min\limits_q \left[E_T(t_{n-1})\right]_q = \infty, \\ E_T^+(t_n) \cup \underbrace{\{t_n + R_T\} \cup \ldots \cup \{t_n + R_T\}}_{u_T(t_n)} \ else. \end{cases} \quad (13)$$

Отметим, что множество значений $\{u_T(t_n)\}$ в общем случае зависит от случайного выбора перехода из множества запускающихся переходов.

Таким образом, имеем следующие уравнения состояний, описывающие функционирование стохастической временной сети Петри с многоканальными и конфликтными переходами:

$$t_n = \min_T\left(\min_q\left[E_T(t_{n-1})\right]_q\right), t_n \geq t_{n-1}, \quad (14)$$

$$\mathbf{S}(t_0) = \left(D^-\right)^m(\mathbf{S}(t_0)), \quad (15)$$

$$\mathbf{S}(t_n) = \left(D^-\right)^m\left(D^+(\mathbf{S}(t_{n-1}))\right), n = 1,2\ldots, \quad (16)$$

где преобразование $D^+ : \mathbf{S}(t_{n-1}) \to \mathbf{S}(t_n)$ описывается формулами (4),(5), преобразование $\left(D^-\right)^m : \mathbf{S}(t_n) \to \mathbf{S}(t_n)$ описывается формулами (12),(13) и условием (10).

Уравнения (14)-(16), которые получены, являются полным математическим описанием алгоритма функционирования временной стохастической сети Петри: уравнение (14) задает продвижение времени, уравнение (15) – преобразование состояния сети Петри, соответствующее начальному моменту времени, а уравнение (16) – преобразование состояния сети Петри, соответствующее всем следующим после начального моментам времени. Поскольку начальное состояние сети Петри обычно задается так, что маркеры находятся только в позициях (и не находятся в переходах), то в начальный момент времени осуществляется вход маркеров в переходы. Во все другие моменты времени осуществляется выход и вход маркеров в переходах.

Хотя уравнения (14)-(16) не накладывают никаких ограничений на размерность сети Петри, тем не менее, реализация с их помощью

больших сетей Петри, содержащих сотни элементов, приводит к непреодолимым трудностям, связанным с проверкой правильности составления сети Петри и отладкой алгоритма имитации. Для решения этой проблемы предлагается конструировать сеть Петри, моделирующей систему в целом, из фрагментов сети Петри, моделирующих элементы системы.

Понятие Петри-объекта

Объектно-ориентированная технология обладает важным свойством создания десятков, сотен и тысяч похожих элементов по одному образцу и справляется с воссозданием структуры очень сложных систем. Однако для того, чтобы объекты могли применяться для воссоздания не только структуры, но и динамики сложных систем, необходим некоторый универсальный единообразный способ задания динамики объектов. Таким способом, как показано в данной работе, могут стать сети Петри.

Введем класс объектов Петри-имитатор (PetriSim) как класс, который реализует имитацию некоторого реального объекта в соответствии с динамикой функционирования, заданной временной сетью Петри с конфликтными и многоканальными переходами (рис. 1).

Petri_Sim
— Net: Petri_net
— timeModeling: double
+ Do_T()
+ Start()
+ NextEvent()
+ DoStatistica()

Рис. 1. Основные поля и методы класса Петри-имитатор

Информация о сети Петри содержится в поле Net Петри-объекта. Метод Start() выполняет вход маркеров в переходы для начальной сети Петри в соответствии с преобразованием $\left(D^-\right)^m$. Метод NextEvent() осуществляет преобразование $\left(D^-\right)^m \circ D^+$ сети Петри, соответствующее моменту времени t_n, и продвигает время в следующий момент времени t_{n+1}. Метод DoStatistica() содержит алгоритм сбора информации о среднем количестве маркеров в позициях и среднем количестве маркеров в переходах. Для размещения информации о дополнительных действиях, которые выполняются при выходе маркеров из переходов, служит метод DoT().

Определение. Объекты моделирования, являющиеся наследниками объекта Петри-имитатор (PetriSim), назовем *Петри-объектами (*PetriObj*):*

$$\text{PetriObj} \xrightarrow{inherit} \text{PetriSim} \qquad (17)$$

Применение механизма наследования обеспечивает воссоздание всех полей и методов супер-объекта в саб-объекте. Сеть Петри объекта создается с помощью статичной функции класса NetLibrary и затем передается конструктору Петри-объекта в качестве аргумента. Конструктор Петри-объекта размещает переданную сеть Петри в поле PetriNet. Такой подход обеспечивает возможность использования одной и той же функции из класса NetLibrary для создания сетей Петри множества однотипных объектов, что, в свою очередь, гарантирует однотипность обращения к позициям и переходам таких объектов.

Петри-объекты, во-первых, владеют всеми свойствами обыкновенного объекта (как элемента ООП), во-вторых, имитируют функционирование объекта на основе сети Петри, описание которой содержится в поле PetriNet, в-третьих, являются конструктивными элементами, из которых составляется сеть Петри сложной системы.

Конструирование модели системы из Петри-объектов

Пусть модель системы состоит из Петри-объектов $Model = \{O_j\}$, $O_j \xrightarrow{inherit} \text{PetriSim}$, принадлежащих классам C_i: $O_j \subset C_i$. Пример диаграммы классов модели, составленной таким способом, представлен на рис. 2. Модель может быть составлена как непосредственно из Петри-объектов, так и из объектов-наследников Петри-объектов или из объектов, которые агрегируют Петри-объекты.

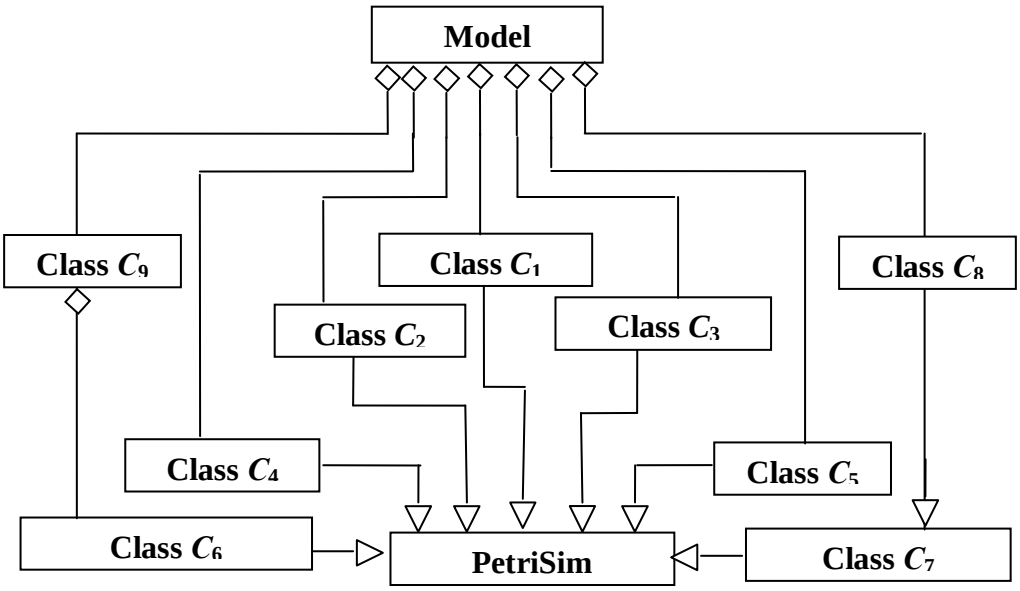


Рис. 2. Диаграмма классов модели системы

Сеть Петри объекта O_j, содержащуюся в его поле PetriNet, обозначим N_j: $N_j = (\mathbf{P}_j, \mathbf{T}_j, \mathbf{A}_j, \mathbf{W}_j, \mathbf{K}_j, \mathbf{I}_j, \mathbf{R}_j)$.

Связи Петри-объектов между собой осуществляются двумя способами:

1) с помощью общих позиций (например, позиция-счетчик либо позиция-ресурс):

$$\exists P \in \mathbf{P}_k \bigcap \mathbf{P}_l \qquad (18)$$

2) с помощью *инициализации событий* (из перехода объекта O_k передаются маркеры в позиции других объектов O_l в заданном количестве $w_{k,l}$):

$$\exists T \in \mathbf{T}_k, \exists P \in \mathbf{P}_l: \quad M_P^+(t_n) = M_P(t_n) + Y(T, t_n) \cdot w_{k,l} \qquad (19)$$

Алгоритмически существование общих позиций обеспечивается совпадением адресов памяти, где хранятся значения маркировок соответствующих позиций. Таким образом, общие позиции характеризуются общим доступом к значению маркировки этой позиции разных Петри-объектов.

Инициализация событий алгоритмически реализуется при запуске дополнительных действий, соответствующих выходу маркеров из перехода, методом DoT(). Тогда инициализацию событий можно выполнить не только для нескольких объектов, но и для множества объектов с помощью цикла. Уравнения (19) дополняют уравнения (4) преобразования D^+, а значит, математически инициализация событий переходом объекта O_k означает добавление к множеству выходящих позиций этого перехода еще одной позиции, принадлежащей объекту $O_l \neq O_k$:

$$w_{k,l} > 0 \Leftrightarrow \left(\exists T \in \mathbf{T}_k, \exists P \in \mathbf{P}_l : P \in T^\bullet\right) \qquad (20)$$

Поставим в соответствие всякой передаче маркеров из перехода одного объекта в позицию другого дугу $(T,P): T \in \mathbf{T}_k, P \in \mathbf{P}_l, w_{k,l} > 0$, и обозначим множество всех таких дуг объекта O_k $U_k = \{(T,P) \mid T \in \mathbf{T}_k, P \in \mathbf{P}_l, w_{k,l} > 0\}$, а соответствующие этим дугам значения кратностей $\mathbf{w}_k = \{(T,P) \mid T \in \mathbf{T}_k, P \in \mathbf{P}_l, w_{k,l} > 0\}$. Тогда сеть Петри модели системы, составленная из сетей Петри-объектов, имеет вид:

$$PetriNet = \left(\bigcup_j N_j\right) \bigcup U_j \qquad (21)$$

где U_j – множество дуг, вдоль которых объект O_j осуществляет инициализацию событий в других объектах, а объединение сетей Петри понимается в смысле объединения множеств ее позиций, переходов, дуг, кратностей, значений для решения конфликта, временных задержек в переходах:

$$\bigcup_j N_j: \ \mathbf{P} = \bigcup_j \mathbf{P}_j, \ \mathbf{T} = \bigcup_j \mathbf{T}_j, \ \mathbf{A} = \bigcup_j \mathbf{A}_j, \ \mathbf{W} = \bigcup_j \mathbf{W}_j, \ \mathbf{K} = \bigcup_j \mathbf{K}_j, \ \mathbf{R} = \bigcup_j \mathbf{R}_j. \qquad (22)$$

Связи (18), (19) обеспечивают, что модель, которая конструктивно состоит из Петри-объектов, описывается сетью Петри

$$PetriNet = (\mathbf{P}, \mathbf{T}, \mathbf{A}, \mathbf{W}, \mathbf{K}, \mathbf{R}) \qquad (23)$$

$$\mathbf{P}=\bigcup_j \mathbf{P}_j,\ \mathbf{T}=\bigcup_j \mathbf{T}_j,\ \mathbf{A}=\bigcup_j (\mathbf{A}_j \cup U_j),$$
$$\mathbf{W}=\bigcup_j (\mathbf{W}_j \cup \mathbf{w}_j),\ \mathbf{K}=\bigcup_j \mathbf{K}_j,\ \mathbf{R}=\bigcup_j \mathbf{R}_j. \quad (24)$$

Таким образом, конструирование модели из Петри-объектов обеспечивает, что функционирование модели описывается сетью Петри (23), являющейся объединением сетей Петри объектов, из которых она состоит.

Алгоритм имитации Петри-объектной модели состоит из следующих действий:

- Формировать список Петри-объектов;
- Осуществить вход маркеров в переходы Петри-объектов (Start());
- Пока не достигнут момент окончания моделирования
  - продвинуть время в момент ближайшего события;
  - определить список конфликтных объектов и выбрать объект из списка конфликтных объектов;
  - для выбранного объекта выполнить выход маркеров из переходов (Do_T(), StepEvent());
  - для всех других объектов осуществить вход маркеров в переходы (Start()).
- Вывести результаты моделирования.

Петри-объектная модель учебного процесса вуза

Технология конструирования модели из Петри-объектов возникла и апробировалась при разработке модели системы управления учебным процессом вуза [10]. На рис. 3 изображена Петри-объектная модель системы управления учебным процессом вуза, составленная из следующих Петри-объектов: Дисциплина, Студент, Преподаватель, Группа, Деканат, Контроль задолженностей. Как следует из этого примера, Петри-объекты представляют структурные элементы системы, очень часто являющиеся очевидными структурными единицами системы, которая исследуется.

Рассмотрим Петри-объект Дисциплина, который представляет изучение дисциплины в соответствии с учебным планом специальности (рис. 4). Общие с объектом Расписание позиции «Начало семестра», «Продолжается экзаменационная сессия» обеспечивают начало обучения и проведение экзамена по дисциплине в установленные деканатом временные интервалы. Для связи с этими позициями используются информационные связи, обозначенные пунктирными линиями. Использование информационных связей для моделирования систем управления описано в [11]. Общая позиция «Преподаватель» позволяет контролировать занятость преподавателя, например, другими дисциплинами.

Инициализация событий происходит в переходах Начало лекции (лабораторной работы, практического занятия), Модульный контроль, Защита лабораторной работы, Экзамен. Например, в результате запуска перехода Начало лекции передаются маркеры в позицию «Есть пара по расписанию» Петри-объекта Преподаватель и Петри-объекта Студент, что создает условия для выполнения событий «Проводит занятие по расписанию» Преподавателя и «Посещает занятие» Студента. Заметим, что при этом передача маркеров осуществляется во все Петри-объекты Студент, относящиеся к объекту Группа.

Метод DoT() в применении к модели учебного процесса, кроме передачи маркеров, содержит также действия, связанные с записями в журнал успеваемости и журнал посещаемости.

Правильность функционирования Петри-объекта сильно зависит от значений, задающих приоритет и вероятность запуска переходов. Например, событие «Начало лекции» и событие «Модульный контроль» при определенных условиях могут оказаться конфликтными. Так как в учебном процессе проведение занятий по расписанию является обязательным, то следует установить большее значение приоритета для перехода «Начало лекции».

Отметим, что создание модели системы управления учебного процесса с помощью Петри-объектов позволило воспроизвести такие процессы, как проведение занятий в группах

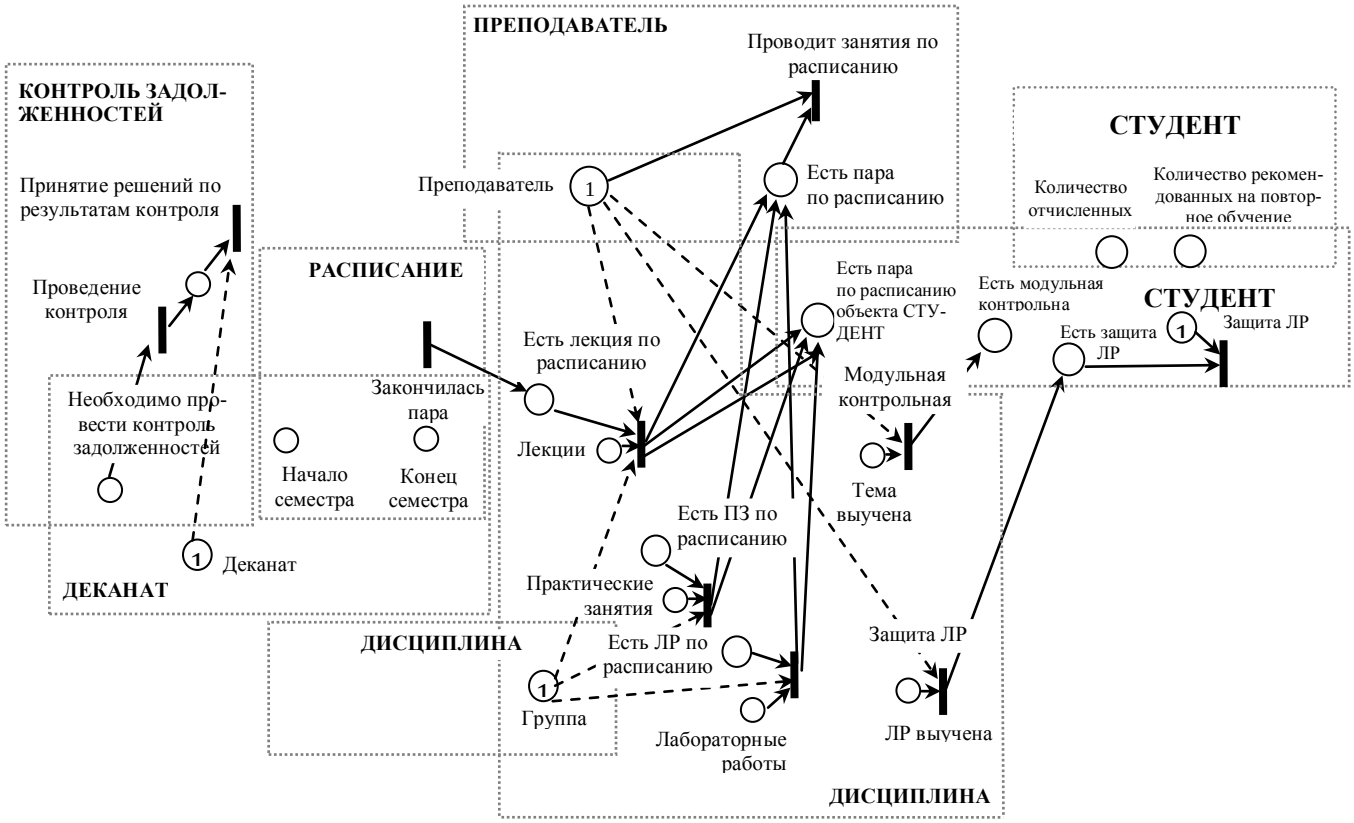


Рис. 3. Петри-объектная модель учебного процесса вуза

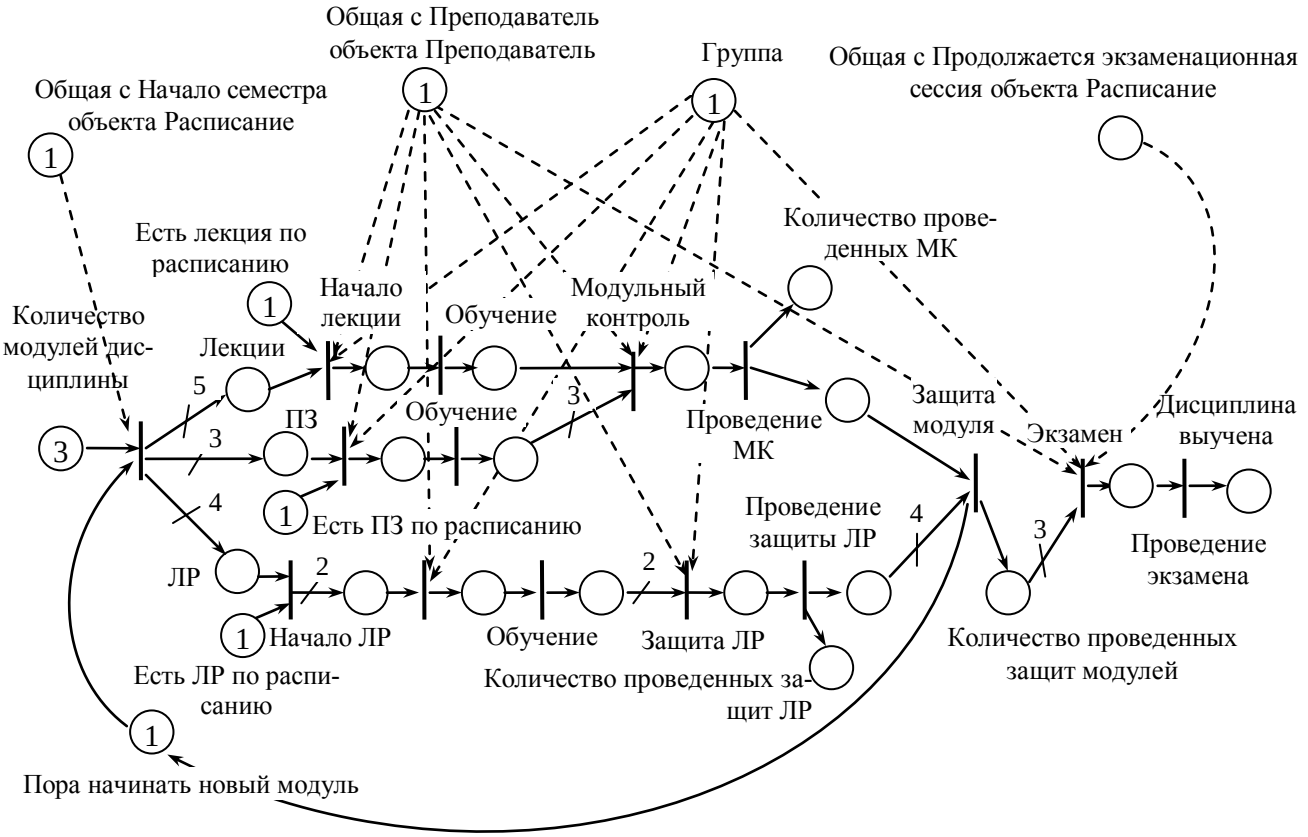


Рис. 4. Сеть Петри-объекта Дисциплина

студентов по расписанию, прием задолженностей преподавателями, сдача задолженностей по разным дисциплинам студентами, допуск студентов к сессии деканатом, принятие решений деканатом об отчислении студента или допуске к пересдаче.Реализация Петри-объектной

модели учебного процесса выполнена средствами языка программирования Java (J2SE) и интегрированной среды Netbeans IDE 6.5. Верификация модели и результаты моделирования подтвердили правильность функционирования модели.

Заключение

В результате научного исследования получены уравнения состояний временной стохастической сети Петри с многоканальными и конфликтными переходами, которые отличаются от известных уравнений состояний временной сети Петри с детерминированными задержками, во-первых, понятием состояния перехода, во-вторых, наличием уравнения, задающим продвижение времени.

Введено понятие Петри-объекта и разработана технология конструирования Петри-объектных моделей, при которой объекты моделирования создаются с использованием механизма наследования на основе класса объектов Петри-имитатор.

Моделирование системы с помощью Петри-объектов позволяет исследователю сосредоточиться на составлении сетей Петри элементов системы. При этом сеть Петри-объекта отображает поведенческие свойства элемента системы. Отладка Петри-объектов может быть выполнена до объединения в систему и не сложна, если Петри-объект достаточно простой. Только после отладки Петри-объектов, представляющих элементы системы, выполняется конструирование модели системы.

Подход к построению модели, который предлагается, позволяет быстро конструировать модели сложных систем и обеспечивает сокращение затрат времени на отладку и реализацию имитационных моделей больших систем. Технология апробирована на примере Петри-объектной модели системы управления учебным процессом вуза.

Список литературы

1. Ямпольський Л.С., Лавров О.А. Штучний інтелект у плануванні та управлінні виробництвом. – К.:Вища шк., 1995. - 255с.
2. Стеценко І.В., Бойко О.В. Система імітаційного моделювання засобами сіток Петрі // Математичні машини і системи. – Київ, 2009. – №1. – С.117-124.
3. Dmitriy A. Zaitsev Functional Petri net // Universite Paris Paris-Dauphine. – Cahier N 224. – mars 2005. – P.1-62.
4. Charles Lakos Object Oriented Modeling with Object Petri Nets // Concurrent Object-Oriented Programming and Petri Nets. - 2001. - P. 1-37.
5. Lakos, C., Keen, C. LOOPN++: a new language for object-oriented Perti nets, Technical Report R94-4, Networking Research Group, Univesity of Tasmania,Australia, April 1994.
6. Hue Xu Timed Hierarchical object-oriented Petri net // Petri Net, Theory and Applications, Book edited by: Vedran Kordic. – I-Tech Education and Publishing, Vienna, Austria. – 2008. – P.253-280.
7. Hong,J.E., Bae D.H. High-level Petri net for incremental specification of object-oriented system requirements // Institution of Engineering and Technology, IEEE Proceedings – Software. – 2001. – Vol. 148, No.1 – P.11-18.
8. Murata T. Petri Nets: Properties, Analysis and Applications. // Proceedings of IEEE. – 1989. - Vol.77, No.4. – P.541-580.
9. Зайцев Д.А. Инварианты временных сетей Петри // Кибернетика и системный анализ. - 2004. – № 2. – С. 92-106.
10. Стеценко І.В. Імітаційне моделювання системи управління навчальним процесом ВНЗ з використанням об’єктно-орієнтованого підходу // П’ята науково-практична конференція з міжнародною участю «Математичне та імітаційне моделювання систем МОДС’2010’». Тези доповідей. – Київ. – 2010. – 21-25 червня 2010р. – С.134-135.
11. Стеценко І.В. Моделювання систем: навч. посіб. [Текст] / Стеценко І.В.; М-во освіти і науки України, Черк. держ. технол. ун-т. - Черкаси: ЧДТУ,2010. – 399с.
12. Jinzhong Niu, Jing Zou, Aihua Ren OOPN: an object-oriented Petri nets and its integrated development environment [Електронний ресурс]. – Режим доступу: http://www.sci.brooklyn.cuny.edu/~jniu/research/publications/files/sea03-oopn.pdf.

УДК 004.021

ПОЛТОРАК В.П.,
ВІТІЩЕНКО Н.С.

КАЛЬКУЛЯТОР GF(q) ДЛЯ ЦИКЛІЧНИХ КОДІВ

Розроблені та реалізовані алгоритми функціонування інструменту для проведення обчислень в скінченних полях Галуа. Наведені приклади його роботи. Створений інструмент виконує арифметичні операції в полях Галуа, піднесення в степінь та ділення поліномів. Він знайшов застосування в наукових дослідженнях та в учбовому процесі НТУУ «КПІ». Калькулятор може бути рекомендований в учбовий процесс інших технічних ВНЗ.

Algorithms for the functioning of the application to perform calculations in finite Galois fields are developed and implemented. Examples of its work are demonstrated. An application calculates the arithmetic GF(q) operations, does exponentiation and division of polynomials in GF(q) too. This tool is used in scientific researches and in the educational process of NTUU "KPI". The tool can be recommended for the educational process of other technical universities.

На сьогоднішній день в інформаційно-комунікаційних технологіях існує ряд задач, пов'язаних з дослідженням надлишкових завадостійких кодів, що коректують помилки. Кінцевою метою таких досліджень є вирішення задач забезпечення надійного відеоконференцзв'язку; створення системи зв'язку, як великої цифрової системи; надійної та достовірної передачі даних між обчислювальними терміналами літаючих апаратів та супутників; передача даних вузько-смуговими каналами (телефон); зберігання великого об'єму даних, чуттєвих до помилок, тощо.

Існує клас циклічних кодів, здатних вирішувати проблему з виправленням помилок «на льоту» [1]. Коди Боуза-Чоудхурі-Хоквінгема (БЧХ) здатні виправляти до t помилок в блоці даних довжини n (t<n) [2]. В якості алфавіту коду обирається множина елементів поля Галуа GF(q), де q – потужність алфавіту, завдяки чому, місце знаходження помилок можна визначити, вирішивши ряд алгебраїчних рівнянь [3].

При дослідженні шляхів вирішення таких задач, необхідно виконати багатократні процедури кодування та декодування цими кодами, вивчити вплив ряду факторів на їхню ефективність.

Відомо, що полем Галуа називають множину зі скінченним числом елементів та з заданими на ній арифметичними операціями.

Найменше поле – двійкове, воно містить два елементи, з відповідними операціями [1].

В роботі поставлена ціль – створити окремий інструмент, котрий дозволив би виконувати обчислення в скінченних полях Галуа та виконувати ділення полінома на поліном, для того, щоб підвищити швидкість моделювання та обробки недвійкових надлишкових кодів в процесі моделювання та дослідження. Крім того, велике значення має розповсюдження навичок та вмінь виконання таких робіт в кадровій підготовці кваліфікованого персоналу в IT сфері.

В теперішній час існують декілька багатофункціональних, дорогих програмних пакетів, з широким функціональним спектром, за допомогою яких можна виконати обчислення в полях Галуа. Ці програми універсальні, тому вони складні в експлуатації, оскільки вирішують велику кількість різнотипових задач. Яскравими прикладами таких пакетів є MatLab та Mathematica.

MatLab 7.0 – програмний пакет, за допомогою якого можна виконувати арифметичні операції складання «+», віднімання «-», множення «*», ділення «/» над елементами кінцевого поля (назви операцій умовні та мають свої визначення, наведені нижче) [4].

Так, наприклад, використовуючи MatLab, для рішення елементарного прикладу складання двох чисел в GF(16) необхідно виконати достатньо велику кількість натискань клавіш. При цьому, структура вводу даних досить складна.

Приклад виконання операції $(10 + 13)_{16}$ в GF(16) з використанням пакету MatLab виглядає так:

```
>> x = gf(10,4)
x = GF(2^4) array. Primitive polynomial = D^4+D+1 (19 decimal)
 Array elements = 10
```

```
>> y = gf(13,4)
 y = GF(2^4) array. Primitive polynomial = D^4+D+1 (19 decimal)
 Array elements = 13
>> z = x + y
z = GF(2^4) array. Primitive polynomial = D^4+D+1 (19 decimal)
 Array elements = 7
```

Команди, що задані після «>>» вводить користувач.

Як видно, спочатку задається перший операнд в спеціальному форматі, і його значення присвоюється змінній x. Другий операнд задається аналогічно і його значення присвоюється змінній y. Після того, як обидва операнди задані, ми створюємо третю змінну z, та їй присвоюємо результат виконання операції складання x та y.

З діленням поліномів ситуація більш складна. Спочатку необхідно задати саме скінченне поле. Воно буде відображатися в незручному для сприйняття вигляді – в не лексикографічному порядку. Елементи полінома задаються в спеціальному вигляді в командному рядочку.

Другий пакет, Wolfram Mathematica 7 - це система комп'ютерної алгебри компанії Wolfram Research. Вона також містить функції, за допомогою яких можна виконувати арифметичні операції в полях Галуа [5].

Для того, щоб виконувати ці операції спочатку необхідно підключити відповідну бібліотеку - "FiniteFields`". Далі, зіткнемось з незручністю – результати виконання операцій будуть відображатися в двійковому вигляді.

Приклад виконання операції $(10 + 13)_{16}$ в GF(16) з використанням пакету Mathematica виглядає так:

```
In[1]:=GF[2,4][{1,0,1,0}]+GF[2,4][{1,0,1,1}]
Out[1]={1,1,0,1}2.
```

Операнди задаються в спеціальному форматі. Спочатку задається поле елементів, в даному випадку $GF(2^4)$. Далі, в фігурних дужках записуємо сам операнд в двійковому вигляді. Програма повертає результат складання операндів в двійковому вигляді, що не завжди зручно для користувача.

Очевидно, що в цих програмних пакетах важко виконувати відповідну обчислювальну роботу.

Тому важливо створити зручний для оператора інструмент, в якого буде вузько направлений функціонал. Щоб за допомогою цього інструменту можна було б обробляти символи алфавіту в полях Галуа, ділити поліном на поліном в GF(q) з отриманням результатів у вигляді частки та залишку від ділення.

Для вирішення поставленої задачі необхідно розробити реалізацію арифметичних операцій складання, множення, та обернених до них операцій – віднімання, ділення, а також піднесення в степінь та ділення полінома на поліном.

В роботі розглядається підготовка поля елементів для окремого випадку БЧХ коду – коду Ріда – Соломона.

В обраній області практичне застосування мають скінченні поля, з потужністю множини, що дорівнює цілій степені двійки. Так, наприклад, елементи $GF(16) = GF(2^4)$ можуть бути подані в декількох формах (табл.1): в векторному поданні елементів поля, адитивною групою та мультиплікативною групою.

Табл. 1. Представлення поля GF(16)

N	A_N (векторне подання)	$R_m(x)$ (адитивна група)	x^i (мультиплікативна група)
0	0000	0	-
1	0001	1	x^0
2	0010	x	x^1
3	0011	x+1	x^4
4	0100	x^2	x^2
5	0101	x^2+1	x^8
6	0110	x^2+x	x^5
7	0111	x^2+x+1	x^{10}
8	1000	x^3	x^3
9	1001	x^3+1	x^{14}
10	1010	x^3+x	x^9
11	1011	x^3+x+1	x^7
12	1100	x^3+x^2	x^6
13	1101	x^3+x^2+1	x^{13}
14	1110	x^3+x^2+x	x^{11}
15	1111	x^3+x^2+x+1	x^{12}
1	0001	1	x^{15}

Розглянемо визначення та виконання вказаних вище арифметичних операцій скінченного поля, з метою підготовки їх алгоритмічної реалізації.

Складання елементів поля виконується за допомогою їх подання адитивною групою або за допомогою векторного подання (табл.1).

Наприклад,

$10 + 7 = x^3 + x + x^2 + x + 1 = x^3 + x^2 + (1 + 1)x+1= = x^3+ x^2 + 1 = 13.$

$(10 + 7)_{16} = (1010)_2 + (0111)_2 = (1101)_2 = (13)_{16}.$

Множення елементів поля виконується за допомогою їх подання мультиплікативною групою (табл.1).

Наприклад,

$10 * 7 = x^9 * x^{10} = x^{19} = x^{15} * x^4 = x^4 = 3$.

Віднімання елементів поля виконується аналогічно складанню.

Наприклад,

$10 - 7 = 10 + (-7) = 10 + 7 = x^3+x+x^2+x+1 =$

$= x^3+x^2+(1+1)x+1 = x^3+ x^2+1 = 13$.

$(10 - 7)_{16} = (10 + (-7))_{16} = (10 + 7)_{16} =$

$= (1010)_2 + (0111)_2 = (1101)_2 = (13)_{16}$.

Ділення елементів поля виконується за допомогою їх подання мультиплікативною групою, і ця операція схожа на операцію множення, тільки при діленні показники степені віднімаються (табл.1). Ділення на нуль не визначено.

Наприклад,

$10 : 7 = x^9 : x^{10} = x^{-1} = x^{15} * x^{-1} = x^{14} = 9$.

При піднесенні в степінь заданого операнду, використовується його мультиплікативне подання. При цьому степені перемножуються.

Наприклад,

$10^7 = (x^9)^7 = x^{63} = x^{15} * x^{15} * x^{15} * x^{15} * x^3 = x^3 =$

$= 8$.

Ділення полінома на поліном – більш складна процедура, яка в явному вигляді виконується за допомогою операндів, поданих поліномами. Розглянемо приклад ділення поліномів поданих у векторній формі на рис.1.

Нехай, наприклад, поліном - ділене – 1 5 8 0 0.

А поліном - дільник – 1 12 5.

В результаті виконання стандартної процедури ділення «в стовпчик» отримаємо такі результати:

Частку від ділення – 1 9 11.

Залишок – 6 1.

Проміжкові від'ємники – 1 12 5, 9 6 11, 11 13 1.

На першому кроці ділимо старший елемент діленого на старший елемент дільника та записуємо його на старшу позицію частки. Далі перемножуємо отриманий елемент частки на кожен елемент дільника, та записуємо проміжковий від'ємник під діленим. Далі, за правилами віднімання (результат якого співпадає з результатом складання для полів характеристики p = 2), додаємо перші f (f – число елементів дільника) елементів діленого з елементами проміжкового від'ємника, нижче записуємо результат.

Ці дії повторюємо до тих пір, поки в діленому ще є елементи (в даному прикладі це «0») для зносу їх до залишку. Коли порядок залишку буде меншим порядку дільника, процес ділення припиняється (тобто степінь полінома – залишку стає меншою степені полінома - дільника).

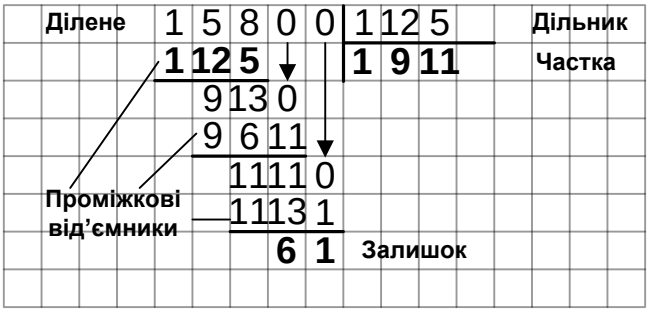


Рис. 1. Ділення поліномів в векторній формі

Інструмент повинен містити декілька обчислювальних модулів: модуль *Калькулятора*, що виконує операції складання, множення, віднімання, ділення, піднесення в степінь, а також, модуль *Дільника поліномів*, що ділить поліном на поліном.

В модулі *Калькулятора* повинен бути передбачений вибір GF(q), для $q = 2^i$. Повинні бути присутні декілька способів вводу даних - з клавіатури або за допомогою миші.

Оператор повинен мати можливість виконувати арифметичні операції з елементами скінченного поля та піднесення в степінь. Користувач повинен вводити перший операнд з клавіатури, або за допомогою миші, далі, обрати бажану операцію та вводити другий операнд. Після цього, натиснувши клавішу «=» або «Enter» на екрані повинен з'явитися покроковий результат обчислювань. При складанні та відніманні проміжковий результат повинен подаватися в двійковому вигляді. При множенні, діленні та піднесенні в степінь – в степеневому поданні елементів.

В модулі *Дільника поліномів* повинен бути передбачений ввід даних з клавіатури, а також із зовнішнього файлу (в даному випадку дані завантажуються з таблиці).

Користувач повинен задавати поліном – ділене та поліном - дільник в векторному вигляді. Після натискання клавіші «Розділити» на екрані повинен з'явитися результат ділення, у вигляді залишку та частки від ділення поліномів. В теорії завадостійкого кодування залишок від ділення поліномів цікавить нас в більшій мірі.

Оператор повинен мати можливість багатократно виконувати ділення поліномів. Для цього необхідно передбачити кнопку

«Скидання». Функціонально – логічна схема *Калькулятора* та *Дільника поліномів* зображена на рис.2.

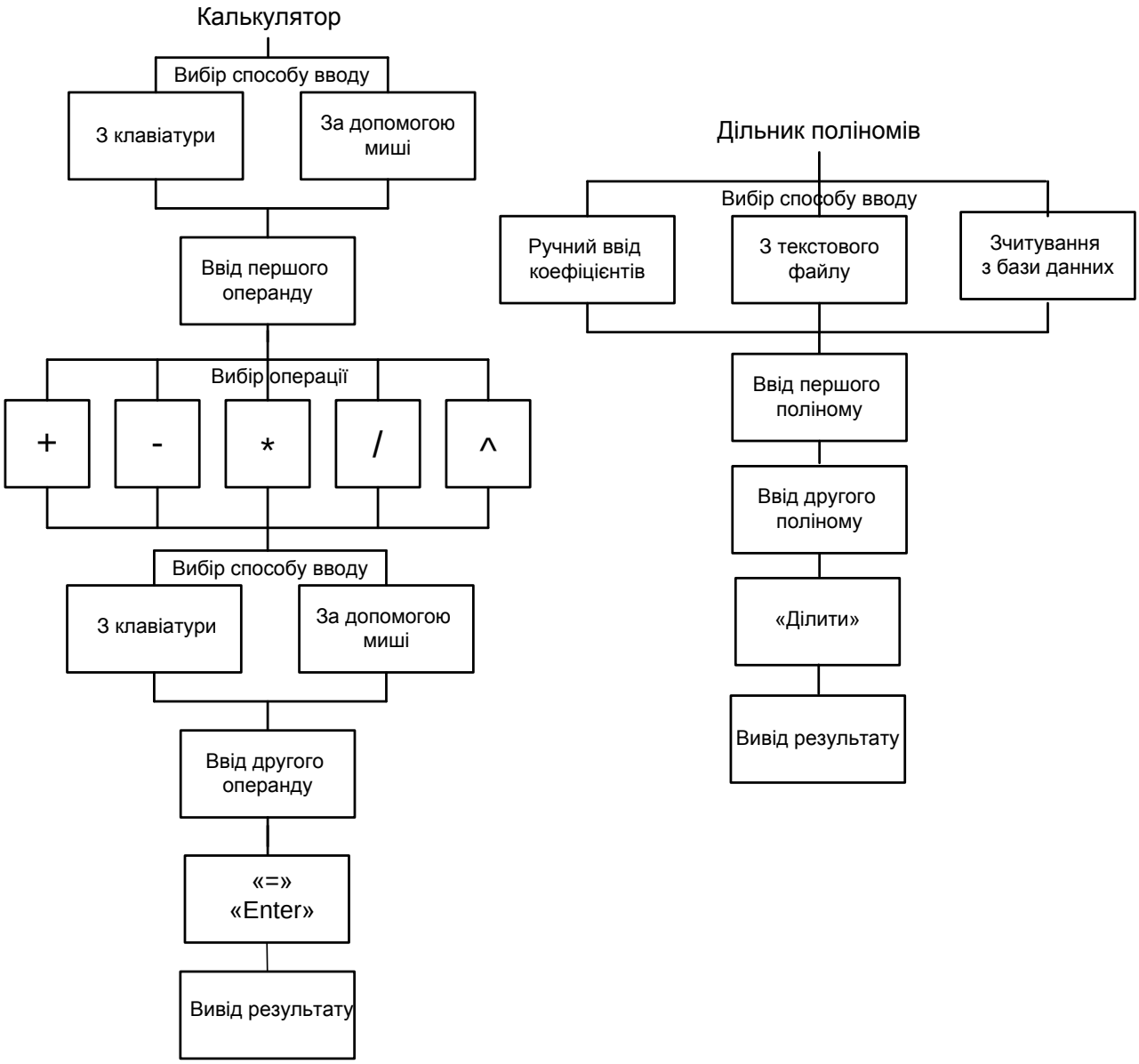


Рис. 2. Функціонально-логічна схема Калькулятора GF(q)

Для виконання операції складання двох елементів скінченного поля, необхідно використати три процедури, які описані в табл.2.

Процедура ToBin – переводить числа з q – ічного вигляду в двійковий еквівалент. Процедура DoPlus – додає два двійкових числа, за mod 2, порозрядно (без переносів). Процедура ToDec – переводить результат додавання з двійкового вигляду в q – ічний.

Табл. 2. Перелік необхідних процедур для виконання складання

ToBin	Передаємо chislo_dec – операнд в q - ічному вигляді та mas_bin – масив для зберігання двійкового числа.
ToDec	Передаємо chislo_bin – операнд в двійковому вигляді.
DoPlus	Передаємо операнд 1 та операнд 2 в двійковому вигляді; mas_result – масив для зберігання результату додавання. алг DoPlus (арг цел таб op1[1:4], цел таб op2[1:4], цел таб mas_result[1:4]) нач нц для i от 0 до 3 mas_result[i] := op1[i] ^ op2[i] кц кон

Для виконання операції множення двох елементів поля Галуа, необхідна процедура, що описана в табл.3.

Процедура DoMult – множить два операнди та вертає результат множення в q – ічному вигляді.

Табл. 3. Процедура для виконання ноження

DoMult	Передаємо операнд 1 та операнд 2; змінна result_mult – повертає результат множення.

```
алг DoMult (арг цел operand1, арг цел operand2, рез цел result_mult)
нач цел result_mult:= 0, temp := 0 | temp – змінна, в якій зберігається результат складання показників степеней двох чисел.
  если operand1 = 0 или operand2 = 0 то result := 0
  если operand1 > 0 и operand2 > 0 то
    нц temp := mas_pow[operand1] + mas_pow[operand2] | mas_pow – масив степеней для мультиплікативної групи
      если temp >= 15 то temp := temp mod 15
        нц для i от 0 до 15
          если mas_pow[i] = temp то result_mult:= i
        кц
      иначе
        нц для i от 0 до 15
          если mas_pow[i] = temp то result_mult:= i
        кц
    кц
кон
```

Операція віднімання по своїй суті аналогічна операції складання, тому в ній вико ристовуються ті ж самі процедури (табл.2).

Операція ділення елементів скінченного поля схожа на операцію множення. Різниця в тому, що показники степені ми віднімаємо (табл.4). При цьому, не існує проблеми ділення на нуль, оскільки відсутнє мультиплікативне подання для нуля.

Процедура DoDiv – ділить перший операнд на другий та повертає результат ділення.

Табл. 4. Процедура для виконання ділення

DoDiv	Передаємо операнд 1 та операнд 2; змінна result_div – повертає результат ділення.

```
алг DoDiv (арг цел operand1, арг цел operand2, рез цел result_div)
нач цел result_div:= 0 | result_div – зміна, що зберігає результат ділення двох чисел
если (operand1 != 0 и operand2 = 0) или (operand1 = 0 и operand2 = 0) то
result_div:= -1| вивід повідомлення про помилку
  если operand1 = 0 и operand2 > 0 то result_div:= 0
  если operand1 != 0 и operand2 != 0 то
    нц temp := mas_pow[operand1] - mas_pow[operand2] | temp – результат віднімання степеней операндів; mas_pow – масив степеней для мультиплікативної групи
      если temp < 0 то temp := temp + 15
        нц для i от 0 до 15 | пошук елемента, що відповідає отриманій встепені
          если mas_pow[i] = temp то result_div:= i
        кц
      иначе
        нц для i от 0 до 15
          если mas_pow[i] = temp то result_div:= i
        кц
    кц
кон
```

Піднесення в степінь здійснюється за алгоритмом, що описаний у табл.5.

Процедура DoPow – підносить перший операнд в степінь, задану другим операндом та повертає результат піднесення в степінь.

Табл. 5. Процедура піднесення в задану степінь операнда

DoPow	Передаємо операнд 1 та pow - степінь, в яку підносимо операнд 1; змінна result_pow – повертає результат піднесення в степінь.

```
алг DoPow (арг цел operand1, арг цел pow, рез цел result_pow )
нач цел result_pow:= 0 | result_pow – змінна, що зберігає результат піднесення в степінь.
  если operand1 = 0 или pow = 0 то result_pow:= 0
    иначе
      нц temp := mas_pow[operand1] * operand2; | temp – результат множення степені операнда1 на задану степінь pow.
        temp = temp mod 15
        нц для i от 0 до 15
          если mas_pow[i] = temp то result_pow:= i
        кц
      кц
кон
```

Алгоритм функціонування модуля ділення поліномів виглядає наступним чином (псевдокод):

ВХІДНІ ДАНІ:

polinom1 – строка з коефіцієнтами полінома - діленого (перший поліном).

polinom2 – строка з коефіцієнтами полінома - дільника (другий поліном).

ВИХІДНІ ДАНІ:

ostatok – масив коефіцієнтів залишку від ділення.

chast – масив коефіцієнтів частки від ділення.

Крок 1.

Перетворення коефіцієнтів першого полінома у масив елементів поля. Підрахунок кількості елементів.

Крок 2.

Перетворення коефіцієнтів другого полінома у масив елементів поля. Підрахунок кількості елементів.

Крок 3.

Записуємо частку від ділення в mas_chast[i].

Крок 4.

Множимо коефіцієнти другого полінома mas_pol_2[i] на перший елемент частки mas_chast[0], за допомогою процедури DoMult (табл.3).

Крок 5.

Додаємо елементи залишку до елементів проміжкового від'ємника mas_temp[i];

нц для і от 0 до pol_2_length | pol_2_length – кількість елементів другого полінома (полінома - дільника).

mas_ ostatok [i]:=mas_ ostatok[i]^mas_ temp [i]

кц

Крок 6.

Зсуваємо всі елементи залишку на одну позицію вліво, тим самим прибираємо нуль з першої позиції.

нц для і от 0 до mas_ostatok.length-1

mas_ ostatok [i] := mas_ ostatok [i+1]

кц

Крок 7.

Перевіряємо чи залишились ще в першому поліномі елементи, які можна знести до залишку.

Якщо залишились – зносимо елемент. Якщо ні – переходимо до кроку 8.

Крок 8.

Вивід залишку та частки від ділення у відповідні Тулбокси діалогового вікна.

При написанні *Калькулятора GF(q)* була використана мова програмування C# платформи .NET.

На рис.3 наведений вигляд головного вікна Калькулятора, наприклад, для GF (16).

За допомогою меню Калькулятора можна обрати роботу безпосередньо з модулем *Калькулятора*, або з модулем *Дільника поліномів.*

Інтерфейс *Калькулятора* містить: кнопки з цифровими значеннями, кнопки зі знаками операцій, а також область, в якій від ображається покроковий результат виконання операцій над двома операндами.

Нижче наведені приклади обчислювань Калькулятором, представлених вище в GF (16) операцій.

Тут враховано, що $x^{15} = 1$.

Так, операція (10 + 7), Калькулятором виконується, як:

$10 + 7 = 1010 + 0111 = 1101 = 13$

Операція (10 * 7) виконується, як:

$10 * 7 = x^9 * x^{10} = x^{19} = x^{15} * x^4 = 3$

Операція (10 - 7) виконується, як:

$10 - 7 = 10 + (-7) = 10 + 7 = 1010 + 0111 = 1101 = 13$

Операція (10 / 7) виконується, як:

$10 / 7 = x^9 / x^{10} = x^9 * x^{-10} = x^{-1} * x^{15} = 9$

Піднесення в степінь (10^7) виконується, як:

$10^7 = (x^9)^7 = x^{63} = x^3 = 8$

На рис.4 зображено вікно *Дільника поліномів*, наприклад, для GF (16).

Його можна запустити, обравши відповідну закладку в опціях Калькулятора GF(16). Інтерфейс вікна *Дільника поліномів* має вигляд звичний для оператора.

Поліноми у вікні дільника задаються у векторному поданні через пробіл.

В результаті ділення полінома на поліном, ми отримаємо залишок та частку від ділення.

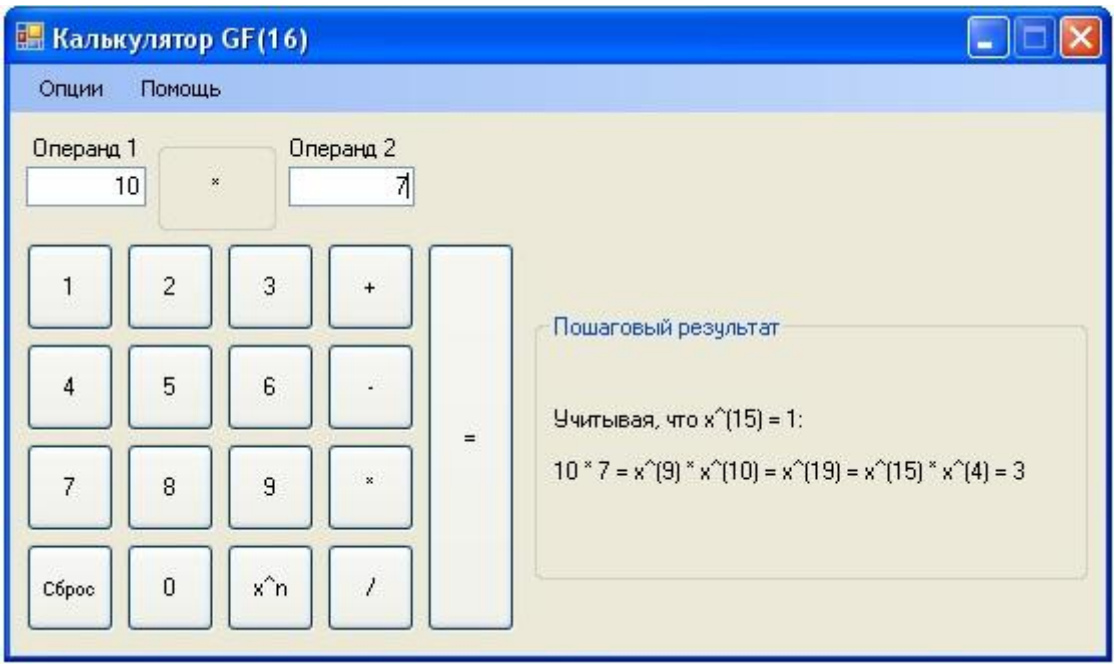


Рис. 3. Головне вікно Калькулятора GF(16)

Можна виконати необмежену кількість ітерацій ділення.

На рис.4 зображено приклад виконання ділення полінома 4 степені, поданого вектором (з набором своїх коефіцієнтів 1 5 8 0 0) на поліном 2 степені (набором своїх коефіцієнтів 1 12 5). Результат ділення відображається у вигляді залишку та частки від ділення.

		хв	**лок**
1.	Вручну	30	12
2.	З використанням *Калькулятора*	10	4
3.	З використанням *Дільника поліномів*	1	1

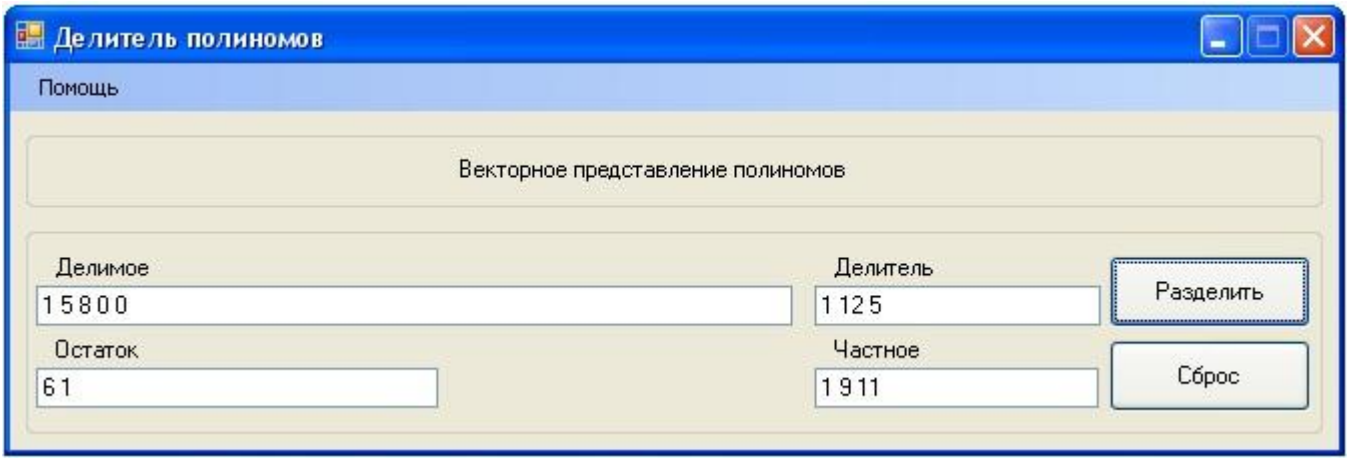


Рис. 4. Вікно Дільника поліномів

З метою перевірки ефективності створеного інструменту, були проведені натурні експерименти. Розглядалися три ситуації, в яких прийняли участь 25 студентів - операторів.

Дослід 1. Їм було доручено виконати операцію ділення полінома на поліном вручну. При цьому порядок діленого був 14, а дільника – 6. Практично всім студентам на вирішення даної задачі знадобилося близько тридцяти хвилин. Частина з них допускала помилки, в результаті чого, були отримані невірні значення розрахунків, що відображено в табл.6.

Дослід 2. На наступному етапі експерименту студенти використовували *Калькулятор* для виконання арифметичних операцій над коефіцієнтами поліномів. В цьому випадку час, який вони затратили на ділення поліномів значно зменшився. Але через людський фактор, все ще, виникали помилки в розрахунках (табл.6).

Дослід 3. На третьому етапі дослідження, обчислення проводились у *Дільнику поліномів*, на що знадобилось 1..2 хвилини щоб ввести дані, та декілька секунд, щоб отримати достовірний та правильний результат. Помилки практично були відсутні. Одна помилка виникла через неправильний ввід даних.

Усереднені результати експерименту від ображені в табл.6.

Табл. 6. Результати експерименту

Номер експерименту	**Обчислювальний тип експерименту**	**Середній час виконання,**	**Середня кількість поми-**

В рамках даної роботи під ефективністю калькулятора gf(q) будемо розуміти затрати часу т на виконання обчислювальної роботи операторами та кількість випадків помилково виконаних розрахунків.

Обчислювати ефективність е будемо, як величину обернену добутку часу т на середню кількість помилок (1) (для уникнення нескінченного значення показника ефективності, при $\varepsilon = 0$ введемо корекцію, у вигляді $\varepsilon + 1$).

$$E = \frac{1}{T \cdot (1 + \varepsilon)} \qquad (2)$$

де т – час затрачений на обчислення,

ε – Кількість виниклих помилок.

За результатами експерименту був побудований графік, що віддзеркалює ефективність застосування отриманого інструменту (рис.5).

В майбутньому планується вдосконалення калькулятора, шляхом додавання додаткового функціоналу: *перемножувач поліномів*, що перемножує поліноми, для обчислення твірного поліному g(x), циклічного коду, та застосовується у алгоритмі декодування.

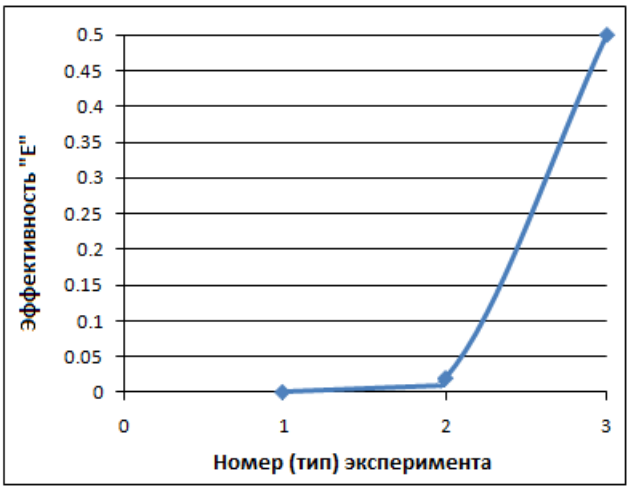


Рис. 5. Графік ефективності Калькулятора GF(q)

Обчислювач значень полінома у заданих точках.

Генератор поля, в якому задавши число елементів поля як цілу степінь двійки, можна буде генерувати подання поля у вигляді таблиці (подібній табл.1), де елементи поля будуть подані в декількох формах: десятковій, у формі залишків від ділення на твірний поліном (адитивна група), в степеневому вигляді (мультиплікативна група) і у векторній формі, а також містити відповідні елементам поля мінімальні поліноми.

Репрезентований обчислювальний інструмент використовується в наукових цілях при дослідженні властивостей надлишкових циклічних кодів.

Розроблений інструмент знайшов також застосування в учбовому процесі в НТУУ «КПІ» та може бути рекомендований для студентів інших технічних ВНЗ.

Список літератури

1. Касами Т., Токура Н., Ивадари Ё., Инагаки Я. Теория кодирования: Пер. с япон. – М.:Мир, 1978. – 568с.
2. Бэрлекэмп Э. Алгебраическая теория кодирования: Пер. с англ. – М.:Мир, 1971. – 463с.
3. Блейхут Р. Теория и практика кодов, контролирующих ошибки: Пер. с англ. – М.:Мир, 1986. – 575с.
4. Ануфриев И.Е., Смирнов А.Б., Смирнова Е.Н. MATLAB 7 (Наиболее полное руководство): БХВ-Петербург, 2005. -1104с.
5. http://www.wolfram.com/

УДК 658:005.5

ТОМАШЕВСЬКИЙ В.М.,
ГРЕГУЛЬ В.В.,
НАЗАРУК О.В.,
ТИМОШЕНКО Я.Я.

ОЦІНКА ЕФЕКТИВНОСТІ ВПЛИВУ НА ФАКТОРИ РИЗИКУ СИСТЕМИ ОХОРОНИ ЗДОРОВ'Я ЗАСОБАМИ ІМІТАЦІЙНОГО МОДЕЛЮВАННЯ

У статті запропоновано новий підхід до дослідження галузі охорони здоров'я з врахуванням динамічній взаємодії всіх підсистеми. Побудована імітаційна модель оцінює вплив різних факторі та ризиків на стан здоров'я та запропонувати найбільш ефективні шляхи нейтралізації негативних впливів. Отримані результати вказують на можливі шляхи покращення ефективності системи охорони здоров'я, а також мінімізації її видатків.

This article proposes a new approach to the study of health, taking into account the dynamic interaction of all subsystems. Simulation model, which was created is able to evaluate the influence of various factors and risks to health and to offer the most effective ways to neutralize the negative impacts. The results indicate possible ways of improving health system efficiency and minimize its costs.

Вступ

Стрімкий розвиток медичних наук, що спостерігається протягом останнього часу, швидке збільшення інформації про причини, патогенез, профілактику та лікування різних захворювань, дозволяє вводити нові методи дослідження впливів на здоров'я людей та модифікувати всю систему охорони здоров'я (СОЗ). Сучасний рівень медичних знань дозволяє оцінювати опосередковані впливи на стан здоров'я людини та попереджувати виникнення хвороб через усунення або нейтралізацію медичних, соціальних, психологічних та інших факторів ризику, зокрема впливу навколишнього середовища, таким чином зменшуючи захворюваність і витрати на лікування. Такий аналіз можливий лише за допомогою використання інформаційних технологій, що спираються на методи моделювання, системного аналізу та теорії прийняття рішень.

Серед існуючих робіт по застосування інформаційних систем у медицині слід відзначити праці [1,…, 5]. Їх метою є ефективний аналіз великого обсягу медичних даних і прийняття відповідних рішень. До того ж, у працях [1, 2] розроблено програмні додатки для автоматизації згаданих задач. Однак жодна з праць не пропонує принципово нову концепцію системи охорони здоров'я в цілому, а орієнтується на порівняно вузькі, спеціалізовані аспекти галузі і не передбачає внесення змін у структуру функціонування системи. Проблема оптимізації та реструктуризації медичної служби в збройних сил України (ЗСУ) залишається актуальною і досі відкритою.

Постановка задачі

Сучасний стан СОЗ не враховує повній мірі можливі причини виникнення будь-яких хвороб. На сьогодні необхідно розглядати впливи фак.-торів, що стосуються не лікування, а недопущення хвороби (далі не медичні фактори). Вони включають здоровий спосіб життя, спортивну активність, збалансовану дієту, гігієну, відмову від шкідливих звичок, а також соціально-економічне та психологічне благополуччя. Як показано у науковій праці [7] такі фактори мають дуже високий вплив на захворюваність і рівень смертності від хвороб. Метою роботи є створення моделі модифікованої СОЗ, що дозволить оцінити ефективність корегування впливів тих чи інших факторів ризику на функціонування СОЗ та запропонувати найбільш вигідні шляхи модифікації системи як з медичної точки зору (зменшення захворюваності) так і з фінансової (ефективне вико ристання коштів).

Досліджується концепцію системи охорони здоров'я, що підвищує рівень здоров'я людей і зменшує витрати на лікування шляхом недопущення виникнення хвороби, впливаючи на медичні та не медичні фактори.

Для порівняння запропонованої концепції з існуючою системою створено і досліджено імі-

таційні моделі СОЗ з метою оцінювання характеристик ефективності.

Загальна концепція СОЗ ЗСУ

Узагальнений причинно-наслідковий цикл погіршення стану здоров'я можна описати наступним чином: здорові військовослужбовці під дією несприятливих факторів набувають шкідливих звичок. Шкідливі звички призводять до розвитку у особи факторів ризику, яки, у свою чергу, спричиняють гострі або хронічні захворювання, що призводить до втрати працездатності, незворотного погіршення стану здоров'я або смерті (рис. 1). На рисунку позначено о\с – особливий склад.

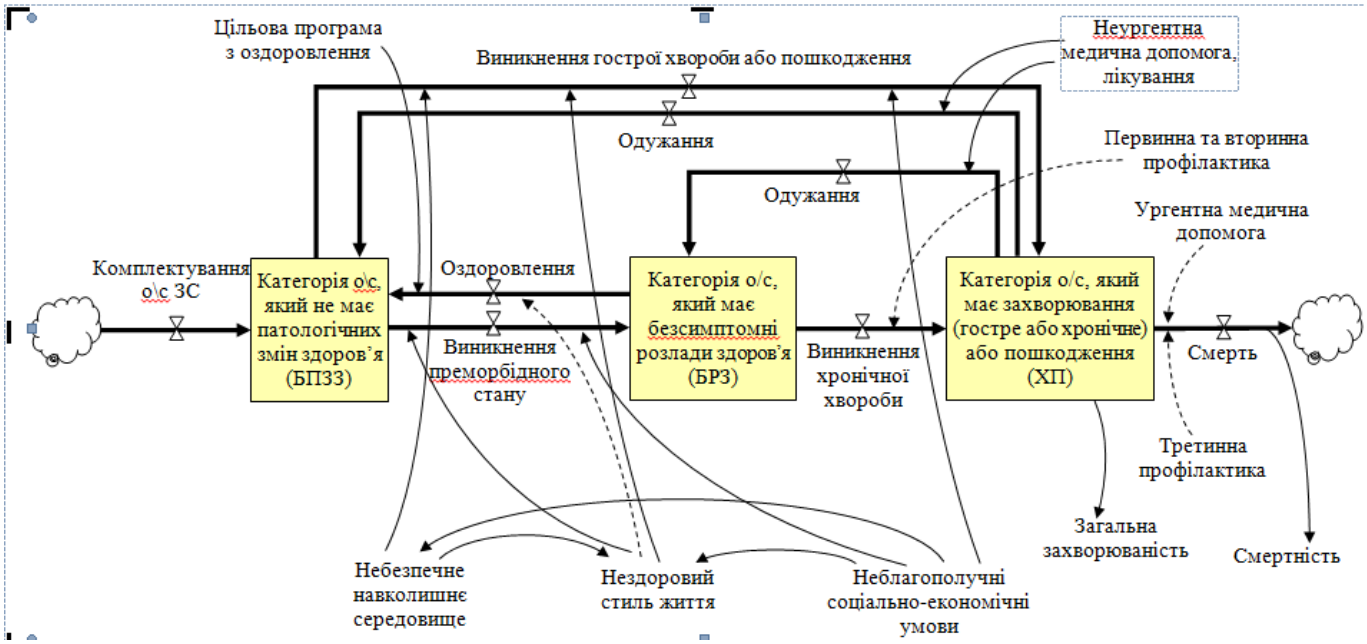


Рис. 1. Вплив різних факторів на стань здоров'я людей

Існує також і обернений цикл – процес відновлення попереднього стану здоров'я або не погіршення поточного стану. Наприклад: боротьба зі шкідливими звичками допоможе частині людей відмовитись від них, таким чином повертаючись у категорію здорових, тобто істотно зменшуючи імовірність набуття цими людьми факторів ризику. Також можливе лікування, яке не допускає розвиток більш тяжких форм захворювання. Пацієнти, яких вилікували від шкідливих звичок переходять або у категорію повністю здорових, або у категорію людей із шкідливими звичками в залежності від ефективності боротьби.

Модель СОЗ була б неповною, якщо не розглядати природне старіння організму. Так, із збільшенням віку особи, збільшується захворюваність, підвищується складність нозології, витрачається більше коштів на лікування і утримання пацієнта. Окрім того, зменшується ефективність лікування і значно зменшується результативність боротьби із шкідливими звичками і пропаганди здорового способу життя [8-11].

Відомо, що СОЗ фінансується із бюджету України, тому в моделі необхідно врахувати вплив фінансових витрат на СОЗ.. Постійні витрати нараховуються незалежно від стану здоров'я військовослужбовця. До них належать: харчування, забезпечення житлом та іншими матеріальними цінностями, а також виплата заробітної платні офіцерам та військовослужбовцям-контрактникам. Змінні витрати – це витрати на профілактику, ранню діагностику, лікування тощо. Змінні витрати збільшуються із збільшенням числа хворих, а також залежать від наявності неперервного фінансування. Тобто можна припустити, що при нестачі коштів припиняється профілактика і лікування хворих. Створена модель дає змогу проаналізувати співвідношення кількості витрачених коштів і стану боєздатності військовослужбовців.

Експерименти з моделлю проводяться для того, щоб знайти відповіді на такі запитання:

- чи існують в СОЗ фактори, що можуть призвести до суттєвого покращення загального стану здоров'я за відносно невеликі кошти;
- чисельно визначити, як профілактика і рання діагностика у молодому віці впливає на зменшення захворюваності у майбутньому;

- знайти таке поєднання зусиль, що за безпечить максимальну працездатність при найбільшій можливій тривалості житті (професійне довголіття);
- найкраще співвідношення кількості боєздатних осіб до витрачених коштів;
- мінімально необхідне фінансування на забезпечення деякого заданого відсотку боєздатних осіб із загального числа військовослужбовців.

Імітаційна модель побудована за принципами системної динаміки і складається з фондів, потоків і конвертерів [12]. Фонди слугують резервуарами для накопичення числа військовослужбовців відповідних категорій з відповідним станом здоров'я. Вони наповнюються та вичерпуються потоками, які за характером використання розділяються на обмежені і необмежені, односпрямовані і двоспрямовані. Інтенсивність потоку задається певною функцією, коефіцієнти якої, як правило, зберігають у конвертерах. Цими коефіцієнтами можуть бути як наперед задані константи, так і регульовані змінні. Коефіцієнти-конвертери, значення яких можна змінювати в процесі моделювання, зображуються із регулятором у середині.

На рис. 2 представлена взаємодія основних фондів і потоків першого, найбільш загального, варіанта моделі зміни стану здоров'я військовослужбовців.

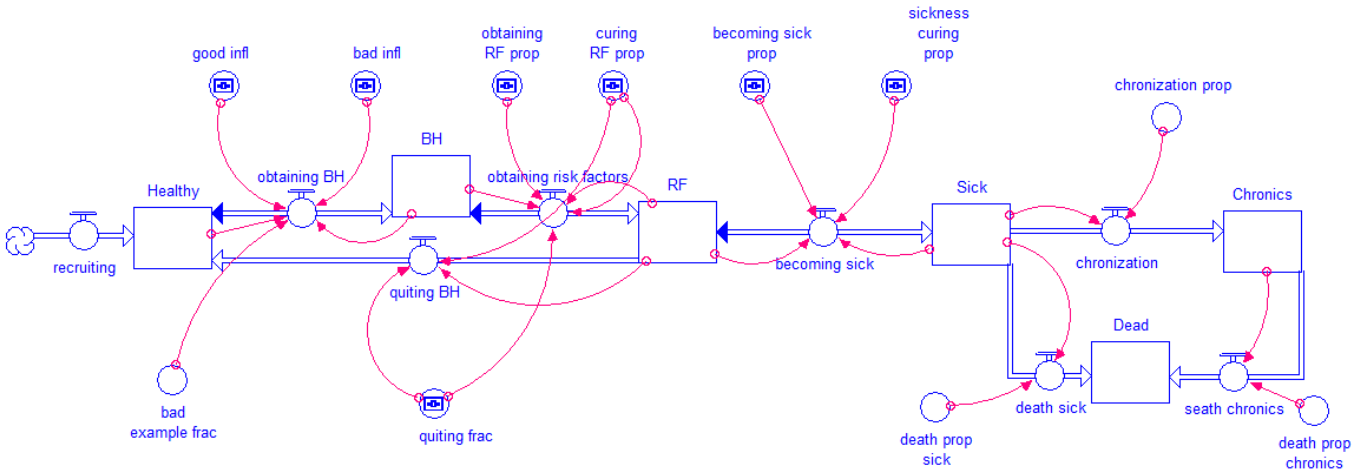


Рис. 2. Узагальнена модель зміни стану здоров'я військовослужбовців

У моделі використовуються наступні фонди:

– HEALTHY – повністю здорові;

– BH (Bad Habits) – із шкідливими звичками (паління, алкоголізм, а також малорухливий спосіб життя, незбалансоване харчування, недотримання гігієни тощо);

– RF (Risk Factors) – мають фактори ризику – розлади здоров'я, що безпосередньо не призводять до втрати працездатності, але без вчасного лікування через певний час викликають серйозніші захворювання: гіпертонію, ожиріння, цукровий діабет і т.ін.;

– SICK – гостро хворі, тимчасово непрацездатні;

– CHRONICS – хронічно хворі, постійні обмеження працездатності;

– DEAD – померли внаслідок хвороби.

Позначені на рис. 2 двоспрямовані потоки (biflow) задають:

– OBTAINING BH – набуття або позбавлення шкідливих звичок;

– OBTAINING RF – набуття або позбавлення факторів ризику;

– BECOMING SICK – розвиток хвороби або одужання.

Односпрямовані потоки задають:

– RECRUITING – поповнення армії (для спрощення розглядається тільки поповнення за рахунок весняного і осіннього призивів);

– QUITTING BH – позбавлення шкідливих звичок після проходження успішного курсу лікування;

– CHRONIZATION – набуття хроніних захворювань;

– DEATH SICK – смерть внаслідок гострої хвороби;

– DEATH CHRONIC – смерть внаслідок загострення хронічної хвороби.

У моделі використовуються наступні регульовані конвертери:

– GOOD INFL – агреговані позитивні впливи, направлені на відмову від шкідливих звичок (включають надзвичайно широкий спектр впли-

вів від пропаганди здорового способу життя до покращення умов життя, зменшення стресу, надання послуг психолога, створення умов для занять спортом і доступність здорового харчування тощо);

– BAD INFL – агреговані негативні впливи, що призводять до появи шкідливих звичок а також стресу, перевантаження на роботі, погані соціальні умови життя тощо;

– QUITING FRAQ – відсоток людей, що відмовляються від шкідливих звичок, який пропорційний витраченому часу спеціалісттами-психологами з боротьби із шкідливими звичками;

Конвертори, які задаються константами:

– BAD EXAMPLE FRAC – коефіцієнт «поганого прикладу» (чим більше людей мають шкідливі звички, тим імовірніше інші люди будуть їх набувати);

– OBTAINING RF PROP – імовірність розвитку факторів ризику в результаті впливу шкідливих звичок;

– CURING RF PROP – імовірність успішного лікування факторів ризику, яка в першу чергу, залежить від ранньої діагностики;

– BECOMING SICK PROP – імовірність розвитку тяжкого (гострого) захворювання із-за факторів ризику;

– SICKNESS CURING PROP – імовірність успішного лікування хвороби;

– CHRONIZATION PROP – імовірність хронічного розвитку хвороби;

– DEATH SICK PROP – імовірність смерті внаслідок хвороби;

– DEATH CRON PROP – імовірність смерті внаслідок гострої хвороби.

Розширена модель враховує процеси старіння, які моделюються за допомогою послідовного переходу людей по віковим групам: до 20, до 30, до 40 і до 50 років [13,14]. Згадані вікові групи представлені у моделі відповідними фондами. У моделі, для наочності, категорії «здорові», «зі шкідливими звичками» та інші виділяються в окремі сектори. Коефіцієнти, що змінюють інтенсивності потоків моделі у конвертерів, задані як масив коефіцієнтів, де індекси цих масивів визначають вікові групи.

Для того, щоб замкнути причинно-наслідковий цикл, введемо залежність від фінансування. Замкнений ланцюг міркувань виглядає так: чим більше здорових військовослужбовців, тим дешевше їх утримувати і тим вища боєздатність армії, але при цьому зростають витрати на підтримання здоров'я військовослужбовців. Основна мета – знайти таке співвідношення витрат на профілактику і на лікування, при якому буде забезпечений бажаний відсоток здорових військовослужбовців при мінімальних витратах коштів.

Повна модель з відповідними блоками представлена на рис. 3.

Модель відтворює процеси фінансування у Збройних сил (ЗС) за допомогою блоку бюджет (**BUDGET**), у якому на початок моделювання є певний залишок коштів. Раз у рік, бюджет збройних сил поповнюється надходженнями з Державного бюджету (потік BUDGETING). Щомісяця з бюджету ЗС фінансуються (потік UPKEEP) служба охорони здоров'я ЗС і утримання армії.

Для визначення управлінських дій в медичній службі на основі вище наведених моделей необхідно провести серію експериментів за заданими сценаріями для визначення найкращих управлінських рішень.

Аналіз результатів експериментів

Розглянемо три сценарії, яки визначають профілактику захворювань, покращення соціального та психологічного стану осіб, що обслуговуються СОЗ ЗСУ і проведення соціальних та економічних заходів, що сприяють відмові від шкідливих звичок.

Для дослідження впливу кожної групи факторів проведемо такі експерименти:

- рівний розподіл зусиль між фак.торами (стандартний бюджет);
- відмова від впливу на соціальні та психологічні фактори на користь профілактики захворювань при стандартному бюджеті;
- близький до найкращого розподіл зусиль при збільшенні бюджету на 40 %.

За результатами експериментів, наведеними у табл. 1 та табл. 2 при рівному розподілі зусиль СОЗ веде себе нестабільно через нестачу коштів (занадто велика частка бюджету іде на задоволення соціальних потреб, і, як наслідок, не вистачає коштів на будь-що інше). Чисельність ЗС при такій стратегії нестабільна і недостатня.

У випадку з виключенням впливу на соціальні та психологічні фактори чисельність і витрати стабілізуються, але рівень здоров'я військовослужбовців незадовільний, занадто висока захворюваність та хронічні захворювання.

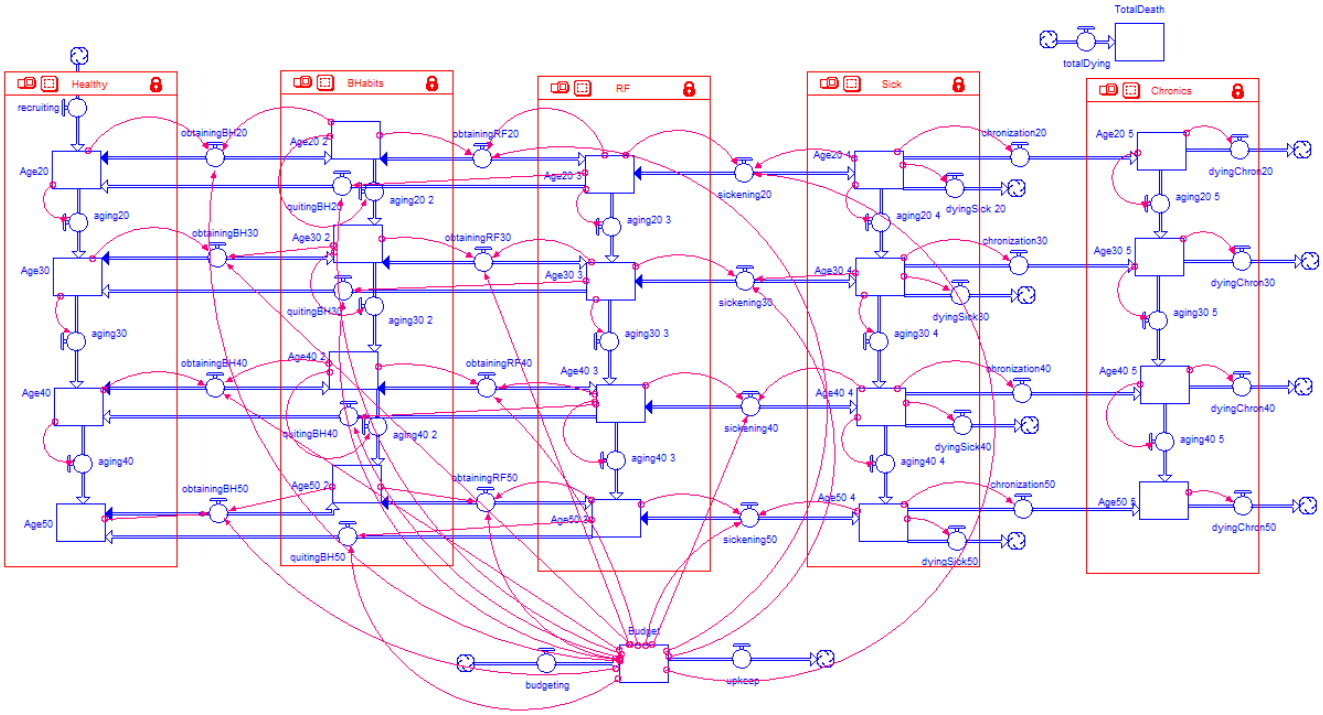


Рис. 3. Модель, яка визначає рівень здоров'я військовослужбовців з урахуванням залежності від фінансування СОЗ

Табл. 1. Бюджет на кінець періоду та періодичні витрати

Bad Infl	**Good Infl**	**Buiting BH**	**5p.**		**10p.**		**15p.**		**20p.**		**25p.**		**40p.**	
			upk	budg	upk	budg	uupk	budg	Uupk	budg	upk	budg	upk	budg
0.5	0.5	0.5	5.5	8	3,45	8.	3,5	7,8	3,5	7,8	3,5	7,8	3,5	7,8
1	0.6	0.5	10	7	11	6	11	8	10	6	12	4	11	6
0.8	0.9	0.9	22	13	23	13	22	13	23	13	22	13	23	13

Табл. 2. Чисельність ЗС за групами здоров'я та загальна чисельність

	BadInfl	**GoodInfl**	**Quiting**	**5p.**	**10p.**	**15p.**	**20.p**	**25p.**	**40p.**
SumHealthy	0.5	0.5	0.5	250	400	300	300	300	300
SumBH	0.5	0.5	0.5	200	350	250	250	250	250
SumRF	0.5	0.5	0.5	50	100	100	100	100	100
SumSick	0.5	0.5	0.5	25	50	50	50	50	50
SumChron	0.5	0.5	0.5	125	150	130	130	130	130
SumHealthy	1	0.6	0.5	350	350	350	350	350	350
SumBH	1	0.6	0.5	300	300	300	300	300	300
SumRF	1	0.6	0.5	75	75	75	75	75	75
SumSick	1	0.6	0.5	50	50	50	50	50	50
SumChron	1	0.6	0.5	100	100	100	100	100	100
SumHealthy	0.8	0.9	0.9	600	600	600	600	600	600
SumBH	0.8	0.9	0.9	350	350	350	350	350	350
SumRF	0.8	0.9	0.9	75	75	75	75	75	75
SumSick	0.8	0.9	0.9	50	50	50	50	50	50
SumChron	0.8	0.9	0.9	80	80	80	80	80	80

У випадку із збільшенням бюджету можемо бачити, що такі дії дозволяють використовувати кошти ефективніше, чисельність ЗС зростає і стабільна, захворюваність значно знизилась, хронічні захворювання також значно зменшились.

Висновки

Проведені дослідження щодо СОЗ ЗС показують, що потрібно збільшення фінансування на 30-40 % для ефективного запровадження різноманітних програм, спрямованих на профілактику

і попередження захворювань і підвищення рівня здоров'я для боєздатності ЗСУ.

Серед неочікуваних результатів потрібно відзначити невисоку ефективність різкого покращення соціальних умов, без проведення великої кількості профілактичних і попереджувальних заходів. Це зумовлено дуже високою вартістю таких дій. Проведення роботи з особовим складом та впровадження профілактичних програм набагато ефективніше покращує СОЗ в цілому. Однак повне знехтування соціальними факторами також може призвести до негативних змін.

Таким чином можна зробити висновок, що кошти на покращення рівня життя та зменшення соціального тиску слід виділяти з огляду на нормальне фінансування профілактичних і попереджувальних заходів. Така стратегія дозволить майже вдвічі покращити показники СОЗ ЗСУ та на 40-50 % підвищити боєздатність ЗСУ.

Література

1. Марценюк В.П. Компьютерно-математическое моделирование процессов с последействием в медицине // Радиоэлектроника. Информатика. Управление. – 2001. – № 2. – С. 102-106
2. Марценюк В.П., Лісничук Н.Є., Баранюк І.О. Системний аналіз медичних наукових досліджень у динаміці патологічних процесів // Штучний інтелект. – 2004. – № 1. – С. 66-72
3. Лопін Євгеній Борисович. Наукове обгрунтування та розробка моделі завантаження лікува ьних закладів охорони здоров'я пацієнтами : Дис... канд. наук: 14.03.11 – 2008
4. В.П. Марценюк. Про Web-інтегроване програмне середовище підтримки медичних системних досліджень // Тернопільська державна медична академія, Україна // http://iai.dn.ua/public/JournalAI_2004_4/Razdel2/08_Martsenyuk.pdf
5. Моделювання процесу управління системою охорони здоров'я Державної прикордонної служби України Ю.В. Вороненко, В.П. Мегедь Національна медична академія післядипломної освіти ім. П.Л. Шупика Управління охорони здоров'я Державної прикордонної служби України http://www.umj.com.ua/wp-content/uploads/archive/71/pdf/1442_ukr.pdf?upload=
6. Томашевський В.М. Моделювання систем. [Текст]// К.: Видавнича група BHV, 2005. – 352 с
7. Левченко Ф.М. Системна динаміка лікувально-профілактичного забезпечення військ (сил): нові технології управління організаційно-медичними процесами // Збірник праць Першого Всеукраїнського з'їзду "Медична та біологічна інформатика і кібернетика" з міжнародною участю. – Київ, 2010. – С. 58.
8. Levi J. Prevention for a healthier America: investments in disease prevention yield significant savings, stronger communities // Levi J., Segal L.M., Juliano C. // Washington, DC: Trust for America's Health; July, 2008. [Електронний ресур] // Режим доступу: http://healthyamericans.org/reports/prevention08.pdf
9. Schoen C. Bending the curve: options for achieving savings and improving value in U.S. health spending // Schoen C., Guterman S., Shih A., Lau J., Kasimow S., Gauthier A., Davis K. // New York, NY: Commonwealth Fund;December18,2007. [Електронний ресурс] // Режим доступу: http://www.commonwealthfund.org/publications/publications_show.htm?doc_id=620087
10. Hirsch G. Achieving health care reform in the United States: toward a whole-system understanding // Hirsch G, Homer J, McDonnell G, Milstein B. // 23rd International Conference of the System Dynamics Society; Boston, MA; July 17-21,2005. // [Електронний ресурс] // Режим доступу: http://www.systemdynamics.org/conf2005/proceed/papers/HIRSC406.pdf
11. Homer J. System Dynamics Applications at the Federal Centers for Disease Control and Prevention [Текст]// Institute on Systems Science and Health University of Michigan School of Public Health May 6, 2009
12. Сайт розробників системи імітаційного моделювання Stella (TM) [Електронний ресур] // Режим доступу: www.iseesystems.com/software/Education/StellaSoftware.aspx
13. Hirsch G. A System Dynamics Model for Planning Cardiovascular Disease Interventions [Текст]/ Hirsch G., Homer J. // American Journal of Public Health Vol. 100, No. 4 616-622 April 2010
14. Milshtein B. "HealthBound" Policy Simulation Game. Overall Design, Preliminary Insights, and Future Directions // Milstein B., Hirch G., Homer J. // [Електронний ресурс] // Режим доступу: http://www.cdc.gov/healthbound/

УДК 681.51

ЯНЧЕВСЬКИЙ С.Л.

МНОГОКРИТЕРИАЛЬНАЯ ОПТИМИЗАЦИЯ ПЛАНИРОВАНИЯ КОСМИЧЕСКОЙ СЪЕМКИ НА ОСНОВЕ ГЕОПРОСТОРАНСТВЕННОЙ ЕКСПЕРТНОЙ ИНФОРМАЦИИ

В данной работе предложен подход к планированию космической съемки на основе методов многоритериальной оптимизации и геопространственной экспертной информации. Частные критерии оптимизации определяют степень удовлетворения потребностей конкретных ведомств в спутниковой информации и формируются на основе нечетких экспертных геопространственных оценок. В результате обобщения (нелинейной свертки) экспертных оценок строится карта обобщенных потребностей ведомств, которая может быть использована в системе поддержки принятия решений процесса планирования съемки для КА «СІЧ-2».

An approach to optimal programming of satellite observations is presented. It is based on multiobjective optimization and geospatial expert information. Each partial criterion to be optimized defines a demand for space images of particular ministry and is formed using fuzzy expert geospatial judgments. As a result of convolution of partial criteria we obtain 2d map of generalized demand of every ministry that can be used to support decision making in programming of satellite observation by “Sich-2”.

Введение

Имеющиеся подходы к планированию съемки земной поверхности космическими аппаратами (КА) дистанционного зондирования Земли (ДЗЗ) до сих пор ориентируются на КА прошлых поколений, которые имели невысокую производительность и использовались преимущественно для нужд оборонных ведомств и, по остаточному принципу, для гражданских потребностей. Увеличение возможностей КА ДЗЗ совпадает со значительным увеличением спроса на их информацию. Данные ДЗЗ все более затребованы МЧС, Минагрополитики, Минэкологии, Госкомземом и т.д. [15]. Выполнить своевременно все заявки от такого количества потребителей, интересы которых могут не совпадать, достаточно сложно.

Соответственно, усложняется процесс планирования съемок КА ДЗЗ. Систем поддержки принятия решений для выполнения таких работ в Украине практически нет. Лицам, принимающим решение (ЛПР), приходится по собственному усмотрению ранжировать заявки, полученные на всех этапах планирования. На ЛПР в процессе планирования работы КА ДЗЗ возлагается все большая нагрузка и ответственность. Современная система планирования является практически полностью экспертной системой. Для ее улучшения необходимы новые алгоритмы планирования работы целевой аппаратуры спутников ДЗЗ.

В 2011 году НКАУ планирует запустить КА ДЗЗ «Сич-2». Один КА не сможет удовлетворить потребности всех заинтересованных министерств и ведомств, однако, оптимизация использования целевой аппаратуры КА «Сич-2» позволит выполнить максимально возможное количество заявок потребителей из разных ведомств [6]. При этом особую актуальность обретают подходы, дающие возможность формализации и проведения соответствующих вычислений с использованием совокупного опыта ЛПР. Очевидно, что такие экспертные знания имеют «нечеткую природу», что еще более усложняет задачу. Интересы органов исполнительной власти (ОИВ) относительно данных ДЗЗ в определенные периоды времени могут не совпадать и порождать перманентный конфликт, решение которого, как правило, заключается в рациональном планировании работы целевой аппаратуры КА. Для решения этой задачи необходимо оценить заинтересованность каждого ОИВ в данных ДЗЗ на общий район интереса (в нашем случае территория Украины). Это позволит принять оптимальное решение при наличии заявок и обеспечит эффективную работу целевой аппаратур на т.н. «холостых» витках (при отсутствии заявок), или при наличии свободного ресурса КА ДЗЗ. В этой связи, планирование предлагается проводить на основе данных экспертного опроса.

Существующие алгоритмы перспективного и долгосрочного планирования достаточно полно рассмотрены в работе [9], отдельные подходы к

планированию работы КА ДЗЗ рассмотрены в монографиях [7], оригинальная методика планирования работы бортовой целевой аппаратуры (ЦА) предложена в статье [8].

Однако все эти научные работы не обеспечивают решения проблем, обусловленных упомянутыми выше тенденциями в развитии и планировании работы современных систем ДЗЗ. Для комплексного решения этих проблем необходимо применять методы многокритериальной оптимизации [4] и системного анализа [5].

В роботах [6,15] предложен подход к многокритериальной оптимизации планирования покупки спутниковых снимков низкого и среднего разрешения с учетом потребностей различных ведомств, а также имеющихся финансовых ограничений. Подход основывается на оптимизации векторозначной функции степени удовлетворения нужд ОИВ, заинтересованных в использовании спутниковой информации.

В основу предложенного подхода положены функции потребностей ведомств, которые строятся по результатам экспертного опроса. Однако в случае использования данных высокого разрешения эксперты должны определить не только ориентировочное время съемки, но и конкретизировать территорию, которая представляет наибольший интерес.

Поэтому в данном случае частные критерии оптимизации, которые определяются уровнем удовлетворения нужд ведомств, зависят от векторного аргумента. Соответственно, в ходе экспертного опроса и обработки оценок возникает необходимость оценки геопространственной информации субъективного характера. Для решения такой задачи целесообразно использовать методы геопространственного интеллекта [1]. В тоже время, для формализации неопределенности субъективного характера целесообразно использовать мараматический аппарат теории нечетких множеств [3]. В данной работе формулируется задача многокритериальной оптимизации плана космической съемки для КА ДЗЗ высокого разрешения «Сич-2» с учетом потребностей различных ОИВ и имеющихся технических ограничений. Для построения функции потребностей государственных ведомств применены методы экспертного опроса, а для обработки результатов предлагается использовать методы геопространственного интеллекта и теории нечетких множеств.

Содержательная постановка задачи многокритериальной оптимизации плана съемки

Пусть заданна определенная территория (в нашем случае Украина и приграничная полоса), относительно которой необходимо составить оперативный план космической съемки для КА ДЗЗ «Сич-2» с учетом всей совокупности заявок от ОИВ.

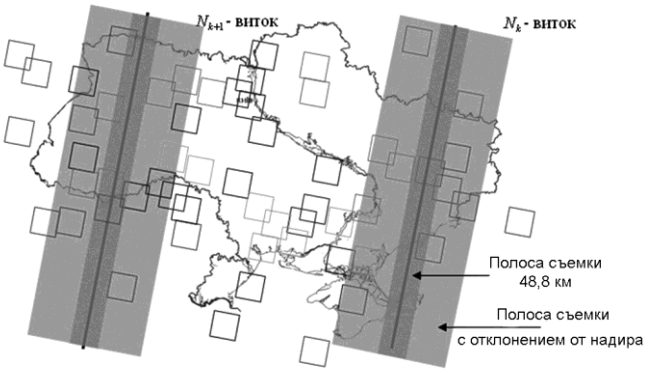


Рис. 1. Вариант заказа на съемку от разных министерств и ведомств (МЧС, Минприроди, Минагрополитики и т.д.)

Для этой территории формируется определенное количество заявок, которые должными быть выполнены почти в одно и то же время (рис.1). Исходя из ТТХ КА «Сич-2», полоса съемки аппарата составляет 48,8 км [12]. Соответственно размер кадра составляет 48,8х48,8 км, полосы – 48,8х300 км. Учитывая возможности КА относительно проведения съемки с отклонением от надира $\pm 35^{\circ}$, полоса захвата может быть расширена до 500 км с ухудшением разрешения с 8,2 м до 15 м. Однако, даже при таких условиях выполнить все заявки в назначенный срок с полным удовлетворением нужд ОИВ практически невозможно.

Поэтому возникает задача составления плана съемок с отклонением КА от надира в антересах наиболее заинтересованных и наиболее важных потребителей. Ввиду необходимости одновременно удовлетворить нужды нескольких ведомств, поставленная задача является многокритериальной задачей оптимизации, в которой, аналогично работе [14], частными критериями оптимальности является среднеквадратичное отклонение от функций потребностей конкретных ведомств. Но в данном случае множество возможных решений определяется не только временами съемки, но и географическими координатами. Т.е. частичные критерии являются скалярными функциями ве-

кторного аргумента, а множество возможных решений $X \subset E^3$ состоит из векторов $x = \{x_i\}_{i=1}^3$ трехмерного Эвклидова пространства, где x_1, x_2 – географические координаты центра сегмента, который снимается, $x_3 = t$ – время съемки.

Учитывая особенности функционирования КА, решение на планирование съемок принимается с учетом внешних влияний, которые опишем вектором *r* заданным на множестве возможных факторов *R*. Под внешними влияниями будем понимать статистические и прогностические данные относительно условий съемки, например облачности. Ситуация, которая оценивается при планировании в результате принятия многокритериального решения на проведение съемки *x* в заданных условиях *r*, характеризуется декартовым произведением $C = X \times R$. Ставится задача обеспечения наивысшей эффективности космической съемки при максимально возможных объемах получения снимков с наиболее возможным соблюдением интересов ОИВ.

Качество решения оценивается по совокупности частных критериев определяющихся целевыми функциями потребностей конкретных ведомств-потребителей, которые формируют s-мерный вектор $y(x) = \{y_k(x)\}_{k=1}^s \subset F$. Выражение $y \subset F$ означает принадлежность вектора потребностей *y* к классу *F* допустимых векторов эффективности выполнения съемки.

Математическая постановка векторной задачи оптимизации

В задаче оптимизации планирования съемки для аппаратов высокого разрешения, раз мерность вектора независимых переменных равна 3. То есть множество возможных решений относительно проведения съемки $X \subset R^3$ состоит из векторов $x = \{x_i\}_{i=1}^3$ трехмерного Эвклидова пространства, где x_1, x_2 – географические координаты центра сегмента, который снимается, а $x_3 = t$ – время съемки.

Задача многокритериальной оптимизации плана проведения космической съемки состоит в определении такого решения $x^* \in X$, которое при заданных условиях, связях и ограничениях оптимизирует векторозначную функцию эффективности планирования зависящую от векторного аргумента

$$x^* = \arg\min_{x \in X} f(x) = \arg\min_{x \in X} (y(x) - \hat{y}(x))^2 \quad (1)$$

при заданных условиях $r \subset R$ и ограничениях

$$s(\hat{y}(x)) \le S, \quad (2)$$

которые для данного КА обусловлены техническими характеристиками целевой аппаратуры «Сич-2».

В соотношении (1) $y(x) = \{y_k(x)\}_{k=1}^s$ – это векторозначная функция потребностей ведомств в спутниковой информации, $\hat{y}(x) = \left\{\hat{y}_k(x)\right\}_{k=1}^s$ – реальная степень удовлетворения потребностей ведомств в информации ДЗЗ.

В (2) $s(\hat{y}(x))$ – это объем информации, который определяется количеством возможных снимков, $S \in R$ – имеющийся бортовой ресурс (объем бортового запоминающего устройства) КА «Сич-2», который составляет 2 ГБт [12]. В частности, объем бортового запоминающего устройства (БЗУ) заполняется информацией за 89 с съемки. При условии, что получение одного снимка занимает до 7 с, КА «Сич-2» может отснять за один проход не более 12 отдельных снимков, или две сплошные полосы до 300 км. Соответственно, возникает потребность рационального распределения этого ресурса между потребителями (ведомствами). В этой ситуации, векторным критерием задачи оптимизации является квадрат нормы отклонения реальной степени удовлетворения потребностей ведомств в данных ДЗЗ от целевой функции потребностей при заданных технических ограничениях. Минимизируя отклонение от целевой функции, мы обеспечиваем максимальную эффективность съемки.

Для определения оптимальных схем проведения съемок КА ДЗЗ составим набор частных критериев. Такие критерии представляют собой количественные показатели, числовые значения которых является мерой качества системы управления [4]. Качество системы управления КА ДЗЗ в части его целевого использования, в целом, зависит от рациональности планирования съемки с учетом ограничений, которые накладываются условиями функционирования КА

на орбите, объемами и сроками выполнения заявок.

Метод решения задачи

Векторный критерий оптимизации отображает отклонение от функций потребностей заинтересованных ОИВ в информации ДЗЗ в определенном временном интервале с учетом технических ограничений. Значение целевых функций на дискретные моменты времени определяется по результатам опроса экспертных групп специалистов ОИВ, заинтересованных в получении информации ДЗЗ.

Решение поставленной задачи многокритериальной оптимизации предусматривает выполнение следующих этапов [4]: формирование частных критериев; определение областей согласия и областей компромиссов; нахождение в области компромиссов решения, которое удовлетворяет условию оптимальности Парето, путем нормализации частных критериев, учета приоритетности ведомств, выбора схемы компромиссов и получения паретооптимального решения.

Частными критериями качества, оптимального планирования съемки, следует считать степень удовлетворения потребностей в получении информации ДЗЗ на определенном интервале времени, для каждого из заинтересованных ведомств, с учетом ограничений, которые накладываются ситуацией С. Ситуация в конкретный момент времени определяется следующими факторами (ограничениями): расчетной оптической видимостью районов съемки (E); геометрической видимостью (K); прогнозируемой (статистической) видимостью районов съемки (γ).

Расчетная оптическая видимость районов съемки определяется по классическим методикам, исходя из количества рабочих витков над определенным районом, величины освещенности и состояния атмосферы над этой территорией [13]. Геометрическая видимость определяется положением космического аппарата (прохождением трассы КА) над заданным районом и, соответственно, накрытием полем зрения аппаратуры БСК одной или нескольких областей интереса. Методика расчета геометрической видимости достаточно полно изложена в работе [13].

Прогнозируемая (статистическая) видимость районов съемки определяется на основе имеющейся статистики относительно конкретной территории (среднее соотношение солнечных и облачных дней). На каждую точку общей зоны интереса можно получить определенный коэффициент или прогнозируемую (статистическую) достоверность отсутствия облачности для успешного проведения космической съемки районов интереса. Рассмотрим детальнее этап формирования частных критериев.

Формирование частных критериев

Подобно скалярной задаче многокритериальной оптимизации плана закупки данных низкого разрешения [14], функции потребностей конкретных ведомств, а значит и частные критерии в (1), формируются на основе экспертного опроса. Членами таких экспертных групп должны быть опытные специалисты ОИВ имеющие четкое понимание того, сколько, по какой территории, с какой оперативностью, на какой отрезок времени необходимо видовой информации для выполнения задач данным ведомством. Поставленная задача решалась с учетом потребностей трех ведомств: МЧС, Минприроды и Минагрополитики. С целью выяснения реальной потребности этих ОИВ в данных с КА «Сич-2» на протяжении года, а также определения степени важности проведения космических съемок по определенным участкам территории Украины и пограничной полосы, в этих ведомствах были сформированы группы экспертов и проведено анкетирование. Количество экспертов в каждой из групп равнялось пяти.

Ситуация усложняется тем, что эксперты должны были определить не только пріоритетное время съемки, но и территорию, которая составляет наибольший интерес. Поэтому для экспертного опроса такого рода оказались неприменимы традиционные способы анкетирования, которые позволяют экспертам оперировать символьной или числовой информацией [10]. Высокое разрешение снимков КА приводит к необходимости учета геопространственной информации в экспертном оце нивании и применения для ее обработки методов геопространственного интеллекта [1]. Поэтому была разработана анкета специального вида, которая позволяла за короткий срок коняретизировать области интереса. Так была получена базовая информация для проведения планирования на весь год (по месяцам). Каждый эксперт, используя

фломастеры красного, синего и зеленого цветов, должен был нанести на схематические контурные карты Украины районы интереса, относительно которых имеется особая заинтересованность его ведомства в получении космических снимков (по месяцам года).

Каждый район интереса выделялся контуром определенного цвета (рис. 2) со следующими семантическими значениями:

«красный» – очень высокая потребность в получении даннях
«синий» – высокая потребность в получении даннях
«зеленый» – средняя потребность в получении даннях

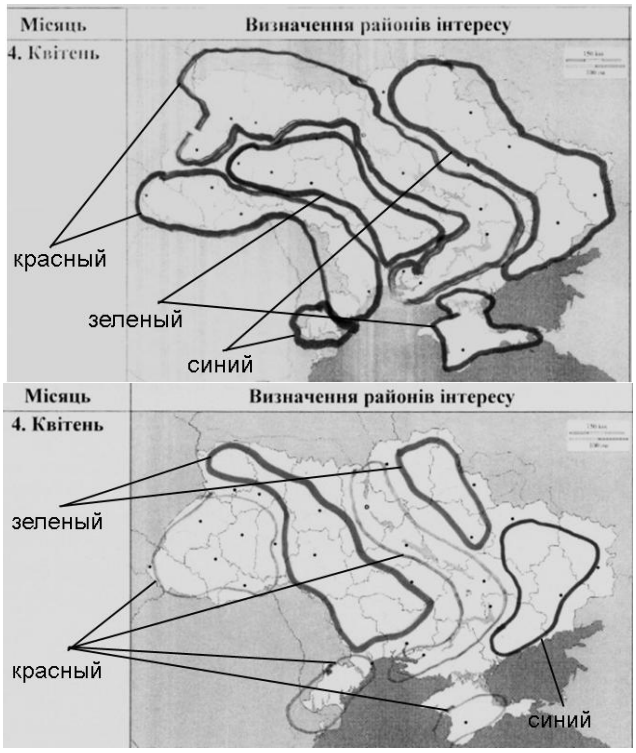


Рис. 2. Примеры заполненных экспертами МЧС анкет (на апрель)

Из рис. 2 видно, что результаты анкетирования являются весьма субъективными и часто могут противоречить друг другу. Кроме того, при нанесении информации вручную, невозможно точно определить границы областей интереса. Поэтому возникает не обходимость определения математического аппарата позволяющего формализовать и обобщить геопространственные данные экспертного опроса с учетом их субъективности. В мировой практике для решения подобных задач используются методы геопространственного интеллекта, в т.ч. с использованием аппарата нечеткой логики.

Анализ экспертной информации на основе методов геопространственного интеллекта

Основным математическим аппаратом формализации представления и обработки экспертных оценок и высказываний является теория нечетких множеств [3]. Нечеткие множества обеспечивают математическую формализацию экспертных оценок в виде лингвистических переменных и функций принадлежности, а также предоставляют методы обработки этих оценок, имеющие простую лингвистическую интерпретацию [2]. При этом конечный пользователь оперирует объектами и понятиями естественного языка, которые автоматически преобразуются к числовому виду для компьютерной обработки. Такой подход обеспечивает способ приближенного описания поведения сложных и слабоструктурированных органи зационно-технических систем [16] в условиях нестохастической неопределенности. Для формализации данных экспертного опроса на геопривязанной контурной карте Украины сформируем регулярную сетку (рис. 3) с размерами ячейки (*а, в)*.

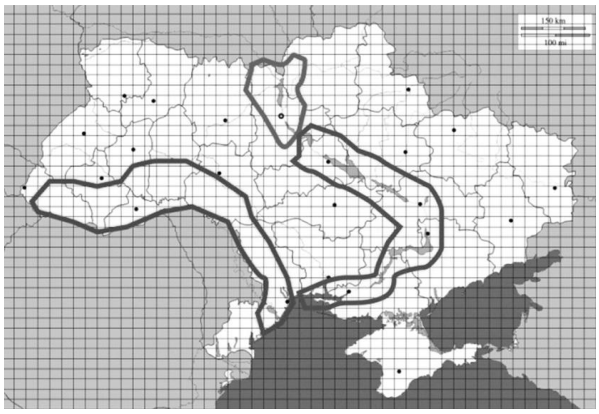


Рис. 3. Пример карты экспертного опроса с нанесенной сеткой

Размер ячейки A определяется допустимой погрешностью и масштабом карты. Каждая ячейка (точка в дискретном пространстве) $A_{(x_1,x_2)}$ с координатами центра (x_1, x_2) характеризуется оценкой потребности, т.е. значением кластера C(A), к которому эту точку отнес эксперт, и мерой достоверности такой оценки $\mu(C_A)$ – значением нечеткой функции принадлежности:

$$A_{(x_1,x_2)} = \{C(A), \mu(C_A)\} \qquad (3)$$

Мера достоверности оценки $\mu(C_A)$ априорно для каждого эксперта принимается равной $\frac{1}{N}$, где N – число экспертов. Таким образом

$$A_{(x_1,x_2)} = \left\{C(A), \frac{1}{N}\right\} \quad (4)$$

где N – число экспертов, $C(A)$ – экспертная оценка потребности, или номер кластера, который может принимать следующие значения (Таблица 1)

Табл. 1. Семантические значения экспертных оценок

Семантическое значение	Цвет	C(A)
очень высокая потребность	«красный»	**3**
высокая потребность	«синий»	**2**
средняя потребность	«зеленый»	**1**
не имеет потребности	-	**0**

Соответственно (4) определяются параметры функции принадлежности для каждой ячейки геопространственной анкеты (точки дискретного пространства) (рис. 4).

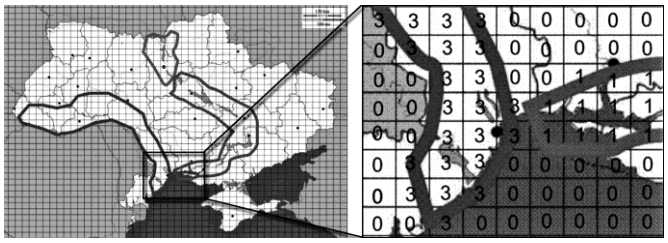


Рис. 4. Общая карта и ее фрагмент с результатами формализации экспертных оценок

Для крайних ячеек, находящихся на границе двух кластеров: C_i та C_j, если $S(C_i) > S(C_j)$, где $S(C_i)$ – площадь, которая принадлежит кластеру C_i, значение функции принадлежности определяется по формуле

$$A_{(x_1,x_2)} = \left\{C_i, \frac{1}{2N}\right\} \quad (5)$$

Тогда для крайних ячеек на границе k кластеров

$$C(A_{(x_1,x_2)}) = \arg\max_{C_i} S(C_i),\ \mu(A) = \frac{1}{kN}$$

То есть

$$A_{(x_1,x_2)} = \left\{\arg\max_{C_i} S(C_i), \frac{1}{kN}\right\} \quad (6)$$

Интегральная «нечеткая» карта, или «тело потребностей» для одного ведомства формируется следующим образом:

$$\forall(x_1,x_2)\ \ A^*_{(x_1,x_2)} = \left\{C^*, \mu^*\right\}, \quad (7)$$

где C^*, μ^* – интегральные оценки потребностей (значения кластера) и меры принадлежности.

В теории экспертного оценивания на основе нечеткой экспертной информации определены различные процедуры вычисления обобщенных оценок C^*, μ^* [10, 16]. В частности, вычисление обобщенных нечетких оценок можно привести к оцениванию средней потребности и стандартному отклонению (а) или наиболее вероятному значению оценки (б).

В случае (а) вычисления значений нечеткой функции принадлежности осуществляется по следующим формулам:

$$C^* = \frac{1}{N}\sum_{i=1}^{N} C_i(A_{(x_1,x_2)}),\ \ , \quad (8)$$

$$\mu^* = \left(\frac{1}{N-1}\sum_{i=1}^{N}(C_i(A_{(x_1,x_2)}) - C^*)^2\right)^{0,5} \quad (9)$$

Геопространственное распределение параметров обобщенных нечетких экспертных оценок для варианта (а) представлено на рис. 5.

В случае (б) определения значений обобщенной нечеткой функции принадлежности на основе наиболее вероятного значения оценки, параметры вычисляются по формулам:

$$C^* = C_{\max} = \arg\max_{c}\left\{\sum_{i=1}^{N}\delta\left(c, C_i(A_{(x_1,x_2)})\right)\right\}, \quad (10)$$

где δ – функция Кронекера.

Если существует несколько глобальных максимумов частоты, то среди них выбирается тот, который имеет максимальную оценку потребности. Для оценки достоверности $C_{\max}$ используем значение частоты для потребности $C_{\max}$:

$$\mu^* = \frac{1}{N}\sum_{i=1}^{N}\delta(C^*, C_i(A_{(x_1,x_2)})). \quad (11)$$

Формулу (11) можно представить иначе:

$$\mu^* = \sum_{i=1}^{N}\mu_i w_i,$$
$$w_i = \begin{cases}1, \textit{ якщо } C_i(A_{(x_1,x_2)}) = C^* \\ 0, \textit{ в інших випадках}\end{cases} \quad (12)$$

То есть, $w_i \neq 0$, если i-й эксперт отнес данную точку к классу C^*.

Геопространственное распределение параметров обобщенных нечетких экспертных оценок для варианта (б) согласно формулам (10)-(11) приведено на рис. 6.

Таким образом, соотношение (7) со значениями параметров (8)-(9) или (10)-(11) обеспечивают этап формирования частных критериев в задаче многокритериальной оптимизации планирования космической съемки (1)-(2). Обобщенную нечеткую оценку потребностей ведомств (7) можно визуализировать в виде интегральной трехмерной поверхности или «тела потребностей» для каждого из заинтересованных ведомств и, в дальнейшем, для всех ведомств одновременно.

Отметим, что соотношение (7) с определенными операциями получения интегральной оценки значения кластера и меры принадлежности, которые в данном случае описываются формулами (8)-(9) или (10)-(11), по сути, является реализацией схемы компромиссов в процессе решения многокритериальной задачи оптимизации на основе геопространственного интеллекта для случая использования нечеткой экспертной информации.

Таким образом, соотношения (8)-(9) или (10)-(11) обеспечивают не только этап формирования частных критериев, но и другие этапы решения задачи многокритериальной оптимизации, вплоть до определения схемы компромиссов и получения паретооптимального решения.

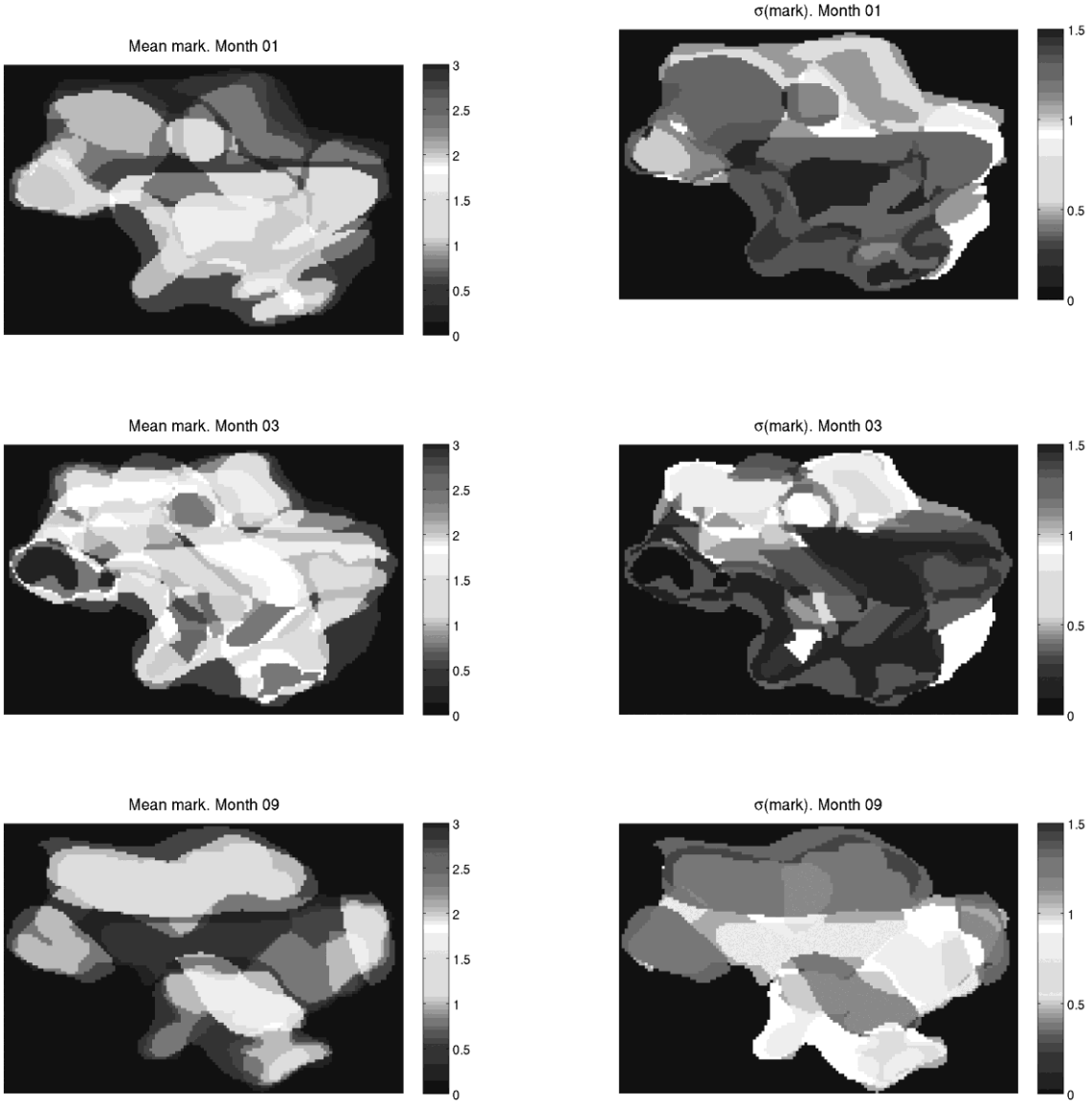


Рис. 5. Вычисление обобщенных нечетких оценок за январь, март и сентябрь на основе оценивания средней потребности (слева) и стандартного отклонения (справа) по формулам (8) и (9).

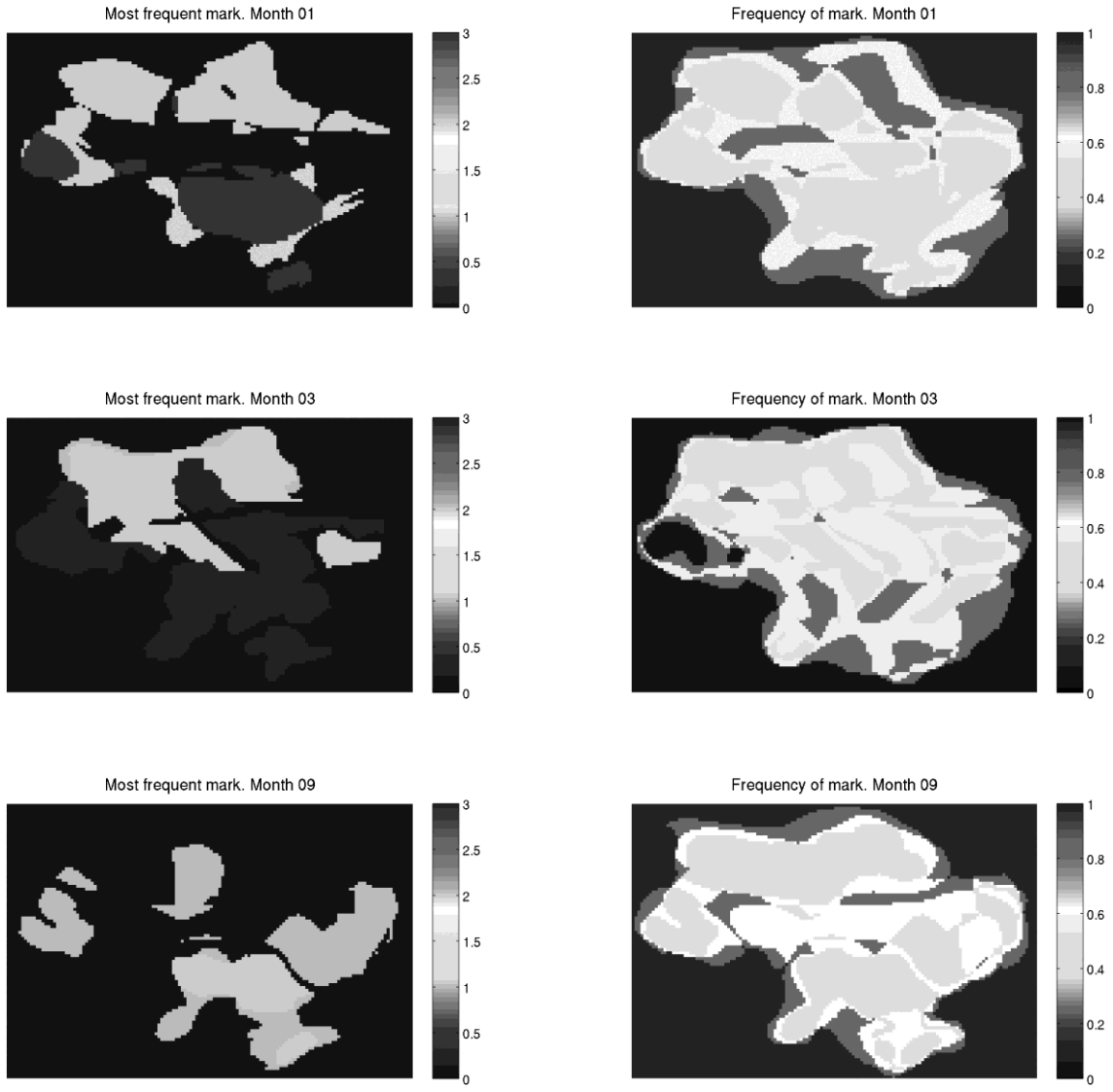


Рис. 6. Вычисление обобщенных нечетких оценок за январь, март и сентябрь на основе наиболее вероятного значения оценки по формулам (10) и (11)

Отметим, что формулами (8)-(9) или (10)-(11) не исчерпываются способы формирования интегральной нечеткой экспертной оценки.

Автором также исследовались и другие возможные варианты определения параметров интегральной нечеткой оценки, но данное исследование выходит за пределы тематики данной статьи.

Формирование оптимального маршрута съемки

Геопространственные карты обобщенных потребностей C* всех ведомств, примеры которых для МНС показаны на рис. 5 и 6, можно использовать в системах поддержки принятия решений (СППР) в процессе планировании космической съемки.

Получение интегральной оценки нужд разных ведомств позволит более обоснованно осуществлять планирование и проводить съемку наиболее актуальных районов, независимо от наличия заявок, в режиме т.н. «холостых проходов», которые имеют место в случае наличия свободного ресурса, при отсутствии заявок или их небольшому количеству на n-проход над определенной территорией.

Кроме того, это значительно упрощает задачу планирования для лица принимающего решение (ЛПР) либо ОПР, процесс оптимизации маршрутов при проведении съемок в режиме программных поворотов (РПП), когда съемка территориально разнесенных районов интереса проводится с отклонением оси визирования це-

левой аппаратур КА ДЗЗ от направления в надир на угол $\pm 35^o$ (рис. 7).

Таким образом, нечеткая экспертная информация может быть формализована и использована в СППР как априорная основа для планирования съемок КА «Сич-2». При таком подходе ОПР получает определенную оценку потенциальной востребованности на оперативную и архивную информацию для каждого района главной зоны интереса (Украина и пограничные территории) и может ее использовать в автоматизированных системах планирования съемки.

Рис. 7. Использование интегральной карты потребностей ведомств в процессе планирования космической съемки

Выводы

В работе сформулирована многокритериальная задача оптимизации планирования космической съемки для КА ДЗЗ «Сич-2» с учетом потребностей различных ведомств в получении информации для конкретного времени и региона.

Для формирования частных критериев оптимизации предложено использовать методы экспертных оценок и геопространственного анализа. Формируемая на основе экспертных оценок интегральная трехмерная поверхность или «тело потребностей» для каждого ОИВ позволяет оптимизировать процесс планирования съемок по заданным районам интереса с максимально возможным учетом потребностей всех ОИВ.

Другие варианты определения параметров интегральной нечеткой оценки исследовались автором, однако такое исследование выходит за пределы тематики данной статьи и результаты будут опубликованы в других роботах.

Разработанный метод позволяет формализовать и учитывать влияние большого количества особенностей и ограничений, которые имеют «нечеткую» природу, но в значительной степени влияют на процесс принятия решения во время планирования космической съемки.

Такой подход обеспечивает ОПР априорной базовой информацией по каждой точке главной зоны интереса на определенные периоды времени, что формализует процесс принятия решений и повышает эффективность использования полезной нагрузки КА ДЗЗ.

Подобный подход может быть использован в рамках организации СППР для ЛПР на всех этапах планирования, для КА ДЗЗ низкого, среднего и высокого разрешения.

Литература

1. Bacastow T.S., Bellafiore D.J. Redefining geospatial intelligence // American Intelligence Journal, 2009. P. 38-40.
2. Shelestov A.Yu., Kussul N.N. Using The Fuzzy-Ellipsoid Method For Robust Estimation Of The State Of A Grid System Node // Cybernetics and Systems Analysis, 2008. Т.44, № 6. С. 847-854.
3. Zadeh L.A. The concept of a linguistic variable and its application to approximate reasoning//. Information Sciences, 1975. Vol. 8. P. 301—357.
4. Воронин А. Н., Ю. К. Зиатдинов, А.В. Харченко Сложные технические и эргатические системы: методы исследования // Харьков: Факт, 1997. 240 с.
5. Згуровский М.З., Панкратова Н.Д. Системный анализ. Проблемы, методология, приложения // Киев: «Наукова думка», 2005.
6. Куссуль Н.М., Фриз В.П., Янчевський С.Л. Возможный подход к рациональному планированию космической съемки Земли на основе многокритериальной оптимизации.// Сборник научных работ №4, Житомир: ЖВІНАУ, 2011. С. 97-105.
7. Лебедев А.А., Несторенко О.П. Космические системы наблюдения. Синтез и моделирование // Москва: Машиностроение, 1991. 224 с.
8. Машков О.А., Фриз С.П. Методика оптимизации планирования работы орбитальных средств космических систем наблюдения // Сборник научных работ. Житомир: ЖВИРЭ, 2003. С. 80-91.
9. Пясковский Д.В. Основы построения систем управления космическими аппаратами. Ч.1. Конспект

лекций./ Пясковский Д.В., Парфенюк В.Г.; - Житомир: ЖВИРЭ, 1998. 187 с.
10. Самохвалов Ю.Я., Науменко Э.М. *Экспертное оценивание. Методический аспект* // К.: ДУИКТ, 2007. 262 с.
11. Скребушевский Б.С. Формирование орбит космических аппаратов // М.: Машиностроение, 1990. 256 с.
12. Тактико-техническое задание на опытно-конструкторскую работу «Космическая система оптико-электронного наблюдения и связи «Сич-2» от 07.02.2002 г. // ДП «КБ «Южное». 7-8 с.
13. Ханцеверов Ф.Р., Остроухов В.В Моделирование космических систем изучения естественных ресурсов Земли // М.: Машиностроение, 1989. 264 с.
14. Янчевский С.Л. Оптимизация планирования получения информации от систем дистанционного зондирования Земли среднего разрешения при условии финансовых ограничений// Научные работы ДонНТУ Серия "Информатика, кибернетика и вычислительная техника". 2011.
15. Янчевский С.Л. Оптимизация использования информационного ресурса космических систем дистанционного зондирования Земле для нужд системы государственного управления// Материалы докладов Второй всеукраинской конференции Аэрокосмические наблюдения в интересах постоянного развития и безопасности GEO-UA 2010. Киев, «Освіта України», 2010. С. 166-168.
16. Ярушкина Н.Г., Афанасьева Т.В., Перфильева И.Г. Интеллектуальный анализ временных рядов: учебное пособие // Ульяновск: УлГТУ, 2010. 320 с.

УДК 004.4

МОЛЧАНОВСЬКИЙ О.І.,
МИСТЕЦКИЙ В. А.,
НГІЄМ ЛЕ КУАН,
БУЙ ХИУ ДАТ

ЗАДАЧА ПОШУКУ ФАЙЛІВ ЗОБРАЖЕНЬ

В даній статті описується проблема вирішення задачі пошуку файлів зображень. Проведено аналіз існуючих рішень та розглянуті відомі системи, що дозволяють знаходити схожі зображення. Описана методологія тестування таких систем та наведені результати тестування системи Retrievr. Розглянуті підходи до проектування системи пошуку зображень, що розробляється. Наведено метод представлення зображень у базі даних та механізм порівняння вхідного зображення з зображеннями в базі даних на предмет схожості. Розглядається механізм порівняння зображень за спектрально-частотною характеристикою наявності кольорів. Описано підхід до оцінки сегментної структури зображень та використання цієї структури при порівнянні зображень. Також наведені висновки, щодо проведеної роботи та сформульовано проблеми для подальшого дослідження.

In this paper we describe the problem of image file search. We analyzed existed methods and systems that allow the user to find similar images. We propose testing methodology for such systems and provide the test results for Retrievr system [2]. In the second part firstly we describe the presentation of images in the corresponding database. Secondary we describe the method of image comparison based on spectral-frequency color characteristics. The estimation approach for segmental structure of images is provided as well as search method based on this approach. At the end we provide overall summary and the trends for the feature work.

Вступ

Цілями даної статті є розгляд існуючих інструментів пошуку схожих зображень, а також розробка оригінального інструментарію для розв'язання вказаної проблеми. При розгляді існуючих інструментів важливо висунути припущення щодо методів, які вико ристовуються ними. Це можна зробити на основі інформації, яка надається в процесі експлуатації цих інструментів. Результати таких припущень є важливими с точки зору побудови власного інструментарію, адже дозволяє з'ясувати, що є найбільш важливим, а що другорядним при пошуку схожих зображень.

В умовах бурхливого розвитку та по пуляризації інформаційних технологій значно збільшується кількість електронної інформації. У зв'язку з цим виникає необхідність створювати засоби її каталогізації та пошуку.

На сьогоднішній момент існує велика кількість систем пошуку текстової інформації, що здатні враховувати популярність ресурсу, на якому ця інформація знаходиться, максимізувати відповідність результатів пошуку до запиту користувача, використовуючи при цьому геолокаційні дані про місцезнаходження користувача, інформацію про його попередні пошуки, вподобання тощо.

Проте значну частину інформації у всесвітній мережі складають мультимедійні файли, такі як музика, відео та зображення. Пошук музикальних файлів спрощений насамперед тим, що для більшості файлів існують закладені в нього метадані – теги, завдяки яким можна сказати про назву пісні, альбому чи виконавця. Відео файли найчастіше зберігаються на спеціалізованих сервісах, в яких слідкують за тим, щоб назва та опис, що супроводжує відео файл відповідав йому.

Найбільшу складність для пошуку складають файли зображень, що найчастіше не мають тегів, назва файлів не несе інформації про їх склад (більшість фотокамер, хостингів та редакторів зберігають фотографії з службовими назвами). Тому на сьогоднішній момент основним типом пошуку файлів зображень є пошук по текстовій інформації, що супроводжує це зображення на інтернет сторінці. Проте такий пошук не може задовольнити користувачів, які хочуть знайди саме певне зображення, що, наприклад, схоже на деяке зображення, яке користувач вже має.

Об'єктом нашого дослідження стала саме така система, яка зможе знаходити у своїй базі зображень зображення, що схоже не те, яке задав користувач у вигляді іншого зображення.

На сьогоднішній момент існує велика кількість вузько-направлених систем розпізнавання та обробки зображень. Такими системами є, наприклад, системи розпізнавання номерних знаків автомобілів, системи цифрового зору роботів, системи розпізнавання будинків на знімках супутників тощо. Головною особивістю таких систем є те, що в них наперед відомий формат вхідних зображень та об'єктів, що необхідно розпізнати. Це дає змогу використовувати найбільш підходящі алгоритми для обробки саме таких об'єктів. В поставленій нами задачі основною проблемою являється велика кількість типів зображень, між якими система повинна провести порівняння та прийняти рішення про схожість чи відмінність їх по деяким ознакам.

Насамперед необхідно визначитись з тим, саме за якими ознаками система повинна порівнювати зображення та визначати схожість між ними. За результатами проведеного дослідження [1] ми виділили наступні основні критерії, за якими можна порівнювати зображення: кольорова насиченість, сегментованість зображення, наявність деяких головних об'єктів (наприклад будинків, тварин, людей тощо) та жанр зображення (пейзаж, портрет, креслення і т.д.). Отже, система повинна враховувати ці критерії для пошуку та порівняння зображень.

Аналіз існуючих рішень

Зараз існує декілька систем, що намагаються вирішувати поставлену задачу: Retrievr [2], Gazopa [3], TinEye [4] та деякі інші. Ми намагались відповісти на питання, яким чином можна оцінити роботу тієї чи іншої системи і створили деяку методологію оцінки системи пошуку зображень [1].

Для цього нами було запропоновано розбиття тестового корпусу зображень на групи (класи) за жанрово-семантичними ознаками. Таких груп було сформовано 12, серед яких: окремі об'єкти (автомобілі, літаки), креслення (блок схеми, таблиці), пейзажі, тварини, малюнки, фотопортрети. Більшість зображень можна віднести до деякої групи, проте присутня неоднозначність (деяке зображення можна віднести одразу до декількох класів).

Для кожного вхідного зображення виставляються оцінки від 0 до 10 балів по чотирьом параметрам: сегментованість (кількість яскраво виражених об'єктів), контрастність (оцінка чіткості переходів між об'єктами), кольорова насиченість (кількість різних кольорів на зображенні), кількість відтінків (оцінка кількості відтінків одного кольору; для всіх кольорів зображення). Оцінка цих параметрів проводиться людиною експертом.

За цими оцінками були відібрані еталонні зображення, в яких певна характеристика оцінюється в 0, 3, 5, 7 та 10 балів. Еталони служать для спрощення оцінки зображень експертами.

До оцінки результуючих зображень, окрім цих чотирьох параметрів, додаються ще дві характеристики від 0 до 10 балів: семантична схожість результату з вхідним зображенням (сюжет, тематика, жанр) та кольорова схожість.

Для тестування була створена база зображень в 150 зображень, в якій були представлені зображення з усіх класів.

Після оцінки за вищевказаною схемою вхідних та результуючих зображень експертами проводиться статистична обробка отриманих даних (розрахунок та порівняння математичних сподівань та середньоквадратичного відхилення оцінок результуючих зображень від оцінок вхідних зображень і т.д.). Отримані залежності можна використовувати в якості оцінки пошуку системою взагалі, в розрізі окремих класів та характеристик.

За цією методологією була оцінена система Retrievr.

На рис. 1 зображений графік розподілення відносної кількості зображень у кожному класі в діапазонах оцінок семантичної схожості.

Для кожного вхідного зображення деякого класу значення середньої семантичної схожості розраховувалось як середнє арифметичне усіх результуючих зображень для цього вхідного зображення.

Як видно, майже у всіх класах для абсолютної більшості вхідних зображень середня оцінка критерію семантичної схожості лежить в діапазоні 0-2. Однак для класу №8 (пейзажі) ми бачимо ярко-виражений глобальний мінімум кривої для діапазону 0-2 та значне зростання кривої діапазону 9-10 та діапазонів 6-8 і 3-5. Ми пояснюємо це тим, що в базі зображень системи Retrievr знаходиться дуже велика кількість зображень на цю тематику (відносно зображень з інших класів).

Рис. 1. Розподілення оцінок результуючих зображень за критерієм семантичної схожості

З цього можна зробити висновок, що система ніяк не враховує семантичну схожість при пошуку зображень.

Також були розраховані середні арифметичні відхилення оцінок сегментованості, контрастності, кольорової насиченості та кількості відтінків і розрізі класів за формулою (1).

$$P_m = \frac{\sum_{i=1}^{n_m} | mark\_res_i - \frac{\sum_{j=1}^{k_i} mark\_inp_j}{k_i}}{n_m} \quad 1)$$

,

де P_m – середнє арифметичне відхилення в m-му класі;

n_m – кількість вхідних зображень в m-му класі;

k_i – кількість результуючих зображень для і-го вхідного зображення;

$mark\_res_j$ – оцінка відповідного результуючого зображення;

$mark\_inp_i$ – оцінка відповідного вхідного зображення.

Отримані результати можна побачити на рис. 2.

Також були розраховані середнє арифметичне дисперсії оцінок сегментованості, контрастності, кольорової насиченості та кількості відтінків в розрізі класів за формулою (2).

$$D_m = \frac{\sum_{i=1}^{n_m} (mark\_inp_i - \frac{\sum_{j=1}^{k_i} mark\_res_j}{k_i})^2}{n_m}, \quad (2)$$

де D_m – середнє арифметичне дисперсії в m-му класі;

n_m – кількість вхідних зображень в m-му класі;

k_i – кількість результуючих зображень для і-го вхідного зображення;

$mark\_res_j$ – оцінка відповідного результуючого зображення;

$mark\_inp_i$ – оцінка відповідного вхідного зображення.

Отримані результати можна побачити на рисунку 3.

З поданих вище графіків можна зробити висновок, що результати мають досить великі відхилення та дисперсії від параметрів вхідного зображення. Виключеннями є значення дисперсій по параметру кольорової насиченості в 2 (блок схеми), 4 (тварини), 9 (портрети), 10 (графіка) та 12 (макрозйомка) класах. Причинами цього, на наш погляд, є кольорова монотонність більшості зображень в цих класах.

Таким чином, основним критерієм пошуку в системі Retrievr є кольорова схожість зображень. Всі інші параметри враховуються слабко, чи не враховуються взагалі.

Постановка задачі

Розглянувши деякі з існуючих систем пошуку зображень, нами була сформульована наступна задача. Необхідно розробити комплекс програмних модулів для пошуку файлів зображень, які розміщуються в локальному сховищі даних (локальний комп'ютер чи локальна мережа) або глобальній мережі Інтернет.

Реалізація даної задачі передбачає розгляд та реалізацію різних критеріїв пошуку. До них можна віднести порівняння зображень на основі кольорового спектру, на основі просторових характеристик (сегментів) тощо. Нижче ми послідовно розглянемо ці критерії.

Рис. 2. Графік середньо-арифметичних відхилень по оцінкам сегментованості, контрасту, кількості кольорів та відтінків

Рис. 3. Графік середньо-арифметичних дисперсій по оцінкам сегментованості, контрасту, кількості кольорів та відтінків

Представлення зображень

Одною з основних проблем при створенні системи пошуку зображень є великі об'єми інформації, що необхідно обробити та порівняти за фіксовано малий час запиту користувача. Для цього потрібно розробити механізм представлення зображень бази даних в такому вигляді, щоб зберегти основні дані про кольори та їх розміщення на оригіналі зображення для подальшого порівняння цих даних з вхідним зображенням від користувача.

В якості такого механізму був обраний апарат модифікованого дискретного вейвлет-

перетворення [5]. Для зменшення обчислювальної складності не проводиться розрахунок часткових різниць на кожній ітерації та сумування із зміною частотно-часової локалізації та зріджування вхідного сигналу. Отриманий після перетворення об'єкт будемо вважати деяким ескізом зображення.

Перед створенням цього ескізу всі зображення нормалізуються до однакового квадратного розміру (наприклад 1024x1024). В якості вікна вейвлет-перетворення було обрано вікно розміром 32x32. Результатом сумування для кожного місцезнаходження вікна зводяться до середньо арифметичного значень кольорів всіх точок в цьому вікні. Отже, ми отримаємо 32x32 значення кольорів по 3 компоненти кожен (використовується стандартне кольорове представлення RGB). Таким чином вимірність ескізу буде 32x32x3.

Значення для компонент r, g та b вікна (i,j) буде розраховуватись за формулами (3), (4) та (5).

$$r_{i,j} = \frac{\sum_{l=32i}^{32i+32}\sum_{k=32j}^{32j+32}(r_{lk})}{32\cdot 32}, \qquad (3)$$

$$g_{i,j} = \frac{\sum_{l=32i}^{32i+32}\sum_{k=32j}^{32j+32}(g_{lk})}{32\cdot 32}, \qquad (4)$$

$$b_{i,j} = \frac{\sum_{l=32i}^{32i+32}\sum_{k=32j}^{32j+32}(b_{lk})}{32\cdot 32}. \qquad (5)$$

В якості міри відстані (оцінки схожості) між ескізами двох зображень була обрана Манхетенська відстань в кольоровому просторі RGB. Ця відстань розраховується, як відстань між двома точками в 32x32x3-мірному просторі.

Кожний об'єкт має 32x32x3 блоки (властивості) (всього 3072). Вважаємо, що 3 компоненти кольору кожного з 32x32 блоків являються тривимірними значеннями блоків, різниця яких усереднюється.

Кожна різниця нормалізується (ділиться на 255), а отримана відстань усереднюється (ділиться на 32x32).

Таким чином, манхетенська відстань між двома ескізами лежить в діапазоні від 0 до 1 та обраховується за формулою (6).

$$P = \frac{\sum_{i=1}^{32\cdot 32}\left(\left|\frac{R_{Ai}-R_{Bi}}{255}\right| + \left|\frac{G_{Ai}-G_{Bi}}{255}\right| + \left|\frac{B_{Ai}-B_{Bi}}{255}\right|\right)}{3}}{32\cdot 32}, \qquad (6)$$

де P – манхетенська відстань між ескізами;

A – перший ескіз;

B – другий ескіз;

R,G,B – кольорові компоненти простору RGB відповідного блоку відповідного ескізу.

Тобто, чим менша отримана відстань, тим ескізі більш схожі.

Дана метрика була обрана в якості компромісу між «жорсткістю» оцінки (в порівнянні, наприклад, с Манхетенською відстанню між ескізами по параметру Hue кольорового простору HSV) та обчислювальною складністю (в порівнянні, наприклад, з евклідовою відстанню в кольоровому просторі RGB).

Наявність модулів в формулі розрахунку відстані дозволяє цьому механізму враховувати не лише кольорове співпадіння, а й враховувати розміщення тих чи інших кольорових блоків на оригінальних зображеннях.

Проте, використовуючи лише цей механізм, ще на етапі створення ескізу сильно різнокольорові зображення вироджувались в сіро-монотонні ескізи через усереднення кольору в кожному з блоків. В результаті порівняння для цих зображень апарат розраховував малі відстані з ескізами дійсно сіро-монотонних зображень в базі, що значно погіршувало результати пошуку.

Для уникнення цієї проблеми було розроблено підсистему розрахунку та врахування частотно-спектральної характеристики [6], яка характеризує процентне включення кольорів того чи іншого діапазону у зображенні.

Частотно-спектральний аналіз зображень

Для побудови частотного спектру діапазонів кольорів, що відповідають кольорам: червоний, оранжевий, жовтий, зелений, голубий, синій та фіолетовий, був застосований кольоровий простір HSV:

Компонента hue (тон) лежить в діапазоні значень 0-360^{o} та відображає тон того чи ін.-шого кольору.

Не дивлячись на математичну точність, у такої моделі є суттєвий недолік: на практиці кількість відтінків, що можна розрізнити змен-

шується при наближенні яскравості до нуля. Також на малих S і V з'являються суттєві похибки округлення при переводі компонент кольорового простору RGB в простір HSV та навпаки.

Тобто, якщо розглядати лише значення компоненти hue в кожному піддіапазоні ми будемо включати білі, сірі та чорні кольори, як показано на рисунку 4.

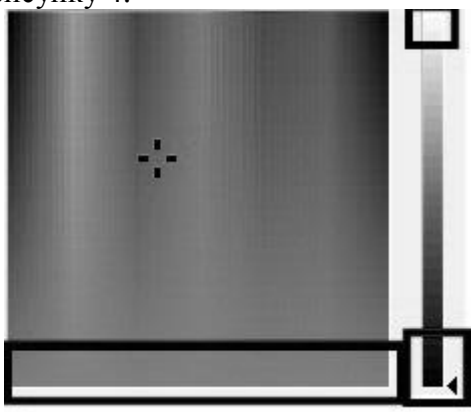


Рис. 4.

Тому для вилучення та врахування білого, сірого та чорного кольору спочатку проводиться перевірка кольору кожного пікселя на належність до цих під діапазонів в кольоровому просторі RGB.

Таким чином ми отримуємо 10 значень для наступних кольорів: червоний, оранжевий, жовтий, зелений, блакитний, синій, фіолетовий, чорний, білий та сірий. Очевидно, що сума цих значень дорівнює 1 (100% всього зображення).

Отже, розрахувавши ці характеристики ми отримуємо відображення кольорової насичення зображення, а саме його різнокольоровість чи монотонність. Тому, на основі цих характеристик, було створено спектральний фільтр, який для вхідних зображень, що мають різногольорову структуру відкидає результати, що мають ярко-виражене домінування певного кольору (мають монотонну кольорову структуру).

Цей фільтр умовно поділений на дві частини. Перша частина визначає тип зображення:

- тип 1. Різнокольорове зображення (всі значення масиву спектру <0.3, що означає, що немає кольору, який би займав більше 30% зображення);
- тип 2.Зображення що не має домінуючих кольорів (є хоча б одне значення масиву спектру, що >0.3, але <0.4);
- тип 3. Монотонне зображення (є хоча б одне значення масиву спектру, що >0.4, тобто є хоча б один колір, що займає більше 40% всього зображення і домінує в ньому).

Після цього він розраховує манхетенську відстань між спектральними характеристиками зображень (точки в 10х3-мірному просторі), що порівнюються.

В залежності від результатів роботи першої частини спектрального фільтру формується комплексна оцінка схожості ескізів. В якості комплексної оцінки виступає середнє зважене оцінки вейвлет-ескізу зображень та їх частотно-спектральних характеристик. Вона обраховується за формулою (7).

$$P = a_1 \cdot w\_mark + (1 - a_1) \cdot s\_mark, \qquad (7)$$

де a_1 – коефіцієнт, що визначений спектральним фільтром;

w_mark – оцінка схожості вейвлет-ескізів;

s_mark – оцінка схожості частотно-спектральних характеристик.

Вагові коефіцієнти задаються спектральним фільтром, як показано в таблиці 1 (конкретні значення коефіцієнтів були підібрані експериментальним шляхом).

Табл. 1. Значення коефіцієнта a_1 для різних комбінацій порівнюваних зображень

Типи	**Тип 1**	**Тип 2**	**Тип 3**
Тип 1	0,8	0,6	*
Тип 2	0,6	0,8	0,5
Тип 3	*	0,5	0,8

* – означає, що оцінка не розраховується взагалі і така пара зображень не порівнюється.

Таким чином система отримує комплексну оцінку кольорово-структурної та спектрально-частотної схожості вхідного ескізу, та деякого ескізу з бази даних. Причому вагові коефіцієнти окремих оцінок в комплексній оцінці змінюються в залежності від визначеного типу зображення.

Сегментація зображень

Іншою важливою характеристикою, що повинна враховуватись при порівнянні зображень є їх сегментна структура [7]. Сегмент – це деяка область зображення, що відокремлена від інших частин зображення достатньо вираженим контрастним переходом кольору (на границі сегменту є досить великий перехід тону чи яскравості). Найчастіше сегменти відповідають деяким об'єктам, що вирізняються на фоні зображення, чи іншого об'єкту. Також це можуть бути частини деякого об'єкту тощо. На наш погляд, сегментна структура є досить вагомою характеристикою зображення, оскільки людина звертає увагу на об'єкти, що відокремлюються від фону та інших об'єктів на зображенні. А

розмір, місце розташування та взаємозв'язок між цими об'єктами є важливою інформацією, яка повинна використовуватись при порівнянні зображень.

Таким чином, задача полягає в тому, щоб реалізувати механізм виокремлення сегментів в зображенні, причому вони повинні як найбільше відповідати реальним об'єктам зображення, а їх кількість повинна корелювати із експертними оцінками сегментованості зображення (наприклад так, як пропонується в описаній вище методології оцінки роботи систем пошуку зображень).

В якості математичного апарата для вирішення цієї задачі був обраний модифікований алгоритм Краскала побудови мінімального остового дерева. Кожний піксель зображення в цій задачі представляється в якості вершини графа, а вага ребра, що їх з'єднує визначається за формулою (8).

$$w(v_i, v_j) = \left| I(p_i) - I(p_j) \right|, \qquad (8)$$

де $I(p_i)$ – інтенсивність (яскравість) пікселя p_i.

В ході роботи алгоритму Краскала, на проміжному етапі ми будемо мати декілька різних сегментів (підмножин пікселів) з мінімальною сумарною вагою ребер всередині: сегменти будуть об'єднані ребрами мінімальної довжини, тобто з мінімальними перепадами інтенсивностей між сусідніми пікселями, але тільки до деякого значення максимального ребра (перепаду інтенсивності).

В результаті роботи алгоритму ми отримаємо множину сегментів, їх корені (корені відповідних остових дерев), потужність (кількість пікселей в кожному сегменті) та координати усіх границь кожного сегменту.

Проте в даній реалізації алгоритму сегментування кількість отриманих сегментів виявляється дуже великою, тобто алгоритм розбиває реальний об'єкт на велику кількість малих сегментів. Таким чином інформація про кількість сегментів та їх розмір не відповідає кількості реальних об'єктів на зображенні.

Тому був розроблений алгоритм, що по закінченню роботи алгоритму сегментування проводить додатковий обхід отриманих сегментів та об'єднує їх з сусідніми сегментами, якщо різниця між кольоровими компонентами (манхетенська відстань) нижче деякого порогу. Даний поріг є динамічним та визначається насамперед спектрально-частотними характеристиками зображення. Так якщо зображення є монотонним з домінуванням синього кольору, то алгоритм враховує це і збільшує пороги для об'єднання сегментів по компоненті Blue, тим самим не даючи об'єднати слабо виражений об'єкт з фоном.

Висновки та подальша робота

Універсальність системи, що розробляється накладає додаткові складності для підбору параметрів алгоритмів. Зміна параметрів в одну чи іншу сторону покращує результати порівняння для деяких типів зображень (наприклад для монотонних з великими об'єктами), але погіршує для інших (наприклад для сильно сегментованих, різнобарвних, з великою кількістю дрібних об'єктів). Тому ми здійснюємо підбір цих параметрів порівнюючи отримані результати (наприклад кількість сегментів на зображенні) з експертними оцінками та мінімізуємо середньоквадратичне відхилення отриманих системою даних із даними від експертів. Крім того, система намагається визначити тип зображення за спектрально-частотними характеристиками для того, щоб підібрати найбільш відповідні параметри алгоритмів, вагові коефіцієнти в комплексній оцінці схожості тощо.

В даний момент проводиться проектування високонавантаженої бази даних, що буде здатна оброблювати запити на великій базі зображень за прийнятний час. Ведеться розробка алгоритмів збереження інформації про структуру та взаємозв'язок сегментів для подальшого порівняння не тільки кількісних характеристик сегментів зображення (їх кількість та розмір), а й якісних (взаємне розташування, сусідство і т.д.). Також проводиться розробка модулю виявлення облич на зображення та збереження інформації про їх площу та розміщення. Це дозволить отримувати додаткову семантичну інформацію з зображення та проводити пошук, що направлений на сюжет зображення. Крім того це дозволить відокремити клас зображень – портрети, серед яких можна реалізовувати нейро-мережеві методи пошуку [8], що дають гарні результати для порівняння облич.

Список літератури

1. Молчановский А.И., Буй Хиу Дат. Методология тестирования систем поиска изображений: тези доповідей Міжнародної науково-практичної конференції [Інформаційні технології та комп'ютерна інженерія], (Вінниця, 19-21 травня 2010). – Вінниця: ВНТУ, 2010. – 487с.
2. Веб-сторінка системи Retrievr: http://labs.systemone.at/retrievr/about**.**
3. Веб-сторінка системи Gazopa: http://www.gazopa.com**.**
4. Веб-сторінка системи TinEye: http://www.tineye.com**.**
5. The engineer's ultimate guide to wavelet analysis: http://users.rowan.edu/~polikar/WAVELETS/WTtutorial.html.
6. Молчановский А.И., Нгием Лэ Куан. Спектральный анализ для задачи поиска изображений: материалы третьей Всеукраинской научно-практической конференции молодых ученых и студентов [Информационные процессы и технологии «Информатика – 2010»], (Севастополь, 26-30 апреля 2010). – Севастополь: СевНТУ, 2010. – 384с.
7. Молчановский А.И., Нгием Лэ Куан. Проблема сегментации для задачи поиска изображений: тези доповідей Міжнародної науково-практичної конференції [Інформаційні технології та комп'ютерна інженерія], (Вінниця, 19-21 травня 2010). – Вінниця: ВНТУ, 2010. – 487с.
8. Васильев В.И. Распознающие системы. – Киев: Наукова думка, 1983.-424с.

УДК 004.75

ЗАЙЧЕНКО Ю.П.,
ОВИ НАФАС АГАИ АГ ГАМИШ

ИССЛЕДОВАНИЕ ЗАВИСИМОСТИ «ДОХОДНОСТЬ-РИСК» В ЗАДАЧЕ ОПТИМИЗАЦИИ НЕЧЕТКОГО ПОРТФЕЛЯ

В статье рассмотрены задачи портфельной оптимизации в нечетких условиях. Построена математическая модель данной задачи. Исследована зависимость «оптимальная доходность - риск» для нечеткого портфеля. Определены достаточные условия при которых эта зависимость будет монотонно-убывающей. Приводятся результаты экспериментальных исследований, подтверждающие теоретические результаты.

The fuzzy portfolio optimization problem is considered. The mathematical model is constructed and the dependence “optimal benefits – risk” is investigated. The sufficient conditions for this dependence to be monotonously decreasing are obtained. The results of experimental investigations are presented.

Задача оптимизации инвестиционного портфеля в условиях неопределенности в последние годы вызывает значительный интерес. Для решения этой проблемы был предложен аппарат нечетких множеств в работах [1,2], согласно которому доходности акций и в целом доходность инвестиционного портфеля рассматриваются как нечеткие числа с заданной функцией принадлежности, а риск трактуется как возможность (субъективная вероятность) ситуации, когда реальная доходность портфеля оказывается ниже ожидаемой доходности нечеткого портфеля.

В ходе экспериментальных исследований разработанного метода были построены зависимости «оптимальная доходность-риск» для нечеткого портфеля, которые во многих случаях имели прямо противоположный вид по сравнении с классической моделью Марковица-Тоббина [3]. В предыдущей работе [4] были получены достаточные условия, при которых зависимость «оптимальная доходность-риск» в задаче нечеткой портфельной оптимизации является монотонно –убывающей для частного случая, когда портфель состоял из двух ценных бумаг (ЦБ). Целью настоящей работы является обобщение полученных результатов на случай, когда портфель состоит из произвольного числа ЦБ, получение аналитических условий, при которых зависимость «оптимальная доходность-риск» для нечеткого портфеля будет монотонно убывающей и их экспериментальная проверка.

Задача нечеткой портфельной оптимизации базируется на допущении, что доходность портфеля является нечетким числом (НЧ), описываемым тройкой параметров:

$$r = (r_{\min} = \sum_{i=1}^{N} x_i r_i ; \tilde{r} = \sum_{i=1}^{N} x_i \tilde{r}_i ; r_{\max} = \sum_{i=1}^{N} x_i r_{i2})$$

где $(r_{1i}, \tilde{r}_i, r_{2i})$ – доходность i-*той* ценной бумаги. x_i-доля i-*той бумаги в портфеле.*

Задается критериальное значение доходности портфеля r^*, которое может быть как четким числом, так и нечетким.

Для того, чтобы определить структуру портфеля, который обеспечит максимальную доходность при заданном уровне риска, требуется решить следующую задачу оптимизации:

$$\tilde{r} = \sum_{i=1}^{N} x_i \tilde{r}_i \to \max \qquad (1)$$

$$\beta = const \qquad (2)$$

$$\sum_{i=1}^{N} x_i = 1 \; x_i \geq 0 \; i = \overline{1, N} \qquad (3)$$

$$0 < \beta < 1$$

где β- уровень риска.

В работе [1] было показано, что этот случай возможен когда

$$r_{\min} = \sum_{i=1}^{N} x_i r_{i1} \leq r^* < \sum_{i=1}^{N} x_i \tilde{r}_i$$

либо когда

$$\sum_{i=1}^{N} x_i \tilde{r}_i \leq r^* < \sum_{i=1}^{N} x_i r_{i2} = r_{\max}$$

Тогда используя результаты работы [3], задача (1)-(3) сводится к следующей задаче нелинейного программирования:

$$\tilde{r} = \sum_{i=1}^{N} x_i \tilde{r}_i \to \max \qquad (4)$$

Рассмотрим случай, когда критериальное значение доходности r^* удовлетворяет условиям

$$\sum_{i=1}^{N} x_i r_{i1} \le r^* \le \sum_{i=1}^{N} x_i \tilde{r}_i = \tilde{r} \quad (5)$$

Тогда величина риска равна [1,2]

$$\beta(x) = \frac{1}{\sum_{i=1}^{N} x_i r_{i2} - \sum_{i=1}^{N} x_i r_{i1}} [(r^* - \\ - \sum_{i=1}^{N} x_i r_{i1}) + (\sum_{i=1}^{N} x_i \tilde{r}_i - \\ -r*)\ln(i=1Nxiri-r*i=1Nxiri-i=1Nxiri1)] \quad (6)$$

Требуется доказать, что функция риска $\beta(x)$ является монотонно убывающей, где

$$\beta(x) = \left(A(x) + B(x)\ln\frac{B(x)}{C(x)}\right) D(x),$$

$$A(x) = r^* - \sum_{i=1}^{N} x_i r_{i1};$$

$$B(x) = \sum_{i=1}^{N} x_i \tilde{r}_i - r^*;$$

$$C(x) = \sum_{i=1}^{N} x_i \tilde{r}_i - \sum_{i=1}^{N} x_i r_{i1},$$

$$D(x) = \frac{1}{\sum_{i=1}^{N} x_i r_{i2} - \sum_{i=1}^{N} x_i r_{i1}}.$$

Заметим, что функция $A(x)$ – линейна и поэтому не строго выпукла, а функции $B(x)$ и $C(x)$также линейны.

Кроме того

$r_i \ge r_{i1}, i = \overline{1,N}$, и $\sum_{i=1}^{N} x_i \tilde{r}_i - r^* > 0$ по предположению (условие (5).

Рассмотрим функцию $D(x)$. Найдем

$$\frac{\partial D(x)}{\partial x_i} = D'(x) = -\frac{r_{i2} - r_{i1}}{(\sum_{i=1}^{N} x_i (r_{i2} - r_{i1}))^2} < 0 \quad (7)$$

Для удобства обозначим

$$A(x) + B(x)\ln\frac{B(x)}{C(x)} = \varphi(x). \quad (8)$$

Докажем, что $\frac{\partial \varphi}{\partial x_i} = \varphi'(x) < 0$.

Рассмотрим

$$\frac{\partial \varphi}{\partial x_i} = \frac{\partial}{\partial x_i}\left(A(x) + B(x)\ln\frac{B(x)}{C(x)}\right) = A'(x) + \\ + B'(x)\ln\frac{B(x)}{C(x)} + B(x)\frac{C(x)}{B(x)}\frac{B'(x)C(x) - C'(x)B(x)}{C^2(x)} = \\ = A'(x) + B'(x)\,ln\frac{B(x)}{C(x)} + B'(x) - C'(x)\frac{B(x)}{C(x)} \quad (9)$$

Подставив значения $A'(x)$и $B'(x)$ в (9), получим:

$$\frac{\partial \varphi}{\partial x_i} = -r_{i1} + \tilde{r}_i \ln\frac{B(x)}{C(x)} + \tilde{r}_i - (\tilde{r}_i - r_{i1})\frac{B(x)}{C(x)} = \\ \tilde{r}_i\left(1 + \ln\frac{B(x)}{C(x)}\right) - r_{i1} - -(\tilde{r}_i - r_{i1})\frac{B(x)}{C(x)} \quad (10)$$

Поскольку $\frac{B(x)}{C(x)} < 1$, то $\frac{B(x)}{C(x)} < 1$.

Отсюда, после упрощения (10), мы получим:

$$\frac{\partial \varphi}{\partial x_i} < \tilde{r}_i\left(1 + \ln\left(\frac{B(x)}{C(x)}\right) - \frac{B(x)}{C(x)}\right), \quad (11)$$

$$1 + \ln\left(\frac{B(x)}{C(x)}\right) - \frac{B(x)}{C(x)} = 1 + \ln\left(\frac{\tilde{r} - r^*}{\tilde{r} - r_{min}}\right) - \frac{\tilde{r} - r^*}{\tilde{r} - r_{min}} \quad (12)$$

Заметим, что $r^* > r_{min} = \sum_{i=1}^{N} x_i r_{i1}$ и $\tilde{r} > r^*$. Покажем, что выражение (12) меньше 0. Обозначим $\tilde{r} - r^* = a$, тогда:

$\tilde{r} - r_{min} = \tilde{r} - r^* + r^* - r_{min} = a + y$,

где $y = r^* - r_{min} > 0$.

Подставляя эти обозначения в (12) , получим:

$$1 + \ln\left(\frac{\tilde{r} - r^*}{\tilde{r} - r_{min}}\right) - \frac{\tilde{r} - r^*}{\tilde{r} - r_{min}} = 1 + \ln\left(\frac{a}{a+y}\right) - \frac{a}{a+y} \quad (13)$$

Покажем, что $\Delta = 1 + \ln\left(\frac{a}{a+y}\right) - \frac{a}{a+y} < 0$ для всех $y > 0$.

Очевидно, что $\Delta = 1 + \ln\left(\frac{a}{a+y}\right) - \frac{a}{a+y} = 0$ при $y = 0$, и кроме того, функция $\Delta(y)$ монотонно убывающая, т.к.

$$\Delta'(x) = -\frac{1}{a+y} + \frac{a}{(a+y)^2} = -\frac{y}{(a+y)^2} < 0 \quad (14)$$

для всех $y > 0$.

Таким образом $\Delta(y) < 0$, для всех $y > 0$, и окончательно $\frac{\partial \varphi(x)}{\partial x_i} < 0$.

Вычислим первые производные:

$$\frac{\partial \beta}{\partial x_i} = \frac{\partial}{\partial x_i}\left(\varphi(x)D(x)\right) = \varphi'(x)D(x) + + D'(x)\varphi(x)$$

И т.к. $\varphi'(x) < 0$, $D'(x) < 0$, а $\varphi(x) > 0$, $D(x) > 0$, то получим окончательно

$$\frac{\partial \beta(x)}{\partial x_i} < 0 \quad (15)$$

Таким образом, мы получили следующие достаточные условия монотонно убывающего характера зависимости « оптимальная доходность –риск» для задачи оптимизации нечеткого портфеля:

$$\sum_{i=1}^{N} x_i r_{i1} \le r^* \le \sum_{i=1}^{N} x_i \tilde{r}_i = \tilde{r} \quad (16)$$

Пример 1

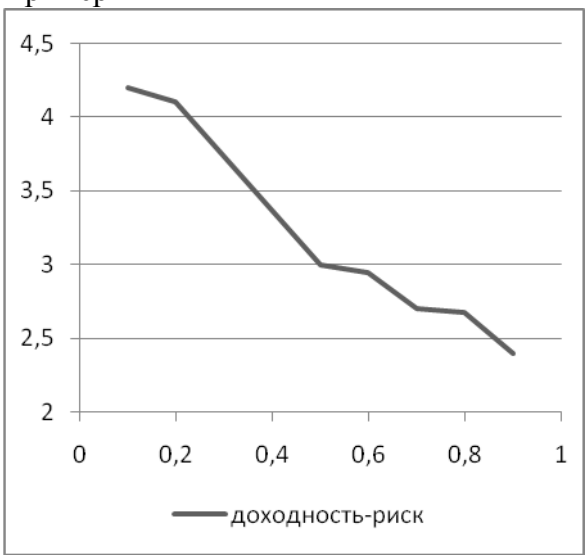


Рис. 1. График зависимости доходность- риск нечеткого портфеля

Пример 1.

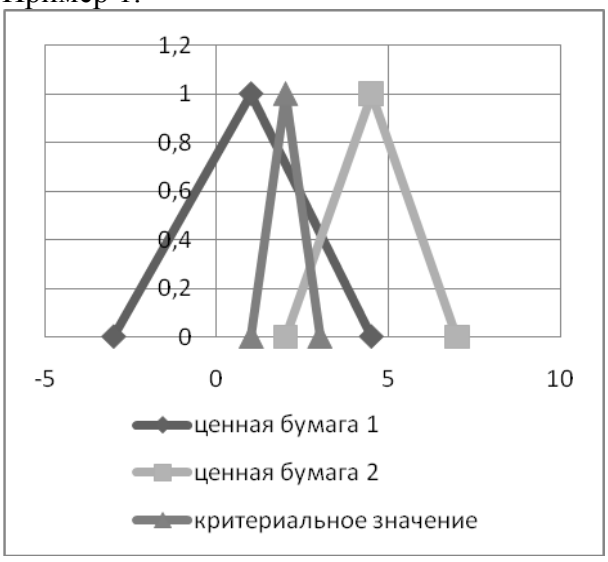


Рис. 2. Взаимное расположение нечетких доходностей ЦБ и критериального значения для треугольных ФП

Пример 2.

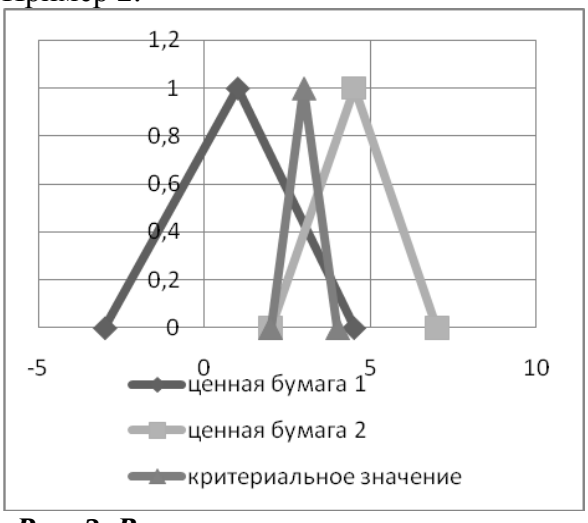


Рис. 3. Взаимное расположение нечетких доходностей ЦБ и критериального значения.

Пример 3.

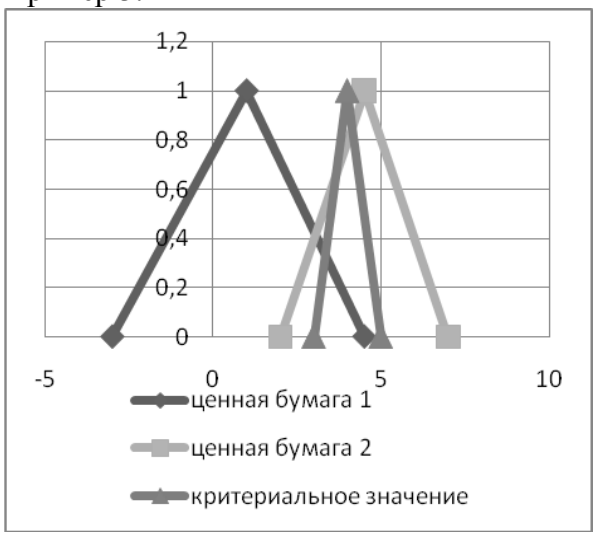


Рис. 4. Взаимное расположение нечетких доходностей ЦБ и критериального значения.

Пример 4.

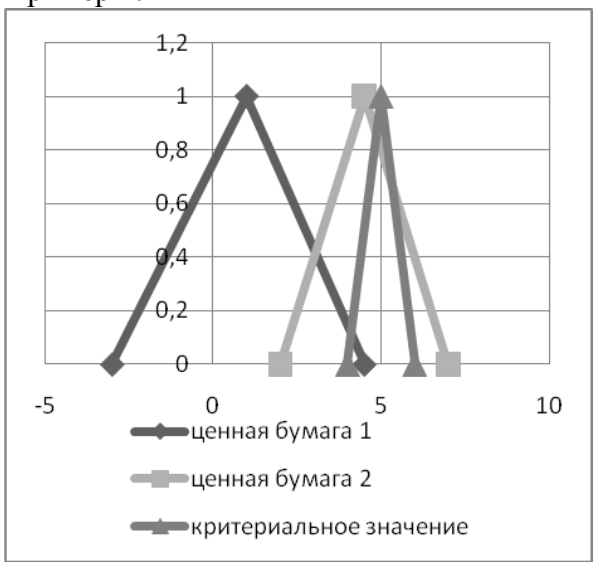


Рис. 5. Взаимное расположение нечетких доходностей ЦБ и критериального значения.

Пример 5.

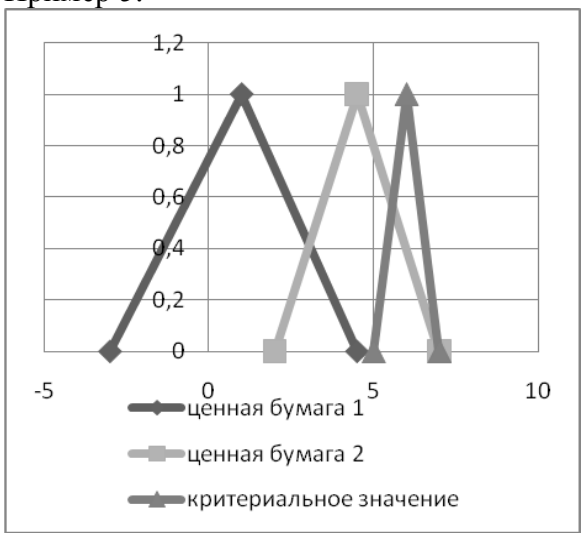


Рис. 6. Взаимное расположение нечетких доходностей ЦБ и критериального значения для треугольных ФП.

Выводы

В работе исследована модель задачи оптимизации нечеткого инвестиционного портфеля. Установлены достаточные условия, при которых зависимость «оптимальная доходность-риск»в задача нечеткой портфельной оптимизации для портфеля с произвольными числом ценных бумаг будет иметь убывающий характер, прямо противоположный зависимости для классической модели Марковица- Тоббина.

Проведены экспериментальные исследования условий и оптимальных решений задачи портфельной оптимизации для убывающей и возрастающей зависимостей «доходность-риск», которые полностью подтвердили теоретические результаты.

Список литературы

1. Зайченко Ю.П., Малихех Есфандиярфард. Анализ инвестиционного портфеля для различных видов функций принадлежности // Системні дослідження та інформаційні технології. - 2008.-№2.- С. 59-76.
2. Зайченко Ю.П., Малихех Есфандиярфард. Нечеткий метод индуктивного моделирования для прогнозирования курсов акций в задачах портфельной оптимизации// Вісник Черкаського державного технологічного університету. – 2008. - № 1.- С. 9-14.
3. Зайченко Ю.П., Малихех Есфандиярфард, Заика А.И. Анализ инвестиционного портфеля на основе прогнозирования курсов акций // Вісник національного технічного університету України «КПІ». «Інформатика, управління та обчислювальнатехніка». – К.: ТОО «ВЕК+», 2007. - № 47. - С. 168-179.
4. Зайченко Ю.П.,Малихех Есфандиярфард, Ови Нафас Агаи Аг Гамиш. Анализ зависимости «доходность- риск» в задачах портфельной оптимизации в нечетких условиях// Вісник національного технічного університету України «КПІ». «Інформатика, управління та обчислювальна техніка». – К.: ТОО «ВЕК+», 2010. - № 51. - С. 168-179.

УДК 004.023

БОЛДАК А.А.,
СУХАРЕВ Д.Л.

ОПРЕДЕЛЕНИЕ КОЛИЧЕСТВА КЛАСТЕРОВ В СТАТИСТИЧЕСКИХ ДАННЫХ

В работе разработаны и экспериментально изучены три эвристических алгоритма автоматического определения количества кластеров выборок данных, использование которых позволяет усовершенствовать известные алгоритмы кластеризации данных. Особенностью предложенных алгоритмов является отсутствие необходимости многократного решения задачи кластеризации для разного количества кластеров, с последующим анализом качества кластерной структуры.

In the work three heuristic algorithms for automatic determination of the clusters amount of data samples were developed and experimentally studied. Usage of them can improve well-known clustering algorithms. Feature of the proposed algorithms is that there is no need to execute multiple calculations of clustering task for different clusters amount, with subsequent analysis of cluster structure quality.

Введение

Алгоритмы кластеризации являются одним из инструментов, которые с успехом используются для решения задач, связанных с интеллектуальным анализом данных (Data Mining) [1], обработкой изображений [2], рас познаванием образов [2], группированием результатов поиска [3], сжатием данных [3] и т.п. Особенность известных методов иерархической и разбивающей кластеризации [2] состоит в том, что для них количество кластеров является параметром, автоматическое определение которого может рассматриваться как самостоятельная задача. Известные алгоритмы её решения (Gap подход [4], G-Means [5], X-Means [6]) предполагают повторное выполнение кластеризации с последующей оценкой качества полученной кластерной структуры. Очевидно, что применение таких методов требует значительных вычислительных затрат для решения этой задачи.

Цель работы состоит в снижении вычислительных затрат, необходимых для автоматического определения количества кластеров за счет разработки и использования алгоритмов решения этой задачи, отличительной особенностью которых является отсутствие необходимости повторного выполнения кластеризации данных.

1. Постановка задачи

Постановка задачи кластеризации связана с определением метрического пространства

$$(O, d : O \times O \to R),$$

где d – метрика, O – множество объектов кластеризации, $O = \{O_i\},\ i = \overline{1, |O|}$.

Кластеризация объектов множества O представляет собой отображение вида:

$$f : O_i \to C_j, i = \overline{1, |O|}, j = \overline{1, k},$$

где C – множество кластеров, k – их количество, $k = \overline{1, |O|}$.

Для каждого кластера C_j с множеством элементов $O_j \subseteq O$ можно определить центроид, т.е. такой объект c_j, для которого выполняется условие:

$$c_j : \sum_{i=1}^{|O|} d(c_j, O_{ji})^2 \to \min.$$

Как правило, формальная кластеризация предполагает, что сведения обо всех объектах множества O заданы с помощью количественных оценок их свойств (показателей) в виде таблицы «объект-свойство» [7]:

$$X = (x_{ij})_{i=1, j=1}^{|O|, m}, x_{ij} \in R,$$

в которой строка X_i соответствует множеству значений показателей для объекта O_i, столбец X^j – множеству значений j-го показателя для всех объектов множества O, который рассматривается как статистическая переменная [7].

В этом случае метрика d выражается как метрика в пространстве R^m и интерпретируется как оценка близости объектов в R^m. В один кластер включаются близкие друг к другу по метрике d объекты.

Пусть качество результата работы алгоритма кластеризации f оценивается количественно с помощью критерия $K(O, k)$, O – множество объектов, k – количество кластеров. Тогда задача определения параметра k может быть поставле-

на как задача поиска аргумента максимизации (argmax) или минимизации (argmin) этого критерия:

$$k = \arg\max(K(X,k)),\ k = \overline{1,|O|},$$
$$k = \arg\min(K(X,k)),\ k = \overline{1,|O|}.$$

Вид и содержание критерия *K*, и связанная с ним задача оптимизации определяются семантикой конкретного прикладного применения метода кластеризации.

2. Эвристические алгоритмы определения количества кластеров

Предлагаемые алгоритмы определения количества кластеров основаны на предположении, что кластерная структура данных характеризуется неравномерностью плотности распределения значений X^j.

Функция плотности распределения значений статистической переменной $p(X^j,n)$, где n – количество уровней дискретизации, соответствующих интервалам группирования значений этой переменной [8], задаёт вероятность событий вида $X^j = x_{ij}$, $i = \overline{1,n}$. Очевидно, что равномерное распределение значений статистической переменной (X_U^j) соответствует отсутствию кластерной структуры в данных. Это означает, что степень ее проявления можно оценить с помощью количественной характеристики отклонения функции плотности распределения значений X^j от функции плотности равномерного распределения X_U^j в области значений X^j.

Таким образом, разработка алгоритма определения количества кластеров для одной статистической переменной требует определения меры μ в пространстве функций плотности распределения вероятностей и такого количества интервалов группирования данных, на котором расстояние между $p(X^j,n)$ и $p(X_U^j,n)$, оцененное по мере μ, будет максимальным.

Информационная энтропия. Для дискретной случайной величины *X*, принимающей значения $\{x_i : i = \overline{1,n}\}$, шенноновская энтропия, мера неопределенности, представляется как [9]:

$$H(X^j) = -\sum_{i=1}^{n} p(x_i)\log_b p(x_i).$$

Для существования кластерной структуры вероятностное распределение данных X^j должно отличаться от равномерного. Поэтому в качестве меры μ можно использовать понятие эффективности алфавита [10]:

$$E(X^j) = -\sum_{i=1}^{n} \frac{p(x_i)\log_b p(x_i)}{\log_b(n)}, \qquad (1)$$

где числителем является энтропия для X^j, знаменателем – энтропия для X_U^j. Это значение всегда меньше или равно единицы. Чем оно меньше, тем более предсказуемым является исходное распределение по отношению к равномерному, и тем легче выделяются интервалы с большей вероятностью попадания в них. В случае кластеризации такие интервалы будут соответствовать кластерам, то есть интервалы группирования можно рассматривать как алфавит мощности *n*.

При решении задачи кластеризации в правой части формулы (1) известны значения всех операндов, кроме *n*. Тогда формула (1) может рассматриваться как выражение для вычисления критерия $K(X^j,n)$, а задача поиска *n* – как

$$n = \arg\min(K(X^j,n)).$$

Если начальным количеством интервалов группирования есть значение $n=N_{max}$, являющееся степенью двойки

$$N_{\max} = 2^p,\ p \in N,$$

то попарное суммирование значений $p(x_i),\ i = \overline{1,N_{\max}}$ даёт значения $p(x_i),\ i = \overline{1,N_{\max}/2}$ для $N_{\max}/2$ интервалов и т.д. Это позволит уменьшить число обращений к исследуемым объектам.

Дивергенция Кульбака-Лейблера. Альтернативным методом определения количества интервалов группирования является дивергенция Кульбака-Лейблера [11][12][13], известная также как относительная энтропия:

$$D_{KL}(P \| Q) = \sum_{i=1}^{n} P(x_i)\log\frac{P(x_i)}{Q(x_i)}.$$

В случае кластеризации наибольший интерес она представляет как расстояние между выборками *P* и *Q*. Однако она не является полноценной метрикой т.к. не удовлетворяет условию симметричности. Поэтому используют её симметризированную форму – дивергенцию Дженсона-Шеннона (JSD) [14]:

$$D_{JS}(P \| Q) = \frac{1}{2} D_{KL}(P \| M) + \frac{1}{2} D_{KL}(Q \| M), \quad (2)$$

где M – средняя плотность распределения для P и Q:

$$M = \frac{1}{2}(P + Q). \quad (3)$$

Пусть в формулах (2),(3) P – распределение исходных данных X^j, Q – равномерное распределение X_U^j. Тогда задача нахождения интервалов группирования может быть поставлена как

$$n = \arg\max(K(P,Q,n)), \; n = \overline{1, N_{\max}},$$

но дивергенция здесь не может использоваться как критерий K, т.к. возрастает с ростом количества интервалов группирования. Однако она поддается аппроксимации функцией $f(n) = a \log n$. Тогда критерий K имеет следующий вид:

$$K(P,Q,n) = D_{JS}(P \| Q) - a \log(n).$$

Для уменьшения количества вычислений используются те же оптимизации, что и в предыдущем алгоритме, основанном на энтропии.

Деление интервалов группирования до образования локального минимума (ДИГОЛМ). С одной стороны, поиск центроидов связан с нахождением плотных интервалов группирования, в которых локализованы эти центроиды. С другой стороны, поиск границ связан с нахождением разреженных интервалов. Таким образом, задача связана с поиском экстремумов: локальных минимумов x_{min} и максимумов x_{max} функции вероятности $p(x_i)$:

$$\begin{aligned} &p(x_{\min}) \le p(x_i) \; при \; |i - \min| \le \varepsilon, \; i = \overline{1,n}, \\ &p(x_{\max}) \ge p(x_i) \; при \; |i - \max| \le \varepsilon, \; i = \overline{1,n}, \end{aligned} \quad (4)$$

где ε – единица, т.к. номера интервалов группирования являются дискретными величинами.

При этом, также как и в предыдущем случае, значение n выбирается среди степеней двойки. Это позволяет сократить объем необходимых вычислений для решения задачи (4).

Поиск центроидов. Задача поиска кластеров может решаться, как задача локализации плотных областей в многопараметрическом пространстве. Центры этих областей соответствуют центроидам предполагаемых кластеров.

Когда данные по каждому показателю сгруппированы в интервалы и при этом наблюдается неравномерность распределения плотности, можно построить *многомерные интервалы группирования (МИГи)*.

Каждому МИГу соответствует один интервал группирования для каждого показателя. Тогда количество МИГов N_B можно определить из формулы:

$$N_B = \prod_{i=1}^{n} l_i,$$

где l_i – количество интервалов группирования i-го показателя, n – размерность пространства.

Затем рассчитывается функция плотности объектов, попадающих в каждый МИГ. Объект x_i считается принадлежащим МИГу когда он попадает во все интервалы I_i, включенные в этот МИГ, то есть

$$\bigcap_{i=1}^{n} (x_i \in I_i) = 1,$$

где n – размерность пространства.

Разбиение пространства на многомерные интервалы группирования и оценка их плотности позволяет определить приблизительное положение центроидов, которые могут быть использованы как начальные центроиды алгоритма k-means. Это дает возможность уменьшить количество итераций этого алгоритма.

Для определения центроидов вычисляется ожидаемое значение функций плотности МИГов

$$M_x = \frac{1}{N_B} \sum_{i=1}^{N_B} p_i.$$

Если значение функции плотности МИГа превышает M_x ($p_j > M_x$), то центр этого МИГа признается центроидом. Количество кластеров определяется количеством центроидов.

Оценка эффективности. В разработанных алгоритмах кластеризации данных, которые автоматически определяют количество кластеров, применяются эвристики поиска количества интервалов группирования, поиска центроидов, а также методы кластеризации. К реализованным алгоритмам поиска количества интервалов группирования относятся эффективность алфавита теории энтропии, дивергенция Дженсена-Шеннона и ДИГОЛМ. Алгоритмы поиска центроидов делятся на аналитические и случайные. В качестве алгоритма кластеризации используется метод k-means. Таким образом, было разработано 6 алгоритмов кластеризации, способных автоматически определять количество кластеров.

Были созданы средства генерации экспериментальных выборок данных, которые возвращают выборки разного вида, в зависимости от

количества кластеров, их плотности и разделенности.

Для статистической достоверности оценка эффективности алгоритмов кластеризации была выполнена для 4000 выборок данных разного вида. В таблице 1 приведены усредненные результаты этих оценок.

Для оценки качества кластерной структуры используются индекс Данна [15] и среднее значение силуэта (average silhouette) [16]. Они имеют достаточно высокую точность и напрямую не зависят от количества кластеров. Кроме того, среднее значение силуэта является нормированной величиной и устойчиво к вбросам. Для этих критериев большее значение свидетельствует о более высоком качестве кластерной структуры.

Оценка производительности алгоритмов кластеризации выполняется на основе количества итераций и длительности их выполнения в миллисекундах. Измерения проводились на ПК с процессором Intel Core 2 Duo E8500 под управлением ОС Ubuntu 10.04 x64.

Табл. 1. Усредненные результаты измерений производительности применения различных эвристик

Определение кол-ва интервалов	Определение начальных центроидов	Кол-во итераций	Длительность, мс	Кол-во кластеров	Індекс Данна	Среднее значение силуэта
Энтропия	Аналитическое	6.696	**302.05**	11.918	**2.307**	0.750
ДИГОЛМ	Аналитическое	**6.632**	316.52	6.210	1.639	0.697
JSD	Аналитическое	6.736	337.74	12	2.297	**0.763**
Энтропия	Случайное	27.473	1048.5	11.918	1.335	0.297
ДИГОЛМ	Случайное	11.775	447.82	6.215	1.375	0.626
JSD	Случайное	27.117	1079.5	12	1.361	0.338

В соответствии с результатами, приведенными в таблице 1, можно сделать вывод, что в общем случае алгоритм на основе метода ДИГОЛМ имеет наименьшее среднее количество итераций, алгоритм на основе метода энтропии – наименьшее время выполнения и наибольший индекс Данна, а алгоритм на основе JSD – наибольшее значение среднего силуэта. Среднее значение количества кластеров, реально возвращаемых генераторами экспериментальных выборок данных, равнялось 13. Самые близкие к этому значению результаты показали алгоритмы, использующие методы энтропии и JSD. Алгоритмы на основе метода ДИГОЛМ плохо различают большое количество (больше шести) кластеров. Алгоритмы со случайным определением начальных центроидов значительно уступают тройке алгоритмов с их аналитическим определением.

Заключение

Результатом работы являются три эвристических алгоритма автоматического определения количества кластеров, использование которых позволяет усовершенствовать известные алгоритмы кластеризации данных, в частности алгоритм k-средних и его модификации. Эти усовершенствования касаются как качества кластерной структуры, так и времени выполнения.

Основой разработанных алгоритмов есть то, что наличие кластерной структуры может количественно оцениваться отклонениями функции плотности распределения статистической переменной от функции плотности равномерного распределения. Для оценки этих отклонений используются эффективность алфавита теории энтропии, дивергенция Дженсена-Шенонна и ДИГОЛМ. Т.к. они оперируют с функциями плотности, которые являются дискретными и могут быть получены за один проход данных, то такие алгоритмы имеют линейную вычислительную сложность.

В результате анализа экспериментальных данных, полученных с использованием разработанных программных средств, можно сделать вывод, что в общем случае, когда вид выборок данных неизвестен, алгоритмы с использованием JSD и эффективности алфавита теории энтропии дают наилучшие результаты.

Использование описанных алгоритмов позволяет значительно снизить вычислительные затраты, необходимые для автоматического определения количества кластеров сравнительно с существующими алгоритмами, требующими множественного выполнения кластеризации данных.

Список литературы

1. Usama F., Piatetsky-Shapiro G., Smyth P. (1996). From Data Mining to Knowledge Discovery in Databases // AI MAGAZINE. – 1996. – pp. 37-54.
2. Jain A.K. and Dubes R.C. Algorithms for Clustering Data // Prentice Hall. – 1988. – 320 c.
3. Tan P.N., Steinbach M., Kumar V. Introduction to Data Mining // Addison-Wesley. – 2005. – 769 c.
4. Tibshirani R., Walther G., Hastie T. Estimating the number of clusters in a dataset via the Gap statistic. // Technical Report. Stanford. – 2000. – pp. 412-423.
5. Hamerly G., Elkan C. Learning the k in k-means. // NIPS. – 2003.
6. Pelleg D., Moore A. X-means: Extending K-means with Efficient Estimation of the Number of Clusters. // Morgan Kaufmann. – 2000.
7. Айвазян С.А., Енюков И.С., Мешалкин Л.Д. Прикладная статистика. Исследование зависимостей. М.: Финансы и статистика. 1985. – 480 с.
8. Кудрявцев Л. Д. Курс математического анализа. — 5-е изд. — М.: «Дрофа», 2003. — Т. 1. — 704 с.
9. Shannon C.E. A Mathematical Theory of Communication // Bell System Tech. – 1948. – vol. 27, pp. 379-423, 623-656.
10. Potamites P. Selective Pressures on Symbolic Systems. [Електроний ресурс] – January 28, 2010. – Режим доступу: http://www-scf.usc.edu/~potamite/philqual2.pdf
11. Kullback S., Leibler R.A. On Information and Sufficiency. Annals of Mathematical Statistics. – 1951. – 22 (1): pp. 79–86.
12. Kullback S. Information theory and statistics. // John Wiley and Sons, NY. – 1959. – 416 c.
13. Kullback S. Letter to the Editor: The Kullback-Leibler distance // The American Statistician. – 1987. – 41(4): pp. 340–341.
14. Fuglede B., Topsoe F. Jensen-Shannon divergence and hilbert space embedding. // University of Copenhagen, Department of Mathematics. – 2004.
15. Dunn J. Well separated clusters and optimal fuzzy partitions // Journal of Cybernetics. – 1974. – 4, pp. 95-104.
16. Rousseeuw P.J. Silhouettes: a graphical aid to the interpretation and validation of cluster analysis // Journal of Computational and Applied Mathematics. – 1987. – 20, pp. 53-65.

УДК 004.9

МУРГА М.О.

МОДИФІКАЦІЯ МЕТОДУ НАВЧАННЯ РАДІАЛЬНОЇ БАЗИСНОЇ МЕРЕЖІ ТА ЇЇ ПОРІВНЯННЯ З НЕЧІТКОЮ НЕЙРОННОЮ МЕРЕЖЕЮ МАМДАНІ ТА ННМАННКП

В статті представлена модифікація методу навчання нейронної мережі на радіальних базисних функціях та досліджено властивості модифікованого методу. Проводиться порівняльний аналіз радиальної базисної мережі, навченої за наведеною модифікацією, та нейронної мережі Мамдані, і мережі ННМАННКП на значеннях денних котирувань фінансових інструментах з фінансових ринків різного типу й на денних значеннях світових фінансових індексів.

Radial basis function network training method modification is presented in this paper and features of the modified method are investigated. Comparative analysis of radial basis network, trained with the modification, Mamdani neural network and FNNAITER is conducted on values of financial instrument daily quotes and on world financial indexes daily values.

Вступ

В даній роботі запропоновано модифікацію методу навчання нейронної мережі на радіальних базисних функціях ([1, С. 57-60, С. 231-241; 2, С. 342-416; 3]). Ця модифікація полягає у тому, що поряд з навчанням ваг радіальної базисної мережі, відбувається паралельне навчання параметру σ нейронів мережі. Метод навчання дозволяє налаштувати мережу за кінцеву кількість кроків. В роботах [1, С. 231-241; 2 С. 239-403; 3] вже були запропоновані методи навчання всіх параметрів мережі, однак це були – генетичні алгоритми та градієнтний метод, що збігаються до прийнятного результату дуже довго, а радіальна базисна мережа відома своєю швидкістю навчання. В роботі запропоновано метод оцінки вищенаведених параметрів, що застосовує МНК для встановлення залежності виходу блоку від одного входу. Це дозволяє вирішити проблему швидкого навчання параметру. Даний факт обумовлює актуальність роботи. Мережу було апробовано на денних котируваннях валютних пар EUR/GBP, EUR/USD, USD/JPY, USD/CHF, що були взяті з [4], та на денних значеннях CAC40, DAX, DJIA, HANG SENG, NASDAQ, індекс ММВБ, RTS2 та на денних котируваннях фінансових інструментів COMEX GOLD, ММВБ GSZP, ICE.BRN, котрі були взяті з [5]. На першій групі даних проводився порівняльний аналіз даної радіальної базисної мережі, мережі, запропонованої в [6], та нечіткої нейронної мережі Мамдані ([7 С. 161-166; 8 С. 100-114]). На першій та другій групі – порівняльний аналіз радиальної базисної мережі та нечіткої нейронної мережі Мамдані.

Робота складається з трьох розділів: «Опис модифікацій», де описується, власне, метод навчання, що пропонується; «Дослідження властивостей запропонованої модифікації», де вивчаються характеристики методу, що пропонується; та – «Порівняльний аналіз нейронних мереж», в якому приводяться результати порівняння вищенаведених мереж та аналіз даних результатів.

Цільовою аудиторією статті є аналітики фінансових ринків, спеціалісти в галузі систем штучного інтелекту та спеціалісти в галузі аналізу часових рядів.

1. Опис модифікації

Радиальна базисна мережа – це нечітка нейронна мережа, з класу мереж, що вико ристовують нечіткий логічний вивід Такагі-Сугено. Її архітектуру можна побачити на Рис. 1.

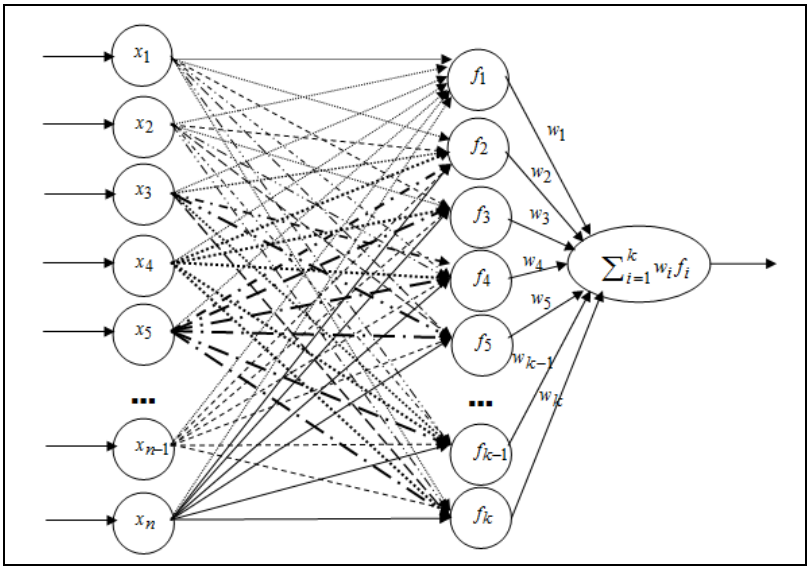


Рис. 1. Архітектура радиальної базисної нейронної мережі

На Рис.1 використані наступні позначення. $x_1,...,x_n$ – це входи нейронної мережі. $f_i, i=1,...,k$ – радиальні базисні функції, значення котрих залежать від входів нейронної мережі. В роботі дані функції взяті гаусівського вигляду (1):

$$f_i(\vec{x}) = \exp\left\{-\frac{\left\|\vec{x}-\vec{c_i}\right\|^2}{2\sigma_i^2}\right\}. \qquad \textbf{(1)}$$

У формулі (1) використані наступні позначення. $\vec{x}=(x_1,...,x_n)$ – вектор аргументів функції (входів нейронної мережі). $\vec{c_i}=(c_{1i},...,c_{ni})$ – «центр» радиальної базисної функції, та σ_i – «розкид» радиальної базисної функції.

Значення входів мережі подаються до блоків розрахунку радіальних базисних функцій. Отримані на виходах значення множаться на відповідні ваги та додаються один до одного. Так отримується вихід радиальної базисної мережі.

В якості «центрів» радіальних базисних функцій прийнято вибирати вхідні значення навчальної вибірки даних, а «розкиди» вибираються експертом рівними один одному ([1, С.57-60; 2, С. 396-398]).

Якщо позначити вихід нейронної мережі через y, то він визначається формулою (2):

$$y = \sum_{i=1}^{k} w_i f_i\left(\vec{x}\right). \qquad \textbf{(2)}$$

Нехай є навчальна вибірка даних $\left(x_{1j}, x_{2j},...,x_{nj}, y_j\right), j=1,...,L$. Якщо ввести матрично-векторні позначення $\vec{y}=(y_1,...,y_L)$ – вектор бажаних виходів мережі; $F=\left\| f_i\left(x_{1j},...,x_{nj}\right)\right\|_{i=1,...k}^{j=1,...,L}$ – матриця відповідних виходів блоків радіальних базисних функцій; $\vec{w}=(w_1,...,w_k)$ – відповідний вектор ваг, то формулу (2) можна переписати у вигляді (3):

$$\vec{y} = F\vec{w} \qquad \textbf{(3)}$$

Так як можна вибрати $L=k$, то в такому випадку F – квадратна, та за теоремою Мічеллі, до неї завжди буде існувати обернена матриця ([2, С.352-353]). Якщо зафіксувати σ_i, то задача навчання нейронної мережі зводиться до обертання матриці F та множення її та вектор $\vec{y}$. Саме так пропонується робити у роботах ([1, С.57-60; 2, С. 396-398].

«Розкиди» зазвичай вибираються рівними один одному. Однак відразу ж помітна вада такого підходу – вибір «розкидів» залежить від конкретної задачі. Окрім того обертання матриці пов’язано зі значними обчислювальними витратами. Для розв’язання цієї проблеми у роботі [1, С. 231-241] для навчання мережі пропонується використовувати генетичний алгоритм. Це – вирішення проблеми, але воно зводить нанівець основну перевагу ради-

альної базисної мережі – висока швидкість навчання.

По-перше, навчання ваг. Пропонується зафіксувати значення «розкидів» для ясності викладень (далі це обмеження буде усунуто), при йому навчання зводиться до застосування методу найменших квадратів ([9, С.49-57]) для відновлення лінійної залежності від одної змінної. В даній роботі буде використано саме цей метод навчання ваг радиальної базисної мережі, а також буде запропоновано метод навчання значень «розкидів», який теж зводиться до застосування МНК для відновлення залежності від однієї змінної.

У подальших викладеннях буде використовуватися позначення (4):

$$k_i = \frac{1}{2\sigma_i^2} \qquad \textbf{(4)}$$

Далі у роботі k_i з формули (4) буде називатися коефіцієнтом звуження.

Отже у роботі пропонується з вибірки даних вибрати дві точки, що розташовані якнайдалі одна від одної за Евклідовою метрикою. Ці дві точки беруться за центри двох радіальних базисних функцій, які утворюють нову мережу. Значення коефіцієнтів звуження для гаусівських функцій вибираються рівними та, за допомогою підходу з використанням теореми Мічеллі, відновлюються обидва ваги радиальної базисної функції. Далі всі виміри навчальної вибірки даних пропускаються через отриману нейронну мережу. Якщо загальна помилка роботи мережі виявилася меншою за допустиме значення, то синтез та навчання мережі завершується, інакше – наступний крок. На кожному кроці значення отриманих раніше ваг не змінюються, а кожна нова вага – мінімізує похибку, що видає мережа на попередньому рівні. Далі пояснюється зміст цього речення.

Нехай вже відновлено p ваг. З вимірів вибірки даних, що ще не брали участь у синтезі мережі та навчанні ваг, вибирається той вимір, що дає найбільшу похибку (ця методика була запропонована у роботі [10, С. (13-177) – (13-180)]). Вхідні значення вибраного виміру ініціюють центр нового нейрона радиальної базисної мережі (нового блоку розрахунку значень радиальної базисної функції). Коефіцієнт звуження вибирається рівним, до коефіцієнтів інших нейронів (далі в роботі це обмеження буде усунуто). Вся вхідна частина вимірів навчальної вибірки даних, що вже брали участь у синтезі та навчанні мережі, та новий вимір, пропускаються через мережу, що була синтезована та навчена на попередньому кроці та отримується $p+1$ відхилень виходів мережі від бажаних значень виходів:

$$e_1 = y_1 - d_1, ..., e_{p+1} = y_{p+1} - d_{p+1}. \qquad \textbf{(5)}$$

В позначеннях (5): e_i – відхилення отриманого значення виходу нейронної мережі – y_i від реального значення – d_i.

Мета навчання – знайти таке w_{p+1}, що виконується загалом несумісна система рівнянь (6):

$$f_{p+1}\left(\overrightarrow{x_i}\right) w_{p+1} = -e_i, i = 1, ..., p+1. \qquad \textbf{(6)}$$

Слід ще раз зазначити, що індекс i показує не номер виміру в навчальній вибірці даних, а номер нейрону радиальної базисної мережі та $\overrightarrow{x_i}$ за домовленістю – це центр відповідного нейрона.

Застосовуючи метод найменших квадратів, можна отримати формулу:

$$w_{p+1} = -\frac{\sum_{i=1}^{p+1} f_{p+1}\left(\overrightarrow{x_i}\right) e_i}{\left(f_{p+1}\left(\overrightarrow{x_i}\right)\right)^2}. \qquad \textbf{(7)}$$

Виміри вибірки даних, що не брали участь у синтезі та навчанні «пропускаються» через мережу. Якщо загальна помилка виявляється меншою за задане допустиме значення, то процес синтезу мережі завершується, інакше – на наступний крок.

Розглянутий вище алгоритм навчання нейронної мережі дозволяє швидко навчити нейронну мережу, не витрачаючи час на обертання великих матриць та дозволяє відмовитися від методів навчання, що забезпечують збіжність процесу навчання за нескінченну кількість ітерацій навчання.

Але у застосованого методу залишається одна вада – задана однаковість значень коефіцієнтів звуження.

Далі пропонується наступна ідея для розв'язання цієї проблеми. По перше, очевидно, що загальний вихід радиальної базисної нейронної мережі має незначний вплив на зна-

чення коефіцієнтів звуження (як буде показано далі в експериментальних дослідженнях, коефіцієнт звуження же сильно впливає на значення загального виходу мережі). Таким чином, і значення ваг w_i можна не брати до уваги при навчанні k_i. Також, можна сказати, що частина радиальної базисної мережі до множення на ваги (див. Рис. 1) виконує класифікацію вектора поданого на вхід мережі. Таким чином, в ідеалі хотілося б, щоб при подачі на вхід певного вектора з навчальної вибірки даних, нейрон, що відповідав цьому входу давав найбільше значення – 1, а інші нейрони – інше значення рівне для всіх, скажімо, ev. З урахуванням формул (1), (4) та приведених в даному абзаці міркувань, можна скласти систему рівнянь (8):

$$\exp\left\{-k_i \cdot \left\|x_i - c_j\right\|^2\right\} = \begin{cases} 1, i = j, \\ ev, i \neq j. \end{cases} \quad \textbf{(8)}$$

В даній системі i та j змінюються від одиниці до значення кількості нейронів в нейронній мережі.

Якщо тепер ввести позначення $h_{ij} = \left\|x_i - c_j\right\|^2$, прологарифмувати за натуральною основою кожне рівняння в системі (8) та виключити з системи тотожні рівняння, то можна отримати систему (9):

$$-k_i h_{ij} = \ln ev, i \neq j. \quad \textbf{(9)}$$

Значення коефіцієнтів звуження можна отримати, застосовуючи МНК:

$$k_i = -\frac{\sum_{z=1}^{p} h_{ij} \ln ev}{\left(h_{ij}^2\right)}, \quad \textbf{(10)}$$

де p – кількість нейронів в нейронній мережі.

Якщо усереднити всі коефіцієнти, то можна отримати класичний випадок, коли у всіх нейронів коефіцієнти звуження рівні.

Слід зазначити, що далі в даній роботі буде досліджуватись поведінка критеріїв якості роботи мережі в залежності від кількості нейронів, і тому, при додаванні нової точки вибірки до процесу навчання нейронної мережі буде створюватися ще один нейрон та мережа буде донавчатись за вищенаведеним методом. При практичному застосуванні ж пропонується створювати нейрони вибірково. Якщо додавання нейрону дійсно значно покращує якість функціонування мережі, то необхідно застосовувати наведений вище алгоритм. У іншому випадку (приріст якості функціонування мережі незначний при додаванні нейрона) – або донавчити мережу за допомогою РМНК без створення нового нейрону, або взагалі знехтувати цією інформацією.

2. Дослідження властивостей запропонованої модифікації

В даній частині приводяться результати дослідження залежності якості навчання та функціонування радіальної базисної мережі від конкретного значення величини ev. Слід відразу зазначити, що, якщо не буде зазначено інше, аналіз отриманих у роботі результатів буде проводитися на значеннях критеріїв (які будуть описані нижче) на перевірочній вибірці. В якості даних для дослідження було вибрано денні котирування валютних пар EUR/USD, EUR/GBP, USD/JPY, USD/CHF за річний період, котрі були взяті з джерела [4]; денні значення світових індексів Dow&Jones, NASDAQ, CAC40, DAX, Hang Seng, індекс ММВБ та RTS2, також досліджувалися денні котирування фінансових інструментів COMEX GOLD, ICE Brent, та денні котирування ММВБ «Газпром Нефть», що були взяті з джерела [5]. Дослідження дозволило виявити ряд цікавих результатів, що наводяться нижче.

Перш, ніж переходити до опису власне результатів досліджень, необхідно описати сам експеримент, що проводився. Попередні дослідження показали, що кращих значень показників якості функціонування мережі вдається добитися при співвідношенні навчальної та перевірочної 70%:30%. Саме для цих значень і проводилися дослідження. Параметрами експерименту були: кількість входів мережі (2, 3 та 4) та коефіцієнт ev (коефіцієнт змінювався в межах від 0,1 до 0,9 з кроком 0,1 та з подальшим уточненням). Якість мережі оцінювалася за допомогою значень критерію $MAPE$, що в даній роботі буде визначатися формулою (11):

$$MAPE = \frac{1}{n} \sum_{i=1}^{n} \frac{|y_i - d_i|}{d_i}, \quad \textbf{(11)}$$

де y_i – значення, отримане на виході мережі при подачі на її вхід i-х вхідних вимірів вибірки даних; d_i – бажаний вихід мережі; та n – об'єм вибірки даних.

Вибірки даних формувалися наступним чином. В залежності від кількості входів мережі, що досліджувалась, входами мережі бралися відповідні попередні значення часового ряду досліджуваного фінансового інструменту, а виходом – дане значення. Така процедура проводилася для усього часового ряду. Після виконувалося перемішування порядку замірів вибірки даних та починався експеримент. Така рандомізація вибірки (перемішування порядку вимірів вибірки) повторювалася певну кількість разів та для кожної реалізації перемішування проводився свій експеримент за принципом, описаним в попередньому абзаці. Це дозволило виділити декілька стійких у сенсі малого значення критерію (11) значень показника ev.

Результати проведених експериментів наведені в Табл. 1. Дана таблиця складається з чотирьох стовбців. В першому стовбці наводяться офіціальні позначення фінансових інструментів та індексів, що досліджуються. У другому стовбці наводиться кількість входів мережі, що використовувалася для експерименту на відповідному фінансовому інст.рументі та індексі. Третій стовбець містить відповідні значення досліджуваного параметра ev, що дозволяли отримувати в ході експериментів найкращі значення критеріїв якості, а четвертий – відповідне краще значення критерію $MAPE$, що було отримано в ході експериментів на відповідному фінансовому інструменті чи індексі на перевірочній вибірці.

Табл. 1. Результати досліджень поведінки параметра ev

Фінансовий інструмент чи індекс	**Кількість входів мережі**	**Краще значення *ev*** ***(якщо їх декілька, то розділяються крапкою з комою)***	**Найкращі значення критерію *MAPE*, що вдалося отримати**
CAC40	2	0,995	0,008
	3	0,3; 0,995	0,01
	4		
COMEX GOLD	2	0,3; 0,99<*ev*<1	0,0721
	3		
	4	0,3; 0,995	
DAX	2	0,995	0,0091
	3		
	4		
DJIA	2	0,995	0,008
	3	0,3; 0,995	0,01
	4		
HANG SENG	2	0,3; 0,995	0,0108
	3		
	4	0,995	
ICE.BRN	2	0,3; 0,995	0,0132
	3	0,995	
	4		
GAZP	2	0,09; 0,3; 0,995	0,0231
	3		
	4		
ММВБ	2	0,09; 0,3; 0,995	0,012
	3	0,995	
	4		
NASDAQ	2	0,995	0,0089
	3	0,09; 0,3; 0,995	
	4		
RTS2	2	0,995	0,0087
	3		
	4		
EUR/USD	2	0,995	
	3		
	4		

EUR/GBP	2	0,3; 0,995	0,004
	3		
	4		
USD/JPY	2	0,995	
	3		
	4		
USD/CHF	2		
	3		
	4		

Якщо підсумувати наведені в таблиці дані, то можна прийти до висновку, що на практиці (але слід відмітити, що дослідження проводилися лише для фінансових ринків) краще за все використовувати $ev = 0{,}3$ та $ev = 0{,}995$. Інколи непогані результати давали випадки рівності ev: 0,9; 0,1; 0,2; 0,4, але ці випадки давали досить погані значення критерію *MAPE* та були наслідками невдалої рандомізації. Через те, що $ev = 0{,}995$ частіше за все виявлявся взагалі кращим варіантом, то автор даної роботи буде використовувати саме це значення для подальших досліджень. Окрім того, аналізуючи значення критерію якості (*MAPE*) на перевірочній вибірці, можна прийти до висновку про високі прогностичні властивості мережі. Так, наприклад, найкраще значення критерію для валютних пар було трохи нижче за 0,004, що всередньому свідчить про відхилення реальних значень від оцінок на 40 пунктів, а якщо ще прийняти до уваги, що в реальному застосуванні мережі тільки 50% всередньому чинять негативний вплив, то це значення слід поділити ще на два (якщо похибки в кращу сторону не враховувати), що є досить солідним результатом.

Також проводилися аналогічні дослідження тих же фінансових інструментів, але перетворених за формулою (12):

$$z_i = \frac{x_i - x_{\min}}{x_{\max} - x_{\min}}, \qquad \textbf{(12)}$$

де x_i – значення котирування фінансового інструменту у момент часу i, $x_{\min}$ – мінімальне значення по вибірці фінансового інструменту, $x_{\max}$ – максимальне значення по вибірці фінансового інструменту. Область значень z_i – інтервал $[0;1]$. Очевидно, що внаслідок можливості ділення на 0, застосовувати критерій *MAPE* для оцінки z_i неможливо. Тому для оцінки використовувався критерій *RMSE* (13):

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - d_i)^2}, \qquad \textbf{(13)}$$

де позначення відповідають позначенням у формулі (1).

Дослідження перш за все показали, що якість оцінок зменшилася на порядок (так як z_i змінюється в інтервалі $[0;1]$, то (13) помножене на 100%, як і (1) дають відсоток похибок) – в залежності від конкретного фінансового інструменту, значення критерію *RMSE* змінювались в межах від 0,02 до 0,62 для кращих випадків. Також було встановлено, що у випадку двох входів мережі отримувалися найкращі результати. Було підмічено важливу закономірність. При оцінці (12) досить стабільних котирувань фінансових інструментів найкращі значення критерію *RMSE* лежали стабільно біля $ev = 0{,}9$, а при оцінці нестабільних фінансових інструментів $ev = 0{,}3$ був найкращим вибором. Так що читачу при практичній реалізації мережі для свого фінансового інструменту рекомендується провести RS-аналіз ([11; 12]) даного інструменту.

3. Порівняльний аналіз нейронних мереж

Для оцінки якості функціонування та інших важливих характеристик запропонованої модифікації методу навчання нейронної мережі на радіальних базисних функціях (далі – Радіальна базисна мережа модифікована, або – РБМ) був проведений порівняльний аналіз даної мережі на валютних парах взятих з [4] з мережею, що використовує нечіткий логічний вивід Мамдані ([7 С. 161-166; 8 С. 100-114]) та з мережею, представленою в [6], котра в даній роботі буде називатися Нечітка нейромережева архітектура з незалежним навчанням кожного правила (ННМАННКП). Також був проведений порівняльний аналіз мережі Мамдані та РБМ на значеннях котирувань CAC40,

COMEX GOLD, DAX, DJIA, ММВБ GAZP (нафта), Hang Seng, ICE.BRN, індекс ММВБ, NASDAQ, RTS2. Для навчання мережі Мамдані використовувалася модифікація градієнтного методу – super SAB ([7, С. 103-104]), методика навчання ННМАННКП представлена в роботі [6].

План експериментів для порівняльного аналізу був наступний. Параметрами експерименту були: співвідношення навчальної і перевірочної вибірки, та кількість входів нейронної мережі. Порівняльний аналіз буде базуватися на значеннях критерію (11), чи (13), в залежності від обставин, що будуть обумовлені нижче. Співвідношення навчальної та перевірочної вибірок (Навч.: пер.) вибрано наступними: 10:90, 20:80, 30:70, 40:60, 50:50, 60:40, 70:30, 80:20, 90:10. Кількість входів було вибрано – 2, 3 чи 4.

Результати експериментів першого порівняльного аналізу наведені в Табл. 2.

Табл. 2. Результати порівняльного аналізу мережі ННМАННКП, Мамдані та РБМ

Значення критерію якості функціонування мережі для котирування	Мережі								
	Кількість входів мережі	ННМАННКП			Радиальна базисна модифікована			Мамдані		
		Навч.: пер.	MAPE навч.	MAPE пер.	Навч.: пер.	MAPE навч.	MAPE пер.	Навч.: пер.	MAPE навч.	MAPE пер.
EUR/GBP	2	70:30	0,0043	0,0033	40:60	0,0048	0,0051	80:20	0,0038	0,0039
	3	70:30	0,0038	0,0033	70:30	0,0075	0,0067	40:60	0,0031	0,0044
	4	50:50	0,0039	0,0036	30:70	0,0073	0,0072	70:30	0,0032	0,0045
EUR/USD	2	80:20	0,0045	0,0037	80:20	0,0048	0,0043	60:40	0,0035	0,0044
	3	70:30	0,0044	0,0046	80:20	0,0056	0,0055	70:30	0,0032	0,005
	4	70:30	0,0051	0,0053	80:20	0,008	0,0062	70:30	0,0033	0,0045
USD/JPY	2	70:30	0,0055	0,0046	60:40	0,0054	0,0049	40:60	0,0043	0,0053
	3	70:30	0,0052	0,0047	80:20	0,0061	0,0049	60:40	0,0036	0,006
	4	80:20	0,007	0,0039	80:20	0,0065	0,0063	80:20	0,0039	0,0054
USD/CHF	2	60:40	0,0049	0,0036	70:30	0,0045	0,0048	80:20	0,0041	0,0037
	3	80:20	0,0049	0,0045	40:60	0,005	0,0054	80:20	0,0037	0,004
	4	70:30	0,0054	0,0033	80:20	0,0056	0,0048	80:20	0,0033	0,0041

В Табл. 2 в колонках «Навч.:пер.» приведені співвідношення навчальної та перевірочної вибірок, що дали найкращі результати. Зліва від цих колонок розташовані колонки «MAPE навч.» та «MAPE пер.» – відповідно до цього співвідношення вибірок значення критерію (11) на навчальній та перевірочній вибірках. На основі аналізу Табл. 1, можна прийти до висновку, що найкраще функціонуючою виявилась ННМАННКП – в більшості випадків вона дала найкращі результати. «На другому місці» – нечітка нейронна мережа Мамдані, а найгірші результати були отримані після застосування РБМ.

В Табл. 3 наведена друга група експериментів, що присвячена порівняльному аналізу мережі Мамдані та РБМ на даних з [5].

Табл. 3. Результати експерімнтів по прогнозуванню фінансових інструментів

Значення критерію якості функціонування мережі для котирування	Мережі						
	Кількість входів мережі	Радиальна базисна модифікована			Мамдані		
		Навч.: пер.	MAPE навч.	MAPE пер.	Навч.: пер.	MAPE навч.	MAPE пер.
CAC40	2	80:20	0,0147	0,0127	50:50	0,0094	0,0117
	3	80:20	0,0165	0,0128	70:30	0,0037	0,0177
	4	50:50	0,0153	0,0155	60:40	0,0073	0,0118
COMEX GOLD	2	80:20	0,008	0,0087	60:40	0,0066	0,0058
	3	60:40	0,0115	0,0096	50:50	0,0058	0,0074
	4	50:50	0,0125	0,0127	80:20	0,0053	0,0077
DAX	2	70:30	0,0098	0,0103	80:20	0,0077	0,0076

	3	70:30	0,0118	0,0115	80:20	0,0064	0,0096
	4	80:20	0,0122	0,0109	70:30	0,0051	0,0105
DJIA	**2**	70:30	0,0103	0,0078	80:20	0,0066	0,0072
	3	70:30	0,0123	0,0109	80:20	0,0054	0,011
	4	80:20	0,0133	0,0132	50:50	0,0031	0,0121
HANG SENG	**2**	50:50	0,0097	0,0097	70:30	0,0077	0,0096
	3	80:20	0,014	0,0121	40:60	0,0081	0,009
	4	30:70	0,0121	0,0119	50:50	0,005	0,011
ICE.BRN	**2**	70:30	0,0141	0,0123	40:60	0,0123	0,0112
	3	80:20	0,0157	0,0137	80:20	0,0102	0,0159
	4	80:20	0,0165	0,015	70:30	0,0075	0,0155
GAZP	**2**	70:30	0,0182	0,0158	80:20	0,0122	0,0182
	3	80:20	0,0255	0,0256	60:40	0,0105	0,0167
	4	50:50	0,0278	0,029	60:40	0,0116	0,0186
ММВБ	**2**	40:60	0,0127	0,0125	60:40	0,0092	0,0123
	3	30:70	0,0139	0,0156	70:30	0,0105	0,0096
	4	80:20	0,0159	0,0143	80:20	0,0065	0,0166
NASDAQ	**2**	80:20	0,0102	0,011	70:30	0,0083	0,008
	3	80:20	0,0135	0,0145	80:20	0,0072	0,0101
	4	80:20	0,015	0,012	60:40	0,066	0,0122
RTS2	**2**	30:70	0,0091	0,0094	70:30	0,0058	0,0061
	3	40:60	0,0091	0,0105	60:40	0,0053	0,0073
	4	30:70	0,0103	0,0125	40:60	0,0039	0,008

Позначення в Таблиці 3 аналогічні позначенням Таблиці 2. Аналізуючи результати, приведені в Таблиці 3, можна прийти до висновку, що кращою за значеннями критерію (11) все одно залишається мережа Мамдані, однак десь у 50% результатів і РБМ була не гіршою за нечітку мережу Мамдані.

В Таблиці 4 приведені результати порівняльного аналізу мережі Мамдані та РБМ на даних [4] та [5], однак значення цих інструментів були перетворені за допомогою (12), і тому в даній групі експериментів порівняльний аналіз базувався на значеннях критерію (13) (при цьому перетворенні він має той самий сенс, що і (11), як було зазначено раніше). Цікавість цієї групи експериментів, на противагу групі експериментів напрямлених на аналіз власне котирувань, в тому, що перетворення (12) дозволяє працювати з реальними коливаннями фінансових інструментів, усунувши вплив тренду та масштабу.

Табл. 4. Результати експериментів по прогнозуванню перетворення (12) фінансових інструментів

Значення критерію якості функціонування мережі для котирування	**Мережі**						
	Кількість входів мережі	**Радиальна базисна модифікована**			**Мамдані**		
		Навч.: пер.	**RMSE навч.**	**RMSE пер.**	**Навч.: пер.**	**RMSE навч.**	**RMSE пер.**
EUR/GBP	**2**	60:40	0,0532	0,0514	40:60	0,0442	0,0498
	3	70:30	0,0721	0,0692	80:20	0,0403	0,056
	4	30:70	0,0578	0,0681	70:30	0,0433	0,0498
EUR/USD	**2**	30:70	0,0354	0,042	50:50	0,0311	0,0365
	3	30:70	0,046	0,0415	70:30	0,0329	0,0337

	4	70:30	0,049	0,042	80:20	0,0332	0,0361
USD/JPY	**2**	70:30	0,0432	0,0418	40:60	0,0334	0,0436
	3	80:20	0,0521	0,0453	60:40	0,0346	0,0404
	4	70:30	0,0523	0,0553	80:20	0,0301	0,0445
USD/CHF	**2**	80:20	0,0386	0,0379	80:20	0,0333	0,0349
	3	40:60	0,0481	0,0471	60:40	0,0319	0,0379
	4	50:50	0,0497	0,043	70:30	0,0237	0,0477
CAC40	**2**	40:60	0,0776	0,0855	80:20	0,0556	0,088
	3	30:70	0,0854	0,0949	80:20	0,049	0,0816
	4	50:50	0,1014	0,0952	80:20	0,0351	0,0989
COMEX GOLD	**2**	80:20	0,0328	0,0339	80:20	0,0289	0,0265
	3	50:50	0,0471	0,0466	60:40	0,0261	0,0296
	4	30:70	0,0451	0,0409	80:20	0,0246	0,0354
DAX	**2**	80:20	0,0471	0,0456	80:20	0,0357	0,0409
	3	30:70	0,0481	0,0523	70:30	0,0307	0,0437
	4	30:70	0,0583	0,0518	60:40	0,0323	0,041
DJIA	**2**	80:20	0,0503	0,0504	80:20	0,041	0,0388
	3	70:30	0,0545	0,0502	60:40	0,035	0,0706
	4	60:40	0,0632	0,061	80:20	0,0219	0,0729
HANG SENG	**2**	80:20	0,0568	0,0409	80:20	0,0385	0,0383
	3	70:30	0,0482	0,0476	70:30	0,0378	0,0388
	4	30:70	0,0507	0,0544	70:30	0,0329	0,0445
ICE.BRN	**2**	80:20	0,0452	0,0387	80:20	0,0361	0,0445
	3	80:20	0,0486	0,051	50:50	0,0283	0,0556
	4	30:70	0,0556	0,0536	70:30	0,0266	0,0538
GAZP	**2**	80:20	0,0522	0,0418	80:20	0,0343	0,0443
	3	50:50	0,0594	0,0642	80:20	0,0316	0,056
	4	80:20	0,0621	0,0454	40:60	0,0177	0,0615
ММВБ	**2**	50:50	0,0436	0,0421	40:60	0,0247	0,0415
	3	70:30	0,0456	0,0415	60:40	0,032	0,038
	4	80:20	0,0492	0,0371	80:20	0,0233	0,052
NASDAQ	**2**	70:30	0,0412	0,0444	80:20	0,0375	0,034
	3	30:70	0,0484	0,0475	70:30	0,0296	0,0462
	4	60:40	0,0594	0,051	80:20	0,0256	0,0475
RTS2	**2**	80:20	0,0283	0,0202	70:30	0,0179	0,018
	3	30:70	0,0378	0,0286	70:30	0,017	0,0195
	4	60:40	0,0382	0,034	50:50	0,0115	0,0217

Детально проаналізувавши Таблицю 4, можна прийти до висновку, що мережа Мамдані в більшості випадків виявилася краще за РБМ (ґрунтуючись на значеннях критерію *RMSE*).

Бігло оглянувши всі результати з експериментів, можна прийти до висновку, що найгіршою виявилася РБМ. Однак це – хибний висновок. Якщо уважно придивитися до значень критеріїв, то можна помітити, що значення мережі Мамдані та РБМ відрізняються на 1-5% в гіршому випадку. В той же час, можна помітити, що кращі результати у РБМ були досягнуті часто при співвідношеннях вибірок 30:70, що свідчить про кращі апроксимаційні властивості РБМ у порівнянні з іншими мережами (значно менше даних потрібно, щоб зробити адекватний висновок і для цього необхідно створювати не дуже багато нейронів). Також в ході експериментів було підмічено, що РБМ виявилася найстабільнішою з мереж,

тобто якщо після навчання отримано значення критерію на навчальній вибірці, то для РБМ можна очікувати, що від цього значення близько буде розташовано значення на перевірочній вибірці, що повинно симпатизувати практикам. В той же час, мережа Мамдані, наприклад, при значенні 0,0454 критерію (13) на навчальній вибірці, дала при одному з експериментів значення 2,0217 на перевірочній вибірці. З цього можна зробити висновок про нестабільність поведінки нечіткої нейронної мережі Мамдані. ННМАННКП навчалася довше за інші дві мережі, що розглянуто в даній роботі. Автор ставив одні і ті ж експерименти на тому ж комп'ютері на мережі Мамдані втричі довше, ніж на РБМ і цей показник більш об'єктивний ніж вимірювання мілісекунд виконання окремої операції.

Що ж стосується донавчання, то уважний читач повинен був помітити, що, як можна побачити в частині «Опис модифікації», механізм до навчання вже закладено в алгоритм навчання РБМ. ННМАННКП донавчити легше за мережу Мамдані внаслідок не залежного навчання правил та антецедентів і консиквентів правил. Важче за все донавчити мережу Мамдані.

Загальний висновок можна зробити наступний: кожна мережа є кращою за інші в залежності від умов, в яких доводиться працювати.

Висновки

В роботі були наведені результати численьних експериментів по застосуванню РБМ, мережі Мамдані та ННМАННКП до аналізу значень світових фінансових індексів та котирувань фінансових інструментів. Найточніші результати дала ННМАННКП, також якість функціонування при збільшенні входів даної мережі залишалася приблизно на одному рівні. В той же час дана нейронна мережа навчалася довше за інші дві. Нечітка нейронна мережа на базі нечіткого виводу Мамдані була «на другому місці» по точності та по швидкості навчання. РБМ виявилася найстабільнішою (варто считати – найнадійнішою) та навчалася найшвидше за інші розглянуті мережі і, в той же час, якість її функціонування була незначно гірше за інші дві мережі. На підставі цього було зроблено висновок про те, що кожну мережу варто застосовувати в залежності від конкретних обставин на час застосування. Що ж стосується фінансових показників, що аналізувалися, то найскладнішими для аналізу (через їх нестабільність) виявилися закордонні фінансові індекси та російський індекс ММВБ, а більш легкими – російський індекс RTS2, золото та валюти. Ціна акцій «Газпром нефть» та ціна північноморської нафти Брент була «посередині» у сенсі складності аналізу. Слід відмітити, що дані результати відповідають результатам роботи [12].

Дана робота була присвячена оцінці здатності екстраполювати мережами часові ряди на крок вперед на підставі передісторії. Хоча, навчивши мережу екстраполювати дані на крок вперед, а потім застосовувати отримані результати для повторення цього процесу є вирішенням проблеми екстраполяції значень на декілька кроків вперед, це є не економічно. У зв'язку з цим, а також у зв'язку з можливістю отримання значно кращих результатів одна з наступних робіт автора буде присвячена саме оцінці здатності нейронних мереж, що були розглянуті у даній роботі до екстраполяції значень фінансових часових рядів на декілька кроків вперед.

Література

1. Снитюк В.Є. Прогнозування. Моделі. Методи. Алгоритми. – К.: «Маклаут», 2008. – 364 с.
2. Хайкин Саймон. Нейронные сети: полный курс, 2-е изд.: Пер. с англ. – М.: Издательский дом «Вильямс», 2006. – 1104 с.
3. Adrian G. Bors. Introduction of the Radial Basis Function (RBF) Networks. http://www-users.cs.york.ac.uk/~adrian/Papers/Others/OSEE01.pdf. - 7 p.
4. Денні котирування валютних пар EUR/GBP, EUR/USD, USD/JPY, USD/CHF за період з 25.03.2009 по 24.03.2010. http://www.finam.ru/analysis/export/default.asp.

5. Щоденні значення світових індексів CAC40, DAX, DJIA, HANG SENG, NASDAQ, RTS2, індекс ММВБ та денні котирування фінансових інструментів COMEX GOLD, ICE.BRN, ММВБ «Газпром нефть» - GAZP за період з 1.02.2010 по 1.02.2011. http://www.finam.ru/analysis/export/default.asp.
6. Мурга Н.А. Нейросетевая архитектура на частичных обучениях // New Trends in Classification and Data Mining. ITHEA (Sofia, Bulgaria). – 2010. – № 16 – С. 170-184.
7. Зайченко Ю.П. Основи проектування інтелектуальних систем. – К.: Видавничий Дім «Слово», 2004. – 352 с.
8. Зайченко Ю.П. Нечёткие модели и методы в интеллектуальных системах. – К.: «Издательский Дом «Слово», 2008. – 344 с.
9. Браммер К., Зиффлинг Г. Фильтр Калмана-Бьюси: детерменированное наблюдение и стохастическая фильтрация – пер. с нем. – М.: Наука. Главная редакция физико-математической литературы, 1982. – 200 с.
10. Mark Hudson Beale, Mavtin T. Hagan, Howard B. Demuth. Neural Network Toolbox 7: User's Guide. – MATLAB. MathWorks, 2010. – 951 p.
11. Петерс Э. Фрактальный анализ финансовых рынков: Примененние теории хаоса в инвестициях и экономике. – М.: Интернет-трейдинг, 2004. – 304 с.
12. Мурга Н.А. Нечёткий логический вывод Херста // Вісник Черкаського державного технологічного університету. Сер. Технічні науки. – 2010. - № 4 – С. 36-41.

УДК 004. 056.5

ФЕДОРЕЧКО О.И.,
ВИНОГРАДОВ Ю.Н.,
ИВАНОВ А.Н.

МЕТОД ИСПРАВЛЕНИЯ “ПАЧКИ” ОШИБОК В КАНАЛАХ С КОДОВО-ИМПУЛЬСНОЙ МОДУЛЯЦИЕЙ НА ОСНОВЕ УМНОЖЕНИЯ БЕЗ МЕЖРАЗРЯДНЫХ ПЕРЕНОСОВ

В статье предложен метод исправления “пачки” ошибок в каналах с импульсно-кодовой модуляцией. Метод основан на математической операции двоичного умножения без переносов. Предложенный метод позволяет ускорить и уменьшить вычислительную сложность процедуры коррекции как по сравнению с кодами Рида-Соломона, так и по сравнению с известными кодами на основе взвешенных контрольных сумм. Вначале приведено математическое обоснование предложенного метода. Подробно изложена процедура коррекции пачки ошибок. Приведен численный пример. Выполнен аналитическое сравнение сложности предложенного и известного методов, которое показывает достигаемые преимущества. Предлагаемый способ может найти широкое применение в современных системах передачи данных реального времени.

In article the method for burst errors correction in channels with pulse-code modulation is proposed. The method is based on the mathematical operation of binary multiplication without carry. The offered method permits accelerated performance with reduced computational complexity as compared to Reed – Solomon codes, as compared to known weighed check sum codes while using the same number of bits for error control. The mathematical background of the proposed method is first presented. The procedure for burst errors is then explained. A numerical example for procedure is given. An analytical comparison of the complexity of both the proposed and the known methods is presented, that demonstrates the improvements attained. The proposed method can find wide application in modern real time data transmission.

Введение

В современных условиях динамичного роста скорости передачи данных, важное значение приобретает время, затрачиваемое на контроль возникающих ошибок. В большинстве систем контроль и исправление ошибок должно производиться без задержек, в темпе передачи данных. Особенно остро проблема быстрой реализации операций контроля стоит для канналов передачи данных с кодово-импульсной модуляцией. Такие каналы широко используются для обмена данными в компьютерных системах. Операции, связанные с контролем и коррекцией ошибок передачи данных выполняются в таких каналах встроенными аппаратными средствами. Для быстрого и эффективного контроля ошибок, схема таких средств должна быть возможно проще. Сложность схемы зависит от метода обнаружения и исправления ошибок передачи данных, а также от типа ошибок и их кратности.

Быстрое развитие технологий обмена данными меняет характер возникающих в каналах ошибок. Если, несколько десятилетий назад основной причиной ошибок были явления гауссова шума [77], то в настоящее время основными негативными факторами в плане возникновения ошибок передачи данных становятся внешние помехи [55]. Это обусловлено расширением использования беспроводных технологий передачи данных и мобильной связи. Меняется и характер внешних помех: возрастает их длительность. Это приводит к том, что они воздействуют на несколько информационных сигналов. Вследствие этого, доминирующим типом ошибок становятся “пачки” битовых искажений [99].

Таким образом, в современных условиях, задача создания метода коррекции “пачки” ошибок, обеспечивающего эффективную аппаратную реализацию встроенных средств контроля передачи цифровых данных в каналах с кодово-импульсной модуляцией является важной и актуальной.

Анализ известных методов коррекции “пачки” ошибок

Значительная практическая значимость задачи коррекции “пачки” ошибок в каналах с кодово-импульсной модуляцией стимулировала создание создано ряд методов ее решения.

Наиболее известный подход к коррекции “пачек” ошибок связан с использованием корректирующих кодов на основе циклических ко-

дов [1]. Корректирующие коды других классов (коды Хемминга, коды Голлея и др.) ориентированы на исправление произвольно локализованного множества искаженных битов, количество их разрядов с увеличением кратности ошибок растет экспоненциально и поэтому они изначально проигрывают циклическим кодам при решении задачи коррекции "пачки" ошибок.

Специально для коррекции "пачки" ошибок на основе циклических кодов созданы коды Файра, коды Миласа, коды Абрамсона и коды Рида-Соломона [1]. Во всех этих кодах контрольный код образуется как результат деления полинома информационного сообщения на образующий код полином. Фактически приведенные коды отличаются способом формирования образующего полинома.

На практике наибольшее распространение получили коды Рида-Соломона.

Коды Рида-Соломона представляют собой недвоичные циклические коды, предусматривающие разделение контролируемого информационного блока на символы. Эти коды позволяют исправлять любые искажение в h символах с использованием $2 \cdot h$ контрольных символов. Поскольку положение символов жестко фиксировано, а m-битовая "пачка ошибок" может возникнуть, начиная с любого из битов передаваемого блока, то для гарантированного ее исправления, код Рида-Соломона должен обеспечивать возможность коррекции 2-х символов (h=2) длиной m бит, то есть использовать 4 контрольных символа или $4 \cdot m$ контрольных разряда.

Кодирование, то есть вычисление контрольных разрядов на передатчике, состоит в нахождении остатка от деления n-символьного информационного блока на 4-символьный полином. Операция деления состоит из n циклов, на каждом из которых выполняется 4 операции умножения символов и одна операция сложения. Операция умножения на полях Галуа m-разрядных символов включает m циклов, в рамках каждого из которых, в среднем, выполняется две операции сдвига и одна операция суммирования над k-битовыми числами. Исходя из этого, время T_{C1} кодирования определяется в виде:

$$T_{C1} = 4 \cdot n \cdot (3 \cdot k \cdot t_{\ddot{e}} + t_{\ddot{e}}), \qquad (1)$$

где $t_л$ – время выполнения логической операции над k-разрядным символом.

Декодирование кодов Рида-Соломона включает два этапа: обнаружение ошибок и их исправление. Для обнаружения ошибок вычисляется 4 k-разрядных синдромов. Если значения всех синдромов равно нулю, то полагается, что сообщение передано без ошибок. При вычислении каждого из синдромов производится n операций умножения и такое же количество операций сложения. Поэтому, время T_{O1} обнаружения ошибок практически совпадает со временем кодирования.

Исправление ошибок передачи производится следующим порядком: вначале решается система из 4-х символьных уравнений для определения компонент локатора ошибок; методом перебора определяются позиции искаженных символов; решается система из 4-х символьных уравнений для определения символов искажений. Общее время T_{s1} коррекции определяется формулой:

$$T_{S1} \approx 4 \cdot n \cdot (3 \cdot m \cdot t_{\ddot{e}} + t_{\ddot{e}}). \qquad (2)$$

Из изложенного следует, что коррекция "пачки" из 2-х искаженных символов в коде Рида-Соломона требует значительного времени и достаточно сложных аппаратных решений, поскольку последние должны решать несколько разнородных математических задач.

Указанные недостатки обусловлены универсальным характером кодов Рида-Соломона, который не учитывает особенностей возникновения одиночной "пачки" ошибок в каналах с кодово-импульсной модуляцией.

Для повышения эффективности коррекции "пачки" ошибок было предложено использовать контрольный код на основе арифметической взвешенной контрольной суммы (АВКС) [2]. Этот контрольный код использует для кодирования операции арифметического умножения и сложения, а для коррекции ошибок – операцию целочисленного арифметического деления. Это позволяет существенно уменьшить время коррекции, сделав его независимым от n-числа символов в блоке. Однако при аппаратной реализации вычисление арифметических операций требует много времени, поскольку необходимо учесть последовательный характер формирования переносов. Кроме того, контрольные коды на основе АВКС требуют большего, по сравнению с кодами Рида-Соломона числа контрольных разрядов.

Целью исследований является разработка метода коррекции одиночной "пачки" ошибок в каналах с кодово-импульсной модуляцией,

обеспечивающего упрощение аппаратной реализации и ускорение выполнения процедур контроля ошибок и их исправления.

Метод исправления “пачки” ошибок на основе умножения без переносов

Предлагаемый метод использует логические операции и ориентирован на эффективную аппаратную реализацию. В основе метода лежит использование взвешенной контрольной суммы, вычисление которой выполняется с применением операции умножения без межразрядных переносов. Эта операция ниже обозначается знаком $\otimes$.

Произведение без межразрядных переносов $D=U\otimes V=\{d_{2\cdot r},\ldots,d_3,d_2,d_1\}=d_1+2\cdot d_2+4\cdot d_3+\ldots+2^{2\cdot r}\cdot d_{2\cdot r}$, $\forall l\in\{1,..,2\cdot r\}$: $d_l\in\{0,1\}$ двух r-разрядных чисел $U=\{u_r,\ldots,u_2,u_1\}=u_1+2\cdot u_2+\ldots+2^r\cdot u_r$ и $V=\{v_r,\ldots,v_2,v_1\}=v_1+2\cdot v_2+\ldots+2^r\cdot v_r$, $\forall i\in\{1,..,r\}$: $v_i,u_i\in\{0,1\}$ вычисляется следующим образом:

$$D=U\cdot v_1\oplus(U\cdot v_2)<<1\oplus\ldots\oplus(U\cdot v_r)<<(r-1), \quad (3)$$

где $\cdot$ – операция логического умножения, $p<<q$ – операция логического сдвига числа p влево на q разрядов.

Если задана максимальная длина m “пачки” битовых искажений, то информационный N-битовый блок $B=\{b_1,b_2,\ldots,b_N\}$, $\forall l\in\{1,..,N\}$: $b_l\in\{0,1\}$ предлагается рассматривать состоящим из $n=N/m$ символов $B=\{X_1,X_2,\ldots,X_n\}$.

Поскольку длина пачки ограничена m битами, то она может исказить непересекающиеся подмножества битов только двух смежных символов X_i и X_{i+1}, $i\in\{1,\ldots,n\text{-}1\}$.

Для исправления такого вида ошибок предлагается использовать контрольную сумму, которая состоит из трех компонент:

1) Первая C_1 – сумма по модулю два символов, с нечетными номерами в блоке:

$$C_1=X_1\oplus X_2\oplus\ldots\oplus X_{n-1}. \quad (4)$$

2) Компонента C_2 – сумма по модулю два символов, которые находятся на четных позициях:

$$C_2=X_2\oplus X_4\oplus\ldots\oplus X_n. \quad (5)$$

3) Третья компонента C_3 представляет собой сумму по модулю два произведений без межразрядных переносов символов блока на коды их порядковых номеров:

$$C_3=(1\otimes X_1)\oplus(2\otimes X_2)\oplus\ldots\oplus(n\otimes X_n). \quad (6)$$

Компоненты контрольной суммы C_{1s}, C_{2s} та C_{3s} для информационного блока B вычисляются на передатчике. Порядок передачи пакета данных приемнику следующий: третья, первая и вторая компоненты контрольного кода, а затем информационный блок B : $C_{3s}, C_{1s}, C_{2s}, B$.

На приемнике по полученному блоку данных B вычисляются компоненты контрольного кода C_{1R}, C_{2R}, C_{3R} и разности Δ_1, Δ_2, Δ_3 компонент контрольного кода приемника и передатчика:

$$\begin{aligned}\Delta_1&=C_{1S}\oplus C_{2R},\\ \Delta_2&=C_{2S}\oplus C_{2R},\\ \Delta_3&=C_{3S}\oplus C_{3R}.\end{aligned} \quad (7)$$

По значениям Δ_1, Δ_2 и Δ_3 определяется наличие ошибки в принятом информационном блоке: если $\Delta_1=0$, $\Delta_2=0$ и $\Delta_3=0$, то принятый информационный блок B не содержит ошибок, в противном случае выполняется классификация типа возникших при передаче ошибок.

Перечень возможных вариантов искажений, вызванных “пачкой” ошибок приведен ниже:

1) $\Delta_1\neq0$, $\Delta_2=0$ и $\Delta_3\neq0$ и при этом $\Delta_3>\Delta_1$ – “пачка” ошибок исказила один информационный символ, который находится на нечетной позиции.

2) $\Delta_1=0$, $\Delta_2\neq0$ и $\Delta_3\neq0$ – “пачка” ошибок исказила один информационный символ, который находится на четной позиции.

3) $\Delta_1\neq0$, $\Delta_2\neq0$ и $\Delta_3\neq0$, $\Delta_1<\Delta_2$ – “пачка” ошибок исказила два смежных символа, причем информационный символ, который находится на нечетной позиции был искажен первым.

4) $\Delta_1\neq0$, $\Delta_2\neq0$ и $\Delta_3\neq0$, $\Delta_1>\Delta_2$ и при этом $\Delta_3>\Delta_1$ и $\Delta_3\neq\Delta_1$–“пачка” ошибок исказила два смежных символа, причем информационный символ, который находится на четной позиции был искажен первым.

5) $\Delta_1\neq0$, $\Delta_2\neq0$ и $\Delta_3\neq0$, при чем $\Delta_3=\Delta_1$, то “пачка” ошибок исказила первый символ информационного блока и вторую компоненту контрольного кода .

6) $\Delta_1\neq0$, $\Delta_2=0$ и $\Delta_3=0$ – “пачка” исказила первую компоненту контрольного кода.

7) $\Delta_1=0$, $\Delta_2\neq0$ и $\Delta_3=0$ – “пачка” исказила вторую компоненту контрольного кода.

8) $\Delta_1=0$, $\Delta_2=0$ и $\Delta_3\neq0$ – “пачка” исказила третью компоненту контрольного кода.

9) $\Delta_1\neq0$, $\Delta_2\neq0$ и $\Delta_3=0$ – “пачка” исказила первую и вторую компоненту контрольного кода.

10) $\Delta_1 \neq 0$, $\Delta_2 = 0$, $\Delta_3 \neq 0$ и при этом $\Delta_3 < \Delta_1$ – “пачка” исказила первую и третью компоненту контрольного кода.

В том случае, если после анализа информационного блока на наличие в нем ошибок, была обнаружена ошибка в информационном блоке, то процедура коррекции в этом случае состоит из следующих действий: определения номера первого искаженного символа и выполнения коррекции. В предложенном методе для выполнения эффективной коррекции используется информация, содержащаяся в компоненте Δ_3: в случае возникновения “пачки” ошибок в ее формировании принимают участие номера позиций искаженных символов в информационном блоке (P_i – номер первого символа искаженного “пачкой” ошибок, а P_{i+1} – номер второго). Если, “пачка” ошибок исказила информационный блок начиная с символа находящегося на нечетной позиции ($\Delta_1 < \Delta_2$), то количество ненулевых разрядов в компоненте Δ_1 указывают сколько раз при формировании компоненты Δ_3 выполнялась сумма по модулю два номера первого искаженного символа, а позиции ненулевых значений в компоненте Δ_1 указывают на сколько разрядов влево были сдвинуты значения номера первого искаженного символа, которые берут участие при формировании компоненты Δ_3. По количеству единиц в компоненте Δ_2 можно определить количество раз, которое номер второго искаженного символа участвует при формировании компоненты Δ_3, а номера позиций единиц в компоненте Δ_2 определяют на сколько разрядов участвующие значения номера второго искаженного символа сдвинуты влево от начала. Так, например, при количестве разрядов k=4 и значениях компонент $\Delta_1 = 0101_2$ и $\Delta_2 = 1000_2$ видно, что при формировании компоненты Δ_3 номера искаженных символов принимали участие в порядка, показанном на рис.1. Так, как в компоненте Δ_1 два ненулевых разряда, это значит, что номер первого искаженного символа принимал участь в формировании разрядов компоненты Δ_3, два раза, причем первый раз без сдвига (так, как ненулевым является младший разряд компоненты Δ_1), а во второй раз со сдвигом на два разряда влево (так, как ненулевым является третий разряд компоненты Δ_1); из компоненты Δ_2 видно, что номер второго искаженного символа принимал участие в формировании компоненты Δ_3 один раз, причем, он был сдвинут влево от начала на три разряда (так как ненулевым является четвертый разряд компоненты Δ_2). Если сначала был искажен символ, находящийся на четной позиции, то по компоненте Δ_2, определяется порядок участия номера первого искаженного символа при формировании компоненты Δ_3, а по компоненте Δ_2 – второго номера искаженного символа.

Для дальнейшего объяснения алгоритма коррекции вводятся следующие условные обозначения:

- $P_i[j]$ – значение j-го разряда номера первого искаженного символа $j \in \{0..\log_2 n\}$;
- $P_{i+1}[j]$ – значение j-го разряда номера второго искаженного символа $j \in \{0..\log_2 n\}$;
- $C[j]$ – значение j-го переменной межразрядных переносов $j \in \{0..\log_2 n\}$;
- $\Delta_1[j]$ – значение j-го разряда номера первой компоненты контрольного кода $j \in \{0..k\}$;
- $\Delta_2[j]$ – значение j-го разряда номера второй компоненты контрольного кода $j \in \{0..k\}$;
- $\Delta_3[j]$ – значение j-го разряда номера третьей компоненты, $j \in \{0..\log_2 n+k-1\}$;
- r – индекс строки матрицы;
- st – индекс столбца матрицы;

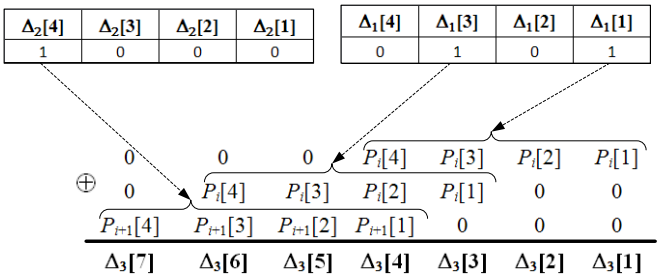


Рис. 1. Пример формирования разрядов Δ_3

Поскольку размер “пачки” ошибок всегда меньше или равен разрядности символа информационного блока, то при искажении двух смежных символов она не может исказить их в одинаковых разрядах. Таким образом, при формировании компоненты Δ_3 номера первого и второго искаженного символа организуют ступенчатую структуру при вычислении разрядов компоненты Δ_3, с возможностью поразрядного определения значений номеров искаженных символов P_i и P_{i+1}. символы информационного блока, то следует отметить еще одну особенность, которая вытекает из очевидного факта того, что значение следующего номера искаженного символа P_{i+1} можно получить, если прибавить единицу к номеру символа P_i и при

этом значение их разрядов отличаются на значение переменной C которая содержит значения межразрядных переносов при выполнении суммы по модулю два $P_{i.}$ с P_{i+1} : $C= P_i \oplus P_{i+1}$. Следовательно, значение номера второго искаженного символа P_{i+1} может быть выражено через значение номера первого искаженного символа $P_{i.}$ и значения межразрядных переносов C:

$$P_{i+1} = P_i \oplus C. \qquad (8)$$

Учитывая описанные выше особенности, и ступенчатый характер формирования компоненты Δ_3 была разработана процедура нахождения номера первого искаженного "пачкой" ошибок символа.

Процедура нахождения номера первого искаженного символа состоит из двух этапов:

1) Определение позиции младшего искаженного "пачкой" ошибок разряда в первом искаженном символе P_i.

2) Восстановление значений разрядов весового коефициента.

Первый этап алгоритма, состоит в определении позиции самого младшего искаженного "пачкой" ошибок разряда в первом искаженном символе, его необходимо найти для того, чтобы правильно локализовать местонахождение начала номера первого искаженного символа P_i, который сдвинутый на наименьшее количество разрядов влево относительно начала, при вычислении компоненты Δ_3. Для этого необходимо определить: где находится начало "пачки" ошибок, а именно в символе с четным или с нечетным порядковым номером в информационном блоке. Если значение Δ_1 меньше значения Δ_2 и при этом $\Delta_1 \neq 0$, то начало пачки ошибок находится в символе с нечетным значением номера позиции в информационном блоке, иначе – в символе с четным значением. В первом случае анализируется значение компоненты Δ_1 начиная с младшего разряда, в другом – выполняется аналогичный анализ Δ_2. Анализ выполняется до тех пор пока не будет найдено первый ненулевой разряд. Номер позиции этого разряда в информационном символе сохраняется в переменных r и st.

Второй этап алгоритма вычисления разрядов номера первого искаженного символа состоит в последовательности действий:

1) Начинается цикл поразрядного нахождения значения номера первого искаженного символа P_i от 1 до $log_2(n)$, счетчику присваивается значение ноль a=0.

2) Инициализация переменной $counter=r+1$.

3) Начинается цикл суммирования по модулю два разрядов номера первого искаженного символа P_i, которые формируют текущий разряд st компоненты Δ_3. Цикл от a-1 до нуля включительно, значению j присваивается значение a-1.

3.1) Если $counter<k$, то переход на пп.3.2 иначе на пп. 4.

3.2) Если $\Delta_1[counter] \neq 0$ или $\Delta_2[counter] \neq 0$,то переход на пп. 3.3 иначе – на пп.3.4.

3.3) $P_i[a] = P_i[a] \oplus P_i[j]$;

3.4) Если $\Delta_1 > \Delta_2$ и при этом $\Delta_1[counter] \neq 0$ или $\Delta_1 < \Delta_2$ и при этом $\Delta_2[counter] \neq 0$, то переход на пп. 3.5 иначе – на пп.4

3.5) $P_i[a] = P_i[a] \oplus C[j]$;

4) Инкремент переменной counter.

5) Декремент переменной j.

6) Проверка окончания цикла: если j=0, то переход на пп 7. Иначе на пп. 3.1.

7) $P_i[a] = P_i[a] \oplus \Delta_3[st]$.

8) Определение переноса и запись его в переменную C. Если $P_i[a]$=1 і $C[a]$=1, то фиксируется наличие переноса $C[a+1]$=1, иначе $C[a+1]$=0.

9) Инкремент переменной st.

10) Инкремент переменной a.

11) Проверка окончания цикла: если $a<\log_2(n)-1$, то переход на пп 2. Иначе -выход на конец процедуры, поскольку определение разрядов номера первого искаженного символа окончено.

После окончания процедуры определения номера первого искаженного символа производится коррекция искаженных символов информационного блока. Коррекция производится в зависимости от типа искажения, которое возникла в информационном блоке в результате воздействия внешней помехи. Перечень возможных искажений блока и соответствующие операции коррекции представлены ниже:

1) Если "пачка" ошибок исказила один информационный символ, который находится на нечетной позиции ($\Delta_1 \neq 0$, Δ_2=0 и $\Delta_3 \neq 0$ и при этом $\Delta_3 > \Delta_1$), то коррекция проводится путем выполнения суммы по модулю два информационного символа находящегося на позиции P_i и компоненты Δ_1.

2) Если “пачка” ошибок исказила один информационный символ, который находится на четной позиции ($\Delta_1=0$, $\Delta_2\neq0$ и $\Delta_3\neq0$), то коррекция проводится путем выполнения суммы по модулю два информационного символа находящегося на позиции P_i и компоненты Δ_2.

3) Если “пачка” ошибок исказила два смежных символа, причем информационный символ, который находится на нечетной позиции был искажен первым ($\Delta_1\neq0$, $\Delta_2\neq0$ и $\Delta_3\neq0$, $\Delta_1<\Delta_2$), то коррекция проводится путем выполнения суммы по модулю два информационного символа находящегося на позиции P_i и компоненты Δ_1 и выполнения суммы по модулю два информационного символа находящегося на позиции P_{i+1} и компоненты Δ_2.

4) Если “пачка” ошибок исказила два смежных символа, причем информационный символ, который находится на четной позиции был искажен первым ($\Delta_1\neq0$, $\Delta_2\neq0$ и $\Delta_3\neq0$, $\Delta_1>\Delta_2$ и при этом $\Delta_3>\Delta_1$ и $\Delta_3\neq\Delta_1$), то коррекция проводится путем выполнения суммы по модулю два информационного символа находящегося на позиции P_i и компоненты Δ_2 и выполнения суммы по модулю два информационного символа находящегося на позиции P_{i+1} и компоненты Δ_1.

5) Если “пачка” ошибок исказила первый символ информационного блока и вторую компоненту контрольного кода ($\Delta_1\neq0$, $\Delta_2\neq0$ и $\Delta_3\neq0$, при чем $\Delta_3=\Delta_1$), то коррекция проводится путем выполнения суммы по модулю два первого информационного символа P_1 и компоненты Δ_1.

Например, k=4 при возникновении «пачки» ошибок искажаются 7-й и 8-й символы. Пусть на передатчике X_{7S}={1,1,1,0} и X_{8S}={0,1,1,1}, а на приемнике X_{7R}={1,0,0,0} и X_{8R}={1,1,1,1}. Соответственно Δ_1={0,1,1,0} Δ_2={1,0,0,0}. $\Delta_3=(7<<1)\oplus(7<<2)\oplus(8<<3)=14\oplus28\oplus64=82_{10}=$ $=1010010_2$.

Разряды номеров искаженных символов формирующие компоненту Δ_3 для этого примера представлены на рисунке 2.

	0	0	$P_i[4]$	$P_i[3]$	$P_i[2]$	$P_i[1]$	0
$\oplus$	0	$P_i[4]$	$P_i[3]$	$P_i[2]$	$P_i[1]$	0	0
	$P_{i+1}[4]$	$P_{i+1}[3]$	$P_{i+1}[2]$	$P_{i+1}[1]$	0	0	0
	Δ_3**[7]**	Δ_3**[6]**	Δ_3**[5]**	Δ_3**[4]**	Δ_3**[3]**	Δ_3**[2]**	Δ_3**[1]**

Рис. 2. Пример формирования разрядов компоненты Δ_3

Проанализировав значения компонент Δ_1 и Δ_2 можно сделать вывод, что самый младший искаженный разряд находится на второй позиции $r=2$ (при нумерации строк начиная с единицы). Следовательно, определение разрядов номера первого искаженного символа необходимо начинать из второй строки и второго столбца. Вычисление третьей компоненты контрольной суммы с учетом формулы 11 изображено на рисунке 3.

	0	0	$P_i[4]$	$P_i[3]$	$P_i[2]$	$P_i[1]$	0
$\oplus$	0	$P_i[4]$	$P_i[3]$	$P_i[2]$	$P_i[1]$	0	0
	$P_i[4]\oplus C[3]$	$P_i[3]\oplus C[2]$	$P_i[2]\oplus C[1]$	$P_i[1]\oplus1$	0	0	0
	Δ_3**[7]**	Δ_3**[6]**	Δ_3**[5]**	Δ_3**[4]**	Δ_3**[3]**	Δ_3**[2]**	Δ_3**[1]**

Рис. 3. Вычисление третьей компоненты контрольной суммы

Вычисление значений разрядов предлагается проводить начиная с того столбца *st*, который соответствует номеру первого искаженного разряда *r*. В приведенном примере необходимо начинать со второго столбца. Воспроизведение значения разрядов номера первого искаженного символа выполняется поразрядно, как это показано на рис.4.

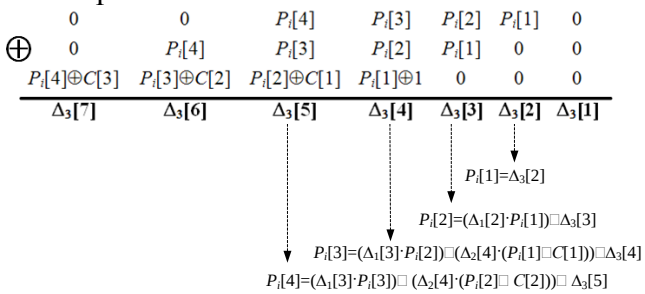


Рис. 4. Вычисление значений разрядов первого искаженного символа

На рисунке 4 видно, что разряды номера первого искаженного символа P_i, можно получить путем выполнения суммы по модулю два значений разрядов, которые находятся ниже него в столбце при формировании соответствующего значения разряда компоненты Δ_3.

Пошаговое выполнение алгоритма вычисления номера первого искаженного символа представлено ниже.

Действия, которые выполняются на первом шаге:

$P_i[1]=\Delta_3[2]=1$;

Так как соседние разряды точно отличаются на единицу, то есть возможность определить $P_{i+1}[1]$ и также можно определить $C[1]$ первый перенос;

$P_{i+1}[1]= P_i[1] \oplus1=0$;

$C[1]=1$, так как $P_i[1]=1$, то для получения младшего разряда прибавлялась единица, а это значит, что перенос был.

Действия, которые выполняются на втором шаге:

Переход к следующему, 3-му столбцу.

$P_i[2]= P_i[1] \oplus 1 \oplus \Delta_3[3]=1\oplus1\oplus1=1$

$P_{i+1}[2]= P_i[2] \oplus C[1]=1\oplus1=0;$

$C[2]=1$, так как $P_i[1]=1$ и $C[1]= 1$, то имел место перенос.

Действия, которые выполняются на третьем шаге:

Переход к следующему, 4-му столбцу.

$P_i[3]= P_i[2] \oplus C[1] \oplus P_i[1] \oplus 1 \oplus\Delta_3[4]= =1\oplus1\oplus1\oplus1\oplus1=1$

$P_{i+1}[3]= P_i[2] \oplus C[2]= 1\oplus1=0;$

$C[3]=1$, так как $P_i[2]=1$ и $C[2]= 1$, то имел место перенос.

Действия, которые выполняются на четвертом шаге:

Переход к следующему, 5-му столбцу.

$P_i[4]= P_i[3] \oplus C[2] \oplus P_i[2] \oplus C[1] \oplus\Delta_3[5]= =1\oplus1\oplus1\oplus1\oplus0=0$

$P_{i+1}[4]= P_i[4] \oplus C[3]= 0\oplus1=1;$

$C[4]=0$, так как $P_i[4]=0$, а $C[3]= 1$, то переноса не было.

В таблице 1 представлены значения переменных после четвертого шага алгоритма.

Табл. 1. Значения переменных P_i, P_{i+1},

№ разряда / Переменные	4	3	2	1
P_i	0	1	1	1
P_{i+1}	1	0	0	0
C	0	1	1	1

Определение номеров искаженных символов закончено: $P_i =7$ и $P_{i+1}=8$. коррекция проводится путем выполнения суммирования по модулю символа, находящегося на позиции P_i с значением первой компоненты Δ_1, и символа, находящегося на позиции P_{i+1} и второй компоненты Δ_2, $X_{7R}\oplus\Delta_1= 1000_2\oplus1000_2= 1110_2$ та $X_{8R}\oplus\Delta_2=1111_2\oplus1000_2=0111_2$. Так, как значения символов $X_{7R}=1110_2$ и $X_{8R}=0111_2$ после коррекции совпадают с значениями соответствующих символов на передатчике $X_{7S}=1110_2$ и $X_{8S}=0111_2$, то можно сделать вывод что коррекция была проведена успешно.

Анализ эффективности предложенного метода

Число k контрольных разрядов, используемых в предложенном методе коррекции одной “пачки” ошибок в каналах с кодово-импульсной модуляцией определяется суммарной разрядностью всех 3-х компонент контрольного кода. Учитывая, что разрядность двух первых компонент равна k, а разрядность третьей: $k+\lfloor\log_2 n\rfloor$, число контрольных разрядов d определяется в виде:

$$d = 3\cdot k + \lfloor \log_2 n \rfloor. \qquad (9)$$

В отличие от кодов Рида-Соломона, которые накладывают ограничение на число n символов информационного блока: $2^k\le n$, в предлагаемом методе число символов может быть любым. Это позволяет эффективно использовать метод для коррекции ошибок при передаче блоков большой длины.

При использовании кодов Рида-Соломона, наибольшая эффективность использования контрольных разрядов достигается при $2^k=n$. Если принять это условие для предлагаемого метода, то формула (9) преобразуется к виду: $d=4\cdot k$. Это означает, что предлагаемый метод и коды Рида-Соломона для решения одинаковой задачи - коррекции одиночной “пачки” ошибок в каналах с кодово-импульсной модуляцией используют одинаковое количество контрольных разрядов.

Основным преимуществом предлагаемого метода коррекции “пачки” ошибок является ускорение выполнения операций кодирования. декодирования и исправления ошибок. Сравнение времен выполнения основных операций, связанных с контролем и исправлением “пачки” ошибки, для предложенного и известных методов приведено в таблице 2.

Табл. 2. Сравнительные характеристики времени выполнения операций

Метод коррекции “пачки” ошибок	Время кодирования	Время декодирования и анализа синдрома	Время коррекции “пачки” ошибок
Предложенный метод	$2\cdot n\cdot k\cdot t_л$	$2\cdot n\cdot k\cdot t_л$	$\frac{k^2}{4}\cdot t_л$
Метод на основе АВКС	$4\cdot n\cdot k^2\cdot t_л$	$4\cdot n\cdot k^2\cdot t_л$	$k^2\cdot t_л$
Коды Рида-Соломона	$12\cdot n\cdot k^2\cdot t_л$	$12\cdot n\cdot k^2\cdot t_л$	$12\cdot n\cdot k\cdot t_л$

Из таблицы 2 следует, что предложенный метод фактически обеспечивает ускорение в k раз критической по времени операции кодирования. Критической, поскольку кодирование должно происходить в темпе передачи данных. Время коррекции “пачки” по сравнении с арифметической взвешенной контрольной суммой уменьшается в 4 раза, а в сравнении с ко-

дами Рида-Соломона на порядки, поскольку количество символов – n на порядки превышает их разрядность – k.

Выводы

Разработан метод обнаружения и коррекции “пачки” ошибок, вызванных внешней помехой в каналах с кодово-импульсной модуляцией. Основным достоинством предложенного метода по сравнению с существующими способами коррекции “пачки” ошибок является малая вычислительная сложность реализации контроля и исправления ошибок. Проведенный теоретический анализ и результаты моделирования показали, что по сравнению с кодами Рида-Соломона, предложенный метод обеспечивает примерно в 8 раз большую скорость кодирования-декодирования и на 2 порядка меньшее время, требующееся на исправления “пачки” битовых искажений при одинаковом количестве контрольных разрядов.

Эффективность предложенного метода обусловлена тем, что он, в отличие от универсальных методов, учитывает особенности возникновения ошибок в каналах с кодово-импульсной модуляцией и, фактически, является специализированным. Метод ориентирован на аппаратную реализацию контроля ошибок. Метод может быть эффективно использован в скоростных каналах обмена информацией между компонентами компьютерных систем управления, для которых основной причиной ошибок являются внешние помехи.

Список литературы

1. Скляр Б. Цифровая связь. Теоретические основы и практическое применение. М.: Издательский дом ”Вильямс”.- 2004.- 1104 с.
2. Турченко Ю. А. Использование арифметических взвешенных контрольных сумм для коррекции ошибок в каналах со спектральной модуляцией // Вісник національного технічного університету України ”КПІ”. Інформатика, управління та обчислювальна техніка. – 2009.- № 50.- С. 106-112.
3. Федоречко О.І. Корекція “пачки помилок” в каналах з імпульсно-кодовою модуляцією // Матеріали XIII-ї Міжнародної науково-технічної конференції “Системний аналіз та інформаційні технології” – Київ, НТУУ “КПІ”, ІПСА.– 2011. – с. 516.

УДК 681.3.06

ПУСТОВАРОВ В.І.

МЕХАНІЗМИ ВКЛЮЧЕННЯ ЗАСОБІВ ФОРМАЛЬНИХ СПЕЦИФІКАЦІЙ ДО БАЗ ЗНАНЬ КОМ'ПЮТЕРНИХ МОВ

Розв'язання задач підвищення рівня автоматизації сучасних систем програмування і верифікації розроблених програмних засобів вимагає збереження специфікацій типів, проблемних галузей, програмних об'єктів та критеріїв ефективності реалізацій програм. Для реалізації задач автоматизованого визначення типів та формального синтезу, верифікації і формування виконавчих кодів пропонується відображення різних потрібних елементів специфікацій комп'ютерних мов в різних форматах графів подання мов та сучасних засобів мультимедіа. Використання таких представлень забезпечує гранично ефективне звертання до баз знань попередніх рішень при доказовій побудови нових і вдосконалених програм і моделей.

Rising of level for the complex task decision in hardware/software co-design systems require storage of types, restriction, relations and affectivity criterion specifications. For implementation of automated type definition task during formal verification and code, there are proposed using of different element of computer and multimedia languages or specifications. It is shown, that application of such representations permit to organize effective processing of CAD knowledge bases and their using for proved synthesis of advanced programs and models.

Вступ

Системи автоматизованого програмування і моделювання (САПМ) [1] для комп'ютерного розв'язання задач при визначенні типів даних спираються на табличну організацію вбудованих структур даних, їх примірників та методів обробки. Таблиці САПМ зберігають характеристики типів аргументів та результатів програмних і апаратних ресурсів розв'язання задач (РРЗ) в сучасних засобах їх оптимізації, та формального синтезу і верифікації [3, 8, 9]. Базовим підходом до доказової автоматизації проектування ефективних РРЗ стало використання баз знань (БЗ) семантичного супроводження [3, 7] моделей з використанням реляційних сховищ даних в складі САПМ РРЗ.

Фактично мови програмування і моделювання мають явні структуровані специфікації типів користувача та даних та неявні, визначені стандартами для семантичної частини мови специфікації вбудованих типів. Більш потужні засобами формальної специфікації РРЗ включено до таких мов, як Z і VDM [10], що базуються на поширених математичних позначеннях і часто визначають задачі в таких спеціальних галузях математики, як теорії множин, чисел та типів, реляційна алгебра, тощо. Найближча перспектива використання описів формальних специфікацій і процесів їх перетворення на коди програм полягає у автоматизованому розв'язанні ряду задач:

- **верифікації програм і моделей** нормальними методами перевірки коректності шляхом взаємних перетворень специфікацій та кодів;
- **вибірки оптимальних програм і моделей алгоритмів** за критеріями, заданими спеціальними конструкціями;
- **частково автоматизованого синтезу моделей і програм** перетвореннями зв'язків і обмежень формальних специфікацій;
- **пошуку реалізацій програм і моделей алгоритмів** в базах БЗ САПМ за образами специфікацій різних рівнів;
- **формування і накопичення показників складності та ефективності** в критеріях оптимального розв'язання задач;
- **впорядковано зберігання примірників реалізації** РРЗ за ключовими елементами специфікацій;
- **автоматизованої ревізії та формування протоколів доведення** відповідності програм формальним специфікаціям та доказового синтезу програм.

Головну проблему використання цих засобів складає обмеженість реалізованих рівнів комп'ютерної обробки цих специфікацій, які звичайно обмежуються в автоматичному режимі лише їх перевіркою на суперечливість і дозволяють ручні адекватні перетворення для розв'язання суміжних задач [8]. Іншу проблему складає необхідність практично необмеженого розширення можливих зв'язків і описів специфікацій і кодів при розв'язанні задач в розширюваних предметних галузях досліджень (ПГД). Для цього треба забезпечити специфікацію окремих структурних одиниць фрагмента про-

грами, як змінних або об'єктів (підпрограми або міні-модуля), так і всієї предметної області у вигляді батьківського макромодуля.

В специфікаціях, включених до модулів на комп'ютерних мовах, треба розділити визначення типів, що відповідають математичним координатам, від специфікацій зв'язків, цілей, обмежень та обчислювальних дій.

Пошук відповідності між цільовими специфікаціями та реалізаціями програм повинен спиратися на ключі, на яких залежно від задачі проектування (синтезу, верифікації або оптимізації) використовують різні відношення порядку. Технології автоматизованого проектування та оптимізації програм передбачають збереження специфікацій програм та їх шаблонів в БЗ [3, 8] і доступ до них за ключами, типи яких наведені в табл. 1.

Табл. 1. Типи узагальнених ключових елементів управління в БЗ САПМ

Мови / ***Мовні об'єкти***	***Математичні мови специфікацій моделей***	***Проектування програм та обладнання***	***Моделювання за послідовностями подій***	***Визначення режимів проектування***	***Комп'ютерні мови специфікацій задач***	***Текстові описи моделей в документах***
Вхідні дані та аргументи	Списки типізованих імен в структурах	Списки типізованих імен в структурах	Списки імен даних управління	Параметри джерел даних і оптимізації	Списки імен типізованих структур	Текстові описи змісту входів
Проміжні дані	Імена типізованих примірників і агрегатів	Імена примірників і агрегатів	Імена примірників і агрегатів	Параметри примірників і агрегатів	Імена примірників і агрегатів	Текстові описи примірників і агрегатів
Результати	Списки типізованих імен	Списки типізованих імен	Списки імен впливів	Імена приймачів даних	Списки типізованих імен	Текстові описи виходів
Операції	Вирази прототипів зв'язків та обмежень	Вирази прототипів заголовків	Прототипи заголовків і умов запуску	Прототипи заголовків та шаблонів	Прототипи заголовків та шаблонів	Текстові описи позначень і аргументів
Функції	Вирази прототипів заголовків і зв'язків	Вирази прототипів заголовків	-	Прототипи заголовків та шаблонів	Прототипи заголовків та шаблонів	Текстові описи імен і аргументів
Підпрограми	Прототипи заголовків, зв'язків і обмежень	Прототипи заголовків і зв'язків	Прототипи заголовків і умов запуску	Прототипи заголовків та шаблонів	Прототипи заголовків та шаблонів	Текстові описи імен і аргументів
Оператори мови	Прототипи заголовків, зв'язків і обмежень	Передумови Післяумови	Прототипи заголовків або шаблонів	Прототипи заголовків або шаблонів	Прототипи заголовків або шаблонів	-

Проблема реалізації змісту комп'ютерних мов полягає в уніфікації дій семантичної обробки комп'ютерних і природних мов для даних основних типів, включаючи безпосередні обчислення, пояснення та адекватні взаємні перетворення. Така уніфікація семантики або змісту дозволяє визначити практично всі комп'ютерні мови і проторувати шлях до створення загальної реалізації обробки семантичної частин всіх комп'ютерних мов.

Ще одна проблема САПМ пов'язана з аналізом і відтворенням образів синтаксису всіх розділів кодів і специфікацій в текстовій, символічній (ієрогліфічній) або структуровано-графічній формі [5]. Вона пов'язана з необхідністю відображення інформації про модулі перед користувачами САПМ різних рівнів і розв'язується на основі використання форм уніфікованого семантичного кодування елементів програм і моделей [6].

Формалізація постановки задачі

Неформальною метою роботи є визначення механізмів доступу до БЗ САПМ при проектуванні РРЗ різних рівнів організації і складності. Для цього потрібно визначити механізми побудови ключів доступу до змістовних та виконавчих характеристик РРЗ через вбудовані відношення порядку ключових елементів. Найпростіші відношення порядку утворюються узагальненими синтаксичними або морфологічними позначеннями назв задач і типів зі значеннями їх аргументів і результатів. Синтакси-

чні ознаки представлень РРЗ на різних рівнях моделей задаються стандартними позначеннями. До них відносяться стандарти символьних імен конструкцій та стандартних підпрограм і функцій РРЗ та їх аргументи у стандартному порядку. Звичайно ці позначення відповідають стандартним математичним позначенням, що дозволяє прискорювати пошук користувачем потрібних РРЗ в БЗ.

Однак впорядкування за синтаксисом лексикографічних або мультимедійних символів не дозволяє систематизувати та згрупувати разом РРЗ однієї ПГД для локального співставлення споріднених рішень. Більш ґрунтовну систематизацію РРЗ в БЗ, що сприяє доведенню їх відповідності специфікаціям, можна організувати на основі використання змістовних характеристик елементів ПГД як ключів в сховищах зберігання. Змістовні або семантичні характеристики РРЗ в назвах значень та їх типів у прив'язці до ПГД, де вони визначені. За аналогією з семантичним кодуванням текстів [6] семантичні характеристики іменованих або ідентифікованих позначень включають кодові поля ідентифікації відповідного рівня ієрархічних ПГД, показані в табл. 2, та коди деталізації полів ролі та стану відповідного елемента семантики іменування даних методів та об'єктів.

Табл. 2. Рівні представлення семантичних ознак в ключових іменах БЗ САПМ

Кодування / Ключові елементи	***Уніфіковані представлення моделей і програм***	***Уніфіковані представлення мов специфікації***	***Уніфіковані представлення математики***	***Описи моделей на природних мовах***	***Мультимедійні представлення моделей***	***Узагальнене семантичне кодування моделей***
Назви ПГД та її елементів	Набори імен типів в описах ПГД	Набори імен ПГД типів даних і об'єктів	Набори імен ПГД та її координат	Текстовий опис назви і змісту ПГД	Мультимедійні позначення ПГД	Ієрархія кодів словників і змісту ПГД
Назви координат	Числових і абстрактних типів	Числових типів і об'єктів	Типів математичних даних	Текстові назви типів	Зразки відтворення типів	Коди назв і змісту типів
Назви зв'язків координат	Назви блоків контролю обмежень	Назви виразів обмежень і зв'язків	Назви формул обмежень і зв'язків	Текстові назви зв'язків і обмежень	Мультимедійні символи зв'язків	Семантичні коди назв зв'язків
Назви типізованих РРЗ	Примірників і агрегатів РРЗ з аргументами	Примірників і агрегатів РРЗ з аргументами	Специфікацій примірників і агрегатів РРЗ	Текстові описи примірників і агрегатів	Мультимедійні символи примірників	Семантичні коди назв і агрегатів РРЗ
Назви аргументів і змінних	Типізованих аргументів і змінних	Типізованих аргументів і змінних	Типізованих аргументів і змінних	Текстові описи аргументів і їх типів	Мультимедійні символи аргументів	Семантичні коди аргументів і змінних
Назви зв'язків і обмежень	Контрольних виразів зв'язків і обмежень	Контрольних виразів зв'язків і обмежень	Контрольних виразів зв'язків і обмежень	Текстові описи зв'язків і обмежень	Мультимедійні позначення зв'язків	Семантичні коди виразів зв'язків
Назви критеріїв складності	Об'ємних, часових, комбінованих	Об'ємних, часових, комбінованих	Об'ємних, часових, комбінованих	Текстові описи критеріїв складності	Мультимедійні символи складності	Семантичні коди виразів складності
Назви критеріїв коректності	Умов збіжності та точності	Умов збіжності та точності	Умов збіжності і точності	Текстові описи умов збіжності	Мультимедійні символи точності	Семантичні коди виразів точності

Якщо назви або ідентифікатори побудовані за так званою угорською схемою іменування, то елементи імені послідовно показують належність до об'єктів або їх груп за іменами, діями та, за необхідності, режимами використання. Імена окремих полів можуть формуватися на основі слів природних мов, відзначених номерами однотипних об'єктів, а для можливості реконфігурації імен доцільно використовувати поля зміщення відносно канонічної позиції при відтворенні [8]. Імена спочатку прив'язуються до типа об'єкта, а потім включають номери та змістовні коди елементів об'єкта. При визначенні ПГД для проектування слід розрізняти їх фізичний зміст та їх математичний зміст з абстрагуванням від фізики значень та одиниць їх виміру. Для розв'язання задач достатньо мати ***повну*** математичну модель і систему пояснень та інтерпретації операцій та операторів і відтворення картин ПГД і результатів.

Опис математичної моделі ПГД включає у своєму заголовку назву ПГД, списки визначень координат $\mathbf{X} = \{X_1,\ldots, X_k,\ldots, X_{k\max}\}$, де k – номери координат або їх однорідних агрегатів X^r_k, де r – мірність агрегатів координат, які утворюють підпростори. Кожний опис координати X_k включає визначення множини або домену припустимих значень dom X_k, а при конкретизації ПГД розширяється специфікаціями фізичних одиниць виміру $u(X_k)$.

Наявність математичних відношень і зв'язків координат $R(X_k)$ скорочує обсяг простору значень і множину можливих варіантів рішень, створюючи обмежені простори і фрактали як простори однорідних координат $X^r{}_k$ з дробовою мірністю r. Таким чином, при визначенні координат конкретної ПГД важливо забезпечити їх несуперечність, тобто наявність непорожніх наборів сумісних значень координат, що дозволяють визначати комплекси дійсних (не уявних) об'єктів ПГД.

Більшість задач в рамках будь-якої ПГД розв'язується відносно окремих примірників об'єктів ПГД, а також картин і сцен, що формуються і відбуваються над комплексами об'єктів. Картини і сцени ПГД утворюються базовими примірниками об'єктів та їх агрегатами елементів. Для доведення коректності списки внутрішніх зв'язків, відношень та обмежень базових об'єктів і агрегатів доцільно розмістити в спеціальних розділах описів. Списки заголовків РРЗ базовими мареметичними методами або списки заголовків правил доведення повинні скласти основу ключів пошуку в БЗ потрібних реалізацій РРЗ та шляхів доведення їх відповідності специфікаціям типів, примірників і зв'язків.

Спираючись на гіпотезу, що будь-яка задача над об'єктами будь-якої складності може розв'язуватися у межах координатних просторів, що включають геометричний та/або топологічний підпростори, підпростори розвитку, які покривають координати динамічного континууму. При повному описі їх доцільно деталізувати через підпростори фізичних координат з визначеними відношеннями порядку та одиницями вимірів. Таким чином, головною задачею цієї роботи стало ***створення набору конструкцій комп'ютерної мови специфікацій*** C_j ***та структур їх подання як вузлів дерева*** $\mathbf{T}$, що включає семантично значимі вузли $V_0,\ldots, V_{li},\ldots, V_{l\max i\max}$, зв'язані в граф, названий в [8] графом алгоритму, який відображує підлеглість операторів і операцій [4], узагальнений для конструкцій опису і специфікації даних. Таке подання повинно однозначно представляти типи, що визначають координати і примірники даних з додержанням відношень семантичного порядку принаймні для канонічних форм представлення. Змістовна обробка операторів опису C_j повинна надавати повну інформацію про характеристики типів даних та їх примірників для розв'язання задач аналізу, синтезу та генерації виконавчих кодів в межах обраної ПГД.

Основні конструкції і структури розділів специфікацій комп'ютерних мов

Конструкції комп'ютерних мов з суворою типізацією звичайно спираються на базовий стиль декларації об'єктів мови. Наприклад, мова C/C++ використовує ім'я типа як попередника до описів його примірника, а мова Pascal – як відповідного наступника. В той самий час конструкції описів типів мають деякі різні деталі (роздільники або ключові слова). Такі самі стилі можна зберегти і при описі ПГД і в додаткових розділах описів класів і методів об'єктів.

Більшість стандартних конструкцій і прийомів об'єктно-орієнтованих мов програмування зручно використати при побудові розширень для представлення специфікацій. Так для опису просторів ПГД зручно вико ристовувати механізм абстрактних класів з успадкуванням базових класів або типів, що описують координати ПГД. До опису будь-якого класу (координати і цілої ПГД) можна додати спеціальний розділ обмежень **`restriction`**, що представляє у вигляді логічних зв'язки, які притаманні координатам або полям класу і обмежують домени значень, та обмеження, характерні для окремих груп задач в ПГД.

Опис ПГД в формі специфікації включає її іменування та описи координат або типів, а також заголовки або шаблони функцій, тобто цей опис визначається як спеціальний клас об'єкта ПГД з супроводженням типів, які визначають координати елементів ПГД. Таким чином, визначення ПГД не потребує включення кардинально нових синтаксичних конструкцій. Описи сцен та дій в ПГД додатково включають специфікації примірників об'єктів і зв'язки обмежень їх координат, тощо. Таким чином, їх можна побудувати як класи об'єктів, успадкованих від класу ПГД з наступною деталізацією або додаванням нових об'єктів.

Вибір деревоподібної структури для узагальненого внутрішнього подання комп'ютерних мов

Спочатку приділимо увагу внутрішньому поданню виконавчої частини комп'ютерної мови. Найбільш поширеними деревоподібними структурами в трансляторах є дерева син таксичного розбору, що формуються синтаксичним аналізатором і двійкові спрямовані ациклічні графи (САГ), що визначають піддерева підлеглих виразів комп'ютерних мов [1, 9]. Для розв'язання задач семантичної обробки ряд авторів [8] використовує так звані графи алгоритмів, які можна розглядати як проміжні форми між раніше описаними видами дерев та графів і цільовою формою, що має мінімальне різноманіття вузлів операцій та частин операторів.

Для раціональної організації семантичної обробки комп'ютерної мови слід сформулювати набір вимог, які визначатимуть порядок обробки графів. Ці вимоги стосуватимуться механізмів формування та обробки типів даних у виконавчих та декларативних конструкціях мов:

- **у вузлі, що відповідає імені або константі,** важливо зберігати код типа даних іменованого або константного об'єкта мови та проміжного результату;
- вузли графа повинні дозволяти подання **елементів будь-яких виконавчих та специфікаційних конструкцій**;
- важливо використовувати кодування типів, що дозволяє зберігати в таблицях **однозначні коди типів та узагальнені коди для споріднених типів**;
- бажано мати **однакові структури для пошуку ключових елементів вузлів дерева в БЗ,** щоб мінімізувати розміри таблиць відповідності для семантичної обробки.

Через дві останні вимоги для семантичної обробки більш зручні двійкові графи. Щоб якомога краще задовольнити перелічені вимоги як основу внутрішнього подання використаємо різновид графа алгоритму, який будемо називати **графом підлеглості** операцій та елементів операторів. Для математичних виразів комп'ютерних мов він повністю співпадає з базовою формою САГ [4] і потребує узагальнення для операторів специфікацій і структурного програмування. Основні правила побудови та організації цього графа можна сформулювати наступним чином:

- Всі знаки операції та базові ключові слова операторів відображуються в графах внутрішнього подання бінарними зв'язками.
- Відображення унарних операцій включає порожній зв'язок на місці відсутнього аргументу.
- Тернарні операції та оператори структурованого програмування також розбиваються на підграфи відповідно порядку їх виконання;
- Індексування елементів масиву розглядається за аналогією з двомісною операцією "[]" мови C++, лівий аргумент якої визначає масив, а правий – номер його елемента. Так само описи та виклики функцій і розділення заголовка і тіла блоку розглядаються як узагальнені операції "()" і "{}".
- Послідовність обробки однорідних підграфів вузлів визначення елементів списків можна розглядати як узагальнення обробки допоміжної операції кома "," і порожнього оператора ";" в мові C/C++.
- Для вузлів передачі управління операторів **`break`**, **`continue`**, **`return`** та **`goto`** передбачаються зв'язки управління на вузли наступної обробки.
- Будь-який підграф має єдиний кореневий вузол, який звичайно переглядається або обробляється першим або останнім в блоці структурного програмування (вхід до внутрішніх вузлів блоку повз кореневий вузол вважається некоректним).

Первинний граф підлеглості доцільно будувати з дерев синтаксичного розбору через **дерева або графи підлеглості операцій та конструкцій** [4]. В таких деревах повинні повторюватися всі підграфи з однаковими діями, а в графах повторні піддерева замінюються посиланнями на їх першу появу. На рис. 1 показано приклад побудови графа підлеглості конструкцій для наступного опису даних на мові C/C++: **`const struct`** `rec rc, *prc; …`

В подібних операторах описів та специфікацій типів і класів даних та абстрактних типів і класів необхідно застосовувати допоміжний кореневий вузол визначення типу, який посилається піддерево визначення типа і має спеціальний тип вузла `UsrDefTp`. Оператори явного визначення типів (**`type`** мови Pascal і **`typedef`** мови C/C++) використовують вузли базових ключових слів як корені для посилань на визначення застосованих в заголовках описів іменованих типів користувачів .

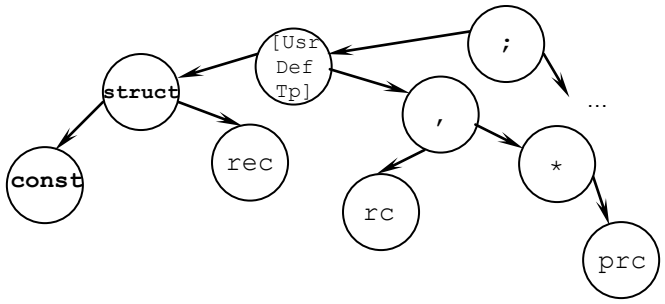


Рис. 1. Дерево оператора визначення даних типа struct
const struct rec rc, *prc;…

Описи операцій, функцій, класів та їх шаблонів в мові C/C++ розділяються на дві частини: описи заголовків та описи тіла визначення. Описи заголовків бібліотечних підпрограм повинні використовуватися як ключі при пошуку потрібних функцій. При формуванні графа мови C/C++ кореневий вузол визначається дужками тіла, ліве підлегле піддерево визначає типізований результат і список типізованих аргументів підпрограми, а праве піддерево – оператори примірника реалізації. Дерево специфікації для початку визначення прикладу функції **int** main(**int** argc, **char***argv[]){initHsh(thStud);…}… показано на рис. 2. Таким чином, дерева визначення підпрограм мають чітко виділений заголовок, який формує ключ пошуку в БЗ.

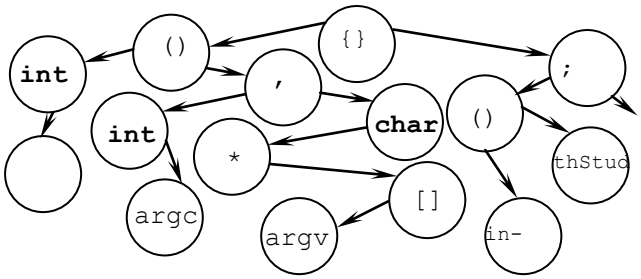


Рис. 2. Дерево оператора визначення функції
int main(**int** argc, **char***argv[]){initHsh(thStud);…}…

Специфікації визначень класів і розширених типів користувача відображується аналогічно визначенню функцій з чітко виділеним заголовком. При цьому важливо включити до опису класу чи типу всі наявні елементи типів примірника та заголовки методів, включених до класу чи типу. Такі додаткові розділи специфікацій як розділ обмежень **restriction** також повинні входити для ключів визначення підпрограм, класів або для описів ПГД. Таким чином, частини обмежень типів, класів і підпрограм повинні бути приєднані до заголовків для полегшеного пошуку в БЗ.

При комп'ютерному зберіганні Z-специфікацій необхідно насамперед визначити механізми деревоподібного відображення таких спеціальних конструкцій, як схеми, в яких спеціальні вузли типа ScmStr відділяють піддерево заголовку, яке використовуватиметься як ключ, від піддерева тіла схеми. Для інших спеціальних позначень Z-специфікацій треба визначити спеціальні вузли теоретико-множинних операцій і спеціальних позначень, зв'язків, серед яких лише деякі, що визначають тип використання підпрограми та її аргументи використовуються в ключових частинах визначень.

Особливості визначення структури ключових полів БЗ для пошуку специфікацій типів, даних і задач

Оскільки процес створення РРЗ пов'язаний з поданнями специфікацій, програм, моделей і процесів доведення [7] важливо використати узагальнений механізм формування семантики функціональних моделей обчислень і всіх потрібних оцінок їх складності.

Такий механізм створюється на базі узагальненого абстрактного типу аналітичного даних [5], в якому використовуються спеціальні канонічні форми зберігання [5, 8] для полегшення зіставлень варіантів рішень програм і доведень і впорядковане кодування [6] семантики імено-

ваних даних і операцій. Вимоги до канонічних форм подання кодів програм та протоколів верифікації [5] спираються на відношення лінійного порядку $\preccurlyeq$ [4] на базі кодів типів різних термінальних і нетермінальних вузлів дерев і графів підлеглості.

Висновки

Для розв'язання задач автоматизованої формальної верифікації і синтезу в системах програмування і моделювання доцільно розширити мови програмування засобами таких мов математичної специфікації як Z, представленими в стилі лінійних конструкцій мов програмування.

1. Для підвищення рівня автоматизації пошуку РРЗ за семантичними ознаками у вбудованій БЗ систем програмування складені ключі пошуку створюють назви цих ресурсів і списки типів аргументів або вхідних та вихідних даних операторів введення-виведення у складі відповідних предметних галузей.

2. Додатковими частинами ключових елементів пошуку потрібного РРЗ є розділи обмежень і критерії вибору модулів програм і моделей та їх складових частин, які дозволяють стати точно розпізнавати підрозділи БЗ, що орієнтовані на задані ПГД та обмеження, притаманні окремим випадкам задач.

2. Крім того узагальнені реалізації комп'ютерних мов і мов специфікації дозволяють організувати за єдиною методикою різні варіанти співставлень, впорядкувань та перетворень в системах програмування і моделювання, а також дають можливість одержати об'єктивно підтримані доведення тверджень і відповідності умовам в різних галузях математики, що може стати істотною комп'ютерною підтримкою при розробці математичних моделей.

3. Розширення комп'ютерних мов формальними специфікаціями координат ПГД як абстрактних типів зв'язків координат і змінних, а також обмежень координат і даних, відкриває можливість створення семантичних описів на базі природних мов і мов представлення різних видів мультимедійих даних, що надає можливість організації доступу до елементів РРЗ через більш зручні елементи текстових пояснень.

Список посилань

1. Ахо А., Сети Р., Ульман Дж. Компиляторы: принципы, технологии, инструменты: Пер. с англ. – М.: Издательский дом Вильямс, 2001. – 768 с.
2. Лисков Б., Гатэг Дж. Использование абстракций и спецификаций при разработке программ. Пер. с англ. М.: Мир, 1989. 424 с.
3. Пустоваров В.І. Побудова баз знань для сумісного створення програм і обладнання та їх формальної верифікації // Вісник НТУУ "КПІ". Інформатика, управління та обчислювальна техніка – К.: «Век+». 2008, 51, с. 41-46.
4. Пустоваров В.І., Коваленко С.Ю. Організація сховищ даних для мов опису та маніпуляцій з інформаційними сутностями // Вісник НТУУ "КПІ". Інформатика, управління та обчислювальна техніка – К.: «Век+». 2009, 51, с. 86 – 91.
5. Пустоваров В.І. Супроводження контролю відповідності програм і моделей формальним специфікаціям задач // Вісник НТУУ "КПІ". Інформатика, управління та обчислювальна техніка – К.: «Век+». 2007, 47, с. 269-279.
6. Пустоваров В.И. Семантически ориентированное кодирование моделей и его эффективное применение для решения задач искусственного интеллекта // Электронное моделирование – К.:, - 1998, 20, № 4. с.32-42.
7. Hehner E.C.R. Practical theory of programming. Springer-Verlag, New York, 1993 – 243 p.
8. Metzger R.C., Zhaofang W. Automatic algorithm recognition and replacement: a new approach to program optimization / The MIT Press, Cambridge, 2000. 219 p.
9. Muchnick S.S. Advanced compiler design and implementation – San Francisco, Morgan Kaufmann Publishers, 1997. – 856 p.
10. Woodcock J., Davies J. Using Z. Specification, Refinement, and Proof. C.A.R. Hoare Series editor, 1995 – 390 p.

УДК 004.032.24:519.614.2

ГРИБЕНКО Д.В.,
СТІРЕНКО С.Г.

РЕАЛІЗАЦІЯ РОЗВ'ЯЗАННЯ ЗАДАЧІ ЗНАХОДЖЕННЯ ВЛАСНИХ ЧИСЕЛ ТА ВЕКТОРІВ МЕТОДОМ ЛАНЦОША НА КЛАСТЕРНІЙ СИСТЕМІ

В статті розглянуто супутні проблеми, що виникають при реалізації методу Ланцоша на ЕОМ, а саме: вибір способу зберігання розрідженої матриці, оптимізація коду за допомогою застосування SIMD команд сучасних процесорів. Виконано реалізацію алгоритмів множення матриці на вектор в різних форматах та їх експериментальне порівняння. Розроблено програмне забезпечення знаходження власних чисел та власних векторів дійсних симетричних розріджених матриць для кластерної обчислювальної системи. Застосовано бібліотеку MPI для передачі повідомлень та бібліотеку LAPACK для пост-обробки результатів алгоритму Ланцоша. ПЗ підтримує введення матриць в форматі Matrix Market, що забезпечує сумісність з існуючими пакетами чисельних обчислень.

This article discusses questions that arise during implementation of Lanczos method for clusters: choice of sparse matrix storage format and code optimization with SIMD instructions of modern CPUs. Discussed algorithms of sparse matrix-vector multiplication have been implemented and an experimental comparison was carried out. A parallel cluster eigenvalue and eigenvector solver cluster for real symmetric sparse matrices was developed. It uses MPI for communication and LAPACK for post-processing. It supports the de-facto standard Matrix Market format.

Вступ

Типовою задачею лінійної алгебри є знаходження для заданої матриці $A \in R^{n \times n}$ власних чисел $\{\lambda_i\}$ та відповідних власних векторів $\{x_i\}$, або, іншими словами, розв'язків рівняння $Ax = \lambda x$.

Необхідність знаходження заданої кількості найбільших за абсолютною величиною власних чисел та відповідних векторів розріджених матриць великої розмірності виникає, наприклад, при дослідженні коливань у фізиці. [1] Коливання пружних конструкцій, коливань в аеродинаміці, коливання в електричних колах, коливання молекул та атомів у фізиці елементарних часток – всі ці задачі є різноманітними прикладними аспектами однієї і тієї самої математичної задачі.

Іншою прикладною задачею, що потребує знаходження власних векторів великих розріджених матриць є задача ранжування результатів веб-пошуку, одним із способів вирішення якої є визначення міри центральності вузлів графу веб-сторінок. [2] Якщо задати граф G, вузлами якого є веб-сторінки, а ребрами якого є гіперпосилання, то міра центральності кожного вузла буде характеризувати імовірність того, що людина, читаючи веб-сторінки та переходячи по гіперпосиланням, опиниться на відповідній веб-сторінці. Так, алгоритм PageRank, що використовується пошуковою системою Google, базується саме на визначенні міри центральності кожної веб-сторінки (яку розробники назвали PageRank цієї веб-сторінки). Звичайно, кількість веб-сторінок в мережі Інтернет є дуже великою (за оцінками Google [3], в 2008 році кількість веб-сторінок в мережі Інтернет досягла 10^{12}), але кожна веб-сторінка має порівняно невелику кількість гіперпосилань (до сотень), тому матриця суміжності графу G буде розрідженою.

Наївні методи знаходження власних чисел є простими для реалізації, але мають велику обчислювальну складність. Універсальні методи, такі як QR-розклад, також мають велику обчислювальну складність, але є обчислювально стійкими.

Найбільш ефективними та обчислювально стійкими серед ітераційних методів є проекційні методи, особливо ті з них, що виконують проекції на підпростори Крилова. Іншою перевагою цих методів є можливість ефективного розпаралелювання. Так, якщо матриця A має деякі властивості, а саме, є дійсною, симетричною та розрідженою, метод Ланцоша має значні переваги з точки зору обчислювальної складності над іншими методами, що використовують підпростори Крилова.

Метод Ланцоша для знаходження власних чисел

Метод Ланцоша призначений для знаходження власних чисел симетричної матриці. Вперше був опублікований в 1950 році, [4] практичне застосування було розпочато після

Вхідні дані : $A \in \mathbb{R}^{n\times n}$, причому $A = A^T$, m
Вихідні дані: $Q \in \mathbb{R}^{n\times m}, T \in \mathbb{R}^{m\times m}$

```
1  begin
2      for i ← 1 to n do
3          q_{1,i} ← rand() ;                          // n разів по 1 операції
4      end
5      q_1 ← q_1 / sqrt(⟨q_1, q_1⟩) ;                  // 3n операцій
6      β_1 = 0
7      q_0 = 0
8      for j ← 1 to m do
9          v_j = A q_j − β_j q_{j−1} ;                 // m разів по 2n + U операцій
10         α_j = ⟨v_j, q_j⟩ ;                          // m разів по 2n операцій
11         v_j ← v_j − α_j q_j ;                       // m разів по 2n операцій
12         β_{j+1} = sqrt(⟨v_j, v_j⟩) ;                // m разів по 2n операцій
13         q_{j+1} = v_j / β_{j+1} ;                   // m разів по n операцій
14     end
15 end
```

Рис. 1. Обчислювальний алгоритм методу Ланцоша

того, як були знайдені оцінки похибок. [5] З сучасної точки зору, в контексті знаходження апроксимацій власних чисел, метод Ланцоша є поєднанням методу Релея–Рітца, процесу Грама–Шмідта (або іншого методу ортогоналізації набору векторів) та підпросторів Крилова. [6]

Викладення усіх міркувань, що приводять до методу Ланцоша, є досить довгим та не доречним в даній статті, тому через брак місця наведемо тільки обчислювальну процедуру методу Ланцоша та стислий її аналіз. Вхідними даними є симетрична дійсна матриця A розміром $n \times n$ та необхідна кількість власних чисел m . Метод Ланцоша задає рекурентну послідовність, що визначає елементи тридіагональної матриці T_m, власні числа якої є наближеннями власних чисел матриці A. [4, 7] Введемо наступні позначення для матриці T_m: діагональні елементи позначимо $\alpha_1, \alpha_2, \ldots, \alpha_m$, позадіагональні елементи позначимо $\beta_2, \beta_3, \ldots, \beta_m$:

$$T_m = \begin{pmatrix} \alpha_1 & \beta_2 & 0 & 0 & 0 \\ \beta_2 & \alpha_2 & \beta_3 & 0 & 0 \\ 0 & \ddots & \ddots & \ddots & 0 \\ 0 & 0 & \beta_{m-1} & \alpha_{m-1} & \beta_m \\ 0 & 0 & 0 & \beta_m & \alpha_m \end{pmatrix},$$

Власне обчислювальний алгоритм у вигляді, зручному для реалізації на ЕОМ, представлений на рис. 1. Проведемо аналіз обчислювальної складності цього алгоритму. В нотатках справа від алгоритму показана обчислювальна складність кожного кроку. В рядку 9 вказана деяка величина U, тому що обчислювальна складність множення матриці на вектор залежить від того, чи є матриця щільною чи розрідженою, а в останньому випадку – ще й від способу зберігання.

У випадку, коли матриця A є щільною, множення матриці на вектор потребує $U = 2n^2$ операцій. Тоді обчислювальна складність алгоритму Ланцоша для щільних матриць становить $2mn^2 + 9mn + 4n = O(mn^2)$.

У випадку, коли матриця A є розрідженою, та при використанні відповідних ефективних способів зберігання матриці в пам'яті, множення матриці на вектор потребує $U = 2k$ операцій, де k – кількість ненульових елементів в матриці. Тоді обчислювальна складність алгоритму Ланцоша для розріджених матриць становить $9mn + 2mk + 4n = O(m(n + k))$.

Так як $k \gg n$ та $k \gg m$, то найбільше часу витрачається на виконання операції множення розрідженої матриці на вектор в рядку 9 (рис. 1). Тому оптимізація цієї виконання операції є першочерговим завданням для отримання високопродуктивної реалізації. Питання роботи з розрідженими матрицями, включаючи вибір форматів зберігання, аналіз алгоритмів множення розрідженої матриці на вектор, розпаралелювання цих алгоритмів, необхідно розглянути окремо в наступних розділах.

Табл. 1. Характеристики форматів зберігання розріджених матриць

Формат	Спеціалізований	Вимоги до пам'яті	Обчислювальна складність множення матриці на вектор	Додаткові операції при розпаралелюванні
Координатний	Ні	$3k$	$O(k)$	$n(p-1)$
CSR	Ні	$2k+n+1$	$O(k)$	–
CSC	Ні	$2k+n+1$	$O(k)$	$n(p-1)$
CSR для симетричних матриць	Ні	$2r+2n+1$	$O(r+n)$	$n(p-1)$
BCSR	Так	$n_b b^2 + n_b + \frac{n}{b} + 1$	$O(n_b)$	–

Вибір формату зберігання розрідженої матриці

При виборі методу зберігання розріджених матриць необхідно звернути увагу на наступні характеристики кожного методу:

- вимоги до пам'яті;
- складність побудови структур даних під час завантаження матриці в пам'ять;
- обчислювальна складність виконання необхідних операцій (в контексті реалізації методу Ланцоша – множення матриці на вектор);
- наявність можливості розподілу даних по вузлам кластерної системи (тобто, відсутність необхідності зберігати повну копію матриці в кожному вузлі) та можливість паралельного виконання необхідних операцій;
- чи виникає необхідність виконання додаткових обчислень після паралельного виконання операції для отримання фінального результату, їх обчислювальна складність;
- ефективність використання кеш-пам'яті при виконанні операцій.

Було розглянуто наступні формати: координатний, [8, 9] CSR (Compressed Sparse Rows), [10] CSC (Compressed Sparse Columns), [9, 10] модифікація формату CSR для симетричних матриць, BCSR (Block Compressed Sparse Row). [11]

В таблиці 1 наведені основні характеристики форматів зберігання розріджених матриць, причому використано наступні позначення:

- n – розмірність квадратної матриці;
- k – кількість ненульових елементів матриці;
- r – кількість ненульових елементів симетричної матриці, що лежать нижче головної діагоналі ($r < \frac{k}{2}$);
- b – розмір щільного блоку;
- n_b – кількість блоків, що мають хоча б одним ненульовим елементом;
- p – кількість процесорів.

Задача зберігання та обробки великих розріджених матриць сама по собі не є тривіальною; до цієї задачі часто неможливо застосувати традиційні прийоми, що застосовуються при обробці щільних матриць. При виконанні операцій над розрідженими матрицями складно досягти високої продуктивності обчислень (на рівні з продуктивністю операцій над щільними матрицями) з наступних причин:

- структури даних є більш складними;
- алгоритми виконання операцій є складними, так як оброблюють складні структури даних;
- при обході елементів матриці є необхідність застосовувати непряму адресацію пам'яті;

- продуктивність обчислень сильно залежить від вхідних даних (можна побудувати такі структури даних, обробка яких буде відбуватись повільніше, ніж в «середньому» випадку).

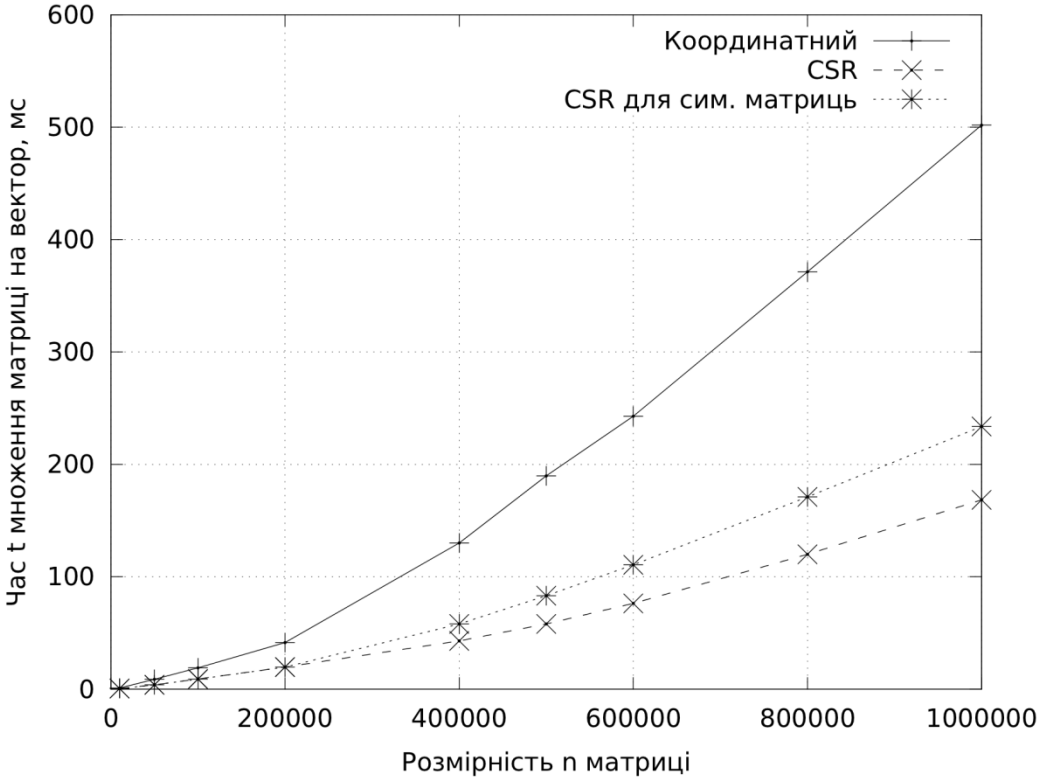


Рис. 2. Залежність часу множення розрідженої матриці на вектор від розмірності матриці при постійній середній кількості ненульових елементів у рядку (30)

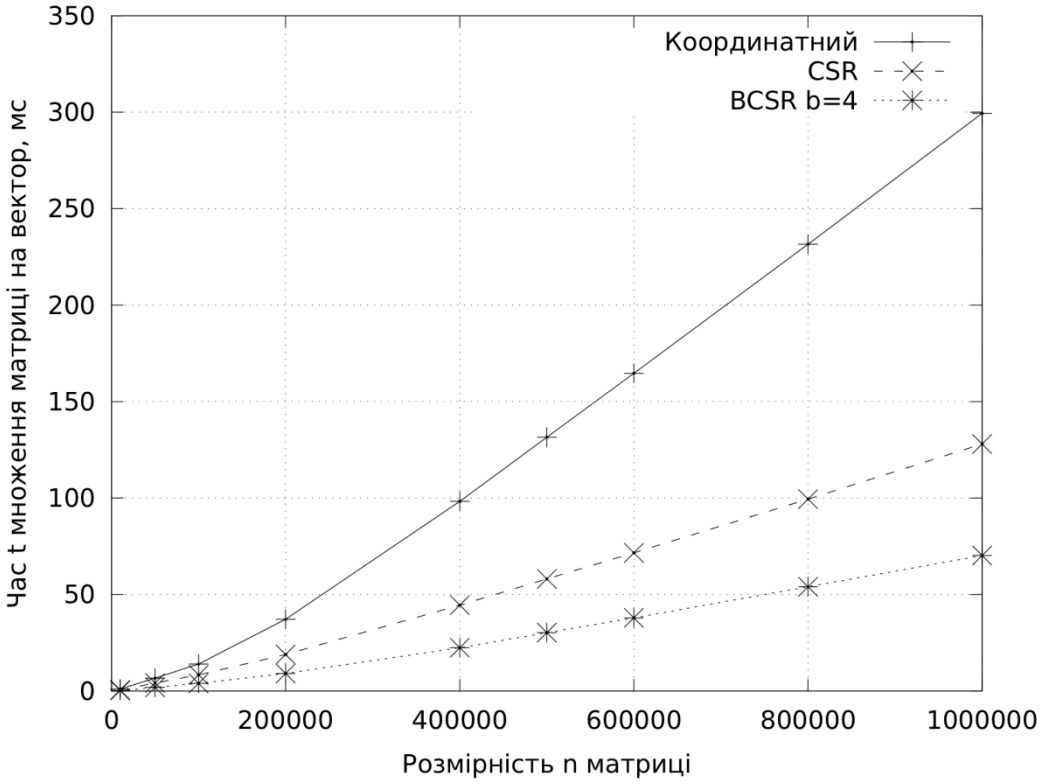


Рис. 3. Залежність часу множення розрідженої матриці з блочно-щільною структурою на вектор від розмірності матриці при постійній середній кількості ненульових елементів у рядку (44)

Експериментальне порівняння форматів загального призначення

Аналіз алгоритмів множення розрідженої матриці (в різних форматах) на вектор не дає кількісних оцінок для порівняння часу роботи. Тому доцільним є проведення експериментального порівняння реалізацій цих алгоритмів з метою визначити різницю в часі роботи та дати рекомендації для подальшої реалізації алгоритмів програмного забезпечення. Було виконано реалізацію послідовного варіанту вказаних методів зберігання розріджених матриць та відповідних алгоритмів множення матриці на вектор. Для вимірювання часу множення необхідні вхідні дані. Були згенеровані симетричні матриці розмірністю n =10, 50, 100, 200, 400, 500, 600, 800, 1000 тисяч без будь-якої характерної структури портрету з 30 ненульовими елементами (в середньому) в рядку.

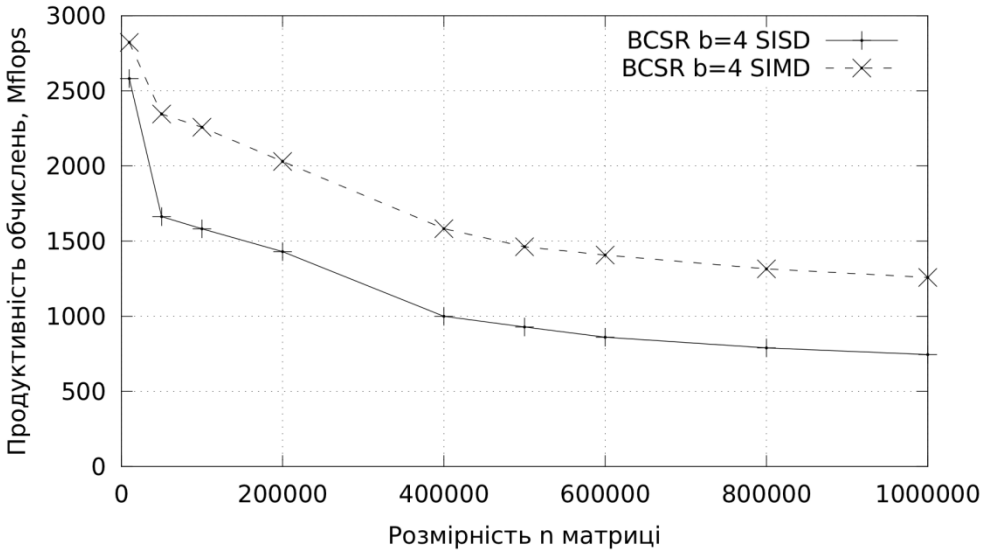


Рис. 4. Залежність продуктивності обчислень від розміру матриці для SISD та SIMD реалізацій операції множення матриці в форматі CSR на вектор

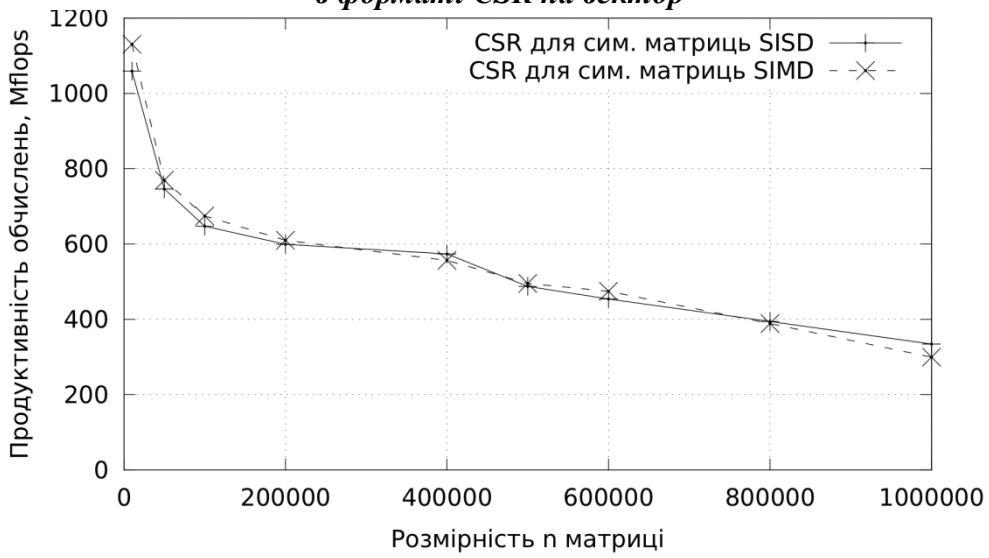


Рис. 5. Залежність продуктивності обчислень від розміру матриці для SISD та SIMD реалізацій операції множення матриці в форматі CSR для симетричних матриць на вектор

Реалізація кожного алгоритму множення була виконана 50 разів з вимірюванням часу, найгірший та найкращий час були відкинуті, а інші 48 результатів були усереднені. Графіки залежності часу виконання від розмірності матриці представлені на рис. 2.

Проаналізуємо ці результати. Так як кількість ненульових елементів $k = 30n$ в даному експерименті (тобто, k пропорційно n), то залежність часу виконання $t(k)$ пропорційна $t(n)$, що побудована на графіку (рис. 2). З графіку видно, що залежності часу виконання для координатного формату, CSR та його модифікації є досить близькими до лінійних, що є в згоді з теоретично обґрунтованою обчислювальною складністю $O(k)$. Найкращий час виконання в усіх випадках показала реалізація алгоритму для формату CSR.

Модифікація формату CSR для симетричних матриць дозволяє зменшити вимоги до пам'яті за рахунок збільшення часу обчислень.

Експериментальне порівняння форматів зберігання блочно-щільних матриць

Для зберігання блочно-щільних матриць можуть бути застосовані формати загального призначення (координатний, CSR) та спеціалізований формат BCSR. Генерування вхідних даних та вимірювання часу були проведені за методикою, аналогічною застосованій в попередньому пункті. Графіки залежності часу виконання від розмірності матриці представлені на рис. 3.

Ці графіки також є близькими до лінійної залежності, а тому є в згоді з теоретично обґрунтованою обчислювальною складністю $O(n_b)$ для алгоритму множення матриці в форматі BCSR. В порівнянні з реалізацією алгоритму для формату CSR, реалізація для формату BCSR показала в середньому в 1,37 разів менший час виконання, що є суттєвим покращенням.

Таким чином, застосування формату BCSR для блочно-щільних матриць є доцільним.

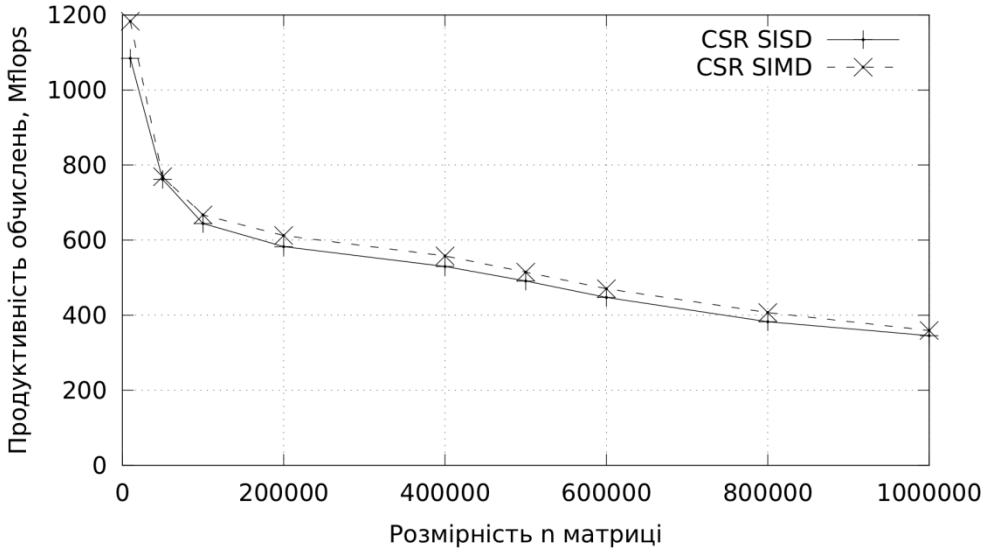


Рис. 6. Залежність продуктивності обчислень від розміру матриці для SISD та SIMD реалізацій операції множення матриці в форматі BCSR при на вектор

Оптимізація алгоритмів множення розрідженої матриці на вектор за рахунок застосування SIMD команд

В даній роботі найбільш цікавою є векторизація операцій множення матриці на вектор з декількох причин. По-перше, ця операція займає більшу частину часу обчислень за методом Ланцоша, і тому її необхідно оптимізувати в першу чергу. По-друге, алгоритми цих операцій є достатньо складними та не реалізуються SIMD командами тривіально.

Була виконана реалізація з використанням наборів команд SSE2, SSSE3, та SSE4 операцій множення матриці на вектор для форматів CSR, модифікації CSR для симетричних матриць, BCSR з фіксованими розмірами блоку $b = 2$, $b = 4$. З набору команд SSE2 було застосовано команди виконання арифметичних операцій, з набору команд SSE4 було використано команду обчислення скалярного добутку DPPD. Набори команд SSE та SSE4 включають команди для зчитування та запису в пам'ять даних в обхід кеш-пам'яті. Це дозволяє більш ефективно використовувати кеш-пам'ять, застосовуючи ці команди для завантаження тих даних, що під час обробки знадобляться тільки один раз.

Експериментальне порівняння SISD та SIMD реалізацій

На рис. 4–6 зображено експериментально отримані залежності продуктивності обчислень від розміру матриці для SISD та SIMD реалізацій операції множення для різних форматів зберігання матриць. Як видно з графіків, найбільший ефект застосування SIMD команд дало для формату BCSR, де підвищення продуктивності склало до 1,7 разів. Для формату CSR підвищення продуктивності становить до 7% Для модифікації формату CSR для симетричних матриць однозначного висновку зробити не можна, але для $n < 200000$ маємо підвищення продуктивності до 5%.

Методика тестування програмного забезпечення

В якості критерію оцінки та порівняння реалізацій алгоритму обрано коефіцієнт прискорення $K_П$ та коефіцієнт ефективності K_E.

Коефіцієнт прискорення показує, у скільки разів менше часу займає виконання паралельної програми в паралельній обчислювальній системі з P процесорами у порівнянні з виконанням послідовної програми в однопроцесорній системі:

$$K_П(P) = \frac{T_1}{T_P}.$$

Коефіцієнт ефективності показує середній рівень завантаження процесорів при виконанні програми в паралельній обчислювальній сис-

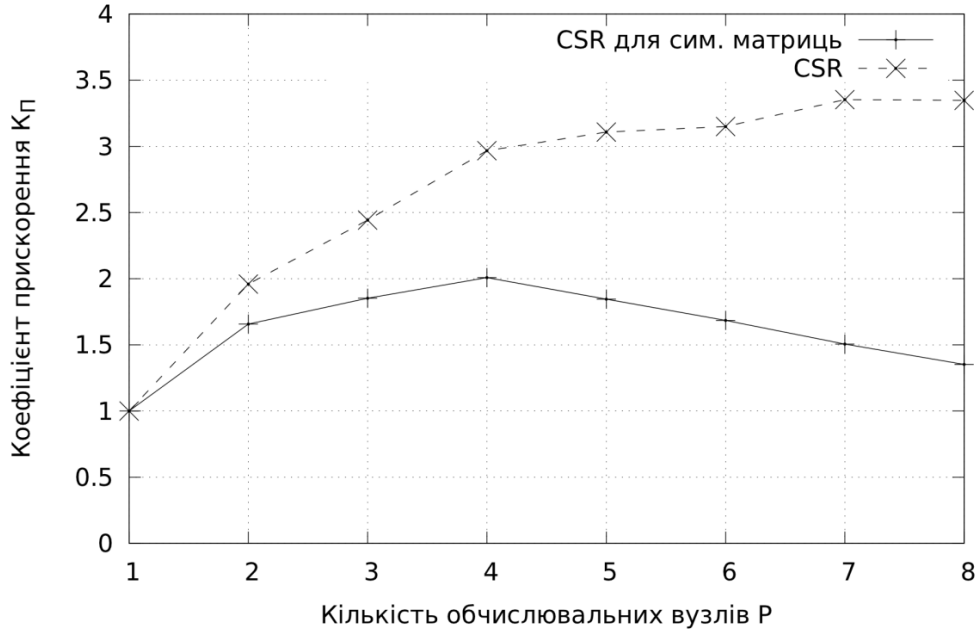


Рис. 7. Залежність коефіцієнту прискорення від кількості запущених задач, система «Б»

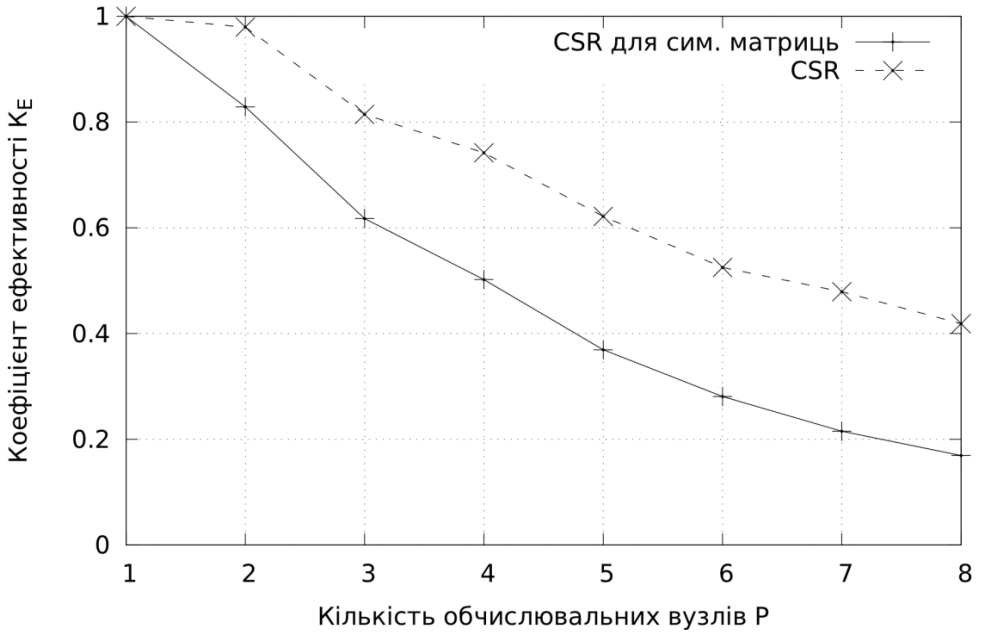


Рис. 8. Залежність коефіцієнту ефективності від кількості запущених задач, система «Б»

темі:

$$K_E(P) = \frac{K_П(P)}{P}$$

Для вимірювання часу виконання використовувався системний виклик clock_gettime(2).

Вимірювався час виконання 100 ітерацій алгоритму Ланцоша для матриці $n = 400000$, використовувався формат CSR.

Наведемо характеристики обчислювальних систем.

- процесор: Intel Core i5-750 (2,66 ГГц, 4 ядра, 8 Мб кеш-пам'яті третього рівня);
- оперативна пам'ять: DDR3 1600 МГц, 2 × 2048 Мб.

В якості програмного забезпечення використовувались:

- операційна система: Debian GNU/Linux «Sid» x86_64;
- компілятор C++: g++ 4.6.0.

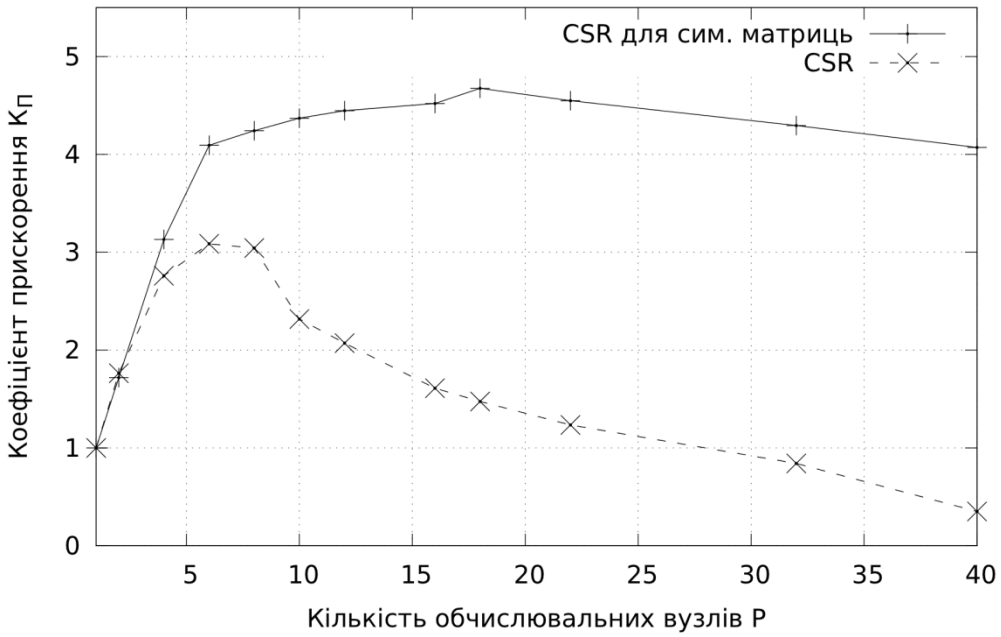


Рис. 9. Залежність коефіцієнту прискорення від кількості запущених задач, система «B»

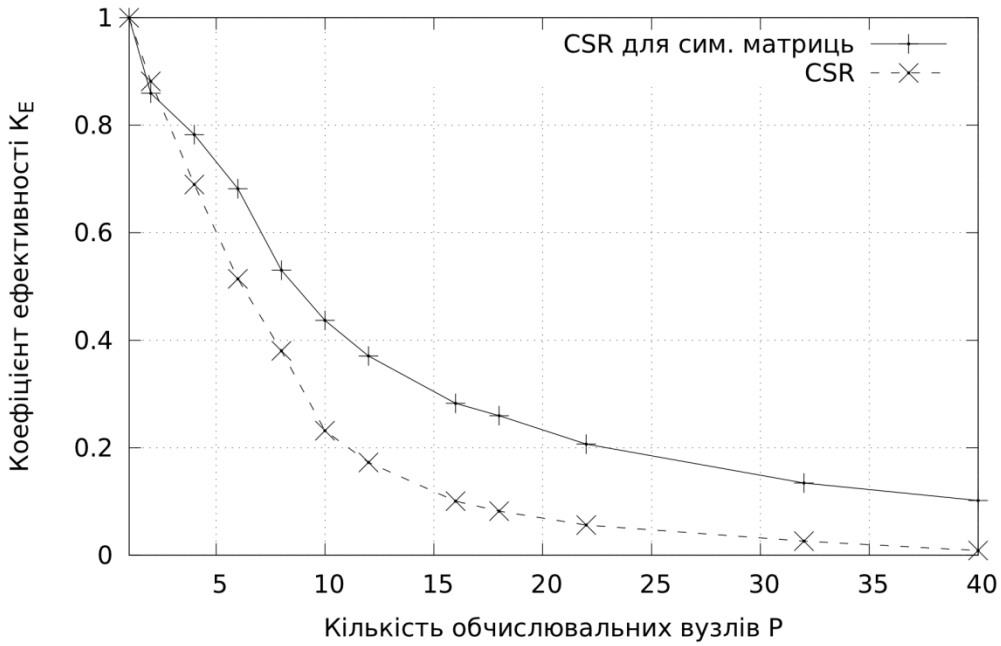


Рис. 10. Залежність коефіцієнту ефективності від кількості запущених задач, система «B»

Обчислювальна система «A» має наступне апаратне забезпечення:

Дана обчислювальна система використовувалась для тестування послідовних варіантів програм у розділах, представлених вище.

Обчислювальна система «Б» Для тестування паралельної реалізації методу Ланцоша використовувалась кластерна обчислювальна система Центру суперкомп'ютерних обчислень НТУУ «КПІ» з наступним апаратним забезпеченням вузла:

- процесори: два Intel Xeon E5345 (2,33 ГГц, 4 ядра, 8 Мб кеш-пам'яті другого рівня);
- оперативна пам'ять: DDR2 1333 МГц, 8192 Мб.

В якості програмного забезпечення використовувались:

- операційна система: CentOS release 5.5 x86_64;
- компілятор C++: g++ 4.6.0;
- реалізація MPI: OpenMPI 1.3.3.

Обчислювальна система «В» Обчислювальна система для пакетного запуску задач Intel Multicore Testing Lab має наступне апаратне забезпечення:

- процесори: чотири Intel Xeon E7-4860 (2,26 ГГц, 10 ядер, 24 Мб кеш-пам'яті другого рівня);
- оперативна пам'ять: DDR2 1333 МГц, 262144 Мб.

В якості програмного забезпечення використовувались:

- операційна система: RedHat Linux RHEL 5.4 x86_64;
- компілятор C++: g++ 4.5.1.

Результати тестування програмного забезпечення

Результати тестування на одному вузлі кластеру КПІ представлені на рис. 7–8. Максимальний коефіцієнт прискорення дорівнює 3,4 при запуску на 7 процесорах для формату CSR. Необхідно відмітити, що оптимальним можна вважати запуск на 4 процессорах, так як він показав $K_П \approx 3$ та високий коефіцієнт ефективності.

Результати тестування на Intel Manycore Testing Lab представлені на рис. 9–10. Максимальний коефіцієнт прискорення дорівнює 4,5 при запуску на 18 процесорах для формату CSR. Оптимальним можна вважати застосування 6 процесорів, коли спостерігається $K_П \approx 4$ та високий коефіцієнт ефективності.

За допомогою інструменту Intel VTune було встановлено, що вузьким місцем програми є швидкість каналу пам'ять-процесор. Це і є головним фактором, що обмежує можливість використання великої кількості процесорів даною програми. Це спостереження підтверджується наступним: в системі «Б» два процесори, кожен з яких має свій контролер доступу до пам'яті, а в системі «В» – чотири. Тому максимальні коефіцієнти прискорення при довільному доступі до пам'яті, що виникають у великій кількості при застосуванні формату CSR для симетричних матриць, дорівнюють приблизно двом та чотирьом відповідно. При застосуванні звичайного формату CSR довільних доступів до пам'яті значно менше, ніж послідовних, що дозволяє використати на свою користь попереднє завантаження даних в кеш, що відбувається автоматично при послідовному доступі до пам’яті.

Висновки

В даній роботі розглянуто не тільки реалізацію власне методу Ланцоша, але всі суміжні питання, що виникають під час створення реалізації для кластерних обчислювальних систем.

Експериментальне порівняння форматів зберігання з точки зору часу виконання операції множення матриці на вектор в послідовній системі показало:

- Правильний вибір формату зберігання розрідженої матриці дозволяє зменшити час обчислень до кількох разів.
- Для розріджених симетричних матриць без характерних особливостей портрету (матриці загального вигляду) з точки зору часу обчислень за алгоритмом множення матриці на вектор в послідовній системі оптимальним є формат CSR.
- Як наслідок попереднього: за замовчанням доцільно використовувати формат CSR.
- Застосування модифікації формату CSR для симетричних матриць дозволяє зменшити вимоги до пам'яті за рахунок збільшення часу обчислень.
- Для розріджених блочно-щільних матриць з точки зору часу обчислень за алгоритмом множення матриці на вектор в послідовній системі оптимальним є формат BCSR.

Для підвищення продуктивності обчислень було застосовано SIMD команди сучасних процесорів, що дало зменшення часу обчислень в окремих випадках до 1,7 разів.

Було встановлено, що головним фактором, що обмежує можливість використання систем з великою кількістю процесорів є швидкість каналу пам'ять–процесор.

Автор виражає вдячність Центру суперкомп'ютерних обчислень НТУУ «КПІ», [12] а також Intel Manycore Testing Lab [13] за наданий машинний час для розробки і тестування програмного забезпечення.

Перелік посилань

1. Sagan Hans. Boundary and Eigenvalue Problems in Mathematical Physics [Text]. – Dover Publications, 1989. – ISBN 0-486-66132-6.
2. Tore Opsahl Filip Agneessens John Skvoretz. Node centrality in weighted networks: Generalizing degree and shortest paths [Text]. – 2010. – Vol. 32. – Pp. 245–251.
3. We knew the web was big [Electronic resource]. – Access mode: http://googleblog. blogspot.com/2008/07/we-knew-web-was-big.html. – Last access: 15.04.2011. – Title from the screen.
4. Lanczos C. An iteration method for the solution of the eigenvalue problem of linear differential and integral operators [Text] // Journal of the Research of the National Bureau of Standards. – 1950. – Vol. 45. – Pp. 255--282.
5. Paige C. C. Computational variants of the lanczos method for the eigenproblem [Text] // Journal of the Institute of Mathematics. – 1972. – Vol. 10. – Pp. 373--381.
6. Saad Y. Iterative Methods for Sparse Linear Systems, 2nd edition [Text]. – Philadelpha, PA: SIAM, 2003.
7. Saad Y. Theoretical error bounds and general analysis of a few Lanczos-type algorithms [Text] // Proceedings of the Cornelius Lanczos International Centenary Conference / Ed. by J. D. Brown, M. T. Chu, D. C. Ellison, R. J. Plemmons. – Philadelphia, PA: SIAM, 1994. – Pp. 123--134.
8. The matrix market exchange formats: Initial design [Electronic resource]. – Access mode: http://math.nist.gov/MatrixMarket/reports/MMformat.ps.gz. – Last access: 15.04.2011. – Title from the screen.
9. Duff Iain S., Erisman A. M., Reid J. K. Direct Methods for Sparse Matrices [Text]. – 1986. – Pp. xiii + 341. – ISBN 0-19-853408-6 (hardcover).
10. Templates for the Solution of Linear Systems: Building Blocks for Iterative Methods, 2nd Edition [Text] / R. Barrett, M. Berry, T. F. Chan et al. – Philadelphia, PA: SIAM, 1994.
11. Karakasis Vasileios, Goumas Georgios, Koziris Nectarios. Perfomance models for blocked sparse matrix-vector multiplication kernels [Text] // Proceedings of the 2009 International Conference on Parallel Processing. – ICPP '09. – Washington, DC, USA: IEEE Computer Society, 2009. – Pp. 356--364. http://dx.doi.org/10.1109/ICPP.2009.21.
12. Центр суперкомп'ютерних обчислень НТУУ «КПІ» [Електронний ресурс]. – Режим доступу: http://hpcc.org.ua/. – Останнє звернення: 15.04.2011. – Назва з екрану.
13. Intel Manycore Testing Lab [Electronic resource]. – Access mode: http://software.intel.com/ en-us/articles/intel-many-core-testing-lab/. – Last access: 15.04.2011. – Title from the screen.

УДК 004.75

РОЛІК О.І.,
МОЖАРОВСКИЙ П.Ф.,
ВОВК В.М.,
ЗАХАРОВ Д.С.

МЕТОД ЗВЕДЕННЯ МЕТРИК ЯКОСТІ ФУНКЦІОНУВАННЯ КОМПОНЕНТІВ ІТ-ІНФРАСТРУКТУРИ ЗА ДОПОМОГОЮ АПАРАТУ НЕПАРАМЕТРИЧНОЇ СТАТИСТИКИ

Предложен метод сведения метрик оценки качества функционирования компонентов ИТ-инфраструктуры. Метод основан на представлении ИТ-инфраструктуры в виде графа и применении аппарата непараметрической статистики, что позволяет сводить разнотипные параметры, влияющие на качество функционирования компонентов, к универсальным обобщенным показателям. Приведена последовательность этапов метода, раскрыты особенности его алгоритмизации и программирования, получены оценки быстродействия и показаны примеры использования. Применение предложенного метода сведения метрик позволяет контролировать качество функционирования компонентов ИТ-инфраструктуры и своевременно реагировать на изменения состояния компонентов.

Method for aggregating quality metrics of IT-infrastructure component functioning estimation is proposed. Method uses graph-based presentation of IT-infrastructure in the IT-infrastructure control system and nonparametric statistics, what enables aggregating parameters having different types and influence on the quality of component functioning in generalized indicators. Method sequence, performance estimates and application examples are given, algorithmization and programming issues are considered. Application of the proposed method for metric aggregation allows controlling quality of IT-infrastructure component functioning and reacting timely on components state change.

Вступ

Ефективна діяльність сучасних комерційних організацій, державних та військових структур безпосередньо залежить від надійності та ефективності роботи середовища функціонування автоматизованих інформаційних технологій (ІТ) – ІТ-інфраструктури. Обробку та обмін інформацією між розподіленими елементами ІТ-інфраструктури здійснює інформаційно-телекомунікаційна система (ІТС), що є сукупністю інформаційної системи та телекомунікаційної мережі. Підтримку експлуатаційних параметрів функціонування ІТС на рівні, визначеному певними договорами, такими як договір про рівень послуг чи договір про управління застосуваннями, забезпечує система управління ІТ-інфраструктурою (СУІ) [1–3]. Однією з основних частин СУІ є підсистема моніторингу, яка здійснює оцінку стану ІТ-інфраструктури шляхом виконання безперервного контролю справності її елементів та розрахунку показників ефективності функціонування ІТС та ІТ-інфраструктури в цілому. Для цього підсистема моніторингу здійснює збір інформації, не обхідної для оцінки стану елементів ІТС, у вигляді значень параметрів функціонування апаратного та програмного забезпечення обчислювальної техніки та телекомунікаційної мережі, її обробку та послідовне зведення до узагальнених метрик, що характеризують функціонування не тільки елементів ІТС, а й функціональних і технологічних підсистем, а також ІТС в цілому. Поряд з числовими оцінками важливе місце займають класи якості функціонування, віднесення до яких дозволяє визначити перелік та пріоритети дій, які застосовуються з метою підвищення ефективності функціонування ІТ-інфраструктури.

Робота присвячена розробці методу розрахунку зведених кількісних характеристик для виділених множин елементів ІТ-інфраструктури, що характеризують надійність, ефективність та інші аспекти безперебійного виконання функцій ІТС на основі значень параметрів функціонування її елементів, зібраних з використанням агентської технології [4, 5]. Суть методу полягає у зведенні групи метрик елементів, пов’язаних функціональними залежностями, до єдиної узагальненої величини, яка однозначно характеризує стан елементів певної функціональної одиниці ІТС. В основу методу покладено апарат непараметричної статистики, що застосовується до представлення поточного стану елементів ІТС у евклідовому просторі їх

параметрів з поетапним узагальненням оцінок до рівня ІТС в цілому згідно її ієрархічного представлення.

Аналіз методів та систем зведення метрик елементів ІТ-інфраструктури

Управління процесами, що протікають в ІТ-інфраструктурі, потребує моніторингу та оцінки тих складових цих процесів, які здійснюють безпосередній або прихований вплив на підконтрольні метрики якості та їх зведення до узагальнених метрик, а отже вимагає створення спеціалізованих підсистем моніторингу та аналізу для отримання відповідних оцінок певних аспектів функціонування ІТ-систем. Останнім часом множинність рішень усе частіше замінюється централізованими СУІ [1, 3], що виконують ряд задач з управління ІТ-інфраструктурою і містять такі підсистеми моніторингу та оцінки якості функціонування, які, застосовуючи новітні методи, охоплюють певний аспект діяльності функціональних одиниць ІТ-інфраструктури або проводять багатогранну оцінку.

У роботі [6] оцінка стану елементів визначається за допомогою аналітичних методів зведення метрик, проте розрахунки вимагають тривалого калібрування схеми оцінювання.

У [7] оцінювання стану виконується у вигляді простої логічної послідовності однотипних операцій і дає змогу представити результати у наглядній формі, але вимагає великого об'єму апріорної інформації та її експертної оцінки, вимагає значних обчислювальних потужностей.

У [8] зведення метрик виконується за допомогою системи нечіткого логічного висновку із залученням експертів, а це потребує участі експертного персоналу, і тому процес оцінювання не піддається автоматизації.

У праці [9] застосовано підхід до оцінки стану окремих елементів ІТС єдиним параметром без урахування ієрархії елементів ІТ-інфраструктури.

У [10] представлено систему моніторингу середовищ великомасштабних паралельних і розподілених обчислень, де застосована методика зведення метрик, яка використовує деревовидне представлення процесів, що виконуються, та дозволяє суттєво зменшити об'єми даних, що збираються і зберігаються на жорстких носіях. Система певним чином вирішує більшість розглянутих вище питань, проте вимагає класифікації даних моніторингу.

У [11] пропонується модель неперервного зваженого зведення метрик, яка спеціально розроблена для оцінки якості функціонування програмного забезпечення. На відміну від [12] дані моніторингу проектуються на неперервну шкалу єдиного масштабу, але тут суворо визначені рівні зведених метрик, що стримує поширення моделі на нові підходи до оцінювання програмного коду, і система оцінювання верхнього рівня застосовує функції зважування. Такий підхід вимагає тривалого налагодження механізму оцінювання та не дозволяє у найкоротші строки адаптуватися до поточних пріоритетів та стратегії корпорації.

В роботі [13] пропонуються система та метод збору і зведення метрик функціонування ЕОМ із застосуванням факторного аналізу, які крім заощадження об'ємів дискового простору дозволяють визначити множини найбільш важливих для даної одиниці техніки показників, однак навіть обертання отриманих характеристик часто унеможливлює їх змістовне тлумачення.

Аналіз відомих реалізацій СУІ, незважаючи на їх розповсюдженість та всеохоплюваність, виявляє певні недоліки. Так, пакет програм HP Operations Manager вимагає високовартісної ліцензії, залучення значної кількості висококваліфікованих спеціалістів та тривалого, витратного й багатоетапного процесу впровадження. Ряд ознак також унеможливлює застосування програмних модулів System Center від Microsoft у якості основи для побудови СУІ будь-яких масштабів. Деякі з наведених проблем вирішуються у [2], проте бракує універсального узагальнюючого математичного апарату оцінки стану об'єктів моніторингу і управління.

Постановка проблеми

Виникає необхідність визначення показників якості функціонування розподілених компотнентів ІТ-інфраструктури, залучених до виконання покладених на систему задач. Такими показниками є узагальнені метрики, що характеризують певні аспекти діяльності функціональних одиниць ІТ-інфраструктури, а саме: окремі задачі, комплекси задач або ж цілі функціональні підсистеми. Для розрахунку узагальнених метрик необхідно мати ієрархію зведення оцінок, яка містить орієнтовану на реально виконувані задачі структуру залежностей між елементами ІТС. Залежності вказують на те, які параметри,

яких елементів ІТ-інфраструктури і яким чином впливають на визначену характеристику певного її компоненту. Тоді задача полягає у розробці методу отримання показників функціонування елементів та їх зведення на основі ієрархічного представлення ІТС до узагальнених метрик, які комплексно описують якість виконання покладених на систему задач.

Метою даної статті є розробка методу розрахунку та зведення метрик якості функціонування груп елементів ІТС, ФПС та ІТ-систем, а також узагальнення цих оцінок до рівня всієї ІТ-інфраструктури єдиним універсальним способом. Для зведення метрик необхідно виробити принципи структурування представлення ІТС у СУІ, виділення груп однорідних елементів, після чого можна застосувати процедури оцінки якості функціонування елементів та функціональних одиниць ІТ-інфраструктури на основі апріорної інформації про їх параметри, ідентифікації порушення режиму штатної роботи та визначення елементів, що сприяють відхиленню функціональної одиниці від стану функціонування у відповідності з визначеним регламентом.

Представлення ІТС в СУІ у вигляді графу ОМУ

Процес моніторингу та оцінки отриманих показників зручно проводити у разі представлення ІТС у вигляді графу. Відсутність явно вираженої ієрархічної структури залежностей між елементами у сучасних багатоелементних та багатофункціональних ІТС веде до необхідності виділення функціональних підсистем (ФПС), тобто груп елементів, виокремлених із повного складу за ознакою участі у виконанні певної функції. ІТС може містити декілька таких груп або складати єдину. У залежності від задач деякі групи можуть залишатися незмінними тривалий час або ж піддаватися частим змінам, наприклад ті, що відповідають за дискретно-виконувані функції [14]. У корпоративних ІТС часто виділяють ФПС стратегічного управління, що виконує комплекси задач фінансового менеджменту, маркетингу, управління документообігом та ін., бухгалтерського обліку, управління виробництвом і т. д., кожна з яких характеризується унікальністю структури та особливістю підходу до оцінки ефективності функціонування.

Серверні вузли, ПК, система кабелів та комутуючого обладнання, офісна периферія, загальносистемне та спеціальне програмне забезпечення утворюють апаратно-програмні комплекси ІТС, на які покладено функції ФПС. Інтегрованість ФПС у межах сучасних підприємств, орієнтованих на мінімізацію задіяних ресурсів, приводить до змішування апаратно-програмних частин підсистем, використання однієї й тієї ж обчислювальної та телекомунікаційної техніки декількома ФПС, наприклад, на одному сервері або ПК можуть знаходитися застосування для управління документообігом та ФПС бухгалтерського обліку. Множинність функціональних зв’язків представимо графом залежностей, вершинами якого є об’єкти моніторингу та управління.

Під об’єктом моніторингу та управління (ОМУ) мається на увазі абстрактна сутність, що є частиною ІТС, яка крім вербальної характеристики може бути описана набором параметрів. На нижньому рівні ОМУ можуть бути апаратні та програмні елементи ІТС, на верхньому – функціональні та технологічні підсистеми, що виконують покладені на ІТС задачі. Кожний ОМУ містить відомості про конфігурацію та поточний стан окремого елементу ІТС, в ньому інкапсульовано ті властивості та характеристики елементів ІТС, що стосуються процесів управління ІТ-інфраструктурою. Так, наприклад, для окремих ЕОМ чи серверних вузлів це процесорна ємність, використання оперативної пам’яті, своп-файлу, жорстких носіїв, поточна швидкість зчитування/запису контролеру жорсткого диску та ін., для Web-серверів – пропускна здатність, кількість вдалих з’єднань між клієнтами та Web-сервером за секунду, середній час відгуку Web-сервера, відносна швидкодія інтерфейсу програмування застосувань, кількість помилок за секунду та ін., для одиниць програмного забезпечення – кількість рядків вихідного коду, циклічна складність, кількість методів на клас, глибина спадковості та ін. Виділення ОМУ у процесі декомпозиції системи складається із введення і формального опису логічних сутностей у відповідності до вагомих функціональних одиниць і формування наборів властивостей, що визначають їх стан.

Сукупність ОМУ та їх взаємозв’язків утворюють орієнтований граф залежностей між ОМУ, у якому останні є вершинами, а направлені ребра виділяють множину впливових об’єктів, показуючи, які параметри і яких ОМУ нижніх рівнів ієрархії впливають на кожен окремий ОМУ верхнього рівня. Граф ОМУ по-

винен максимально наближатися до дерева та не містити циклів, а кожній його вершині необхідно поставити у відповідність набір параметрів ОМУ. Граф залежностей дозволяє окреслити множину об'єктів, що вірогідно впливають на даний ОМУ, та їх параметрів, повинен відображати функціональну інфраструктуру, охопити максимум параметрів ОМУ різних рівнів, відповідати внутрішній будові ІТС і виходити на узагальнені цільові показники. При цьому необхідно здійснити відображення отриманих показників на якісну шкалу. Крім того означаються функціональні групи (ФГ) – множини однорідних об'єктів, виділені для вирішення однієї або декількох сумісних задач і описувані однаковим набором параметрів.

Для забезпечення запропонованого методу вихідними даними на основі графу ОМУ необхідно узгодити апріорну інформацію про взаємодію залежних ОМУ, окреслити множину параметрів, суттєвих для оцінюваного параметра розглянутого об'єкту, та визначити математичний апарат для розрахунку кіль кісних показників функціонування об'єкту та їх проектування на якісну вісь. Для формування множини суттєвих параметрів оберемо випадок, коли оцінюваний об'єкт характеризується лише однією метрикою, яка залежить від усіх параметрів ОМУ нижчого рівня ієрархії. Цей випадок без ускладнень узагальнюється для розрахунку декількох зведених метрик ОМУ, який може бути проведений за кожною з них окремо. Загальність методу вимагає рівності наборів параметрів у межах оцінюваної метрики, тобто у кожного з впливових ОМУ до уваги беруться набори однакових параметрів, значення кожного з яких за необхідності приводяться до одного масштабу.

Суть запропонованого методу зведення метрик

Якісна оцінка виражає поняттєву характеристику процесу, прив'язана до його природи і очікуваного результату, у якомусь сенсі є суб'єктивною і тому в основі своїй майже не піддається формалізації та узагальненню. Така оцінка повинна враховувати взаємозв'язок між елементами ІТС та актуальними значеннями їх метрик, а отже, мати статистичну природу. Непередбачуваність впливу широкого спектру факторів на елементи ІТС та складні взаємозв'язки між ними не дозволяють застосувати відомі параметричні статистичні моделі. Так, самоподібність моделей систем, поведінка яких визначається інтернет-активністю [15, 16], множинність, непередбачуваність та різноманітність задач, які вирішуються засобами ІТС, не дозволяють як застосовувати параметричні моделі, так і прогнозувати поведінку елементів на визначені проміжки часу.

Для зведення метрик інтерес становлять непараметричні статистичні моделі, які зазвичай не ґрунтуються на жодних припущеннях про функції розподілу і водночас застосовують усю наявну інформацію. Серед них найбільш цікавою є функція статистичної глибини [17, 18], оскільки саме вона дає оцінку подібності, яка сходиться до ймовірнісних показників. Ідеологія підходу, заснованого на застосуванні функції глибини та її глибинно-впорядкованих регіонів до представлення ФГ у параметричному евклідовому просторі, де координатними осями слугують параметри функціонування елементів ІТ-інфраструктури, викладена у [9]. Показник глибини універсальний і може бути застосований для аналізу роботи компонентів ІТ-систем, ФПС чи ФГ на основі різних наборів параметрів.

Апарат статистичної глибини дозволяє звести множину метрик, що характеризують стан об'єктів ФГ, до єдиного показника, який надає переваги роботи з одномірними даними, тобто простоту упорядкування чи ранжування та незалежність від поточного положення групи елементів ІТС у параметричному просторі. Розподіл об'єктів групи у даному просторі визначається множиною їх глибинно-впорядкованих регіонів, що дозволяє вирішувати задачі оцінки стану групи, ідентифікації та локалізації несправностей та порівнювати її з іншими групами об'єктів того ж простору.

Умову нормального функціонування групи можна задати за допомогою глибинного регіону, наприклад обмеженням його форми, об'єму чи положення. Обмеження форми будуть виражати відносну незмінність певного взаємозв'язку між об'єктами групи, обмеження об'єму покликані стримувати їх розходження за параметрами, а фіксація положення деяким підпростором виражатиме наявність зони нормального функціонування. В останньому випадку можна стверджувати, що група справно виконує поставлені задачі у разі належності всього регіону певної глибини до зони штатного функціонування.

Особливої уваги заслуговують ОМУ, які у даному параметричному просторі знаходяться віддалік від головної хмари, а отже є претендентами на сторонні точки. Для їх ідентифікації пропонується спершу виділяти такі допустимі межі, що об'єкти, які їм не належать, майже напевно є сторонніми. Такий підпростір можна окреслити, наприклад, гіперплощинами, перпендикулярними до координатних осей, що забезпечить простоту їх задавання оператором та високу швидкодію диференціації. Крім того, існують нотації глибини, що здатні виділити сторонні точки самостійно, наприклад півпросторова [19] та проекційна глибини [20, 21].

У запропонованому методі зведення метрик вибір нотації функції статистичної глибини припав на глибину зоноїда [22, 23] завдяки наявності ряду бажаних властивостей та особливостей, які роблять її застосовною у якості математичного апарату для оцінки функціонування компонентів ІТ-інфраструктури. Серед них виділимо афінну інваріантність, що забезпечує фільтрацію впливу зміни положення у параметричному просторі оцінюваної групи цілком на оцінку якості функціонування групи, та неперервність за глибиною та точками. Самі ж глибинно-впорядковані регіони, які в даній нотації іменуються зоноїдами, є випуклими, а також для них існує апарат побудови випуклих оболонок у просторі будь-якої розмірності [24].

Суть запропонованого методу викладемо у вигляді послідовності етапів, частина з яких є попередніми або підготовчими, а інші виконуються з різною періодичністю.

Етап 1. Виконується попереднє групування елементів, виділення залежних ОМУ та організація зв'язків впливу ОМУ нижчого рівня ієрархії на вищі. Функціональною одиницею, до якої застосовується метод, є ОМУ $O_l, l=1,...,L$, де L – кількість вершин у графі ОМУ. Стан кожного *l*-го ОМУ характеризується K_l параметрами $P_{l,k}, k=1,...,K_l$, де K_l – кількість параметрів *l*-го ОМУ, кожен з яких характеризує певний аспект функціонування ОМУ O_l. Значення *k*-го параметру *l*-го ОМУ $P_{l,k}$ визначається значеннями параметрів множини ОМУ нижчого рівня ієрархії, яка складає оцінювану групу $S_{l,k}$ параметру $P_{l,k}$, що містить $N_{l,k}$ однорідних ОМУ $X_{l,k,n}, n=1,...,N_{l,k}$. Серед параметрів групи $S_{l,k}$ виділяються ті, що чинять вплив на $P_{l,k}$: $P_{l,k,d}, d=1,...,D_{l,k}$, де $D_{l,k}$ – кількість параметрів оцінки функціонування об'єктів, що складають оцінювану групу $S_{l,k}$. Однорідність ОМУ означає ідентичність набору параметрів ОМУ у межах оцінюваної групи, тобто стан параметру $P_{l,k}$ визначається набором $D_{l,k}$ $N_{l,k}$ елементів ІТ-інфраструктури.

Таке представлення забезпечує загальність апарату, яка передбачає залежність різних параметрів оцінюваного ОМУ від різних груп параметрів одного й того ж або різних ОМУ нижчих рівнів ієрархії, проте зайве не ускладнює його, оскільки граничним частинним випадком є наявність у ОМУ лише одного параметру, що визначається усіма параметрами ОМУ єдиної оцінюваної групи.

Етап 2. Виконується збір даних функціонування елементів ІТ-інфраструктури протягом деякого періоду T, який визначається поточною динамікою групи, та представлення їх у вигляді точок простору розмірності $D_{l,k}$, де осями є параметри об'єктів групи. Виділимо два види ОМУ – такі, що їх параметри або частина з них розрахункові, та ті, що характеризуються вимірюваними параметрами. Значення параметрів ОМУ першого виду розраховуються на четвертому кроці поточної процедури і зберігаються у пам'яті. Параметри ОМУ другого виду, а також параметри ОМУ першого виду, що не є розрахунковими, отримуються шляхом вимірювання чи виконання передбачених тестових перевірок. Тут не розглядається механізм відсіювання сторонніх точок, оскільки він залежить від задач, поставлених перед компонентами ІТС, та поточної реалізації і може бути визначений оператором.

Представленню ОМУ у $D_{l,k}$-мірному параметричному просторі передує їх приведення до єдиних масштабу та координатної сітки, або ж проекція на відносну шкалу, наприклад, відрізок реальної осі $[0,1]$.

Етап 3. Виділяється підпростір $\Theta_{l,k}$ нормального функціонування об'єктів групи $S_{l,k}$, що формується у вигляді обмежень гіперповерхнями деякої зони $D_{l,k}$-мірного параметричного простору. У найпростішому випадку межі можуть бути гіперплощинами, ортогональними координатним осям, і задавати зону, вихід за межі якої є критичним, оскільки означає пору-

шення штатного режиму функціонування компонентів ІТ-систем.

Етап 4. Виконується розрахунок стану параметру функціонування $P_{l,k}$ як оцінки ймовірності виходу параметрів функціонування довільного ОМУ $X_{l,k,n}$ групи $S_{l,k}$ за межі підпростору нормального функціонування. Нехай $X_{l,k,n} = \left(x_{l,k,n,1}, x_{l,k,n,2}, ..., x_{l,k,n,D_{l,k}}\right)^T$, де $x_{l,k,n,d}$ — значення d-го параметру n-го ОМУ групи $S_{l,k}$, а $(\cdot)^T$ означає транспонування вектору. Тоді значення стану параметру розраховується за формулою

$$x_{l,k} = \begin{cases} \inf\{\alpha_m : \mathrm{Z}(\alpha_m, S_{l,k}) \in \Theta_{l,k}\} \forall \alpha_m \in [0,1] : \exists Z(\alpha_m, S_{l,k}) \in \Theta_{l,k} \\ \gamma_{l,k,\text{н.в.}} - \text{в усіх інших випадках} \end{cases} \quad (1)$$

де $x_{l,k}$ – значення параметру $P_{l,k}$, $\alpha_m \in \left[\frac{m}{n}, \frac{m+1}{n}\right]$, $m = 1,...,n-1$, $\gamma_{l,k,\text{н.в.}}$ – ідентифікує надмірне відхилення групи $S_{l,k}$ від зони нормального функціонування і може бути визначене по-різному у залежності від конкретної реалізації і значення $\alpha_{l,k,\max}$, яке обговорюється на наступному кроці, а $Z\left(\alpha_m, S_{l,k}\right)$ визначається як [22]

$$Z\left(\alpha_m, S_{l,k}\right) = conv\left\{\frac{1}{\alpha_m N_{l,k}} \sum_{j=1}^{m} X_{l,k,i_j} + \left(1 - \frac{m}{\alpha_m N_{l,k}}\right) X_{l,k,i_{m+1}} : \{i_1,...,i_{m+1}\} \subset \{1,...,N_{l,k}\}\right\} \quad (2)$$

у разі, якщо $\alpha_m \in \left[\frac{1}{N_{l,k}}, 1\right]$, та як

$$Z\left(\alpha_m, S_{l,k}\right) = conv\left\{X_{l,k,1}, ..., X_{l,k,N_{l,k}}\right\}, \quad (3)$$

якщо $\alpha_m \in \left[0, \frac{1}{N_{l,k}}\right]$.

Етап 5. Визначені на попередньому кроці значення параметрів $P_{l,k}$ перевіряються на належність установленим межам, які можуть бути гарантованою якістю обслуговування, ймовірністю доступу до ресурсів чи іншими характеристиками, передбаченими у договорі про надання послуг, або ж визначені емпіричним шляхом у межах оцінюваної групи $S_{l,k}$. За наявності відхилення, тобто перевищення встановленого відсотка $x_{l,k} > \alpha_{l,k,\max}$, необхідно задіяти механізми усунення можливих несправностей, наприклад балансування навантаження у межах групи, попередньо отримавши рекомендації щодо порядку надання таких дій. Тоді для кожного ОМУ $X_{l,k,n} \notin \Theta_{l,k}$ обчислюється значення

$$\beta_{k,l,n} = \begin{cases} \sup\{\beta_{k,l,n} : X_{k,l,n} \in \mathrm{Z}(\beta_{k,l,n}, S_{l,k})\} \forall X_{l,k,n} : \exists Z(\beta_{k,l,n}, S_{l,k}) : X_{l,k,n} \in \mathrm{Z}(\beta_{k,l,n}) \\ 0 - \text{в усіх інших випадках} \end{cases}, \quad (4)$$

Дії з усунення несправностей у першу чергу застосовуються до тих ОМУ, які мають менші значення $\beta_{k,l,n}$, що особливо критично у випадку $x_{l,k} = \gamma_{l,k,\text{н.в.}}$.

Перший етап методу виконується одноразово і вимагає повторення лише у разі змін в ІТ-інфраструктурі, при цьому часто достатньо застосовувати структурування лише до зміненої частини системи. Те саме можна сказати й про алгоритми другого кроку, які визначають масштаб чи відносну вісь, проте операції вимірювання та чисельного перетворювання вимагають виконання на кожній ітерації процедури методу. Обмеження підпростору нормального функціонування $\Theta_{l,k}$, які застосовуються на третьому кроці, а також величини $\gamma_{l,k,\text{н.в.}}$ та $\alpha_{l,k,\max}$ з четвертого та п'ятого кроків також можуть уточнюватися згідно з цілями й задачами, покладеними на ІТ-систему.

Особливості алгоритмічної реалізації методу

Найбільш затратним у обчислювальному плані являється четвертий крок процедури, оскільки він може вимагати багаторазового розрахунку $Z\left(\alpha_m, S_{l,k}\right)$, однак питання застосування алгоритмів оптимізації залишається відк-

ритим і значним чином залежить від необхідної точності розрахунків та доступного часу, тому обмежимося лише перевіркою виконання умови

$$Z\left(\alpha_m, S_{l,k}\right) \in \Theta_{l,k}. \quad (5)$$

У найпростішому випадку межі підпростору нормального функціонування $\Theta_{l,k}$ можуть бути ортогональними до координатних осей параметричного простору гіперплощинами, що відповідають верхнім і (або) нижнім обмеженням на ресурси без урахування можливого зв'язку між

$$\theta_{l,k,d,\min} \le z\left(\alpha_m, S_{l,k}\right)_{q,d} \le \theta_{l,k,d,\max} \forall q = 1,...,Q\left(Z\left(\alpha_m, S_{l,k}\right)\right),\ d = 1,...,D_{l,k}, \quad (6)$$

де $z\left(\alpha_m, S_{l,k}\right)_{q,d}$ – проекція q-ї вершини зоноїда $Z\left(\alpha_m, S_{l,k}\right)$ на d-у вісь параметричного підпростору, $Q\left(Z\left(\alpha_m, S_{l,k}\right)\right)$ – кількість вершин фігури $Z\left(\alpha_m, S_{l,k}\right)$, а $\theta_{l,k,d,\min}$ і $\theta_{l,k,d,\max}$ – мінімальна та максимальна границі по d-й осі прямокутного $D_{l,k}$-мірного паралелепіпеду нормального функціонування відповідно.

Якщо наявний зв'язок між параметричними обмеженнями підпростору нормального функціонування, $\Theta_{l,k}$ може приймати складніші форми і тоді перевірку виконання умови (5) необхідно алгоритмізувати окремо для кожного такого випадку.

Розрахунок вершин та граней $Z\left(\alpha_m, S_{l,k}\right)$ може бути виконаний згідно алгоритму [24], або за [25] якщо $D_{l,k} = 2$. У першому випадку швидкодія алгоритму складатиме $O\left(D_{l,k}^2 \cdot \left(D_{l,k}^2 + N_{l,k}\right) \cdot F\left(N_{l,k}, D_{l,k}\right)\right)$, де $F\left(N_{l,k}, D_{l,k}\right)$ – середня кількість граней зоноїду, при затратах пам'яті $O\left(F\left(N_{l,k}, D_{l,k}\right) \cdot D_{l,k} + C\right)$, де C – деяка константа, а у другому швидкодія алгоритму оцінюватиметься як $O\left(N_{l,k}^2 \cdot \log N_{l,k}\right)$. Обчислення глибини на п'ятому кроці може бути виконане згідно симплекс методу або його версії за допомогою декомпозиції Данціга-Вульфа [26].

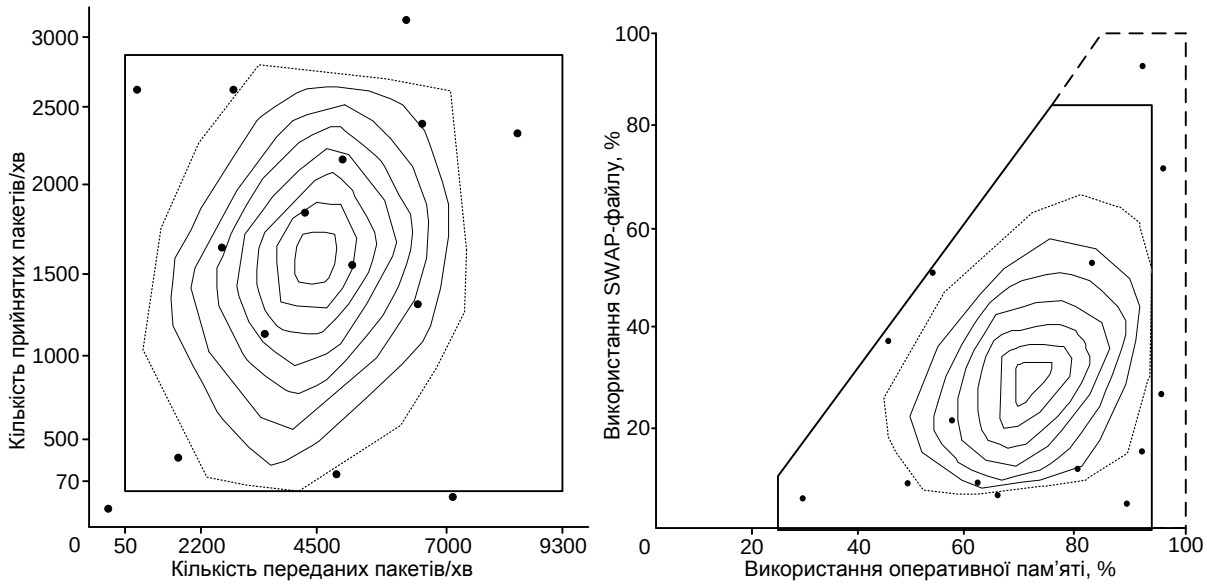
Особливої уваги заслуговує питання визначення періодичності ітерацій процедури методу. Розрахунок параметрів $P_{l,k}$ можна починати у разі змін відповідних значень вимірюваних чи розрахункових параметрів ОМУ групи $S_{l,k}$, проте, зважаючи, що вимірювані параметри поними. Тоді підпростір нормального функціонування є прямокутним $D_{l,k}$-мірним паралелепіпедом, а перевірка входження фігури $Z\left(\alpha_m, S_{l,k}\right)$ до нього спрощується завдяки тому, що $Z\left(\alpha_m, S_{l,k}\right)$ є випуклим многогранником. Це дозволяє визначати входження $Z\left(\alpha_m, S_{l,k}\right)$ як дотримання кожної з системи нерівностей:

требують постійного моніторингу та інтегрування на протязі встановленого періоду T, можна застосувати інтелектуальні механізми визначення частоти розрахунку [27].

Реалізація методу та приклади

Процедура методу запрограмована та підключена у вигляді модуля до СУІ, розробленої в НТУУ «КПІ». Представлення IT-інфраструктури у вигляді графу ОМУ, групування об'єктів та їх параметрів, а також встановлення меж підпростору нормального функціонування проводилося за допомогою інтуїтивного графічного інтерфейсу клієнтської частини, а збір даних робочих станцій, серверів та комутуючого обладнання виконувався агентським ПЗ. Проведення випробувань показали застосовність методів і підтвердили правильність оцінок швидкодії та використання оперативної пам'яті.

На рис. 1,а приведено оцінку стану ФГ, що складається із 15 робочих станцій зі встановленими агентами моніторингу СУІ. Низька кількість переданих та прийнятих пакетів свідчить про відхилення агенту від режиму нормального функціонування чи відсутність його авторизації на сервері СУІ відповідно, у той час як надмірне використання мережевого інтерфейсу протягом тривалого часу свідчить про можливу несанкціонованість трафіку, неполадки у роботі агентів або наявність стороннього ПЗ на робочій станції. Рис. 1,б демонструє оцінку тієї ж групи стосовно балансу використання свопу та оперативної пам'яті. На обох рисунках звичайною лінією визначено межі стану штатного функціонування, а пунктирною – зоноїд найбільшої глибини, що належить встановленим межам.



а) *б)*

Рис. 1. Оцінка стану ФГ робочих станцій зі встановленими агентськими модулями стосовно: а) мережевого інтерфейсу; б) балансу використання свопу і оперативної пам'яті

Оцінка завантаженості групи серверів технології SaaS приведена на рис. 2. Низька задіяність ресурсів дозволяє перенести сервер на іншу віртуальну машину з меншими вимогами до ресурсів або згорнути його, а висока вказує на необхідність рознесення виконуваних на ньому задач або виділення додаткових ресурсів.

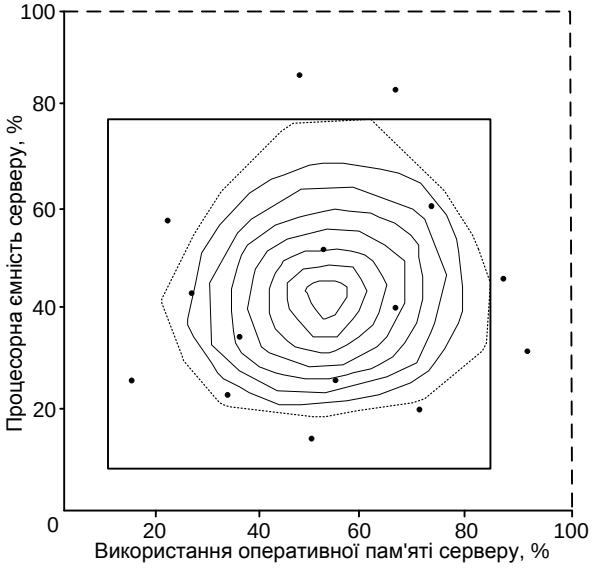


Рис. 2. Оцінка стану групи серверів SaaS

Висновки

У роботі запропоновано метод зведення метрик якості функціонування компонентів ІТ-інфраструктури, який вирішує задачу розрахунку кількісних характеристик компонентів ІТС, що залежать від множини параметрів функціонування елементів нижчих рівнів ієрархії. Структурування представлення ІТС у СУІ дозволяє виділити групи параметрів, які найкраще характеризують діяльність ІТ-інфраструктури, а в подальшому провести межу автоматизації вирішення задач у СУІ та подати ту їх частину, що вимагає втручання оператора, за допомогою інтуїтивного графічного інтерфейсу. У статті викладено концептуальні положення та особливості алгоритмізації і програмування методу, вказано відповідні алгоритми, а їх швидкодія та можливість застосування підтверджені експериментами.

Даний метод вирішує питання узагальнення показників групи елементів шляхом їх представлення у єдиному параметричному просторі з можливістю подальшого проектування на якісну вісь. Таке узагальнення враховує ймовірнісну сторону розгляду елементів, не прив'язуючись при цьому до якого-небудь розподілу завдяки застосуванню непараметричних моделей, використовуючи максимум наявної інформації та не роблячи припущень щодо моделі розподілу. Єдиність підходу дозволяє застосовувати метод до будь-якого вузла графу об'єктів ІТ-інфраструктури.

Список літератури

1. 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.
2. Теленик С.Ф. Система управления ИТ-инфраструктурой – путь к повышению эффективности функционирования предприятия / С.Ф. Теленик, А.И. Ролик, М.М. Букасов, А.В. Волошин, Д.А. Галушко // Інформаційні технології – інструмент підвищення конкурентоздатності підприємств: 9–10 груд. 2009.: матеріали конф. – К.: УкрНЦ РІТ, 2009. – С. 30–33.
3. Теленик С.Ф. Система управления информационной инфраструктурой транспортного предприятия / С.Ф. Теленик, А.И. Ролик, П.Ф. Можаровский, А.В. Волошин // Автомобільний транспорт. Вип. 25. – Харків.: ХНАДУ, 2009. – С. 242–245.
4. Ролик А. И. Распределение мобильных компонентов системы управления информационно-телекоммуникационной системой / А.И. Ролик, Р.Л. Соколовский // Вісник НТУУ «КПІ». Інформатика, управління та обчислювальна техніка. – К.: «ВЕК+», 2007. – № 47. – С. 113 –.
5. Ролік О.І. Застосування агентського підходу до управління інформаційно-телекомунікаційною системою АСУ спеціального призначення / О.І. Ролік, П.Ф. Можаровський, О.О. Покотило // Пріоритетні напрямки розвитку телекомунікаційних систем та мереж спеціального призначення: V наук.-практ. семін., 22 жовт. 2009 р.: доповіді та тези доповідей: – К.: ВІТІ НТУУ «КПІ», 2009. – С. 228–229.
6. Ролик А.И. Анализ качества функционирования элементов информационно-телекоммуникационных систем / А.И. Ролик, Е.В. Глушко // Вісник НТУУ «КПІ». Інформатика, управління та обчислювальна техніка. – К.: «ВЕК+», 2008. – № 48. – С. 113–120.
7. Теленик С.Ф. Оцінювання стану корпоративних інформаційно-телекомунікаційних систем / С.Ф. Теленик, О.І. Ролік, О.М. Моргаль // Контроль і управління в складних системах (КУСС-2010): X між нар. конф., 19–21 жовт. 2010 р. Вінниця: труди. – С. 122. http://www.vstu.vinnica.ua/mccs2010/materials/subsection_2.2.pdf. Дата доступу: 21.10.10.
8. 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.
9. Ролик А.И. Применение глубинно-упорядоченных регионов в системах управления ИТ-инфраструктурой / А.И. Ролик, П.Ф. Можаровский, Б.А. Март // Интеллектуальный анализ информации «ИАИ-2010»: X междунар. науч. конф. им. Т.А.Таран, 19–21 мая 2010 г.: сб. тр. – К.: Просвіта, 2010. – С. 214–221.
10. Boehm S. Aggregation of Real-Time System Monitoring Data for Analyzing Large-Scale Parallel and Distributed Computing Environments / S. Boehm, C. Engelmann, S.L. Scott // High Performance Computing and Communications (HPCC), 2010 12th IEEE International Conference. 1–3 Sept. 2010. – 2010. – P. 72–78.
11. Mordal-Manet K. An Empirical Model for Continuous and Weighted Metric Aggregation / K. Mordal-Manet, J. Laval, S. Ducasse, N. Anquetil, F. Balmas, F. Bellingard, L. Bouhier, P. Vaillergues, T.J. McCabe // 15th European Conference on Software Maintenance and Reengineering. 1–4 March 2011. – Oldenburg. Germany. 2011. – P. 141–150.
12. ISO/IEC 9126-1. Software engineering – product quality. Part 1: Quality model, 2001.
13. Патент США №6377907. МПК G06F 11/34. System and method for collation UNIX performance metrics (Anthony C. Waclawski) – Опубл. 23.04.2002.
14. Надійність техніки. Проектна оцінка надійності складних систем з урахуванням технічного і програмного забезпечення та оперативного персоналу. Основні положення: ДСТУ 3524-97. – К.: Держстандарт України, 1997. – 54 с.
15. Crovella M.E. Self-similarity in world wide web traffic: Evidence and possible causes / M.E. Crovella, A. Bestavros // IEEE/ACM Trans. on Networking – 1997, Dec. – Vol. 5. – P. 835–846.
16. Paxson V. Wide-Area Traffic: The Failure of Poisson Modeling / V. Paxson, S. Floyd // IEEE/ACM Trans. on Networking – 1995, June. – Vol. 3, No. 3. – P. 226–224.
17. Cascos I. Data Depth: Multivariate Statistics and Geometry / I. Cascos // New Perspectives in Stochastic Geometry / S. Kendall, I. Molchanov. – Oxford.: Oxford University Press, 2009. – P. 398–425.
18. Zuo Y. General notions of statistical depth functions / Y. Zuo, R. Serfling // Ann. Statist. – 2000. – Vol. 28, No. 2. – P. 461–482.
19. Tukey J. W. Mathematics and the Picturing of Data / J.W. Tukey// Proceedings of the International Congress of Mathematicians, Vancouver – 1974.

20. Stahel W. A. Breakdown of covariance estimators / W.A. Stahel // Research Report 31, Fachgruppe fuer Statistik, ETH, Zurich – 1981, Dec.
21. Donoho D. L. Breakdown properties of multivariate location estimators / D.L. Donoho // Ph. D. Qualifying Paper, Harvard University – 1982.
22. Koshevoy G. Zonoid trimming for multivariate distributions / G. Koshevoy, K. Mosler // Ann. Statist. – 1997, Oct. – Vol. 25, No. 5. – P. 1998–2017.
23. Mosler K. Multivariate dispersion, central regions and depth / K. Mosler. – New York.: Springer, 2002. – 291 p.
24. Mosler K. Computing zonoid trimmed regions of dimension d>2 / K. Mosler, T. Lange, P. Bazovkin // Computational Statistics and Data Analysis. – 2009. – Vol. 53, Issue 7. – P. 2500–2510.
25. Dyckerhoff R. Computing zonoid trimmed regions of bivariate data sets. / R. Dyckerhoff // COMPSTAT Proceedings in Computational Statistics. – 2000. – P. 29–30.
26. Dyckerhoff R. Zonoid data depth: theory and computation / R. Dyckerhoff, G. Koshevoy, K. Mosler // In A. Pratt, ed., COMPSTAT Proceedings in Computational Statistics. – 1996, Nov. – P. 235–240.
27. Ролик А.И. Планировщик задач системы управления информационно-телекоммуникационной системой / А.И. Ролик, П.Ф. Можаровский, А.В. Волошин // Інженерія програмного забезпечення 2009: Міжнар. наук.-практ. конф. аспірантів і студентів, 14–16 вересня 2009 р., Київ: тези доповідей. – Київ:, 2009. – С. 43.

УДК 681.3.016

АМОНС А.А.,
ЗАЙЦЕВ С.Ю.,
КИРИЧЕК А.А.

ВЫЯВЛЕНИЕ ПЛАГИАТА В ПРОГРАММНОМ КОДЕ C#

В статье рассматривается вопрос о том, что такое плагиат. С начала описываются модели представления программ, а так же рассматриваются алгоритмы определения плагиата в программном коде C#. По каждому алгоритму проведен анализ и сравнение для выяснения достоинств и недостатков каждого алгоритмы. В результате сделаны выводы и определены два алгоритма, которые считаются наилучшими, и внесены предложения по созданию новых алгоритмов, без недостатков в уже существующих. Приведен практический эксперимент для доказательства того, что новые алгоритмы работают лучше чем существующие.

The article discusses s problem of a plagiarism. At the beginning describes the models of program representation, and algorithms for determining plagiarism in software code C #. Each algorithm was analyzed and compared to determine the strengths and weaknesses. As a result, conclusions were made and identified two algorithms, which are considered the best, and made proposals to develop new algorithms without the disadvantages of existing ones. The practical experiment was held to prove that the new algorithms work better than existing ones.

Введение

Возможность легкого копирования информации, представленной в электронном виде, породила множество проблем, связанных с нарушением авторских прав. В бумажных и электронных изданиях часто можно встретить публикации об очередном противостоянии пиратов и владельцев авторских прав на распространяемые этими пиратами произведения. Как правило, в таких случаях единственным (но не всегда действенным) способом улаживания конфликтов являются судебные разбирательства.

Сейчас в век интенсивного развития информационных технологий интеллектуальная собственность становиться все более ценной. В связи со значительным увеличением объемов этого вида собственности назрела необходимость в мощных автоматических инструментах для защиты авторских прав, для инспектирования и проверки авторства, для нахождения плагиата.

- Плагиат – вид нарушения авторских прав. Принуждение к соавторству также рассматривается как плагиат [1].

Виды представления исходных кодов программ

В виде элемента n-мерного пространства. Ранние системы по обнаружению плагиата (например,[2]) представляли программу, как точку в n-мерном пространстве натуральных чисел с нулем, i-ая координата которой - количественная характеристика какого-либо свойства (атрибута) всей программы. Например, средняя длина строки кода, количество объявленных и используемых переменных, средняя длина имен переменных, количество операторов ветвления и так далее. Если точки двух программ лежат рядом, то одна из них предполагается плагиатом другой.

Исходный код

Некоторые системы обнаружения плагиата рассматривают исходный код “как есть”. Например, так поступают детекторы плагиата, которые работают с кодом так же, как и с обычными текстами. Но они крайне неэффективны для решения нашей задачи, так как переименование функций и переменных или несущественные изменения в коде являются серьезными препятствиями для их правильной работы. Иногда используется параметризованное представление кода (см. [3]).

Токенизация

Пусть у нас есть две строки кода f or(int i = 0; i <= n; i + +) и f or(int j = 0; (j 1) < n; j + +). Очевидно, что у них одинаковая функциональность, и плагиатор мог из одной получить

другую без особых усилий, а для ранее описанных представлений они совершенно различны. Чтобы бороться с такого рода средствами сокрытия плагиата было придумано токенизированное представление кода. Основная идея этого представления это сохранение существенных и игнорирование всех поверхностных (то есть легко модифицируемых) деталей кода программы. Процедура токенизации выглядит примерно так:

1) Каждому оператору языка (кроме пустого его игнорируем), который не является операндом, приписываем код, назначенный заранее для каждого класса операторов. Также коды можно приписывать блочным операторам (наприме, begin/end, {}), подключениям библиотек и заголовочных файлов.

2) Строим строку из полученных кодов, сохраняя порядок следования их в исходном коде программы. Один символ строки (токен) код одного оператора.

Таким образом мы автоматически игнорируем названия функций и переменных (классов, объектов и так далее), разделительные символы, предотвращаем влияние мелких изменений кода программы.

Нужно отметить, что процесс токенизации и разбиение операторов на классы зависим от используемого в исходном коде языка программирования.

К одному классу операторов обычно относят те, которым соответствует один идентификатор языка программирования, все вызовы функций, вызовы методов классов, объявления переменных элементарных типов, объявление экземпляров классов. Более подробную информацию можно посмотреть в приложении к [4].

Анализ существующих алгоритмов

В данном алгоритме над кодом программы выполняется токенизация в результате получается строка токенов, которая впослідствии разбиваются на k-граммы, строится таблица в которой записан каждый k-грамм, а так же его частота встречаемости в строке, чем меньше частота встречаемости, тем больше вес данного k-грамма [5]. По частоте встречаемости высчитывается процент каждого k-грамма в коде.

При сравнении находятся все совпавшие k-граммы и в программе из базы суммируются проценты данных k-граммов в результате выводится процент схожести.

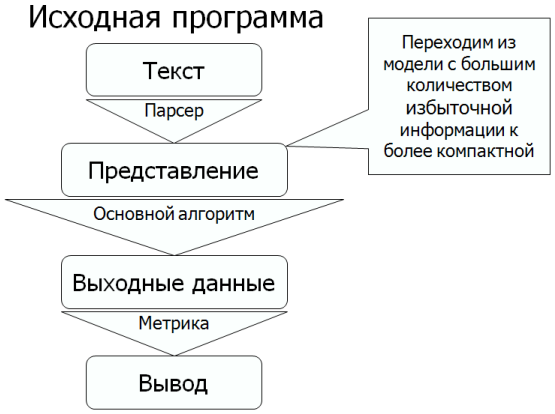


Рис.1. Общая схема поиска плагиата.

Алгоритм Хескела

Достоинства:

1) Линейная трудоемкость алгоритма

Недостатки:

2) Возможность совпадения токенизированного представления программ, но отсутствия совпадения в исходных кодах программ.

3) Небольшое количество уникальных k-граммов в больших программах, соответственно многие совпадения, не содержащие в себе таких k-граммов будут проигнорированы.

4) Вставка в найденный блок или изменение на семантически эквивалентный оператора во многих случаях будет приводить к игнорированию той части блока, в которой не содержится уникальный k-грамм.

5) Нельзя организовать базу данных, ускоряющую проверку один- против-всех.

Метод отпечатков

В этом алгоритме токенезированная программа представляется в виде набора отпечатков (меток, fingerprints), так чтобы эти наборы для похожих программ пересекались. Этот метод позволяет организовать эффективную проверку по базе данных. Метод отпечатков можно представить в виде четырех нижеследующих шагов:

1) Последовательно хэшируем подстроки токенизированной программы P длины k (фиксированный параметр).

2) Выделяем некоторое подмножество их хэш-значений, хорошо характеризующее P. Проделываем те же шаги для токенизированных программ $T_1, T_2 ... T_n$ и помещаем их выбранные хэш-значения в хэш-таблицу.

3) С помощью хэш-таблицы (базы) получаем набор участков строки P , подозрительных на плагиат.

4) Анализируем полученные на предыдущем шаге данные и делаем выводы.

Существует несколько реализаций шага 2, но наиболее качественный это использование метода просеивания [6].

Достоинства:

1) Можно организовать базу данных, ускоряющую проверку один-против-всех.

2) Преимущества токенизированного представления.

3) Общие подстроки, меньше пороговой длины игнорируются, поэтому алгоритм не принимает в расчет малые случайно совпавшие участки кода.

4) При разбиении совпавшего участка кода на две и более части вставкой одного-нескольких блоков или одиночных операторов, а также перестановкой небольшого количества независимых операторов, функция схожести слабо изменяется. (Длина совпадения должна быть значительно больше некоторой пороговой длины.)

5) Алгоритм нечувствителен к перестановкам больших фрагментов кода.

Недостатки:

1) Возможность совпадения токенизированного представления программ, но отсутствия совпадения в исходных кодах программ.

2) Разбиение совпадения на блоки, вставкой или заменой оператора на похожий (например, for на while), каждый длиной меньше k, ведет к полному игнорированию совпадения.

Алгоритм выравнивания строк

Пусть у нас есть две программы, представим их в виде строк токенов s и t соответственно (возможно различной длины). Теперь мы можем воспользоваться методом локального выравнивания строк, разработанным для определения схожести строк ДНК [7]. Выравнивание двух строк получается с помощью вставки в них пробелов таким образом, чтобы их длины стали одинаковыми. Заметим, что существует большое количество различных выравниваний двух строк.

Достоинства:

1) Использование процедуры кластеризации.

2) Токенизированное представление программ

Недостатки:

1) На базе этого алгоритма нельзя организовать базу данных, ускоряющую проверку один-против-всех.

Алгоритм жадного строкового замощение

Эвристический алгоритм получения жадного строкового замощения (The Greedy String Tiling, см. [8]), получает на вход две строки символов над определенным алфавитом (у нас это множество допустимых токенов), а на выходе дает набор их общих непересекающихся подстрок близкий к оптимальному.

Достоинства:

1) Преимущества токенизированного представления.

2) Общие подстроки меньшей длины, чем MinimumMatchLength игнорируются, поэтому алгоритм не принимает в расчет небольшие случайно совпавшие участки кода.

3) При разбиении совпавшего участка кода на две и более части вставкой одного-нескольких блоков или одиночных операторов, а также перестановкой небольшого количества независимых операторов, функция схожести слабо изменяется.

4) Алгоритм нечувствителен к перестановкам больших фрагментов кода.

Недостатки:

1) Возможноть совпадения токенизированного представления программ, но отсутствия совпадения в исходных кодах программ.

2) Разбиение совпадения на блоки, вставкой или заменой оператора на похожий (например, for на while), каждый длиной меньше MinimumMatchLength, ведет к полному игнорированию совпадения.

3) Из-за эвристик, используемых в алгоритме, совпадения, длиной меньшей чем MinimumMatchLength, будут проигнорированы.

4) Нельзя организовать базу данных, ускоряющую проверку один-против-всех.

Выводы

Наибольшим количеством достоинств является «Метод отпечатков», который разрешает поддержку полноценной базы большого размера, позволяющую ускорить процедуру провер-

ки один-против-всех, так же относительно данной базы можно искать плагиат в любом другом исходном коде за приемлемое время, а не проверять образец с каждым элементом базы, это происходит из-за того, что этот алгоритм использует хеш-таблицы.

Все описанные алгоритмы, кроме метода отпечатков, не могут эффективно работать с большой базой, так как требуют проведения сравнений образца с каждым элементом базы.

Алгоритм Хескела имеет множество недостатков относительно метода отпечатков и метода жадного строкового замощения, потому, что производит сравнение по k-граммам, а не по хеш значениям, однако если применить данный алгоритм к уже полученным меткам в методе отпечатков, то можно ускорить процесс сравнения.

Предложения по улучшению рассмотренных алгоритмов

При выполнении токенизации, ввести одинаковые коды на схожие операторы, например for и while или double и float, что бы соответствовали одному коду токенизации. Далее ввести понятие вложенный токен, например если у нас есть целочисленная функция Test, которая содержит в себе цикл for и который в свою очередь содержит переменные типа int и float.

```
int Test ()
{
    for
   {
     int x;
     float y;
    }
}
```

И например целочисленной переменной соответствует токен a, а циклу for токен b и переменным int и float соответственно токены c и d, то вложенный токен будет иметь вид: a{b{cd}}, который при указании k-грамма будет приравниваться одному символу, например если k=3, то он будет иметь вид (рисунок 2):

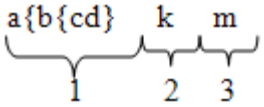


Рис. 2. Схема вложенного токина.

где a{b{cd}} – первый символ, k-второй, m-третий.

Так же предлагается преобразовать метод просеивания который используется для получения меток из хеш значений, что бы уже когда создано окно для поиске меток, то если в нем имеются хеш значения, полученные из внутреннего токена, то они обязательно становятся метками, а уже если таких значений нет, то выбирается наименьшее значение в окне, данный подход дожжен позволить находить частичный плагиат, например одинаковые функции. Когда будут получены метки, то они сохраняются в таблице поиска типа:

Табл. 2. Таблица поиска

Метка	k-грамм	Имя файла	строка	Метод (цикл)
12 4	3	Code.txt .	5 .	+ -

Когда будет производится сравнение на плагиат, то сравнимая программа которая имеет свой набор меток, будет сверятся с каждой меткой в таблице поиска, так можно будет сделать вывод в процентном соотношении к какому файлу она наиболее близка. Последний столбец Метод (цикл) существует для того, что бы показать что в данной метке находится вложенный токен, который изначально имеет также свое хеш значение, но оно хранится в таблице сравнения , например если мы имеем k-грамм a{b{cd}}k m и он например имеет значение 12, то мы видим что в данном k-грамме есть вложенный токен, которому отдельно выдается хеш значение, например 8 которое записывается в таблицу сравнения, которая будет иметь вид:

Табл. 3. Таблица Сравнения

Хеш значение вложенного точена	Хеш значеие всего k-грамма
8	12

Это делается для того, что бы даже если у k-граммов которые имеют разные значения хеш функций, но имеют одинаковый вложенный токен было найдено по нему совпадение.

Предложенные Алгоритмы

Вариант 1:

1) Производим токенизацию, так же создаются вложенные токены.

2) Выполняем разбиение на K-граммы (n = m-(k-1), где n- количество K-грамм, m- длинна текста).

3) Получаем хеш значения с помощью хеш-функции для каждого K-грамма, а также для каждого вложенного токена.

4) Используем метод просеивания для выбора меток из всех существующих хеш значений (выбираем окно размерностью w=t–k+1, если в данном окне есть хеш значение вложенного токена, то оно обязательно становится меткой, если же таких значений нет, то в окне выбираем найменьшое хеш значение, которое становиться меткой, сравниваем все метки, и если есть метки с одинаковыми значениями, то они записываются в таблицы только один раз, метки полученные от K-граммов записываются в таблицу поиска, а метки от вложенных токенов в таблицу сравнения.)

Вариант 2:

1) Производим токенизацию, так же создаются вложенные токены.

2) Выполняем разбиение на K-граммы (n = m-(k-1), где n- количество K-грамм, m- длинна текста).

3) Получаем хеш значения с помощью хеш-функции для каждого K-грамма, а также для каждого вложенного токена.

4) Используем метод просеивания для выбора меток из всех существующих хеш значений (выбираем окно размерностью w=t–k+1, если в данном окне есть хеш значение вложенного токена, то оно обязательно становится меткой, если же таких значений нет, то в окне выбираем найменьшое хеш значение, которое становиться меткой, метки полученные от K-граммов записываются в таблицу поиска, а метки от вложенного токена в таблицу сравнения.)

5) Используем метод Хескела к полученным меткам, после чего создается таблица, где каждой метке указывается ее вес (чем меньше раз данная метка встречается в таблице поиска и в таблице сравнения, тем больше ее вес).

На рисунке 3 представлена Диаграмма классов для разрабатываемой системы

Анализ предложенных вариантов

В данных вариантах предлагается ввести понятие вложенный токен, что позволит находить плагиат отдельных функций в кодах программ. Введение вложенных токенов влечет за собой модификацию метода просеивания, в котором изменяется способ отбора меток в окне. Первый вариант алгоритма должен работать быстрее второго, так как во втором варианте будет строится еще и третья таблица для метода Хескела. За счет уменьшения быстродействия должно увеличится качество поиска относительно первого варианта.

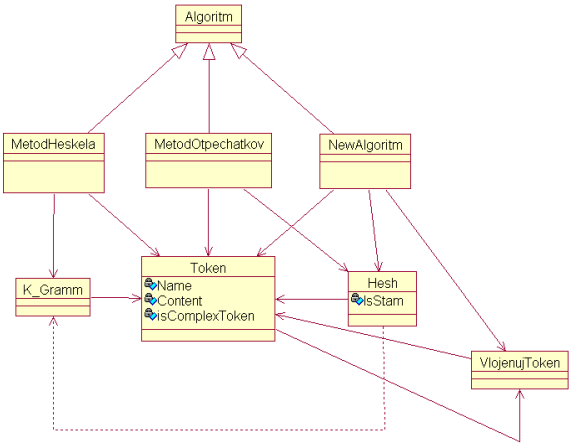


Рис. 2. Диаграмма классов

Экспериментальный анализ работы алгоритмов

Проведем экспериментальный анализ описанных выше алгоритмов, целью которого является поиск скрытого плагиата.

Анализ работы алгоритмов средствами ПК

В эксперименте используем несколько идентичных проектов на языке C# платформы .NET framework компании Microsoft (в данном эксперименте используются три проекта с приблизительным количеством строк кода равным 2150).

Один из проектов оставляем без изменений, в остальных моделируем некоторые попытки скрыть плагиат:

- замена названий переменных (методов);
- изменение последовательности чередования методов (функций), переменных;
- изменение типов объявления переменных;
- изменение типов объявления циклов.

В таблице 4 представлен результат сравнения проектов разными алгоритмами поиска плагиата.

На рисунке 3 полученные данные представлены в виде точечной диаграммы.

Как видно из полученных результатов, лучшие показатели поиска плагиата показал алгоритм описан в варианте 1 (Z-nested 1).

Табл. 4. Результаты сравнения проектов разными алгоритмами поиска плагиата

№	Файл проекта с базы	Алго-ритм	Время срав-нения (мс)	Похожесть (%)
1	CodePlagiat2.csproj	Z-nested 1	661	100
2	CodePlagiat.csproj	Z-nested 1	498	96,4285714285714
3	CodePlagiat1.csproj	Z-nested 1	368	95,7908163265306
4	CodePlagiat2.csproj	Heskel	99	100
5	CodePlagiat.csproj	Heskel	670	78,5714285714289
6	CodePlagiat1.csproj	Heskel	193	77,9038718291058
7	CodePlagiat2.csproj	Z-nested 2	519	100
8	CodePlagiat.csproj	Z-nested 2	263	83,9080459770115
9	CodePlagiat1.csproj	Z-nested 2	69	85,0574712643678

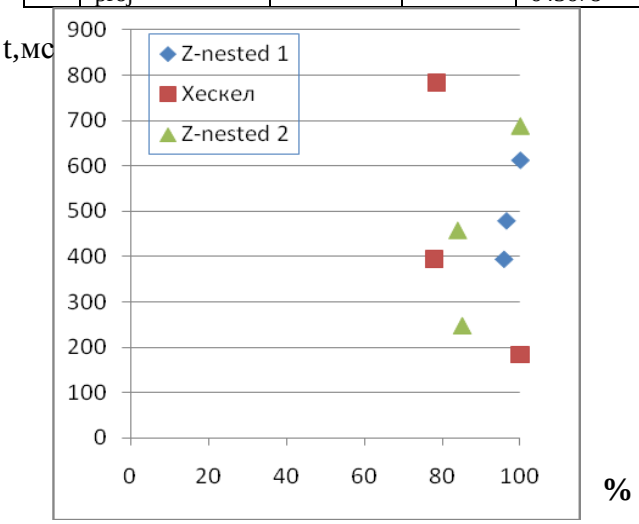


Рис. 3. Точечная диаграмма сравнения алгоритмов поиска плагиата

Анализ работы алгоритмов пошаговым расчетом

Представлено подробное описание работы алгоритмов с указанными примерами.

Алгоритм Хескела

Рассмотрим на приведенных примерах программ работу алгоритма Хескела.

1)Строим строку проекта в виде последовательности токенов.

Пример №1.

080|011|080|004|076|080|060|037|060|017|081|076|080|034|077|080|029|033|080|060|081|081|081|081|081

Пример №2.

080|011|080|004|076|080|034|077|080|029|033|080|060|081|081|081|076|080|060|037|060|017|081|081|081

Пример №3.

080|011|080|004|076|080|034|029|033|080|028|082|080|060|081|081|081|076|080|060|037|060|017|081|081|081

Пример №4.

080|011|080|004|076|080|029|033|080|028|082|080|060|081|081|034|081|076|080|060|037|060|017|081|081|081

Далее выбираем проект, который хотим сравнить с остальными. Допустим, это будет проект Пример №1 (далее основной проект).

2) Строим последовательность k-граммов каждого проекта с учетом количества их встречаемости. Допустим, количество токенов в одном к-грамме равно 3. Тогда имеем следующее:

Табл. 4. Пример №1.

№	К-грамм	Частота встреча-емости	Вес (%)
1	080|011|080	1	4,34782608695652
2	011|080|004	1	4,34782608695652
3	080|004|076	1	4,34782608695652
4	004|076|080	1	4,34782608695652
5	076|080|060	1	4,34782608695652
6	080|060|037	1	4,34782608695652
7	060|037|060	1	4,34782608695652
8	037|060|017	1	4,34782608695652
9	060|017|081	1	4,34782608695652
10	017|081|076	1	4,34782608695652
11	081|076|080	1	4,34782608695652
12	076|080|034	1	4,34782608695652
13	080|034|077	1	4,34782608695652
14	034|077|080	1	4,34782608695652
15	077|080|029	1	4,34782608695652
16	080|029|033	1	4,34782608695652
17	029|033|080	1	4,34782608695652
18	033|080|060	1	4,34782608695652
19	080|060|081	1	4,34782608695652
20	060|081|081	1	4,34782608695652
21	081|081|081	3	13,0434782608696

Табл. 5. Пример №2.

№	К-грамм	Частота встреча-емости	Вес (%)
1	080|011|080	1	4,34782608695652
2	011|080|004	1	4,34782608695652
3	080|004|076	1	4,34782608695652
4	004|076|080	1	4,34782608695652
5	076|080|034	1	4,34782608695652
6	080|034|077	1	4,34782608695652
7	034|077|080	1	4,34782608695652
8	077|080|029	1	4,34782608695652
9	080|029|033	1	4,34782608695652

10	029|033|080	1	4,34782608695652
11	033|080|060	1	4,34782608695652
12	080|060|081	1	4,34782608695652
13	060|081|081	1	4,34782608695652
14	081|081|081	2	8,69565217391304
15	081|081|076	1	4,34782608695652
16	081|076|080	1	4,34782608695652
17	076|080|060	1	4,34782608695652
18	080|060|037	1	4,34782608695652
19	060|037|060	1	4,34782608695652
20	037|060|017	1	4,34782608695652
21	060|017|081	1	4,34782608695652
22	017|081|081	1	4,34782608695652

Табл. 5. Пример №3

№	К-грамм	Частота встречаемости	Вес (%)
1	080|011|080	1	4,16666666666667
2	011|080|004	1	4,16666666666667
3	080|004|076	1	4,16666666666667
4	004|076|080	1	4,16666666666667
5	076|080|034	1	4,16666666666667
6	080|034|029	1	4,16666666666667
7	034|029|033	1	4,16666666666667
8	029|033|080	1	4,16666666666667
9	033|080|028	1	4,16666666666667
10	080|028|082	1	4,16666666666667
11	028|082|080	1	4,16666666666667
12	082|080|060	1	4,16666666666667
13	080|060|081	1	4,16666666666667
14	060|081|081	1	4,16666666666667
15	081|081|081	2	8,33333333333333
16	081|081|076	1	4,16666666666667
17	081|076|080	1	4,16666666666667
18	076|080|060	1	4,16666666666667
19	080|060|037	1	4,16666666666667
20	060|037|060	1	4,16666666666667
21	037|060|017	1	4,16666666666667
22	060|017|081	1	4,16666666666667
23	017|081|081	1	4,16666666666667

Табл. 6. Пример №4

№	К-грамм	Частота встречаемости	Вес (%)
1	080|011|080	1	4,16666666666667
2	011|080|004	1	4,16666666666667
3	080|004|076	1	4,16666666666667
4	004|076|080	1	4,16666666666667
5	076|080|029	1	4,16666666666667
6	080|029|033	1	4,16666666666667
7	029|033|080	1	4,16666666666667
8	033|080|028	1	4,16666666666667
9	080|028|082	1	4,16666666666667
10	028|082|080	1	4,16666666666667
11	082|080|060	1	4,16666666666667
12	080|060|081	1	4,16666666666667
13	060|081|081	1	4,16666666666667
14	081|081|034	1	4,16666666666667
15	081|034|081	1	4,16666666666667
16	034|081|076	1	4,16666666666667
17	081|076|080	1	4,16666666666667
18	076|080|060	1	4,16666666666667
19	080|060|037	1	4,16666666666667
20	060|037|060	1	4,16666666666667
21	037|060|017	1	4,16666666666667
22	060|017|081	1	4,16666666666667
23	017|081|081	1	4,16666666666667
24	081|081|081	1	4,16666666666667

3)Основываясь на описанных выше таблицах выполняем сравнение проектов. Если k-грамм присутствует в основном и в сравниваемом проектах, то учитывается его вес в %, который он занимает в основном. Далее в таблице приведен результат сравнения.

Табл. 7. Результат сравнения

№	Основной проект	Сравниваемый проект	Процент схожести (%)
1	Пример №1	Пример №2	91,304347826087
2	Пример №1	Пример №3	66,6666666666667
3	Пример №1	Пример №4	62,5

Алгоритм Z-nested 1

Рассмотрим на приведенных примерах программ работу алгоритма описанного в варианте №1 (Z-nested 1).

1) Переводим проект в последовательность токенов. Далее получаем хеш значения к-граммов. Допустим, размер к-грамма равен 3, тогда имеем следующий вид проектов:

Табл. 8. Пример №1

№	Хеш-значение к-грамма	К-грамм
1	-1494513560	080|011|080
2	378461068	011|080|004
3	-390511174	080|004|076{080060060081}
4	1776990261	004|076{080060060081}|028{080060081}
5	95065247	076{080060060081}|028{080060081}|080028{080060081}081
6	-1505237569	028{080060081}|080028{080060081}081|028{080028{080060081}081}
7	-1005543780	080028{080060081}081|028{080028{080060081}081}|080034028{080028{080060081}081}081
8	-1407541811	028{080028{080060081}081}|080034028{080028{080060081}081}0

		81|076{080034028{080028{08006 0081}081}081}
9	-499040212	080034028{080028{080060081}08 1}081|076{080034028{080028{08 0060081}081}081}|081

Табл. 8. Пример №2.

№	Хеш-значение к-грамма	К-грамм
1	-1494513560	080|011|080
2	378461068	011|080|004
3	-1906288165	080|004|028{080060081}
4	-823501287	004|028{080060081}|080028{080 060081}081
5	-1505237569	028{080060081}|080028{0800600 81}081|028{080028{080060081}0 81}
6	-1005543780	080028{080060081}081|028{0800 28{080060081}081}|080034028{0 80028{080060081}081}081
7	-1407541811	028{080028{080060081}081}|080 034028{080028{080060081}081} 081|076{080034028{080028{0800 60081}081}081}
8	-1573902207	080034028{080028{080060081}0 81}081|076{080034028{080028{0 80060081}081}081}|076{0800600 60081}
9	783746616	076{080034028{080028{0800600 81}081}081}|076{080060060081} |081

Табл. 9. Пример №3.

№	Хеш-значение к-грамма	К-грамм
1	-1494513560	080|011|080
2	378461068	011|080|004
3	-1906288165	080|004|028{080060081}
4	-823501287	004|028{080060081}|080028{0800 60081}081
5	-1505237569	028{080060081}|080028{0800600 81}081|028{080028{080060081}0 81}
6	-1005543780	080028{080060081}081|028{0800 28{080060081}081}|080034028{0 80028{080060081}081}081
7	-1407541811	028{080028{080060081}081}|080 034028{080028{080060081}081}0 81|076{080034028{080028{08006 0081}081}081}
8	-1573902207	080034028{080028{080060081}08 1}081|076{080034028{080028{08
		0060081}081}081}|076{08006006 0081}
9	783746616	076{080034028{080028{08006008 1}081}081}|076{080060060081}|0 81

Табл. 10. Пример №4

№	Хеш-значение к-грамма	К-грамм
1	-1494513560	080|011|080
2	378461068	011|080|004
3	-1906288165	080|004|028{080060081}
4	-823501287	004|028{080060081}|080028{080 060081}081
5	-1505237569	028{080060081}|080028{080060 081}081|028{080028{080060081 }081}
6	-1555898206	080028{080060081}081|028{080 028{080060081}081}|080028{08 0028{080060081}081}034081
7	670774381	028{080028{080060081}081}|08 0028{080028{080060081}081}03 4081|076{080028{080028{08006 0081}081}034081}
8	-1386322015	080028{080028{080060081}081} 034081|076{080028{080028{080 060081}081}034081}|076{08006 0060081}
9	-144113634	076{080028{080028{080060081} 081}034081}|076{080060060081 }|081

2) Используем метод просеивания для выбора меток из всех существующих хеш значений к-граммов. Допустим, окно w используемое в алгоритме равно 2, тогда имеем следующие метки проектов:

Табл. 11. Пример №1

№	Хеш-значение к-грамма	К-грамм
1	-1494513560	080|011|080
2	-390511174	080|004|076{080060060081}
3	1776990261	004|076{080060060081}|028{080 060081}
4	95065247	076{080060060081}|028{080060 081}|080028{080060081}081
5	-1505237569	028{080060081}|080028{080060 081}081|028{080028{080060081 }081}
6	-1005543780	080028{080060081}081|028{080 028{080060081}081}|080034028 {080028{080060081}081}081
7	-1407541811	028{080028{080060081}081}|08 0034028{080028{080060081}081 }081|076{080034028{080028{08 0060081}081}081}
8	-499040212	080034028{080028{080060081}0

		81}081|076{080034028{080028{080060081}081}081}|081

Табл. 12. Пример №2

№	Хеш-значение к-грамма	К-грамм
1	-1494513560	080|011|080
2	-1906288165	080|004|028{080060081}
3	-823501287	004|028{080060081}|080028{080060081}081
4	-1505237569	028{080060081}|080028{080060081}081|028{080028{080060081}081}
5	-1005543780	080028{080060081}081|028{080028{080060081}081}|080034028{080028{080060081}081}081
6	-1407541811	028{080028{080060081}081}|080034028{080028{080060081}081}081|076{080034028{080028{080060081}081}081}
7	-1573902207	080034028{080028{080060081}081}081|076{080034028{080028{080060081}081}081}|076{080060060081}
8	783746616	076{080034028{080028{080060081}081}081}|076{080060060081}|081

Табл. 13. Пример №3

№	Хеш-значение к-грамма	К-грамм
1	-1494513560	080|011|080
2	-1906288165	080|004|028{080060081}
3	-823501287	004|028{080060081}|080028{080060081}081
4	-1505237569	028{080060081}|080028{080060081}081|028{080028{080060081}081}
5	-1005543780	080028{080060081}081|028{080028{080060081}081}|080034028{080028{080060081}081}081
6	-1407541811	028{080028{080060081}081}|080034028{080028{080060081}081}081|076{080034028{080028{080060081}081}081}
7	-1573902207	080034028{080028{080060081}081}081|076{080034028{080028{080060081}081}081}|076{080060060081}
8	783746616	076{080034028{080028{080060081}081}081}|076{080060060081}|081

Табл. 14. Пример №4

№	Хеш-значение к-грамма	К-грамм
1	-1494513560	080|011|080
2	-1906288165	080|004|028{080060081}
3	-823501287	004|028{080060081}|080028{080060081}081
4	-1505237569	028{080060081}|080028{080060081}081|028{080028{080060081}081}
5	-1555898206	080028{080060081}081|028{080028{080060081}081}|080028{080028{080060081}081}034081
6	670774381	028{080028{080060081}081}|080028{080028{080060081}081}034081|076{080028{080028{080060081}081}034081}
7	-1386322015	080028{080028{080060081}081}034081|076{080028{080028{080060081}081}034081}|076{080060060081}
8	-144113634	076{080028{080028{080060081}081}034081}|076{080060060081}|081

Далее выбираем проект, который хотим сравнить с остальными. Допустим это будет проект Пример №1 (далее основной проект).

3) При сравнении основного проекта с сравниваемыми учитываются вложенные к-граммы. Т.е., если хеш-значения к-граммов не совпадают, но при этом вложенные токены к-граммов одинаковые, то к-граммы считаются одинаковыми.

Табл. 15. Результат сравнения

№	Основной проект	Сравниваемый проект	Процент схожести (%)
1	Пример №1	Пример №2	100
2	Пример №1	Пример №3	100
3	Пример №1	Пример №4	87,5

Алгоритм Z-nested 2

Рассмотрим на приведенных примерах программ работу алгоритма описанного в варианте №2 (Z-nested 2).

Пункты 1 и 2 аналогичны описанным в работе алгоритма **Z-nested 1.**

3) Далее применяем алгоритм Хескела к полученным меткам. Допустим, количество меток в одном к-грамме равно 3. Тогда имеем следующие значения к-граммов построенных из меток:

Табл. 16. Пример №1

№	К-грамм меток хеш значений	К-грамм токенов хеш значений
1	-1494513560|-390511174|1776990261	080|011|080*080|004|076{080060060081}*004|076{080060060081}|028{080060081}

2	95065247|-1505237569|-1005543780	076{080060060081}|028{080060081}|080028{080060081}081*028{080060081}|080028{080060081}081|028{080028{080060081}081}*080028{080060081}081|028{080028{080060081}081}|080034028{080028{080060081}081}081

Табл. 17. Пример №2

№	К-грамм меток хеш значений	К-грамм токенов хеш значений
1	-1494513560|-1906288165|-823501287	080|011|080*080|004|028{080060081}*004|028{080060081}|080028{080060081}081
2	-1505237569|-1005543780|-1407541811	028{080060081}|080028{080060081}081|028{080028{080060081}081}*080028{080060081}081|028{080028{080060081}081}|080034028{080028{080060081}081}081*028{080028{080060081}081}|080034028{080028{080060081}081}081|076{080034028{080028{080060081}081}081}

Табл. 18. Пример №3

№	К-грамм меток хеш значений	К-грамм токенов хеш значений
1	-1494513560|-1906288165|-823501287	080|011|080*080|004|028{080060081}*004|028{080060081}|080028{080060081}081
2	-1505237569|-1005543780|-1407541811	028{080060081}|080028{080060081}081|028{080028{080060081}081}*080028{080060081}081|028{080028{080060081}081}|080034028{080028{080060081}081}081*028{080028{080060081}081}|080034028{080028{080060081}081}081|076{080034028{080028{080060081}081}081}

Табл. 19. Пример №4

№	К-грамм меток хеш значений	К-грамм токенов хеш значений
1	-1494513560|-1906288165|-823501287	080|011|080*080|004|028{080060081}*004|028{080060081}|080028{080060081}081
2	-1505237569|-1555898206|670774381	028{080060081}|080028{080060081}081|028{080028{080060081}081}*080028{080060081}081|028{080028{080060081}081}|080028{080028{080060081}081}034081*028{080028{080060081}081}|080028{080028{080060081}081}034081|076{080028{080028{080060081}081}034081}

4)При сравнении основного проекта с сравниваемыми учитываются вложенные токены. Т.е., если к-граммы разные, но при этом вложенные токены к-граммов одинаковые, то к-граммы считаются одинаковыми.

Табл. 20. Результат сравнения

№п.п	Основной проект	Сравниваемый проект	Процент схожести (%)
1	Пример №1	Пример №2	100
2	Пример №1	Пример №3	100
3	Пример №1	Пример №4	100

Список литературы

1. Большой юридический словарь Под ред. А.Я. Сухарева, В.Е. Крутских., М., 2002
2. Faidhi, J. A. W. and S. K. Robinson. An Empirical Approach for Detecting Program Similarity within a University Programming Environment. Computers and Education 11(1): pp. 1119 (1987).
3. B. S. Baker. On Finding Duplication and Near-Duplication in Large Software Systems. In Proceedings of the second IEEE Working Conference on Reverse Engineering (WCRE), July 1995, pp. 86–95.
4. L. Prechelt, G. Malpohl, and M. Philippsen. JPlag: Finding plagiarisms among a set of programs. Technical Report No. 1/00, University of Karlsruhe, Department of Informatics, March 2000.
5. Heckel, Paul. A Technique for Isolating Di erences Between Files. Communications of the ACM 21(4), pp. 264–268 (April 1978).
6. Aiken, S. Schleimer, D. Wikerson. Winnowing: local algorithms for document fingerprinting. In Pro ceedings of ACMSIGMOD Int. Conference on Management of Data, San Diego, CA, June 9–12, pp. 76–85. ACM Press, New York, USA, 2003.
7. X. Huang, R. C. Hardison, AND W. Miller. A space-e cient algorithm for local similarities. Computer Applications in the Biosciences, 6 (1990), pp. 373-381. Michael J. Wise. String similarity via greedy string tiling and running Karp-Rabin matching. Dept. of CS, University of Sydney, December 1993.

УДК 681.3.016

УЛЬЯНИЦКАЯ К.А.,
КУНЩИКОВ Е.О.,
ОСТРОВСКИЙ С.М.

ПОДХОД К ИСПОЛЬЗОВАНИЮ МАТРИЧНОГО МЕТОДА ДЛЯ АВТОМАТИЗАЦИИ БИЗНЕС ПРОЦЕССОВ

This material is devoted to the use of matrix methods to create and automate business processes. Considered a specific business process "Manufacturing production" and his support through SEDO. Was carried out general analysis of documents prepared by the state diagram of BP formed BZ logical rules movement control documents. To the obtained system of logical rules applied matrix method for the evaluation of the probability of working off for the business processes. Module was designed to optimize the business process. And the comparative analysis was made. A system of logical rules will be used to automate BP "Making products" based on SEDO. The matrix method will be used to determine the probability of working off of documents entering.

Данный материал посвящен использованию матричного метода при создании и автоматизации бизнес процессов. Рассмотрен конкретный бизнес-процесс «Изготовление продукции» и его поддержка на основе Седо. Был проведен общий анализ документооборота организации, составленная диаграмма состояний БП, сформированная БЗ логических правил управления движением документов. К полученной системы логических правил применен матричный метод для оценки достоверности отработки документа для данного БП. Был разработан модуль для оптимизации бизнес процесса. И проведен сравнительный анализ. Полученная система логических правил будет использована для автоматизации БП «Изготовление продукции» на основе Седой. Матричный метод будет использован для определения вероятности отработки документов, поступающих.

Поддержка бизнес-процесса «Изготовление продукции» на основе СЕДО

Необходимо начать с общего анализа документооборота организации, который должен помочь определить набор операций над документами, политику маршрутизации документа на всех этапах его жизненного цикла. В общем случае документооборот организации включает в себя следующие процессы:

- регистрация документа: после получения документа извне или создания внутри, документ регистрируется, т.е. на документ заводится регистрационная карточка, ему присваивается уникальный «идентификатор;
- анализ документа: результатом такого анализа документа будет в простейшем случае ответный документ и переход документа на завершающую стадию, в других же случаях, и их большинство, – результатом анализа будет документ (резолюция), прикрепляющаяся к документу, и являющаяся поручением, которое нужно будет выполнить соответствующим лицам, определенным в резолюции;
- обработка документа: движение документа и его копий; документ/копии документа после анализа чаще всего попадают к исполнителям, которых может быть достаточно большое количество; выполнение документа, естественно, контролируется определенными должностными лицами, что также должно быть учтено при построении системы управления движением документов; при этом схемы движения документа имеют достаточно сложную структуру с циклами, разветвлениями и последовательно выполняемыми операциями;
- завершающая стадия – отправка документа.

Определено три основных состояния: когда документ находиться в канцелярии, у главного технолога и на обработке. Все три состояния являются составными, так как состоят из не скольких подсостояний.

Опишем переходы на диаграмме:

- Зарегистрировать документ;
- Проанализировать документ;
- Если документ исходящий или в результате анализа не требуется его дальнейшая обработка и рассмотрение, то его отправляют по назначению;
- Документ поставлен на контроль;
- На документ наложена резолюция;
- Документ отправлен на обработку;
- Обработать документ;

- Сформировать отчет по отработке документа;
- Согласование с ПЭО;
- Отправка на подпись к главному технологу;
- Снятие с контроля;
- Отправка документа.

Формирование БЗ логических правил управления движением документов

Это наиболее ответственный этап. Неверные правила, отсутствие правил для некоторых ситуаций приводят к неправильным решениям, сбоям в функционировании документооборота. Правила имеют вид клауз Хорна: [1]

$A_1 \leftarrow B_1, B_2, \ldots, B_n$

Подразумевается, что A_1 выполняется, если выполняются условия $B_1, B_2, \ldots, B_n$.

Клаузальная логика – это форма стандартной логики (классической) и отличается от нее системой обозначений. В основе стандартной формы логики лежит логика высказываний (пропозициональная логика) и логика предикатов. При записи предложений в клаузальной форме кванторы в явном виде не указываются, но предполагается, что все переменные связаны квантором всеобщности. В примере выше B1,...,Bm, A1,...,An – это атомарные формулы ($n \geq 0$, $m \geq 0$), A1,...,An – это совместные посылки клаузы, а B1,...,Bm − это альтернативные заключения. В стандартной форме клауза равносильна записи A1 & ...& An → B1 ∨ ... ∨ Bm.

Таким образом, запятые в посылке клаузы означают конъюнктивные связки между предложениями, составляющими посылку, а запятые в заключении клаузы символизируют дизъюнктивные связки между предложениями, составляющими посылку. Множество клауз невыполнимо, если из него можно вывести пустое предложение (обозначаемое). [2]

Наиболее важной характеристикой, используемой в правилах управления движением документов, является состояние документа. Вводим переменную состояния **z,** которая может принимать значения из множества состояний определенных в схеме.

Определим основные состояния документа:

Z1- документ зарегистрирован
Z2- документ проанализирован
Z3- документ поставлен на контроль
Z4- документ рассмотрен, резолюция
Z5.1, Z5.2, Z5.3, Z5.4– документ исполнен соответствующими отделами: отдел снабжения, отдел комплектации, отдел электротехнической подготовки, инструментальное производство.
Z6.1, Z6.2, Z6.3, Z6.4- создан отчет об обработке документа: отдел снабжения, отдел комплектации, отдел электротехнической подготовки, инструментальное производство.
Z7- формирование ответа
Z8- согласование
Z9- документ подписан
Z10- документ снят с контроля
Z11- документ отправлен

Клаузы Хорна нашей системы будут иметь следующий вид, где число рядом с стрелочкой отображает вероятность с какой условие обеспечивает последствие:

1) $|\rightarrow_1 Z1$
2) $|\rightarrow_1 (Z1 \rightarrow Z2)$
3) $|\rightarrow_{0.7} (Z2 \rightarrow Z3)$
4) $|\rightarrow_{0.95} (Z2 \rightarrow Z4)$
5) $|\rightarrow_{0.9} ((Z2 \wedge Z4) \rightarrow (Z5.1 \wedge Z5.2 \wedge Z5.3 \wedge Z5.4))$
6) $|\rightarrow_1 (Z5.1 \rightarrow Z6.1)$
7) $|\rightarrow_1 (Z5.2 \rightarrow Z6.2)$
8) $|\rightarrow_1 (Z5.3 \rightarrow Z6.3)$
9) $|\rightarrow_1 (Z5.4 \rightarrow Z6.4)$
10) $|\rightarrow_1 ((Z6.1 \wedge Z6.2 \wedge Z6.3 \wedge Z6.4) \rightarrow Z7)$
11) $|\rightarrow_{0.9} (Z7 \rightarrow Z8)$
12) $|\rightarrow_{0.95} ((Z7 \wedge Z8) \rightarrow Z9)$
13) $|\rightarrow_1 (Z9 \rightarrow Z10)$
14) $|\rightarrow_1 (Z10 \rightarrow Z11)$

Так как свойства и функции отделов одинаковы, то для упрощения последующих расчетов представляем состояния Z5.1, Z5.2, Z5.3, Z5.4 в общем виде Z5, а Z6.1, Z6.2, Z6.3, Z6.4 в виде Z6.

Для построения атомов логических правил также используются определенные значения атрибутов документа, для учета которых вводим переменные:

- переменная “вид” документа **v,** которая может принимать значения из множества **{V_1(**внутренний документ), **V_2(**входящий документ) и др.**}**;
- переменная “тип” документа **a,** которая может принимать значения из множества **{A_1 (**справка)**,** **A_2 (**поручение)**,** **A_3** (заявка)**,** **A_4 (**приказ) и т.д.**}**;
- переменная “организация” (или человек) откуда документ пришел **o,** которая может принимать значения из множества **{O_1** (НТУ «КПИ»)**,** **O_2** (НАУ)**,** **O_3** (ВАТ «ВаБанк») и т.д.**}**;

- переменная “срочность” документа **c,** которая может принимать значения из множества **{C_1** (срочный)**, C_2** (неотложный)**, C_3**(несрочный)**}**;
- переменная “тематика” документа **t,** которая может принимать значения из множества **{T_1**(финансовый документ)**, T_2**(банковские расчеты), **T_3** (договор) и т.д.**}**.

Рассмотрим примеры правил вывода системы:

Для того, чтобы перевести документ из состояния Z_0 в одно из под-состояний состояния Z_1, необходимо чтобы были выполнены определенные операции и соблюдался ряд условий. Например, для перехода документа в состояние Z_1^1(«Документ зарегистрирован в журнале для входящих общих документов») необходимо, чтобы тип документа был «заявка» ($a=A_3$), организация, из которой он поступил – ВАТ «ВаБанк» ($o=O_3$), документ должен быть срочным ($c=C_1$), а тематика документа – заказ. Для осуществления такого рода действий надо составить соответствующие логические правила:

- Для перехода из состояния Z_0 в состояние «Документ зарегистрирован в журнале для внутренних документов» Z_1^0:

$$(z= Z_1^0) \leftarrow (v=V_1)$$

- Для перехода из состояния Z_0 в состояние «Документ зарегистрирован в журнале для входящих общих документов» Z_1^1:

$$(z= Z_1^1) \leftarrow (a=A_3 , o=O_3 , c=C_1 , t=T_3)$$

- Для перехода из состояния Z_0 в состояние «Документ зарегистрирован в журнале для переходящих документов» Z_1^2:

$$(z= Z_1^2) \leftarrow (a=A_3 , o=O_2 , c=C_3 , t\neq T_2)$$

- Для перехода из состояния Z_0 в состояние «Документ зарегистрирован в журнале для входящих специфических документов» Z_1^3:

$$(z= Z_1^3) \leftarrow (a=A_2 , o=O_1 , c=C_3)$$

- Для перехода из состояния Z_0 в состояние «Документ зарегистрирован в журнале для исходящих документов» Z_1^4:

$$(z= Z_1^4) \leftarrow (a=A_3 , c=C_4)$$

Отобразим схематически один из возможных планов передвижения документа из начального состояния Z_0 в конечное состояние Z_{12} (рис. 1.):

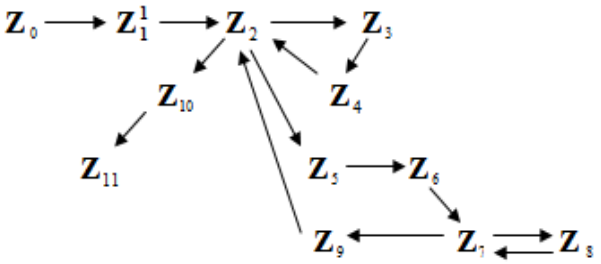


Рис. 1. План (схема) передвижения документа из начального состояния Z_0 в конечное состояние Z_{11}

Переходы между состояниями обозначены направленными стрелками, каждому переходу соответствует определенный набор логических правил.

На рисунке 1 отображен простейший жизненный цикл пришедшего на обработку документа: документ поступил из внешней организации (Z_0), зарегистрирован в соответствующем журнале (Z_1^1), после чего проанализирован (Z_2), после анализа документа он был поставлен на контроль(Z_3) и на него была наложена резолюция (Z_4); после того как документ опять пройдет анализ, в случае необходимости его переслали на исполнение (Z_5), после отработки,в каждом отделе был написан отчет об исполнении документа (Z_6). Затем в отделе кооперации формируется общий отчет(Z_7), который перед тем как пойти на подпись к главному технологу проходит согласование в ПЭО(Z_8). Отчет вместе с документом поступает на подпись вышестоящему начальству (Z_9). Если документ подписан (а в нашем случае это так), то он возвращается на стадию анализа (Z_2), после чего документ снимается с контроля(Z_{10}), и отсылается (Z_{11}).

Реализация метода

Предлагается 2-х этапная схема реализации:

1) Этап вывода. Выводится план (схема) движения документа на основе правил, заложенных в системе. Этот план включает операции, которые предстоит выполнить и условия, которые нужно проверить перед выполнением операции. Поскольку следующее действие зависит от результатов предыдущего, план должен достраиваться в процессе его реализации.

2) Этап осуществления планов. Когда план построен или построена определенная часть плана, необходимо отслеживать условия вы-

полнения операций плана и передавать документ для выполнения соответствующих действий над ним.

Реализацию первого этапа необходимо осуществлять с помощью механизмов вывода. Предлагается рассмотреть матричный метод. [3]

Система должна быть непротиворечивой, то есть у нее должен быть закладений алгоритм анализа правил на противоречивость. Идея заключается в применении матричного представления выходных данных и реализации резолютивного метода в виде операции умножения матриц.

Нечеткий логический вывод

Одной из проблем создания применений есть работа с неопределенностью, неточностью, неполнотой информации. Достаточно часто она бывает нечеткой по своей природе. В классических условиях применяют логико-математические исчисления, причем механизмы выведения лишают алгоритмический подход от некоторых его существенных недостатков. Эффективным и удобным средством для проработки нечетких знаний является нечеткая логика. Предлагая аппарат определения численных значений нечеткости, вычисления выражений с элементами нечеткости, она дает возможность находит решение даже в случае недостаточности данных.

Допустимо, что существует множественное число правил вида:

$$(A1 \wedge A2 \wedge \ldots) \rightarrow (B1 \vee B2 \vee \ldots) \qquad (1)$$

где A1, A2,… – это условия правила, B1, B2,… – возможные последствия.

Для каждого следствия экспертами определена оценка уверенности в том, что он имеет место для выражения, которое удовлетворяет условия правила. Последствия правила могут составлять условия для других правил, и таким образом мы получаем цепочки правил. Каждому следствию определена оценка уверенности в интервале [0,1]. Если условие имеет одно следствие, его оценка уверенности не обязательно равняется 1.

В большинстве случаев существует две и больше возможных последовательностей применения правил выведения. Чтобы определить, которая из них дает конечный результат с большей степенью уверенности, необходим метод вычисления промежуточных результатов на основе степени уверенности условия и степени уверенности того, что определенное условие обеспечит определенное следствие. Другими словами, если известно, что уверенность в A1 равняется к, а уверенность в A1→A2 равняется l, то для последующих вычислений по цепочке или (в случае, если A2 – конечная точка) получения конечной уверенности в целесообразности применения этой цепочки правил необходимо иметь оценку уверенности в A2.

Чтобы получить оценку для A2, обозначим с помощью a′ “не a”, $a \vee b$ – “або”, $a \wedge b$, “і”, $a \rightarrow b$ – импликацию “если a, то b”. В теории нечетких множеств $a' = 1 - a$. Логическому выражению “і” отвечает операция $\otimes$, а выражению “или” – операция $\oplus$. Эти операции определяются через Т-нормы (отражаются знаком *T) и Т-конорми (отражаются знаком +T) соответственно. Самыми распространенными являются такие определения Т-норм и Т-конорм:

$$a *_T b = ab;\ a +_T b = a + b - ab \qquad (2)$$

$$a *_T b = \min\{a, b\};\ a +_T b = \max\{a, b\} \qquad (3)$$

Вычисление выполнять за такой схемой:

Шаг 1. Рассматривать лишь правила вида (1), в каких консеквент содержит не больше одного возможного последствия;

Шаг 2. Если $\vdash_{\rightarrow} к$ A помечает утверждение о том, что A имеет место со степенью уверенности к, то степени уверенности вычислять с помощью формул:

а) если $\vdash_{\rightarrow} k\ P$ і $\vdash_{\rightarrow} l\ Q$, то $\vdash_{\rightarrow} k \otimes l\ P \wedge Q$;

б) если $\vdash_{\rightarrow} k\ P$ і $\vdash_{\rightarrow} l\ Q$, то $\vdash_{\rightarrow} k \oplus l\ P \vee Q$;

в) если $\vdash_{\rightarrow} k\ P$ і $\vdash_{\rightarrow} l\ P \rightarrow Q$, то $\vdash_{\rightarrow} k \otimes l\ Q$;

Шаг 3. Перебор всех возможных цепочек заменить применением матричного метода вывода.

Рассмотрим наш пример:

Сначала применением определения операций (3) и правил выведения шага 2 приведенной выше схемы:

1) вычислим $\vdash_{\rightarrow 1} \otimes_1 Z2$:
   $1 \otimes 1 = \min\{1, 1\} = 1$,
   то есть $\vdash_{\rightarrow 1} Z2$;
2) вычислим $\vdash_{\rightarrow 1} \otimes_{0.7} Z3$:
   $1 \otimes 0.7 = \min\{1,0.7\} = 0.7$,
   то есть $\vdash_{\rightarrow 0.7} Z3$;
3) вычислим $\vdash_{\rightarrow 1} \otimes_{0.95} Z4$:
   $1 \otimes 0.95 = \min\{1,0.95\} = 0.95$,
   то есть $\vdash_{\rightarrow 0.95} Z4$;
4) вычислим $\vdash_{\rightarrow 1} \otimes_{0.95} (Z2 \wedge Z4)$:
   $1 \otimes 0.95 = \min\{1,0.95\} = 0.95$,
   то есть $\vdash_{\rightarrow 0.95} (Z2 \wedge Z4)$;
5) вычислим $\vdash_{\rightarrow 0.95} \otimes_{0.9} (Z5.1 \wedge Z5.2 \wedge Z5.3 \wedge Z5.4)$:

$0.95 \otimes 0.9 = \min\{0.95,0.9\} = 0.9$,
то есть $|\rightarrow_{0.9}$ (Z5.1$\wedge$ Z5.2 $\wedge$ Z5.3$\wedge$ Z5.4);

6) i = 1..4
вычислим $|\rightarrow_{0.9} \otimes_1$ (Z6.i):
$0.9 \otimes 1 = \min\{0.9,1\} = 0.9$,
то есть $|\rightarrow_{0.9}$ (Z6.i);

7) вычислим $|\rightarrow_{0.9} \otimes_{0.9} \otimes_{0.9} \otimes_{0.9}$ (Z6.4$\wedge$ Z6.4$\wedge$ Z6.4$\wedge$ Z6.4):
$0.9 \otimes 0.9 \otimes 0.9 \otimes 0.9 = \min\{0.9,0.9,0.9,0.9\} = 0.9$,
то есть $|\rightarrow_{0.9}$ (Z6.4$\wedge$ Z6.4$\wedge$ Z6.4$\wedge$ Z6.4);

8) вычислим $|\rightarrow_{0.9} \otimes_1$ Z7:
$0.9 \otimes 1 = \min\{0.9,1\} = 0.9$,
то есть $|\rightarrow_{0.9}$ Z7;

9) вычислим $|\rightarrow_{0.9} \otimes_{0.9}$ Z8:
$0.9 \otimes 0.9 = \min\{0.9,0.9\} = 0.9$,
то есть $|\rightarrow_{0.9}$ Z8;

10) вычислим $|\rightarrow_{0.9} \otimes_{0.9}$ (Z7$\wedge$Z8):
$0.9 \otimes 0.9 = \min\{0.9,0.9\} = 0.9$,
то есть $|\rightarrow_{0.9}$ (Z7$\wedge$Z8);

11) вычислим $|\rightarrow_{0.9} \otimes_{0.95}$ Z9:
$0.9 \otimes 0.95 = \min\{0.9,0.95\} = 0.9$,
то есть $|\rightarrow_{0.9}$ Z9;

13) вычислим $|\rightarrow_{0.9} \otimes_1$ Z10:
$0.9 \otimes 1 = \min\{0.9,1\} = 0.9$,
то есть $|\rightarrow_{0.9}$ Z10;

14) вычислим $|\rightarrow_{0.9} \otimes_1$ Z11:
$0.9 \otimes 1 = \min\{0.9,1\} = 0.9$,
то есть $|\rightarrow_{0.9}$ Z11;

Теперь применим матричный метод. Сначала построим матрицу для проверки вывода Z11. Наличие оценок уверенности правил и утверждений требует усовершенствовать процедуру построения матриц. Они, как и для двусмысленной логики, должны иметь столбец и строку для каждого литерала, а элемент на пересечении строки i и столбца j будет иметь не нулевое значение лишь тогда, когда найдется клауза, в безусловной части которой находится литерал строки i, а условная часть содержит литерал столбца j. Но это значение будет нечеткой оценкой. Добавляем клаузу $\leftarrow$ Z11. Учитывая возможность существования нескольких клауз с одинаковой безусловной частью и клауз с условной частью с несколькими литералами (например, сравните $|\rightarrow$ 0.4 A1 $|\rightarrow$ 0.6 (A2 $\rightarrow$ A3) $|\rightarrow$ 0.2 (A1 $\rightarrow$ A3) $|\rightarrow$ 0.7 A2 и $|\rightarrow$ 0.4 A1 $|\rightarrow$ 0.7 A2 $|\rightarrow$ 0.8 (A1,A2 $\rightarrow$ A3)), наличие первого случая будем помечать индексом „$\vee$” возле каждого литерала условной части одной клаузи, а наличие последнего случая - индексом „$\wedge$” возле каждого литерала условной части одной клаузи. Матрица имеет вид:

	◊	Z1	Z2	Z3	Z4	Z5	Z6	Z7	Z8	Z9	Z10	Z11
◊	0	0	0	0	0	0	0	0	0	0	0	1
Z1	1	0	0	0	0	0	0	0	0	0	0	0
Z2	0	1	0	0	0	0	0	0	0	0	0	0
Z3	0	0	0.7	0	0	0	0	0	0	0	0	0
Z4	0	0	0.95	0	0	0	0	0	0	0	0	0
Z5	0	0	0.9$\wedge$	0	0.9$\wedge$	0	0	0	0	0	0	0
Z6	0	0	0	0	0	1	0	0	0	0	0	0
Z7	0	0	0	0	0	0	1	0	0	0	0	0
Z8	0	0	0	0	0	0	0	0.9	0	0	0	0
Z9	0	0	0	0	0	0	0	0.95$\wedge$	0.95$\wedge$	0	0	0
Z10	0	0	0	0	0	0	0	0	0	1$\wedge$	0	0
Z11	0	0	0	0	0	0	0	0	0	0	1	0

Рис. 2. Матрица M1

Применяя правила A и B приведем матрицу к виду матрицы M1. Модифицируем также алгоритм умножения матриц. В случае наличия правил $|\rightarrow$ к A1 $|\rightarrow$ l (A1 $\rightarrow$ A2), умножение $k \otimes l$ необходимо выполнять за формулой min{к, l}. В случае наличия правил $|\rightarrow$ к A1 $|\rightarrow$ l A2 $|\rightarrow$ q (A1, A2 $\rightarrow$ A3), в матрице в строке появляются индексы „$\wedge$” в столбцах для литералов A1, A2 и умножение $(k \otimes l) \otimes q$ необходимо выполнять за формулой min{min{к, q}, min{l, q}}. В случае наличия правил $|\rightarrow$ к A1 $|\rightarrow$ l A2 $|\rightarrow$ q (A1 $\rightarrow$ A3) $|\rightarrow$ r (A2 $\rightarrow$ A3), в матрице в строке появляются два ненулевих элемента с индексами „$\vee$” в столбцах для литералов A1, A2 и при умножении необходимо выбрать максимальный из двух произведений в строке max {min{к, q}, min{l, r}}.

	◊	Z1	Z2	Z4	Z5	Z6	Z7	Z8	Z9	Z10	Z11
◊	0	0	0	0	0	0	0	0	0	0	1
Z1	1	0	0	0	0	0	0	0	0	0	0
Z2	0	1	0	0	0	0	0	0	0	0	0
Z4	0	0	0.95	0	0	0	0	0	0	0	0
Z5	0	0	0.9$\wedge$	0.9$\wedge$	0	0	0	0	0	0	0
Z6	0	0	0	0	1	0	0	0	0	0	0
Z7	0	0	0	0	0	1	0	0	0	0	0
Z8	0	0	0	0	0	0	0.9	0	0	0	0
Z9	0	0	0	0	0	0	0.95$\wedge$	0.95$\wedge$	0	0	0
Z10	0	0	0	0	0	0	0	0	1$\wedge$	0	0
Z11	0	0	0	0	0	0	0	0	0	1	0

Рис. 3. Матрица M1

Перемножение матрицы приводит к подтверждению противоречивости исходного множественного числа клауз при p = 10 по признаку противоречивости для традиционной клаузальной логики, которая доводит приведенное выше предположение о логическом следствии Z11.

	◊	Z1	Z2	Z4	Z5	Z6	Z7	Z8	Z9	Z10	Z11
◊	0.9Λ	0.9Λ	0	0	0	0	0	0	0	0	0.9Λ
Z1	0.9Λ	0.9Λ	0.9Λ	0	0	0	0	0	0	0	0
Z2	0	0.9Λ	0.9Λ	0.9Λ	0	0	0	0	0	0	0
Z4	0	0	0.9Λ	0.9Λ	0.9Λ	0	0	0	0	0	0
Z5	0	0	0.9Λ	0.9Λ	0.9Λ	0.95Λ	0	0	0	0	0
Z6	0	0	0	0	0.9Λ	0.9Λ	0.9Λ	0	0	0	0
Z7	0	0	0	0	0	0.9Λ	0.9Λ	0.9Λ	0	0	0
Z8	0	0	0	0	0	0	0.9Λ	0.9Λ	0.9Λ	0	0
Z9	0	0	0	0	0	0	0.9Λ	0.9Λ	0.9Λ	0.9Λ	0
Z10	0	0	0	0	0	0	0	0	0.9Λ	0.9Λ	0.9Λ
Z11	0.9Λ	0	0	0	0	0	0	0	0	0.9Λ	0.9Λ

Рис. 4. Матрица M10

Теперь остается определиться с нечеткой оценкой результата. Поскольку применением определения операций (3) и правил выведения шага 2 приведенной выше схемы мы получили |→ 0.9 Z11, то нечеткая оценка результата принадлежит диагональному элементу последней матрицы на пересечении строки и столбца Z11.

Проблема использования матричного метода

Существует ряд ограничений и неудобств использования матричного метода. Одну из таких проблем мы рассмотрим. Проблема связана с тем, что при проектировании больших бизнес процессов с использованием матричного метода сама матрица получается очень больших размеров, и пути, получаемые в результате, выходят очень большой длинны. В результате потребуется больше памяти на их хранение, а также, и возможно главное, так как может повлиять на скорость их обработки оператором, это то, что из их внешнего вида очень сложно будет сразу определить сам путь. Представим, что состояния бизнес процесса у нас отмечаются цифрами, а переходы из состояния в состояние стрелкой.

(1 ⟶ 2)(2 ⟶ 3)…(49 ⟶ 64)…(85 ⟶ 73)…

Естественно, имея такой путь, будет очень неудобно его обрабатывать. Поэтому был написан модуль, который позволяет многократно оптимизировать исходный бизнес процесс. Принцип работы модуля состоит в том, чтобы преобразовывать прямые участки в одно целое. То есть если существует участок (1 ⟶ 2)(2 ⟶ 3)(3 ⟶ 4) (4 ⟶ 5) то он заменится на аналогичный только будет иметь вид (1 ⟶ 5) и информация о том из каких частей состоит данный отрезок, будет сохраняться в хранилище данных. Это позволяет сделать схему бизнес процесса более простой и воспринимаемой. Многократная оптимизация означает то, что мы можем применять данный модуль некоторое количество раз на схеме одного бизнес процесса. То есть, оптимизировав исходную схему, опять применим уже к выходной матрице (оптимизированной). Это позволит значительно упростить отображение нашего бизнес процесса. Так же всегда есть возможность получить полный путь по сокращенным. Далее будет рассмотрено применение модуля на нашем бизнес процессе «Изготовление продукции».

Использование модуля оптимизации при расчетах

Принцип работы модуля состоит в том, что прямые участки отработки бизнес процесса представляются как одно целое. То есть если есть участок, который представляет собой определенную последовательность действий, без ветвлений, то можно представить данный участок как начальный и конечный пункт, скрывая промежуточные состояния бизнес процесса. Промежуточные состояния будут сохраняться в хранилище, поэтому всегда будет возможность узнать эти промежуточные состояния и возобновить полную схему бизнес процесса. Выше был произведен расчет бизнес процесса «Изготовление продукции» без применения данного модуля. Рассмотрим его применение на данном бизнес процессе и сравним полученные результаты.

Исходная матрица, отображающая наш бизнес процесс, имеет следующий вид (рисунок 5).

	◊	Z1	Z2	Z4	Z5	Z6	Z7	Z8	Z9	Z10	Z11
◊	0	0	0	0	0	0	0	0	0	0	1
Z1	1	0	0	0	0	0	0	0	0	0	0
Z2	0	1	0	0	0	0	0	0	0	0	0
Z4	0	0	0.95	0	0	0	0	0	0	0	0
Z5	0	0	0.9Λ	0.9Λ	0	0	0	0	0	0	0
Z6	0	0	0	0	1	0	0	0	0	0	0
Z7	0	0	0	0	0	1	0	0	0	0	0
Z8	0	0	0	0	0	0	0.9	0	0	0	0
Z9	0	0	0	0	0	0	0.95Λ	0.95Λ	0	0	0
Z10	0	0	0	0	0	0	0	0	1Λ	0	0
Z11	0	0	0	0	0	0	0	0	0	1	0

Рис. 5. Исходная матрица, отображающая бизнес процесс

Применив модуль оптимизации один раз на исходной схеме, мы получим следующий вид матрицы (рисунок 6).

	◊	Z2	Z7	Z11
◊	0	0	0	1
Z2	1	0	0	0
Z7	0	0.9Λ	0	0
Z11	0	0	0.9Λ	0

Рис. 6. Оптимизированная матрица

После преобразования видно, что остались те узлы, в которых происходит ветвление, а прямые участки представляются единым целым. Ниже показано, каким образом произошло преобразование.

(◊ - Z2) : Path (◊ - Z1)(Z1 – Z2)

(Z2 – Z7) : Path (Z2 – Z4)(Z4 – Z5)(Z5 – Z6)(Z6 – Z7)

(Z2 – Z7) : Path (Z2 – Z5)(Z5 – Z6)(Z6 – Z7)

(Z7 – Z11) : Path (Z7 – Z8)(Z8 – Z9)(Z9 – Z11)(Z10 – Z11)

(Z7 – Z11) : Path (Z7– Z9)(Z9 – Z10)(Z10 – Z11)

(Z11 - ◊) : Path (Z11 - ◊)

Располагая такими данными, в любой момент, возможно, будет отобразить исходный вид матрицы. Как видно, результат оптимизации значительно упростил существующую матрицу. Сохраняя все исходные данные мы получаем матрицу, которая значительно отличается по размерам от начальной. Во – первых, это позволит сэкономить память, необходимая для хранения данных, так как видно, что размеры матриц отличаются практически в три раза. Во – вторых, при отображении пути движения документа, схема становится более читаемой, понятной и прозрачной. И трете, самое главное, полученные данные позволят значительно ускорить дальнейший расчет для анализа бизнес процесса. Так как расчет связан с перемножением матриц, а при их больших размерах, данная задача становиться ресурсоемкой, требуется большее количество памяти на расчеты, и значительно большее количество времени на обработку информации.

Теперь применим наши полученные результаты после оптимизации к нашим расчетам. Собственно проделаем ту же процедуру что и в основном примере только с оптимизированной матрицей. Матрица для проверки вывода Z11 это и будет наша матрица на рисунке 5.

	◊	Z2	Z7	Z11
◊	0	0	0	1
Z2	1	0	0	0
Z7	0	0.9Λ	0	0
Z11	0	0	0.9Λ	0

Рис. 7. Матрица 1

	◊	Z2	Z7	Z11
◊	0	0	0.9Λ	0
Z2	0	0	0	1
Z7	0.9Λ	0	0	0
Z11	0	0.9Λ	0	0

Рис. 8. Матрица 2

	◊	Z2	Z7	Z11
◊	0	0.9Λ	0	0
Z2	0	0	0.9Λ	0
Z7	0	0	0	0.9Λ
Z11	0.9Λ	0	0	0

Рис. 9. Матрица 3

	◊	Z2	Z7	Z11
◊	0.9Λ	0	0	0
Z2	0	0.9Λ	0	0
Z7	0	0	0.9Λ	0
Z11	0	0	0	0.9Λ

Рис. 10. Матрица 4

Перемножение матрицы приводит к подтверждению противоречивости исходного множественного числа клауз при p = 4 по признаку противоречивости для традиционной клаузальной логики, которая доводит приведенное выше предположение о логическом следствии Z11.

Теперь остается определиться с нечеткой оценкой результата. Поскольку применением определения операций (3) и правил выведения шага 2 приведенной выше схемы мы получили |→ 0.9 Z11, то нечеткая оценка результата принадлежит диагональному элементу последней матрицы на пересечении строки и столбца Z11.

Как видно, по сравнению с результатами, полученными при обычных расчетах, и количество информации и количество шагов расчетов значительно отличаются. При обычных расчетах потребовалось провести 10 шагов с матрицей размером 11 на 11, а во втором случае 4 шага с матрицей размером 4 на 4. Так как наш пример не является полным отображением ре-

ального бизнес процесса и полученные результаты являются довольно удовлетворительными, то можно представить какую пользу может принести данный модуль при расчетах реальных бизнес процессов, где размеры матриц смогут составлять 50 на 50, а то и гораздо больших размеров.

Реализация подхода

Программная реализация механизмов оптимизации и расчетов разработана с помощью среды разработки Microsoft Visual Studio 2010 + .Net Framework 4.0 и языка программирования C#. Отображение было разработано с помощью технологии WPF(Windows Presentation Foundation) и языка разметки xaml. При написании данного модуля использовался шаблон проектирования MVP (MODEL – VIEW - PRESENTER).[4][5] Суть паттерна в том, чтобы форма (View) реализовывала интерфейс, с которым сможет работать представление данных (Presenter), которое в свою очередь оперирует объектами предметной области (Model). Таким образом, логика приложения отделяется от представления данных. Также, появляется возможность протестировать эту логику.

Presenter выступает в роли связующего модели и представления. По сути это самое сложное место так как View и Model это две отдельные сущности а цель Presenter связать их. Данный подход позволяет создавать абстракцию представления. Реализовать данный паттерн можно при помощи вынесения интерфейсов представления. У каждого представления будут интерфейсы с определенными наборами методов и свойств, необходимых презентеру, презентер в свою очередь инициализируется с данным интерфейсом, подписывается на события представления и по необходимости подсовывает данные.

В нашем модуле вся логика для оптимизации и расчетов была вынесена в отдельный компоновочный блок (библиотеку), который и есть Model в данном случае. View был создан ради демонстрации данного модуля. В этом и есть основное преимущество применения паттерна MVP, так как можно легко и четко разграничить логику и отображение и в случае необходимости подменять эти блоки. То есть для интегрирования данного модуля достаточно подменить или отключить View, если модуль необходимо использовать для промежуточных расчетов, не требующих отображения.

Библиотека называется CalculationLibrary.dll. Данная библиотека предоставляет два интерфейса для работы с ней:

- IMultiplyMatr
- IOptimize

По названию данных интерфейсов легко можно понять их предназначение, интерфейс IMultiplyMatr предназначен для произведения расчетов над матрицами, а интерфейс IOptimize предназначен оптимизаций матриц. Классы, реализующие данные интерфейсы соответственно:

- MultiplyMatr
- Optimize

Данные классы предоставляют все необходимые методы и свойства для реализации данного механизма.

Интерфейс IOptimize предоставляет 4 основные методы и одно свойство для работы:

- FillDefaulutGrid - метод
- OptimizeTable - метод
- GenerateFile - метод
- GetFileData – метод
- Iteration - свойство

Основным является, конечно, метод OptimizeTable. Метод предназначен для оптимизации матрицы. В качестве аргумента принимает объект типа DataTable, отображающий исходную матрицу, которую необходимо оптимизировать, и в качестве возвращаемого значения выступает тоже объект DataTable, содержащий уже новую оптимизированную матрицу.

Остальные методы написаны специально для демонстрации модуля и вряд ли будет необходимость их использования при внедрении, в какие либо системы.

Метод FillDefaulutGrid предназначен для заполнения матрицы значениями по умолчанию, то есть нулями. В качестве аргумента принимает целое число и в результате выполнения метода будет возвращен объект типа DataTable, содержащий матрицу заполненную нулями и размером, который был указан в качестве входного аргумента в метод.

Метод GenerateFile предназначен для генерации файла с данными. Для автоматизации ввода данных в матрицу. При выполнении данного метода на жестком диске в каталоге с исполняемым файлом программы создается файл

Settings.xml, содержащий множество объектов Point, каждый из которых содержит три свойства:

- Col – колонка;
- Row – рядок;
- Probability – вероятность.

При последующей загрузки файла xml каждый объект Point займет свое место в матрице по данным свойствам колонки и рядка, и иметь значение вероятности, которое отобразиться пользователю.

Метод имеет один входной целочисленный аргумент, который означает, сколько объектов Point будет содержать данный файл. В качестве возвращаемого значения методом выступает переменная типа bool (булевый тип), принимающая значение true если удалось создать файл и false если нет.

Метод GetFileData предназначен для считывания матрицы из файла. Не содержит входных аргументов. В качестве возвращаемого значения выступает объект типа DataTable, содержащий матрицу заполненную значениями из файла.

Последним членом данного интерфейса является свойство Iteration. Данное свойство целочисленного типа. На каждой итерации происходят записи в базу данных и при этом в каждой записи есть поле итерация, что бы впоследствии можно было возобновить данные на каждой итерации. Это и есть основное предназначение данного свойства, инкрементировать его на каждой операции оптимизации и записывать в базу с данными.

Интерфейс IMultiplyMatr предоставляет всего лишь два метода:

- Multiply
- ShowExcel

Первый метод Multiply является основным методом данного класса, и предназначен для перемножения двух матриц. Метод имеет два входных аргумента, в качестве которых выступают два экземпляра класса DataTable, содержащие исходные матрицы, необходимые для перемножения. Возвращаемым значением метода является также экземпляр класса DataTable, содержащий уже готовую перемноженную матрицу.

Второй метод ShowExcel предназначен для отображения данных в Excel. Для этого используется COM библиотека Microsoft.Office.Interop.Excel, которая предоставляет необходимый набор методов и свойств для работы с пакетом Microsoft.Excel. Метод предполагается для отображения матрицы полученной в результате перемножения двух других. В качестве входного аргумента является экземпляра класса DataTable, который и содержит необходимую для отображения матрицу. Отображаемый пользователю файл Excel, сгенерированный в результате выполнения метода, содержит не только матрицу, полученную в качестве аргумента в метод и отображающую вероятности перехода из одного состояния бизнес процесса в другой, но и матрицу отображающую пути перехода. Пример представления данных на рисунке 11.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
1	0	1	1	0	1	1	0		0	(1->6)(6->0)	(2->6)(6->0)	0	(4->6)(6->0)	(5->6)(6->0)	0
2	0	0	0	0	0	0	1		0	0	0	0	0	0	(6->0)(0->1)
3	1	0	0	0	0	0	0		(0->1)(1->2)	0	0	0	0	0	0
4	1	1	0	0	0	0	0		(0->1)(1->3)	(1->2)(2->3)	0	0	0	0	0
5	1	1	1	0	0	0	0		(0->1)(1->4)	(1->3)(3->4)	(2->3)(3->4)	0	0	0	0
6	1	1	1	1	0	0	0		(0->1)(1->5)	(1->4)(4->5)	(2->3)(3->5)	(3->4)(4->5)	0	0	0
7	1	1	0	1	1	0	0		(0->1)(1->6)	(1->5)(5->6)	0	(3->5)(5->6)	(4->5)(5->6)	0	0

Рис. 11. Пример отображения данных в Excel

Существует еще ряд классов, как видно из диаграммы, необходимых для работы модуля. Класс OptimNode является основным классом, над объектами которого происходят различные вычисления, и является отображением ячейки таблицы. Состоит из четырех свойств и одного метода. Свойства:

- Num – свойство предназначено для отображает текущего состояния бизнес процесса.
- NodeToGo – предназначено для хранения значения связанного, дочернего состояния с текущим.
- Path – так как при оптимизации прямые участки графа преображаются в одно целое, а промежуточные состояния скрываются, то весь путь, включая скрытые состояния, между двумя точками хранится в данном свойстве.
- Probability – в данном свойстве хранится вероятность перехода между состояниями. Поле типа double. Может принимать значение от 0 до 1.

Классы Settings, SettingsBuilder, Coordinates – принимают участие при формировании файлов со сгенерированными значениями, отображающие бизнес процесс. При создании файла и при считывании данных из него используется технология сериализации. [6]

Сериализация (в программировании) – процесс перевода какой-либо структуры данных в последовательность битов. Обратной к опера-

ции сериализации является операция десериализации – восстановление начального состояния структуры данных из битовой последовательности.

Сериализация используется для передачи объектов по сети и для сохранения их в файлы. Например, нужно создать распределённое приложение, разные части которого должны обмениваться данными со сложной структурой. В таком случае для типов данных, которые предполагается передавать, пишется код, который осуществляет сериализацию и десериализацию. Объект заполняется нужными данными, затем вызывается код сериализации, в результате получается, например, XML-документ. Результат сериализации передаётся принимающей стороне, например, по электронной почте или HTTP. Приложение-получатель создаёт объект того же типа и вызывает код десериализации, в результате получая объект с теми же данными, что были в объекте приложения-отправителя. По такой схеме работает, например, сериализация объектов через SOAP в Microsoft .NET.

Сериализация предоставляет несколько полезных возможностей:

- Метод реализации сохраняемости объектов, который более удобен, чем запись их свойств в текстовый файл на диск и повторная сборка объектов чтением файлов;
- метод осуществления удалённых вызовов процедур, как, например, в SOAP;
- метод распространения объектов, особенно в технологиях компонентно-ориентированного программирования, таких как COM и CORBA;
- метод обнаружения изменений в данных, изменяющихся со временем.

В нашем случае используется xml сериализация. Класс Settings помечается атрибутом XmlRoot – получает или задает объект, задающий сериализацию с помощью XmlSerializer для класса как корневого элемента XML. Единственное свойство класса Settings – ColumnList помечается атрибутом XmlElement, это означает, что значения данного свойства будут формировать записи в файле XML. ColumnList является коллекцией экземпляров класса Coordinates. В сою очередь свойства класса Coordinates: Col, Row, Probility – помечаются атрибутами XmlAttribute. Это означает, что именно эти три свойства будут отображаться в каждой записи XML файла. Формат записи на рисунке 12.

```
<Point Col="0" Row="0" Probability="0" />
<Point Col="0" Row="1" Probability="1" />
```

Рис. 12. Формат хранения данных в xml

Оставшийся класс SettingsBuilder и занимается сериализацией, то есть формированием и занесением данных в xml файл, десериализацией, то есть вычиткой данных их файла. Для этого в классе предоставляется два метода:

- Serialize – данный метод имеет один входной аргумент, в качестве которого выступает экземпляр класса Settings, который и будет сериализован в файл. Данный метод не имеет возвращаемого значения.
- Deserialize – метод предназначен для считывания данных из XML файла. Не имеет входных аргументов. В качестве возвращаемого значения метода выступает экземпляр класса Settings.

На диаграмме классов остался еще один класс о котом следует рассказать и им является NodeCache. Данный класс является кэшем для хранения состояния бизнес процесса. Для того чтобы не обращаться каждый раз к базе данных при расчетах и создан такой класс буфер. Данные запишутся в кэш при первом обращении, и при последующей необходимости будут использовать данный класс. При проектировании данного класса – кэша использовался шаблон проектирования SingleTon.[7]

Основная его задача – гарантировать, что класс будет иметь лишь единственный экземпляр и глобальную точку доступа к нему. В программировании часто встречаются ситуации, когда необходимо создать класс, который будет существовать только таким образом. Глобальные переменные здесь не годятся, поскольку дают доступ к классу, но не могут запретить его инстанцирование в нескольких экземплярах. В случае работы с шаблоном Singleton сам класс контролирует наличие единственного своего экземпляра и предоставляет доступ к нему.

Итак, Singleton используют в следующих случаях:

- когда необходимо иметь лишь один экземпляр какого-либо конкретного класса, доступного любым клиентам;
- когда единственный экземпляр данного класса способен расширяться через порождения подклассов, а клиенты имеют возможность работать с уже расширенным экземпляром, не внося никаких изменений в свой код.

У класса, реализующего шаблон Singleton, присутствует операция Instance, требующаяся

для того, чтобы предоставить доступ к его единственному экземпляру. Надо понимать, что клиенты получают доступ к экземпляру данного класса только через операцию Instance и никак иначе.

Рассматриваемый паттерн обладает несомненными достоинствами. Выше уже упоминалось, что класс, реализующий шаблон Singleton, инкапсулирует свой единственный экземпляр. Именно благодаря этому он полностью контролирует то, каким образом и когда клиенты будут получать доступ к нему. Кроме того, Singleton позволяет избежать использования глобальных переменных, а значит, не придется засорять ими пространство имен для хранения уникальных экземпляров классов.

Довольно легко создавать и подклассы от класса, реализующего Singleton. Само приложение допустимо сконфигурировать уже расширенным классом. Естественно, можно будет конкретизировать приложение определенным экземпляром класса, нужным в данный момент для выполнения.

Также несложно изменить класс, реализующий Singleton, и предоставить ему возможность создавать более одного экземпляра класса. Это порождает некоторую гибкость, которая всегда может пригодиться в будущем.

К минусам данного шаблона проектирования стоит отнести лишь то, что глобальные объекты, как таковые, могут быть вредны для программы, поскольку препятствуют ее масштабируемости.

Анализ производительности

Производительность данного модуля так же является немало важной частью роботы, так как в случае длительного времени потраченного на расчеты, использование системы становится просто не актуальным. Ни один оператор, ни одна система не захочет использовать данный модуль в случае, если на оптимизацию и анализ требуются 10 минут, а то и десятки.

Проанализируем производительность модуля в зависимости от объема данных, над которыми производятся операции.

1) Анализ оптимизации.

Табл. 1. Данные производительности при оптимизации

Кол-во состояний бизнес процесса.	10	50	100	150
Время потраченное на оптимизацию	36 Млсек.	661 Млсек.	20 Сек	2 мин 22сек

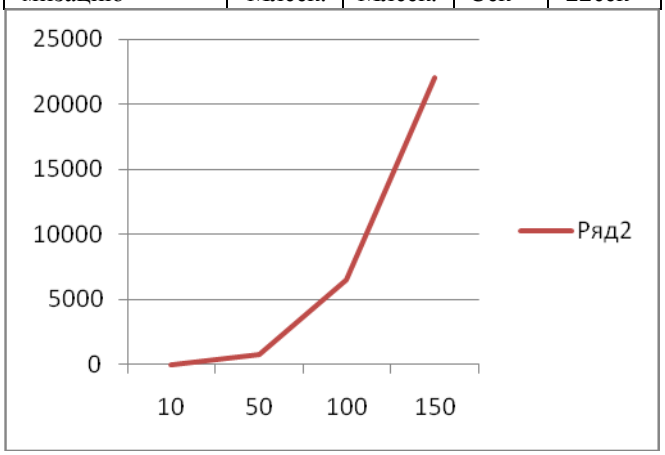


Рис. 13. График производительности по оптимизации

На рисунке 13 отображено график производительности системы при оптимизации бизнес процесса. По горизонтальной оси показано количество состояний, по вертикальной оси показано количество миллисекунд необходимые для оптимизации матрицы. Как видно зависимость имеет экспоненциальный вид. И говорит о том, что при изменении количества состояний бизнес процесса – количество времени необходимое для оптимизации увеличивается в разы.

2) Анализ расчетов.

Табл.1. Данные производительности при расчетах

Кол-во состояний бизнес процесса.	10	50	1	150
Время потраченное на расчеты	24 Млсек.	817 Млсек.	6.5 Сек	22 Сек

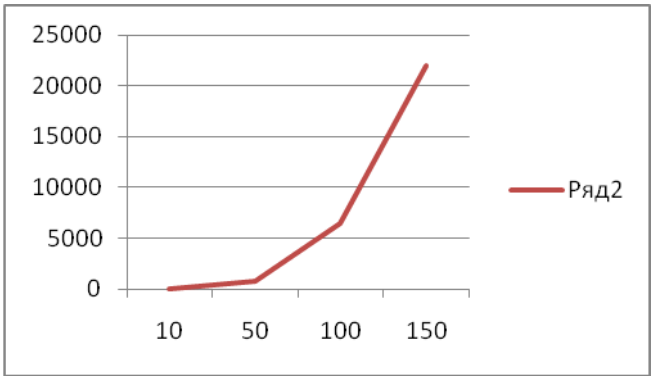


Рис. 14. График производительности по расчетам

На рисунке 14 отображено график производительности системы при расчетах, анализе бизнес процесса. По горизонтальной оси показано количество состояний, по вертикальной оси показано количество миллисекунд необходимые для расчетов матрицы. Как видно с те-

кущего графика зависимость так же имеет экспоненциальный вид, но рост графика не такой значительный как при оптимизации.

Данные исследования полностью не отображают реальные данные по производительности, так как для расчетов и оптимизации использовались бизнес процессы сгенерированные приложением, а не настоящие, имеющие практическое применение. Бизнес процесс, полученный с помощью приложения, представляет собой схему, в которой практически каждое состояние имеет связь с остальными. Из-за этого прямых участков в схеме остается малое количество, и оптимизация не приносит большой выгоды. В такой ситуации, когда у графа почти все состояния связаны друг с другом, сама суть использования бизнес процессов теряется. В реальной жизни такие процессы имеют более простой вид, более прямолинейные зависимости, что делает применение модуля очень полезным, и это было продемонстрировано на примере бизнес процесса «Изготовление продукции».

При использовании данного модуля на бизнес процессе «Изготовление продукции» во время оптимизации было достигнуто значение практически в 65%, так как исходная схема содержала 11 состояний, а оптимизированная – 4. Вследствие чего для расчетов матрицы соответствующего размера необходимо было перемножить 10 раз для исходной матрицы, и 3 раза для оптимизированной.

Время потраченное на оптимизацию составляет 45 миллисекунды. На расчет одной итерации 17 миллисекунды. Учитывая то, что необходимо произвести 3 итерации общее время составляет:

T = 3* 17 + 45 = 96 миллисекунды

Время потраченное на одну итерацию расчетов исходной матрицы составляет 27 миллисекунды. Общее время на расчеты составляет:

T = 27 * 10 = 270 миллисекунд

Учитывая полученные данные, можно получить график производительности для данного реального бизнес процесса (рисунок 15).

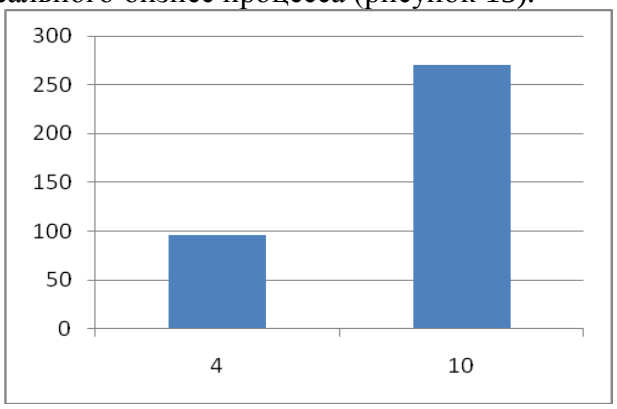


Рис. 15. График производительности бизнес процесса «Изготовление продукции»

Из графика видно разницу в производительности при использовании модуля оптимизации и без него. Во-первых, выигрыш во времени очевиден. Во – вторых, данные полученные в результате расчетов представляются в удобном виде для пользователя.

Выводы

Настоящий материал посвящен проблеме создания системы построения бизнес-процессов посредством реализации схем (планов) действий на основании системы бизнес-правил для автоматизированных систем управления любого уровня технической и программной насыщенности с использованием матричного метода.

Главными преимуществами внедрения модуля является возможность анализа составленных бизнес процессов, получения вероятности отработки конкретных БП конкретными путями, а также решена проблема отображения путей бизнес процессов.

Список литературы

1 Ершов Ю.Л., Палютин Е.А. Математическая логика: Учебное пособие для вузов. Изд.2. 1987. – 336 с.
2 Математическая логика и теория алгоритмов: Учебное пособие. - 2-е изд.120-125стр.
3 Теленик С.Ф., Амонс А.А, Смічик Р.В., Хмелюк В.С. Матрична резолюція для клаузальних логік // Вестник ХНАДУ. Сб. науч. тр. Вып.28, 2003. – С.243-246.
4 http://www.martinfowler.com/eaaDev/PresentationModel.html by Martin Fowler
5 http://msdn.microsoft.com/msdnmag/issues/06/08/DesignPatterns/default.aspx by Jean-Paul Boodhoo
6 Эндрю Троелсен «Язык программирования C# 2005 и платформа .Net 2.0». Глава 17.
7 Дмитрий Федоров “Примеры использования Паттерн Singleton (Одиночка)”

УДК 519.854.2

ВАЩУК Ф.Г.,
ПАВЛОВ О. А.,
МІСЮРА О. Б.,
МЕЛЬНИК О.О.

СКЛАДАННЯ РОЗКЛАДІВ СУМАРНОГО ВИПЕРЕДЖЕННЯ І ЗАПІЗНЕННЯ ІЗ НАЛАГОДЖЕННЯМИ, ЩО ЗАЛЕЖАТЬ ВІД ПОСЛІДОВНОСТІ

Розглядається задача складання розкладів за критерієм мінімізації сумарного випередження і запізнення відносно директивних строків при виконанні незалежних завдань одним приладом (МВЗ) при наявності налагоджень. Запропоновано евристичний алгоритм пошуку локального оптимального розв'язку задачи, що розглядається. Експериментальні дослідження показали ефективність розробленого алгоритму, що дозволяє розв'язувати задачи великої розмірності за короткий час.

We consider the one machine scheduling problem of minimizing the total earliness and tardiness of independent tasks against due dates (ET) with setups. We propose a heuristic algorithm to search a local optimal solution of the problem. Experimental studies have shown the effectiveness of the algorithm, which allows to solve large dimensional problems for a short time.

Вступ

Складанню розкладів відносно директивних строків приділяється значна увага в літературі. Одна з причин цього явища – зростаючий тиск конкуренції на міжнародних ринках: фірми повинні запропонувати велику розмаїтість різних та індивідуальних виробів, у той час як клієнти очікують, що замовлені товари будуть поставлені вчасно. Принцип виробництва «точно в строк» установлює, що необхідна кількість товарів повинна бути вироблена або поставлена точно в заданий час. Виконання роботи з випередженням приводить до витрат на складування, у той час як запізнення робіт – до штрафів і, в остаточному підсумку, втраті доброзичливості клієнтів і репутації фірми.

Постановка задачі. Множина з n незалежних завдань $J = \{j_1, j_2, ..., j_n\}$ повинна бути призначена на виконання без переривань на одному приладі, який може працювати не більш, ніж з одним завданням одночасно. Прилад і завдання передбачаються безупинно доступними з моменту часу нуль, а простої приладу не допускаються. Завдання j, де $j = 1, 2, ..., n$, вимагає часу виконання p_j і в ідеалі повинне бути закінчене у свій директивний строк d_j. Для окремих завдань задано час налагодження s_{ij}, це означає, що в розкладі, у якому завдання j виконується відразу після завдання i, повинен бути час налагодження s_{ij} одиниць часу між моментом завершення завдання i, позначеним через C_i, і часом початку завдання j, що дорівнює $C_j - p_j$. Протягом цього періоду налагодження ніяке інше завдання не може бути виконано приладом. Часи налагодження є *залежними від послідовності*, тому що вони залежать як від i, так і від j. Для будь-якого заданого розкладу випередження та запізнення завдання j можуть бути визначені виразами (2) і (3). Ціль полягає в тім, щоб знайти розклад, який мінімізує сумарне випередження і запізнення всіх завдань:

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

де випередження і запізнення визначаються як, відповідно,

$$E_j = \max(0, d_j - C_j) = (d_j - C_j)^+,$$
$$T_j = \max(0, C_j - d_j) = (C_j - d_j)^+.$$

Ця задача (позначимо її МВЗН) узагальнює задачу МВЗ і відноситься до класу *NP*-повних. В [1] розглядається актуальність задачі та приводиться огляд літератури по складанню розкладів «точно в строк» і розкладів з налагодженнями. Представлено алгоритм віток і границь і ефективна евристична процедура одержання локального оптимуму.

Алгоритми локального пошуку широко використовуються як практичний підхід для розв'язання задач комбінаторної оптимізації. Починаючи із припустимого розв'язку, ці алгоритми ітеративно пробують поліпшити поточний розв'язок, вишукуючи кращий розв'язок в околиці поточного розв'язку, поки не знайдений локальний оптимум. Ефективність цих алгоритмів якісно залежить від визначення околиці: при більших околицях якість розв'язку краще, але час обчислень більше. Тому на прак-

тиці більші околиці не корисні, якщо тільки вони не можуть ефективно досліджуватися.

В основу розв'язання задачі покладено новий підхід до розв'язання задачі МВЗ, викладений у [2, 3]. У даній статті ми розширюємо цей підхід для більш загальної та більш практичної задачі, що з'являється особливо у виробничих задачах складання розкладів – задачі із часами налагодження.

Алгоритм розв'язання задачі мвзн

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

Алгоритм, що представляється, складається із двох етапів. На першому етапі (блоки 1 і 2) налагодження не враховуються. У блоці 1 розв'язується задача мінімізації сумарного запізнення при виконанні незалежних завдань одним приладом МСЗ [3]. Алгоритм побудовано на перестановках і полягає в оптимальному використанні завданнями, що запізнюються, резервів завдань, що не запізнюються. Таким чином, реалізується зменшення сумарного запізнення за рахунок зменшення сумарного випередження. У блоці 2 здійснюється зменшення значення сумарного випередження і запізнення за допомогою послідовного збільшення моментів початку виконання завдань.

Позначимо $r_{\min} = \min\{r_j\}$; N_r – число завдань з резервами ($r_j > 0$); $N_з$ – число завдань, що запізнюються (в їх число включаються завдання з нульовим резервом).

Твердження [4] (використання резервів завдань, що випереджають). Якщо в послідовності σ виконується $N_r > N_з$, то при збільшенні початку виконання завдань на величину, рівну $r_{\min}$, значення функціонала випередження/запізнення зменшується на величину $(N_r - N_з)\, r_{\min}$.

Розглядається послідовність, що отримана в результаті розв'язання задачі МСЗ. На кожній ітерації за умови $N_r \geq N_з$ збільшуються моменти початку виконання завдань у поточній послідовності на $r_{\min}$. Такі процедури виконуються, поки не виконається умова $N_r < N_з$. Отриману послідовність позначаємо σ^R.

На другому етапі здійснюється оптимізація послідовності σ^R з урахуванням налагоджень приладу, що залежать від послідовності, за допомогою реалізації процедури локального пошуку ефективного розв'язання поставленої задачі. Аналізується послідовність σ^R. Для тих завдань, для яких задані налагодження приладу, включаємо ці налагодження у тривалості виконання завдань. Отриману послідовність позначаємо σ^{R1}.

Використовуються наступні типи перестановок: API (adjacent pairwise interchange – суміжна попарна перестановка), NAPI (nonadjacent pairwise interchange – несуміжна попарна перестановка), EBSR (extraction and backward-shifted reinsertion – витяг і повторна вставка зі зрушенням назад) и EFSR (extraction and forward-shifted reinsertion – витяг і повторна вставка зі зрушенням уперед), які діють на послідовність σ таким чином. Нехай задана пара індексів $i < j$, така, що $\sigma = \rho\, i\, \pi\, j\, \omega$.

API(i, j): $\rho\, i\, j\, \omega \Rightarrow \rho\, j\, i\, \omega$ (потрібне $\pi = \varnothing$);

NAPI(i, j): $\rho\, i\, \pi\, j\, \omega \Rightarrow \rho\, j\, \pi\, i\, \omega$;

EBSR(i, j): $\rho\, i\, \pi\, j\, \omega \Rightarrow \rho\, j\, i\, \pi\, \omega$;

EFSR(i, j): $\rho\, i\, \pi\, j\, \omega \Rightarrow \rho\, \pi\, j\, i\, \omega$.

Опис етапу 2 алгоритму

1. У послідовності σ^{R1} знаходимо завдання j, виконання якого вимагає максимального часу налагодження приладу: s_{kj}, де завдання k займає позицію, що безпосередньо передує завданню j.
2. Знаходимо в σ^{R1} завдання i, для яких $s_{ij} < s_{kj}$. Організуємо список S цих завдань, упорядкований за неспаданням значень часу налагодження.
3. Будуємо нові поточні послідовності завдань таким чином. Завдання j з позиції k+1 за допомогою виконання відповідного оператора переносимо на позицію i+1, на якій налагодження для завдання j мінімальне. У зв'язку зі зміною послідовності змінюємо значення налагоджень: налагодження завдання j на новій позиції і налагодження завдання, що безпосередньо слідує за новою позицією завдання j. Перераховуємо часи виконання завдань і визначаємо значення функціонала. Аналогічно будуємо наступну поточну послідовність, переставляючи в послідовності σ^{R1} завдання j після наступного завдання зі списку S і виконуючи всі процедури, описані в п. 3. Така побудова поточної послідовності виконується, поки не буде переглянутий весь список можливих позицій перестановки завдання j.
4. Із всіх побудованих поточних послідовностей вибираємо послідовність із найменшим значенням функціонала, у цій послідовності завдання j позначається зірочкою з метою виключення його з подальшого розгляду, позначимо цю послідовність σ^{R2}, перехід на

п. 1, розглядаючи в якості σ^{R1} послідовність σ^{R2}. У послідовності σ^{R2} знаходимо непомічене зірочкою завдання *j*, виконання якого вимагає максимального часу налагодження приладу, і виконуємо кроки 1–3. Якщо в черговій поточній послідовності відсутні непомічені завдання, що вимагають налагодження, то алгоритм закінчує роботу.

Обчислювальні результати

Алгоритм був закодований мовою C# у середовищі розробки Visual Studio 2010 під бібліотеку Microsoft .NET 4.0. Випробування проводилися на персональному комп'ютері із процесором Pentium CORE 2 Duo 2.0 ГГц із оперативною пам'яттю 2 Гбайта під управлінням ОС Microsoft Windows Vista. Досліджувалися задачі розмірності до 500 завдань.

Для визначення ефективності алгоритму були проведені дослідження залежності часу розв'язання задачі від загального числа завдань і кількості завдань, що вимагають налагодження приладу, заданого у відсотковому відношенні до загального числа завдань.

Схема генерації даних, запропонована Фішером [5], використовувалася для тестування алгоритму на різних типах прикладів, тип задачі визначається комбінацією фактора запізнення *T* і діапазону директивних строків *R*. Для кожної задачі спочатку генеруються тривалості виконання і часи налагоджень із рівномірного розподілу із заданими границями. Потім обчислюються директивні строки з розподілу, рівномірного на $\left[p^*(1-T-R/2), p^*(1-T+R/2)\right]$, де p^* – сума всіх тривалостей. Значення *T* і *R* вибираються з множин {0.2, 0.4, 0.6, 0.8} і {0.2, 0.4, 0.6, 0.8, l.0}, відповідно, даючи по 20 задач кожного типу.

Результати, що наведені в табл. 1, дані для випадку $T = 0{,}4$; $R = 0{,}8$. Найбільш складні задачі виникають в області $T = 0{,}6$; $R = 0{,}9$.

Табл. 1. Середній час розв'язання задач (мс)

Розмірність	Кількість завдань, що потрібують налагодження, %									
	10	20	30	40	50	60	70	80	90	100
50	32	32	33	33	33	34	34	34	35	35
100	103	104	105	106	107	109	110	111	113	114
150	181	183	184	186	188	189	191	193	195	197
200	233	235	236	237	239	241	242	244	246	248
250	268	270	273	276	279	282	286	289	293	297
300	326	330	335	340	346	352	358	365	371	378
350	338	349	361	374	387	401	416	431	448	464
400	444	453	464	475	487	499	512	526	540	555
450	476	491	508	525	544	562	583	604	626	649
500	542	560	580	600	622	644	669	693	720	746

Висновки

Ми представили алгоритм локального пошуку розв'язку задачі МВЗН, заснований на методі розв'язання задачі МВЗ, викладеному в [2, 3]. У нашій евристиці на першому етапі будується послідовність, у якій всі завдання виконуються максимально близько до директивних строків без урахування налагоджень. На етапі 2 задача розв'язується за допомогою операторів перестановок. Дослідження показали, що наш алгоритм дозволяє за прийнятний час ефективно розв'язувати задачі великої розмірності. Майбутні дослідження будуть спрямовані на вивчення поводження алгоритму при збільшенні числа завдань і різних значеннях *R* і *T* та на побудову відсікань, що скорочують час розв'язання.

Список літератури

1. F. Sourd. Earliness-tardiness scheduling with setup considerations / Computers & Operations Research № 32 (7). – 2005, – P.1849-1865.
2. Павлов О.А., Місюра О.Б., Мельников О.В. Дослідження властивостей та розв'язання задачі «Мінімізація сумарного штрафу як за випередження, так і за запізнення відносно директивних строків при виконанні незалежних завдань одним приладом» / Вісник НТУУ "КПІ". Інформатика, управління та обчислювальна техніка. К.: "ВЕК+", 2008.– №48.– С.3-6.
3. Згуровский М.З., Павлов А.А. Принятие решений в сетевых системах с ограниченными ресурсами: Монография.– К.: Наукова думка, – 2010.– 573 с.
4. Згуровский М.З., Павлов А.А., Мисюра Е.Б. ПДС-алгоритмы и труднорешаемые задачи комбинаторной оптимизации // Системні дослідження та інформаційні технології.– 2009.– №.4. – С.14-31.
5. Fisher M.L. A dual algorithm for the one-machine scheduling problem / Math. Programming 11. – 1976, – P.229-251.

УДК 004.7(045)

ИГНАТОВ В.А.,
ГУЗИЙ Н.Н.,
СОРАЯ М.А.

ОПТИМАЛЬНОЕ ОБЕСПЕЧЕНИЕ ГАРАНТОСПОСОБНОСТИ ТЕЛЕКОММУНИКАЦИОННЫХ И КОМПЬЮТЕРНЫХ СЕТЕЙ

Предложены модели и алгоритмы расчета оптимального обеспечения гарантоспособности специализированных телекоммуникационных и компьютерных сетей на основе *ZG(n,m)*- преобразования для оптимального управления избыточностью телекоммуникационных и компьютерных сетей, обеспечения заданных требований гарантоспособности при минимальных затратах сетевых ресурсов.

Models and algorithms of calculation of the optimum providing of telecommunication and computer networks dependability are offered. *ZG(n,m)* transformations are used for an optimum management with redundancy, providing of the set requirements to networks dependability, optimum redundancy of telecommunication and computer networks at the minimum expenses of network resources.

Постановка проблемы. Анализ последних исследований

Задача обеспечения гарантоспособных вычислений (dependable computing) в условиях отказов и неисправностей устройств возникла уже на первых этапах развития вычислительной техники [1,2]. В дальнейшем понятие гарантоспособности (dependability) компьютерных систем и сетей существенно расширилось. Гарантоспособность является средством (технологией), гарантирующим достоверность информации в результате ее накопления, преобразования, хранения, обработки и передачи при наличии внешних и внутренних возмущений при функционировании вычислительной системы. Построение распределенных вычислительных систем потребовало учета гарантоспособности телекоммуникационных систем, объединяющих компьютеры в сети, а также воздействия множества деструктивных факторов, таких как несанкционированный доступ, пассивные и активные помехи [3,4]. Обеспечение необходимого уровня гарантоспособности тесно связано с такими понятиями, как надежность, живучесть, отказоустойчивость и другими, хорошо известными современной науке [5,6].

Проблемы повышения вероятности безотказной работы отказоустойчивых реконфигурируемых многопроцессорных систем управления рассматриваются в [7,8].

Дальнейшее развитие информационных компьютерных технологий сопровождается ростом требований к качеству функционирования специализированных систем критического назначения. Проблемы обеспечения качества обслуживания трафика (QoS) особо актуальны в системах критического назначения. Особую роль играет гарантоспособность в работе автономных вычислительных средств, в частности бортовых авиационных и космических систем [9-11].

В гражданской авиации интенсивно ведутся работы по разработке и внедрению всемирной авиационной телекоммуникационной сети (ATN), построенной на принципах архитектуры открытых информационных систем и концепции CNS/ATM “single sky.

Основным средством предупреждения и парирования опасных ситуаций, связанных с отказами компонентов систем критического назначения является использование различного вида избыточности [9-12].

При системном подходе к выбору способов применения избыточности и построении гарантоспособных систем из элементов ограниченного качества и надежности, как правило, решают три проблемы:

выбор альтернативных принципов действия, структур и элементов;

анализ гарантоспособности альтернативных систем с учетом ограничений на характеристики систем и ресурсы;

принятия решений относительно оптимального обеспечения гарантоспособности.

Первые постановки указанных проблем относятся к 50-70 годам прошлого столетия. Они обозначены в ставших уже классическими работах

Э. Мура и К. Шеннона, которые были посвящены методу поэлементного резервирования [1]; Дж. Немана, который предложил метод мажоритарной обработки сигналов – «метод голосования большинством сигналов» [2].

Основные идеи, предложенные авторами, были направлены на расширение функций систем в направлении самоконтроля и самодиагностирования, на заблаговременное введение такой структурной и информационной избыточности, которая позволяла парировать возможные отказы элементов и устройств, бороться с помехами в каналах управления.

Системы, при построении которых в той или иной форме используются и развиваются эти идеи, получили название «функционально-избыточных систем (ФИС)» или «структурно-избыточных систем (СИС)». Такие системы находят все более широкое применение при построении специализированных компьютерных сетей, необходимыми компонентами которых служат функционально-избыточные телекоммуникационные системы (ФИТС).

В настоящее время остаются недостаточно изученными проблемы оценивания функционирования телекоммуникационных и компьютерных сетей по интегральным показателям обеспечения гарантоспособности, самодиагностирования в штатном режиме, самовосстановления, реконфигурации архитектуры и структуры телекоммуникационных и компьютерных сетей при обнаружении отказавших элементов.

Телекоммуникационные и компьютерные сети можно рассматривать как сетевые системы массового обслуживания вычислительных и логических процедур, а всю совокупность процессов обслуживания этих процедур обвединять одним общим понятием „обслуживание трафика данных”[13,14]. Под работоспособностью телекоммуникационной и компьютерной сети в дальнейшем будем понимать ее способность обеспечивать заданное качество обслуживания сетевого трафика в условиях отказов элементов, действия внутренних и внешних помех, а также других видов несанкционированных воздействий на сеть. Из этого определения следует, что для количественного определения показателей гарантоспособности необходимо задавать эталоны *QoS* - качества обслуживания трафика для определения показателей качества, возможные виды отказов, сбоев, атак, помех, несанкционированных воздействий и т.п.

Цель данной работы – разработать модели и алгоритмы оптимального управления гарантоспособностью специализированных телекоммуникационных и компьютерных сетей, в которых для оптимального управления гарантоспособностью используют прямое и обратное *ZG*-преобразования [9-12].

Для достижения этой цели ставятся и решаются следующие задачи:

построение моделей и алгоритмов расчета и оптимального обеспечения гарантоспособности телекоммуникационных и компьютерных сетей;

исследование построенных моделей и алгоритмов расчета оптимального обеспечения гарантоспособности телекоммуникационных и компьютерных сетей;

выявление закономерностей изменения параметров моделей;

разработка практических рекомендаций по результатам анализа предложенных моделей и алгоритмов.

Изложение основного материала исследования

Предложенные Игнатовым В.А. и Захаренковым В.В. прямое и обратное *ZG*-преобразования для повышения надежности систем автоматизированного самолетовождения [10] являются дальнейшим обобщением скалярных методов мажорирования на векторный случай. Они позволяют управлять функционально-сигнальной избыточностью так, чтобы оптимально обеспечивать требуемое качество функционирования систем.

На рис.1 приведена обобщенная структурная схема *ZG(n,m)*- системы, выполняющая функции многоканальной телекоммуникационной системы между двумя узлами сети. Через *n* обозначено число каналов в неизбыточной телекоммуникационной системе (НТКС), через *m* обозначено число резервных каналов в функционально-избыточной телекоммуникационной системе (ФИТКС).

Упрощенно принцип действия *ZG(n,m)*- системы можно пояснить следующим образом. В прямом *ZG*-преобразовании выходной векторный сигнал *Xn,* размерности *n,* узла связи (маршрутизатора), выполняющего функции источника сигналов (данных), с помощью первого генератора опорных сигналов (ГОС1) преобразуется *A* – преобразователем в *n + m* избыточный векторный сигнал *Yn+m,* размерности *n+m,* так, что

каждая координата выходного сигнала $Yn+m$ преобразователя содержит информацию о всех координатах векторного входного сигнала Xn. При прохождении по многоканальной линии связи, составляющие сигнала $Yn+m$ подвергаются разного рода несанкционированным воздействиям $\xi n+m$ от источников несанкционированных воздействий (ИНВ), описанных оператором L. В обратном ZG-преобразовании, при оптимальной обработке в D - преобразователе принятых избыточных сигналов с помощью второго генератора опорных сигналов (ГОС2) обнаруживают и исправляют искаженные из-за помех и несанкционированных воздействий передаваемые сигналы. На выходе D – преобразователя получают оптимальные по методу максимального правдоподобия оценки X^*_n.переданных сигналов X_n.

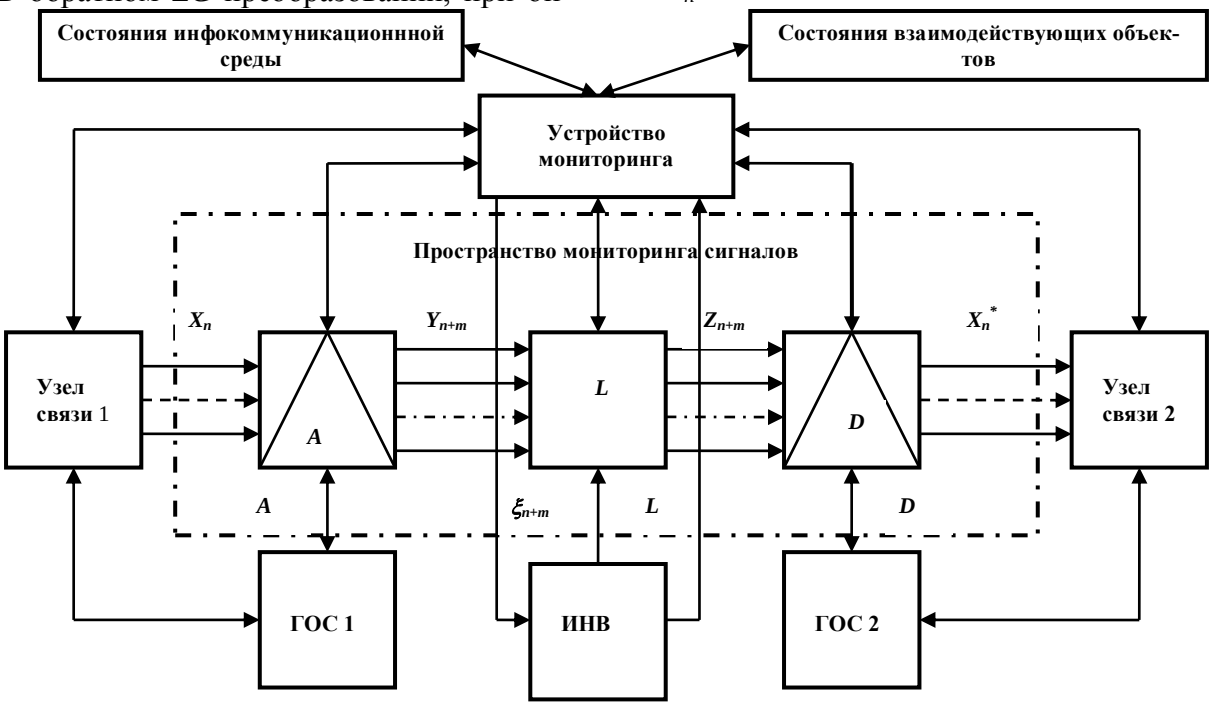


Рис. 1. Структурная схема ZG(n,m)- системы

Нетрудно заметить, что в $ZG(n,m)$- системе используется фрактальная избыточность, которую можно количественно измерить показателем фрактальной избыточности

$$r(n,m) = m / (n+m) \qquad (1)$$

Эта избыточность порождает общее число возможных решений в D –преобразователе

$$N(n,m) = C^n_{n+m} = (n+m)!/ n!\, m! \qquad (2)$$

Пример 1. Рассмотрим простейшую бинарную ФИТКС $ZG(n,m)$- систему с параметрами $n = 2$, $m = 1$, $r1(2,1) = m/(n+m) = 1/(2+1) = 1/3$. Используя (2), получим $N(2,1) = C^2_{2+1} = (2+1)!/ 2!1! = 1.2.3/1.2.1 = 3$. Следовательно, в обратном D – преобразователе $ZG(2,1)$- системы формируется не одно, как обычно принято, единственное решение, а множество из трех решений, из которых D – преобразователь должен сформировать одно, по определенному критерию оптимальное решение относительно переданных двух сигналов. Такая система остается гарантоспособной при отказе или искажении сигнала в любом одном канале, так как позволяет правильно вычислить два переданных сигнала по принятым сигналам «оставшимся в живых» двух каналов. Необходимо отметить существенное отличие векторного мажорирования от скалярного мажорирования. При скалярном мажорировании к каждому одному каналу необходимо было бы организовывать еще два канала. Это, в конечном счете, привело бы к избыточному введению избыточных четырех каналов, то есть, к шестиканальной системе. Таким образом, использование принципа в формировании избыточных сигналов «один за всех и все за одного» позволяет в данном примере в 4 раза уменьшить избыточность и, соответственно, затраты всех видов ресурсов.

Пример 2. Рассмотрим более сложную бинарную ФИТКС $ZG(n,m)$-систему с параметрами $n = 2$, $m = 2$, $r2(2,2) = m / (n+m) = 2/(2+2) = 1/2$. Используя (2), получим $N(2,2) = C^2_{2+2} = (2+2)!/2!2! = 1.2.3.4/ 1.2.\ 1.2 = 6$. Следовательно, в D – преобразователе $ZG(2,2)$- системы формируется множество из шести решений. Из них

преобразователь формирует одно, по определенному критерию оптимальное, решение относительно двух переданных сигналов. Такая система остается гарантоспособной при отказе любых двух каналов.

Пример 3. Рассмотрим бинарную ФИТКС *ZG(n,m)*-систему с параметрами *n = 2, m = 3, r3(2,3) = m / (n+m) = 3/(2+3) = 3/5*. Используя (2), получим $N(2,3) = C^2_{2+3} = (2+3)!/\ 2!3! = 1.2.3.4.5/\ 1.2.1.2.3 = 10$. Следовательно, в *D* – преобразователе *ZG(2,3)*-системы существует множество из десяти решений. Из них формируется единственное, по определенному критерию оптимальное решение, относительно двух переданных сигналов. Такая система остается гарантоспособной при отказе любых трех каналов.

Устройство мониторинга (embedded system) состояния каналов *ZG(n,m)*- системы обеспечивает контроль и диагностирование сигналов во всех каналах, выявление отказов и искажений сигналов, отключение отказавших каналов, то есть адаптивную реконфигурацию структуры системы, самовосстановление сигналов отказавших каналов по сигналам работоспособных каналов, взаимодействия с управляющими устройствами узлов связи 1, 2 и другие функции. Пространство мониторинга сигналов (рис.1) выделено штрихпунктирной линией. Основное назначение мониторинга в этой *ZG(n,m)*-системе – обеспечить гарантоспособность передачи сигналов узла сети 1 в узел 2, при воздействии на сеть внутренних и внешних деструктивных факторов.

Рассмотрим кратко методику математического моделирование работы *ZG(n,m)*- системы. Выходной сигнал X_n. узла связи *1* преобразуется *A*–преобразователем в избыточный векторный сигнал Y_{n+m}, размерности *n+m,* в соответствии с операторным уравнением

$$Y_{n+m} = \grave{A}\, X_n , \qquad (3)$$

где *A(n+m)Чm* – матрица *A* характеризует ориентацию в *(n+m)*-ом пространстве проекций *n*-вектора X_n., на Y_{n+m} так, что *(n+m)*-вектор Y_{n+m} содержит все проекции X_n. и удовлетворяет уравнению (3).

По умолчанию рассматриваются только те матрицы *A*, ранг которых равен *n,* а любая комбинация из *n* строк такой матрицы также есть *nЧn*-матрица ранга *n*. Предполагается, что в каналах многоканальной линии связи составляющие вектора Y_{n+m} искажаются случайными несанкционированными воздействиями и помехами. Эти искажения описываются оператором *L:*

$$Z_{n+m} = L(Y_{n+m,}\, X_{n+m}) \qquad (4)$$

В результате на вход *D* – преобразователя поступает сигнал Z_{n+m} , который может существенно отличаться от Y_{n+m} .

Различают два вида искажений: малые искажения и «выбросы» (отказы). Предполагается, что выбросы в рассматриваемый момент времени могут изменять не более *m* координат избыточного вектор-сигнала.

На основе принципов «голосования большинством или меньшинством» определяется диагностическая процедура и операторы алгоритма, позволяющие устройству мониторинга идентифицировать координаты выбросов и нейтрализовать последствия выбросов, выполняя оптимальное оценивание вектора X_n по искаженному вектору Z_{n+m} [8, 15].

Оптимальное по методу наименьших квадратов единственное преобразование избыточного *n+m* мерного векторного сигнала Z_{n+m} в оценку X^*_n векторного сигнала X_n , *n* -мерного базиса, выполняется на основе расширенной псевдоматрицы

$$D = [B\,A]^{-1} B, \qquad (5)$$

где определенный выбор из конечного множества *nЧm*-матрицы *B* ранга *n* для заданной *(n+m)Чn*-матрицы *A* делает оптимальное преобразование единственным.

Для $B=A^T$ матрица *D* в (5) является псевдообратной матрицей [7,8].

При построении алгоритмов диагностирования и реконфигурации структуры *ZG(n,m)*-системы используется проверочная матрица вида

$$S = AD - E \ , \qquad (6)$$

где *E* – единичная матрица.

Для реконфигурации структуры ФИТКС введена коммутирующая диагональная *(n+m)Ч(n+m)*-матрица $G_{ij\ldots k}$, в которой *i-й, j-й, …, k-й* элементы нулевые, а все другие равны единице. Индексы *i, j, …, k* по определению матрицы $G_{ij\ldots k}$ не могут принимать значения более чем *n+m.*

Авторами доказано следующее утверждение: если число *h* индексов матрицы $G_{ij\ldots k}$, равных нулю, удовлетворяет условию

$$h \le m \ , \qquad (7)$$

где *m* – число избыточных каналов, то обратное *ZG*-преобразование

$$X^*_n = \left[\ B\ G_{ij\ldots k}\ A\ \right]^{-1} B\ G_{ij\ldots k}\ Z_{n+m} \ , \qquad (8)$$

полученное с помощью матрицы (5) путем подстановки в нее матрицы $G_{ij...k}$, инвариантно относительно числа h.

Для оптимального обеспечения гарантоспособности *ZG(n,m)*-системы необходимо выб-рать значение критерия оптимальности (целевой функции), построить математическую модель критерия, эталонные значения для расчета показателей качества обслуживания трафика, ввести показатели гарантоспособности системы в установившемся режиме статистического равновесия.

В качестве критерия оптимальности Π выберем критерий среднего риска как мареметическое ожидание потерь, обусловленных, с одной стороны, нарушениями гарантоспо-собности, а, с другой стороны, затратами на создание и поддержание качественной работы системы обеспечения гарантоспособности

$$P(y,\ \pi_1;\ \pi_2;\ p_1;\ p_2;\ \alpha;\ \beta) = \pi_1\, p_1\, y^{-\alpha} + \pi_2\, p_2\, y^{-\beta}\ , \quad (9)$$

где нормированный показатель гарантоспособности

$$y = q\ /\ q_0\ , \quad (10)$$

q и q_0 – соответственно, показатели гарантоспособности рассматриваемой системы и системы аналога или прототипа;

π_1 и π_2 – соответственно, потери из-за нарушения гарантоспособности и затраты на обеспечение гарантоспособности;

p_1 и p_2 – соответственно, вероятности противоположных событий – нарушения гарантоспособности и обеспечения гарантоспособности, для этих вероятностей соблюдается условие нормировки как для полной группы событий

$$p_1 + p_2 = 1\,; \quad (11)$$

α и β – показатели влияния избыточности *ZG(n,m)*-системы на нормированное значение показателя гарантоспособности.

Значение критерия (9) удобно рассчитывать для одного канала избыточной и неизбыточной системы. Можно предположить, что затраты на обеспечение работы одного канала избыточной системы будут линейно возрастать с увеличением числа резервных каналов, тогда

$$p_2 = p_1\,(n + m)\ /\ n = p_1\, r(n,m)/n/m \quad (12)$$

Для определения вероятностей p_1 и p_2 необходимо составить систему дифференциальных уравнений Колмогорова–Чепмена, которые описывают динамику изменения состояний *ZG(n,-m)*–системы, и для установившегося режима статистического равновесия определить эти вероятности. Граф переходов *ZG(n,m)*- системы представлен на рис.2, где через S_k, $k = 0, m+n$, обозначено k-ое состояние системы, Λ и M – соответственно, интенсивность отказа любого одного канала и интенсивность восстановления работоспособности этого канала после отказа, $P_k(t)$ – вероятность пребывания системы в состоянии S_k в момент времени t. Система дифференциальных уравнений имеет вид:

$$\begin{aligned}
&P'_0(t) = -\Lambda P_0(t) + M P_1(t),\\
&P'_1(t) = \Lambda P_0(t) - (\Lambda + M) P_1(t) + M P_2(t);\\
&P'_k(t) = \Lambda P_{k-1}(t) - (\Lambda + M) P_k(t) + M P_{k+1}(t);\ k = 2,\ (n+m-1);\\
&\sum P_k(t) = 1.
\end{aligned} \quad (13)$$

Для исследования динамики изменения гарантоспособности необходимо задать начальные условия

$$P_k(t=0) = P0_k,\ k = 0,\ (n+m)\ ; \quad (14)$$

и решить задачу Коши. Состояния $S_0,\ S_1,\ldots\ S_m$ образуют подмножество состояний гарантоспособности системы, а состояния $S_{m+1},\ S_{m+2},\ldots\ S_{m+n}$ образуют подмножество состояний нарушения гарантоспособности системы. Поэтому вероятность обеспечения гарантоспособности системы в момент времени t равна

$$p_2(t) = \sum P_j(t)\ ,\ j = 0,\ m. \quad (15)$$

Вероятность нарушения гарантоспособности системы в момент времени t равна

$$p_1(t) = \sum P_k(t),\ k = (m+1),\ (n+m)\ . \quad (16)$$

Формулы (15), (16) позволяют исследовать переходный и установившийся режимы обеспечения гарантоспособности системы.

На практике чаще всего важны характеристики установившегося режима статистического равновесия, который определяется условием

$$P'_k(t) = 0,\ k = 0,\ (n+m)\ . \quad (17)$$

Для этого режима система дифференциальных уравнений Колмогорова–Чепмена переходит в систему алгебраических уравнений, неизвестными в которой являются вероятности

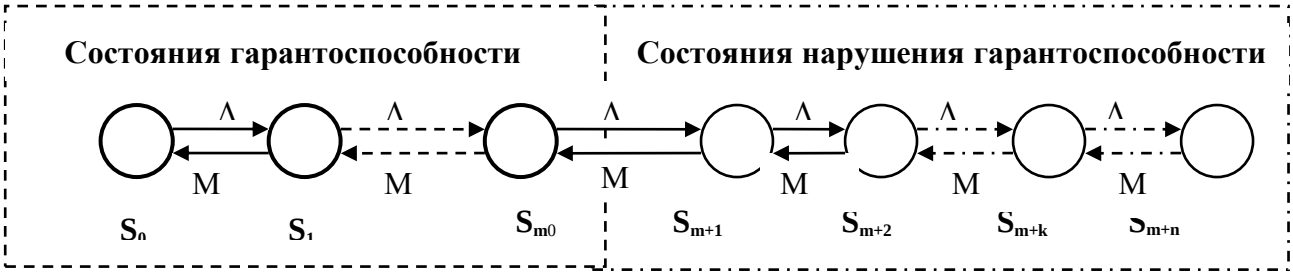


Рис. 2. Граф изменения гарантоспособности ZG(n,m)- системы

состояний, не зависящие от времени. Решая эту систему относительно этих вероятностей, получим

$$P_k = \rho^k \; / \; \sum \rho^k \;,\; \mathrm{k} = 0,\ (n+m)\;, \qquad (18)$$

где отношение интенсивностей

$$\rho = \Lambda / M\;. \qquad (19)$$

Вероятность обеспечения гарантоспособности системы в установившемся режиме статистического равновесия равна

$$p_{2\infty} = \sum\ r^i \; / \sum\ r^k\;;\ i = 0,\ m;\ k = 0,\ (n+m)\;. \qquad (20)$$

Вероятность нарушения гарантоспособности системы в установившемся режиме статистического равновесия равна

$$p_{1\infty} = \sum\ r^j \; / \sum\ r^k;\ j = m+1, (n+m);\ k = 0,\ (n+m)\;. \qquad (21)$$

Из формулы (18) следует, что исходными данными для расчета вероятностей (20), (21) являются интенсивности Λ и M, которые по формуле (19) позволяют рассчитать отношение ρ, а по нему с помощью формул (20), (21) искомые вероятности.

Пример 4. Рассмотрим простейшую бинарную ФИТКС *ZG(n,m)*- систему с параметрами $n = 2$, $m = 1$, $r = m / (n+m) = 1/(2+1) = 1/3$. Рассчитаем показатели гарантоспособности в установившемся режиме при $\rho = 0.25$.

Используя формулы (20), (21), получим

$p_{1\infty} = (\rho^2 + \rho^3) / (\rho^0 + \rho^1 + \rho^2 + \rho^3) = (0.063 + 0.016)/(1 + 0.25 + 0.063 + 0.016) = 0.058824$

$p_{2\infty} = (\rho^0 + \rho^1) / (\rho^0 + \rho^1 + \rho^2 + \rho^3) = (1.0 + 0.25) / (1 + 0.25 + 0.063 + 0.016) = = 0.941176$

Как видим, расчет показателей гарантоспособности *ZG(n,m)*-систем в установившемся режиме не представляет принципиальных трудностей. Исходные данные в виде интенсивностей Λ и M определяют известными методами теории надежности.

Перейдем к определению параметров α и β в формуле (9). Можно предположить, что параметр α будет зависеть обратно пропорционально от числа резервных каналов m, а параметр β - прямо пропорционально числу m, тогда эти зависимости удобно аппроксимировать следующими функциями:

$$a = n \; / \; (n + m) = 1 \; / \; (1 + r); \qquad (22)$$

$$b = (n + m) \; / \; n = 1 + r. \qquad (23)$$

Подставляя эти функции в формулу (9), получим итоговую зависимость критерия среднего риска от всех определяющих гарантоспособность *ZG(n,m)*-системы параметров. В формуле (9) остается один свободный параметр q_0, которым, как будет показано в дальнейшем, задается база сравнительного анализа гарантоспособности различных *ZG(n,m)*- систем.

Решая полученное уравнение оптимизации гарантоспособности из (9) найдем оптимальное значения y_{opt}:

$$y_{opt} = \left[\pi_1 n^2 \sum_{k=2}^{3} \rho^k \Big/ \pi_2 (n+m)^2 \sum_{k=0}^{1} \rho^k \right]^Z \;, \qquad (24)$$

где безразмерный показатель степени

$$z := \frac{n(n+m)}{n^2 + (n+m)^2}\;. \qquad (25)$$

Подставляя оптимальное значение y_{opt} в формулу (9), найдем минимальное значение критерия среднего риска

$$\Pi_{\min}(y_0) = \pi_1 \sum_{k=2}^{3} \rho^k \Big/ \sum_{k=0}^{1} \rho^k * (y_0)^{\frac{-n}{n+m}} + \pi_2 \sum_{k=0}^{1} \rho^k \Big/ \sum_{k=0}^{1} \rho^k * (y_0)^{\frac{n+m}{n}}\;. \qquad (26)$$

Типовой алгоритм решения задачи оптимального обеспечения гарантоспособности *ZG(n,m)*-систем содержит следующие шаги: выбор критерия оптимальности и параметров, определяющих гарантоспособность системы; установление связей и отношений между критерием и определяющими параметрами, построение графа изменения состояний гарантоспособности системы, составление дифференциальных уравнений; определение вероятностей состояний для режима статистического равновесия системы; расчет показателей гарантоспособности *ZG(n,m)*- системы в установившемся режиме; выбор аппроксимаций (22) и (23); решение уравнения оптимизации нормированного параметра гарантоспособности и получение аналитических соотношений типа (24) и (25) для координат экстремума критерия

среднего риска; параметрический анализ оптимального решения; формулировка выводов и практических рекомендаций.

Пример 5. Покажем работу этого алгоритма при оценивании эффективности использования избыточности в бинарных *ZG(n,m)*- системах оптимального обеспечения гарантоспособности при $n = 2$, $\pi_1 = 1$, изменении числа резервных каналов m от 1 до 3 при $\rho = 0.25$.

Индексные показатели уменьшения нормированных оптимальных значений параметра гарантоспособности и нормированных минимальных значений критерия среднего риска, рассчитанные в системе MathCAD, имеют следующие значения:

$$W_{У21}=1.962,\ W_{У31}=2.829,\ W_{У32}=1.442,$$
$$W_{П21}=4.499,\ W_{П31}=21.554,\ W_{П32}=4.791 \qquad (27)$$

Рис. 3 иллюстрирует особенности формирования оптимального значения параметра гарантоспособности *ZG(n,m)*- систем. Через $\Pi_1(y)$ обозначена составляющая среднего риска, обусловленная нарушениями гарантоспособности, через $\Pi_2(y)$ обозначена составляющая среднего риска, обусловленная обеспечением и поддержанием гарантоспособности *ZG(n,m)*- системы. На рис.4 показаны графики зависимости критерия среднего риска для бинарных *ZG(n,m)*- систем при изменении числа резервных каналов от 1 до 3 при $\rho = 0.25$.

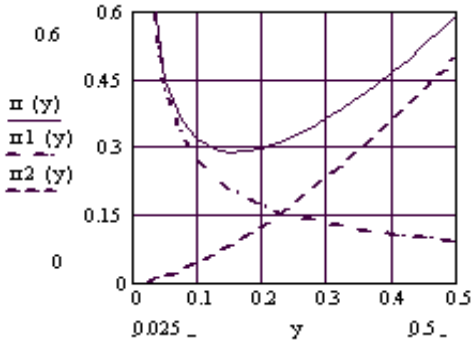


Рис. 3. Формирование оптимального значения параметра гарантоспособности

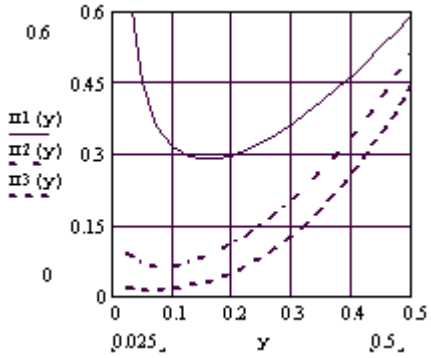


Рис. 4. Изменение критерия среднего риска для ZG(n,m)- систем при n = 2, m = 1; 2; 3

Анализ результатов позволил выявить следующие закономерности: особо значимый рост эффективности *ZG(n,m)*- систем наблюдается при первом увеличении числа резервных каналов – от 1 до 2; индексные показатели уменьшения оптимальных значений нормированного параметра гарантоспособности убывают с ростом числа резервных каналов; индексные показатели уменьшения минимальных значений критерия среднего риска растут с ростом числа резервных каналов. В заключение рассмотрим использование показателя гарантоспособности системы аналога или прототипа q_0 для определения оптимального значения q_{opt} *ZG(n,m)*- систем:

$$q_{opt} = y_{opt}\, q_0. \qquad (28)$$

Пример 6. Предположим, что вероятность нарушения гарантоспособности одного канала системы-прототипа равна $q_0 = 0.015$. Определим оптимальное значение вероятности нарушения гарантоспособности может быть у одного канала *ZG(n,m)*- системы оптимального обеспечения гарантоспособности при $n = 2$, $m = 1, 2, 3$.

Используем результаты примера 5 как исходные данные: $y1_{opt} = 0.158648$, $y2_{opt} = 0.080875$, $y3_{opt} = 0.056083$. Подставляя эти значения в формулу (28), получим оптимальные значения вероятностей нарушения гарантоспособности соответствующих *ZG(n,m)*: $q1_{opt} = y1_{opt}\, q_0 = 0.158648.\ 0.015 = 2.38\ 10^{-3}$, $q2_{opt} = y2_{opt}\, q_0 = 0.0808750.\ 0.015 = 1.213.\ 10^{-3}$, $q3_{opt} = y3_{opt}\, q_0 = 0.0560830.\ 0.015 = 8.412\ 10^{-4}$.

По результатам исследования можно сделать следующие выводы

1. Для оптимального обеспечения гарантоспособности телекоммуникационных и компьютерных сетей основными являются задачи построения моделей и алгоритмов оптимального обеспечения гарантоспособности телекоммуникаци-

онных и компьютерных сетей; параметрический анализ моделей оптимизации гарантоспособности, разработка практических рекомендаций.

2. Использование *ZG(n,m)*- систем позволяет оптимально управлять избыточностью специализированных телекоммуникационных и компьютерных сетей, обеспечивать заданные требования к их гарантоспособности при минимальных затратах всех видов сетевых ресурсов.

3. Полученные в этой работе результаты полезны для определения эталонов качества обслуживания трафика на разных уровнях модели открытых систем.

Список литературы

1. Neumann J. Probabilistic logic and the synthesis of reliable organisms from unreliable components. // Neumann J. Automata Studies. Ann. of Math. Studies. №34. 1952.
2. Moore E.F. Reliable circuits using less reliable relays // Moore E.F., Shannon C. E. J. Franklin Inst.,– 1956, p. 262.
3. Харченко В.С. Гарантоспособность и гарантоспособные системы: элементы методологии // Радіоелектронні та комп'ютерні системи. – 2006. – №5. – С.7-19
4. Харченко В.С. Парадигмы и принципы гарантоспособных вычислений: состояние и перспективы развития // Радіоелектронні і комп'ютерні системи. – 2009. – №2(36) – С. 91-100
5. Теслер Г.С. Концепция построения гарантоспособных вычислительных систем // Математичні машини і системи. 2006. – №1, – С. 134-145.
6. Мудла Б.Г., Єфімова Т.І., Рудько Р.М. Гарантоздатність як фундаментальний узагальнюючий та інтегруючий підхід // Математичні машини і системи. – 2010. – №2, – С. 148-165.
7. Романкевич А.М., Романкевич В.А., Мораведж Сейед Милад О повышении надежности реконфигурируемых отказоустойчивых многопроцессорных систем управления сложными объектами // Электронное моделирование. – 2010. – Т.32. № 4. – С.85-92
8. Романкевич А.М., Майданюк И.В., Романкевич В.А. Частный случай граничных оценок при построении и преобразовании GL-модели // Радіоелектронні і комп'ютерні системи – 2010, – №6(47) – С. 236-243
9. Игнатов В.А., Маньшин Г.Г., Трайнев В.А. Статистическая оптимизация качества функционирования электронных систем. М.: Энергия. – 1974. 264 с.
10. Игнатов В.А. Диагностические комплексы систем автоматического самолетовождения. Игнатов В.А., Паук С.М., Конахович Г.Ф Под ред. В.А.Игнатова. М.: Транспорт. - 1975, 272 с.
11. Игнатов В.А., Захаренков В.В. Мажоритарное устройство для выделения проекций векторной величины. А.С. СССР №782162. Б.И. СССР. №6. 1978.
12. Игнатов В.А., Захаренков В.В. Устройство выбора непрерывного сигнала по принципу большинства. А.С. СССР №828448. Б.И. СССР. №8. 1981.
13. Игнатов В.А. Теория информации и передачи сигналов. М.: Сов. радио, 2-ое изд. 1990, 280 с.
14. Даниліна Г.В., Гузій М.М., Ігнатов В.О., Милокум Я.В. Методи і алгоритми оптимального управління трафіком в обчислювальних мережах // – Проблеми інформатизації та управління. К.: НАУ, – 2006. – Вип.17. – С. 32-37.

УДК 519.614

ПЕТРЕНКО О.О.

МЕТОД ОБЧИСЛЕННЯ ВЛАСНИХ ВЕКТОРІВ МАТРИЦІ, ЩО БАЗУЄТЬСЯ НА ЗШИРЕНОМУ МЕТОДІ ДІАГОНАЛЬНОЇ МОДИФІКАЦІЇ

Обгрунтована автоматична процедура обчислення власних векторів матриці, яка застосовується для рішення відповідної виродженої системи рівнянь методом діагональної модифікації, яка забезпечує також обчислення власних векторів для кратних власних значень і можливість змінювати власні значення з збереженням власних векторів.

Proved automatic procedure for calculating the eigenvectors of the matrix, which is used for the solution of the degenerate system of equations using a diagonal modification, which provides the calculation of eigenvectors for multiple eigenvalues and the ability to change their meaning with preservation of eigenvectors.

Вступ

Відомо, що вектори x називають *власними векторами* матриці A, а коефіцієнти пропорційності λ – власними її значеннями, якщо виконується рівність

$$Ax = \lambda x, \qquad (1)$$

тобто після множення матриці A на власний вектор x маємо новий вектор $y = Ax$, який відрізняється від вектора x тільки своєю довжиною (модулем) , зберігаючи напрямок (орієнтацію) у багатовимірному просторі.

Вираз (1) звичайно переписується у вигляді тотожності

$$(A - \lambda E)x = 0, \qquad (2)$$

з якої при $x \neq 0$ робиться висновок, що матриця $B = (A - \lambda E)$ є виродженою матрицею, виводиться відома формула для обчислення власних значень матриці A:

$$\det (A - \lambda E) = P(\lambda) = 0 \qquad (3)$$

Таким чином, будь яка матриця A не може мати нульовий власний вектор x_i, але може мати нульове власне значення $\lambda_i = 0$. Тоді згідно (2) сама матриця A повинна бути виродженою, і як слідство її не можна обернути.

Якщо визначник (3) розкрити відносно значень λ, то отримаємо так зване *характеристичне рівняння* матриці A у вигляді полінома n–степеня $P(\lambda)$ відносно власних значень. Розв'язок цього рівняння визначає множину всіх власних значень матриці, при цьому існує багато методів (Крилова, Фадєєва , Данилевського та інш.) обчислення коефіцієнтів поліному $[P(\lambda)$ без розкриття самого визначника (3) [1].

Кожному власному значенню матриці λ відповідає свій власний вектор x, але для його обчислення необхідно вирішити рівняння (2) з виродженою матрицею $B = (A - \lambda E)$, при цьому однозначність рішення втрачається, бо визначник матриці $\det B = 0$. Це більш складна задача, і в існуючій навчальній і монографічній літературі автоматизація її рішення майже зовсім не розглядається, особливо для випадку кратних власних значень матриці [1,2,3,4]. В цій роботі пропонується новій підхід до вирішення задачі власних векторів, що базується на використанні розширеного методу діагональної модифікації [5].

1. Постановка задачі

Умова $detB = 0$ при LU-розкладанні збігається з умовою $\det U = 0$, тому що завжди $detL = 1$ (при виборі одиничних діагональних елементів цієї матриці).В свою чергу, $detU = 0$ через появу ненульового діагонального елемента (звичайно в самому кінці U матриці, якщо застосовується впорядкування рівнянь при їх рішенні).

Пропонується наступний метод виходу з тупика і отримання безлічі можливих рішень лінійних систем рівнянь з виродженими матрицями коефіцієнтів:

1). Проводиться довільне коректування матриці U шляхом заміни її нульового діагонального елемента довільним значенням g, що вибирається ,наприклад , з урахуванням середнього значення елементів першого рядка матриці U.

2). Знаходиться рішення $\boldsymbol{x}^{(1)}$ для нової (вже не виродженої) задачі і заданого вектора правої частини систем рівнянь b.

3).Для знаходження інших можливих рішень виродженої системи пропонується знаходити додаткове рішення $\boldsymbol{x}^{(2)}$ для тієї ж нової (не ви-

родженої) задачі і допоміжного вектора правої частини *b1*, який *містить* одиницю в позиції, визначуваній номером рядка матриці *U*, нульовий діагональний елемент якої коректувався, і нулі у всій решті позицій

4). Можливі рішення виродженої задачі знаходиться комбінацією двох рішень не виродженої скоректованої задачі

$$\boldsymbol{x}=\boldsymbol{x}^{(1)}+\boldsymbol{k}\,\boldsymbol{x}^{(2)} \qquad (4)$$

де $k=i\,g$, i= +/- 1,2,3

Оскільки базове рішення $\boldsymbol{x}^{(1)}$, як було виявлено, не залежить від вибору корегуючої константи g, то коефіцієнт $\boldsymbol{k}$ в наведеній композиції можна вибрати з умови отримання заданого значення одного з компонентів можливого вектора рішення.

Приклад 1.

Задана система лінійних рівнянь

$$\begin{bmatrix}1 & 2 & 3\\ 4 & 5 & 6\\ 3 & 2 & 1\end{bmatrix}\begin{bmatrix}x_1\\ x_2\\ x_3\end{bmatrix}=\begin{bmatrix}14\\ 32\\ 10\end{bmatrix},$$

особливість якої виявляється при *LU*-розкладанні

$$L=\begin{bmatrix}1 & 0 & 0\\ 4 & 1 & 0\\ 3 & 4/3 & 1\end{bmatrix}\quad U=\begin{bmatrix}1 & 2 & 3\\ 0 & -3 & -6\\ 0 & 0 & 0\end{bmatrix},$$

тому що матриця *U* має нульовий діагональний елемент $u_{33}=0$.

Вибираємо $g=3$ і виправляємо матрицю *U,* а потім згідно виразу

$$Ly=b \qquad (5)$$

знаходимо проміжний вектор $\boldsymbol{y}^{(1)}$= [14 -24 0]^t

Далі згідно

$$U1\cdot x=y \qquad (6)$$

розв'язується трикутна система рівнянь з корегованою матрицею *U1* і знаходиться рішення $\boldsymbol{x}^{(1)}$=[-2 8 0]^t**.**

$$\begin{bmatrix}1 & 2 & 3\\ 0 & -3 & -6\\ 0 & 0 & 3\end{bmatrix}\begin{bmatrix}x_1^{(1)}\\ x_2^{(1)}\\ x_3^{(1)}\end{bmatrix}=\begin{bmatrix}14\\ -24\\ 0\end{bmatrix}$$

Примітка 1: Можна переконатися, що одержуване рішення $\mathbf{x}^{(1)}$=[-2 8 0]^t не залежить від вибору корегуючої константи *g* і співпадає з рішенням**,** формованим пакетом `Mathematica` при використанні стандартного оператора ***`LinearSolve[A,b]`*** для вирішення заданої виродженої системи рівнянь.

Враховуючи положення нульового елементу u_{33}, вибираємо вектор *b1*=[0 0 1]^t і повторюємо рішення. Як наслідок отримуємо проміжний вектор $\boldsymbol{y}^{(2)}$=[0 0 1]^t і додаткове рішення $\boldsymbol{x}^{(2)}$= [1/3 -2/3 1/3]^t:

Тепер, комбінуючи базові рішення $\boldsymbol{x}^{(1)}$ і $\boldsymbol{x}^{(2)}$ для різних коефіцієнтів *i* , отримуємо

Послідовність можливих рішень:

при i=1 $\boldsymbol{x}=\boldsymbol{x}^{(1)}+3\,\boldsymbol{x}^{(2)}$=[-1 6 1]^t

при i=2 $\boldsymbol{x}=\boldsymbol{x}^{(1)}+6\,\boldsymbol{x}^{(2)}$= [0 4 2]^t

при i=3 $\boldsymbol{x}=\boldsymbol{x}^{(1)}+9\,\boldsymbol{x}^{(2)}$= [1 2 3]^t

і т.д. в справедливості яких переконуємося прямою підстановкою в вихідні рівняння .

Цікаво відмітити, що сума всіх складових векторів можливих рішень інваріантна і рівна для даного прикладу шести.

Примітка 2: Якщо заданий вектор правої частини виродженої систем рівнянь *b* нульовий, то проміжний вектор $\boldsymbol{y}$ з виразу (5) завжди буде теж нульовий при вирішенні системи з трикутною матрицею ***L.*** Рішення системи (6) *з* корегованою трикутною матрицею *U1* і нульовим вектором правої частині $\boldsymbol{y}$, завжди також буде нульовим $\boldsymbol{x}^{(1)}=\mathbf{0}$. В цьому випадку згідно виразу (4) рішення виродженої задачі визначається лише одним рішенням не виродженої скоректованої задачі

$$\boldsymbol{x}=\boldsymbol{k}\,\boldsymbol{x}^{(2)} \qquad (7)$$

де *k*- масштабний коефіцієнт.

3. Основний матеріал

3.1. Власні вектори для дійсних різних власних значень

Розглянемо метод обчислення власних векторів, що пропонується, спочатку для випадку дійсних рознесених власних значень матриці $|\lambda_1| > |\lambda_2| \geq |\lambda_3| > \ldots \geq |\lambda_n|$. Враховуючи співвідношення (7) , можна з використанням операторів пакету *Mathematica* скласти відповідну програму “Vector” (рис.1), на вхід якої подаються дані про матрицю *A*, вибране її власне значення *lamda* і нульовий вектор *b* правої частини тотожності (2). На виході програми отримуємо нормований власний вектор, що відповідає вибраному *lamda.*

```
In:=A;b;lamda;
n=Length[A];
B=A-lamda*IdentityMatrix[n];
( формується тотожність(2))
{lu,p,c}=LUDecomposition[B]
u=lu SparseArray[{i_,j_}/;j≥i→1,{n,n}];
  (виконується lu-перетворення і  обчислюється вироджена матриця u)
u1=ReplacePart[u,{n,n}->1];
```

(усувається виродження матриці u переходом до u_1 з елементом u_{nn}=1):

```
b1= ReplacePart[b,n->1];
```

(корегується вектор правої частини введенням елемента b_n=1);

```
x1=LinearSolve[u1,b1];
```

(вирішується система рівнянь (6))

```
x=Normalize[N[x1]]
нормується отримане рішення)
```

Рис.1. Програма "Vector" обчислення власних векторів для дійсних простих власних значень

Приклад 2.Проілюструємо запропонований підхід до обчислення власних векторів на прикладі матриці A:

$$A=\begin{bmatrix}5&1&2&0&4\\1&4&2&1&3\\2&2&5&4&0\\0&1&4&1&3\\4&3&0&3&4\end{bmatrix} \qquad (8)$$

яка має характеристичний поліном $F(z)= -1222+1153z-146z^2-79z^3+19z^4-z^5$.

П'ять дійсних коренів цього поліному можна визначити з допомогою стандартного оператора пакета *Mathematica* для вирішення нелінійних рівнянь *NSolve[F[z]==0 ,z]* і отримати $\lambda_1= 12.0258$, $\lambda_2=5.67255$, $\lambda_3=3.36188$, $\lambda_4=1.49766$ і $\lambda_5= -3.55784$.

Для обчислення першого власного вектора матриці(8) на вхід програми "Vector", приведеної на рис.1., треба подати:

A={{5,1,2,0,4},{1,4,2,1,3},{2,2,5,4,0},{0,1,4,1,3},{4,3,0,3,4}};

lamda= 12.0258;

b={0,0,0,0,0};

і отримати результат у вигляді:

x1={0.485107,0.411358,0.450672,0.343351,0.52389}

Згідно виразу (7) всі інші власні вектори, що визначаються вибраним власним значенням, будуть відрізнятися лише довільним масштабним множником.

При обчисленні наступних власних векторів для інших власних значень матриці також використовується програма"Vector" , лише міняється в вхідних даних величина власного числа *lamda*. Отримані результати зведені в табл.1.

Табл. 1. Власні значення і власні вектори матриці (8)

Власне значення	Власний вектор
$\lambda_1= 12.0258$	$\boldsymbol{x_1}$***={0.485107,0.411358,0.450672,0.343351,0.52389}***
$\lambda_2=5.67255$	$\boldsymbol{x_2}$***={0.440209,-0.0115007,-0.712073,-0.334081,0.432927}***
$\lambda_3=3.36188$	$\boldsymbol{x_3}$***={-0.616027,0.690172,-0.2933,0.0830426,0.226389}***
$\lambda_4=1.49766$	$\boldsymbol{x_4}$***={-0.306812,-0.56669,-0.152227,0.570039,0.486426}***
$\lambda_5= -3.55784$	$\boldsymbol{x_5}$***={-0.311869,-0.182146,0.425024,-0.662313,0.500256}***

Можна переконатися, що для отриманих власних векторів виконуються умови

$$x_i^T x_j = \begin{cases}0, & i \neq j\\ 1, & i = j\end{cases},$$

тобто власні вектори матриці, що притаманні дійсним рознесеним власним значенням матриці, утворюють *ортогональну систему векторів*.

Цікаво відмінити, що стандартні оператори *Eigenvalues[A]* та *Eigenvectors[A]* пакета *Mathematica* не в змозі знайти власні значення і власні вектори для матриці (8), але оператор *Eigensystem [N[A]]* дає результати, які співпадають з даними табл.1.

3.2. Власні вектори для кратних дійсних власних значень

Якщо матриця має кратні власні значення, то попередня процедура, яка базується на рішенні виродженої системи рівнянь (2), не дозволяє обчислити кілька незалежних власних векторів для одного значення λ_i, яке є кратним. Тому пропонується інший підхід, що передбачає при збереженні власних векторів таке перетворення матриці A, за яким один з її кратних коренів заміняється нульовим значенням (обнуляється), і задача зводиться до попередньої, коли всі власні значення нової матриці A_1 різні.

Для кратного власного значення λ_1 можна програмою рис.1 обчислити відповідний власний вектор x_1 , як то було зроблено вище, і побудувати нову матрицю A_1, яка на відміну від матриці A матиме просте власне значення λ_1 :

$$A_1 = A - \lambda_1 x_1 \otimes x_1, \qquad (9)$$

де застосовується множення векторів за Кронекером і власний вектор x_1. Відмінність операції множення векторів за Кронекером від інших операцій множень пояснюється на прикладі двох простих векторів у табл.2.

Табл. 2. Операції множення в пакеті Mathematica

Операція множення	Результат	Форма
Векторне *{a,b} *{c,d}*	*{ac,bd}*	*вектор*
Скалярне *{a,b}.{c,d}*	*ac+bd*	*скаляр*
KroneckerProduct [{a,b},{c,d}]	*{{ac,ad},{bc,bd}}*	*матриця*

Формулу (9) можна отримати з перетворення матриці за Йорданом

$$: A = \begin{bmatrix} x_1 \\ x_{11} \\ x_2 \\ \ldots\ldots \\ x_i \\ \ldots\ldots \\ x_n \end{bmatrix}^T \cdot \begin{bmatrix} \lambda_1 & 0\ldots & 0\ldots & 0\ldots & 0\ldots & 0 \\ 0 & \lambda_1 & \ldots\ldots & & \ldots & 0 \\ 0 & 0 & \lambda_2 & \ldots & & 0 \\ \ldots & \ldots & \ldots\ldots & \ldots & & \\ 0\ldots & 0\ldots & \lambda_i\ldots & 0\ldots & \ldots & 0 \\ \ldots & \ldots & \ldots & \ldots & & \\ 0\ldots & 0\ldots & 0\ldots & 0\ldots & \ldots & \lambda_n \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ x_{11} \\ x_2 \\ \ldots\ldots \\ M_i \\ \ldots\ldots \\ M_n \end{bmatrix}, \qquad (10)$$

якщо вираз (10) перепишемо так:

$$A_1 = \begin{bmatrix} x_1 \\ x_{11} \\ x_2 \\ \ldots\ldots \\ x_i \\ \ldots\ldots \\ x_n \end{bmatrix}^T \cdot \begin{bmatrix} 0.. & 0\ldots & 0\ldots & 0\ldots & 0\ldots & 0 \\ 0 & \lambda_1 & 0 & \ldots & & 0 \\ 0 & 0 & \lambda_2 & \ldots\ldots & & 0 \\ \ldots & \ldots & \ldots\ldots & \ldots & & \\ 0\ldots & 0\ldots & \lambda_i\ldots & 0\ldots & \ldots & 0 \\ \ldots & \ldots & \ldots & \ldots & & \\ 0\ldots & 0\ldots & 0\ldots & 0\ldots & \ldots & \lambda_n \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ x_{11} \\ x_2 \\ \ldots\ldots \\ x_i \\ \ldots\ldots \\ x_n \end{bmatrix} = A - \begin{bmatrix} x_1 \\ x_{11} \\ x_2 \\ \ldots\ldots \\ M_i \\ \ldots\ldots \\ M_n \end{bmatrix}^T \cdot \begin{bmatrix} \lambda_1.. & 0\ldots & 0\ldots & 0\ldots & 0\ldots & 0 \\ 0 & 0\ldots & 0\ldots & & 0\ldots & 0 \\ 0 & 0\ldots & 0\ldots & & 0\ldots & 0 \\ \ldots & \ldots & \ldots\ldots & \ldots & & \\ 0\ldots & 0\ldots & 0\ldots & 0\ldots & 0\ldots & 0 \\ \ldots & \ldots & \ldots & \ldots & & \\ 0\ldots & 0\ldots & 0\ldots & 0\ldots & \ldots & 0 \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ x_{11} \\ x_2 \\ \ldots\ldots \\ M_i \\ \ldots\ldots \\ M_n \end{bmatrix}$$

або $A_1 = A - \lambda_1 x_1 \otimes x_1$.

Програма “Vector1” , що обчислює другий власний вектор для кратного власного значення матриці шляхом здійснює обнуління одного з власних значень матриці A з збереженням власних векторів приведена на рис.2.

Приклад 3. Проілюструємо запропонований підхід до обчислення власних векторів для кратних власних значень на прикладі матриці A:

$$A = \begin{bmatrix} 5 & 4 & 2 \\ 4 & 5 & 2 \\ 2 & 2 & 2 \end{bmatrix}, \qquad (11)$$

Користуючись програмою “Vector” (рис.1), для власних значень λ_1=1 і λ_3=10 обчислюємо відповідні власні вектори x_1 і x_2, підставляючи такі вхідні дані:

A={{5,4,2},{4,5,2},{2,2,2}}; lamda; b={0,0,0};

Отримаємо значення x_1 та x_2 приведені в табл.3.

Тепер використовуємо програму “Vector1”, сформувавши для неї наступні вхідні дані:

A={{5,4,2},{4,5,2},{2,2,2}}; lamda=1;

x1={-0.447214,0.,0.894427};

і підраховуємо другий власний вектор x_{11} для кратного власного значення і вносимо його в табл.3.

яка має наступні власні значення {1,1,10}, тобто кратне значення $\lambda_{1,2}$=1 і дійсне значення λ_3=10.

```
In:=A;lamda;x1;
n=Length[A];
A1=A-lamda* KroneckerProduct[x1,x1]
```
(отримується перетворена матриця з простими власними значеннями)
```
B=A1-lamda*IdentityMatrix[n]
{lu,p,c}=LUDecomposition[B]
u=lu SparseArray[{i_,j_}/;j≥i→1,{n,n}];
u1=ReplacePart[u,{n,n}->1];
b1= ReplacePart[b,n->1];
X1=LinearSolve[u1,b1];
x11=Normalize[N[X1]]
```

Рис.2. Программа "Vector1" обчислення другого власного вектора для кратного власного значення матриці

Табл. 3. Власні значення і вектори матриці (11)

Власні значення	Обчислені власні вектори	Нормовані власні вектори на виході програми *Mathematica*
λ_1= *1*	$\boldsymbol{x_1}$ **={-0.447214,0.,0.894427}**	x_1 ={-0.447214,0., 0.894427}
λ_2=*1*	$\boldsymbol{x_{11}}$**={0.596285,-0.745356, 0.298142}**	x_{11}={-0.707107,0.707107,0.}
λ_3=*10*	$\boldsymbol{x_2}$***={0.666667,0.666667, 0.333333}***	x_2={0.666667, 0.666667, 0.333333}

Можна переконатися, що перетворена програмою "Vector1" (рис.2), матриця

$$A_1=\begin{bmatrix} 4.8 & 4 & 2.4 \\ 4 & 5 & 2 \\ 2.4 & 2 & 1.2 \end{bmatrix}$$

має прості дійсні власні значення{10.,1.,-2.01249□ 10^8}

Як свідчить табл.3, значення власного вектора x_{11}, обчисленого запропоновую процедурою і програмою *Mathematica* , відрізняються, хоч дві множини власних векторів створюють *ортогональну систему векторів,* в чому неважко переконатися. Наприклад, для обчислених значень

$x_2.x_1$= -7.80547□ 10^7, $x_{11}.x_1$= 4.44523□ 10^{18} і $x_2.x_{11}$= -1.49071□ 10^7.

Крім того, перевірка показує, що значення λ_2=*1* і власний вектор x_{11} задовольняють рівняння (1): $A.x_{11}-\lambda_2*x_{11}$={-2.22045x$10^{-16}$, -2.22045x$10^{-16}$,-1.11022x$10^{-16}$}

Це свідчить, що для обчислення x_{11} задіяні різні процедури і що множена власних векторів матриці необмежена.

Якщо кратність власного значення матриці *k>2*, то виконуються *k-1* ітерацій, на кожній з яких застосовуються послідовно програми "Vector" (рис.1) і "Vector1" (рис.2) з проміжним формуванням матриці, кратність власного значення якою зменшується на одиницю.

3.3. Власні вектори для комплексних власних значень

Матриця *A*, як відомо, може мати комплексоспряжені власні значення. Оскільки матриця А має дійсні елементи, то в цьому випадку згідно тотожності (2) власні вектори матриці повинні бути теж комплексними. Принципово для їх обчислення може безпосередньо вико ристов.-уватися програма "Vector", а у випадку їх кратності – також програма "Vector1".

Приклад 4. Обчислимо власні вектори матриці

$$A=\begin{bmatrix} 2.27107 & -0.271061 & -1.72893 \\ -1.31412 & 0.685878 & -1.31412 \\ -1.72894 & 0.271061 & 2.27106 \end{bmatrix}, \quad (12)$$

яка має власні значення $\boldsymbol{\lambda_1=4,\lambda_{2,3}}$ **= *0.613999 ±0.84098* □.**

Послідовно знаходимо нормовані власні вектори x_1,x_2 і z_3 для кожного з наведених трьох власних значень, користуючись приведеною вище програмою "Vector", на вхід якої подаються, наприклад, дані:

A={{2.27106,0.271056,-1.72894},
{-1.31412,0.685878,-1.31412},
{-1.72894,0.271056,2.27106}};

lamda= 0.613999+0.84098 □;
b={0,0,0}
Отримані результати зведені у табл.4.

Табл. 4. Нормовані значення власних векторів матриці (12)

	Обчислені значення за пропонованою процедурою	*Значення, що визначаються програмою Mathematica*
x_1	***{-0.707107,0.,0.707107}***	*{0.707107, 0.,-0.707107}*
x_2	***{0.292401,0.0775337 +0.907189 i, 0.292401}***	*{0.0249009 - 0.291339 i, 0.910496, 0.0249009 - 0.291339 i}*
x_3	***{0.292401, 0.0775337 -0.907189 i, 0.292401}***	*{0.0249009 + 0.291339 i,0.910496, 0.0249009 + 0.291339 i}*

Література

1. Фельдман Л.П. Чисельні методи / Фельдман Л.П., Петренко А.І., Дмитрієва О. А.- Київ, ВНУ, 2006. – 600 с.
2. Бахвалов Н.С. Численные методы. -М.: Наука, 1978, – Т.1. – 632 с.
3. Березин И., Жидков Н.Л. Методы вычислений. - М.: Физматгиз, 1962-1966, - Т.1-464 с.
4. Воеводин В.В. Вычислительные основы линейной алгебры. - М.:Наука,1977. - 303с.
5. Петренко О.О. // Системный анализ и информационные технологии: 12-я международная научно-техническая конференция "САИТ-2010", 25-29 мая 2010, Киев, Украина : материалы. – К. : УНК "ИПСА" НТУУ "КПИ", 2010. – С. 140.
6. http://www.arndt-bruenner.de/mathe/scripts/engl_eigenwert.htm

УДК 519.854.2

ПАВЛОВ А.А.
МИСЮРА Е.Б.
ЛИСЕЦКИЙ Т.Н.

ОБЪЕДИНЕНИЕ РАБОТ В ГРУППЫ С УЧЕТОМ ИХ ПРИОРИТЕТОВ, ГОТОВНОСТИ К ВЫПОЛНЕНИЮ И ДИРЕКТИВНЫХ СРОКОВ

Розглядається алгоритм об'єднання робіт у групи при плануванні складних систем. У результаті об'єднання робіт істотно підвищується ефективність функціонування систем за рахунок більш дешевого або більш швидкого виконання робіт групами. Алгоритм розроблений для випадку, коли тривалість налагодження приладу істотно перевищує максимальну із тривалостей виконуваних робіт.

We consider the one machine scheduling problem of minimizing the total earliness and tardiness of independent tasks against due dates (ET) with setups. We propose a heuristic algorithm to search a local optimal solution of the problem. Experimental studies have shown the effectiveness of the algorithm, which allows to solve large dimensional problems for a short time.

Введение

В последнее время наблюдается существенный интерес к составлению расписаний работ, использующему объединение работ в группы или семейства [1], т.к. в результате объединения работ существенно повышается эффективность функционирования систем за счет более дешевого или более быстрого выполнения работ группами. Например, уменьшается время переналадки, которое может превышать длительность выполнения самих работ. В химической промышленности объединение подготовительных работ может существенно сократить стоимость за счет более рационального использования дорогостоящих компонентов.

Представленный алгоритм входит в состав трехуровневой модели планирования функционирования сложных систем, имеющих сетевое представление технологических процессов и ограниченные ресурсы, описанной в [2]. На первом уровне модели на основе детальной информации, связанной с заданиями (комплексами взаимосвязанных работ, выполняемых в системе), ресурсами и технологией производства, строится агрегированная модель с помощью объединения отдельных ресурсов и операций в бóльшие единицы. Агрегация осуществляется до уровня мультиресурсов (устойчивых групп совместно работающих ресурсов, например, бригад, групп однотипного оборудования, однопрофильных подразделений), и агрегированных работ (совокупностей работ, выполняемых в одном мультиресурсе в рамках одного захода в мультиресурс по одному заданию). На основе агрегированной информации для каждого задания строится агрегированный граф, определяющих взаимосвязи и порядок выполнения агрегированных работ. В свою очередь, для каждого построенного агрегированного графа строится критический путь, определяющий минимальное время выполнения комплекса агрегированных работ, составлящих задание. Длительность выполнения задания и агрегированных работ, входящих в его состав, определяется критическим путем.

Завершающим этапом агрегации является построение агрегированной модели, в которой задача планирования сводится к задаче календарного планирования для одного прибора. На основе критических путей заданий строится граф, называемый далее графом на критических путях. Вершины полученного ориентированного ацикличного графа – агрегированные работы, лежащие на критических путях заданий. Дуги отражают связи между агрегированными работами, которые выполняются в соответствующих мультиресурсах, регламентирующие технологию выполнения заданий. Конечные вершины соответствуют выполненным заданиям.

Некоторые агрегированные работы, принадлежашие критическим путям различных заданий и выполняемые в одном мультиресурсе, объединяются в общие агрегированные работы так, чтобы не требовалась наладка для работы, если она принадлежит той же вершине, что и агрегированная работа, выполненная перед ней. На графе связности это отображено общими вершинами. При объединении агрегированных работ в «общие вершины» исключается время переналадки для объединяемых работ, которое в отдельных случаях может существенно превышать длительность выполнения работ. Если объединение в «общие вершины» не реализует-

ся, то при выполнении каждой агрегированной работы необходимо учитывать время переналадки, что существенно увеличит время прохождения заданий в системе.

Веса всех вершин построенного графа на критических путях заданий равны нулю, веса конечных вершин равны весам выполненных заданий. Длительность выполнения «общей вершины» равна сумме длительностей выполнения входящих в ее состав агрегированных работ.

Для определения очередности назначения заданий на выполнение на построенном графе на критических путях заданий решается задача минимизации суммарного взвешенного момента окончания выполнения работ одним прибором (МВМ) при условии, что веса всех вершин, кроме конечных, равны нулю. Результат решения этой задачи – приоритетно-упорядоченная последовательность выполнения агрегированных работ, входящих в состав заданий, которая является основной информацией для второго уровня модели – согласованного планирования. В этой последовательности в первую очередь выполняются задания высшего приоритета, что обеспечивает получение эффективного решения задачи по рассматриваемым в системе критериям [2].

В данный статье рассматривается разработанный на основе требований к формированию «общих вершин» и правил их построения [2] алгоритм объединения агрегированных работ в «общие вершины» при построении графа на критических путях заданий.

Алгоритм построен на основе информации о приоритетах заданий, готовности агрегированных работ к выполнению, длительностях переналадок и директивных сроках и состоит из двух этапов. На первом этапе определяются группы агрегированных работ, претендующих на объединение в «общие вершины». На втором этапе производится построение «общих вершин» в соответствии с правилами, изложенными в [2].

Алгоритм определения претендентов для объединения в «общие вершины»

5. Упорядочиваем мультиресурсы по невозрастанию значений их времен наладок (последовательность σ^1).
6. Упорядочиваем критические пути заданий по невозрастанию их приоритетов (последовательность σ^2).
7. Выбираем из последовательности σ^1 мультиресурс i_s, требующий максимального времени наладки при переходе от одного типа работ к другому.
8. Просматриваем последовательность σ^2, начиная с критического пути задания наивысшего приоритета. Находим первую агрегированную работу j_r, требующую выполнения в мультиресурсе i_s. Пусть она находится на критическом пути задания J_p. Формируем группу претендентов на объединение с работой j_r следующим образом. В последовательности σ^2 на интервале $[p+1, n]$ находим следующую агрегированную работу j_u, требующую выполнения в мультиресурсе i_s. Пусть она находится на критическом пути задания J_t. Проверяем условия:

$$\Omega_{J_t}\sum_{q=p+1}^{t-1} L_{J_q} + l_{i_s}^{\mathrm{H}}\sum_{q=t+1}^{n}\Omega_{J_q} > \left(L_{J_t} - l_{i_s}^{\mathrm{H}}\right)\sum_{q=p+1}^{t-1}\Omega_{J_q}, \quad (1)$$

$$\frac{\Omega_{J_p}}{L_{J_p}} - \frac{\Omega_{J_p}+\Omega_{J_t}}{L_{J_p}+L_{J_t}-l_{i_s}^{\mathrm{H}}} \le \Delta,$$

где Ω_{J_q} – вес задания J_q; L_{J_q} – длительность его критического пути; $l_{i_s}^{\mathrm{H}}$ – длительность наладки в мультиресурсе i_s; $\Delta \approx 8$–10% от максимального приоритета. В левой части неравенства (1) отражено уменьшение значения функционала МВМ в случае встраивания задания J_t (агрегированных работ, принадлежащих его критическому пути) после задания J_p в последовательности σ^2 и в результате уменьшения числа переналадок, а в правой части – увеличение значения функционала в результате смещения заданий на интервале $\overline{p+1, t-1}$ на более поздние позиции. В соответствии с (2), приоритет «общей вершины» не должен быть меньше приоритета задания J_p более, чем на величину Δ. Если не выполняется объединение, то длительность агрегированной работы включает время переналадки $l_{i_s}^{\mathrm{H}}$. Если условия (1) и (2) выполняются, j_u включается в группу претендентов на объединение с работой j_r. Аналогичным образом находим всех претендентов на объединение с работой j_r, просматривая критические пути заданий $\overline{p+1, n}$.
9. Переходим к следующему мультиресурсу в последовательности σ^1, переход на шаг 3. Если все мультиресурсы рассмотрены, конец алгоритма.

Формирование «общих вершин» на основе групп претендентов

Приведенный выше алгоритм разработан для случая, когда длительность наладки прибора существенно превышает максимальную из длительностей выполняемых работ. Поэтому «общие вершины» на графе критических путей заданий формируются на основе Правил 1–2, изложенных в [2]. Правило 1 используется в случае отсутствия директивных сроков заданий, Правило 2 – при их наличии.

Правило 1. В случае отсутствия директивных сроков заданий объединение агрегированных работ выполняется при выполнении условий (1) и (2), а также если разность моментов готовности агрегированных работ к выполнению (длительностей путей от начала критических путей заданий, которым принадлежат рассматриваемые агрегированные работы, до этих работ) не превышает длительности переналадки в мультиресурсе. Если критические пути некоторых заданий имеют несколько «общих вершин», то объединение агрегированных работ выполняется при выполнении условий (1) и (2), а также если разность моментов готовности агрегированных работ к выполнению с минимальным временем начала выполнения не превышает суммарного времени переналадок для объединяемых вершин на этом интервале.

Правило 2. В случае наличия директивных сроков агрегированные работы объединяются в «общие вершины» при выполнении условий (1) и (2). Объединение выполняется, если их директивные сроки (определенные как директивный срок задания минус длительности выполнения агрегированных работ, следующих за назначаемой работой по критическому пути до конечной вершины) равны или отличаются не более, чем на величину $\Delta \approx 5–7\%$ от меньшего директивного срока, определяемую на основе экспериментальных исследований.

Выводы

Предложен алгоритм формирования «общих вершин» на критических путях выполнения заданий с учетом их приоритетов, моментов готовности агрегированных работ к выполнению и директивных сроков. Эффективность предложенного алгоритма подтверждается результатами просчета ряда примеров построения календарных планов выполнения заданий в компьютерной системе на основе реальных данных практической производственной размерности.

Список литературы

6. Chris N. Potts, Mikhail Y. Kovalyov. Scheduling with batching: A review // European Journal of Operational Research.– 2000, Vol.120.– P.228-249.
7. Згуровский М.З., Павлов А.А. Принятие решений в сетевых системах с ограниченными ресурсами: Монография.– К.: Наукова думка, – 2010.– 573 с.

Відомості про авторів

Абу Усбах О.Н.	к.т.н., доцент кафедри ОТ НТУУ «КПІ»
Альмураді Ваджих	студент кафедри ОТ НТУУ «КПІ»
Амонс А.А.	к.т.н., доцент кафедри АУТС НТУУ «КПІ»
Ашаєв Ю.М.	студент кафедри ОТ НТУУ «КПІ»
Болдак А.А.	к.т.н, доцент кафедры ВТ НТУУ «КПИ»
Буй Хиу Дат	студент НТУУ «КПІ»
Ващук Ф.Г.	д.т.н., проф., Закарпатський державний університет
Виноградов Ю.Н.	старший викладач кафедри ОТ НТУУ ”КПІ”
Вітіщенко Н.С.	студентка кафедри АУТС НТУУ «КПІ»
Вовк В.М.	магистр кафедри АУТС НТУУ «КПІ»
Грегуль В.В.	студент кафедри АСОІУ НТУУ «КПІ»
Грибенко Д.В.	студент кафедри ОТ НТУУ «КПІ»
Гузий Н.Н.	к.т.н., доцент НАУ
Демчинський В.В.	к.т.н., доцент кафедри ІБ ФТІ НТУУ «КПІ»
Дорогий Я.Ю.	асистент кафедри АУТС НТУУ «КПІ»
Дорошенко К.С.	асистент кафедри АУТС НТУУ «КПІ»
Єфремов К.В.	директор Світового центру даних НТУУ “КПІ”
Зайченко Ю.П.	д.т.н., проф. кафедры ММСА НТУУ «КПИ» УНК «ИПСА»
Зайцев С.Ю.	студент кафедри АУТС НТУУ «КПІ»
Захаров Д.С.	студент НТУУ «КПІ»
Иванов А.Н.	студент кафедри обчислювальної техніки НТУУ ”КПІ”
Игнатов В.А.	д.т.н., проф. НАУ
Киричек А.А.	студент кафедри АУТС НТУУ «КПІ»
Костик Д.Ю.	аспирант кафедры АСОИУ НТУУ «КПИ»
Кунщиков Е.О.	студент кафедри АУТС НТУУ «КПІ»
Куренёв А.С.	студент кафедри ОТ НТУУ «КПІ»
Лисецкий Т.Н	аспирант кафедры АСОИУ НТУУ "КПИ"
Мазур Р.Ф.	студент кафедри ОТ НТУУ “КПІ”
Марковський О.П.	к.т.н., доцент кафедри ОТ НТУУ «КПІ»
Мельник О.О.	ст. викл. Закарпатського державного університету
Мисюра Е.Б.	к.т.н, с.н.с. кафедры АСОИУ НТУУ «КПИ»
Мистецький В.А.	студент НТУУ «КПІ»
Можаровский П.Ф.	магістр НТУУ «КПІ»
Молчановський О.І.	викладач НТУУ «КПІ»
Мурга Н.А.	аспірант кафедри ММСА ННК «ІПСА »НТУУ «КПІ»
Назарук О.В.	студент кафедри АСОІУ НТУУ «КПІ»
Нгіем Ле Куан	студент НТУУ «КПІ»
Ови Нафас Агаи Аг Гамиш	аспірант кафедри ММСА ННК «ІПСА »НТУУ «КПІ»
Островский С.М.	студент кафедри АУТС НТУУ «КПІ»
Павлов О.А.	проф. кафедры АСОИУ НТУУ «КПИ», декан ФИВТ
Петренко О.О.	студент кафедри СП ННК «ІПСА »НТУУ «КПІ»
Полторак В.П.	к.т.н., доцент кафедри АУТС НТУУ «КПІ»
Порєв В.М.	к.т.н., ст. викладач кафедри ОТ НТУУ «КПІ»
Пустоваров В.І.	к.т.н., доцент кафедри ОТ НТУУ ”КПІ”
Пустовит М.А.	студент кафедри ОТ НТУУ «КПІ»
Ролік О.І.	к.т.н., доцент, ст. науквий співробітник НТУУ «КПІ»
Симоненко А.В.	ст. викл. кафедри ОТ НТУУ ”КПІ”
Симоненко В.П.	д.т.н., проф. кафедри ОТ НТУУ ”КПІ”
Сорая М.А.	викладач НАУ
Стеценко И.В.	к.т.н., доцент кафедры компьютерных технологий ЧГТУ

Стіренко С.Г.	к.т.н., доцент кафедри ОТ НТУУ ”КПІ”
Сухарев Д.Л	студент кафедри ОТ НТУУ «КПІ»
Тимошенко Я.Я.	студент кафедри АСОІУ НТУУ «КПІ»
Томашевський В.М.	д.т.н., проф. кафедри АСОІУ НТУУ «КПІ»
Ульяницкая К.А.	студентка кафедри АУТС НТУУ «КПІ»
Федоречко О.И.	інженер кафедри ОТ НТУУ ”КПІ”
Шаршаков А.С.	студент кафедри ОТ НТУУ «КПІ»
Шило С.О.	к.ф-м.н., с.н.с. ІПРІ НАН України
Шиховець О.В.	м.н.с. інституту проблем реєстрації інформації НАН України
Юрлов В.І.	к.т.н., провідний науковий співробітник компанії Самсунг Електро- Механікс, Південна Корея
Янчевский С.Л.	соискатель ИКД НАНУ-НКАУ
Яцишин А.Ю.	аспірант кафедри АСОІУ НТУУ «КПІ»