ВІСНИК

НАЦІОНАЛЬНОГО ТЕХНІЧНОГО УНІВЕРСИТЕТУ УКРАЇНИ
"Київський політехнічний інститут"

Інформатика, управління та обчислювальна техніка

Кафедра обчислювальної техніки 50 років

52

2010

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 52

Київ “ВЕК+” 2010

УДК 004

Вісник НТУУ «КПІ». Інформатика, управління та обчислювальна техніка: Зб. наук. пр. – К.: Век+, – 2010. – № 52. – 144 с.

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

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

Заст. гол. ред.: Стіренко С.Г., к.т.н., доц.,

Пустоваров В.І., к.т.н., доц.

Відповідальний секретар: Поспішний О.С.

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

Описано результати дослідження і створення компонентів обчислювальних й інформаційних систем і комплексів, пристроїв автоматики та передавання 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 23.12.2010. Формат 60×84 1/16. Гарнітура Times. Папір офсетний №1. Наклад 150 прим.
Надруковано в ЗАТ “ВІПОЛ”, 03151, г. Київ, вул. Волинська, 60.

© Національний технічний університет України “КПІ”, 2010

© Національний технічний університет України “КПІ”, 2010

ЗМІСТ

Павлов О.А., Місюра О.Б., Мельников О.В., Аракелян Г.А., Щербатенко О.В., Михайлов В.В., Лисецький Т.Н. Інформаційна технологія ієрархічного планування та прийняття рішень в організаційно-виробничих системах …….....3
Луцкий Г.М., Волокита А.Н., Иванов Д.Г. Метод организации Защищенных файловых систем на основе криптографических примитивов контроля доступа……...…….....15
Кулаков О. Ю., Бролінський С.М., Ашаєв Ю.М. Система моделювання роботи GRID систем з застосуванням ієрархічного планувальника та методу віртуальних кластерів…….....22
Карпінський В.М., Євтух П.С., Кінах Я.І. Теоретико-графовий підхід до моделювання Розподілених безпровідних сенсорних мереж……....27
Роковой А.П. Анализ эффективности VOIP кодеков в глобальной сети……33
Ролик А.И., Волошин А.В., Галушко Д.О., Можаровський П.Ф., Покотило О.О. Система управления корпоративной информационно-телекоммуникационной инфраструктурой на основе агентского похода……...39
Писаренко А.В., Цвєлодуб А.О. Розробка інструментальних засобів matlab/simulink для проектування керуючих пристроїв систем управління на базі ПЛІС…….....54
Луцький Г.М., Блінова Т.О., Порєв В.М. Використання деяких способів кодування для підвищення точності обчислень……....60
Зайченко Е.Ю., Зайченко Ю.П., Лавринчук А.Н. Инструментальный комплекс алгоритмов и программ оптимального проектирования сетей с технологией MPLS……..65
Мурга Н.А. Применение нечётких нейронных сетей к задаче анализа поведений котировок валют……..72
Костик Д.Ю. Оптимізація програмної реалізації модифікованого ПДС-алгоритму задачі мінімізації сумарного запізнення виконання завдань……..80
Салапатов В.І. Методи оптимізації програм на рівні машинних кодів…..……85
Сергиенко А.М., Лепеха В.Л. Специализированный контроллер локальной сети.……89
Сергиенко А.М., Лесик Т.М. Реализация перестраиваемых рекурсивных цифровых фильтров на ПЛИС……..93
Ладогубець В.В., Фіногенов О.Д., Безносик О.Ю. Особливості оптимізації макромоделей механічних компонентів……..97
Бузовский О.В., Болдак А.А., Невдащенко М.В. Семантика моделей и размерность задач проектирования……104
Стіренко С.Г., Зіненко О.І. Зберігання розріджених матриць великих розмірностей в системі з локальною пам'яттю……112
Томашевський В.М., Новіков Ю.Л., Камінська П.А. Складання розкладів занять у дистанційних системах навчання……...119
Лактіонов Є.Ю., Малицький А.Ю. Використання XML для підготовки спеціалізованої програмної документації……..132
Вельма А.М., Лактіонов Є.Ю. Вибір системи відстеження помилок в залежності від конфігурації програмного забезпечення……....138

УДК 519.854.2

ПАВЛОВ О. А.,
МІСЮРА О. Б.,
МЕЛЬНИКОВ О. В.,
АРАКЕЛЯН Г. А.,
ЩЕРБАТЕНКО О. В.,
МИХАЙЛОВ В. В.,
ЛИСЕЦЬКИЙ Т.Н.

ІНФОРМАЦІЙНА ТЕХНОЛОГІЯ ІЄРАРХІЧНОГО ПЛАНУВАННЯ ТА ПРИЙНЯТТЯ РІШЕНЬ В ОРГАНІЗАЦІЙНО-ВИРОБНИЧИХ СИСТЕМАХ

Наведено короткий опис комплексної системи оперативного планування та прийняття рішень у складних організаційно-виробничих системах та схема розв'язання задачі. Детальний приклад розв'язання задачі планування у системі для випадку дискретного виробництва показує усі переваги використання системи для кінцевого користувача. У висновках наведено галузі застосування системи та економічний ефект від її впровадження.

A short description of the complex system of operational planning and decision making in complex organizational systems is provided together with the scheme for solving the problem. A detailed example of solving the planning problem in the system for discrete production case shows all the advantages for end user. The conclusions contain the system application areas and the economic effect of its implementation.

Загальна характеристика системи

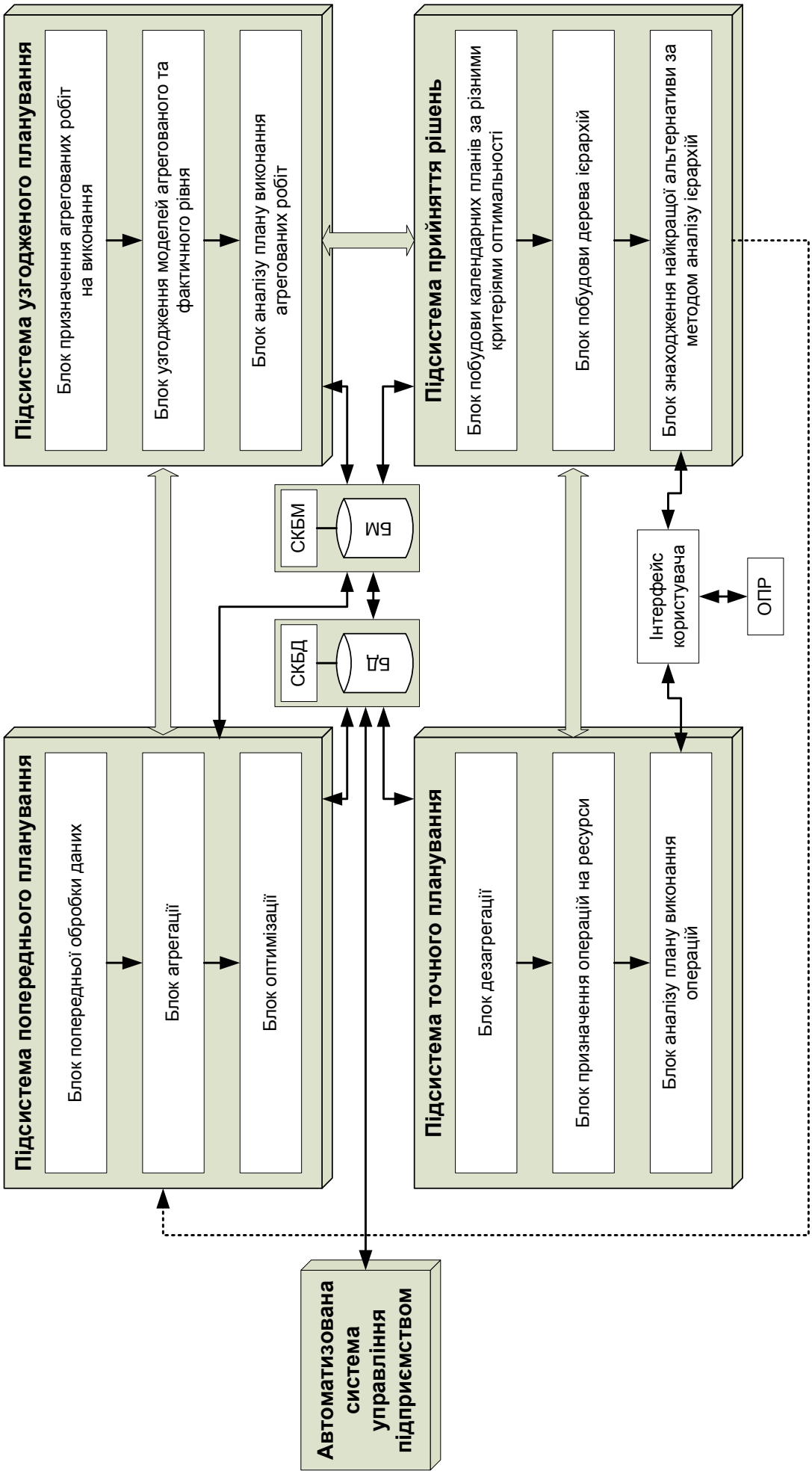
На підставі проведених теоретичних досліджень та розроблених ПДС-алгоритмів розв'язання важкорозв'язуваних задач комбінаторної оптимізації [1] авторами створена комплексна система оперативного планування та прийняття рішень (СОППР). Вказана система є пакетом програм, призначених для розв'язання задач оперативного планування у складних організаційно-виробничих системах (ОВС) та вибору найкращого плану із побудованих за 7 критеріями оптимальності [1].

В основу математичного забезпечення системи покладена трирівнева модель планування, функціональна схема якої представлена в [2], та розроблена система взаємозв'язаних моделей та методів, яка дозволяє враховувати складність сучасного виробництва. На відміну від існуючих методів планування, кращі з яких містять лінійну чи випадкову комбінацію різних правил переваги, що не гарантує якості отриманих розв'язків, в процесі розв'язання задачі планування визначається стратегія пошуку глобального оптимуму, що дозволяє отримати розв'язки, близькі до оптимальних. В основі моделі лежить розв'язання задачі мінімізації сумарного зваженого моменту закінчення виконання завдань (МЗМ), що визначає пріоритети виконання завдань та дозволяє мінімізувати час їх проходження у виробничому циклі.

Система СОППР має модульну структуру та може бути без будь-яких суттєвих доробок впроваджуватись як модуль планування та прийняття рішень у складі АСУ ОВС різних галузей народного господарства. Архітектура системи показана на рис. 1. Система включає ряд блоків, об'єднаних у чотири підсистеми. За результатами роботи системи особа, що приймає рішення (ОПР), отримує найкращий з можливих планів виконання завдань за різними критеріями оптимальності. Розв'язання задачі реалізується за блок-схемою, показаною на рис. 2. Схема вибору алгоритму розподілу агрегованих робіт в ієрархічній моделі планування та управління (перший блок II рівня моделі) для кожного з 7 критеріїв оптимальності показана на рис. 3. Ілюстративний приклад використання технології розв'язання задачі планування у дискретному виробництві наведений нижче.

Ілюстративний приклад розв'язання задачі планування

Покажемо на прикладі використання технології розв'язання задач планування на основі ієрархічної моделі планування та управління для дискретного виробництва. У даному прикладі розв'язується задача за критерієм 1 [2] – максимізація прибутку (МЗМ) при відсутності директивних строків.



*БД – база даних; БМ – база моделей; СКБД – система керування базою даних; СКБМ – система керування базою моделей.

Рис. 1. Архітектура комплексної системи оперативного планування та прийняття рішень СОППР*

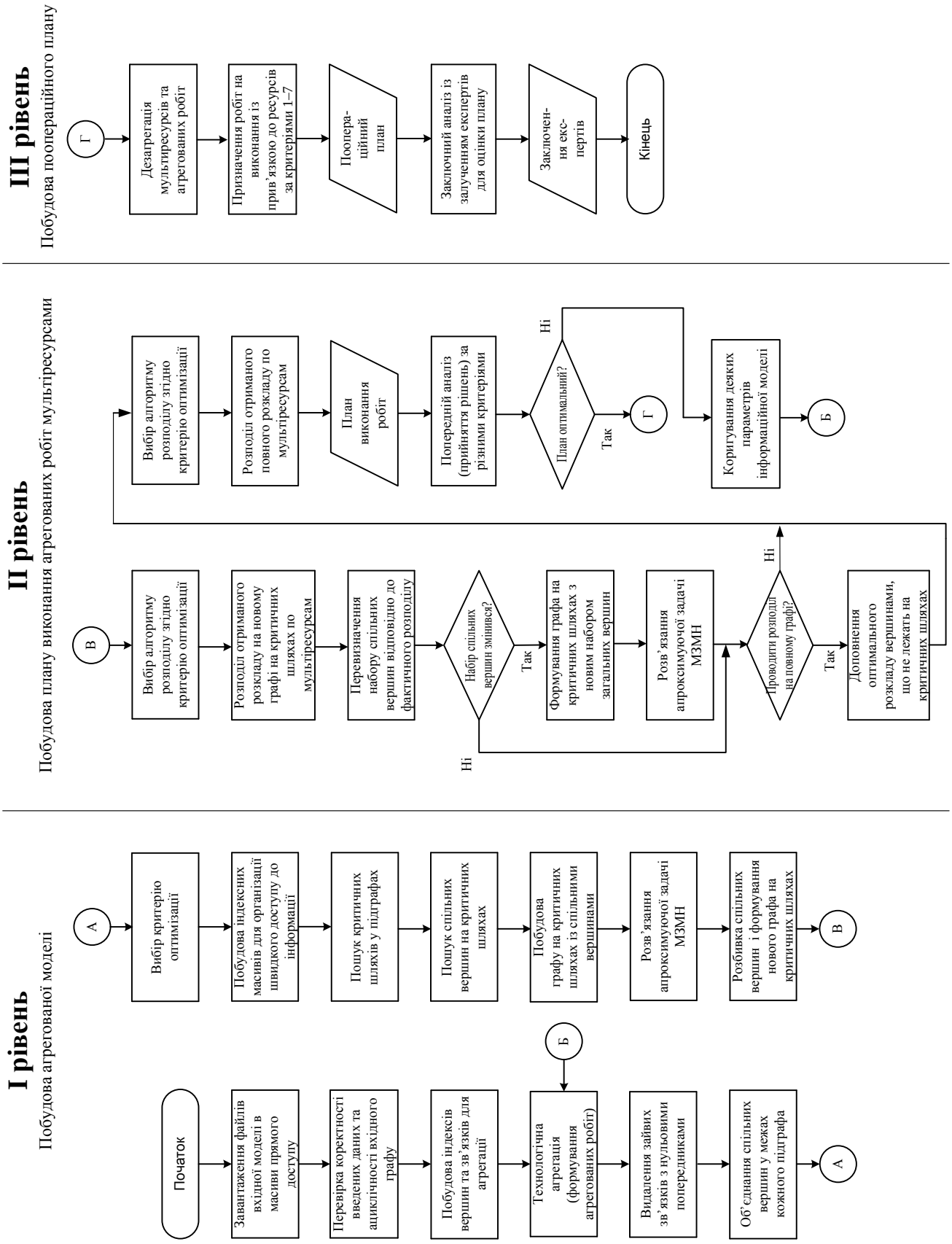


Рис. 2. Блок-схема алгоритму розв'язання задач планування на основі ієрархічної моделі планування та прийняття рішень

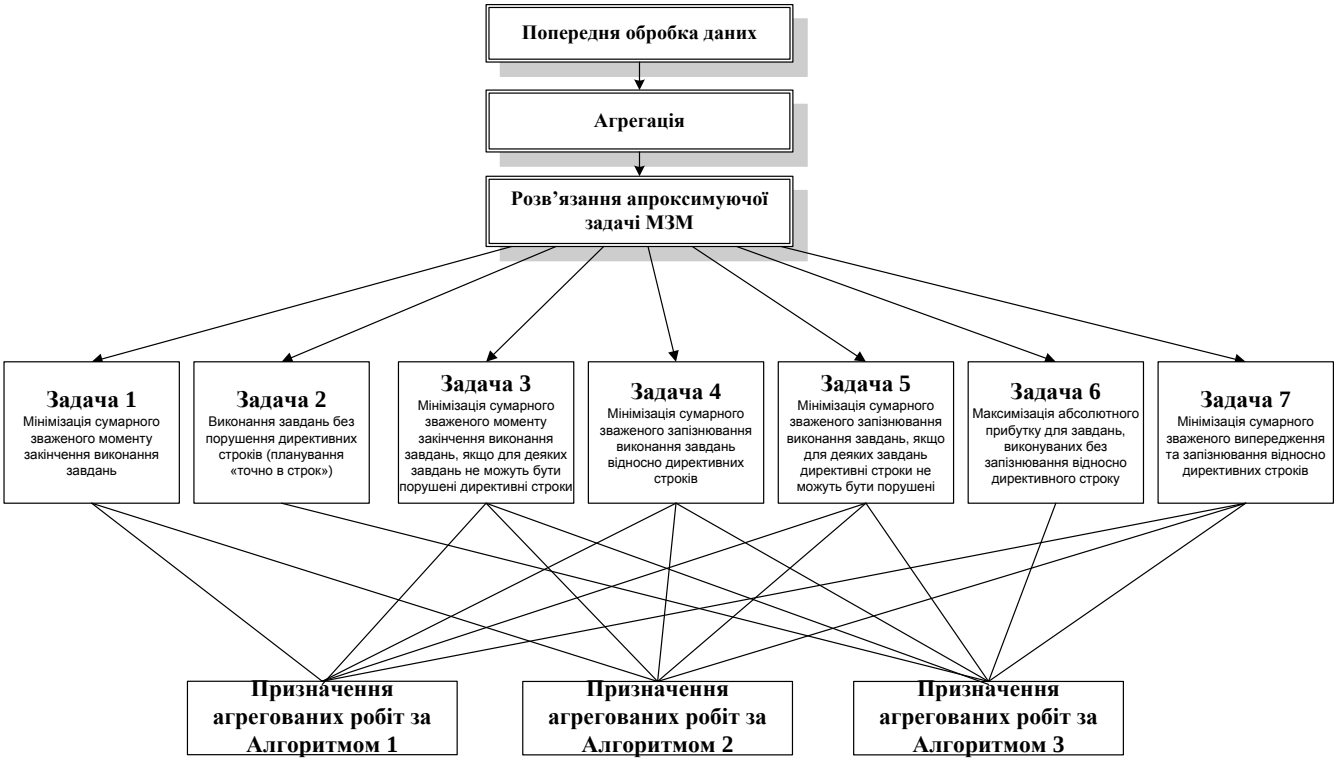


Рис. 3. Схема вибору алгоритму розподілу агрегованих робіт в ієрархічній моделі планування та прийняття рішень в ОВС

Рівень І. Побудова моделі технологічної й конструкторської агрегації

Вхідна модель інформації, як правило, являє собою бази даних з інформацією про завдання (вироби), операції, взаємозв'язки між ними, мультиресурси (комірки), групи комірок і календар робіт (інформація подається із АСУ підприємства). Вхідний ациклічний орієнтований граф операцій представляється у вигляді ланцюжків технологічних операцій (переліку робіт) і конструкторської входимості (переліку взаємозв'язків між операціями).

Вхідні дані: є 3 вироби (серії однотипних виробів), їхні характеристики наведені в табл. 1. Кожний виріб складається з деталей і складань, технологія виготовлення яких описана в табл. 2.

Табл. 1. Вхідні дані: довідник виробів

Виріб	Кількість виробів у серії	Ваги виробів (ω_i), що входять у критерій 1
И1	80	1
И2	90	2
И3	65	3

Табл. 2. Вхідні дані: технологія виконання виробів (деталей, збірок)*

Деталь	Група комірок	Захід	НМ	*l*, год
Д1	5	1	3	0,01
Д1	1-3	1	2	0,15
Д1	1-2	1	1	0,03
Д2	1-1	1	4	0,02
Д2	1-3	2	3	0,01
Д2	4	1	2	0,01
Д2	1-3	1	1	0,03
Д3	1-1	1	3	0,02
Д3	2	1	2	0,02
Д3	1-3	1	1	0,03
Д5	5	1	2	0,1
Д5	2	1	1	0,3
Д6	5	1	4	0,05
Д6	2	1	3	0,02
Д6	3	1	2	0,03
Д6	4	1	1	0,04
Д7	5	1	2	0,05
Д7	2	1	1	0,02
С1	5	1	4	0,05
С1	3	1	3	0,13
С1	1-2	1	2	0,12
С1	1-1	1	1	0,01
С2	1-1	1	1	0,03
С3	5	1	2	0,5
С3	1-1	1	1	0,11
С4	1-1	1	2	0,11
С4	1-3	1	1	0,12
С5	1-3	1	4	0,11
С5	2	1	3	0,02
С5	4	1	2	0,04
С5	5	1	1	0,05
С6	5	2	3	0,05
С6	4	1	2	0,4
С6	5	1	1	0,5
С7	5	1	2	0,5
С7	1-1	1	1	0,11
И1	5	1	1	0,55
И2	5	1	1	0,4
И3	5	1	1	0,5

*НМ – номер у маршруті, *l* – тривалість виконання

Роботи виконуються на множині з 14 комірок, об'єднаних у 7 груп, дані по яких наведені в табл. 3. В окремих комірках устаткування вимагає тривалого переналагодження при виконанні різних видів операцій.

Графи конструкторської входимості виробів наведені на рис. 4–6.

Табл. 3. Вихідні дані: довідник груп комірок*

Група	Цех	Ділянка	Комірки	Робочих місць	$l_{нал}$
1-1	1	1-1; 1-2	1-1-1; 1-1-2	9	1
1-2	1	2	1-2	11	–
1-3	1	3-1; 3-2; 3-3	1-3-1; 1-3-2; 1-3-3	24	–
2	2	1; 2; 3	2-1; 2-2; 2-3	14	2
3	3		3	11	4
4	4	1; 2	4-1; 4-2	18	4
5	5	1; 2	5-1; 5-2	20	5

*Кількість змін дорівнює 3 для всіх груп комірок.

Перетворення вхідного графа в граф коміркокомплектів здійснюється за допомогою процедури первинної агрегації. Агрегація такого технологічного типу сприяє об'єднанню суміжних операцій (по технологічних ланцюжках), які виконуються в однієї і тієї ж комірці по одному виробу, в один елемент, названий коміркокомплектом.

Пунктирною лінією на рис. 4–6 показано вершини, поєднувані в коміркокомплекти. У кружку – код (назва) роботи, поруч із ним – код групи комірок (у дужках – номер заходу). Стрілки позначають входження, числа над стрілками – кількість робіт, які мають потребу у виконанні для передачі попередникові (застосовність).

Тривалість виконання коміркокомплекта визначається критичним шляхом у графі виконання робіт, що входять до нього, час переналагодження для коміркокомплекта дорівнює максимальному із часів переналагодження його перших робіт (листів графа). При цьому тривалість коміркокомплекта *в одному виробі* визначається технологією робіт. Загальна тривалість коміркокомплекта дорівнює сумі тривалостей операцій по всіх виробах у даній комірці по критичному шляху на одне робоче місце, тобто

$$Lop_i = \sum_{j \in CP_i} \frac{Lop_{ij} \cdot Nnp_i \cdot N_i}{KRM_i}, \qquad (1)$$

де L_i – тривалість i-го коміркокомплекта; Nnp_i – кількість операцій у виробі (застосовність); Lop_{ij} – тривалість j-ї операції i-го коміркокомплекта; N_i – кількість виробів, у які входить i-й коміркокомплект; KRM_i – кількість робочих місць у всіх комірках групи (з розрахунком на паралельне виконання в однотипних комірках); CP_i – критичний шлях операцій коміркокомплекта. Величину $Lop_i = \max\left[\sum_{j \in CP_i} Lop_{ij}, 1\right]$ назвемо тривалістю однієї *агрегованої операції*. Тоді $Nop_i = \frac{Nnp_i \cdot N_i}{KRM_i}$ – кількість агрегованих операцій у коміркокомплекті.

Отже, $L_i = Lop_i\, Nop_i$. З наведеної формули видно, що фактично тривалість коміркокомплекта виявляється кратної тривалості однієї агрегованої операції Lop_i, що визначається як сума тривалостей операцій по критичному шляху коміркокомплекта. Агрегована операція – найменша частина коміркокомплекта, що не допускає дроблення на більш дрібні частини.

Після одержання первинних коміркокомплектів методом технологічного об'єднання мо-

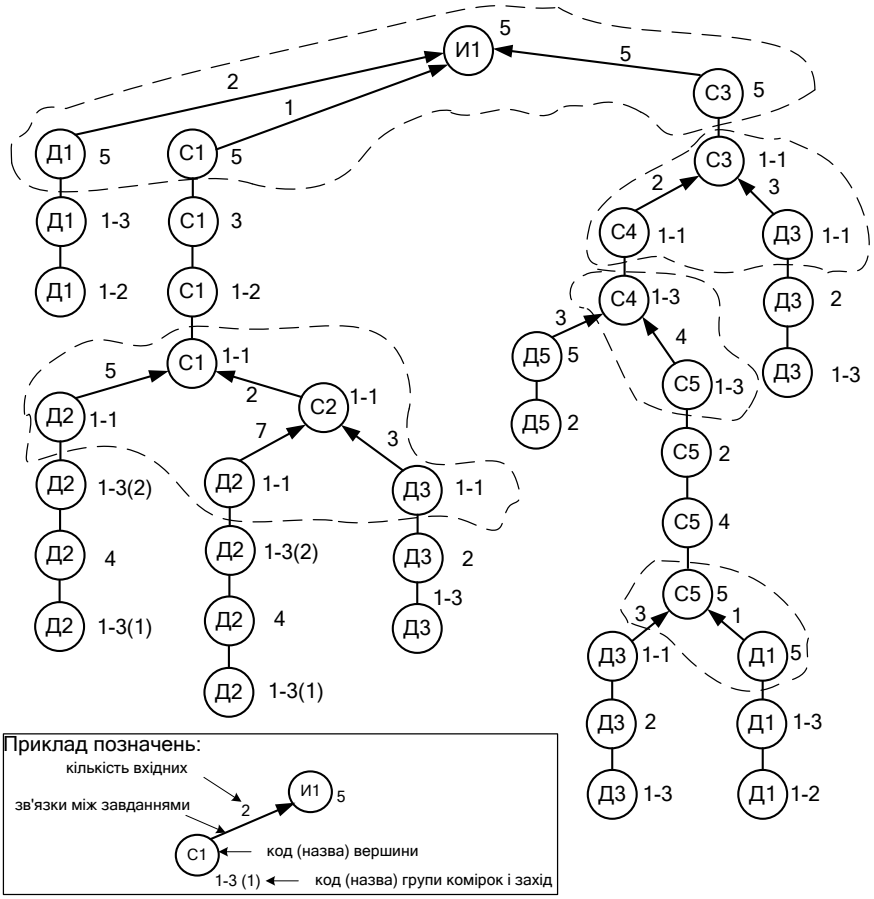


Рис. 4. Граф конструкторської входимості: виріб И1

же здійснюватися конструкторське об'єднання множина коміркокомплектів і нові графи їхнього взаємозв'язку для кожного виробу (рис. 7, табл. 4). На рис. 7 числа ліворуч від кружків позначають номери агрегованих робіт після перенумерації.

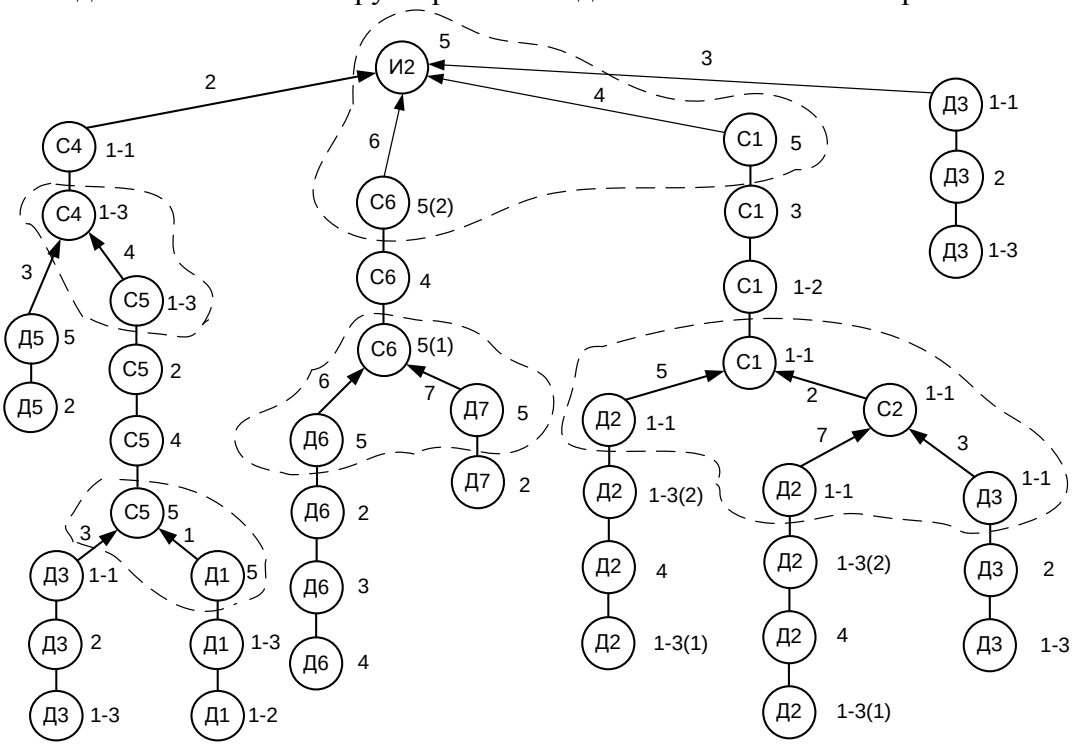


Рис. 5. Граф конструкторської входимості: виріб И2

коміркокомплектів, що мають однакові коди вершин і комірок в одному виробі, час запуску яких по критичному шляху виробу різниться не більш, ніж на максимальне значення із часів їхнього переналагодження. При цьому коміркокомплекти утворять один комплект із сумарною тривалістю й часом переналагодження, що дорівнює максимальному із всіх часів їхнього переналагодження.

Конструкторська агрегація виконується рідко, тому що вона звичайно приводить до великих тривалостей агрегованих робіт, що погано впливає на результат розподілу. Однак об'єднання робіт у спільні вершини, розглянуте далі, також є методом конструкторської агрегації.

У результаті виконання процедури технологічної агрегації отримана

Рівень II. Побудова плану виконання коміркокомплектів в комірках

Далі здійснюється побудова спеціальних індексних масивів для швидкого доступу до інформації, необхідної для виконання наступних блоків. При індексації визначаються значення окремих реквізитів моделі відповідно до критерію, обраного користувачем. На сформованій у результаті перенумерації моделі інформації здійснюється пошук критичних шляхів виробів. Критичний шлях у графі, де кожна

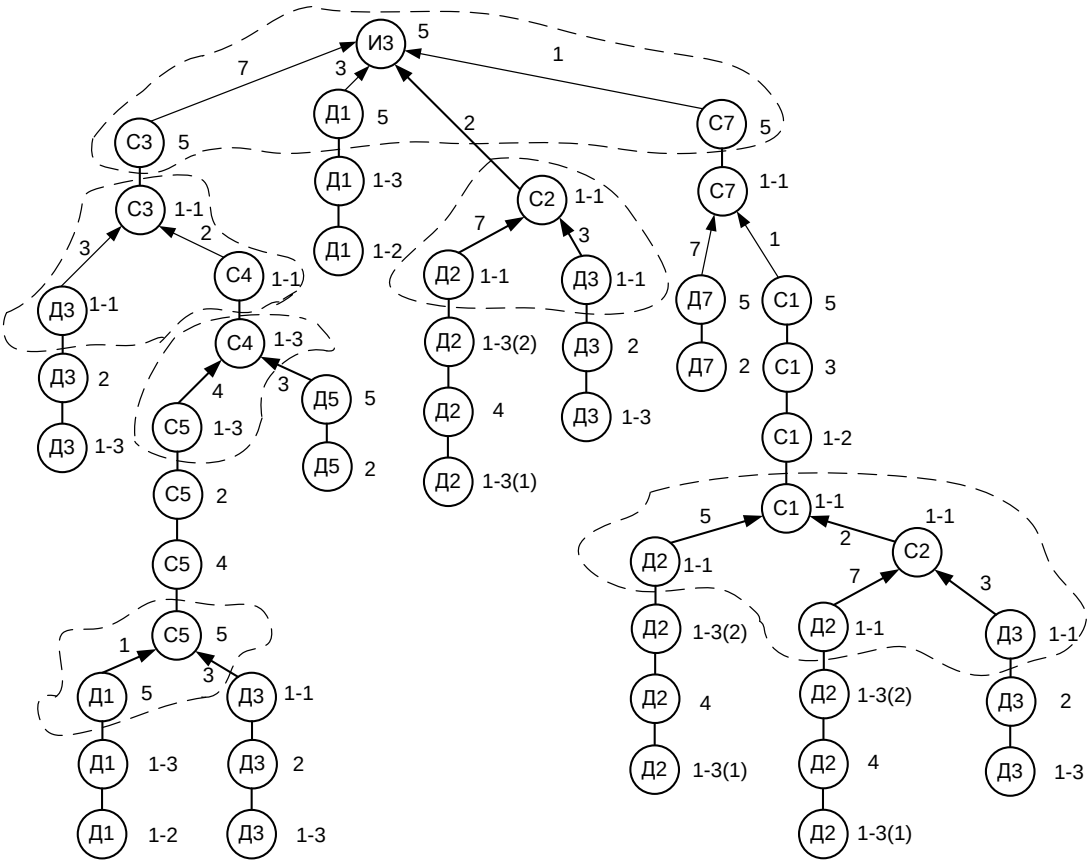
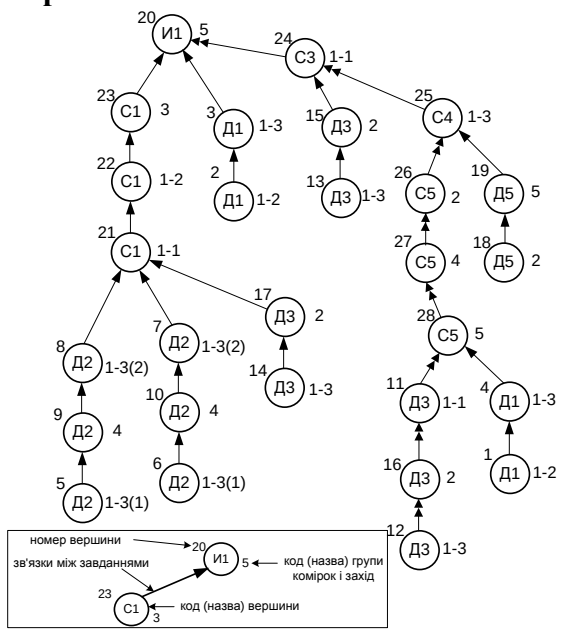


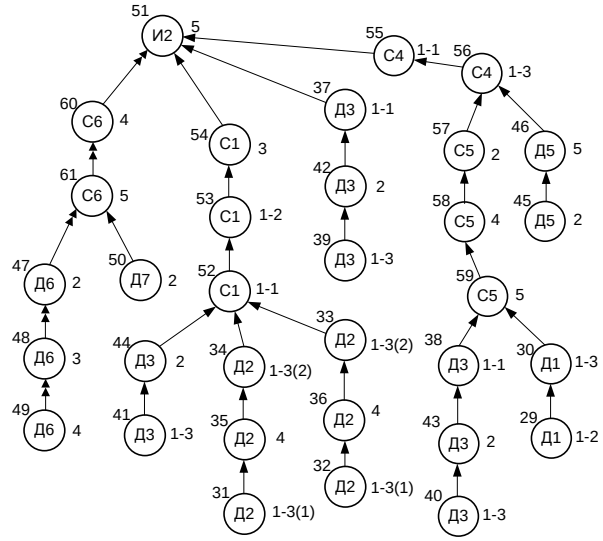
Рис. 6. Граф конструкторської входимості: виріб ИЗ

вершина навантажена тривалістю, – це шлях з максимальною тривалістю. Критичні шляхи шукають за допомогою спрямованого повного перебору вершин графа коміркокомплектів. Для пошуку критичного шляху перед його виконанням перевіряється ацикличність побудованого графа. Використовувані процедури пошуку критичного шляху у графі детально розглянуто у [2]. На графах взаємозв'язку коміркокомплектів (див. рис. 7) критичні шляхи відзначені лініями з подвійною стрілкою.

Виріб И1



Виріб И2



Виріб И3

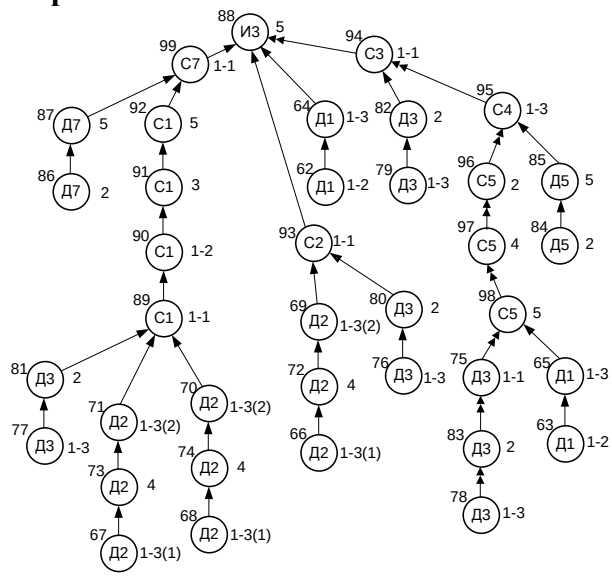


Рис. 7. Графи коміркокомплектів, отримані після технологічної агрегації, і критичні шляхи виробів (подвійні стрілки)

Табл. 1. Агреговані тривалості коміркокомплектів із тривалостями переналагоджень в комірках

i	l_i	*i*	l_i	*i*	l_i
1	8,727	34	8,727	67	8,727
2	0,436	35	0,436	68	0,436
3	1,000	36	1,000	69	1,000
4	20,000	37	20,000	70	20,000
5	1,400	38	1,400	71	1,400
6	0,500	39	0,500	72	0,500
7	0,167	40	0,167	73	0,167
8	0,467	41	0,467	74	0,467
9	0,622	42	0,622	**75**	**0,622**
10	0,222	43	0,222	76	0,222
11	**21,333**	44	21,333	77	21,333
12	**12,000**	45	12,000	**78**	**12,000**
13	1,500	46	1,500	79	1,500
14	0,600	**47**	**0,600**	80	0,600
15	1,714	**48**	**1,714**	81	1,714
16	***13,714***	**49**	**13,714**	82	13,714
17	0,686	50	0,686	***83***	***0,686***
18	51,429	**51**	**51,429**	84	51,429
19	12,000	52	12,000	85	12,000
20	**12,200**	53	8,727	86	8,727
21	3,111	54	0,436	87	0,436
22	0,873	55	1,000	**88**	**1,000**
23	0,945	56	20,000	89	20,000
24	**14,667**	57	1,400	90	1,400
25	**18,667**	58	0,500	91	0,500
26	**4,571**	59	0,167	92	0,167
27	***7,111***	**60**	**0,467**	93	0,467
28	***9,600***	**61**	**0,622**	**94**	**0,622**
29	2,945	62	0,222	**95**	**0,222**
30	6,750	63	21,333	**96**	**21,333**
31	6,300	64	12,000	***97***	***12,000***
32	2,250	65	1,500	***98***	***1,500***
33	0,750	66	0,600	99	0,600

**i* – номер вершини; жирним шрифтом відзначені вершини, що лежать на критичних шляхах, курсивом – поєднувані в загальні вершини.

Побудований за такими правилами граф (рис. 8) є графом меншої розмірності, тому що він включає тільки вершини на критичних шляхах. Таким чином, реалізується агрегування моделі до рівня «одного приладу». Вершини № 3 і № 4, а також № 14 і № 15 не об'єдналися з тієї причини, що група комірок «1-3», у якій вони виконуються, не є унікальною й не містить устаткування, що потребує переналагоджень. Вершини № 1 і № 2, №16 і №17, №12 і

№13 не об'єдналися тому, що не виконались умови їхнього об'єднання [1].

При об'єднанні спільних вершин можуть з'явитися однакові зв'язки. Тому при нумерації виконується процедура їхнього видалення. Крім того, потрібно виключити ситуації, коли поєднуються вершини, з яких не всі становлять листи. Тоді потрібно видалити виникаючі при цьому «зайві» нульові зв'язки.

Після побудови графа на критичних шляхах розраховуються розміри партій агрегованих операцій для кожного коміркокомплекта. Коміркокомплекти розбиваються на однакове число партій (при цьому процедура розподілу найбільш прост і швидка, тому що не вимагає перегляду попередників по кожній призначуваній партії й кожному спадкоємцю). Мінімальне число партій визначається максимальною тривалістю партії, рівної тривалості зміни, тому що не повинна бути порушена умова безперервності виконання партії. Тому мінімальна кількість партій дорівнює цілій частині максимальної із тривалостей коміркокомплектів, поділеній на тривалість зміни, плюс одиниця. Нарешті, кількість коміркокомплектів у партії визначається як тривалість агрегованої операції, помножена на кількість комірок у групі та розділена на кількість партій. У нашому прикладі число партій для коміркокомплектів дорівнює 1.

Центральною при розв'язанні задач погодженого планування ієрархічної моделі планування та управління є задача «Мінімізація сумарного зваженого моменту закінчення виконання завдань при відношенні порядку, заданому орієнтованим ациклічним графом, у якій для всіх вершин графа, крім кінцевих, вага дорівнює нулю» (МЗМН), розглянута в [3]. Ця задача служить для визначення черговості призначення агрегованих робіт на виконання, що у свою чергу є основною інформацією для розв'язання задач другого й третього рівня моделі.

Тому на графі критичних шляхів коміркокомплектів розв'язується оптимізаційна задача МЗМН (ваги задані тільки для кінцевих вершин – у цьому випадку це вершини 9, 10, 11 з вагами 1, 2, 3 відповідно. Ваги для інших вершин дорівнюють нулю). Алгоритм розв'язання описаний у [3]. Послідовність виконання коміркокомплектів з розбивкою на підпослідовності максимального пріоритету (ПМП) наведена в табл. 5.

Послідовність, отримана після розв'язання задачі МЗМН, побудована на агрегованому графі, що включає спільні вершини. Хоча існує практика об'єднання деяких коміркокомплектів

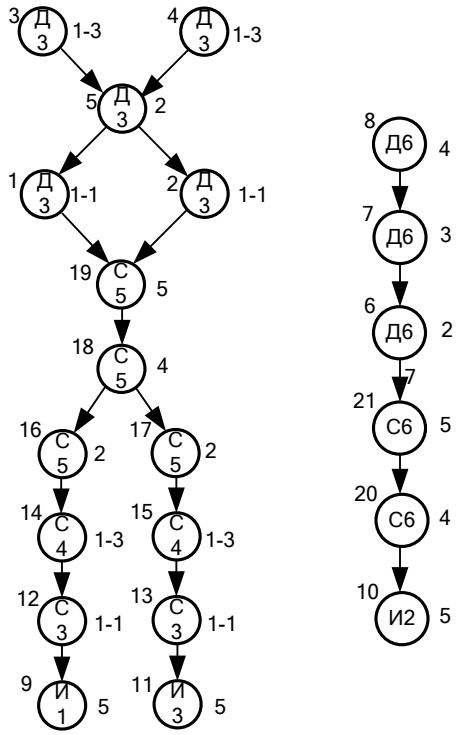


Рис. 8. Граф на критичних шляхах виробів

у спільні блоки, але при фактичному виконанні цих коміркокомплектів можливі ситуації, коли їхнє реальне об'єднання невигідно. Це може

Табл. 5. Пріоритетно-упорядкована послідовність виконання коміркокомплектів – результат розв'язання задачі МЗМН*

n	$n_{поч}$	ω_i	l_i	Спільна	C_i
ПМП №1:					
8	49	0	11,200	Ні	11,2
7	48	0	12,836	Ні	24,0
6	47	0	6,629	Ні	30,7
21	61	0	27,950	Ні	58,6
20	60	0	16,000	Ні	74,6
10	**51**	**2**	**8,150**	**Ні**	**82,8**
ПМП №2:					
3	12	0	12,000	Ні	94,8
4	78	0	13,650	Ні	108,4
5	16 і 83	0	31,314	Так	139,7
1	11	0	22,333	Ні	162,1
2	75	0	25,267	Ні	187,3
19	28 і 98	0	25,520	Так	212,8
18	27 і 97	0	19,200	Так	232,0
17	96	0	7,200	Ні	239,2
15	95	0	21,233	Ні	260,5
13	94	0	17,683	Ні	278,2
11	**88**	**3**	**18,000**	**Ні**	**296,2**
16	26	0	6,571	Ні	302,7
14	25	0	18,667	Ні	321,4
12	24	0	15,667	Ні	337,1
9	**20**	**1**	**17,200**	**Ні**	**354,3**

*n – номер вершини у графі на критичних шляхах, $n_{поч}$ – номер вершини у початковому графі коміркокомплектів, функціонал f розраховується для вершин з вагою: f_{10} = 165,5; f_{11} = 1054,0; $f_9 = f_\Sigma$ = 1408,3.

відбутися, наприклад, якщо фактичне завантаження ресурсів перевищує планові показники,

внаслідок чого виникають простої встаткування та зменшуються резерви часу. Більше того, фактична інформація про розподіл визначає більш точну модель графа, чим отримана при порівнянні тривалостей виконання з початку критичного шляху. Агрегована модель є неточним представленням моделі планування, так як вона працює з агрегованими показниками. Тому агрегована модель може не відповідати дійсності при розподілі коміркокомплектів на виконання. Якщо в результаті неточності моделі в спільну вершину поєднуються коміркокомплекти, які будуть фактично виконуватися в різні проміжки часу, то це приводить до затримки виконання всіх спадкоємців відповідних коміркокомплектів по графу зв'язності.

Для узгодження моделей першого (агрегованого) і другого (фактичного) рівнів потрібно проаналізувати фактичну інформацію про розподіл і сформувати новий набір спільних вершин, що відповідає їхньому фактичному об'єднанню. Очевидно, спільні вершини, які були об'єднані при агрегації, не можна розподілити на виконання як спільні. Агрегована модель повинна бути дезагрегована, тобто спільні вершини повинні бути розбиті на окремі коміркокомплекти, але потрібно дотриматись порядку виконання коміркокомплектів, визначеного розв'язанням задачі МЗМН. Тому будується новий граф зв'язності, у якому кожна із спільних вершин перетвориться в ланцюжок вершин критичних шляхів виробів зі збереженням відносин передування графа на критичних шляхах. При цьому попередники кожної об'єднаної в графі на критичних шляхах спільної вершини стають попередниками першої (починаючи з листів графа) вершини в ланцюжку, а спадкоємці кожної об'єднаної спільної вершини – спадкоємцями останньої вершини в ланцюжку.

Наприклад, якщо вершині *i* графа на критичних шляхах відповідає вершина B_i загального графа, а спільні вершини мають номера 3 і 6, і їм відповідають по дві вершини загального графа (позначимо їх умовно B_{31}, B_{32}, B_{61}, B_{62}; вони здобувають свої власні номери в даній процедурі), то здійснюється перетворення графа на критичних шляхах, показане на рис. 9.

Далі виконується розподіл коміркокомплектів побудованого графа критичних шляхів з розбивкою спільних вершин для уточнення інформації про їхнє об'єднання на стадії розподілу. У користувача в діалозі запитуються дати початку й закінчення планового періоду. За календарем робочих днів визначаються номери робочих днів початку та закінчення планового періоду, а також кількість робочих днів у плановому періоді. По кожній комірці розраховується фонд часу комірці в днях, який дорівнює добутку кількості робочих днів у плановому періоді та коефіцієнта змінності.

Процедура розподілу може застосовуватися для будь-якого вхідного графа (загальний граф зв'язності коміркокомплектів, граф на критичних шляхах або граф на критичних шляхах з розбивкою спільних вершин). Для правильної роботи відповідно до обраної підзадачі відкриваються відповідні заданому графу масиви зв'язків і вершин. Це можливо завдяки уніфікації їхніх структур. На даному етапі алгоритму виконується розподіл по графу з розбивкою спільних вершин і вводяться наступні припущення:

а) коміркокомплекти не розбиваються на партії: при розбивці на партії неможливо визначити фактичний набір спільних вершин, тому під «партією» в алгоритмі треба розуміти «коміркокомплект»;

б) у зв'язку з тим, що тривалість не розбитих на партії коміркокомплектів може перевищувати тривалість робочого дня в деяких комірках, уважається, що всі комірки працюють цілодобово.

Алгоритми розподілу описані в [2]. Для нашого приклада вони використовуються в припущенні, що завдання надходять у систему одночасно: $\forall j\ r_j = 0$. Згідно рис. 3 застосовується алгоритм розподілу 1, тому що розв'язується задача за критерієм 1 (максимізація прибутку підприємства при відсутності директивних строків).

Після першого розподілу на графі критичних шляхів проводиться перевизначення набору

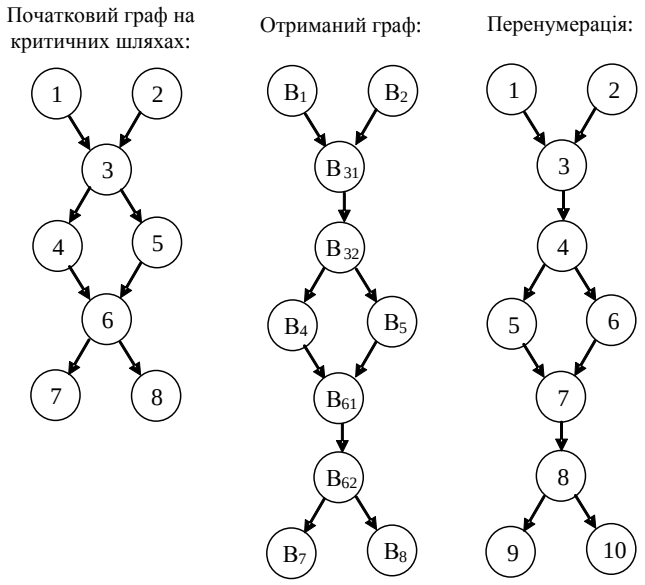


Рис. 9. Приклад перетворення графа при розбивці спільних вершин

спільних вершин відповідно до фактичній інформації про розподіл. Це здійснюється наступним алгоритмом: для кожної спільної вершини перевіряються ознаки об'єднання в графі на критичних шляхах і об'єднання при розподілі, і якщо вони не збігаються, то необхідно поставити ознаку об'єднання в графі на критичних шляхах в ознаку об'єднання при розподілі, ознаку необхідності перерозподілу – в «Істина».

Якщо набір спільних вершин не змінився, то модель не вимагає зміни й, отже, потрібно тільки перерозподілити коміркокомплекти з розбивкою на партії та доповненням до повного графа (наступний крок). Інакше модель перебудовується та шукається нова послідовність виконання агрегованих робіт (повторне розв'язання задачі МЗМН). Для цього знову виконуються процедури, починаючи з формування графа на критичних шляхах виробів. Для побудови графа використовується набір спільних вершин, отриманий при розподілі коміркокомплектів.

У результаті повторного розв'язання задачі МЗМН отримана припустима приоритетно-упорядкована послідовність, розбита на підпослідовності спадаючого пріоритету. Однак вона включає тільки коміркокомплекти, що лежать на критичних шляхах виробів, а для розподілу виробничої програми необхідно врахувати всі коміркокомплекти, необхідні для виконання всіх виробів. Тому для подальшої роботи алгоритму отримана послідовність доповнюється коміркокомплектами, що не лежать на критичних шляхах виробів, із присвоєнням їм відповідного номера ПМП. Весь надграф кожного з коміркокомплектів, що входять у приоритетно-упорядковану послідовність, буде мати той же номер ПМП, що й цей коміркокомплект.

Розподіл отриманого розкладу по комірках із прив'язкою до планового періоду виконується по одному з алгоритмів розподілу, описаних вище, з такими виключеннями:

– алгоритм виконується по загальному графу зв'язності коміркокомплектів або графу на критичних шляхах, залежно від обраної підзадачі;
– виконується розбивка коміркокомплектів на партії (кількість ітерацій дорівнює кількості партій і виконується для партій замість однієї ітерації для коміркокомплектів повної тривалості);
– комірки мають відповідну позмінну роботу, а не цілодобову.

Приклад отриманої виробничої програми показаний на рис. 10. На цьому рисунку час наведений у годинах з моменту початку планового періоду. Фактична дата й час запуску коміркокомплектів визначається за допомогою виробничого календаря, коефіцієнта змінності комірки, дати початку планового періоду, заділа на початок планового періоду й інших планових показників.

Рівень III. Побудова виробничої програми із прив'язкою до ресурсів

На третьому рівні ієрархічної моделі планування здійснюється формування повного плану виконання всіх операцій з розбивкою по ресурсах всіх комірок і з урахуванням розподілу коміркокомплектів, отриманого на другому рівні. При цьому можуть розв'язуватись оптимізаційні задачі.

Процедури розподілу операцій по ресурсах аналогічні алгоритмам 1, 2, 3 розподілу коміркокомплектів по комірках на другому рівні, що описані у [2]. Вибір алгоритму розподілу здійснюється залежно від необхідного критерію оптимізації розкладу: для одержання компактних розкладів вибирається аналог алгоритму 1 другого рівня, для незатримуючих – аналог алгоритму 2, для розкладів без запізнювання (критерії, що мають завдання з директивними строками) – аналог алгоритму 3. В алгоритмах третього рівня не здійснюється аналіз об'єднання спільних вершин. Розбивка агрегованих робіт на партії (для виробничих проектів) здійснюється за принципом забезпечення складання виробів за 1 зміну роботи складального цеху.

Оскільки в нашому прикладі розв'язується задача по критерію 1 [2], то застосовується алгоритм розподілу для одержання компактних розкладів операцій (аналог алгоритму 1 другого рівня), що забезпечить мінімізацію сумарного зваженого моменту виконання всіх операцій. Розподіл операцій по ресурсах проводиться за наступними правилами. Розглядається кожна комірка і здійснюється перегляд всіх коміркокомплектів (партій операцій), призначених на виконання в даній комірці. На кожному кроці на вільне робоче місце (ресурс) призначаються операції з мінімальним часом закінчення, що визначається як час початку виконання партії плюс тривалість виконання операції. Виходячи із принципів побудови плану виконання коміркокомплектів, всі ресурси будуть мати достатні резерви для виконання операцій, що лежать на критичному шляху операцій коміркокомплекта.

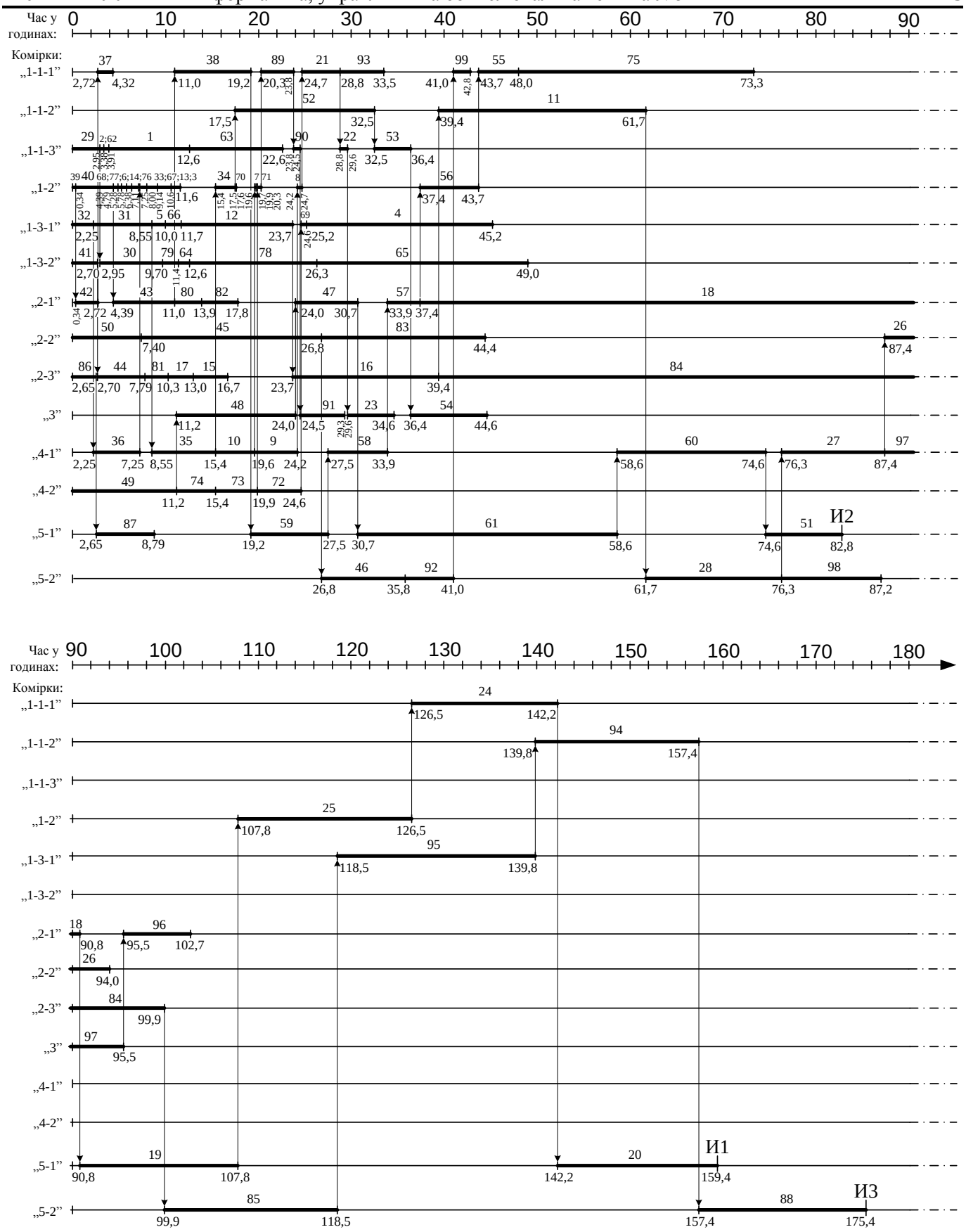


Рис. 10. Результат розподілу виробничої програми по комірках

За правилами побудови, кожний коміркокомплект містить ціле число операцій.

Схематично процес розподілу операцій показаний на рис. 11.

Як видно з рисунка, у результаті розподілу через округлення тривалості операції при визначенні тривалості коміркокомплектів у формулі (1) у ресурсах можуть виникати резерви. Ці резерви можна використовувати для виконання інших паралельних операцій, не потребуючі переналагодження в комірці.

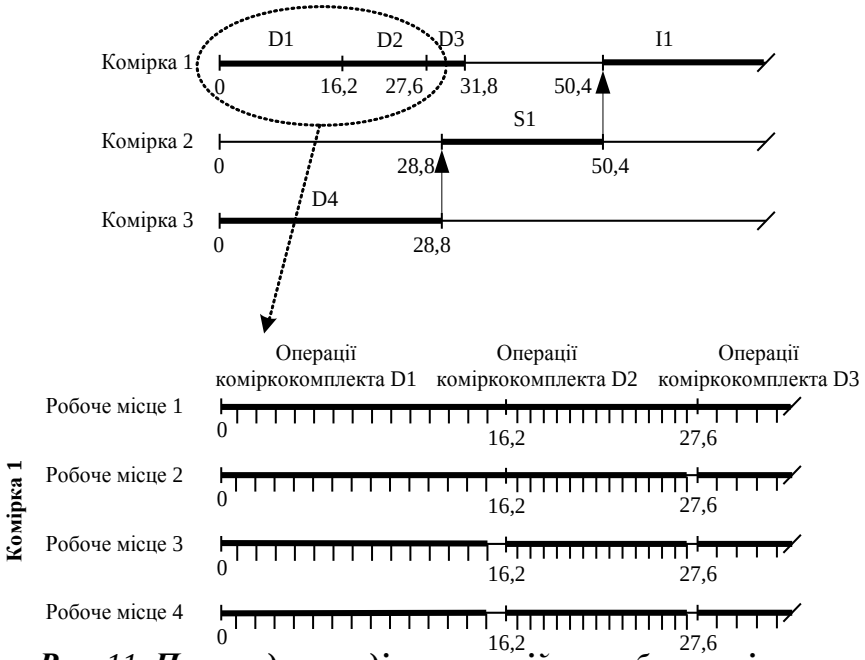


Рис. 11. Приклад розподілу операцій по робочих місцях

Висновки

Дана стаття є продовженням циклу робіт зі створення методології планування і управління в організаційно-виробничих системах у різних прикладних областях. Загальна фундаментальна проблема, що вирішується у даному циклі – розробка математичних моделей та методів ієрархічного планування та прийняття рішень в складних організаційно-виробничих системах з мережним представленням технологічних процесів та обмеженими ресурсами.

Створено інформаційну технологію та комплексну систему оперативного планування та прийняття рішень, що реалізує розв’язання задач планування в складних організаційно-виробничих системах за різними критеріями оптимальності. Інформаційна технологія без будь-яких суттєвих доробок може бути впроваджена у автоматизовані системи планування виробництва та системи планування проектами, що показано на прикладі дискретного виробництва.

Розроблене математичне та програмне забезпечення дозволить в умовах жорсткої ринкової конкуренції максимізувати прибуток підп.риємств, забезпечуючи найбільш повне завантаження обладнання, економію енергоресурсів за рахунок: ефективного використання обладнання, максимального скорочення виробничого циклу виготовлення виробів, виконання замовлень точно в строк, зниження витрат на проектування та виробництво та більш повного і своєчасного забезпечення потреб ринку у продукції. Економічний ефект для виробничих систем та систем управління проектами становить не менш 15% від вартості розробки нових виробів та технологій. На відміну від існуючих систем планування, в розробці враховано обмеженість потужностей та фактичне навантаження при виконанні робіт.

Запропоновані моделі та методи носять універсальний характер та знайдуть впровадження при розробці організаційно-виробничих систем у різних галузях народного господарства України, зокрема, для планування та управління робочим цехом, планування виробництв «на замовлення», планування виробництв дрібносерійного типу; ієрархічного планування виробництв по виготовленню партій; ієрархічного планування та управління проектами. Можливе також застосування розробленого комплексу моделей та методів ієрархічного планування в складних організаційно-виробничих системах в інших прикладних областях, наприклад, в системах автоматизованого проектування, інформаційних управляючих системах, системах автоматизації наукових досліджень, системах штучного інтелекту, системах проектування у будівництві та ін.

Перелік посилань

1. Згуровский М.З., Павлов А.А. Принятие решений в сетевых системах с ограниченными ресурсами: Монография.– К.: Наукова думка, – 2010.– 573 с.
2. Загальна схема планування та управління складними об’єктами, що мають мережне представлення технологічних процесів й обмежені ресурси. / Павлов О.А., Місюра О.Б., Мельников О.В., Щербатенко О.В., Михайлов В.В. // Вісник НТУУ “КПІ”. Інформатика, управління та обчислювальна техніка. К.: “ВЕК+”, 2009.– №49.– С.77-87
3. Павлов А.А., Теленик С.Ф. Информационные технологии и алгоритмизация в управлении.– К.: Техника.– 2002.– 344 с.

УДК 004.75

ЛУЦКИЙ Г.М.,
ВОЛОКИТА А.Н.,
ИВАНОВ Д.Г.

МЕТОД ОРГАНИЗАЦИИ ЗАЩИЩЕННЫХ ФАЙЛОВЫХ СИСТЕМ НА ОСНОВЕ КРИПТОГРАФИЧЕСКИХ ПРИМИТИВОВ КОНТРОЛЯ ДОСТУПА

Современные организации все больше ориентированы на облачные вычисления, одним из направлений которых являются удаленные системы хранения данных. Среди ключевых вопросов использования этого сервиса можно выделить обеспечение безопасности данных организации. В данной статье предложен метод организации защищенных файловых систем, отличающийся совместным использованием симметричного шифрования и внутренней схемы управления ключами, поддерживающий расширенную модель контроля доступа, и обеспечивающий более высокую производительность при необходимой степени безопасности.

Modern organizations are increasingly focused on cloud computing, one of the commonly used service of cloud computing is remote storage. Among the key issues surrounding the use of this service is provide security of organization's data. In this paper we present the method of secured file system organization based on cryptographic primitives for access control supporting enhanced access control model, using the internal key management and provides the necessary security and productivity.

Введение

В последнее время все более популярным становится использование удаленных систем хранения данных (СХД), в которых организации используют услуги внешнего поставщика сервиса хранения данных (Storage Service Provider, SSP). На данный момент основными мировыми компаниями по предоставлению данного типа услуг являются: Amazon, BC Grid, Arsenal Digital и Iron Mountain Digital.

Использование удаленных СХД для организации имеет следующие преимущества: экономия затрат на обслуживание СХД, простота управления, повышение гибкости и отказоустойчивости.

Однако широкому распространению данного сервиса препятствует целый ряд технических проблем, основной из которых является проблема обеспечения безопасности удаленных СХД. Так как, во-первых, данные находятся на удаленной системе, доступ к которой осуществляется через глобальную сеть, по умолчанию являющейся ненадежной средой обмена данными, а во-вторых, система хранения данных находится под управлением сторонней компании, от которой информация также должна быть защищена.

Реализация задач безопасности для удаленных СХД выполняется с помощью шифрования данных организации. Шифрование данных может быть реализовано на двух уровнях: уровне приложений и на уровне файловой системы. Если говорить о реализации на уровне приложений, то среди преимуществ можно выделить селективность шифрованных данных, а если говорить о реализации на уровне файловой системы, то однообразность работы и более высокий уровень безопасности.

На данный момент преобладают решения на уровне файловых систем, которые называются криптографическими файловыми системами (КФС) [1] – файловые системы с встроенными функциями криптографических преобразований.

В данной статье предложен метод организации криптографических файловых систем, который позволит разграничить права доступа и обеспечить целостность данных в удаленной СХД за счет использования криптографических примитивов контроля доступа.

Модели организации взаимодействия с удаленными системами хранения данных

Модели организации взаимодействия с удаленными системами хранения данных отличаются путями передачи данных и метаданных, централизованным или децентрализованным доступом, а также сложностью управления ключами. Под метаданными будем понимать структурированные данные, представляющие собой характеристики (атрибуты) описываемых сущ-

ностей для целей их идентификации, поиска, оценки и управления ими [2]. Существует три модели организации взаимодействия с удаленным СХД: централизованная, децентрализованная и модель с централизованным доступом к метаданным и децентрализованным доступом к данным. Будут рассмотрены особенности централизованной и децентрализованной моделей.

Централизованная модель доступа к СХД

В централизованной модели используется единая точка доступа для записи и чтения данных, хранящихся в уделенной СХД. Как показано на рис. 1, узлы организации используют защищенную файловую систему для шифрования/дешифрования данных, хранящихся на стороне SSP. Доступ к защищенной файловой системе осуществляется с помощью шлюза (файл-сервера), который обслуживает запросы от пользователей, находящихся на других узлах организации.

Для перехода к использованию удаленной СХД, согласно централизованной модели, организации необходимо настроить защищенную файловую систему, которая будет шифровать все данные при их записи на удаленную СХД и расшифровывать при чтении. При этом для пользователей доступ к данным является прозрачным. Обычно используются протоколы сетевых файловых систем Network File System (NFS) и Common Internet File System (CIFS). В этой модели конфиденциальность данных обеспечивается за счет использования простых схем управления ключами: данные могут быть зашифрованы единственным ключом, доступным только защищенной файловой системе шлюза. Кроме того, система контроля доступа функционирует на стороне организации, что позволяет использовать существующую локальную базу пользователей.

Основным недостатком данной модели является наличие шлюза доступа к данным, так как может привести к перегрузке канала передачи данных при больших рабочих нагрузках, не обладает достаточной масштабируемостью, кроме того, шлюз становится местом, при отказе которого прекращается работа всей системы.

Таким образом, использование централизованной модели является целесообразным при доступе к корпоративным данным из одного узла организации. Централизация и дополнительные расходы на криптографические операции снижают её эффективность при множественном доступе к данным.

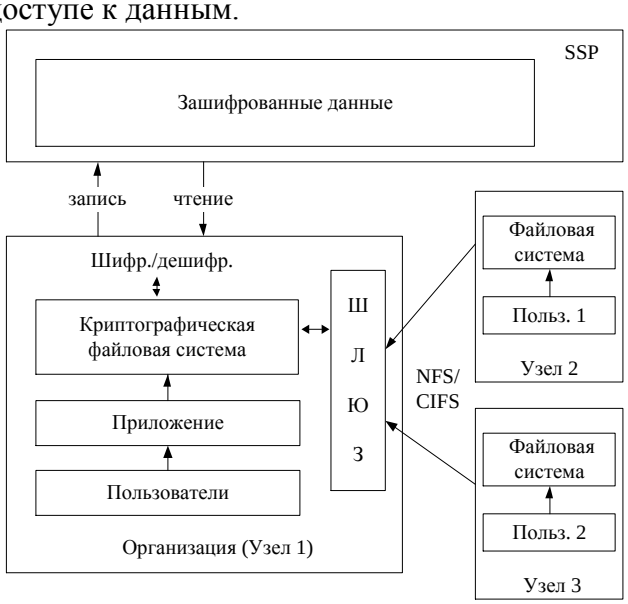


Рис. 1. Централизованная модель доступа к СХД

Децентрализованная модель доступа к СХД

В децентрализованной модели взаимодействия с удаленной СХД пользователи при доступе к данным или метаданным обращаются непосредственно с SSP (рис. 2).

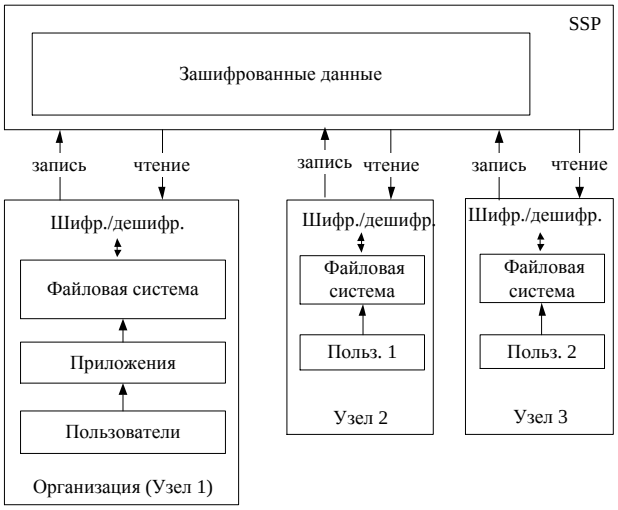


Рис. 2. Децентрализованная модель доступа к СХД

Использование данной модели исключает все узкие места, связанные с каналами передачи данных и метаданных, т.к. каждый пользователь имеет непосредственный доступ к SSP. Преимуществом децентрализованной модели является отсутствие необходимости управления шлюза доступа к данным, недостатком – проблема управления ключами, так как каждый

файл необходимо зашифровывать новым ключом.

Проведенный анализ моделей взаимодействия показывает, что использование централизованной модели в StaaS [3] нецелесообразно, т.к. снижается производи-тельность и увеличиваются расходы. Таким образом, в статье будут рассмотрены методы обеспечения безопасности удаленных СХД для децентрализованной модели организации взаимодействия.

Современные методы организации защищенных файловых систем

Рассмотрим основные методы организации КФС в зависимости от типов алгоритмов, используемых для шифрования данных и метаданных [4,5,6,7,8,9]:

1. Метод организации КФС без шифрования метаданных.

Идея метода заключается в том, что не выполняется шифрование метаданных, а сами данные шифруются с помощью симметричных алгоритмов шифрования. Среди реализаций данного метода можно выделить криптографические файловые системы CFS, CryptFS и NCryptFS.

2. Метод организации КФС с шифрованием метаданных. Метаданные шифруются с использованием асимметричных алгоритмов, а данные – симметричных алгоритмов. Криптографические файловые системы, получившие наибольшее распространение: SiRiUS и SNAD.

3. Метод организации КФС с шифрованием ключа метаданных. Отличительной чертой данного метода является шифрование данных и метаданных симметричными алгоритмами, а ключ метаданных шифруется ассиметрично.

В табл. 1 представлен сравнительный анализ методов и средств обеспечения информационной безопасности удаленных систем хранения данных.

Среди реализаций данного метода можно выделить криптографическую файловую систему Plutus.

Как видно из таблицы, ни одно из существующих средств не объединяет в себе безопасность и производительность: CFS, CryptFS и NCryptFS не поддерживают шифрование метаданных, SiRiUS и SNAD требуется большие временные затраты для работы с зашифрованными данными, так как используются медленные алгоритмы RSA, а Plutus предполагает наличие отдельно реализованного сервиса управления ключами, что повлияет на стоимость управления системы и повысит затраты време-

Табл. 1. Сравнительная характеристика современных методов и средств защиты информации в удаленных СХД

Характеристики	**Без шифрования метаданных**			**С шифрованием метаданных**		**С шифрованием ключа метаданных**
	CFS	**Cryptfs**	**NCryptfs**	**SiRiUS**	**SNAD**	**Plutus**
Обеспечение целостности	-	-	-	+	+	+
Внутренняя схема управления ключами	-	-	-	+	+	-
Проверка времени использования ключей	+	-	+	-	-	-
Алгоритм шифрования метаданных	-	-	-	RSA	RSA	3DES CBC/CTS
Длина ключа шифрования метаданных	-	-	-	2048	1024	168
Алгоритм шифрования данных	DES OFB+ ECB	Blowfish CBC	Blowfish CFB	AES CBC	Blowfish CFB	3DES CBC/CTS
Длина ключа шифрования данных, бит	56	128	128	128	128	168
Размер шифруемых блоков данных	8B	4KB/8KB	4KB/16KB	4KB/16KB	4KB/16KB	4KB
Поддержка групп пользователей	-	-	+	-	+	+
Поддержка разделения прав пользователей	-	-	+	+	+	+
Поддержка разделённого доступа к файлам	-	-	+	+	+	+
Метод реализации	На основе NFS	Стековая ФС	Стековая ФС	На основе NFS	Полноценная ФС	Полноценная ФС

ни на выполнения операции работы с данными.

Поэтому необходима разработка нового метода организации КФС с высоким уровнем безопасности и производительности, а также внутренней схемой управления ключами.

Метод организации защищенных файловой системы на основе криптографических примитивов контроля доступа

Предложен метод организации КФС на основе криптографических примитивов контроля доступа, максимально использующий преимущества симметричного шифрования, одним из которых является отсутствие временных затрат. Данный метод организации защищенных файловых систем, отличающийся совместным использованием симметричного шифрования и внутренней схемы управления ключами, поддерживающий расширенную модель контроля доступа, и обеспечивающий более высокую производительность при необходимой степени безопасности.

Модель контроля доступа метода является дискреционной, в которой каждый объект (файл, каталог или ссылка) имеет связанного с ним владельца, контролирующего к нему доступ. При этом доступ может быть предоставлен на основе субъектов трех типов: владелец (owner), группа-владелец (group) и все остальные (other).

Каждый пользователь для идентификации имеет пару ключей (открытый/секретный), которая обозначается <Bu, Pu>, где Bu является открытым ключом, и Pu – секретным ключом, который известен только пользователю.У каждой группы пользователей также есть аналогичная пара ключей <Bg, Pg>.

Пример структуры распределения ключей для группы пользователей student (<Bstud, Pstud>) показан в Табл. 2. Таблица группы состоит из двух частей: хеш-свертки идентификатора пользователя и зашифрованных ключей группы открытым ключом пользователя.

Зашифрованные ключи группы хранятся на стороне SSP. Когда пользователь выполняет вход в систему (то есть, монтирует файловую систему SSP), он получает зашифрованные ключи групп, к которым принадлежит, и использует свой секретный ключ для их расшифровки.

Табл.2. Таблица группы пользователей student

Пользователь	Ключи группы
Хэш-свертка (bob)	E_{Bbob}\{<Bstud, Pstud>\}
Хэш-свертка (alice)	E_{Balice}\{<Bstud, Pstud>\}

Для эффективной обработки файлы большого размера, как правило, разделяются на несколько блоков данных, которые шифруется отдельно. Это позволяет выполнять обновление данных более эффективно, т.к. нет необходимости повторно шифровать весь файл после внесения изменений в одну из его частей. В дальнейшем, для простоты описания, предполагается, что файлы помещаются в один блок данных.

Каждый блок данных имеет набор связанных с ним ключей:

1. Ключ шифрования данных, DEK (Data Encryption Key): DEK является уникальным симметричным ключом шифрования, используемым для шифрования блоков данных.

2. Ключ подписи данных, DSK (Data Signing Key) и ключ проверки данных, DVK (Data Verification Key): DSK и DVK являются парой асимметричных ключей, так что любые подписанные с использованием DSK данные, могут быть проверены только с DVK и, наоборот.

Аналогично, блоки метаданных также имеют связанные с ними ключи:

1. Ключ шифрования метаданных, MEK (Metadata Encryption Key): MEK является уникальным симметричным ключом, используемым для шифрования блоков метаданных.

2. Ключ подписи метаданных, MSK (Metadata Signing Key): MSK доступен только для пользователей, которые могут выполнить операцию записи в блоке метаданных.

3. Ключ проверки метаданных, MVK (Metadata Verification Key): MVK используется для проверки целостности блока метаданных.

Все типы ключей показаны в Табл. 3.

Табл. 3. Типы ключей доступа

Субъект	Обозначение
Пользователь	Bu
	Pu
Группа пользователей	Bg
	Pg
Блок данных	DEK
	DSK
	DVK
Блок метаданных	MEK
	MSK
	MVK

Блоки данных шифруются симметричными алгоритмами с использованием заранее сгене-

рированных ключей, поэтому пользователю необходимы соответствующие ключи. Для этого в блок метаданных было добавлено три новых поля: DEK, DSK и DVK. Таким образом, метаданные не только указывают на блоки данных, но и предоставляют пользователю ключи для работы с ними.

Также дополнительно выполнены следующие модификации. Во-первых, добавлено поле, содержащее MSK, доступное исключительно владельцу файла (каталога) и позволяющее подписывать блок метаданных при его обновлении, например при изменении прав доступа. Во-вторых, из блока метаданных удалены поля, изменяющиеся при записи модифицированного файла (каталога).

Блок данных каталога содержит таблицу, которая состоит из двух колонок, содержащих номер I-узла [10] и имена файлов и подкаталогов, содержащихся в этом каталоге. В таблицу каталога были добавлены два новых столбца, содержащие MEK и MVK для метаданных файлов и подкаталогов. Таким образом, таблица каталога предоставляет информацию о расположении блока метаданных, а также ключи для расшифровки и проверки.

Доступ к полям метаданных и таблиц каталога, содержащие ключи, является избирательным. Такая избирательность необходима для обеспечения контроля доступа и поддерживается с помощью криптографических примитивов CAP (Cryptographic Access Control Primitives) [11].

CAP объекта файловой системы заменяет контроль доступа в удаленной системе хранения данных, селективно предоставляя ключи доступа в метаданных и таблице каталога. Например, для установки разрешения только на чтение, CAP файла содержит поля DEK и DVK, а ключ DSK – недоступен.

Варианты метаданных и записей таблицы каталога с использованием CAP представлены на рис 3. Слева приведены права доступа, в среднем столбце – поля метаданных, а в правом столбце – таблица каталога. Для поддержки «нулевого» права доступа, все поля метаданных недоступны.

Право доступа «только для чтения» позволяет просматривать содержимое каталога (команда ls), но не позволяет выполнять переход или изменения (команды cd, rm и т.д.). Поэтому, CAP содержит ключи DEK и DVK в метаданных и поле «имя» в таблице каталога. Право «чтение-запись» каталога имеет одинаковое значение с «только для чтения», так как право на запись невозможно использовать без права на выполнение.

Разрешение «чтение-выполнение» позволяет пользователю переходить в каталог и получать доступ к содержимому, но не допускает изменений. В конструкции CAP для этого разрешения в метаданных доступны ключи DEK и DVK. В таблице каталога все четыре поля являются доступными, поскольку с разрешением на выполнение пользователи имеют возможность перехода в каталог и получения доступа ко всем метаданным файлов и подкаталогов. Используя номер I-узла, пользователи могут получить метаданные, используя MEK расшифровать их, после чего проверить целостность с помощью MVK.

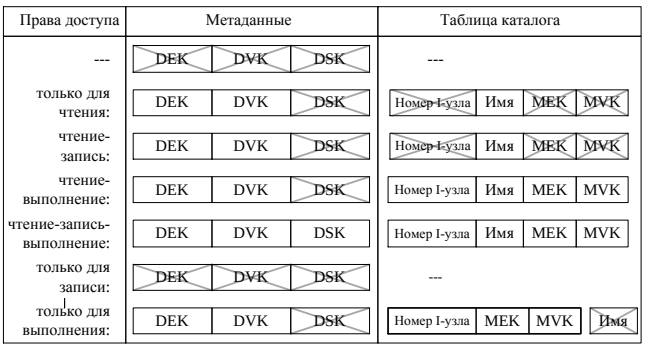


Рис. 3. Варианты CAP для каталогов

Право доступа «чтение-запись-выполнение» разрешает пользователям также изменять содержимое таблицы каталога, при этом поле с ключом DSK доступно.

Конструкция CAP для разрешения «только для записи» соответствует «нулевому» разрешению, так как такое разрешение не может работать без разрешенного права на «выполнение».

Наиболее интересна конструкция CAP разрешения «только для выполнения». Данное разрешение позволяет пользователям переходить в каталог и подкаталоги, однако пользователь не может просматривать их содержимое. Поскольку пользователи могут переходить в подкаталоги, то необходимо обеспечить доступ к таблице каталога, для этого DEK и DVK должны быть доступными, а поле «имя» недоступно. Это достигается за счет шифрования полей номера I-узла, MEK и MVK с помощью нового ключа, полученного из имени файла или подкаталога. Новый ключ генерируется следующим образом: с помощью ключевой хэш-функции такой как MD5 или SHA-1, где входным сообщением является имя, а ключом DEK. Таким образом, любой пользователь, знающий

имя файла или подкаталога, может сгенерировать ключ, а затем использовать его для расшифровки соответствующей строки в таблице каталога и получения доступ к метаданным.

Существует вариант прав доступа, который не поддерживается в описываемом методе – «запись-выполнение». Это связано с использованием симметричных ключей для шифрования блоков данных. Любой пользователь, который имеет право записи, может расшифровать блоки данных, используя тот же ключ и, следовательно, сможет прочитать содержимое. Тем не менее, такое разрешение очень редко встречается в реальных системах, и данный недостаток не является существенным.

Примеры САР для всех вариантов прав доступа к файлам представлены на рис 4. В случае с файлами, блоки данных не используются в модели доступа, поэтому они опущены. При «нулевом» разрешении, все поля метаданных, содержащие ключи, недоступны. При разрешении «только для чтения» DEK и DVK доступны, что позволяет расшифровывать и проверять блоки данных. САР для разрешения «чтение-запись» предоставляет также доступ к DSK. Разрешение «чтение-выполнение» предоставляет доступ к тем же ключам, как и «только для чтения», поскольку после расшифровки файла пользователь может выполнить его как программу. САР «чтения-записи-выполнения» соответствует САР «чтение-запись».

Права доступа	Метаданные		
---	~~DEK~~	~~DVK~~	~~DSK~~
только для чтения:	DEK	DVK	~~DSK~~
чтение-запись:	DEK	DVK	DSK
чтение-выполнение:	DEK	DVK	~~DSK~~
чтение-запись-выполнение:	DEK	DVK	DSK

Рис. 4. Варианты САР для файлов

Как и в каталогах, в САР файлов не поддерживается разрешение «только для записи». Кроме того, для удаленных систем хранения данных разрешение «только для выполнения» является недопустимым, так как при выполнении программы предварительно необходимо прочитать её файл.

При выполнении операции монтирования файловой системы пользователю необходимо считать суперблок, содержащий описание базовой структуры и атрибутов файловой системы, и номер I-узла корневого каталога. В данном случае также с номером I-узла хранятся MEK и MVK корневого каталога, что позволяет выполнить расшифровку его блока метаданных. Для каждого зарегистрированного пользователя суперблок зашифрован с помощью открытого ключа и хранится на стороне SSP. Таким образом, когда пользователь монтирует файловую систему, он расшифровывает суперблок, используя свой секретный ключ, и получает доступ к метаданным корневого каталога.

В данном методе присутствует недостаток избыточного хранения метаданных объектов файловой системы, так как двум пользователям с различными правами доступа необходимо предоставить два варианта САР. Для решения этой задачи была разработана схема дифференциации САР, предусматривающая использование для каждого файла трёх типов САР, определяющих права доступа владельца, группы-владельца и всех остальных соответственно.

Выводы

В данной статье предложен метод организации защищенных файловых систем на основе криптографических примитивов контроля доступа, отличающийся совместным использованием симметричного шифрования и внутренней схемы управления ключами, и обеспечивающий более высокую производительность при необходимой степени безопасности. Данный метод имеет ряд существенных преимуществ:

1. Поддержка расширенной модели контроля доступа, позволяющая более точно описывать права доступа к объектам файловой системы.

2. Использование внутреннего управле-ния ключами. В предложенном методе все ключи находятся внутри самой файловой системы и хранятся в зашифрованном виде на стороне SSP.

3. Совмещение уровня безопасности метода с шифрованием метаданных и уровня производительности метода без шифрования метаданных.

Одним из существенных недостатков данного метода является сложная структура метаданных объектов файловой системы и таблиц каталогов, что является предметом дальнейших исследований в данной области.

Список литературы

1. Muller, R . "How IT Works: Encrypting File System". TechNet Magazine. Microsoft. 2006-05
2. Воройский Ф.С. Информатика. Новый систематизированный словарь-справочник (Вводный курс по информатике и вычислительной технике в терминах). – 2-е изд., перераб. и доп.. – М.: Издательство Либерия, 2001. – С. 536.
3. Roettgers, Janko. "Piracy Beyond P2P: One-Click Hosters", Retrieved: 5 January 2008.
4. Blaze, M., “A Cryptographic File System for Unix”, in Proceeding of the ACM Conference on Computer and Communications Security, 1993
5. Zadok, Erez, Ion Badulescu and Alex Shender, "Cryptfs: A Stackable Vnode Level Encryption File System", CUCS-021-98.
6. Wright, C., Martino, M., and Zdok, E., “Ncryptfs: A secure and Convenient Cryptographic File System.” In Proceedings of the USENIX AnnualTechnical Conference, 2003
7. Goh, E., Shacham, H., Modadugu, N., and Dohen,D., “SiRiUS: securing reote untrusted storage,” in Proceedings of the Network and Distributed System Security Symposium (NDSS), 2003
8. Ethan Miller “Strong Security for Distributed File Systems”, 20th International Performance, Computing and Communications Conference, April 2001
9. Kallahalla M., Reidel, E.,Swaminathan, R., Wang, Q., and Fu, K., “Plutus: Scalable secure file sharing on untrusted storage,” in Proceedings of the USENIX Conference on File and Storgae Technologies (FAST), 2003
10. Робачевский А. Н., Немнюгин С. А., Стесик О. Л. Индексные дескрипторы / Базовая файловая система System V / Глава 4. Файловая система // Операционная система UNIX. – 2-е изд. – СПб.: БХВ-Петербург, 2008. – С. 334-. – 656 с.
11. Singh, A., and Liu, L., “Cryptographic Access Primitives in cryptographic file systems,” Georgia Tech CERCS Technical Report GIT-CERCS-07-08, 2008.

УДК 004.04 (043.3)

КУЛАКОВ О.Ю.,
БРОЛІНСЬКИЙ С.М.,
АШАЄВ Ю.М.

СИСТЕМА МОДЕЛЮВАННЯ РОБОТИ GRID СИСТЕМ З ЗАСТОСУВАННЯМ ІЄРАРХІЧНОГО ПЛАНУВАЛЬНИКА ТА МЕТОДУ ВІРТУАЛЬНИХ КЛАСТЕРІВ

У даній статті розглянуті алгоритми планування у Grid-системах, серед них метод віртуальних кластерів. В основу планувальника було покладено ієрархічний спосіб планування. Описана розроблена програма моделювання Grid системи, яка дозволяє використовувати вищезгадані та інші популярні алгоритми. Досліджено отримані результати алгоритмів їх недоліки та переваги у порівнянні з іншими методами планування, описано принципи реалізації моделювання за допомогою віртуальних кластерів.

This article describes the scheduling algorithms for Grid-systems including the method of virtual clusters. Hierarchical scheduling method was chosen as the basic method of scheduling. The Grid system simulation program, which allows the use of the mentioned and other popular algorithms, is described. The article includes received results and the comparison of the scheduling algorithms and describes the principles of the simulation using virtual clusters.

1. Вступ

Останнім часом до обчислювальних кластерів виявляється підвищений інтерес зі сторони науки, освіти і промисловості. Кількість кластерів постійно збільшується, але більшість із них рідко бувають завантажені на 100 відсотків, у той час, як деякі не здатні обслужити необхідну кількість заявок через високу завантаженість, або недостатню обчислювальну потужність. Тому постає задача об'єднання кількох кластерів в єдину grid систему з метою розподілення навантаження між підключеними кластерами. Але одночасно з перевагами об'єднання кластерів в єдину grid систему постає проблема планування задач між цими кластерами з метою балансування навантаження. Але оскільки кластери можуть знаходитися на значній відстані один від одного, планувальник має бути децентралізованим, тобто окремим для кожного кластеру, але мати можливість обміну інформацією з кількома "сусідніми" кластерами. У той самий час важливе місце займає моделювання можливого навантаження на систему перед реальним створенням цієї системи.

2. Існуючі рішення

На сьогоднішній день існують багато систем моделювання grid систем, зокрема Bricks, MicroGrid, OptorSim, SimGrid та GridSim. Всі ці системи мають свої недоліки і переваги, але серед головних недоліків слід зазначити такі, як вузька спеціалізація цих систем та відсутність, або застарілість графічного інтерфейсу користувача.

Але головною відмінністю запропонованого рішення від існуючих є можливість застосування планувальника, що використовує віртуальні grid системи[2] та застосування ієрархічного планувальника[1].

3. Програма моделювання grid

Нами була розроблена програма моделювання grid систем з графічним інтерфейсом користувача і відповідним графічним відображенням результатів моделювання. Для відображення результатів моделювання у програмі використовуються як діаграми так і лог файл з подіями, що відбулися при моделюванні на кожному його етапі та результатами обробки цих подій системою.

Для відображення використовуються діаграми, що відображають наступні параметри:

– Кількість виконаних задач кожним кластером за час моделювання.

– Завантаженість кожної ноди кожного кластеру за період моделювання.

– Час, який кожна задача знаходилася в черзі на обслуговування (а саме кількість задач, що очікували менше 10% максимального часу очікування, менше 20% і тд.)

У програмі кожен кластер містить у собі VDHCP сервер (планувальник та маршрутизатор) а також набір логічних обчислювальних елементів, об'єднаних в довільну топологію (створену користувачем), що містять одно- або багатоядерні процесори. Також кожний елемент має деякі, не нульові, обчислювальні потужності (набирається користувачем). Графічний інтерфейс користувача програми моделювання grid систем складається з 2 частин – зліва користувач має змогу додавати кластери, з'єднувати їх між собою та редагувати внутрішнє представлення кластеру. Справа користувач може редагувати властивості кожного елементу, що був попередньо розміщений зліва.

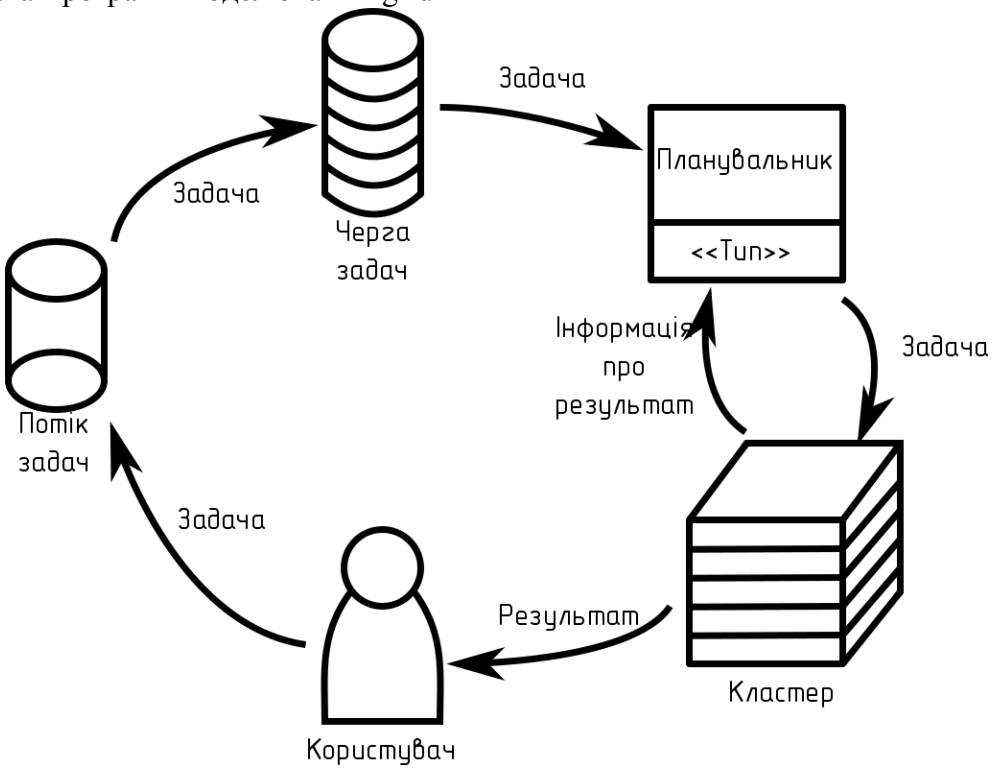


Рис.1. Діаграма процесу моделювання

4. Алгоритми моделювання

Діаграма процесу моделювання представлена на рис.1.

Для моделювання користувачеві дається на вибір 5 алгоритмів планування для grid системи:

– BackFillLocal[3,5]
– BestFitLocal[3,5]
– FirstFitLocal[3,5]
– RandomFitGlobal[3,5]
– VirtualGrid[2]

Кожен з цих алгоритмів має свої особливості.

Особливістю BackFillLocal є те, що невелика задача може бути запущена поза чергою, якщо вона не заважатиме роботі основної задачі, тобто займатиме менше відповідного відсотка (заданого у налаштуваннях моделювання) обчислювальних потужностей елемента.

Особливість BestFitLocal є те, що планувальник намагається знайти кластер, який би якнайкраще відповідав задачі – в ідеалі задача має завантажувати увесь кластер.

FirstFitLocal шукає перший (найближчий) кластер, що здатний обслужити заявку, а RandomFitGlobal обирає будь-який кластер, який може обслужити заявку.

VirtualGrid виділяє віртуальну grid систему, яка містить необхідну кількість ресурсів для обслуговування задачі, при цьому перевага надається найбільш зв’язаним між собою елементам (тобто тим, які розташовані найближче один до одного).

При реалізації планувальника було обрано ієрархічний тип планувальника, оскільки він об’єднує переваги централізованого та децентралізованого способів планування. Ієрархічний спосіб планування забезпечує високу ефективність планування, підтримує високу

надійність і може бути застосований для grid систем з необмеженою масштабованістю[1].

Результати моделювання відображені на рис. 2 і рис.3.

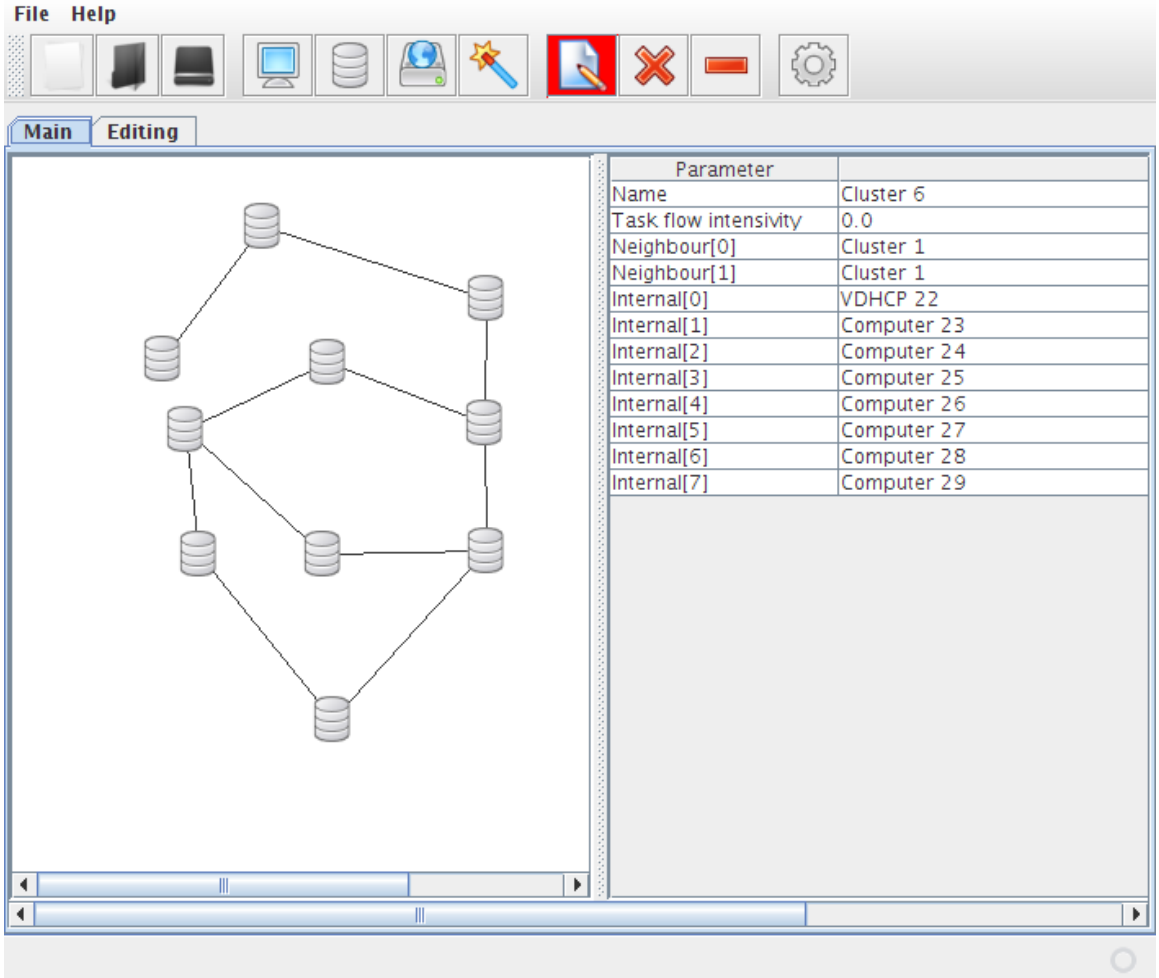


Рис. 2. Графічній інтерфейс користувача програми моделювання grid систем

Рис. 3. Відображення результатів моделювання

5. Моделювання за допомогою віртуальних grid систем

Об'єднання багатьох кластерів в єдину інтегровану систему забезпечує підвищену надійність такої системи за рахунок динамічної заміни вузлів у разі відключення та відмови частини з них.

Така система реалізує найголовнішу концепцію віртуалізації – приховування реальної архітектури та надання користувачеві деякої абстракції, зручної для користування. Користувач повинен лише вказати параметри необхідних ресурсів, а система в свою чергу відшукає необхідні вільні ресурси на локальному кластері, чи на інших віддалених кластерах та об'єднає ці ресурси у єдиний віртуальний кластер[4].

Для моделювання у програмі віртуальний кластер унаслідується від звичайного кластеру

і перевизначає усі методи, що пов'язані з моделюванням і включає у себе колекцію посилань на обчислювальні елементи.

Структура програми моделювання зображена на рис.4. Як видно з рисунку моделювання grid системи за допомогою віртуальних кластерів відбувається так само, як і моделювання системи фізичними кластерами, єдиним недоліком є неможливість точно визначити скільки саме задач обчислив кожен з кластерів, оскільки 1 задача може обчислюватись одночасно на кількох фізичних кластерах.

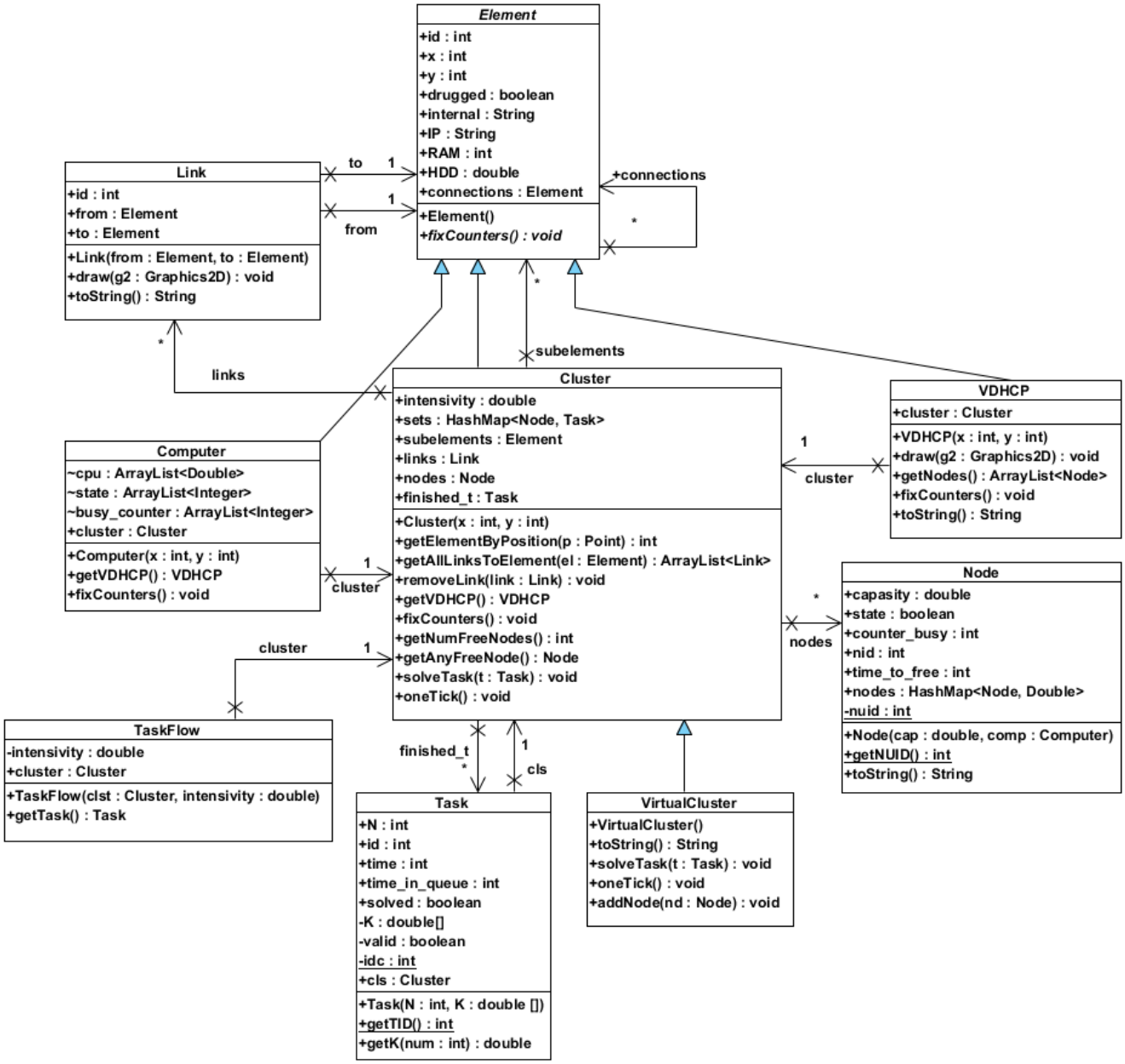


Рис. 4. Структура програми моделювання (модель даних)

6. Порівняльна характеристика

Для порівняння алгоритмів моделювання було проведено експеримент з використанням розробленої системи моделювання, у результаті чого були отримані дані, занесені в табл.1. З якого видно з результатів алгоритм VirtualGrid дає гарні результати при моделюванні і має кращі показники з максимального часу перебування в черзі ніж BackFillLocal та середнього часу перебування в черзі ніж BestFitLocal.

Табл. 1. Результати моделювання

Алгоритм	Кількість обчислених задач	Середній час перебування задачі у черзі	Максимальний час перебування задачі у черзі
BackFillLocal	70	615	1005
BestFitLocal	66	661	815
FirstFitLocal	56	814	1354
RandomFitGlobal	63	789	1275
VirtualGrid	70	610	990

7. Застосування

Дана програми моделювання grid систем може бути використана для планування організації топології обчислювальної мережі та порівняння ефективності використання того чи іншого планувальника. Користувач може оцінити показники ефективності використання ресурсів grid системи, максимальний час очікування задачі у черзі і навіть докладну статистику для кожного типу задачі. За допомогою використання ієрархічного планувальника[1] стає можливим оцінка виробничих потужностей системи навіть якщо вона не отримує задач напряму, а усі задачі надходять їй від суміжних систем.

Список посилань

1. Кулаков Ю. А., Русанова О.В., Шевело А. П. "Иерархический способ планирования для Grid" // Вісник НТУУ "КПІ". – Інформатика, управління та обчислюватьна техніка.–2009.–№51.–С.13– 21.
2. Кулаков О. Ю., Бролінський С. М., Ашаєв Ю. М. "Динамічне створення віртуальних Grid систем для вирішення розподілених задач на основі менеджера ресурсів" // Вісник НТУУ "КПІ".– Інформатика та обчислювальна техніка. – 2009.–№51.–С.125–129.
3. Sergii Brolinskyi "Scheduling Problem in Grid Systems" // Book of Abstracts "Innovations in Science and Technology". – Information Technologies.–2010.–С.182–183.
4. Yuriy Ashaev "Server Virtualization" // Book of Abstracts "Innovations in Science and Technology". – Information Technologies.–2010.–С.167–168.
5. Грушин Д. А. , Поспелов А. И. "Система моделирования Grid: реализация и возможности применения" [Електроний документ] Режим доступу: http://citforum.ru/nets/articles/grid_modelling/.

УДК 004.94

КАРПІНСЬКИЙ В.М.,
ЄВТУХ П.С.,
КІНАХ Я.І.

ТЕОРЕТИКО-ГРАФОВИЙ ПІДХІД ДО МОДЕЛЮВАННЯ РОЗПОДІЛЕНИХ БЕЗПРОВІДНИХ СЕНСОРНИХ МЕРЕЖ

Розглянуто теоретичну інтерпретацію для задач керованості, контрольованості та спостережуваності в безпровідних сенсорних мережах (БСМ) із застосуванням графів. Удосконалено підхід до оцінки розподілених БСМ, який дозволяє підвищити продуктивність. Досліджено задачу спостережуваності для цього підходу оцінки та отримані необхідні умови.

The theoretical interpretation using the graphs for the problems of control, controllability and observability in wireless sensor networks (WSN) was considered. Approach to the evaluation of distributed WSN, which allows to increase performance was improved. The problem of observability for this approach and assess the necessary conditions was investigated.

1. Постановка проблеми дослідження сенсорних мереж та її зв'язок із важливими науковими та практичними завданнями

Розподілені безпровідні сенсорні мережі (БСМ) становять важливий клас мереж, який в зв'язку з наявністю аномальних умов функціонування вимагають нового підходу до оцінки їх властивостей та на його підставі перспективного теоретико-графового методу та засобів аналізу задачі спостережуваності. Розв'язувані задачі пов'язані з тематикою НДР кафедри ЕЕ Тернопільського національного технічного університету (ТНТУ) та актуальність їх підсилюється запланованим виконанням важливих наукових і практичних завдань згідно з програмою "Наука та інженерія поза межами закону Мура" [1].

2. Аналіз останніх досліджень і публікацій в галузі теорії, аналізу та синтезу БСМ

2.1. Генезис проблеми

До витоків теорії створення об'єднаних в мережу систем можна віднести модель стада Рейнольдса і роботи авторів Vicsek та ін. В останні десятиріччя цій напрям набув подальшого інтенсивного розвитку з отриманням численних результатів. Значний комплекс досліджень зосереджений на двох основних складових – гомогенних мережах, де агенти або вузли ідентичні один одному, та гетерогенних мережах, в яких деякі представники володіють більш ширшими можливостями. Слід зазначити, що останнім часом моделям систем і мереж на підставі графів присвячується щодалі більша кількість наукових робіт і дисертаційних досліджень [2-5].

2.2. Аналіз розподілених безпровідних сенсорних мереж

Розподілена БСМ загалом складається з комплекту сенсорних вузлів, кожен з яких інтегрований з мікропроцесором, приймачем-передавачем (трансівером), а в деяких випадках виконавчим механізмом. Таким чином, сприймаючи сигнали на відстані, вузли здатні здійснювати нескладну обробку інформації та комунікуватися на короткі відстані. Незважаючи на поширення в довкіллі, цю інформацію слід повідомити до однієї або більше базових станцій для синтезу даних і прийняття рішень. Наявність таких центральних вузлів зумовлює віднесення БСМ до гетерогенної мережі.

Загальна мета БСМ полягає в збиранні інформації про довкілля, наприклад, освітленість, температуру, радіоактивність, переміщення і локалізацію, електричні, оптичні та теплові величини різноманітних засобів, параметри систем енергоспоживання, тощо, а також в моніторингу загроз безпеці в покритій сенсорною мережею зоні. Сприйняття за допомогою БСМ ставить важливий виклик і суттєву складну задачу для контролю та комунікації, тому що вимагається поєднання добре скоординованого зчитування даних, комунікації та принципів керування і контролю для подолання обмежень щодо потужності та пропускної здатності (Rahmani A., Ji M., Mesbahi M., Egerstedt M., 2007). В типовій БСМ, де наявна централізова-

на комунікація, тобто кожен вузол передає дані безпосередньо до центрального вузла, це неможливо здійснити із-за обмеженості ширини смуги. Обмежена відстань передавання сенсорними вузлами, а також той факт, що вони завжди розміщені на великій площі, зумовлюють надалі неможливість використання централізованої схеми.

В літературі запропоновані два підходи до координації чи узгодження – багатоетапну передачу, в якій беруть участь декілька проміжних вузлів-ретрансляторів (multi-hop), та із застосуванням мобільних центральних вузлів [6]. У мережах з ретрансляторами сенсорні вузли передають свої дані до базової станції безпосередньо або через інші вузли. В такому оточенні проміжний вузол виконує функцію маршрутизатора. Він передає свої власні дані разом з даними верхнього каналу зв'язку до наступного вузла. В наукових працях запропонована низка ретрансляційних протоколів маршрутизації із розв'язанням деяких задач, зокрема стабільності, втрати пакетів. Тим не менше, передбачені на комунікацію витрати призначені для подолання двох основних перешкод методу з використанням ретрансляторів, які в основному пов'язані з потужністю та накопиченням помилок.

Згідно з другим підходом центральний вузол (вузли) може переміщатися повсюди та збирати дані з вузлів, що знаходяться в межах дальності передачі. Уникнувши труднощів, з якими стикаються в мережах з ретрансляторами, принцип мобільних сенсорних вузлів теж має обмеження. Мобільний вузол може накопичувати інформацію лише від невеликої частини мережі, причому розмір цієї області та швидкість її переміщення визначаються оновленням частоти. Якщо площа велика та швидкість низька, частота може бути досить низькою, щоб вплинути на паралелізм або співпадання в часі даних.

Інші пов'язані на цю тему результати можна знайти в роботі R. Olfati-Saber (2005), зокрема в контексті узгоджувального фільтру, в якому автор звів розподілений фільтр Кальмана до двох окремих динамічних узгоджувальних задач. Іншими авторами – Gupta V., Jeffcoat D., Murray R. M. (2006), – представлений стохастичний підхід для формування оптимальних траєкторій давачів для висвітлення завдань мобільної сенсорної мережі. Оцінка задач в мережі з пакетним виділенням каналів досліджена в публікаціях Gupta V., Dana A. F., Hespanha J. P., Murray R. M. (2006), де наведено оптимальний підхід. Відповідно до нього вимагається, щоб кожен вузол в каналі зв'язку запам'ятав та опрацював отриманий пакет.

3. Формулювання мети статті (постановка завдання)

До мети статті належить розробка графобазованих засад оцінки розподілених БСМ підвищеної продуктивності з врахуванням спостережуваності і керованості в аномальних станах.

4. Розробка та дослідження теоретико-графових моделей та засобів аналізу БСМ

4.1. Моделі безпровідної сенсорної мережі на підставі графів

До об'єкту дослідження належать процеси функціонування компонентів БСМ в умовах дії збурювальних факторів. Графи в подальшому будуть адаптовані для представлення обмеженої поточної інформації в мережі, в якій дуги між вузлами відповідають спільно використовуваній інформації. Надалі вводиться низка основних понять і позначень щодо моделювання на основі графів та висвітлюються деякі корисні властивості, пов'язані з матричним поданням графів. Топологію БСМ можна описати орієнтованим графом $G\,(V, E, w)$, який визначається набором вузлів $V = \{1, ..., n\}$ і дуг $e \in E$, пов'язаних з набором вузлів декартовим добутком, тобто $E \subset V \times V$, причому w – ваговий коефіцієнт (рис. 1). Кількість вузлів графа G – це його порядок, а загальна кількість дуг – розмір. Нехай набір вузлів і дуг графа G іменуються $V(G)$ та $E(G)$, відповідно. Якщо застосовується символ $|\ |$ для позначення кардинального числа (кількості елементів множини), то отримується, що порядок графа – це $|V(G)|$ та його розмір $|E(G)|$ – це кількість його дуг.

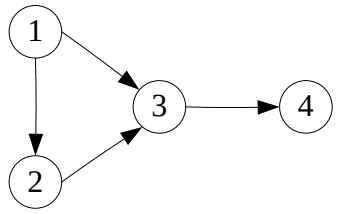


Рис. 1. Орієнтований граф для V={1, 2, 3, 4} та набору дуг E={(1, 2), (1, 3), (2, 3), (3, 4)}

Розглянемо стан моделювання БСМ на підставі графів. Співставимо граф взаємодії з наявними інформаційними потоками, в якому вузли відповідають сенсорам, а дуги – доступним міжсенсорним каналам зв'язку. Такі графи взаємодії – відповідні представники основної мережевої топології, викликаної обмеженістю інформації та локалізованою взаємодією.

Введемо математичне поняття площі мережі:

$$S_{i|i=\overline{1,n}} = \beta_i \sum_{j=1}^{n} w_{ij}\left(x_i(t) - x_j(t)\right)\Delta t_{ij}, \qquad (1)$$

де $x_i(t) \in \Re^n$ – вектор стану сенсора ϑ_i в момент часу *t*;

β_i, w_{ij} – вагові коефіцієнти, що визначаються для конкретної БСМ;

n_i – набір сусідів сенсора ϑ_i, наприклад, комплект сенсорів ϑ_j, які взаємодіють з сенсором ϑ_i,

Δt_{ij} – середній час обміну даними між сусідніми сенсорами.

Застосуємо правило найближчих сусідів:

$$\dot{x}_i(t) = \frac{S_{i|i=\overline{1,n}}}{\Delta t_{ij}}, \ \Delta t_{ij} = const. \qquad (2)$$

Для визначення швидкості передачі даних від групи сусідніх сенсорів до визначеного вузла доцільно скористатись математичною моделлю (2). Для цього розділимо значення відстані між вузлами на час передачі інформації. Отримуємо математичну модель залежності швидкості передачі даних від часу та відстані між вузлами:

$$v_*\left(\Delta t_{ij}, x\right) = \frac{\beta_i \sum_{j \in N_i} w_{ij}\left(x_i(t) - x_j(t)\right)}{\Delta t_{ij}\ v_{nom}}, \qquad (3)$$

де $v_*\left(\Delta t_{ij}, x\right)$ – швидкість передачі даних у відносних одиницях.

Результати чисельного експерименту наведені на рис. 2. Дослідження показали, що із збільшенням відстані між вузлами, необхідно збільшувати швидкість передачі даних за законом v_*. За такої умови забезпечується стабільна робота мережі, оскільки час передачі з довільного вузла буде однаковий.

Дослідження показали, що із збільшенням відстані між вузлами, необхідно збільшувати швидкість передачі даних за законом v_*. За такої умови забезпечується стабільна робота мережі, оскільки час передачі з довільного вузла буде однаковий.

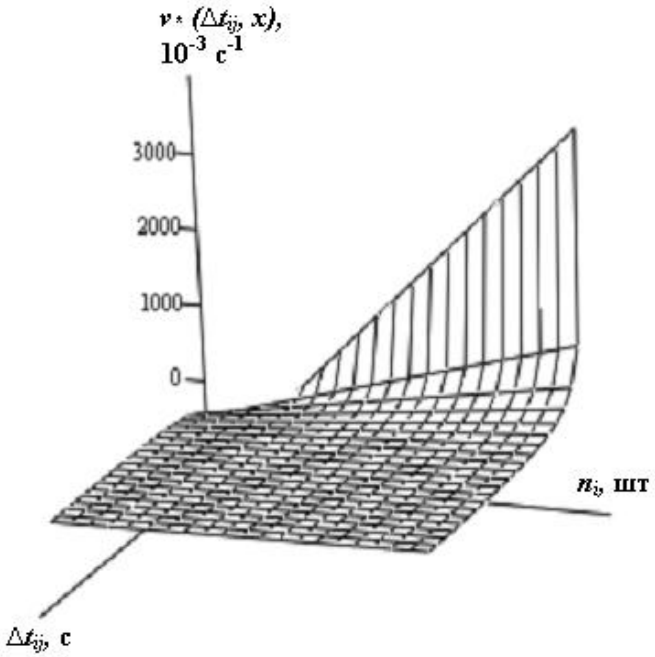


Рис. 2. Залежність швидкості передачі даних від часу та відстані між вузлами

Особливе значення для подальших досліджень відіграють графи з R_p-кругом суміжності (рис. 3) [2, 6]. У графі з R_p-кругом суміжності дуги встановлені між вузлами ϑ_i та ϑ_j, якщо та лише якщо сенсори знаходяться на відстані R_p один від одного, тобто, коли $\left|x_i - x_j\right| \le R_p$.

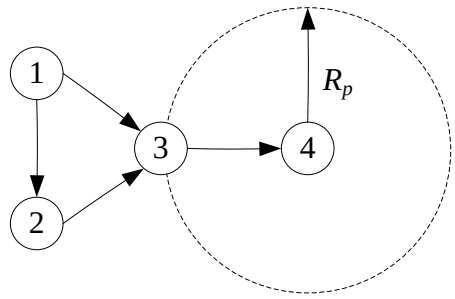


Рис. 3. Граф з R_p-кругом суміжності для V = {1, 2, 3, 4}

Слід зазначити, що такі графи носять динамічний характер, зокрема можуть з'явитися чи зникнути в цих ребрах в якості сенсорів або переміститися поза відстань сприйняття чи комунікації один з одним. Крім цього, можна припустити, що сенсори приєднуються або видаляються, в результаті чого не тільки набір дуг, але й комплект вузлів створює динамічну структуру. Надалі не досліджуватиметься цієї останньої ситуації і тим самим обмежується статич-

ним характером набору вузлів, в той же час комплекти дуг піддаються динамічним змінам.

4.2. Оцінка безпровідних сенсорних мереж на підставі графів

Розглядається БСМ з n сенсорами, базовий сенсорний граф $G_s = (V_s, E_s, w_s)$, вершини якого містять давачі, а дугами є канали зв'язку між парами вершин. Не відходячи від узагальнень, нехай $u_i \in \Re$ – вихідна фізична величина чи сигнал або параметр давача вузла (вершини) ϑ_i, $i = \{1, ..., N\}$. Потім за $x \in \Re^n$ приймається стан розподіленого оцінювача та за x_i – сигнал суміжного компонента, пов'язаного з вершиною ϑ_i. Слід зауважити, що x_i відрізняється формою від дійсного сигналу сприйняття u_i, причому x_i визначається на підставі алгоритму розподіленої оцінки. Ці сигнали x_i, а не u_i, передаються по каналах зв'язку та обчислюються процесорами, вбудованими в сенсорні вузли БСМ. Іншими словами, кожній вершині притаманні три функції: перша – давача, який моніторує значення параметра u_i; друга – компонента оброблення системи розподілених обчислень, який оновлює свій стан x_i при обробці інформації, отриманої з її каналів зв'язку; третя – трансівера, що обмінюється інформацією про стан зі своїми сусідами $(x_j, j \in n(i))$.

Базуючись на результатах (Ji M., 2007), необхідний алгоритм оцінки можна описати таким чином. В початковий момент відліку часу вузол (вершина) ϑ_i записує значення перетвореної або виміряної (сприйнятої чи зчитаної) ним величини u_i та використовує його початковим значенням для x_i. Потім вузол оновлює власний стан, грунтуючись на локальній інформації, тобто на усіх станах суміжних вузлів і своєму стані. Алгоритм можна сформулювати так:

$$\frac{dx_i}{dt} = \beta_i \sum_{j \in n(i)} w_{ij}(x)\ f\left(x_j(t) - x_i(t)\right),\ x_i(0) = u_i, \quad (4)$$

де $w_{ij}\ (x) = f\ (J_0(j\alpha u_i), J_n(j\alpha u_i))$, причому

$$J_0(j\alpha e_i) = \frac{\left(j\frac{\alpha e_i}{2}\right)^2}{1!^2} - \frac{\left(j\frac{\alpha e_i}{2}\right)^4}{2!^2} + \frac{\left(j\frac{\alpha e_i}{2}\right)^6}{3!^2} - ..., \quad (5)$$

$$J_n(j\alpha e_i) = \frac{\left(j\frac{\alpha e_i}{2}\right)^n}{0!n!} - \frac{\left(j\frac{\alpha e_i}{2}\right)^{n+2}}{1!(n+1)!} + \frac{\left(j\frac{\alpha e_i}{2}\right)^{n+4}}{2!(n+2)!} - ... \quad (6)$$

є функціями Бесселя з уявним аргументом нульового та вищих порядків.

На основі цього удосконаленого підходу центральні вузли здійснюють вибірку даних спостереження

$$y(t) = \varphi(x(t)),\ \ y(t) \in \Re^q,\ q << n, \quad (7)$$

де q – наявна кількість центральних вузлів, причому

$$x(t) = [x_1(t), ..., x_n(t)]^T. \quad (8)$$

Задача в подальшому зводиться до однозначного визначення $u = [u_1, ..., u_n]^T$ із $y(t)$.

Якщо прийняти правило найближчого суміжного вузла для $f(\cdot)$ згідно з (4), то отримується:

$$\frac{dx_i}{dt} = \beta_i \sum_{j \in n(i)} \beta_{ij}\ (x_j - x_i),\ i = 1, ..., n. \quad (9)$$

Використовуючи умовне позначення керування на основі графа, вираз (9) можна записати у вигляді

$$\begin{cases} \dot{x} = -L(G_s)x,\ x(0) = u, \\ y = Mx, \end{cases} \quad (10)$$

де $L(G_s)$ – лапласівська матриця графу, що відповідає БСМ,

$M \in \Re^{q \times n}$ – матриця даних моніторингу.

Оскільки граф вважається статичним, то надалі використовується L_s для $L(G_s)$, щоб це не викликало жодної плутани. Грунтуючись на доведенні (Ji M., 2007), можна констатувати, що система (10) стійка та $x_i(t)$ збігається до $\sum_1^n u_i / n$ для всіх $i \in \{1, n\}$ та $w_{ij} = const$. Тим не менш, з теорії керування відомо: якщо система $(-L_s, M)$ є спостережуваною, то можна повністю відновити первісний її стан $x(0)$ на підставі даних моніторингу $y(t)$. Слід відзначити, що для випадку декількох центральних вершин, кожний центральний вузол збирає сирі неопрацьовані відомості з сенсорної підмножини та накопичує їх разом, – ця дія рівнозначна множенню y на кожен рядок M. Потім сумарні результати повідомляються на одну суперверщину для остаточного оброблення інформації та прийняття рішення.

Також обмежено M до (0, 1)-матриці, а це означає, що всі канали однаково зважені. Якщо M містить ненульовий елемент у стовпці j,

рядок i, то це означає, що є комунікація між давачем j та центральним вузлом i.

4.3. Дослідження спостережуваності БСМ

Отримані в цьому пункті результати тісно пов'язані з наведеними іншими авторами (Egerstedt M. та ін., 2007) розв'язками із-за дуалізму між задачами спостереження та керування. Надалі використовуються деякі позначення, пов'язані з нетривіальним об'єктивним розділенням (НОР). При цьому очевидний тривіальний розклад зводиться до *n*-розділів, тобто $p = \{\{1\}, \{2\}, ..., \{n\}\}$.

Розглядаючи БСМ, припустимо, що її основному сенсорному графу притаманний НОР p_s з $p_s = r$ та C_{ps} – характеристична матриця НОР p_s. Внісши деякі зміни до позначень, нехай в подальшому $\overline{C_{ps}} = \left(C_{ps}^T C_{ps}\right)^{-0,5} C_{ps}$ буде нормалізованою характеристичною матрицею для C_{ps} та H_{ps}, які вибираються таким чином, що формується ортогональна матриця $T = \left[\overline{C_{ps}} \middle| \overline{H_{ps}}\right]$.

Беручи до уваги БСМ, розцінимо центральні вузли лідерами в структурі лідер-послідовник та позначимо результуючий граф розширеним сенсорним графом G. G – ніщо більше, як доповнення сенсорного графа G_s центральними вершинами та додання в G_s дуг, які представляють собою потоки інформації від сенсорних вузлів до центрального вузла. Сенсорний граф G_s відіграє аналогічну до повторюваного графа G_f роль в структурі лідер-послідовник. Відмінність полягає в тому, що в БСМ інформаційні потоки спрямовані від сенсорних вузлів до центрального вузла, тоді як у лідер-повторюваній структурі інформаційні потоки напрямлені від лідерів до послідовників.

За такої побудови справджується наступна лема:

- Для БСМ з графом комунікації G_s і розширеним графом G система (10) не є повністю спостережуваною, якщо існують НОР на G і G_s, іншими словами такі p та p_s, що всі нетривіальні соти НОР p містяться в НОР p_s, тобто такі $\exists\, p$ та p_s, що $|C_i| = 1$, $\forall C_i \in p \setminus p_s$. Більше того, $T_s = \left[\overline{P_s} \middle| \overline{Q_s}\right]$ є перетворенням подібності, що дає декомпозицію моніторованості системи.

Таким чином, сформульована достатня умова для не повністю керованої БСМ. Зворотне твердження цієї леми забезпечує необхідні умови для цілком моніторованої БСМ. Стверджується це таким наслідком:

- Маючи в наявності сенсорний граф комунікації G_s та його доповнений граф G, необхідна умова для системи $(-L_s, M)$ щодо спостережуваності полягає в тому, що не існує жоден НОР p та p_s над G та G_s, так що p та p_s спільно використовують всі нетривіальні соти.

Висновки та перспективи подальших досліджень у даному напрямку

Удосконалено принцип оцінки розподіленої БСМ. Згідно з цією концепцією кожен сенсорний вузол вільно переміщується за зближувально-подібною схемою в локальній інформаційній системі, а також центральні вузли оцінюють первинну дійсну інформацію шляхом спостереження невеликої групи сенсорів. Висвітлено одне з найважливіших питань удосконаленого підходу – спостережуваність та отримано низку необхідних умов. Подано теоретичну інтерпретацію із застосуванням графів для задач керованості, контрольованості та спостережуваності в розподілених БСМ.

Удосконалений метод менш складний порівняно із ретрансляційною концепцією чи принципом мобільного центрального вузла. Кожен сенсорний вузол нескладно обчислює середнє значення, навіть не знаючи ідентифікаційних ознак своїх сусідів. Він не повинен займатися задачею маршрутизації – найважливішою частиною в системах із застосуванням ретрансляторів чи багатократним відбиттям радіохвиль. З точки зору центральних вузлів, необхідно пройти через зону для зібрання даних, так що їм не доведеться турбуватися про задачу покриття чи планування руху.

Оскільки центральні вузли мають широкий діапазон комунікаційних можливостей, то вони можуть спільно використовувати чи сумісно розподілити інформацію та зберегти копію для кожного з них. Таким чином, кожен центральний вузол може відігравати роль засобу прийн-

яття рішення. А подальше розширення може бути таким, що кожен вузол виступає центральним вузлом, використовує хронологію (передісторію) свого власного стану, та робить логічний висновок про стан інших. Хоч ця схема є витонченою та стійкою в сенсі толерантності до помилок, вбачається за перспективне займатися цим напрямом.

Список літератури

1. Science and Engineering Beyond Moore's Law. – Режим доступу: http://www.nsf.gov/about/budget/fy2009/pdf/44_fy2009.pdf
2. Дунець Р.Б. Знаходження шляху обходу вершин дугами при візуалізації топологій на площині / Р. Б. Дунець, Т. М. Басюк // Вісник Нац. ун-ту “Львівська політехніка” “Комп’ютерні системи та мережі”. – 2007. – С. 43-48.
3. Кочкаров Азрет Ахматович. Новые теоретико-графовые подходы в моделировании сложных систем: дис. ... канд. физ.-мат. наук : 05.13.18 / Кочкаров Азрет Ахматович. – М., 2005. – 118 с.
4. Кулаков О.Ю. Програмно-апаратна організація GRID-систем на основі технології віртуальних мереж: автореф. дис. на здобуття наук. ступеня канд. техн. наук : спец. 05.13.05 “Компютерні системи та компоненти” / Кулаков Олексій Юрійович; Нац. техн. ун-т України "Київ. політехн. Ін-т". – Київ, 2010. – 22 с.
5. Парасюк І. М. Моделе-орієнтовані методи побудови та оцінювання програмних архітектур на основі нечітких графів / І. М. Парасюк, С. В. Єршов // Проблеми програмування. – 2008. – №2-3. – С. 181-190.
6. Kościelnik D. Overview of the Methods for Information Broadcast in ad-hoc Wireless Networks / D. Kościelnik, J. Stępień // Telecommunication Review. – 2010. – No 5. – Pp. 168-173.

УДК 004.77

РОКОВОЙ А.П.

АНАЛИЗ ЭФФЕКТИВНОСТИ VOIP КОДЕКОВ В ГЛОБАЛЬНОЙ СЕТИ

Предложен способ анализа эффективности VoIP кодеков в глобальной сети. По результатам эксперимента даны рекомендации по выбору подходящего кодека в зависимости от значений полосы пропускания, задержки пакетов, дисперсии задержки пакетов и потери пакетов.

A method for analyzing the efficiency of VoIP codecs in the global network is offered. Recommendations for choice suitable codec depending on values of bandwidth, packet delay, jitter and packet loss are given by results of experiment.

1. Введение

Передача речевой информации в IP сетях или IP-телефония (VoIP) [1] предъявляет к сетевой инфраструктуре гораздо более жесткие требования, чем любое другое приложение. Степень доступности сети, потеря пакетов и временные задержки оказывают существенное влияние на качество передачи речи в IP сетях. К сожалению, в большинстве случаев конечные узлы не содержат механизмов для эффективной борьбы с этими проблемами. Поскольку и потери пакетов, и временные характеристики их доставки являются сетевыми характеристиками, и конечные узлы могут их только наблюдать. Это особенно актуально для глобальной сети, где нет возможности вносить изменения в настройки промежуточных устройств – маршрутизаторов, которые обеспечивают продвижение пакетов от источника к приемнику.

На качество передачи речи в IP сетях оказывают влияние следующие параметры:

1) полоса пропускания канала передачи данных;
2) потери пакетов;
3) задержка;
4) дисперсия задержки (джиттер);
5) VoIP кодек и его параметры.

VoIP кодеки обеспечивают различную степень сжатия аудиоинформации, являются более или менее устойчивыми к потерям пакетов, а также к временным характеристикам доставки пакетов. Поэтому правильный выбор VoIP кодеков, а также их параметров позволит повысить качество передачи речи в IP сетях.

При этом возникает задача: определить зависимость качества передачи речи в IP сетях для различных кодеков от потери пакетов и временных характеристик их доставки.

2. Обзор voip кодеков

Под кодеками, как правило, понимают различные математические модели, используемые для цифрового кодирования и сжатия аналоговой аудиоинформации. Цель различных алгоритмов кодирования – обеспечить баланс между эффективностью и качеством. Изначально термин кодек был образован от слов кодер/декодер – это устройство, которое выполняет преобразования между аналоговым и цифровым сигналом. Теперь этот термин больше относится к понятиям компрессия/декомпрессия.

G.711 – основной кодек PSTN. Используется два метода компандирования: μlaw в Северной Америке и alaw во всем остальном мире. Любой из них обеспечивает передачу 8-битового слова 8000 раз в секунду. Это требует передачи 64000 бит/с. На самом деле G.711 является базовым кодеком, от которого были произведены все остальные. G.711 требует минимальные вычислительные ресурсы для своей реализации.

G.723.1 – это кодек с малой требуемой полосой пропускания канала (6.3Kbps или 5.3Kbps) и в основном используется со стандартом H.323. Использование этого кодека в продуктах требует лицензии.

G.726 – одним из исходных кодеков со сжатием. Эта технология известна как адаптивная дифференциальная импульсно-кодовая модуляция (Adaptive Differential Pulse-Code Modulation, ADPCM), она обеспечивает разную скорость передачи данных. Чаще всего используются скорости 16, 24 и 32 Кбит/с. G.726 предлагает качество практически такое же, как у G.711, но использует только половину полосы пропускания. Это возможно пото-

му, что он отправляет не результат измерения, а только достаточную информацию для описания разницы между текущим и предыдущим за мерами.

G.729A – учитывая, насколько малую полосу пропускания использует кодек G.729A, он обеспечивает впечатляющее качество звука. Делает он это за счет технологии Conjugate-Structure Algebraic-Code-Excited Linear Prediction (CS-ACELP). G729A является запатентованным продуктом, поэтому его нельзя использовать без лицензии; однако он чрезвычайно популярен и, соответственно, поддерживается многими телефонами и системами. Чтобы достичь такой значительной степени сжатия, этот кодек требует такой же значительной мощности ЦП.

GSM – самый популярный кодек Asterisk. Он не обременен лицензионными соглашениями, как G.729A, и предлагает превосходную производительность, если учитывать требования, которые он предъявляет к ЦП. Качество получаемого звука, в общем, считается ниже, чем обеспечивает G.729A.

iLBC (Internet Low Bitrate Codec) обеспечивает привлекательное сочетание низкого коэффициента использования полосы пропускания и приемлемого качества. Он особенно хорошо подходит для обеспечения целесообразного качества в сетевых соединениях с потерями.

Speex – это кодек с переменной скоростью передачи цифровых данных (variable bitrate, VBR). Это означает, что он может динамически менять скорость передачи данных в ответ на изменение условий сети. Speex – абсолютно бесплатный кодек, лицензированный по версии Xiph.org лицензии BSD. Данный кодек может использоваться для каналов со скоростью передачи данных от 2,15 до 22,4 Кбит/с благодаря его способности менять свою скорость передачи данных.

3. Постановка задачи

Для оценки качества передачи речи в IP сетях используют усредненную субъективную оценку MOS (Mean Opinion Score) [2]. Алгоритм MOS основан на пятибалльной шкале – от единицы (самое плохое качество связи) до пяти (самое хорошее качество). В соответствии с этим алгоритмом группа людей оценивает качество звучания тестовых сообщений, передаваемых по сети. Среднее значение этих оценок и определяет качество передачи речи. Недостатком такого подхода является его субъективность.

Существуют алгоритмы, которые автоматизируют процесс получения оценок MOS, обеспечивая их объективность: PSQM (Perceptual Speech Quality Measure), PSQM+, PESQ (Perceptual Evaluation of Speech Quality) [3] и PAMS (Perceptual Analysis Measurement System). Но они также имеют недостатки: сложны в реализации, требуют аппаратной поддержки либо не учитывают особенностей передачи данных в IP сетях.

В электроакустике искажения, которые вносятся аппаратурой во всех звеньях звуковых трактов, разделяют на две категории: линейные и нелинейные. В большинстве случаев линейные искажения не связаны с потерей качества звуковых сигналов. Нелинейные искажения характеризуются появлением в спектре сигнала новых составляющих, которые изменяют временную структуру первоначального сигнала в зависимости от его уровня. Они воспринимаются как резкость, грубость звучания и характеризуются появлением шумов, хрипов и тресков в сигнале. В настоящей статье предлагается использовать коэффициент нелинейных искажений для оценки качества передачи речи в IP сетях.

Для этого необходимо определить зависимость коэффициента нелинейных искажений звукового сигнала при передаче его от узла источника к узлу приемнику для заданных VoIP кодеков, при изменении параметров: ширина полосы пропускания, потери пакетов, задержки пакетов и дисперсии задержки пакетов. Значения изменяемых параметров должны соответствовать средним значениям для глобальной сети.

Если на вход устройства обработки звука подать синусоидальный сигнал, считая, что он состоит только из одной первой гармоники, то на выходе этого устройства, в общем случае, получится сигнал, состоящий из суммы синусоидальных сигналов с частотами: $f, 2f, 3f \ldots Nf$, где f – частота исходного синусоидального сигнала. Эти синусоидальные сигналы называются гармониками сигнала. Соотношение исходного сигнала (первой гармоники) и гармоник с более высокими номерами, выраженное через формулу (1), и называется коэффициентом нелинейных искажений K [4].

$$K = \frac{\sqrt[2]{U_2^2 + U_3^2 + \cdots + U_n^2}}{U_1} \qquad (1)$$

где:

U_n – амплитуда n-ой гармоники сигнала.

Учитывая, что сигнал представлен в цифровом виде, то для определения амплитуд гармоник необходимо выполнить дискретное преобразование Фурье [5] по формуле:

$$X_k = \sum_{n=0}^{N-1} x_n \, e^{-\frac{2\pi i}{N} kn} \qquad (2)$$

где:

X_k – комплексное значение k-ой гармоники сигнала;

N – количество значений сигнала;

x_n – значение сигнала.

По полученным комплексным значениям X_k вычислить амплитуды гармоник U_n по формуле:

$$U_n = \frac{|X_k|}{N} \qquad (3)$$

где:

$|X_k|$ – модуль комплексного значения k-ой гармоники.

4. Описание эксперимента

На узле источнике генерируется цифровой синусоидальный сигнал и воспроизводится с помощью стандартных утилит операционной системы. С выхода усилителя мощности звуковой карты сигнал подается на микрофонный вход этой же звуковой карты и средствами IP-телефонии передается с одного узла на другой. На приемном узле сигнал с выхода усилителя мощности звуковой карты подается на микрофонный вход и записывается в файл с помощью стандартных утилит операционной системы.

Для сигнала, полученного на выходе, выполняется дискретное преобразование Фурье по формуле (2). По полученным значениям амплитуд гармоник вычисляется коэффициент нелинейных искажений K по формуле (1).

Эксперименты проводятся для разных кодеков, при этом изменяются параметры: полоса пропускания, время задержки, дисперсия времени задержки и количество теряемых пакетов.

На рисунке 1 показана схема сети для проведения эксперимента.

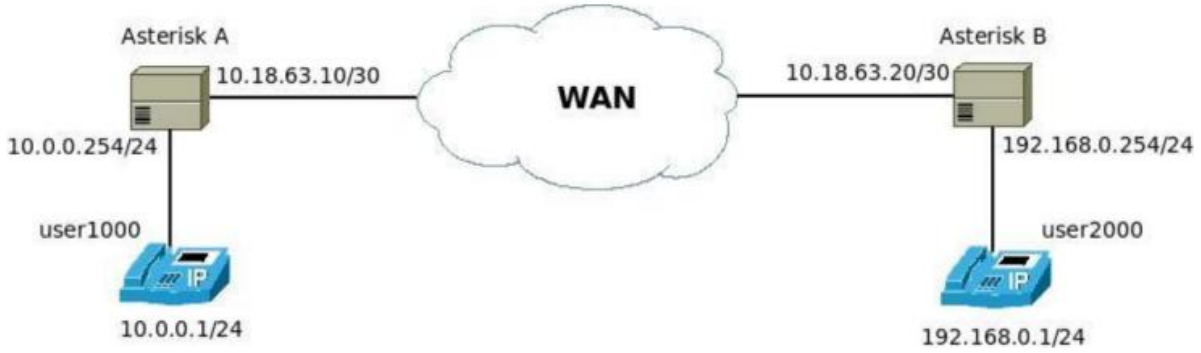


Рис.1. Схема сети эксперимента

В состав сети для проведения экспериментов входят компоненты:

1) WAN эмулятор – реализован средствами IPFW и Dummynet[6] в операционной системе FreeBSD 7.2 i386. Сервер содержит 2 сетевых интерфейса для соединения двух сегментов сети.

2) два PBX сервера – реализованы на базе операционной системы Linux Centos 5.4 i386 и ПО Asterisk [1]. Сервера содержат по два сетевых интерфейса.

3) два IP-телефона реализованы программным обеспечением X-Lite для операционной системы Windows XP.

Аппаратная часть всех компонент:

1) процессор Intel Pentium 4 с частотой 2 ГГц;

2) ОЗУ 2 Гб;

3) Сетевой адаптер Intel PRO/100.

5. Полученные результаты

Качество передачи голосовых данных в IP сетях измерялось коэффициентом нелинейных искажений (гармоник). То есть все графики представляются в виде зависимости коэффициента гармоник от значений полосы пропускания, задержки пакетов, дисперсии задержки пакетов и потери пакетов.

Зависимость коэффициента гармоник от пропускной способности представлена на рисунке 2.

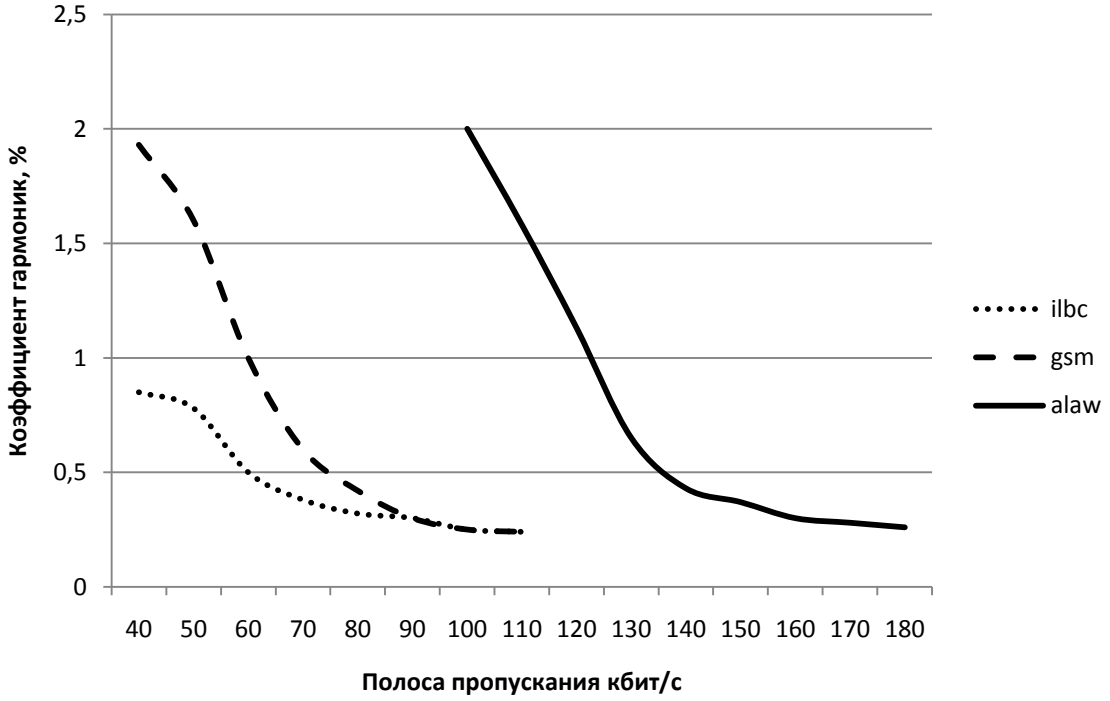


Рис.2. График.

Зависимость коэффициента гармоник от задержки представлена на рисунке 3.

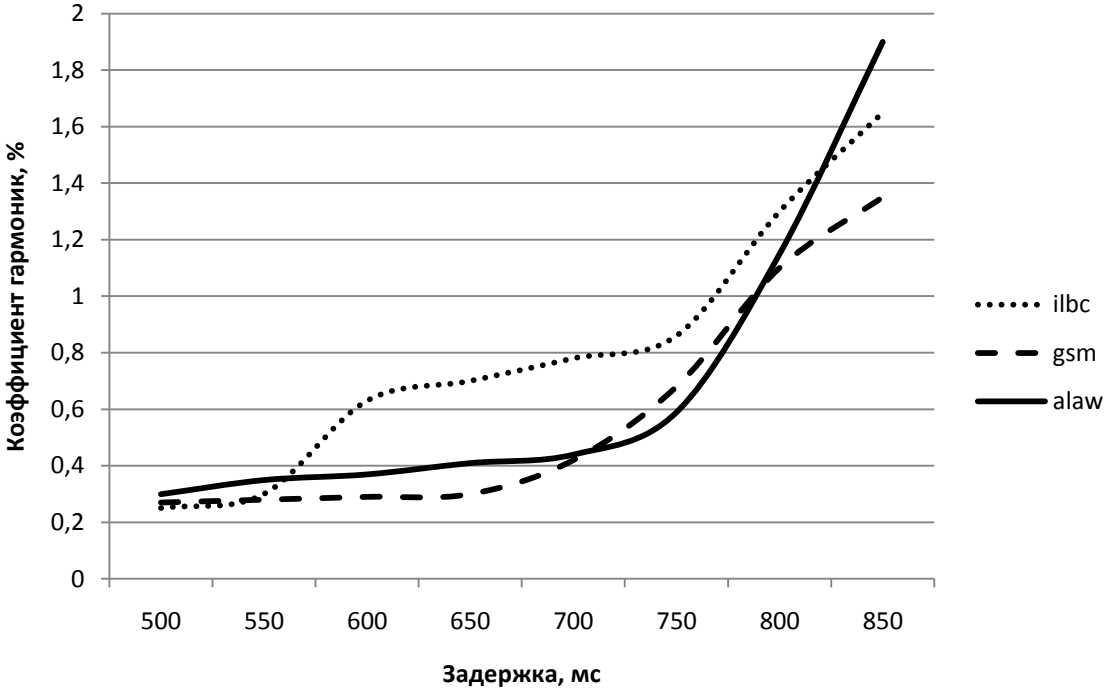


Рис.3. График.

Зависимость коэффициента гармоник от джиттера представлена на рисунке 4.

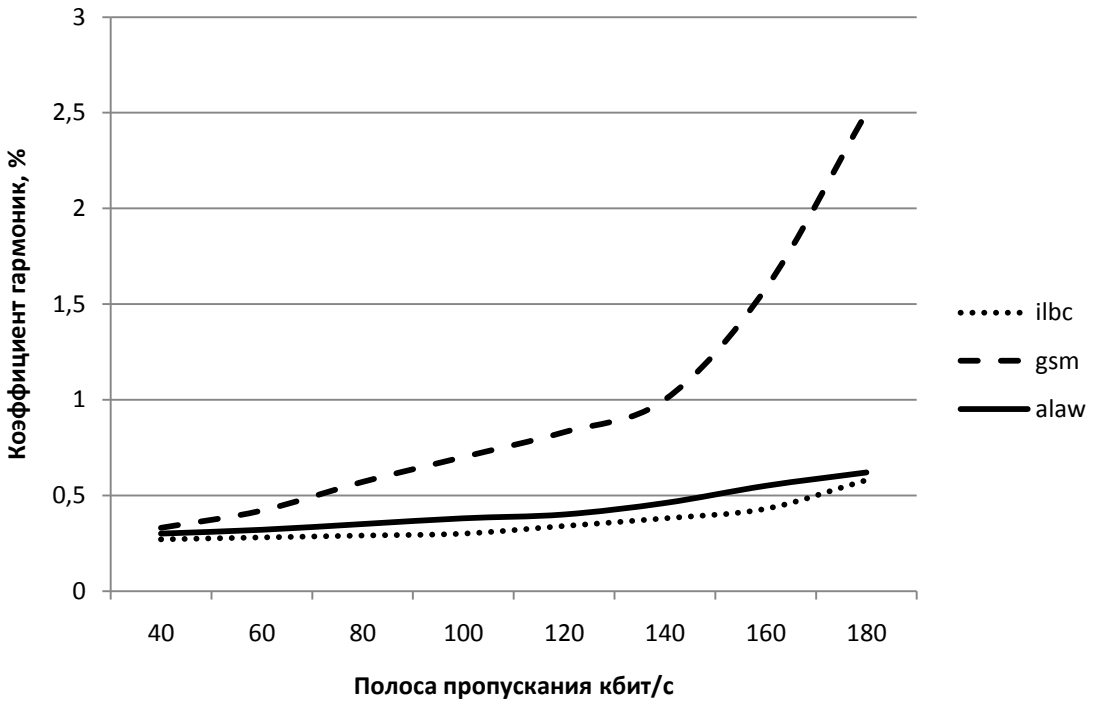


Рис.4. График.

Зависимость коэффициента гармоник от потерь пакетов представлена на рисунке 5.

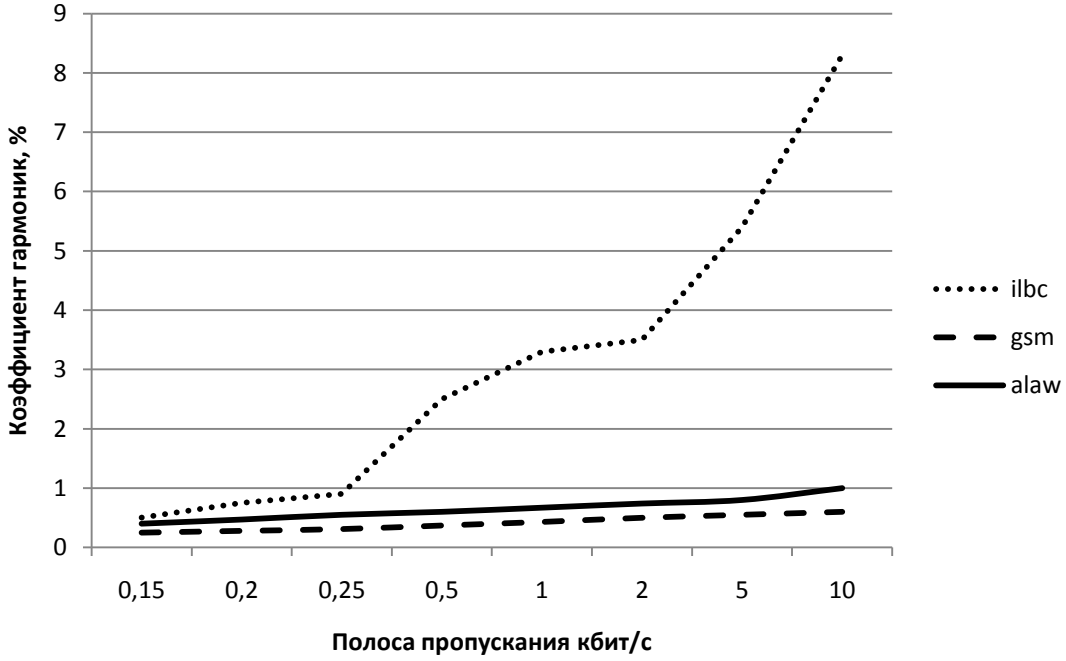


Рис.5. График.

9. Заключение

1. Пропускная способность большим образом влияет на коэффициент гармоник. При эмулировании пропускной способности меньшей чем скорость кодировании речи разных кодеков наблюдался достаточно большой коэффициент искажений. Причина этому – отброс пакетов на узле маршрутизации. При небольших потерях пакетов кодеки пытаются сгладить искажения. Но при достаточном увеличении процента потерь коэффициент гармоник стремительно растет вверх. При влиянии джиттера и задержек кодеки ведут себя стабильно, но при сильно больших значениях коэффициент линейным образом резко увеличивается.

2. Кодек alaw требует самой большой пропускной способности – более чем 150кбит/с.

Кодеки gsm и ilbc неплохо работаю при 70-80кбит/с. Это говорит о том, что эти кодеки достаточно хорошо сжимают речь.

3. Кодек ilbc совсем не стойкий при больших потерях. При 0,25% потерь пакетов коэффициент стремительно возрастает. Намного лучше себя ведут кодеки alaw и gsm. Тут прослеживается линейная зависимость, а также то, что gsm наиболее лучше использовать при каких-либо потерях в сети.

4. При влиянии джиттера все три кодека стабильно работают при значении 50мс. Это и показывает о существовании джиттер-буфера. А при увеличении джиттера при использовании кодека gsm наблюдаются большие искажения. Это объясняется большими потерями пакетов непосредственно на клиенте (приемнике).

5. Влияние задержек в сети имеют приблизительно одинаковые результаты для кодеков alaw и gsm. И при увеличении задержек более чем на 550мс дают плохие результаты. Стандарт ITU-T G.114 рекомендует, что задержка при передаче голоса не должна превышать 150 мс, поэтому в этом случае измерение коэффициента гармоник носит чисто экспериментальный характер.

Список литературы

1. Джим Ван Меггелен, Лейф Мадсен, Джаред Смит – Asterisk: будущее телефонии, 2-е издание, – Пер. с англ. – СПб: Символ-Плюс, 2009. – 656с.
2. «Methods for subjective determination of transmission quality», ITU-T Recommendation P.800 August 1996.
3. «Perceptual evaluation of speech quality (PESQ): An objective method for end-to-end speech quality assessment of narrow-band telephone networks and speech codecs» ITU-T Recommendation P.862 February 2001.
4. А. П. Гогучев, «Теоретические основы радиотехники», Высшая школа, 2002. – 306с.
5. Сергиенко А. Б. «Цифровая обработка сигналов». – 2-е. – Спб: Питер, 2006. – 751с.
6. Luigi Rizzo, «Dummynet», RFC2398, August 1998.

УДК 681.518

РОЛИК А.И.,
ВОЛОШИН А.В.,
ГАЛУШКО Д.А.,
МОЖАРОВСКИЙ П.Ф.,
ПОКОТИЛО А.А.

СИСТЕМА УПРАВЛЕНИЯ КОРПОРАТИВНОЙ ИНФОРМАЦИОННО-ТЕЛЕКОММУНИКАЦИОННОЙ ИНФРАСТРУКТУРОЙ НА ОСНОВЕ АГЕНТСКОГО ПОДХОДА

В статье разработана структура системы управления корпоративной информационно-телекоммуникационной инфраструктурой на основе агентского подхода с интеграцией методов активного и пассивного мониторинга и консолидацией централизованного и децентрализованного управления. Определены структуры серверного и агентского модулей, механизмы их взаимодействия. Разработан подход к распределению объектов мониторинга и управления по агентам, а также реактивный метод обновления состояний элементов ИТ-инфраструктуры.

The article describes the structure of the agent-based corporative information-telecommunication infrastructure control system, using the integration of active and passive monitoring methods and the consolidation of centralized and decentralized control methods. The article also determines the structure of the server and agent modules and algorithms of their interaction. The approach to the distribution of monitoring objects among the agents and the reactive method of updating their states were also developed.

Введение

В настоящее время показатели эффективности работы практически всех предприятий и организаций существенно зависят от эффективности функционирования корпоративных информационно-телекоммуникационных систем (ИТС). ИТС представляет собой совокупность информационной системы и телекоммуникационной сети и является базисом ИТ-инфраструктуры крупных предприятий или организаций. ИТС предоставляет аппаратно-программные средства для автоматизации бизнес-процессов, обеспечивает информационное и телекоммуникационное взаимодействие между функциональными и обеспечивающими системами, подразделениями, сотрудниками и пр. Для управления корпоративными ИТС используются универсальные фирменные или разрабатываются собственные адаптированные под потребности предприятия системы управления ИТ-инфраструктурой (СУИ) [1, 2]. СУИ реализуется в виде программного продукта, осуществляющего мониторинг состояния и анализ работоспособности как отдельных элементов ИТС, так и подсистем. Для этого СУИ производит мониторинг сетевого оборудования и решение задач сетевого управления, проверку функционирования и контроль работоспособности сетевых сервисов и приложений, управление аппаратными и программными элементами ИТС, и пр.

Увеличение стоимости информационно-телекоммуникационных ресурсов и технологий, существенная зависимость бизнеса от эффективности работы ИТС, постоянная модернизация и появление новых информационных технологий, ужесточение требований к эффективности и бесперебойности работы ИТС делают актуальной задачу совершенствования существующих или разработки новых перспективных СУИ.

Постановка проблемы

Для решения задач управления ИТС прежде всего необходимо получить информацию о состоянии компонентов ИТС. В настоящее время для решения этой задачи применяются различные методы мониторинга [3], среди которых можно выделить два наиболее простых и очевидных альтернативных метода – активный и пассивный мониторинги. В первом случае со стороны СУИ производится опрос элементов с целью получения значений параметров функционирования, во втором, – элементы ИТС путем отправки специальных сообщений самостоятельно информирует СУИ об изменении своего состояния в случае пересечения текущими значениями контролируемых параметров

установленных пороговых значений. Каждый из этих методов имеет свои преимущества и недостатки, и вполне логично, что перспективным является комбинированный – интегральный подход, сочетающий в себе преимущества обоих методов. При этом возникают задачи распределения функций между методами, интеграции результатов работы каждого из них, разработки структуры центральной и агентской частей СУИ. Это же касается и реализации в СУИ управляющих функций, которые могут в виде команд управления поступать из центра управления или генерироваться в агентах при отслеживании и реализации общей политики управления.

Разработке СУИ с интеграцией методов активного и пассивного мониторинга, а также реализации соответствующих управляющих функций и посвящена данная статья.

Анализ публикаций

В работе [1] определены требования, которым должна удовлетворять перспективная СУИ. Разработана структура серверной части, но не предложены принцип построения агентов и схема взаимодействия серверной и агентской частей СУИ.

В [3] методы мониторинга элементов ИТ-инфраструктуры рассматриваются без привязки к структуре СУИ.

В [4] рассматривается агентская схема построения СУИ, определена концепция мобильных агентов и мета-агентов, однако отсутствуют принципы построения и структура агентов, а также не представлены механизмы взаимодействия агентов с сервером.

Работа [5] посвящена интеграции методов активного и пассивного мониторинга элементов коммуникационной сети в GRID системах. В ней также не рассматриваются принципы построения агентов и вопросы их реализации.

Целью статьи является разработка системы управления информационно-телекоммуникационной инфраструктурой на основе агентского подхода с интеграцией активного и пассивного мониторинга и консолидацией методов централизованного и децентрализованного управления.

Структура СУИ

Современные СУИ представляют собой аппаратно-программные комплексы, предназначенные для мониторинга состояния, анализа работоспособности и управления аппаратными и программными средствами ИТС.

СУИ осуществляют:

— мониторинг и анализ функционирования программно-технических средств функциональных и технологических подсистем;

— организационное и автоматизированное управление функционированием программно-технических средств функциональных и технологических подсистем;

СУИ обеспечивают:

— контроль параметров функционирования аппаратного и программного обеспечения серверов и рабочих станций ИТС;

— анализ данных о функционировании подсистем и распределенных приложений;

— накопление, анализ и отображение результатов контроля состояния и работоспособности элементов ИТС;

— своевременное обнаружение, локализацию и управление устранением неисправностей в ИТС;

— прогнозирование критических состояний системы и предотвращение возникновения отказов в работе ИТ-системы на ранней стадии;

— интеграцию средств администрирования функциональных и технологических подсистем;

— выработку команд и сигналов управления техническими и программными средствами подсистем в автоматическом и ручном режиме для принятия решения системным администратором;

— оптимизацию планирования задач обслуживания системы с учетом загруженности ресурсов ИТС;

— оптимальное распределение общих вычислительных и коммуникационных ресурсов между бизнес-процессами с учетом их значимости;

— автоматизацию планирования деятельности персонала по управлению функционированием при техническом обслуживании и других регламентных работах;

— регистрацию и документирование событий в ИТС, действий администраторов по управлению функциональными и технологическими подсистемами.

Для реализации этих и множества других функций в СУИ используется централизованное или децентрализованное управление, активный или пассивный мониторинг. Как правило, СУИ, построенная по централизованному

принципу, использует метод активного мониторинга, когда сервер СУИ получает информацию от элементов ИТС путем запроса, и метод непосредственного управления элементами путем отправки соответствующих команд. Децентрализованная СУИ реализуется на интеллектуальных агентах, при этом основная часть задач мониторинга и управления реализуется агентом, а сервер собирает информацию о событиях, генерируемых агентами, и вырабатывает политику управления, обязательную для всех агентов.

Наибольший интерес представляет принцип построения СУИ, основанный на рассмотрении в тесной взаимосвязи методов централизованного и децентрализованного управления, активного и пассивного мониторинга. В этом случае производится рациональное разделение задач централизованного и децентрализованного управления и интегрированное использование активного и пассивного мониторинга. Именно такой принцип и реализован в предлагаемой СУИ.

В крупных, территориально распределенных на существенных расстояниях ИТС, содержащих большое количество элементов, СУИ целесообразно строить по иерархическому принципу (см. рис. 1).

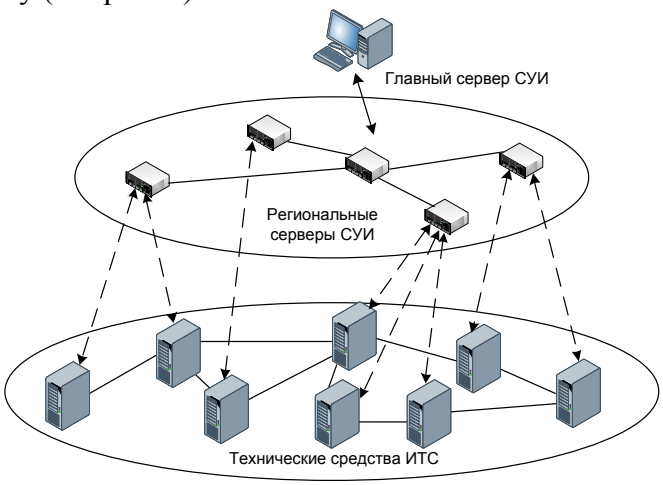


Рис. 1. Иерархия серверов СУИ

В этом случае на верхнем уровне располагается главный или центральный сервер СУИ, определяющий состояние крупных составляющих ИТС – функциональных и технологических подсистем, телекоммуникационной сети и пр., а на нижних уровнях – региональные или периферийные серверы СУИ, каждый из которых осуществляет мониторинг, анализ функционирования и управление элементами ИТС в своей географической или административной зоне ответственности. С нижних уровней на верхние поступают обобщенные данные мониторинга и информация о функционировании подсистем ИТС, расположенных в зоне ответственности периферийных серверов СУИ. С верхних на нижние – распространяются политики управления и макрокоманды. Все серверы СУИ управляются администраторами, которые находятся в состоянии иерархической соподчиненности в соответствии с организационной структурой департамента, обеспечивающего функционирование ИТ-инфраструктуры.

Взаимодействие и совместная работа компонентов СУИ осуществляется через телекоммуникационную сеть. Через сеть осуществляется взаимодействие между серверами СУИ и элементами функциональных и технологических подсистем ИТС, АРМ администраторов СУИ с администраторами функциональных, технологических подсистем и телекоммуникационной сети.

Специальное программное обеспечение разработанной СУИ состоит из трех отдельных программных компонентов – серверного, клиентского и агентского программных модулей.

Серверный программный модуль (сервер) обеспечивает взаимодействие между клиентами, предоставляет доступ к данным, обеспечивает их целостность и защиту, уведомляет администратора о критических или интересующих его событиях в ИТС, осуществляет планирование действий по мониторингу, анализ и управление элементами ИТС. Для сбора данных мониторинга и реализации управляющих воздействий сервер взаимодействует с элементами ИТС через агентские модули. Важной функцией сервера является синхронизация и обработка данных, а также уведомление подключенных агентов и клиентов о произошедших изменениях.

Клиентский программный модуль (клиент) реализует интерфейс администратора для настройки СУИ, отслеживания работоспособности и производительности элементов ИТС, контроля состояния функциональных и технологических подсистем и ИТС в целом, а также для введения команд управления при реконфигурации или проведении восстановительных мероприятий. Непосредственно мониторинг элементов ИТС и реализация управляющих воздействий осуществляется агентскими и серверным модулями, в то время как основная функция клиентского модуля – предоставление графического интерфейса для взаимодействия администратора с системой.

Агентский программный модуль (агент) контролирует элементы ИТС, анализирует их состояние и поведение. Кроме того, агенты используются при моделировании перспективного состояния, как отдельных элементов, так и всей ИТС. Это позволяет предвосхищать и предотвращать функциональные отказы еще при первых признаках их проявления, что, в свою очередь, повышает надежность функционирования ИТС. Агент запускается в виде отдельного сервиса, специализирующегося на выполнении индивидуального набора функций мониторинга и управления, конфигурируемых в зависимости от задач, решаемых агентом и объектами, которые он обслуживает. Собираемые агентом данные могут обрабатываться как на этом же, так и на других агентах, и вместе с результатами их обработки хранятся в локальной БД или отправляются на сервер.

Предлагаемая СУИ оперирует с объектами мониторинга и управления (ОМУ), а вся информация о структуре ИТС представляется в виде дерева ОМУ, при этом каждый ОМУ содержит сведения о конфигурации и текущем состоянии отдельного элемента ИТС. ОМУ – универсальное и обобщенное понятие, элемент логической модели ИТС, основная функциональная единица представления элементов ИТС в СУИ, в которой инкапсулированы те свойства и состояния элементов ИТС, которые имеют отношение к процессам управления ИТ-инфраструктурой. Состояние ОМУ, определяемое агентами СУИ, может зависеть от состояния других ОМУ. В зависимости от физического расположения ОМУ может реально обслуживаться только одним агентом, а для удаленной работы с ОМУ могут создаваться его копии на других агентах и клиентах. В таких случаях копии выступают заместителями реального ОМУ и переадресовывают ему все обращения. Относительно ОМУ определяются параметры мониторинга, механизмы выявления неисправностей и исполнения команд управления. В качестве ОМУ выступают не только физические и логические объекты, но и подсистемы, состояния процессов деятельности, бизнес-процессы и пр. Каждый компонент ИТС всех уровней иерархии содержит дерево ОМУ, листья которого – датчики контроля (сенсорная часть), а остальные вершины – ОМУ. Такой подход позволяет использовать универсальные методы работы СУИ с ОМУ, а также единый или несколько универсальных методов для оценки состояния ОМУ.

Программный сервер СУИ

Структура сервера СУИ приведена на рис. 2.

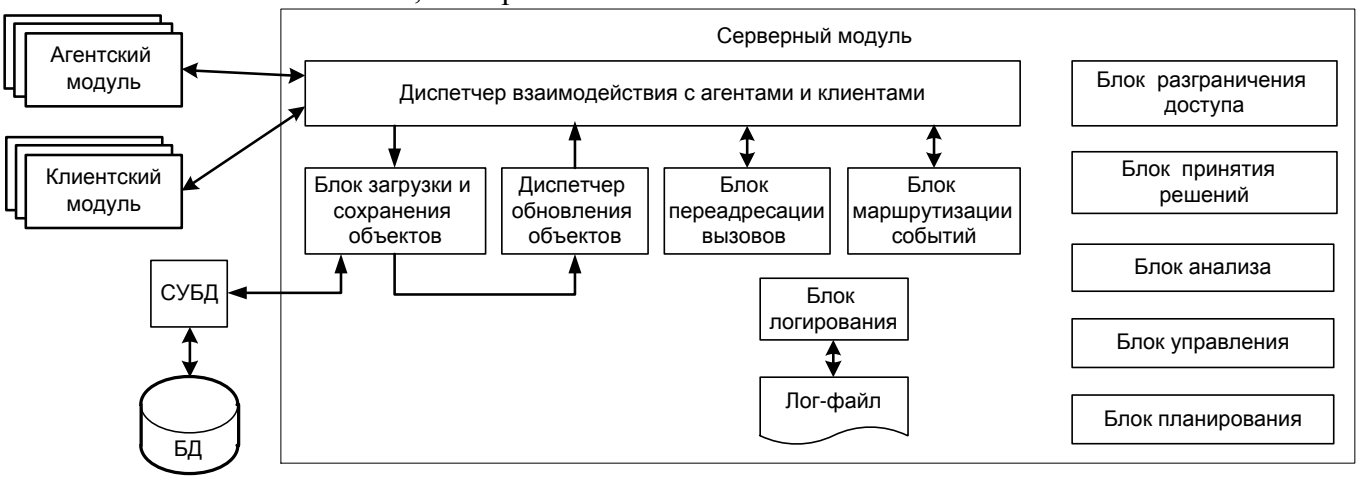


Рис. 2. Структура сервера СУИ

Сервер СУИ обеспечивает сбор данных мониторинга и управление ОМУ через агентские модули, взаимодействие с клиентскими модулями, предоставляет доступ к БД, обеспечивает защиту и целостность данных, осуществляет принятие решений по управлению ИТС, генерирует команды управления ОМУ и сигналы для администраторов, а также выполняет другие функции.

Сервер состоит из диспетчера взаимодействия с агентами и клиентами, а также следующих блоков: загрузки и сохранения объектов, диспетчера обновления объектов, переадресации вызовов, маршрутизации событий, разграничения доступа, принятия решений, анализа, управления, планирования и логирования.

Диспетчер взаимодействия с агентами и клиентами поддерживает взаимодействие и мар-

шрутизацию информационных потоков между программными компонентами СУИ, производит корректное подключение и отключение агентов и клиентов с рассылкой соответствующих уведомлений. Для получения доступа к серверу клиенты проходят процедуру регистрации, при которой проводится аутентификация и проверка прав. Любые запросы незарегистрированных клиентов, кроме запроса на регистрацию, автоматически отклоняются. Для поддержания связи подключенный клиент с заданной частотой путем передачи соответствующего запроса проверяет доступность сервера. В случае недоступности сервера, клиент осуществляет попытки повторного подключения. Сервер производит аналогичную проверку только в случае отсутствия в течение заданного времени запросов от клиента. При недоступности клиента сеанс разрывается. Для сокращения загруженности телекоммуникационной сети запрос доступности не производится при наличии активного информационного обмена между агентами и клиентами с сервером.

Блок загрузки и сохранения объектов осуществляет обработку запросов, кэширование ОМУ и запросов, а также реализует механизм синхронизации. Блок загрузки и сохранения объектов вместе с диспетчером обновления объектов производят загрузку и сохранение логических отображений ОМУ, а также обеспечивают целостность данных и своевременное уведомление клиентов о новых версиях объектов.

При получении некоторого запроса на загрузку ОМУ проверяется его наличие в кэше и, в случае отсутствия, запрос перенаправляется в БД. При этом предоставляются находящиеся в кэше готовые ОМУ, а отсутствующие конструируются из объектов, находящихся в БД, с последующим добавлением в кэш. Благодаря этому значительно повышается скорость выполнения запросов и обеспечивается гарантированное наличие в системе не более одной копии каждого ОМУ. Механизм синхронизации используется для получения от клиента обновленных версий ОМУ, а для применения изменений, внесенных данным клиентом, отслеживаются версии атрибутов ОМУ и его связей. Данный механизм позволяет редактировать один и тот же объект на разных клиентах. После выполнения синхронизации все затронутые изменениями ОМУ сохраняются в БД и передаются диспетчеру обновления объектов для оповещения подключенных клиентов.

Для обеспечения целостности и актуальности данных реализован механизм, отвечающий за своевременное оповещение клиентов о произошедших изменениях структуры дерева ОМУ и его составляющих. Для каждого подключенного клиента диспетчер строит выборки объектов из списка обновленных. Построение выборок сводится к решению NP-полной задачи, т. к. для решения задачи определения необходимости оповещения клиента об обновлении некоторого ОМУ, необходимо решить аналогичную задачу для каждого из связанных с ним объектов. Затраты на определение необходимости оповещения клиента минимальны для объектов, связанных непосредственно с клиентом, и велики для объектов, привязанных к клиентам через цепь других объектов. Поэтому нецелесообразно проводить выборку объектов некоторых типов, отправляя все измененные объекты клиенту. Для снижения сетевого трафика клиентам передаются не измененные объекты, а список их идентификаторов, а клиент запрашивает обновление только тех объектов, которые им используются.

Блок переадресации вызовов и маршрутизации событий, а также диспетчер обновления объектов позволяют клиентам работать с удаленными объектами, своевременно получая уведомления об изменениях их состояния и структуры.

При построении систем клиент-серверного взаимодействия, при котором клиенты взаимодействуют друг с другом только через сервер, возникает проблема обращения к ОМУ, месторасположение которого неизвестно. ОМУ может находиться только на одном агенте, а все его воплощения (копии) на других машинах являются заместителями, поэтому необходимо неявно перенаправлять вызов некоторой функции объекта-копии реальному объекту. При этом обращение к удаленным объектам осуществляется как к локальным, а все механизмы, осуществляющие переадресацию, скрываются от администратора СУИ. Это обеспечивает взаимную анонимность вызывающего и конечного клиентов, поскольку принятие решения о правомерности такого вызова и нахождение конечного клиента осуществляет сервер. Если при обращении к объекту выясняется, что он является копией, то задействуется встроенный механизм переадресации, который обращается к серверу, где происходит определение конечной машины, на которой находится объект, и ее доступность, после чего запрос перенаправля-

ется конечному клиенту. Конечный клиент возвращает результаты запроса на сервер, который, в свою очередь, перенаправляет результаты запросившему клиенту. В случае недоступности сервера или конечного клиента, ошибки или отказа в доступе, клиенту возвращается описание причины неудачного выполнения запроса.

Для обеспечения работы механизма реактивного обновления состояний ОМУ вне рамок одного агента должна осуществляться своевременная передача уведомлений о событиях, произошедших на данном агенте, другим агентам, имеющим зависящие ОМУ. Для подписки на события и их передачи агенты не обязаны знать о местонахождении конкретных ОМУ, обращаясь с находящимися у них копиями, как с реальными ОМУ. Реализация процесса подписки показана на рис. 3.

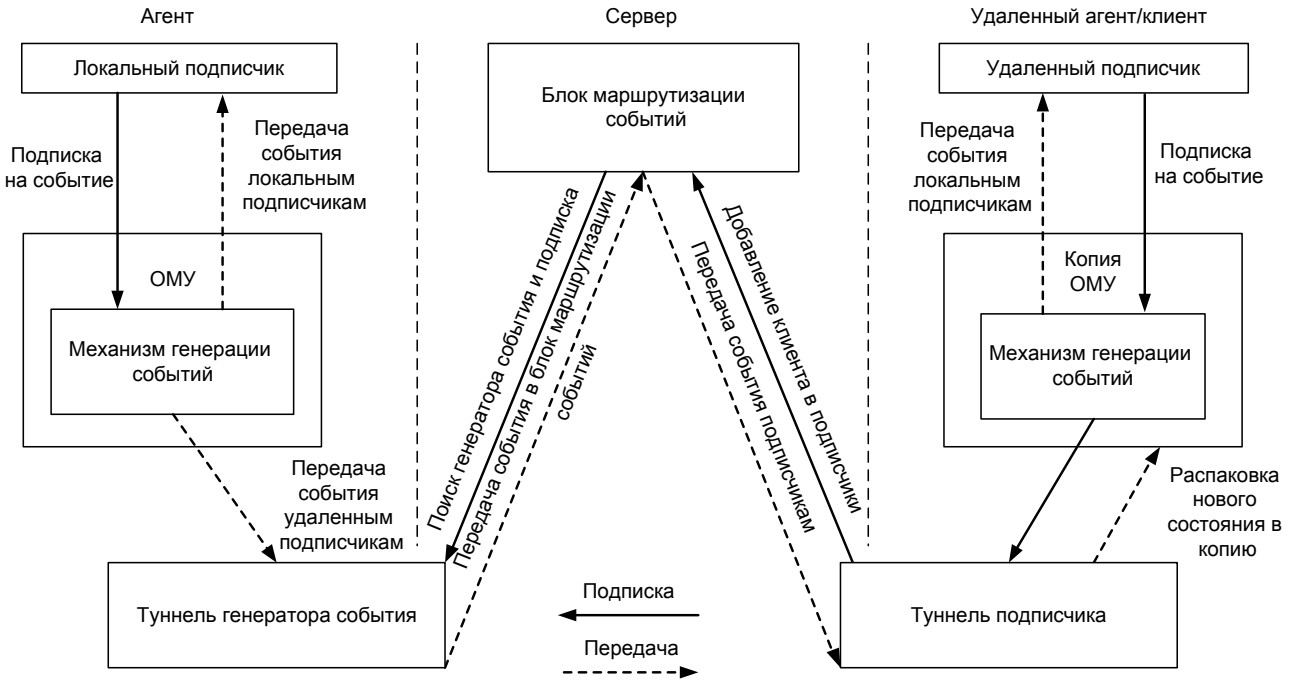


Рис. 3. Процесс подписки на удаленное событие и его вызова

Если при подписке на событие определяется, что данный объект является копией (заместителем), производится подписка на получение данного события на локальном туннеле. Туннель представляет собой общий для данного клиента механизм подписки, отписки и маршрутизации событий. Туннель переадресует подписку на серверный блок маршрутизации событий, где существует реестр событий и клиентов – подписчиков и источников. Сервер определяет клиента, которому принадлежит генерирующий событие объект, и добавляет кортеж (объект–клиент-источник–клиент-подписчик) в реестр событий. Далее диспетчер взаимодействия с агентами и клиентами определяет, подключен ли клиент-источник к серверу, и отправляет ему запрос на отсылку подписываемого события.

Подписка автоматически восстанавливается после разрывов сеансов связи. При отключении клиента-подписчика блок маршрутизации событий автоматически аннулирует все его подписки, а после восстановления подключения клиент заново осуществляет подписку. При отключении клиента-источника, подписки на его события остаются в реестре, и при повторном подключении клиента к сети блок маршрутизации событий сообщает об ожидающих его подписках. При возникновении события обновления состояния, агент делает рассылку локальным подписчикам и, в случае наличия удаленных подписок, происходит отправка этого события и сопутствующих атрибутов на серверный блок маршрутизации событий, который производит его ретрансляцию всем клиентам-подписчикам. Получив событие, клиентский туннель распаковывает его атрибуты в объект-копию, после чего генерирует аналогичное событие на локальной машине от имени объекта-копии. Обычно в качестве атрибутов события отправляется текущее значение или обновленное состояние объекта. Таким образом, использованные механизмы и подходы позволяют реализовать поддержку реактивного обновле-

ния состояний ОМУ и своевременное обновление состояний копий ОМУ на всех подписанных клиентах.

Блок разграничения доступа проверяет права агентов и клиентов при попытках получения доступа, редактирования или удаления объектов и связей, редактирования прав, вызова удаленных функций некоторого объекта или подписки на события об изменении его состояния.

Серверный блок анализа в отличие от агентского, который анализирует состояния отдельных ОМУ, анализирует состояние ИТС в целом. Блок анализирует полученные от агентов данные мониторинга и анализа параметров с использованием различных статистических методов. В частности, выявляются тренды поведения, проводится работа по архивации данных мониторинга и пр. Проактивный мониторинг осуществляется путем анализа и определения тенденции динамики изменения состояния определенных параметров функционирования системы и прогнозирования возможных будущих состояний.

На основании результатов анализа блок принятия решений вырабатывает определенные варианты воздействия на ИТС в соответствии с используемыми критериями, политиками и алгоритмами. Изначально администратор вручную задает алгоритмы поведения системы при возникновении определенных видов неисправностей, а в процессе обучения, запоминая и анализируя действия администратора, блок принятия решений отбирает наиболее часто использующиеся решения, и предлагает их администратору, который в свою очередь может на основании этих рекомендаций совершенствовать алгоритмы поведения системы. Принятые решения могут быть следующих типов: уведомление администратора о существующей или возможной в будущем неисправности; рекомендации администратору по устранению неисправности; автоматизированное принятие решения или предложение выбора из готовых вариантов решения; автоматическое устранение неисправности или реконфигурация ИТС.

Блок управления воздействует на элементы ИТС посредством передачи управляющих запросов соответствующим им ОМУ. При этом блок переадресации вызовов перенаправляет данный запрос агенту, который может непосредственно влиять на поведение данного ОМУ. Блок управления работает в ручном и автоматическом режимах, что позволяет управлять ИТС посредством клиентского интерфейса, а также приводит в исполнение принятые блоком принятия решений автоматические действия по реконфигурации ИТС, перераспределению ограниченных ресурсов [6], управлению информационными потоками [7] и пр.

Блок планирования позволяет выполнять задачи с заданным периодом или в заданное время, а блок логирования сохраняет историю системных событий.

Программный агент СУИ

Мониторинг компонентов ИТС осуществляется посредством программных агентов СУИ. Агенты СУИ – универсальные, гибкие инструменты, с модифицируемым набором функций, позволяющие контролировать элементы ИТ-инфраструктуры, анализировать их поведение и состояние. Каждый агент может специализироваться на выполнении индивидуального набора функций мониторинга и управления, задаваемого администратором, в зависимости от возлагаемых на агента задач.

Модель программного агента СУИ базируется на пяти основных элементах: сущности, действия, отношения, сообщения, проверочные функции. Эти элементы обеспечивают функционирование агентского модуля, а также связи между агентами и сервером или клиентами.

Каждый агент содержит **сущность** – логический объект, активный элемент агента, выполняющий некоторое подмножество его функций, обладающий набором свойств, характеризирующийся определенным шаблоном поведения и способностью взаимодействовать с другими сущностями, в роли которых могут выступать логические объекты агентов, клиентов или сервера СУИ.

Поведение сущностей определяется **действиями** – набором функций, которые определяют работу модели сущности и протокол взаимодействия с другими сущностями. В первом случае инициализируется библиотека функций, которая контролирует и определяет внутреннее поведение агента. Во втором – определяется стратегия внешнего взаимодействия и правила передачи и обмена информацией. Действия между различными сущностями возможны исключительно в случае установленных доверительных отношений между ними.

Отношения определяют возможность взаимодействия между собой агентов и могут быть следующих видов: доверительные и отношения контроля.

Доверительные отношения устанавливаются при необходимости взаимного обмена информацией. Такие отношения используются между группой агентов в случае общего мониторинга и анализа состояния комплексных объектов ИТ-инфраструктуры.

Отношения контроля устанавливаются между двумя сущностями и позволяют одной из них (контролеру) запрашивать и получать необходимую информацию от второй (подконтрольной) сущности. Такие отношения в первую очередь устанавливаются между сервером и агентами СУИ (см. рис. 4) при централизованном управлении.

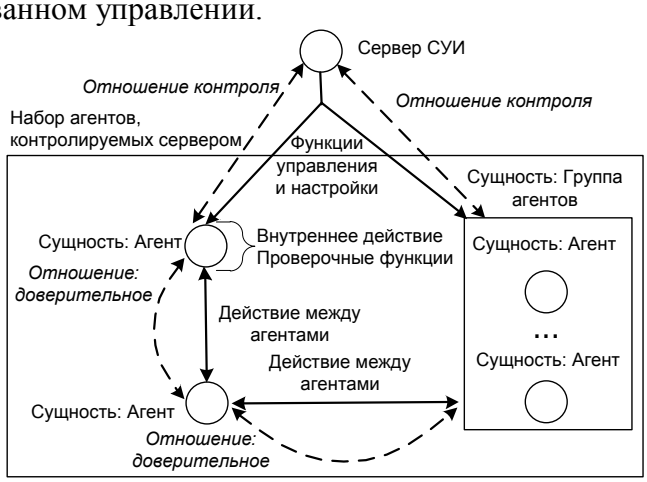


Рис. 4. Связи между сущностями

Агент обменивается информацией с другими агентами, а также с сервером посредством **сообщений**. Сообщения позволяют передавать необходимые данные и осуществлять удаленное управление. Они могут быть двух видов: служебные и системные.

Системные сообщения – сообщения, которые генерируются СУИ, для автоматического или планового обмена системной информацией и для автоматического модифицирования конфигурации отдельных сущностей. Они используются для мониторинга элементов ИТС с возможностью последующего анализа.

Служебные сообщения генерируются администратором СУИ или планировщиком для удаленного управления элементами ИТС или получения информации об их состоянии. С помощью сообщений реализован механизм оперативного реагирования на возможные неисправности.

Для контроля функционирования отдельного агента или группы агентов используются **проверочные функции**. Проверочные функции могут запускаться агентом для проведения самоконтроля и вызываться со стороны сервера. Посредством проверочных функций строится график работы как отдельных элементов, так и всей ИТС, что позволяет в режиме реального времени автоматически определять возможные «узкие места» в производительности системы и перераспределять ресурсы для оперативного решения возникающих проблем. На основе статистической обработки результатов выполнения проверочных функций строятся графики изменения значений ключевых показателей работоспособности как отдельных элементов, так и ИТС в целом, что позволяет автоматически выявлять неполадки в работе ИТС и перераспределять ресурсы для оперативного реагирования на возникающие проблемы.

Структура агентского модуля

Структура агентского модуля СУИ приведена на рис.5.

Рис. 5. Структура агентского модуля

Агентский модуль состоит из исполнительной, сенсорной, элементной частей и блока взаимодействия.

Исполнительная часть обеспечивает штатное функционирование агента, осуществляет управление агентом, планирование действий, вызов функций мониторинга и состоит из блоков мониторинга, управления, планирования и обработки информации.

Блок мониторинга обеспечивает мониторинг ОМУ, находящихся в зоне ответственности агента. Мониторинг может осуществляться стандартными методами с использованием протоколов SNMP или ICMP и инструментария WMI, или с помощью предложенной авторами и реализованной в предлагаемой СУИ технологии DALLF (Dynamic Auto Link Library Function), использующей рефлексию типов. Стандартные средства мониторинга агенты СУИ используют для получения информации с локального элемента ИТС, на который установлен агент, и, при наличии доверительных отношений, с удаленных ОМУ. Расширяемые методы мониторинга, представленные в виде DALL (Dynamic Auto Link Library) функций, выбираются администратором для каждого агента из библиотеки шаблонов поведения. Каждый элемент шаблона соответствует определенному типу агента в зависимости от возложенных на него функций и состоит из набора DALL функций.

Исполнительная часть обеспечивает управление агентом, планирование выполнения функций мониторинга и обработку информации, которая поступает от сенсорной части. При этом контролируются показатели качества функционирования отдельного элемента ИТС, на котором расположен агент и, в случае выхода их за критические пределы, производится локализация потенциальной неисправности. Блок управления обеспечивает работу с деревом ОМУ, к которому привязан агент, и может устранить возникшую, или потенциальную неисправность при наличии необходимых для устранения неисправности алгоритмов, о чем информируется сервер СУИ, который может скорректировать принятые агентским блоком управления решения. При отсутствии необходимых для устранения неисправности алгоритмов, агент уведомляет сервер СУИ о локализированной неисправности, и сервер принимает решение по ее устранению. При недоступности сервера агент принимает решения по устранению неисправностей самостоятельно.

Блок планирования обеспечивает контроль запускаемых функций проверки, обеспечивает непрерывный процесс мониторинга и распределяет ресурсы между всеми решаемыми агентом задачами. Кроме того, для реализации механизмов пассивного мониторинга, можно подписывать DALL функции на внешние системные события.

Блок обработки информации осуществляет обработку данных, поступающих с сенсорной части и от блока мониторинга, и путем статистического анализа выявляет закономерности изменения значений измеряемых параметров и определяет соответствие состояний ОМУ предопределенным политикам, что необходимо при использовании децентрализованного управления.

Элементная часть состоит из дерева ОМУ, которые закреплены за агентом. Состояние каждого ОМУ оценивается по ряду параметров. Блок управления передает ОМУ управляющие команды для поддержания контролируемых параметров в заданных пределах. Расчетные значения параметров поступают на сенсорную часть.

Сенсорная часть состоит из набора датчиков, счетчиков, указателей и пр., которые собирают данные для определения состояний ОМУ, контролируемых агентом, и передает полученные данные в блок обработки информации.

Блок взаимодействия осуществляет информационный обмен между агентами и агентов с сервером. Связь между агентскими модулями может осуществляться напрямую, при установлении доверительных отношений между агентами, или через канал связи агентов в противном случае (см. рис 6.).

Рис. 6. Связи между агентами

Канал связи агентов обеспечивает связь между агентами, когда между ними нет прямых доверительных отношений. При этом создается новое сообщение для передачи, которое поступает в стек сообщений, где обрабатывается посредством планировщика. Каждое сообщение имеет свой уровень важности от 0 до 9 (0 – наивысший уровень). Сообщения с уровнем важности 0 всегда передаются напрямую неза-

висимо от уровня доверенности между агентами.

Каждый программный агент СУИ имеет определенный жизненный цикл, на протяжении которого он исполняет возложенные на него функции, и может находиться в одном из следующих состояний (см. рис. 7): инициализации (идентификатор: 0), активности (идентификатор: 1), перемещения (идентификатор: 2), ожидания (идентификатор: 3), паузы (идентификатор: 4) и уничтожения (идентификатор: 5).

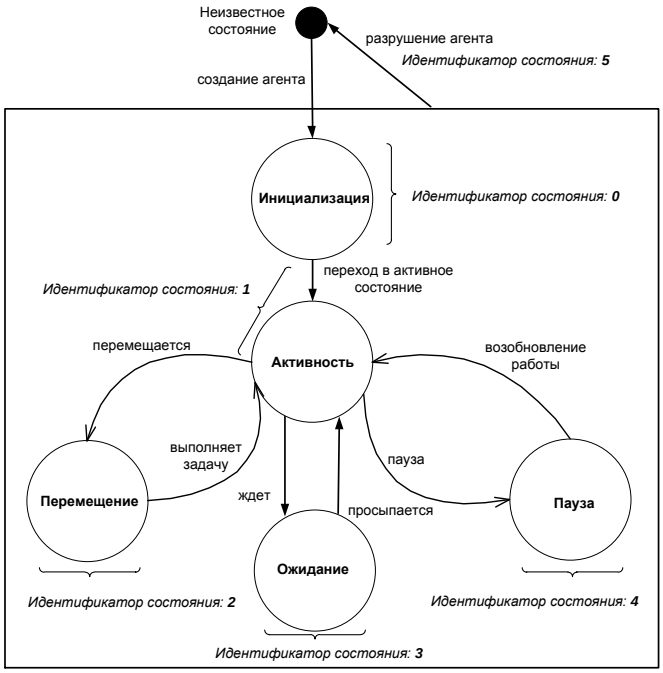


Рис. 7. Жизненный цикл агента

Первоначально программный агент находится в состоянии инициализации, в котором происходит инициализация набора функций самоконтроля, задание параметров для планировщика и загрузка дерева ОМУ. После завершения инициализации агент переходит в состояние активности. В этом состоянии агент запускает планировщик и работает с деревом ОМУ, анализирует состояние всех составных элементов и может производить начальный анализ полученных данных. На данном этапе главной задачей агента является контроль состояний ОМУ и передача необходимой информации на сервер. Если за некоторый промежуток времени не происходит изменения параметров подконтрольных ОМУ, то агент переходит в состояние ожидания.

Главной задачей планировщика становится проверка возобновления активности подконтрольных ОМУ. Если долгое время параметры объектов не изменяются, то по указанию сервера агент может переместиться на другую машину, на которой ОМУ возобновили свою деятельность или на котором повысилась расчетная нагрузка. В этом случае агент переходит в состояние с идентификатором 2 – состояние перемещения и прекращает активность, для чего приостанавливает внутренний планировщик и сворачивает активность дерева ОМУ. Сервер СУИ готовит туннель для перемещения агента и служебным сообщением передает его идентификатор. По идентификатору туннеля блок управления агента находит нужный путь перемещения и мигрирует к заданному месторасположению в ИТС.

Если услуги отдельно взятого агента перестают быть полезными для СУИ, то сервер может принять решение о приостановке работы агента (идентификатор: 4) или разрушить его.

Особенности реализации распределенных агентов

В СУИ использован подход к распределению ОМУ по агентам, при котором все ОМУ закрепляются за агентами в зависимости от производительности агентских машин и пропускной способности линий связи между ними. Логические ОМУ могут обрабатываться на агентах, физическое расположение которых может отличаться от физического расположения ОМУ. К конкретным агентам привязываются только ОМУ, управляющие и мониторинговые функции которых выполняются непосредственно на целевой машине, остальные ОМУ могут обслуживаться любыми агентами. Это позволяет получать структуры, которые просто изменяются и гибко настраиваются.

Такой подход позволяет решить проблему поддержки слабых машин, которые могут не обладать достаточной производительностью и быстродействием для анализа всего дерева находящихся на них ОМУ, а также проблему безопасности, когда возникает необходимость скрыть от агента структуру дерева ОМУ или отдельные его элементы.

Агенты могут быть вынесены за пределы, например, сервера приложения, с тем, чтобы не создавать процессору сервера дополнительную нагрузку решением задач мониторинга и анализа ОМУ. Реализация предлагаемого подхода позволяет также разделить серверный агент на несколько физических машин, что позволяет существенно разгрузить сервер, распараллелить обслуживание возложенных на него ОМУ, и тем самым повысить надежность СУИ.

Упрощенный пример распределения дерева ОМУ по агентам отображен на рис. 8, где тон-

кий агент обслуживает только те ОМУ, для обслуживания которых необходимо их присутствие на целевой машине, толстый агент обслуживает все логические ОМУ своей машины, региональный серверный агент обслуживает ОМУ своей сети, главный серверный агент обслуживает ОМУ всей системы.

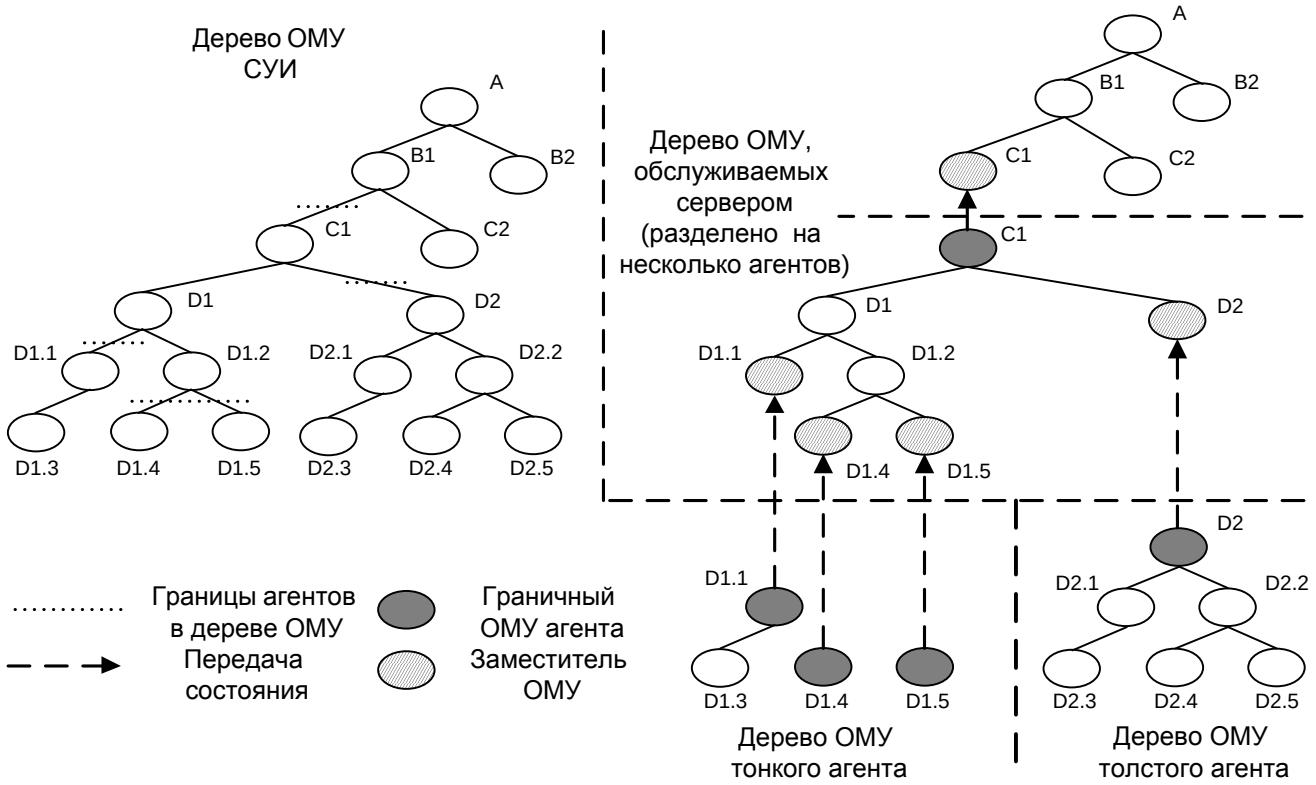


Рис. 8. Пример распределения дерева ОМУ по агентам

Предлагаемый подход позволяет легко реализовать толстые, тонкие и мигрирующие агенты, применяя один и тот же программный модуль, но по-разному задавая граничные ОМУ. Тонкий агент занимается только съемом данных мониторинга с целевой машины и выполнением управляющих функций на целевой машине, а анализ и принятие решений производятся на сервере СУИ. В мигрирующих агентах граница разделения в дереве ОМУ перемещается во время работы СУИ, перенося группы ОМУ с одной машины на другую.

Для реализации предложенного подхода к распределению ОМУ по агентам необходимо, чтобы каждый конкретный ОМУ, вне зависимости от его физического расположения, логически находился (обрабатывался) только на одном агенте, и в то же время обеспечивалась оперативная передача информационных и управляющих сигналов между граничными ОМУ разных агентов. Для этого введены понятия реального ОМУ и его заместителя (копии) соответственно для ОМУ, которые обрабатываются на данном агенте, и тех, которые необходимы на данном агенте для обеспечения взаимодействия с реальными ОМУ. Объекты-заместители, используя встроенный механизм переадресации вызовов, передают запросы реальному объекту, а также синхронизируют свое состояние в соответствии с принятыми событиями.

Программное представление объектов мониторинга и управления

С точки зрения мониторинга, оценки состояния и управления ОМУ, ИТС удобно представлять в виде дерева, узлами которого являются ОМУ, а связи отображают функциональные зависимости. Свойства, наиболее значимые с точки зрения мониторинга и оценки элементов ИТ-инфраструктуры, составляют набор параметров ОМУ.

В дереве ОМУ могут присутствовать одинаковые или схожие ветви. При построении дерева администратор тратит время на дублирование информации, поэтому возникает необходимость стандартизированной обработки однотипных ОМУ. Для этого в предлагаемой СУИ гомогенные ОМУ объединены в шаблоны, которые определяют набор параметров, функций и связей ОМУ данного шаблона. Механизм шаблонов позволяет настроить поведение и структуру шаблонного ОМУ и при создании

новых ОМУ использовать настройки шаблона, при необходимости уточняя и дополняя их. При этом можно легко менять поведение и структуру группы ОМУ путем изменения настроек их шаблона.

Ряд механизмов взаимодействия модулей СУИ инкапсулированы непосредственно в программном отображении ОМУ. Это значительно упрощает работу с объектами в различных программных модулях и позволяет взаимодействовать с удаленными объектами посредством их локальных копий-заместителей, и производить описанные ниже действия без использования отдельных блоков на каждом из программных компонентов СУИ.

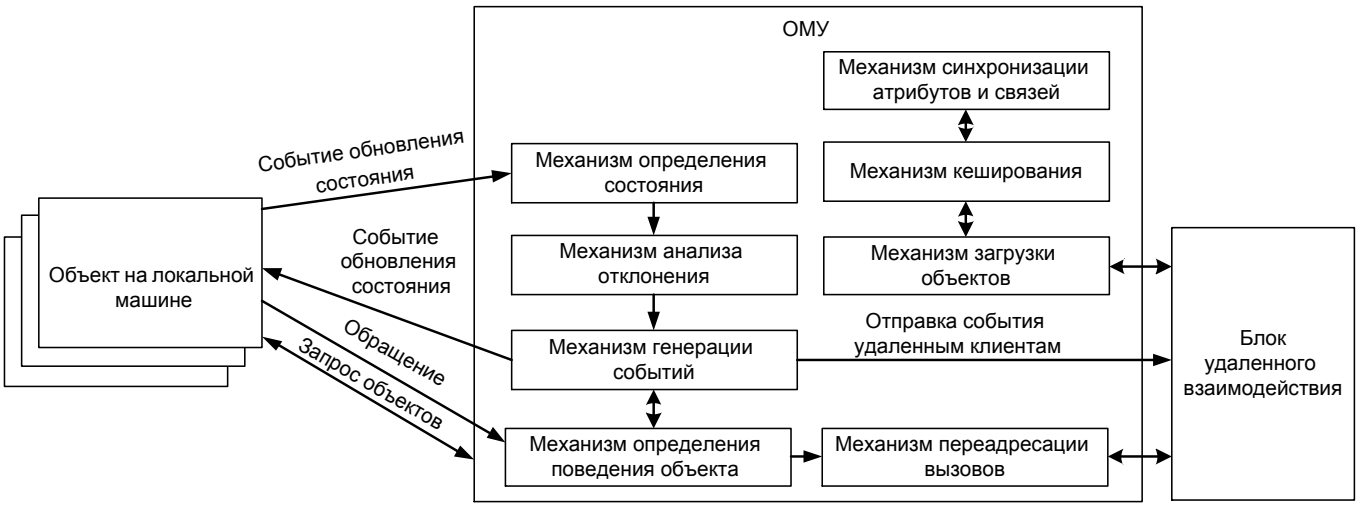


Рис. 9. Инкапсулированные в ОМУ механизмы

В программном отображении ОМУ инкапсулированы следующие механизмы (рис. 9):

— механизм синхронизации атрибутов и связей – осуществляет синхронизацию текущей версии ОМУ с обновленной, полученной с сервера, либо наоборот, сохраняемой клиентом;

— механизм кэширования – обеспечивает хранение используемых на данной машине ОМУ и наличие не более одной копии объекта в системе. В случае получения новой версии уже существующего объекта происходит его синхронизация;

— механизм определения поведения объекта – определяет, является ли копия объекта на данной машине реальной (оперирующей с ОМУ напрямую) или заместителем (переадресовывающим команды на реальный удаленный ОМУ в зависимости от истинного расположения отображаемого им ОМУ), и обеспечивает в системе наличие не более одной реальной копии каждого объекта;

— механизм определения состояния – определяет текущее состояние ОМУ в зависимости от значений мониторинговых показаний или состояний влияющих на него ОМУ, используя различные передаточные функции;

— механизм анализа отклонения – принимает решение о необходимости уведомления зависимых ОМУ об изменении своего состояния;

— механизм генерации событий – сообщает зависимым ОМУ-подписчикам об изменении своего состояния;

— механизм переадресации вызовов – для объектов-заместителей переадресует вызов управляющих функций на сервер, где определяется местонахождение реального объекта;

— механизм загрузки объектов – динамически подгружает необходимые объекты, связанные с данным, используя в качестве источника локальную или удаленную (серверную) базу данных.

Механизм реактивного обновления состояний ОМУ

В ИТС состояние ОМУ N-го иерархического уровня зависит от состояния ОМУ (N-1)-уровня, поэтому возникает задача определения необходимой частоты обновления состояний всех ОМУ.

Если осуществлять пересчет состояний ОМУ по указаниям планировщика, то каждому ОМУ N-уровня необходимо задавать период обновления, а при пересчете использовать текущие значения состояний ОМУ (N-1)-уровня, частота обновления которых может отличаться от частоты опроса ОМУ N-уровня. Недостатки такого метода очевидны: сложность синхронизации обновления сложных иерархий во времени, достаточно медленное реагирование на измене-

ние состояния ОМУ (N-1)-уровня и использование устаревших значений, при низкой частоте опроса, либо излишне высокая требовательность к вычислительным ресурсам при частом опросе.

Реализованный в СУИ механизм обновления построен на принципах реактивного программирования. Планировщик управляет периодом обновления состояний только для ОМУ нижних уровней. ОМУ верхних уровней реагируют на поток событий изменением состояний и не нуждаются в использовании планировщика. При обновлении состояния ОМУ (N-1)-уровня генерирует событие, на уведомление о котором подписываются зависящие от него ОМУ N-уровня. Механизм анализа отклонения ОМУ обладает свойством гистерезиса, что предотвращает генерацию события и излишне частые срабатывания, а также обеспечивает остановку волны перерасчета состояний ОМУ вверх по дереву, если состояние иерархически нижерасположенных ОМУ изменилось незначительно. Кроме того, для пассивного мониторинга в СУИ реализована возможность подписки функций ОМУ на внешние системные события.

Механизм реактивного обновления обеспечивает своевременное обновление состояний всех ОМУ, уменьшает количество избыточных пересчетов состояния верхних уровней иерархии ОМУ, позволяет гибко настраивать чувствительность системы к изменениям состояний ОМУ.

Выводы

Предложенная система управления корпоративной информационно-телекоммуникационной инфраструктурой реализована на основе агентского подхода с интеграцией методов активного и пассивного мониторинга, и консолидацией методов централизованного и децентрализованного управления.

Предложены структуры серверного и агентского модулей СУИ. Рассмотрены особенности реализации мобильных агентов. Разработана схема взаимодействия серверного модуля с агентским и клиентским модулями. Реализованы механизмы поддержания на всех программных компонентах системы актуальных версий объектов мониторинга и управления, позволяющие оперативно отслеживать их состояние и удаленно взаимодействовать. В системе использован реактивный подход обновления состояний ОМУ, что обеспечивает своевременное обновление состояний всех ОМУ, снижает количество избыточных пересчетов состояния верхних уровней иерархии ОМУ. Функциональность системы управления может легко наращиваться за счет использования динамических библиотек, что расширяет возможности ее применения. Благодаря разработанному подходу к распределению объектов мониторинга и управления по агентам нагрузка системы управления на вычислительные ресурсы и телекоммуникационную сеть может перераспределяться.

Предложенная система управления SmartBase ITS Control разработана в НТУУ «КПИ», прошла апробацию в двух системах специального назначения и может быть использована в различных системах управления корпоративными информационно-телекоммуникационными системами для повышения надежности и эффективности использования информационно-вычислительных и телекоммуникационных ресурсов, а также повышения результативности применения информационных технологий.

Список литературы

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 информационной инфраструктурой транспортного предприятия / С.Ф. Теленик, А.И. Ролик, П.Ф. Можаровский, А.В. Волошин // Автомобільний транспорт: зб. наук. праць: Вип. 25. – Харків.: ХНАДУ, 2009. – С. 242-245.
3. Ролик А.И. Управление устранением неисправностей в ИТ-системах / А.И. Ролик, Ю.С. Тимофеева, Н.И. Турский // Вісник НТУУ «КПІ». Інформатика, управління та обчислювальна техніка. – К.: «ВЕК+», 2008. – № 49. – С. 94-107.
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. 113-124.

5. Andreozzi S. On the Integration of Passive and Active Network Monitoring for Grid Systems / S. Andreozzi, A. Ciuffoletti, A. Ghiselli, D. Antoniades, E. P. Markatos, M. Polychronakis, P. Trimintzios // Integrated Research in GRID Computing, Springer-Verlag, January 2007. – p. 147–161.
6. Ролик А.И. Модель управления перераспределением ресурсов информационно-телекоммуникационной системы при изменении значимости бизнес-процессов// Автоматика. Автоматизация. Электротехнические комплексы и системы. – 2007. – №2 (20). – С. 73–82.
7. Ролик А.И. Система управления информационными потоками в корпоративной IP-сети / А.И. Ролик, В.А. Иосифов // Адаптивні системи автоматичного управління. Міжвідомчий науково-технічний збірник. – Дніпропетровськ: Системні технології, 2009. – Вип. 14 (34). – С. 73-85.

УДК 681.5.015

ПИСАРЕНКО А.В.,
ЦВЄЛОДУБ А.О.

РОЗРОБКА ІНСТРУМЕНТАЛЬНИХ ЗАСОБІВ MATLAB/SIMULINK ДЛЯ ПРОЕКТУВАННЯ КЕРУЮЧИХ ПРИСТРОЇВ СИСТЕМ УПРАВЛІННЯ НА БАЗІ ПЛІС

У статті розглядається проблема створення інструментальних засобів для використання методології швидкого прототипування та сумісного моделювання при розробці керуючих пристроїв систем управління. Детально розглянуто мету, засоби і переваги підходу швидкого прототипування. Також розглянуто зв'язок швидкого прототипування та сумісного моделювання, переваги їх використання у комплексі. Як рішення запропоновано універсальний спосіб зв’язку між середовищем моделювання та апаратним прототипом.

Результатом роботи є розроблена програма, що дозволяє використовувати апаратні прототипи керуючих пристроїв, створені на ПЛІС, для сумісного моделювання з моделлю системи створеної у програмному пакеті Matlab.

This article addresses the problem of creating a design environment for rapid prototyping and co-simulation to be used in field of control systems. Purposes, techniques and advantages of rapid prototyping approach are covered in details. As well a co-simulation approach is covered in conjunction with rapid prototyping; the advantages of using them in combination are presented. A universal way of communication between a simulation environment and a prototype is proposed as a part of the solution.

The study resulted in a development of a program that allows hardware prototypes implemented in FPGA to be used for co-simulation with a model of a system created in MATLAB/Simulink environment.

Вступ

Метою підходу швидкого прототипування є створення прототипів нових виробів на ранніх етапах розробки. Прототипи, зазвичай, реалізують тільки частину потрібної функціональності, або не відповідають усім вимогам до кінцевого виробу, щоб прискорити процес їх створення. Використання прототипів дозволяє, по-перше, виявити можливі недоліки та виправити їх на ранніх стадіях розробки, а по-друге, скорегувати подальші плани розробки в залежності від змін у вимогах проекту, які можуть бути сформульовані на основі випробувань прототипу. При такому підході розробка ведеться ітеративно, що додає гнучкості процесу. Загалом це пришвидшує цикл розробки кінцевого продукту та дозволяє зменшити час потрібний для завершення проекту, що є вигідним при дрібносерійному виробництві та виробництві у високо конкурентних галузях промисловості. Різницю між традиційним підходом та підходом з використанням швидкого прототипування можна побачити на рисунку 1.

Розглянемо технологію швидкого прототипування у зв’язку з розробкою електронної апаратури. Є декілька засобів, які можуть бути використані для створення прототипів електронної апаратури. Це мікроконтролери, програмовані логічні пристрої та програмовані логічні інтегральні схеми.

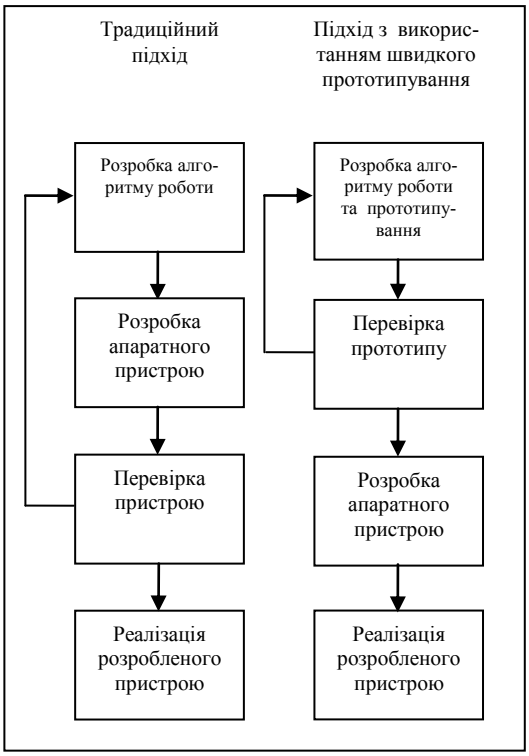


Рис. 1. Схеми традиційного підходу та підходу з використанням швидкого прототипування

Від спеціалізованих інтегральних схем, які зазвичай використовуються при виробництві кінцевої продукції, їх відрізняє одна спільна особливість – можливість швидкої зміни функціональності, що і є необхідним при створенні прототипів. Завдяки стабільному прогресу у області надвеликих інтегральних схем та облас-

ті засобів проектування електронних пристроїв, ПЛІС досягли рівня при якому їх можна вважати передовим засобом розробки електронних пристроїв. Вони мають переваги над іншими засобами прототипування, маючи більшу швидкість та ефективність роботи відносно до вартості. Саме тому, в останні роки ПЛІС використовуються, як у промисловості, для створення прототипів, так і в науці, для дослідження ефективності їх використання у різних областях застосування.

Саме можливість запрограмувати у ПЛІС довільну функціональність, поєднана з високою швидкодією отриманого пристрою робить їх надзвичайно вигідними для створення регуляторів для систем управління та перевірки роботи складних алгоритмів управління.

Отримавши прототип пристрою необхідно перевірити його функціонування та виявити недоліки у його роботі. Для цього необхідно перевірити його працездатність у робочих умовах, наприклад, використавши у справжній системи або макеті системи. Однак, коли існує досить детальна модель системи, розроблена у спеціалізованих програмних пакетах, або при занадто дорогій розробці макету системи, можливо провести моделювання програмної моделі системи разом із розробленим прототипом.

Такий підхід до моделювання отримав назву сумісного моделювання. Він полягає у поєднанні обчислювальних засобів різного походження, для створення загальної моделі. Апаратний блок обчислень, наприклад, створений раніше прототип, за допомогою каналу зв'язку підключається до моделі складної системи, і, коли потрібно провести обчислення закладені у апаратній частині, через канал зв'язку на неї надсилаються вхідні данні, пристрій проводить складні обчислення та через канал зв'язку відсилає назад вихідні данні. Каналом зв'язку може бути будь-який з інтерфейсів, які підтримують апаратний засіб та середовище моделювання.

Огляд матеріалів

Проблемі створення прототипів пристроїв, а також розробці середовища для сумісного моделювання, яке задовольнило б вимогам науковців та інженерів присвячені наступні роботи.

У роботі [1] у загальному вигляді описано процес використання MATLAB/Simulink як засобу розробки пристроїв на ПЛІС. Як приклад розглянуто створення передавача стандарту HSDPA, описано загальний процес перетворення моделі MATLAB/Simulink у код на мовах опису апаратного забезпечення.

Стаття [2] містить приклад розробки регулятора для асинхронного електричного двигуна у MATLAB/Simulink. Як засіб перетворення моделі на код на мові опису апаратного забезпечення використано програмний пакет Xilinx System Generator, який дозволяє створити модель для ПЛІС фірми Xilinx.

У [3] описується приклад зв'язку між MATLAB/Simulink та демо-платою з ПЛІС фірми Altera за допомогою програмних засобів фірми Celoxica.

У статтях [4], [5] наведено приклади побудови ПІД-регуляторів за допомогою програмного пакету Xilinx System Generator. У [5] детально розглянута побудова ПІД-регулятора та проведено аналіз використання ресурсів ПЛІС таким регулятором.

Робота [6] детально розглядає використання ПЛІС для прототипування регуляторів. Розглянуто методику створення регуляторів з можливістю обробки аналогових даних за допомогою Xilinx System Generator, а також інтеграцію створених регуляторів у Систему-на-чіпі (SoC) з написанням інтерфейсу для обміну даними з ПК з ОС Linux.

Постановка проблеми

Як видно з огляду приведених вище робіт, на даний момент не існує єдиного загального способу забезпечення комунікацій між апаратним пристроєм, реалізованим на ПЛІС, та моделлю на ПК, реалізованій у програмному пакеті MATLAB. Існуючі способи обмежені у своєму використанні до ПЛІС окремих виробників, або до демонстраційних плат з ПЛІС окремих моделей. До того ж більшість середовищ сумісного моделювання вимагають використання окремих програмних засобів, ускладнюючи створення такого середовища. В даній роботі буде розглянуто спосіб створення середовища для сумісного моделювання на основі програмного пакету MATLAB, що є універсальним у сенсі незалежності від моделі та марки використовуваної ПЛІС, та дозволить провести сумісне моделювання з прототипом створеним за допомогою MATLAB\Simulink HDL Prototype Control Toolbox[7].

Вирішення проблеми

Основи цієї розробки закладено у статті [7]. У ній описано створення засобів та загальної методики перетворення векторно-матричних моделей керуючих пристроїв у код на мові Verilog, який потім може використовуватись для програмування ПЛІС.

Для сумісного моделювання разом з моделлю у Simulink код регулятора поєднується з підсистемою, що забезпечує обмін даними з ПК. Структурна схема рішення представлена на рисунку 2.

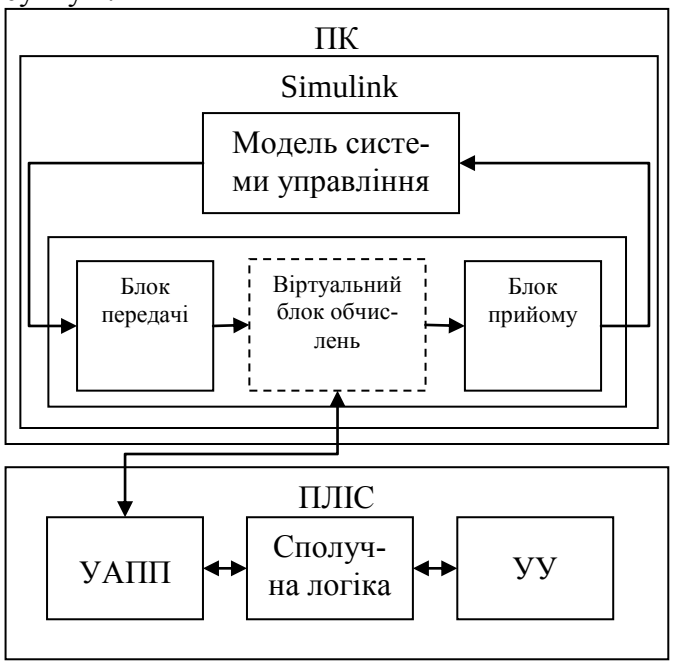


Рис. 2. Структурна схема середовища сумісного моделювання на базі MATLAB/Simulink

На боці комп'ютера знаходиться пакет Simulink із зібраною в ньому схемою передачі і прийому квантованих за часом значень величин впливів. У цей час ПЛІС налаштована на виконання таких функцій:

– прийом і передача даних;

– обробка (обчислення) даних.

За прийом і передачу відповідає модуль універсального асинхронного приймача-передавача (УАПП – UART). Параметри режиму передачі задаються до конфігурування пристрою, на стадії опису апаратного рівня.

Сполучна логіка (СЛ) об'єднує заздалегідь задану кількість послідовно переданих байт в одне слово, яке являє собою одне, квантоване за часом, значення, прийняте від ПК. Даний модуль також проводить зворотні перетворення над обробленими даними для їх передачі на ПК. Після формування одного слова даних керуючий модуль виставляє це значення на вхідну шину обчислювального модуля і видає одиночний імпульс, тривалістю в один період тактової частоти, на вхід дозволу обчислення. Вже на наступному такті обчислювальний модуль виставляє на вихідну шину обчислене значення, яке згодом розбивається на байти і передається на ПК. Після виділення одного байта зі слова даних модуль сполучної логіки виставляє на вхідні шину модуля УАПП даний байт і формує одиночний імпульс дозволу передачі. По завершенні передачі чергового байта модуль УАПП інформує модуль СЛ про закінчення передачі, що дає останньому можливість вибору наступної дії: або передачі наступного байта або переходу в режим очікування, якщо все слово даних було передано.

Модуль сполучної логіки реалізований у вигляді кінцевого автомата типу Мілі для забезпечення синтезування вищеописаного алгоритму.

Параметр довжини слова в байтах, який є умовою для переходу в станах прийому і передачі даних, встановлюється у конфігураційному файлі при генерації моделі.

Обчислювальним модулем виступає блок «HDL State-Space» трансльований в код на мові Verilog[7] й об'єднаний з керуючою програмою.

Для доведення можливості функціонування приведеної вище системи її було реалізовано у експериментальному режимі. При цьому, як засіб зв’язку було обрано широко поширений інтерфейс RS-232, що забезпечує обмін даних між ПЛІС та ПК. До недоліків такої реалізації слід віднести невелику швидкість передачі даних, яка не дозволяє системі працювати в режимі реального часу. Але, слід зазначити, що структура системи побудована таким чином, що, по-перше, прототип не залежить від реалізації інших частин системи, тобто він може використовуватись, як з моделлю на ПК, так і з моделлю створеною на апаратних компонентах, та, по-друге, при необхідності легко може бути змінений інтерфейс, що поєднує прототип з іншими частинами системи, тобто можливо реалізувати інтерфейси, які дозволять всій системі працювати в режимі реального часу, і це не буде вимагати внесення змін до структури системи.

Приклад використання

Для прикладу проведемо сумісне моделювання двовимірної системи з наявними неконтрольованими збуреннями (із застосуванням фільтру Калмана).

Об'єкти системи задамо у векторно-матричному вигляді, яке буде відповідати рівнянням (1), де A, B, C, D – матриці, $\mathbf{X}(t)$ – вектор станів об'єкту, $\mathbf{U}(t)$ – вектор управління, $\mathbf{Y}(t)$ – вектор виходів об'єкту.

$$\begin{aligned} \dot{\mathbf{X}}(t) &= A\mathbf{X}(t) + B\mathbf{U}(t), \\ \mathbf{Y}(t) &= C\mathbf{X}(t) + D\mathbf{U}(t). \end{aligned} \qquad (1)$$

Система задана за допомогою векторно-матричних моделей об'єкта управління та керуючого пристрою (розрахунок виконувався окремо), (2) та (3), відповідно.

$$A = \begin{bmatrix} -3 & 2 \\ 2 & -6 \end{bmatrix}, B = \begin{bmatrix} 3 & 2 \\ 4 & 1 \end{bmatrix}, \quad C = \begin{bmatrix} 0.4 & 0 \\ 0 & 1.2 \end{bmatrix} \qquad (2)$$

$$A_r = \begin{bmatrix} 0 & 0 & -33.3 & 0 \\ 0 & 0 & 0 & -100 \\ 3 & 0 & -20.547 & -3.5921 \\ 0 & 1 & -0.3991 & -23.355 \end{bmatrix},$$

$$B_r = \begin{bmatrix} 83.333 & 0 & 0 & 0 \\ 0 & 83.333 & 0 & 0 \\ 0 & 0 & 1.2617 & 2.9934 \\ 0 & 0 & 0.9978 & 2.7125 \end{bmatrix}, \qquad (3)$$

$$C_r = \begin{bmatrix} -0.6 & 0.4 & 2.6 & -5.2 \\ 2.4 & -0.6 & -12.4 & 6.8 \end{bmatrix},$$

$$D_r = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

Модель системи у MATLAB\Simulink матиме вигляд як на рисунку 3.

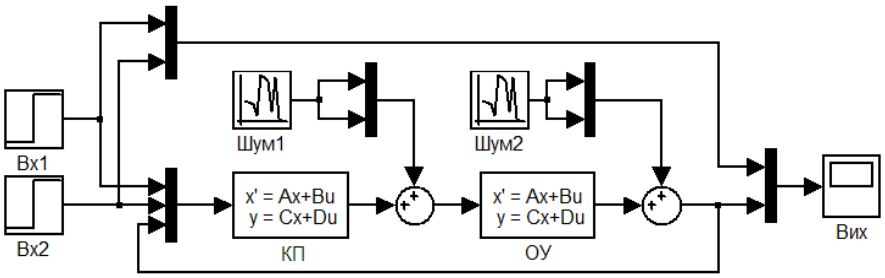


Рис. 3. Схема системи з фільтром Калмана

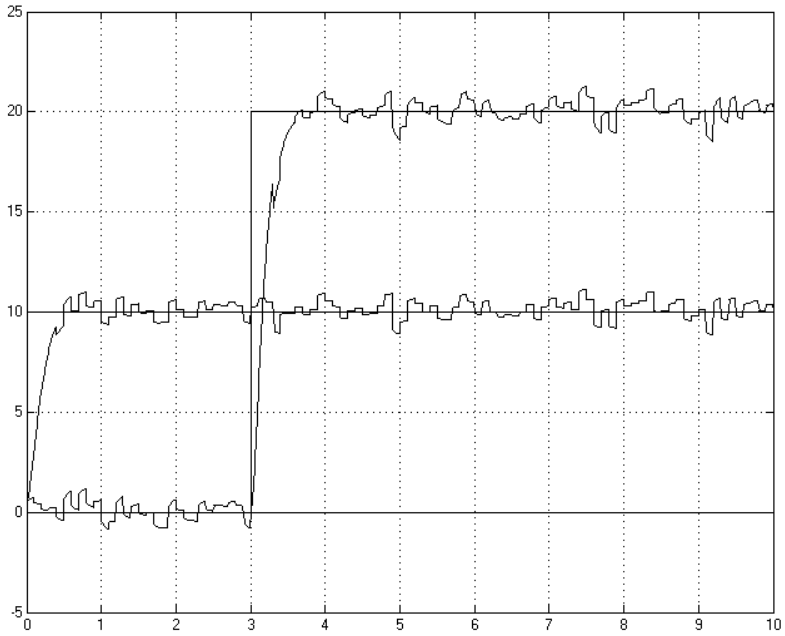


Рис. 4. Перехідні процеси у системі

Наступним кроком додамо до моделі блок з розробленої бібліотеки інструментальних засобів MATLAB\Simulink HDL Prototype Control Toolbox, за допомогою якого створимо програму для ПЛІС. Далі отримаємо можливість генерувати програму функціонування керуючого пристрою для ПЛІС за допомогою Simulink HDL Coder[7], об'єднаємо отриману програму із сполучною логікою та згенеруємо файл прошивки для ПЛІС. Під час синтезу логіки для ПЛІС було отримано дані про витрати апаратних ресурсів ПЛІС обраної моделі, що приве-

дені у таблиці 1. Ці дані свідчать про те, що ресурси ПЛІС було використано приблизно на 75%, та подальше ускладнення регулятора є неможливим без певної оптимізації, або використання мікросхеми ПЛІС з більшими апаратними ресурсами.

Табл. 1. Витрати апаратних ресурсів ПЛІС та обмеження тактової частоти

Ресурс	Використано	Доступно	Процент використання
Логічні таблиці	5 119	7 168	71%
Конфігуровані блоки	2 685	3 584	74%
Блоки множення	16	16	100%
Мультиплексори	2	8	25%
Блоки вводу-виводу	12	264	4%
Тактова частота	22 Мгц	33 Мгц	66%

Отримавши бінарний файл, запрограмуємо ПЛІС, у нашому випадку Xilinx Spartan 3E, створеною програмою.

Після цього модифікуємо модель Simulink таким чином, щоб забезпечити обмін даними між Simulink та ПЛІС, та виконаємо моделювання.

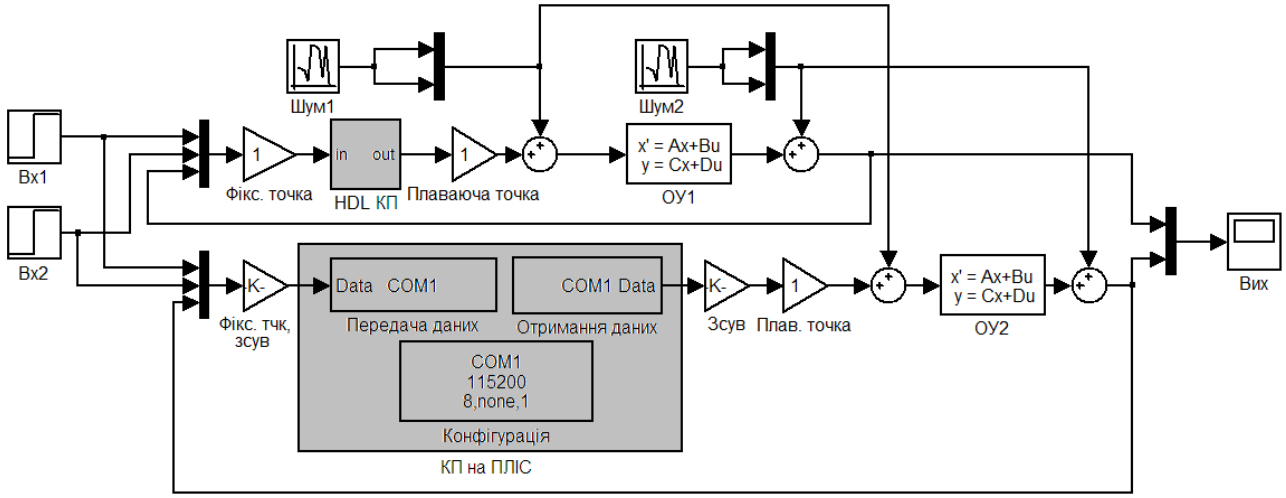


Рис. 5. Модель системи управління, що реалізує методологію сумісного моделювання

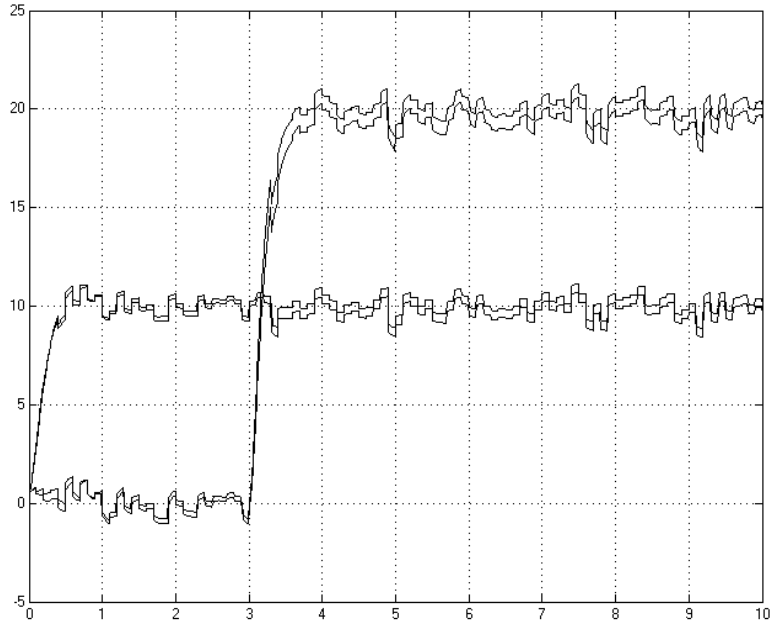


Рис. 6. Результати сумісного моделювання системи

Результати моделювання свідчать, що апаратна система веде себе аналогічно до моделі у Simulink з невеликою розбіжністю, що обумовлена використанням у ПЛІС типів даних з обмеженою точністю.

Висновки

Результатом роботи є середовище інструментальних засобів MATLAB\Simulink для розро-

бки прототипів керуючих пристроїв, та проведення їх сумісного моделювання на базі програмного пакету MATLAB. Запропоновані засоби забезпечують роботу в рамках одного програмного середовища для роботи з моделлю системи та незалежність від виробника та марки ПЛІС.

Подальшою перспективою досліджень може бути розробка інфраструктурних блоків для Simulink та покращення сполучної логіки для додання інших інтерфейсів комунікації між Simulink та ПЛІС, а також автоматизація програмування ПЛІС з загального середовища розробки.

Список літератури

1. Tejas M. Bhatt, Dennis McCain. Matlab as a Development Environment for FPGA Design // Proceedings of the 42nd annual Design Automation Conference.– New York, NY, USA.– 2005.– 607.
2. R.Rajendran, Dr.N.Devarajan. FPGA Based Implementation of Space Vector Modulated Direct Torque Control For Induction Motor Drive // International Journal of Computer and Electrical Engineering. – June, 2010. – Vol. 2, No. 3. – 1793-8163
3. Bartosinski, R., Kadlec, J. Hardware Co-simulation with Communication Server from MATLAB/Simulink // KONFERENCE TECHNICAL COMPUTING. – PRAGUE. – 2006.
4. Yuen Fong Chan, Moallem M., Wei Wang. Design and Implementation of Modular FPGA-Based PID Controllers // Industrial Electronics, IEEE Transactions on. – 2007. – Volume 54 Issue:4 – 1898 – 1906
5. The use of FPGA in field-oriented control of an induction machine, Ozkan AKIN, Irfan ALAN
6. Stephane Simard, Jean-Gabriel Mailloux, and Rachid Beguenane. Prototyping Advanced Control Systems on FPGA // EURASIP Journal on Embedded Systems. – 2009.– Volume 2009.– 12.
7. Писаренко А.В., Репнікова Н.Б., Цвєлодуб А.О. Розробка Verilog/VHDL орієнтованого блоку MATLAB/Simulink // Наукові вісті Галицької академії. – Івано-Франківськ: ГА, 2010. – № 17(1). – С.115-121.

УДК 004.3

ЛУЦЬКИЙ Г. М.,
БЛІНОВА Т.О.,
ПОРЄВ В.М.

ВИКОРИСТАННЯ ДЕЯКИХ СПОСОБІВ КОДУВАННЯ ДЛЯ ПІДВИЩЕННЯ ТОЧНОСТІ ОБЧИСЛЕНЬ

При здійсненні в обчислювальних системах операцій, особливо векторних, над числами з плаваючою точкою дуже важливі питання точності. Описується алгоритм роботи обчислювального пристрою, орієнтованого на виконання над числами із плаваючою точкою операції послідовності додавання парних добутків. Для забезпечення підвищеною точністю проміжні результати виконання арифметичних операцій представлені в системі числення зі збільшеною основою, причому цифри представлені двійковим кодом. Це дає можливість отримувати результати обчислень, що містять, у загальному випадку, тільки помилку, що виникає через представлення мантиси кінцевим числом розрядів

At realization in the computer systems of operations, especially vector operations, on floating point numbers the questions of exactness are very important. The work algorithm of floating point number computing device, which oriented to performance of add sequence of paired products, is described. For providing enhanceable exactness the intermediate arithmetic results represent in system with scaled-up basis and with digits are represented by binary code. It enables to get the calculation results which contain, in general case, only error which arises up through mantissa represent of a finite number of digits.

Через обмежену, навіть дуже велику, розрядність цифрових обчислювальних машин, питання точності дуже важливі, особливо для обчислень із плаваючою точкою.

Обчислення над числами у формі з плаваючою точкою неточні по своїй природі, можуть спостерігатися порушення закону асоціативності, закону дистрибутивності, що пов'язує операції × та +: $u \times (v + w) \neq (u \times v) + (u \times w)$ тощо. Віднімання майже рівних величин, наприклад, може значно збільшити відносну похибку. Вважається, що для виконання арифметичних дій над числами із плаваючою точкою подвійна точність необхідна майже завжди (у відмінність від фіксованої точки). Для знаходження, наприклад, точного співвідношення для додавання величин із плаваючою крапкою, потрібно виконати сім операцій додавання [1].

Виконання рекомендації по зменшенню похибок, зокрема, попереднього сортування чисел за розміром, рутинного перетворення формул і т.п. програмістом або транслятором утруднюється через ряд причин, наприклад, через незнання до початку роботи програми можливих значень змінних. Забезпечення цих правил, наприклад, зміна порядку подачі операндів, створює труднощі при реалізації паралельних обчислень.

У роботі [2] показано, що при виконанні векторних операцій особливу увагу необхідно приділяти точності операцій, які пов'язані з виконанням послідовності операцій алгебраїчного додавання, а саме: віднімання-додавання, додавання парних добутків тощо.

Отже, уявляються актуальними дослідження, спрямовані на реалізацію виконання арифметичних операцій над числами з плаваючої точкою, що забезпечують зберігання максимально можливого числа стандартних математичних законів.

У роботах [3,4] описаний спосіб додавання послідовності чисел із плаваючої точкою, що дозволяє одержати результат обчислень із заданою точністю. При цьому передбачається, що на вході пристрою кожне число x із плаваючою точкою має вигляд:

$$x = M_x \cdot 2^{Px}, \qquad (1)$$

де M_x – n-розрядна нормалізована ($½ \leq | M_x | < 1$, при $x \neq 0$) дробова частина числа x (мантиса); Px – порядок числа (ціле число з інтервалу [$Pmax$, $Pmin$]); M_x та Px представлено додатковим двійковим кодом.

Відображення довільного дійсного числа у вигляді (1) містить, у загальному випадку, помилку, що виникає через представлення мантиси кінцевим числом розрядів. За винятком цієї помилки, результат додавання послідовності чисел може бути отриманий із заданою точністю, що досягається через відсутність нормалізації й округлення проміжних сум, а також унаслідок того, що молодші розряди

мантис доданків при вирівнюванні їхніх порядків не втрачаються.

Для зменшення сумарних апаратурних витрат запропоновано використовувати для представлення чисел у систем числення з основою більше двох.

Через те, що при розробці пристроїв обчислювальної техніки рідко використовують багатопозиційні цифрові елементи використовуються системи числення з основами 2^k (k – ціле позитивне число), і кодування кожної цифри цих систем числення визначеною комбінацією станів декількох двопозиційних елементів. Причому, як показують дослідження, приведені в книзі, шуканий мінімум апаратурних витрат досягається при $k=n$-1, тобто при $\varepsilon = 2^{n-1}$.

Вибір таких систем числення обумовлений тим, що вони є найбільше економічними з погляду необхідного устаткування для кодування цифр за допомогою двопозиційних елементів.

Іншою важливою властивістю систем числення з основою 2^k є той факт, що для перекладу числа з фіксованою точкою з однієї системи числення в іншу не потрібно ніяких дій. У зв'язку з цим, видача операндів на входи перетворювача інформації, що працює в системі числення з основою 2^k і зчитування результату з його виходів може супроводжуватися мінімальними перетвореннями.

Таким чином, вираз (1) для довільного числа x с плаваючої точкою у системі числення з такою основою ε буде мати наступний вигляд:

$$x = M_x(\varepsilon)\cdot \varepsilon^{Px(\varepsilon)} = M_x\, 2^{Px \bmod (n-1)}\varepsilon^{\lfloor Px/(n-1)\rfloor}.$$

Зазначимо, що тут і далі $\lceil X \rceil$ – найменше ціле число, що більше або дорівнює X; $\lfloor X \rfloor$ – найбільше ціле число, що менше або дорівнює X; $X \bmod Y$ – залишок від ділення X на Y.

Отже, для переводу довільного числа з плаваючою точкою з двійкової системи числення у систему числення з основою $\varepsilon = 2^{n-1}$ необхідно обчислити $P_x \bmod (n-1)$ і відповідно зсунути мантису, формуючи дві цифри числа у системі обчислення з основою ε, а також обчислити $\lfloor Px/(n-1)\rfloor$, щоби визначити, на які позиції числа, представленого у цій системі числення, поставити отримані дві цифри (інші цифри числа дорівнюватимуть нулю).

Аналогічно, для зворотного переводу довільного числа x із плаваючою точкою з системи числення з основою $\varepsilon = 2^{n-1}$ у двійкову систему числення необхідно обчислити, скільки нульових старших цифр у числі, це і буде величина $\lfloor Px/(n-1)\rfloor$, потім відповідно зсунути уліво число на $\lfloor Px/(n-1)\rfloor$ позицій; проаналізувати двійковий код старшої цифри зсунутого числа, визначаючи, скільки нульових (уже двійкових) розрядів вона має, це і буде $P_x \bmod (n-1)$; потім зсунути уліво на $P_x \bmod (n-1)$ розрядів (двійкових) усе число, формуючи тим самим нормалізовану мантису M_x числа x, представленого у двійковій системі числення, а також обчислити порядок $P_x = (P_x \bmod (n-1)) + (n-1)\lfloor Px/(n-1)\rfloor$ нормалізованого числа x.

Роботи [5,6] присвячені побудові та моделюванню конвеєрного суматора чисел із плаваючою точкою, що забезпечує підвищену точність, робота [7] присвячена побудові пристрою для додавання послідовності добутків пар чисел із плаваючою точкою.

Мета даної статті – надати опис алгоритму роботи розробленого пристрою для виконання векторної операції додавання послідовності добутків пар чисел із плаваючою точкою.

Пристрій для виконання векторної операції додавання послідовності добутків пар чисел із плаваючою точкою (ПДПД) містить блок множення, блок керування, блок підсумовування, блок формування результату (рис.1). Функціональна схема ПДПД представлена на рис.2.

При обчисленні суми послідовності добутків пар чисел на вхід блока множення на кожному такті роботи ПДПД подаються представлені у прямому коді n-розрядні нормалізовані мантиси обох співмножників (у регістри РгМ1 та РгМ2), їхні порядки (у регістри РгП1 та РгП2) та знаки (у тригери ТгЗ1 та ТгЗ2).

Нехай на i-му такті роботи на вхід пристрою подано i-у пару співмножників і у блоці множення відбувається обчислення n-розрядного добутку цих співмножників (за допомогою вузла Пам'ять добутків), порядок (за допомогою суматора) та знак (за допомогою логічного вузла ЛогВ, що обчислює функцію нерівнозначності).

На $(i+1)$-му такті роботи у блоці керування відбувається перетворення цього добутку (далі він виступає у ролі доданка) із двійкової системи обчислення у систему обчислення з основою $\varepsilon=2^{n-1}$.

А саме, з приходом переднього фронту тактового імпульсу мантиса M_x доданка (добутку), що представлений в прямому коді, порядок P_x доданка (добутку), знак доданку (добутку) записуються відповідно у регістри мантиси

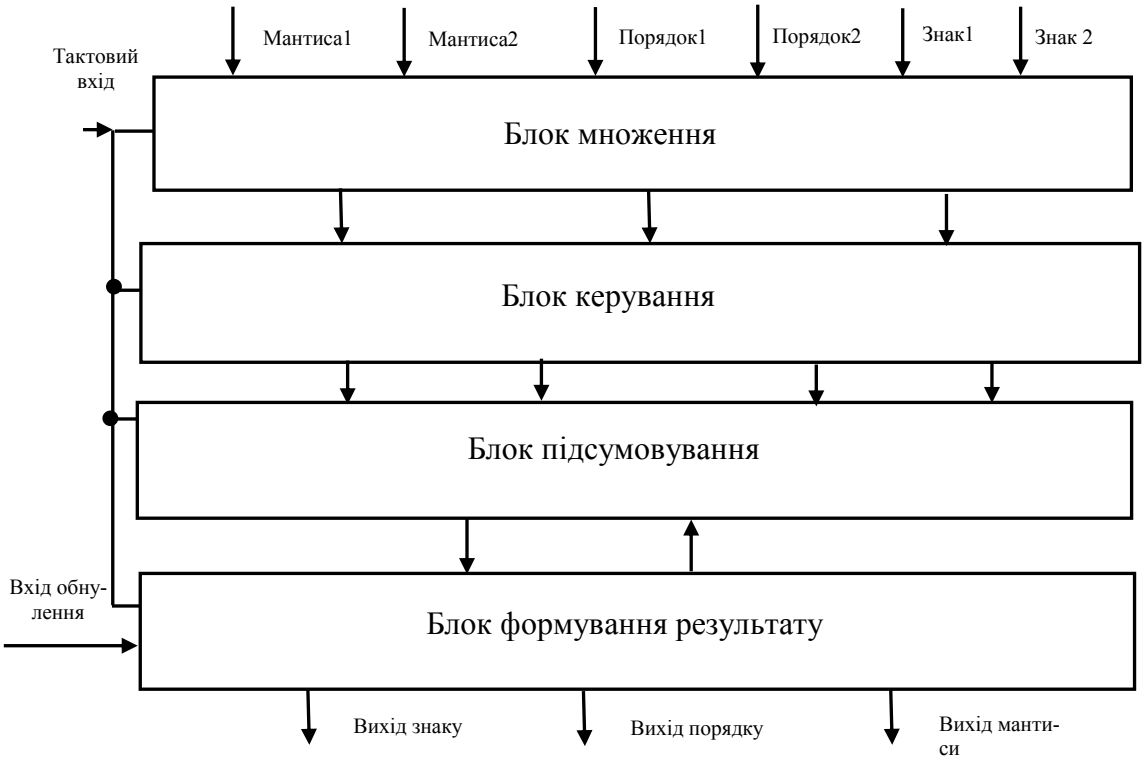


Рис.1. Структурна схема пристрою

(РгМ) та порядку (РгП) і тригер керування (ТгК).

За допомогою вузла Пам'ять 1 визначається P_x mod (n-1), і відповідно на P_x mod (n-1) розрядів зсовується мантиса M_x, формуючи дві цифри числа $M_x(\varepsilon)$ у системі обчислення з основою ε, а також обчислюється $\lfloor Px\ /(n\text{-}1)\rfloor$, щоби визначити, на які позиції числа $M_x(\varepsilon)$ поставити отримані дві цифри (інші цифри числа дорівнюватимуть нулю). Отже, на перший вихід вузла Пам'ять 1 подається цифра числа $M_x(\varepsilon)$, що повинна потрапити на якусь непарну позицію, а на другий вихід вузла Пам'ять 1 подається цифра числа $M_x(\varepsilon)$, що повинна потрапити на якусь парну позицію числа $M_x(\varepsilon)$. Третій вихід вузла Пам'ять 1 визначає, на які саме позиції будуть записані нулі, а отже у ті позиції, що залишилися, будуть записані ці дві цифри.

На (i+2)-му такті роботи у блоці підсумовування відбувається перевід мантиси $M_x(\varepsilon)$ у доповняльний код та підсумовування її з накопиченою сумою, що по ланцюгу зворотного зв'язку поступає з регістра блока формування результату (на початку обчислення нової послідовності ця накопичена сума дорівнює нулю, тому що на (i+2)-му такті роботи регістр був встановлений у нуль).

Це здійснюється наступним чином. З приходом (i+2)-го тактового імпульсу по його передньому фронту відбувається запис числа $M_x(\varepsilon)$ у (n-1)• m- розрядний регістр мантиси РгММ, де n – розрядність мантиси, $m = \left\lceil \frac{P_{макс}}{n-1} \right\rceil + 1$, а $P_{макс}$ – максимальний порядок доданків. Перевід мантиси $M_x(\varepsilon)$ у доповняльний код проходить два етапи. Спочатку комутатор формує обернений код числа (інвертує число, якщо знак доданка дорівнює одиниці), потім при виконанні підсумовування на вхід переносу в молодший розряд суматора мантиси подається знак доданка.

На (i+3)-му такті роботи у блоці формування результату відбувається перевід накопиченої суми у прямий код, перевід її у двійкову систему числення та починається формування нормалізованого n - розрядного результату.

Перевід результату $M_x(\varepsilon)$ у прямий код здійснюється у два етапи. Спочатку комутатор результату інвертує число, якщо знак результату дорівнює 1, потім при виконанні підсумовування на вхід переносу в молодший розряд суматора результату подається знак доданка, а на другу групу входів суматора результату подаються сигнали "Логічний нуль".

Формування нормалізованого n- розрядного результату теж проходить у декілька етапів. Спочатку за допомогою першої групи елементів "АБО" із m (n-1)- входових елементів "АБО", виявляються всі нульові (n-1)- розрядні

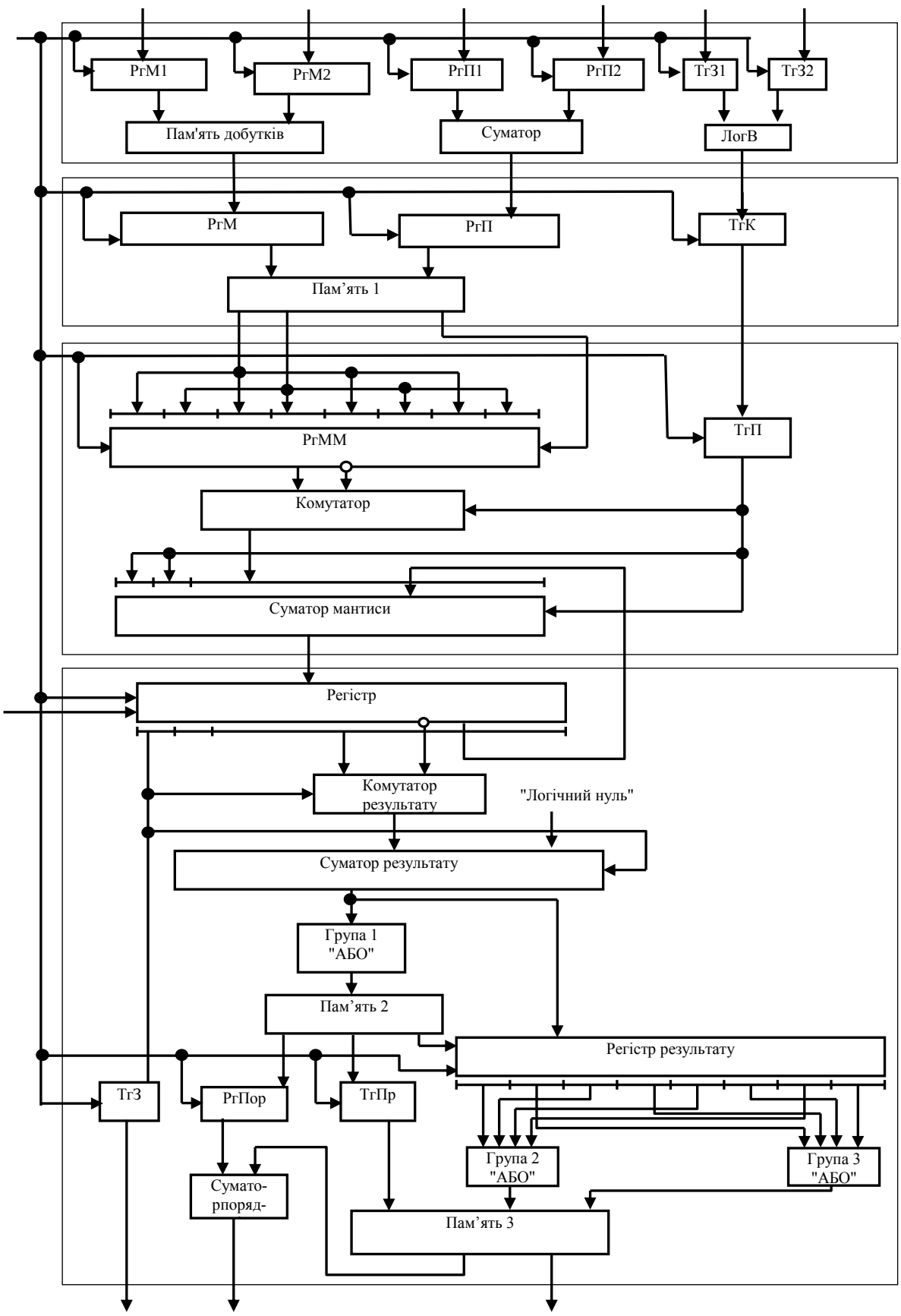


Рис.2. Функціональна схема пристрою

цифри результату. Потім у вузлі Пам'ять2 визначається, скільки старших цифр результату дорівнює нулю, і формується $((n\text{-}1)\lfloor Px/(n\text{-}1)\rfloor)$ – частина порядку числа. Крім того, у вузлі Пам'ять 2 формуються сигнал перестановки та сигнали встановлення в нуль регістра результату.

На $(i+4)$-му такті роботи у блоці формування результату продовжується перевід результату у двійкову систему числення та формування нормалізованого n - розрядного результату.

По передньому фронту тактового сигналу у регістр результату записуються тільки дві старші $(n\text{-}1)$-розрядні групи ненульових розрядів, позаяк інші находилися під впливом сигналів встановлення в нуль. За допомогою другої (з $\lceil 0{,}5m\rceil$ $(n\text{-}1)$-входових елементів "АБО") та третьої $((m\text{-}\lceil 0{,}5m\rceil)$ $(n\text{-}1)$-входових елементів "АБО") груп елементів "АБО" їх відокремлено, тим самим відокремлено дві старші цифри мантиси, представленої у системі числення з основою ε. Перша відокремлена цифра може бути як на парній позиції, так і на непарній позиції. Отже, коли на другий адресний вхід вузла Пам'ять 3 поступає непарна цифра, вона може бути молодшою. Для того, щоб знати це, на перший адресний вхід вузла Пам'ять 3 подається сигнал перестановки із тригеру перестановки ТгПР, він дорівнює одиниці, якщо потрібно поміняти місцями старшу і молодшу цифри. Таким чином формується $2(n\text{-}1)$-розрядне двійкове число. За допомогою вузла Пам'ять 3 на основі аналізу того, скільки нульових (уже двійкових) старших розрядів міститься у цьому числі, визначається P_x mod $(n\text{-}1)$, це число на P_x mod $(n\text{-}1)$ розрядів зсовуються уліво і формується n- розрядний результат, відкидаючи, якщо це потрібно зайві розряди). Частина порядку P_x mod $(n\text{-}1)$ результату з першого виходу вузла Пам'ять 3 поступає в суматор порядку, де підсумовується з частиною порядку $((n\text{-}1)\lfloor Px/(n\text{-}1)\rfloor)$. Отже, на $(i+4)$-му такті формується знак суми, порядок та нормалізована мантиса результату.

Таким чином, для підсумовування послідовності добутків із l пар чисел із плаваючою крапкою потрібно $(l+4)$ тактів роботи пристрою. Перед початком обчислення нової послідовності чисел має бути хоча б один холостий такт.

ПДПД легкий у керуванні, тому що виконує обчислення за передбачувану кількість тактів, що залежить тільки від числа доданків.

ПДПД створено з урахуванням вимог IEEE Стандарту на двійкову арифметику із плаваючою точкою (P754) та Стандарту на арифметику із плаваючою точкою з різними основами системи числення й словами довільної довжини (P854). Вхідні операнди та остаточний результат нормалізовано й представлено у прямому коді, порядок чисел зміщений. У самому ПДПД виконуються операції з підвищеною точністю. На виходах пристрою формуються результати, що знову представлені у відповідності зі Стандартами P754 і P854.

Запропонований пристрій для виконання з підвищеною точністю векторної операції підсумовування послідовності добутків пар чисел із плаваючою точкою може без змін використовуватися, як закінчений вузол, у процесорах.

Список літератури

1. Кнут. Д. Искусство программирования для ЭВМ. В семи томах. Т.2. Получисленные алгоритмы, пер. с англ. − М.: Мир, 1977. − 724 с.
2. Луцький Г.М., Блінова Т.О.Долголенко О.М Пристрій для виконання з підвищеною точністю векторних операцій над числами зі плаваючою точкою //Вісник НТУУ "КПІ". Сер. Інформатика, управління та обчислювальна техніка. – 2002. № 37– С.130-135.
3. АС СССР № 849205. Конвейерное устройство для выполнения арифметических операций над множеством чисел /Вилкул М.А., Долголенко А.Н.,Луцкий Г.М. –1981. Бюл. № 27
4. Самофалов К.Г. Луцкий Г.М. "Основы теории многоуровневых конвейерных вычислительных систем". М.: Радио и связь. – 1989. -272 с.
5. Патент України №50428А, Пристрій для додавання послідовності чисел із плаваючою точкою / Блінова Т.О., Долголенко О.М., Луцький Г.М., Порєв В.М.– 2002. Бюл. № 10.
6. Аксьоненко С.В., Долголенко О.М. VHDL-реалізація конвеєрного пристрою для матричних обчислень високих порядків на FPGA, що реконфігуруються //Вісник НТУУ "КПІ". Сер. Інформатика, управління та обчислювальна техніка. – 2007.№ 46– С. 32-44.
7. Патент України №58920А, Пристрій для додавання послідовності добутків пар чисел із плаваючою точкою / Блінова Т.О., Долголенко О.М., Луцький Г.М., Порєв В.М. -2003.Бюл. № 8.

УДК 681.324

ЗАЙЧЕНКО Е.Ю.,
ЗАЙЧЕНКО Ю.П.,
ЛАВРИНЧУК А. Н.

ИНСТРУМЕНТАЛЬНЫЙ КОМПЛЕКС АЛГОРИТМОВ И ПРОГРАММ ОПТИМАЛЬНОГО ПРОЕКТИРОВАНИЯ СЕТЕЙ С ТЕХНОЛОГИЕЙ MPLS

В статті описано інструментальний комплекс алгоритмів та програм для оптимального проектування мереж з технологією MPLS. Розроблені програми комплексу базуються на оригінальних методах та алгоритмах, запропонованих авторами, та включають алгоритми для вирішення різних задач проектування: вибору пропускних спроможностей каналів, розподілу потоків, аналізу та оптимізації живучості та структурного синтезу мереж MPLS. Розроблені моделі та алгоритми враховують специфіку технології MPLS. Наведені результати експериментальних досліджень алгоритмів.

Software tools for MPLS networks design are elaborated and presented in this work. The software tools are based on original methods and algorithms suggested by authors and include algorithms for solution of numerous tasks of channels capacities assignment, flows distribution, survivability analysis and structural synthesis. The elaborated models and algorithms take into account the specificity of MPLS technology. The results of experimental investigations and practical implementation of the suggested software tools are presented.
Keywords: MPLS technology, capacities assignment, flows distribution, topological optimization, survivability analysis.

Введение

Бурный рост сети Интернет, острые потребности пользователей в передаче разнотипных видов информации-аудио, видео и данных, в частности мультимедийной информации, потребовали разработки новой коммуникационной технологии, способной передавать эти виды информации на высоких и сверхвысоких скоростях и обеспечивать при этом требуемое качество обслуживания, различное для разных видов информации. Такой технологией явилась технология многопротокольной коммутации меток (MPLS), которая возникла на рубеже двух веков и в настоящее время интенсивно развивается, охватывая все новые и новые области приложений [1]. Технология MPLS стала унифицированным транспортным механизмом для большинства сетевых технологий, в частности, технологий IP, ATM, быстрой передачи кадров Frame Relay, а также технологии локальных сетей - Ethernet.

Специфическими особенностями технологии MPLS являются следующие:

1. Наличие различных классов сервиса (Class of Service – CoS) до 8 классов, которые обслуживаются в маршрутизаторах (LSR) с относительными приоритетами.
2. Введение показателей качества обслуживания (QoS), а именно средняя задержка в доставке пакетов $T_{cp}(k)$ для разных классов и доля потерянных пакетов – PLR (Packets Loss Ratio).

Появление новой технологии поставило на повестку дня разработку моделей и методов для анализа и оптимального проектирования сетей MPLS. Такие модели и методы должны учитывать особенности технологии MPLS. В работах авторов разработан комплекс моделей и методов на их основе, для анализа и оптимизации характеристик и структур сетей с технологией MPLS [1-4].

Целью настоящей статьи является краткое изложение основных моделей, методов, а также структуры и функций инструментального комплекса для анализа и проектирования сетей “MPLS NetBuilder”.

Комплекс моделей и методов анализа и оптимизации сетей с технологией MPLS

1) Для решения задач анализа и синтеза сетей разных классов сервиса необходима прежде всего разработка **аналитических моделей оценки показателей качества сервиса** – средней задержки $T_{cp,k}$ и доли потерянных пакетов-PLR (Packets Loss Ratio) для различных классов сервиса в сетях MPLS. В работе [2] получены соответствующие модели для расчета основных показателей качества обслуживания в зависимости от пропускных способностей каналов

связи и распределения потоков разных классов сервиса. Эти модели учитывают специфику технологии MPLS и используются для решения многих задача анализа и синтеза сетей MPLS. В частности, для расчета средней задержки в сети потока k-ого класса приоритета получено следующее выражение [1,2]

$$T_{cp,k} = \frac{1}{H_{\Sigma}^{(k)}} \sum_{(r,s)\in E} \frac{f_{rs}^{(k)} \sum_{i=1}^{k} f_{rs}^{(i)}}{\left(\mu_{rs} - \sum_{i=1}^{k-1} f_{rs}^{(i)}\right)\cdot\left(\mu_{rs} - \sum_{i=1}^{k} f_{rs}^{(i)}\right)} \quad (1)$$

В [2] сформулирована **задача распределения потоков** (РП) различных классов в сети MPLS, разработан и исследован алгоритм РП. Данная задача формулируется следующим образом:

Задана сеть с технологией MPLS в виде графа $G=(X,E)$, где $X=\{x_j\}_{j=\overline{1,n}}$ - узлы сети (коммутаторы MPLS) и $E=\{(r,s)\}$ - множество дуг (каналов связи). Имеем возможный набор пропускных способностей (скоростей передачи) каналов связи – $\{\mu_{rs}\}$, $(r,s)\in E$.

Пусть также введены классы обслуживания (CoS – Class of service), задана матрица требований для k-го класса $H(k)=\|h_{i,j}(k)\|$, $k=\overline{1,K}$, где $h_{i,j}(k)$ – интенсивность потока, который необходимо передать от узла i в узел j. Кроме того, введены показатели качества QoS для каждого класса в виде ограничения на среднюю задержку – $T_{зад}(k)$.

Необходимо выбрать маршруты передачи $\{\pi_{ij}\}$ и найти такое распределение потоков всех классов $F(k)=[f_{rs}(k)]$, при которых обеспечивается передача заданных величин потоков со средней задержкой, которая не превышает $T_{зад}(k)$.

В работе исследованы свойства оптимального потока и предложен алгоритм ее решения, который базируется на этих свойствах. Алгоритм сводится к последовательному распределению потоков различных классов, начиная с наиболее приоритетного, по кратчайшим маршрутам в некоторой специальной метрике, которая, как показано, является оптимальной для данного критерия.

2) Следующий класс задач, решаемых в комплексе, это комбинированная задача **выбора пропускных способностей и распределения потоков (ВПСРП)** в сетях MPLS. Она формулируется так [2,3].

Задана сеть MPLS в виде орграфа $G=(X,E)$, где $X=\{x_j\}$ $j=\overline{1,n}$ множество узлов сети, $E=\{(r,s)\}$ – множество каналов связи (КС), набор пропускных способностей (ПС) каналов $D=\{d_1,d_2,...,d_k\}$ и их удельных стоимостей $C=\{c_1,c_2,...,c_k\}$. Заданы также матрицы требований входящих потоков соответствующих классов $H=\|h_{ij}^{(k)}\|$, $i,j=\overline{1,n}$, $k=\overline{1,K}$ и ограничения на среднюю задержку для классов потоков $T_{cp,k}$,

Требуется выбрать такие ПС каналов связи $\{\mu_{rs}^{(0)}\}$ и найти распределение потоков всех классов $F(k)=[f_{rs}(k)]$, при которых стоимость сети будет минимальной, а установленные ограничения на задержки по классам будут выполняться полностью. Математическая модель данной задачи будет иметь следующий вид:

Найти

$$\min C_{\Sigma} = \sum_{(r,s)\in E} C_{rs}(\mu_{rs}), \quad (2)$$

при ограничениях:

$$T_{cp,к}(F(k),\mu_{rs}) \le T_{зад,k} \quad k=\overline{1,K} \quad . \quad (3)$$

В работах [1.2] предложен и исследован метод решения задачи ВПСРП для сетей MPLS.

3) Важным показателем эффективности функционирования сетей является живучесть.

Поскольку основное назначение сети с технологией MPLS является передача заданных величин входящих потоков различных классов, то *живучесть сети MPLS предлагается оценивать величиной максимального потока, который возможно передать в сети при отказах ее элементов*- каналов и узлов при сохранении заданных показателей качества.

В работах авторов [1,4] были впервые предложены **показатели живучести** (ПЖ) сетей с технологией MPLS и разработан метод анализа и сетей по этим показателям.

Для анализа живучести сетей MPLS был введен следующий комплексный показатель;

$$P\{H_{\Sigma}^{\Phi}(1)\ge r\% H_{\Sigma}^{0}(1)\}...P\{H_{\Sigma}^{\Phi}(k)\ge r\% H_{\Sigma}^{0}(k)\}, \quad (4)$$

где $H_{\Sigma}^{0}(k)$ – величина потока k-го класса в безотказном состоянии сети (номинального потока), $H_{\Sigma}^{\Phi}(k)$ - фактическая величина

потока класса k в случае действия отказов, r =(50÷100), $k=\overline{1,K}$.

Таким образом, *показатели живучести определяют распределение вероятностей передачи заданной доли номинального потока при отказах ее элементов*. Предложен алгоритм оценки показателя живучести для сетей MPLS, который реализован в комплексе, и проведены его исследования для различных примеров сетей.

Пусть рассматривается сеть MPLS G = (X, E), состоящая из элементов (каналов и узлов), подверженных воздействию внешней среды, в результате которого они выходят из строя. Предполагается, что заданы надежностные характеристики элементов сети – коэффициенты готовности каналов $k_{\Gamma r,s}$ и узлов k_{Γ_i}, $(r,s)\in E$, $i=\overline{1,n}$.

Будем учитывать следующие отказовые состояния сети:

1. выход из строя 1 КС: Z_1;
2. выход из строя 1 УС: Z_2;
3. выход из строя 2КС : Z_3;
4. отказ 1КС и 1УС: Z_4;
5. выход из строя 3КС: Z_5.

Считая отказы элементов сети статистически независимыми, можно определить вероятность каждого состояния $P(Z_i)$. Например, если Z_i – это выход из строя КС (r_i, s_i), то

$$P(Z_i)=(1-K_{\Gamma r_i s_i})\prod_{(r,s)\neq(r_i,s_i)}K_{\Gamma r,s}\prod_{i=1}^{n}K_{\Gamma_i}\quad,(5)$$

где $K_{\Gamma r,s}$ – вероятность исправного состояния (коэффициент готовности) КС

$(r,s)\neq(r_i,s_i)$, $K_{\Gamma r_i,s_i}$ – вероятность выхода из строя КС (r_i,s_i).

В работе [4] был предложен алгоритм оценки показателей живучести сети MPLS, суть которого состоит в следующем.

1. Вычисляем общую величину потока в безотказном состоянии для всех классов сервиса: $H_\Sigma^{(0)}(1), H_\Sigma^{(0)}(2),..., H_\Sigma^{(0)}(K)$.

2. Моделируем различные отказовые состояния: Z_1, Z_2, Z_3, Z_4, Z_5. Для каждого из них рассчитываем вероятности $P(Z_i)$ согласно (5).

3. Находим величину максимального потока всех классов в состоянии $Z_j: H_\Sigma^{\Phi}(k,z_j)$, $k=\overline{1,K}$.

Для этого используется специально разработанный в [1] алгоритм нахождения максимального потока.

Вычисляем комплексный показатель живучести для каждого класса сервиса: для первого класса

$$P\{H_\Sigma^{\Phi}(1)\geq r\%H_\Sigma^{0}(1)\}=\sum_{Z_j}P(Z_j),\qquad(6)$$

где суммирование в (6) происходит по всем Z_j таким, что $H_\Sigma^{\Phi}(1)\geq r\%H_\Sigma^{0}(1)$;

для k-го класса

$$P\{H_\Sigma^{\Phi}(k)\geq r\%H_\Sigma^{0}(k)\}=\sum_{Z_i}P(Z_i),\qquad(7)$$

где суммирование в (7) происходит по всем состояниям Z_i таким, что:

$H_\Sigma^{\Phi}(k)\geq r\%H_\Sigma^{(0)}(k)$, $H_\Sigma^{0}(k)$ – величина потока k-го класса в безотказовом состоянии сети; $H_\Sigma^{\Phi}(k)$ – фактическая величина потока класса k в случае действия отказов, r =(50÷100), $k=\overline{1,K}$.

Полученные зависимости

$P\{H_\Sigma^{\Phi}(1)\geq r\%H_\Sigma^{0}(1)\},...P\{H_\Sigma^{\Phi}(k)\geq r\%H_\Sigma^{0}(k)\}$

Далее построим в координатах $P\{H_\Sigma^{\Phi}(k)\}$ - $r\%H_\Sigma^{0}$ зависимость показателей живучести (см. рис. 7) от варьируемых параметров. По изменению этих зависимостей можно судить о живучести соответствующей сети.

В ходе проектирования сетей по результатам анализа ее показателей живучести возникает проблема обеспечения требуемого уровня живучести. Естественно, что эта задача может быть решена путем резервирования ее каналов и узлов и структурной оптимизации сети, что требует дополнительных затрат материальных средств. Поэтому в работе рассматривается следующая постановка задачи структурной **оптимизации сети по показателям живучести** [1,5].

Пусть имеется сеть MPLS, которая описывается орграфом $G=\{X,E\}$, где $X=\{x_j\}$ – множество узлов сети (УС), $E=\{(r,s)\}$ – множество каналов связи (КС), μ_{rs} – пропускные способности КС.

Заданы надежностные характеристики элементов сети – коэффициенты готовности каналов $K_{\Gamma_{rs}}$ и узлов K_{Γ_i} и вероятности отказов – $P_{отк_i}=1-K_{\Gamma_i}$. Для каждого

класса k введен показатель качества (QoS) в виде заданной величины средней задержки $T_{ср,k}$. Пусть исходя из функционального назначения сети, установлены требуемые значения показателей живучести для потока k-го класса: $P_{0,зад}^{(k)}, P_{1,зад}^{(k)}, \ldots, P_{5,зад}^{(k)}$.

Требуется определить такую структуру сети, для которой для всех классов K будут обеспечиваться следующие ограничения по уровню живучести:

$$P\left\{H_{\Sigma}^{ф}(k) \geq r\% H_{\Sigma}^{(0)}(k)\right\} \geq P_{r,зад}^{(k)},$$

r=(50÷100), $k = \overline{1,K}$, (8)

а дополнительные затраты средств будут при этом минимальными:

$$C_{\Sigma} = \sum C_{rs}^{рез}(\mu_{rs}) \to \min \qquad (9)$$

Достижение требуемого уровня живучести будем обеспечивать путем введения соответствующего резервирования наиболее ответственных элементов сети (КС и УС). Для определения очередности резервирования предложен следующий показатель эффективности резервирования для каналов

$$\alpha_{r_i s_i} = -\frac{\Delta P(Z_i)}{C_{r_i s_i}}, \qquad (10)$$

где Z_i – состояние выхода из строя КС (r_i, s_i);

$\Delta P(Z_i)$ – изменение вероятности состояния Z_i в случае резервирования, $C_{r_i s_i}$ – стоимость такого резервирования

4) Важной задачей завершающего этапа процесса проектирования сетей является **задача структурного синтеза сетей по критерию стоимости** при ограничениях на показатели QoS и живучести.

В работе [6] была впервые сформулирована постановка задача структурного синтеза сетей

MPLS, предложен и исследован генетический алгоритм синтеза структуры.

Задано множество узлов сети $X = \{x_j\}$ $j = \overline{1,n}$ – маршрутизаторов MPLS (так называемых LSR–Label Switching Routers), их размещение по территории региона, набор пропускных способностей каналов связи $D = \{d_1, d_2, \ldots, d_k\}$, из которых ведется синтез, их удельных стоимостей на ед. длины $C = \{c_1, c_2, \ldots, c_k\}$, определены классы обслуживания CoS (Class of Service), известны матрицы входящих требований для k-го класса $H(k) = \|h_{ij}(k)\|$ $i, j = \overline{1,n}$; $k = 1, 2, \ldots, K$, где $h_{ij}(k)$ – интенсивность k-го класса, который необходимо передавать из узла i в узел j за единицу времени. Кроме того, введены ограничения на показатели качества (QoS) для каждого класса k в виде ограничения на среднюю задержку $T_{зад,k}$, $k = \overline{1,K}$.

Требуется найти структуру сети в виде набора каналов связи (КС) $E = \{(r,s)\}$, выбрать пропускные способности (ПС) каналов связи $\{\mu_{rs}\}$ и найти распределение потоков всех классов $F(k) = [f_{rs}(k)]$, таким образом, чтобы обеспечить передачу требований всех классов $H(k)$ в полном объеме и с задержками $T_{ср}$, не превышающими заданные $T_{зад,к}$ и при этом бы выполнялись ограничения на долю потерянных пакетов CLP_k, а стоимость сети была бы при этом минимальной. В работах [1,6] предложен и исследован генетический алгоритм синтеза структуры сетей MPLS.

Структура и функции инструментального комплекса “MPLS NetBuilder”

Программный комплекс “MPLS NetBuilder” состоит из следующих функциональных модулей:

- Модуль структурного синтеза сетей

MPLS Модуль оценки показателей качества (QoS), а именно, средней задержки $T_{ср,k}$ и доли потерянных пакетов $PLR_к$..

- Модуль анализа и оптимизации функцио

нальных характеристик сетей MPLS, включающий программы РП и ВПС РП.

- Модуль анализа живучести сетей с

технологией MPLS, включающий программы нахождения максимального потока (НМП) в сети при отказах её элементов.

Модуль структурного синтеза сетей MPLS.

Информационные взаимосвязи между модулями приведены на рис 1.

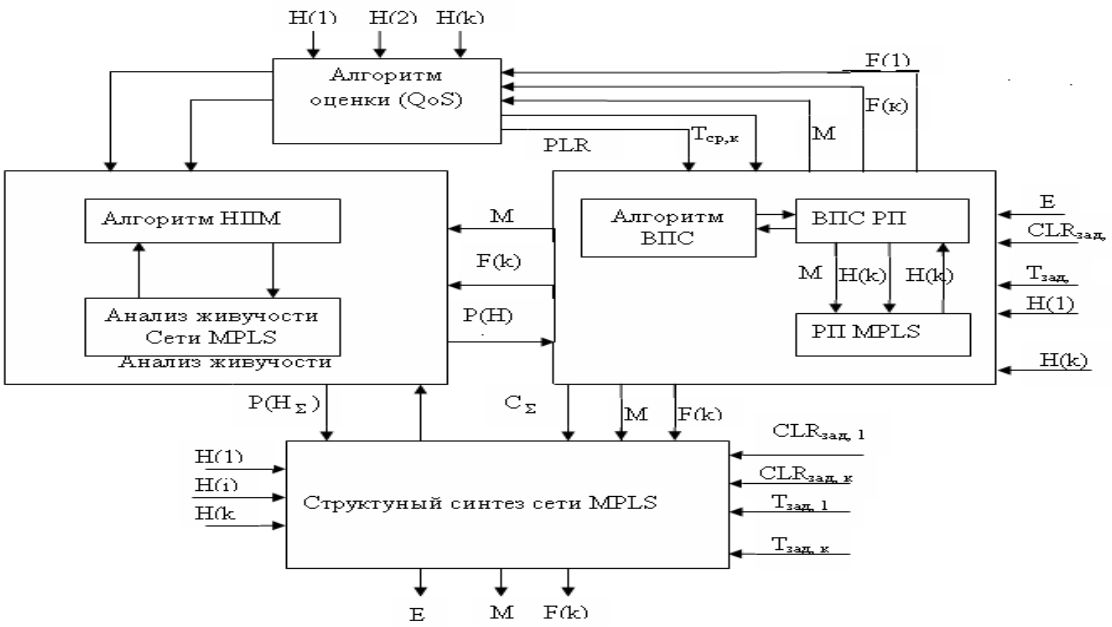


Рис.1. Структурная схема инструментального комплекса анализа и синтеза сетей "MPLS NetBuilder"

Кроме основных функциональных модулей, в состав комплекса входят служебные программные модули, включающие программный модуль интерфейса с пользователем – проектировщиком.

Этот программный модуль позволяет выводить синтерованную структуру в виде карты, осуществлять процедуры проектирования структуры в интерактивном режиме, выводить по указанию пользователя на экране монитора распределения потоков и маршруты передачи, характеристики каналов связи, распределения потоков в КС и их загрузки и характеристики узлов.

Для работы программного комплекса "MPLS NetBuilder" разработан и реализован банк данных, состоящий из отдельных баз данных, таких, как:

- База данных требований входящих потоков.
- База данных каналов связи и база данных узлов сети с их характеристиками.
- База данных распределения потоков различных классов и др.

Все функциональные программные модули взаимодействуют друг с другом через банк данных.

В качестве СУБД используется My SQL Server.

Программный комплекс "MPLS NetBuilder" позволяет решать широкий комплекс задач анализа, оптимизации и синтеза структуры сетей с технологией MPLS при обеспечении заданных показателей качества средней задержки T_{cp} и доли потерянных ячеек PLR для различных классов сервиса, а также анализировать показатели живучести и оптимизировать структуру сетей MPLS при ограничениях на установленные значения показателей живучести.

Результаты экспериментальных исследований

Проведены экспериментальные исследования методов и алгоритмов инструментального комплекса MPLS NetBuilder". Некоторые из них представлены ниже. Алгоритмы РП и ВПСРП исследовались на сети Украины, структура которой приведена на рис.2.

Рассматривалось три класса сервиса. В ходе экспериментов в алгоритме РП увеличивалась матрица требований по передаче соответствующего трафика с помощью умножения на коэффициент k .

На этот коэффициент умножаются все требования матрицы H(k) по передаче соответствующего трафика. При увеличении входящих требований по передаче трафика увеличиваются потоки в каналах связи, а, следовательно, увеличивается и задержка по типам сервисов.

Зависимость задержки для первого класса (K_1) приводится на рис. 3, а для второго класса (K_2)- на рис. 4.

Из представленного графика на рис.3 видно, что время задержки начинает стремительно расти при достижении полной пропускной способности канала связи.

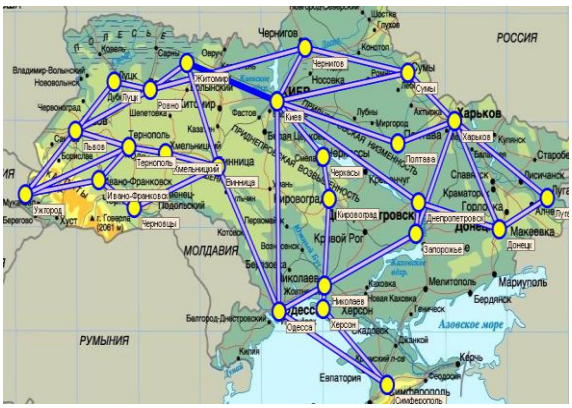


Рис . 2. Пример структуры синтезированной сети

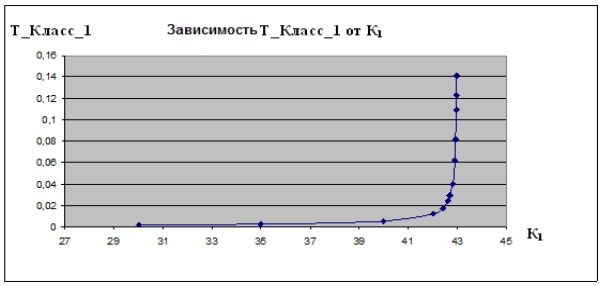


Рис.3. Зависимость средней задержки для первого класса от интенсивности входящего пот ока

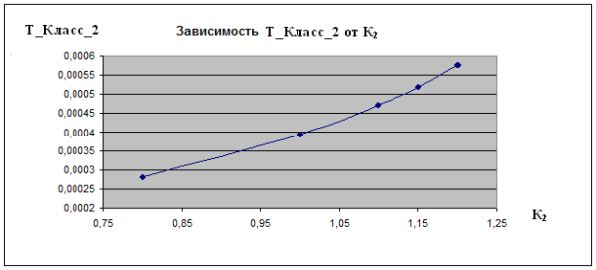


Рис.4. Зависимость средней задержки для второго класса сервиса

В следующей серии экспериментов исследовался алгоритм ВПСРП. В этих экспериментах исследовались зависимости стоимости сети от вариации ограничений на среднюю задержку для различных классов сервиса. Соответствующая зависимость для класса 2 приводится на рис. 5. В следующих экспериментах проводились исследования алгоритмов анализа и оптимизации сетей по показателям живучести.

Как и следовало ожидать, ужесточение требований по средней задержке приводит к увеличению стоимости сети.

Это объясняется тем, что для обеспечения заданных значений T_{cp} приходится увеличивать ПС каналов.

В следующем эксперименте исследовалась зависимость стоимости сети от изменения объемов входящих требований.

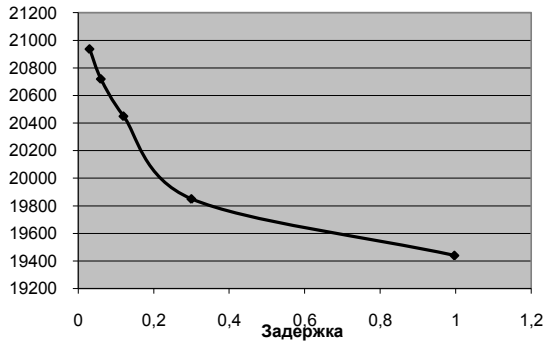


Рис. 5. Зависимость стоимости сети от ограничения на среднюю задержку

При этом исходная матрица H умножалась на коэффициент пропорциональности k. Соответствующие результаты приведены на рис.6. Полученный вид зависимости хорошо согласуется с теорией.

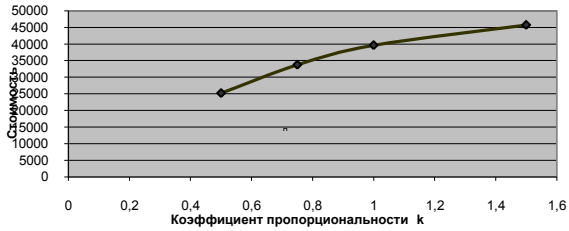


Рис. 6. Зависимость стоимости сети от интенсивности входящего потока

Все эксперименты проводились для коэффициентов готовности каналов, распределенных нормально на интервале 0.9 – 0.95, и для коэффициентов готовности узлов, распределенных нормально на интервале 0.95 – 0.99.

В первой серии экспериментов оценивалась чувствительность показателей живучести первого класса сервиса к вариации ограничения на среднюю задержку [4].

Соответствующие результаты приведены на рис. 7

В следующей серии экспериментов исследовалась чувствительность показателей живучести второго и третьего классов к варьированию ограничения на соответствующую среднюю задержку.

Полученные зависимости имеют вид, аналогичный графикам на рис. 7.

В последующих экспериментах решались задачи оптимизации сети по показателям живучести (ПЖ). Исходные значения ПЖ до оптимизации для четырех классов сервиса приводятся в табл.1, а после оптимизации сети путем определения наиболее эффективного плана резервирования – в таблице 2.

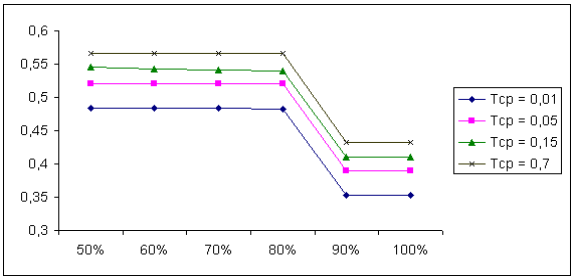


Рис.7. Показатели живучести для класса 1 для разных ограничений на задержку

Табл. 1. Показатели живучести сети MPLS до оптимизации

Н,%	класс1	класс2	класс3	класс4
0,5	0,385	0,385	0,385	0,385
0,6	0,385	0,385	0,385	0,385
0,7	0,385	0,385	0,385	0,385
0,8	0,385	0,385	0,385	0,385
0,9	0,385	0,385	0,385	0,357
1	0,385	0,385	0,385	0,357

Табл. 2. Показатели живучести сети MPLS после оптимизации

Н,%	класс1	класс2	класс3	класс4
0,5	0,814	0,75	0,75	0,72
0,6	0,814	0,75	0,72	0,72
0,7	0,814	0,75	0,72	0,72
0,8	0,814	0,75	0,72	0,72
0,9	0,814	0,75	0,72	0,72
1	0,75	0,75	0,72	0,72

Анализируя и сравнивая результаты в таблицах 1,2, можно легко убедиться, в том, что после оптимизации показатели живучести для всех классов сервиса заметно возросли. В последующих экспериментах исследовался генетический алгоритм структурного синтеза сетей, При этом варьировались матрицы входящих требований и ограничения на показатели качества. Анализируя полученные структуры сети при варьировании коэффициента пропорциональности k в диапазоне 0.2-2.0 сделан вывод об *устойчивости базовой структуры (остова) сети* в широком диапазоне вариации входящей нагрузки (при десятикратном её увеличении). Это свидетельствует о высокой эффективности разработанного метода структурного синтеза сетей.

Выводы

В статье описан инструментальный комплекс алгоритмов и программ анализа и синтеза сетей с технологией MPLS. Созданный комплекс предназначен для решения многих задач оптимального проектирования сетей MPLS: распределения потоков, выбора пропускных способностей каналов связи, анализа показателей живучести и структурного синтеза сетей MPLS.

Проведены экспериментальные исследования алгоритмов анализа и синтеза сетей MPLS. Разработанный комплекс по своим функциям и возможностям не имеет аналогов и позволяет сократить время проектирования, повысить обоснованность проектных решений, снизить капитальные затраты на создание сетей.

Список литературы

1. Зайченко Е.Ю. Зайченко Ю.П. Сети с технологией MPLS: моделирование, анализ и оптимизация. К.: Изд. « Политехника». 2008.- 320 с.
2. Зайченко Ю.П., Ахмед А.М. Шарадка. Оптимизация распределения потоков в сетях с технологией MPLS с учетом задержки в маршрутизаторах // Электроника и связь. –2007. –№1. –С.84 –87.
3. Зайченко Ю.П., Лавринчук А.Н. Выбор метрики для оптимизации распределения потоков трафика в сетях MPLS. Вісник Черкаського державного технологічного університету.–2009.–№3.–с.3-7.
4. Зайченко Ю.П., Мохаммадреза Моссавари.Анализ показателей живучести компьютерной сети с технологией MPLS.//Вісник національного технічного університету “КПІ”.Інформатика, управління та обчислювальна техніка.–43,2005,с.73-80.
5. Зайченко Ю.П., Мохаммадреза Моссавари. Оптимизация компьютерных сетей с технологией MPLS по показателям живучести в случае активной внешней среды. //Вісник національного технічного університету “КПІ”.Інформатика, управління та обчислювальна техніка.–.45,2006,с.163-172.
6. Зайченко Ю.П., Е. Ю. Зайченко, Мохаммадреза Моссавари, Ашраф Абдель-Карим Хилал Абу-Аин. Структурный синтез компьютерных сетей с технологией MPLS// Системні дослідження та інформаційні технології.– 2006.–№ 4.–с.65-70.

УДК 004.8

МУРГА Н.А.

ПРИМЕНЕНИЕ НЕЧЁТКИХ НЕЙРОННЫХ СЕТЕЙ К ЗАДАЧЕ АНАЛИЗА ПОВЕДЕНИЯ КОТИРОВОК ВАЛЮТ

В статті приводяться результати порівняльного аналізу застосувань радиальної базисної нейронної мережі та нечіткої нейронної мережі, запропонованої автором у [1], до задачі прогнозувань денних котирувань валютних пар EUR/GBP, EUR/USD, USD/JPY, USD/CHF. В роботі також приводиться опис нейронної мережі [1] та метод її навчання.

Results of comparative analysis of application of radial-basis neural network and fuzzy neural network, was proposed by author in [1], for problem of forecasting daily currency EUR/GBP, EUR/USD, USD/JPY, USD/CHF quotes is proposed in the paper. The neural network [1] and its training method are also shown in the paper.

Введение

В работе [1] была представлена нечёткая нейросетевая архитектура, которой в качестве нечёткого логического вывода используется нечёткий вывод Такаги-Сугено ([2,3]). В консеквенте нечётких правил указанной нечёткой нейронной сети в качестве функциональных зависимостей входов сети от выходов используются тригонометрические полиномы ([2,3]). Обучение сети из работы [1] происходит не с целью минимизации единого для всей сети критерия качества, а для каждого правила нечёткой системы формируется своё подмножество множества точек обучающей выборки и каждое правило обучается отдельно, с целью оптимизации своего критерия качества. В работе [1] исследовалось применение сети к задаче прогноза значений котировок валютных пар на основании их предыстории. В качестве котировок были взяты котировки EUR/GBP, EUR/USD, USD/JPY, USD/CHF из [4].

Далее, с целью сокращения названия, метод обучения, предложенный в [1], и нейросетевая архитектура, рассматриваемая в [1], будет называться – Нечёткая нейросетевая архитектура с независимым обучением каждого правила – ННСА НОКП – и в ходе повествования будет использоваться эта аббревиатура.

Данная работа посвящена сравнительному анализу применения радиальной базисной нейронной сети и ННСА НОКП к той же задаче, что и в работе [1] при той же выборке данных [4]. Следует отметить, что описание радиальной базисной сети в данной работе приводиться не будет вследствие ограничений на объём статьи, однако можно найти её описание в работе [5].

Структура работы построена следующим образом. В первой части статьи, которая носит название «Нечёткая нейросетевая архитектура с независимым обучением каждого правила логического вывода», приводится общая методика обучения нейронных сетей, после чего описывается модификация общей методики, а также предлагаются к рассмотрению способы оценки качества работы обученных нейронных сетей. В этой же части описывается ННСА НОКП. Во второй части, которая носит название «Результаты применения ННСА НОКП и радиальной базисной сети к анализу котировок валют» представлен сравнительный анализ применения к вышеуказанной задаче ННСА НОКП и радиальной базисной нейронной сети. В данной части обосновывается адекватность проведения такого сравнительного анализа и излагаются результаты сравнения сетей по ряду параметров. Несмотря на то, что в статье рассматривалась только задача прогнозирования значений котировок валютных пар, целевая аудитория данной

работы – это специалисты в области анализа временных рядов, специалисты в области интеллектуальных систем, а также аналитики финансовых рынков.

Нечёткая нейросетевая архитектура с независимым обучением каждого правила логического вывода

Пусть будет выбран для определённости нечёткий вывод Такаги-Сугено ([2,3]). Рассуждения для данной системы нечёткого логического вывода легко переносятся на другие типы систем нечёткого логического вывода.

С целью оценки качества функционирования нейронной сети и качества её обучения, вся выборка данных, на которых будет функционировать сеть, разбивается на обучающую выборку и проверочную. На обучающей выборке происходит обучение сети на проверочной, соответственно, проверка способности сети обрабатывать вновь поступающие данные. При этом качество оценивается по определённому критерию. Обычно качество функционирования нейронной сети оценивается с помощью критерия ***RMSE***. Для дальнейших выкладок необходимо ввести слідующие обозначения: n – количество точек выборки данных, y_i – выход модели, который должен получаться при i-х входных значениях и это значение взято из выборки; $\overline{y}_i$ – значение, которое выдаёт система обработки данных выборки (в данном случае – нечёткая система) при тех же входных значениях. Согласно введённым обозначениям критерий ***RMSE*** определяется формулой (1):

$$\boldsymbol{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}\left(y_i - \overline{y}_i\right)^2}, \qquad (1)$$

Однако данный критерий является абсолютным критерием, а это означает, что он является чувствительным к масштабу данных. Критерием, который не чувствителен к масштабу данных, является относительны критерий, например ***MAPE***, который определяется формулой (2):

$$\boldsymbol{MAPE} = \frac{1}{n}\sum_{i=1}^{n}\frac{\left|y_i - \overline{y}_i\right|}{y_i}. \qquad (2)$$

Обучение сети происходит с целью минимизации другого, упрощённого для выполнения расчётов критерия, например, приведённого в формуле (3):

$$E = \frac{1}{2}\sum_{i=1}^{n}\left(y_i - \overline{y}_i\right)^2. \qquad (3)$$

Приведённый критерий удобен тем, что функция E дифференцируема и при взятии производной от неё по определённому аргументу коэффициент будет равен 1.

Классический подход к обучению нейронных сетей подразумевает обучение сети в целом по всей обучающей выборке. Таким образом, ищется усреднённая по всем точкам выборки данных закономерность. Следует, однако, отметить, что динамика процесса в одной части пространства состояний может подчиняться одной закономерности, а в другой части (или частях) – другой. Способом разрешения данной проблемы, собственно, который и предлагался в работе [1], является разбиение обучающей выборки данных на подвыборки и поиск своих закономерностей в каждой из подвыборок.

Формально это описывается следующим образом. Пусть A – множество точек выборки данных. A_k, где $k = \overrightarrow{1,K}$ – подмножества множества A, которые удовлетворяют следующим условиям: $A = \bigcup_{k=1}^{K} A_k$ и $A_i \cap A_j = \varnothing$ при $i \neq j$. E_{A_k} – значение критерия (3) для подмножества A_k множества точек выборки. Подход предполагает обучение сети путём $\min E_{A_k}, \forall k = \overrightarrow{1,K}$ в отличие от классического подхода $\min E_A$.

Кроме того, данный метод настройки параметров нейронной сети позволяет получить следующее свойство: каждое частичное описание в консеквенте каждого нечёткого правила имеет свой собственный порядок (имеется в виду – порядок функции) и данный порядок настрашивается в ходе обучения сети.

Схема ННСА НОКП представлена на рис.1.

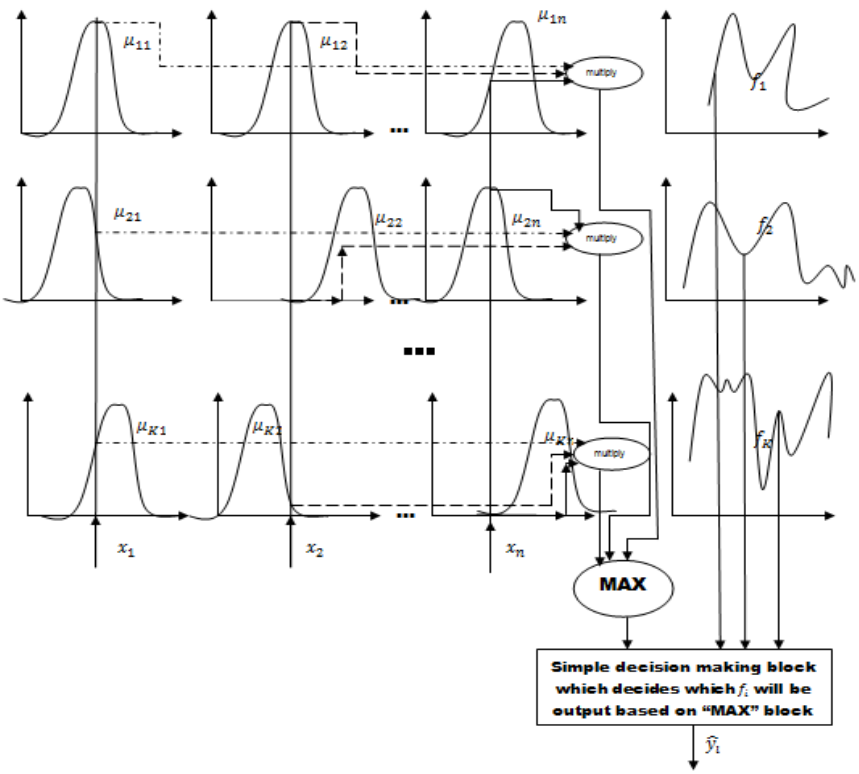


Рис. 1. Система нечёткого логического вывода Такаги-Сугено

Прежде чем переходить к объяснению рисунка, необходимо пояснить некоторые символы, которые в нём используются.

Выборка данных представляет собой матрицу размерности $L \times (n+1)$, где n – число входов сети (размерность пространства состояний), а L – количество точек выборки данных. Данная матрица имеет следующий вид:

$$A = \begin{pmatrix} x_{11} & \dots & x_{1n} & y_1 \\ \dots & \dots & \dots & \dots \\ x_{L1} & \dots & x_{Ln} & y_L \end{pmatrix}. \qquad (4)$$

Если будет подразумеваться некоторая (неважно какая) точка i выборки данных, то вместо $x_{i1}, x_{i2}, \dots, x_{in}, y_i$ будет записано $x_1, x_2, \dots, x_n, y$.

μ_{kj} – это функции принадлежности определённого признака j терму k_j.

В данной работе будут использованы функции принадлежности гауссовского вида, то есть μ_{kj} описывается выражением (5):

$$\mu_{kj}(x_j) = e^{-\frac{(x_j - c_{kj})^2}{2\sigma_{kj}^2}}. \qquad (5)$$

В качестве функций $f_k(x_1, \dots, x_n), k = \overrightarrow{1, K}$ будут использованы тригонометрические полиномы, которые описаны в [1] и [3], и восстанавливаться данные функциональные зависимости будут с помощью МНК по процедуре, описанной в [1]. Следует ещё раз отметить, что каждое частичное описание $f_{kj}(x_j)$, такое что $f_k(x_1, \dots, x_n) = \sum_{j=1}^{n} f_{kj}(x_j)$, имеет собственный порядок зависимости.

Таким образом, обучение сети состоит в нахождении таких c_{kj}, σ_{kj} и $f_k(x_1, \dots, x_n)$, которые бы наилучшим образом описывали динамику рассматриваемого процесса.

Процесс обучения предлагается строить следующим образом.

Первоначально происходит кластеризация той части выборки данных, которая соответствует входам сети. Для этого применяется сначала метод разностного группирования ([2,3]) для первоначально-

го задания центров кластеров, а после – нечёткий метод k-средних ([2,3]). Так определяются параметры c_{kj} функций принадлежностей, которые описываются формулой (5). Также нечёткий метод k-средних позволяет получить степень принадлежности каждой точки выборки $x_{i1}, x_{i2},..., x_{in}$ каждому полученному правилу. По максимальной степени принадлежности правилу происходит разбивка точек выборки на множества A_k, где $k = \overrightarrow{1,K}$, которые были описаны ранее.

Значения σ_{kj}, как отмечается в работах [2] и [3], влияют на качество функционирования сети незначительно. В текущей работе для каждого терма k_j каждого правила они будут выбраны свои, но одинаковые – это будет сделано для ускорения процесса обучения сети.

Однако для полноты выкладок можно предложить следующий метод нахождения σ_{kj}.

Так как множества A_k, $k = \overrightarrow{1,K}$, уже сформированы, то для каждого правила построение σ_{kj} и $f_k(x_1,..., x_n)$ будет происходить отдельно, основываясь на выборке данных, представленной множеством A_k.

На рис. 1 показано, что для определения k, а соответственно f_k, которое будет вычислять значение $\overline{y}_i$, выполняется следующая процедура:

$$k = \arg\max_S \prod_{j=1}^{n} \mu_{sj} \qquad (6)$$

Таким образом, очевидно, что для каждого правила k при $(x_1,..., x_n) \in A_k$ значение $\prod_{j=1}^{n} \mu_{sj}$ должно быть как можно большим, а в противном случае – наименьшим.

Таким образом, для нахождения оптимальных σ_{kj}, задавшись некоторым $\varepsilon \to 0, \varepsilon > 0$ для каждого правила, в среднеквадратическом смысле необходимо найти решение системы (7) относительно σ_{kj}.

$$\begin{cases} \prod_{j=1}^{n} \mu_{kj}(x_{ij}) = 1, & if\ (x_{i1}, x_{i2},..., x_{in}) \in A_k, \\ \prod_{j=1}^{n} \mu_{kj}(x_{ij}) = \varepsilon, & else. \end{cases} \qquad (7)$$

Если взять натуральный логарифм от обеих частей равенств системы, то систему можно переписать в следующем виде:

$$\begin{cases} \sum_{j=1}^{n} \ln(\mu_{kj}(x_{ij})) = 0, & if\ (x_{i1}, x_{i2},..., x_{in}) \in A_k, \\ \sum_{j=1}^{n} \ln(\mu_{kj}(x_{ij})) = \ln \varepsilon, & else. \end{cases} \qquad (8)$$

С учётом формулы (5), выражение (8) можно переписать в виде:

$$\begin{cases} \sum_{j=1}^{n} \ln\left(e^{-\frac{(x_{ij} - c_{kj})^2}{2\sigma_{kj}^2}} \right) = 0, & if\ (x_{i1}, x_{i2},..., x_{in}) \in A_k, \\ \sum_{j=1}^{n} \ln\left(e^{-\frac{(x_{ij} - c_{kj})^2}{2\sigma_{kj}^2}} \right) = \ln \varepsilon, & else. \end{cases} \qquad (9)$$

С учётом того, что $\ln e^a = a$, и введя обозначения (10), (11) и (12) :

$$h_{kij} = -\frac{\left(x_{ij} - c_{kj}\right)^2}{2}, \quad (10) \qquad p_{kj} = \frac{1}{\sigma_{kj}^2}, \quad (11)$$

$$\begin{cases} g_{ki} = 0, & if \ \left(x_{i1}, x_{i2}, ..., x_{in}\right) \in A_k, \\ g_{ki} = \ln \varepsilon, & else, \end{cases} \quad (12)$$

получается:

$$\sum_{j=1}^{n} h_{kij} p_{kj} = g_{ki}, \ i = \overrightarrow{1, L}, \ j = \overrightarrow{1, n}. \quad (13)$$

Если ввести обозначения (14)-(16):

$$H_k = \left\| h_{kij} \right\|_{i=\overrightarrow{1,L}}^{j=\overrightarrow{1,n}}, \quad (14)$$

$$p_k = \left\| p_{kj} \right\|_{j=\overrightarrow{1,n}}, \quad (15)$$

$$g_k = \left\| g_{ki} \right\|_{i=\overrightarrow{1,L}}, \quad (16)$$

то выражение (13) можно переписать в матричном виде (17):

$$H_k p_k = g_k. \quad (17)$$

Таким образом, для нахождения σ_{kj} необходимо решить задачу квадратичного программирования (18)-(19) для каждого правила $k = \overrightarrow{1, K}$:

$$\min_{p_{kj}} \left(H_k p_k - g_k\right)^T \left(H_k p_k - g_k\right), \quad (18)$$

$$p_{kj} \geq 0, \ j = \overrightarrow{1, n}, \quad (19)$$

Следует отметить, что p_{kj} может равняться нулю, так как если $x_{ij} = x_{mj}$, $\forall i, m$, то $\sigma_{kj} \to \infty$, что означает $p_{kj} \to 0, \ \forall k$.

Методика решения оптимизационных задач квадратичного программирования детально описана в [6].

Методика восстановление зависимостей $f_k\left(x_1, ..., x_n\right)$ приводится в данной работе не будет, так как она описана в [1]. Только необходимо отметить, что тригонометрические полиномы являются квазилинейными функциями, а, следовательно, их коэффициенты могут быть восстановлены с помощью метода наименьших квадратов. Также необходимо ещё раз отметить, что каждая $f_k\left(x_1, ..., x_n\right)$ восстана - вливается на своём множестве A_k

На этом обучение сети завершается.

При поступлении на вход сети очередного $\left(x_{i1}, x_{i2}, ..., x_{in}\right)$ вычисляется $k = \arg\max_S \prod_{j=1}^{n} \mu_{sj}$ и $\overline{y}_i = f_k\left(x_{i1}, x_{i2}, ..., x_{in}\right)$.

Оценка качества работы сети производится на полной обучающей выборке A и проверочной выборке по значениям критериев (1) и (2).

Результаты применения ННСА НОКП и радиальной базисной сети к анализу котировок валют

Для оценки применимости вышеприведённых нейронных сетей и для сравнительного анализа разных показателей их работы, радиальная базисная сеть и ННСА НОКП применялись к задаче анализа значений валютных котировок на основании их предыстории.

Эксперимент был поставлен следующим образом (аналогичен постановке эксперимента в работе [1]). Параметрами эксперимента было соотношение обучающей выборки и проверочной, а также число входов сетей. Соотношение обучающей выборки принимало значение 10:90, 20:80, 30:70, 40:60, 50:50, 60:40, 70:30, 80:20, 90:10 – в соотношении на первом месте стоит процент объёма обучающей выборки от объёма всей выборки данных, а на втором – процент проверочной выборки от всей выборки данных. Входами сетей было значение котировки валюты для 2, 3 и 4 периодов времени до прогнозного значения, а выходом сети было значение котировки валюты, которое необходимо

спрогнозировать. Эксперимент ставился для котировок валютных пар EUR/GBP, EUR/USD, USD/JPY, USD/CHF, которые были взяты из [4].

Анализ значений критериев (1) и (2) на обучающих выборках показал, что качество обучения сетей было приблизительно одинаково, о чём свидетельствует «близкость» значений критериев для обоих сетей при одинаковых параметрах эксперимента. Это даёт основание для проведения данного сравнительного анализа.

Исходя из значений тех же критериев на проверочных выборках, было продемонстрировано одинаковое их поведение при сравнении обоих сетей и значения были очень близки друг к другу при одинаковых параметрах эксперимента – сети имеют одинаковую прогностическую способность.

Следует отметить ещё раз факт, приведённый в работе [1].

Значения всех критериев для обеих выборок (обучающей и проверочной) для обоих сетей, при объёме обучающей выборки 10-20% от всей выборки данных, значительно отличаются от значений при других объёмах обучающей выборки. Как отмечается в работе [1], и как подтверждают результаты применения радиальной базисной сети, это вызвано тем, что при столь малых объёмах обучающих выборок растёт влияние каждой отдельной точки выборки на процесс обучения. Следовательно, выше (по сравнению со случаями других соотношений обучающей и проверочной выборки) влияние случайных шумов, не связанных с природой процесса на процесс обучения.

Исследования динамики рассматриваемых процессов, которые опираются на анализ поведения значений критериев качества (1) и (2) на обучающих и проверочных выборках, показывают незначительный прирост качества работы сетей при увеличении размерности пространства входных данных. Отсюда следует вывод, сделанный уже в работе [1]. При значении размерности входного пространства для нейронных сетей около 3 и 4, значения критериев начинают входить в «насыщение», и значения критериев качества позволяют сделать заключение, что размерность 4 можно считать «точкой насыщения». Однако данный факт ещё необходимо проверить в последующих исследованиях с усеченным диапазоном изменения параметра эксперимента «объём обучающей выборки».

В ходе проведённых опытов были отмечены «всплески» значений критериев качества на проверочной выборке при объёме обучающей выборки - 90%. Это объясняется (как отмечается также в работе [1]) тем же малым объёмом проверочной выборки, а, следовательно, большим влиянием каждой отдельной точки выборки на значение критерия. Таким образом, для оценки работоспособности сети «всреднем» данное соотношение выборок не является адекватным.

Анализ обеих сетей позволяет сделать вывод об их относительно стабильном поведении при соотношении обучающей и проверочной выборок 50:50 – 80:20.

Если анализировать скорость обучения сети, то радиальная базисная сеть была обучена значительно быстрее ННСА НОКП. Таким образом, по скорости обучения радиальная базисная сеть превосходит ННСА НОКП.

Однако применение сетей в реальном времени подразумевает их переобучение при поступлении новой информации при заданных ограничениях на время переобучения (минутные котировки, часовые, дневные…). Система прогнозирования, которая не удовлетворяет установленному ограничению, непригодна для применения в реальном времени. При поступлении новой точки выборки весовой вектор радиальной базисной сети ([5]) должен быть найден заново. Для ННСА НОКП определяется лишь принадлежность точки ваборки данных одному из кластеров, которые задаются, де-факто, правилами нечёткой системы. Точка принадлежит тому кластеру, для которого получается максимальная степень принадлежности антецеденте нечёткого правила, которое соответствует данному кластеру. После происходит процесс «доучивания» рекуррентным методом коэффициентов соответствую-

щей функциональной зависимости в консеквенте нечёткого правила. Выходит, что по параметру «способность к доучиванию» ННСА НОКП превосходит радиальную базисную сеть.

Если анализировать объём занимаемой сетями памяти, то с ростом обучающей выборки будет расти и объём памяти, занимаемой радиальной базисной сетью. Если же допустить конечное реальное количество кластеров точек обучающей выборки и при задании ограничения на рост порядка частичных описаний,

ННСА НОКП будет занимать ограниченный объём памяти вычислительной системы. По этому параметру ННСА НОКП превосходит радиальную базисную сеть.

Заключение

Работа была посвящена сравнительному анализу ННСА НОКП и радиальной базисной нейронной сети применительно к задаче прогнозирования котировок валютных пар на основании их предыстории. В качестве выборок данных были взяты котировки EUR/GBP, EUR/USD, USD/JPY, USD/CHF с [4].

Применение сетей к данной задаче позволило сделать ряд общих выводов. Первое, с увеличением размерности пространства данных, которые поступают на вход сети, прирост качества функционирования незначителен. Это позволяет допустить существования периода «насыщения» в зависимости «размерность пространства входных данных – качество функционирования прогнозирующей модели» для данной задачи. Исходя из значений критериев для обоих сетей, был сделан вывод, что при размерности 4 входного пространства прогнозирующая модель способна адекватно оценивать процесс. Второе, объём обучающей выборки влияет на качество работы сети как на обучающей, так и на проверочной выборке. Третье,– вывод подобен предыдущему – для оценки прогнозирующей способности сети, объём проверочной выборки не должен быть мал. Анализ динамики значений критериев качества показал, что соотношение обучающей и проверочной выборки 90%:10% от исходной выборки является неприемлемым для оценки прогнозирующей способности сети.

Сравнительный анализ применения обоих рассматриваемых в данной работе сетей дал следующие результаты. Подобный анализ – адекватен, так как соответствующие значения критериев на обучающих выборках были относительно близки друг к другу для обеих сетей – сети были одинаково хорошо обучены. На проверочных выборках соответствующие значения критериев были близки для обеих сетей. Таким образом, по качеству функционирования сети примерно идентичны. Что касается времени обучения, то ННСА НОКП уступает по этому параметру радиальной базисной нейронной сети. В то же время, при «дообучении» радиальной базисной нейронной сети нужно переобучать весь весовой вектор, что при больших размерностях может оказаться весьма проблематичным, когда для ННСА НОКП требуется переобучить лишь одно правило рекурентным методом при заданной стартовой («нулевой») точке. Так что, по этому параметру ННСА НОКП превосходит радиальную базисную нейронную сеть. Третьим фактом сравнительного анализа был следующий результат.

В то время, когда рост объёма поступающих данных сопровождается ростом объёма памяти вычислительной системы, которую занимает радиальная базисная нейронная сеть, рост объёма памяти, занимаемой ННСА НОКП, – ограничен. Данное ограничение связано с заданием верхнего значения порядка частичных описаний и количеством реально существующих кластеров выборки данных.

Безусловно, требуется проводить дальнейшие сравнительные анализы ННСА

НОКП с другими нейронными сетями. В последующих работах необходимо экспериментально исследовать полезность метода (18)-(19) для обучения параметра σ_{kj} для ННСА НОКП.

Необходимо расширение множества исследуемых выборок данных на другие типы финансовых рынков.

Литература

1. Мурга Н.А. Нейросетевая архитектура на частичных обучениях //New Trends in Classification and Data Mining. ITHEA (Sofia, Bulgaria). – 2010. – № 16 – С. 170-184.
2. Зайченко Ю.П. Основи проектування інтелектуальних систем. – К.: Видавничий Дім «Слово», 2004. – 352 с.
3. Зайченко Ю.П. Нечёткие модели и методы в интеллектуальных системах. – К.: «Издательский Дом «Слово»», 2008 – 344 с.
4. Дневные котировки валют за период с 25.03.2009 по 24.03.2010. http://www.finam.ru/analysis/export/default.asp.
5. Хайкин С. Нейронные сети: полный курс, 2-е издание.: Пер. с англ. – М.: Издательский дом “Вильямс”, 2006. – 1104 с.
6. Зайченко Ю.П. Дослідження операцій. Сьоме видання, перероблене та доповнене. – К.: Видавничий Дім «Слово», 2006. – 816 с.

УДК 519.854.2

КОСТИК Д. Ю.

ОПТИМІЗАЦІЯ ПРОГРАММНОЇ РЕАЛІЗАЦІЇ МОДИФІКОВАНОГО ПДС-АЛГОРИТМУ ЗАДАЧІ МІНІМІЗАЦІЇ СУМАРНОГО ЗАПІЗНЕННЯ ВИКОНАННЯ ЗАВДАНЬ

Наводиться порівняння програмних реалізацій різних версій, дається опис оптимізації програмної реалізації ПДС-алгоритму задачі мінімізації сумарного запізнення. На прикладах проілюстровано дві проблеми: неоптимальне використання оперативної пам'яті та обчислення тільки за допомогою одного ядра на багатоядерних комп'ютерах.

A comparison of different versions of software implementations, describes the software optimization of PDS algorithm to minimize total tardiness. The examples illustrate two problems: non optimal usage of memory and calculation using only one core on multicore computers.

1. Вступ

На практичну ефективність алгоритму впливають дві складові, а саме: ефективність математичного алгоритму та ефективність програмної реалізації. Аналізуючи обидві складові можемо отримати по-справжньому ефективний алгоритм та його програмну реалізацію. Найбільш проблемні місця у будь якого програмного продукту, який реалізує певний математичний алгоритм, це: витік пам'яті, неоптимальність простих операцій (впорядкування, вставка, і т.д.), занадто часте використання постійної пам'яті, використання тільки одного потоку або одного процесору, дуже часті рекурсивні виклики, мале навантаження на центральний процесор, невикористання графічного процесору для математичних операцій. У статті розглядається програмна реалізація алгоритму, яка біла наведена у [2].

2. «Мінімізація сумарного запізнення виконання незалежних завдань з директивними строками одним приладом» (МСЗ)

Припустимо, що задано множину незалежних завдань $J = \{j_1, j_2, \ldots, j_n\}$, кожне з яких складається з однієї операції. Для кожного завдання відома тривалість виконання l_j і директивний строк виконання D_j. Завдання надходять у систему одночасно в момент $d_j = 0, j = \overline{1, n}$. Переривання не допускаються. Необхідно побудувати розклад виконання завдань для одного приладу, що мінімізує сумарне запізнення при виконанні завдань:

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

де C_j – момент завершення виконання завдання j.

Огляд відомих методів розв'язання цієї задачі приведений у [1]. Згідно Ду і Люнгу, задача є NP-складною.

На кафедрі АСОІУ було запропоновано ефективний точний ПДС-алгоритм (алгоритм із поліноміальною й експоненційною складовими) розв'язання задачі, заснований на новому підході до розв'язання задач з директивними строками, що полягає в оптимальному використанні резервів часу незапізнених завдань, що дозволяє розв'язувати задачі з числом завдань, істотно більшим, ніж 500.

3. Ефективність програмної реалізації

Найбільш проблемні місця у будь якого програмного продукту, який реалізує певний математичний алгоритм, це:

- витік пам'яті;
- неоптимальність простих операцій (впорядкування, вставка, і т.д.);
- занадто часте використання постійної пам'яті;
- використання тільки одного потоку або одного процесору;
- дуже часті рекурсивні виклики;
- мале навантаження на центральний процесор;
- невикористання графічного процесору для математичних операцій.

Якщо розглянути всі алгоритми в сукупності, то їх можна поділити на дві категорії:

— ті, що використовують переважно пам’ять;

— ті, що використовують переважно процесор.

Щодо модифікованого ПДС-алгоритму, то його перша реалізація використовувала переважно процесор. Розмір всіх даних у оперативній пам’яті ледве сягав 20 мегабайт. Ще однією неприємністю було використання лише одного процесора – це пов’язано з структурою самого алгоритму. Математичний алгоритм є послідовним у всіх його місцях, тобто не містить блоків для розгалуження.

На рис. 1 показано навантаження на ЦП, та розмір оперативної пам’яті.

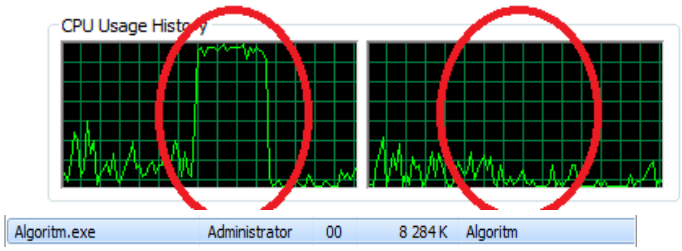


Рис. 1. Навантаження на ЦП та розмір ОП

Більшість персональних комп’ютерів сьогодні є багатоядерними та з підтримкою технології віртуальних ядер. Тобто навіть найдешевший ноутбук має 4-и ядра (2-а фізичних, та 2-а віртуальних). А якщо брати до уваги серверні комп’ютери, навіть одноюнітові, то в них кількість незалежних ядер може сягати 32. Тобто, запустивши алгоритм на такому комп’ютері, ми будемо використовувати лише 1/32 ресурсів, що не є оптимальним ні з точки зору використання такого комп’ютера, ні з точки зору швидкодії та часу вирішення великих прикладів. Розмір оперативної пам’яті такого комп’ютера може бути від 8 до 128 гігабайт. А як зазначалося вище алгоритм потребував лише 20 мегабайт. Тобто знову таки, маємо ситуацію таку ж як і з процесорами – ресурсів багато, а використовуємо лише маленьку частину.

Проаналізувавши ці дві проблеми, біли проведені дослідження алгоритму на можливість адаптації до розгалужування розрахунків, тобто виділення незалежних операцій в один момент часу, а також перенос навантажень з процесору на оперативну пам'ять.

Як зазначалося вище, і це проілюстровано на блок-схемі алгоритму [2], математично алгоритм не містить місць, які можна робити паралельно. Тобто всі етапи залежні один від одного та повинні виконуватися у чіткій послідовності. Так, єдиний варіант, який є можливим для розгалужування, – це впорядкування.

Продивившись та проаналізувавши детальні протоколи роботи алгоритму на задачах з різною кількістю завдань, було виявлено, що у деяких випадках, коли алгоритм занурюється у рекурсію, він робить оптимізацію на під інтервалі, на якому, можливо, вже виконувалась оптимізація минулими етапами алгоритму. Тобто є ймовірність того, що деякі дії повторюються. Це є невід’ємною рисою всіх NP-складних задач.

Було розроблено два додаткові етапи алгоритму:

— етап побудови таблиць з найкращими результатами;

— етап заміщення рекурсивних викликів вже оптимізованої послідовністю.

Таким чином отримали метод, який дозволяє значно знизити навантаження на центральний процесор, та перенести їх на оперативну пам’ять. Головна ідея цього метода в тому, що ми можемо заповнювати оперативну пам'ять тимчасовими проміжними даними. В цих даних зберігаються проміжні результати оптимальних підпослідовностей. Це дає змогу підставляти вже готові результати у випадку, коли ми вже робили такі оптимізації. Тобто, після кожного глобального циклу (одна ітерація), ми додаємо до оперативної пам’яті саму послідовність та вже отриманий оптимальний розклад. Таким чином формується таблиця з найкращими результатами для кожної підпослідовності.

Далі, коли є таблиця з кращими результатами, перед кожним рекурсивним кроком намагаємося знайти вже оптимізований розклад. Тут, замість того, щоб процесор робив обчислення рекурсивного виклику, йому достатньо знайти комірку в оперативній пам’яті, де зберігається вже оптимальний розклад. Дійсно, пошук у масиві даних, найчастіше, займає менше часу, ніж побудова оптимального розкладу за експоненційною складовою. Але продивившись протоколи отримані при розв’язані прикладів різної розмірності, виявилося, що у 24% прикладів, оптимізація займає менше часу, ніж пошук оптимального розкладу.

Наступний етап для покращення ефективності був етап розробки ефективного пошуку у таблиці тимчасових послідовностей. Таким

чином ми отримали наступну структуру для зберігання даних про тимчасові оптимальні результати:

— контрольна сума – число, яке є ідентифікатором послідовності;

— масив завдань – таблиця завдань підпослідовності;

— масив індексів початкової послідовності, мається на увазі ще не оптимізованої послідовності;

— масив індексів вже після оптимізації, тобто та таблиця, яку ми підставляємо замість рекурсивних викликів.

Після кожної ітерації зберігаємо такі відомості. Контрольна сума була введена для швидкого пошуку, щоб звести процесорне навантаження зовсім до мінімуму. При зберіганні послідовності по масиву індексів у початковій послідовності, ми рахуємо контрольну суму. А при пошуку в таблиці тимчасових результатів для послідовності, яку треба оптимізувати, ми рахуємо контрольну суму за тим самим алгоритмом (32-бітна контрольна сума), що були розраховані перед занесенням в таблицю. Таким чином, перший етап пошуку зводиться до порівняння лише одного 32 бітного числа. Другий етап залишився, тобто по елементний пошук, але він викликається тільки тоді, коли була знайдена вже невелика кількість збігань контрольних сум. Найчастіше маємо збігання один до одного.

Після аналізу ефективності нового методу і розподілу навантажень між процесором та оперативною пам’яттю, ми отримали 81% ефективності, у порівняні з частими рекурсивними викликами. Ще 10% припадають на приклади малої розмірності (до 50 завдань), тобто на ті, котрі вирішуються за прийнятний час. І які нема сенсу оптимізувати.

Одну проблему, а саме мале навантаження на оперативну пам’ять, вирішили. На рис. Рис. ***2*** показано завантаженість оперативної пам’яті комп’ютера.

Algoritm.vshost.exe	Administrator	50	201 084 K	vshost.exe

Рис. 2. Розмір ОП

На рис. 3 показано завантаженість оперативної пам’яті комп’ютера. Тобто, як можна переконатися з графіку, найбільшу частину пам’яті займає саме етап алгоритму 14.3, в якому відбувається запис до тимчасових таблиць.

Name	Bytes %
PDS.Core.PDS.nStep14_3(int32,int32,int32)	94,17
PDS.Core.PDS.PDSRec(int32,int32,int32,int32)	2,50
PDS.Core.PDS.FreeExchange(int32,int32)	2,41
System.BitConverter.GetBytes(int32)	0,73

Рис. 3. Розподіл пам’яті між етапами алгоритму

На рис. 4 показано, як розподіляється пам’ять між типами даних.

Name	Bytes %
Algoritm.Interface.Record[]	95,02
System.Boolean[]	2,41
System.Byte[]	2,38
Algoritm.Interface.AlgoritmBase.Comparer_L	0,14
PDS.Core.PDSRecResult[]	0,04

Рис. 4. Розподіл пам’яті між типами даних

На рис. 5 показано, як розподіляється звертання до пам’яті між типами даних.

Name	Instances %
System.Byte[]	50,22
Algoritm.Interface.Record[]	25,30
System.Boolean[]	18,67
Algoritm.Interface.AlgoritmBase.Comparer_L	5,78
System.String	0,01

Рис. 5. Розподіл звертань пам’яті між типами даних

На вищенаведених рисунках можна побачити, що найбільш вимогливими до оперативної пам’яті комп’ютера є тип даних Algoritm.Interface.Record[], в ньому зберігаються всі тимчасові таблиці з проміжними оптимальними результатами підпослідовностей. Але найчастіше пам'ять використовує тип даних Syste.Byte[], в ньому зберігаються індекси тимчасових таблиць, по яким відбувається пошук.

Для оптимізації алгоритму було використано дуже потужний інструмент для розробників програмного забезпечення Microsoft Performance Profiler, який входить до складу останньої версії Microsoft Visual Studio 2010. З його допомогою можна проаналізувати «вузькі» місця алгоритму, продивитися ієрархію викликів методів.

Як зазначалося вище, вузьким місцем алгоритму є навантаження тільки на онде ядро процесора (рис. 6).

При досліджені алгоритму на прикладах різної розмірності було отримані наступні дані (рис. 7, 8).

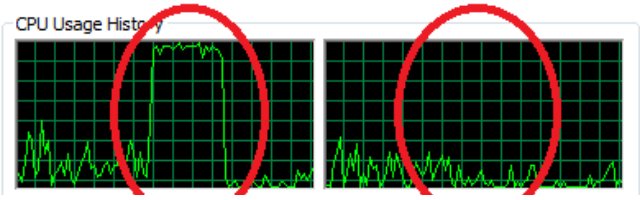


Рис. 6. Навантаження на ЦП

Function Name	Inclusive Samples %	Exclusive Samples %
PDS.Core.PDS.PDSRec(uint32,uint32,int32,int32)	21,92	0,00
PDS.Core.PDS.nStep14(int32,int32&,int32,int32)	8,22	2,05
PDS.Core.PDS.nStep14_3(int32,int32,int32)	6,16	0,00
PDS.Core.PDS.PDSRec(uint32,uint32,int32,int32)	5,48	0,00
PDS.Core.PDS.FreeExchange(uint32,uint32)	**4,11**	**4,11**

Рис. 7. Найбільш ресурсоємні функції

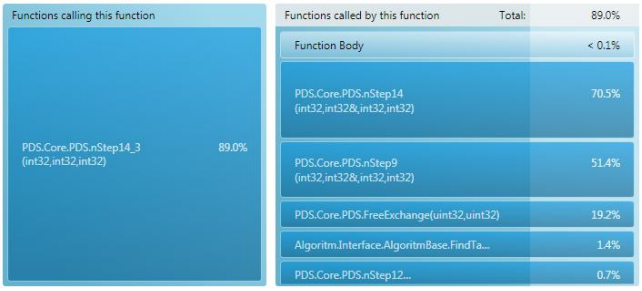


Рис. 8. Найбільш ресурсоємні функції

З протоколу можна побачити, що функцією, яка найбільш часто викликається, є nStep14_3, вона є точкою входу до рекурсивного виклику. Тобто саме з цього місця починається експоненційна складова алгоритму. Проаналізувавши багато прикладів різної розмірності, та той факт, що треба розгалужувати алгоритм, було запропоновано та реалізовано алгоритм тимчасових таблиць. Після впровадження алгоритму вдалося знизити навантаження на експоненційну складову алгоритму з 70% до 39%. Але проблема розгалужування ще не була вирішена.

Єдине місце, де це можливо було зробити, є пошук по тимчасовим таблицям. Так ми можемо перекласти частину навантаження за рахунок хоча б 11% пошуку по тимчасовим таблицям.

Для паралельного програмування була використана бібліотека PLINQ. Результати другої версії алгоритму наведені на рис. 9–10.

Function Name	Inclusive Samples %	Exclusive Samples %
PDS.Core.PDS.PDSRec(uint32,uint32,int32,int32)	77,23	0,00
PDS.Core.PDS.nStep14(int32,int32&,int32,int32)	34,65	3,96
PDS.Core.PDS.nStep14_3(int32,int32,int32)	26,73	0,00
PDS.Core.PDS.PDSRec(uint32,uint32,int32,int32)	25,74	0,99
System.Linq.ParallelEnumerable.ToList(class System.Linq.ParallelQuery`1...	**6,93**	**6,93**

Рис. 9. Найбільш ресурсоємні функції

З протоколів видно, що за рахунок паралельного пошуку нам вдалося знизити рекурсивні виклики на 5 відсотків (з 39% до 34%), та «перекласти» частину навантаження на 2-е ядро центрального процесора (рис. 11).

Name	Exclusive Samples %
PDS.Core.PDS.FreeExchange(uint32,uint32)	42,57
System.Linq.ParallelEnumerable.ToList(class System.Linq.ParallelQuery`1<!0>)	23,76
[clr.dll]	9,90
PDS.Core.PDS.nStep14(int32,int32&,int32,int32)	5,94
PDS.Core.Crc32..ctor()	2,97

Рис. 10. Найбільш ресурсоємні функції за індивідуальними викликами

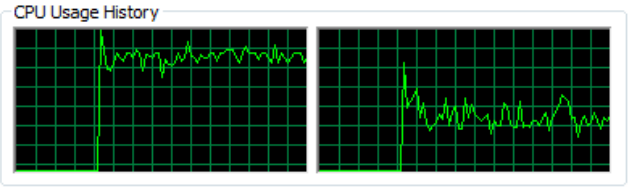


Рис. 11. Навантаження на ЦП

Такий алгоритм для пошуку проміжних результатів по тимчасовим таблицям залишимо у версії для прикладів з кількістю завдань до 200, та для задач з кількістю завдань більше 200 запропоновано дворівневий пошук. Це нагадує індексний пошук в сучасних реляційних базах даних, але є деякі відмінності, специфічні тільки для нашого випадку.

В якості індексу для пошуку було обрано контрольну суму, яка у нас все одно рахується для кожної послідовності. Але на відміну від першої реалізації, де ми використовували 16 біт для зберігання контрольної суми, в третій версії алгоритму було прийнято рішення використовувати 8 біт для зберігання контрольної суми. У цього рішення є як позитивні, так і негативні риси. До позитивних рис можна віднести:

— зменшення математичних операцій при підрахунку контрольної суми;

— менший об’єм оперативної пам’яті;

— прискорення пошуку (перебір 256 комірок, замість 65536).

До негативних рис відноситься:

— не унікальність контрольної суми;

— дворівневий пошуку.

Після визначення індексу пошук перетворився на дворівневий, за умов того, що ми використовуємо 8 біт для зберігання контрольної суми, а тимчасових таблиць, розмірність яких вище 200, більше ніж 256 можливих значень.

Першим кроком є пошук всіх однакових контрольних сум, які відповідають тимчасовим послідовностям. Ділі, коли ми вже отримали набір таблиць з однаковими контрольними сумами, серед них шукаємо необхідну нам таблицю, якщо така є.

У іншому алгоритму пошук заміни та вставки, такі самі, як і в першій його версії.

Результати, які вдалося отримати за рахунок нової оптимізації, наведені у протоколі рис. 12 – 13.

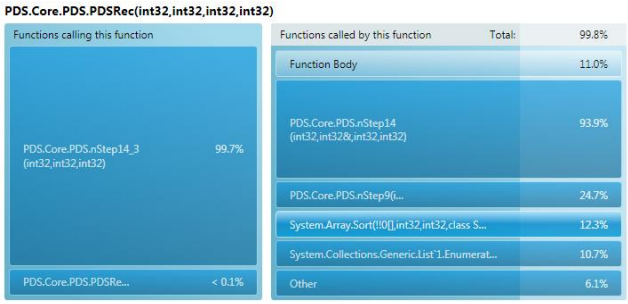


Рис. 12. Найбільш ресурсоємні функції

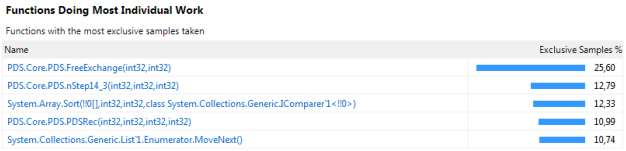


Рис. 13. Найбільш ресурсоємні функції за індивідуальними викликами

З протоколів видно, що за рахунок нової версії напошуку вдалося покращити показники швидкодії на 11 відсотків (з 23% до 12%).

8. Висновок

Показана ефективність модифікацій до ПДС-алгоритму. Показана різниця в розподілах оперативної пам'яті комп'ютера, та навантаження на центральний процесор та розподіл задач між багатоядерними системами. Розроблено методику для дослідження ефективності програмної реалізації алгоритму. Проведені експериментальні дослідження розв'язування генерованих задач за допомогою програмної реалізації ПДС-алгоритму, інтегрованої до розробленої системи моделювання, дозволили зібрати значиму статистичну інформацію як про класи вхідних задач, так і про активність використання окремих процедур алгоритму та провести оптимізацію програмної реалізації ПДС алгоритму.

Перелік посилань

1. Павлов А.А., Теленик С.Ф. Информационные технологии и алгоритмизация в управлении.– К.: Техника.– 2002.– 344 с., ISBN 966-575-045-3.
2. Система моделювання для дослідження ефективності ПДС-алгоритму задачі мінімізації сумарного запізнення виконання завдань // Павлов А.А., Місюра О.Б., Халус О.А, Беньковський С.Б., Лисецький Т.М., Костик Д.Ю. / Вісник НТУУ "КПІ". Інформатика, управління та обчислювальна техніка. К.: "ВЕК+", 2007.– №47.– С.215-220
3. Павлов О.А., Місюра О.Б., Халус О.А. Модифікований ефективний ПДС-алгоритм рішення задачі мінімізації сумарного запізнення виконання незалежних завдань одним приладом / Радиоэлектроника и информатика.– Харьков: ХНУР, 2007.– №1(36).– С.21-23
4. Павлов А.А., Мисюра Е.Б. Эффективный точный ПДС-алгоритм решения задачи о суммарном запаздывании для одного прибора // Системні дослідження та інформаційні технології. – 2004.– №4.– С.30-59
5. Шейко В., Кушнаренко Н. Організація та методика науково-дослідницької діяльності: Підручник. – К.: Знання-Прес, 2002. – 295 с.
6. Fisher, M.L. (1976) A dual algorithm for the one machine scheduling problem. Mathematical Programming, 11, 229-251. Тестові задачі доступні в Інтернет на сторінці http://www.bilkent.edu.tr/~bkara.

УДК 004.415.2.043

САЛАПАТОВ В.І.

МЕТОДИ ОПТИМІЗАЦІЇ ПРОГРАМ НА РІВНІ МАШИННИХ КОДІВ

В статті наводяться методи оптимізації машинного коду на рівні мови Асемблер, які враховують систему команд та архітектурні особливості цільових процесорів. Крім того, створюється база даних операндів, які існують у програмно доступних регістрах, які можуть бути використані у наступних командах. Розглядається, також, оптимальне використання стеку стосовно мінімізації звернень до нього.

Methods of the machine code optimization on Assembler language level, which are used instruction set and architectural feature of target processors. Besides, the data base operands are created. This operands are exist in program accessed registers, which are can be used on the next commands. There are considered else the optimal stack using and minimal accessed time quantity to it.

1. Вступ

Процес породження оптимальних машинних кодів програм засобами компіляції дедалі стає все ще актуальною задачею особливо для спеціалізованих пристроїв, зокрема для цифрових сигнальних процесорів. Це обумовлено, з одного боку, вимогами швидкісної обробки даних у реальному часі, а з іншого – обмеженим розміром оперативної пам'яті. Родина цифрових сигнальних процесорів ADSP-21xx має низку архітектурних особливостей, які дозволяють суміщення кількох дій в одній команді. Сучасні засоби компіляції не забезпечують у повній мірі використання усіх таких архітектурних можливостей. Неефективно, також, використовується інформація у програмно доступних регістрах, яка міститься в кожний конкретний момент під час виконання програми. На фазі генерації кодів стандартний компілятор породжує надмірні команди пересилки даних у певні регістри, тоді як ця інформація може вже знаходитись в інших регістрах. Такі команди пересилки складають левову частку надмірності машинного коду програм. На даний час оптимізація машинних кодів програм поки що здійснюється на рівні мови Асемблер, що є досить важким, довготривалим та витратним процесом. При цьому найбільш критичні фрагменти машинного коду, як правило, перероблюються на мові Асемблер, що викликає великі труднощі й займає багато часу. Тому, створення компіляторів, які забезпечують породження оптимальних машинних кодів програм з урахуванням архітектурних особливостей процесорів та існуючої інформації в їхніх програмно доступних регістрах (ПДР) в кожний конкретний момент часу виконання програми, є актуальною задачею і являє собою як теоретичний, так і з практичний інтерес. Тому метою даної статті є опис методів породження оптимального машинного коду на останній фазі роботи компіляторів з мов програмування високого рівня.

2. Особливості машинно-залежної оптимізації програм для сигнальних процесорів родини ADSP-21xx

В системах компіляції теоретично розвинені та практично застосовуються методи машинно-незалежної оптимізації, які забезпечують оптимізацію виразів, циклів, підстановки з тимчасовими змінними, тощо. Перелічені прийоми значно покращують структуру програм на абстрактному рівні, але основна надмірність машинного коду програм виникає при його породженні на останній фазі генерації роботи компілятора. Для порівняння ефективності роботи компіляторів для цифрового сигнального процесора ADSP-2188, наведемо приклад простої програми, що складається з оператора присвоєння на мові С. Розглянемо команди, що породжують компілятори з одного боку фірми Analog Device, а з іншого - створеного за описаною методикою. Текст цієї програми на мові С має наступний вигляд.

```
main()
{//4
   int E;//6
     int A = 3;
     int B = 2;
     int C = 5;
     int D = 4;
    E = (A+B) * 5 + (C-B)*D;   //12      }
```

Компілятор фірми Analog Device породжує наступні оператори Асемблера.

```
AR =  3;    // A
I6=-5;MODIFY(I6+=M4); DM(I6+=M6)=AR;
AR=PASS 2; // B
I6= -4; MODIFY(I6+=M4); DM(I6+=M6)=AR;
AR=5; // C
I6 = -3; MODIFY(I6+=M4);DM(I6 +=M6)=AR;
AR=PASS 4; // D
I6= -2; MODIFY(I6+=M4); DM(I6 +=M6)=AR;
I6 = -4; MODIFY(I6+=M4); AY1=DM(I6+=M6); // B
I6 = -3; MODIFY(I6+=M4); AX1=DM(I6+=M6); // C
AR = AX1 - AY1;
I6 = -2; MODIFY(I6 +=M4);MY1=DM(I6+=M6); // D
MR = AR * MY1(SS);
I6 =  -5; MODIFY(I6 += M4);
AX0 = DM(I6 +=M6); // A
AR = AX0+AY1; MY0 =  5;
AY0 = MR0 ;
MR0 = AY0 ;  //-----------------------------
MR = MR+AR*MY0(SS);AR=MR0 ;  //-------------
I6 = -6;MODIFY(I6 +=M4); // ---------
DM(I6 += M6) = AR; // ---------E
```

Початкові оператори пов’язані з ініціюванням змінних і тому не підлягають оптимізації. Оператори, які виділені коментарями з пунктирами, новим компілятором спрощуються наступним чином.

```
I6 = -6;
MODIFY(I6 += M6);
MR=MR+AR*MY0(SS),
DM(I6+=M6)=MR0;
```

Як бачимо, шість операторів Асемблера, що породжуються фірмовим компілятором, замінюються на три оператори. Причому останній оператор об’єднує дії множення-додавання з пересилкою. Для більш складних програм ефективність коду буде ще більш відчутною.

Як було показано вище, фірмовий компілятор мови С не враховує ані архітектурних особливостей цифрових сигнальних процесорів родини ADSP-21xx, зокрема таких, як суміщення за певною логікою кількох дій в одній команді, умовне виконання команд та деякі інші, ані вже існуючої інформації в регістрах у кожній точці програми під час її виконанні. Інформація про особливості архітектури та системи команд цільового процесора повинна надаватись компілятору у вигляді спеціальної бази даних [1] з описом семантики кожної команди разом з операндами та способами адресації до них, розміщенням результатів виконання команди, а також зміни стану процесора. Особливість оптимізації машинних кодів полягає у тому, що необхідно описати та врахувати правила можливого суміщення кількох дій в одній машинній команді, організацію циклу повтору за лічильником та умовне виконання команд. Такі правила враховуються шляхом доповнення бази даних опису особливостей архітектури та системи команд додатковими атрибутами [2]. Також треба зазначити про особливості синтаксису Асемблера цифрових

сигнальних процесорів родини ADSP-21xx, де оператори використовують природну форму запису арифметичних виразів та їх умовне виконання. Отже основною задачею, що розглядається в даній статті є створення таких компіляторів, що породжують ефективні машинні коди.

3. Методи породження ефективних машинних кодів програм

При породженні машинного коду програм сучасні компілятори використовують перетворення їх у асемблерний еквівалент. Причому внутрішнє уявлення програми перетворюється в трьохадресовий код із стандартних фрагментів, для яких створені спеціальні універсальні програмні асемблерні фрагменти. Така універсальність породжує надлишковість машинного коду програм, оскільки для реалізації цих фрагментів використовуються наперед визначені ПДР.

Оскільки більшість команд процесорів для свого виконання використовують ПДР, то в процесі генерації машинного коду програм компілятор породжує підготовчі команди пересилок R_v^i в ці регістри певних даних як операндів для інших допоміжних команд, де v – певна змінна, а i – певний регістр. Таким чином, перед програмною реалізацією кожного наступного оператора програми маємо множину інформаційних ресурсів (ІР) $\{R_v^i\}$, яку можна використовувати як операнди для команд при реалізації цього оператора. Зауважимо, що використовуватись при цьому можуть усі припустимі ПДР. Для породження допоміжних команд спочатку здійснюється пересилка даних у вільні ПДР. Поряд з невикористаними ПДР вільними будемо вважати, також ті, інформація в яких в подальшому не буде використовуватись. Для позначки про зайнятість ПДР додано спеціальний атрибут у БД опису особливостей архітектури та системи команд цільового процесора [2]. Коли всі ПДР стають зайнятими, необхідно обрати один з існуючих ПДР, зберегти його в стеку та на його місце переслати потрібний операнд. Після програмної реалізації поточного оператора збережений у стеку ІР необхідно відновити. При зайнятості всіх наявних ПДР для чергової підготовки операнду в першу чергу обирається той регістр, ІР якого в подальшому буде використовуватись для породження машинного коду після всіх попередніх ПДР з відповідними ІР. В подальшому, якщо необхідно використати ще один ПДР, то для чергового заміщення наступного ПДР застосовується така сама методика, коли попередній заміщений ПДР не приймає участі в заміщенні. Такий перегляд здійснюється на фазі синтаксичного аналізу та машинно-незалежної оптимізації при формуванні внутрішнього подання програми. При цьому фіксується порядок використання змінної чи елементу масиву у вигляді номера оператора і обирається той ресурс, який найближчим часом не знадобиться. Такий порядок використання стеку мінімізує кількість звернень до стеку, код програми та швидкість її виконання в цілому. Це відбувається через те, що з кількох ПДР $R_i(n)$ з ресурсами R_v^i доцільно обрати той регістр, який вміщує ІР, що буде звільнений з них першим. В даному випадку аргумент n вказує на останнє використання ресурсу в регістрі R_i. Таким чином, порядковий номер оператора програми досить зручно і природно використовувати як аргумент n. Тому для заміщення ПДР обирається той $R_i(n)$, де n = min. Отже, першим методом оптимізації машинного коду є ефективне використання стеку за описаним алгоритмом.

Крім того, за наявністю потрібного ІР в одному з ПДР не потрібно готувати операнди допоміжними підготовчими командами пересилок. В разі розвинутої адресації в системі команд цільового процесора один і той самий ресурс може бути присутнім з різними методами доступу, тобто видами адресації. У цьому випадку застосовується реляційна операція пошуку за зразком для визначення потрібного ІР. При наявності множини однакових ІР з різними видами адресації необхідно обрати той ІР, для якого значення заданої цільової функції [3] буде мінімальним.

Формат команди з певною семантикою являє собою множину припустимих сполучень операндів, які визначають спосіб адресації до них. Формати команд описуються у БД опису системи команд і поєднують припустимі види адресацій для кожної конкретної команди мікропроцесора. У разі пошуку ІР для формування операнду в команді Асемблера використовується реляційна операція вибірки, яка відповідає, з одного боку, команді із заданою

семантикою, а з іншого – наявності потрібного операнду у ПДР. Другий метод оптимізації машинного коду передбачає багаторазове використання ІР, які формуються в процесі в ПДР в процесі породження машинних команд в кожному місці виконання програми.

Треба зазначити, що при породженні асемблерного еквіваленту програми кожен оператор мови Асемблер включає в себе операцію або сукупність операцій над певними операндами. Стосовно операндів, то в кожному місці програми вони або вже існують, або мають бути підготовлені за допомогою команд пересилок. Тобто, якщо усі необхідні складові для породження асемблерного еквіваленту програми знаходяться у БД ІР, то для створення оператора Асемблера необхідно виконати реляційну операцію з'єднання з тих відношень БД ІР, які відповідають обраному критерію оптимальності згідно із значенням цільової функції. Задача зводиться до пошуку в БД оптимальних ІР та наступного з'єднання їх разом із потрібною семантикою в оператор Асемблера. При цьому семантики операторів внутрішнього подання програми мають охоплювати семантики машинних команд цільового процесора. Тоді з'єднання такого головного оператора Асемблера можна представити як S $\{R_v^i\}$. За відсутністю необхідних ІР формуються підготовчі оператори Асемблера, які за наявності вільного ПДР готують операнд для головного оператора. За відсутністю вільного ПДР за допомогою реляційної операції пошуку за зразком знаходиться потрібний ПДР для заміщення та наступного відновлення. Таким чином, програмна реалізація оператора мови зводиться до породження асемблерних команд для заміщення ПДР із записом у стек, команд підготовки ІР, головних команд, що реалізують оператор мови та команди відновлення ІР із стеку. Такий порядок синтезу машинного коду програм дозволяє отримати оптимальну програму за обраним критерієм оптимальності.

4. Висновки

Проблема породження оптимального машинного коду за допомогою компіляторів, з одного боку залежить від ефективного використання архітектурних особливостей цільового процесора, а з іншого боку – від максимально повного використання ІР в кожному місці програми під час її виконання.

За даною методикою на факультеті обчислювальної техніки, інтелектуальних та управляючих систем в Черкаському національному університеті імені Богдана Хмельницького створено кросс-компілятор мови С для цифрового сигнального процесора родини ADSP-21xx, який на даний час знаходиться на дослідній перевірці і вже підтвердив справедливість обраної методики породження оптимального машинного коду програм. Машинні коді, які сьворюються через проміжну мову Асемблер, відповідають обраному критерію ефективності щодо мінімального машинного коду при максимальній швидкості виконання програм.

Список літератури

1. Салапатов В.И. Структура данных для описания семантики и синтаксиса команд целевой ЭВМ. Вісник національного технічного університету України «КПІ», Інформатика, управління та обчислювальна техніка, № 41. Київ, 2004, с. 191-198.
2. Салапатов В.І. Особливості підвищення якості програм у сигнальних процесорів. Вісник національного технічного університету України «КПІ», Інформатика, управління та обчислювальна техніка, №51. Київ, 2010. с.74-77.
3. Салапатов В.І. Структура системи синтезу оптимальних машинних кодів програм. Вісник Черкаського інженерно-технологічного університету. № 4. Черкаси. 2004. с. 128-132.

УДК 681.322

СЕРГИЕНКО А.М.,
ЛЕПЕХА В.Л.

СПЕЦИАЛИЗИРОВАННЫЙ КОНТРОЛЛЕР ЛОКАЛЬНОЙ СЕТИ

В статье рассмотрена реализация контроллера локальной сети на базе программируемой логической интегральной схемы, в котором с помощью автоматов реализованы протоколы UDP и ARP. Благодаря этому, контроллер обеспечивает быстродействующую пересылку данных между персональным компьютером и сопроцессором, в который встроен этот контроллер.

The implementation of the LAN controller based on FPGA is proposed, in which the UDP and ARP protocols are implemented on the base of the finite state machines. Such implementation of the controller provides the high-speed communication between PC and coprocessor, in which the controller is built in.

Введение

Благодаря своей высокой производительности, программируемые логические интегральные схемы (ПЛИС) часто используется в качестве сопроцессора к персональному компьютеру (ПК) для ускорения критических вычислений или как мощное периферийное устройство ввода, например, как процессор видеокамеры или аппарата УЗИ. Однако остается проблема организации быстрого ввода-вывода данных в/из ПЛИС.

Среди таких современных быстрых каналов ввода-вывода, как PCI Express, SATA, USB, Ethernet, последний обладает рядом преимуществ при подключении через него ПЛИС. Во-первых, при этом нет необходимости в разработке сложного специального драйвера, а дополнительное матобеспечение для PC – сравнительно простое и основано на стандартных файловых операциях ввода-вывода. Во-вторых, вычислительный модуль, устанавливаемый в ПЛИС для контроля шиной Ethernet, т.е. контроллер локальной сети, существенно проще в разработке, чем такой же модуль для PCI Express или SATA. В-третьих, шина Ethernet обеспечивает пропускную способность не ниже, чем у PCI Express или SATA, но существенно выше, чем у USB. Наконец, подключение сопроцессора через шину Ethernet дает возможность передавать данные на практически любую дистанцию и разделять сопроцессор между несколькими PC с различными операционными системами [1].

Для связи по интерфейсу Ethernet наиболее часто используется Интернет-протокол версии 4 (IPv4). Однако для соединения с большинством сопроцессоров достаточно применить более простой протокол передачи дейтаграмм UDP.

Благодаря меньшим накладным расходам на организацию и программирование протокола, UDP обеспечивает большую пропускную способность и меньшие затраты на внедрение, чем IPv4. Меньшая надежность соединения UDP теряет важность при непосредственном соединении сопроцессора и ПК. Чтобы упростить соединение с периферийным процессором через локальную сеть, он должен также иметь реализацию протокола определения сетевого адреса ARP [2,3].

Многие фирмы – поставщики ПЛИС предлагают также вычислительные модули контроллера доступа к шине Ethernet (Media Access Controller – MAC). Однако для полноценного доступа к сети через контроллер MAC необходим также модуль быстродействующего микроконтроллера с инсталлированной программой стека TCP/IP, работающей в среде на основе ОС Linux [4]. Однако из-за невысокого быстродействия такого стека (порядка 5 Мбит в секунду), а также необходимости покупки дорогостоящих лицензий на модуль MAC, микроконтроллер и матобеспечение (20-40 тыс. долл.) такое техническое решение не годится для многих приложений сопроцессора на базе ПЛИС.

В статье описывается специализированный контроллер локальной сети, который, благодаря встроенной реализацией протоколов UDP и ARP, обеспечивает при малых аппаратных затратах высокую пропускную способность при передаче данных между ПК и сопроцессором на базе ПЛИС.

Архитектура контроллера локальной сети

Физический уровень интерфейса Ethernet реализуется на микросхеме приемопередатчика локальной сети (ППЛС) со стандартными функциями приема-передачи синхронных битовых последовательностей. При этом после установки синхронизма приема, ППЛС выдает непрерывный поток тетрад или байтов принятых данных. Для настройки режима работы ППЛС имеет стандартный последовательный интерфейс MII.

Каждый пакет Ethernet на физическом уровне начинается 8-байтовой преамбулой 55…55D5h, которая служит для нахождения начала пакета и заканчивается контрольным кодом FCS, служащим для проверки целостности пакета. Пакет UDP, кроме этого, содержит физические адреса приемника и передатчика, тип пакета, его заголовок, номер протокола, контрольную сумму заголовка IP, IP-адреса, UDP-адреса, длину пакета, его контрольную сумму и данные пакета.

Контроллер MAC должен запрограммировать режим ППЛС, принимать пакеты данных от него и подавать синхронно на вход ППЛС пакеты на передачу. При приеме контроллер должен выбрасывать из потока преамбулу и код FCS, а при передаче – их вычислить и добавить в поток.

Контрольные суммы IP, UDP рассчитываются как сумма полуслов с дополнением до 1 с инверсией результата. Код FCS определяется как циклическая свертка CRC путем двоичной свертки полинома с битами пакета, начиная с первого переданного бита MAC-адреса приемника и кончая последним битом последнего переданного данного. Сверточный полином задан кодом 04C11DB7h. Начальное значение свертки CRC равно FF…FFh. Результирующий код FCS – это двоичная инверсия кода CRC, а его передача в канал начинается с бита, отвечающего старшей степени полинома, т.е. разряды передаваемых тетрад или байтов кода FCS должны иметь двоично-инверсный порядок. Корректный принятый пакет вместе с контрольным кодом FCS должен дать после своей свертки результат CRC, равный C704DD7Bh [2,3,5].

Проектирование контроллера

Для реализации логического уровня локальной сети были выбраны протоколы 10BASE-T и 100BASE-TX, пропускной способности которых достаточно для большинства приложений сопроцессора на базе ПЛИС. При этом прием и передача данных между ППЛС и контроллером MAC выполняются тетрадами. Причем используется только дуплексный режим работы контроллера, который обеспечивается практически во всех локальных сетях, оборудованных кабелями с витой парой.

Кроме протокола UDP, контроллер MAC реализует протокол ARP, что обеспечивает стыковку сопроцессора с ПК с использованием матобеспечения ОС ПК, благодаря чему существенно упрощается разработка и отладка приложений сопроцессора.

Контроллер MAC состоит из трех независимых блоков: блоков приема, передачи и управления ППЛС. Последний при начальной установке программирует необходимые настройки ППЛС и контролирует его работу.

Структура блока приема показана на рис.1.

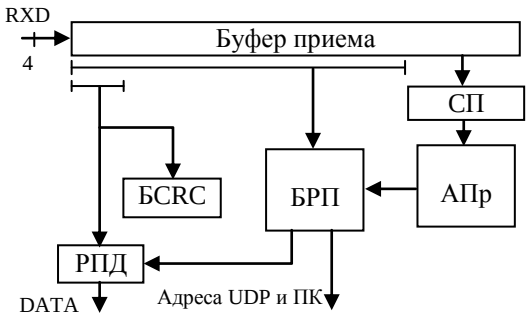


Рис.1. Блок приема контроллера MAC

Входные данные RXD непрерывно поступают в буфер приема, представляющий собой восьмибайтовый сдвиговый регистр. Селектор преамбулы (СП) срабатывает, если в буфере оказывается код преамбулы пакета и запускает автомат приема (АПр). АПр, кроме состояния ожидания пакета, имеет также состояния загрузки заголовка, загрузки данных и проверки FCS. В момент нахождения преамбулы в АП запускается счетчик принятых тетрад, состояние которого строго соответствует номеру тетрады заголовка пакета и передается в блок распаковки пакета БРП.

БРП управляет приемом пакетов с протоколами UDP и ARP. При этом в соответствующих тактах из буфера приема выделяются необходимые поля заголовка пакета, сравнение принятых адресов с адресами данного контроллера MAC, определяется тип протокола, число принимаемых данных. На выходы БРП в режиме ARP выдаются полученные Ethernet

адреса ПК. Если условия приема пакета совпали с заданными, то принятые байты данных пакета записываются в выходной буферный регистр РПД. Блок контроля циклической свертки (БCRC) выдает сигнал отмены принятых данных, если он обнаружил нарушение целостности пакета.

При приеме пакетов буферная память, кроме РПД, не требуется, так как устройства, подключаемые к контроллеру и реализованные в ПЛИС, всегда имеют быстродействие, достаточное, чтобы не допустить потери данных.

Структура блока передачи показана на рис.2.

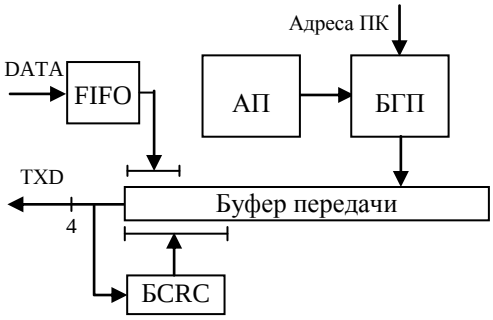


Рис.2. Блок передачи контроллера MAC

Для передачи пакета UDP инициатор его передачи запускает автомат приема (АП) и подает поток передаваемых данных DATA в буфер FIFO. АП определяет основные фазы передачи пакета и ведет отсчет тактов для блока генерации пакета (БГП). БГП формирует и подает в сдвиговый регистр буфера передачи преамбулу и заголовок пакета. Также он вычисляет и передает в буфер контрольную сумму заголовка. Блок контроля циклической свертки (БCRC) вычисляет контрольный код FCS передаваемых тетрад TXD, подставляемый в конец пакета. После приема контроллером пакета ARP с запросом адреса, БГП обеспечивает передачу ответного пакета ARP.

Экспериментальные результаты

Контроллер MAC был использован для соединения ПК и сопроцессора для решения задач линейной алгебры [6]. Также контроллер был использован в составе сопроцессора для ускорения выполнения криптографических алгоритмов.

Сопроцессор реализован на отладочной плате фирмы Avnet, содержащей ПЛИС XC4VSX-35 фирмы Xilinx и микросхему ППЛС DP83847 фирмы National.

Испытания контроллера показали его работоспособность и надежность соединения. При этом максимальная скорость пересылки пакетов (до 10 Мбайт в сек.) сдерживалась матобеспечением ПК. Результаты конфигурирования контроллера в ПЛИС Xilinx различных серий показаны в таблице.

Параметр	Spartan-3E	Virtex-4	Virtex-5
Аппаратные затраты, CLBs	810	956	408
Максимальная тактовая частота, МГц	200	330	390

Анализ данных в таблице показывает, что у схемы контроллера MAC есть большой запас быстродействия. Это означает, что после несложной доработки он может реализовать протокол 1000BASE-T, благодаря чему его пропускная способность увеличится вдесятеро. При этом следует лишь заменить формат данных в виде тетрад на байтовый формат.

Также планируется увеличение функциональности контроллера путем реализации протоколов стека TCP/IP. Чтобы упростить реализацию сложных протокольных алгоритмов, необходимо добавление программируемого микроконтроллера, такого как i8051. Микроконтроллер с такой архитектурой с быстродействием до 100 млн. команд в сек. описан в [7]. Авторами разрабатывается также специализированный микроконтроллер для реализации Интернет-протоколов, архитектура которого адаптирована к их реализации.

В дальнейшем планируется построение специализированного сетевого процессора для фильтрации опасных пакетов с использованием данного контроллера.

Выводы

Для соединения быстродействующего сопроцессора с ПК наиболее эффективно применение шины Ethernet, которая обеспечивает как высокую скорость передачи данных на любые расстояния, так и малые затраты на доработку матобеспечения ПК. Однако широко распространенные технические решения (готовые блоки соединения с Ethernet, вычислительные модули для ПЛИС) обладают недостатками, такими как низкая скорость передачи данных или высокая стоимость, а также

большие затраты на доработку матобеспечения.

Реализация протоколов UDP и ARP в специализированном контроллере шины Ethernet обеспечивает стыковку сопроцессора с ПК с использованием матобеспечения ОС ПК, благодаря чему достигается не только высокая скорость передачи данных, но и существенно упрощается разработка и отладка приложений сопроцессора.

Сопроцессор на базе ПЛИС со встроенным специализированным контроллером локальной сети может найти широкое применение для решения большого круга задач, как например, обработка видеосигналов, ультразвуковая диагностика, защита локальных сетей, ускорение задач сортировки и поиска, реализация сложных криптоалгоритмов и многие другие.

Список литературы

1. Alachiotis N., Berger S.A., Stamatakis A. Efficient PC-FPGA communication over Gigabit Ethernet // Proc. 7-th IEEE Int. Conf. on Embedded Software and Systems, ICESS'10. Bradford, UK. –2010. –288-296.
2. Axelson J. Embedded Ethernet and Internet Complete. Designing and Programming Small Devices for Networking. – Madison: Lakeview Research LLC. –2003. –482p.
3. Held G. Ethernet Networks: Design, Implementation, Operation, Management. –John Wiley & Sons. – 2003. – 583p.
4. Каршенбойм И. Контроллеры Fast Ethernet для встроенных применений// Компоненты и технологии. –2003. –№ 5. –с.12-19.
5. Borrelli C. IEEE 802.3 Cyclic Redundancy Check // XAPP209 (v1.0) –2001. –March 23. –8 p. Available at httm://www.xilinx.com
6. Sergiyenko A., Maslennikow O., Lepekha V., Tomas A., Wyrzykowski R. Parallel Implementation of Cholesky LL^T Algorithm in FPGA–Based Processor //Lecture Notes in Computer Science. –Berlin: Springer. –2008. –p. 137-147.
7. Сергієнко А.М., Лепеха В.Л. Деякі особливості проектування мікроконтролерів для СНК // Вісник НТУУ «КПІ», сер. Інформатика, управління та обчислювальна техніка. –Т.50. –2009. –с.70-73.

УДК 681.322

СЕРГИЕНКО А.М.,
ЛЕСИК Т.М.

РЕАЛИЗАЦИЯ ПЕРЕСТРАИВАЕМЫХ РЕКУРСИВНЫХ ЦИФРОВЫХ ФИЛЬТРОВ НА ПЛИС

Рассмотрены особенности рекурсивных цифровых фильтров, реализованными в программируемых логических интегральных схемах. Благодаря использованию фазовых фильтров, эффектов маскирования, кратных задержек, конвейеризации и разделения ресурсов, получены малые аппаратные затраты и высокая тактовая частота фильтров, а также возможность динамического изменения характеристик.

Features of the dynamically tuned IIR filters, which are configured in FPGA, are considered. The filters utilize the frequency masking properties of the all-pass digital filters, which have the delay factors z^{-k}. The mapping of the filter algorithm is implemented using pipelining and *C*-slow retiming techniques, which provide the minimized hardware volume and high clock frequency of the resulting filters.

Введение

В цифровой обработке сигналов часто используются рекурсивные цифровые фильтры (РЦФ), требующие перестройки в процессе своей работы. Настройка такого фильтра сводится к расчету коэффициентов РЦФ и записи набора коэффициентов в ПЗУ фильтра, которое для плавной перестройки фильтра должно иметь большой объем. Перестройка РЦФ представляет собой переходный процесс, который искажает выходной сигнал пропорционально добротности фильтра и шагу изменения его параметров [1].

РЦФ на базе программируемой логической интегральной схемы (ПЛИС) обладает высокой пропускной способностью. Но для возможности перестройки фильтра в состав вычислительной системы должен входить микроконтроллер для вычисления коэффициентов фильтра или память большого объема для их хранения, что существенно увеличивает аппаратные затраты.

В статье предлагается реализация динамически перестраиваемых РЦФ на базе фазовых фильтров, имеющих плавную перестройку своих параметров, благодаря использованию эффектов маскирования частот и фильтров с кратными задержками.

Маскирующие фильтры

При последовательном соединении ступеней фильтров результирующая амплитудо-частотная характеристика (АЧХ) является пересечением АЧХ этих ступеней. При этом говорят, что АЧХ ступени маскирует АЧХ других ступеней, т.е. такая ступень является маскирующим фильтром (рис. 1). Благодаря маскированию, результирующий фильтр, состоящий из простых ступеней фильтра, имеет высокое качество АЧХ [2,3].

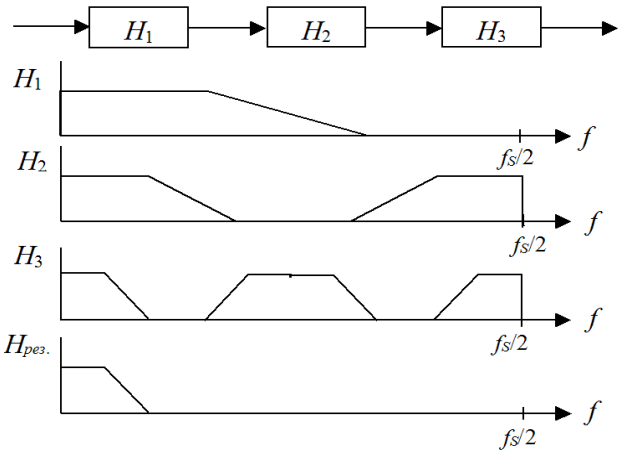


Рис.1. Пример трехступенчатого РЦФ с маскирующими фильтрами

Фильтры с кратными задержками

Каждому члену z^{-k} в передаточной характеристики $H_0(z)$ в сигнальном графе РЦФ отвечает задержка на k циклов или цепочка k регистров задержки в структуре РЦФ. Если в РЦФ количество регистров задержки увеличить в n раз, то получается фильтр с АЧХ $H_n(z) = H_0(z^n)$. АЧХ в этом фильтре имеет форму такую же, как у фильтра – прототипа $H_0(z)$, но в диапазоне $0 - f_S$ она повторяется n раз, где f_S – частота дискретизации [4]. На рис. 1

$H_1 = H_0(z)$, $H_2 = H_0(z^2)$, $H_3 = H_0(z^4)$.

РЦФ на базе фазовых фильтров

Фазовый фильтр имеет модуль передаточной функции $H(z)$, равный $|H(z)| = 1$. Его фазочастотная характеристика (ФЧХ) на частоте f_R имеет фазовый сдвиг, равный 180°. Если складываются сигналы от двух фазовых фильтров, выходной сигнал подавляется на частотах, для которых разность фаз равна 180°. Результирующая передаточная функция:

$$H_S = (H_1(z) \pm H_2(z))/2, \qquad (1)$$

соответствует различным фильтрам низких (ФНЧ), высоких частот (ФВЧ), полосовому или режекторному фильтру в зависимости знака суммирования (ФНЧ или ФВЧ) и от порядка фазовых фильтров [5].

РЦФ на базе фазовых фильтров отличаются устойчивостью при малой разрядности коэффициентов, высокой линейностью АЧХ и ФЧХ в полосе пропускания, а также высоким быстродействием. Параметры их АЧХ, такие как положение частоты среза, крутизна переходной полосы, прямо зависят от коэффициентов фильтра [1,5].

Динамически перестраиваемый ФНЧ

Для получения узкополосных фильтров с шириной полосы пропускания менее 0,05 от частоты дискретизации используют многоступенчатые схемы фильтров с децимацией и интерполяцией или двухшаговый гетеродинный перенос частоты с фильтрацией промежуточных частот [1,5]. Но комбинация узко- и широкополосных фильтров требует комплексных технических решений.

Для построения ФНЧ, перестраиваемого в широком диапазоне частот, предлагается структура фильтра на рис. 2, которая состоит из ступеней маскирующих фильтров $H(z),\ldots, H(z^k)$ и ступени фильтра-формирователя $H(z^k,a,b)$. При этом маскирующие фильтры выполняют фильтрацию частот с формой АЧХ, свойственной фильтрам-дециматорам, но непосредственно децимацию не производят (как H_1, H_2, H_3, на рис.1)[6]. Передаточная функция одной (первой) ступени маскирующего фильтра равна:

$$H_S(z) = 0{,}5z^{-1} + 0{,}5\frac{a + b(1+a)z^{-1} + z^{-2}}{1 + b(1+a)z^{-1} + az^{-2}}, \qquad (2)$$

где первое и второе слагаемое представляют $H_1(z)$ и $H_2(z)$ в (1), $b = \cos(2\pi f_R)$ – регулирует частоту среза f_R, $a = (1-t)/(1+t)$ – задает ширину переходной полосы Δf, $t = \mathrm{tg}(\pi\Delta f)$ [6]. Таким образом, изменяя b в (2), полоса среза регулируется в пределах (0,1 – 0,4) f_S с подавлением в полосе задержания до 50 дб.

Следует отметить, что все ступени фильтра выполняются по одному алгоритму, но с различными коэффициентами a,b и числом k регистров задержки.

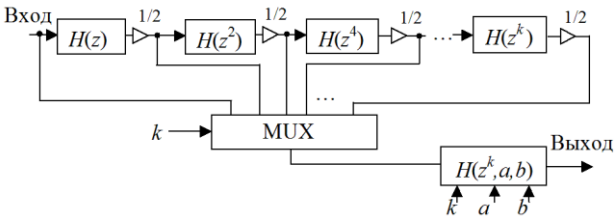


Рис.2. Структура ФНЧ с регулируемыми параметрами

Динамически перестраиваемый ФВЧ

ФВЧ с узкой полосой задержания может быть получен при вычитании результатов ФНЧ, описанного выше, из задержанного входного сигнала. Однако из-за неидеальности передаточной характеристики ФНЧ в полосе пропускания уровень подавления такого ФВЧ на практике не превосходит 30-40 дб.

Для получения ФВЧ с регулируемой шириной полосы задержания менее 0,05 от частоты дискретизации, кроме каскадирования ступеней, предлагается использовать свойство двойного взаимного дополнения АЧХ ФНЧ и ФВЧ по формуле (1). Два фильтра называются взаимно дополняющими, если АЧХ одного фильтра в полосе пропускания соответствует АЧХ другого фильтра в полосе задержания. Если еще суммарная энергия выходных сигналов этих фильтров равна энергии входного сигнала, то такие фильтры называются дважды взаимно дополняющими [5]. Таким образом, складывая отсчеты выходов таких ФНЧ и ФВЧ, получается сигнал, равный входному с точностью до ошибок вычислений.

На рис. 3 показана структура ФВЧ с дополняющим ФНЧ. Первая ступень фильтра разделяет полосу входного сигнала на две: нижнечастотную (верхняя ветвь) и высокочастотную (нижняя ветвь) полосы. Вторая ступень выполняет то же самое с нижнечастотной полосой первой ступени. Результирующий сигнал полосы верхних частот второй ступени формируется как сумма верхнечастотного сигнала второй ступени и задержанного сигнала полосы верхних частот первой ступени. Выходной сигнал ФВЧ на рис. 3 имеет полосу частот, равную 7/16 f_S, а сигнал ФНЧ – 1/16 f_S.

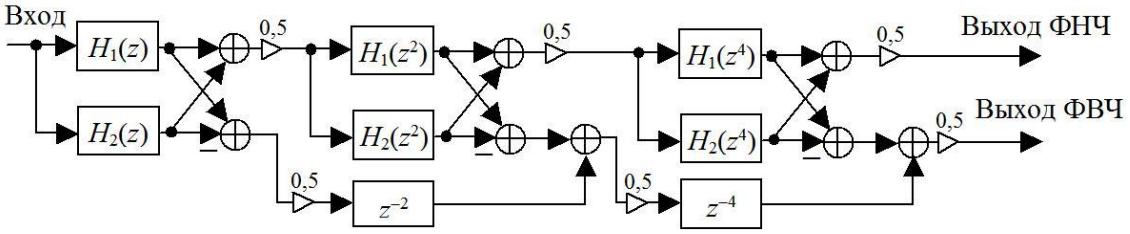


Рис.3. Структура ФВЧ с дополняющим ФНЧ

Регулировка полосы пропускания ФВЧ реализуется также, как в вышеописанном ФНЧ – путем перестройки параметров последней ступени и изменения числа ступеней. Во многих случаях, например, при построении банка фильтров эквалайзера, используются оба выхода этой структуры, что экономит апаратные затраты и обеспечивает верное разделение полос, благодаря их свойству двойного взаимного дополнения.

Поскольку данный ФВЧ имеет ступенчатую структуру со ступенями с k–кратными регистровыми задержками, то он может быть реализован, используя те же методы синтеза конвейерных схем, которые используются при построении вышеописанного ФНЧ.

Пример реализации динамически перестраиваемого ФНЧ

Для реализации проекта динамически перестраиваемого ФНЧ выбрано схему фильтра на рис.2. Чтобы получить уровень подавления более 80 дб и крутизну переходной полосы 100 дб/октава ступени фильтра реализуют передаточную функцию 10k–го порядка:

$$H(z^k)=\left[z^{-2k}+\frac{a+b(1+a)z^{-k}+z^{-2k}}{1+b(1+a)z^{-k}+az^{-2k}}\cdot\frac{c+z^{-k}}{1+cz^{-k}}\right]^2.$$

Здесь коэффициент c незначительно влияет на полосу среза и уровень подавления. Приближенное значение частоты среза и крутизны фильтра определяется аналогично (2).

При включении одной формирующей ступени фильтра (рис.2) частота среза регулируется в пределах (0,125–0,4)f_S, а при включении трех ступеней маскирующих фильтров – в пределах (0,015–0,031)f_S. При добавлении до восьми ступеней маскирующих фильтров возможно регулирование частоты среза в диапазоне от 20 Гц до 20 кГц при соответствующей частоте дискретизации.

Синтез структуры фильтра выполняется с использованием метода отображения периодических алгоритмов, представленных графом синхронных потоков данных, в конвейерный вычислитель, реализованный в ПЛИС. Метод заключается в размещении графа алгоритма в многомерном индексном пространстве и отображении его в подпространства структур и времени. Ограничения на процесс отображения позволяют минимизировать в ПЛИС как тактовый интервал, так и аппаратные затраты, включая мультиплексоры [6,7].

Синтезированный фильтр имеет три блока умножения на a,b,c, которые разделяются во времени между 4 ступенями фильтра. При этом структура фильтра работает с периодом 8 тактов с высокой степенью конвейеризации и поэтому – с высокой тактовой частотой.

Для вычисления коэффициентов a,b,c в зависимости от частоты среза f_R в структуру фильтра введен блок калькулятора этих коэффициентов. Частота среза f_R фильтра задается 12-разрядным кодом, 3 старшие разряда которого управляют мультиплексором на рис.2, а остальные – настройкой f_R. Новая частота среза устанавливается сразу после записи кода частоты и начинает действовать после соответствующего переходного процесса. Фильтр стабилен при всех возможных комбинациях кодов частоты и входных данных.

На рис. 4 показаны АЧХ данного фильтра в зависимости от кода частоты среза, которые свидетельствуют о его высоком качестве. АЧХ получены путем моделирования фильтров по методике, представленной в [8]. Фильтр имеет высокий уровень подавления и узкую переходную полосу при различных настройках. Неравномерность АЧХ в полосе пропускания – не более –0,23дБ. Характеристики фильтра, реализованного в ПЛИС Xilinx, представлены в табл. 1. Аппаратные затраты фильтра измеряются в числе эквивалентных конфигурируемых логических блоков (ЭКЛБ, CLB Slices) и числе блоков умножения DSP48. Тактовая частота достигает 280 МГц в ПЛИС Xilinx Virtex-5, т.е. фильтр может обрабатывать сигналы с частотой дискретизации до 35 МГц. Модуль данного ФНЧ, описанный на языке VHDL, размещен на сайте открытых вычислительных модулей [9] и может быть использован в разных проектах.

Табл. 1. Параметры фильтра при реализации в ПЛИС Xilinx Virtex

Микросхема ПЛИС	XC3 00A	XC5 VLX30
Аппаратные затраты, ЭКЛБ	616	339
Аппаратные затраты, DSP48	3	3
Максимальная тактовая часта, МГц	161	280
Максимальная частота дисетизации f_S, МГц	20	35

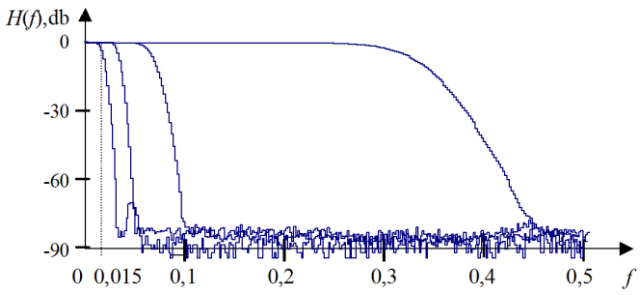


Рис.4. АЧХ фильтра при различных значениях кода частоты среза

Выводы

Благодаря использованию фазовых рекурсивных фильтров, эффектов маскирования частот и фильтров с кратными задержками появляется возможность проектировать динамически перестраиваемые РЦФ с высокими характеристиками. Синтез структуры РЦФ выполняется с использованием метода отображения периодических алгоритмов, представленных графом синхронных потоков данных. Такие фильтры эффективно реализовать в ПЛИС, где они имеют малые аппаратные затраты и высокое быстродействие, благодаря конвейерной обработке. Также они могут быть эффективно реализованы программно.

Список литературы

1. Оппенгейм А.В.,Шафер Р.В. Цифровая обработка сигналов. – М.: Связь. –1979. –416 с.
2. Lim Y.C., Lian Y. Frequency-response masking approach for digital filter design: Complexity reduction via masking filter factorization // IEEE Trans. Circuits Syst. II: Analog and Digital Signal Processing. –1994.–41. –N4. –P. 518–525.
3. Lu W.-S., Hinamoto T. Optimal Design of Frequency-Response-Masking Filters Using Semidefinite gramming // IEEE Trans. Circuits Syst. I – Fundamental theory and applications. –2003. –41. –N4. –Aug –P. 557–568.
4. Гольденберг Л.М., Матюшкин Б.Д., Поляк М.Н. Цифровая обработка сигналов: Справочник – М Радио и связь. –1985. –312 с.
5. Regalia P.A., Mitra S.K. and Vaidyanathan P.P. The Digital All-Pass Filter: A Versatile Signal Processing Building Block // Proc. IEEE. –1988.–V.76. –№1. –P.19–37.
6. Сергиенко А.М. Методика проектирования цифровых фильтров с помощью VHDL. //Моделювання та інформаційні технології. Зб. наук. праць. ІПМЕ НАН України. –2002. – 12. –С. 99-107.
7. Сергиенко А.М., Симоненко В.П. Отображение периодических алгоритмов в логические интегральные схемы //Электрон. моделирование. –2007. – 29. –№ 2. –С. 49–61.
8. Testbench for the filter testing. http://kanyevsky.kpi.ua/usefulcores_VHDLlab.html
9. Low-Pass IIR Filter : Overview //OpenCores project site. http://opencores.org/project,lp_iir_filter

УДК 62-82:658.512.011.56

ЛАДОГУБЕЦЬ В.В.,
ФІНОГЕНОВ О.Д.,
БЕЗНОСИК О.Ю.

ОСОБЛИВОСТІ ОПТИМІЗАЦІЇ МАКРОМОДЕЛЕЙ МЕХАНІЧНИХ КОМПОНЕНТІВ

В данной статье исследованы особенности целевых функций схемных макромоделей механических компонентов МЭМС, а также проведен анализ существующих средств САПР, которые могут решать задачи параметрической оптимизации. Экспериментально исследовано влияние качества и количества варьируемых параметров, диапазонов их изменения на результаты решения тестовых задач оптимизации схемных макромоделей механических компонентов МЭМС с помощью метода случайного поиска с уменьшением интервала поиска. Разработаны рекомендации по использованию блока оптимизации пакета NetALLTED для уточнения параметров схемных макромоделей.

In the paper presented the features of the objective functions of MEMS mechanical component circuit macromodels are investigated as well as the analysis of the existing CAD tools which can solve parametrical optimization tasks is carried out. The influence of a quality and a quantity of variable parameters, their change range on the results of MEMS mechanical component circuit macromodel optimization test tasks solution by means of the random search method with search interval reduction is investigated experimentally. The recommendations on usage of the NetALLTED optimization block to adjust circuit macromodel parameters are given.

1. Особливості цільових функцій схемних аналогів макромоделей МЕМС

В [1] запропоновано метод формування макромоделей механічних компонентів у вигляді еквівалентних скорочених RLC-схем для моделювання складених електромеханічних пристроїв пакетами схемотехнічного проектування.

У випадку, коли точність отриманої скороченої еквівалентної схеми заміщення не є достатньою при прийнятному для моделювання у пакетах схемотехнічного проектування розмірі, пропонується скористатися стандартними методами оптимізації, характерними для пакетів схемотехнічного проектування, для «підгонки» макромоделі у діапазоні частот, що розглядається [2, 3].

Уточнення параметрів схемних макромоделей МЕМС за допомогою оптимізаційних процедур з метою максимальної відповідності характеристик макромоделі вихідним характеристикам моделі МЕМС, отриманої за допомогою скінченно-елементного програмного забезпечення, неможливе без детального дослідження особливостей цільових функцій макромоделей МЕМС, аналізу існуючих на сьогодні засобів САПР, які можуть вирішувати задачі параметричної оптимізації, та ефективності вбудованих в них математичних алгоритмів. Крім цього, важливе значення має наявність в пакеті схемотехнічного проектування засобів оцінки вторинних параметрів вихідних характеристик та вибір чисельних параметрів оптимізаційних процедур. До числа найбільш важливих особливостей макромоделей неелектричних об’єктів варто віднести:

1. Великий (до 20 порядків) розкид значень елементів макромоделі неелектричного об’єкту. Це робить досить складним процес визначення величин інкрементів при розрахунку значень градієнтів цільових функцій, що, у свою чергу, може призводити до значних похибок при розрахунках. Тобто, враховуючи велику кількість локальних екстремумів цільових функцій, що застосовуються для оптимізації макромоделей неелектричних об’єктів, найбільш прийнятним методом оптимізації слід вважати метод випадкового пошуку зі зменшенням інтервалу пошуку [4].
2. Великий (до 30 порядків) розкид чутливості вихідних характеристик об’єкту до зміни окремих компонентів макромоделі, що передбачає “яружний” характер ЦФ.
3. Складний багатоекстремальний характер цільових функцій. Найчастіше це приводить до того, що значення ЦФ у знайденій точці локального оптимуму не задовольняє вимогам завдання, і виникає задача визначення координат іншого, більш придатного локального екстремуму. Крім цього, виникає велика залежність результату оптимізації від координат початкової точки оптимізації.
4. Порівняно невелика трудомісткість однократної оцінки цільової функції. Цей факт

також свідчить на користь методів оптимізації, що є найбільш простими з алгоритмічної точки зору, але найменш чутливими до вибору координат початкової точки та точності оцінки складових градієнтів.

Чисельні експерименти показали, що у якості ЦФ найбільш доцільно вибрати ЦФ виду [4]:

$$F8 = \sum_{i=1}^{m} \left(\frac{PV_i - y_i}{PV_i} \right)^2$$

де PV_i і y_i – задане та підраховане значення АЧХ, m – кількість параметрів, значення яких треба оптимізувати.

2. Можливості пакетів схемотехнічного проектування для рішення задач параметричної оптимізації

У таблиці 1 наведено інформацію, що відображує можливості блоків параметричної оптимізації чотирьох найбільш розповсюджених пакетів схемотехнічного проектування: NetALLTED (НТУУ "КПІ", Україна), HSpice (Synopsys, США), SPECTRE (Cadence, США), Saber (Synopsys, США) [5]. Аналіз цих даних свідчить, що пакет NetALLTED не тільки не поступається за своєю ефективністю, з точки зору можливостей блоку параметричної оптимізації, іншим пакетам, але найчастіше перевершує їх. Таким чином, з урахуванням особливостей цільових функцій схемних реалізацій макромоделей МЕМС, зроблено вибір пакету NetALLTED у якості засобу оптимізації макромоделей МЕМС.

Табл. 1. Можливості блоків параметричної оптимізації пакетів схемотехнічного проектування

Вид оптимізації	Кількість методів оптимізації			
	NetALLTED	HSpice	SPECTRE	Saber
Безумовна	8	1	1	-
Умовна	3	-	1	-
Мінімаксна	2	-	-	-

3. Опис тестових задач параметричної оптимізації схемних макромоделей механічних складових МЕМС

Модель закріпленої однорідної балки з одним ступенем свободи (Схема 1)

На рис. 1 представлено схемну макромодель однорідної балки з одним ступенем свободи, отриману при $\tau_{min}=3*10^{-5}$, та її опис на вхідній мові NetALLTED [4].

Для цієї схемної макромоделі необхідно, змінюючи параметри будь-яких елементів схеми, досягти того, щоб перша та друга резонансні частоти мали значення 1336.3 Гц та 4009.3 Гц відповідно, тобто співпадали з розрахованими за допомогою пакету ANSYS з сумарною похибкою не більше 0,1%. Значення резонансних частот, які є аргументами відповідної цільової функції (ЦФ) типу F8, сама ЦФ та необхідна передаточна функція на вхідній мові NetALLTED наведені на рис. 2.

```
Circuit Beam;
J1(100,0)=-100;
C1(82,100)=-0.11667;
L10(100,0)=-34161.8;
L11(23,50)=1.35e-10;
L12(23,0)=1.15e-10;
L13(50,0)=-16435.9;
L14(50,82)=1.6e-10;
C2(100,0)=6.3;
C3(23,50)=-0.11667;
C4(23,0)=17.3833;
C5(50,82)=-0.11667;
C6(0,50)=20.65;
C7(0,82)=17.5;
L8(82,100)=9e-11;
L9(82,0)=-580750;
```

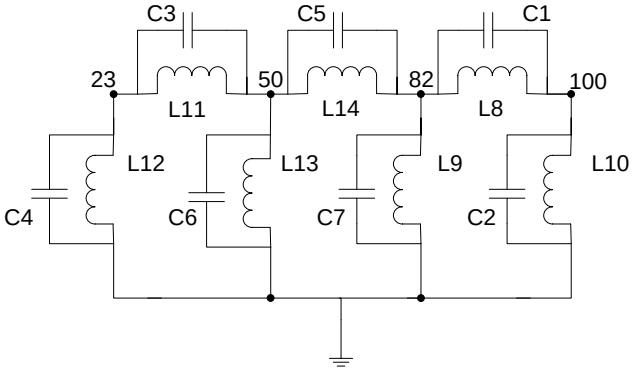


Рис. 1. Схемна макромодель однорідної балки із одним ступенем свободи

```
TF K1=UJ1/IJ1;
FIX T1=MAXA(db.K1,100,1600);
FIX T2=MAXA(db.K1,1700,4100);
OF ERROR1=F8(1336.3,4009.3/T1,T2);
```

Рис. 2. Фрагмент завдання на моделювання на вхідній мові ППП NetALLTED

Модель закріпленої однорідної мікробалки з трьома ступенями свободи (Схема 2)

На рис. 3 представлено схемну макромодель однорідної мікробалки з трьома ступенями свободи, отриману при $\tau_{min}=8*10^{-4}$, а на рис. 4 – її опис на вхідній мові ППП NetALLTED.

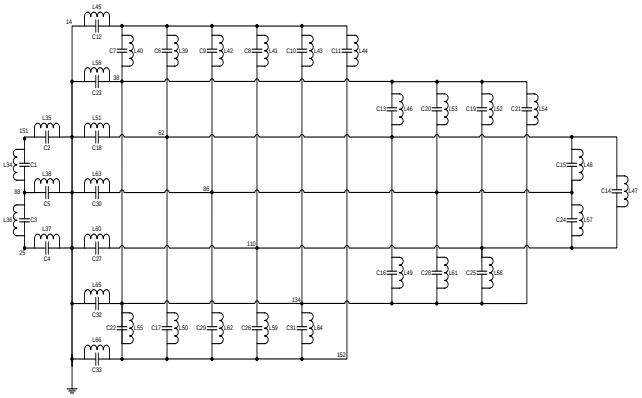


Рис. 3. Схемна макромодель мікробалки з трьома ступенями свободи

Для цієї схемної макромоделі необхідно, змінюючи параметри будь-яких елементів схеми, досягти того, щоб перші чотири резонансні частоти мали значення 18.745, 50.89, 97.859 та 115.49 МГц відповідно, тобто співпадали з розрахованими за допомогою пакету ANSYS з сумарною похибкою не більше 1%. Значення резонансних частот, які є аргументами відповідної цільової функції (ЦФ) типу F8, сама ЦФ та необхідна передаточна функція на вхідній мові NetALLTED наведені на рис. 5.

```
Circuit MICROBALKA_3D;  L40(14,38)=10.9703;
    J1(152,0)=-250;     L41(14,110)=-214.444;
    J2(151,0)=-250;      L42(14,86)=57.7349;
 C1(88,151)=-1.2e-08;   L43(14,134)=771.999;
C10(14,134)=-8.62e-12; L44(14,152)=-3377.49;
C11(14,152)=-9.13e-12;L45(14,0)=2.07521e+06;
C12(14,0)=5.82569e-07;  L46(62,38)=4.20676;
 C13(62,38)=2.084e-08; L47(62,110)=-8.93519;
C14(62,110)=-7.68e-09;  L48(62,86)=3.76376;
C15(62,86)=2.4426e-08;  L49(62,134)=32.1667;
C16(62,134)=2.593e-09;  C5(88,0)=1.512e-06;
C17(62,152)=-8.59e-10; L50(62,152)=-140.729;
C18(62,0)=5.83515e-07;L51(62,0)=1.57424e+07;
C19(38,110)=2.236e-09;  L52(38,110)=35.7407;
  C2(151,0)=7.56e-07;   L53(38,86)=-9.62251;
C20(38,86)=-6.497e-09; L54(38,134)=-128.667;
C21(38,134)=-7.36e-10;  L55(38,152)=562.917;
C22(38,152)=2.349e-10;L56(38,0)=-4.2182e+06;
C23(38,0)=5.49237e-07;  L57(110,86)=3.70797;
C24(110,86)=2.509e-08;  L58(110,134)=3.5059;
C25(110,134)=2.49e-08;L59(110,152)=-10.0521;
C26(110,152)=-8.7e-09;C6(14,62)=3.60298e-09;
C27(110,0)=5.8209e-07;L60(110,0)=2.1814e+08;
C28(86,134)=-7.82e-09; L61(86,134)=-8.57778;
C29(86,152)=2.786e-09;  L62(86,152)=37.5278;
  C3(25,88)=-1.2e-08;   L63(86,0)=-5.873e+07;
C30(86,0)=5.72711e-07; L64(134,152)=6.21245;
C31(134,152)=3.69e-08;L65(134,0)=-7.849e+08;
C32(134,0)=5.5842e-07;L66(152,0)=-3.394e+06;
C33(152,0)=1.7144e-07;C7(14,38)=-1.4409e-08;
L35(151,0)=-2.569e+14;C8(14,110)=1.0005e-10;
L37(25,0)=-2.6187e+14;C9(14,86)=-6.7338e-10;
L38(88,0)=-6.7123e+13;  L34(88,151)=2.1875;
 L39(14,62)=-15.4753;    L36(25,88)=2.1875;
  C4(25,0)=1.332e-06;
```

Рис. 4. Опис макромоделі мікробалки з трьома ступенями свободи

```
  TF K1=UJ1/IJ1;
  TF K2=UJ2/IJ2;
  FIX T1=MAXA(db.K1,15,25);
  FIX T2=MAXA(db.K1,40,60);
  FIX T3=MAXA(db.K1,85,105);
  FIX T4=MAXA(db.K2,106,120);
  OF
ERROR1=F8(18.745,50.89,97.859,115.49/T1,
  T2,T3,T4);
```

Рис. 5. Фрагмент завдання на моделювання на вхідній мові ППП NetALLTED

Модель пластини

Опис схемної макромоделі пластини (рис. 6), отриманий при $\tau_{\min}$ =5*10^{-5}, на вхідній мові ППП NetALLTED наведено на рис. 7.

Для цієї макромоделі необхідно, змінюючи параметри будь-яких елементів схеми, досягти того, щоб четверта резонансна частота мала значення 3427.8 МГц, тобто співпадала з розрахованою за допомогою пакету ANSYS з сумарною похибкою не більше 0.01%. Значення резонансної частоти, яка є аргументом відповідної цільової функції (ЦФ) типу F8, сама ЦФ та необхідна передаточна функція на вхідній мові NetALLTED наведені на рис. 8.

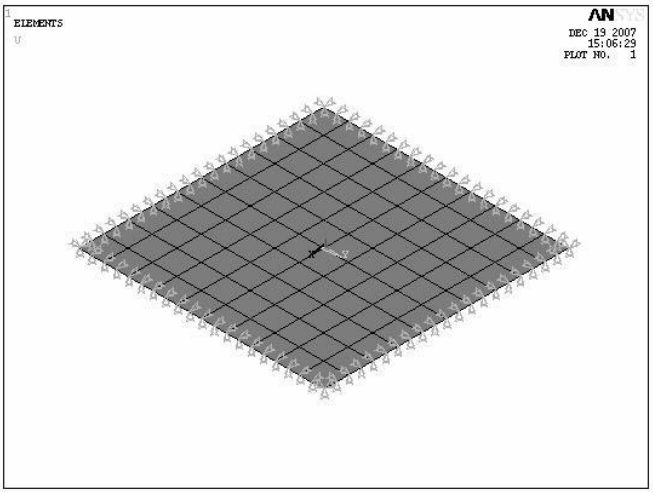


Рис. 6. Скінченноелементна модель пластини акселерометра

```
   Circuit membrana;      L22(681,0)=1749.7;
    J1(827,0)=100;      L23(681,827)=81.0134;
C1(749,681)=9.8187e-12;L24(681,1041)=293.884;
C10(1023,0)=9.1175e-11;  L25(1023,0)=1748.35;
C11(1023,827)=-1.4e-11;L26(1023,827)=81.0241;
C12(1023,1041)=9.3e-12;L27(1023,1041)=145.914;
C13(0,827)=3.39931e-10;  L28(0,827)=-769.548;
C14(0,1041)=9.1135e-11;  L29(0,1041)=1779.07;
C15(827,1041)=-1.3e-11;C3(749,0)=9.11471e-11;
 L16(749,681)=145.945;  L30(827,1041)=81.0423;
 L17(749,1023)=293.424;C4(749,827)=-1.450e-11;
   L18(749,0)=1743.16;  C5(749,1041)=9.051e-12;
 L19(749,827)=81.0146; C6(681,1023)=8.851e-12;
C2(749,1023)=-6.76e-12;C7(681,0)=9.11518e-11;
L20(749,1041)=-116.299;C8(681,827)=-1.258e-11;
 L21(681,1023)=-116.31;C9(681,1041)=-6.56e-12;
```

Рис. 7. Схемний аналог макромоделі пластини

```
  TF K1=UJ1/IJ1;
  FIX T1=MAXA(db.k1,3200,3500);
  OF ERROR1=F8(3427.8/T1);
```

Рис. 8. Фрагмент завдання на моделювання на вхідній мові ППП NetALLTED

4. Дослідження впливу кількості параметрів, що варіюються, та діапазону їх зміни на результати оптимізації схемних макромоделей механічних складових МЕМС

Успіх та витрати на вирішення задач параметричної оптимізації, незалежно від обраного методу оптимізації, в значні мірі залежать від якості та кількості обраних параметрів, що

варіюються. Крім цього, для методу випадкового пошуку зі зменшенням інтервалу пошуку важливим є також визначення інтервалу зміни кожного з таких параметрів.

Для вибору множини параметрів, що варіюються, пропонується використати метод оцінки чутливості цільових функцій до зміни окремих компонентів схеми. Ті компоненти схеми, чутливість цільової функції до зміни значень яких є максимальною серед усіх розрахованих, доцільно включати в множину параметрів, що варіюються.

За допомогою такого підходу для схем, що досліджуються, були вибрані наступні множини параметрів, що варіюються:

для Схеми 1: C5, C3, C4, C7, L8, L11, L12, L14;

для Схеми 2: L53, L61, L62, L64, L36, L34, L39, L40;

для Схеми 3: C1, C11, C12, C10, C15, C14, L16, L26.

Наступним етапом було дослідження впливу кількості параметрів, що варіюються, на результати оптимізації. При цьому для випадків, коли кількість таких параметрів становила менше 8, їх перелік обирався випадково із вказаних вище множин. Щоб нейтралізувати вплив розмірності вибірки для всіх випадків розмір вибірки N обирався рівним 200. Діапазон зміни кожного з параметрів обирався рівним ±20% від їх номінальних значень.

В таблицях 2-4 представлені результати оптимізації тестових задач в залежності від кількості параметрів, що варіюються (в цій та подальших таблицях: КО – кількість оцінок ЦФ, які були витрачені на отримання результату; ПЗЦФ, КЗЦФ – початкове та кінцеве значення ЦФ відповідно; позначка «**» означає, що бажане рішення не отримано).

Табл. 2. Залежність результату оптимізації від кількості параметрів, що варіюються, для Схеми 1 (N=200)

Кількість параметрів	ПЗЦФ	Назви параметрів	КО	КЗЦФ
2	$7.3*10^{-3}$	L8, L11	818	$7.3*10^{-3**}$
4		L8, L11, L12, L14	624	$5.45*10^{-7}$
6		L8, L11, L12, L14, C5, C3	494	$5.45*10^{-7}$
8		C5, C3, C4, C7, L8, L11, L12, L14	767	$5.45*10^{-7}$

Табл. 3. Залежність результату оптимізації від кількості параметрів, що варіюються, для Схеми 2 (N=200)

Кількість параметрів	ПЗЦФ	Назви параметрів	КО	КЗЦФ
2	$8.2*10^{-3}$	L34, L36	1022	$2.8*10^{-3**}$
4		L53, L61, L62, L64	626	$5.5*10^{-3**}$
6		L53, L61, L62, L64, L36, L34	850	$5.25*10^{-5}$
8		L53, L61, L62, L64, L36, L34, L39, L40	1082	$4.42*10^{-5}$

Ці дані свідчать про те, що для вирішення задач параметричної оптимізації схемних аналогів макромоделей механічних складових МЕМС доцільно у якості множини параметрів, що варіюються, обирати 4-6 компонентів з числа можливих.

Табл. 4. Залежність результату оптимізації від кількості параметрів, що варіюються, для Схеми 3 (N=200)

Кількість параметрів	ПЗЦФ	Назви параметрів	КО	КЗЦФ
2	$3.5*10^{-3}$	C10, C12	618	$9.02*10^{-4}$
4		C1, C11, C12, C10	438	$1.36*10^{-9}$
6		C1, C11, C12, C10, C15, C14	450	$1.36*10^{-9}$
8		C1, C11, C12, C10, C15, C14, L16,L26	454	$1.36*10^{-9}$

Наступні дані (таблиці 5-8) дають можливість оцінити вплив розмірів діапазонів можливих значень параметрів, що варіюються, на результати оптимізації. Для цих досліджень у якості множини таких параметрів обирались такі варіанти, які згідно таблиць 2-4 відповідали найкращим варіантам з точки зору як кількості оцінок цільових функцій, які було необхідно витратити для отримання рішення, так і якості отриманого результату (кінцеве значення цільової функції).

Табл. 5. Залежність результату оптимізації від відсоткового діапазону змін параметрів, що варіюються, для Схеми 1 (N = 100)

Відсотковий діапазон змін параметрів, що варіюються	ПЗЦФ	КО	КЗЦФ
10%	$7.3*10^{-3}$	242	$5.45*10^{-7}$
20%		326	$8.17*10^{-5**}$
30%		434	$5.74*10^{-5**}$
40%		434	$3.37*10^{-5**}$

Табл. 6. Залежність результату оптимізації від відсоткового діапазону змін параметрів, що варіюються, для Схеми 1 (N = 200)

Відсотковий діапазон змін параметрів, що варіюються	ПЗЦФ	КО	КЗЦФ
10%	$7.3*10^{-3}$	624	$5.45*10^{-7}$
20%		834	$9.65*10^{-6}$
40%		881	$5.45*10^{-7}$

Табл. 7. Залежність результату оптимізації від відсоткового діапазону змін параметрів, що варіюються, для Схеми 2 (N = 200)

Відсотковий діапазон змін параметрів, що варіюються	ПЗЦФ	КО	КЗЦФ
5%	$8.2*10^{-3}$	850	$5.25*10^{-5}$
10%		1062	$2.94*10^{-5}$
20%		1062	$5.31*10^{-4**}$

Табл. 8. Залежність результату оптимізації від відсоткового діапазону змін параметрів, що варіюються, для Схеми 3 (N = 200)

Відсотковий діапазон змін параметрів, що варіюються	ПЗЦФ	КО	КЗЦФ
10%	$3.54*10^{-3}$	438	$1.36*10^{-9}$
20%		438	$1.36*10^{-9}$
30%		438	$1.36*10^{-9}$

Наведені дані з високим ступенем ймовірності дозволяють зробити висновок, що для вирішення задач параметричної оптимізації схемних аналогів макромоделей механічних складових МЕМС найбільш доцільно обирати у якості інтервалів можливої зміни кожного з параметрів, що варіюються, інтервали, які дорівнюють ±10÷20% від значень відповідного компонента на початку оптимізації. Окремо слід відзначити, що при малих розмірах вибірки (табл. 5), результат оптимізації виявляється дуже чутливим до розмірів можливих інтервалів зміни параметрів, що варіюються. Навпаки, результати табл. 8 свідчать про те, що в окремих випадках зміна розміру можливих інтервалів значень параметрів, що варіюються, мало впливає на результат оптимізації.

5. Дослідження впливу розміру вибірки на результати оптимізації схемних макромоделей механічних складових МЕМС

Успіх вирішення задачі параметричної оптимізації за допомогою методу випадкового пошуку зі зменшенням інтервалу пошуку в значній мірі залежить від розміру вибірки на кожному кроці оптимізації. В таблицях 9-11 наведено результати оптимізації тестових задач за допомогою методу випадкового пошуку зі зменшенням інтервалу пошуку при зміні розміру вибірки.

Табл. 9. Залежність результату оптимізації від розміру вибірки для Схеми 1

Розмір вибірки	ПЗЦФ	КО	КЗЦФ
50	$7.3*10^{-3}$	276	$3.9*10^{-5**}$
100		242	$5.45*10^{-7}$
150		518	$5.45*10^{-7}$
200		624	$5.45*10^{-7}$
300		719	$5.45*10^{-7}$
400		2042	$2.43*10^{-7}$

Табл. 10. Залежність результату оптимізації від розміру вибірки для Схеми 2

Розмір вибірки	ПЗЦФ	КО	КЗЦФ
50	$8.2*10^{-3}$	250	$1.97*10^{-4**}$
100		562	$5.86*10^{-5}$
150		812	$9.05*10^{-5}$
200		850	$5.25*10^{-5}$
300		1250	$8.96*10^{-5}$

Табл. 11. Залежність результату оптимізації від розміру вибірки для Схеми 3

Розмір вибірки	ПЗЦФ	КО	КЗЦФ
50	$3.54*10^{-3}$	176	$5.34*10^{-6**}$
100		434	$3.32*10^{-7**}$
150		528	$1.36*10^{-9}$
200		669	$1.36*10^{-9}$
300		928	$1.36*10^{-9}$
400		1235	$1.36*10^{-9}$

Ці дані з високим ступенем ймовірності дозволяють зробити висновок, що для вирішення задач параметричної оптимізації схемних аналогів макромоделей механічних складових МЕМС за допомогою методу випадкового пошуку зі зменшенням інтервалу пошуку найбільш доцільно обирати розмір вибірки 100÷200.

На рис. 9-11 представлені АЧХ для всіх тестових задач до та після оптимізації.

6. Рекомендації щодо використання параметричної оптимізації для уточнення параметрів схемних макромоделей механічних складових МЕМС

Перевагою підходу, заснованого на оптимізації параметрів схеми, є малий розмір скороченої еквівалентної схеми, а також, при вузькому діапазоні робочих частот, можливість одержати необхідні частоти з високою точністю. Недоліком є необхідність проведення додаткового аналізу схеми для виділення найбільш чутливих елементів. При занадто малому розмірі скороченої схеми чутливість до кожного елементу буде зростати, що створює складності як для визначення діапазонів зміни параметрів, що варіюються, так і для роботи методів оптимізації при пошуку глобального оптимуму.

Результати проведених досліджень дають можливість сформулювати такі рекомендації щодо використання параметричної оптимізації для уточнення параметрів схемних аналогів макромоделей механічних складових МЕМС за допомогою пакету NetALLTED:

1. За допомогою аналізу чутливості обрати множину параметрів, що варіюються, у кількості 8-12 компонентів.
2. Випадковим чином з цієї множини вибрати 4-6 параметрів.
3. Призначити для кожного параметру, що варіюється, діапазон можливих змін у розмірі ±10÷20%.
4. Обрати значення максимально допусти-

мої величини цільової функції, виходячи з параметрів необхідної точності (як правило, 1.0E-4÷1.0E-6, що відповідає похибці аргументу цільової функції в 1÷0.1%).

5. За допомогою методу випадкового пошуку зі зменшенням інтервалу пошуку зробити спробу вирішити завдання, задавши розмір вибірки 100÷200.
6. Якщо у пункті 5 успіху не було досягнуто, змінити множину параметрів, що варіюються, вибравши їх з числа перелічених в пункті 1, і перейти до пункту 3.

7. Висновки

1. Детально досліджено особливості цільових функцій схемних макромоделей МЕМС та проведено аналіз існуючих на сьогодні засобів САПР, які можуть вирішувати задачі параметричної оптимізації.
2. Експериментально, за допомогою вирішення тестових задач оптимізації схемних макромоделей механічних складових МЕМС, досліджено вплив якості та кількості параметрів, що варіюються, діапазонів їх зміни та розміру вибірки на результати вирішення тестових задач за допомогою методу випадкового пошуку зі зменшенням інтервалу пошуку.
3. Розроблено рекомендації по використанню блоку оптимізації пакету NetALLTED для уточнення параметрів схемних макромоделей механічних складових МЕМС.

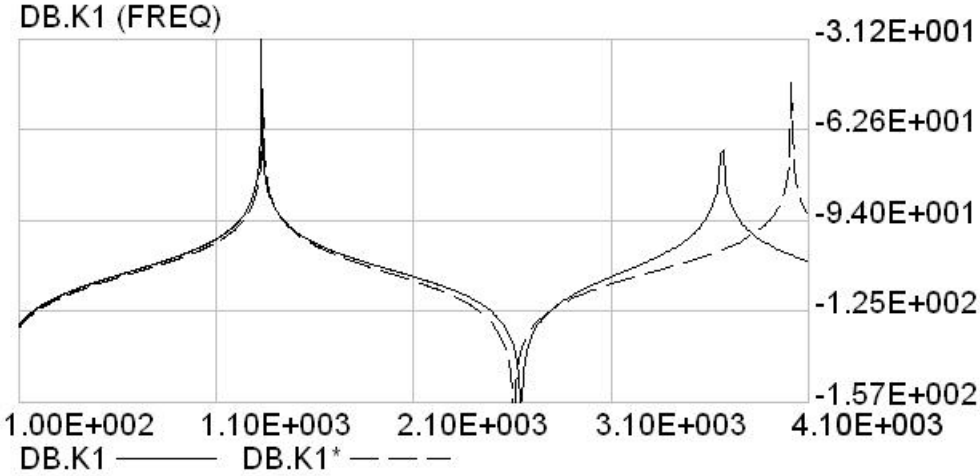


Рис. 9. Вигляд АЧХ Схеми 1 до і після (*) оптимізації

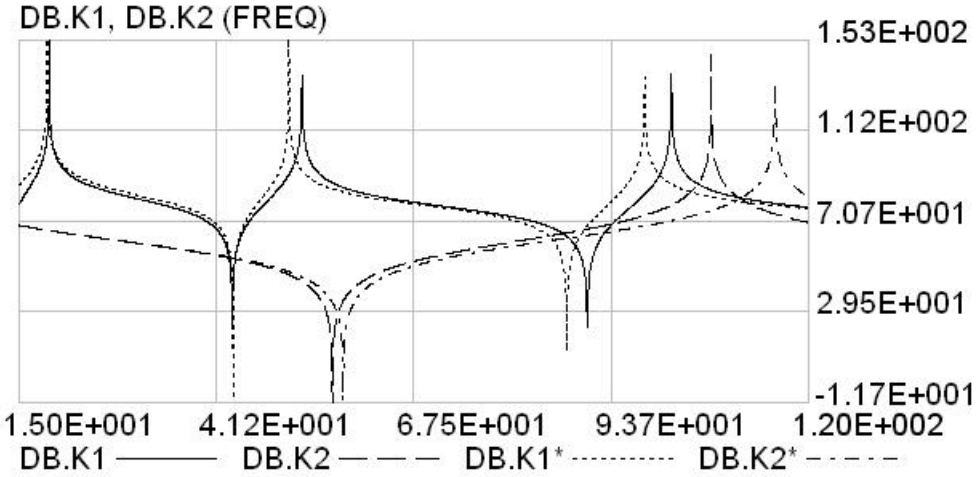


Рис. 10. Вигляд АЧХ Схеми 2 до і після (*) оптимізації

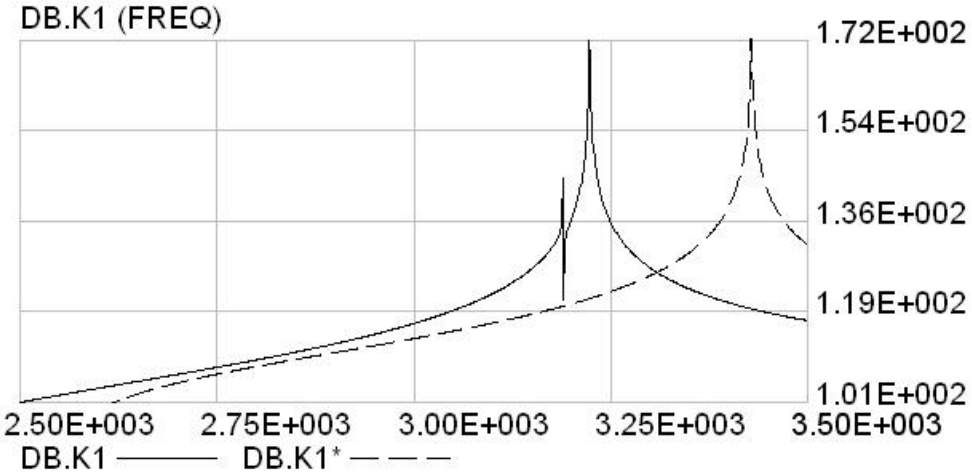


Рис. 11. Вигляд АЧХ Схеми 3 до і після (*) оптимізації

Список літератури

1. Ладогубец В.В. Методика построения моделей механических компонентов МЭМС для пакетов схемотехнического проектирования / Ладогубец В.В., Чкалов А.В., Безносик А.Ю., Финогенов А.Д. // Электроника и связь: тематический выпуск «Электроника и нанотехнологии». – 2009. – Ч. 1, № 2/3. – С. 298–305.
2. Petrenko A. Using Optimization Procedures to Calculate Parameters of MEMS Macromodels / Petrenko A., Ladogubets V., Beznosyk O., Finogenov O. // The Experience of Designing and Application of CAD Systems in Microelectronics : 10-th Anniversary Intern. Conf. «CADSM'2009», 24-28 February 2009, Polyana-Svalyava (Zakarpattya), Ukraine : proc. – Lviv, 2009. – P. 511–514.
3. Безносик А.Ю. Особенности оптимизации макромоделей механических компонентов с несколькими степенями свободы / Безносик А.Ю., Ладогубец Т.С., Ладогубец А.В. // 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. 356.
4. Petrenko A. ALLTED – a computer-aided engineering system for electronic circuit design / Petrenko A., Ladogubets V., Tchkalov V., Pudlowski Z. – Melbourne: UICEE, 1997. – 205 p.
5. Ладогубец В.В. Состояние и перспективы развития автоматизированного схемотехнического проектирования / Ладогубец В.В. // Электроника и связь: Тематический выпуск «Проблемы электроники». – 2005. – Ч. 1. – С. 121–127.

УДК 004.932.2

БУЗОВСКИЙ О.В.,
БОЛДАК А.А.,
НЕВДАЩЕНКО М.В.

СЕМАНТИКА МОДЕЛЕЙ И РАЗМЕРНОСТЬ ЗАДАЧ ПРОЕКТИРОВАНИЯ

В статье предложен ряд формализмов, связанных с определением моделей объекта и процесса проектирования. Анализируется иерархическая структура процесса проектирования с позиции оценки сложности модели и взаимосвязь между семантикой модели и размерностью проектных задач. В качестве иллюстрации такой взаимосвязи сформулированы задачи, решаемые при распознавании штриховых изображений. Приведена общая структура системы для распознавания штриховых изображений.

A number of formalisms related to the definition of object models and processes of self projection are suggested in the article. The hierarchical structure of the design process from the position of estimating the complexity of the model and interdependence between the semantics of the model and the dimension of the project objectives are analyzed. As an illustration of such interdependence the tasks solved by recognition of strip images are formulated. General structure of strip images recognition is given.

Введение

Вычислительные возможности современных компьютеров позволяют использовать компьютерную обработку изображений при решении различных прикладных задач, в том числе и задач автоматизации проектирования. Последние, так или иначе, связаны с разработкой проектной документации – различного рода спецификаций и схем, которые отображают семантику проектных решений. Подтверждением прогресса в этой области является наличие средств оптического распознавания [1-3], которые достаточно легко интегрируются в технологический процесс проектирования с элементами автоматизации.

При этом сам термин «распознавание» по отношению к изображению может обозначать как его кодирование, так анализ с целью классификации полученного кода. К сожалению, последняя задача сегодня еще не решена в общем виде, чем и объясняется отсутствие на рынке соответствующих программных средств. Одной из причин этого является отсутствие подходов к моделированию изображений и математическому аппарату их обработки с учетом семантики изображений.

В связи с этим целью работы является снижение трудоемкости разработки программных средств анализа изображения за счет формулировки и применения подходов к построению моделей изображений и способов их преобразования с учетом определенной семантики.

Таким образом, в рамках области анализа изображений исследуется модели и преобразования изображений и схем, отличительной особенностью которых является семантика, однозначно заданная с помощью формальных правил построения.

Формирование семантики проектного решения

Прежде чем говорить о моделировании изображения схем необходимо определить принципы и подходы к формированию семантики проектного решения, неразрывно связанной с понятиями объекта и процесса проектирования.

Для различного рода технических задач общий успех их решения в значительной мере определяет формулировка представлений об объекте, относительно которого такая задача решается. Обычно это представление называют моделью, причем сам термин «модель» требует уточнения. Ниже под моделью будем понимать абстракцию в соответствии с определением, введенным Хоаром [4]. Сложность формулировки модели, исключая тривиальные случаи, определяется сложностью самих объектов и проявляется в неформальном характере процедур или правил, которые можно применять для этой цели с учетом характера и особенностей конкретной задачи.

Сложность объекта и, соответственно, его модели, которая обычно определяет размерность задачи анализа, может быть рассмотрена с различных точек зрения. Прежде всего, можно выделить сложность структуры объекта (см. ниже), которая характеризуется количеством элементов, образующих объект, а также количеством и разнообразием типов взаимосвязей между этими элементами. Если количество элементов превышает некоторое пороговое значение, которое не является строго фиксированным, то такой объект может быть назван сложным.

Вторым аспектом сложности является сложность процесса функционирования объекта. Это может быть связано как с непредсказуемым характером его поведения, так и невозможностью формального представления правил преобразования входных воздействий в выходные.

Организация иерархии абстракции (моделей, понятий) позволяет исключить или существенно уменьшить сложность. При этом следует отметить, что существует два независимых подхода к построению таких иерархий. Первый из них акцентирует внимание на отношении обобщения и позволяет перейти от набора частных абстракций к небольшому числу более общих понятий. Второй подход использует отношение агрегации и позволяет представить систему с помощью более простых взаимодействующих компонент.

Цели и задачи проектирования определяются исходя из сформулированной, как правило, в виде технического задания семантики объекта проектирования (ОП) O для формального описания которой можно использовать тройку вида:

$$O = \{F, S, P\},$$

где F, S и P – соответственно функциональное, структурное и параметрическое описания объекта или (F-описание, S-описание и P-описание).

Функциональное описание на этом уровне представления является обязательным и должно отражать заданную траекторию O в пространстве времени-состояний, которая определяется реакцией объекта на управляющие и, возможно, пассивные воздействия внешней среды. При этом на форму F-описания в общем случае не накладывается никаких ограничений; это может быть просто вербальное определение назначения, например, «система управления воздушным движением» или «система управления сотовой телефонной связью». Если объект с заданным F-описанием или документация, по которой он может быть изготовлен, существуют, то задачу проектирования решать не имеет смысла при условии, что параметрическое описание удовлетворяет техническим требованиям. В противном случае такой объект необходимо проектировать.

В процессе проектирования используется известный с древних времен принцип «разделяй и властвуй», связанный с декомпозицией F-описания на некоторые составляющие F_i (второй подход), причем

$$F = S(F_i),\ i = \overline{1..n},$$

где S-оператор, определяющий такую композицию F_i, которой соответствует исходное F-описание. S-оператор задает S-описание или структуру ОП на рассматриваемом уровне декомпозиции. Некоторой части F_i могут соответствовать существующие объекты, относительно которых не имеет смысла решать задачу декомпозиции и в этом смысле их можно считать элементами несмотря на то, что они могут быть достаточно сложными. Следовательно для таких объектов S-описание отсутствует , что является формальным признаком самого понятия «элемент».

Для оставшейся части F_i вновь необходима декомпозиция и т.д. до тех пор, пока все F-описания не станут элементами. Таким образом,

$$F_i = S_i(F_{i+1}, j),\ i = \overline{1..n},\ j = \overline{1..m},$$

где j определяет номер шага, а i – индекс компоненты на соответствующем шаге декомпозиции. По сути каждый новый шаг соответствует очередному уровню абстракции.

Процесс декомпозиции F-описаний, как правило, неоднозначен, причем множество допустимых вариантов достаточно велико. Выбор варианта декомпозиции обычно определяется его качеством, которое количественно может быть оценено с помощью P-описанием, элементами которого являются количественные оценки свойств ОП называемые параметрами p_i, $i = \overline{1..k}$. Тогда P-описание есть $P = \{p_i\}$, причем P-описание всегда однозначно связано с конкретным S-описанием. В этом случае задача проектирования может быть поставлена как задача оптимизации P-описания на множестве вариантов декомпозиции.

По мере накопления удачного опыта проектирования однотипных объектов у разработчиков обычно складываются шаблонные решения задачи декомпозиции. Эти шаблоны являются обобщенными и связаны с построением иерархии обобщений. Последняя по внешнему виду напоминает схему классификации понятий формальной логики. В этой связи следует заметить, что каждое понятие в логике имеет некоторый объем и содержание. При этом под объемом понятия понимают все другие понятия, для которых исходное может служить определяющей категорией или главной частью. Содержание понятия составляет совокупность всех его признаков или атрибутов, отличающих данное понятие от всех других.

Иерархия понятий имеет следующую структуру. В качестве наиболее общего понятия или категории используется понятие, имеющее наибольший объем и, соответственно, наименьшее содержание. Это самый высокий уровень абстракции для данной иерархии. Менее общие понятия, которые конкретизируют некоторую абстракцию на схеме иерархии будут располагаться на уровень ниже исходного понятия. По мере продвижения по дереву абстракций на более низкие уровни данное общее понятие некоторым образом конкретизируется, тем самым уменьшается его объем и увеличивается содержание.

В общем случае, процесс формирования иерархии обобщений представляет собой разработку методики проектирования, которая сформулирована в терминах, соответствующих категориям этой иерархии и правилам декомпозиции.

Если описанная выше иерархия агрегации строится, как правило, «сверху вниз», то иерархия обобщений – наоборот, «снизу вверх», и в последствии может эффективно использоваться для решения задачи декомпозиции, поскольку ограничивает множество допустимых проектных решений.

Фактически, иерархия обобщений может быть описана четверкой (Vp, Vc, Pr, Pl), где $Vp = \{vp_i\}$, $i = \overline{1..n}$ – множество примитивов, соответствующих нижнему уровню иерархии, $Vc = \{vc_i\}$, $i = \overline{1..m}$ – множество конструкций – понятий более высоких уровней иерархии, $Pr = \{pr_i\}$, $i = \overline{1..k}$ – множество допустимых правил (операций), с помощью которых могут быть заданы отображения $Vp \cdot Vp \to Vc$, $Vp \cdot Vc \to Vc$, $Vc \cdot Vc \to Vc$, $Pl = \{pl_i\}$, $i = \overline{1..l}$ – множество локальных параметров, используемых для оптимизации проектных решений.

Не трудно заметить, что определенная таким образом иерархия представляет собой замкнутую математическую систему (алгебру) относительно множества объектов $Vp \cup Vc$ и набора операций Pr.

Отсутствие в этой системе правил вида $Vp \cdot Vp \to Vp$ определяет ограничения, связанные с применением данной иерархии обобщений для решения задачи декомпозиции. Эти ограничения состоят в том, что в рамках этой замкнутой системы объекты Vp являются элементарными, но в природе они не существуют, и, следовательно, для них требуется решать задачу проектирования.

В этом случае необходима смена иерархии понятий, которая используется для проектирования, и определение однозначных соответствий между понятиями новой и элементами предыдущей иерархий. Таким образом, определен уровень отрыва методик проектирования, который связан со сменой набора понятий и операций, используемых для решения задачи декомпозиции.

Формальные правила построения схем как формы отображения описанной выше семантики охватывают одну иерархию обобщений и предназначены для построения отображения иерархии агрегации. При этом понятиям иерархии обобщения соответствуют условные графические обозначения (УГО), а операциям (отношениям агрегации) – связи соединяющие УГО. Таким образом, схема может рассматриваться как графовая модель $G = \{N, R \subseteq N \times N\}$.

В общем случае интерпретация (чтение) схемы связано с определением спецификации элементов множества N и агрегации заданным с помощью R.

Штриховые изображения

Схемная документация как форма отображения проектного решения обладает рядом свойств, позволяющих на ее основе формально восстанавливать структуру объекта проектирования (ОП). К таким свойствам относятся [5]:

- использование при начертании простейших геометрических примитивов;

- использование ограниченного множества условных графических обозначений (УГО), которые ставятся в однозначное соответствие элементам структуры ОП;
- наличие правил оформления схем, однозначно определяющих соответствие между отношениями на множестве элементов ОП и геометрическими соотношениями между примитивами и УГО на поле чертежа.

Сказанное выше определяет предпосылки создания системы обработки схемной документации, основная задача которой состоит в распознавании схем.

Решение задач обработки изображений связано с использованием моделей, сформулированных в рамках двух подходов. Суть первого подхода состоит в том, что изображение рассматривается как многомерный информационный сигнал. Чаще всего изображение рассматривают как двумерную функцию распределения некоторой физической величины (в большинстве задач – интенсивности освещения) [6]. Второй подход определяет модель изображения как множество геометрических примитивов, наглядных знаков и их геометрических соотношений, в соответствие которым поставлена определенная семантика [7]. Очевидно, что решение задачи распознавания схем предполагает использование обеих моделей. Более того, необходимо определить соответствие между ними.

Примером семантической модели изображения может служить схема, содержащая геометрические примитивы, образующие УГО, в соответствие которым правила оформления ставят семантику структуры ОП. При этом примечательно то, что множество используемых в схемах примитивов обладают общим свойством – они представляют собой траектории движения пера (обычно круглой формы с постоянным диаметром). В этом случае отличительной особенностью примитивов является форма траектории пера, а не его толщина или цвет. Геометрические примитивы, обладающие перечисленными свойствами можно назвать штрихами.

Таким образом, под штриховым изображением в рамках семантического подхода в дальнейшем понимается изображение, сформированное с помощью штрихов.

В отличие от изображения рукописного текста, схемная документация характеризуется использованием ограниченного множества штрихов (прямая, дуга, окружность), которые могут быть получены с помощью простейших чертежных инструментов. Это ограничение позволяет в значительной степени упростить задачу распознавания штрихов на схемах. Но при этом следует отметить, что УГО представляют собой структуры штрихов, а схема, в свою очередь, воспринимается как структура УГО. Следовательно, семантическая модель схемы является иерархической и, соответственно, требует применения многоуровневых схем распознавания.

Ограничения, наложенные на класс штриховых изображений, позволяют сформулировать множество классификационных признаков в рамках сигнальной модели.

Поскольку устойчивая идентификация штрихов возможна только в случае использования равномерного фона изображения, то гистограмма интенсивности идеального штрихового изображения представляет собой функцию вида:

$$H(i) = \alpha_1 \cdot \delta(I_1, i) + \alpha_2 \cdot \delta(I_2, i), \qquad (1)$$

где I_1 и I_2 – значения интенсивности для фона и штрихов соответственно, $\delta(x, y)$ – функция Кронекера, α_1 и α_2 – весовые коэффициенты $(\alpha_1 + \alpha_2 = 1)$.

На Рис.1 приведена гистограмма интенсивности идеального штрихового изображения, которая существенно отличается, например, от типовой гистограммы фотографии (Рис.2).

Анализ гистограммы реального штрихового изображения (Рис.3), позволяет определить множество задач предварительной обработки, необходимой для приведения реальных изображений к виду, пригодному для дальнейшего распознавания.

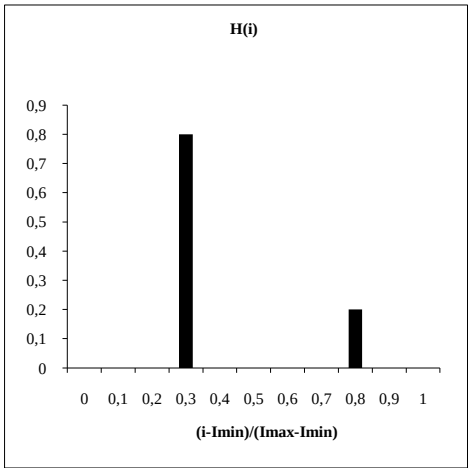


Рис.1. Гистограмма интенсивности идеального штрихового изображения

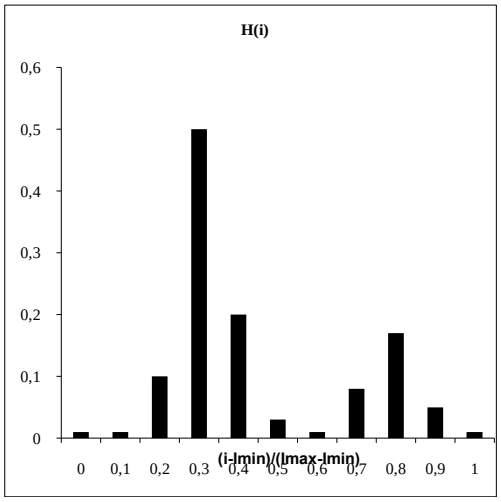


Рис.2. Гистограмма интенсивности фотографического изображения

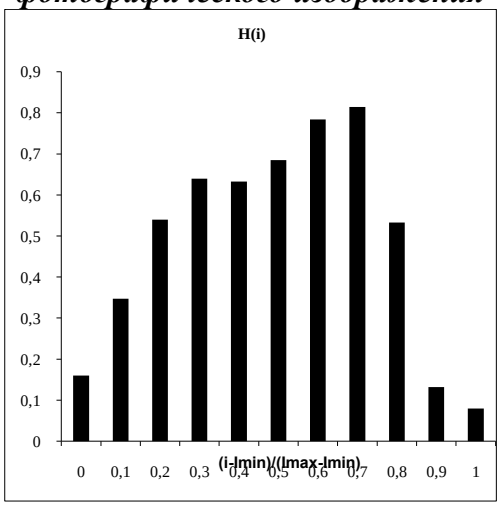


Рис.3. Гистограмма интенсивности реального штрихового изображения

Штриховые изображения характеризуются также специфическим распределением $I(x,y)$, где I – интенсивность, x и y – геометрические координаты, которое представляет собой кусочно-постоянную функцию с уровнями интенсивности I_1 и I_2. При этом штрихам соответствуют связные геометрические области с постоянной интенсивностью I_2. Более того, можно указать минимально допустимую площадь такой области – она равна площади пера. Это особенность дает возможность использовать эффективные методы кодирования, основанные на прослеживании границ областей одинаковой интенсивности (контуров) [5]. Вид функции распределения $I(x,y)$ для штриховых изображений обеспечивает также высокую помехоустойчивость, поскольку предопределяет эффективную реставрацию реальных изображений с помощью методов цифровой фильтрации.

Таким образом, выделенный класс штриховых изображений обладает рядом специфических свойств как на уровне сигнальной, так и на уровне семантической модели. Следовательно, для этого класса может быть очерчено множество задач обработки таких изображений и определены эффективные методы их решения.

Обработка штриховых изображений рассматривается в контексте решения задачи распознавания схемной документации. При этом обработка изображений может быть разделена на этапы, связанные с предварительной обработкой, препарацией и распознаванием.

Под предварительной обработкой понимается преобразование реальных изображений к виду, пригодному для решения последующих задач.

Применительно к штриховым изображениям предварительная обработка состоит в преобразованиях, позволяющих привести гистограмму интенсивности к виду (1), реставрировать форму распределения $I(x,y)$, осуществить геометрическую привязку отдельных фрагментов изображения, произвести контурное кодирование штрихового изображения.

Решение первой задачи сводится к повышению контрастности изображения, которое может эффективно осуществляться с помощью известного метода эквализации [6], с последующим применением порогового фильтра, формирующего бинарное изображение.

Вторая задача определяет необходимость подавления пиксельного шума в бинарном изображении и решается с помощью методов ранговой фильтрации. При этом размер маски фильтра выбирают в зависимости от диаметра пера.

Необходимость геометрической привязки фрагментов обусловлена тем фактом, что в большинстве случаев форматы сканирующих устройств существенно меньше размеров реальных чертежей. В связи с этим производят сканирование отдельных, частично перекрывающихся фрагментов и их наложение с целью получения единого изображения схемного документа. Известные методы корреляционной привязки, к сожалению, обладают высокой вычислительной сложностью, что не позволяет использовать их в полной мере для автоматического решения данной задачи. (Функция оператора в данном случае состоит в указании окна перекрытия фрагментов, в пределах которого осуществляется привязка.)

Методы контурного кодирования позволяют снизить информационную избыточность

изображения, что в большой степени определяет эффективность методов решения задач препарирования.

Под препариванием в контексте рассматриваемой задачи понимается получение множества признаков изображения, необходимых для распознавания. Фактически, препарирование – это этап обработки, позволяющий связать сигнальную и семантическую модели. Применительно к обработке штриховых изображений препарация предполагает получение описания штрихов, достаточного для их однозначной идентификации. Исходя из определения штриха, можно заключить, что такое описание должно быть инвариантным по отношению к толщине штриха и, по возможности, полно описывать форму траектории пера. Это означает, что основным признаком штриха можно считать серединную ось. (Эффективным методам скелетизации посвящена работа [5].) При этом следует отметить, что при скелетизации следует отдавать предпочтение методам, позволяющим получить описание формы штриха в аналитическом виде, поскольку в этом случае имеется возможность использования эффективных аналитических и численных методов анализа форм кривых.

Следует отметить, что в том случае, когда разрешающая способность сканирующих устройств недостаточна, наблюдается значительное влияние помех дискретизации на форму штриха. В этом случае необходима предварительная реставрация формы контуров. Задача реставрации формы контура может рассматриваться как задача фильтрации одномерного сигнала с равномерно распределенным шумом. В дальнейшем с помощью методов численного дифференцирования определяются узловые точки аппроксимации контура. Описанный метод позволяет классифицировать сегменты контура по степени аппроксимирующего полинома.

Как уже отмечалось, семантическая модель схемы является иерархической. В связи с этим схема решения задачи распознавания приобретает следующий вид.

На первом этапе на основе анализа формы производится классификация штрихов, и определяются их геометрические соотношения. Второй этап связан с распознаванием таких геометрических объектов как многоугольники. Задача выделения УГО сводится к определению пространственных соотношений между первичными штрихами и объектами, полученными на втором этапе. Результатом решения задачи является набор распознанных УГО (входящих во множество предопределенных) и набор неклассифицированных примитивов. При этом должна предоставляться возможность ведения статистики по результатам распознавания, ошибкам, неклассифицированным объектам и т.п. На последнем этапе производиться классификация УГО (согласно имеющейся базе данных схемных элементов), обучение системы в случае введения/выявления новых классов элементов или изменения существующих, а также приведение распознанного документа к требуемому формату.

Каркас программной системы обработки зображений

Множество математических моделей и их преобразований, применяемых при решении задач обработки изображений схем, может быть представлено иерархической структурой, в которой на нижнем уровне иерархии расположены сигнальные модели, верхние уровни соответствуют семантическим моделям изображений. Решение конкретной задачи соответствует пути в такой иерархии, который начинается на нижних, и заканчивается на верхних ярусах.

На основе такого подхода можно предложить математическую схему для решения задач обработки изображений (Рис.4), которая включает в себя отдельные стадии, каждая из которых может рассматриваться как отдельная задача. Более того, такие задачи могут быть классифицированы по признаку моделей, используемых ими в качестве исходных и результирующих. Следовательно, можно определить множество программных классов и их интерфейсов, составляющих устойчивый каркас системы обработки изображения (Рис.5).

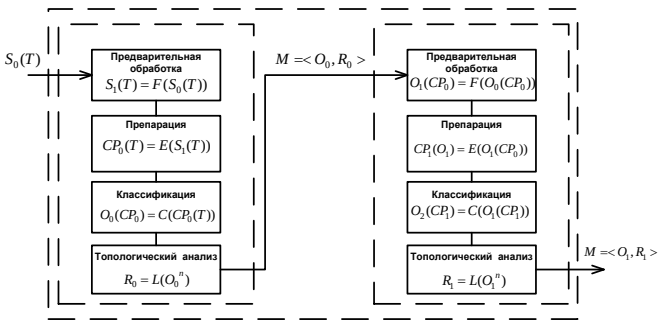


Рис.4. Математическая схема решения задач обработки изображений

$S(T)$ – функция пространственного распределения вектора характеристик сигнала;

T – пространственный базис;

I – вектор характеристик сигнала;

$SP_i, i = 1,...,n$ – скалярная характеристика сигнала;

O – множество экземпляров;

$CP_j, j = 1,...,m$ – классификационный признак;

R – структура изображения;

$v_1 : C(S) \to O, v_2 : L(O^l) \to R$ – классификация экземпляров и определение структуры изображения.

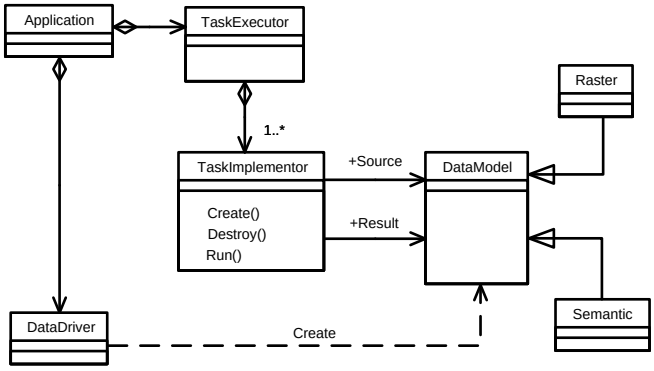


Рис.5. Схема классов для каркаса системы обработки изображений

Класс Application, который реализует интерфейс пользователя программы, использует экземпляр класса TaskExecutor, представляющий собой механизм последовательного запуска стадий обработки. Стадии обработки на диаграмме представлены абстрактным классом TaskImplementor, который имеет методы, необходимые для создания, запуска и уничтожения экземпляра, а также использует в качестве исходных данных и результатов экземпляры абстрактной модели данных (DataModel). Сигнальная и семантическая модели являются разновидностями такого класса. Создание экземпляров данных поручается драйверу (DataDriver), который ориентирован на конкретный формат внешних файлов и находятся под управлением самого приложения. Эта схема дает возможность, используя различные форматы внешних визуальных данных, преобразовать их во внутренний формат реализовать последовательность стадий обработки, которые зарегистрированы в списке механизма последовательного запуска.

Описанный каркас, основанный на использовании абстрактных классов, может повторно использоваться при разработке различных программных систем обработки изображений, освобождая тем самым ресурсы команды разработчиков, необходимые для выполнения научно-исследовательских работ.

Известно, что разработчики программного обеспечения стремятся к узкой специализации, что позволяет им сократить затраты на исследовательские работы за счет использования опыта, полученного в предыдущих проектах. Такой опыт, как правило, оформлен в виде типовых решений (шаблонов), программных библиотек и т.п. Рассмотренный каркас создает предпосылки для создания подобных библиотек, начиная со средств доступа к данным, и заканчивая реализациями стадий обработки изображений.

Библиотека программных ресурсов имеет четыре уровня инструментария для реализации средств обработки изображений (Рис.6). Базовый уровень связан с реализацией сигнальной и семантической моделей изображений, которые могут использоваться как при обработке на различных стадиях, так и при реализации драйверов данных, опирающихся на это базовое представление моделей изображений. Уровень математического аппарата содержит модули средств, реализующих математические методы преобразования изображений, которые используются для решения базовых задач обработки визуальной информации. Реализация этих задач сконцентрирована на прикладном уровне. Верхний уровень содержит средства для организации интерфейса пользователя и контроллеров задач обработки, что позволяет уменьшить затраты на разработку новых или модификацию функциональности существующих программ.

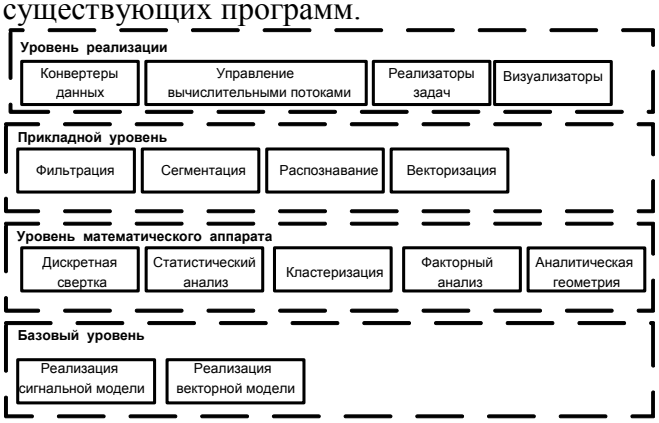


Рис. 4. Библиотека программных ресурсов для построения систем обработки изображений

Заключение

В работе сформулирован подход к построению моделей и преобразований изображений, который отличается от известных тем, что учитывает семантику последних. Это позволяет получать формализованные математи-

ческие схемы решения задач анализа изображений с учетом их семантики. Формализация математической схемы решения задачи анализа изображений создает предпосылки к созданию унифицированного программного каркаса, использование которого существенно снижает затраты на разработку специализированных программных средств связанных с анализом изображений.

В качестве примера применения сформулированного подхода приведено решение задачи анализа изображений схем в технологических процессах автоматизированного проектирования.

Список литературы

1. ABBYY FineReader [Электронный ресурс] / Компания ABBYY – ведущий разработчик ПО в области распознавания, обработки данных и лингвистики – Электрон. дан.. – ABBYY, 2004. Режим доступа: http://www.abbyy.ru/finereader/, свободный. – Загл. с экрана. – Яз. рус.
2. Readiris [Электронный ресурс] / I.R.I.S. – OCR software and Document Management solutions – Электрон. дан.. – I.R.I.S. s.a., 2009. Режим доступа: http://www.irislink.com/, свободный. – Загл. с экрана. – Яз. англ.
3. SILVERCODERS OCR Server [Электронный ресурс] / SILVERCODERS – Quality and Competence – Электрон. дан.. – SILVERCODERS, 2005. Режим доступа: http://silvercoders.com/, свободный. – Загл. с экрана. – Яз. англ.
4. Дал У., Дейкстра Э., Хоор К. Структурное программирование. – М.: Мир, 1975. – 245с.
5. Александров В.В., Горский Н.Д. Алгоритмы и программы структурного метода обработки данных. – Л.: Наука, 1983, – 208с.
6. Александров В.В., Горский Н.Д. Представление и обработка изображений: рекурсивный подход. – Л.: Наука, 1985, – 190с.
7. Анисимов Б.В., Курганов В.Д., Злобин В.К. Распознавание и цифровая обработка изображений. – М.: Высшая школа, 1983, – 296 с.

УДК 004.67:004.428.4

СТІРЕНКО С.Г.,
ЗІНЕНКО О.І.

ЗБЕРІГАННЯ РОЗРІДЖЕНИХ МАТРИЦЬ ВЕЛИКИХ РОЗМІРНОСТЕЙ В СИСТЕМІ З ЛОКАЛЬНОЮ ПАМ'ЯТТЮ

В цій статті запропоновано ієрархічну схему зберігання розріджених матриць великих розмірностей в системі з локальною пам'яттю, який може бути використаний при розробці застосунків з лінійної алгебри. Проведено аналіз ефективності застосування цієї схеми та порівняння з іншими схемами. Запропоновано метод обробки таких матриць в розподіленій системі.

In this article the scheme of large scale sparse matrix storage in the distributed memory system, which can be useful in linear algebra applications development, is proposed. The comparative efficiency analysis of proposed and existing schemes is made. Also the method of proceeding such matrices in a distributed memory computer system is proposed.

Вступ

Велика кількість наукових задач та прикладних задач при реалізації їх розв'язку на ЕОМ може бути зведена до роботи з матрицями та векторами, для яких існує багато розроблених алгоритмів та структур даних. Зокрема для розв'язку диференційних рівнянь методом скінченних елементів або скінченних різниць, розв'язку систем лінійних алгебраїчних рівнянь, задачі знаходження власних чисел та векторів лінійних операторів тощо, існує декілька математичних методів розв'язку, що базуються на використанні матриць та інших алгебраїчних структур [1].

З розвитком засобів обчислювальної техніки став можливим розв'язок задач, що містять великі обсяги вхідних даних [2], таких як системи рівнянь з декількома мільйонами невідомих. Серед них окремо виділяються задачі, вхідні дані яких є розрідженими матрицями, які містять відносно невелику кількість ненульових елементів. Для таких задач розроблені сучасні швидкі алгоритми, що базуються на теорії підпросторів Крилова, зокрема алгоритми Ланцоша та Відемана для розв'язку задачі власних чисел та розв'язку системи лінійних рівнянь відповідно [3]. Ці методи були успішно застосовані, наприклад, до атаки на криптографічний алгоритм RSA.

Стримуючим фактором ефективного застосування програмних реалізацій алгоритмів розв'язку задач такого класу на ЕОМ є необхідність зберігання великої кількості даних під час обробки. На даному етапі розвитку техніки вхідні матриці не можуть бути розміщені в оперативній пам'яті однієї ЕОМ [4], хоча повністю вони необхідні лише на початковому етапі алгоритму. Тому необхідно запропонувати таку схему зберігання розріджених матриць великої розмірності, яка б дозволила завантажувати всі вхідні дані (або їх частину) в локальну оперативну пам'ять кожного вузла кластера та виконувати їх ефективну обробку для подальшого використання в програмних реалізаціях математичних алгоритмів Криловського типу. Також необхідно, щоб вхідні дані могли б бути приведені до запропонованої схеми з найбільш поширеного формату зберігання розріджених матриць на зовнішніх носіях Matrix Market [5].

Схеми зберігання розріджених матриць

Всі схеми зберігання розріджених матриць базуються на зменшенні розміру даних, що зберігаються, завдяки зберіганню лише ненульових елементів. Оскільки в розріджених матрицях за визначенням таких елементів значно менше ніж ненульових, використання таких схем дозволяє зберігати матриці більших розмірностей. Позначимо ω кількість ненульових елементів матриці, та $k_\omega = \dfrac{\omega}{n \times m}$ коефіцієнт розрідженості матриці. Важливою властивістю схеми зберігання є необхідність наявності способу визначити елемент матриці за його рядковим та стовпцевим індексами.

Найпростіший підхід запропоновано в схемі Matrix Market [5]. Вона передбачає збері-

гання впорядкованого за індексом рядка та стовпця в вихідній матриці лінійного масива трійок "елемент, індекс рядка, індекс стовпця".

Більш складний, проте більш ефективний метод запропоновано в схемах CSR (compressed sparse row) та CSC (compressed sparse column) [6], які є симетричними та відрізняються лише використанням рядків або стовпців в якості основного індексу. В схемі CSR зберігається лінійний масив пар "елемент, індекс стовпця" та лінійний масив індексів в головний масив, що показує початок кожного рядка.

Схема Кнута-Ларкомба [6] є прикладом зв'язної схеми зберігання. Для симетричних матриць в ній застосовується структура з елемента та двох покажчиків на правий та верхній ненульові елементи матриці. Для матриць загального вигляду легко розробити модифікацію, що містить інші покажчики. В залежності від потреб алгоритму, що реалізується, існує можливість використати декілька покажчиків за необхідними напрямами обходу. Ця схема потребує достатньо великого обсягу пам'яті для службових даних та незручна в зберіганні на зовнішніх носіях, але є найшвидшою при виконанні ітерацій за ненульовими елементами.

Гіперматрична схема

Дамо визначення гіперматриці. Гіперматрицею k-го порядку називається така матриця, елементами якої є гіперматриці $(k-1)$-го порядку. Гіперматрицею першого порядку є така матриця, елементами якої є скаляри. Гіперматриця k-го порядку може бути представлена в вигляді гіперматриці $(k+1)$-го порядку шляхом групування її сусідніх елементів. Оскільки матриця з одним елементом з точки зору алгебри залишається матрицею, процес групування та підвищення порядку може продовжуватися нескінченно. Наведемо приклад гіперматриці

$$\begin{pmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \\ 9 & 10 & 11 & 12 \\ 13 & 14 & 15 & 16 \end{pmatrix} \leftrightarrow \begin{pmatrix} \begin{pmatrix} \begin{pmatrix} 1 \\ 5 \end{pmatrix} & \begin{pmatrix} 2 \\ 6 \end{pmatrix} \end{pmatrix} & \begin{pmatrix} \begin{pmatrix} 3 \\ 7 \end{pmatrix} & \begin{pmatrix} 4 \\ 8 \end{pmatrix} \end{pmatrix} \\ \begin{pmatrix} \begin{pmatrix} 9 & 10 \end{pmatrix} \\ \begin{pmatrix} 13 & 14 \end{pmatrix} \end{pmatrix} & \begin{pmatrix} \begin{pmatrix} 11 & 12 \end{pmatrix} \\ \begin{pmatrix} 15 & 16 \end{pmatrix} \end{pmatrix} \end{pmatrix}.$$

Розглянемо обернений процес: мається гіперматриця k-го порядку, з якої необхідно отримати гіперматрицю першого порядку. Для цього виконати наступні ітерації. Одночасно замінити всі гіперматриці першого порядку на їх елементи, сформувавши таким чином нові гіперматриці першого порядку замість таких другого порядку, але більшої розмірності. При цьому порядок загальної гіперматриці зменшиться на одиницю. Продовжувати ітерації доки порядок загальної гіперматриці не дорівнюватиме одиниці.

Однак при такому підході існує наступна проблема: гіперматриця за визначенням є більш складною структурою ніж просте групування елементів матриці, тому може містити матриці-елементи різних розмірностей. В цьому випадку, їх знищення і використання лише їх елементів напряму неможливе. Таким чином необхідно перевіряти умови знищення: сумарна кількість елементів в кожному рядку та кожному стовпці після знищення матриць-елементів має бути відповідно однаковою. Для спрощення, розглядатимемо тільки такі гіперматриці, які задовольняють більш строгим умовам: всі матриці-елементи в рядку гіперматриці мають містити однакову кількість елементів в стовпці; всі матриці-елементи в стовпці гіперматриці мають містити однакову кількість елементів в рядку.

Застосуємо цей підхід до зберігання даних в ЕОМ з локальною пам'яттю. Нехай задана матриця великої розмірності. Виділимо підматриці однакової розмірності зі скалярів та зберігатимемо їх за певною схемою. Додатково створимо службову структуру даних, в якій зберігатиметься розмірність таких підматриць та їх взаємне розташування. Таким чином отримали гіперматричну схему зберігання другого порядку. Отримані підматриці (або гіперматриці першого порядку) можуть зберігатися в локальній пам'яті окремих вузлів незалежно; необхідно лише кожному вузлу надати інформацію про розташування інших частин матриці, якщо це необхідно за алгоритмом.

Схема зберігання підматриць або гіперматриці в цьому випадку може бути будь-якою. Але при застосуванні, наприклад, традиційної щільної схеми, використання запропонованого методу недоцільне, оскільки потребує лише додатової пам'яті для службових даних, і має перевагу лише при використанні блочних ал-

горитмів, для яких такий формат представлення є зручним. У випадку розріджених матриць з великою кількістю нульових елементів підматриця може виявитися складеною повністю з нульових елементів. Зберігати таку підматрицю недоцільно, тому в структурі гіперматриці можна зберігати лише сигнальне значення. Таким чином, гіперматриця другого порядку також може виявитися розрідженою, і до неї може бути застосована одна з схем зберігання розріджених матриць. Таким чином процес підвищення порядку розріджених гіперматриць можна продовжувати, доки наступна може бути ефективно збережена за розрідженою схемою або не буде досягнуто певного порядку. Проте недоліком збільшення порядку є збільшення часу доступу до окремого елементу за відомими індексами. Аналіз ефективності та швидкості роботи запропонованої схеми наведено далі.

Порівняльний аналіз схем зберігання

Наведемо теоретичні розрахунки ефективності згаданих схем та перевіримо їх теоретично. Для спрощення аналізу, вважатимемо що розглядається збереження матриць чисел з плаваючою комою розмірністю 64 двійкових розряди на 64-розрядній системі, оскільки такі системи є найбільш розповсюдженими існуючими кластерними системами. Тобто розмір як елемента даних, так і покажчика або індекса становитиме 8 байт.

Схема Matrix Market потребує зберігати для кожного елемента безпосередньо значення та два індекси: рядковий та стовпцевий. Тому для зберігання за цією схемою необхідно

$$M_{MM} = 8 \cdot 3 \cdot \omega \text{ байт.}$$

Однак класична реалізація цієї схеми як впорядкованого за індексами лінійного масиву потребуватиме $O(\log_2 k_\omega n \cdot \log_2 k_\omega m)$ операцій для пошуку елемента за індексами, $O(1)$ – для вибору наступного елемента в рядку та $O(k_\omega \cdot m)$ – для вибору наступного елементу стовпця.

Схема CSR потребує окрім значення ненульового елементу зберігання також його стовпцевого індексу та окремого масиву індексів початку рядків, тобто необхідний обсяг пам'яті

$$M_{CSR} = 8 \cdot (2 \cdot \omega + n) \text{ байт.}$$

При чому пошук елемента за індексами може бути виконаний за $O(\log_2 k_\omega m)$ операцій, перехід до наступного елементу в рядку за $O(1)$ операцій, а до наступного елементу в стовпці за $O(\log_2 k_\omega m)$ операцій.

Схема Кнута-Ларкомба з застосуванням лише двох покажчиків: на елементи знизу та справа – тобто формуванням однозв'язних списків, та застосуванням додаткових масивів покажчиків на початки рядків та стовпців потребуватиме

$$M_{KL} = 8 \cdot (3 \cdot \omega + n + m) \text{ байт.}$$

Для цієї схеми пошук елементів за індексами матиме складність $O(\min(k_\omega m, k_\omega n))$ операцій, але перехід до наступного елементу як в рядку, так і в стовпці – лише $O(1)$ операцій.

Безперечною перевагою гіперматричних схем є можливість застосування різних схем, як розріджених, так і щільних, при зберіганні гіперматриць різних порядків. Тому теоретично можливо розрахувати необхідні обсяги лише якщо всі такі схеми відомі, або в термінах обсягів пам'яті, необхідних для зберігання за зазначеними схемами. Для зберігання матриць найбільших розмірностей доцільно використовувати таку схему, яка потребує найменшого обсягу пам'яті. З розглянутих схем в загальному випадку це CSR схема. Знайдемо необхідний обсяг пам'яті для збереження гіперматриці k-го порядку за такою схемою

$$M_H = 8 \cdot \left(\sum_{i=1}^{p_1} (2 \cdot \omega_{1,i} + \frac{n}{p_1^{0.5}}) + \sum_{k=2}^{g} \sum_{i=1}^{p_k} (2 \cdot \omega_{k,i} + \sqrt{p_k}) \right),$$

Де $\omega_{i,k}$ – кількість ненульових елементів в i-тій підматриці гіперматриці k-го порядку, p_k – кількість елементів гіперматриці k-го порядку, g – найбільший порядок гіперматриці.

Очікувані з теоретичних розрахунків обсяги використаної пам'яті в залежності від розрідженості наведені на рис. 1. Розглядалися квадратні розріджені матриці, в яких ненульові елементи розподілені рівномірно. Вважаємо, що розмірність матриць постійна, а кількість ненульових елементів змінюється, що призводить до зміни коефіцієнту розрідженості. При розрахунках для гіперматричної схеми, вважаємо, що розмір гіперматриці першого порядку визначається як $\sqrt[g]{n}$, де g – максимальний порядок.

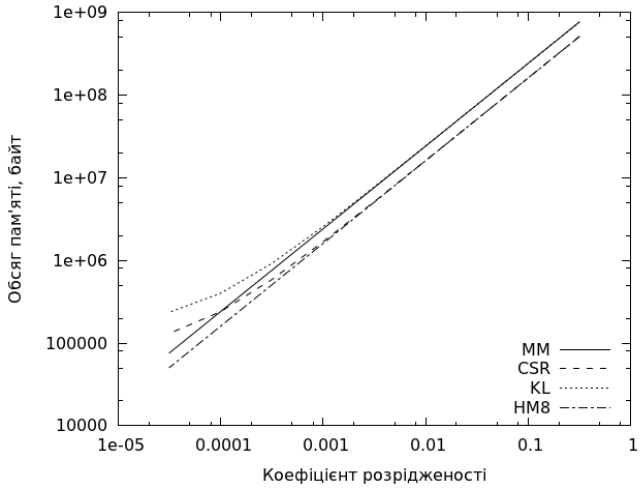


Рис. 1. Обсяг пам'яті, необхідна для зберігання матриці розміром $10^4 \times 10^4$ елементів.

Тут і далі в графіках використано наступні позначення: MM – схемаMatrix Market; CSR – розріджена рядкова схема; KL – схема Кнута-Ларкомба; HMi – гіперматрична схема і-го порядку.

Також було розраховано ефективність застосування запропонованого методу у порівнянні з переліченими вище класичними методами (рис. 2).

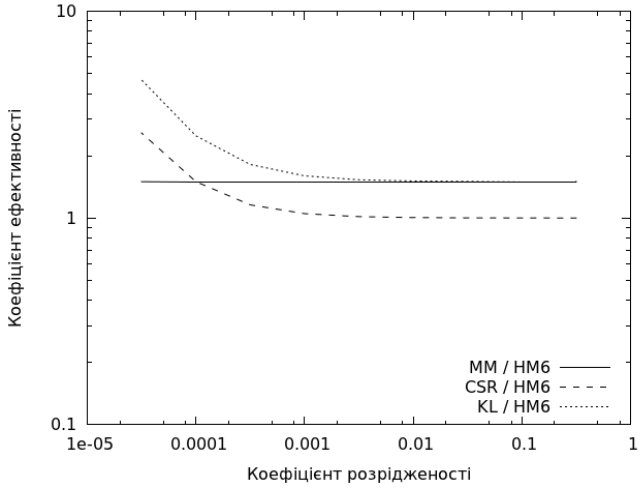


Рис. 2. Відношення пам'яті, необхідної для існуючих методів, до пам'яті, що потребується для запропонованого методу.

Для підтвердження теоретичних розрахунків та практичного порівняння схем було розроблено програми, що реалізують відповідні схеми зберігання розріджених матриць на мові C++ з використанням бібліотеки паралельних обчислень для систем з локальною пам'яттю MPI. Робота виконувалась на кластері центру суперкомп'ютерних обчислень НТУУ "КПІ". Для перевірки часу роботи алгоритмів зі схемами зберігання, що порівнюються, було також розроблено програмну реалізацію генерації підпросторів Крилова, що включає виконання матрично-матричних та матрично-векторних операцій. Було проведено серію запусків для різних згенерованих довільним чином розріджених матриць великих розмірностей з фіксованою кількістю ненульових елементів. Ненульові елементи розподілені рівномірно по всій матриці в першій серії експериментів, та нормально відносно головної діагоналі. Результати приведені на рис. 3 та рис. 4.

Експериментальні дані також показують переваги запропонованого методу зберігання над існуючими методами при його застосуванні до розріджених матриць, розмірності яких вимірюються мільйонами елементів, а коефіцієнт розрідженості відносно невеликий і становить відповідно $10^{-6} \div 10^{-4}$ для різних розмірностей.

Зі зменшенням коефіцієнту розрідженості доцільно збільшувати порядок використовуваних гіперматричних схем, оскільки збільшується ймовірність того, що гіперматриці також будуть розрідженими.

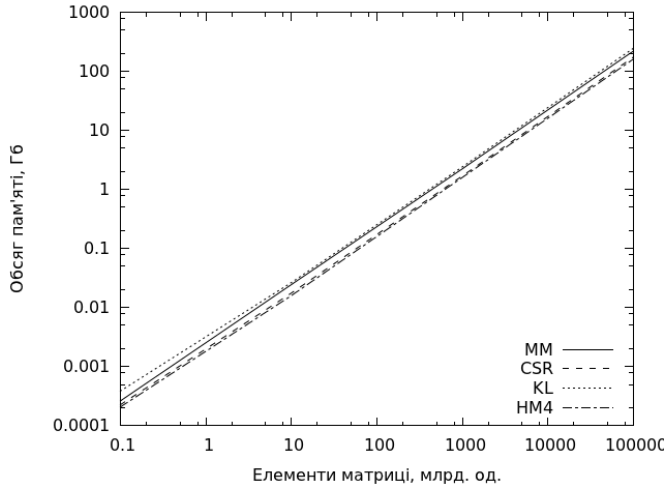


Рис. 3. Середній обсяг пам'яті, необхідний для зберігання розріджених матриць з коефіцієнтом розрідженості $k_\omega = 0.0001$.

Також підтверджено емпіричний висновок, що застосування схем зберігання розріджених матриць доцільне лише для таких матриць, які містять таку кількість ненульових елементів, яка більша ніж одна з розмірностей матриці не більше ніж на декілька порядків (див. рис. 2).

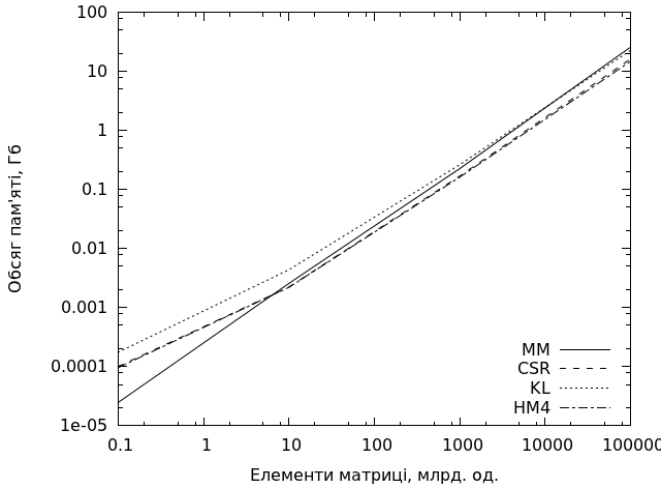


Рис. 4. Середній обсяг пам'яті, необхідний для зберігання розріджених матриць з коефіцієнтом розрідженості $k_\omega = 0.0001$.

Метод організації обчислень та деталі програмної реалізації

Однією з поставлених при розробці схеми задач була можливість ефективного використання запропонованої схеми при розробці програм для копм'ютерної системи з локальною пам'яттю. Оскільки вирішення сучасних задач потребує використання матриць, що містять декілька трильйонів елементів, розмістити всі вихідні дані в пам'яті одночасно неможливо. Запропонована схема дозволяє розбити вихідні дані на незалежні частини, які можуть зберігатися окремо в локальній пам'яті різних вузлів. Тобто, якщо вихідна матриця була представлена в вигляді гіперматричної схеми g-го порядку, то гіперматриці всіх нижчих порядків можуть зберігатися окремо, за умови наявності у всіх вузлів інформації про них. Алгоритмічна обробка найпоширеніших операцій може виконуватись за допомогою блочних методів, таких як методи Фокса, Ланцоша та Відемана-Копперсміта.

Іншою поставленою задачею була можливість перетворення з існуючих схем на запропоновану за обчислювально простим алгоритмом. Гіперматричні схеми можуть мати підматриці, що зберігаються за різними схемами, тому застосування для гіперматриць першого порядку такої ж схеми, як і для вхідних даних дозволить перетворити їх шляхом тривіального введення коригуючого доданку при індексації. Однак, такий підхід може потенційно збільшити необхідний обсяг пам'яті для зберігання даних. Найбільш ефективним виявився підхід з використанням CSR схеми в якості базової, тому розглянемо перетворення з формату Matrix Market до гіперматричного CSR. Оскільки в першому зберігаються як стовпцеві, так і рядкові індекси, а в останньому в явному вигляді тільки стовпцеві, їх перетворення виконується тривіальним відніманням індексу першого стовпця в гіперматриці першого порядку. Індекси початку рядків можуть бути сформовані під час зміни рядкового індексу в вихідних даних. Таким чином маємо наступне обмеження на розмір вхідних даних: в пам'яті вузла мають бути розміщені один рядок під матриць другого порядку. Як буде показано нижче, це обмеження можна обійти при використанні більш складних алгоритмів роботи обчислювальних процесів.

Читання програмою вихідних даних з зовнішнього накопичувача може бути організоване наступним чином. Оскільки в багатьох кластерних системах фізичний доступ до системи зберігання даних має один або декілька вузлів, в той час як інші отримують дані мережею, недоцільно намагатися виконувати читання паралельно. Натомість рекомендується організувати обчислення за парадигмою «диспетчер-робітник», в якій процес-диспетчер виконуватиме введення даних та їх розсилку робітникам. При цьому він формуватиме загальну інформацію про знаходження певних частих даних в обчислювальних процесах. Оскільки швидкість обробки даних процесором значно вище швидкості передачі мережею, безпосереднє перетворення вхідних даних може виконуватись на керуючому вузлі, після чого дані для виконання алгоритмічної обробки можуть передаватися обчислювальним вузлам. Передачу даних та введення наступної їх частини можна сумістити в часі використовуючи асинхронні передачі, які доступні в MPI.

Разом з тим при виконанні обчислювального алгоритму в залежності від архітектури зв'язків кластера можна виконувати передачу дних як через керуючий вузол, так і між обчислювальними вузлами безпосередньо.

Авторська програмна реалізація виконана з використанням шаблонів мови C++ для оформлення класів різних схем зберігання. Параметризованим є тип елементу матриці. Це дозволяє використовувати каскадування різних схем зберігання в гіперматричну схему і разом

з тим накладає необхідне обмеження на однаковість типів елементів однієї матриці. Використання шаблонів було також викликане відсутністю накладних витрат на перевірку типів під час роботи програми та диспетчеризацію віртуальних викликів.

Архітектурно-залежні оптимізації

В залежності від архітектури процесора цільової системи та організації кластера в цілому можливе застосування деяких оптимізацій або існування додаткових обмежень.

Найбільш простим та ефективним розбиттям є таке, в якому кількість гіперматриць першого порядку кратна кількості обчислювальних процесів комп'ютерної системи. Такий підхід також значно спрощує програмування прикладних алгоритмів. У випадках, коли через інші обмеження така кількість підматриць не може бути створена, можливо до визначити вхідні дані певною кількістю нульових елементів. Це не вплине на роботу більшості алгоритмів, та незначно збільшить необхідний обсяг пам'яті, оскільки нульові елементи не зберігаються безпосередньо.

При застосуванні гіперматричних схем вищих порядків доцільно обирати їх розміри таким чином, щоб матриця першого порядку (або декілька їх) вміщувалась в кеш-пам'яті першого рівня процесора, а другого порядку – в кеш пам'яті другого рівня відповідно. Наприклад для використовуваних в кластері НТУУ «КПІ» процесорів розміри матриць з елементами з плаваючою комою становитимуть не більше 640x640 та 2560x2560 елементів відповідно (що відповідає розмірам кеш-пам'яті 512 кб та 8 Мб).

Висновки та перспективи

Запропонована схема зберігання розріджених матриць великих розмірностей задовольняє вимогам, що висуваються до таких схем та дозволяє зменшити обсяг пам'яті, що необхідний для зберігання таких даних, на 7-10%.

Рекомендований метод організації обчислень дозволить використовувати локальну пам'ять вузлів кластерної системи для збереження таких даних.

Отримані результати можуть створити передумови для подальшого розвитку програмних комплексів для розв'язку сучасних математичних та фізичних проблем та моделювання. Вони також можуть бути корисні при розробці програмного забезпечення для виконання аналізу криптографічної стійкості алгоритмів шифрування та для аналізу даних методом головних компонент, наприклад, в системах пошуку інформації в мережі.

Експерименти з запропонованою схемою показують, що застосування надмірно великого порядку гіперматриці призводить до використання більшого обсягу пам'яті та збільшення часу доступу до окремого елементу. Рекомендується для матриць розмірністю порядку $10^n \times 10^n$ застосовувати схеми не більше ніж $(n-4)$-го порядку.

В подальшому планується дослідити можливість застосування неквадратних підтмариць в запропонованій схемі, а також використання нерівномірних розбиттів для отримання більшого коефіцієнту ефективності гіперматричної схеми.

Важливим напрямом розвитку запропонованих ідей є розробка евристичного методу визначення областей матриці великої розмірності, що містять лише нульові елементи.

Автори висловлюють подяку суперкомп'ютерному центру НТУУ «КПІ» [7] та лабораторії Intel Manycore Testing Lab [8] за надані ресурси для досліджень.

Перелік посилань

1. *Opsahl, T.* Node centrality in weightednetworks: Generalizing degree and shortest paths [Text] / Opsahl T., Agneessens F., Skvoretz J. // Social Networks. – 2010. – Vol. 32, no. 3. – Pp. 245-251. – ISSN 03788733.
2. *Стиренко С.Г.* Форма представления функциональных программ для автоматического распараллеливания / Стиренко С.Г., Грибенко Д.В., Зиненко А.И. // Вісник НТУУ «КПІ». Інформатика, управління та обчислювальна техніка: Зб. наук. праць. – К.: «Век+», 2009. – №50. – с. 184-187.

3. *Wiedemann D.H.* Solving sparse linear equations over finite fields [Text] // IEEE Transactions of Information Theory. – New York, NY, USA: IEEE Press, 1986. – Vol. 32, no. 1. – Pp. 54-62. – ISSN 0018-9448.
4. *Sutter, H.* Software and concurrency revolution / H. Sutter, J. Laurus // Queue. – New York: ACM, 2005. – Vol. 3, № 7. – P. 54-62. – ISSN 1542-7730.
5. *Boisvert R.F.* Matrix market: a web resource for test matrix collections / R. Boisvert, R. Pozo, K. Remington, et. al. // In Proceedings of the IFIP TC2/WG2.5 working conference on Quality of numerical software: assessment and enhancement, *Ronald F. Boisvert (Ed.)* – : London, UK: Chapman & Hall, Ltd., 1997. – Pp. 125-137.
6. *Montagne E.* An optimal storage format for sparse matrices [Text] / Montagne E., Ekambaram A.// Information Processing Letters. – San Francisco, CA, USA: Information processing society, 2004. – Vol. 90, no. 2. – Pp. 87-92. – ISSN 0020-0190.
7. Суперкомп'ютерний центр НТУУ «КПІ» [електронний ресурс]. – Режим доступу: http://www.hpcc.org.ua/. – Останнє звернення 10.09.2010, назва з екрану.
8. Intel® Manycore Testing Lab [electronic resource]. – Access mode: http://software.intel.com/en-us/articles/intel-many-core-testing-lab/. – Останнє звернення 12.09.2010, назва з екрану.

УДК 681.513.2

ТОМАШЕВСЬКИЙ В.М.,
НОВІКОВ Ю.Л.,
КАМІНСЬКА П.А.

СКЛАДАННЯ РОЗКЛАДІВ ЗАНЯТЬ У ДИСТАНЦІЙНИХ СИСТЕМАХ НАВЧАННЯ

У статті розглянуто особливості побудови розкладів для дистанційної форми освіти з огляду на відмінності процесу класичного та дистанційного навчання. Подано математичну постановку загальної задачі побудови розкладу. Розглянуто методи та підходи до вирішення цієї задачі. Обгрунтовано вибір методу генетичного алгоритму та розроблено алгоритм побудови розкладу для дистанційного навчання, реалізований у якості підсистеми для системи дистанційного навчання «Віртуальний Університет».

The features of construction schedules for the distance form of education according to differences of classical and distance learning are described in the article. The mathematical formulation of the general problem of constructing the schedule is represented. Methods and approaches for solving this problem are considered. The choice of the method of genetic algorithm is grounded and the algorithm for constructing a schedule for distance learning, implemented as a subsystem for the distance learning system "Virtual University", is developed.

Вступ

Якість підготовки спеціалістів у вузах і особливо ефективність використання науково-педагогічного потенціалу залежать в значній мірі від рівня організації навчального процесу, який в значній мірі залежить від розкладу занять, що регламентує трудовий режим і впливає на творчу віддачу викладачів. Тому розклад занять можна розглядати як фактор оптимізації використання обмежених трудових ресурсів – викладацького складу.

Метою даного дослідження є автоматизація складання розкладу занять для системи дистанційного навчання „Віртуальний університет” з врахуванням завантаженості викладацького складу. Якість навчання, його економічна ефективність, зручність навчання студентів і роботи професорсько-викладацького складу залежить від вдало складання розкладу.

Потреба в розробці механізму складання розкладів виникла в зв’язку з необхідністю спрощення й прискорення розв’язання складної задачі – складення розкладу занять для системи дистанційного навчання „Віртуальний університет”. Задача складання розкладу є NP-складною, тобто такою, для якої досі невідомий алгоритм розв’язку, час виконання якого виражався б поліноміальною функцією. Складність задачі виявляться також у тому, що до розкладу занять, як правило, залучена велика кількість людей, причому кожен зі своїми інтересами та вимогами, які часто є суперечливі.

Тому для розв’язку таких задач застосовуються наближені методи, які дозволяють скласти субоптимальний розклад занять.

1. Аналіз умов і предметного середовища

Складаючи розклад занять, необхідно врахувати такі умови:

– забезпечення найбільш рівномірного навантаження на учасників навчального процесу;

– забезпечення гнучкості графіку з навантаження та з розміщення занять з урахуванням базових вимог,

– виключення різного роду „накладок” в розкладі,

– максимальна відповідність вихідним даним з урахуванням обмежень.

Обмеженнями в даному випадку можуть бути наступні:

1) жорсткі обмеження:

– режим роботи викладачів у системі дистанційного навчання;

– виключення святкових і неробочих днів.

2) відносно гнучкі обмеження:

– план-графік тестових контролів;

– план-графік інших занять;

– періоди підвищеної зайнятості у деяких викладачів або груп студентів.

Для складання розкладу важливою вхідною інформацією є організаційна структура підрозділів навчльного закладу. Наприклад, візьмемо типову структуру університету, що включає:

– факультети (потрібна наявність інформації про назву, декана);

– кафедри (потрібна наявність інформації про назву, завідуючого);

– спеціальності (потрібна наявність інформації про назви);

– групи (потрібна наявність інформації про назви, дату утворення, поточний семестр навчання і курс);

– дисципліни (потрібна наявність інформації про викладача, максимальний бал, номер навчального семестру);

– викладацький склад;

– студентський склад.

Для складання реального розкладу занять для дистанційного способу навчання необхідно мати наступну інформацію.

1. Кількість годин роботи викладача у семестрі, що відведені на певну дисципліну. Цей час включає:

– відеоеонференції (лекції, віртуальні заняття, тренінги, конференції, презентації, доклади, он-лайн семінари)

– консультаційні заняття (пояснення щодо пройденого матеріалу дисципліни, передтестові консультації, відповіді на запитання);

– практичні (лабораторні) заняття (віртуальна присутність викладача як спостерігача і координатора та консультування через інтерактивне спілкування);

– перевірку тестових (контрольних та екзаменаційних) робіт.

2. Орієнтовний обсяг академічного навантаження для студента (кількість годин у семестрі) по певній дисципліні. Цей час включає самостійну роботу з:

– вивчення лекційного матеріалу;

– виконання практичних, лабораторних і контрольних робіт;

– тестування (здавання екзаменів і заліків).

3. Розподілення видів освітньої діяльності за періодами:

– послідовність вивчення матеріалів і перевірки знань по них згідно графіку навчання;

– час на опанування тієї або іншої дисципліни (підготовка до тестування, обробка результатів тестування, робота над помилками, тощо).

2. Математична постановка задачі складання розкладу

Задача складання розкладу є однією з найбільш розповсюджених задач, які вирішує кожна людина (свідомо або ні) практично кожен день. У загальній постановці вона є процесом упорядкування деякою скінченою множиною подій в часі за умов ресурсних та інших обмежень. Мета полягає в тому, щоб при заданих властивостях для кожної події (завдання) і обмеженнях на ресурси знайти ефективний алгоритм впорядкування завдань, який оптимізував би певний критерій ефективності. За критерій оцінки розкладу звичайно беруть середні, середні зважені або максимальні значення:

– моментів часу закінчення робіт;

– тривалостей виконання робіт;

– часового зміщення;

– запізнювання;

– випередження;

– тривалості очікування.

Математичні моделі для цих задач є дискретними, детермінованими тому, що вся інформація, на основі якої приймаються рішення щодо упорядкування, відома заздалегідь. Більшість задач складання розкладу є багатокритеріальними, тому що до навчального процесу залучена велика кількість людей, кожен з яких має свої інтереси та вимоги, задоволення яких часто призводить до суперечливих ситуацій.

Задача складання розкладу може бути представлена як задача цілочисельного математичного програмування. При даному підході інтереси суб'єктів навчального процесу враховуються у вигляді обмежень або редукованих критеріїв оптимальності. Наведемо математичну модель задачі складання розкладу навчальних занять студентів дистанційної форми навчання. Введемо наступні позначення.

1. У вузі є N учбових груп, що навчаються дистанційно, які об'єднані в R потоків, r – номер потоку ($r = \overline{1,R}$); k_r – номер учбової групи на потоці r, ($k_r = \overline{1,G_r}$), G_r – кількість груп на потоці r.

2. Заняття проводяться в робочі дні. t – номер робочого дня тижня, T – загальна кількість робочих днів учбового закладу ($t = \overline{1,T}$). Для кожної групи k_r визначається множина номерів

робочих днів цієї групи T_{k_r}, яка є підмножиною всіх робочих днів учбового закладу.

На відміну від класичного аудиторного навчання, дистанційна форма навчання дозволяє проводити заняття у найбільш зручний для учасників навчального процесу час, тому класичним розподілом навчального часу, прив'язаним до часових інтервалів, які прийнято називати „парами", можна знехтувати. Отже, кожен учбовий день розбивається на години, що починаються з J_s і закінчуються годиною J_f. Значення J_s і J_f можуть залежати від робочого часу викладачів (деяким викладачам зручніше проводити заняття у першій або у другій половині дня); j – конкретна учбова година ($j = \overline{J_s, J_f}$).

3. На основі учбового плану для кожного потоку складається список із S_r лекційних занять, з номером конкретної дисципліни в списку $s_r = \overline{1, S_r}$. Для кожної групи k_r складається список із Q_{k_r} запланованих консультаційних занять, q_{k_r} – номер дисципліни в списку ($q_{k_r} = \overline{1, Q_{k_r}}$). Загальний список занять W_{k_r} для кожної групи буде складатися із усіх дисциплін, що присутні в даних списках ($W_{k_r} = S_r + Q_{k_r}$). Крім того, якщо по деякій дисципліні протягом тижня проводиться більше одного заняття, то вона вказується в списку лекцій або консультаційних занять стільки раз, скільки це передбачено навчальним планом.

4. Нехай p – номер (ім'я) викладача, який проводять лекційні або консультаційні заняття, P – множина всіх викладачів ($p = \overline{1, P}$).

Для кожного викладача p введемо наступні булеві змінні:

$$\delta^{p}_{rs_r} = \begin{cases} 1, & \text{якщо на потоці } r \text{ лекцію } s_r \text{ читає викладач } p, \\ 0, & \text{в протилежному випадку;} \end{cases}$$

$$\Delta^{p}_{rk_r q_{k_r}} = \begin{cases} 1, & \text{якщо в групі } k_r \text{ заняття } q_{k_r} \text{ проводить викладач } p, \\ 0, & \text{в протилежному випадку;} \end{cases}$$

Оскільки учбове навантаження викладачів планується до складання розкладу занять, то ці величини вважаються заданими. Також вражається заданою для кожного викладача p його навантаження – N_p годин за тиждень ($p = \overline{1, P}$).

Дистанційне навчання не прив'язане до аудиторій, тому це дозволяє не обмеження на аудиторії, що є дуже критичним класичному навчанні. Це дозволить скласти розклад, який задовольняє потребам і побажанням учасників навчального процесу.

При даному представленні початкових даних задача складання розкладу занять полягає у визначенні для кожного заняття дня тижня і пари в ньому з урахуванням накладених на розклад обмежень. Вводяться наступні групи булевих змінних, які необхідно знайти.

$$\eta^{tk_r}_{j} = \begin{cases} 1, & \text{якщо в день } t \text{ група } k_r \text{ закінчує заняття о } j\text{-й годині} \\ 0, & \text{в протилежному випадку;} \end{cases}$$

$$\xi^{tk_r}_{j} = \begin{cases} 1, & \text{якщо в день } t \text{ група } k_r \text{ починає заняття о } j\text{-й годині} \\ 0, & \text{в протилежному випадку;} \end{cases}$$

$$y^{ts_r}_{rtj} = \begin{cases} 1, & \text{якщо на потоці } r \text{ в день } t \text{ у час } j \text{ читається лекція } s_r \\ 0, & \text{в протилежному випадку;} \end{cases}$$

$$x^{qk_r}_{rk_rtj} = \begin{cases} 1, & \text{якщо в групі } k_r \text{ в день } t \text{ у час } j \text{ проводиться консультація } q_{k_r} \\ 0, & \text{в протилежному випадку.} \end{cases}$$

Обмеження «кожного дня на кожній парі для кожної групи може проводитися не більше одного заняття» має наступний вигляд:

$$\sum_{q_{k_r}=1}^{Q_{k_r}} x^{q_{k_r}}_{rk_rtj} + \sum_{s_r=1}^{S_r} r^{s_r}_{rtj} \le 1,$$

де $r = \overline{1, R}$, $k_r = \overline{1, G_r}$, $t \in T_{k_r}$, $j = \overline{J_s, J_f}$.

Інша обов'язкова умова складання розкладу для учбових закладів «для кожної учбової групи повинні виконуватися всі передбачені планом види робіт протягом тижня» має наступний вигляд:

$$\sum_{t \in T_{k_r}} \sum_{j=1}^{J} \left(\sum_{q_{k_r}=1}^{Q_{k_r}} x^{q_{k_r}}_{rk_rtj} + \sum_{s_r=1}^{S_r} y^{s_r}_{rtj} \right) = W_{k_r},$$

де $r = \overline{1, R}$, $k_r = \overline{1, G_r}$.

Ще одним важливим обмеженням є відсутність «накладок», тобто розклад дистанційних занять у кожного викладача не повинен перетинатися з розкладом його аудиторних занять.

У конкретних задачах список обмежень може бути продовжений.

Після складання обмежень, необхідно обрати критерій оптимальності розкладу. Це складне питання через принципову багатокритеріаль-

ність задач складання розкладу. Зокрема, можна розглядати які критерій максимальне сукуне ущільнення навантаження викладачів. У такому випадку критерій має вигляд:

$$\sum_{t=1}^{T}\sum_{p=1}^{P}\xi_p z_t^p \to \max ,$$

де ξ_p – ваговий коефіцієнт, який визначається статусом викладача; z_t^p – булева змінна, яка приймає значення 0, якщо даний викладач *p* має заняття в день *t*, і 0 – в протилежному випадку.

Дана модель відображає основні фактори, які необхідно враховувати при складанні розкладу і відноситься до класу задач лінійного булевого програмування. Однак, розв'язання таких задач навіть відносно малої розмірності (для невеликих навчальних закладів) звичайно вимагає значних витрат часу. Побудована таким чином модель, при дотриманні основних вимог до розкладу, повинна включати тисячі обмежень і десятки тисяч змінних. Стає зрозуміло, що не лише цілочисельна, а й проста задача лінійного програмування подібної розмірності не має перспектив на оптимізацію точними методами. Цей факт цілком пояснює, чому при зіставленні розкладу занять на практиці користуються зазвичай евристичними методами [1].

Задача складання розкладу в дистанційному навчанні є *NP*-повною. Для розв'язання *NP*-повних задач досі невідомо алгоритмів, час роботи яких виражався б поліноміальною функцією, але й не доведено, що для якоїсь із них такого алгоритму не існує. Тому розв'язувати дану задачу методом повного перебору немає сенсу і слід використовувати наближені методи:

– імітації відпалу;
– розфарбування графу;
– імітаційного моделювання;
– логічного програмування з обмеженнями;
– генетичного алгоритму.

Розглянемо подані методи

3. Метод імітації відпалу

Ідея алгоритму імітації відпалу [2] запозичена із досліджень поведінки атомів металу в процесі його відпалу. В металі, нагрітому до температури, яка перевищує точку його плавлення, атоми знаходяться в безладному русі. При цьому, як і в усіх фізичних системах, вони прагнуть до стану мінімуму енергії (єдиного кристалу), але при високих температурах енергія атомного руху перешкоджає цьому. В процесі поступового охолодження металу виникають все більш низькоенергетичні стани, поки не буде досягнуто найнижчий з можливих станів – глобальний мінімум.

Переходячи до математичної моделі, визначимо, що в задачі складання розкладу може грати роль енергії, і стан якого об'єкту така «енергія» може характеризувати. Один із можливих варіантів – це розгляд такої енергії у вигляді цільової функції, яка основана на штрафах, що додаються до поточного розкладу за кожний невдалий розклад, а коректний (хоча і невідомий) розклад – розглядати як низькоенергетичний стан.

Загальну ідею алгоритму імітації відпалу для задачі складання розкладу можна представити наступною схемою.

1. На першій ітерації генерується деякий коректний початковий розклад X^0, який вважається поточним розв'язком задачі ($X = X^0$).

2. Задається початкове, високе значення температури T^0 і операція мутації розкладу. Мутація може бути реалізована наступними діями:

– зміна часу проведення занять;
– обмін місцями в розкладі двох занять.

3. На основі введених операцій мутації та поточного розв'язку генерується новий коректний розклад X', який дещо відрізняється від попереднього.

4. Визначається зміна цільової функції $\Delta f = f(X) - f(X')$:

– якщо $\Delta f < 0$ (розв'язок не погіршився), то новий варіант розкладу стає поточним ($X = X'$);

– якщо $\Delta f > 0$ (розв'язок погіршився), то новий розклад стає поточним лише з ймовірністю $p = e^{-\Delta f / T}$. Відповідно, з ймовірністю (1-*p*) попередній розклад зберігається як поточ-

ний.

Таким чином, допускається збільшення цільової функції, але ймовірність цього експоненціально зменшується з ростом Δf. Такий підхід дозволяє долати локальні екстремуми.

5. Виконується функція зміни поточної температури. Температура, і, відповідно, ймовірність прийняти поточного розкладу з більшим значенням цільової функції зменшується на кожній ітерації. Величину зменшення температури можна зв'язати з числом виконаних ітерацій, або з величиною зміни цільової функції. При високих температурах спостерігається стрибкоподібна зміна цільової функції. Поступово зі зменшенням температури її значення прагне до мінімуму.

6. Якщо не виконується критерій зупину, то перейти до п. 3. Для критерію зупину можуть бути прийняті наступні умови:

– виконання заданого числа ітерацій;

– виконання заданого числа ітерацій без покращення значення цільової функції на задане значення.

Для підвищення ефективності алгоритму можна реалізувати збереження *m* кращих варіантів, а також *n* останніх згенерованих розкладів. Це дозволить запобігти зацикленню процесу складання розкладу.

Точність розв'язку, отриманого за даним алгоритмом, можна збільшити за рахунок більш повільної зміни температури. В такому випадку алгоритм більш детально досліджує простір пошуку, але час його роботи суттєво збільшується. Тому функцію температури доводиться підбирати для кожної конкретної задачі індивідуально. Незважаючи на зовнішню простоту, такий підхід може виявитися цілком ефективним для складання невеликих розкладів.

4. Метод розфарбування графу

Задачу складання розкладу можна розглядати як задачу розфарбування графу. Задачею розфарбування графу називають пошук хроматичного числа графу або, іншими словами, пошук мінімального числа кольорів, необхідних для розфарбування вершин деякого графу таким чином, щоб для кожної пари сусідніх вершин використовувалися різні кольори [4]. Сама задача пошуку хроматичного числа є *NP*-повною, для розв'язання якої в більшості випадків використовують різного роду жадібні алгоритми.

Для постановки задачі складання розкладу як задачі розфарбування графу будується граф, в якому кожна вершина представляє собою заплановане навчальним планом заняття. У випадку, коли між деякими двома вершинами можливі конфлікти (наприклад, обидва заняття проводяться в однаковий час з одним викладачем), то вони з'єднуються ребром. Це еквівалентно забороні одночасного проведення цих занять. Тоді задача складання розкладу представляється як мінімізація числа кольорів, нео бхідних для розфарбування графа. Кожен колір відп овідає одному періоду занять.

Застосування цього підходу для розв'язку реальних задач, очевидно, малоефективне. Але задача розфарбування графа при складанні розкладу може бути корисною у випадку її комбінації з іншими алгоритмами.

5. Метод імітаційного моделювання

Враховуючи *NP*-повноту задачі, для її розв'язання можна застосувати імітацію дій диспетчера при складанні розкладу. В цьому випадку алгоритм оперує безпосередньо розкладом і списком занять, які необхідно включити до розкладу (навчальним планом). Процес складання розкладу розпочинається з пустого розкладу, коли всі заняття знаходяться в списку неврахованих занять. Далі алгоритм переходить від одного незавершеного розкладу до другого, намагаючись найліпшим чином розставити усі заняття, які внесені до списку. Процес триває доти, поки не буде сформований повний розклад або виконається фіксоване число ітерацій.

Наведемо схему цього алгоритму більш детально. Даний алгоритм складається з трьох основних дій [4].

1. Вибирається ще не включене в розклад заняття. Вибір робиться на підставі аналізу «вузьких місць». В даному випадку вузькими місцями є найбільш дефіцитні ресурси: викладачі, студенти та час. В першу чергу зіставляється розклад для найбільш дефіцитних ресурсів. Це можуть бути заняття, що проводяться викладачами, які ставлять жорсткі умови щодо часу їх проведення та інше.

2. Для обраного заняття визначаються всі можливі варіанти його розміщення в розкладі, які задовольняють усі жорсткі обмеження. Далі

кожна така позиція оцінюється за допомогою спеціальної евристичної цільової функції і заняття розміщується в найкращу із можливих позицій.

3. Якщо у випадку п. 2 в розкладі виникла конфліктна ситуація, то «конфліктуючі» заняття видаляють з розкладу і вносять знову у список неврахованих занять.

При реалізації алгоритму, який оснований на принципах імітаційного моделювання, особлива увага приділяється розробці еВристичних правил вибору наступного заняття зі списку, визначенні найкращої для нього позиції в розкладі та оцінці отриманого розкладу з виконанням обов'язкових обмежень.

До позитивних сторін такого підходу можна віднести можливість детального врахування специфіки задачі, що розв'язується, у випадку складання розкладу для конкретного вузу. Однак, у такому випадку сильно обмежується можливість застосування розробленої системи в інших учбових закладах. Крім цього, очевидно, доведеться вносити суттєві зміни до алгоритму при незначних внутрішніх змінах у вузі. Оскільки алгоритм оснований на діях, які виконує диспетчер при складанні розкладу, то існує можливість організувати діяльний діалог між користувачем і системою при пошуку найкращого розкладу. Хоча це справедливо лише для відносно невеликих задач, так як в протилежному випадку значна залежність алгоритму від дій користувача може зробити такий діалог малоефективним.

6.Метод логічного програмування з обмеженнями

Складання розкладу можна представити як задачу задовільнення обмежень. Для розв'язку таких задач розроблено достатньо алгоритмів, починаючи с класичного методу Гаусса і закінчуючи складними методами, які застосовуються в системах доведення теорем і в системах символьних обчислень. Виник навіть цілий напрямок в програмуванні – програмування з обмеженнями.

Програмування з обмеженнями тісно пов'язано з традиційним логічним програмуванням, в рамках якого воно сформувалося. Більшість систем програмування в обмеженням представляють собою звичайний інтерпретатор мови Пролог з вбудованим механізмом для розв'язку певного класу задач задовільнення обмеженням. Програмування в таких системах називають логічним програмуванням з обмеженнями (Constraint Logic Programming або CLP).

Ідея розв'язання задач полягає в тому, що програміст визначає деяку множину змінних $x_1,\ldots,x_n$ та область їх значень $X_1,\ldots,X_n$, описує додаткові обмеження, яким повинні задовольняти змінні, а система повинна знайти такі значення цих змінних, які б задовольняли одночасно усім заданим обмеженням.

Наведемо невеликий приклад вищезгаданого методу. Розглянемо вуз, в якому працює M викладачів, в учбовому плані зафіксовано N дисциплін. Визначимо множину $P = 1,2,\ldots,D$, елементами якої є всі періоди навчання (години) у вузі протягом робочого тижня, а D – число всіх періодів навчання протягом тижня. Нехай y_{ij} - період навчання, в якому i -й викладач веде предмет j, $y_{ij} \in P$. Тоді обмеження „кожен викладач в кожен момент часу може вести не більше одного заняття" має вигляд:

$$y_{ij} \neq y_{ij'}, 1 \leq i \leq M, 1 \leq j \leq N, 1 \leq j' \leq N, j \neq j'.$$

Аналогічним чином складаються інші обмеження.

Результатом роботи алгоритму є множина значень кожної змінної, що не суперечать вказаним обмеженням. При цьому область визначення змінних, що входять до строгих обмежень, може суттєво зменшитися або навіть матиме єдине значення. Основна перевага, яку можна отримати при використанні CLP, – скорочення простору пошуку. Це досягається не шляхом оцінки кожного варіанту розкладу, а за рахунок того, що система сама виключає з розгляду неперспективні варіанти.

7. Метод генетичного алгоритму

Усі розглянуті вище методи в своїй основі використовують ітераційну техніку для покращення результатів. На протязі однієї ітерації вони шукають розв'язок, який буде кращий в околі даного. Якщо таке рішення знайдено, воно стає поточним і починається нова ітерація. Так триває доти, поки приріст цільової функції

не зменшиться практично до нуля або не виконається задане число ітерацій. Очевидно, що такі методи орієнтовані на пошук лише локальних оптимумів, до того ж розташування знайденого оптимуму залежить від стартової точки. Глобальний оптимум може бути знайдений лише випадково. Для збільшення ймовірності знаходження глобального оптимуму використовується множинний експеримент з різними початковими точками, що суттєво збільшує час пошуку.

В зв'язку з цим представляє інтерес розробка алгоритмів, які б зберігали переваги описаних методів і не мали б вказаного недоліку. Саме до таких відносяться генетичні алгоритми. Тому вони були обрані як метод розв'язування задачі складання розкладу в даній дипломній роботі.

Генетичні алгоритми – це стохастичні евристичні оптимізаційні методи, основна ідея яких взята з теорії еволюційного розвитку видів [5]. Основним механізмом еволюції є природний відбір, суть якого полягає в тому, що більш пристосовані особини мають більше шансів на виживання та розмноження і, відповідно, приносять більше потомства, ніж менш пристосовані особини. При цьому завдяки передачі генетичної інформації нащадки успадковують від батьків основні їх якості. Носіями генетичної інформації індивіда виступають молекули ДНК. При розмноженні живих організмів відбувається злиття двох батьківських статевих клітин. Їх ДНК взаємодіють, утворюючи ДНК нащадка. Основний спосіб взаємодії – кросовер. При кросовері ДНК предків діляться на дві частини, а потім обмінюються своїми половинками. При спадкуванні можливі мутації через радіоактивність або інші впливи, в результаті яких можуть змінитися деякі гени в статевих клітинах одного з батьків. Змінені гени передаються нащадку і надають йому нових властивостей. Якщо ці нові властивості є корисними, то вони, швидше за все, збережуться в даному виді и при цьому відбудеться стрибкоподібне підвищення пристосованості виду.

Першим кроком при розробці математикної моделі, основаної на генетичному алгоритмі, є розробка структури хромосоми, в якій буде зберігатися розв'язок. У нашому випадку такою „хромосомою" є розклад.

Обрана структура повинна враховувати всі особливості і обов'язкові обмеження, які висуваються до шуканого розкладу, а також те, що від її вибору напряму залежить реалізація алгоритмів кросоверу та мутації. Загалом, вибір „хромосоми" впливає не лише на швидкість, але й на збіжність алгоритму в цілому.

На наступному кроці алгоритму створюється початкова популяція, розмір якої залежить від розмірності задачі та складає звичайно декілька сотень рішень.

Для організації оптимізаційного процесу необхідно створити направляючу силу розвитку популяції. Такою силою може виступати вимога мінімізації цільової функції або, в термінах генетичних алгоритмів, функція придатності. Зазвичай як функцію придатності використовують адитивний показник оптимальності, який оснований на штрафах, що встановлюються кожному розв'язку за будь-який незручний момент в розкладі. Перевагою такого вибору є можливість налаштування алгоритму під конкретну задачу шляхом варію вання коефіцієнтів і, тим самим, зміни пріоритету при пошуку оптимального розкладу.

Таким чином, розмістивши початкову популяцію в створене штучне середовище та реалізувавши процеси селекції, кросоверу та селекції, ми отримаємо ітераційний алгоритм пошуку оптимального рішення, на кожній ітерації якого виконуються такі дії:

1) кожна особина популяції оцінюється за допомогою функції придатності;

2) кращі рішення (звичайно близько 5 %) копіюються в нову популяцію без змін. Такий принцип (принцип елітизму) не дозволяє втратити кращі рішення і забезпечує підвищену збіжність алгоритму;

3) на основі пропорційного відбору з поточної популяції вибираються два рішення, які піддаються рекомбінації. Для цього хромосоми батьків обмінюються відповідними частинами;

4) отриманий на попередньому етапі розклад може виявитися некоректним, наприклад, може не відповідати навчальному плану. В цьому випадку можна повторювати операцію рекомбінації до того часу, коли не буде отриманий коректний розклад, у такому випадку краще передбачити евристичний механізм виправлення розкладу;

5) якщо нова популяція сформована, то стара видаляється, після чого переходимо до п. 1. У протилежному випадку переходимо до п. 3.

Розглянутий алгоритм є не лише стійким до локальних мінімумів, але й завдяки внутрішньому паралелізму, який працює не з окремими розв'язками, а з цілими класами розв'язків, забезпечується відносно швидкий пошук оптимального рішення.

8.Обґрунтування доцільності використання генетичного алгоритму

Серед існуючих методів розв'язку задач складання розкладів занять навчальних закла-дів необхідно вибрати такий, який дозволить врахувати специфіку дистанційного навчання. Для таких закладів проблема навчальних аудиторій вирішена, адже студентам і викладачам не має необхідності бути присутніми у одному й тому самому місті в один і той же час. Це, в свою чергу, накладає нові вимоги до вибору методу розв'язування задачі складання учбового розкладу. Також необхідно вибрати оптимальний метод з точки зору якості його роботи. Найбільш важливим є вибір критеріїв оптимізації, а також накладання різного роду обмежень на результати роботи методу. Вибраний метод повинен задовольняти вимогам швидкодії, бажана відсутність нестійкості до локальних оптимумів.

Не менш складною є задача реалізації обраного методу в рамках системи дистанційного навчання «Віртуальний Університет». Також необхідно внести зміни у саму постановку задачі, що в свою чергу потребує модифікації алгоритму.

Врахувавши специфіку задачі та переваги і недоліки методів, метод генетичного алгоритму найбільш пристосований та ефективний для розв'язання задачі складання розкладів.

Ідея генетичних алгоритмів (ГА) перейнята у самої природи. Розробник ГА повинен правильно встановити закони еволюції, щоб досягти бажаної мети якомога швидше. Сама ідея здається досить цікавою, а багаточисельні позитивні результати ще більше розпалюють інтерес до цього методу.

Незважаючи на цей факт, існує чимало думок, які ставлять під сумнів успішне використання ГА. Наведемо обґрунтування дієздатності ГА [3] та його спроможність ефективно вирішувати поставлені задачі.

Звісно, ГА не претендує на роль універсального засобу розв'язання задач: в одному випадку він хороший, а в іншому – не дуже. В загальному випадку ГА не знаходить оптимального розв'язку дуже складних задач, але з іншого боку, ГА може знайти непогане рішення. Особливо, якщо розмірність задачі є значною. Існує багато прикладів, коли в дуже складних задачах за допомогою ГА були отримані хороші результати.

Визначимо загальні переваги ГА:

– він не потребує ніякої інформації про поверхню відгуку;

– розриви, що присутні на поверхні відгуку, мають несуттєвий вплив на повну ефективність оптимізації;

– він стійкий до попадання в локальні оптимуми;

– добре працює при вирішенні крупномасштабних проблем оптимізації;

– використовується для широкого класу задач;

– простий та прозорий в реалізації;

– може використовуватися в задачах зі змінним середовищем.

– До недоліків ГА можна віднести:

– не бажано використовувати його, якщо необхідно знайти точний глобальний оптимум;

– не бажано використовувати його, якщо необхідно знайти всі розв'язки задачі, а не один;

– значний час роботи алгоритму.

Необхідність пошуку всіх рішень також не є проблемою для цього алгоритму. Якщо дослідженню простору пошуку приділити достатньо уваги, то ГА зможе знайти декілька (або навіть всі) екстремуми.

В обох випадках простори пошуку є одномірними. Хромосома описує одне число, яке є розв'язком задачі. Розмірність простору пошуку для більш складних задач може сягати кілька сотень. У такому випадку зображати будь-які графіки неможливо, а програми пошуку рішень є досить складними.

Розглянемо один з простих методів пошуку – метод сходження. Він працює наступним чи-

ном: спочатку генерується випадковий розв'язок, далі в нього вносять одиничне збурення і такий збурений розв'язок порівнюють з вихідним. Якщо новий розв'язок кращий, то він зберігається, а старий відкидається, в протилежному випадку – зберігається попередній (старий) розв'язок. Після цього процес повторюється зі збереженим розв'язком, але вже з іншим одиничним збуренням і так у подальшому. Якщо, наприклад, на протязі тисячі ітерацій розв'язок не покращився, то він вважається оптимальним і процес зупиняється. При сходженні завжди рухаються вверх (в розумінні функціоналу якості розв'язку). Якщо простір відгуку простий і містить лише один пік, то методом сходження можна знайти оптимальний розв'язок. Якщо ж він більш складний, і сходження відбувається поблизу одного з невеликих піків (локальних максимумів), то глобальний оптимум навряд або буде знайдений.

Генетичний алгоритм діє більш тонко. Генетична популяція містить багато розв'язків, які розосереджені всюди на просторі пошуку. Мутація діє аналогічно одиничному збуренню в методі сходження і приводить, в основному, до блукання поблизу того піку, на якому знаходиться розв'язок. Однак процедура кросоверу породжує нові точки, які можуть попадати на віддалені піки.

У цьому полягає основна відмінність генетичних алгоритмів від більш простих, таких як сходження. ГА ефективно проглядає простір пошуку, зберігаючи хороших „батьків", які частіше, ніж погані, дають хороше потомство (кращі розв'язки). Тому такий алгоритм з більшою ймовірністю знайде кращий розв'язок задачі.

Існує два випадки, коли ГА дуже хороші. Перший випадок: коли не відомий спосіб точного розв'язання задачі. Якщо відомо, як оцінити пристосованість хромосом, то завжди можна примусити ГА розв'язувати дану задачу. Другий випадок: коли спосіб для точного розв'язання існує, але він дуже складний в реалізації і потребує значних витрат часу.

9. Застосування методу генетичного алгоритму для розв'язку задачі складання розкладу занять в системі дистанційного навчання «Віртуальний Університет»

Для задачі складання розкладу відобразимо у блоці занять навчальний план, перелік викладачів, перелік учбових груп, перелік учбових дисциплін. Блок занять представлений у вигляді двовимірного масиву $Z[i,j]. Стовпець масиву містить інформацію про викладача, дисципліну, що він викладає, групу, для якої викладає, та тип дисципліни. Таким чином, блок занять являє собою агрегований об'єкт (рис. 1).

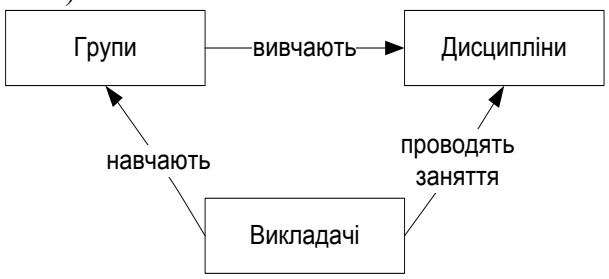


Рис. 1. Структура об'єкта «блок занять»

Множину блоків занять представимо у такому вигляді:

$$Z = \{z_i\},\ \{z_i\} = \{z_i^p, z_i^d, z_i^g, z_i^t\},$$

де z_i^p – викладач, що виклалає даний блок занять, $z_i^p = \overline{1, N_{pr}}$, N_{pr} – загальне число викладачів; z_i^d – дисципліна, що викладається в даному блоці занять, $z_i^d = \overline{1, N_{dsc}}$, N_{dsc} – загальне число дисциплін; z_i^g – навчальна група, для якої проводяться заняття даного блоку, $z_i^g = \overline{1, N_{grp}}$, N_{grp} – загальне число груп; $z_i^t \in \{1, N_{type}\}$ – тип, до якого відноситься дисципліна даного блоку занять (тип дисципліни відображає її приналежність до математичних, гуманітарних або іншої групи дисциплін (вводиться в модель для усунення випадків, коли в один і той самий день групі студентів будуть викладатися дисципліни одного типу, що значно зменшить ефективність навчання і сприняття матеріалу); N_{type} – загальне число типів дисциплін.

Часова сітка занять, кількість пар у дні тижня, кількість учбових днів у тижні задані у блоці часових інтервалів. Блок часових інтервалів представлений у вигляді двовимірного масиву *$T*[*i,j*]. Стовпець масиву містить інформацію
про тиждень, день і час.

Множину часових інтервалів представимо у такому вигляді:

$$T = \{t_k\},\ \{t_k\} = \{t_k^w, t_k^d, t_k^p\},$$

де t_k^w – номер тижня, $t_k^w = \overline{1, N_{wps}}$, N_{wps} – число навчальних тижнів у семестрі; t_k^d – номер дня тижня, $t_k^d = \overline{1, N_{dpw}}, N_{dpw}$ – число навчальних днів у тижні; t_k^p – номер години протягом дня, $t_k^p = \overline{N_S, N_F}$, N_S – початок навчального дня, N_F – кінець навчального дня. Доцільно використовувати найбільший відрізок часу, наприклад, з 8:00 до 21:00, адже потрібно призначити заняття у найбільш зручний для учасників навчального процесу час, а деякі викладачі можуть надавати перевагу другій половині дня.

Побажання викладачів стосовно часу проведення занять задані у блоці переваг викладачів. Блок переваг викладачів представлений у вигляді двовимірного масиву *$P*[*i,j*]. Для кожного викладача є стовпець, який містить інформацію про бажані часові інтервали (години для навчання).

Розклад дистанційних занять повинен враховувати аудиторне навантаження викладачів, оскільки вони здебільшого спершу викладають за класичною схемою аудиторні заняття, а вже потім дистанційні. Тому потрібно знати розклад занять викладачав, щоб уникнути «накладок» дистанційних занять на аудиторні. Тому розклад аудиторних занять представимо у вигляді двовимірного масиву *$PA*[*i,j*]. Для кожного викладача є стовпець, який містить інформацію про часові інтервали, зайняті аудиторними заняттями.

Окрім цього важливою вхідною умовою, що може бути врахована у дистанційному навчанні, це часові інтервали, у які викладачі не хотіли б (не можуть) проводити заняття.

Зберігатимемо ці умови у вигляді двовимірного масиву *$PX*[*i,j*]. Для кожного викладача є стовпець, який містить інформацію про небажані часові інтервали.

У результаті роботи буде отримано варіант розкладу, що відповідним чином враховує задані умови. Розклад зберігається у хромосомі.

Хромосома представлена у вигляді асоціативного масиву *$Chromosome*[*i*], що зв'язує блок занять та блок часових інтервалів. Індекси масиву – номери блоків занять (посилання на індекс). Значення масиву – номери блоків часових інтервалів.

Кожна хромосома особини складається із числа генів, рівних числу блоків занять. Інформаційним наповненням хромосоми є часові інтервали. Для кожної хромосоми визначена оцінка (штраф).

Побудову хромосоми проілюструє приклад, показаний на рис. 2.

Для заданої множини об'єктів *T, Z* потрібно знайти варіант розкладу, що забезпечує мінімальне значення критерію *P* втрат "якості" розкладу:

$$P = f(\tau) = \sum_{i=1}^{N} c_i w_i(\tau) \to \min .$$

де w_i значення коефіцієнта штрафу за невиконання *i*-ї частки критерію, c_i оцінка, що визначає ступінь невиконання *i*-ї частки критерію, τ – елемент з множини часових інтервалів *T*.

При цьому необхідно забезпечити виконання всіх заданих обмежень: відсутність накладок всіх типів (для викладачів, навчальних груп, аудиторних і дистанційних занять викладачів), необхідність проведення всіх запланованих на семестр і передбачених навчальним планом занять, врахування часових інтервалів, у які викладачі не можуть проводити заняття.

Вимоги при яких досягається найкращий розклад можуть бути наступні:

– дотримання рівномірності розподілу занять протягом тижня;

– дотримання необхідної відповідності між характером проведених занять і часовим інтервалом його проведення (наприклад, не варто проводити поспіль заняття одного й того ж типу, зокрема, лабораторні);

– урахування побажань викладачів щодо свого розкладу занять;

– вимоги, зв'язані із забезпеченням комфортності умов роботи викладачів і навчання сту-

дентів даного вузу.

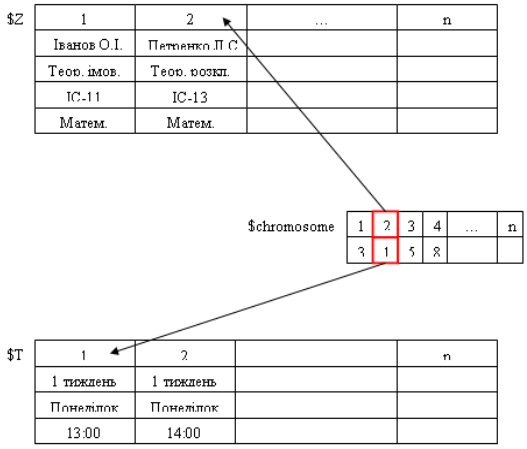


Рис. 2. Зв'язок між хромосомою, блоками занять і блоками часових інтервалів

Для запропонованої структури особин генетичного алгоритму реалізовані модифіковані процедури схрещування й мутації. Процедура схрещування в генетичному алгоритмі відбувається в такий спосіб: для двох обраних особин провадиться обмін частками генетичного коду між відповідними хромосомами (рис. 3). Точка поділу L вибирається випадково.

Оператор мутації (рис. 4) з деякою ймовірністю змінює значення декількох генів у хромосомі "нової" особини на інші значення, що входять у число припустимих значень даного гена. Наприклад, у результаті мутації i-того гена хромосоми, де i – визначає номер навчальної пари, i-тому гену буде привласнений номер навчальної пари із припустимої підмножини навчальних пар, випадково обраний з підмножини навчальних пар, призначених для проведення саме даного виду заняття.

Запропоновані процедури схрещування й мутації дозволяють враховувати ряд вимог, запропонованих до розкладу занять. Для роботи генетичного алгоритму також необхідним є використання ефективного методу відбору особин у наступну популяцію (популяцію нащадків). Для цього в алгоритмі використаний метод елітарного відбору, що сприяє збереженню кращих рішень.

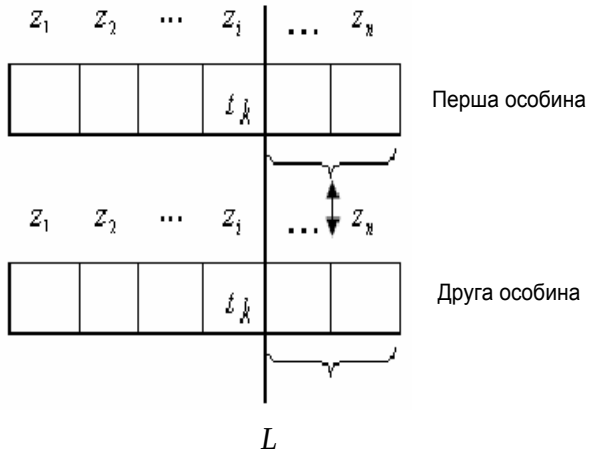


Рис. 3. Схема схрещування двох особин

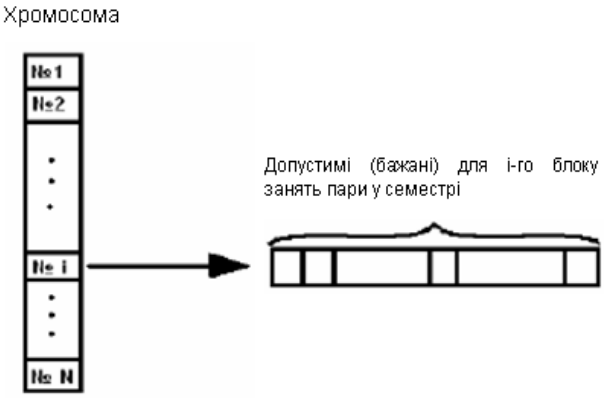


Рис. 4. Схема мутації особин

Отже, генетичний алгоритм складання розкладу занять включає наступні кроки.

1. Формування множини блоків занять.
2. Формування множини часових інтервалів.
3. Формування матриці переваг викладачів.
4. Генерація початкової популяції хромосом.

4.1. Генерація екземпляру хромосоми.

Генерація хромосоми – це процес співставлення блоків занять, що попередньо визначені, з часовими інтервалами. Тобто для кожної зв'язки «викладач-дисципліна-група» призначається відповідний часовий інтервал у відповідний день відповідного тижня. Цей процес проходить випадково для кожного блоку.

4.2. Перевірка згенерованої хромосоми на допустимість

Тобто визначається, чи існують ситуації типу:

– «одна група одночасно знаходиться на заняттях у різних викладачів»;

– «викладач одночасно проводить аудиторне та дистанційне заняття»;

– «сформовано заняття у неприйнятний для викладача час»;

– «один викладач одночасно веде пару у різних групах».

4.3. Якщо таких ситуацій не існує, то маємо допустимий екземпляр розкладу і додаємо його до початкової популяції.

4.4. Перехід до п. 4.1.

5. Визначення оцінки (штрафу) хромосом у початковій популяції.

Оцінка формується виходячи з того, наскільки хромосома не задовольняє умовам, що накладаються. Цими умовами можуть бути переваги кожного викладача, рівномірність розподілу занять протягом тижня, відсутність «вікон» у розкладі, відсутність дисциплін однакового типу, що викладаються однієї групі підряд протягом дня.

6. Вибір найкращої хромосоми з популяції (з меншою оцінкою), а також вибір ще однієї з 10 кращих.

7. Виконання операції кросингування для обраних двох хромосом.

Застосування кросоверу – це обмін ділянками хромосом між хромосомами. Тобто, випадковим чином вибирається точка поділу хромосом та проходить обмін.

8. Перевірка на допустимість отриманих врезультаті кросингування хромосом.

9. Якщо отримано допустимі хромосоми, оцінюємо їх.

9.1. Якщо отримані хромосоми кращі існуючих, додаємо їх у популяцію замість найгірших.

9.2. Якщо ж отримані екземпляри не кращі, тоді:

9.2.1. Виконується мутація.

Мутація – це внесення випадкових змін у хромосому. Випадковим чином обирається елемент хромосоми, що відповідає за співвідношення «тиждень – день - година» і змінюється у відповідності з перевагами викладачів. Тобто замість випадково обраної зв'язки «тиждень - день - година» на початковому етапі генерації хромосоми, ставиться зв'язка, що найкращим чином враховує побажання відповідного викладача.

9.2.2. Перевірка на допустимість: якщо отримали допустимі хромосоми, оцінюємо їх.

9.2.3. Якщо отримані хромосоми кращі існуючих, додаємо їх у популяцію замість найгірших.

10. Повторення пп. 6-9.

11. У будь-який момент можна припинити виконання алгоритму. Результатом роботи буде хромосома з найменшою оцінкою (штрафом).

Висновки

У поданому дослідженні були розглянуті питання створення підсистеми складання розкладу для системи дистанційного навчання «Віртуальний Університет» для організації найбільш зручного та ефективного навчального процесу. Розглянуто умови функціонування та вимоги складання розкладу для дистанційної форми навчання. Проведено аналіз існуючих методів розв'язання задачі складання розкладу, виявлені їх позитивні та негативні сторони, та обґрунтовано метод генетичного алгоритму. Запропонований алгоритм, реалізований мовою програмування PHP, враховує особливості системи дистанційного навчання «Віртуальний Університет», а саме: побажання викладачів стосовно часу проведення їх занять; часові інтервали, у які викладачі не можуть проводити заняття; обмеження на відсутність накладок усіх типів (для викладачів, навчальних груп, аудиторних і дистанційних занять викладачів); необхідність проведення всіх запланованих на семестр і передбачених навчальним планом занять.

Результати роботи алгоритму зберігаються у формі, придатній для подальшого оброблення. Керування параметрами роботи алгоритму винесені у конфігураційні файли, що дає можливість ефективнішого використання та внесення оптимізаційних змін у його роботу.

Закладена можливість подальшого удосконалення алгоритму та розширення його функціональності.

Список літератури

1. Логоша Б. А., Петропавловская А. В. Комплекс моделей и методов оптимизации расписания занятий в вузе // Экономика и математические методы, Т. 29, 1993, № 4.
2. Уоссермен Ф. Нейрокомпьютерная техника // Теория и практика. М. Мир. 1992.
3. Клемент Р. Генетические алгоритмы: почему они работают? Когда их применять? // Компьютерра. – 1999. – № 11.
4. Безгинов А. Н. Обзор существующих методов составления расписаний / Безгинов А. Н., Трегубов С. Ю. // Информационные технологии и программирование. Межвузовский сборник статей, М. – 2005. – Выпуск 2 (14).
5. Низамова Г.Ф. Автореферат диссертации «Математическое и программное обеспечение составления расписания учебных занятий на основе агрегативных генетических алгоритмов» // Уфа – 2006.

УДК 004.42

ЛАКТІОНОВ Є. Ю.
МАЛИЦЬКИЙ А. Ю.

ВИКОРИСТАННЯ XML ДЛЯ ПІДГОТОВКИ СПЕЦІАЛІЗОВАНОЇ ПРОГРАМНОЇ ДОКУМЕНТАЦІЇ

У даній статті розглядаються питання створення спеціалізованої документації для програмного забезпечення. Розроблені інструментальні засоби базуються на широко поширених технологіях XML і DocBook.

The paper discusses the creation of specialized documentation for the software. The developed tools are based on widely accepted technologies, XML and DocBook.

Вступ

Досить часто програмна документація має специфічні особливості. Наприклад, вико ристовуються спеціальні для даної предметної області шаблони опису, або є суттєві обмеження технологічного процесу підготовки документтації.

У даній статті розглядаються різні системи документування програмного забезпечення і робиться вибір відповідного інструментарію. Також робиться вивід про необхідність доопрацювання готової системи документтування і описуються розроблені для конкретного випадку технологія і програмні засоби, що дозволяють вирішити поставлену задачу більш економічним чином.

Постановка задачі

У досить великий за обсягом комплекс програмного забезпечення входить підсистема «швидких звітів», побудованих на основі SQL-запитів. Підсистема повинна мати програмну документацію. Кількість звітів – досить велика, декілька сотень, готувати і, тим більше, супроводжувати опис звітів традиційними методами достатньо складно і дорого.

Треба вибрати найбільш придатний інструментарій для документування програмної підсистеми і, у разі потреби, внести необхідні доробки у технологію та програмні засоби.

Принцип єдиного джерела

Складання програмної документації - дуже важлива складова розробки програмного продукту [1 - 2]. Стандарт ISO/IEC 12207, що визначає процеси життєвого циклу програмного забезпечення, навіть передбачає спеціальний процес, присвячений вказаному питанню. При цьому на кожен програмний продукт повинна розроблятися документація двох типів: для користувачів різних груп і для розробників. Відсутність документації будь-якого типу для конкретного програмного продукту є недопустимою.

Модульний підхід укорінився в програмуванні багато років тому. Ніхто не набирає один і той же програмний код багато разів, а зловживання командами текстового редактора «Copy» і «Paste» програмісти, поза сумнівом, визнають грубою роботою. Якщо програма в різних ситуаціях повинна виконувати одні і ті ж дії, їх оформляють у вигляді функцій або процедур. Якщо дії однакові загалом, але розрізняються залежно від конкретної ситуації, то програміст передбачає можливість передати функції або процедурі певні аргументи.

У технічній документації повтор – явище не менш рідкісне, ніж у програмі. Доводиться описувати схожі функції програм і систем, схожі екранні форми, схожі дії користувачів. Описи одних і тих же об'єктів доводиться повністю або частково дублювати в різних документах. Оскільки матеріал, з якого складається технічна документація, – текст на природній мові, застосувати до нього модульний підхід виявляється складнішим, ніж до програмного коду. У тексті документації досить особливостей, обумовлених лексичними і граматичними виклю-

ченнями, традиціями, що склалися, міркуваннями стилістики і естетики. Багато розробників документації роблять те, що було б ганьбою для будь-якого програміста: «копипейстят» власний текст і при необхідності уручну правлять його, пристосовувавши до чергового контексту.

Принцип єдиного джерела і технології, що реалізовують його, дозволяють застосувати модульний принцип до документування [3]. Завдяки цьому принципу ми можемо не дивлячись на всю своєрідність тексту «виносити за дужки» однакове, параметризувати майже однакове і формувати з цих блоків документи в тому вигляді, в якому їх вважає за краще отримувати їх аудиторія.

Застосування технології єдиного джерела вимагає підготовки. Спочатку проектується структура єдиного джерела, розробляються шаблони і стилі оформлення, встановлюється і настроюється інструментарій для формування документів. Ця стадія віднімає час на початку проекту і пред'являє досить високі вимоги до кваліфікації розробника технічної документації. Але потім починається рутинна робота по написанню тексту і його завантаженню в єдине джерело. У будь-який момент на основі введеного тексту можна сформувати документи, з більшою чи меншою мірою готовності. На цій стадії ми отримуємо віддачу від зроблених вкладень. Розділ, малюнок, таблиця, абзац, будь-який інший фрагмент, який повинен з'являтися в декількох місцях, при необхідності достатньо виправити одноразово в єдиному джерелі. Чудово, що це стосується не тільки тексту, але і структурних рішень. Одноразового внесення змін до шаблону, припустимо, керівництва користувача, достатньо, щоб всі документи цього типу після чергової автоматичної обробки були змінені потрібним чином.

Принцип єдиного джерела в документуванні, як і модульний принцип в програмуванні, допомагає організувати роботу колективу співавторів, розподіливши між ними більш-менш ізольовані підзадачі. Таким чином, єдине джерело – це не тільки технічне, але ще і організаційне рішення. Взаємодія між учасниками розробки технічної документації теж може бути автоматизованою. Крупні інтегровані середовища для автоматизації документування, такі, як AUTHORIT, SiberSafe, ROBOHELP надають для цього різні засоби: управління правами доступу, контроль версій, планування робіт і тому подібне

Існуючі технології підготовки програмної документації

Залежно від цілей і об'ємів документації, що готується, розробниками створюються різні технологічні ланцюжки з використанням наступних найбільш популярних програмних продуктів [4 - 6]:

- *Microsoft Word – PDF*

Текст пишеться з використанням текстового редактора Microsoft Word і потім конвертується у формат PDF. Технологічний ланцюжок дозволяє готувати паперові публікації, а також комплекти електронної документації у форматі PDF.

Використовується в тих випадках, коли вимоги до видавничої якості документації невисокі. Перевагами є легкість впровадження і можливість рецензування і редагування матеріалів всіма, хто має до них доступ. Недоліком є невисока якість верстки.

- *Adobe FrameMaker – PDF*

Текст верстається з використанням програмного комплексу Adobe FrameMaker і потім конвертується у формат PDF. Технологічний ланцюжок дозволяє готувати паперові публікації, а також комплекти електронної документації у форматі PDF. Можливо також перетворення в інші формати (RTF, HTML, різні застосування XML і т. д.).

Використовується для підготовки публікацій професійної видавничої якості. До недоліків відноситься необхідність витрат на впровадження і неможливість редагування матеріалів «на льоту».

- *Технології підготовки контекстної довідки*

Технологічні ланцюжки цього типу використовуються для створення контекстної довідки в одному з поширених форматів: WinHelp, HTML Help і так далі Ланцюжок може включати спеціалізовану програмну систему (ForeHelp, eHelp, RoboHelp) або

обходитися без неї.

Технології підготовки контекстної довідки вимагають витрат на впровадження і узгодження дій між технічним письменником і командою розробників.

- *Технологія єдиного джерела на основі DocBook*

Текст технічної документації спочатку пишеться на мові розмітки DocBook і використовується як єдине джерело. Два паралельні технологічні ланцюжки дозволяють отримати файл у форматі PDF (який надалі розповсюджується в електронному вигляді або служить основою для паперової публікації) або файл у форматі HTML Help.

Технологія єдиного джерела на основі DocBook є оптимальним рішенням в тих випадках, коли необхідно скласти і підтримувати технічну документацію як у формі паперових (або електронних) документів, так і у формі системи контекстної довідки. При цьому всі зміни вносяться до єдиного джерела, вихідні ж файли будь-якого формату при необхідності генеруються з єдиного джерела. Таким чином, їх вміст завжди відповідає актуальному стану початкового тексту, написаного на DocBook.

Технологія потребує значних витрат зусиль на впровадження.

- *Технологія DITA*

Основний постулат технології DITA полягає в тому, що текст технічної документації складається з фрагментів (так званих топиків), що типізуються, причому кожному типу відповідає певний план опису. При цьому технологія DITA дозволяє авторові декларувати нові типи топиків для потреб конкретного проекту і задавати для кожного типу план опису.

У комплекті інструментарію DITA вже є готові стилі, що реалізовують усі найбільш важливі функції формування документів приблизно на тому ж рівні, на якому це роблять стилі DocBook. Технологія DITA дозволяє, з одного боку, створювати нові типи топиків, а з іншою, використовувати для їх обробки вже наявні стилі. Це досягається завдяки прийому, який в об'єктно-орієнтованому програмуванні відомий як успадкування.

Технологія DITA є більш складною у порівнянні з DocBook і потребує ще більш витрат на впровадження.

Зі всіх розглянутих технологій найбільш прийнятною для початкового впровадження принципу єдиного джерела здається технологія DocBook. Вона є у достатній мірі гнучкою та потужною і водночас не такою складною, як DITA. До того ж DocBook є відкритою системою (Open Source проект).

Технологія DocBook/XML

Основою технологічної платформи DocBook/XML служить однойменна проблемно-орієнтована мова розмітки [7]. Вона призначена для запису тексту технічної документації на програми, алгоритмічні мови, комп'ютерне устаткування і інші рішення в області інформаційних технологій, чим принципово відрізняється від більшості форматів зберігання текстових даних (але не XML-мов!). У мові DocBook/XML передбачені засоби опису фрагментів, властивих технічній документації, наприклад, спеціальними елементами належить виділяти назви елементів інтерфейсу, позначення клавіш, назви змінних, терміни, різні врізання (зауваження, підказки, попередження), лістинги, описи виконуваних користувачем процедур. Розмітка, що задається мовою DocBook/XML, носить переважно функціональний характер: автор указує роль, яку той або інший фрагмент відіграє в тексті, а не спосіб його зовнішнього оформлення. Такий підхід сковує автора, зате дозволяє добитися явної незалежності змісту і оформлення вихідного документа та уніфікувати деякі важливі якості стилю викладу при роботі декількох авторів в одному проекті.

Для набору тексту вхідного документа можуть застосовуватися різні програми, від звичайного «Блокнота» до розвинених XML-редакторів. Оскільки вимоги до формату вхідного документа визначаються специфікаціями використовуваних мов розмітки, вибір редактора перестає бути важливим питанням, яке необхідно вирішити на рівні проекту. Кожен автор може працювати в тому редакторові, який йому зручний.

По відношенню до завдання набору тексту у форматі DocBook/XML редактори, що існують сьогодні, можна розділити на наступні групи:

– звичайні текстові редактори;

– XML-редактори з інтерфейсом текстового процесора;
– XML-редактори, що спрощують набір розмітки.

Вимоги до оформлення вихідних документів в різних проектах істотно різні, тому спроби випустити вичерпний набір XSLT-стилів приречені на невдачу. Створювати XSLT-стилі наново в кожному проекті теж не можна, тому що терміни і вартість цієї роботи вийдуть неприйнятними. На практиці зазвичай застосовується вільно поширюваний комплект стандартних стилів DocBook XSL [7]. Він підтримує основні елементи макетів і цільові формати, а також добре пристосований до доопрацювання для потреб конкретного проекту. Завдяки архітектурі XSLT адаптація стандартних стилів не вимагає модифікації їх коду. Нові правила, що розташовуються в окремих файлах, доповнюють або заміщають стандартні. Виконані доопрацювання не прив'язані до конкретної копії стандартних стилів, отже:

– доопрацювання не конфліктують один з одним, і ними легко обмінюватися;
– можливість оновлення версій DocBook XSL обмежена тільки їх зворотною сумісністю.

DocBook успішно використовується при розробці і підтримці технічної документації, особливо коли мова йде про об'єми в тисячі і десятки тисяч сторінок. Технологія дозволяє вирішити ряд типових завдань, що виникають при розробці технічної документації, таких як:

– розробка і оформлення документації в суворій відповідності з галузевими і державними стандартами (ЕСКД, ЕСПД, КСАС);
– автоматизація процедур нормоконтролю документації, що готується;
– підготовка документації в різних електронних форматах, призначених для друку і перегляду на екрані монітора: PDF, HTML, HTML Help, Java Help, Unix Man Pages;
– підтримка багатокористувачевого режиму підготовки документації.

Технологія DocBook чудово підходить підприємствам, чия продукція супроводиться технічною документацією, підготовленою та оформленою відповідно до вимог ДСТ 2, 19 і 34 серії (ЕСКД, ЕСПД, КСАС).

Проте, навіть така розвинена технологія, як DocBook, потребує деякого доопрацювання при конкретному її застосуванні.

Підготовка спеціалізованої програмної документації

Великий за обсягом програмний комплекс «Лабораторні інформаційні системи» призначений для ведення інформаційних систем для обслуговування лікарень і медичних лабораторій.

Однією з підсистем цього комплексу є підсистема «Швидкі звіти» SQL, яка служить для видачі даних про таблиці-довідники системи на підставі SQL запитів.

Підсистема «Швидкі звіти» SQL повинна мати повноцінну програмну документацію, яка включає розділи керівництва користувача і керівництва програміста (див. рис. 1).

До складу програмної документації також входить екранна допомога користувачеві звітів.

Підготовка програмної документації для підсистеми «Швидкі звіти» SQL має ряд важливих особливостей:

– SQL запити можуть достатньо часто змінюватися під час експлуатації системи, разом з цим також змінюється зовнішній вид звітів, і, отже, повинна змінюватися документація;
– програмна документація повинна підтримуватися у всіх вказаних вище видах – для кінцевого користувача, для програміста, екранна допомога;
– документація для користувача має бути представлена в декількох форматах – у форматі набору HTML-сторінок і форматі PDF-файлу ;
– програмна документація у вигляді екранної допомоги реалізується за допомогою спеціальної програмної системи і тому вихідна інформація для неї має бути представлена у вигляді XML-файлів спеціального формату;
– під час експлуатації системи можуть з'являтися нові звіти SQL і видалятися існуючі.

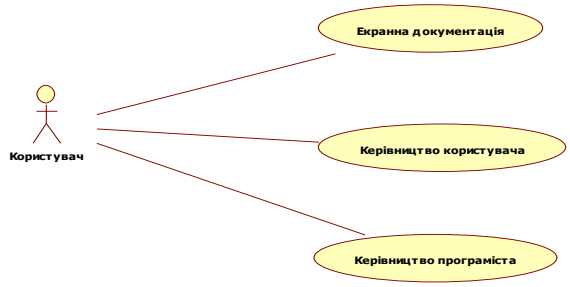


Рис. 1. Діаграма використання документації підсистеми

Всі ці чинники приводять до того, що система підготовки програмної документації, а особливо – система супроводу програмної документації мають бути дуже динамічними і повинні дозволяти швидку підготовку різних форматів документації з одного джерела.

Як основна система документування, на підставі проведеного аналізу, вибрана система документування DocBook (реалізація для Windows).

Перше рішення було таким: підготувати вихідну документацію у форматі DocBook і засобами DocBook отримати результуючу документацію у форматі HTML (у вигляді ієрархічної структури сторінок) і у форматі PDF.

Для отримання необхідної документації екранної допомоги у вигляді XML-файлу, структура якого визначена зовнішньою системою підготовки екранної допомоги, було вирішено розробити скрипт конвертації засобами XSLT (мова перетворення XML-документів). Схема підготовки документації для підсистеми «Швидкі звіти» SQL придбала вигляд, зображений на рис. 2.

Проте, підготовка описів у форматі DocBook для великого числа звітів виявилася вельми трудомісткою (через гнучкість, об'ємність і складність цього формату), і тоді було вирішено готувати початкові описи за значно простішим шаблоном «XML в спеціальному форматі» (для системи допомоги). Модифікована схема підготовки документації приведена на рис. 3.

Така невелика зміна напряму потоку даних в робочому процесі привела зрештою до економії багатьох годин роботи операторів (програмістів) по підготовці вихідних описів звітів.

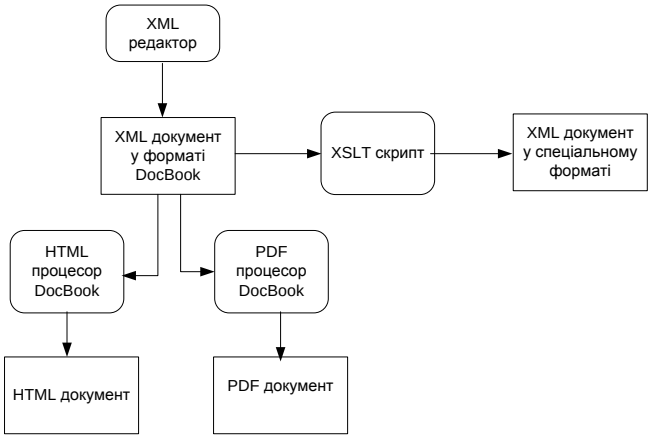


Рис. 2. Схема підготовки документації (варіант 1)

Такий процес підготовки документації є еквівалентним добре відомому в програмуванні процесу компілящії і збірки великого програмного продукту.

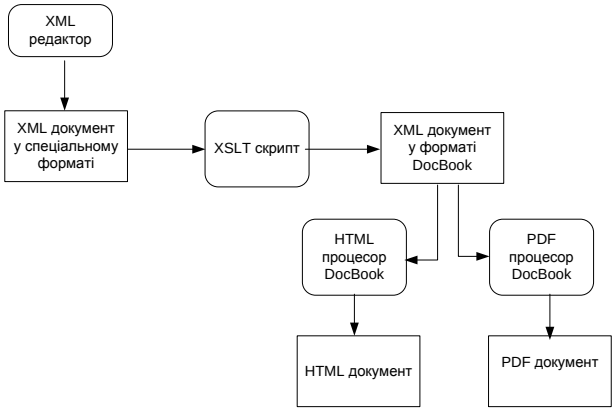


Рис. 3. Схема підготовки документації (варіант 2)

Висновки

В процесі дослідження розглянуто різні системи документування програмного за безпечення та обґрунтовано вибір технології DocBook. Наведений приклад доопрацювання готової системи документування в конкретному випадку завдання підготовки спеціалізованої програмної документації (системи видачі програмних звітів) вирішене більш економічним чином.

Перелік посилань

1. Липаев В.В. Документирование и управление конфигурацией программных средств / В.В. Липаев. — М.: СИНТЕГ, 1998. — 220 с.
2. Липаев В.В. Качество программных средств / В.В. Липаев. — М: Янус-К, 2002. 400 с.
3. Технология DocBook и принцип единого источника [Электронный ресурс]. Режим доступа к ресурсу: http://glossterm.org/index.php
4. Белайчук А. Кратчайший путь к DocBook [Электронный ресурс]. Режим доступа к ресурсу: http://docbook.ru/doc/sw/
5. Технический писатель. Базовые компетенции специалиста [Электронный ресурс]. Режим доступа к ресурсу: http://philosoft.ru/twskills.zhtml
6. Технология DITA. Обзор возможностей и основные преимущества [Электронный ресурс]. Режим доступа к ресурсу: http://www.philosoft.ru/dita-intro.zhtml
7. Н.Уолш. DocBook: The Definitive Guide [Электронный ресурс]. Режим доступа к ресурсу: http://www.docbook.org/tdg/en/html/docbook.html

УДК 004.054

ВЕЛЬМА А.М.
ЛАКТІОНОВ Є.Ю.

ВИБІР СИСТЕМИ ВІДСТЕЖЕННЯ ПОМИЛОК В ЗАЛЕЖНОСТІ ВІД КОНФІГУРАЦІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

В статті представлений огляд деяких систем відстеження помилок, їх порівняння і рекомендації до вибору. На сьогоднішній день спостерігається перехід від продуктивності до якості в основних вимогах до програмного забезпечення. Тому компаніям необхідно організувати свої бізнес-процеси так, щоб відповідати цим вимогам. Актуальність даної статті полягає в тому, що компаніям-розробникам доцільно використовувати системи відстеження помилок для забезпечення ефективності в їх виявленні і виправленні.

The paper presents the review of some tracking systems, their comparison and recommendations to choice. Today there is a transition from the productivity to quality in the basic requirements to software. Therefore companies must organize the business process to conform to these requirements. Actuality of this article consists of that developing companies are expedient to utillize the tracking systems for providing of efficiency in their discovery and correction

1. Вступ

У даній статті розглядаються різні системи відстеження помилок в програмному за безпеченні і виконується аналіз критеріїв вибору такої системи залежно від конфігурації програмного забезпечення, що розробляється.

2. Постановка задачі

Існує багато систем відстежування помилок у досить широкому діапазоні функціональльності і вартості. Як критерій вибору відповідної системи для конкретної розробки можна вибрати оцінку надійності програмної системи, що розробляється, керуючись її мірою складності по Холстеду.

3. Місце систем відстеження помилок у розробці програмних засобів

Методи тестування ПЗ, що існують на сьогоднішній день, не дозволяють однозначно і повністю виявити всі дефекти і встановити коректність функціонування аналізованої програми, тому всі існуючі методи тестування діють в рамках формального процесу перевірки ПЗ, що досліджується або розробляється.

Такий процес формальної перевірки або верифікації може довести, що дефекти відсутні з погляду використовуваного методу (тобто немає ніякої можливості точно встановити або гарантувати відсутність дефектів в програмному продукті з урахуванням людського чинника, присутнього на всіх етапах життєвого циклу ПЗ).

Якість програмного продукту характеризується набором властивостей, що визначають, наскільки продукт «гарний» з погляду зацікавлених сторін, таких як замовник продукту, спонсор, кінцевий користувач, розробники і тестувальники продукту, інженери підтримки, співробітники відділів маркетингу, навчання і продажу [1]. Кожен з учасників може мати різне уявлення про продукт і про те, наскільки він гарний або поганий, тобто про те, наскільки висока якість продукту. Таким чином, постановка завдання забезпечення якості продукту виливається в завдання визначення зацікавлених осіб, їх критеріїв якості і потім знаходження оптимального рішення, що задовольняє цим критеріям. Тестування є одним з найбільш сталих способів забезпечення якості розробки програмного забезпечення і входить в набір ефективних засобів сучасної системи за безпечення якості програмного продукту.

З технічної точки зору тестування полягає у виконанні застосування на деякій множині початкових даних і звірці отримуваних результатів із заздалегідь відомими (еталонними) з метою встановити відповідність різ-

них властивостей і характеристик застосування замовленим властивостям. Як одна з основних фаз процесу розробки програмного продукту (Дизайн додатку – Розробка коду – Тестування), тестування характеризується достатньо великим внеском в сумарну трудомісткість розробки продукту. Широко відома оцінка розподілу трудомісткості між фазами створення програмного продукту: 40% – 20% – 40% (рис. 1) з чого виходить, що найбільший ефект в зниженні трудомісткості може бути отриманий перш за все на фазах дизайну і тестування.

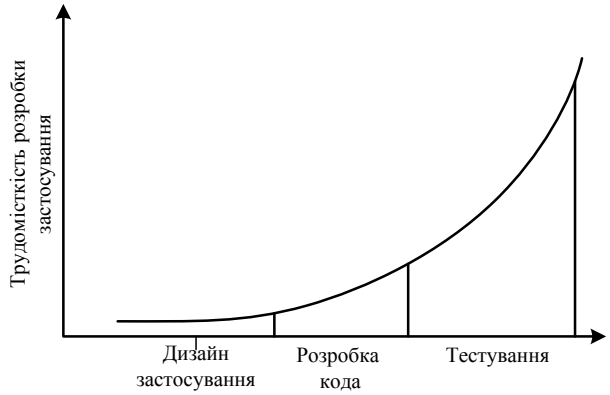


Рис. 1. Розподіл трудомісткості розробки застосування

Для скріплення роботи програміста і тестувальника при розробці програмного забезпечення, як великі, так і маленькі софтверні компанії використовують системи обліку завдань, помилок, управління проектами (bug tracker, issue tracking system, project management application).

На даний момент такого роду продуктів існує немало. Є прості системи, функціонал яких обмежується обліком помилок і відстеженням їх статусу. Є складніші, які дозволяють, наприклад, будувати різні графіки з проектних ризиків, інтегруватися з системами контролю версій, здійснювати складний пошук у проектній документації і так далі.

В ідеалі система управління проектами - це деяке серверне застосування, яке дозволяє робити наступне:

- в реальному часі відстежувати поточний стан проектів, збирати статистику проектів;
- вести облік помилок, завдань, поліпшень відповідно до заданого життєвого циклу;
- зберігати проектну документацію (мати вбудовану базу знань, або Wiki);
- конфігурувати права доступу користувачів, їх ролі, відправку нотифікацій;
- інтегруватися з різними third-party продуктами (наприклад, з тими ж системами контролю версій);
- доступ до функцій системи програмним способом (через відповідний API).

Принцип використання системи відстеження помилок зображений на рис. 2.

Також можливі і інші способи вико ристання, наприклад без web-сервера система відстеження помилок працюватиме як standalone сервер.

Слід відмітити, що автоматизована система відстеження проблем, перш за все, вирішує зовсім не технічні, а політичні питання. Це могутній організаційний засіб з великими можливостями.

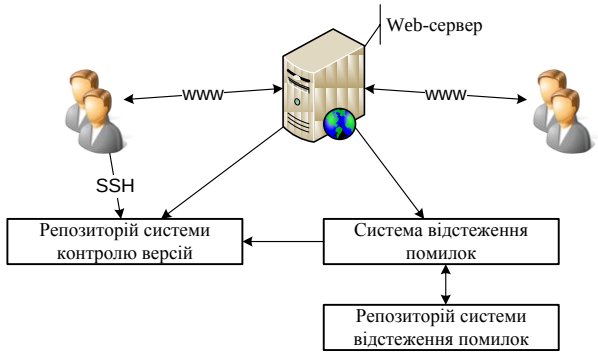


Рис. 2. Використання системи відстеження помилок і інтеграція з системою контролю версій

Система є засобом відстеження ходу робіт. Інформація, яка традиційно була в розпорядженні тільки у керівництва і декількох програмістів, стає доступною широкому кругу співробітників різних рівнів. Система надає своїм користувачам об'єктивну і незалежну оцінку ходу виконання робота і їх відповідності графіку. У будь-який момент можна проглянути повний список завдань, які ще мають бути виконані. Поточний стан продукту і його якість завжди відомі. І кожен може побачити, як просуваються роботи і наскільки швидко вирішуються поставлені завдання.

Система також є засобом організації взаємодії між співробітниками. У кожній компанії, так або інакше, вирішується цілий ряд організаційних питань. Коли процес їх рішення автоматизований, дії співробітників стають більш регламентованими і впорядкованими. Більш того, система висвічує деякі внутрішні проблеми компанії, що раніше

залишилися прихованими. Гарна система, особливо мережева, дозволяє практично повністю відстежувати взаємодію між співробітниками, що беруть участь у виявленні недоліків програмного продукту, щоб вчасно вирішувати спірні питання і приймати заходи по оптимізації робіт. Зокрема, з її допомогою можна зафіксувати випадки некоректної поведінки окремих співробітників або груп, зловживання чужим робочим часом, необґрунтованих претензій і тому подібне

Система відображає продуктивність роботи кожного співробітника. З бази даних неважко отримати будь-яку статистичну інформацію, наприклад, про кількість звітів, що здаються тестувальником за день, середній кількості помилок що допускаються програмістом за тиждень. Такі дані можуть бути корисні для аналізу ходу розробки і вирішення поточних проблем.

4. Існуючі системи відстеження помилок і їх властивості

Розглянемо деякі системи відстеження помилок [2 – 3].

Bugzilla – застосування для відстеження помилок, розроблене компанією Mozilla. Фактично, Bugzilla використовується в багатьох корпораціях для розробки власного програмного забезпечення для корпоративних потреб.

За минулий час Bugzilla став зрілим продуктом, що по праву гордиться розвиненою функціональністю, куди, зокрема, входять:

- інтегрована система безпеки з можливістю визначення безпеки на рівні продуктів;
- система залежностей помилок і виведення залежностей помилок в графічному вигляді;
- розвинена система складання звітів;
- стабільний, перевірений часом back-end на основі RDBMS;
- розвинена система конфігурації;
- дуже зрозумілий і добре продуманий протокол виправлення помилок
- API для електронної пошти, XML, HTTP і консолі;
- доступна інтеграція з системами управління автоматичної конфігурації програмного забезпечення, зокрема Perforce and CVS (через інтерфейс електронної пошти Bugzilla і скрипти для checkin/checkout).

Trac – це досить проста система відстеження помилок, що проте помітно спрощує життя розробника при належному терпінні. Основою для Trac'a є SVN репозиторій. Тому найбільш функціональним рішенням буде робота в зв'язці Subversion – Trac.

Основними функціями цієї системи є:

- управління проектом
  - а) розділення проекту на етапи (milestones);
  - б) контроль виконання (roadmap);
  - в) всі зміни за проектом заносяться на часову шкалу (timeline);
  - г) підтримка RSS;
- tickets – облік помилок, зауважень, побажань з можливістю фільтрації і занесення відповідно в milestone, roadmap;
- перегляд репозиторія – достатньо зручний модуль перегляду Subversion репозиторія проекту. Дозволяє переглядати вихідний код з урахуванням ревізії, а також змін;
- управління користувачами. Проста система – вказується, що можуть робити користувачі, а що ні;
- Wiki – в Trac вбудована система wiki з можливістю робити посилання на milestone, roadmap, ticket. Органічно вписується і зручна у використанні при веденні проекту.

Redmine – відкрите серверне застосування для управління проектами і відстеження помилок. Redmine написаний на Ruby і є застосуванням на основі широко відомого веб-фреймворка Ruby on Rails.

Даний продукт надає наступні можливості:

- ведення декількох проектів;
- гнучка система доступу, що базується на ролях;
- система відстеження помилок;
- діаграми Ганта і календар;
- ведення новин проекту, документів і управління файлами;
- сповіщення про зміни за допомогою RSS-потоків і електронної пошти;
- вики для кожного проекту;
- форуми для кожного проекту;

– облік витрат часу;
– довільні поля, що настроюються, для інцидентів, витрат часу, проектів і користувачів;
– легка інтеграція з системами управління версіями (SVN, CVS, Git, Mercurial, Bazaar і Darcs);
– створення записів про помилки на основі одержаних листів;
– підтримка множинної аутентифікації LDAP;
– можливість самостійної реєстрації нових користувачів;
– багатомовний інтерфейс (зокрема російський);
– підтримка СУБД MYSQL, POSTGRESQL, SQLite, Oracle.

Порівняння представлених систем відстеження помилок представлене у табл. 1.

Також слід відмітити, що функціональність даних систем можна розширити або змінити за допомогою доповнень (plugins). Також, оскільки дані системи відстеження помилок є проектами з відкритим кодом, можливо самостійно додати необхідну функціональність.

Розглянемо випадки, коли необхідно використовувати системи відстеження помилок.

Можна представити ситуацію, коли в отриманому продукті кількість передбачуваних помилок не перевищуватиме певного показника (наприклад, для програми в 1000 рядків можна припустити, що кількість помилок не перевищить 20). У такому разі на установку і налаштування системи відстеження помилок може витратитися більше ресурсів і часу, ніж на виправлення отриманих помилок. Тобто можна вважати, що використання системи відстеження помилок є недоцільним.

Але як можна оцінити кількість помилок в програмному забезпеченні? На жаль, поки немає загальної моделі оцінювання надійності ПЗ, яка дає точні оцінки надійності довільного програмного забезпечення. Тому в світовій практиці використовується декілька десятків різних моделей оцінки надійності. Кожна з них дає більш-менш прийнятні результати, всі мають своє найбільш раціональні сфери застосування.

Табл. 1. Критерії вибору систем відстеження помилок

Критерій	**Bugzilla**	**Trac**	**Redmine**
Ліцензія	Mozilla Public License	BSD variant	GPL v.2
Мова реалізації	Perl	Python	Ruby on Rails
Призначений для користувача інтерфейс	Web, e-mail, RSS, Web service, command-line	Web	Web, E-mail, Atom
Інтеграція з системами управління версіями	CVS, Subversion, Perforce, AccuRev	Subversion, Darcs, Bazaar, Mercurial, Perforce, Git	Subversion, CVS, Bazaar, Darcs, Mercurial, Git
Інтеграція або генерація динамічної документації	Інтегровані звіти і діаграми, планування розкладу звітів	Інтегрована wiki	Інтегрована wiki, дискусійні форуми, новинні блоги, email інтеграція, календарі
Інтеграція з плануванням тестів	Testopia	–	–
Workflow, що настроюється	Так	Так	Так
Підтримка юнікода	Так	Так	Так
LDAP авторизація	Так	–	Так

Математичні моделі по-різному підходять до оцінки надійності ПЗ, використовуючи для цього різні характеристики [4 – 5]:

– інтервали часу між відмовами (моделі Джелінського-Моранді, Шика-Волвертона, геометричні);
– внесені штучно до ПЗ помилки (модель Міллса);
– простір вхідних даних (модель Нельсона);
– складність ПЗ (модель Холстеда).

Скористаємося мірою складності Холстеда. Експерименти показують, що кількість помилок в непротестованих програмах пропорційна:

$$E_n = K \cdot E^{2/3}, \quad K = \frac{1}{3200},$$

де E_n – міра складності Холстеда, і визначається за формулою:

$$E = \frac{n_1 N_2 N \log_2 n}{2n_2},$$

де n_1 – кількість різних операторів;

n_2 – кількість різних операндів;

$n = n_1 + n_2$ – словник програми;

N_1 – кількість появ операторів;

N_2 – кількість появ операндів;

$N = N_1 + N_2$ – довжина програми.

Розрахуємо кількість помилок для малої програми (близько 1000 рядків), середньої (близько 100000 рядків) і великої програми (більше 1 млн. рядків).

Для малої програми $N_1 = 1000$, $N_2 = 2000$, $n_1 = 200$, $n_2 = 100$ – оцінна кількість складатиме 37 помилок.

Для середньої програми кількість помилок зросте до 2500 і для великої – ця кількість перевищить 20 тис. помилок.

Розглянувши ці випадки, можна зробити вивід, що людських здібностей може вистачити на роботу з помилками у разі малої програми, але у випадку з великими і середніми програмами вкрай необхідне впровадження і супровід системи відстеження помилок.

5. Висновок

Багато компаній знаходять, що інтегрована система відстеження помилок зменшує час простою, збільшує продуктивність і підвищує задоволеність клієнта від роботи з їх системами. Разом з повним розкриттям інформації, відкрита система відстеження помилок дозволяє виробникам програмного забезпечення підтримувати контакт зі своїми клієнтами і реселерами, для передачі повідомлень про помилки по всьому ланцюгу управління даними. Багато корпорацій також виявляють, що відстеження помилок зменшує витрати, забезпечуючи службі підтримки IT систему обліку, телефонній підтримці - бази знань, а всім - добре зрозумілу систему для обліку проблем з програмним забезпеченням.

Система відстеження помилок може бути легко адаптована до різних ситуацій. У зв'язці з такими системами як CVS або Subversion, система відстеження помилок забезпечує могутнє, легке у використанні рішення для управління конфігурацією і проблемами, що виникають при реплікації.

Вибір системи відстеження помилок необхідно робити, спираючись на урахування наступних особливостей системи:

- функції, які необхідні в роботі людини або організації (слід також враховувати розширюваність за допомогою додаткових модулів);
- зручність у використанні і обслуговуванні;
- швидкість роботи в конкретній конфігурації сервера і системи відстеження помилок.

Найбільш вірогідна ситуація, коли необхідно випробувати конкретну систему на тестовому комп'ютері, для затвердження цієї системи відстеження помилок або відмови від неї.

Перелік посилань

1. Канер С. Тестирование программного обеспечения. Фундаментальные концепции менеджмента бизнес-приложений / Канер С., Фолк Д., Нгуен Е.; пер. с англ. – К.: Издательство ДиаСофт, 2001. – 544 с.
2. The Bugzilla Guide (электронный ресурс). Режим доступа к ресурсу: https://bugzilla.mozilla.org/
3. Trac and Sunversion (электронный ресурс). Режим доступа к ресурсу: http://trac.edgewall.org/
4. Черноножкин С.К. Меры сложности программ (обзор). Системная информатика. / Черноножкин С.К. – Новосибирск: Наука, 1997. – Вып. 5: Архитектурные, формальные и программные модели. – 560 с.
5. Лаврищева Е.М. Методы и средства инженерии программного обеспечения / Е.М. Лаврищева, В.А. Петрухин. – М.: МФТИ, 2006. – 304 с.

Відомості про авторів

Аракелян Г.А.	с.н.с. кафедры АСОИУ НТУУ «КПИ»
Ашаєв Ю.М.	студент кафедри ОТ НТУУ «КПІ»
Безносик О.Ю.	ст. викладач каф. СП ІПСА НТУУ «КПІ»
Болдак А.А.	к.т.н, доцент кафедры ВТ НТУУ «КПИ»
Блінова Т.О.	к.т.н., с.н.с. кафедри ОТ НТУУ «КПІ»
Бролінський С.М.	студент кафедри ОТ НТУУ «КПІ»
Бузовский О.В.	д.т.н, профессор кафедры ВТ НТУУ «КПИ»
Вельма А.М.	студент Бердянського університету менеджменту і бізнесу, ФІС та Т
Волокита А. М.	к.т.н., докторант кафедри ОТ НТУУ «КПІ»
Волошин А.В.	студент кафедри АУТС НТУУ «КПІ»
Галушко Д.О.	студент кафедри АУТС НТУУ «КПІ»
Зайченко Е.Ю.	д.т.н., проф. УНК «ИПСА» НТУУ «КПИ»
Зайченко Ю.П.	д.т.н., проф., проф. кафедры ММСА УНК «ИПСА» НТУУ «КПИ»
Зіненко О.І.	студент кафедри ОТ НТУУ «КПІ»
Іванов Д. Г.	магістр, аспірант кафедри ОТ НТУУ «КПІ»
Камінська П.А.	аспірант кафедри АСОІУ НТУУ «КПІ»
Карпінський В.М.	аспірант кафедри ЕЕ ТНТУ ім. І.Пулюя
Кінах Я.І.	к.т.н., старший викладач кафедри КІ ТНЕУ
Костик Д.Ю.	аспирант кафедры АСОИУ НТУУ «КПИ»
Кулаков О. Ю.	к.т.н., асистент кафедри ОТ НТУУ «КПІ»
Лавринчук А.Н.	аспирант кафедры ММСА НТУУ «КПИ» УНК «ИПСА»
Ладогубець В.В.	доцент, к.т.н. каф. СП ІПСА НТУУ «КПІ»
Лактіонов Є.Ю.	доцент, к.т.н. Бердянського університету менеджменту і бізнесу, ФІС та Т
Лепеха В.Л.	н.с. кафедри ОТ НТУУ «КПІ»
Лесик Т.М.	інж. кафедри ОТ НТУУ «КПІ»
Лисецкий Т. Н.	аспирант кафедры АСОИУ НТУУ «КПИ»
Луцький Г.М.	д.т.н. наук, проф., завідувач кафедри ОТ НТУУ «КПІ»
Малицький А.Ю.	студент Бердянського університету менеджменту і бізнесу, ФІС та Т
Михайлов В.В.	к.т.н, с.н.с. кафедры АСОИУ НТУУ «КПИ»
Мельников О. В.	н.с. кафедры АСОИУ НТУУ «КПИ»
Мисюра Е.Б.	к.т.н, с.н.с. кафедры АСОИУ НТУУ «КПИ»
Можаровський П.Ф.	студент кафедри АУТС НТУУ «КПІ»
Мурга Н.А.	аспірант кафедри ММСА ННК «ІПСА »НТУУ «КПІ»
Невдащенко М.В.	ассистент кафедры ВТ НТУУ «КПИ»
Новіков Ю.Л.	аспірант кафедри АСОІУ НТУУ «КПІ»
Павлов О.А.	проф. кафедры АСОИУ НТУУ «КПИ», декан ФИВТ
Писаренко А.В.	ст. викладач кафедри АУТС НТУУ «КПІ»
Покотило О.О.	студент кафедри АУТС НТУУ «КПІ»
Порєв В.М.	к.т.н., ст. викладач кафедри ОТ НТУУ «КПІ»
Роковой А.П.	ассистент кафедры ВТ НТУУ «КПИ»
Ролік О.І.	к.т.н., ст. наук. спів., доцент кафедри АУТС НТУУ «КПІ»
Салапатов В.І.	к.т.н., доцент кафедры кибернетики, ЧНУ
Сергієнко А.М.	к.т.н., с.н.с. кафедри ОТ НТУУ «КПІ»
Стіренко С.Г.	к.т.н, доцент кафедры ВТ НТУУ «КПИ»
Томашевський В.М.	д.т.н., професор кафедри АСОІУ НТУУ «КПІ»
Щербатенко О.В.	к.т.н., с.н.с. кафедры АСОИУ НТУУ «КПИ»
Фіногенов О.Д.	асистент, к.т.н. каф. СП ІПСА НТУУ «КПІ»
Цвєлодуб А.О.	студент кафедри АУТС НТУУ «КПІ»
Євтух П.С.	д.т.н., професор, зав. кафедри ЕЕ ТНТУ ім. І.Пулюя