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

ВІСНИК

НАЦІОНАЛЬНОГО ТЕХНІЧНОГО УНІВЕРСИТЕТА УКРАЇНИ «КПІ»

Інформатика, управління та обчислювальна техніка

Заснований у 1964 р.
Випуск 63

Київ “ВЕК+” 2015

УДК 004

Вісник НТУУ «КПІ». Інформатика, управління та обчислювальна техніка: Зб. наук. пр. – К.:Век+, – 2015. – № 63. – 131 с.

Рекомендований до друку Вченою радою факультету інформатики та обчислювальної техніки, протокол № 9 від 25.05.2015

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

Заст. гол. ред.: *Стіренко С.Г., к.т.н., доц.,*
Відповідальний секретар: *Коган А.В.*

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

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

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

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

Журнал входить до переліку наукових фахових видань МОН України.

ISSN2310-3620

Свідоцтво про державну реєстрацію № 16949-5719 Р від 17.06.2010

Збірник наукових праць українською, англійською та російською мовами
Web-ресурс – http://it-visnyk.kpi.ua

Підп. до друку 25.05.2015. Формат 60×84 1/8. Гарнітура Times. Папір офсетний № 1.
Наклад 150 прим.
Надруковано в ЗАТ “ВІПОЛ”, 03151 м.Київ, вул. Волинська, 60.

© Національний технічний університет України “КПІ”, 2015

ЗМІСТ

Кравець П.І., Шимкович В.М., Федорчук В.В., Гой А.А. Нейромережевий контролер системи стабілізації рухомого об'єкта з апаратно-програмною реалізацією на пліс…….……. 4

Сперкач М.О. Задача визначення максимально пізнього моменту початку виконання завдань із спільним жорстким директивним терміном паралельними пристроями різної продуктивності……..... 12

Марковський О.П., Терещенко С.Ю. Метод корекції подвійних помилок в каналах передачі цифрових даних зі спектральною модуляцією…….…… 19

Ролик А.И., Галушко Д.А., Барна В.В., Томащук А.В., Ясочка М.В. Оценка качества предоставления мультимедийных сервисов с использованием нейросетевого классификатора…….. 25

Габинет А. В. Анализ эффективности протоколов доступа к удаленным хранилищам видеоданных…….. 31

Теленик С.Ф., Амонс О.А., Троцький С.О., Білорус Д.С. Автоматизація підтримки бізнес-процесів у web-середовищі…….……..... 35

Болдак А.О., Мазуревич Ю.В. Спосіб оптимізації відображення графічної інформації в клієнт-серверних інформаційних системах……...….…….. 43

Теленик С.Ф., Букасов М.М., Карнаухов О.К., Філімонов В.Ф., Моргун М.В. Моделі оптимізації багаторівневого зберігання даних……...……..... 48

Минухин С.В. Методы распределения заданий между процессорами вычислительного кластера, использующими технологии масштабирования частоты и напряжения……...….. 54

Диброва М.А., Коган А.В., Воробьева А. Л. Способ формирования множества путей в сетевых центрах данных……..……..……. 60

Марковський О.П., Великий М.М. Метод резервування та прискореного відновлення даних в системах їх віддаленого зберігання…….……..…….…… 65

Павлов А.А., Мисюра Е.Б., Мельников О.В., Муха И.П., Лисецкий Т.Н. Алгоритмизация третьего и четвертого уровней четырехуровневой модели календарного и оперативного планирования и принятия решений в сетевых системах с ограниченными ресурсами…….. 72

Клименко І.А. Оптимізація реконфігурації в динамічно реконфігурованих обчислювальних системах……... 93

Дорогий Я.Ю., Дорога-Іванюк О.О.,Ференс Д.А. Реалізація алгоритму структурної оптимізації нейронної мережі……. 101

Порєв В.М. Евристичні алгоритми растрового моделювання динаміки процесів розповсюдження для геоінформаційних систем……..……. 107

Гапак О.М. Оцінка якості генератора голлманна, реалізованого на основі FCSR…… 119

Готович В.А., Назаревич О.Б. Застосування методу “Гусениця-SSA” для аналізу річного часового ряду електронавантаження організації…….. 123

Відомості про авторів……..……...…. 130

Summary…….……...……..... 131

УДК 681.5.13

КРАВЕЦЬ П.І.,
ШИМКОВИЧ В.М.,
ФЕДОРЧУК В.В.,
ГОЙ А.А.

НЕЙРОМЕРЕЖЕВИЙ КОНТРОЛЕР СИСТЕМИ СТАБІЛІЗАЦІЇ РУХОМОГО ОБ'ЄКТА З АПАРАТНО-ПРОГРАМНОЮ РЕАЛІЗАЦІЄЮ НА ПЛІС

В даній роботі розроблений та досліджений нейромережевий контролер системи стабілізації рухомого об'єкта з його апаратно-програмною реалізацією на FPGA, а саме розроблено макет балансування кульки на платформі, апаратне та програмне забезпечення даного макету, що демонструє роботу раніше розроблених нейромережевих компонентів систем керування.

In this paper developed and investigated neural network controller stabilization system of a moving object with its hardware and software implementation on FPGA, namely model developed balancing ball on a platform, hardware and software of this model, showing the work previously developed neural components of control systems.

Вступ

Нейромережеві системи управління являють собою новий високотехнологічний напрямок в теорії управління та відносяться до класу нелінійних динамічних систем [1,2]. Висока швидкодія за рахунок розпаралелювання вхідної інформації в поєднанні зі здатністю до навчання нейронних мереж робить цю технологію вельми привабливою для створення пристроїв управління в автоматичних системах [2]. Нейронні мережі можуть бути використані для побудови регулюючих та коректуючих пристроїв, еталонної, адаптивної, номінальної та інверсно-динамічної моделей об'єкта, на основі яких виконується спостереження та оцінка параметрів об'єкта керування (ОК), спостереження та оцінка величини діючих в системі збурень, пошук або обчислення оптимальної програми зміни керуючого впливу, ідентифікація ОК, прогнозування стану ОК та інше [2,3]. Здатність до навчання на будь-який заданий принцип функціонування дозволяє створити системи автоматичного керування, оптимальні по швидкодії, по енергоспоживанню і т. д., при цьому, природно, можлива реалізація декількох принципів функціонування та перехід з одного на інший. Навчені нейронні мережі не потребують для обчислень значних часових затрат, а тому системи з нейронними мережами мають значно кращу динаміку. Вони є універсальним засобом для моделювання складних нелінійних ОК та знаходження рішень в некоректних задачах [2-5].

Засоби реалізації нейромережевих систем управління повинні орієнтуватися на широке застосування в промислових умовах, бути універсальними і гнучкими, навчатися і адаптуватися в реальному часі, бути простими і дешевими, тому найбільш перспективними засобами можна вважати ПЛІС [6,7]. З появою ПЛІС проектування цифрових мікросхем перестало бути долею виключно великих підприємств з обсягами випуску в десятки і сотні тисяч кристалів. Проектування і випуск невеликої партії унікальних цифрових пристроїв став можливий в умовах проектно-конструкторських підрозділів промислових підприємств, в дослідницьких і навчальних лабораторіях і навіть в умовах домашніх радіоаматорських місць. Промислово випускаються «заготовки» програмованих мікросхем з електричним програмуванням і автоматизованим процесом перекладу схеми користувача в послідовність імпульсів програмування роблять проектування нових цифрових пристроїв порівнянним з розробкою програмного забезпечення [8-10].

Для побудови макета нейромережевого контролера в якості об'єкта управління обрано систему балансування кульки на платформі, здатну встановити кульку в задану точку. Платформа нахиляючись по кожній з двох горизонтальних осей, контролює положення кульки на платформі. По кожній осі нахил платформи виконується під дією електричного двигуна. Положення кульки на платформі фіксується за допомогою відеокамери. Таке завдання балансування є розширенням задачі балансування «перевернутого маятника». Такий об'єкт управління є складним багатовимірним, багатозв'язним об'єктом та розроблений для демонстрації роботи нейромережевих систем управління синтезованих на розроблених раніше компонентах [11-16].

Метою роботи є розробка макета нейромережевого контролера системи стабілізації кульки на платформі з апаратно-програмною реалізацією на ПЛІС.

Розробка макету

Першочерговим завданням є визначення положення кульки, точно, надійно і в не громіздкий та недорогий спосіб. Різні варіанти, які розглядалися, перераховані нижче. Відносні переваги і недоліки вказані також:

1. За допомогою сенсорної плівки, коли на платформу наклеюється сенсорний екран: не вистачає інформації, може бути складно реалізувати.

2. Відстеження положення кульки цифровою камерою: не дорогий, вимагає використання додаткового програмного забезпечення, вимагає використання додаткової конструкції, щоб встановити камеру.

3. Використання резистивної сітки на платформі (2D потенціометра): спосіб є надмірний і громіздкий.

4. Сітка інфрачервоних датчиків: спосіб є громіздким та надмірним.

5. Відстеження 3D-руху з кулі за допомогою інфрачервоного ультразвукової транспондера, прикріпленою до кульки, який обмінюється сигналами з трьома дистанційно розташованими вежами: дуже точні вимірювання, потрібен додатковий апарат у цілому, дуже дорого.

На підставі перелічених вище достоїнств і недоліків, пов'язаних з кожним вибором, було прийнято рішення відстежувати положення кульки за допомогою цифрової відеокамери.

Наступним завданням було спроектувати механізм нахилу пластини. Пластина має нахилятися по двох своїх осях, щоб мати можливість збалансувати кульку. Для цієї конструкції, були розглянуті наступні варіанти:

1. Два приводи прикріплюються по двох кутах пластини, яка підтримується кульовим шарніром в центрі, в якості двох необхідних ступенів руху.

2. Встановлення пластини на кільця підвісу. Один двигун перетворює карданний підвіс, що забезпечує одну ступінь обертання; другий двигун обертає пластину по відношенню до кільця забезпечуючи тим самим другий кут повороту.

3. Використання пристрою – ремінь та шків, повертати пластину за допомогою двох двигунів.

У цьому випадку, обрано перший варіант. На рисунку 1 показаний вигляд всієї системи в зборі, включаючи механізм нахилу пластини і зчитування положення кульки за допомогою відеокамери.

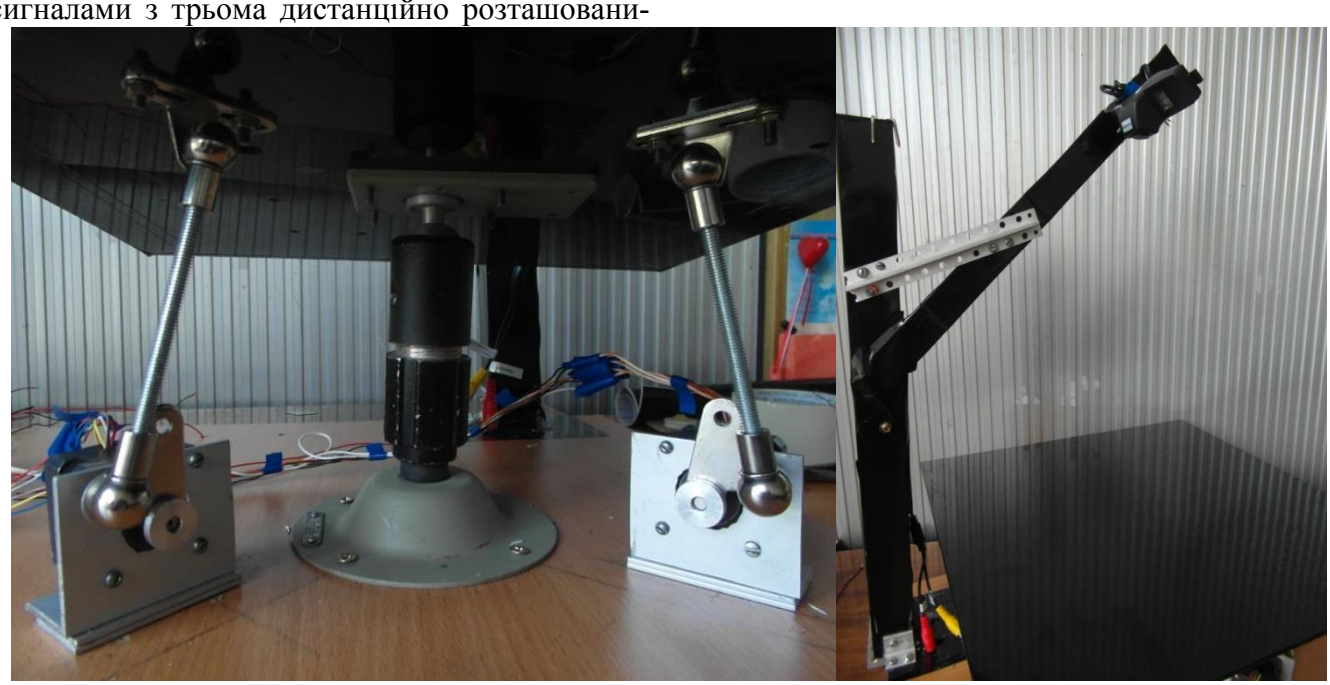


Рис. 1. Механізм керування платформою та відеокамера

Система балансування кульки на платформі складається з пластикової пластини обклеєної чорною акриловою плівкою на якій буде знаходитись біла кулька для настільного тенісу, приводним механізмом для нахилу пластини навколо двох осей, цифрової відеокамери, для відстеження положення кульки, апаратного та програмного забезпечення, що опрацьовує ін-

формацію та керує системою в режимі реального часу.

Кожен двигун приводить в одну вісь кута пластини обертання і з'єднаний з пластиною, за допомогою просторового механізму зв'язку, як показано на рисунку 1. Кожна зі сторін механізму просторової ув'язки складається з паралелограма. Це гарантує, що для малих рухів по всьому рівноваги, кути плити дорівнюють відповідним кутам двигуна. Пластина з'єднана з підставкою за допомогою центральної рухомої опори.

Для приведення в рух пластини використовуються крокові двигуни ПБМГ 200-265, які мають наступні характеристики:

– Тип двигуна: кроковий, уніполярний.
– Кут повороту за один крок: 1.8 градуса.
– Кількість кроків на повний оберт вала: 200.
– Кількість виводів: 6.
– Кількість обмоток: 2 з середньою точкою.
– Напруга живлення: 12В.
– Опір обмоток: 70Ом.
– Споживаний струм: 0,17 А.

Для відстеження положення кульки на платформі використано цифрову відеокамеру OV7670.

Рис. 2. Цифрова відеокамера OV7670

VGA камера моделі OV7670 з максимальною роздільною здатністю 640x480 пікселів (0,3 Мегапікселя), низькою напругою живлення, однокристальним CMOS датчиком зображення має широкий спектр форматів кадру і контролюється з допомогу шини Serial Camera Control Bus (SCCB), яка є аналогом шини I2C, IIC.

Камера OV7670 використовується для захоплення, первинної обробки та передачі зображення на контролер, дисплей, інший пристрій обробки зображення. Вона може управлятися за допомогою мікропроцесорного керуючого пристрою, а також з комп'ютера за допомогою спеціальних програм для управління послідовними портами.

Камера OV7670 формує зображення з максимальною роздільною здатністю 640x480 і може видати його зі швидкістю до 30 кадрів в секунду. Відеоінтерфейс використовує синхроімпульси по кадрам VSYNC, по рядках HREF і по пикселям PCLK. Дані пікселя, що представляють собою закодовану інформацію про його колір, видаються по 8-ми розрядному паралельному інтерфейсу D7-D0 по тактам PCLK. Один піксель не дорівнює одному байту. У разі кодування RGB дані про колір пікселя видаються у двох байтах. Також можливе YCbCr кодування. Для OV7670 існують безліч налаштувань. Наприклад, можна використовувати такі формати кадру:

– VGA (640 x 480);
– QVGA (320 x 240);
– CIF (352 x 240);
– QCIF (176 × 144);
– Ручне масштабування.

RGB доступна в наступних варіантах: RGB565, RGB555 і RGB444. Цифри позначають, кількість біт на кожен колір. Приміром, RGB565 – 5 біт на червоний, 6 біт на зелений і 5 біт на синій. Крім зазначених є ще безліч налаштувань (регулювання підсилень за кольорами, рівні сірого, полярності сигналів, внутрішня PLL та інші)

Камера OV7670 має один 18 піновий інтерфейс для підключення вхідних, вихідних сигналів та живлення:

– VDD – живлення;
– GND – загальний;
– SDIOC – (вхід) тактовий сигнал послідовного інтерфейсу SCCB управління камерою;
– SDIOD – (вхід / вихід) інформаційний сигнал (дані) послідовного інтерфейсу SCCB управління камерою;
– VSYNC – (вихід) імпульс кадрової синхронізації;
– HREF – (вихід) імпульс рядкової синхронізації;
– PCLK – (вихід) тактовий імпульс видачі байта з паралельного порту D7-D0;
– XCLK – (вхід) головний тактовий імпульс для роботи з OV7670;
– D7-D0 – 8-ми бітний паралельний відеовихід;
– RESET (Скидання) – виведення апаратного скидання камери;
– PWDN – висновок апаратного включення / вимикання камери.

Живлення камери може здійснюватися від мікропроцесорного керуючого пристрою, зовнішнього джерела живлення (блоку живлення, батарей). Напруга живлення модуля 3 - 5 В.

Фізичне моделювання системи

Наступні припущення використовуються в моделюванні вищеописаної фізичної системи:

1. Передбачається, що тертя-ковзання між кулею і пластиною досить високе, щоб запобігти кульку від ковзання на пластині. Це обмежує ступені свободи системи, а також робить рівняння руху простішим.

2. Обертання кулі навколо своєї вертикальної осі, як передбачається, буде незначним.

3. Тертям-коченням між кулькою та пластиною – знехтувати.

4. Передбачається, що буде невеликий рух пластини при рівноважній конфігурації. Це гарантує, що кути пластини будуть приблизно рівні кутам двигуна.

5. Пластина, як передбачається, має максимальну симетрію.

Фізична модель керованого об'єкта представлена на рисунку 3, де *X-Y-Z* є підставою рами. Платформа має два ступені свободи, а його орієнтація визначається двома кутами (q_1 і q_2), які представляють собою тіла (1-2) обертання. Рамка x"-y"-z" є пластина, закріплена система відліку, в той час як x'-y'-z' є проміжними.

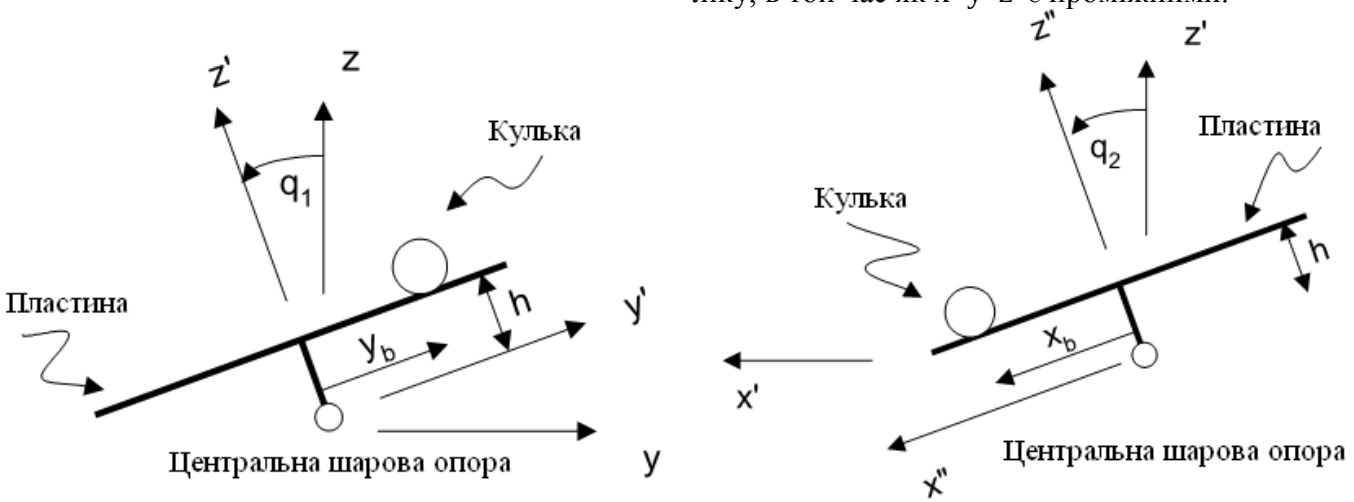


Рис. 3. Фізичне моделювання руху кульки по платформі

Схема просторового зв'язку показана на рисунку 4. З'єднання жорстко прикріплене до основи пластини і з'єднане з землею в точці O. Два двигуни з'єднані з тягами нерухомими з'єднаннями. Решта на шарових шарнірах.

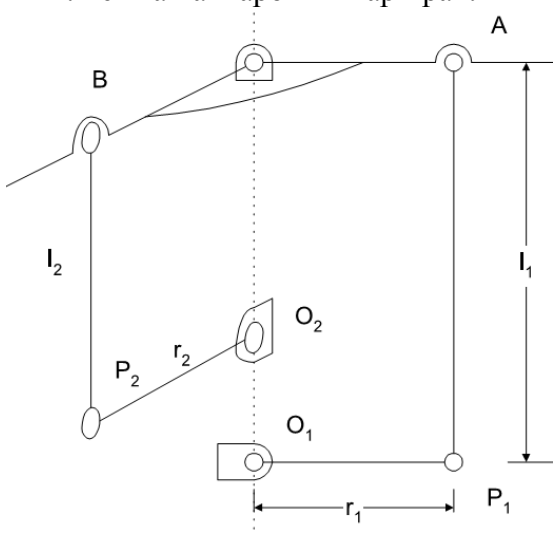


Рис. 4. Механізм керування платформою

Розрахунок ступеня свободи: кількість твердих тіл (N) = 5; кількість осей (P) = 2; кількість U-образних сполук (U) = 1; кількість шарових з'єднань = 4;

Кількість надлишкових ступенів свободи два (Повороти двох вертикальних зв'язків по своїх осях складають два резервних ступеня свободи, так як вони не впливають на стан системи в будь-якому випадку)

Таким чином, плита і механізм просторового зв'язку мають два ступені свободи, як і очікувалося. Це також дорівнює числу входів в систему, керуючий сигнал двома двигунами.

Виходячи з вищевикладеного, очевидно, що з чотирьох змінних – $\theta_{m1}, \theta_{m2}, q_1, q_2$, але тільки дві є незалежними, так як механізм має два ступені свободи. Таким чином, існують наступні два кінематичні рівняння зв'язку, які трансформують кут повороту валів двигунів в кут нахилу платформи

$$\begin{aligned}&\left(r_1\cos q_1 - r_1\cos\theta_{m1}\right)^2 + \\ &+\left(r_1\sin q_1 - r_1\sin\theta_{m1} + l_1\right)^2 = l_1^2, \\ &\left(r_2\cos\theta_{m2} - r_2\sin q_2\cos q_1 + l_2\right)^2 + \\ &+\left(r_2\cos q_2 - r_2\cos\theta_{m2}\right)^2 + r_2\sin q_2\sin q_1 = l_2^2.\end{aligned} \quad (1)$$

Помічено, що, в загальному випадку кут повороту пластини q_2 пов'язаний з кутами пово-

роту двигунів θ_{m1} та θ_{m2} складними нелінійними рівняннями, наведеними вище. Проте, для малих рухів на рівновагу кульки на платформі, можна зазначені вирази звести до наступних лінійних виразів:

$$q_1 = \theta_{m1},$$
$$q_2 \cong \theta_{m2}.$$

Справедливість цього припущення також перевірена експериментально. Встановлено, що x – координата:

$$m_b g r_b \sin q_2 \cos q_1 - m_b r_b \begin{bmatrix} (h+r)\ddot{q}_2 - y_b \ddot{q}_1 \sin q_2 - x_b \dot{q}_2^2 - x_b \dot{q}_1^2 \sin^2 q_2 + \\ +(h+r_b)\dot{q}_1^2 \sin q_2 \cos q_2 - 2\dot{y}_b \dot{q}_1 \sin q_2 + \ddot{x}_b \end{bmatrix} - I_b\left(\left(\ddot{x}_b / r_b\right) + \ddot{q}_2\right) = 0 \quad (2)$$

y – координата:

$$m_b g r_b \sin q_1 - m_b r_b \begin{bmatrix} x_b\left(\ddot{q}_1 \sin q_2 + \dot{q}_2 \dot{q}_1 \cos q_2\right) - (h+r_b)\left(\ddot{q}_1 \cos q_2 + \dot{q}_2 \dot{q}_1 \sin q_2\right) + \\ +\dot{q}_2 \dot{q}_1 (h+r_b) \sin q_2 - \dot{y}_b \dot{q}_1^2 + x_b \dot{q}_2 \dot{q}_1 \cos q_2 + 2\dot{x}_b \dot{q}_1 \sin q_2 + \ddot{y}_b \end{bmatrix} - \quad (3)$$
$$-I_b\left(\ddot{y}_b / r_b + \ddot{q}_1 \cos q_2 \dot{q}_2 \dot{q}_1 \sin q_2\right) = 0$$

Рівняння руху є нелінійними та малопридатні для синтезу системи управління на базі лінійної теорії управління. Такий об'єкт управління, як кулька на платформі є складнім нелінійним об'єктом, як видно з рівнянь (2) та (3), тому для управління ним потрібна нейромережева систему управління, яка відноситься до класу нелінійних динамічних систем. Далі розглянемо загальну структурну схему нейронконтролера.

Загальна структурна схема нейроконтролера

Сучасні засоби управління технічними та технологічними об'єктами мають в своєму складі обчислювальне ядро, засоби вводу аналогової та цифрової інформації, засоби виводу аналогової та цифрової інформації, засоби обміну даними з іншими обчислювальними пристроями, елементи візуалізації та оперативного управління. Такі засоби управління будуються на основі промислових комп'ютерів, вільно програмованих контролерів, спеціалізованих контролерів та мікроЕОМ. Далі розглядається контролер для управління динамічними об'єктами, в якому в якості обчислювального ядра ПЛІС (англ. FPGA).

На рисунку 5 представлена узагальнена структурна схема нейроконтролера, де в якості обчислювального ядра використовується нейрообчислювач на основі ПЛІС, що реалізує штучні нейроні мережі.

для відповідного діапазону роботи, відповідність між кутом повороту двигуна і кутом нахилу є задовільною.

Рівняння руху для цієї системи можуть бути отримані за допомогою законів Ньютона або рівнянь Лагранжа. Для цього випадку ці методи були використані для перевірки остаточних результатів. Повна нелінійна математична модель описується системою рівнянь:

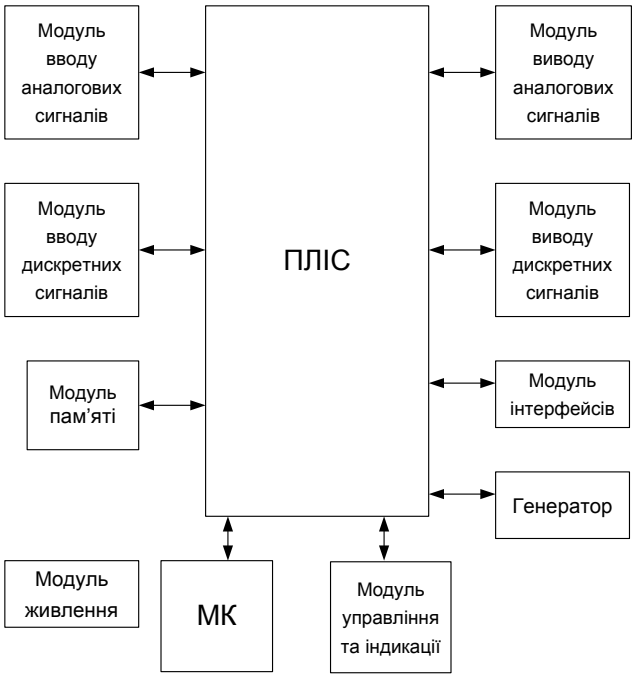


Рис.5. Загальна структурна схема нейроконтролера

До складу нейроконтролера входять: ПЛІС – програмована логічна інтегральна схема; МК – мікроконтролер; модуль вводу аналогових сигналів; модуль виводу аналогових сигналів; модуль вводу дискретних сигналів; модуль виводу дискретних сигналів; модуль пам'яті; модуль інтерфейсів; модуль управління та індикації; генератор;

ПЛІС є центральним елементом даної структури, яка реалізує нейромережеві алгоритми управління, включаючи алгоритм навчання мережі в режимі «on line».

Мікроконтролер (МК) виконує допоміжні функції, такі як завантаження конфігураційної

послідовності в мікросхему ПЛІС, реалізує функцію спостереження.

Модулі вводу/виводу аналогових та цифрових сигналів для вводу/виводу інформації з/в систему або об'єкт управління.

Модуль пам'яті забезпечує зберігання даних та конфігураційної послідовності ПЛІС.

В даній структурній схемі присутні два обчислювальних ядра: мікроконтролер (МК) і ПЛІС. В залежності від задач управління, мікроконтролер може бути: основним елементом нейромережевого контролера, а ПЛІС відводиться роль «швидкого» обчислювача фрагментів загального алгоритму роботи контролера пов'язаного з нейрообчисленями; доповнювати на рівних програму роботи ПЛІС, або бути чисто технічним елементом нейромережевого контролера, що забезпечує режими роботи ПЛІС. Але, в усіх випадках ПЛІС призначена для реалізації нейромережевих елементів системи управління, тому в подальшому основну увагу будемо приділяти обчислювачу на ПЛІС.

Апаратна реалізація системи управління

Для управління кроковими двигунами розроблені драйвери, схема електрична принципова яких зображена на рисунку 6. На рисунку dir – сигнал, який вказує напрямок обертання, якщо dir=1(2.6-5В) – обертання в одну сторону, 0 – обертання в іншу, step – сигнал, що задає кількість кроків.

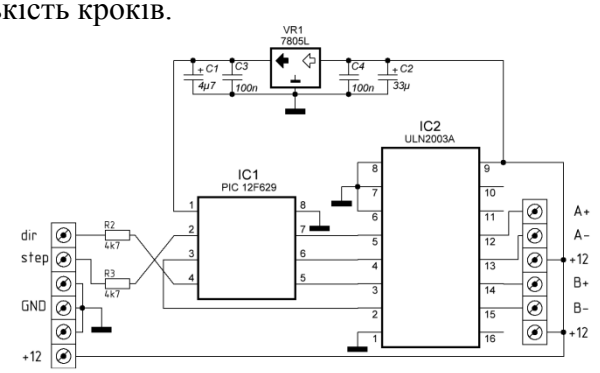


Рис.6. Схема електрична принципова драйвера крокового двигуна

В якості МК та модулів вводу/виводу аналогових сигналів обрано плату STM32F4 Discovery. Дана плата є відладочною платою на мікроконтролері ARM Cortex M4 від ST Microelectronics. Параметри плати:

–Мікроконтролер STM32F407VGT6 з частотою 168Mhz на 32-бітному ядрі ARM Cortex-M4F, 1 Мб флеш-пам'яті, 192 Кб оперативної пам'яті в корпусі LQFP100.

–Вбудований ST-LINK / V2 з можливістю використання ST-LINK / V2 як окремого пристрою (з SWD роз'ємом для програмування і налагодження). Не впевнений в ньому, т.к. JTAG є перевіреним методом налагодження, і потрібно побачити, що можна робити з новим роз'ємом.

–LIS302DL, ST MEMS датчик руху, 3-осьовий акселерометр з цифровим виходом.

–MP45DT02, ST MEMS датчик звуку, всенаправлений цифровий мікрофон.

–CS43L22, аудіо ЦАП з вбудованим підсилювачем класу D.

–Чотири світлодіода, LD3 (помаранчевий), LD4 (зелений), LD5 (червоний) і LD6 (синій).

–Дві кнопки (призначена для користувача і reset)

–Порт USB OTG FS і конектор micro-AB

–Штирі для швидкого з'єднання з іншими пристроями підключення до всіх висновків LQFP100.

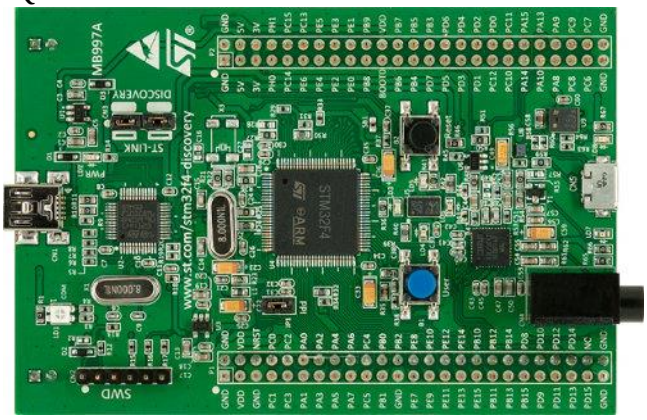


Рис.7. Плата STM32F4 Discovery

До даної плати підключено цифрову відеокамеру OV7670 по 18-ти піновому інтерфейсу та шині SCCB, яка є аналогом шини I^2C, IIC. В STM32F4 виконується початкова обробка зображення, виділення координат кульки на платформі та передача їх до ПЛІС та інші функції. Плата STM32F4 Discovery зображена на рисунку 7.

В якості ПЛІС обрано плату Xilinx Spartan-3 з чіпом – XC3S200. Інструментальний модуль Xilinx Spartan-3 Starter Kit може використовуватися для реалізації автономних систем керування, збору і обробки цифрової інформації, вбудованих цифрових пристроїв з різними комп'ютерними інтерфейсами.

Плата має три додаткові 40-ка пінові роз'єми для вводу та виводу дискретних сигналів. На ПЛІС сімейства FPGA даної плати розміщено нейромережеву систему управління на основі компонентів розроблених в попередніх роботах [16]. Інструментальний модуль Xilinx Spartan-3 Starter Kit зображений на рисунку 8.

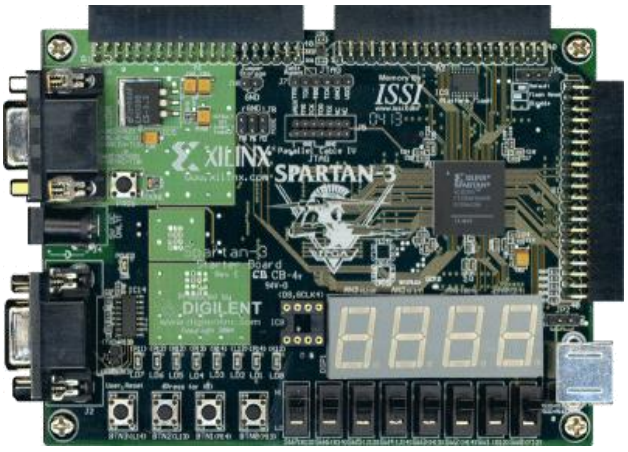


Рис.8. Плата Xilinx Spartan-3 Starter Kit

Функціональні можливості і конструктивне виконання інструментального модуля дозволяють не тільки виконувати налагодження проектів, що реалізуються на основі ПЛІС сімейства Spartan-3, але і використовувати його в якості промислового серійного варіанту розроблюваного пристрою. Застосування серійно випускаємих модулів дозволяє значно скоротити сумарну тривалість розробки макета за рахунок виключення етапів проектування, виготовлення друкованої плати пристрою і монтажу компонентів.

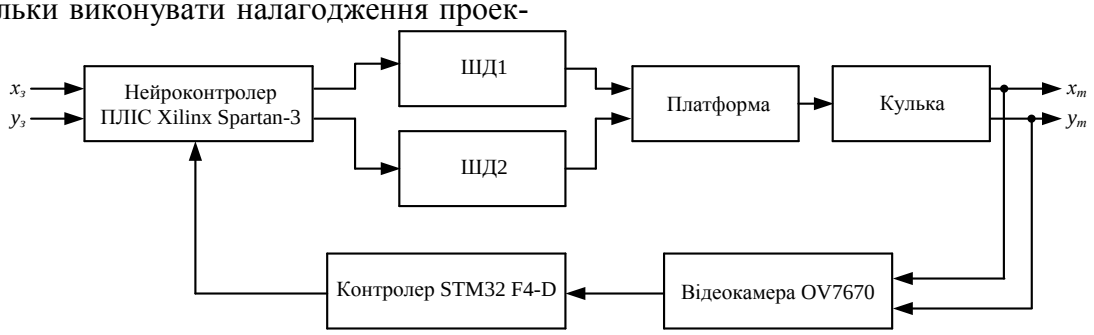


Рис.9. Система управління кульки на платформі

Розробка системи управління і ПО

Для керування положенням кульки на платформі розроблено нейромережеву систему управління з зворотнім зв'язком, представлену на рисунку 10. Схема є класичною схемою спеціалізованого інверсного навчання. За методом описаним в [16] синтезуємо нейромережевий компонент такий як, обернена(інверсна) модель об'єкта управління. На рисунку 11 нейроконтролер включає в себе інверсну модель об'єкта керування, компонент її налагодження, розроблений за методом описаним в [14] та обернений зв'язок.

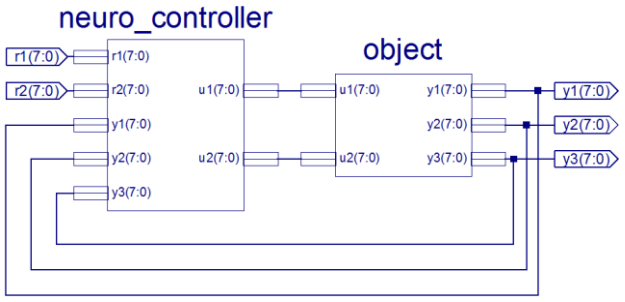


Рис.10. Модель системи управління

Схема представлена на рисунку 10 є моделлю всієї системи управління включно з об'єктом на основі розроблених раніше нейромережевих компонентах, в FPGA завантажено сам нейроконтролер.

Висновок

Розроблено узагальнену структурну схему нейроконтролера на основі ПЛІС, що реалізує базові компоненти нейромережевих систем управління.

Реалізовано макет нейроконтроллера для системи стабілізації рухомого об'єкта на обмеженій площині та проведено його дослідження, які підтвердили високу ефективність роботи.

Такий об'єкт управління, як кулька на платформі є складнім нелінійним об'єктом тому для управління ним розроблено нейромережеву систему управління, яка відноситься до класу нелінійних динамічних систем. Розроблена в даній роботі нейромережева система управління функціонує та адаптуються в режимі реального часу та формує досить складні функції управління. Коли система управління ввімкнена, кулька буде збалансована в будь-якій заданій точці пластини. Вона також може бути спрямована на перехід від однієї точки до іншої, і залишитися там. Використовуючи цю систему управління можна задати траєкторію

руху кульки по платформі, наприклад рух по колу.

Побудовану і випробувану систему можна додатково використовувати як відмінний випробувальний стенд для тестування різних інших схем управління. В подальшому можуть бути розроблені на базі ПЛІС більш оптимальні контролери для досягнення ще більш високої продуктивності.

Перелік посилань

1. Саймон Хайкин. Нейронные сети. Полный курс. – М.: Вильямс, 2006. – 1104с.
2. Терехов В.А., Ефимов Д.В., Тюкин И.Ю. Нейросетевые системы управления: Учеб. пособие для вузов. – М: Высшая школа. 2002. – 183с.
3. Сигеру Омату. Нейроуправление и его приложения. – М.: ИПРЖР, 2000. – 272с.
4. Рутковская Д., Пилиньский М., Рутковский Л. Нейронные сети, генетические алгоритмы и нечеткие системы: Пер. с польск. И.Д. Рудинского. – М.: Горячая линия – Телеком, 2007. – 452 с.
5. Егупов Н.Д. Методы робастного, нейро-нечеткого и адаптивного управления. – М.: Изд-во МГТУ им. Н.Э. Баумана, 2001. – 744 с.
6. Гильгурт С.Я. Анализ применения реконфигурируемых вычислителей на базе ПЛИС для реализации нейронных сетей // Моделювання та інформаційні технології. Зб. наук. пр. ІПМЕ НАН України. – Вип. 37. – Київ: 2006. – С. 168-174 с.
7. Логовский А. Технология ПЛИС и ее применение для создания нейрочипов. // Открытые системы. – 2000. – № 4. – С. 100-102 с.
8. Соловьев В. Проектирование цифровых систем на основе ПЛИС. – М.: Радио и связь, 2003. – 376с.
9. Тарасов И.Е. Разработка цифровых устройств на основе ПЛИС Xilinx® с применением языка VHDL. – М.: Горячая линия. – Телеком, 2005. – 252 с.
10. Сергиенко А.М. VHDL для проектирования вычислительных устройств – К ЧП «Корнейчук», ООО «ТИД «ДС», 2003. – 208 с.
11. Petr I. Kravets, Tatyana I. Lukina, Valeriy A. Zherebko, Vladimir N. Shimkovich. Methods of Hardware and Software Realization of Adaptive Neural Network PID Controller on FPGA-Chip. // Journal of Automation and Information Sciences. New York, USA, 2011, Vol. 43 / Issue 4, P. 70-77.
12. Кравець П.І., Шимкович В.М., Зубенко Г.А. Технологія апаратно-програмної реалізації штучного нейрона та штучних нейронних мереж засобами FPGA / Вісник НТУУ «КПІ». Інформатика, управління та обчислювальна техніка: Зб. наук. пр. – К.: Век+, – 2012. – №55. – С.174-180.
13. Кравець П.І., Лукіна Т.Й., Шимкович В.М., Ткач І.І. Розробка та дослідження технології оцінювання показників нейромережевих моделей MIMO-об'єктів управління / Вісник НТУУ «КПІ». Інформатика, управління та обчислювальна техніка: Зб. наук. пр. – К.: Век+, – 2012. – №57. – С.144-150.
14. Кравец П.И., Шимкович В.Н. Метод оптимизации весовых коэффициентов нейронных сетей с помощью генетического алгоритма при реализации на программируемых логических интегральных схемах / Международный научно-технический журнал «Электронное моделирование». – 2013. – 35, №3. – С. 65-75.
15. Кравець П.І., Шимкович В.М., Зубенко Г.А. Моделі штучних нейронних мереж при їх апаратно-програмній реалізації на FPGA / Сборник трудов XIV международной научной конференции им. Т.А.Таран «Интеллектуальный анализ информации ИАИ-2014», К., 14-16.05.14.

Кравець П.І., Шимкович В.М., Омельченко П. Нейромережеві компоненти систем керування динамічними об'єктами з їх апаратно-програмною реалізацією на FPGA / Вісник НТУУ «КПІ». Інформатика, управління та обчислювальна техніка: Зб. наук. пр. – К.: Век+, – 2013. – №59. – С.78-85.

УДК 519.854.2

СПЕРКАЧ М.О.

ЗАДАЧА ВИЗНАЧЕННЯ МАКСИМАЛЬНО ПІЗНЬОГО МОМЕНТУ ПОЧАТКУ ВИКОНАННЯ ЗАВДАНЬ ІЗ СПІЛЬНИМ ЖОРСТКИМ ДИРЕКТИВНИМ ТЕРМІНОМ ПАРАЛЕЛЬНИМИ ПРИСТРОЯМИ РІЗНОЇ ПРОДУКТИВНОСТІ

В статті досліджені властивості задачі календарного планування виконання завдань зі спільним жорстким директивним терміном паралельними пристроями різної продуктивності з метою максимізації моменту запуску пристроїв за умови, що усі завдання не запізнюються. Сформульована допоміжна оптимізаційна задача, за результатами якої визначені достатні умови оптимальності розкладів. На основі достатніх умов оптимальності визначено множину перестановок, які дозволяють послідовно покращувати значення критерію. Розроблена поліноміальна складова ПДС-алгоритма розв'язання задачі.

This article studies the properties of the problem of scheduling tasks with a common tough due date on parallel machines with different speeds to maximize the launch moment of the machines with the condition that all the tasks are not delayed. An auxiliary optimization problem is formulated, the results of which are the sufficient conditions for a schedule optimality. On their basis the set of permutations is defined that allow to consistently improve the criterion value. The polynomial component of PDC-algorithm for the problem solution is developed.

1. Вступ

Оперативно-календарне планування виробництва є частиною внутрішньоцехового планування та полягає в організації ритмічної роботи виробництва з випуску продукції, в заданих обсягах і у встановлені терміни при найбільш ефективному використанні всіх виробничих ресурсів підприємства. Змістом внутрішньоцехового планування є розробка оперативних планів і складання поточних графіків роботи виробничих ділянок. Для цього доцільно застосовувати моделі та методи теорії розкладів.

Стаття присвячена дослідженню однієї задачі теорії розкладів.

2. Постановка задачі

Задано множину завдань $J = \{1,2,...,n\}$ та кількість пристроїв *m*. Пристрої працюють паралельно і є взаємозамінними у тому сенсі, що кожний з пристроїв може виконувати будь-яке завдання з множини J. Пристрої відрізняються один від одного продуктивністю виконання завдань. При цьому пристрої можна впорядкувати за швидкістю виконання завдання і цей порядок однаковий для всіх завдань: для кожного пристрою i існує коефіцієнт k_i такий, що тривалість виконання завдання j на пристрої i дорівнює $k_i p_j$. «Еталонним» будемо називати пристрій з коефіцієнтом продуктивності $k = 1$. В цьому сенсі величина p_j є тривалістю виконання завдання j на еталонному пристрої. Величину k_i будемо називати *коефіцієнтом продуктивності* (якщо $k_i > 1$,то пристрій i менш продуктивний, ніж еталонний, якщо $k_i < 1$ – більш продуктивний).

Передбачається, що всі завдання множини *J* надходять одночасно та мають спільний жорсткий директивний термін *d*, процес обслуговування кожного завдання протікає без переривань до завершення обслуговування завдань. Всі пристрої працюють без переривань. Необхідно знайти максимальний момент запуску пристроїв $r_{\max}$, що дозволяє отримати допустимий розв'язок (розклад, у якому усі завдання не запізнюються) Під моментом запуску розуміється момент початку виконання множини завдань (мінімальний з моментів початку виконання завдань з множини J).

Далі, не втрачаючи загальності, будемо вважати, що виконуються такі умови: еталонним є найбільш продуктивний пристрій і у нього $k_i = 1$ (звідси витікає, що усі $k_i \geq 1$); усі p_j є цілими числами.

Задача, що розглядається належить до класу NP.

3. Дослідження задачі

Визначимо теоретично мінімальній час, за який усі пристрої могли б виконати усі завдання в об'ємі $\sum_{j=1}^{n} p_j$. Цю величину можна отримати, якщо вважати, що усі пристрої можуть паралель-

но виконувати кожне з завдань. Для цього m паралельних пристроїв розглядаються як один, а фактичні тривалості завдань замінюються їх так званими узагальненими значеннями, рівними:

$$\tilde{p}_j = \frac{1}{\sum_{i=1}^{m} \frac{1}{t_{ij}}},$$

де $t_{ij} = k_i p_j$ – тривалість виконання завдання j пристроєм i ($i = \overline{1,m},\ j = \overline{1,n}$);

$\frac{1}{t_{ij}}$ – частка завдання j, яка виконується пристроєм i за одну одиницю часу;

$\sum_{i=1}^{m} \frac{1}{t_{ij}}$ – частка завдання j, яка виконуються всіма пристроями за одну одиницю часу;

$\frac{1}{\sum_{i=1}^{m} \frac{1}{t_{ij}}}$ – тривалість виконання завдання j всіма пристроями.

Тоді, мінімально можливий час, за який усі пристрої могли б виконати усі завдання, становить:

$$C^* = \sum_{j=1`}^{m} \tilde{p}_j, \qquad (1)$$

$$C^* = \sum_{j=1`}^{n} \frac{1}{\sum_{i=1}^{m} \frac{1}{k_i p_j}}. \qquad (2)$$

Підійдемо до визначення величини C^* з іншого боку. Як і раніше, визначимо C^* – як теоретично мінімально можливий час, за який усі пристрої могли б виконати усі завдання в об'ємі $\sum_{j=1}^{n} p_j$, але тепер за умови, що кожне завдання виконується тільки одним пристроєм. В ідеальному випадку розклад є *рівномірним* – у ньому усі пристрої обслуговують усі свої завдання за час C^*. Очевидно, що рівномірний розклад є оптимальним. Це пояснюється наступним. Переміщення будь-якого завдання з одного пристрою на інший збільшує час завершення усіх завдань на пристрої-реціпієнті. Будь-який обмін груп завдань з двох різних пристроїв також збільшує час завершення усіх завдань на одному з пристроїв.

Введемо величину $c_i^* = \frac{C^*}{k_i}$ – «ідеальну» зведену тривалість зайнятості пристрою i ($i = \overline{1,m}$). У разі рівномірного розкладу справедливо наступне:

$$\sum_{j=1}^{n} p_j = \sum_{i=1}^{m} c_i^*, \qquad (3)$$

$$\sum_{j=1}^{n} p_j = \sum_{i=1}^{m} \frac{C^*}{k_i}, \qquad (4)$$

$$\sum_{j=1}^{n} p_j = C^* \sum_{i=1}^{m} \frac{1}{k_i}$$

(сумарний час виконання усіх завдань в еталонних значеннях тривалостей дорівнює сумі ідеальних зведених тривалостей зайнятості пристроїв).

Звідси отримаємо значення C^*:

$$C^* = \frac{\sum_{j=1}^{n} p_j}{\sum_{i=1}^{m} \frac{1}{k_i}}, \qquad C^* = \sum_{j=1}^{n} \frac{p_j}{\sum_{i=1}^{m} \frac{1}{k_i}}.$$

Поділимо чисельник та знаменник j-го доданку на величину p_j ($j = \overline{1,n}$):

$$C^* = \sum_{j=1}^{n} \frac{1}{\sum_{i=1}^{m} \frac{1}{k_i p_j}}. \qquad (5)$$

Вираз (5) співпадає з отриманим раніше виразом (2). За умови ідентичності пристроїв ($k_i = 1,\ i = \overline{1,m}$) вираз для величини C^* матиме вигляд $C^* = \frac{\sum_{j=1}^{n} p_j}{m}$ (в цьому випадку результати розрахунків C^* співпадають із результатами, отриманими при дослідженні задачі з ідентичними пристроями [1]).

Величина C^* може бути як цілим так і не цілим числом.

Алгоритм побудови початкового розкладу

Мета цього евристичного алгоритму – рівномірно навантажити пристрої з урахуванням того, що їх продуктивності різні: поточне завдання призначається на той пристрій, у якого невикористаний зведений фонд робочого часу найбільший.

КРОК 1. Перенумерувати завдання множини J за незростанням тривалостей їх виконання на еталонному пристрої.

КРОК 2. Розрахувати C^* (теоретично мінімальній час виконання пристроями усіх завдань) та $c_i^* = \frac{C^*}{k_i}$ ($i = \overline{1,m}$) – «ідеальні» зведені тривалості зайнятості пристроїв.

КРОК 3. Встановити значення величин невикористаного зведеного фонду робочого часу пристроїв: $f_i = c_i^*$, $i = \overline{1,m}$.

КРОК 4. Встановити (реальний) час зайнятості пристроїв: $C_i = 0$, $i = \overline{1,m}$.

КРОК 5. Обрати завдання $j = 1$.

КРОК 6. Обрати пристрій i з максимальним невикористаним зведеним фондом робочого часу f_i (а якщо таких декілька, то обрати пристрій з найменшим значенням k_i).

КРОК 7. Призначити завдання j на пристрій i: $C_i = C_i + k_i p_j$; $f_i = f_i - p_j$.

КРОК 8. Перейти до наступного завдання: $j = j+1$. Якщо $j > n$, кінець алгоритму, в іншому випадку переходимо до **КРОКУ 6**.

Відмітимо, що наведений алгоритм за умови ідентичності пристроїв (тобто $k_i = 1$, $i = \overline{1,m}$) дає той же самий результат, що і алгоритм побудови початкового розкладу для задачі з ідентичними пристроями [1].

Розглянемо деякий допустимий розклад σ. Позначимо в цьому розкладі:

$J_i(\sigma)$ – множина завдань, що виконується пристроєм i;

$C_i(\sigma) = \sum_{j \in J_i(\sigma)} k_i p_j$ – тривалість зайнятості пристрою i;

$\Delta_i(\sigma) = \max\{0; C_i(\sigma) - C^*\} = \max\left\{0; \sum_{j \in J_i(\sigma)} k_i p_j - C^*\right\}$ – виступ пристрою i;

$R_i(\sigma) = \max\{0; C^* - C_i(\sigma)\} = \max\left\{0; C^* - \sum_{j \in J_i(\sigma)} k_i p_j\right\}$ – резерв пристрою i ($i = \overline{1,m}$).

Позначимо через $C_i^/(\sigma) = \dfrac{C_i(\sigma)}{k_i}$ зведену тривалість зайнятості пристрою i ($i = \overline{1,m}$) в розкладі σ – час зайнятості пристрою, розрахований у еталонних тривалостях виконання завдань.

Відповідні зведені величини:

$\Delta_i^/(\sigma) = \max\{0; C_i^/(\sigma) - c_i^*\}$; $i = \overline{1,m}$;

$R_i^/(\sigma) = \max\{0; c_i^* - C_i^/(\sigma)\}$, $i = \overline{1,m}$.

Визначимо:

$I_\Delta(\sigma)$ – множина пристроїв, для яких $\Delta_i(\sigma) > 0$;

$I_R(\sigma)$ – множина пристроїв, для яких $R_i(\sigma) > 0$;

$I_0(\sigma)$ – множина пристроїв, для яких $\Delta_i(\sigma) = R_i(\sigma) = 0$.

З урахуванням того, що величина d є константою (оскільки не залежить від розкладу) критерій оптимізації вихідної задачі (максимізація моменту r запуску пристроїв)

$$r = d - (C^* + \max_{1 \le i \le m} \Delta_i(\sigma)) \to \max$$

є еквівалентним критерію мінімізації загального часу виконання усіх завдань:

$$C^* + \max_{1 \le i \le m} \Delta_i(\sigma) \to \min .$$

В свою чергу, з урахуванням того, що $C^* = const$, критерій спрощується і зводиться до мінімізації максимального виступу:

$$\max_{1 \le i \le m} \Delta_i(\sigma) \to \min .$$

Рисунок 1 ілюструє взаємозв'язок величин, що входять до вказаних критеріїв.

Рис. 1. Взаємозв'язок величин

Для введених величин має місце наступне:

$$\sum_{j \in J_i(\sigma)} k_i p_j = C^* - R_i(\sigma) + \Delta_i(\sigma),\ i = \overline{1,m}. \qquad (6)$$

Розділимо i-те рівняння в (6) на k_i ($k_i \ne 0$):

$$\sum_{j \in J_i(\sigma)} p_j = \frac{C^*}{k_i} - \frac{R_i(\sigma)}{k_i} + \frac{\Delta_i(\sigma)}{k_i},\ i = \overline{1,m},$$

$$\sum_{j \in J_i(\sigma)} p_j = c_i^* - R_i^/(\sigma) + \Delta_i^/(\sigma),\ i = \overline{1,m}.$$

Застосовуючи методологію побудови ПДС-алгоритмів [2], згідно з якою поліноміальна складова алгоритму породжується логіко-аналітичними умовами, виконання яких гарантує оптимальність отриманого рішення, визначимо ознаки оптимальності розкладів.

4. Ознаки оптимальності розкладів

Як було вказано раніше, «ідеальним» розкладом є рівномірний розклад, тобто розклад, у якого $\sum_{i=1}^{m} \Delta_i(\sigma) = \sum_{i=1}^{m} R_i(\sigma) = 0$. Рівномірний розклад можливо отримати тільки тоді, коли C^* –

ціле і усі $c_i^* = \dfrac{C^*}{k_i}$ також є цілими числами (інакше не можливі рівності $\sum_{j \in J_i} p_j = c_i^*$, $i = \overline{1,m}$ – ліві частини рівностей за припущенням є сумами цілих величин).

Отже, ми визначили таку ознаку оптимальності розкладу (достатню умову оптимальності): якщо в розкладі $\sum_{j \in J_i} p_j = c_i^*$,

($\sum_{i=1}^{m} \Delta_i^{/}(\sigma) = \sum_{i=1}^{m} R_i^{/}(\sigma) = 0$), $i = \overline{1,m}$, то поточний розклад є оптимальним.

Якщо виконується умова:

$$\exists i \mid c_i^* \notin Z . \tag{7}$$

(серед величин c_i^* є не цілі), то у цьому випадку принципово неможливо побудувати рівномірний розклад. Визначимо ознаку оптимальності для цього випадку.

Позначимо:

$$\delta = \sum_{j=1}^{n} p_j - \sum_{i=1}^{m} \left\lfloor \frac{C^*}{k_i} \right\rfloor = \sum_{j=1}^{n} p_j - \sum_{i=1}^{m} \lfloor c_i^* \rfloor .$$

Враховуючи (3) маємо, що $\delta \geq 0$, ціле.

За умови (7) маємо, що $\delta > 0$, ціле. Визначимо, який вигляд матиме «ідеальний» розклад у цьому випадку. Іншими словами, визначимо *контур* оптимального розкладу.

Визначення. Розклад, у якому на пристрої призначені не усі завдання із множини J, будемо називати неповним.

Представимо, що ми розподілили між пристроями деяку кількість завдань (нехай вони складають множину $\overline{J} \subset J$) і отримали неповний розклад $\overline{\sigma}$, у якому:

$$\sum_{j \in \overline{J_i}} t_{ij} = k_i \lfloor c_i^* \rfloor, \ i = \overline{1,m}; \tag{8}$$

або

$$\sum_{j \in \overline{J_i}} p_j = \lfloor c_i^* \rfloor, \ i = \overline{1,m}, \tag{9}$$

тут $\overline{J_i}$ – множина завдань, які в отриманому неповному розкладі $\overline{\sigma}$ виконуються пристроєм i). Відмітимо, що залишились недорозподіленими завдання, сумарна тривалість виконання яких становить:

$$\delta = \sum_{j=1}^{n} p_j - \sum_{i=1}^{m} \sum_{j \in \overline{J_i}} p_j . \tag{10}$$

З урахуванням (9), рівняння (10) має вигляд:

$$\delta = \sum_{j=1}^{n} p_j - \sum_{i=1}^{m} \lfloor c_i^* \rfloor . \tag{11}$$

Припустимо, що залишилось розподілити δ завдань, еталонна тривалість кожного з яких дорівнює 1.

Отже, перед нами постає така задача: розподілити одиничні завдання в кількості δ між пристроями так, щоб максимальний з виступів повного розкладу був мінімальним. В результаті ми отримаємо розклад, у якому робота в об'ємі $\sum_{j=1}^{n} p_j$ розподілена найкращим чином (ідеальний розклад для сукупності завдань з заданою сумою тривалостей їх виконання).

Позначимо:

$$e_i = c_i^* - \sum_{j \in \overline{J_i}} p_j , \ i = \overline{1,m}; \tag{12}$$

де e_i – резерв пристрою i у неповному розкладі $\overline{\sigma}$, в якому недорозподілено δ одиничних завдань.

З урахуванням (9) маємо:

$$e_i = c_i^* - \lfloor c_i^* \rfloor, \ i = \overline{1,m} . \tag{13}$$

З (13) слідує, що $\forall\, e_i \geq 0$.

На рисунку 2 відображено контур найкращого неповного розкладу $\overline{\sigma}$, у якому на пристрої призначені завдання в об'ємі $\sum_{i=1}^{m} \lfloor c_i^* \rfloor$ (в еталонних тривалостях).

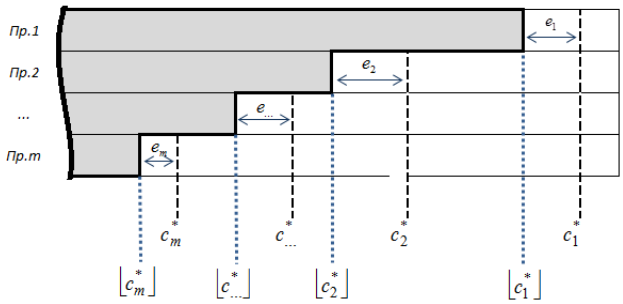


Рис. 2. Контур найкращого розкладу, у якому на пристрої призначені завдання в об'ємі $\sum_{i=1}^{m} \lfloor c_i^* \rfloor$ ***(в еталонних тривалостях)***

Просумуємо рівняння (13) по i:

$$\sum_{i=1}^{m} e_i = \sum_{i=1}^{m} c_i^* - \sum_{i=1}^{m} \lfloor c_i^* \rfloor ,$$

з урахування (3) маємо:

$$\sum_{i=1}^{m} e_i = \sum_{j=1}^{n} p_j - \sum_{i=1}^{m} \lfloor c_i^* \rfloor = \delta .$$

Таким чином, можна сформулювати наступну ***допоміжну оптимізаційну задачу***: необхідно δ завдань одиничної довжини розподілити між m пристроями, за умов, що пристрій i має

резерв e_i, $i=1,m$ і $\sum_{i=1}^{m} e_i = \delta$ з метою мінімізації максимального з виступів (з урахуванням продуктивностей пристроїв). Математична модель цієї задачі має наступний вигляд.

Змінні: x_i – кількість «одиничних» еталонних завдань, що повинні бути призначені на пристрій i, $i=\overline{1,m}$.

Обмеження – кількість «одиничних» еталонних завдань становить δ:

$$\sum_{i=1}^{m} x_i = \delta;\ \ x_i \ge 0,\ \ i=1,m\text{, цілі.} \qquad (14)$$

Цільова функція: мінімізація максимального з виступів (з урахуванням продуктивностей пристроїв):

$$\max_i \{k_i(x_i - e_i)\} \to \min. \qquad (15)$$

Алгоритм розв'язання допоміжної оптимізаційної задачі

Дано: δ; e_i, $i=1,m$ ($\sum_{i=1}^{m} e_i = \delta$); k_i, $i=1,m$; $T_i = C^* - e_i k_i$ (тривалість виконання завдань пристроєм i в неповному ідеальному розкладі $\overline{\sigma}$), $i=1,m$.

КРОК 1 Ініціалізація: $x_i = 0$, $i=1,m$.

КРОК 2 Поки $\delta \ge 1$ (поки є нерозподілені одиничні завдання)

2.1 Знайти

$$\min\{T_1 + k_1;\ T_2 + k_2;\ldots;\ T_m + k_m\}, \qquad (16)$$

($T_i + k_i$ – тривалість виконання усіх завдань пристроєм i за умови, що на цей пристрій буде призначено поточне одиничне завдання). Нехай мінімум в (16) відповідає пристрою $i = q$.

2.2 Призначити на пристрій q одиничне завдання: $x_q = x_q + 1$, $\delta = \delta - 1$.

2.3 Перерахувати тривалість виконання усіх завдань пристроєм q: $T_q = T_q + k_q$.

КІНЕЦЬ АЛГОРИТМУ

Отже, отримали такі ознаки оптимальності (достатні умови оптимальності).

Ознака оптимальності 1

Якщо $C^* = \sum_{j=1}^{n} \dfrac{1}{\sum_{i=1}^{m} \dfrac{1}{k_i p_j}}$ – ціле та усі $c_i^* = \dfrac{C^*}{k_i}$ також є цілими числами – для заданого набору завдань можливо побудувати рівномірний розклад.

Якщо у розкладі σ виконується:

$$C_i^{/}(\sigma) = \sum_{j \in J_i} p_j = c_i^*,\ i=\overline{1,m},$$

$$(C_i(\sigma) = k_i \sum_{j \in J_i} p_j = k_i c_i^* = C^*,\ i=\overline{1,m})$$

або

$\sum_{i=1}^{m} \Delta_i(\sigma) = \sum_{i=1}^{m} R_i(\sigma) = 0$, то це означає, що поточний розклад є рівномірним (оптимальним).

Ознака оптимальності 2

Якщо $\delta = \sum_{j=1}^{n} p_j - \sum_{i=1}^{m} \lfloor c_i^* \rfloor > 0$

($\exists i \mid c_i^* = \dfrac{C^*}{k_i} \notin Z$), то у цьому випадку отримати рівномірний розклад не можливо.

Нехай x_i^*, $i=\overline{1,m}$ – оптимальний розв'язок допоміжної оптимізаційної задачі (14)-(15), $T_i^* = C^* + k_i(x_i^* - e_i)$ – тривалість виконання завдань в «ідеальному» розкладі.

Тоді розклад σ, в якому:

$$C_i^{/}(\sigma) = \lfloor c_i^* \rfloor + x_i^*,\ i=\overline{1,m}$$

або

$C_i(\sigma) = T_i^*$, $i=\overline{1,m}$, є оптимальним.

На основі ознак оптимальності розкладів визначимо множину перестановок, яка дозволить послідовно покращувати значення критерію.

5. Розробка множини перестановок

Після розв'язання допоміжної задачі отримали контур оптимального розкладу для заданої сумарної довжини завдань множини J, у якого i-й пристрій завершує свої завдання у момент:

– C^*, якщо $\delta = 0$;

– T_i^*, якщо $\delta > 0$ (рис. 3).

Введемо позначення:

$$Z_i(\sigma) = \max\{0;\ C_i(\sigma) - T_i^*\} = \max\left\{0;\ \sum_{j \in J_i(\sigma)} k_i p_j - T_i^*\right\}$$

(аналог величини $\Delta_i(\sigma)$, але в даному випадку розраховується відхилення від «ідеальної» тривалості зайнятості пристрою T_i^* в оптимальному контурі розкладу);

$$E_i(\sigma) = \max\{0;\ T_i^* - C_i(\sigma)\} = \max\left\{0;\ T_i^* - \sum_{j \in J_i(\sigma)} k_i p_j\right\}$$

(аналог величини $R_i(\sigma)$;

$I_Z(\sigma)$ – множина таких пристроїв, для яких $Z_i(\sigma) > 0$;

$I_E(\sigma)$ – множина таких пристроїв, для яких $E_i(\sigma) > 0$.

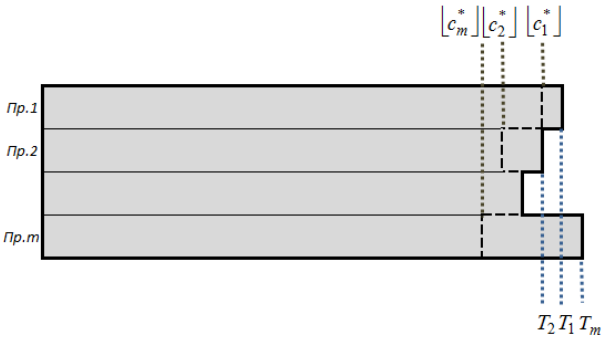


Рис. 3. Оптимальний за другою ознакою оптимальності контур розкладу

Для покращення розкладу необхідно направити зусилля на зменшення величини $\max_i Z_i(\sigma)$. Для цього пропонується використовувати обмін завдань між пристроями з множин $I_Z(\sigma)$ та $I_E(\sigma)$: коли деяка підмножина завдань з пристрою h (позначимо її як $K_h(\sigma)$, $K_h(\sigma) \subseteq J_h(\sigma)$) обмінюється з деякою підмножиною завдань з пристрою s (позначимо цю підмножину як $L_s(\sigma)$, $L_s(\sigma) \subseteq J_s(\sigma)$). Позначимо через θ різницю між сумами еталонних тривалостей завдань, які приймають участь в обміні:

$$\theta = \sum_{j \in K_h(\sigma)} p_j - \sum_{j \in L_s(\sigma)} p_j .$$

Далі такий обмін завданнями між підмножинами $J_h(\sigma)$ та $J_s(\sigma)$будемо називати перестановкою.

Нехай σ^1 – розклад, який отримано після виконання перестановки. Для того, щоб перестановка призвела до покращення (тобто $\sum_{i=1}^{m} Z_i(\sigma) - \sum_{i=1}^{m} Z_i(\sigma^1) > 0$), необхідно, щоб $\theta > 0$.

В роботі розроблена множина перестановок, які в залежності від умов виконання та наслідків розділено на чотири типи: ***А k***, ***Б k***, ***В k*** та ***Г k***. Перестановки усіх типів, в свою чергу, можна розділити на підтипи в залежності від кількості завдань, які приймають участь в перестановці (потужностей множин $K_h(\sigma)$ та $L_s(\sigma)$).

Таблиця 1 містить узагальнену характеристику та властивості перестановок усіх типів. Ці перестановки покладені в основу розробленої поліноміальної складової ПДС-алгоритму розв'язання задачі.

6. ПДС-алгоритм розв’язання задачі

На основі ознак оптимальності та розробленої множини перестановок побудований алгоритм визначення максимально пізнього моменту початку виконання завдань паралельними пристроями

Табл. 1. Властивості перестановок

Тип перестановки	***Умови виконання перестановки***	***Характеристики результуючого розкладу*** σ^1
А k	$\theta > 0$, $\theta k_h \le Z_h(\sigma)$, $\theta k_s \le E_s(\sigma)$	$Z_h(\sigma^1) = Z_h(\sigma) - \theta k_h$, $E_s(\sigma^1) = E_s(\sigma) - \theta k_s$
Б k	$\theta > 0$, $\theta k_s \le Z_h(\sigma)$, $\theta k_h > E_s(\sigma)$	$Z_h(\sigma^1) = Z_h(\sigma) - \theta k_s$, $Z_s(\sigma^1) = \theta k_h - E_s(\sigma)$
В k	$\theta > 0$, $\theta k_s > Z_h(\sigma)$, $\theta k_h \ge E_s(\sigma)$	$Z_h(\sigma^1) = \theta k_s - Z_h(\sigma)$, $Z_s(\sigma^1) = \theta k_h - E_s(\sigma)$
Г k	$\theta > 0$, $\theta k_s < Z_h(\sigma)$, $\theta k_h < Z_h(\sigma) - Z_s(\sigma)$	$Z_h(\sigma^1) = Z_h(\sigma) - \theta k_s$, $Z_s(\sigma^1) = Z_s(\sigma) + \theta k_h$

різної продуктивності із загальним жорстким директивним терміном в допустимому розкладі.

Опис алгоритму

КРОК 1 Побудувати початковий розклад σ^0

КРОК 2 $\sigma = \sigma^0$. Визначити δ.

КРОК 3 *Перевірити виконання ознак оптимальності*

3.1 ЯКЩО $\delta = 0$ і виконується перша ознака оптимальності

ТО перейти на **КРОК 6** (σ – оптимальний розклад)

ІНАКШЕ перейти на КРОК **4**

3.2 ЯКЩО $\delta > 0$

ТО

3.2.1 Розв'язати допоміжну оптимізаційну задачу.

3.2.2. ЯКЩО виконується друга ознака оптимальності

ТО перейти на **КРОК 6** (σ – оптимальний розклад)

КРОК 4 Визначити пристрій h, якому відповідає максимальне значення виступу: $Z_h(\sigma) = \max\limits_{i \in I_Z(\sigma)} Z_i(\sigma)$.

КРОК 5 Для пристрою h перебираючи усі пристрої $s \in I_E(\sigma)$ виконати перестановку типу $\boldsymbol{A}\,k$, $\boldsymbol{Б}\,k$ або $\boldsymbol{B}\,k$,

ЯКЩО таких перестановок не знайшлось,

ТО

5.1 Для пристрою h перебираючи пристрої $s \in I_0(\sigma) \cup I_Z(\sigma)$ виконати перестановку типу $\boldsymbol{Г}\,k$.

5.2 ЯКЩО таких перестановок не знайшлось,

ТО перейти на **КРОК 6**,

ІНАКШЕ $\sigma = \sigma^1$, перейти на **КРОК 3.**

ІНАКШЕ $\sigma = \sigma^1$, перейти на **КРОК 3.**

КРОК 6 Визначити максимально пізній момент запуску завдань на виконання: $r_{\max} = r(\sigma) = d - \max\limits_i C_i(\sigma)$.

КІНЕЦЬ АЛГОРИТМУ

Складність алгоритму складає $O(n^2 W)$, де $W = \sum\limits_{i=1}^{n} p_i$. Це пояснюється наступним. На кожному кроці значення цільової функції (значення максимального з виступів) зменшується, щонайменше на 1. Значить, в гіршому випадку алгоритм зробить кількість кроків, рівне значенню суми виступів в початковому розкладі σ^0. Величина W являється дуже грубою верхньої оцінкою значення $\sum\limits_{i=1}^{m} \Delta_i(\sigma^0)$. На кожному кроці самою трудомісткою операцією є пошук допустимих перестановок завдань між двома пристроями, яка за умови $|K_h(\sigma)| = |L_s(\sigma)| = 1$ зводиться до аналізу тривалостей всіх пар завдань двох обраних пристроїв, (верхня межа кількості пар завдань на пристрої дорівнює $n(n-1)/2$).

7. Висновок

Досліджені властивості задачі календарного планування виконання завдань із спільним жорстким директивним терміном паралельними пристроями різної продуктивності з метою максимізації моменту запуску пристроїв за умови, що усі завдання не запізнюються. Сформульована допоміжна оптимізаційна задача, за результатами якої визначені достатні умови оптимальності розкладів. Визначено множину перестановок, які дозволяють послідовно покращувати значення критерію. Розроблена поліноміальна складова ПДС-алгоритму розв'язання задачі.

Перелік посилань

1. Павлов О.А., Жданова О.Г., Сперкач М.О. Задача составления допустимого расписания с максимально поздним моментом запуска выполнения идентичными параллельными приборами работ с общим директивным сроком / О.А. Павлов, М.О. Сперкач, О.Г. Жданова // Вісник НТУУ "КПІ". Серія «Інформатика, управління та обчислювальна техніка». – К.: "ВЕК+", 2014. – №61 – С.93-102.
2. Згуровский, М. З. Принятие решений в сетевых системах с ограниченными ресурсами [Текст]: монография / М. З. Згуровский, А. А. Павлов. – К.: Наукова думка, 2010.– 573 с.

УДК 004.056.5

МАРКОВСЬКИЙ О.П.,
ТЕРЕЩЕНКО С.Ю.

МЕТОД КОРЕКЦІЇ ПОДВІЙНИХ ПОМИЛОК В КАНАЛАХ ПЕРЕДАЧІ ЦИФРОВИХ ДАНИХ ЗІ СПЕКТРАЛЬНОЮ МОДУЛЯЦІЄЮ

Стаття присвячена проблемі підвищення ефективності виправлення двократних помилок передачі даних в каналах зі спектральною модуляцією за рахунок спрощення та прискорення обчислень, пов'язаних з корекцією помилок.

В роботі запропоновано, теоретично обґрунтовано і досліджено підхід до корекції багатократних помилок в каналах зі спектральною модуляцією на основі позиційних корегуючих сум, що дозволяє визначати позиції спотворених символів та вектори їх спотворення без перебору всіх символів блоку. Детально представлено математичну ідею методу та процедури виявлення та корекції помилок. Використання процедури корекції помилок ілюструється прикладами. Наведено теоретичні та експериментальні оцінки ефективності запропонованого методу.

The paper is dedicated to the efficiency improving of double error correction of data transmission in channels with spectral modulation by simplifying and accelerating the computations associated with error correction.

The paper presents a theoretically grounded and investigated approach of double errors correction in channel with spectral modulation based on positional correcting sums that allows to determine the positions of distorted symbols and their distortion vectors without enumeration through all the symbols of block. The method mathematical basis and procedure of errors detecting and correcting are presented in detail. The use of the procedure error correction is illustrated via the thorough presentation of an example of erroneous data transmission. The theoretical and experimental estimations of effectiveness of proposed method are given.

Вступ

Розвиток розподілених комп'ютерних систем в останнє десятиліття стимулювався, головним чином, потужним прогресом засобів передачі даних між їх компонентами. Чільне місце в сучасних технологій передачі даних в комп'ютерних мережах та розподілених системах займають канали передачі зі спектральною модуляцією. Цифрові дані в таких каналах передаються посимвольно і на фізичному рівні передача символу співвідноситься з відповідною стрибкоподібною зміною фази та амплітуди несучого синусоідального сигналу [1].

Продуктивність розподіленої обробки даних значною мірою визначається швидкістю обміну даними між компонентами комп'ютерних систем. Тому, в найближчій перспективні, швидкість передачі даних буде зростати, в тому числі і за рахунок зменшення надійності, маючи на увазі те, що воно, в принципі, може бути скомпенсоване використанням більш потужних методів виправлення помилок [2].

Відповідно, зростання швидкості обміну цифровими даними між компонентами комп'ютерних систем диктує необхідність вдосконалення існуючих і створення нових методів та засобів виправлення помилок, що виникають при передачі цифрових даних.

Таким чином, наукова задача прискорення підвищення ефективності корекції помилок даних в каналах передачі даних зі спектральною модуляцією з огляду на сучасний стан розвитку технологій комп'ютерної обробки інформації є актуальною.

Аналіз існуючих способів корекції помилок в каналах зі спектральною модуляцією

Вважаючи на практичну важливість проблеми виправлення помилок передачі даних в бездротових каналах, на які вливають зовнішні електромагнітні завади, до теперішнього часу створено ряд методів корекції помилок передачі символів [3].

Ефективність засобів корекції помилок характеризується через три групи критеріїв:

– функціональні, що характеризують здатність вирішувати задачу виявлення та виправлення помилок;

– обчислювальні, що характеризують реалізацію корекції на програмному та апаратному рівнях;

– інформаційні, що характеризують рівень інформаційної надлишковості для виправлення помилок.

До функціональних критеріїв відносяться:

1) кратність помилок, що можуть бути гарантовано виправлені без повторної передачі;

2) ймовірності виправлення помилок кратності, більшої за гарантовану;

3) кратність помилок, які гарантовано виявляються.

До розряду обчислювальних критеріїв відносяться:

1) обчислювальна та часова складність обчислювальних процедур виявлення та корекції помилок,

2) часова складність обчислювальних процедур виявлення та корекції помилок,

3) складність апаратних засобів корекції помилок.

В якості інформаційного критерію ефективності засобів корекції помилок найчастіше виступає кількість контрольних розрядів, що передаються разом з блоком даних.

Динаміка розвитку технологій передачі даних має наслідком зміщення акцентів значимості наведених критеріїв ефективності методів контролю помилок. Багатократне підвищення швидкості передачі зумовлює зростання значимості оцінки обчислювальної складності операцій декодування, виявлення помилок та їх корекції, яка визначає можливість виконання цих операцій в темпі передачі. Цей же чинник знижує значимість, як критерію ефективності, числа контрольних розрядів, що додатково передаються.

Стосовно каналів передачі даних зі спектральною модуляцією засоби корекції помилок мають локалізувати спотворені при передачі дані, а також визначити вектори спотворення кожного із пошкоджений символів.

Якщо вважати, що інформаційний блок B містить $n=2^k-1$ m-розрядних канальних символів $B=\{X_1, X_2,...,X_n\}$, кожен j-тий ($j\in\{1,..,n\}$ з яких X_j передається одним канальним сигналом і складається з m бітів: $X_j=\{x_{j1},x_{j2},...,x_{jm}\}$, $\forall i\in\{1,..,m\}$: $x_{ji}\in\{0,1\}$, то теоретичний мінімум k_m кількості контрольних розрядів, потрібних для виправлення h спотворених символів становить:

$$k_m = h\cdot(\log_2 n + \log_2 m) \qquad (1)$$

Класичним засобом виправлення багатократних помилок в каналах зі спектральною модуляцією є корегуючі коди Ріда-Соломона [4]. Ці коди являють собою недвійкові циклічні коди, що здатні виправляти довільку кількість h спотворених при передачі символів з використанням $2\cdot h$ контрольних символів. Використання кодів накладає обмеження на довжину блоку:

$$n \le 2^m - 1 \qquad (2)$$

Таким чином, при урахуванням обмеження (2), кількість контрольних символів в кодах Ріда-Соломона відповідає теоретичному мінімуму.

Найбільш суттєвим недоліком кодів Ріда-Соломона вважається [3] висока обчислювальна та часова складність. Це зумовлено тим, що для розв'язання системи нелінійних рівнянь на полях Галуа, до якого в математичному сенсі, зводиться локалізація помилок та визначення векторів спотворень, використовується авторегресійна технологія, яка фактично реалізується перебором всіх n символів блоку.

Час T_0 виконання виявлення помилок для кодів Ріда-Соломона визначається формулою:

$$T_o = 2\cdot h\cdot n\cdot(t_m + t_{XOR}), \qquad (3)$$

де t_m – час виконання множення на полях Галуа, t_{XOR} – час виконання операції додавання за модулем 2. Враховуючи, що операція множення складається з m циклів зсуву та додавання за модулем 2, обчислювальна складність виявлення помилок складає $O(8\cdot h\cdot n\cdot m)$.

Процес корекції помилок включає розв'язання двох систем з h лінійних рівнянь та обчислення синдрому наявності помилки для кожного з символів блоку, так, що загальний час корекції оцінюється формулою [3]:

$$T_s = (n\cdot h + 8\cdot h^2)\cdot(t_m + t_{XOR}) \qquad (4)$$

Відповідно, обчислювальна складність операцій, пов'язаних з корекцією помилок становить $O(n\cdot m\cdot h^2)$.

В сучасних умовах збільшення швидкості передачі, розрядності символів та довжини блоку, використання кодів Ріда-Соломона не забезпечує для багатьох важливих для практики застосувань можливість корекції в темпі передачі даних.

Ціллю дослідження є розробка методу прискореної корекції помилок в каналах зі спектральною модуляцією та спрощення апаратної реалізації корекції.

Метод корекції двох помилок з використанням позиційних корегуючих кодів

Основним резервом досягнення поставленої мети є збільшення кількості контрольних розрядів, що передаються приймачу разом з інформаційним блоком. В роботі [6] була започаткована ідея використання для корекції багатократних помилок позиційних корегуючих кодів. Суть цієї ідеї полягає в тому, що контрольний код складається з сум позицій підмножини інформаційних об'єктів так, що при виникненні помилки змінюється підмножина позицій об'єктів. Відповідно, різниці контрольних сум на передавачеві та приймачеві утворюють систему лінійних рівнянь, компонентами якої є позиції спотворених помилками об'єктів.

Ідея може бути конкретизована для каналів передачі даних зі спектральною модуляцією.

Для виявлення багатократних помилок і гарантованого виправлення двох помилок пропонується метод, яких передбачає передачу разом з інформаційним блоком контрольного блоку, який складається з $2\cdot k+1$ компонентів: $C=\{C_0,C_1,C_2,...,C_k, S_1,S_2,...,S_k\}$.

Нульову компоненту контрольного коду C_0 пропонується обчислювати у вигляді суми за модулем 2 всіх символів блоку:

$$C_0 = X_1 \oplus X_2 \oplus ... \oplus X_n \qquad (5)$$

Кожен j-тий номер символу X_j блоку B може бути представлений у вигляді двійкового k-розрядного коду: $j = j_1 + j_2\cdot 2 + j_3\cdot 2^2 +...+j_k\cdot 2^{k-1}$. Друга компонента C_1 у являє собою суму за модулем 2 тих символів блоку B, молодший розряд номерів котрих дорівнює одиниці:

$$C_1 = \bigoplus_{j=1}^{n} j_1 \cdot X_j \qquad (6)$$

Аналогічно, кожна l-та компонента контрольного коду C_l, $l = 1,...,k$ обчислюється як сума по модулю 2 лише тих символів блоку даних B, l-тий розряд номери яких дорівнює одиниці:

$$C_l = \bigoplus_{j=1}^{n} j_l \cdot X_j \qquad (7)$$

Компоненти контрольного коду $S_1,S_2,...,S_k$ формуються з використанням лінійної згортки.

Лінійна згортка $\lambda(Y)$ m-розрядного коду $Y=\{y_1,y_2,...,y_m\}$, де $\forall j\in\{1,2,...,m\}$: $y_j\in\{0,1\}$ являє собою h-розрядний ($h=\lfloor \log_2 m \rfloor$) код $Z=\{z_1,z_2,...,z_h\}$, причому, молодший розряд z_1 коду згортки формується як сума за модулем два всіх двійкових розрядів коду Y:

$$z_1 = \bigoplus_{j=1}^{m} y_j \qquad (8)$$

Другий розряд z_2 коду лінійної згортки Z формується як сума за модулем два всіх розрядів коду Y, номери яких мають одиницю в другому розряді, третій розряд z_3 коду згортки являє собою суму за модулем 2 всіх тих розрядів коду Y, номери яких мають одиницю в третьому розряді. Загалом, l-тий розряд z_l, $l\in\{2,3,...,h\}$ коду лінійної згортки формується згідно наступної формули:

$$z_l = \bigoplus_{j=1}^{m} y_j \cdot ((j \bmod 2^{l-1}) / 2^{l-2}) \qquad (9)$$

Наприклад, якщо $Y=\{1,0,1,1,0,1,0,1\}$, то $z_1 = 1\oplus0\oplus1\oplus1\oplus0\oplus1\oplus0\oplus1=1$, $z_2 = 0\oplus1\oplus1\oplus1 = 1$, $z_3 = 0\oplus1\oplus0\oplus1 = 0$, $Z=\lambda(Y) = \{1,1,0\}$.

Головна властивість коду лінійної згортки $\lambda(Y)$ полягає в тому, щоб при будь-якій зміні коду Y зазнає змін його лінійна згортка.

Кожну l-ту компоненту S_l контрольного коду, $l=1,...,k$ пропонується обчислювати як суму за модулем 2 поліноміальних добутків лінійних згорток символів блоку даних B, l-тий розряд порядкових номерів в блоці яких дорівнює одиниці, на їх номери:

$$S_j = \bigoplus_{j=1}^{m} (\lambda(X_j) \otimes j) \cdot j_l \qquad (10)$$

Запропонований спосіб формування контрольного коду може бути ілюстровано наступним прикладом. Нехай передається блок з 7-ми символів, що нумеруються від одиниці до семи: B = {0101, 1011, 1100, 1001, 1111, 1101, 0011}. Для цього прикладу розрядність символів m=4, кількість символів в блоці $n=2^3-1=7$, розрядність порядкового номеру символів в блоці k=3.

Нульова компонента контрольного коду C_0 згідно (5) обчислюється як логічна сума : C_0 = 0101⊕1011⊕1100⊕1001⊕1111⊕1101⊕ 0011 = 1010. Компонента C_1 обчислюється згідно (6) як логічна сума тих символів, молодший розряд номерів яких дорівнює одиниці: $C_1 = X_1\oplus X_3\oplus X_5\oplus X_7$ = 0101⊕ 1100⊕1111⊕0011 = 0101. Аналогічно, компонента C_2 обчислюється як логічна сума тих символів другий розряд номеру яких дорівнює одиниці: $C_2 = X_2\oplus X_3\oplus X_6\oplus X_7$ = 1011⊕ 1100⊕1101⊕0011 = 1001. Код компоненти C_3 являє собою суму за модулем 2 символів, старший розряд номеру яких дорівнює одиниці: $C_3 = X_4\oplus X_5\oplus X_6\oplus X_7$ = 1001⊕ 1111⊕1101⊕0011 = 1000. Лінійні згортки символів блоку утворюють множину:

{00,01,10,10,00,11,10}. Відповідно з (10) код S_1 формується як сума поліноміальних добутків згорток символів, молодший розряд номеру яких дорівнює одиниці на самі ці номери, тобто: $S_1=\lambda(X_1)\otimes 1 \oplus \lambda(X_3)\otimes 3 \oplus \lambda(X_5)\otimes 5 \oplus \lambda(X_7)\otimes 7 = 00\otimes 001 \oplus 10\otimes 011 \oplus 00\otimes 101 \oplus 10\otimes 111 = 0110 \oplus 1110 = 1001$. Аналогічно, $S_2=\lambda(X_2)\otimes 2 \oplus \lambda(X_3)\otimes 3 \oplus \lambda(X_6)\otimes 6 \oplus \lambda(X_7)\otimes 7 = 01\otimes 010 \oplus 10\otimes 011 \oplus 11\otimes 110 \oplus 10\otimes 111 = 010 \oplus 110 \oplus 1010 \oplus 1110 =0$; $S_3=\lambda(X_4)\otimes 4 \oplus \lambda(X_5)\otimes 5 \oplus \lambda(X_6)\otimes 6 \oplus \lambda(X_7)\otimes 7 = 10\otimes 100 \oplus 00\otimes 101 \oplus 11\otimes 110 \oplus 10\otimes 111= 1000 \oplus 1010 \oplus 1110 = 1100$.

Позначимо компоненти контрольного коду, обчислені на стороні передавача як $C_{S,0}$, $C_{S,1}$, $C_{S2},...,C_{S,k}$, $S_{S,0}$, $S_{S,1}$, $S_{S2},...,S_{S,k}$, а компоненти контрольного коду, обчислені приймачем за прийнятим блоком, як $C_{R,0}$, $C_{R,1}$, $C_{R2},...,C_{R,k}$, $S_{R,0}$, $S_{R,1}$, $S_{R2},...,S_{R,k}$. На стороні приймача обчислюються різниці контрольних кодів:

$$\begin{aligned} &\Delta_0 = C_{S,0} \oplus C_{R,j} \\ &\forall j = 1,..,k : \Delta_j = C_{S,j} \oplus C_{R,j} \\ &\delta_j = S_{S,j} \oplus S_{R,j} \end{aligned} \qquad (11)$$

Очевидно, що в разі, коли жоден з символів блоку не зазнав спотворення при передачі, всі різниці дорівнюють нулю: $\forall j=0,...,k$: $\Delta_j=0$.

В разі спотворення лише одного q-го символу блоку, $q\in\{1,2,...,n\}$, на стороні приймача код цього символу $X_{R,q}$ відрізняється від відповідного символу на передавачеві $X_{S,q}$, так, що вектор спотворення $\Delta X_q = X_{R,q} \oplus X_{S,q} \neq 0$. Очевидно, що коди різниць в разі спотворення одного символу визначаються наступним чином: $\Delta_0 = \Delta X_q$, $\forall j\in\{1,2,..,k\}$: $\Delta_j = 0$, якщо j-тий розряд q_j номеру q дорівнює нулю: $q_j = 0$ і $\Delta_j = \Delta X_q$, якщо j-тий розряд q_j номеру q дорівнює одиниці: $q_j = 1$. Таким чином, за кодами перших k+1 різниць Δ_0 $\Delta_1,...,\Delta_k$ однозначно визначаються код вектору спотворення символу: $\Delta X_q = \Delta_0$ та розряди $q_1,q_2,...,q_k$ номеру q спотвореного при передачі символу: $\forall j\in\{1,2,..,k\}$: $q_j = 0$ якщо $\Delta_j=0$, $q_j = 1$, якщо $\Delta_j \neq 0$. Відповідно, виправлення спотвореного символу виконується у вигляді: $X_q = X_{R,q}\oplus\Delta X_q$. Сама ситуація виникнення однократної помилки визначається на умовою: $\forall j\in\{1,2,..,k\}$: $\Delta_j\in\{0,\Delta_0\}$.

В разі спотворення при передачі пари символів, що мають в блоці порядкові номери q та p, причому $p < q$, постає задача визначення як самих позицій q і p, так і векторів спотворень обох символів ΔX_q та ΔX_p .

Якщо вектори спотворень обох пошкоджених при передачі символів відрізняються, тобто $\Delta X_q \neq \Delta X_p$, ця задача вирішується доволі просто. Код різниці нульових компонентів контрольного коду в цій ситуації дорівнює в цій ситуації значення: $\Delta_0 = \Delta X_p \oplus \Delta X_q \neq 0$. Кожна j-та різниця, $j\in\{1,2,..,k\}$, k компонентів контрольного коду може приймати лише 4 значення:

– $\Delta_j = 0$, якщо j-ті розряди q_j і p_j номерів q та p дорівнюють нулю: $q_j = p_j = 0$;

– $\Delta_j = \Delta_0 = \Delta X_p \oplus \Delta X_q$, якщо j-ті розряди q_j і p_j номерів q та p дорівнюють одиниці: $q_j = p_j = 1$;

– $\Delta_j = \Delta X_p$ якщо j-тий розряд q_j номеру q дорівнює нулю, а j-тий розряд p_j номеру p дорівнює одиниці: $q_j = 0$, $p_j = 1$;

– $\Delta_j = \Delta X_q$ якщо j-тий розряд q_j номеру q дорівнює одиниці, а j-тий розряд p_j номеру p дорівнює нулю: $q_j = 0$, $p_j = 1$.

Відповідно, процедура відновлення номерів q та p пошкоджених символів представлена у вигляді наступної послідовності дій:

1) Встановити індекс j поточного розряду номеру в k: $j = k$, прапорець f виявлення першого розряду, в якому різняться коди q та p встановлюється в нуль: f=0.

2) Якщо $\Delta_j = 0$, то j-ті розряди p та q встановлюються в нулі, $p_j = 0$, $q_j = 0$. Перехід на п.6.

3) Якщо $\Delta_j = \Delta_0$, то j-ті розряди p та q встановлюються в 1: $p_j = 1$, $q_j = 1$. Перехід на п.6.

4) Якщо $\Delta_j \neq\Delta_0$, $\Delta_j \neq 0$ і $f = 0$, то j-тий розряд q встановлюється в одиницю, j-тий розряд p встановлюється в нуль, $q_j = 1$, $p_j = 0$, $\Delta X_q = \Delta_j$ і f=1. Перехід на п.6.

5) Якщо $\Delta_j \neq\Delta_0$,$\Delta_j \neq 0$ і $f = 1$, то, якщо $\Delta_j =\Delta X_q$, j-тий розряд q встановлюється в одиницю, j-тий розряд p встановлюється в нуль, $q_j = 1$, $p_j = 0$, інакше j-тий розряд p встановлюється в одиницю, а j-тий розряд q встановлюється в нуль, $p_j = 1$, $q_j = 0$.

6) Декремент індексу поточного розряду номеру $j = j – 1$. Якщо $j > 0$, то повернення на виконання п.2.

7) Визначення вектору $\Delta X_p = \Delta_0 \oplus \Delta X_q$. Корекція пошкоджених символів: $X_p= X_{R,p}\oplus\Delta X_p$; $X_q= X_{R,q}\oplus\Delta X_q$. Кінець.

Конструктивність розробленої процедури визначається тим, що в силу того, що номери q та p обов’язково розрізняються, причому $q>p$, то в старшому розряді, в якому ці коди відмінні відповідний розряд q дорівнює нулю, а одноіменний розряд p дорівнює одиниці. Це значить, що обов’язково виконується п.4 наведеної вище

процедури, який визначення вектор спотворення q-го символу – ΔX_q.

Наведена процедура корекції двох спотворених при передачі символів блоку за умови, що вектори їх спотворення відрізняються може бути ілюстрована наступним чином. Якщо припустити, що в рамках наведеного вище прикладу спотворені 4-й та 6-й символи (q=6, p=4), причому вектор першого із спотворених символів ΔX_p = 1100, а другого ΔX_q = 1001, то прийнятий блок B={ 0101, 1011, 1100, **0101**, 1111, **0100**, 0011}. Значення обчислених на стороні приймача перших k+1 компонентів контрольного коду становлять: $C_{R,0}$= 0101⊕1011⊕1100⊕**0101**⊕1111⊕**0100**⊕0011=1111, $C_{R,1}$=0101⊕ 1100⊕1111⊕0011 = 0101; $C_{R,2}$=1011⊕1100⊕**0100**⊕0011 = 0000; $C_{R,3}$= **0101**⊕ 1111⊕**0100**⊕0011 = 1101. Відповідно, Δ_0= 0101; Δ_1=0000; Δ_2=1001; Δ_3=0101.

Згідно п.1 наведеної вище процедури, індекс j=3, f=0. Так як Δ_3=0101=Δ_0, то згідно з п. 3 процедури старші розряди номерів пошкоджених символів дорівнюють одиниці: p_3=1 і q_3=1. При j=2 виконуються умови п.4 процедури: $\Delta_2 \neq \Delta_0$, $\Delta_2 \neq 0$ і f = 0: відповідно 2-тий розряд q встановлюється в одиницю, 2-тий розряд p встановлюється в нуль, q_j = 1, p_j = 0, $\Delta X_q = \Delta_2$ = 1001 і f=1. При j=1 виконується умова п.2 процедури: Δ_1=0, відповідно згідно з п.2 p_1=1 і q_1=1. Таким чином, позиції пошкоджених при передачі визначено: q=6, p=4. Останнім п.7 визначається вектор спотворення 4-го символу: $\Delta X_4 = \Delta_0 \oplus \Delta X_6$ = 0101⊕1001=1100 та відновлюються символи: $X_4 = X_{R,4} \oplus \Delta X_4$= 0101⊕1100 =1001, $X_6 = X_{R,6} \oplus \Delta X_6$= 0100⊕1001 = 1101.

Якщо вектори спотворень обох пошкоджених символів однакові, тобто $\Delta X_p = \Delta X_q$, то Δ_0=0 і для корекції використовуються різниці $\delta_1,\delta_2,...,\delta_k$ компонентів $S_1,S_2,...,S_k$. Процедура корекція пари символів з порядковими номерами p та q, причому $q>p$ полягає в виконанні наступної послідовності дій:

1) Встановити індекс j поточного розряду номеру в k: $j = k$, прапорець f виявлення першого розряду, в якому різняться коди q та p встановлюється в нуль: f=0.

2) Якщо $\delta_j = 0$, то j-ті розряди p та q встановлюються в нулі, $p_j = 0$, $q_j = 0$. Перехід на п.6.

3) Якщо $\delta_j \neq 0$, а $\Delta_j = 0$, то j-ті розряди p та q встановлюються в одиниці, $p_j = 1$, $q_j = 1$. Перехід на п.6.

4) Якщо $\delta_j \neq 0$, $\Delta_j \neq 0$ і $f = 0$, то j-тий розряд q встановлюється в одиницю, j-тий розряд p встановлюється в нуль, $q_j = 1$, $p_j = 0$, $\Delta X_q = \Delta_j$, збереження значення δ_j в змінній d: $d = \delta_j$, установка прапорця f=1 Перехід на п.6.

5) Якщо $\delta_j \neq 0$, $\Delta_j \neq 0$ і $f = 1$, то, якщо $\delta_j = d$, j-тий розряд q встановлюється в одиницю, j-тий розряд p – в нуль, $q_j = 1$, $p_j = 0$, інакше j-тий розряд p встановлюється в одиницю, а j-тий розряд q – в нуль, $p_j = 1$, $q_j = 0$.

6) Декремент індексу поточного розряду номеру $j = j - 1$. Якщо $j > 0$, то перехід на п.2.

7) Визначення вектору $\Delta X_p = \Delta X_q$. Корекція пошкоджених символів: $X_p = X_{R,p} \oplus \Delta X_p$; $X_q = X_{R,q} \oplus \Delta X_q$. Кінець.

Процедура корекції двох однаково спотворених символів блоку може бути ілюстрована наступним чином. Якщо припустити, що в рамках наведеного вище прикладу спотворені 3-й та 6-й символи (q=6, p=3), причому вектори спотворення обох символів обох символів дорівнюють $\Delta X_p = \Delta X_q$ =1001, то прийнятий блок має вигляд B={ 0101,1011,**0101**,1001,1111, **0100**, 0011}. Значення обчислених на стороні приймача 2·k+1 компонентів контрольного коду становлять: $C_{R,0}$= 1010, $C_{R,1}$= 1101; $C_{R,2}$=1001; $C_{R,3}$= 0001, $S_{R,1}$=1111, $S_{R,2}$= 1010, $S_{R,3}$= 0. Відповідно, Δ_0= 0; Δ_1=1001; Δ_2=0; Δ_3=1001, δ_1 = 110; δ_2 =1010; δ_3 = 1100.

Згідно п.1 наведеної вище процедури, індекс j=3, f=0. Так як δ_3 = 1100≠0, Δ_3=1001≠0 і f=0, то згідно з п. 4 процедури старший розряд більшого номеру q дорівнює одиниці: q_3=1, старший розряд меншого номеру p дорівнює нулю: p_3=1; в змінній d фіксується значення δ_3 = 1100, f=1. При j=2 виконуються умови п.3 процедури: Δ_2=0, $\delta_2 \neq 0$: відповідно p_2=1 і q_2=1. При j=1 виконується умова п.5 процедури: δ_1 = 110 ≠ 0, Δ_1 = 1001 ≠ 0 і f = 1, при цьому δ_1 = 110 ≠ d=1100, відповідно молодший розряд p встановлюється в одиницю, а молодший розряд q встановлюється в нуль, p_j = 1, q_j=0. Таким чином, позиції пошкоджених при передачі визначено: q=6, p=3. Останнім п.7 визначається вектор спотворення 3-го символу: $\Delta X_3 = \Delta X_6$ = 1001 та відновлюються спотворені символи: $X_3 = X_{R,3} \oplus \Delta X_3$= 0101⊕1001 = 1100, $X_6 = X_{R,6} \oplus \Delta X_6$= 0100⊕1001 = 1101.

Аналіз ефективності

Запропонований метод корекції двократних помилок передачі даних в каналах зі спектральною модуляцією дозволяє виправляти до 2-х помилок, що відповідає сучасним вимогам [2].

Головна перевага методу у порівнянні з кодами Ріда-Соломона полягає в суттєвому зменшенні обчислювальної складності процесу корекції двох символів. Як слідує з наведеного алгоритму корекції, кількість операцій для локалізації спотворених символів не перевищує розрядності номеру, тобто обчислювальна складність корекції становить $O(\log_2 n)$.

Згідно з формулою (4) обчислювальна складність процедури корекції для кодів Ріда-Соломона становить $O(n \cdot m \cdot h^2)$. Враховуючи, що кількість помилок, що можуть бути скорегованими h=2, і, відповідно до (2) $m=\log_2 n$, то зазначена оцінка складності може бути представлена у вигляді $O(n \cdot \log_2 n \cdot 4)$. Таким чином, запропонований метод дозволяє спростити і, відповідно, прискорити процедуру корекції в $4 \cdot n$. Наприклад для n=1024, теоретична оцінка прискорення корекції при використанні запропонованого методу становить 4096. Проведене експериментальне дослідження показало, що на практиці виграш у швидкодії більш суттєвим, оскільки оцінка (4) не враховує час розв'язання двох систем лінійних рівнянь на полях Галуа.

Ще більший виграш у часі корекції досягається при апаратній реалізації, оскільки пропонований метод використовує гранично прості процедури та операції.

Крім того, запропонований метод не накладає обмежень на довжину блоку даних.

Загальна довжина l контрольного коду C та S обчислюється як сума розрядів всіх k її компонентів. Розрядність перших k компонентів коду C дорівнює розрядності символу – m, компонентів S – $\log_2 k + \log_2 m - 1$. Загальна довжина L контрольного коду дорівнює:

$$L = k \cdot (m + \log_2 m + k - 1) + m \qquad (12)$$

Наприклад, для типових для практики значень розрядності символу m = 8 (QAM-256) і довжини блоку n =1024, довжина контрольного коду становить L =208 бітів, або 26 символів, що становить 2.5% об'єму блоку.

З (12) слідує, що кількість контрольних символів помітно перевищує аналогічний показник для коду Ріда-Соломона. Проте для сучасних швидкостей передачі даних, які вимірюється Мегабайтами за секунду, зменшення на 2-4 порядки часу корекції помилок вагоміше за передачу десятка додаткових символів.

Висновки

В результаті проведених досліджень запропоновано метод прискореної корекції помилок в каналах зі спектральною модуляцією, кратність не більше 2-х.

За рахунок використання більшої кількості контрольних розрядів, в $4 \cdot n$ разів зменшено обчислювальну складність корекції в порівнянні з кодами Ріда-Соломона, що дозволяє на порядки прискорити процес корекції та спростити апаратну реалізацію.

Метод орієнтовано для швидкісних каналів передачі даних між компонентами розподілених систем комп'ютерного управління, що працюють у режимі реального часу.

Список посилань

1. Скляр Б. Цифровая связь. Теоретические основы и практическое применение. / Б. Скляр // М.: Издательский дом "Вильямс". – 2004. – 1104с.
2. Ирвин Дж. Передача данных в сетях: инженерный подход. / Дж. Ирвин, Д. Харль // СПб.: БХВ-Петербург, 2002. – 448 с.
3. Морелос-Сарагоса Р. Искусство помехоустойчивого кодирования. Методы, алгоритмы, применение / Р. Морелос-Сарагоса. – М.: Техносфера, 2005. – 319с.
4. Reed I.S. Polynomial codes over certain finite fields / I.S. Reed, G. Solomon // Journal of the Society for Industrial and Applied Mathematics. – 1960. – № 8(2). – Pp. 300-304.
5. Wickers S.B. Reed-Solomon Codes and Their Applications / S.B.Wickers, V.K. Bhargava – IEEE Press. Piscataway, New Jersey. – 1983. – P.433.
6. Марковський О.П. Метод виправлення трьохкратних помилок передачі даних в двійкових симетричних каналах / О.П. Марковський, С.Ю. Терещенко, О.І. Федоречко О.І. // Вісник Національного технічного університету України "КПІ" Інформатика, управління та обчислювальна техніка: Зб. наук. пр. – Київ: ВЕК+. – 2014. – № 60. – С.33-40.

УДК 681.518

РОЛИК А.И.,
ГАЛУШКО Д.А.,
БАРНА В.В.,
ТОМАЩУК А.В.,
ЯСОЧКА М.В.

ОЦЕНКА КАЧЕСТВА ПРЕДОСТАВЛЕНИЯ МУЛЬТИМЕДИЙНЫХ СЕРВИСОВ С ИСПОЛЬЗОВАНИЕМ НЕЙРОСЕТЕВОГО КЛАССИФИКАТОРА

Предложен метод оценки качества предоставления мультимедийных сервисов. Метод использует технологию DPI для получения значений параметров, оказывающих влияние на качество предоставления сервисов. Для оценки качества предоставления мультимедийных сервисов используется нейросетевой классификатор РБФ типа. Продемонстрирована реализация метода на примере услуги VoIP.

Maintaining the high quality of services is necessary in order to be competitive for internet provider. In this article the method for determine the quality of multimedia services is proposed. Method uses DPI to obtain the values of the parameters affecting the quality of multimedia services. The RBF neural network classifier is used for evaluating the quality of multimedia services. Implementation of the method is demonstrated on the example of service VoIP.

Введение

Жесткая конкурентная борьба на рынке предоставления телекоммуникационных услуг вынуждает операторов телекоммуникационных сервисов (ОТС) постоянно вкладывать существенные средства в поддержание высокого качества предоставляемых ими услуг. Особенно требовательными к качеству являются услуги, связанные с передачей мультимедийной информации, такие как VoIP, IPTV и др. Это объясняется тем, что при оценке качества мультимедийных сервисов большой вес имеет субъективная составляющая, определяемая мнением пользователей. Для управления уровнем услуг ОТС используют различные системы управления инфраструктурой (СУИ). Основными задачами СУИ являются: оценка текущего состояния телекоммуникационной сети, оценка фактического уровня услуг и последующее управляющее воздействие на сетевую инфраструктуру для восстановления высокого качества услуг в случае его ухудшения. Поэтому данная работа, посвященная разработке метода оценки качества предоставления мультимедийных сервисов, является актуальной.

Анализ проблемы контроля качества предоставления мультимедийных сервисов

Конкурентоспособность операторов существенно зависит от стабильности поддержания высокого уровня предоставления услуг по передаче мультимедийной информации и от величины затрат на поддержание этого уровня. Значения показателей качества предоставления услуг согласовываются с пользователями и фиксируются в соответствующем соглашении об уровне обслуживания – SLA. При этом каждый ОТС организовывает свою работу с учетом следующего критерия:

$$C\big|_{q_i=const} \to \min,\ \forall i=\overline{1,N}, \qquad (1)$$

где C – затраты оператора на поддержание согласованного уровня услуг, q_i — согласованное качество предоставления отдельной i-й услуги, $i=\overline{1,N}$.

Поскольку поддержание $q_i = const, i=\overline{1,N}$ – это сложная техническая и организационная задача, то ОТС ставят перед собой задачу не по поддержанию качества услуг на стабильно высоком уровне, а осуществляют оперативное управление сетевой инфраструктурой таким образом, чтобы обеспечивалось

$$q_{k,i} - q^*_{k,i} \to 0, \forall k, i, \qquad (2)$$

где $q_{k,j}$ и $q^*_{k,j}$, $k=\overline{1,M}$ – соответственно, целевое и фактическое значения k-го показателя качества i-й услуги.

Для выполнения условия (2) может использоваться предложенный в [2] декомпозиционно-компенсационный подход к управлению уровнем услуг, предполагающий выделение дополнительных сетевых и вычислительных ресурсов сервисам, качество которых ниже требуемого.

Целевые значения $q_{k,i}, \forall k,i$ фиксируются в SLA. За нарушение SLA ОТС выплачивают штрафы, а частые и существенные нарушения SLA приводят к потере клиентов. Что же касается определения фактических значений показателей качества $q^*_{k,i} \to 0, \forall k,i$, то ОТС стремятся решать эту достаточно сложную и трудоемкую техническую задачу с минимальными затратами.

В [1] решается задача оценки ожидаемого качества предоставления мультимедийных услуг на основании анализа значений параметров трафика мультимедийной информации, проходящего через узлы сети. При этом производится определение значений совокупности параметров трафика, оказывающих влияние на качество предоставления мультимедийных сервисов, с последующим применением методов теории распознавания образов для оценки уровня этих сервисов.

Предложенный в [1] метод, отличаясь простотой и низкими затратами на его реализацию, позволяет оценить предполагаемое качество сервиса у группы абонентов, подключенных к рассматриваемому узлу сети и не позволяет оценить текущее качество для каждого абонента в отдельности. Кроме того, метод отличается существенной погрешностью. Это не позволяет использовать предложенный метод для управления уровнем услуг (SLM) относительно отдельного пользователя.

Поэтому в подсистемах SLM, являющихся частью СУИ, особое внимание уделяется методам получения оценок качества сервисов в месте подключения отдельного пользователя. В таком случае для оценки качества мультимедийных сервисов, предполагающих передачу аудио и видео данных, применяются методы, связанные с субъективной и объективной оценкой качества. Субъективные методы основаны на использовании MOS-подобных оценок [3] качества передачи голосовых сообщений или изображения. Субъективная оценка требует участия экспертов, выставляющих оценки и располагающихся в том месте, где производится оценка качества сервисов. Это делает невозможным применение подобных методов при оперативном управлении уровнем услуг, а также для автоматизации процессов мониторинга текущего качества мультимедийных сервисов. Поэтому ОТС при оперативном управлении применяют в СУИ методы объективного оценивания качества мультимедийных сервисов, таких как VoIP или IPTV [4-8].

Автоматизация мониторинга уровня предоставления телекоммуникационных услуг предполагает наличие в СУИ модуля оценки качества сервисов (МОКС), сопоставляющего значения параметров сетевого трафика, показателей качества функционирования сетевой инфраструктуры и других данных, которые можно измерить в сетевом оборудовании с MOS-оценками [1]. При этом неотъемлемой частью МОКС является классификатор, который предварительно обучается с использованием значений параметров $p_{j,k,i}$, $j=\overline{1,L}$, оказывающих влияние на k-й показатель $q^*_{k,j}$ качества i-й услуги s_i, $i=\overline{1,N}$, и MOS-оценок m_l, $l=\overline{1,K}$, где K – количество градаций качества при MOS-оценивании.

Среди множества методов получения значений параметров $p_{j,k,i}$, $j=\overline{1,L}$, $k=\overline{1,M}$, $i=\overline{1,N}$ особый интерес представляют методы, основанные на анализе значений пакетов данных, содержащих мультимедийную информацию. Такую возможность предоставляет технология Deep Packet Inspection (DPI). Повышенный интерес ОТС к применению технологии DPI для решения разнообразных задач в современных СУИ обусловлен тем, что в настоящее время для анализа сетевых пакетов по их содержимому не требуется дорогостоящего оборудования. Интерес авторов к применению технологии DPI для оценки качества предоставления мультимедийных сервисов обусловлен тем, что технология DPI позволяет выделять и анализировать пакеты, содержащие мультимедийные данные и относящиеся к текущему сеансу связи отдельного пользователя. Поэтому целью статьи является разработка метода оценки качества мультимедийных сервисов на основе нейросетевого классификатора, анализирующего значения параметров, полученных с применением технологии DPI.

Суть предлагаемого метода оценки качества мультимедийных сервисов

При контроле качества предоставляемых услуг ОТС выделяет зоны сети и определяет свои возможности по мониторингу параметров сети и оценке качества сервисов в этих зонах, а также полномочия по выдаче управляющих

воздействий на сетевое оборудование для повышения качества услуг. Для каждой из зон определяется набор параметров, на основе которых СУИ может произвести оценку качества предоставляемых услуг. Например, на качество мультимедийных сервисов воздействуют десятки различных факторов. На часть из них оператор может повлиять непосредственно, на часть – лишь косвенно. При этом учитываются рентабельность управляющих воздействий и временны́е ограничения на их выработку и реализацию. Поэтому, как правило, зона, в которой определяются значения параметров и осуществляются управляющие воздействия, ограничивается только сетью передачи трафика, которую контролирует ОТС. Все остальные значения параметров, на которые ОТС не может воздействовать, считаются такими, что находятся в пределах нормы. Для достижения нормативных значений параметров вне зоны контроля ОТС оператор может, например, дать рекомендацию абонентам использовать конкретное оборудование с заведомо известными характеристиками и указать параметры настройки этого оборудования.

Технология DPI позволяет получать значения параметров $p_{j,k,i}$, $j=\overline{1,L}$, $k=\overline{1,M}$, оказывающих существенное влияние на качество предоставления i-й услуги s_i, $i=\overline{1,N}$ и характеризующих условия прохождения пакетов с мультимедийными данными, анализируя значения определенных полей в кадрах данных, проходящих по сети ОТС. Например, для услуги VoIP из множества параметров $p_{j,k,i}$, $j=\overline{1,L}$, $k=\overline{1,M}$, чаще всего используются величина задержки передачи пакетов, потери пакетов и флуктуация задержки передачи пакетов (джиттер). Применение технологии DPI в СУИ для оценки качества предоставления мультимедийных услуг позволяет не только оценивать текущее качество предоставления услуги s_i, $\forall i=\overline{1,N}$ по ходу предоставления услуги, но и получать такую оценку отдельно для каждого пользователя и для каждого сеанса связи этого пользователя.

Суть предлагаемого метода на примере сервиса VoIP заключается в следующем. Поскольку VoIP является услугой реального времени, то наибольшее влияние на качество этой услуги оказывают факторы, связанные с временны́ми параметрами передачи голосовых сообщений (джиттер и задержка передачи голосовых пакетов) и наличия данных как таковых (потери пакетов). В узлах коммутации (УК) и шлюзах на основе технологии DPI осуществляется выделение и последующий анализ пакетов данных, имеющих отношение к каждой отдельной сессии VoIP. Отчеты с информацией о значении джиттера, потерях пакетов и задержке передачи из МЭК передаются в модуль обработки данных, в котором производится определение значений этих трех параметров для каждой отдельной сессии. На основании значений этих параметров определяется MOS-подобная оценка, которая передается в подсистему управления качеством услуги VoIP.

Структура фрагмента распределенной СУИ, осуществляющей мониторинг качества предоставления мультимедийных услуг, представлена на рис. 1.

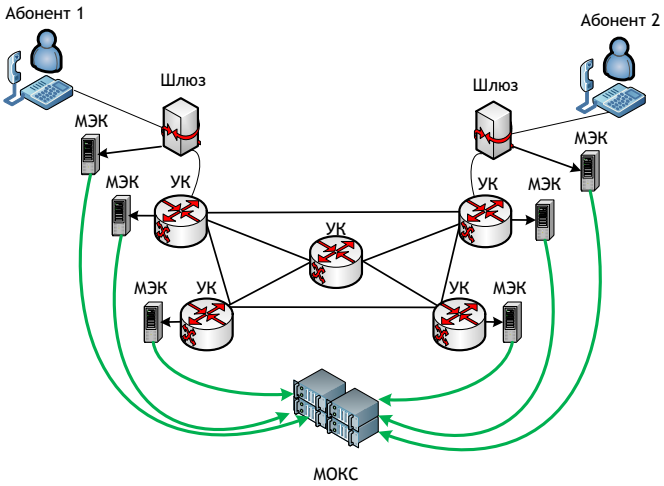


Рис. 1. Структура фрагмента СУИ

Система на рис. 1 работает следующим образом.

К УК и шлюзам подключены модули экстракции данных (МЭК), которые осуществляют захват всего трафика, проходящего через связанные с ними УК и шлюзы. При этом производится зеркалирование трафика, проходящего через УК и шлюзы, на входной интерфейс МЭК. Каждый МЭК, используя технологию DPI, осуществляет выделение пакетов или кадров данных, относящихся к определенному типу трафика и конкретным сессиям. Например, для VoIP это будут пакеты протоколов SIP/SDP, RTP, RTCP. На основании анализа этих пакетов выделяются данные о джиттере, односторонней задержке передачи, потере пакетов или сведения, на основании которых значения этих параметров можно рассчитать в МОКС. Значения

этих параметров могут быть получены на основании анализа полей пакетов протокола RTCP. Пример извлечения данных для анализа качества услуг VoIP из RTCP пакетов приведен на рис. 2.

Рис. 2. Экстракция данных из RTCP пакетов

Если в сети RTCP трафик отсутствует или поля RTCP пакетов не несут необходимой информации, например, по причине нестрогого следования ОТС требованиям соответствующих RFC, то значения параметров определяются на основании анализа пакетов SIP/SDP и RTP.

Полученные данные в виде отчета о звонке в формате JSON из МЭК передаются по сети по сети в МОКС.

МОКС на основании отчетов МЭК выносит оценку относительно качества каждого звонка. Эта оценка передается в подсистему SLM.

МЭК, структура которого приведена на рис. 3, осуществляет захват VoIP трафика, анализ и сбор данных о звонке, а также генерацию и отправку отчетов. Четкое разделение функций МЭК позволяет разрабатывать компоненты МЭК по отдельности, на основании специфицированных интерфейсов с указанием формата и способа обмена между компонентами. Кроме того, такое решение предоставляет возможность простой модификации МЭК для анализа качества других мультимедийных сервисов.

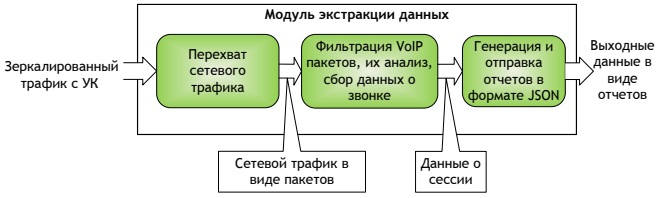


Рис. 3. Структура МЭК

МЭК может быть реализован, например, на основе комбинации программных комплексов Netmap [9] и BlockMon [10], первый из которых осуществляет захват трафика, а BlockMon позволяет реализовать функции МЭК путем композиции программных модулей, логика взаимодействия между которыми задается сценариями в формате xml.

Упрощенный алгоритм работы МЭК можно представить следующим образом:

1. Старт захвата трафика.
2. Трафик отфильтровывается на три группы: SIP, RTP и RTCP.
3. Из данных, содержащихся в полях SIP пакетов, определяются участники сессии, их идентификаторы и адреса. Из данных тела SIP/SDP пакетов выделяются сведения о нюансах обмена медиа-данными.
4. Используя полученные идентификаторы, из голосового трафика выделяются RTP и RTCP пакеты, относящиеся к отдельным сессиям.
5. Из пакетов сессии выделяются RTP-идентификаторы абонентов и тип медиа-данных.
6. При наличии корректных RTCP значений определяются значения параметров передачи голосовых сообщений. В противном случае осуществляется анализ RTP пакетов, из которых извлекаются данные о порядковом номере пакета и его временна́я RTP-метка, производится сбор дополнительных временны́х данных, которые можно будет использовать для поиска значения задержки передачи.
7. На основе собранных данных осуществляются предварительные расчеты значений джиттера и других параметров.
8. Для каждой сессии генерируется отчет с предполагаемыми значениями величины задержки передачи, джиттера и потери пакетов или с данными, на основе которых МОКС сможет вычислить интересующие параметры. Все отчеты пересылаются в МОКС.

Структура МОКС приведена на рис. 4.

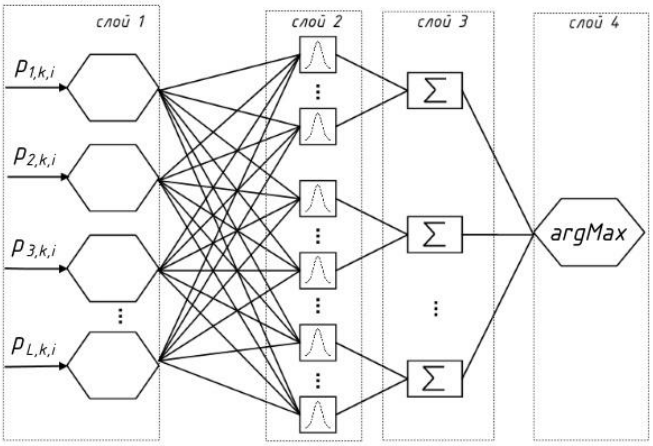


Рис. 4. Структура МОКС

Основой МОКС является совокупность нейросетей, каждая из которых представляет собой отдельную РБФ сеть. МОКС ставит в соответствие каждой точке в пространстве параметров $p_{j,k,i}$, $j=\overline{1,L}$, $k=\overline{1,M}$, $i=\overline{1,N}$, MOS-оценку m_l, $l=\overline{1,K}$.

Поскольку МОКС должен определять принадлежность заданной точки не к двум, а более классам, то нейросеть на рис. 4 состоит из K РБФ сетей, каждая из которых определяет степень близости к отдельной MOS-оценке m_l, $l=\overline{1,K}$.

В МОКС выделено четыре слоя. Слой входных параметров содержит L нейронов, каждому из которых соответствует один из параметров, используемых для оценки качества i-го сервиса.

Следующий слой – скрытый слой нейронов, преобразовывающих расстояние от точки, определяемой значениями параметров $p_{j,k,i}$, $j=\overline{1,L}$, $k=\overline{1,M}$, $i=\overline{1,N}$, до центра облака точек, соответствующего отдельной MOS-оценке из K оценок.

Третий слой образуют K сумматоров, выдающие число, определяющее степень близости точки к каждой из MOS-оценок.

Последний слой предназначен для вынесения решения о качестве отдельного сеанса предоставления мультимедийной услуги путем выставления этому сеансу MOS-оценки.

Выбор РБФ сети в качестве основы для МОКС обусловлен следующими факторами. Количество слоев РБФ сети заведомо известно. РБФ отличается высокой скоростью обучения, поскольку параметры линейной комбинации в выходном слое можно оптимизировать с помощью методов линейного моделирования.

В то же время применение линейной оптимизации в выходном слое сети РБФ требует определения числа радиальных элементов, положения их центров и величины отклонений. Кроме того, РБФ сеть менее пригодна для поиска субоптимальных решений, чем ряд других сетей. Однако данная проблема может быть решена путем структурного обучения нейронной сети.

Проведены эксперименты, в ходе которых изменялись значения ключевых параметров, влияющих на качество передачи голосовых сообщений, а эксперты выставляли MOS-оценки по шкале от 1 до 5, оценивая качество предоставления услуги VoIP по критериям, предложенным в [3].

В ходе дополнительных экспериментов были подтверждены работоспособность и эффективность предложенного метода. В то же время, неумение РБФ сети экстраполировать выводы за область данных, используемых для обучения, требует перебора всех возможных значений параметров. С другой стороны, экстраполяция на данные, находящиеся за пределами обучающего множества, не всегда является преимуществом нейросетей и может приводить к неверной оценке качества сервисов со стороны МОКС. Поэтому данная особенность РБФ сетей не является существенным недостатком, препятствующим ее применению при решении задачи оценки качества мультимедийных сервисов.

Выводы

Рассмотрена проблема оценки качества предоставления мультимедийных услуг операторами телекоммуникационных сервисов.

Предложен метод оценки качества предоставления мультимедийных сервисов. Метод основан на использовании технологии DPI и нейростевого классификатора. Предложена структура подсистемы, осуществляющей мониторинг качества предоставления услуги VoIP. Разработаны модуль экстракции данных, осуществляющий захват и анализ сетевых пакетов, и модуль оценки качества мультимедийных сервисов на основе нейросетевого классификатора РБФ типа.

Список литературы

1. Ролик А.И. Метод потенциальных функций в задачах оценки уровня телекоммуникационных сервисов / А.И. Ролик, Т.И. Ланге, А.А. Покотило, Б.А. Март, М.В. Ясочка // Вісник НТУУ «КПІ»: Інформатика, управління та обчислювальна техніка. – К.: «ВЕК+», 2012. – № 57. – С. 133–143.

2. Ролик А.И. Декомпозиционно-компенсационный подход к управлению уровнем услуг в корпоративных ИТ-инфраструктурах / А.И. Ролик // Вісник НТУУ «КПІ». Інформатика, управління та обчислювальна техніка: Зб. наук. пр. – К.: Век+, – 2013. – № 58. – С. 78-88.
3. Rec. ITU-T P.800. Methods for subjective determination of transmission quality. – 1996. – 37 p.
4. NIQA — Non-Instrisuve voice Quality Analyzer. Режим доступа: http://www.sevana.fi/non-intrusive-voice-quality-testing-software.php
5. Rec. ITU-R BS.1387-1. Method for objective measurements of perceived audio quality. – Recommendation ITU-R. – 2001. – 100 p.
6. Rec. ITU-T J144. Measurement of the quality of service. Objective perceptual video quality measurement techniques for digital cable television in the presence of a full reference. – ITU-T Recommendation. – 2001. – 70 p.
7. Rec. ITU-R BT.1683. Objective perceptual video quality measurement techniques for standard definition digital broadcast television in the presence of a full reference. – Recommendation ITU-R (Question ITU-R 44/6). – 2004. – 107 p.
8. Grancharov V. Non-intrusive speech quality assessment with low computational complexity / Volodya Grancharov, David Y. Zhao, Jonas Lindblom, W. Bastiaan Kleijn // In INTERSPEECH 2006 – ICSLP, Ninth International Conference on Spoken Language Processing, Pittsburgh, PA, USA, September 17–21, 2006. ISCA. – 2006.
9. Rizzo L. netmap: A Novel Framework for Fast Packet I/O // USENIX Annual Technical Conference ATC’12, Boston, MA. USENIX Association. – 2012. – 12 p.
10. Simoncelli D. Stream-monitoring with blockmon: convergence of network measurements and data analytics platforms / D. Simoncelli, M. Dusi, F. Gringoli, S. Niccolini // SIGCOMM Comput. Commun. Rev. – 2013. – 6 p.

УДК 004.724.4(045)

ГАБИНЕТ А. В.

АНАЛИЗ ЭФФЕКТИВНОСТИ ПРОТОКОЛОВ ДОСТУПА К УДАЛЕННЫМ ХРАНИЛИЩАМ ВИДЕОДАННЫХ

Предложен способ повышения эффективности протоколов доступа к внешним хранилищам видеоданных за счет изменения параметров: потери и задержки пакетов. Проведен анализ эффективности использования протоколов доступа к хранилищам видеоданных в глобальной сети. По результатам эксперимента даны рекомендации по выбору подходящего протокола в зависимости от значений задержки и потери пакетов.

A method for increasing the efficiency of the protocols to access external storage of video data by changing parameters: packet delay and loss. The analysis of efficiency using the protocols to access the repositories of video data in the global network. According to the results of the experiment are given advice on choosing a suitable protocol depending on the values of latency and packet loss.

Введение

Система хранения видеоданных представляет собой комплексное решение по организации надежного хранения информационных ресурсов и предоставления гарантированного и безотказного доступа к ним серверов центра обработки видеоданных. При разработке и создании системы хранения видеоданных в обязательном порядке требуется решение, обладающее оптимальным соотношением производительности, доступности, надежности, отказоустойчивости и совокупной стоимости владения. Сохранение общих видеоданных на централизованных файловых серверах (а не на индивидуальных компьютерах) упрощает такие административные задачи, как резервирование и восстановление файлов и каталогов. Это так же централизует такие стандартные административные задачи, как контроль за использованием файловой системы, и представляет новые возможности управления хранением видеоданных – например, балансирование загрузки.

Однако, в случае территориального отдаления потребителей дисковых ресурсов от основной части системы хранения видеоданных (например, сервера филиалов организации или отдельные компьютеры пользователей в других городах), к протоколу обмена данными между потребителем и поставщиком дисковых ресурсов предъявляются определенные требования. Хотя и существуют протоколы, специально предназначенные для передачи файлов в глобальных сетях, по медленным и ненадежным каналам (например, FTP), они не всегда удобны и зачастую не обладают возможностями, предоставляемыми протоколами доступа к хранилищу типа SAN или NAS локальной сети [1]. Сейчас, с развитием высокоскоростных каналов передачи в глобальных сетях актуальной становится задача использования этих протоколов для построения распределенных систем хранения видеоданных.

Обзор и анализ протоколов доступа к видеоданным по сети

Протоколы можно разделить на два типа: доступ на уровне файлов, и на уровне блоков. Здесь будут рассмотрены два протокола доступа на уровне файлов – NFS и SMB, получившие широкое распространение в настоящее время, и один протокол доступа на уровне блоков – iSCSI.

Протокол NFS. Одной из самых известных сетевых файловых систем является Network File System (NFS) фирмы Sun Microsystems [2]. NFS была первоначально создана для UNIX-компьютеров. Сейчас она поддерживает как UNIX, так и другие ОС, включая MS DOS. NFS поддерживает неоднородные системы, например, MS-DOS-клиенты и UNIX-серверы.

Основная идея NFS - позволить произвольному набору пользователей разделять общую файловую систему. Чаще всего, но не обязательно, все пользователи принадлежат одной локальной сети. Каждый NFS-сервер предоставляет один или более своих каталогов для доступа удаленным клиентам. Клиенты получают доступ к экспортируемым каталогам путем монтирования.

Протокол SMB. Протокол SMB (Server Message Block) – это протокол, обеспечивающий сов-местное использование ресурсов в се-

тях MS-Net, LAN Manager и Windows [3]. Также имеются со-ответствующие решения для OS/2, Netware, VMS и Unix-систем, разработанные такими производителями, как AT&T, HP, SCO, а при использовании Samba – свыше 33 других. Протокол SMB используется в среде клиент-сервер для обеспечения доступа к файлам, принтерам, почтовым ячейкам (mail slots), именованным каналам (named pipes) и интерфейсам прикладного программирования (Application programming interface, API). Он был совместно разработан компаниями Microsoft, IBM и Intel в середине 80-х годов. SMB может использоваться поверх нескольких сетевых протоколов.

Протокол SMB неоднократно изменялся. Последней версией протокола SMB является протокол CIFS (Common Internet File System) ОС Windows 2000, который является незначительно измененным вариантом протокола NT LM 0.12, использовавшимся ранее. В следующем разделе детально описывается последняя реализация протокола SMB.

Протокол iSCSI. Это технология, построенная по стандартам IETF, которая заключает блоки SCSI в Ethernet-пакеты [4]. Протокол iSCSI – это протокол транспортировки SCSI команд по IP-сетям. Характеристики сетей хранения видеоданных, построенных на базе протокола iSCSI:

-Высокая доступность – наличие нескольких путей между серверами и хранилищами видеоданных обеспечивает возможность постоянного подключения, даже при отказе ряда каналов.

-Масштабируемость – расширение емкости хранилищ видеоданных без необходимости отключения приложений.

-В рамках гетерогенных платформ возможность использования дисковых и ленточных накопителей.

-Использование хорошо известной технологии Ethernet.

-Использование стандартных коммутаторов и другого сетевого оборудования.

Постановка задачи

На рисунке 1 показана схема подключения к серверам приложений хранилища видеоданных, которое находится в локальной сети. При использовании такой схемы подключения можно говорить об архитектуре NAS, так как для передачи видеоданных приложений и трафика между серверами и хранилищем видеоданных используется та же самая сеть.

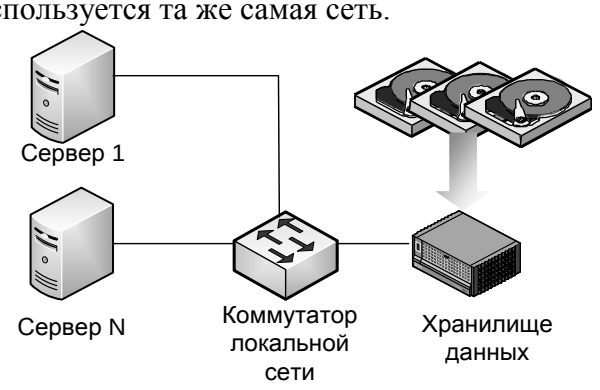


Рис. 1. Схема подключения хранилища видеоданных

При подключении удаленного хранилища видеоданных к серверам через глобальную сеть в ней присутствует также трафик между многими другими узлами, что существенно влияет на качество обслуживания.

Поэтому для проведения эксперимента по исследованию поведения протоколов доступа к хранилищам видеоданных в глобальной сети был построен тестовый стенд, конфигурация которого показана на рисунке 2.

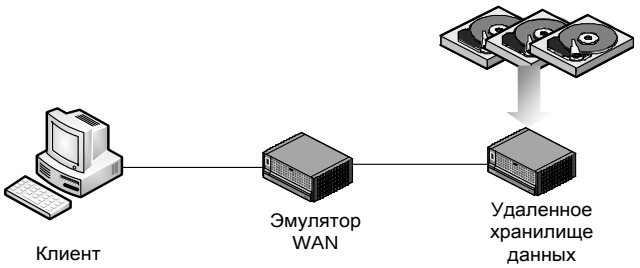


Рис. 2. Конфигурация глобальной сети для проведения экспериментов

Клиентский узел – это рабочая станция Pentium 4, 2ГГц, 256 Мб ОЗУ, сетевой адаптер Intel Ether Express PRO/100+ (Intel 82559). Для протоколов NFS и iSCSI использовалась ОС CentOS 5.0 i386. Для протокола SMB/CIFS использовалась ОС Windows Server 2003 SP2.

Эмулятор WAN был реализован средствами программного обеспечения Dummynet в ОС FreeBSD 6.3.

Удаленное хранилище видеоданных построено на базе машины Pentium 4 D, 3 ГГц, 2Гб ОЗУ, сетевой адаптер Intel Ether Express PRO/100+ (Intel 82559), ОС CentOS 5.0 x86_64 с использованием программного обеспечения, реализующего протоколы доступа NFS, SMB/CIFS, iSCSI.

Все узлы объединены в сеть по технологии Fast Ethernet (IEEE 802.3u) на скорости в 100Мбит/с.

Тестирование протоколов выполнялось с помощью программного обеспечения Bonnie++ 1.03c [5].

Проводилась серия опытов для каждого из протоколов с изменением времени задержки и количества теряемых пакетов в эмуляторе WAN. Были выбраны следующие значения для задержки пакетов: < 1мс, 5мс, 10мс и количества теряемых пакетов: 0%, 0,5%, 1%, 2%.

На стороне клиента использовались стандартные компоненты ОС, которые включаются в базовый набор системы, без дополнительных настроек.

Полученные результаты

На рисунках 3 - 8 представлены зависимости средней скорости чтения/записи от количества теряемых пакетов для задержек < 1мс, 5мс, 10мс.

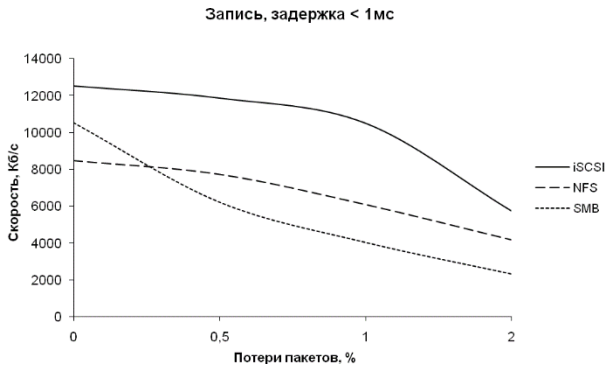


Рис. 3. Зависимость скорости записи от потерь пакетов для задержки меньше 1 мс

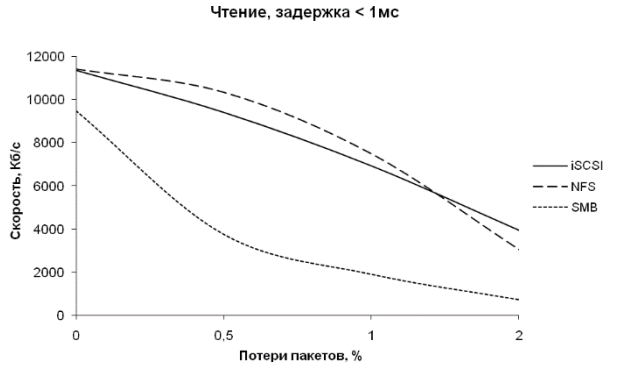


Рис. 4. Зависимость скорости чтения от потерь пакетов для задержки меньше 1 мс

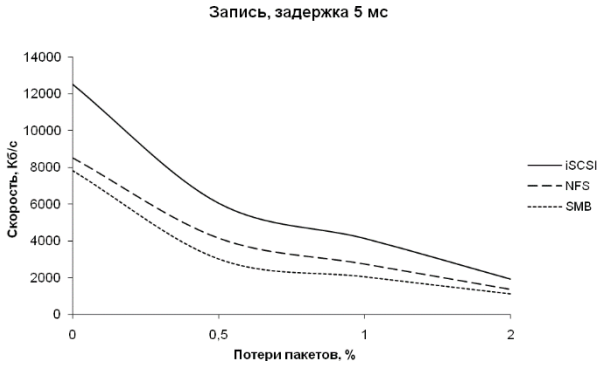


Рис. 5. Зависимость скорости записи от потерь пакетов для задержки 5 мс

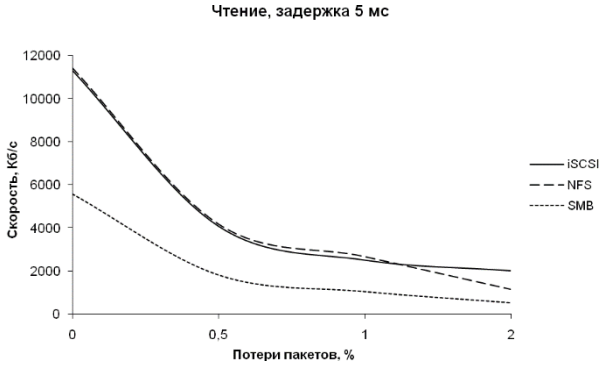


Рис. 6. Зависимость скорости чтения от потерь пакетов для задержки 5 мс

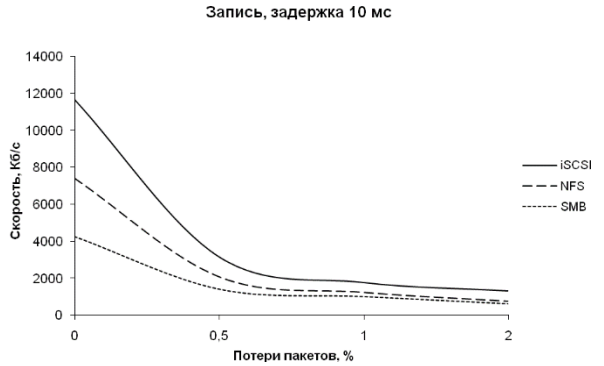


Рис. 7. Зависимость скорости записи от потерь пакетов для задержки 10 мс

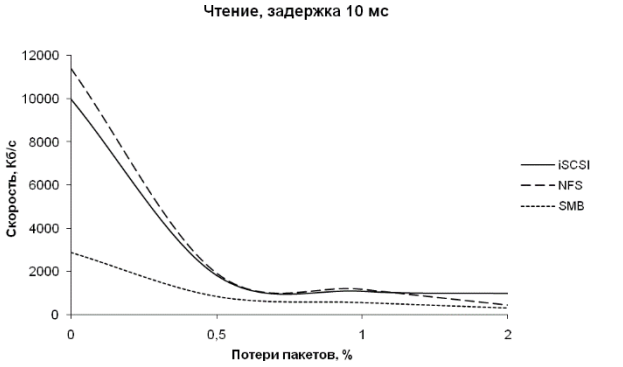


Рис. 8. Зависимость скорости чтения от потерь пакетов для задержки 10 мс

Рекомендации:

1. Результаты тестирования показывают, что в ненадежных и сильно загруженных сетях iSCSI предпочтительнее NFS и SMB, так как он более устойчив к потерям пакетов.

2. В надежных сетях рекомендуется использовать протоколы NFS и SMB, так как они позволяют обеспечить совместный доступ к одним и тем же видеоданным, в то время как iSCSI не предоставляет такой возможности.

3. В протоколе NFS при увеличении задержки от 0 до 10 мс наблюдается незначительное уменьшение скорости чтения. Этот протокол можно рекомендовать для доступа к удаленным хранилищам видеоданных в сетях без потерь, где преобладают операции чтения.

4. Протокол SMB не рекомендуется использовать для доступа к удаленным хранилищам видеоданных, так как у него наблюдается значительное снижение скорости чтения и записи даже при небольших задержках и потерях пакетов.

5. Для всех протоколов наблюдается сильная зависимость скорости чтения/записи от количества теряемых пакетов, при этом зависимость скорости чтения/записи от задержки не такая большая. Поэтому для доступа к удаленным хранилищам видеоданных рекомендуется выбирать надежные не перегруженные каналы, то есть каналы без потерь.

Выводы

В работе предложен способ повышения эффективности протоколов доступа к внешним хранилищам видеоданных. Основными параметрами оценки качества обслуживания были выбраны потеря и задержка пакетов.

Проведен анализ эффективности использования протоколов доступа к хранилищам видеоданных в глобальной сети. Для этого использовалась обобщенная модель, эмулирующая поведение трафика в глобальной сети. По результатам эксперимента даны рекомендации по выбору подходящего протокола в зависимости от значений задержки и потери пакетов.

Список литературы

1. Олифер В.Г., Олифер Н.А. Сетевые операционные системы. СПб.: Питер, 2003. – 539 с.
2. Brian Pawlowski, Spencer Shepler, Carl Beame, Brent Callaghan, Michael Eisler, David Robinson “The NFS Version 4 protocol”, Proceedings of 2nd International SANE Conference, May, Maastricht, The Netherlands.
3. Thomas Lee, Joseph Davies “Microsoft Windows 2000 TCP/IP Protocols and Services Technical Reference”, Microsoft Press, January , 2000. – 345 p.
4. Margaret Rouse "iSCSI (Internet Small Computer System Interface)". SearchStorage. Retrieved 3 November 2012.
5. Bonnie++ [Электронный ресурс] // – Режим доступа: http://www.coker.com.au/bonnie++

УДК 004.94

ТЕЛЕНИК С.Ф.,
АМОНС О.А.,
ТРОЦЬКИЙ С.О.,
БІЛОРУС Д.С.

АВТОМАТИЗАЦІЯ ПІДТРИМКИ БІЗНЕС-ПРОЦЕСІВ У WEB-СЕРЕДОВИЩІ

У статті розглянуто основи функціонування і реалізації системи автоматизації підтримки бізнес-процесів, яка дозволяє значно спростити розробку та підтримку бізнес-процесів підприємства за рахунок впровадження сервіс-орієнтованої архітектури та виконання системних завдань за допомогою наявних у Web-середовищі сервісів. Система підбирає комплекс необхідних для підтримки бізнес-процесу Web-сервісів та надає єдину точку входу для взаємодії з ними системним завданням.

The subject of the article is basic activities of functioning and implementation of business process automation support system that significantly streamline the development and support of business processes by implementing a service-oriented architecture and perform system's tasks with using services which is currently exist in web environment. The system searches and selects complex of Web-services to support business process and provides a single entry point to interact with them by system's tasks.

Вступ

В основі організації діяльності підприємств лежать бізнес-процеси. Інтегровані у єдину систему вони визначають поведінку підприємства за рахунок координованої взаємодії співробітників, спрямованої на визначення і ефективну реалізацію стратегію в умовах мінливого бізнес-середовища. Для автоматизації бізнес-процесів використовують спеціальні інформаційні системи виконання бізнес процесів. Використання сервіс-орієнтованої архітектури в таких системах дозволяє швидко адаптуватися до мінливих умов бізнес-середовища. В свою чергу, автоматичний пошук та композиція існуючих Web-сервісів може значно полегшити впровадження та підтримку цих систем. Розробленню моделей і методів співставлення операцій бізнес-процесу та існуючих Web-сервісів, їх вибору і виконанню і присвячена стаття.

Огляд проблеми

У загальному випадку бізнес-процес – це сукупність взаємопов'язаних заходів і завдань, спрямованих на створення певного продукту або послуги для споживачів. Бізнес-процеси виконуються всередині кожної організації, незалежно від того, формалізовані вони чи ні. Популярна сьогодні система умовних позначень (нотація) BPMN визначає скінченний набір графічних елементів для описання бізнес-процесу [1]:

- Об'єкти потоку управління: події, дії та логічні оператори;
- З'єднувальні об'єкти: потік управління, потік повідомлень та асоціації;
- Ролі: пули і доріжки;
- Артефакти: дані, групи і текстові анотації.

Найменшою елементарною дією в бізнес-процесі є завдання (task). З точки зору автоматизації бізнес-процесів, саме завдання є головним цільовим об'єктом. Інші елементи пов’язані з розв’язанням задач оркестрування, хореографії, координування, та ін. Для цих задач успішно застосовують мови виконання бізнес-процес, наприклад, BPEL.

На Рис. 1 у нотації BMPN 2.0 зображений спрощений узагальнений бізнес-процес. Він складеться з двох учасників: деякого програмного забезпечення (Customer system) та оператора (Operator), що взаємодіє з ним.

У наведеному процесі дії, що автоматично виконуються системою (Auto-task) межують з діями, що потребують втручання людини (User task). При цьому, логіка виконання системних дій є невід’ємною частиною програмного забезпечення, що відповідає за виконання бізнес-процесу. Процес, побудований таким чином, може добре виконувати покладені на нього функції та має як переваги, насамперед, простоту та повний контроль над системою, так і недоліки, серед яких найбільш суттєвими є статичність процесу, висока складність розробки, складність підтримки та модифікації, склад-

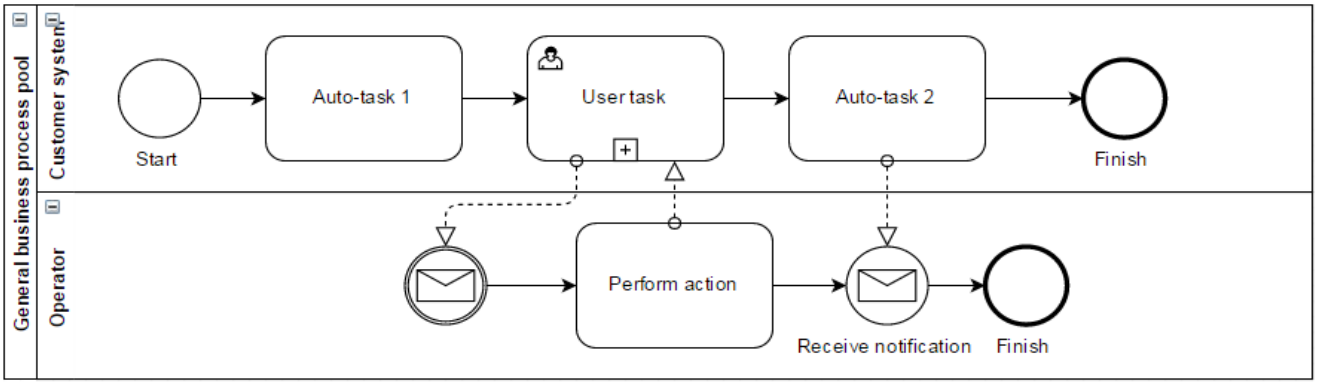


Рис. 1. Приклад бізнес-процесу

ність повторного використання логіки виконання системних дій.

Частково усунути недоліки описані вище можна за рахунок впровадження сервіс-орієнтованої архітектури. Сервіс-орієнтована архітектура (Service-Oriented Architecture, SOA) становить собою стиль створення архітектури IT, спрямований на перетворення бізнесу в ряд пов'язаних сервісів – стандартних бізнес-задач, які можна при необхідності викликати через мережу [2]. Це може бути Інтернет, локальна мережа або географічно розподілена мережа, яка об'єднує різні технології і поєднує сервіси так чином, ніби вони встановлені на одній локальній машині. Для виконання певного бізнес-завдання можна об'єднувати множину таких сервісів. Це дозволяє компаніям швидко адаптуватися до змін умов і вимог ринку.

Визначимо Web-сервіси як автономні, модульні програми, призначені для реалізації бізнес-процесів. Побудова і надання Web-сервісів спираються на ряд галузевих стандартів: WSDL (для описання сервісів), UDDI (для інформування та публікації); SOAP (для обміну повідомленнями) [3]. Ці специфікації не залежить від платформи і мови, завдяки чому користувачі можуть пов'язувати різні компоненти з різних організаційних структур. Розглянемо сучасний підхід, при якому існуючий бізнес процес буде виконуватися за допомогою інформаційної системи управління бізнес-процесами. А сам бізнес процес, за виключенням логічних операцій та ручних дій, має підтримуватися існуючими Web-сервісами, задовольняючи основоположні принципи SOA. На Рис. 2 зображено модифікований бізнес-процес, що задовольняє умови описаного підходу та характеризується такими перевагами:

- розділення логіки управління бізнес-процесу та бізнес-логіки, що додає гнучкості системи та зменшує вартість підтримки;
- можливість розробки функціоналу бізнес-логіки зручними для кожного окремого випадку способами;
- можливість використання Web-сервісів третіх сторін, що зменшує час на розробку системи;
- можливість повторного використання Web-сервісів у різних частинах та процесах системи.

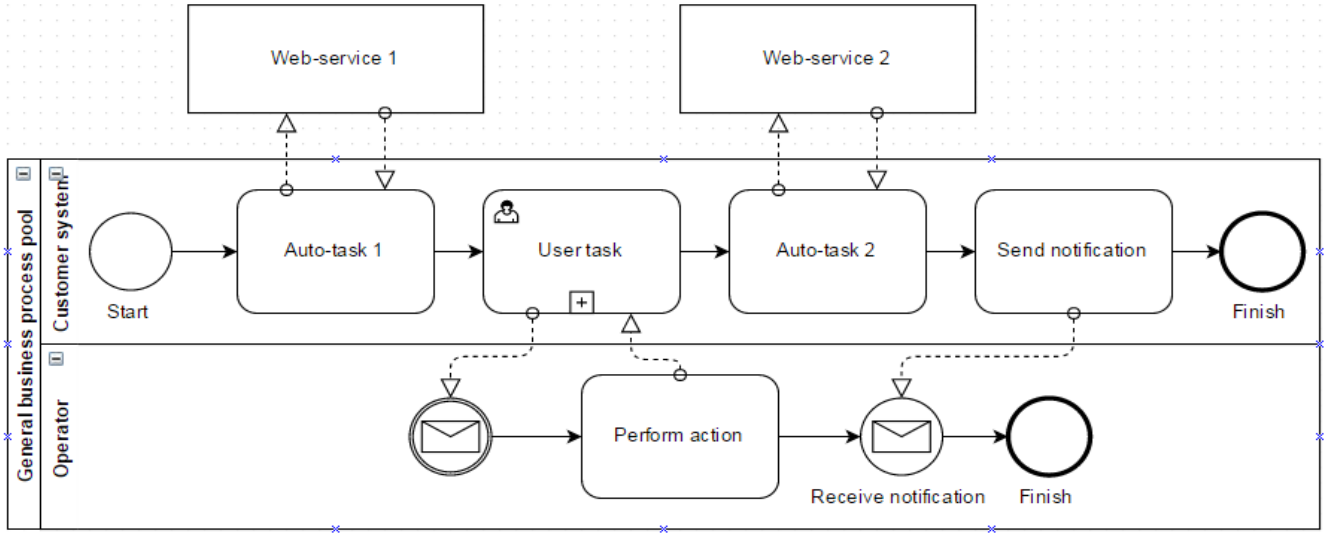


Рис. 2. Бізнес процес з використанням СОА

В даному процесі логіка виконання системних дій винесена у Web-сервіси, що можуть бути реалізовані будь-якою мовою програмування та за допомогою будь-яких технологій. В основному потоці при цьому лишаються користувацькі дії та дії виклику Web-сервісів. Таким чином, ми можемо повністю перекласти виконання основного потоку на один з серверів виконання бізнес-процесів. Промисловим стандартом у даній галузі є Oracle BPEL Process Manager Server [4], що є сервером виконання бізнес-процесів, описаних мовою виконання бізнес-процесів BPEL (Business Process Execution Language).

Головним недоліком, описаного вище сучасного і досить розповсюдженого підходу є необхідність ручного пошуку Web-сервісів та їх жорстка прив'язка до кожної дії бізнес процесу. При цьому, за будь-яких змін у структурі Web-сервісу або виходу його з ладу – вся система буда непрацездатною. Для уникнення такої ситуації і надання більшої незалежності системам управління бізнес-процесів підприємства автори пропонують побудувати окрему систему (далі за текстом – система автоматизації підтримки бізнес-процесу), яка візьме на себе задачу пошуку та композиції Web-сервісів для підтримки будь-якого бізнес-процесу та зведе до мінімуму необхідність ручного втручання в бізнес-логіку процесу.

Побудова системи автоматизації підтримки бізнес процесів

запропонований підхід, який дозволяє вирішувати проблему підтримки бізнес-процесів за рахунок існуючих технологій і легко адаптуватися до нових, насамперед, семантичних технологій організації Web-сервісів.

Беручи до уваги переваги та недоліки існуючих підходів, автори розробили систему підтримки бізнес-процесів, що буде єдиною точкою входу для взаємодії з зовнішніми Web-сервісами. Тобто, замість звернення безпосередньо до зовнішніх Web-сервісів, сервер виконання взаємодіє лише з системою автоматизації бізнес-процесу, що значно спрощує опис бізнес-процесу. При цьому, відповідальність за підбір та забезпечення працездатності Web-сервісів повністю лягає на систему автоматизації. Для збереження усіх переваг СОА, система автоматизації має надавати інтерфейс взаємодії у вигляді Web-сервісу. Приклад бізнес-процесу, що використовує дану систему зображений на Рис. 3.

Система, побудована таким чином, успішно вирішує такі завдання:

- Аналіз бізнес-процесу, виявлення системних дій;
- Динамічна генерація методів Web-сервісу у відповідності до описаних системних дій;
- Пошук реалізацій завдань бізнес-процесу;
- Вибір оптимальної реалізації серед множини можливий реалізацій;
- Забезпечення зв'язку між згенерованими методами та їх реалізаціями.

Відповідно до підходу, що пропонується, система автоматизованої підтримки буде працювати у таких режимах:

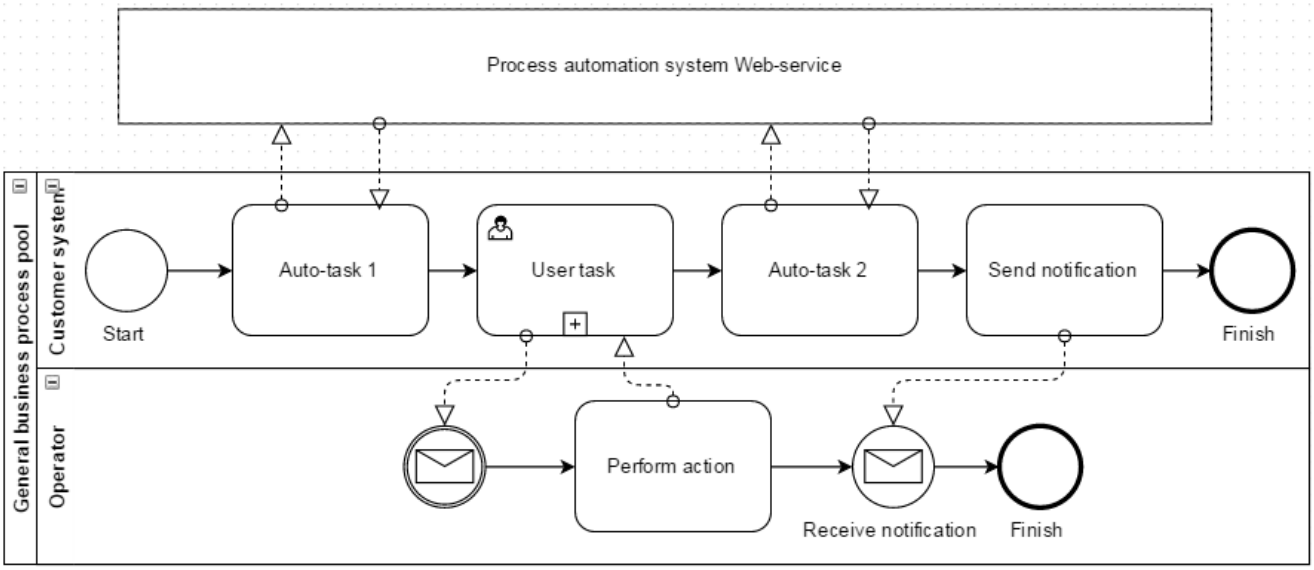


Рис. 3. Бізнес-процес з використанням системи автоматизації

Виходячи із реалій сьогодення, але й враховуючи перспективи розвитку IT-галузі, в роботі

1. Базовий – використовуючи засоби автоматизації роботи з нотацією для моделювання бізнес-процесів BPMN, оператор вибудовує схеми бізнес-процесів, визначаючи елементи, які система буде автоматично підтримувати за рахунок явно визначених Web-сервісів;

2. Автоматизований (реальний) – система, на основі попередньо визначеного оператором опису бізнес-процесів, здійснює пошук Web-сервісів під кожний елемент, що може бути виконаний автоматично. Серед знайдених Web-сервісів оператор відбирає ті, що дійсно задовольняють необхідні потреби, після чого система компонує остаточний комплекс Web-сервісів підтримки бізнес процесів на основі моделей оптимізації;

3. Автоматичний (перспективний) – на основі семантичних описів бізнес-процесу і Web-сервісів система здійснює пошук останніх, відбирає їх комплекс на основі моделей оптимізації і автоматично формує та виконує бізнес-процеси.

Такий підхід враховує реалії і вже сьогодні дозволяє підтримувати автоматизоване проектування і виконання бізнес-процесів в організації підприємства чи компанії. Ці можливості забезпечуються засобами базового і автоматизованого режимів роботи системи.

Важливим недоліком перших двох режимів є “жорстка” прив’язка Web-сервісів до бізнес-процесів. У випадку будь-яких змін структури чи повної відмови Web-сервісів, відповідний бізнес-процес не зможе бути правильно виконаний.

Щоб забезпечити певний рівень незалежності бізнес-процесів від Web-сервісів і потрібен третій автоматичний режим роботи. Однак, його експериментальне дослідження, здійснюване в університетських лабораторіях, ще не перейшло в фазу дослідного впровадження на реальних підприємствах.

Тут є декілька причин: по-перше, семантичні описи ще не досягли того рівня, щоб на їх основі однозначно автоматично визначити чи здатен Web-сервіс виконати елемент бізнес-процесу, по-друге, підготовка і перехід до такої схеми вимагають часу і витрат, що враховуючи першу причину не є наразі доцільним.

Зазначимо, що від обраного режиму роботи залежить лише метод пошуку можливих реалізацій завдань бізнес-процесу. Таким чином перехід від базового і автоматизованого режимів до більш перспективного автоматичного може бути реалізований без суттєвого втручання до всієї системи.

Аналіз бізнес-процесу та генерація методів

Нехай бізнес процес, який необхідно автоматизувати, описаний мовою BPEL. Мова BPEL, використовуючи мову XML за основу, підтримує стек технологій розробки та надання Web-сервісів, в тому числі SOAP, WSDL, UDDI, WS-Reliable Messaging, WS-Addressing, WS-Coordination, WS-Transaction. Процес, описаний мовою BPEL, складається з кроків, які називаються “завданнями”. Мова BPEL підтримує примітивні, а також структуровані завдання.

Структуровані завдання, або підпроцеси, є невід’ємною складовою частиною мови BPEL. Вони дозволяють будувати бізнес-процес, використовуючи рівні абстракції. Таким чином, при правильній побудові, на кожному рівні бізнес-процес залишається простим і зрозумілим. Структуровані завдання повністю складаються з примітивних завдань.

Примітивні завдання є основними конструкціями і використовується для загальних задач, однією з яких є виклик інших Web-сервісів. Таке завдання в XML представленні бізнес-процесу позначається тегом <invoke>. Тобто, для виявлення системних завдань як у складі процесу, так і у складі структурованих завдань достатньо віднайти такі теги у тексті XML. Приклад такої частини наведений нижче.

```
<invoke partnerLink="concretePartnerLink"
portType="pt:concretePortType"
operation="concreteOperation"
inputVariable="in_variable"
outputVariable="out_variable" />
```

Для кожного знайденого системного завдання має бути згенерований відповідний метод Web-сервісу системи. Наприклад, для наведеного прикладу має бути згенерований метод “concreteOperation”. Після цього, за допомогою одного з режимів роботи, згенерованому методу ставиться у відповідність виклик зовнішнього Web-сервісу результат якого буде повернений згенерованим методом.

Пошук реалізацій завдань бізнес процесу в залежності від режиму роботи

Введемо ряд позначень.

Процес P складається з ряду завдань $T_k, k = 1, \dots, N_t$, де N_t – кількість завдань процесу.

$W_i, i = 1, \dots, N_w$ – i-ий сервіс в реєстрі Web-сервісів, де N_w – кількість Web-сервісів в реєстрі.

$M_{ij}, j = 1, \dots, N_{mi}$ – j-ий метод, що виконується i-им Web-сервісом, N_{mi} – число методів i-го Web-сервісу.

$M = \{M_{ij}\}$ – множина всіх методів Web-сервісів.

$impl(T_k, M_{ij}) \in \{0,1\}$ – функція реалізації, що приймає значення 1, якщо метод M_{ij} реалізує завдання T_k процесу P, і 0 в супротивному випадку.

$X(T_k)$ – множина реалізацій завдання T_k, де

$$X(T_k) = \{M_{ij} | impl(T_k, M_{ij}) = 1\}$$

$x(P) \in \prod_{k=1}^{n} X(T_k) = \{M_{ij}^1, \dots, M_{ij}^k\}$ – одна з реалізацій процесу.

$X(P)$ – множина реалізацій процесу P, де

$$X(P) = \{x(P)\} = \prod_{k=1}^{n} X(T_k)$$

Таким чином, процес пошуку необхідних Web-сервісів можна описати наступним чином:

1. В базовому режимі оператор для кожного завдання бізнес-процесу T_k ставить у відповідність деякі методи M_{ij}, вручну формуючи множину реалізацій бізнес-процесу $X(P)$.

2. В автоматизованому режимі для кожної операції T_k оператору пропонується множина методів можливих реалізацій завдання $M' \in M, |M'| \ll |M|$, яка сформована методами синтаксичного пошуку серед доступних UDDI реєстрів. Тепер, для кожного завдання бізнес-процесу T_k оператор відбирає методи M_{ij}, такі що $impl(T_k, M_{ij}) = 1$, формуючи множину реалізацій завдання $X(T_k)$, з яких формується множина реалізацій процесу $X(P)$. При чому, для $\forall k\ X(T_k) \neq \emptyset$, інакше процес не може бути підтримано.

3. В автоматичному режимі для кожної операції T_k система автоматично знаходить множину реалізацій завдання $X(T_k)$ з якої формується множина реалізацій процесу $X(P)$. Це стає можливим за допомогою семантичних методів опису Web-сервісів. Проте, як вказувалось раніше, ці технології недостатньо розвинені і розповсюджені.

Синтаксичний пошук Web-сервісів.

Розглянемо Web-сервіс як програмну систему, що визначається рядком URI і інтерфейсами, визначеними на мові XML. Опис цієї програмної системи може бути знайдено іншими програмними системами, які можуть взаємодіяти з нею згідно з описом, за допомогою XML повідомлень. Шукати Web-сервіси можна вручну на сайтах виробників або сайтах-агрегаторах. Для машинного пошуку Web-сервісів використовують власні UDDI реєстри та реєстри бізнес-партнерів. Також для машинного пошуку можна використовувати відкриті UDDI реєстри, що підтримуються ентузіастами, а також їх альтернативи. Пошук Web-сервісів може бути виконаний по класифікаційному коду Web-сервісу, що присвоюються за класифікаторами (галузевого класифікатора NAICS, класифікатору продуктів і послуг UNSPSC, географічному класифікатору ISO 3166) при реєстрації Web-сервісу в реєстрі та/або ключовим словами, які приписуються Web-сервісу виробником. Тобто використовуючи набір ключових слів можна суттєво зменшити область пошуку необхідного Web-сервісу.

Таким чином, пошукова модель Web-сервісу являє собою двійку <КК, КС>, де КК – це класифікаційний код сервісу, а КС – множина ключових слів, що описують Web-сервіс. Запити на пошук також будуються відповідно до цієї моделі: або задається класифікаційна рубрика, по якій знаходяться всі коди, що входять в цю рубрику, і відповідні їм сервіси; або задається набір ключових слів, за яким будується ешелонована видача сервісів, релевантних запиту; або задається і те й інше. У будь-якому випадку результати запиту вимагають осмислення з точки зору відповідності реальним вимогам. Тобто, система автоматизації має створювати запит на перевірку отриманих результатів.

Для звуження кола пошуку необхідно також перевірити на відповідність методи Web-сервісу. Припустимо, що Web-сервіс W має два набори параметрів: $W^i = \{I_1, I_2, \dots\}$ для SOAP запиту (вхідні параметри) та $W^o = \{O_1, O_2, \dots\}$ для SOAP відповіді (вихідні параметри). Коли W виконується з усіма вхідними параметрами, W^i, він повертає вихідні параметри, W^o. Припустимо, що для того, щоб викликати W, всі вхідні па метри в W^i повинні бути вказані (тоб-

то, W^i є обов'язковими параметрами). Позначимо також набір параметрів як P, а набір Web-сервісів як W.

Коли "значення" двох параметрів, $p_1 \in P$ та $p_2 \in P$, є взаємозамінними, загалом, ми можемо сказати, що вони знаходяться "у відповідності" один з одним. Найпростіший спосіб перевірки на відповідність коли два параметри мають однакове ім'я і тип: $(p_1.name = p_2.name) \cap (p_1.type = p_2.type)$.

Тобто, у найпростішому випадку, до області допустимих методів, потрапляють методи Web-сервісів, що підходять за ключовими словами і параметри цих методів відповідають необхідним.

Семантичний пошук Web-сервісів.

Основні існуючі стандарти Web-сервісів (SOAP, WSDL і UDDI) спроектовані для представлення інформації про інтерфейси сервісів, їх розгортання та виклики, але практично не представлені можливості сервісів, що обмежує автоматичну інтеграцію додатків, написаних у відповідності зі стандартами Web-сервісів.

Проблема автоматичної інтеграції Web-сервісів викликала появу підходу побудованого на семантичних описах Web-сервісів. Семантично описані Web-сервіси становлять собою Web-сервіси, чиї «властивості, можливості, інтерфейси і результати кодуються в однозначній формі, що піддаються машинному опрацюванню» [5]. У цьому випадку, Web-сервіси можуть бути проаналізовані, як з точки зору зіставлення знання мети та інтерфейсів, так і з точки зору семантичного опису сервісів для реалізації пошуку, композиції, запуску, моніторингу та ін.

На сьогодні, загальновизнаний реєстр для семантичних описів Web-сервісів відсутній, хоча є пропозиції про включення такого опису в стандарт UDDI [6]. Передбачається, що семантичний опис сервісу буде виконуватися розробником відповідно до моделі <I, O, P, E>. Це означає, насамперед, що описуються не сервіси, а пов'язані з ними абстрактні процеси, що плануються до виконання в конкретній предметній області. Зазначена область задається концептуальною моделлю у вигляді OWL-онтології. I, O у цій моделі – це множина типів даних, що задаються класами онтології на вході і виході процесу відповідно; P – це множини логічних виразів, що визначають передумови, при виконанні яких можливе успішне виконання процесу; E – множина логічних виразів, що відображають ефекти виконання процесу з точки зору зміни стану зовнішнього середовища.

Очевидно, що і запит на пошук процесів повинен будуватися відповідно до моделі <I, O, P, E> – по суті, користувач, складаючи запит, повинен сформувати семантичний опис потрібного процесу. Це можливо, якщо онтології, що використовуються для формування запиту і процесу, ідентичні або зводяться одна до іншої.

Пошук процесів на основі відповідності їх входів/виходів запитом теоретично становить тільки проміжний етап у вирішенні проблеми пошуку сервісів за їх семантичним описам. Багато дослідників цієї проблеми зупинилися на цьому етапі, оскільки не могли знайти місце передумовам і ефектам в алгоритмах пошуку. Тому, переважна більшість досліджень, пов'язаних з пошуком процесів семантичних Web-сервісів, будується на основі моделі запиту у вигляді <I, O>. Оскільки I, O – це множина типів даних на вході і виході процесу, що задаються класами онтології предметної області, то вноситься семантична складова в пошук процесів. Цим пошук процесів семантичних Web-сервісів відрізняється від пошуку Web-сервісів за ключовими словами.

Вибір оптимальної реалізації серед множини можливий реалізацій

Після пошуку реалізацій завдань бізнес-процесу маємо сформовану множину реалізацій бізнес-процесу $X(P)$. Для того, щоб обрати єдину оптимальну реалізацію $x(P) \in X(P)$ необхідно визначити критерії оптимальності.

У статті для оцінки методів Web-сервісів Mij використовують такі функції:

$f_1(M_{ij}) = \mathrm{ExecutionTime}(M_{ij})$ – час виконання і обробки методу M_{ij} в секундах;

$f_2(M_{ij}) = \mathrm{Cost}(M_{ij})$ – вартість 1000 запитів до Web-сервісу W_i у гривнях;

$f_3(M_{ij}) = \mathrm{FreeLimit}(M_{ij})$ – кількість безкоштовних запитів до Web-сервісу W_i на день.

Використовуючи, функції оцінки методів, визначимо відповідні функції оцінки реалізацій процесів $x(P)$:

$$f_1(x(P)) = \sum_k f_1\left(M_{ij}^k\right),$$
$$f_2(x(P)) = \sum_k f_2\left(M_{ij}^k\right),$$
$$f_3(x(P)) = \sum_k f_3\left(M_{ij}^k\right), k = 1, \dots, N_t, \text{ де}$$

M_{ij}^{k} – метод M_{ij}, що реалізує k-те завдання бізнес-процесу P.

Введемо також функцію оцінки реалізації бізнес-процесу x(P), що не визначається суммою функцій оцінок відповідних методів:

$f_4\big(x(P)\big) = \mathrm{ServiceCount}(x(P))$ – кількість унікальних Web-сервісів задіяних в реалізації $x(P)$

Сформулюємо проміжну задачу наступним чином: серед множини можливих реалізацій процесу $X(P)$ вибрати таку реалізацію $x(P) \in X(P)$ для яких

$f_1(x) = f_1\big(\mathrm{x(P)}\big) \to \min,$
$f_2(x) = f_2(\mathrm{x(P)}) \to \min,$
$f_3(x) = f_3(\mathrm{x(P)}) \to \max,$
$f_4(x) = f_4(\mathrm{x(P)}) \to \min.$

Спростимо цільову функцію, ввівши векторний критерій, який приймає значення у просторі m-мірних векторів R^m, та звівши все до задачі мінімізації:

$$f(x) = f\big(x(P)\big) \to min,$$

$$f(x) = \big(f_1(x), f_2(x), -1 * f_3(x), f_4(x)\big),$$
$$f(x) \in R^m, x \in X(P)$$

Для більш точного визначення оптимальної реалізації згідно нагальних потреб у системі використовується поняття відносної важливості критеріїв, за яким деякий i-ий критерій f_i важливіший за j-ий критерій f_j з заданими додатними параметрами w_i, w_j. Оскільки важливість може змінюватися з часом, інформація про відносну важливість критеріїв винесена в конфігураційний файл. Якщо відносна важливість не вказана використовується відношення важливості за замовчуванням, за яким кількість сервісів, що використовується важливіша за час виконання, час виконання важливіший за ціну запиту, яка в свою чергу важливіша за ліміт безкоштовних запитів на добу:

$$\Omega = \{ServiceCount \succ \mathrm{ExecutionTime},$$
$$\mathrm{ExecutionTime} \succ \mathrm{Cost}, \mathrm{Cost} \succ \mathrm{FreeLimit}\} =$$
$$= \{f_4(x) \succ f_1(x), f_1(x) \succ f_2(x), f_2(x) \succ f_3(x)\}$$

Остаточна задача оптимізації виглядає наступним чином:

$$\begin{cases} f(x) \to min \\ x \in X(P) \\ \Omega \end{cases}$$

Для вирішення задачі в системі використовується метод цільового програмування з евклідовою метрикою.

Приведемо алгоритм вирішення задачі:

1. Провести оцінку множини можливих реалізацій $X(P)$;
2. Сформувати множину Парето-оптимальних реалізацій [7];
3. Звузити множину Парето за рахунок інформації про важливість критеріїв [8];
4. Вирішити задачу цільового програмування $x^* \in X^*(P)$ [9]:

$$inf_y \rho(f(x^*), y) = min_{x \in X} inf_y \rho(f(x), y), \text{де}$$

$\rho -$ евклідова відстань;
$X^*(P) -$ звужена множина Парето;
$y = (0{,}0, -1 * max_{x \in X} f_3(\mathrm{x}), 0)$ – ідеальний вектор критеріїв оцінки.

Висновки

Запропонована система автоматизації підтримки бізнес-процесів, яка розв'язує ряд важливих завдань, насамперед, автоматизації пошуку можливих реалізацій завдань бізнес-процесу та вибору оптимальної реалізації серед усіх можливих. При цьому надається єдиний інтерфейс для взаємодії між методами бізнес-процесу та їх реалізаціями. Система, побудована таким чином, робить основний бізнес-процес майже нечутливим до зовнішніх змін. Крім того, система побудована з урахуванням можливого переходу до третього автоматичного режиму, в якому множина реалізацій бізнес-процесу буде визначатися повністю машиною за рахунок семантичного пошуку, що дозволить однозначно визначати необхідні Web-сервіси та позбавить від необхідності втручання людини.

Список посилань

1. Information technology – Object Management Group Business Process Model and Notation, ISO/IEC 19510, 2013;

2. Мохаммед И. Мабрук. Краткие основы SOA // IBM Developer Works [Електронний ресурс]. – 2010. – Режим доступу до ресурсу: http://www.ibm.com/developerworks/ru/edu/ws-soa-ibmcertified/index.html.
3. Web Services Architecture – W3C Working Group Note 11. – The World Wide Web Consortium (W3C) [Електронний ресурс]. – 2004. – Режим доступу до ресурсу: http://www.w3.org/TR/ws-arch/.
4. ORACLE Data Sheet – ORACLE BPEL Process Manager – Режим доступу: http://www.oracle.com/technetwork/middleware/bpel/overview/ds-bpel-11gr1-1-134826.pdf;
5. McIlraith S. Semantic Web services / McIlraith S., Son T.C., Zeng H. // IEEE Intelligent Systems. Special Issue on the Semantic Web. – 2001. – Vol. 16. – N. 2. – P. 46-53.
6. Климов В.В. Система поиска и интеграции веб-сервисов, основанная на семантических описаниях / В.В. Климов // Приоритетные направления инновационного развития. – 2009. – №1. – С. 33.
7. Подиновский В. В. Парето-оптимальные решения многокритериальных задач / В. В. Подиновский, В. Д. Ногин. – М: Наука. Гл. ред. физ.-мат. лит.-ры, 1982. – 256 с..
8. Ногин В. Д. Принятие решений в многокритериальной среде: количественный подход (2-е изд.) / В. Д. Ногин. – М: Физматлит, 2005. – 176 с.
9. Поляк Б. Т. Введение в оптимизацию / Б. Т. Поляк. – М: Наука. Гл. ред. физ.-мат. лит.-ры, 1983. – 384 с.

УДК 004.414.22

БОЛДАК А.О.,
МАЗУРЕВИЧ Ю.В.

СПОСІБ ОПТИМІЗАЦІЇ ВІДОБРАЖЕННЯ ГРАФІЧНОЇ ІНФОРМАЦІЇ В КЛІЄНТ-СЕРВЕРНИХ ІНФОРМАЦІЙНИХ СИСТЕМАХ

В роботі запропоновано спосіб скорочення часу завантаження та об'єму даних, необхідних для відображення веб сторінки за рахунок попередньої обробки на стороні сервера. Проведено оцінку ефективності реалізації такого способу. Виявлені умови, в яких він буде найбільш ефективний, а також його недоліки та наведено шляхи зменшення їх впливу.

This paper presents an approach to reduce load time and volume of data necessary to display web page due to server side preprocessing. Measurement of this approach's effectivity has been conducted. There were discovered conditions in which this approach will be the most effective, its disadvantages and presented ways to reduce them.

Вступ

Представлення даних у вигляді графіка — дуже хороший спосіб донесення даних до кінцевого користувача. Також цей спосіб полегшує знаходження закономірностей в даних, завдяки чому часто використовується не тільки для презентацій, а й як інструмент досліджень. Для використання графіків в дослідженнях важливу роль починає відігравати інтерактивність.

Існує два способи генерації графічного відображення: на стороні клієнта та на стороні сервера.

Для реалізації побудови графіків на стороні сервера можуть використовуватись бібліотеки як:

JFreeChart - відкрита бібліотека для мови програмування Java, що спрощує створення різноманітних складних діаграм. Через різноманітні методи набору класів надає майже повний контроль над областю діаграми. Так реалізовані механізми збільшення/зменшення, обробки подій, можливості створення кількох діаграм на одній області, текстові підказки, задання вигляду кривих, точок та фону і т.п. [1]

Charts4j – це безкоштовний легковаговий Java API для побудови діаграм та графіків. Він дозволяє розробникам програмно створювати діаграми, що доступні в Google Chart Tools через простий та інтуїтивний Java API. [2]

Такі бібліотеки орієнтовані на побудову статичних зображень, а ті, які підтримують інтерактивність, створені для використання в окремих додатках.

Таким чином, генерація графіків на стороні сервера — це створення растрових чи векторних зображень, що позбавляє відображення інтерактивності. А динамічна зміна можлива лише через нові запити до сервера.

Для вирішення цієї задачі на стороні клієнта можуть використовуватись:

Flot — це бібліотека для побудови графіків на чистому JavaScript для jQuery, з фокусом на простоту використання, привабливий вигляд та інтерактивні елементи;

Ember Charts — бібліотека для створення графіків, що побудована на Ember.js та d3.js фреймворках. Вона включає часові, стовпчикові, кругові та точкові діаграми, які легко використовувати та модифікувати. [3]

Використання JavaScript бібліотек для побудови графіків, з одного боку, дає необхідні можливості, а, з іншого, вимагає завантаження власне бібліотек, які повинні надавати достатній функціонал як для створення графічного відображення, так і для надання користувачу можливості взаємодіяти з цим відображенням, що в свою чергу вимагає відносно значного розміру таких бібліотек, а також значної обчислювальної роботи, виконаної на стороні клієнта. В наслідок цього зростає час, необхідний на завантаження даних та час необхідний на власне рендерінг. [4]

Проблема в тому, що JavaScript бібліотеки повинні містити надлишковий код, велика частина якого не буде використана для побудови практично кожної окремо взятої сторінки.

Отже, на стороні сервера необхідне попереднє очищення коду: без цього складно буде досягти швидкої реакції клієнта.

Мета роботи — зменшити надлишковість програмного коду, що реалізує інтерактивність

графіків за рахунок попередньої підготовки сторінки на стороні сервера.

Пропозиція полягає в тому, щоб використовувати інтерпретатор JavaScript на стороні сервера для попереднього створення варіанту сторінки, що буде показаний користувачу, та прибирання зайвого коду та HTML тегів.

Для цього потрібно мати функціонал, аналогічний браузеру на стороні сервера. Повний чи майже повний функціонал браузера надають проекти:

PhantomJS — це заснований на рушії WebKit веб-браузер, без графічного інтерфейсу, що має JavaScript API. Він підтримує різні веб стандарти, такі як DOM handling, CSS selector, JSON, Canvas та SVG. Також має плагін для Node.js. [5]

HtmlUnit — браузер без графічного інтерфейсу, написаний на Java. Він надає програмам на Java можливість працювати з веб-сторінкою, має хорошу підтримку JavaScript, та може працювати зі складними AJAX бібліотеками. Здатний симулювати Firefox чи InternetExplorer залежно від конфігурації. [6]

Також надають функціонал браузера, але не будують відображення DOM (через що підвищується швидкість роботи, проте знижується точність відображення сторінки):

Zombie.js — середовище симульованого браузера для Node.js. [7]

ENVJS — середовище симульованого браузера, написане на JavaScript для рушія Rino. [8]

Існують бібліотеки, що дозволяють завантажити весь необхідний код та створити початковий варіант сторінки на стороні сервера. Необхідно лише створити систему, яка буде аналізувати, який код використовується, та прибирати зайвий.

Оцінка

Для оцінки була взята типова сторінка. Офіційний приклад використання бібліотеки Flot [9] рис. 1. Бібліотека Flot була обрана для проведення оцінки, оскільки це бібліотека, що широко використовується для побудови графіків. А також, це бібліотека з відкритим вихідним кодом, що дозволить зробити необхідні зміни в коді.

Початковий код даної сторінки був залишений без змін, а робота була здійснена над файлом jquery.flot.js, що містить код, який відповідає за відображення графіка

Вручну було визначено ділянки коду, що не використовуються. Для цього в усіх функціях були поставлені мітки. В якості мітки було використано console.log(«$FUNCTION_NAME»), де $FUNCTION_NAME це назва функції в якій знаходиться мітка. Таким чином, був створений файл, що містить усі назви функцій, що були викликані. Також, таким же чином, були відмічені деякі блоки if та if/else.

В зв’язку з тим, що сторінка не буде змінюватись в майбутньому, необхідність в деяких перевірках відпадає.

В файлі з бібліотеки слово function зустрічається 132 рази. 3 рази в коментарях і 10 разів в допоміжному блоку, який не розглядався. Таким чином, бібліотека має 119 функцій. В кожній з них була розміщена мітка. Для побудови графіка було викликано 53 функції без врахування повторів. Отже, можна безпечно видалити 66 з них. Також, були видалені деякі перевірки.

Файл бібліотеки до видалення містив 122971 символ. Модифікований файл містить 89522 символи. Отже, розмір скоротився приблизно на 28 відсотків.

Проте такі виміри не враховують той факт, що в бібліотеці наявні коментарі, а також символи, що не є необхідними для виконання кодом покладених на нього функцій, а потрібні лише для полегшення розуміння коду людиною, а також довгі назви функцій, з цією ж метою.

Для оптимізації був використаний сервіс [http://jscompress.com/]. Оптимізований код початкової бібліотеки 36668 символів. Розмір бібліотеки з видаленими функціями, що не є необхідними для даної веб сторінки складає 25195 символів. Тобто розмір файлу скоротився на 32 відсотки.

Проте даний сервіс не скорочує назви функцій для того, щоб не руйнувати інтерфейс бібліотеки. Але, в нашому випадку, система повинна мати можливість робити таку оптимізацію. Також система повинна перевіряти всі розгалуження коду та видаляти ті, які не будуть використовуватись в документі, що дозволить ще більше збільшити виграш в скороченні об’єму даних, що завантажуються.

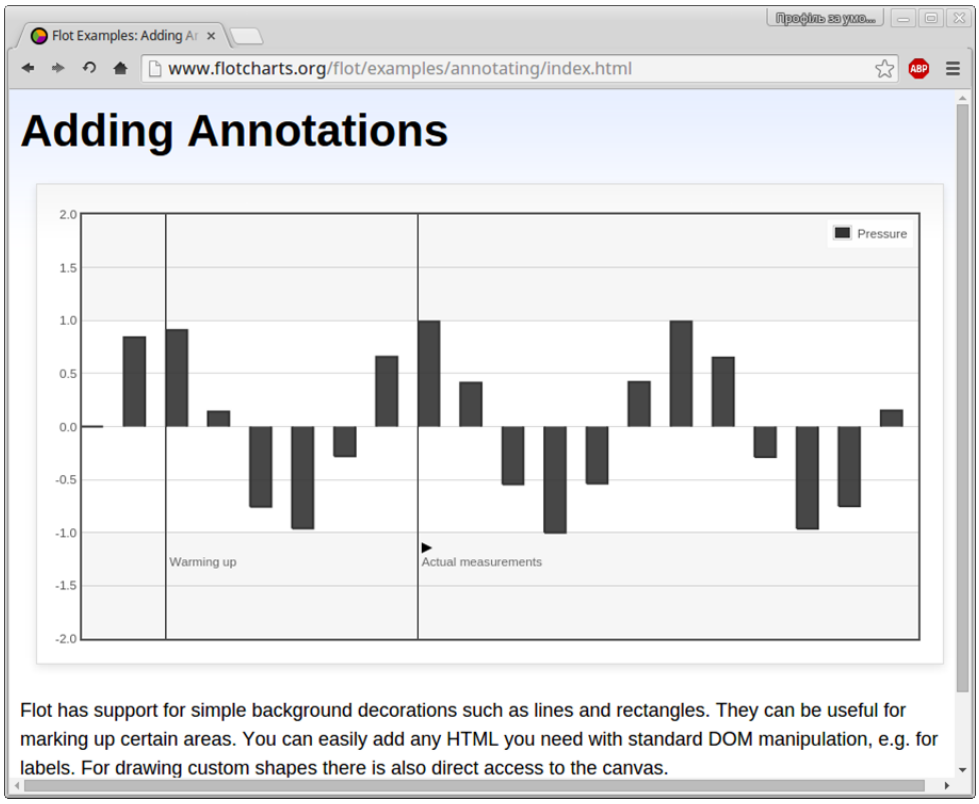


Рис.1 Стовпчикова діаграма з аннотаціями

Тепер оцінимо виграш в часі. Для цього, було використано Google Drive [https://drive.google.com] в якості хоста. Створимо таку ієрархію файлів:

```
├── annotating
│   └── index.html
├── examples.css
├── jquery.flot.js
└── jquery.flotO.js
```

Де index.html — це сам документ, examples.css — файл каскадної таблиці стилів, jquery.flot.js — файл, що містить бібліотеку з вирізаними зайвими функціями і jquery.flotO.js — файл, що містить оригінальний файл бібліотеки.

Для оцінки часу було використано Chrome DevTools.

Розмір оригінальної бібліотеки 31.3 KB. Розмір модифікованої бібліотеки 23.3 KB.

Було проведено 3 досліди. Результати дослідів наведені в таблицях 1, 2 та 3.

Табл. 1. Час завантаження скрипта

№ експерименту	Модифікована бібліотека (с)	Оригінальна бібліотека (с)
1	494	759
2	347	439
3	394	180
Середнє	411.(6)	459.(3)

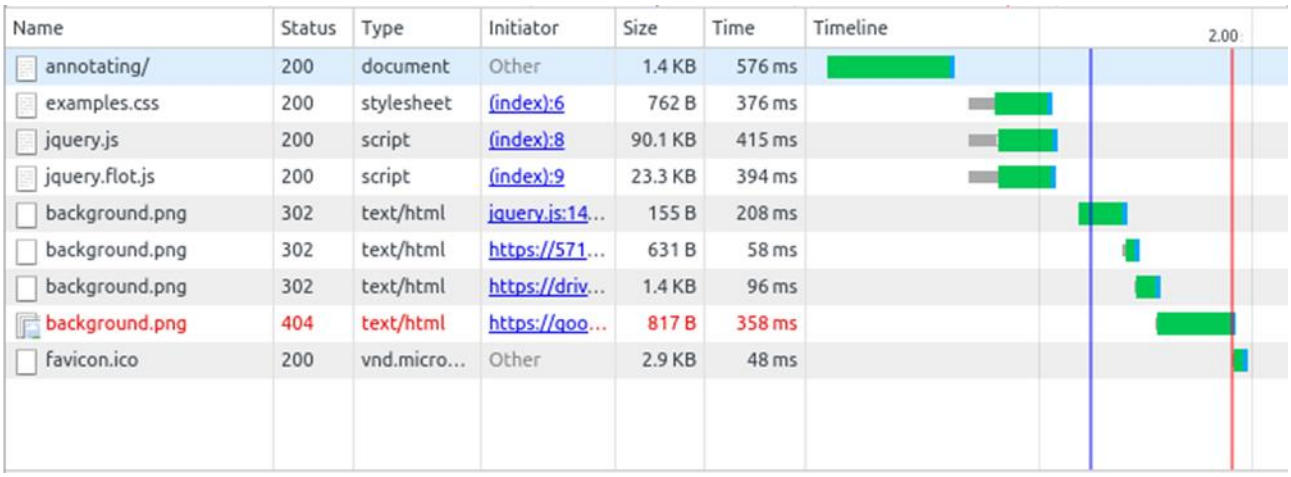
Табл. 2. Час завантаження DOM

№ експерименту	Модифікована бібліотека (с)	Оригінальна бібліотека (с)
1	1.35	1.71
2	1.24	1.37
3	1.24	0.633
Середнє	1.27(6)	1.237(6)

Табл. 3. Час завантаження документу

№ експерименту	Модифікована бібліотека (с)	Оригінальна бібліотека (с)
1	2.08	2.41
2	1.86	1.99
3	1.91	1.24
Середнє	1.95	1.88

Поглянемо на рис. 2.



Name	Status	Type	Initiator	Size	Time	Timeline
annotating/	200	document	Other	1.4 KB	576 ms	
examples.css	200	stylesheet	(index):6	762 B	376 ms	
jquery.js	200	script	(index):8	90.1 KB	415 ms	
jquery.flot.js	200	script	(index):9	23.3 KB	394 ms	
background.png	302	text/html	jquery.js:14...	155 B	208 ms	
background.png	302	text/html	https://571...	631 B	58 ms	
background.png	302	text/html	https://driv...	1.4 KB	96 ms	
background.png	404	text/html	https://goo...	817 B	358 ms	
favicon.ico	200	vnd.micro...	Other	2.9 KB	48 ms	

Рис.2 Часова діаграма завантаження

Хоча новий файл завантажується на 10.2% швидше в порівнянні з початковим, це не вплинуло на швидкість завантаження документу. Після завантаження HTML-сторінки виконуються асинхронні запити на три файли: examples.css, jquery.js та jquery.flot.js. На час завантаження документу впливає запит, який виконується найдовше. Отже, система повинна мати змогу оброблювати і бібліотеку jQuery. Без цього збільшення швидкості завантаження буде незначним. Потрібно також об'єднувати необхідні файли в один у випадку, якщо документ створюється один раз і потім не буде змінюватись.

У зв'язку з наявністю виграшу в швидкості завантаження за рахунок відмови від універсальності та винесення спільного коду, така система буде найбільше підходити для випадку, коли користувач буде переглядати один документ, що буде містити однотипні графіки.

Така ситуація може виникнути у випадку використання певного типу конструктора або редактора документів, коли користувач має можливість додавати графіки, налаштовувати їх, додавати інтерактивність, а потім використовувати цей документ для звітів або презентацій. Тоді такий конструктор матиме певний набір бібліотек JavaScript і матиме усі необхідні умови, щоб надати користувачу саме мінімальний набір даних та коду для відображення документу. Кешування буде відбуватись в рамках одного документа, а універсальність в такому контексті, не вимагається.

Якщо це буде сайт з великою кількістю документів, які використовують можливості однієї бібліотеки, то така система втрачає сенс, оскільки файл бібліотеки просто буде кешуватись в браузері і в разі необхідності буде завантажений не з мережі, а з локального кешу для різних документів. Тому, загальний час завантаження кількох сторінок буде меншим.

Отже, якщо виконувати таку роботу на стороні сервера, то користувач отримає мінімальний необхідний набір даних, для відображення кінцевого результату.

Окрім того, що таке рішення пришвидшить завантаження сторінки без втрати інтерактивності, воно буде "бібліотеконезалежним", тобто таким, яке дозволить використовувати будь-яку наявну бібліотеку JavaScript, яка має хорошу модульну структуру.

Якщо включити код бібліотеки в документ, то буде необхідний лише один запит до сервера, але тоді втратиться перевага кешування, і для кожного нового документу необхідно буде завантажувати один і той же код заново. З іншого боку, можна залишити оброблений файл спільним для кількох документів. Проте, чим більшою буде їх різноманітність, тим менший буде виграш.

Таким чином алгоритм мінімазації даних буде мати вигляд рис.3.

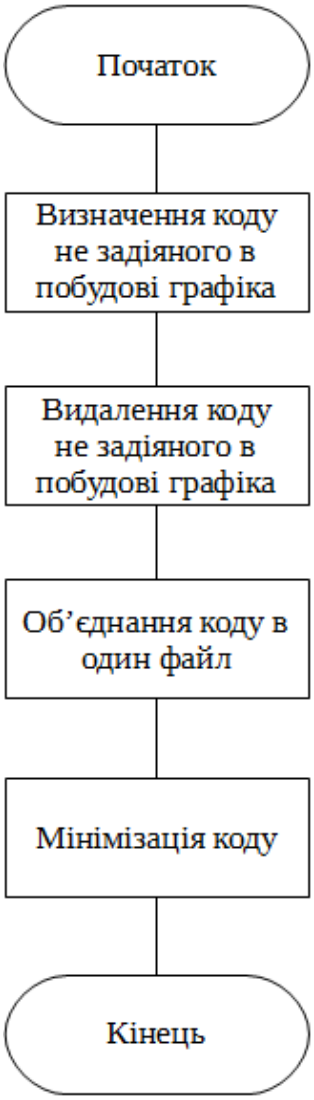


Рис. 3 Алгоритм обробки документу

Висновок

Запропоновано спосіб зменшення часу та об'єму даних, що передаються за рахунок попередньої обробки веб-сторінки на стороні сервера. Проведені експерименти показали, що застосування такого способу дозволяє скоротити об'єм файлу, який оброблюється на 30%. Це дозволило знизити час, необхідний для завантаження файлу, на 10%.

Використання цього способу не вплине на кінцевий вигляд сторінки. Тобто рішення дозволяє зберегти інтерактивність. Також запропонований спосіб є універсальним. Тобто таким, який буде працювати з будь-якою JavaScript бібліотекою в якій наявний код, що не є необхідним для побудови сторінки.

Найкраще спосіб працює в контексті одного документу, з відносною одноманітністю графіків та невисоким використанням функцій бібліотеки.

Недоліком способу є те, що якщо збільшується використання можливостей бібліотеки, знижується його ефективність в зв'язку зі зростанням кількості коду, що використовується.

Подальше вдосконалення способу буде направлене на зменшення цього недоліку за рахунок використання завантаження за вимогою (lazy loading).

Список посилань

1. Welcome To JFreeChart! [Електронний ресурс] // Режим доступу: http://www.jfree.org/jfreechart/
2. Let the computer in the cloud build your charts. [Електронний ресурс] // Режим доступу: https://code.google.com/p/charts4j/
3. Attractive JavaScript plotting for jQuery [Електронний ресурс] // Режим доступу: http://www.flotcharts.org/
4. A powerful and easy to use charting library for Ember.js. [Електронний ресурс] // Режим доступу: http://addepar.github.io/ember-charts/#/ember-charts/overview
5. Full web stack no browser required. [Електронний ресурс] // Режим доступу: http://phantomjs.org/
6. HtmlUnit. [Електронний ресурс] // Режим доступу: http://htmlunit.sourceforge.net/
7. Insanely fast, headless full-stack testing using Node.js [Електронний ресурс] // Режим доступу: http://zombie.js.org/
8. ENVJS – Brining the browser [Електронний ресурс] // Режим доступу: http://www.envjs.com/
9. Flot examples: Adding Annotations [Електронний ресурс] // Режим доступу: http://www.flotcharts.org/flot/examples/annotating/index.html

УДК 004.75

ТЕЛЕНИК С.Ф.,
БУКАСОВ М.М.,
КАРНАУХОВ О.К.,
ФІЛІМОНОВ В.Ф.,
МОРГУН М.В.

МОДЕЛІ ОПТИМІЗАЦІЇ БАГАТОРІВНЕВОГО ЗБЕРІГАННЯ ДАНИХ

Розглянуто особливості збереження даних у багаторівневих сховищах в умовах «великих даних». Сформульовано задачу оптимального розташування блоків даних у багаторівневих сховищах з урахуванням технічних характеристик окремих рівнів збереження даних. Запропоновано три формальних моделі задачі, що враховують особливості сховищ та запитів користувачів до даних і політики щодо мінімізації витрат. Запропоновано методи розв'язання цих задач. Розроблено варіант генетичного алгоритму для визначення оптимального розміщення блоків даних на різних рівнях зберігання даних.

The particularities of data storing in multilevel data storages under "big data" conditions are considered. For multilevel data storages the task of data blocks optimal allocation with taking in consideration technical characteristics of individual storage levels is formulated. Three models for different variants of data storage types, user requests and expenditures minimization policies are proposed. Methods of solving these problems were proposed. The variant of genetic algorithm to determine the optimal allocation of data blocks onto different levels of data storage is developed.

Вступ

З кожним роком в умовах глобалізації і проникнення інформаційних технологій (ІТ) у всі галузі людської діяльності породжуються все більші обсяги даних. Відповідно зростають потреби у зборі, збереженні і обробленні великих обсягів даних, відборі з них корисних і прийнятті на їх основі ефективних рішень. Так, в системах управління великих провайдерів інфокомунікаційних сервісів накопичуються терабайти даних про послуги, їх надання, клієнтів, характеристики обладнання і технологій, відмови, технічне обслуговування тощо.

Процес постійного накопичення даних ускладнює роботу з ними, оскільки одночасно скорочується час, відведений на прийняття рішень. Накопиченні дані є дуже цінними для бізнесу, але якщо ними не можна скористатися вчасно, то їхня корисність починає викликати сумніви. Тому необхідно навчитися контролювати великі інформаційні потоки, ефективно опрацьовувати та використовувати накопичені дані. Постає складний комплекс проблем організації і контролю великих динамічних інформаційних потоків даних, збору, збереження і оброблення великих обсягів даних, оперативного забезпечення користувачів адекватною для прийняття рішень інформацією.

Класичний підхід до збереження і оброблення даних у різноманітних інформаційних та інформаційно-управляючих системах все гірше виконує покладені на нього функції.

Великі постачальники програмно-апаратних засобів збору, збереження і оброблення даних, насамперед EMC, Cisco, NetApp, Veritas, пропонують нові рішення на базі концепції "big data" («великих даних») [1, 2]. Важливою особливістю цих рішень є намагання розробити нові або модифікувати відомі системи і технології, щоб адаптувати системи збереження даних (СЗД) під нові більш жорсткі умови.

Система збереження даних – це комплексне програмно-апаратне рішення з організації надійного збереження інформаційних ресурсів і забезпечення гарантованого доступу до них [3].

При роботі з даними у СЗД в сучасних умовах в цих системах визначають наступні загальні властивості:

- доступність;
- захищеність;
- масштабованість;
- місткість;
- керованість.

Також окремі частини або блоки збереженої інформації мають певні статичні і динамічні властивості. До них найчастіше відносять цінність, інтенсивність використання та інші інтегральні характеристики в залежності від задач. Значення і динаміка змін значень визначають життєвий цикл даних, управління яким визначає ефективність використання збереженої ін-

формації і інформаційної системи у цілому. На сьогодні управління життєвим циклом даних – Information Lifecycle Management (ILM) становить невід'ємну складову інформаційних та інформаційно-управляючих систем підприємств і організацій [4].

Загалом ILM охоплює всі процеси – від створення даних до їх видалення – і все, що відбувається з даними у цей період. Призначення ILM полягає в тому, щоб організувати процеси і технології для управління даними з метою підтримання найбільш вигідного і ефективного життєвого циклу інформації. При цьому необхідно враховувати залежність IT-інфраструктури підприємства чи організації від об'єктів бізнесу і зміну значимості даних з плином часу. Дійсно, дані, що є важливими для бізнесу сьогодні, можуть бути майже незначними завтра. Дані досліджень, наприклад [5], свідчать, що приблизно 90 відсотків збереженої інформації можуть ніколи не бути використанні після 90 днів від моменту їх створення. Тому, коли дані старіють, вони повинні бути переміщенні автоматично на дешевші сховища і технології, в залежності від їх релевантності, щоб зберегти кошти і звільнити ресурси сховищ для нових даних [5].

Отже, ILM – це не програмно-апаратне рішення, а організаційно-технологічна концепція, що за рахунок реалізації системи визначених процесів забезпечує:

- стратегію управління даними;
- орієнтацію на бізнес;
- централізацію;
- цілісність;
- гетерогенність;
- оптимізацію ресурсів.

Реалізація ILM включає в себе певний спектр дій. Перш за все, має бути здійснена класифікація самих даних і програм їх оброблення. На наступному кроці визначаються стратегії створення і видалення даних, в залежності від їх класу, об'ємів та інших параметрів. Окремим аспектом виділяють управління процесами збереження потоків вхідних даних, їх збереженням та екстракцією. Логічним завершенням концепції ILM виглядає описання багаторівневого збереження даних на різних видах носіїв, які, в свою чергу, розділяють за параметрами ціни, швидкості доступу, надійності та ін.

Для інтеграції ILM у інформаційну чи інформаційно-управляючу систему необхідно виконати такі дії:

- встановити систему зберігання даних;
- класифікувати дані (рівні, цикли, політики);
- виконати розподіл даних між рівнями;
- автоматизувати основні процеси роботи з даними;
- повністю інтегрувати засоби ILM в усі програми.

Ручний розподіл даних між рівнями потребує значних зусиль аналітиків на кожній окремій реалізації багаторівневої СЗД. Тому все більшої популярності набирає архітектура СЗД з повністю автоматизованим багаторівневим зберіганням – Fully Automated Storage Tiering (FAST).

Наприклад, ієрархія СЗД може мати такі рівні:

- cache;
- SSD;
- SATA;
- IDE;
- Tape Library.

Підхід, при якому забезпечення ефективності системи збереження відбувається за рахунок переміщення між її рівнями (tiers) збереження називається міжрівневим перерозподілом даних (Tiering). Кожен із рівнів характеризується власним набором параметрів, таких як ціна, швидкодія, ємність, захищеність та інші. Зазвичай міжрівневий перерозподіл даних реалізується механізмами переміщення даних між дисками різних рівнів, або між дисками та магнітною стрічкою. Часто він використовується для організації ILM-сховищ даних у відповідності з визначеними статусом даних та рівнем QoS.

Сьогодні найбільшого поширення набули такі рішення для міжрівневого перерозподілу даних:

- EMC FAST;
- EMC FASTcache;
- NetApp FlashCache.

Диски, які найчастіше використовуються в СЗД відрізаються продуктивністю, ємністю, вартістю. Причому за однакової ціни продуктивність та ємність зазвичай величини обернено пропорційні. Так, для дисків SATA характерна більша ємність при меншій продуктивності, для дисків SAS – менша ємність при більшій продуктивності, а для SSD – ще більша продуктивність при ще меншій ємності. У цілому СЗД характеризуються широко вживаним відношенням «IOPS/вартість».

У загальному випадку невелика частина так званих активних даних використовується більшу частину часу роботи, і тому в більшості випадків визначають швидкодію системи. В противагу, значна частина даних належить до так званих «холодних» даних, які впливають на загальну швидкодію набагато менше. Оскільки в окремих системах холодні дані можуть містити частини або блоки, швидкість доступу до яких є критичною, це треба враховувати при проектуванні.

Розглянемо систему, у якій всі дані є рівноцінними і 20% з них становлять активні даних, швидкість доступу до яких визначає загальну продуктивність системи. Очевидно, що такі дані доцільно розмістити на максимально швидкому носієві (хоч і дорогому). Результатом буде суттєвий приріст швидкодії роботи системи у цілому.

Саме така ідея лежить в основі технологій міжрівневого перерозподілу даних. Якщо певний тип даних є активним, він автоматично переноситься на більш дорогі та швидкі носії, що призводить до підвищення продуктивності. Одночасно менш активні дані переносяться на менш продуктивні та дешевші типи дисків. Таким чином можна контрольовано підвищувати співвідношення IOPS/вартість.

Для великих систем розбиття на рівні є набагато складнішою задачею, оскільки сьогодні велика частина застосувань розташована в центрах обробки даних (ЦОД), а розподілені застосування розташовуються одразу у декількох з них. Також такі застосування можуть використовувати не одну СЗД, а декілька, що додатково ускладнює розбиття усієї множини СЗД на рівні.

Кожен ЦОД має свої СЗД, сервери застосувань та інші компоненти IT-інфраструктури, що дозволяє забезпечувати швидкий доступ застосувань до інформації, що розташована безпосередньо у СЗД цього ЦОД. Але якщо система розподілена, то застосування має отримувати інформацію, що розміщена у різних ЦОД, а швидкість доступу до даних сильно залежить від відстані до інших ЦОД, швидкості, відгуку і інших параметрів мережі.

Говорячи про ЦОД, важливо зазначити його основні властивості, можливості та структуру. Узагальнена структура СЗД з використанням декількох ЦОД наведена на рисунку 1.

ЦОД може складатися з великої кількості різнотипних сховищ (в тому числі готових рішень від відомих компаній, таких як EMC). Сховища, в свою чергу, складаються з стійок дисків та окремих накопичувачів.

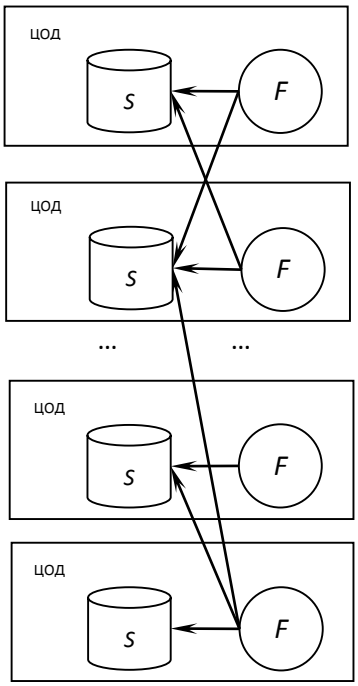


Рис. 1. Схема IT-інфраструктури. S - сховище, F - застосування

Однією з найважливіших характеристик СЗД, ЦОД і сховищ є їх живучість і надійність. Основним з відомих методів підвищення цієї характеристики лишається реплікація даних в кілька незалежних сховищ. В ідеалі, місця збереження дубльованих даних мають бути залежні від різних локальних чи глобальних мереж, а також різних джерел живлення. Важливо розуміти, що дублюватися, в залежності від потреб, можуть як окремі дані, так і диски, стійки, сховища в цілому.

Загальновідомо, що всі бізнес-рішення мають акцент на зменшенні витрат без втрат якості. Ще одним способом економії, а значить і оптимізації, є знеструмлення окремих сховищ (стійок, дисків) з непотрібними в певний час даними. Наприклад нічне виключення пристроїв, що зберігають «холодні» дані, або дані, які точно не будуть потрібні у неробочий час. Маємо ще одну проблему – аналіз і подальше налаштування системи таким чином, щоб відключення в цілях економії не завадили бізнес-процесам, а також налаштування самого процесу знеструмлення без фатальних наслідків, оскільки ЦОД, окрім самих носіїв має сервери налаштувань та іншу інфраструктуру.

Стосовно оптимізації розміщення даних у розподілених системах, очевидно, що чим швидше застосування отримує інформацію з СЗД, тим швидше воно працює. Тому для покращен-

ня роботи системи доцільно розміщувати дані на носії, до яких має швидший доступ з врахуванням конкретного застосування, що їх використовує. Тобто постає проблема оптимального розташування даних у розподіленій СЗД для покращення показників швидкодії певного застосування. І, на відміну від системи, що складається з однієї СЗД і одного серверу застосувань, виникає ситуація, коли вигідніше тримати інформацію на менш швидких носіях, але менших за часом доступу до серверу застосувань. В такій системі також важливо враховувати інтенсивність використання даних окремими серверами застосувань, щоб правильно розрахувати оптимальне розташування.

Для серверів застосувань розташованих в одному ЦОД, можливо розбити СЗД з різних ЦОД на рівні за такими критеріями:

- Швидкість доступу до сховища. Сховище виступає як одиниця.
- Продуктивність носія. Диск виступає як одиниця.
- Швидкість доступу до носія. Диск виступає як одиниця.

Під одиницею мається на увазі нероздільна частина рівня.

Для серверів застосувань, що розташовані в різних ЦОД:

- Сума добутку показників швидкості доступу до сховища і інтенсивності звернень серверу застосувань з яким виконується обмін. Сховище – одиниця.
- Сума добутку показників швидкості доступу до носія і інтенсивності звернень серверу застосувань з яким виконується обмін. Сховище – одиниця.

На жаль, формальні моделі і відповідні методи розв’язання задач, пов’язаних з підвищенням ефективності застосування багаторівневих систем збереження даних, у літературних джерелах не описані. Тому виникає нагальна потреба у таких моделях і методах, насамперед оптимального переміщення даних між сховищами з різними технічними характеристиками.

Мета дослідження

Метою дослідження є підвищення ефективності використання багаторівневих систем збереження даних за рахунок розроблення і використання моделей і методів розв’язання задачі оптимального переміщення блоків даних між дисками сховищ з різними технічними характеристиками.

Математична модель

Розглянемо випадок, коли функції застосування звертаються до блоків даних таким чином, що кожна функція може звертатись до декількох блоків, а до кожного блоку можуть звертатись декілька функцій. При цьому, кожен блок даних має бути розміщеним у одному і лише одному сховищі даних. Модель доступу до наведена на рисунку 2.

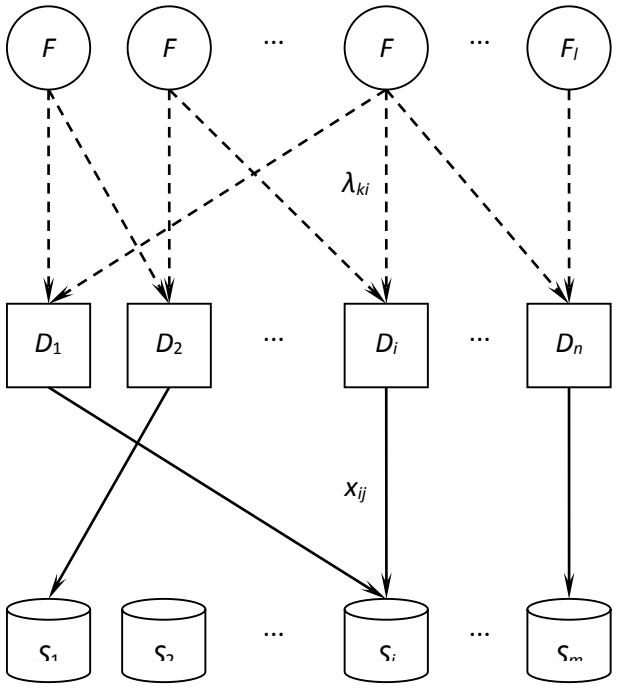


Рис. 2. Модель доступу до даних

Введемо такі позначення:

F_k – функції застосувань, $k = 1, \dots, l$;

μ_k – інтенсивність викликів функції F_k;

D_i – блоки даних, які використовуються застосуваннями, $i = 1, \dots, n$;

d_i – розмір блоку даних D_i;

λ_{ki} – інтенсивність звернень функції F_k до блоку даних D_i;

S_j – фізичні сховища, на яких розташовуються блоки даних, $j = 1, \dots, m$;

t_j – час довільного доступу до сховища S_j;

s_j – розмір сховища S_j;

x_{ij} – булева змінна, що визначає, чи розташовано блок D_i даних у сховищі S_j;

Оскільки кожний блок даних може бути розташований лише у одному сховищі, має виконуватись умова:

$$\sum_{j=1}^{m} x_{ij} = 1,\ i = 1, \dots, n. \qquad (1)$$

Крім того, сума розмірів блоків даних не повинна перевищувати розмір сховища, на якому вони розташовані:

$$\sum_{i=1}^{n} d_i \cdot x_{ij} \le s_j \text{ , } j = 1, \dots , m. \quad (2)$$

Задача 1. В умовах обмежених ресурсів системи збереження даних і відсутності жорстких вимог до часу відгуку функцій, виникає задача оптимального розміщення блоків даних по дисках таким чином, щоб середня продуктивність системи була максимальною, тобто середній час доступу до даних був мінімальним.

Середня інтенсивність звернень до блоку даних D_i може бути представлена наступним виразом:

$$\sum_{k=1}^{l} \mu_k \lambda_{ki} \text{ .} \quad (3)$$

Тоді інтенсивність звернень до сховища S_j буде визначатись виразом:

$$\sum_{i=1}^{n} \sum_{j=1}^{m} \left(\sum_{k=1}^{l} \mu_k \lambda_{ki} \right) \cdot x_{ij} \text{ ,} \quad (4)$$

а критерій мінімізації середнього часу доступу можна записати наступним чином:

$$\min \sum_{i=1}^{n} \sum_{j=1}^{m} \left(\sum_{k=1}^{l} \mu_k \lambda_{ki} \right) \cdot x_{ij} \cdot t_j \text{ .} \quad (5)$$

Тоді задачу мінімізації середнього часу доступу можна сформулювати наступним чином: мінімізувати (5) при виконанні обмежень (1), (2).

Задача 2. Розглянемо випадок, коли є обмеження на час відгуку при викликах функцій додатків. У найгіршому випадку при виконанні функції доступ до блоків даних буде виконуватись послідовно. Позначимо максимально допустимий час доступу до даних функції F_k як T_k. Вочевидь, він має перевищувати сумарний час доступу до усіх блоків даних, які використовуються даною функцією. Тобто,

$$\sum_{i=1}^{n} \sum_{j=1}^{m} \lambda_{ki} \cdot x_{ij} \cdot t_j \le T_k \text{ , } k = 1, \dots , l. \quad (6)$$

Тоді задачу мінімізації середнього часу доступу можна сформулювати наступним чином: мінімізувати (5) при виконанні обмежень (1), (2), (6).

Задача 3. У випадках, коли сховище даних побудовано за концепцією MAID (massive array of idle disks) [5], яка передбачає автоматичне відключення живлення накопичувачів, які тривалий час не використовуються, природньою постає задача розміщення блоків даних у сховищах якнайбільш щільно, при цьому саме таким чином, щоб вивільнити частину накопичувачів і за рахунок їх вимкнення мінімізувати витрати на їх живлення та охолодження.

Позначимо енергоспоживання сховища S_j як e_j, $j = 1, \dots , m$.

Ознакою того що у сховищі S_j не розміщено жодного блоку даних буде визначати наступний вираз:

$$\prod_{i=1}^{n} \overline{x_{ij}} = 1, \; j = 1, \dots, m \text{ .} \quad (7)$$

Для мінімізації енергоспоживання потрібно розмістити блоки даних таким чином, щоб можна було вимкнути частину накопичувачів з максимальним енергоспоживанням, тобто:

$$\max \sum_{j=1}^{m} e_j \prod_{i=1}^{n} \overline{x_{ij}} \text{ .} \quad (8)$$

Тоді задачу мінімізації енергоспоживання можна сформулювати наступним чином: максимізувати (8) при виконанні обмежень (1), (2), (6).

Наведені задачі являють собою лінійні задачі булевого програмування і при невеликій кількості незалежних змінних можуть бути ефективно розв’язані шляхом застосування точних методів [6].

Якщо ж розмірність задачі висока, то для її розв’язання найбільш доцільно буде запропонувати використання різних варіантів генетичного алгоритму (ГА) [7]. При цьому пропонується наступна схема кодування генів.

Оскільки кожний блок даних може бути розташований не більше ніж у одному сховищі, для кодування генів перейдемо від матриці $n \times m$ булевих змінних x_{ij} до вектора довжини n дискретних змінних $y_i \in [1, m]$, кожний елемент якого містить номер сховища S_j, (j=1, ... , m), у якому розміщений відповідний блок даних (рис. 3).

Такий спосіб кодування генів дозволяє, по-перше, скоротити розмірність задачі в m разів, а по-друге, забезпечити автоматичне виконання обмежень (1), оскільки така схема кодування гарантує, що кожний блок даних буде розташований у одному і лише одному сховищі. При цьому операція мутації фактично буде відповідати переносу блоку даних з одного сховища у інше, а операція кросинговеру – обміну декількох блоків даних між сховищами.

$$x_{ij} = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \longrightarrow y_i = \begin{bmatrix} 3 \\ 2 \\ 4 \\ 2 \\ 1 \\ 3 \\ 1 \\ 1 \end{bmatrix}$$

Рис. 3. Приклад кодування генів

Для розв'язання подібного класу задач добре себе зарекомендував керований генетичний алгоритм (КГА) [8].

Висновки

У статті розглянуто підхід до підвищення продуктивності багаторівневої системи збереження даних, що базується на періодичному переміщенні даних між носіями різної продуктивності. Запропоновані три моделі оптимального розміщення блоків даних за різними критеріями, що враховують статистику звернень до інформації, обмеження на ресурси і бізнес-вимоги до системи. Ці моделі можуть бути використані для підвищення швидкодії систем збереження даних. Розв’язання поставлених задач виконується на основі відомих точних методів або запропонованого варіанта генетичного алгоритму в залежності від кількості вільних змінних.

Перелік посилань

1. Mayer-Schönberger V. Big Data: A Revolution That Will Transform How We Live, Work, and Think / V. Mayer-Schönberger, K. Cukier. – Houghton Mifflin Harcourt, 2013. – 240 p.
2. Marz N. Big Data: Principles and Best Practices of Scalable Realtime Data Systems / N. Marz, J. Warren. – Manning Publications Company, 2015. – 328 p.
3. Schulz G. Cloud and Virtual Data Storage Networking / G. Schulz. – CRC Press Taylor & Francis Group, 2011. – 370 p.
4. Information Lifecycle Management [Електронний ресурс]. – 2005. – Режим доступу до ресурсу: http://i.bnet.com/whitepapers/sun00.pdf
5. Colarelli D. Massive arrays of idle disks for storage archives / D. Colarelli, D. Grunwald. // Proceedings of the 2002 ACM/IEEE conference on Supercomputing. – 2002. – С. 1-11.
6. Зайченко Ю.П. Дослідження операцій / Ю.П. Зайченко – [7-е видання, перероблене та доповнене.] – К.: Видавничий Дім «Слово», 2006. – 816 с.
7. Goldberg D.E. Genetic algorithms in search, optimiza-tion, and machine learning / D.E. Goldberg. – Addison-Wesley, 1989. – 412 p.
8. Теленик С.Ф. Генетичні алгоритми вирішення задач управління ресурсами і навантаженням центрів оброблення даних / С.Ф. Теленик, О.І. Ролік, М.М. Букасов, С.А. Андросов // Автоматика. Автоматизація. Електротехнічні комплекси та системи. – 2010. – № 1(25). – С. 106-120.

УДК 004.021, 681.324

МИНУХИН С.В.

МЕТОДЫ РАСПРЕДЕЛЕНИЯ ЗАДАНИЙ МЕЖДУ ПРОЦЕССОРАМИ ВЫЧИСЛИТЕЛЬНОГО КЛАСТЕРА, ИСПОЛЬЗУЮЩИМИ ТЕХНОЛОГИИ МАСШТАБИРОВАНИЯ ЧАСТОТЫ И НАПРЯЖЕНИЯ

Рассмотрены методы распределения независимых заданий, одновременно поступающих на выполнение на параллельные процессоры вычислительного кластера. Предложены методы распределения заданий для однородных и неоднородных кластерных архитектур, отличающиеся предварительной сортировкой заданий входного потока. Для распределения заданий использованы подходы, базирующиеся на определении текущего значения коэффициента загрузки процессоров кластера и методе Round Robin, а также методы масштабирования частоты и напряжения одиночных процессоров. Приведены результаты моделирования исследуемых методов распределения заданий, подтверждающие их эффективность для обеспечения требуемого уровня балансировки загрузки для однородных и неоднородных вычислительных кластеров.

The methods of scheduling of independent tasks, which are arrival at the same time to perform on the parallel processors of computing cluster are developed. The methods of job distribution for homogeneous and heterogeneous cluster architectures, different previously sorted of the input stream. To distribute the task to use the approach based on determining the current value of the utilization of processor and based on the Round Robin method which utilized embedded technologies of scaling the frequency and voltage of single processors. The results of the simulation study methods of tasks distribution are given which proving their effectiveness to achieve the desired level of load balancing for computing clusters with homogeneous and heterogeneous architecture.

1. Введение

Одной из актуальных задач эксплуатации вычислительных ресурсов (кластеров, ЦОД) является обеспечение эффективности функционирования распределенных вычислительных систем. Для их решения необходимы методы, обеспечивающие требуемый уровень загруженности вычислительных ресурсов, качество обслуживания пользователей (минимизацию времени или стоимости выполнения заданий), минимизацию затрат на их содержание и эксплуатацию, включающую оптимизацию энергопотребления процессорами, полученную на основе технологий масштабирования частоты и напряжения. В общем случае эти задачи связаны между собой: минимизация времени выполнения заданий (времени работы процессора) обеспечивает качество обслуживания, однако снижение при этом уровня энергопотребления процессора не всегда представляется возможным, так как уменьшение времени выполнения задания требует использования более производительных режимов его работы. Важным при этом является поддержка штатного режима эксплуатации процессора (узла) кластера, так как его нарушение приводит к увеличению затрат на использование вспомогательного оборудования (например, системы охлаждения). С другой стороны, простаивание процессора приводит к уменьшению его коэффициента загрузки (использования) и увеличению холостого времени его работы. При этом при обеспечении работы процессора в штатном режиме также возникает задача ограничения коэффициента загрузки, который определяется некоторым пороговым значением (в пределах 80 – 90 % от пиковой). В условиях высокой интенсивности потоков заданий приведенные задачи дополняются необходимостью равномерного распределения заданий между параллельно работающими процессорами вычислительного кластера.

Целью данной работы является усовершенствование, исследование и сравнительный анализ методов распределения выполняющихся на параллельных процессорах вычислительного кластера заданий, имеющих произвольные длительности и директивные сроки, которые обеспечивают балансировку загрузки. Данная работа представляет собой дальнейшее развитие и исследование методов распределения заданий на параллельных, использующих технологии масштабирования частоты и напряжения, процессорах кластера, предложенных в работах [1, 3].

2. Постановка задачи

Пусть на вычислительный кластер, состоящем из m процессоров (узлов), одновременно

поступает n независимых заданий t_j (l_j, d_j), описываемых длительностью l_j и директивным сроком выполнения d_j.

Процессоры вычислительного кластера имеют разные штатные частоты и напряжения работы, причем, в зависимости от уровня их загрузки и характеристик поступающих на обработку потока (пакета) заданий, на каждом из них могут быть использованы режимы масштабирования частоты и напряжения, переключения между которыми управляются операционной системой.

Требуется разработать методы распределения заданий между параллельно работающими процессорами вычислительных кластеров с однородной и неоднородной архитектурой, обеспечивающие максимальное значение коэффициента балансировки загрузки процессоров β, определяемого по формуле

$$\beta = 1 - \frac{d}{k_{aver}}, \qquad (1)$$

где

$$k_{aver} = \frac{\sum_{i=1}^{m} k_i}{m}, \; d = \sqrt{\frac{\sum_{i=1}^{m} (k_{aver} - k_i)^2}{m-1}},$$

k_{aver} – средний коэффициент загрузки всех процессоров; k_i – коэффициент средней загрузки i-го процессора.

Коэффициент средней загрузки i-го процессора на каждом шаге распределения заданий рассчитывается как отношение времени завершения текущего (последнего) задания к его директивному сроку.

Методы решения поставленной задачи базируются на выполнении следующих этапов.

Этап 1. Задания сортируются по правилам, использующим их длительность, директивный срок выполнения и плотность [1, 3].

Этап 2. Осуществляется распределение заданий на процессоры (узлы) вычислительного кластера с целью обеспечения их равномерной загрузки, включающее построение допустимых расписаний выполнения заданий на всех процессорах, доставляющих максимум (1).

Этап 3. После распределения заданий осуществляется их выполнение с учетом методов масштабирования частоты и напряжения для одиночного процессора [2].

3. Методы распределения заданий между процессорами вычислительного кластера с однородной архитектурой

Метод 1 [1, 3].

Этап 1. Сортируем все одновременно поступившие на выполнение задания в порядке убывания их длительности (Largest Job First, LJF) [1, 3]. Если таких заданий несколько, первым располагается задание, имеющее меньший директивный срок выполнения.

Этап 2. Распределяем множество заданий, равное количеству процессоров, по методу Round Robin. Определяем коэффициент загрузки каждого процессора и время завершения выполнения задания.

Этап 3. Распределяем из оставшихся нераспределёнными количество заданий, равное количеству процессоров, согласно следующей процедуре: самое длинное задание распределяется на процессор, имеющий наименьшее текущее значение коэффициента средней загрузки; если таких процессоров несколько, выбирается тот, на котором время завершения последнего задания наименьшее. Следующее задание распределяется на процессор с большим коэффициентом средней загрузки и т. д. до тех пор, пока все задания текущего множества не будут распределены.

Этапы 2, 3 повторяются до тех пор, пока все задания входного потока не будут распределены. После этого для каждого из процессоров строится допустимое расписание выполнения заданий с использованием методов масштабирования частоты одиночного процессора [2].

Метод 2.

Этап 1. Все задания, одновременно поступившие на выполнение, сортируются в порядке убывания величины их плотности (Highest Density First, HDF). Если таких заданий несколько, первым располагается задание, имеющее большую длительность.

Этап 2. Множество заданий, равное количеству процессоров, распределяется между процессорами по методу Round Robin. Рассчитывается текущий коэффициент загрузки каждого процессора и время завершения выполнения заданий.

Этап 3. Оставшиеся нераспределёнными задания в количестве, равном количеству процессоров, распределяются согласно следующей процедуре: первым распределяется задание с наибольшей плотностью на процессор, имею-

щий наименьшее текущее значение среднего коэффициента загрузки; если таких процессоров несколько, выбирается тот, на котором время завершения последнего задания наименьшее. Следующее задание распределяется на процессор с большим коэффициентом использования и т. д. до тех пор, пока все задания текущего множества не будут распределены.

Этапы 2, 3 повторяются до тех пор, пока все задания входного потока не будут распределены. После этого для каждого из процессоров строится допустимое расписание выполнения заданий с использованием методов масштабирования частоты одиночного процессора [2].

Метод 3.

Этап 1. Все задания, одновременно поступившие на выполнение, сортируются в порядке убывания их длительности (LJF). Если таких заданий несколько, первым располагается задание, имеющее меньший директивный срок.

Этап 2. Множество заданий, равное количеству процессоров, распределяется по методу Round Robin.

Этап 2 повторяется до тех пор, пока все задания входного потока не будут распределены на процессоры.

После этого для каждого из процессоров строится допустимое расписание выполнения заданий с использованием методов масштабированием частоты одиночного процессора [2].

4. Методы распределения заданий между процессорами вычислительного кластера с неоднородной архитектурой

Метод 1.

Этап 1. Все задания, одновременно поступившие на выполнение, сортируются в порядке убывания их длительности (LJF). Если таких заданий несколько, первым располагается задание, имеющее меньший директивный срок выполнения.

Этап 2. Процессоры сортируются в порядке убывания их относительной скорости (частоты) [1, 3]; если процессоров с одинаковой скоростью несколько, то выбирается любой из них.

Этап 3. Множество заданий, равное количеству процессоров, распределяется согласно следующей процедуре: задание с большей длительностью распределяется на процессор с большей относительной скоростью (частотой). Рассчитывается коэффициент текущей средней загрузки каждого процессора и время завершения выполнения на нем заданий.

Этап 4. Оставшихся нераспределёнными задания в количестве, равном количеству процессоров, распределяются согласно следующей процедуре: первое, то есть самое длинное задание из текущего множества, распределяется на процессор, имеющий наименьшее текущее значение среднего коэффициента загрузки; если таких процессоров несколько, выбирается тот, на котором время завершения последнего задания наименьшее. Следующее задание из множества распределяется на процессор с большим коэффициентом средней загрузки и т. д. до тех пор, пока все задания текущего множества не будут распределены.

Этапы 3, 4 повторяются до тех пор, пока все задания входного потока не будут распределены. После этого для каждого из процессоров строится допустимое расписание выполнения заданий с использованием методов масштабирования частоты одиночного процессора [2].

Метод 2 [1].

Этап 1. Все задания, одновременно поступившие на выполнение, сортируются в порядке убывания величины их плотности (HDF). Если таких заданий несколько, первым располагается задание, имеющее большую длительность.

Этап 2. Множество заданий, равное количеству процессоров, распределяется по методу Round Robin. В этом методе использован подход [4, 5], реализующий идею классификации заданий по их плотностям, отличающийся правилами определения и сортировки заданий по плотностям.

Этап 2 повторяется до тех пор, пока все задания входного потока не будут распределены на процессоры.

После этого для каждого из процессоров строится допустимое расписание выполнения заданий с использованием методов масштабированием частоты одиночного процессора [2].

Метод 3.

Этап 1. Все задания, одновременно поступившие на выполнение, сортируются в порядке убывания их длительности (LJF). Если таких заданий несколько, первым располагается задание, имеющее меньший директивный срок выполнения.

Этап 2. Процессоры сортируются в порядке убывания их относительной скорости (частоты).

Этап 3. Задания в количестве, равном количеству процессоров, распределяются в соответствии с алгоритмом Round Robin.

Этап 3 повторяется до тех пор, пока все задания входного потока не будут распределены на процессоры. После этого для каждого из процессоров строится допустимое расписание выполнения заданий с использованием методов масштабирования частоты одиночного процессора [2].

Метод 4 [1].

Этап 1. Все задания, одновременно поступившие на выполнение, сортируются в порядке убывания их плотности (HDF). Если таких заданий несколько, первым располагается задание, имеющее большую длительность.

Этап 2. Процессоры сортируются в порядке убывания их относительной скорости (частоты).

Этап 3. Множество заданий, равное количеству процессоров, распределяется на них согласно следующей процедуре: задание с большей плотностью распределяется на процессор с большей относительной скоростью (частотой). Далее рассчитывается текущий коэффициент средней загрузки каждого процессора и время завершения выполнения на нем заданий.

Этап 4. Оставшиеся нераспределёнными задания, количество которых равно количеству процессоров, распределяется, следуя процедуре: первое из оставшихся заданий, то есть задание с наибольшей плотностью из текущего множества заданий, распределяется на процессор, имеющий наименьшее текущее значение среднего коэффициента загрузки; если таких процессоров несколько, выбирается тот, на котором время завершения последнего задания наименьшее. Следующее задание распределяется на процессор с большим коэффициентом загрузки и т. д. до тех пор, пока все задания текущего множества не будут распределены.

Этапы 3, 4 повторяются до тех пор, пока все задания входного потока не будут распределены на процессоры. Далее для каждого из процессоров строится допустимое расписание выполнения заданий с использованием методов масштабирования частоты одиночного процессора [2].

5. Экспериментальное исследование методов и анализ полученных результатов

Для проведения вычислительных экспериментов, моделирующих распределений заданий между процессорами, использующими технологии масштабирования частоты и напряжения, выбраны следующие параметры заданий: длительности заданий определялись по равномерному закону распределения в интервале [1,10], директивные сроки рассчитывались путем умножения их длительности на множитель *D*, распределенный по равномерному закону в интервале [1,3], в качестве процессоров для проведения моделирования использованы: для однородного кластера – Intel Core i5, для неоднородного – Intel Core i5 и Intel Xeon E5 2690. Для получения статистически достоверных данных экспериментов на каждую точку результатов расчетов генерировалось 100 наблюдений. В качестве метода масштабирования частоты одиночного процессора выбран метод, предложенный в [2], обеспечивающий наименьший коэффициент загрузки одиночного процессора с масштабированием частоты и напряжения.

Результаты моделирования рассмотренных методов распределения заданий на параллельные процессоры вычислительного кластера с однородной архитектурой приведены в табл. 1, кластера с неоднородной архитектурой – в табл. 2. Они показывают, что для кластера с однородной архитектурой наиболее предпочтительным является метод Round Robin, использующий предварительную сортировку заданий по убыванию их длительности; далее следует метод с балансировкой по текущему значению коэффициента средней загрузки, и потом – метод с предварительной сортировкой по убыванию плотности и балансировкой загрузки на основе текущего значения коэффициента загрузки. Для кластера с неоднородной архитектурой предпочтительным является метод с сортировкой заданий по убыванию плотности и балансировки на основе текущего значения коэффициента загрузки (метод 4), далее следует метод, использующий сортировку по убыванию плотности заданий и процедуру Round Robin (метод 2), а затем с практически одинаковыми результатами располагаются методы 1 и 3.

Табл. 1. Результаты моделирования методов распределения заданий на процессоры однородного кластера

Метод	Количество заданий	Количество процессоров	Диапазон значений d	Диапазон значений β
Метод 1 (LJF)	500	50	0,04 – 0,27	0,90– 0,91
	500	100	0,02 – 0,28	0,90 – 0,91
	1000	100	0,04 – 0,23	0,91–0,92
	5000	100	0,02 – 0,35	0,91–0,92
Метод 2 (HDF)	500	50	0,18 – 0,52	0,88–0,91
	500	100	0,14 – 0,54	0,90–0,92
	1000	100	0,19 – 0,4	0,89–0,92
	5000	100	0,25 – 0,43	0,89–0,91
Метод 3 RR (LJF)	500	50	0,18 – 0,33	0,91–0,92
	500	100	0,16 – 0,34	0,90 –0,93
	1000	100	0,1 – 0,28	0,91–0,93
	5000	100	0,03 – 0,13	0,92–0,94

Табл. 2. Результаты моделирования методов распределения заданий на процессоры неоднородного кластера

Метод	Количество заданий	Количество процессоров	Диапазон значений d	Диапазон значений β
Метод 1 (LJF)	500	50	0,6 – 5,0	0,88–0,93
	500	100	0,5 – 4,3	0,90–0,93
	1 000	100	0,4 – 5,6	0,87–0,93
	5 000	100	0,6 – 4,6	0,88–0,93
Метод 2 (HDF RR)	500	50	0,5 – 5,1	0,88–0,92
	500	100	0,4 – 5	0,88–0,90
	1 000	100	0,4 – 4,8	0,89–0,90
	5 000	100	0,4 – 4,6	0,88–0,92
Метод 3 (RR)	500	50	0,6 – 5,6	0,87–0,90
	500	100	0,5 – 5,6	0,87–0,90
	1 000	100	0,4 – 5,8	0,87–0,92
	5 000	100	0,5 – 5,2	0,87–0,92
Метод 4 (HDF)	500	50	0,6 – 5	0,89–0,93
	500	100	0,4 – 4,8	0,90–0,94
	1 000	100	0,4 – 4,8	0,90–0,94
	5 000	100	0,45 – 4,85	0,90–0,95

6. Выводы

Полученные в данном исследовании результаты позволяют сформулировать следующие выводы.

1. Выбор метода распределения заданий между параллельными процессорами вычислительного кластера в значительной степени зависит от особенностей построения его архитектуры. В случае неоднородной архитектуры требуется учет характеристик заданий и процессоров, влияющих на эффективность распределения.

2. Использование величины текущего значения коэффициента загрузки процессора и сортировки по убыванию плотности позволяет получить для неоднородного кластера более высокий уровень балансировки, чем при использовании процедуры Round Robin.

3. После распределения заданий на процессоры кластера необходимо использовать методы масштабирования частоты процессора, обеспечивающие минимальную загрузку и работу в штатном режиме его эксплуатации для оптимизации энергопотребления.

Список литературы

1. Минухин С.В. Алгоритмы, программная архитектура и информационная технология моделирования методов масштабирования скоростей параллельных процессоров вычислительного кластера / С.В. Минухин, М.И. Сухонос // Вісник НТУ «ХПІ». Серія: Системний аналіз, управління та інформаційні технології. – Х.: НТУ «ХПІ». – 2013. – № 62 (1035). – С.78-92.

2. Минухин С.В. Алгоритмы оптимизации энергопотребления и повышения эффективности процессоров с масштабированием частоты и напряжения гетерогенного кластера / С.В. Минухин, М.И. Сухонос // International Conference Parallel and Distributed Computing Systems PDCS 2013 (Ukraine, Kharkiv, March 13–14, 2013). – 2013. – P. 209-217.
3. Минухин С.В. Энергоэффективные алгоритмы масштабирования скорости процессоров вычислительного кластера / С.В. Минухин // Second International Conference "Cluster Computing" CC 2013 (Ukraine, Lviv, June 3–5, 2013). – P. 131-140.
4. Bell P.C. Multiprocessor Speed Scaling for Jobs with Arbitrary Sizes and Deadlines / Paul C. Bell, Prudence W. H. Wong // Theory and Applications of Models of Computation. Lecture Notes in Computer Science. – 2011. – Vol. 6648. – P. 27-36.
5. Albers S. Speed scaling on parallel processors / S. Albers, F. Müller, S. Schmelzer // Algorithmica. – 2014. – Vol. 68. – Issue 2. – P. 404-425.

УДК 004.724.4 (045)

ДИБРОВА М.А.,
КОГАН А.В.,
ВОРОБЬЕВА А.Л.

СПОСОБ ФОРМИРОВАНИЯ МНОЖЕСТВА ПУТЕЙ В СЕТЕВЫХ ЦЕНТРАХ ДАННЫХ

В работе рассмотрена типовая структура сетевого центра данных. Обоснована целесообразность использования многопутевой маршрутизации в сетевых топологиях Fat Tree. Предложен способ повышения эффективности алгоритма поиска в глубину за счет учета свойства самоподобия топологии Fat Tree. Предложен модифицированный алгоритм поиска в глубину, приведено сравнение его с базовым алгоритмом.

The paper deals with the structure of a typical data center network. Substantiates the desirability of using multipath routing network topologies Fat Tree. A method for increasing the efficiency of the search algorithm in depth by taking into account the properties of self-similarity topology Fat Tree. A modified algorithm for depth-first search, compares it to the baseline algorithm.

1. Введение

В настоящее время в связи с возрастающими потребностями в вычислительной мощности и объемов информации актуальными становятся сетевые центры данных (DCN). Функциональным назначением DCN является высокоскоростная обработка и хранение большого объема данных. Сетевые центры данных (DCN) должны обеспечить высокую пропускную способность и надежность передачи информации. Это, в свою очередь, предъявляет высокие требования к конструированию трафика (TE) в таких системах.

При построении современных DCN в основном используется топология Fat Tree [1], которая является одной из наиболее распространенных топологий для построения распределенных систем, ориентированных на решение высокопроизводительных задач. Топология Fat Tree представляет собой дерево, листьями которого являются вычислительные устройства, а узлами – коммутаторы. При этом у коммутаторов более высоких уровней пропускная способность каналов больше, т.е. связи с другими вершинами более «толстые». Поэтому эта топология и получила название Fat Tree (толстое дерево).

DCN при использовании топологии Fat Tree образуют четырех уровневую структуру [2], на верхнем уровне располагаются основные коммутаторы (Core swiches), уровнем ниже располагаются агрегацийные коммутаторы (Aggregation swiches), затем располагаются граничные коммутаторы (Edle swiches) к которым подключаются вычислительные устройства (Hosts) (рис. 1).

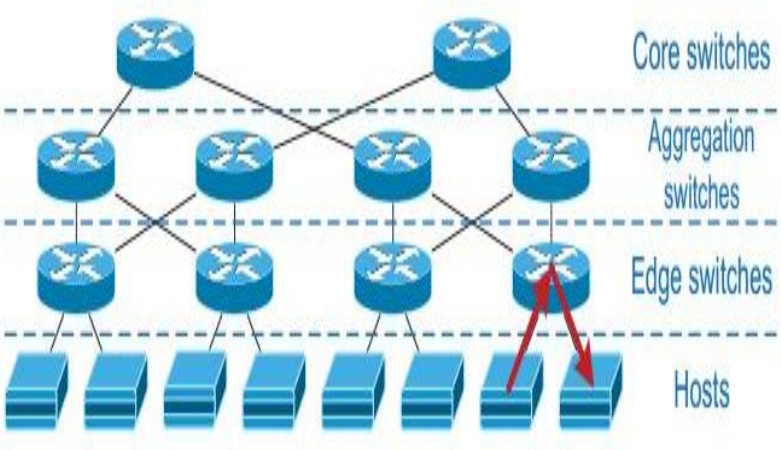


Рис. 1. Топология Fat Tree

В свою очередь, для DCN, состоящих из сотен и тысяч серверов большой размерности, топология Fat tree, на основе принципов самоподобия, преобразуется в так называемую сеть Dragonfly [3]. Учет фрактального характера топологии сети Dragonfly позволяет

упростить процесс конструирования трафика, в частности процесс маршрутизации. Главными требованиями, предъявляемыми к маршрутизации в DCN, являются надежность, минимальная задержка и отказоустойчивость. Этим критериям в большей степени удовлетворяют способы многопутевой маршрутизации, которые нашли широкое применение в DCN [4, 5, 6]. Одним из основных недостатков комбинаторных алгоритмов многопутевой маршрутизации является значительная их временная сложность, которая, как правило, носит нелинейный характер. В работе [7] предложен волновой алгоритм формирования множества путей между произвольными вершинами с линейной временной сложностью. Дальнейшее уменьшение временной сложности связано с учетом топологии графа сети.

2. Модифицированный способ доступа на основе многопутевой маршрутизации

Для топологии Fat tree эффективным является метод поиска в глубину (DFS, сокращение от Depth-first search), в основе которого лежит рекурсивный способ обхода вершин графа [8]. В процессе обхода узлы отмечаются как посещенные и непосещенные, вначале все узлы являются непосященными. Узел, который принимает управляющий пакет отмечается как посещенный и анализирует информацию о смежных с ним узлах. Затем данный узел согласно определенного правила передает пакет одному из неотмеченных смежных с ним узлов. Если у данного узла нет неотмеченных смежных с ним узлов, то он передает управляющий пакет обратно.

К недостаткам базового алгоритма следует отнестито, что он не учитывает:

1) пропускную способность каналов связи;
2) особенности топологии сети.

Существуют три основных стратегии выбора каналов:

• Worst-Fit – выбирается канал с наибольшей доступной пропускной способностью;

• First-Fit – выбирается канал с любой из доступной пропускной способностью, которая отвечает требованиям;

• Best-Fit – выбирается канал с доступной пропускной способностью, которая наилучшим способом отвечает требованиям.

Следует отметить, что при формировании множества путей в DCN с помощью метода DFS наиболее эффективной является стратегия Worst-Fit. Дальнейшее повышение эффективности метода DFS при формировании множества путей связано с учетом особенности иерархической организации DCN и пропускной способности каналов связи между узлами сети.

Метод DFS использует централизованный способ маршрутизации при котором центральный контроллер содержит всю информацию, необходимую для формирования маршрутов. Кроме топологии сети центральный контроллер содержит информацию о пропускной способности каналов связи, которая используется для выбора канала связи на более нижнем уровне или для возврата на уровень выше по дереву. После формирования очередного пути допустимая пропускная способность каналов связи, принадлежащих выбранному пути уменьшается на заданную величину.

На первом шаге алгоритма определяется уровень топологии на котором могут быть объеденены узлы отправитель и получатель информации. Это делается на основании того, что центральный контроллер содержит информацию о топологии сети. Если узлы отправитель и получатель информации подключены к одному и тому же граничному коммутатору, то он напрямую связывает эти узлы. В противном случае путь проходит через несколько коммутаторов, количество которых зависит от взаимного расположения узлов.

Определение уровня соединения узлов позволяет упростить процесс маршрутизации. В этом случае исключаются лишние переходы между уровнями.

Для ускорения перехода между уровнями в модифицированном алгоритме вводится глобальная переменная next, которая указывает направление обхода узлов сети. При next=1 перемещение должно осуществляется вверх по дереву. При next=-1 следует выбирать путь вниз по дереву. При прохождении через уровень на котором могут быть связаны узлы отправителя и получателя значение переменной next становиться равной – 1. Это говорит о том, что обход достиг локального максимума и нужно двигаться вниз по дереву.

Если в результате обхода достигается крайний коммутатор, который не связан с получателем, переменная next принимает значение next=1, что бы вернуться на предыдущий уровень. Использование переменной next является ключевым при

реализации алгоритма DFS, так как позволяет учитывать особенности иерархической топологии.

Таким образом, модифицированный алгоритм DFS состоит из следующей последовательности операций:

1. Начало.
2. Определение уровня соединения пути.
3. Next:=1.
4. Установить узел отправитель текущим узлом.
5. Добавить текущий узел к узлам пути.
6. Если текущий узел не является конечным коммутатором, то переход к пункту 9.
7. Если текущий узел является конечным коммутатором получателя, то переход к пункту 14
8. Установить предыдущий узел текущим.
9. Если достигнут верхний уровень, то установить next = -1.
10. Выбрать следующий узел среди смежных узлов.
11. Если не удалось выбрать следующий узел, то переход к пункту 8.
12. Установить выбранный узел текущим.
13. Переход к пункту 5.
14. Добавить узел получатель к пути.
15. Конец.

3. Моделирование и анализ эффективности предлагаемого способа формирования множества путей в DCN.

С целью анализа эффективности предлагаемого метода формирования множества путей была разработана специальная система моделирования. На рис.2 представлен пример формирования путей между 1 и 5 узлами DCN. Длина *i*-го пути (L_i) определяется количеством переходов между исходной и конечной вершинами и равна *N*-1, где *N* – общее число вершин данного пути.

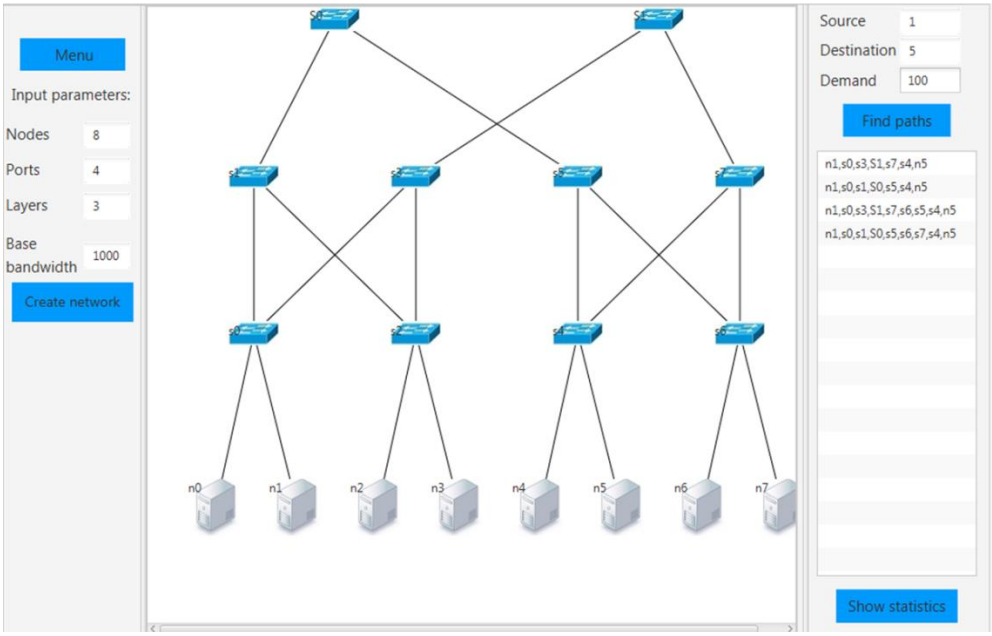


Рис.2. Формирование путей в DCN

В результате работы алгоритма формируется четыре пути между вершинами n_1 и n_5:

1. $n_1,s_0,s_3,S1,s_7,s_4,n_5$; L_1 =6.
2. $n_1,s_0,s_1,S0,s_5,s_4,n_5$; L_2 =6.
3. $n_1,s_0,s_3,S1,s_7,s_6,s_5,s_4,n_5$; L_4=8.
4. $n_1,s_0,s_1,S0,s_5,s_6,s_0,s_4,n_5$; L_4 =8.

В данном случае средняя длина пути L_{cp}=7.

Ниже представлен псевдокод модифицированного алгоритма DFS формирования пути в DCN:

```
DFS(G, a, b, d) {// G: network, a: source, b: destination, d: demand
1 H=necessary-layer-to-connect(G, a, b);
2 path={};
3 u=a; // temp variable indicating current location
4 next=1; // search direction flag, 1: upstream, –1: downstream
5 return SEARCH(u, path, next);
}
SEARCH(u, path, next) {
1 path=path+u;
```

```
2 if(u=b) return true;
3 if ( layer-of(u)=H) next=−1;// reverse search direction after reaching connecting layer
4 if(next=−1&& layer-of(u)=1) return false;// failure at bottom layer
5 neighbors={v| layer-of(v)= layer-of(u)+next, and available bandwidth of link(u, v)≥d};
6 found=false;
7 while (neighbors!=0&&found=false) {
8 v=worst-fit(neighbors); neighbors=neighbors\{v};
9 found=SEARCH(v,path,next);
10 };
11 return found;}
```

В табл. 1 приведена зависимость длины путей от количества узлов DCN, полученная в результате моделирования базового и модифицированного алгоритма маршрутизации.

Таблица 1. Зависимость длины пути от количества узлов DCN

Количество узлов	Длина пути	
	Базовый алгоритм	Модифицированный алгоритм
8	9	7
16	15	7
32	27	7
64	51	7
128	99	7
256	195	7
512	387	7
1024	771	7

Из таблицы видно, что при базовом алгоритме, зависимость длины пути от количества узлов носит линейный характер, а для модифицировнного метода остается постоянной. Это связано с тем, что при базовом алгоритме осуществляется последовательный перебор всех путей. При модифицированном алгоритмы учитывается самоподобная топология сети, что позволяет каждый раз выбирать оптимальный путь и сократить количество итераций. В табл.2 приведена зависимость числа итераций при обычном и модифицированном алгоритме в зависимости от количества узлов DCN. Как видно из табл.2 преимущество предлагаемого алгоритма увеличивется с ростом количества узлов в сети DCN.

Таблица 2. Зависимость числа итераций от количества узлов DCN

Количество узлов	Число итераций	
	Базовый алгоритм	Модифицированный алгоритм
8	8	6
16	14	10
32	26	18
64	50	34
128	98	66
256	194	130
512	386	258
1024	770	514

5. Выводы по работе

В работе предложен и обоснован алгоритм формирования множества путей в DCN. Данный алгоритм является модифицированным алгоритмом поиска в глубину. Предложенный алгоритм максимально учитывает особенности сетевой топологии Fat tree, что способствует более эффективному поиску множества путей передачи данных. В свою очередь применения режима worst-fit позволяет в рамках сетевой топологии Fat tree позволяет обеспечить линейную временную сложность формирования множества путей и обеспечивает лучшую балансировку нагрузки.

Список литературы

1. M. Al-Fares “A scalable, commodity data center network architecture,”/ M. Al-Fares, A. Loukissas, and A. Vahdat, // in Proceedings of the ACM SIGCOMM 2008 Conference on Data Communication. New York, NY, USA: ACM, 2008, p. 6374. [Online]. Available: http://doi.acm.org/10.1145/1402958.1402967
2. Mysore R. N. Portland: a scalable fault-tolerant layer2 data center network fabric. [Text] / Mysore, R. N., A. Pamboris, N. Farrington, N. Huang, P. Miri, S. Radhakrishnan, V. Subramanya, A. Vahdat // ACM SIGCOMM Computer Communication Review – SIGCOMM '09 – ACM New York, NY, USA, 2009. – URL: http://dl.acm.org/citation.cfm?id=1592575.

3. J. Kim “Technology-driven, highly-scalable dragonfly topology,”/ . J. Kim, W. J. Dally, S. Scott, and D. Abts, // in Proceedings of the 35th Annual International Symposium on Computer Architecture, ser. ISCA ’08. Washington, DC, USA: IEEE Computer Society, 2008, p. 7788. [Online]. Available: http://dx.doi.org/10.1109/ISCA.2008.19
4. Eric Jo. A simulation and emulation study of sdn-based multipath routing for fat-tree data center networks. [Text] / Eric Jo, Deng Pan, Jason Liu, Linda Butler // Proceedings of the 2014 Winter Simulation Conference. IEEE Press Piscataway, NJ, USA, 2014. – URL: http://dl.acm.org/citation.cfm?id=2694235.
5. Omair Fatmi Distributed multipath routing for data center networks based on stochastic traffic modeling. [Text] / Omair Fatmi, Deng Pan.// Networking, Sensing and Control (ICNSC), IEEE 11th International Conference. – Miami, FL, USA, 2014. – URL: http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber =6819683.
6. Eun-Sung Jung Distributed Multipath Routing Algorithm for Data Center Networks/ Eun-Sung Jung, Venkatram Vishwanath, Rajkumar Kettimuthu// 2014 International Workshop on Data Intensive Scalable Computing Systems 2014 IEEE , S. – 49-56.
7. Кулаков Ю.А. Разработка и моделирование процесса безопасной многопутевой передачи информации в мобильных сетях / Кулаков Ю.А., Коган А.В., Пирогов А.А. // Вісник НТУУ «КПІ». Інформатика, управління та обчислювальна техніка: збірник наукових праць. – К.: Век+, 2011. – № 54. – С. 145-149.
8. Гуровиц В.М. Алгоритм поиска в глубину. [Електронний ресурс] – Информатика (Теория графов), 2014. – Режим доступу: http://foxford.ru/wiki/informatika/algoritm-poiska-v-glubinu

УДК 004.052.042

МАРКОВСЬКИЙ О.П.,
ВЕЛИКИЙ М.М.

МЕТОД РЕЗЕРВУВАННЯ ТА ПРИСКОРЕНОГО ВІДНОВЛЕННЯ ДАНИХ В СИСТЕМАХ ЇХ ВІДДАЛЕНОГО ЗБЕРІГАННЯ

В статті запропоновано метод резервування та відновлення даних. Ці дані зберігаються на різних віддалених носіях, до *p* з яких постійно або тимчасово втрачено доступ. Висока ефективність запропонованого методу забезпечується тим, що використовується теоретично мінімальна кількість резервних носіїв, яка дорівнює кількості носіїв, до яких втрачено доступ. Детально викладені математична ідея та процедури відновлення даних, доступ до яких втрачено. Розроблений метод ілюстровано прикладами. Наведено теоретичні та експериментальні оцінки запропонованого методу. Розроблений метод має на меті забезпечення високого рівня надійності систем розподіленого зберігання даних.

In this work, a method for backup and restoring data is proposed. The data is stored on different remote disks and temporary or permanent loss of access to any n of them appears. The high efficiency of the proposed method is achieved by using a theoretically proved minimum number of backup copies of the data, which is equal to the number of the disks to which access has been lost. The mathematical idea of proposed method and procedure for recovering of data from access lost storage unit are described in details. A numerical example for developed recovering procedure are given. The theoretical and experimental effectiveness evaluation of the proposed method is demonstrated as well. The proposed method is aimed to ensure a high level of survivability of system of distributed data storage.

Вступ

Швидкий прогрес засобів телекомунікацій та технології комп'ютерних мереж на початку нового тисячоліття призвів до динамічного розвитку систем віддаленого зберігання та обробки даних. Основною перевагою віддаленої обробки інформації в рамках хмарних технологій є можливість надання користувачам значних за обсягом обчислювальних ресурсів та пам'яті [1].

Динамічний розвиток розподілених систем обробки та зберігання даних неможливий без ефективного вирішення ряду наукових задач, чільне місце серед яких посідає забезпечення надійності та безперебійності доступу до віддаленої інформації користувачів. Вирішення цих важливих задач вимагає застосування спеціальних засобів резервування та відновлення даних, до яких втрачено доступ. За умови швидкого зростання об'ємів такої інформації і розширення кола користувачів технологій віддаленого зберігання даних набуває ваги задача підвищення ефективності методів та засобів резервування та відновлення інформації в рамках таких технологій. Це вимагає постійного пошуку нових підходів, методів та розробки засобів забезпечення неперервності доступу до даних шляхом їх відновлення з використанням резервної інформації.

Таким чином, задача підвищення ефективності та розширення функціональних можливостей засобів резервування та відновлення даних при їх зберіганні на віддалених та розподілених носіях є актуальною та практично важливою з огляду на сучасний стан розвитку технологій комп'ютерної обробки інформації.

Аналіз існуючих систем резервування та відновлення даних

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

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

В якості критеріїв ефективності систем резервування найчастіше виступають:

– обчислювальна складність процедури відновлення даних з використанням резервних носіїв;

– кількість носіїв (K_B), дані з яких можуть бути відновлені в разі втрати доступу до них;

– відношення кількості резервних носіїв (K_P) до числа носіїв, дані з яких можуть бути відновлені ($\alpha=K_P/K_B$).

Проведений аналіз літературних джерел [2] показав, що найчастіше причинами втрати доступу до даних віддалених користувачів стає вихід з ладу окремих носіїв (або стирання інформації на них), а також тимчасова нездатність вузлу обслужити запит користувача.

При втраті даних на окремих носіях в результаті виходу їх з ладу, або стирання (помилкового чи цілеспрямованого) даних на них, кількість q таких носіїв залежить від часу t, що пройшов від моменту останнього звертання до них. Тимчасова нездатність вузлу обслужити запит користувача здебільшого пов'язана з перевантаженістю вузла зберігання даних, виходом з ладу його апаратних чи програмних компонентів, економічними чи природними катаклізмами.

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

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

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

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

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

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

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

Доведено також, що існуючі методи резервування даних мають високий рівень надлишковості і що число резервних носіїв може бути зменшено без втрати здатності відновлювати дані.

В роботах [4-7] запропоновано організацію резервування та відновлення даних не більше ніж з трьох носіїв на основі діагональних контрольних сум. Ця організація забезпечує можливість відновлення даних з використанням не більше трьох резервних носіїв, тобто забезпечує теоретичний мінімум інформаційної надлишковості резервування. В роботі [7] запропоновано варіант згаданої організації, який дозволяє відновлювати окремі фрагменти не більше ніж 3-х носіїв, до яких втрачено доступ.

Недоліком організації є те, що вона не забезпечує можливості відновлення даних більш з 3-х носіїв. Аналіз використання систем розподіленого зберігання даних показує, що в ситуаціях, коли носії, на яких зберігаються дані користувача фізично знаходяться в рамках одного вузла, частими є ситуації відмови доступу до більш ніж 3-х носіїв.

Ціллю досліджень є підвищення ефективності резервування та відновлення даних в системах їх віддаленого зберігання за рахунок

зняття обмежень на кількість носіїв, доступ до яких втрачено.

Метод відновлення даних з довільної заданої кількості носіїв

Інформація користувача зберігається на m віддалених носіях. Інформація на кожному з них може розглядатися як така, що складається з n фрагментів. Поняття фрагменту не прив'язано до певної фізичної природи і фрагментом може бути сектор жорсткого диску або блок інформації довжина якого співпадає з розрядністю процесора. Відповідно кожен j-ий фрагмент, $j\in\{1,..,n\}$, i-го віддаленого носія може бути позначений як $s_{i,j}$. Тоді інформація користувача, що зберігається на віддалених носіях може бути представлена у вигляді матриці S:

$$S=\begin{pmatrix} s_{1,1} & s_{1,2} & \dots & s_{1,n} \\ s_{2,1} & s_{2,2} & \dots & s_{2,n} \\ \dots & \dots & \dots & \dots \\ s_{m,1} & s_{m,2} & \dots & s_{m,n} \end{pmatrix}$$

В такій матриці кожен стовпець містить однойменні фрагменти всіх носіїв, а рядок – фрагменти одного носія.

Пропонується метод для відновлення інформації з будь якої непарної кількості віддалених носіїв до яких втрачено доступ. При використанні запропонованого методу розробник такої системи має обрати граничне значення кількості p віддалених носіїв, інформація з яких може бути відновлена. Особливістю методу є те, що кількість резервних носіїв, що використовуються для відновлення даних з p носіїв також становить p. Це значить, що метод забезпечує теоретичний мінімум інформаційної надлишковості резервування. Тоді кожен j-ий фрагмент, $j\in\{1,..,n\}$, i-го резервного носія, $i\in\{1,..,p\}$, може бути позначений як $d_{i,j}$, а інформація, що зберігається на резервних носіях може бути представлена у вигляді матриці D:

$$D=\begin{pmatrix} d_{1,1} & d_{1,2} & \dots & d_{1,n} \\ d_{2,1} & d_{2,2} & \dots & d_{2,n} \\ \dots & \dots & \dots & \dots \\ d_{p,1} & d_{p,2} & \dots & d_{p,n} \end{pmatrix}$$

Для відновлення даних використовуються p резервних носія, які пропонується розділити на три групи:

1) група, яка складається з носіїв з індексами від 1 до $(p-1)/2$ включно;

2) група, яка складається з єдиного носія з індексом $(p+1)/2$;

3) група, яка складається з носіїв з індексами від $(p+3)/2$ до p включно.

Дані для кожної групи резервних носіїв пропонується формувати наступним чином.

Фрагменти для першої групи резервних носіїв пропонується формувати з фрагментів, що належать висхідним діагоналям матриці під різними кутами, а формально як суму за модулем два наступних фрагментів:

$$\forall i\in\{1,..,(p-1)/2\},\forall j\in\{1,..,m\cdot i\}:$$
$$d_{(p-1)/2-i+1,j}=\bigoplus_{q=1}^{\lceil j/i\rceil} s_{\lceil j/i\rceil-q+1,(j-1)\,mod\,i+1+i(q-1)} \quad (1)$$

$$\forall i\in\{1,..,(p-1)/2\},\forall j\in\{m\cdot i+1,..,n\}:$$
$$d_{(p-1)/2-i+1,j}=\bigoplus_{q=1}^{m} s_{m-q+1,i\cdot q+j-i\cdot m} \quad (2)$$

Фрагменти резервного носія з другої групи пропонується формувати з фрагментів, що належать стовпцям матриці, а формально як суму за модулем два наступних фрагментів:

$$\forall j\in\{1,..,n\}:d_{(p+1)/2,j}=\bigoplus_{q=1}^{m} s_{q,j} \quad (3)$$

Фрагменти для третьої групи резервних носіїв пропонується формувати з фрагментів, що належать низхідним діагоналям матриці під різними кутами, а формально як суму за модулем два наступних фрагментів:

$$\forall i\in\{1,..,(p-1)/2\},\forall j\in\{1,..,m\cdot i\}:$$
$$d_{i+(p+1)/2,j}=\bigoplus_{q=1}^{\lceil j/i\rceil} s_{m-\lceil j/i\rceil+q,(j-1)\,mod\,i+1+i(q-1)} \quad (4)$$

$$\forall i\in\{1,..,(p-1)/2\},\forall j\in\{m\cdot i+1,..,n\}:$$
$$d_{i+(p+1)/2,j}=\bigoplus_{q=1}^{m} s_{q,j-i\cdot m+i\cdot q} \quad (5)$$

Запропонований порядок формування фрагментів резервних носіїв може бути ілюстровано прикладом, який зображено на рис. 1-5. В рамках цього прикладу вважається, що загальна кількість віддалених носіїв дорівнює семи (m=7), а граничне значення кількості носіїв до яких втрачено доступ – п'яти (p=5).

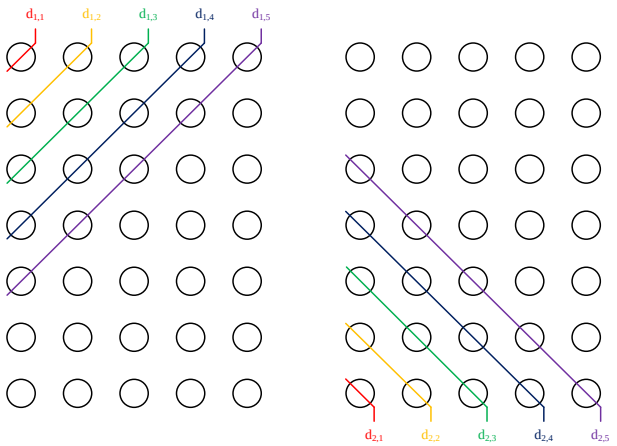


Рис.1 – Формування фрагментів першого резервного носія

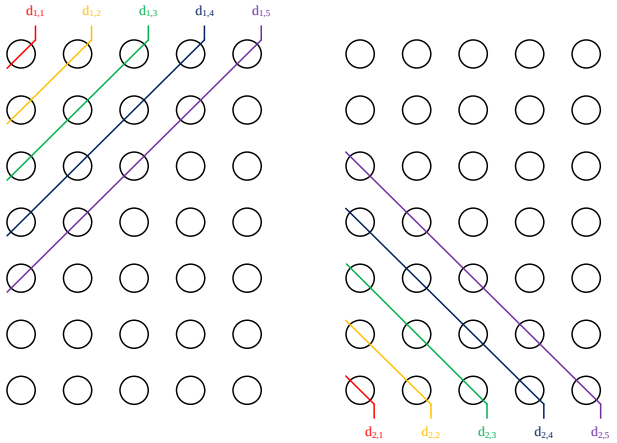


Рис. 2 – Формування фрагментів другого резервного носія

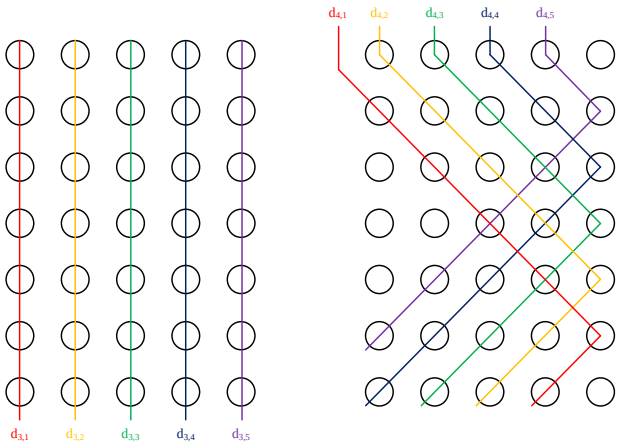


Рис. 3 – Формування фрагментів третього резервного носія

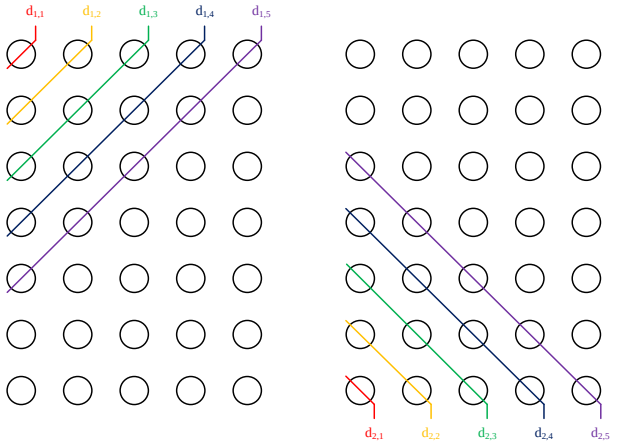


Рис. 4 – Формування фрагментів четвертого резервного носія

Запропонований метод передбачає відновлення майже всіх фрагментів носіїв, до яких втрачено доступ. В найкращому випадку метод надає можливість відновити всі фрагменти, в найгіршому випадку метод не дає змоги відновити останні $(m\text{-}p)/2$ однойменних фрагментів носіїв. Зважаючи на те, що кількість фрагментів n на порядки більша за m та p, на цей недолік можна не звертати уваги.

Запропонований метод передбачає впорядковане відновлення фрагментів носіїв.

Індекси носіїв, до яких втрачено доступ, в порядку зростання позначимо як $X= (x_1, x_2,...,x_p)$, де $x_1 < x_2 < ... < x_p$.

Процес відновлення фрагментів складається двох наступних етапів:

1) відновлення першого фрагменту з носія $x_{(p+1)/2}$, до якого втрачено доступ, і всіх потрібних для цього фрагментів з інших недоступних носіїв;

2) Відновлення по одному фрагменту з кожного носія, до якого втрачено доступ доки не будуть відновленні всі фрагменти.

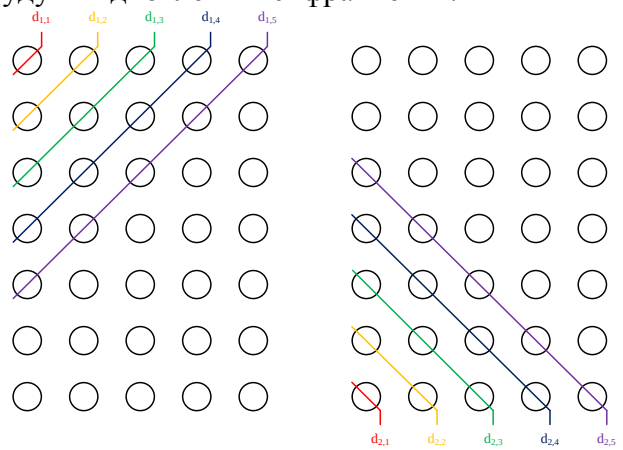


Рис. 5 – Формування фрагментів n'ятого резервного носія

Визначимо кількість фрагментів $B=\begin{pmatrix} b_1 & b_2 & \dots & b_p \end{pmatrix}$, яку потрібно відновити з кожного недоступного носія для реконструювання першого фрагменту носія $x_{(p+1)/2}$. Для цього визначимо різниці індексів сусідніх носіїв, до яких втрачено доступ. Позначимо різниці індексів сусідніх носіїв як $A=\begin{pmatrix} a_1 & a_2 & \dots & a_{p-1} \end{pmatrix}$, де

$$\forall i \in \{1,\dots,p-1\}: a_i = x_{i+1} - x_i \qquad (6)$$

Тоді маємо наступну рівність:
$\forall i \in \{1,\dots,p\}$:

$$b_i = \begin{cases} a_i \cdot ((p-1)/2 - i) + b_{i+1}, i < (p-1)/2 \\ 1, (p-1)/2 \le i \le (p+3)/2 \\ a_{i-1} \cdot (i - (p+3)/2) + b_{i-1}, i > (p+3)/2 \end{cases} \qquad (7)$$

Введемо поняття останнього відновленого фрагмента кожного недоступного носія $C=\begin{pmatrix} c_1 & c_2 & \dots & c_p \end{pmatrix}$. Цілком очевидним є факт того, що на початку відновлення:

$$\forall i \in \{1,\dots,p\}: c_i = 0 \qquad (8)$$

Алгоритм відновлення для першого етапу складається з наступної послідовності дій:

1. Встановити номер кроку $i = b_1$.
2. Встановити номер відновлюваного носія $j = 1$.
3. Якщо кількість фрагментів, яку потрібно відновити для *j*-го носія, менша за номер кроку, тобто умова $b_j \ge i$ не виконується перейти до кроку під номером 6.
4. Збільшити індекс останнього відновленого фрагменту *j*-го носія c_j на одиницю.
5. Відновити фрагмент s_{x_j, c_j}.
6. Збільшити j на 1.
7. Якщо умова $j < (p+1)/2$ виконується перейти до кроку під номером 3.
8. Зменшити i на одиницю.
9. Якщо умова $i \ge 1$ виконується перейти до кроку 2.
10. Встановити номер кроку $i = b_p$.
11. Встановити номер відновлюваного носія $j = p$.
12. Якщо кількість фрагментів, яку потрібно відновити для *j*-го носія, менша за номер кроку, тобто умова $b_j \ge i$ не виконується перейти до кроку під номером 15.
13. Збільшити індекс останнього відновленого фрагменту *j*-го носія c_j на одиницю.
14. Відновити фрагмент s_{x_j, c_j}.
15. Зменшити j на 1.
16. Якщо умова $j > (p+1)/2$ виконується перейти до кроку під номером 12.
17. Зменшити i на одиницю.
18. Якщо умова $i \ge 1$ виконується перейти до кроку 11.
19. Збільшити індекс останнього відновленого фрагменту (*p*+1)/2-го носія $c_{(p+1)/2}$ на одиницю.
20. Відновити фрагмент $s_{x_{(p+1)/2}, c_{(p+1)/2}}$.

Другий етап відновлення складається з *p* кроків, по одному фрагменту з кожного носія, до яких втрачено доступ. На всіх не парних кроках, окрім останнього, реконструюється фрагмент носія з найменшим порядковим номером за допомогою резервних носіїв першої групи. На всіх парних кроках реконструюється фрагмент носія з найбільшим порядковим номером за допомогою резервних носіїв третьої групи. На останньому кроці реконструюється останній фрагмент за допомогою резервного носія другої групи.

Фрагменти, які відновляються на непарних кроках, окрім останнього, належать висхідним діагоналям, сума яких зберігається в резервних носіях першої групи. Такий фрагмент може бути реконструйований, так як фрагменти, що належать діагоналі і лежать на ній вище мають порядковий номер носія менший за j, тобто вже відновленні на даному кроці або зберігаються на непошкоджених носіях з меншими порядковими номерами. Фрагменти, що лежать на діагоналі нижче мають номери менші за c_j, тобто відновлені на одному з минулих кроків або належать доступним носіям.

Фрагменти, які відновляються на парних кроках, належать низхідним діагоналям, сума яких зберігається в резервних носіях третьої групи. Такий фрагмент може бути реконструйований, оскільки фрагменти, що належать діагоналі і лежать на ній нижче мають порядковий номер носія більший за j, тобто вже відновленні на даному кроці або зберігаються на непошкоджених носіях. Фрагменти, що лежать на діагоналі вище мають номери менші за c_j, тобто відновлені на одному з минулих кроків або належать доступним носіям.

Фрагмент, який відновлюється на останньому кроці належить вертикальній прямій, сума якої зберігається в резервному носієві другої

групи. Фрагмент може бути реконструйований так, як всі інші однойменні c_j-ті фрагменти вже відновлені на даному або минулих кроках, а інші фрагменти відомі.

Алгоритм відновлення для другого етапу:

1. Встановити номер кроку $i = c_1$.
2. Встановити номер відновлюваного носія $j = 1$.
3. Збільшити індекс останнього відновленого фрагменту j-го носія c_j на одиницю.
4. Відновити фрагмент s_{x_j,c_j}.
5. Збільшити j на 1.
6. Якщо умова $j < (p+1)/2$ виконується перейти до кроку під номером 3.
7. Встановити номер відновлюваного носія $j = p$.
8. Збільшити індекс останнього відновленого фрагменту j-го носія c_j на одиницю.
9. Відновити фрагмент s_{x_j,c_j}.
10. Зменшити j на 1.
11. Якщо умова $j > (p+1)/2$ виконується перейти до кроку під номером 8.
12. Збільшити індекс останнього відновленого фрагменту $(p+1)/2$-го носія $c_{(p+1)/2}$ на одиницю.
13. Відновити фрагмент $s_{x_{(p+1)/2},c_{(p+1)/2}}$.

Для відновлення конкретного фрагменту s_{x_j,c_j} віддаленого носія, до якого втрачено доступ, будемо посилатись на групи описані на етапі резервування. Тоді для відновлення фрагменту з індексом резервного носія j, що належить носіям з першої групи достатньо використати одну з наступних формул:

$$\forall j \in \{1,..,(p-1)/2\}, \forall c_j \in \{1,..,m\cdot i\}:$$
$$i = (p-1)/2 - j + 1$$
$$s_{x_j,c_j} = d_{j,c_j} \oplus \bigoplus_{\substack{q=1 \\ q \neq \lceil c_j/i \rceil - x_j + 1}}^{\lceil c_j/i \rceil} s_{\lceil c_j/i \rceil - q + 1, (c_j-1) mod i + 1 + i(q-1)} \quad (9)$$

$$\forall j \in \{1,..,(p-1)/2\}, \forall c_j \in \{m\cdot i+1,..,n\}:$$
$$i = (p-1)/2 - j + 1$$
$$s_{x_j,c_j} = d_{j,c_j} \oplus \bigoplus_{\substack{q=1 \\ q \neq m - x_j + 1}}^{m} s_{m-q+1, i\cdot q + c_j - i\cdot m} \quad (10)$$

Для відновлення фрагменту з індексом резервного носія j, що належить носію з другої групи

$$j = (p+1)/2, \forall c_j \in \{1,..,n\}:$$
$$s_{x_j,c_j} = d_{(p+1)/2,c_j} \oplus \bigoplus_{\substack{q=1 \\ q \neq x_j}}^{m} s_{q,c_j} \quad (11)$$

Для відновлення фрагменту з індексом резервного носія j, що належить носіям з третьої групи достатньо використати одну з наступних формул:

$$\forall j \in \{(p+3)/2,..,p\}, \forall c_j \in \{1,..,m\cdot i\}:$$
$$i = j - (p+1)/2$$
$$s_{x_j,c_j} = d_{j,c_j} \oplus \bigoplus_{\substack{q=1 \\ q \neq x_j - m + \lceil c_j/i \rceil}}^{\lceil c_j/i \rceil} s_{m - \lceil c_j/i \rceil + q, (c_j-1) mod i + 1 + i(q-1)} \quad (12)$$

$$\forall j \in \{(p+3)/2,..,p\}, \forall c_j \in \{m\cdot i+1,..,n\}:$$
$$i = j - (p+1)/2$$
$$s_{x_j,c_j} = d_{j,c_j} \oplus \bigoplus_{\substack{q=1 \\ q \neq x_j}}^{m} s_{q, c_j - i\cdot m + i\cdot q} \quad (13)$$

Зважаючи на алгоритм відновлення можна зробити висновок, що запропонований метод відновлює не всі фрагменти носіїв, до яких втрачено доступ. В середньому кількість не відновлювальних фрагментів дорівнює $max\{c_1, c_p\}$ після першого етапу відновлення.

Оцінка ефективності

Зважаючи на те, що кожен віддалений носій складається більш як з тисячі фрагментів, при підрахунку обчислювальної складності процедури резервування даних можна знехтувати формулами (1) та (4) при не великих значеннях p, та вважати, що всі фрагменти резервних носіїв першої групи формуються за формулою (2), а третьої – за формулою (5). Тоді очевидно, що обчислювальна складність резервування даних складає $O(p \cdot m \cdot n)$. Звертаючи увагу на той самий факт, при підрахунку обчислювальної складності відновлення даних можна знехтувати алгоритмом відновлення першого етапу. Та вважати, що всі n однойменних фрагментів відновлюються за алгоритмом другого етапу. Тоді очевидним є факт того, що обчислювальна складність відновлення даних складає $O(p \cdot m \cdot n)$.

Таким чином, показано, що обчислювальна складність процесів формування резервного коду та відновлення будь-яких p з m носіїв, до яких втрачено доступ, лінійно залежно від ос-

новних параметрів – кількості носіїв, кількості фрагментів та кількості носіїв, дані з яких відновлюються.

Основною перевагою запропонованого методу є те, що він не накладає обмежень на кількість носіїв, дані з яких можуть бути відновлені на відміну від відомих методів [4-7].

Висновки

В результаті проведених досліджень запропоновано метод резервування та відновлення даних при їх зберіганні на віддалених від користувача носіях на основі діагональних контрольних сум. Основною відмінністю запропонованого методу від відомих є те, що він не накладає обмежень на кількість носіїв, дані з яких можуть бути відновлені.

Для досягнення можливості відновлення даних з довільної кількості носіїв, доступ до яких втрачено, запропоновано використання контрольних сум фрагментів, що належать діагоналям, розташованими під різними кутами. Оскільки для відновлення даних з довільних p носіїв використано таку ж кількість резервних носіїв, то метод забезпечує теоретичний мінімум інформаційної надлишковості резервування даних.

Розроблений метод орієнтовано для використання в системах з великою кількістю користувачів та віддалених від них рознесених вузлах зберігання інформації.

Список посилань

1. Patterson D. Case of Redundant Array of Inexpensive Disk (RAID) / Patterson D., Gibson G., Katz R.- Berkeley: University of California. – 1987. – 238 p.
2. Plank J. S., Thomasson M.G., On Practical Use of LDPC Erasure Codes for Distribute Storage Application: Technical Report UT-CS-03-510. – Department of Computer Science, University of Tennessee. – 2004.
3. Berger C.R. Optimizing Joint Erasure – and Error-Correction Coding for Wireless Packet Transmissions // Willett P., Pattirati K. / IEEE Transactions on wireless communications. – Vol.7 – No. 11. – 2008. – Pp. 4586-4595.
4. Иванов Д. Г. Организация резервирования в системах распределенного хранения данных // Вісник Національного технічного університету України ”КПІ” Інформатика, управління та обчислювальна техніка, – Київ: ВЕК+ – 2012 – № 56. – С.160-164.
5. Иванов Д.Г. Метод восстановления данных на основе скошенных матриц в системах распределенного хранения / Иванов Д.Г. Луцкий Г.М. // Известия высших учебных заведений. Проблемы полиграфии и издательского дела. Информационные технологии. – М.:УПИПК МГУП им. И.Федорова. – 2013. – № 2. – С.47-52
6. Марковський О.П. Організація резервування та відновлення даних при їх віддаленому зберіганні / Марковський О.П., Іванов Д.Г., Ванчугов Б.Ю. // Вісник Національного технічного університету України ”КПІ” Інформатика, управління та обчислювальна техніка, – Київ: ВЕК+ – 2013. – № 59. – С. 50-55.
7. Марковський О.П. Метод резервування та прискореного відновлення даних в системах їх віддаленого зберігання / Марковський О.П., Іванов Д. Г., Великий М. М., Невдащенко М. В. // Вісник Національного технічного університету України ”КПІ” Інформатика, управління та обчислювальна техніка, – Київ: ВЕК+ – 2014 – № 60. – С.46-54.

УДК 519.854.2

ПАВЛОВ А.А.
МИСЮРА Е.Б.
МЕЛЬНИКОВ О.В.
МУХА И.П.
ЛИСЕЦКИЙ Т.Н.

АЛГОРИТМИЗАЦИЯ ТРЕТЬЕГО И ЧЕТВЕРТОГО УРОВНЕЙ ЧЕТЫРЕХУРОВНЕВОЙ МОДЕЛИ КАЛЕНДАРНОГО И ОПЕРАТИВНОГО ПЛАНИРОВАНИЯ И ПРИНЯТИЯ РЕШЕНИЙ В СЕТЕВЫХ СИСТЕМАХ С ОГРАНИЧЕННЫМИ РЕСУРСАМИ

В статье приводится алгоритмическое обеспечение решения задачи планирования на третьем и четвертом уровнях четырехуровневой модели календарного и оперативного планирования и принятия решений в системах с сетевым представлением технологических процессов и ограниченными ресурсами. Приведен пример решения задачи на сети общего вида.

In the article the algorithmware for the planning problem solution on the third and fourth levels of the four-level scheduling, operational planning and decision making model in systems with a network representation of the technological processes and limited resources is given. An example of the problem solution on the common view network is shown.

Введение

В статье [1] представлено алгоритмическое обеспечение первого и второго уровня четырехуровневой модели планирования, принятия решений и оперативного планирования в сложных организационно - технологических системах с ограниченными ресурсами. На первом уровне строится агрегированная модель планирования выполнения изделий, в которой все производство агрегируется в один прибор, на втором уровне модели – согласованные планы выполнения агрегированных работ в соответствии с одним из 31 критериев оптимальности. На третьем уровне модели строится и решается многоэтапная сетевая задача календарного планирования (выполняется точное планирование). Четвертый уровень предназначен для оперативного планирования – оперативной коррекции построенного на третьем уровне плана, в случае его частичного невыполнения.

Алгоритмическое обеспечение модели состоит из пяти блоков. Действия лиц, принимающих решение, и экспертов определяются в блоке принятия решений (блок 1), а формальные модели реализуются в блоках 2–5:

Блок 1. Блок принятия решений.
Блок 2. Построение модели первого уровня.
Блок 3. Построение модели второго уровня.
Блок 4. Построение модели третьего уровня.
Блок 5. Оперативное планирование.

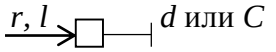
В [1] приведены формальные модели предварительного (агрегированного) и согласованного планирования. Показан пример решения задачи. Также в общих словах описаны модели точного планирования и оперативного корректирования (третий и четвертый уровни модели).

В данной статье третий и четвертый уровни модели планирования получают формализованное описание, приведены несколько различных методов решения задачи на обоих уровнях, а также пример решения задачи.

Постановка задачи

Имеется сетевая модель, отражающая реальную технологию изготовления изделий (групп изделий), состоящая из пяти базовых элементов, и элементов типа накопитель:

Элемент типа 1. Обозначение:

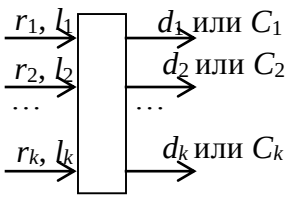


где l – длительность выполнения работы, r – момент времени готовности работы к выполнению на приборе, C – момент окончания выполнения работы, d – директивный срок работы. В допустимом расписании $C \le d$. Символом ⊣ на сети обозначается окончание выполнения конечной работы (не имеющей преемников).

Стрелка, *входящая* в кружок ○, обозначает выполненную работу. Стрелка, *выходящая* из кружка – новая невыполненная работа. □ – обозначение прибора (ресурса), на котором выполняется работа. Приборы и работы обозначаются уникальным номером, причем цифра в начале номера работы

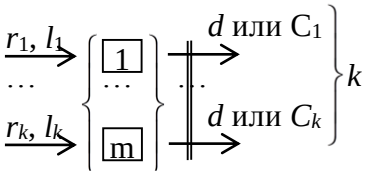
означает уровень входимости (на первом уровне входимости данной работе не предшествует ни одна другая работа). Значком ◌ обозначены фиктивные работы, соответствующие окончанию выполнения изделия. Под изделием понимается либо физически одно изделие, либо партия физически однотипных изделий.

Элемент типа 2. Обозначение:



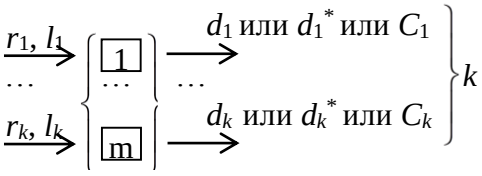
Прибор последовательно без прерываний выполняет k работ. r_i – моменты готовности выполнения работ на приборе, d_i – директивный срок выполнения i-й работы на приборе. В допустимом расписании $C_i \le d_i$.

Элемент типа 3. Обозначение:



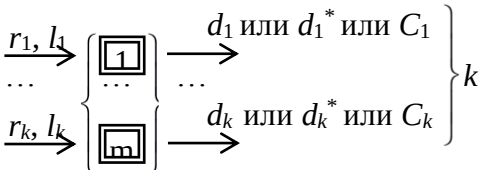
m независимых (идентичных) параллельных приборов без прерываний выполняют k работ, которые должны быть выполнены к одному директивному сроку d. В допустимом расписании $C_i \le d$, $i = \overline{1,k}$. Моменты запуска приборов должны быть одинаковы. Работы, следующие за элементом типа 3, могут начинаться только после окончания всех работ на данном элементе.

Элемент типа 4. Обозначение:



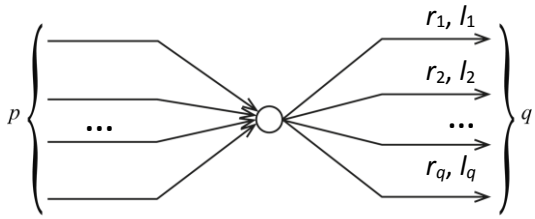
m независимых (идентичных) параллельных приборов *равной* производительности, работающих без прерываний, выполняют k работ, которые должны быть выполнены к различным директивным срокам d_i. $d_i^* = [d_i - \Delta_i, d_i]$, $i = \overline{1,k}$, определяет диапазон получения допустимого решения: $C_i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

Элемент типа 5. Обозначение:



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

Назовем накопителем элемент такого вида:



Элемент типа накопитель отражает появление нового множества, состоящего из q невыполненных работ ($q \ge 1$) в результате окончания выполнения предыдущего множества из p работ ($p \ge 1$). Накопитель описывает отношения предшествования между работами, но не является каким-либо объединением работ. Накопитель может, в частности, описывать сборку одной работы из нескольких других и разузловку одной работы на несколько других. Элемент типа накопитель всегда должен стоять после элемента типа 3, в соответствии с условиями его работы.

Для элементов типа накопитель выполняется следующее общее правило: директивный срок работ слева от накопителя равен минимальному из моментов запуска работ справа от накопителя, а наиболее ранний момент запуска работ справа от накопителя равен максимальному из моментов окончания работ слева от накопителя.

Номера работ, соответствующие индексам i их параметров l_i, d_i, r_i и т.д., для каждого элемента свои и определяются технологией выполнения работ.

Полное описание сетевой модели и критериев оптимальности планирования (пяти базовых критериев и 26 синтетических критериев, являющихся всеми возможными линейными свертками базовых) см. в [1].

При планировании по одному из *базовых критериев* оптимальности на исходную сеть накладывается ограничение: внутри сети могут находиться несколько виртуальных приборов,

соответствующих одному и тому же реальному прибору только в том случае, если они разнесены по времени технологией (чтобы исключить возможность выполнения разных работ в один и тот же интервал времени на одном и том же реальном приборе). Для этого, например, можно использовать накопитель между элементами, соответствующими одному и тому же реальному прибору, для разделения множеств работ, выполняемых на них.

При соблюдении этого ограничения есть для многоэтапной сетевой задачи календарного планирования изложить эффективные методы решения и корректировки, основанные на получении решения непосредственно от конечных работ к начальным, с использованием математически строгих решений одноэтапных задач календарного планирования, в то время как для произвольной виртуальной сети задача решается с помощью эвристических методов, с меньшей гарантией оптимальности свойств решения.

При планировании по *синтетическому критерию* оптимальности изделия разбиваются на $N \in [1, 5]$ отдельных групп по числу базовых составляющих синтетического критерия, ни одно изделие не может входить в две разные группы. Топология, связанная с технологией, задается в виде N отдельных подсетей, соответствующих N отдельным группам изделий, каждой подсети соответствует свой базовый критерий. При этом подсети не могут иметь отношений предшествования друг с другом, но могут быть связаны общим оборудованием, что не позволяет планировать выполнение работ в подсетях независимо друг от друга. Кроме того, общее оборудование (виртуальные приборы, соответствующие одному и тому же реальному прибору) может быть и внутри каждой подсети. Приведенная ниже методология планирования для синтетического критерия позволяет осуществить разведение работ, выполняемых на общем оборудовании.

Алгоритмическое обеспечение третьего уровня планирования (блок 4)

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

После утверждения в блоке 3 согласованного плана выполнения агрегированных работ производится дезагрегация агрегированных работ, т.е. возврат к исходной модели сети, с учетом изменений, внесенных при согласованном планировании. При разбивке общих вершин соответствующие агрегированные работы (в агрегированной сети) и соответствующие им исходные работы (в исходной сети), вместе с их отношениями предшествования, разносятся на элементы с тем же номером для выполнения в разное время (то есть, условие непрерывности выполнения работ теперь действует для отдельных групп работ, а не для одной группы, которую они представляли ранее, выполняясь на одном элементе). Если в принятом на реализацию на третьем уровне плана согласованного планирования некоторые элементы типа 2–5 оказались разбитыми на части, то в исходной технологической сети каждый такой элемент заменяется аналогичными независимыми элементами того же типа, с соответствующим перераспределением входных и выходных связей между новыми элементами, причем каждая выходящая связь входит в ту же вершину, что и в исходной технологической модели.

При планировании *по базовому критерию* необходимо соблюсти ограничение об отсутствии возможности наложения интервалов выполнения работ в сети, поэтому множества работ, выполняемых на разбиваемых приборах, следует принудительно разнести во времени с помощью вставкинакопителя между ними. Накопитель можно использовать и для планирования по синтетическому критерию (в сети произвольного вида), в таком случае гарантируется разнесение разбиваемых множеств работ во времени, чтобы в последующем набор общих вершин и соответствующая ему сетевая модель уже не могли измениться. Примеры такого разбиения при планировании *по базовому критерию* см. ниже на рис. 1 (разносятся работы на элементе №23), при планировании *по синтетическому критерию* – на рис. 2 (разносятся работы на элементе №23) и рис. 3 (разносятся работы на элементе №12).

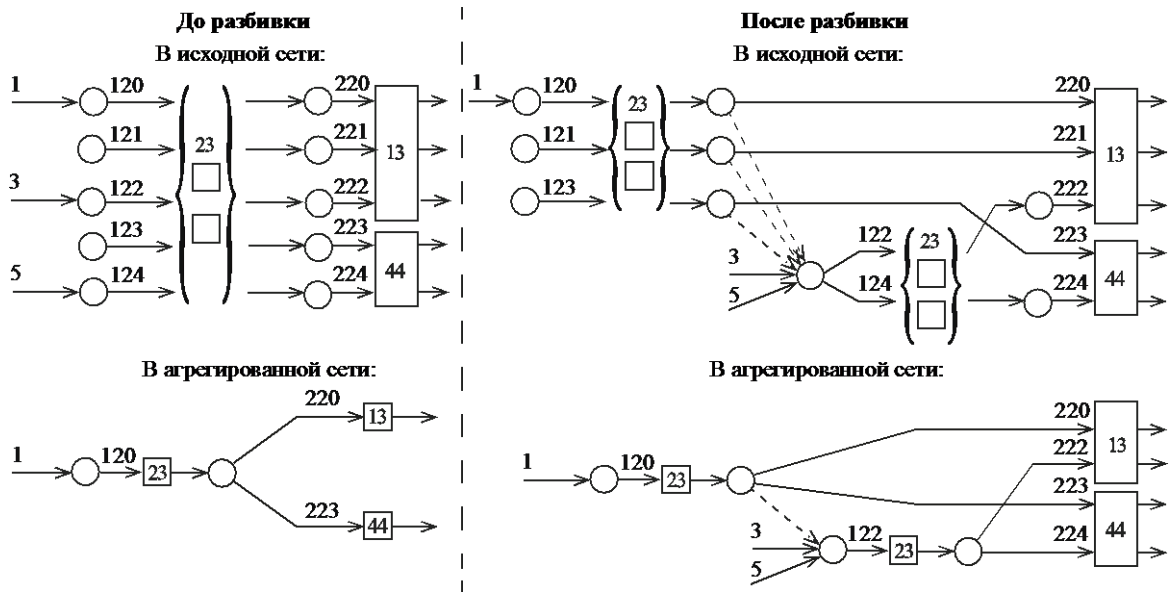


Рис. 1 – Изменение модели по результатам распределения:базовый или синтетический критерий

Пример для базового и синтетического критерия. Если работы 120–125 (агрегированная работа 125) выполнялись на одном элементе 23 в исходной и агрегированной сети, но при согласованном планировании выполняются непрерывно только две группы работ – сначала 120, 121 и 123, а позже 122 и 124, то эта часть агрегированной сети и, соответственно, исходной сети, изменяется так, как показано на рис. 1. Работы 1, 3, 5 на рисунке приведены для примера как непосредственные предшественники работ 120, 122, 124. При объединении новых общих вершин используются общие правила, в том числе ограничения, установленные для выходных работ накопителей. Штрихпунктирными линиями на рис. 1 отражены фиктивные работы с нулевой длительностью, отражающие лишь отношения предшествования между накопителями: работы 122 и 124 на втором приборе №23 могут начаться лишь после окончания работ 120, 121 и 123 на первом приборе №23. Таким образом, эти множества работ разнесены во времени.

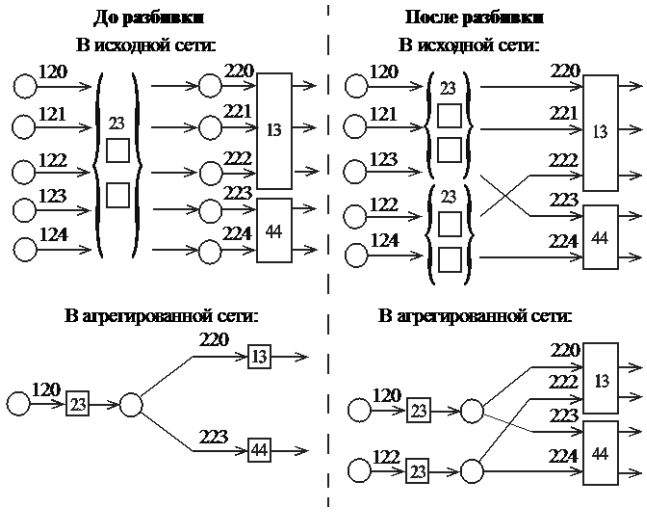


Рис. 2. Изменение модели по результатам распределения: синтетический критерий

Примеры для синтетического критерия:

1) сетевая модель для описанного выше примера (рис. 1) может быть упрощена, если не требуется принудительное разделение разбиваемых множеств работ, так, как показано на рис. 2.

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

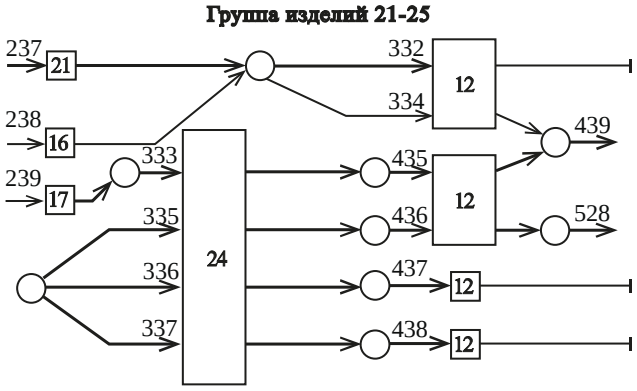


Рис. 3. Изменение агрегированной сети по результатам распределения: синтетический критерий, разбивка работ на элементе №12

Если для такого случая следует применить накопители для разделения работ, то показанная на рис. 1 методика используется 3 раза.

На третьем уровне планирования строится многоэтапная сетевая задача календарного пла-

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

Метод решения многоэтапной сетевой задачи календарного планирования, соответствующей базовому критерию оптимальности

Предлагаемый метод решения задачи реализуется для случая, когда для всех конечных работ сети задаются директивные сроки, выполнение конечной работы до ее директивного срока является допустимым.

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

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

Определение 2. Расписание является *оптимальным*, если оно отвечает следующим условиям:

1) расписание допустимое;

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

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

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

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

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

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

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

Примечание. Пусть конечная работа должна быть выполнена точно в срок. Излагаемый метод, основанный на ПДС-алгоритмах [2 (гл. 3), 5, 6, 7], предлагает минимально возможное отклонение допустимого расписания от директивных сроков. Таким образом, достаточно обоснованно будем считать, что в этом случае штрафом за опережение (выполнение работы ранее директивного срока) можно пренебречь.

1. В силу свойств технологической сети, соответствующей модели планирования по одному из базовых критериев, однозначно директивные сроки конечных работ, где необходимо, пересчитываются в директивные сроки выполнения работ элементов типа 2–5 последнего (в общем случае, предпоследнего или того, с которым непосредственно связаны конечные работы) ряда сети.

1.1. Для каждого элемента типа 2 последнего ряда сети решается одна из следующих одноэтапных задач календарного планирования:

1.1.1. Минимизация суммарного опережения заданий для допустимого расписания на одном приборе при максимально позднем моменте начала выполнения заданий и произвольных директивных сроках;

1.1.2. Минимизация суммарного опережения заданий с произвольными директивными сроками для допустимого расписания на одном приборе.

Примечание. Задачу 1.1.2 лучше использовать для случая, когда штраф за суммарное опережение

выполнения заданий относительно их директивных сроков является существенным.

1.2. Для каждого элемента типа 3 последнего ряда сети решается следующая задача календарного планирования: минимизация суммарного опережения заданий для допустимого расписания на идентичных параллельных приборах с общим директивным сроком.

Примечание. Полученному расписанию соответствует максимально поздний одновременный момент запуска приборов.

1.3. Для каждого элемента типа 4 последнего ряда сети решается одна из следующих двух задач календарного планирования:

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

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

где r_j – момент запуска прибора j, j_1 – номер прибора, у которого момент запуска в оптимальном расписании самый ранний (он является самым поздним для всех допустимых расписаний); j_l, $l = \overline{2,m}$ – номер прибора, у которого момент запуска следующий по величине после приборов j_k, $k = \overline{1,l-1}$ (он является самым поздним для всех допустимых расписаний с фиксированным r_{j_k}, $k = \overline{1,l-1}$).

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

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

где r_j – момент запуска прибора j, j_1 – номер прибора, у которого момент запуска в оптимальном расписании самый ранний (он является самым поздним для всех допустимых расписаний).

1.4. Для каждого элемента типа 5 последнего ряда сети решается одна из следующих двух задач календарного планирования:

1.4.1. Имеется m независимых параллельных приборов разной производительности, работающих без прерываний, которые выполняют n работ. l_i^j – длительность выполнения i-й работы на j-м приборе, длительности работ не связаны между собой (возможна ситуация, что одна работа выполняется быстрее другой на одном приборе и медленнее другой на другом приборе). Работы должны быть выполнены к директивным срокам d_i. Моменты запуска приборов произвольны. Необходимо построить допустимое расписание, оптимальное по критерию (1).

1.4.2. Имеется m независимых параллельных приборов разной производительности, работающих без прерываний, которые выполняют n работ. l_i^j – длительность выполнения i-й работы на j-м приборе, длительности работ не связаны между собой. Работы должны быть выполнены к директивным срокам d_i. Моменты запуска приборов произвольны. Необходимо построить допустимое расписание, оптимальное по критерию (2).

Примечание. Выбор задач 1.3.1 или 1.3.2, 1.4.1 или 1.4.2 определяется выбранным критерием оптимальности для многоэтапной сетевой задачи календарного планирования.

2. Полученные расписания для элементов типа 2–5 последнего ряда сети однозначно задают наиболее поздние моменты готовности на выполнение работ на входе элементов типа 2–5. Эти моменты времени в силу свойств технологической сети однозначно задают директивные сроки выполнения работ для элементов типа 2–5 предпоследнего ряда сети либо элементов типа 2–5, непосредственно (с точностью до элементов типа 1) предшествующих элементам типа 2–5 последнего ряда сети. Та же ситуация касается элементов типа 2–5 любых рядов сети. Для элементов типа 2–5 предпоследнего ряда сети реализуется п.1 общего метода.

Примечание. Пункты 1 и 2 реализуются до тех пор, пока не будут получены расписания элементов типа 2–5 первого ряда сети.

3. Расписания элементов типа 2–5 первого ряда сети в силу ее свойств однозначно задают расписание всех работ начальной части сети. Субоптимальное расписание построено.

Примечание 1. Изложенный метод обладает свойствами ПДС-алгоритма для элементов типа 2–3 конечной части сети (минимизация суммарного опережения выполнения конечных работ

относительно директивных сроков, которые определяются этими элементами).

Примечание 2. Для остальных составляющих критерия оптимальности многоэтапной сетевой задачи календарного планирования метод является эвристическим, в котором оптимизация по лексикографическому либо скалярному критерию (максимизация момента времени начала выполнения технологического процесса) реализуется последовательно с конца сети к началу ПДС-алгоритмами локально оптимально для каждого элемента типа 2–5.

Пример. Пусть работы 11–15 выполнялись на одном элементе 1 в исходной сети, но при согласованном планировании разбились на две группы работ – сначала 11, 12 и 14, а позже 13 и 15, в соответствии с этим в исходной сети элемент 1 разбивается на два элемента – 1А и 1Б (рис. 4).

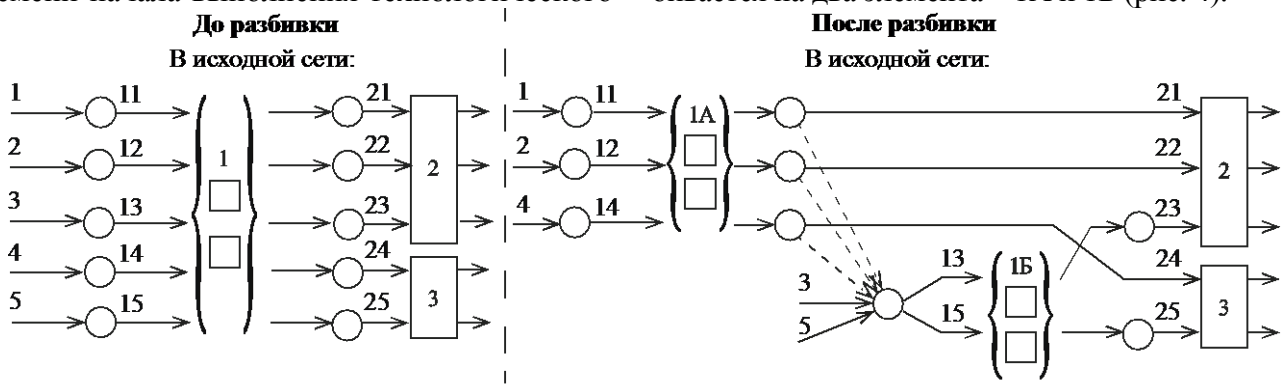


Рис. 4. Пример решения задачи третьего уровня на части сети с разбивкой элемента

Штрихпунктирными линиями на рис. 4 отражены фиктивные работы с нулевой длительностью, отражающие лишь отношения предшествования между накопителями: работы 13 и 15 на элементе 1Б могут начаться лишь после окончания работ 11, 12 и 14 на элементе 1А.

При решении задачи третьего уровня, имея моменты готовности работ на элементах 2 и 3 (C_{21} … C_{25}), находим оптимальное расписание выполнения этих работ на элементах 2 и 3. В результате получаем моменты начала выполнения работ 21–25 (r_{21} … r_{25}).

Переходим к предыдущему ряду: сначала решаем одноэтапную задачу для элемента 1Б, затем для элемента 1А, т.к. у элемента 1А директивные сроки работ будут не больше, чем моменты начала выполнения работ на элементе 1Б.

Моменты начала выполнения работ 23 и 25 автоматически становятся директивными сроками работ 13 и 15. Решаем одноэтапную задачу на элементе 1Б, получаем моменты начала выполнения работ 13 и 15. Выбираем минимальный из них. Пусть это будет момент $r_{1Б} = \min(r_{13}, r_{15})$. $r_{1Б}$ – это момент запуска элемента 1Б (элемент запускается в момент готовности первых работ). Очевидно, что $r_{1Б}$ является директивным сроком для работ 3 и 5, предшествующих работам 13 и 15.

Далее, сравниваем полученный момент $r_{1Б}$ с моментами начала выполнения работ 21, 22 и 24. Директивные сроки для работ 11, 12, 14 на элементе 1А, в соответствии с заданными отношениями предшествования и правилом работы накопителей, определяются следующими выражениями: $d_{11} = \min(r_{21}, r_{1Б})$; $d_{12} = \min(r_{22}, r_{1Б})$; $d_{14} = \min(r_{24}, r_{1Б})$.

Имея директивные сроки работ 11, 12 и 14, решаем одноэтапную задачу на элементе 1А, получаем моменты начала выполнения работ 11, 12 и 14, которые в свою очередь, являются директивными сроками для предшествующих им работ 1, 2 и 4.

Методы решения многоэтапной сетевой задачи календарного планирования, соответствующей синтетическому критерию оптимальности

При планировании по *синтетическому критерию* оптимальности, в силу того, что исходная сеть представляет собой совокупность независимых подсетей, и в одной или разных подсетях может использоваться общее оборудование, метод, представленный для базового критерия, не может быть использован без изменений, т.к. расписание, полученное в одной подсети, может не позволить получить расписание в другой подсети, если оборудование будет уже занято расписанием выполнения предыдущей группы изделий. Поэтому предлагаются четыре альтернативных метода решения сетевой многоэтапной задачи календарного планирования и построения пооперационного плана. *Первый метод* более быстродействующий и может использоваться в случае, если решение одноэтапной задачи на элементах не

приводит к значительному уменьшению моментов начала выполнения работ на этих элементах, полученных при согласованном планировании. Тогда критерием оптимальности для планирования по синтетическому критерию является получение пооперационного плана, максимально близкого к плану, полученному на этапе согласованного планирования. Данный метод может использоваться и для планирования по базовому критерию оптимальности, а также в сети произвольного вида.

В первом альтернативном методе директивные сроки работ на элементах исходной сети задаются моментами окончания агрегированных работ, в которые эти работы вошли в результате агрегации, в утвержденном согласованном плане второго уровня. Моменты запуска агрегированных работ в согласованном плане определяют минимальные моменты запуска работ на приборах. Благодаря согласованному планированию работы, выполняемые на элементах с одинаковым номером, разнесены по времени. Многоэтапная сетевая задача календарного планирования решается с конца в начало, начиная с элементов последнего уровня входимости, в произвольном порядке рассмотрения элементов сети.

Для построения пооперационного плана выполняем дезагрегацию агрегированных работ: к согласованному плану выполнения агрегированных работ добавляем работы исходной сети, прежде вошедшие в агрегированные работы, помечая их как неназначенные. Длительности всех работ приводим в соответствие исходной сети. Для каждого из элементов типа 2–5 решаются одноэтапные задачи календарного планирования следующего вида: найти допустимое (не нарушающее директивные сроки) расписание выполнения работ, оптимальное по следующему критерию оптимальности: для элементов типа 2, 3: max r, где r – момент запуска прибора (приборов); для элементов типа 4, 5 – минимизирующее лексикографический критерий оптимальности [7]. При таком решении для элементов типа 3–5 косвенно минимизируется сдвиг предшественников, если он необходим.

Пример решения одноэтапных задач календарного планирования на элементах типа 3–5 показан на рис. 7 ниже. Выполнение работ на элементе типа 2 представляет собой цепочку непрерывно выполняемых работ на одном физическом приборе.

В результате решения одноэтапной задачи получаем моменты начала выполнения работ на элементе сети. Устанавливаем новый директивный срок для каждой предшествующей работы: он равен минимуму из момента окончания выполнения предшествующей работы и момента начала ее преемника. Переходим к предыдущему ряду и продолжаем аналогичное решение задач на элементах. Если директивный срок работ на элементе-предшественнике уменьшился, в результате производится сдвиг интервала выполнения работ на нем в сторону уменьшения максимального момента окончания выполнения работ. Для уменьшения сдвига, если он необходим, можно устанавливать директивные сроки работ без учета их моментов окончания в согласованном плане, как минимум из моментов начала выполнения их преемников по сети.

Второй метод более трудоемкий, но универсальный, он также может использоваться для планирования по базовому критерию оптимальности и в сети произвольного вида. Согласно второму методу, на исходной сети выполняется процедура согласованного планирования с помощью алгоритма распределения работ с конца в начало, начиная с утвержденных директивных сроков изделий и в соответствии с отношениями предшествования исходной сети, для одного порядка выполнения групп изделий, соответствующего утвержденному согласованному плану, со следующими особенностями реализации:

1) выполняется назначение исходных (не агрегированных) работ, на ресурсы элементов (физических приборов);

2) назначение работ на элементах типа 2–5 выполняется с помощью решения одноэтапных задач оптимизации (см. выше), когда становятся известными директивные сроки всех работ на элементе после назначения их преемников;

3) после решения одноэтапной задачи при занятости ресурса директивный срок назначаемых работ уменьшается для выполнения их в подходящий интервал времени;

4) проверки общих вершин и совместимости технологии не производятся.

Описание алгоритма распределения по второму методу (алгоритм применяется для каждой группы изделий в порядке назначения групп, соответствующем выбранному на втором уровне согласованному плану).

1. Инициализация и очистка списков. Создать пустой список приборов на всех элементах, без назначенных работ.

2. Выбираем все конечные работы изделий. Заносим их в список неназначенных работ. Директивный срок (наиболее пóзднее время окончания) для конечных работ изделий определя-

ется моментами их окончания в выбранном согласованном плане.

3. Если список неназначенных работ пуст, переход на п.7, иначе на п.4.

4. Выбрать из списка неназначенных работ очередную работу. Помечаем работу как готовую к назначению (для нее определен директивный срок), переход на п. 5.

5. Пусть все неназначенные работы на элементе сети уже готовы к назначению. Обозначим это множество работ через *I*. Решаем на данном элементе одноэтапную задачу оптимизации, в результате которой определяются моменты начала и окончания выполнения работ множества *I*. Если после решения одноэтапной задачи имеются работы, конфликтующие по времени с работами множества *I*, то моменты начала и окончания выполнения работ множества *I* (каждой работы по отдельности, кроме элемента 2, на котором нужно сдвигать все множество работ *I*) уменьшаются таким образом, чтобы избежать конфликта по времени с другими работами на этом элементе. Для этого ищем по списку назначенных работ элемента место для вставки работы, просматривая резервы в «реальном» времени элемента в порядке с конца в начало в поиске резерва, оканчивающегося не позже, чем директивный срок назначаемой работы, и длиной не менее ее длительности. Если требуемый резерв не найден в фонде времени данного элемента, то продолжаем поиск до начала планового периода (если начало выполнения работ будет меньше начала планового периода, эти работы будут сдвигаться к началу планового периода после выполнения алгоритма, используя существующие резервы).

6. Перенос всех назначенных работ из списка неназначенных в список назначенных. Выбор всех предшественников назначенных работ и добавление их в список неназначенных работ по общим правилам, определив для каждого предшественника директивный срок как момент запуска назначенной работы – ее преемника, с учетом отношений предшествования по сети. Переход на п.3.

7. Выдать список назначенных работ с информацией о временах запуска и окончания агрегированных работ как пооперационный план выполнения работ.

Третий альтернативный метод является модификацией метода для базового критерия, со следующими изменениями:

1) метод применяется для всех групп изделий, которые рассматриваются последовательно, в порядке, соответствующем выбранному согласованному плану;

2) п.п. 1 и 2 метода для базового критерия применяются без изменений, если ресурс свободен для назначения работы (в этом случае работы назначаются в соответствии с расписанием, полученным в результате решения одноэтапных задач оптимизации). Для случая, если ресурс может быть частично занят ранее назначенными конфликтующими работами, п. 1 и 2 записываются следующим образом:

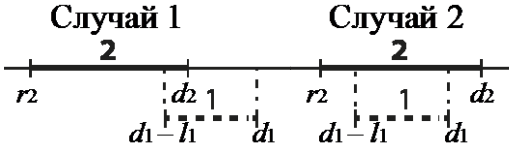
1. В силу свойств технологической сети, соответствующей модели планирования по одному из базовых критериев, однозначно директивные сроки конечных работ, где необходимо, пересчитываются в директивные сроки выполнения работ элементов типа 1–5 последнего (в общем случае, предпоследнего или того, с которым непосредственно связаны конечные работы) ряда сети. Если назначению работ препятствуют ранее назначенные конфликтующие работы, то рассчитанный для них директивный срок уменьшается до момента запуска конфликтующих работ, либо моменты окончания конфликтующих работ уменьшаются до предполагаемого момента запуска назначаемых работ с соответствующей корректировкой директивных сроков предшественников конфликтующих работ, если такая корректировка выгоднее.

2. Полученные расписания для элементов типа 2–5 последнего ряда сети однозначно задают наиболее поздние моменты готовности на выполнение работ на входе элементов типа 1–5. Эти моменты времени в силу свойств технологической сети однозначно задают директивные сроки выполнения работ для элементов типа 1–5 предпоследнего ряда сети либо элементов типа 1–5, непосредственно предшествующих элементам типа 1–5 последнего ряда сети. Если назначению работ препятствуют ранее назначенные конфликтующие работы, то рассчитанный для них директивный срок уменьшается до момента запуска конфликтующих работ, либо моменты окончания конфликтующих работ уменьшаются до предполагаемого момента запуска назначаемых работ с соответствующей корректировкой директивных сроков предшественников конфликтующих работ, если такая корректировка выгоднее. Та же ситуация касается элементов типа 1–5 любых рядов сети. Для элементов типа 1–5 предпоследнего ряда сети реализуется п.1 общего метода.

При изменении директивных сроков предшественников конфликтующей работы следует

также пересчитать одноэтапные задачи на элементах типа 2–5 предшествующих рядов сети, если такие задачи уже были решены. Поэтому, корректировка позиций конфликтующих работ может оказаться более трудоемкой, чем корректировка директивных сроков назначаемых работ, хотя и, возможно, более выгодной.

Какая из двух корректировок выгоднее, определяется величиной уменьшения директивного срока. Покажем на примере для одной работы различные варианты корректировки. Пусть назначаемая работа 1 имеет длительность l_1 и минимально возможный момент запуска d_1, но на тот же ресурс назначена конфликтующая работа 2 с моментом запуска $r_2 = d_2 - l_2$ и моментом окончания d_2. Возможны два случая:



Случай 1: $d_1 \geq d_2$, $d_1 - d_2 < l_1$. При корректировке директивного срока назначаемой работы его изменение равно $\Delta_{1Н} = d_1 - r_2$. При корректировке директивного срока конфликтующей работы он уменьшается на $\Delta_{1К} = d_2 - (d_1 - l_1)$.

Случай 2: $r_2 < d_1 \leq d_2$. При корректировке директивного срока назначаемой работы его изменение равно $\Delta_{2Н} = d_1 - r_2$. При корректировке директивного срока конфликтующей работы он уменьшается на $\Delta_{2К} = d_2 - (d_1 - l_1)$.

Как видим, $\Delta_{1Н} = \Delta_{2Н}$ и $\Delta_{1К} = \Delta_{2К}$. Таким образом, выбирается та корректировка, которой соответствует наименьшее изменение директивного срока:

а) если $d_1 - r_2 \leq d_2 - (d_1 - l_1)$ (или, что то же самое, $2(d_1 - d_2) \leq l_1 - l_2$), уменьшаем директивный срок назначаемой работы на $d_1 - r_2$;

б) в противном случае уменьшаем директивный срок конфликтующей работы и ее назначенных предшественников на $d_2 - (d_1 - l_1)$.

В общем случае, пусть множество работ I на элементе сети и множество K конфликтующих работ следует разнести по времени. Третий метод заключается в том, что расписание всех работ на данном элементе сети должно сдвигаться влево с целью расхождения с множеством K конфликтующих работ, либо конфликтующие работы и их предшественники должны сдвигаться влево для расхождения с множеством работ I (под множеством конфликтующих работ понимается множество всех работ на элементе, на котором находятся конфликтующие работы). Поэтому одноэтапная задача на элементе решается без учета конфликтующих работ, а затем выполняется проверка: если хотя бы для одной работы из множества работ I на элементе сети выполняется $d_i \geq d_k$, $d_i - d_k < l_i$ или $d_k - l_k < d_i \leq d_k$, $i \in I$, $k \in K$, то моменты запуска и окончания *всех* работ на назначаемом элементе уменьшаются на величину $\max_i d_i - \min_k(d_k - l_k)$, $i \in I$, $k \in K$, либо моменты запуска и окончания *всех* работ на конфликтующем элементе и их предшественников уменьшаются на $\min_i(d_i - l_i) - \max_k d_k$, $i \in I$, $k \in K$, в зависимости от того, какая из величин меньше. Такой сдвиг осуществляется, пока не будет найден резерв для выполнения всего расписания работ на назначаемом элементе.

Четвертый метод является модификацией третьего метода. В нем используется три варианта разнесения работ: если бóльшая часть назначаемых работ не может быть назначена из-за ранее назначенных конфликтующих работ, то применяется сдвиг всего расписания работ на элементе, в противном случае (если пересекается с конфликтующими меньшая часть назначаемых работ) используется сдвиг либо всех работ на назначаемом элементе (работ множества I), либо всех работ на конфликтующем элементе (работ множества K), в зависимости от того, какой из сдвигов меньше. Таким образом, минимизируется нежелательный сдвиг расписания уже назначенных работ. Итак, если для большей части работ из множества I на элементе сети выполняется:

$$d_i \geq d_k,\ d_i - d_k < l_i \text{ или } d_k - l_k < d_i \leq d_k,\ i \in I,\ k \in K, \qquad (3)$$

то моменты запуска и окончания *всех* работ на элементе уменьшаются на величину $\max_i d_i - \min_k(d_k - l_k)$, $i \in I$, $k \in K$. Такой сдвиг осуществляется, пока для всех работ не будет найден свободный резерв на элементе. Если же условие (3) выполняется для меньшей части работ из множества работ I, то моменты запуска и окончания *всех* работ на назначаемом элементе уменьшаются на величину $\max_i d_i - \min_k(d_k - l_k)$, $i \in I$, $k \in K$, либо моменты запуска и окончания *всех* работ на конфликтующем элементе и их предшественников (с повторным решением одноэтапных задач на элементах-предшественниках) уменьшаются на $\min_i(d_i - l_i) - \max_k d_k$, $i \in I$, $k \in K$, в зависимости от того, какая из величин меньше. Такой сдвиг осуществляется, пока не будет найден резерв для выполнения всего расписания работ на назначаемом элементе.

Во всех четырех методах для элементов типа накопитель при определении директивных сро-

ков выполнения работ используется общее правило: директивный срок работ слева от накопителя равен минимальному из моментов запуска работ справа от накопителя.

Во всех методах для более эффективного решения одноэтапных задач на элементах типа 3–5 (с целью увеличения максимального момента начала выполнения работ на элементе и уменьшения сдвига предшественников) можно использовать разбивку работ на партии. Например, если изделие делается в количестве 3 единицы, то все или некоторые работы изделия при необходимости можно разбить на 3 партии одинаковой длительности, равной 1/3 от исходной длины работы для всей серии изделий, и размещать эти партии на различных приборах элемента типа 3–5, если это приводит к улучшению значения функционала одноэтапной задачи календарного планирования на этом элементе.

Если в результате пересчета моментов выполнения некоторые работы выполняются до начала планового периода, то осуществляется их сдвиг к моменту начала планового периода, в сторону увеличения интервала выполнения работ на элементах-преемниках. При этом порядок выполнения всех работ сохраняется, а моменты запуска элементов сети типа 2–5 сдвигаются минимально для выполнения условия непрерывности их работы, с состыковкой задач на элементах по принципу разнесения по времени, с выполнением условия: максимальный момент окончания выполнения работ на элементе-предшественнике меньше или равен минимальному моменту запуска работ на выполнение на элементе-преемнике. Такой сдвиг нужно начинать с работ, имеющих наиболее ранний момент начала выполнения. Имеющиеся на элементах свободные резервы времени позволяют иногда сдвинуть выполнение работ без нарушения директивных сроков изделий. После соответствующего сдвига получаем окончательный пооперационный план.

Из пооперационных планов, полученных всеми альтернативными методами, выбирается план с наибольшим значением функционала или (если значение функционала одинаково) по другим характеристикам. В выбранном пооперационном плане анализируются моменты окончания последних работ изделий на элементах последнего уровня входимости.

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

Если полученное решение оказалось недопустимым (моменты выполнения некоторых конечных работ больше директивных сроков, установленных заказчиками), для этого расписания заново рассчитывается прибыль по исходному функционалу. Если новая рассчитанная прибыль не меньше ожидаемой минимальной, то необходимо перейти на блок принятия решений для согласования новых директивных сроков с заказчиком. Если прибыль меньше минимальной ожидаемой, то в случае планирования по базовому критерию оптимальности реализуем п. 9) блока принятия решений [1]: В блоке принятия решений при планировании по базовому критерию оптимальности, в случае необходимости, экспертным путем устанавливаются желаемые моменты окончания выполнения изделий, которые передаются на модель третьего уровня в качестве директивных сроков. Если критерий синтетический, то непосредственно в блоке принятия решений изменяется портфель заказов. Если в результате реализации п. 9) блока принятия решений новая прибыль меньше минимальной ожидаемой, то в блоке принятия решений изменяется портфель заказов.

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

Блок 5. Модель оперативного планирования (четвертый уровень модели планирования)

На четвертом уровне модели реализуется оперативное планирование в случае частичного невыполнения плана, полученного в блоке 4.

Задача оперативной корректировки (оперативного планирования) возникает, если в результате срыва производственного процесса на

отдельных элементах отдельные работы пооперационного плана выполнены позже запланированных сроков.

Формализация модели оперативного планирования. Постановка задачи. Имеется пооперационный план выполнения работ элементами исходной сети, построенный на третьем уровне модели. Корректировка выполняется в произвольный момент времени (назовем его моментом пересчета), определяемый в блоке принятия решений. К этому моменту план частично выполнен (часть сети содержит выполненные работы, часть еще не выполнены, часть еще выполняются), часть выполненных работ закончены позже запланированных сроков (фактический момент окончания $t_ф > t_{кон}$). Если на момент пересчета есть работы, которые должны были быть выполнены, но еще выполняются, то оценивается их предполагаемый момент окончания выполнения. Задача оперативной корректировки заключается в построении нового пооперационного плана выполнения работ, оптимального по критериям, связанным с запаздыванием относительно утвержденных ранее директивных сроков, учитывающего фактические моменты окончания уже выполненных работ и предполагаемые моменты окончания выполняющихся работ.

Во всех моделях четвертого уровня минимизируются аддитивные функционалы, компонентами которых являются новые весовые коэффициенты в функционале качества – штрафы за запаздывание либо за опережение/запаздывание относительно директивных сроков (базовые критерии 3 и 5). Штрафы устанавливаются вместе с директивными сроками на 3-м уровне в блоке принятия решений. Соответствие исходного критерия оптимальности и критерия на четвертом уровне задается следующим образом: если исходный базовый критерий – 1 или 3, то на четвертом уровне для данной подсети используется базовый критерий 3. Если исходный базовый критерий 4 или 5, то на четвертом уровне он не меняется.

На четвертом уровне не рассматривается критерий оптимизации, требующий выполнения изделий «точно в срок» (критерий 2). При изменении пооперационного плана получение нового пооперационного плана с выполнением изделий «точно в срок» в общем случае является неразрешимой математической задачей. Для групп изделий, планируемых по критерию 2, должен быть выполнен пооперационный план 3-го уровня. Если этот план уже нарушен, в блоке принятия решений специалистами новый пооперационный план корректируется таким образом, чтобы он мог быть выполнен в прежние сроки. При этом может меняться технология (добавляться оборудование), увеличиваться мощность (производительность, количество смен) и т.п. Эта эргатическая процедура должна быть выполнена до рассмотрения групп изделий по критериям 3, 4 и 5, которые подвергаются корректировке.

Для синтетического критерия оптимальности группы изделий, планируемые по критерию 3, объединяются в одну подсеть, и группы изделий, планируемые по критериям 4 и 5, остаются в своих подсетях. Для каждой группы изделий решается своя задача МВМ, а согласованный план строится для одного или нескольких (если имеются группы изделий по разным критериям) порядков распределения групп изделий.

Если в результате согласованного планирования получено несколько альтернативных планов, то для выполнения на третьем уровне выбирается план, имеющий минимальное суммарное взвешенное запаздывание относительно утвержденных на третьем уровне директивных сроков. При этом, от штрафов за запаздывание отнимается суммарная прибыль, полученная для группы изделий, планируемых по критерию 1.

Универсальный метод оперативного планирования с помощью модернизированного метода согласованного планирования

Очевидно, что решение задачи не может быть реализовано обычным сдвигом невыполненных работ вперед по времени, т.к. такой сдвиг не учитывает заданных критериев оптимизации. Поэтому для решения предлагается модернизированный метод согласованного планирования. Его суть состоит в том, что общая процедура планирования на уровнях 1–3 модели повторяется с учетом следующих модификаций:

1) из исходной сети исключаются выполненные работы и связи между ними; выполняющиеся (на момент пересчета) задания считаются выполненными, но по общим правилам учитываются запланированные или предполагаемые моменты окончания их выполнения, как наиболее ранние моменты запуска их преемников; при пересчете остальных работ во время согласованного планирования учитывается занятость оборудования работами, не включенными в пересчет;

2) если критерий оптимизации базовый, то пересчитываются все оставшиеся работы; если критерий синтетический, то группы изделий, в которых нет нарушений, не включаются в пересчет (часть работ, предшественники которых выполнены без задержки, и которые не связаны общим оборудованием с работами, по которым есть задержка – для этого реализуется человеко-машинная процедура анализа того, какая часть сети не меняется до конца изготовления изделий); однако, при пересчете остальных групп изделий во время согласованного планирования учитывается занятость оборудования работами групп изделий, не включенных в пересчет;

3) в оставшейся части сети добавляются новые условия для невыполненных работ: они могут начинаться не раньше, чем фактически выполнены их исключенные из сети предшественники (и не раньше начала нового планового периода, которым является момент пересчета); момент пересчета считается нулевым моментом времени для расчета критических путей, задачи МВМ, согласованного и точного планирования;

4) выполненные и выполняющиеся работы не участвуют в планировании на первом и втором уровнях, а учитываются только при решении одноэтапных задач календарного планирования на третьем уровне, не изменяя своих моментов окончания выполнения;

5) если в момент пересчета на элементе есть как выполненные (выполняющиеся), так и невыполненные работы, одноэтапная задача на этом элементе решается одним из следующих методов:

а) если выполненные или выполняющиеся работы имеются на всех приборах элемента, то одноэтапная задача решается одним из алгоритмов, позволяющих иметь начальное расписание, которым в данном случае являются выполненные или выполняющиеся на момент пересчета работы (если такие алгоритмы существуют);

б) одноэтапная задача решается для невыполненных работ. Если в полученном расписании момент запуска работ на каком-либо приборе меньше момента окончания уже назначенных (выполненных или выполняющихся) работ, то пробуются варианты:

– элемент разбивается на два последовательных элемента того же типа, между которыми вставляется элемент типа накопитель (пример см. на рис. 1 или 4). При этом выполненные работы помещаются в предыдущий элемент, а невыполненные – в следующий за ним элемент-преемник, и минимальный момент запуска работ элемента-преемника равен максимальному из моментов окончания работ на элементе-предшественнике. Расписание принимается, если все работы в элементе-преемнике выполняются без запаздывания;

– некоторые или все работы полученного расписания сдвигаются влево, располагаясь в имеющихся резервах до уже назначенных на элементе работ, таким образом, чтобы приборы работали непрерывно. При этом должно соблюдаться ограничение: каждая сдвигаемая работа не должна начинаться раньше момента пересчета;

– некоторые или все работы полученного расписания сдвигаются вправо или переставляются на другие приборы элемента, располагаясь после уже назначенных на элементе работ таким образом, чтобы приборы работали непрерывно, а директивные сроки работ не были нарушены;

– некоторые или все работы полученного расписания сдвигаются вправо или переставляются на другие приборы элемента, располагаясь после уже назначенных на элементе работ таким образом, чтобы приборы работали непрерывно, а работы, по которым директивный срок нарушен, переносятся влево или на другие приборы таким образом, чтобы начинаться после момента пересчета и чтобы приборы работали непрерывно;

в) если ни один из методов не позволил получить допустимое расписание на элементе, то оно корректируется специалистами в блоке принятия решений. При этом может меняться технология (добавляться физическое оборудование), применяться разбивка на партии и т.п.

6) при построении пооперационного плана по базовому критерию возможны ситуации, не предусмотренные для базового критерия, когда требуемый ресурс элемента оказывается частично или полностью занятым ранее назначенными работами. В этом случае, директивные сроки рассматриваемых работ уменьшаются и становятся равными моментам начала выполнения уже назначенных работ, либо для назначения работ используются другие универсальные методы, описанные для синтетического критерия, т. к. они подходят и для базового критерия;

7) после получения пооперационного плана для невыполненных работ полный пооперационный план получается совмещением его с исходным пооперационным планом для выполненных работ.

8) Если нарушены директивные сроки, или прибыль, рассчитанная для нового пооперационного плана, меньше минимальной ожидаемой, в блоке принятия решений специалистами новый пооперационный план корректируется таким образом, чтобы он мог быть выполнен в прежние сроки. При этом может меняться технология (добавляться оборудование), увеличиваться мощность (производительность, количество смен) и т.п.

Данный метод может без изменений применяться и при планировании в сети произвольного вида как по базовому, так и по синтетическому критерию оптимальности.

Методология оперативного планирования для случая, когда задача планирования третьего уровня решалась по базовому критерию

Задачу коррекции пооперационного плана, полученного в результате решения многоэтапной сетевой задачи календарного планирования по базовому критерию 1, 3, 4 или 5, будем решать в следующей постановке. Частичное выполнение пооперационного плана, не совпадающее с пооперационным планом, полученным на третьем уровне, зафиксировало наиболее ранние моменты запуска всех либо части приборов (элементов сети типа 2–5) последнего ряда сети, которые могут оказаться позже моментов запуска приборов в пооперационном плане, полученном на третьем уровне планирования, а также зафиксированы работы, порядок выполнения которых оказался известным и не совпадающим с пооперационным планом третьего уровня. В этом случае коррекция плана проводится следующим образом.

1. Для приборов (элементов типа 2–5) последнего ряда сети уточняются множества работ на выполнение в произвольном порядке: моменты готовности на выполнение или очередность выполнения остальных работ оказались известными и не совпадающими с расписанием, полученным на третьем уровне. Для этого множества работ, в соответствии с топологией конечной части сети, уточняются (в случае необходимости) директивные сроки их выполнения.

Пример. Если у работы 11 (см. рис. 5) момент окончания выполнения T_{11} увеличился, и в результате увеличился момент начала работы 21 T_{21}, то можно увеличить и директивные сроки работ 12 и 13 (не больше, чем до T_{21}), что увеличит резервы для выполнения этих работ.

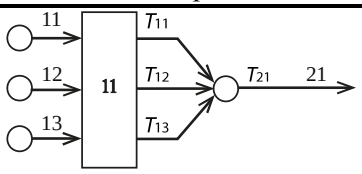


Рис. 5. Пример коррекции директивных сроков

2. Расписание выполнения работ для приборов (элементов типа 2–5) последнего ряда сети, которые не включены в п. 1, соответствуют расписанию третьего уровня планирования.

3. Для приборов (элементов типа 2–5) последнего ряда сети с измененными моментами запуска решаются следующие задачи:

3.1. Для элемента типа 2 (один прибор; работы, назначенные на обсулуживание, имеют разные директивные сроки) строится расписание, минимизирующее суммарное (задача 3.1) либо суммарное взвешенное (задача 3.1а) запаздывание работ относительно директивных сроков либо расписание, минимизирующее суммарное опережение/запаздывание работ относительно директивных сроков (задача 3.1б).

3.2. Для элемента типа 3 строится расписание выполнения работ на идентичных параллельных приборах, минимизирующее суммарное запаздывание работ относительно директивных сроков (задача 3.2а) либо минимизирующее максимальное отклонение работ от директивных сроков (задача 3.2б).

3.3. Для элементов типа 4–5 (параллельные приборы равной либо разной производительности, у обслуживаемых работ различные директивные сроки) точных эффективных алгоритмов решения задачи получения расписания, минимизирующего суммарное (суммарное взвешенное) запаздывание, не существует. Предлагается следующий эргатический алгоритм ее решения.

1) Для каждого из независимых приборов оставляется множество работ на выполнение, соответствующее пооперационному плану, полученному на третьем уровне планирования.

2) Для каждого прибора решается задача 3.1 либо 3.1а либо 3.1б.

3) Полученное расписание для независимых приборов (элементов типа 4–5) анализируется в блоке принятия решений, и расписание принимается, либо экспертным путем изменяется множество работ, назначенных на выполнение на каждом приборе. Затем снова реализуется п. 2) алгоритма. В случае необходимости, эта процедура может быть повторена несколько раз.

4) В блоке принятия решений специалистами корректируется текущий пооперационный план таким образом, чтобы, в соответствии с технологическим процессом, расписание, полученное

для элементов типа 2–5 последнего ряда сети, могло быть выполнено. При этом может меняться технология (добавляться оборудование), увеличиваться мощность (увеличить производительность, количество смен) и т.п.

Метод, описанный в данном параграфе, эффективен, если возможна реализация п. 4) этого метода (корректировка расписания элементов последнего ряда сети экспертами). В случае невозможности такой корректировки, для решения задачи по базовому критерию следует использовать универсальный метод, описанный выше.

Пример решения задачи

Рассмотрим группу изделий 1–5 из примера, приведенного в [1]. Планирование по данной группе изделий выполнялось по базовому критерию 1. На рис. 6 показана исходная сеть с фиктивными работами (технологическая сеть, задающая технологию выполнения изделий). Над стрелками написаны номера работ и элементов, а их характеристики сведены, соответственно, в табл. 1 и 2. Удельная длительность l_i выполнения работы на одно изделие задается в минутах.

В качестве примера, построим пооперационный план по методу планирования для синтетического критерия и рассмотрим его корректировку в случае частичного невыполнения.

Табл. 1. Характеристики работ

№(i)	l_i	№(i)	l_i	№(i)	l_i	№(i)	l_i
11	1	**24**	6	**41**	7	**55**	3
12	4	**25**	8	**42**	15	**56**	6
13	2	**26**	8	**43**	2	**57**	9
14	5	**27**	12	**44**	15	**58**	3
15	15	**28**	20	**45**	3	**59**	5
16	1	**29**	11	**46**	3	**510**	8
17	8	**210**	5	**47**	6	**61**	11
18	3	**30**	7	**48**	1	**62**	12
19	3	**31**	9	**49**	5	**63**	15
110	1	**32**	12	**410**	9	**64**	11
111	4	**33**	18	**51**	6	**65**	15
21	4	**34**	2	**52**	2	**71**	24
22	2	**35**	4	**53**	6		
23	1	**36**	7	**54**	7		

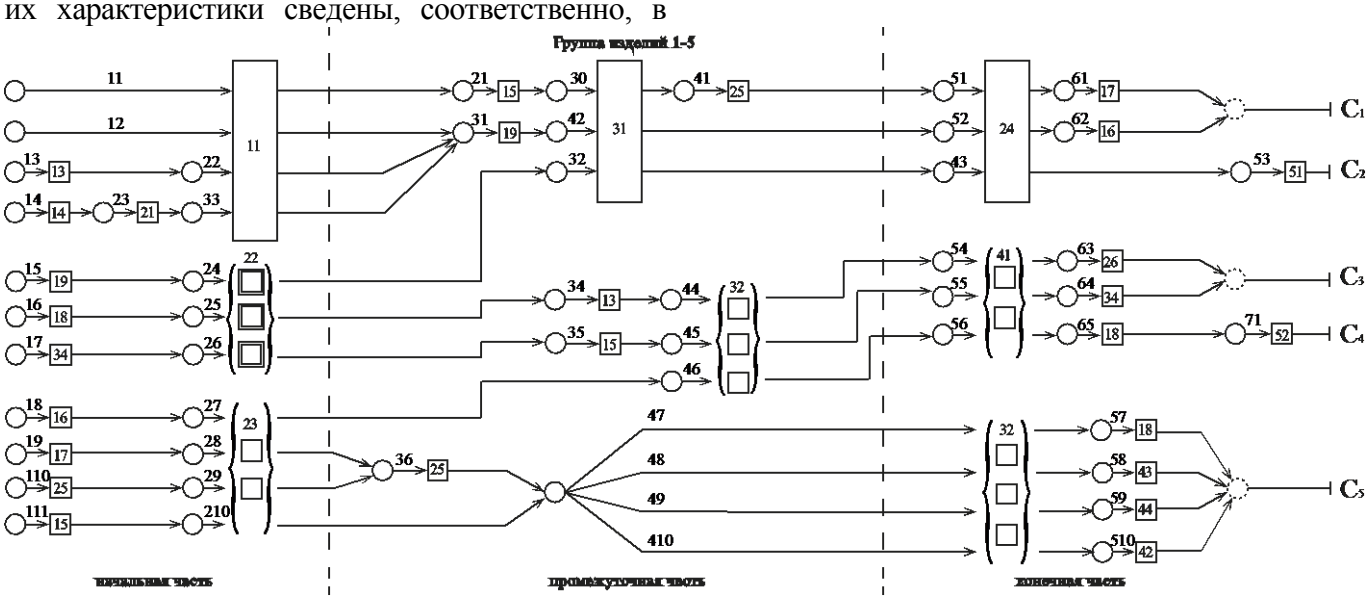


Рис. 6. Исходная сеть (группа изделий 1–5)

Табл. 2. Характеристики ресурсов

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

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

Характеристики изделий зададим в табл. 3 (N_i – число изделий в портфеле заказов, ω_i – вес, j_{end} – номера конечных работ изделия).

Табл. 3. Характеристики изделий

№(i)	j_{end}	N_i	ω_i
1	61, 62	2	1
2	53	4	2
3	63, 64	3	3
4	71	5	4
5	57–59, 510	10	5

Согласованный план выполнения агрегированных работ по данной группе изделий приведен в табл. 4 [1].

Табл. 4. Распределение агрегированных работ группы изделий 1–5

при-бор	ра-бота	изде-лие	$t_{нач}$	$t_{кон}$	при-бор	ра-бота	изде-лие	$t_{нач}$	$t_{кон}$
11	11	1	12	62	24	43	2	170	178
13	13	1	0	4	24	51	1	178	194
13	34	3	60	66	25	110	5	0	10
14	14	1	0	10	25	41	1	164	178
15	111	5	0	40	25	36	5	225	295
15	35	3	60	72	26	63	3	105	150
15	21	1	72	80	31	32	2	72	120
16	18	4	0	15	31	30	1	120	164
16	62	1	194	218	32	46	4	67	72
17	19	5	0	30	32	44	3	72	90
17	61	1	194	216	32	47	5	295	365
18	16	3	0	3	34	17	3	0	24
18	65	4	90	165	34	64	3	105	138
18	57	5	365	455	41	56	4	75	90
19	15	2	0	60	41	54	3	90	105
19	31	1	62	80	42	510	5	365	445
21	23	1	10	12	43	58	5	365	395
22	25	3	36	60	44	59	5	365	415
22	24	2	60	72	51	53	2	178	202
23	27	4	15	45	52	71	4	165	285
23	28	5	45	225					

Все общие вершины этой группы изделий объединились при согласованном планировании, поэтому исходная сеть не изменилась. В блоке 4 формируется многоэтапная сетевая задача календарного планирования, директивные сроки для конечных работ изделий определяются моментами C_i окончания их конечных работ, полученными при согласованном планировании (табл. 5), решение многоэтапной сетевой задачи календарного планирования определяет пооперационный план выполнения работ на исходной сети.

Табл. 5. Моменты окончания изделий C_i

№(i)	j_{end}	C_i
1	62	218
2	53	202
3	63	150
4	71	285
5	57	455

j_{end} – номер конечной работы изделия.

Выполняем дезагрегацию агрегированных работ: к согласованному плану выполнения агрегированных работ добавляем работы исходной сети, прежде вошедшие в агрегированные работы, помечая их как неназначенные. Длительности всех работ теперь соответствуют исходной сети. Используем первый альтернативный метод для синтетического критерия оптимальности.

Решение одноэтапных задач календарного планирования на элементах типа 3–5 показано на рис. 7 (для наглядности выбран различный масштаб времени на разных элементах).

В результате размещения неназначенных работ на приборах и соответствующих сдвигов предшественников получаем следующий недопустимый пооперационный план (табл. 6; работы 15, 18 и 19 выполняются до начала планового периода).

Табл. 6. Предварительный пооперационный план для группы изделий 1–5

при-бор	ра-бота	изде-лие	$t_{нач}$	$t_{кон}$	при-бор	ра-бота	изде-лие	$t_{нач}$	$t_{кон}$
11	11	1	12	14	24	43	2	170	178
11	12	1	14	22	24	51	1	178	190
11	22	1	22	26	24	52	1	190	194
11	33	1	26	62	25	110	5	0	10
13	13	1	0	4	25	41	1	164	178
13	34	3	33	39	25	36	5	205	275
14	14	1	0	10	26	63	3	105	150
15	111	5	0	40	31	32	2	72	120
15	35	3	60	72	31	30	1	120	134
15	21	1	72	80	31	42	1	134	164
16	18	4	**-15**	0	32	46	4	60	75
16	62	1	194	218	32	44	3	39	84
17	19	5	**-25**	5	32	45	3	81	90
17	61	1	194	216	32	47	5	305	365
18	16	3	0	3	32	48	5	305	315
18	65	4	90	165	32	49	5	315	365
18	57	5	365	455	32	410	5	275	365
19	15	2	**-27**	33	34	17	3	0	24
19	31	1	62	80	34	64	3	105	138
21	23	1	10	12	41	54	3	84	105
22	25	3	9	33	41	56	4	75	90
22	26	3	30	60	41	55	3	90	99
22	24	2	33	57	42	510	5	365	445
23	27	4	0	60	43	58	5	365	395
23	28	5	5	205	44	59	5	365	415
23	29	5	60	170	51	53	2	178	202
23	210	5	205	255	52	71	4	165	285

Сдвиг расписания работ, начинающихся до начала планового периода, к моменту времени 0, нужно начинать с работ, имеющих наибольшее такое отклонение (в данном случае, с работы 15, потом 19 и 18). Имеющиеся на элементах свободные резервы времени позволяют иногда сдвинуть выполнение работ без нарушения директивных сроков изделий. После соответствующего сдвига получаем окончательный пооперационный план (табл. 7, рис. 8).

Табл. 7. Окончательный пооперационный план для группы изделий 1–5

при-бор	ра-бота	изде-лие	$t_{нач}$	$t_{кон}$	при-бор	ра-бота	изде-лие	$t_{нач}$	$t_{кон}$
11	11	1	12	14	24	43	2	170	178
11	12	1	14	22	24	51	1	178	190
11	22	1	22	26	24	52	1	190	194

прибор	работа	изделие	$t_{нач}$	$t_{кон}$	прибор	работа	изделие	$t_{нач}$	$t_{кон}$
11	33	1	26	62	25	110	5	0	10
13	13	1	0	4	25	41	1	164	178
13	34	3	60	66	25	36	5	230	300
14	14	1	0	10	26	63	3	132	177
15	111	5	0	40	31	32	2	84	132
15	35	3	60	72	31	30	1	132	146
15	21	1	72	80	31	42	1	146	176
16	18	4	0	15	32	46	4	75	90
16	62	1	194	218	32	44	3	66	111
17	19	5	0	30	32	45	3	96	105
17	61	1	194	216	32	47	5	305	365
18	16	3	0	3	32	48	5	305	315
18	65	4	105	180	32	49	5	315	365
18	57	5	365	455	32	410	5	300	390
19	15	2	0	60	34	17	3	0	24
19	31	1	62	80	34	64	3	114	147
21	23	1	10	12	41	54	3	111	132
22	25	3	36	60	41	56	4	90	105
22	26	3	30	60	41	55	3	105	114
22	24	2	60	84	42	510	5	390	470
23	27	4	15	75	43	58	5	315	345
23	28	5	30	230	44	59	5	365	415
23	29	5	75	185	51	53	2	178	202
23	210	5	185	235	52	71	4	180	300

В этом плане изделия 3, 4, 5 выполнены с нарушением директивных сроков: $\mathbf{C}_3 = 177$ ($d_3 = 150$), $\mathbf{C}_4 = 300$ ($d_4 = 285$), $\mathbf{C}_5 = 470$ ($d_5 = 455$). Поэтому, если прибыль, получаемая от выполнения данного плана, не меньше ожидаемой минимальной, то необходимо перейти на блок принятия решений для согласования новых директивных сроков с заказчиком, а иначе использовать п. 9) блока принятия решений.

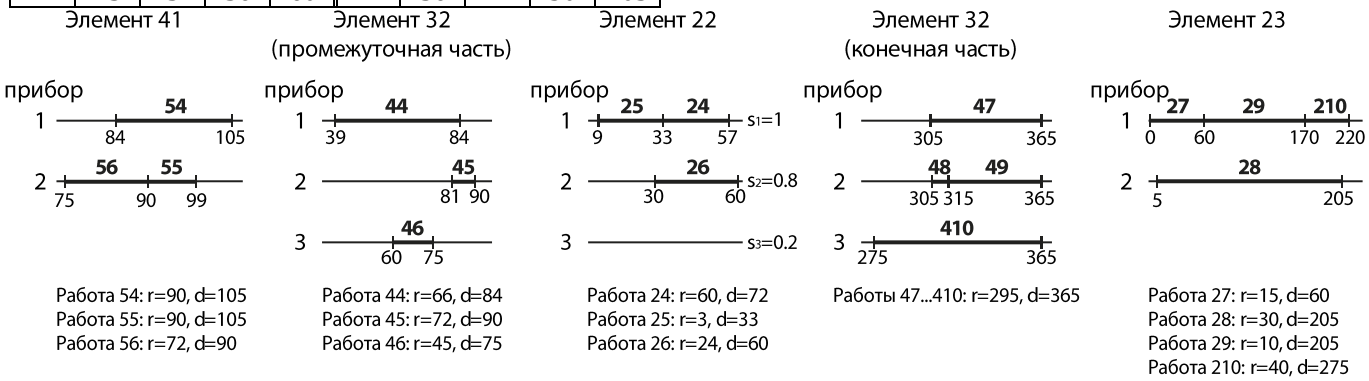


Рис. 7. Решение одноэтапных задач на элементах типа 3–5 (предварительный етап

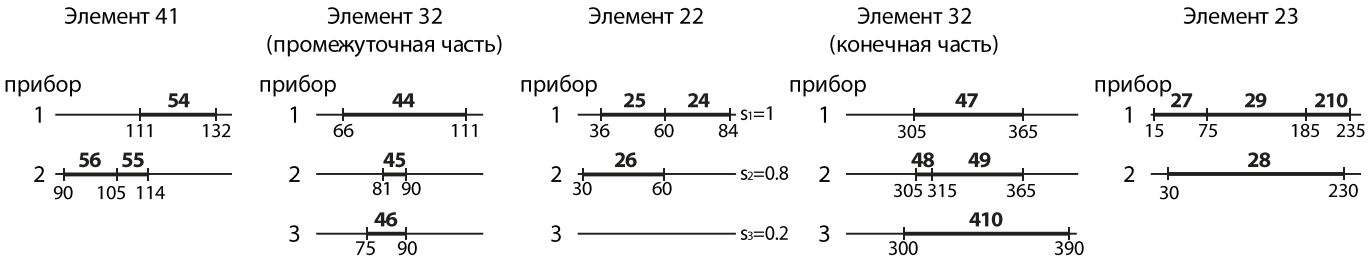


Рис. 8. Решение одноэтапных задач на элементах типа 3–5 (окончательный этап)

Рассмотрим пример *оперативной корректировки* пооперационного плана на некоторый текущий момент времени.

Пусть пооперационный план, приведенный в табл. 7, частично выполнился, и на момент времени 180 работа 30 выполнилась позже назначенного срока, а работы 42, 51 и 28 задерживаются (табл. 8).

В этом случае стоит задача уточнить моменты запуска $t_{нач}$ и окончания $t_{кон}$ работ, начиная с момента пересчета (180) таким образом, чтобы максимизировался функционал качества (критерий 3).

Рассмотрим решение этой задачи. В исходной сети отметим, в соответствии с табл. 8, серым цветом выполненные работы ($t_ф \le 180$), жирным черным цветом – задерживающиеся работы (рис. 9). Работы 28, 29, 51, 53 на момент пересчета еще выполняются. Работа 42 еще выполняется из-за задержки, и предполагаемый момент ее окончания равен 194. Предполагаемый момент окончания работы 28 равен 250.

Очевидно, что для решения задачи следует построить план для невыполненных работ с учетом моментов окончания уже выполненных работ и предполагаемых моментов окончания выполняющихся работ.

Табл. 8. Пооперационный план для группы изделий 1–5, требующий корректировки

при-бор	ра-бота	изде-лие	$t_{нач}$	$t_{кон}$	$t_ф$	при-бор	ра-бота	изде-лие	$t_{нач}$	$t_{кон}$	$t_ф$
18	16	3	0	3	3	31	32	2	84	132	132
13	13	1	0	4	4	41	54	3	111	132	132
14	14	1	0	10	10	**31**	**30**	**1**	**132**	**146**	**164**
25	110	5	0	10	10	34	64	3	114	147	147
21	23	1	10	12	12	26	63	3	132	177	177
11	11	1	12	14	14	25	41	1	164	178	178
16	18	4	0	15	15	24	43	2	170	178	178
11	12	1	14	22	22	18	65	4	105	180	180
34	17	3	0	24	24	23	29	5	75	185	–
11	22	1	22	26	26	**24**	**51**	**1**	**178**	**190***	–
17	19	5	0	30	30	**31**	**42**	**1**	**164**	**194***	–
15	111	5	0	40	40	24	52	1	190	194	–
19	15	2	0	60	60	51	53	2	178	202	–
22	26	3	30	60	60	17	61	1	194	216	–
22	25	3	36	60	60	16	62	1	194	218	–
11	33	1	26	62	62	**23**	**28**	**5**	**30**	**250***	–
13	34	3	60	66	66	23	210	5	185	235	–
15	35	3	60	72	72	52	71	4	180	300	–
23	27	4	15	75	75	25	36	5	230	300	–
19	31	1	62	80	80	32	48	5	305	315	–
15	21	1	72	80	80	43	58	5	315	345	–
22	24	2	60	84	84	32	47	5	305	365	–
32	46	4	75	90	90	32	49	5	315	365	–
41	56	4	90	105	105	32	410	5	300	390	–
32	45	3	96	105	105	44	59	5	365	415	–
32	44	3	66	111	111	18	57	5	365	455	–
41	55	3	105	114	114	42	510	5	390	470	–

*Предполагаемый момент окончания (фактический момент запуска плюс длительность)

Для решения задачи используем частичную исходную сеть (рис. 10). При этом выполненные и выполняющиеся работы 27, 28, 29, 30, 32, 41, 42, 43, 51, 53 задают наиболее ранние моменты начала выполнения следующих работ, не участвуя в планировании на первом и втором уровнях. Эти работы учитываются при решении одноэтапных задач календарного планирования на третьем уровне, но не изменяют своих моментов окончания выполнения.

На первом уровне получаем агрегированную сеть, показанную на рис. 11. При этом длительности агрегированных работ совпадают с полученными на первом уровне общей модели, кроме следующих: $l_{210} = 2{,}5$ (25 по серии изделий), $l_{52} = 2$ (4 по серии изделий).

Граф на критических путях показан на рис. 12. В результате первого уровня планирования получаем решение задачи МВМ (табл. 9, значение функционала равно 2635).

Табл. 9. Приоритетно-упорядоченная последовательность агрегированных работ

i	ПМП	l_i	ω_i	C_i	f_i
52	1	4		4	
62	1	24	1	28	28
71	2	120	4	148	592
210	3	25		173	
36	3	70		243	
47	3	70		313	
57	3	90	5	403	2015

На втором уровне в результате согласованного планирования по алгоритму 3 получаем частичный план выполнения агрегированных работ (табл. 10; используются директивные сроки, утвержденные на третьем уровне, а именно: $d_1 = 218$, $d_2 = 202$, $d_4 = 300$, $d_5 = 470$; работы назначаются с конца, начиная с директивных сроков изделий).

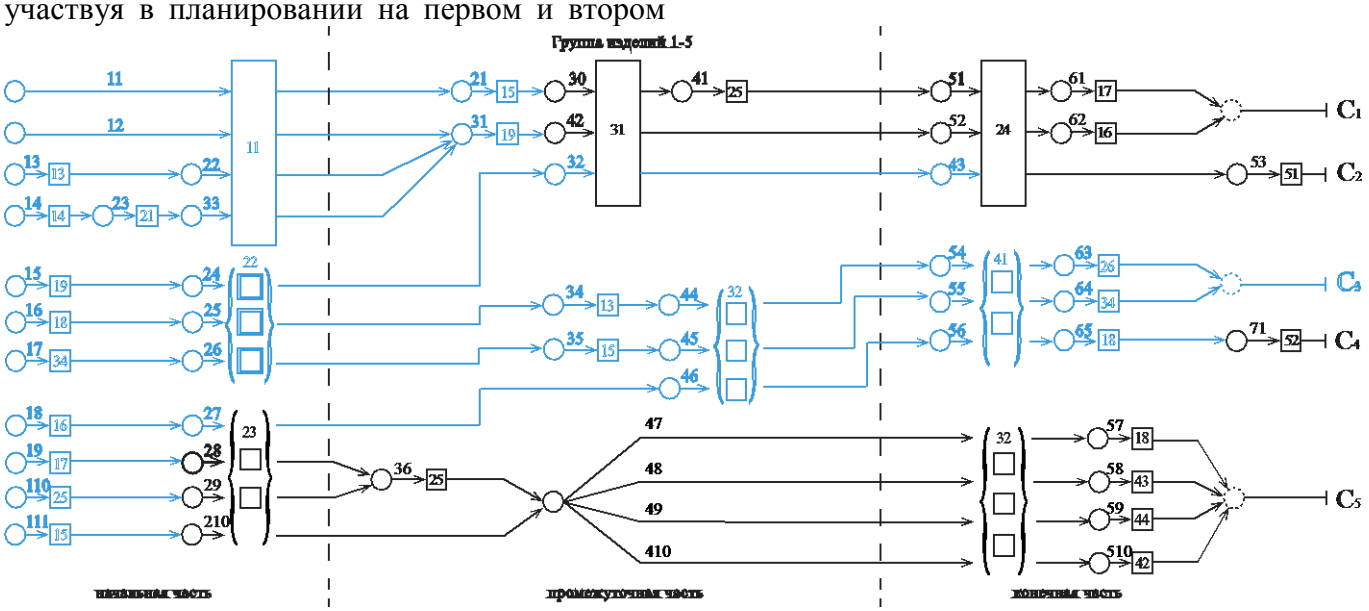


Рис. 9. Исходная сеть для операционного планирования на момент времени 180

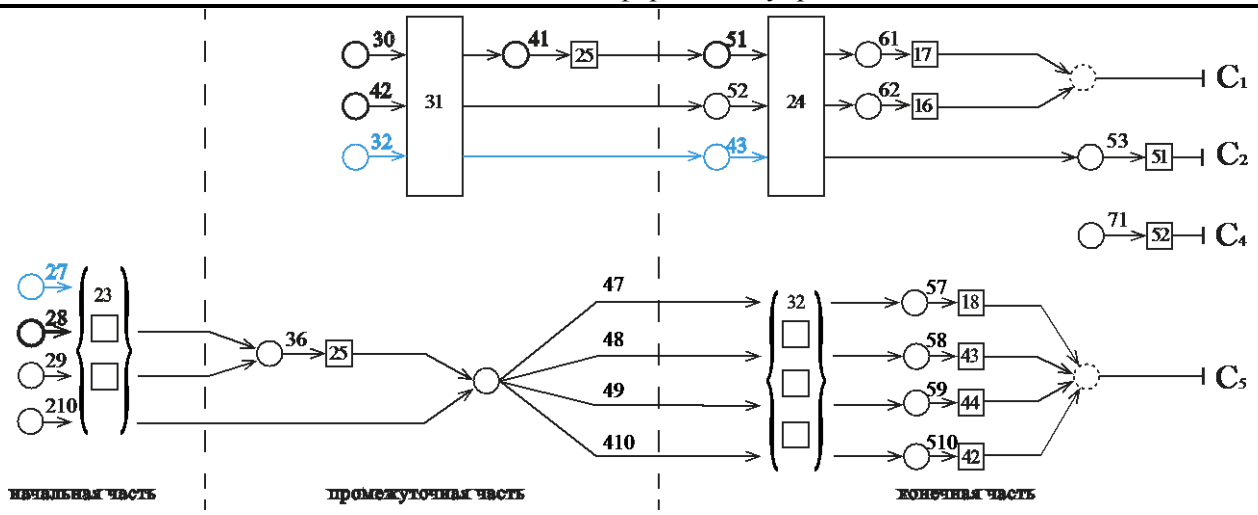


Рис. 10. Частичная исходная сеть из работ, подлежащих оперативной коррекции

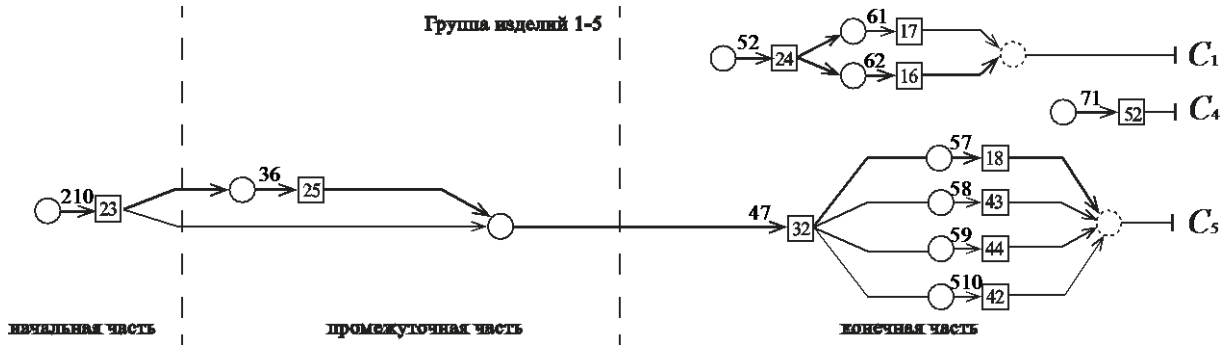


Рис. 11. Агрегированная сеть для оперативной коррекции

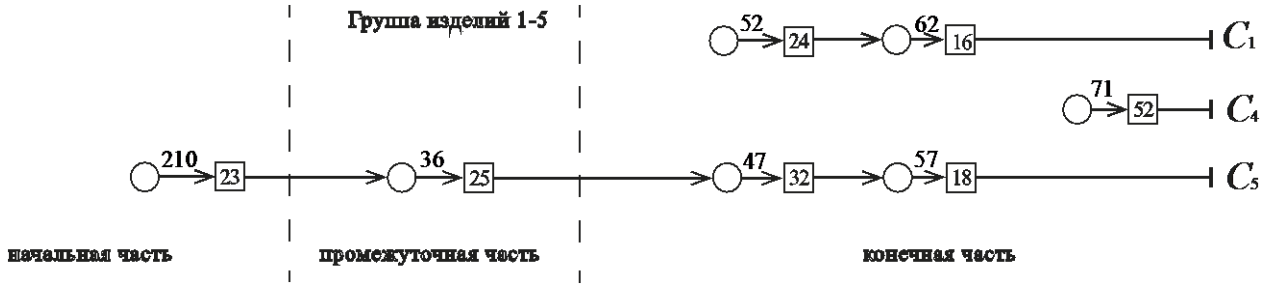


Рис. 12. Граф на критических путях изделий для оперативной коррекции

Табл. 10. Распределение невыполненных агрегированных работ

при-бор	ра-бота	изде-лие	$t_{нач}$	$t_{кон}$	при-бор	ра-бота	изде-лие	$t_{нач}$	$t_{кон}$
16	62	1	194	218	32	47	5	310	380
17	61	1	196	218	42	510	5	390	470
18	57	5	380	470	43	58	5	440	470
23	210	5	215	240	44	59	5	420	470
24	52	1	190	194	52	71	4	180	300
25	36	5	240	310					

На третьем уровне при решении одноэтапной задачи на элементе 24 (тип 2) имеем следующую ситуацию: выполненная работа 43 закончилась в момент 178 и выполняюшаяся за ней работа 51 должна закончиться в момент 190. Однако, работа 52 может начаться не ранее момента 194, т.к. выполнение ее предшественника было задержано. Получаем разрыв в расписании, чего на элементе типа 2 быть не должно. Поэтому, элемент 24 в исходной сети разобьем на два аналогичных (рис. 13):

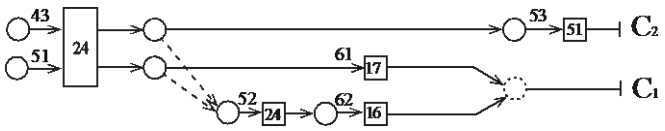


Рис. 13. Разбивка элемента 24 в результате решения одноэтапной задачи

В итоге, получаем следующий частичный пооперационный план:

Табл. 11. Пооперационный план для невыполненных работ

при-бор	ра-бота	изде-лие	$t_{нач}$	$t_{кон}$	$t_ф$	при-бор	ра-бота	изде-лие	$t_{нач}$	$t_{кон}$	$t_ф$
23	29	5	75	185	–	25	36	5	250	320	–
24	51	1	178	190	–	32	48	5	320	330	–
31	42	1	164	194	–	43	58	5	330	**360**	–
24	52	1	194	198	–	32	47	5	320	380	–
51	53	2	178	**202**	–	32	49	5	330	380	–
17	61	1	202	**224**	–	32	410	5	320	410	–
16	62	1	198	**222**	–	44	59	5	380	**430**	–
23	28	5	30	250	–	18	57	5	380	**470**	–
23	210	5	185	235	–	42	510	5	410	**490**	–
52	71	4	180	**300**	–						

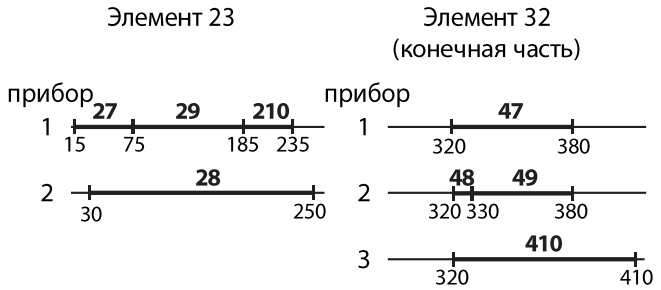


Рис. 14. Решение одноэтапных задач на элементах типа 3–5 (4-й уровень)

Жирным шрифтом отмечены моменты окончания конечных работ изделий. Таким образом, момент окончания выполнения изделия 5 увеличился на 20 и равен 490. Директивный срок по изделию 5 нарушен. Решение экспертов – использовать для работы 410 разбивку на партии и увеличить число приборов на элементе 32 до четырех: изделие 5 изготавливается в количестве 10 шт., поэтому работу 410 можно разбить на число партий, являющееся делителем десяти. Разобьем ее на 2 партии с длительностью 45, при этом можно по одной партии разместить на приборах 3 и 4 элемента 32. Получаем следующее расписание работ на элементе 32 (рис. 15):

Рис. 15. Решение одноэтапной задачи с разбивкой работы 410 на две партии

Тогда работа 410 заканчивается в момент 365, а ее преемник – работа 510 – в момент 445, что меньше, чем $d_5 = 470$.

Аналогично, в полученном пооперационном плане нарушен директивный срок по изделию 1 ($d_1 = 218$, $\mathbf{C}_1 = C_{61} = 224$, $C_{62} = 222$). Уменьшение функционала (суммарной прибыли) для такого плана составит всего 6 из 16888 у.е. Однако, если это неприемлемо, можно разбить работы 61 и 62 каждую на две партии, а элементы 16 и 17 преобразовать в элемент типа 2 с двумя приборами, в результате получаем, соответственно, длительности работ 11 и 12 и моменты их окончания 213 и 210.

Полный откорректированный пооперационный план приведен ниже:

Табл. 12. Откорректированный пооперационный план для группы изделий 1–5

при-бор	ра-бота	изде-лие	$t_{нач}$	$t_{кон}$	$t_ф$	при-бор	ра-бота	изде-лие	$t_{нач}$	$t_{кон}$	$t_ф$
18	16	3	0	3	3	41	54	3	111	132	132
13	13	1	0	4	4	31	32	2	84	132	132
14	14	1	0	10	10	31	30	1	132	164	164
25	110	5	0	10	10	26	63	3	132	177	177
21	23	1	10	12	12	25	41	1	164	178	178
11	11	1	12	14	14	24	43	2	170	178	178
16	18	4	0	15	15	18	65	4	105	180	180
11	12	1	14	22	22	23	29	5	75	185	–
34	17	3	0	24	24	24	51	1	178	190	–
11	22	1	22	26	26	31	42	1	164	194	–
17	19	5	0	30	30	24	52	1	194	198	–
15	111	5	0	40	40	51	53	2	178	202	–
19	15	2	0	**60**	60	17	61	1	202	213	–
22	26	3	30	60	60	16	62	1	198	210	–
22	25	3	36	60	60	23	28	5	30	250	–
11	33	1	26	62	62	23	210	5	185	235	–
13	34	3	60	66	66	52	71	4	180	300	–
15	35	3	60	72	72	25	36	5	250	320	–
23	27	4	15	**75**	75	32	48	5	320	330	–
19	31	1	62	80	80	43	58	5	330	360	–
15	21	1	72	80	80	32	47	5	320	380	–
22	24	2	60	84	84	32	49	5	330	380	–
32	46	4	75	90	90	32	410	5	320	365	–
41	56	4	90	105	105	44	59	5	380	430	–
32	45	3	96	105	105	18	57	5	380	470	–
32	44	3	66	111	111	42	510	5	365	445	–
41	55	3	105	114	114						

Выводы

Приводится методология реализации третьего и четвертого уровня четырехуровневой модели планирования, принятия решений и оперативного планирования в сетевых системах с ограниченными ресурсами в соответствии с общей схемой реализации алгоритмического обеспечения, представленной в [1]. Решение задачи показано на примере. Таким образом, в данной статье логически завершено формализованное описание методологии планирования, в резуль-

тате чего могут быть созданы информационные технологии, готовые к реализации в качестве программных систем планирования и поддержки принятия решений для сетевых систем с ограниченными ресурсами. Данные системы, по мнению авторов, найдут широкое применение для планирования производственных процессов дискретного типа как в Украине, так и за рубежом.

Список литературы

1. Згуровский М.З. Методология построения четырехуровневой модели планирования, принятия решений и оперативного планирования в сетевых системах с ограниченными ресурсами [Текст] / М.З. Згуровский, А.А. Павлов, Е.Б. Мисюра, О.В. Мельников, Т.Н. Лисецкий // Вісник НТУУ “КПІ”. Серія «Інформатика, управління та обчислювальна техніка». – К.: “ВЕК+”, 2014. – №61. – С.60–84.
2. Згуровский М.З., Павлов А.А. Принятие решений в сетевых системах с ограниченными ресурсами: Монография.– К.: Наукова думка. – 2010. – 573 с.
3. Конвей Р.В., Максвелл У.Л., Миллер Л.У. Теория расписаний.– М.: Наука, 1975.– 359 с.
4. Павлов А.А., Теленик С.Ф. Информационные технологии и алгоритмизация в управлении.– К.: Техника.– 2002.– 344 с.
5. Павлов А.А. Составление допустимого расписания выполнения работ на одном приборе с целью минимизации суммарного опережения работ / А.А. Павлов, Е.А. Халус // Вісник НТУУ “КПІ”. Серія «Інформатика, управління та обчислювальна техніка». – К.: “ВЕК+”, 2014. – №61. – С.27–34
6. Згуровский М.З. Задача построения допустимого расписания с максимально поздним моментом запуска и минимальным суммарным опережением / М.З. Згуровский, А.А. Павлов, Е.А. Халус // Системні дослідження та інформаційні технології, – 2015. – №1. – 8 с. (в печати)
7. Згуровский М.З. Минимизация лексикографического критерия для допустимого расписания на независимых параллельных приборах с произвольными директивными сроками [Текст] / М.З. Згуровский, А.А. Павлов, Е.Б. Мисюра // Вісник НТУУ “КПІ”. Серія «Інформатика, управління та обчислювальна техніка». – К.: “ВЕК+”, 2014. – №61. – С.4–17.

УДК 004.27

КЛИМЕНКО І.А.

ОПТИМІЗАЦІЯ РЕКОНФІГУРАЦІЇ В ДИНАМІЧНО РЕКОНФІГУРОВАНИХ ОБЧИСЛЮВАЛЬНИХ СИСТЕМАХ

Розроблений метод оптимізації процесу реконфігурації, за критеріями забезпечення необхідного додатку часу обчислення та врахування просторових обмежень реконфігурованої структури ПЛІС під час розміщення динамічного потоку задач, що дозволяє підвищити ефективність реконфігурації.

The method of optimization reconfiguration with the criteria providing timing requirements of applications and providing space requirements of reconfigurable field of FPGA when mapping of the dynamic flow of tasks is proposed. The proposed method increases the efficiently reconfiguration in the dynamic reconfigurable computer.

Вступ

В області реконфігурованих високопродуктивних обчислювальних систем [1 - 4] зменшення часу реконфігурації є актуальною проблемою сьогодення. Час виконання реконфігурації є критичним параметром зважаючи на його високу непродуктивну складову, пов'язану з необхідністю передавання великих обсягів конфігураційних даних, технологічними обмеженнями параметрів інтерфейсів обміну даними, складним процесом управління реконфігурованими ресурсами. Витрати часу на реконфігурацію, можуть бути навіть порівняні у часі з апаратним прискоренням, зводячи нанівець весь позитивний ефект від здійснення реконфігурації архітектури. Відомі способи та технології інтенсивного прискорення реконфігурації пропонують максимальне видалення непродуктивних витрат часу незважаючи на просторові обмеження реконфігурованих ресурсів ПЛІС. В той час як механізми подолання цих обмежень, такі як тимчасове вивантаження конфігурацій функціональних блоків, дефрагментація реконфігурованої області, зменшення кількості відмов під час розміщення задач, тощо, потребують додаткових ресурсів часу та продуктивності на їх реалізацію.

За умови виставлених додатком часових обмежень виконання обчислень стає задача оптимізації процесу прискорення реконфігурації, за рахунок оптимізації обсягу зменшення часових витрат, що дозволить мінімізувати непродуктивне використання обчислювального простору реконфігурованої області настільки, щоб забезпечити виставлені додатком часові вимоги. Ця задача не вирішується відомими методами та засобами прискорення реконфігурації, що обумовлює актуальність та доцільність виконаних в статті досліджень.

Огляд існуючих рішень

Відомо багато методів та засобів зменшення непродуктивних видатків часу реконфігурації, такі як повторне використання обчислювальних ресурсів [3, 5, 6], попередня вибірка конфігурацій [7], кластерізація конфігурацій [2], реконфігурація за принципом «як можна раніш» [7], різноманітні апаратні засоби прискорення процедури обміну даними та збільшення швидкодії інтерфейсів [8]. Всі вони забезпечують максимально можливе прискорення обчислень, за рахунок максимально зменшення накладних видатків часу реконфігурації, частіше за все ціною надлишкового використання як апаратних ресурсів, так і надлишкових витрат проективності і енергоспоживання, що призводить до додаткових накладних видатків реконфігурації. Одне з найбільш ефективних із відомих рішень прискорення реконфігурації, що спрямовано на використання необхідного об'єму реконфігурованого обчислювального ресурсу, запропоноване для кластероподібної обчислювальної системи із багатоядерними процесорами в вузлах і розділеної між ними однорідної реконфігурованої структури [4, 9]. Але задіяна реконфігурована структура має передумовлену архітектуру і складається з набору однотипних дрібнозернистих і крупнозернистих модулів, поєднаних між собою загальною комунікаційною мережею. Основна ідея розподілу задач складається з визначення ефективної апаратної структури для реалізації задачі, що дозволить забезпечити необхідний час виконання вирішуваного додатку це дозволяє позбавитись від збитковості використання апаратури забезпечуючи загальні її обмеження. Негативні риси даного підходу складаються в наявності передумовленої архітектури, що вимагає попереднього при-

ведення будь якої вхідної задачі до визначної форми, та реалізації спеціалізованого програмного прошарку із залучанням вузькопрофільних спеціалістів для їх підтримки та експлуатації. Проблематика зменшення накладних видатків не розглядається, фактично задача вирішується абстраговано від фізичного процесу реконфігурації і зводиться до класичної концепції апаратного прискорення.

На підставі огляду відомих рішень проблеми прискорення реконфігурації в сучасних реконфігурованих обчислювальних системах визначено необхідність вдосконалення відомих, або розробки нових, методів та засобів прискорення реконфігурації, які забезпечують необхідний додатку час обчислення та мінімізуючи при цьому витрати обчислювального ресурсу ПЛІС.

Постановка задачі

В якості критерію оцінки ефективності функціонування реконфігурованої обчислювальної системи приймають наступний показник прискорення:

$$\rho = \frac{T_{SW}}{T_{Rconf} + T_{HW}} \qquad (1)$$

де T_{SW} – час обчислення задачі на процесорному ядрі, T_{HW} – час обчислення задачі апаратними засобами, T_{Rconf} – час реконфігурації, витрачений на розміщення конфігурації відповідної апаратної реалізації задачі на поверхні реконфігурованої обчислювальної структури. Тоді проблема прискорення реконфігурації може бути вирішена за рахунок зменшення часу реконфігурації, що формально виражається наступним чином:

$$\min(T_{Rconf}) = \min(T_{Contr}) + \min(T_{Comm}) + T_{Mapp}. \qquad (2)$$

Час T_{Rconf} в виразах (1) і (2) характеризує обсяг непродуктивних видатків часу реконфігурації і складається з наступних складових, відповідно виразу (2): тривалості циклу управління процесом реконфігурації $T_{control}$ – обумовлює витрати продуктивності; часу передавання конфігураційних даних T_{comm} – обумовлює комунікаційні витрати; часу прошивання мікросхеми T_{mapp} – залежить від технологічних особливостей реконфігурованої елементної бази. На підставі визначення (2) в роботі [10] запропонований метод інтенсивного прискорення реконфігурації (МПР), який забезпечує мінімізацію непродуктивних видатків часу реконфігурації ($T_{Rconf} \to \min$), за рахунок максимального видалення непродуктивних комунікаційних витрат ($T_{comm} \to \min$) шляхом повторного використання апаратних ресурсів функціональних блоків. Ефективність запропонованого методу МПР залежить від просторових параметрів реконфігурованої області. За невідповідності розміру реконфігурованої області і кількості розміщуваних апаратних задач процес реконфігурації потребує додаткового часу, який негативно впливає на загальний час обчислень і часто призводить до зворотного ефекту від прискорення реконфігурації.

В реальних реконфігурованих обчислювальних системах стає задача оптимізації реконфігурації, яка може бути вирішена за рахунок оптимізації обсягу видалення непродуктивних витрат часу, що дозволить мінімізувати використання обчислювальних ресурсів ПЛІС, забезпечуючи затребувані вирішуваним додатком обмеження часу виконання.

Проблема оптимізації ре конфігурації

Геометричне обґрунтування оптимізації реконфігурації за критеріями забезпечення обмежень часу обчислення і просторових обмежень реконфігурованої області ПЛІС зображено на рис. 1.

Непродуктивні видатки часу на здійснення реконфігурації визначаються складовою T_{Rconf} виразу (1). Коефіцієнт реконфігурації K_{Rconf} характеризує обсяг виконаної реконфігурації з ціллю збільшення степені адаптації обчислювальної структури до вимог вирішуваної задачі. Збільшення коефіцієнта реконфігурації призводить до прискорення обчислень за рахунок зменшення складової часу T_{HW} виразу (1) ($T_{HW} \to \min$). Але прискорення за рахунок збільшення степені адаптації обчислювальної структури часто призводить до зворотного ефекту з причини високих часових витрат на здійснення реконфігурації ($T_{Rconf} \to \max$).В цьому контексті на діаграмі визначена область оптимізації реконфігурації (I) з точки зору забезпечення збільшення показника прискорення відповідно виразу (1). Застосування механізмів інтенсивного прискорення реконфігурації збільшує область оптимізації реконфігурації (II), дозволяючи реалізувати більшу степінь адаптації архітектури та прискорити обчислення.

Криві T_{Count} і $T_{Count}^{\text{МПР}}$ зображують зміну загального часу обчислення відповідно за стандартної послідовності реконфігурації та засто-

сування механізмів інтенсивного прискорення реконфігурації в межах області оптимізації реконфігурації.

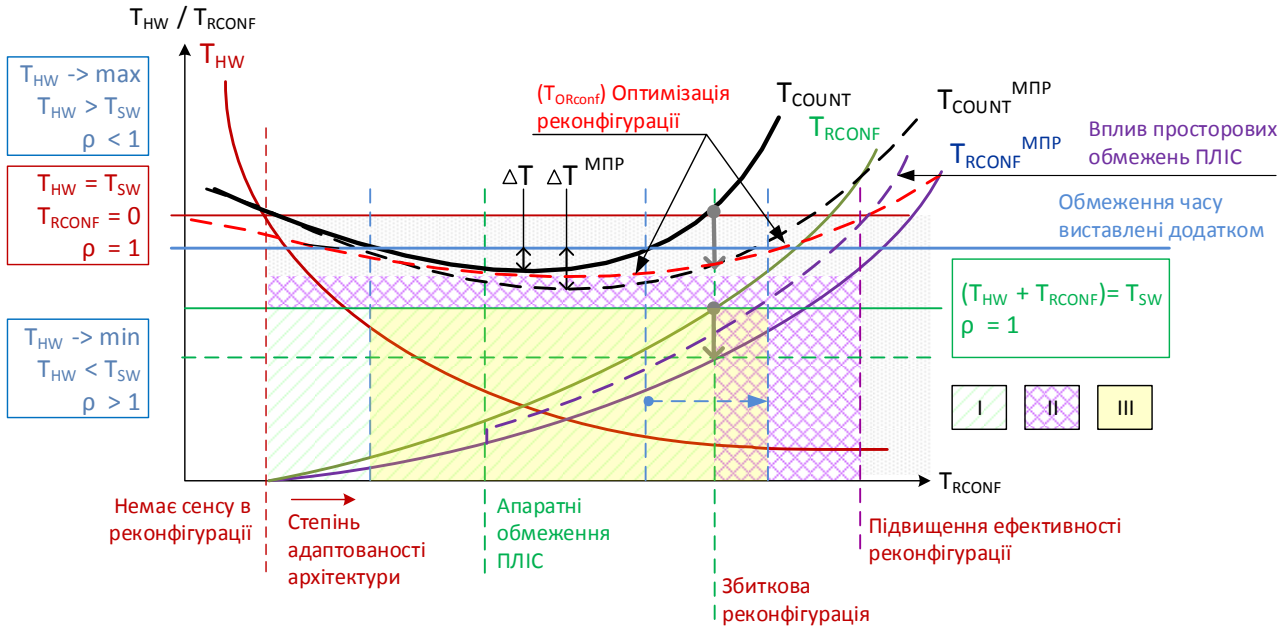


Рис. 1. Визначення критеріїв оптимізації реконфігурації

Прискорення обчислень природньо обмежується недостачею апаратних ресурсів ПЛІС, як для реалізації найбільш адаптованої до структури задачі конфігурації на поверхні кристалу так і для досягнення максимального прискорення реконфігурації. Подолання обмежень апаратних ресурсів ПЛІС потребує додаткових витрат часу, що ще більше збільшує накладні видатки реконфігурації ($T_{Rconf} \to \max$). Апаратні обмеження ПЛІС зменшують ефективність механізмів інтенсивного прискорення реконфігурації та , як зображено на діаграмі (рис. 1), звужують область оптимізації реконфігурації (II).

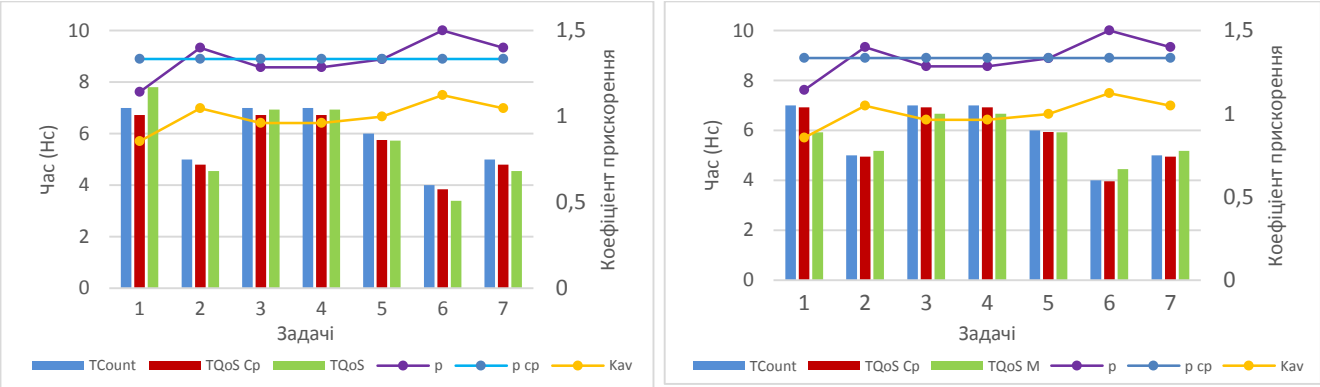
Область оптимізації реконфігурації за критерієм виставлених додатком часових обмежень (III) також розширюється за рахунок механізмів прискорення реконфігурації, що, з іншого боку, обмежується недостачею апаратних ресурсів ПЛІС. При цьому за наближення параметрів реконфігурації до точки перетину графіків T_{Rconf} та $T_{Rconf}^{МПР}$ є деякий проміжок часу ($\Delta T^{МПР}$), який дозволяє вкластися в обмеження часу виставлені додатком без використання, або часткового використання, механізмів прискорення реконфігурації. На підставі цього висновку_в роботі запропоновано метод оптимізації процесу реконфігурації, що забезпечує видалення оптимального обсягу непродуктивних витрат часу, зменшуючи час реконфігурації настільки, щоб забезпечити при цьому мінімальне використання просторових ресурсів ПЛІС і задовольнити вимоги додатку до максимального часу виконання обчислень (крива T_{ORconf} на рис. 1).

Модифікований спосіб розподілу часу

Для реалізації методу оптимізації реконфігурації запропонований модифікований спосіб розподілу часу, який застосовується для розподілу заданого проміжку часу виконання додатку між послідовністю виконуваних макрозадач [11], що є складовими даного додатку. На відміну від відомого способу, заданий проміжок часу виконання розподіляється між послідовністю задач обернено пропорційно часу їх очікуваного виконання. Це призводить до виділення задачам, коефіцієнт прискорення яких більше ніж середній, додаткового часу виконання за рахунок задач, прискорення яких нижче ніж середнє. Таким чином, запропонований спосіб дозволяє визначити критичні до часу виконання задачі, що потребують максимального прискорення реконфігурації. Задачі з високим коефіцієнтом прискорення ефективно вирішуються на наявному устаткуванні засобами стандартної послідовності процесу реконфігурації.

Виконано теоретичні дослідження ефекту від застосування запропонованого модифікованого способу розподілу часу. Досліджений додаток є гіпотетичною послідовністю задач, за часові параметри яких прийняті часові характеристики

синтезованих на ПЛІС Cyclone II Altera арифметичних прискорювачів. На діаграмах (рис. 2) видно, що обернено пропорційна корекція T_{QoS_Mod} усереднених проміжків часу T_{QoS_Av} надає задачам, шо мають високий коефіцієнт прискорення додаткового часового ресурсу. При цьому задачі, коефіцієнт яких наближається до середнього, також отримають додаткового часу для виконання.



а *б*

Рис. 2. Модифікований спосіб розподілу часу: а – пропорційна корекція (відомий спосіб), б – обернено пропорційна корекція усереднених проміжків часу обмеження

Математичний запис запропонованого модифікованого способу розподілу часу представлений виразом (3), де T_{QoS} – виставлені додатком обмеження часу виконання, $T_{QoS_M\,i}$ – обмеження часу, розраховані для кожної *i*-ї задачі, $T_{Count\,i}$ – очікуваного час виконання *i*-ї задачі, ρ_i – прискорення продуктивності, розраховане за формулою (1), усереднене прискорення продуктивності ρ_{Av}, $i=\overline{1,n}$, де *n* – кількість задач в послідовності *B*, що надходить на виконання. На відміну від відомого способу в вираз (3) введена корекція похибки усереднення, що дозволило підвищити точність розрахунків.

$$T_{QoS_Mod\,i} = T_i \times \left[\frac{T_{QoS}}{\sum_{i=1}^{n} T_{C\,Count}} \times \frac{\sum_{i=1}^{n} T_{SW\,i}}{\sum_{i=1}^{n} T_{Count} \times \rho_{Av}} \right] \times \frac{\rho_i}{\rho_{Av}}. \quad (3)$$

Виконання часових обмежень в загальному випадку визначається наступним виразом:

$$T_{Count\,i} \le T_{QoS\,i}, \quad (4)$$

Таким чином, модифікований спосіб розподілу часу складається в наступному:

1. Заданий час обмеження виконання розподіляється на частини обернено пропорційні часу виконання кожної задачі згідно виразу (3) у наступній послідовності:

– виконується корекція часу обмеження до середнього значення,

– виконується розподіл середнього часу обмеження пропорційно середньому значенню прискорення продуктивності – отримання усереднених проміжків часу обмеження виконання кожної макрозадачі,

– корегуються усереднені проміжки часу згідно зворотного напрямку коефіцієнта усереднення кожної задачі.

2. Отримані проміжки часу аналізуються згідно загальної ознаки виконання обмежень (4).

На підставі (3) та (4) визначені наступні умови застосування способу розподілу часу та сформовані наступні межі його застосування, за яких раціонально використання запропонованого методу:

$$\begin{cases} \rho_{Cp} > 1 \\ \rho_{\max} > \rho_{QoS} > 1 \\ \rho_{\max} \to \rho_{Cp},\ \rho_{\min} \to \rho_{Cp} \end{cases}, \quad (5)$$

де ρ_{QoSk} – прискорення, що вимагає накладені часові обмеження (шукане прискорення), $\rho_{\max}, \rho_{\min}$ – граничні обмеження шуканого прискорення, в межах яких раціонально оптимізація реконфігурації.

Випадки, коли усереднене значення прискорення обчислювального процесу (яке є вихідним параметром для обчислення) менше за одиницю ($\rho_{Cp} < 1$), тобто коли фактично від-

бувається уповільнення загального обчислювального процесу, не розглядаються. Такі алгоритми можуть бути виконані засобами процесорних ядер. Ме має сенсу розглядати випадки, коли жодна задача не вкладається в часові обмеження, якщо $\rho_{\max} < \rho_{QoS} > \rho_{\Sigma \mathrm{Cp}}$, або всі задачі вкладаються в обмеження, коли $\rho_{\min} > \rho_{QoS} > 1$. Значення $\rho_{\max}, \rho_{\min}$ розраховуються на підставі показника середньоквадратичного відхилення δ, який є мірою розкидання величин від їх середніх значень. Задачі, у яких прискорення продуктивності наближаються до значення $\pm 3\delta$, значним чином впливають на рівномірність розподілу часу обмеження, що з практичного боку реалізації призведе до надлишкових дії щодо мінімізації часу виконання задачі, і, як слідство, до збиткового використання апаратних та програмно-апаратних ресурсів системи. За цима межами застосування оптимізації реконфігурації недоцільно.

Метод оптимізації реконфігурації

Вихідний обчислюваний додаток є паралельною програмою зі змішаним типом паралелізму, для опису якого застосовується модель програмування М-задач [11]. Відповідно до цього програма (М-програма) задається макрографом потоку даних *MDG* (*Macro Dataflow Graphs*) в вершинах якого розміщуються макрозадачі (М-задачі), а ребра вказують на залежності між вершинами. Тоді, представимо вихідну програму макрографом $G_M = (V_M, E)$, де V_M – множина вершин, що відповідають макрозадачам, а E – множина ребер, що визначають відношення між макрозадачами. Зважаючи на архітектурні особливості реконфігурованої обчислювальної системи, розповсюдженим методом відображення задач на архітектуру є відображення рівнями. При цьому граф програми подається у ярусно-паралельній формі (ЯПФ) і кожний рівень підграфу послідовно відображується на архітектуру обчислювальної системи [11]. Час виконання обчислюваної програми визначається, як сума часу виконання послідовності найтриваліших задач кожного рівня, і описується виразом:

$$T_{G_{M_ЯПФ}} = \sum_{k=1}^{w} \max \{T_{v_k} \big| v_k = \overline{1, H_k}\},$$

де $k = \overline{1, w}$ – номер ярусу, w – кількість ярусів обчислювального алгоритму, $v_k = \overline{1, H_k}$ – номер вузла на ярусі k, H_k – кількість вузлів на ярусі k. Відповідно до цього наступна упорядкована множина макрозадач:

$$B_{G_M} = \{T_{\max_k} \big| k = \overline{1, w}\}, \qquad (6)$$

де $T_{\max_k}$ – час виконання найтривалішої задачі k-го ярусу, складає найтривалішу взаємозв'язану послідовність задач, яка є вихідною для визначення загального часу виконання програми.

Під обмеженнями часу виконання розуміють обмеження часу встановлені на підставі певних зовнішніх факторів, наприклад, вихідних часових вимог виконуваного додатку.

Метод оптимізації процесу реконфігурації для динамічно реконфігурованих обчислювальних систем складається в наступному:

Етап 1

Визначення доцільної межі забезпечення часовим умовам:

$$\begin{cases} \rho_{\mathrm{Cp}} > 1 \\ \rho_{\max} > \rho_{QoS} > 1 \end{cases}.$$

Етап 2

Видалення із послідовності задач B (6), множини задач D ($T_i \in D$, якщо $T_{Rconf\,i} = \max$) час яких неможливо зменшити, зокрема задачі першого рівня ЯПФ [11]:

$$T_{v_1} \in D, \text{ якщо } T_{v_1} = \max\{T_{v_1} \big| v_1 = \overline{1, H_1}\},.$$

тоді

$$T_{QoS} = T_{QoS} - \sum_{i=1}^{n} (T_i \big| \forall T_i \in D).$$

Етап 3

Забезпечення виконання умови

$$\rho_{\max} \to \rho_{QoS}.$$

Етап 4

Розрахунок часу обмеження для задач, що залишилися в послідовності, на підставі (3):

$$T_{\mathrm{QoS}\,i} = T_i \times \left[\frac{T_{QoS}}{\sum_{i=1}^{n} T_{Count\,i}} \times \frac{\sum_{i=1}^{n} T^{\uparrow}_{SW\,i}}{\sum_{i=1}^{n} T_{\mathrm{Count}} \times \rho_{Cp}} \right] \times \frac{\rho_i}{\rho_{Cp}} \Bigg| \forall (T_i \notin D).$$

Етап 5. Визначення концепції зменшення часу виконання задач, що не вкладаються в часові обмеження, шляхом застосування методів інтенсивного прискорення:

$$\rho_i \left| \forall (T_{Count\,i} < T_{QoS\,i}) = \frac{T_{SW\,i}}{T_{Count\,i}} = \frac{T_{SW\,i}}{T_{Rconf\,i} + T_{HW\,i}} \right. \Rightarrow \begin{cases} T_{Rconf\,i} \to \min \\ T_{Count\,i} \to T_{HW\,i} \end{cases}.$$

Для зменшення часу реконфігурації використовується запропоновані авторами робіт [10, 12] метод та засоби інтенсивного прискорення реконфігурації.

Моделювання методу оптимізації реконфігурації

Розроблений емулятор реконфігурованої обчислювальної системи (РОС) та програмна модель реалізації прискорення реконфігурації. Достовірність функціонування програмної моделі забезпечена шляхом застосування наближених до реальних часових характеристик функціональних елементів РОС, що отримані на базі розроблених моделей складових обчислювального модуля та функціональних блоків апаратних задач. Блоки апаратних задачі синтезовані на мові Verilog та реалізовані на ПЛІС Cyclone II Altera.

Дослідження проводились для серії додатків, поданих графами алгоритмів в ЯПФ. Досліджувались графи алгоритмів з різною кількістю однотипних задач. На підставі проведених експериментів отримані залежності часу реконфігурації від місця розташування конфігурацій них даних й від кількості виконуваних задач одного типу (рис. 3), та показник прискорення реконфігурації для алгоритмів з різною кількістю наборів однотипних задач (рис. 4). Виконані порівняння процесів інтенсивного оптимального прискорення реконфігурації.

Реконфігуровані обчислення виконані засобами стандартної послідовності реконфігурації потребують значного часу, як зображено на рис. 3, у зв'язку зі значною перевагою часу реконфігурації над часом виконання апаратної задачі, і не залежать від типів виконуваних задач. Метод прискорення реконфігурації за рахунок видалення всіх непродуктивних комунікаційних витрат часу забезпечує інтенсивне прискорення реконфігурації за умови відповідності структури задачі до структури реконфігурованого середовища, тобто коли ширина графу ЯПФ алгоритму відповідає розміру ПЛІС [11]. Інакше відбуваються витіснення апаратних задач в локальну пам'ять і завантаження затребуваних алгоритмом конфігурацій. Це потребує додаткового часу, але у порівнянні зі стандартною послідовністю реконфігурації МПР забезпечує загальне прискорення виконання обчислювального алгоритму.

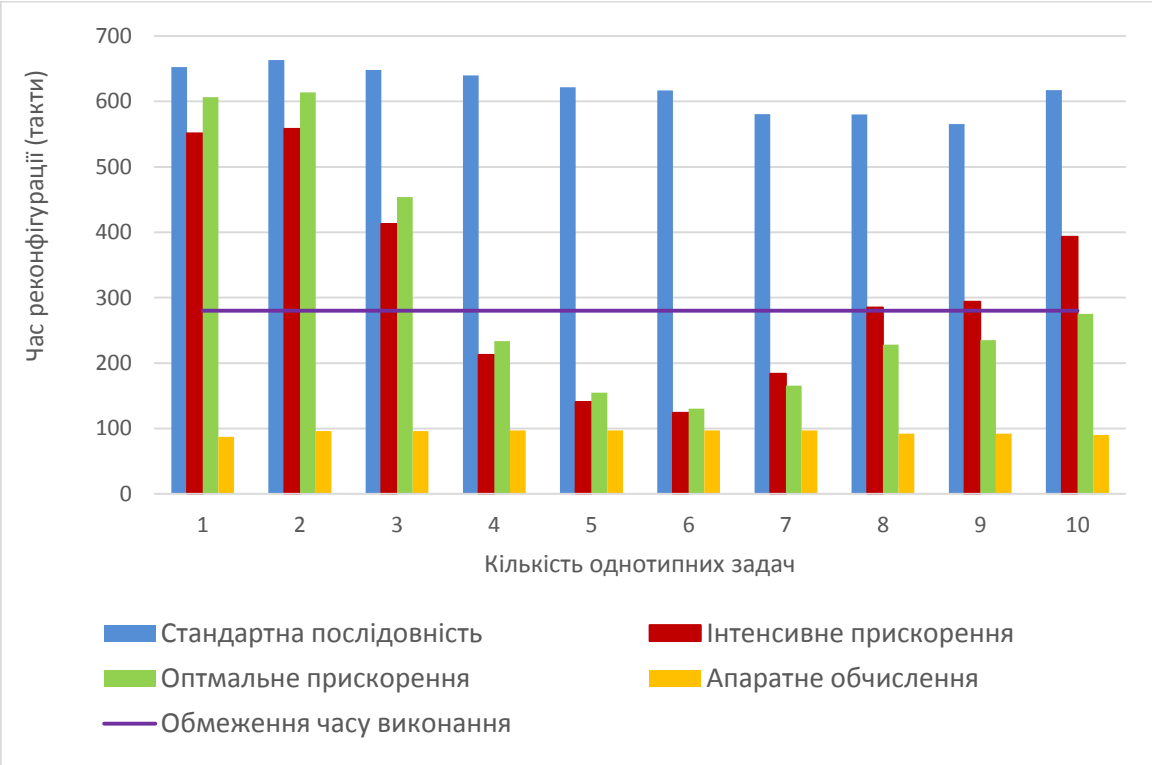


Рис. 3. Дослідження часу реконфігурації

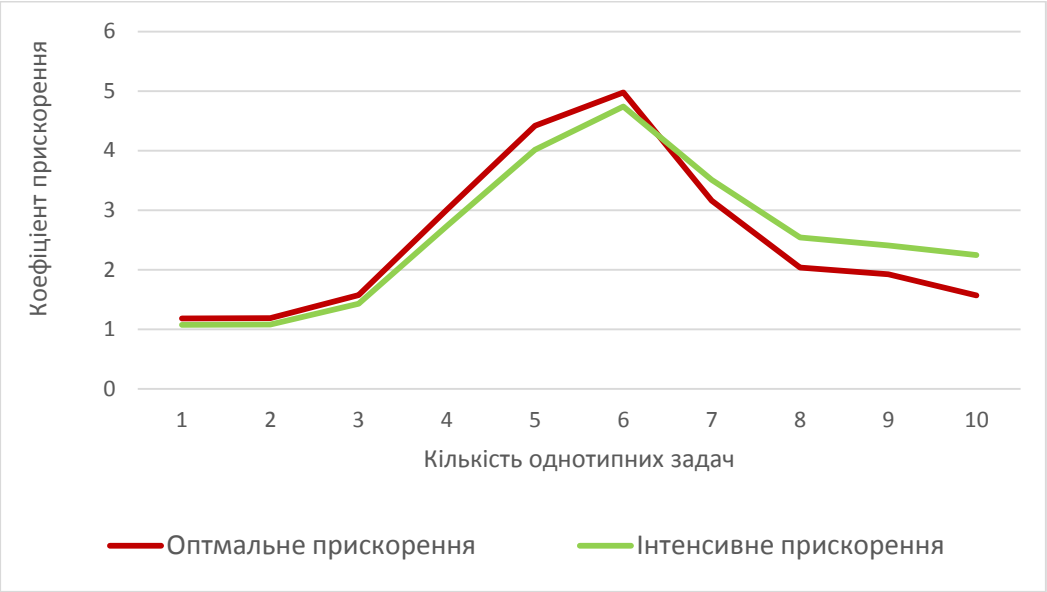


Рис. 4. Дослідження показників прискорення

За результатами експериментів (рис. 3, рис. 4) отримано, що механізм повторного використання ресурсів, на якому базується МПР, надає інтенсивне прискорення реконфігурації зі збільшенням типів задач до кількості сумірної з шириною графа ЯПФ. За сумірної з розміром реконфігурованої області ширини графа ЯПФ та високої кількості однотипних задач досягається збільшення інтенсивності прискорення реконфігурації в середньому на 63%. При цьому в процесі подолання просторових обмежень ПЛІС відбувається різке зменшення інтенсивності прискорення реконфігурації в середньому на 85%.

Метод оптимізації реконфігурації зменшує інтенсивність впливу просторових обмежень на швидкість реконфігурації приблизно на 10% у порівнянні з методом інтенсивного прискорення. З діаграми (рис. 4) видно, що більша кількість задач потрапила в межі часових обмежень. Метод оптимізації реконфігурації доцільно також застосовувати для зменшення кількості відмов виконання задач.

Висновки

Вперше запропоновано та обґрунтовано метод оптимізації процесу реконфігурації, що шляхом визначення та видалення оптимального обсягу накладних витрат часу, за критеріями забезпечення часових обмежень, виставлених вирішуваним додатком та забезпечення апаратних обмежень ресурсів ПЛІС під час розподілу динамічного потоку задач, дозволяє підвищити швидкодію реконфігурованих обчислень та зменшити кількість відмов виконання задач.

Модифіковано спосіб розподілу обмежень часу виконання між задачами обчислювального алгоритму, що дозволило забезпечити його використання в межах вирішення проблеми оптимізації реконфігурації за рахунок визначення задач, що не потребують залучання механізмів інтенсивного прискорення реконфігурації, а також підвищити точність розрахунків, у порівнянні з відомим способом.

Запропонований метод оптимізації реконфігурації дозволяє забезпечити широкий клас задач найбільш ефективною цільовою структурою для досягнення необхідної продуктивності обчислень, а також зменшити кількість відмов під час розподілу потоку динамічно надходжуваних задач.

Список посилань

1. El-Araby E. Exploiting Partial Runtime Reconfiguration for High-Performance Reconfigurable Computing / E. El-Araby, I. Gonzalez, T. El-Ghazawi // ACM Transactions on Reconfigurable

Technology and Systems (TRETS). - USA, NY, New York, ACM, 2009. – Vol. 1, Issue 4, Article № 21.

2. Huang M. Reconfiguration and Communication-Aware Task Scheduling for High-Performance Reconfigurable Computing / M. Huang, V.K. Narayana, H. Simmler, O. Serres, T. El-Ghazawi // Transactions on Reconfigurable Technology and Systems (TRETS). – USA, NY, New York, ACM, 2010. – Vol. 3, Issue 4, Article № 20.
3. Bassiri M.M. Mitigating Reconfiguration Overhead In On-Line Task Scheduling For Reconfigurable Computing Systems / M.M. Bassiri, S.H. Shahriar // Proceeding of the 2nd International Conference on Computer Engineering and Technology (ICCET), (China, Chengdu, 16 – 18 April 2010). – IEEE, 2010. – Vol. 4. – P. V4-397 – V4 – 402.
4. Ahmed W. Adaptive Resource Management for Simultaneous Multitasking in Mixed-Grained Reconfigurable Multi-core Processors /W. Ahmed, M. Shafique, L. Bauer, J. Henkel // Proceedings of the 9th International Conference on Hardware/Software Codesign and System Synthesis (CODES+ISSS) (Taiwan, Taipei, 9-14 October 2011). – IEEE, 2011. – P. 365 - 374.
5. Panella A. A design workflow for dynamically reconfigurable multi-FPGA systems / A. Panella, M.D. Santambrogio, F.Redaelli [and all] // Proceeding of the 18th VLSI System on Chip Conference (VLSI-SoC), (Spain, Madrid, 27 – 29 September, 2010). – IEEE/IFIP, 2010. – P. 414 - 419.
6. Al-Wattar A. Efficient On-line Hardware/Software Task Scheduling for Dynamic Run-time Reconfigurable Systems / A. Al-Wattar, S. Areibi, F. Saffih // Proceeding of the 26th International Parallel and Distributed Processing Symposium Workshops & PhD Forum (IPDPSW) (China, Shanghai, 21 – 25 May 2012). – IEEE, 2012. – P 401 - 406.
7. Taher M. Virtual Configuration Management: A Technique for Partial Runtime Reconfiguration / M. Taher, T. El-Ghazawi // IEEE Transactions on Computers. – IEEE, 2009. – Vol. 58, Issue: 10. – P. 1398 - 1410.
8. Liu S. Achieving Energy Efficiency through Runtime Partial Reconfiguration on Reconfigurable Systems / S. Liu, R.N. Pittman, A. Forin, J.-L. Gaudiot // Transactions on Embedded Computing Systems (TECS). – USA, NY, New York, ACM, 2013. – Volume 12, Issue 3, Article № 72. – 21 p.
9. Ahmed W. mRTS:Run-Time System for Reconfigurable Processors with Multi-Grained Instruction-Set Extensions / W. Ahmed, M. Shafique, L. Bauer, J. Henkel // Proceedings of the Design, Automation & Test in Europe Conference & Exhibition (DATE) (France, Grenoble, 14 – 18 March 2011). – IEEE, 2011. – P. 1 - 6.
10. Кулаков Ю.О. Розробка методу прискорення реконфігурації в динамічно реконфігурованих обчислювальних системах Ю. О. Кулаков, І. А. Клименко, М. В. Рудницький // Восточно-европейский журнал передовых технологий. – 2015. – Том 4, № 4(76) (2015). – С. 25 – 30.
11. Кулаков Ю.О. Метод оптимізації ярусно-паралельної форми подання задачі для реконфігурованих обчислювальних систем / Ю.О. Кулаков, І.А. Клименко // Електроніка та зв’язок. – К: НТТУ «КПІ», 2014. – Том 19, №4(81). – С. 90 - 96.
12. Кулаков Ю.О. Організація багаторівневої пам’яті в реконфігурованих обчислювальних системах // Ю.О. Кулаков, І.А. Клименко / Вісник НТУУ «КПІ». Інформатика, управління та обчислювальна технік: Зб. Наук. Пр. – К.: Вєк+, 2014. - №61. – С. 18 - 26.

УДК 621.391

ДОРОГИЙ Я.Ю.,
ДОРОГА-ІВАНЮК О.О.,
ФЕРЕНС Д.А.

РЕАЛІЗАЦІЯ АЛГОРИТМУ СТРУКТУРНОЇ ОПТИМІЗАЦІЇ НЕЙРОННОЇ МЕРЕЖІ

В статті розглянуте питання реалізації алгоритму структурного оптимізації нейронної мережі та її застосування для задач розпізнавання даних.

In the article the question of implementation of algorithm for neural network architecture optimization and its application for recognition tasks were considered.

Введення в проблему

Апарат нейронних мереж широко використовується для розв'язання різноманітних задач, в тому числі і для задач розпізнавання. Наявність методу автоматичного пошуку оптимальної структури нейронної мережі дала б можливість досліднику швидше отримати структуру нейронної мережі, яка б найкраще відповідали предметній області та наявним вхідним даним.

Аналіз існуючих рішень

Для побудови оптимальної структури нейронної мережі використовується досить широке коло алгоритмів. Першим з таких алгоритмів є алгоритм черепичної побудови [1]. Ідея алгоритму полягає в додаванні нових шарів нейронів таким чином, щоб вхідні навчальні вектори, які мають різні відповідні вихідні значення, мали б в ньому різне внутрішнє представлення. Ще одним яскравим представником є алгоритм швидкої надбудови [2]. Нові нейрони за цим алгоритмом додаються між вихідними шарами. Роль цих нейронів – корегування помилки вихідних нейронів. В загальному вигляді нейронна мережа, що побудована за таким алгоритмом, має форму бінарного дерева. Широко відомими є алгоритми Monoplan, NetLines та NetSphere [3], метод редукції [4]. Але всі ці алгоритми мають достатньо широкий перелік недоліків, тому було вирішено розробити свій алгоритм. В роботі [6] запропоновано алгоритм, реалізацію якого для задач розпізнавання і буде розглянуто в цій роботі.

Мета роботи

Метою даної роботи є аналіз алгоритму структурної оптимізації нейронної мережі в ході її навчання для задач розпізнавання образів та імплементація даного алгоритму програмними засобами.

Алгоритм структурної оптимізації при навчанні

Алгоритм структурного навчання використовується на багатошарових мережах прямого поширення та має ітеративний характер: на кожній ітерації виконується пошук структури мережі кращої за попередню. Пошук мережі виконується шляхом перебору усіх можливих мутацій мережі, вибору та комбінації кращих (селекція та схрещення).

Розглянемо основні параметри алгоритму.

Параметри навчання:

- швидкість навчання, η;
- коефіцієнт інерції, μ;
- коефіцієнт затухання ваг, ε;
- вірогідність активації нейрону прихованого шару, p_h;
- вірогідність активації нейрону вхідного шару, p_i.

Параметри структурного навчання:

- початкова кількість нейронів прихованого шару;
- функція активації прихованого шару;
- функція активації та функція ціни вихідного шару;
- максимальна кількість мутацій при схрещенні;
- кількість епох навчання початкової мережі;
- кількість епох навчання у ітерації;
- види допустимих мутацій;
- частина навчальної вибірки, що використовується для навчання.

Елементарні структурні операції над нейронною мережею

Згідно з [5] введено наступні елементарні структурні операції над мережею:

- додавання синапсу між двома випадково вибраними незв'язаними вузлами або нейронами мережі – операція Syn_{ADD};
- видалення синапсу між двома випадково вибраними незв'язаними вузлами або нейронами мережі – операція Syn_{DEL};
- переміщення синапсу між двома випадково вибраними незв'язаними вузлами або нейронами мережі – операція Syn_{MOD};
- зміна функції активації нейрона для випадково вибраного нейрона – операція A_{MOD};
- серіалізація вузла або нейрона – операції - Ser_{NODE} і Ser_{NR};
- паралелізація вузла або нейрона – операції Par_{NODE} і Par_{NR};
- додавання вузла або нейрона – операції Add_{NODE} і Add_{NR};
- створення нового шару – операція L_{ADD};
- видалення шару НМ – операція L_{DEL}.

Використання чи невикористання наведених структурних операцій залежить від складності поставленої задачі.

Для задач розпізнавання, що будуть наведені в цій статті, використані операції (мутації), описані в [7].

Імплементація алгоритму

Внутрішньо нейронні мережі представлені у вигляді послідовності числових матриць ваг кожного шару окрім вхідного. На рис. 1 зображена послідовність матриць для мережі типу [2-3-2]: матриця 2×3 для прихованого шару та 3×2 для вихідного.

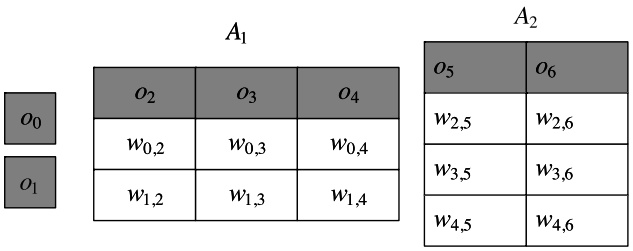


Рис.1. Приклад внутрішньої реалізації для мережі [2-3-2]

Кожен елемент a_{ij} матриці A_k дорівнює значенню ваги між i-тим та j-тим нейронами мережі.

Для реалізації різних видів мутацій використовуються операції над матрицями. При додаванні нових нейронів до шару виконується комбінація із операцій додавання нового рядка та стовпця матриці. На рис. 2, 3 та 4 зображена реалізація додавання нейрону до вхідного, прихованого та вихідного шарів.

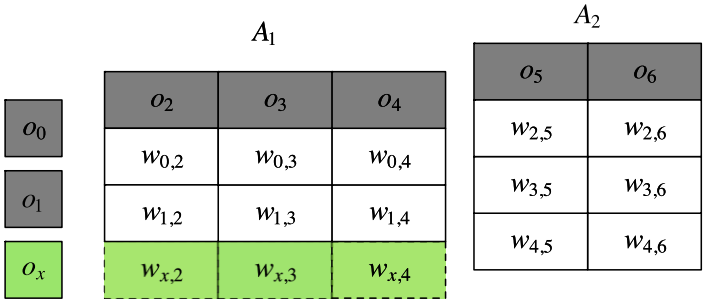


Рис.2. Додавання нейрону до вхідного шару

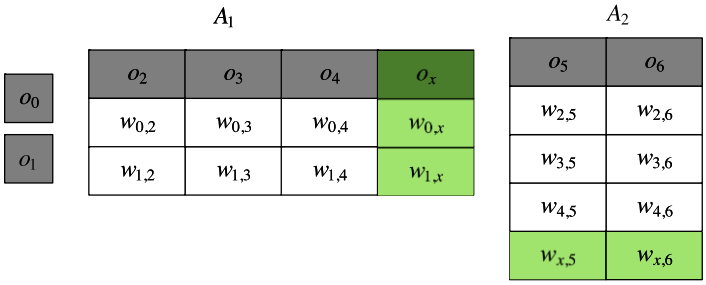


Рис.3. Додавання нейрону до прихованого шару

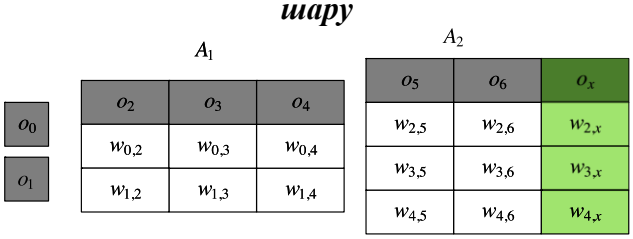


Рис.4. Додавання нейрону до вихідного шару

При видаленні нейронів використовуються протилежні операції. На рис. 5 зображена реалізація видалення 2-го нейрону прихованого шару.

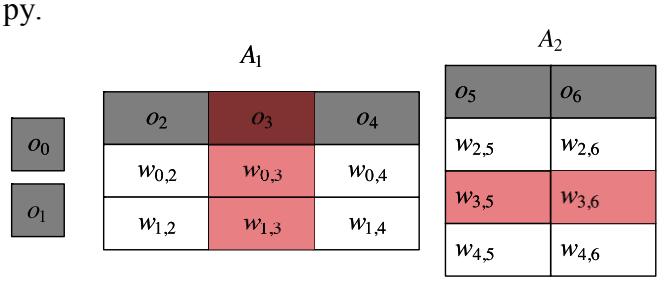


Рис.5. Видалення нейрону прихованого шару

При додаванні нового шару виконується операція вставки нової матриці ваг у послідовність матриць.

Оскільки певні операції змінюють структури матриць, існує певна складність при їх комбі-

нації. Наприклад, при видаленні нейрону O_3 прихованого шару у мережі [2-3-2] у результуючій мережі нейрон O_4 зміститься на 1 позицію та стане нейроном O_3; при додаванні нового прихованого шару, що містить 4 нейрони перед існуючим прихованим шаром, наступний шар зміститься на 1 позицію. При комбінації різних мутацій їх покрокове виконання необхідно здійснювати у чіткому порядку, що залежить від виду та параметрів кожної мутації. У лістингу 1 нижче наведений фрагмент коду, який правильно виконує комбіновану мутацію. Першим чином виконуються мутації, що не змінюють структури – додавання та видалення зв'язків, згодом виконуються додавання нових нейронів та видалення існуючих; наприкінці виконуються додавання шарів. Мутації, що виконують видалення нейронів, виконуються у порядку зменшення номеру нейрону, аналогічно додавання шарів – у порядку зменшення індексу нового шару.

```
(defmethod mutate ::combined
 [net {:keys [mutations]}]
 (let [grouped-ms (group-by :operation mutations)
      {add-node-ms ::add-node del-node-ms ::del-node
       layer-ms ::add-layer} grouped-ms
      safe-ms (mapcat grouped-ms [::identity ::add-edge ::del-
edge])
      safe-del-node-ms (reverse
                  (sort-by #(second (:deleted-node %)) del-
node-ms))
      safe-layer-ms (reverse
                 (sort-by :layer-pos layer-ms))
      ms (concat safe-ms add-node-ms safe-del-node-ms safe-
layer-ms)]
  (reduce mutate net ms)))
```

Лістинг 1. Фрагмент коду що здійснює комбіновану мутацію

Однією із переваг Clojure над іншими мовами програмування є використання незмінних (immutable) структур даних – колекцій та контейнерів, вміст яких неможливо змінити. Натомість, при спробі додати новий елемент до колекції буде створений новий об'єкт колекції, що містить цей елемент. Операція створення нової колекції оптимізована: обидва об'єкти будуть використовувати частину колекції, що є спільною. На рис. 6 зображений результат додавання об'єкту 5 у кінець масиву [1, 2, 3, 4].

Програмування з використанням незмінних структур даних значно спрощує розуміння програм, а також має наступні переваги:

- простота паралелізації програми – незмінні дані можуть використовуватись паралельно без необхідності синхронізації потоків;
- немає проблем з витоками пам'яті (memory leak);
- простота кешування;
- значна економія пам'яті в деяких випадках.

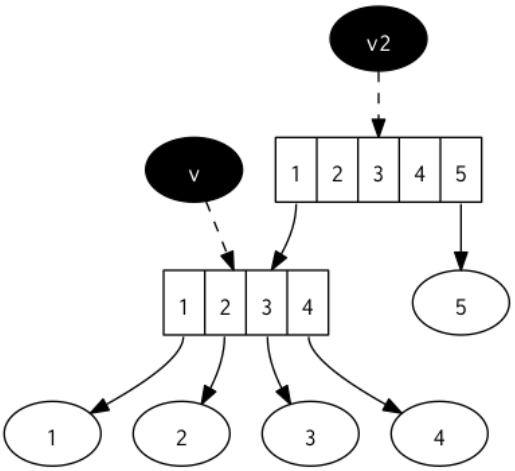


Рис.6. Принцип роботи структур даних у Clojure

Саме завдяки цим властивостям незмінних структур значна частина роботи алгоритму виконується паралельно, максимально використовуючи обчислювальні ресурси.

Розроблена система має архітектуру типу "клієнт-сервер". Діаграма розгортання системи наведена на рис. 7. Загалом, система складається з двох частин:

1. серверний додаток, що виконує навчання нейронної мережі та реалізує алгоритм структурної оптимізації;
2. клієнтський додаток, що реалізує графічний інтерфейс користувача.

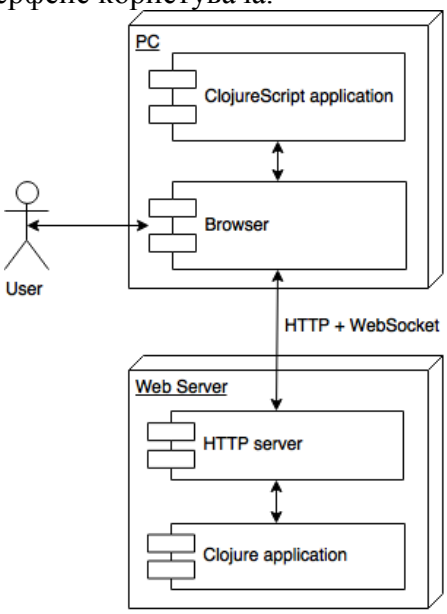


Рис.7. Діаграма розгортання системи

Для реалізації серверного додатку була використана мова програмування Clojure – функціональна мова загального призначення. В якості середовища виконання була вибрана платформа Java.

Для реалізації графічного інтерфейсу була використана мова програмування ClojureScript – діалект Clojure, що використовується для виконання у середовищі JavaScript.

Експериментальні дослідження

Реалізована програмна система використана для дослідження задачі розпізнавання облич людини. В якості вихідних даних була взята база зображень облич Єльського університету [**Ошибка! Источник ссылки не найден.**].

Формування вибірки. Було вибрано 10 різних осіб та відібрано 50 випадкових зображень кожної особи. Кожне зображення було масштабовано до розміру 26 на 26 пікселів та закодовано у вигляді 676-мірного вектору, значення яскравості пікселів були нормалізовані до діапазону 0..1. Номер кожної особи був закодований вектором розміру 10, що містить 9 нулів та одиницю в позиції номеру персони. Отримані 500 прикладів були випадково поділені на навчальну та тестувальну вибірки у відношенні 2:1.

На рис. 8 зображені вихідні зображення та зображення, що використовуються для навчання нейронної мережі.

Рис.8. Приклад формування набору даних

Архітектура початкової мережі. Для оцінки роботи алгоритму була використана мережа, архітектура якої зображена на рис. 9.

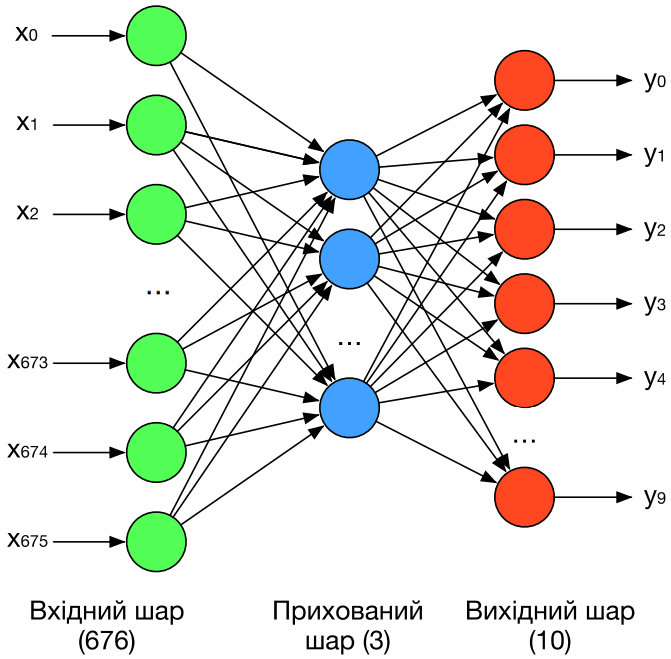


Рис.9. Архітектура мережі для розпізнавання зображень

Дослідження роботи алгоритму. Були використані наступні значення параметрів навчання:

- швидкість навчання: $\eta = 0.002$;
- коефіцієнт інерції: $\mu = 0.1$;
- коефіцієнт затухання ваг: $\varepsilon = 0.1$;
- вірогідність активації нейрону прихованого шару: $p_h = 1$;
- вірогідність активації нейрону вхідного шару: $p_i = 1$.

Параметри алгоритму вибрані наступні:

- початкова кількість нейронів прихованого шару: 3;
- функція активації прихованого шару: ReLU;
- функція активації та функція ціни вихідного шару: softmax;
- максимальна кількість мутацій при схрещенні: $M = 50$;
- кількість епох навчання початкової мережі: $T_0 = 100$;
- кількість епох навчання у ітерації: $T_i = 5$;
- види допустимих мутацій: додавання та видалення синапсів;
- частина навчальної вибірки що використовується для навчання: 1.

Протягом 40 ітерацій алгоритму було здійснено 300 видалень та 128 додавань синапсів. На

рис. 10 та 11 зображені залежності значення ціни та точності класифікації від кількості виконаних епох навчання, отримана точність розпізнавання наведена в табл.1.

Табл.1. Результуюча точність класифікації зображень для $T_i = 5$

Тип НМ	Навчання, %	Тестування, %
Звичайна	97.59	93.41
Оптимізована	98.19	95.80

Завдяки оптимізації структури зв'язків вдалось знизити відсоток помилкової класифікації з 6.6% до 4.2% на тестувальній вибірці.

Також, був проведений експеримент, при якому прийнято $T_i = 3$, зображено на рис. 12 та 13. Протягом 100 ітерацій алгоритму було здійснено 645 видалень та 457 додавань синапсів. Відсоток помилкового розпізнавання вдалось знизити з 7.8% до 6.0%. на тестувальній вибірці. Результати класифікації наведені в табл.2.

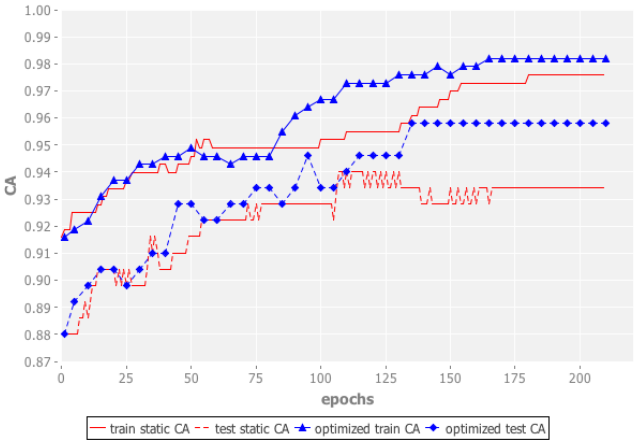


Рис.10. Точність класифікації зображень для $T_i = 5$

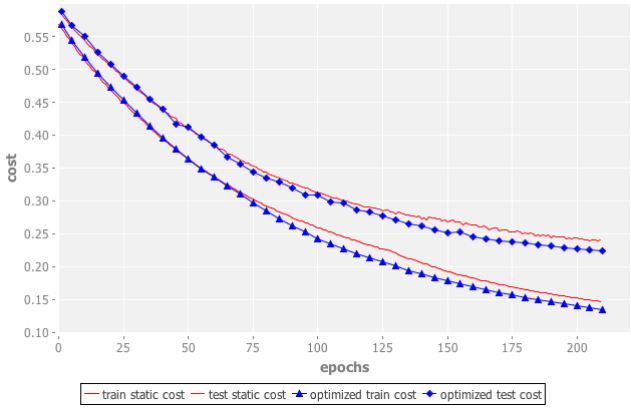


Рис.11.Значення ціни при для класифікації зображень для $T_i = 5$

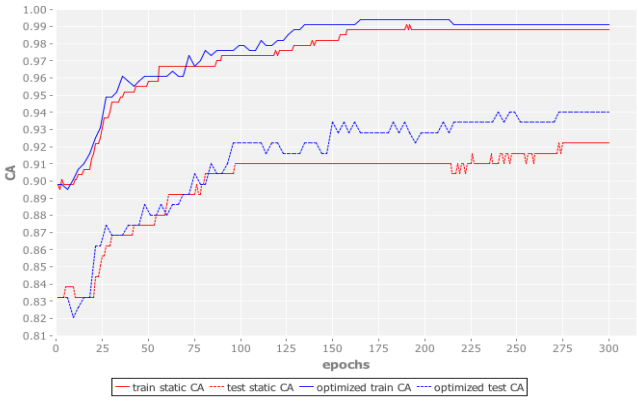


Рис.12. Точність класифікації зображень для $T_i = 3$

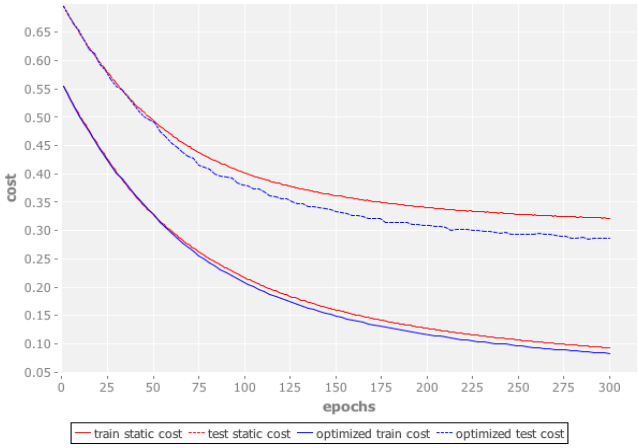


Рис.13. Значення ціни при для класифікації зображень для $T_i = 3$

Табл.2. Результуюча точність класифікації зображень для $T_i = 3$

Тип НМ	Навчання, %	Тестування, %
Звичайна	98.79	92.21
Оптимізована	99.09	94.01

Висновки

В роботі розглянуті питання імплементації алгоритму структурної оптимізації, який запропонований в [6], а також проаналізована можливість використання даного алгоритму для задач розпізнавання образів..

Список посилань

1. Mezard M., and Nadal J.P. Learning in feedforward layered networks: The Tiling algorithm // Journal of Physics. – 1989. – V. A22. – P. 2191 - 2203,

2. Frean M. The Upstart Algorithm: A Method for Constructing and Training Feed-Forward Neural Networks // Tech. Rep. 89/469, Edinburgh Univ., 1989.
3. Ash T. Dynamic Node Creation in Back-Propagation Networks // Connection Science. – 1989. – V. 1.
4. Mozer M.C., Smolensky P. Skeletonization: a technique for trimming the fat from a network via relevance assessment // Advances in Neural Information Processing Systems. – 1989. – V. 1. – P. 107 - 115.
5. Дорогий Я.Ю. Ускоренный алгоритм обучения сверточных нейронных сетей / Я.Ю. Дорогий // Вісник НТУУ «КПІ», «Інформатика, управління та обчислювальна техніка», №57. – 2012. – С. 150-154.
6. Дорогий Я.Ю. Алгоритм структурної оптимізації нейронної мережі / Я.Ю.Дорогий // Вісник НТУУ «КПІ», «Інформатика, управління та обчислювальна техніка», №61. – 2014. – С. 169-173.
7. Дорогий Я.Ю. Застосування алгоритму структурної оптимізації нейронної мережі в задачах класифікації даних / Я.Ю.Дорогий, В.В.Цуркан, О.О.Дорога-Іванюк, Д.А.Ференс// Вісник НТУУ «КПІ», «Інформатика, управління та обчислювальна техніка», №62. – 2015. – С. 169-173.

УДК 621

ПОРЄВ В.М.

ЕВРИСТИЧНІ АЛГОРИТМИ РАСТРОВОГО МОДЕЛЮВАННЯ ДИНАМІКИ ПРОЦЕСІВ РОЗПОВСЮДЖЕННЯ ДЛЯ ГЕОІНФОРМАЦІЙНИХ СИСТЕМ

В статті пропонуються евристичні алгоритми для моделювання деяких процесів розповсюдження на основі метода кінцевих елементів. Виконаний аналіз запропонованих алгоритмів, а також наведені відомості щодо їхньої корисності для геоінформаційних систем.

The article proposed heuristic algorithms to model some processes distribution based on the finite element method. The analysis of the proposed algorithms, and provides information on their usefulness for GIS.

1. Вступ

Ця стаття присвячена моделюванню процесів розповсюдження у просторі. Поняття розповсюдження загалом може позначати перенос, розсіювання, дифузію тощо. Роботи по моделюванню таких процесів виконуються вже протягом багатьох десятиріч в усьому світі, проте поява та масове сучасне поширення комп'ютерних геоінформаційних систем (ГІС) надає додатковий поштовх робіт у цьому напрямку.

Геоінформаційні системи орієнтовані на аналіз просторових даних, накопичених у базах різноманітних відомостей про об'єкти, розташовані на Землі. Такі системи можуть ефективно допомагати у вирішення проблем, як глобальних, так і в межах окремих невеличких територій. Наприклад, моделювання в ГІС розповсюдження небезпечних речовин внаслідок аварії необхідно для адекватного реагування та планування ефективних дій по мінімізації на певній території наслідків надзвичайної ситуації. Інший приклад – моделювання розповсюдження інфекційних захворювань. Можна сказати, що обидва наведені приклади об'єднує поняття процесу розповсюдження у просторі. Просторовий аналіз об'єктів, процесів та явищ є одним з головних аспектів геоінформаційних систем, задля цього розроблені відповідні інструментальні засоби ГІС. Одним з інструментів є підсистема растрового аналізу. Растровий аналіз є одним з різновидів метода кінцевих елементів. Починаючи з деякого порогу, стає доцільним використання метода кінцевих елементів для вирішення тих задач, які раніше вирішувалися виключно розв'язуванням систем диференціальних рівнянь. Використання метода кінцевих елементів дає змогу моделювати складні процеси простими засобами на основі ланцюжків елементарних подій, які відповідають взаємодії сусідніх елементів простору.

2. Аналіз досліджень та публікацій

Темі моделювання процесів розповсюдження присвячено достатньо багато публікацій. Одним з найпопулярніших у цій галузі є напрямок моделювання розповсюдження забруднень [1, 2, 3]. Відомі методики опису та розрахунків параметрів розповсюдження – концентрацій забруднень та розмірів зон забруднень [4, 5, 6, 7]. Наприклад, у стандартній методиці РД 52.04.253-90 затверджені розрахунки розмірів та форми зони небезпечного забруднення після вибуху, причому форма зони залежить від швидкості вітру і окреслюється як коло, напівколо або круговий сектор [4]. У методиці ОНД-86 надані оціночні формули на основі нормального розподілу, зони розповсюдження окреслюються еліпсами [5].

Можна вказати, що опис та відображення зон розповсюдження забруднень у вигляді кругів, секторів або еліпсів є достатньо примітивним. Наприклад, при змінах напрямку та швидкості вітру геометрія зони розповсюдження забруднень стає достатньо складною й зовсім некоректно відображати її сектором або еліпсом. Це є проблемою для використання офіційно затверджених методик розрахунків [4, 5]. Таким чином, потрібно вдосконалювати методики моделювання і, відповідно, шукати або створювати інструменти, які здатні допомогти у цьому.

Що стосується ГІС, то відомі декілька публікацій, які присвячені використанню ГІС для відображення зон забруднення, наприклад [8]. Можна сказати, що ГІС використовується переважно як засіб відображення зон (ізоліній) забруднення фіксованої геометрії та форми, а

розрахунки розмірів зон виконується деяким іншим аналітичним модулем, у якому закладені оціночні формули згідно визначених методик. Те, що ГІС відображатиме зони розповсюдження на електронній карті, принципово нічого не змінює порівняно з відображенням на папері і без ГІС. Проте ГІС здатні на більше. Наявність у ГІС (наприклад, закордонної ArcGIS [9], вітчизняної ГІС "ОКО" [10] і деяких інших) розвинутих інструментальних засобів растрового моделювання надає можливості моделювати самі процеси, замість простого відображення зразків-шаблонів у вигляді деяких заздалегідь розрахованих геометричних форм. Необхідно використовувати відповідні методи та алгоритми, а якщо потрібно – адаптувати їх.

Одним з перспективних напрямків моделювання процесів розповсюдження є використання методу кінцевих елементів, який полягає у тому, що замість безперервного простору аналізується дискретний простір у вигляді множини елементарних елементів об'єму, площини тощо. У цьому напрямку треба згадати клітинні автомати, запропоновані ще фон Нейманом [11]. Клітинний автомат функціонує на основі закладеного простого алгоритму, якій аналізує поточний стан елементу простору та обмеженої кількості сусідніх елементів, і відповідно змінює стан. Незважаючі на те, що у взаємодію вступають тільки сусідні клітини (у фон Неймана це найближчі 4 сусіда на площині), навіть достатньо прості правила функціонування автоматів породжують складні структури. Використанню клітинних автоматів для моделювання процесів переносу, дифузії тощо, присвячено багато публікацій, зокрема [12, 13, 14, 15].

3. Постановка задачі

Потрібно побудувати алгоритми моделювання процесів розповсюдження з урахуванням можливостей підсистеми растрового аналізу ГІС.

Проаналізувати основні характеристики та результати використання растрової моделі розповсюдження на основі метода кінцевих елементів.

4. Виклад основного матеріалу

Розповсюдження можна представити як рух часток чогось у просторі, наприклад, часток речовини від одних елементів дискретного об'єму до інших. Які саме частки та у яких напрямках рухаються – це основні характеристики такої моделі. Потрібно визначити взаємодію між елементами простору. У роботі [16] було введено поняття кубу та матриці розсіювання.

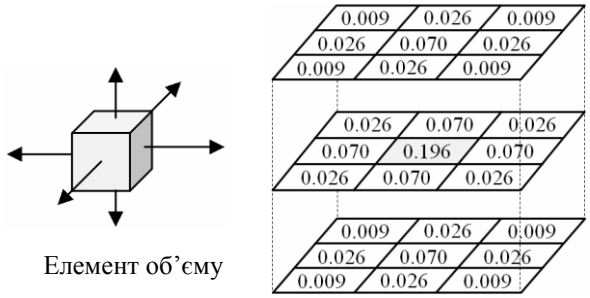


Рис. 1. Куб розсіювання 3×3×3

Куб розсіювання означає взаємодію центрального елементу об'єму із 26 сусідами. Для випадку моделі на площині у центрального елементу є 8 сусідів.

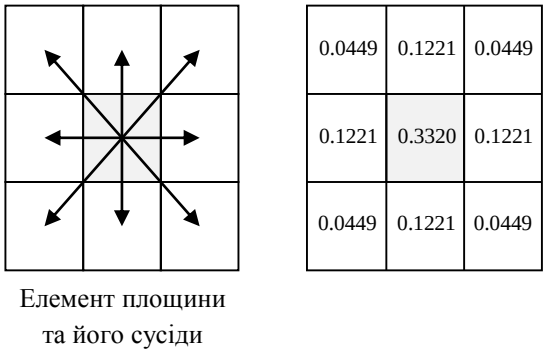


Рис. 2. Матриця розсіювання 3×3

Коефіцієнти кубу та матриці розсіювання відображають кількість речовини, яку буде перенесено у відповідні сусідні елементи за одиницю часу. Позначимо через $F_{3d}(x, y, z)$ розподіл значень елементів в кубі розсіювання де x, y, z – координати елементів відносно його центру. Для матриці розсіювання це буде $F_{2d}(x, y)$. Набір значень коефіцієнтів повністю визначає модель розповсюдження для одного акту взаємодії елементів впродовж певної ітерації моделювання. Однією з вимог може бути те, що сума коефіцієнтів кубу або матриці розсіювання повинна дорівнювати 1, якщо речовина нікуди не зникає, або сама не створюється (проте, така вимога не є обов'язковою для усіх моделей).

Моделювання виконується за декілька ітерацій. Кожна ітерація означає перенос речовини у сусідні елементи дискретного простору за один квант часу. Простір 3d або 2d відобра-

жається числовим масивом – відповідно трьохвимірним або двовимірним (матрицею). Для виконання ітерації потрібні два масиви – вихідний масив ***A*** та допоміжний масив ***B***. Спочатку масив ***B*** обнулюється. Впродовж одної ітерації скануються усі елементи масиву-джерела, і кожний елемент цього масиву множиться на коефіцієнти кубу (матриці) розсіювання – отримуються 27 значень для об'єму (або 9 для площини), які додаються у відповідні елементи масиву результату. Це можна описати так:

B = *0*
B = *Розповсюдження(**A**)* //1-*а ітерація*
A = *0*
A = *Розповсюдження(**B**)* //2-*а ітерація*
B = *0*
B = *Розповсюдження(**A**)* //3-*а ітерація*

і так далі, виконується *I* ітерацій. Залежно від того, парна чи непарна кількість ітерацій, кінцевий результат моделювання буде або у масиві ***A***, або у масиві ***B***.

Розміри масивів визначаються межами простору та роздільною здатністю моделі. Розміри кубу (матриці) розсіювання зовсім необов'язково повинні бути 3×3×3 (або 3×3 для матриці). Значення коефіцієнтів кубу або матриці визначаються характером моделі переносу речовин.

4.1. Розсіювання рівномірне по усім напрямкам

У цьому випадку для визначення коефіцієнтів матриці або кубу розсіювання треба використовувати симетричну відносно усіх напрямків функцію $F(d)$, де d – відстань до центру матриці або кубу розсіювання. Приклади таких функцій: $F(d) = c/(1+a\cdot d^p)$, $F(d) = c\cdot exp(-a\cdot d^2)$, де a, c та p – параметри-константи.

Після виконання декількох ітерацій моделювання отримаємо растр, кожна точка якого буде зберігати якесь значення. Функцію розподілу значень у тривимірному растрі (моделі об'єму) позначимо як $P(x,y,z)$, а для площини – як $P(x,y)$. У найпростішому випадку, коли моделюється рівномірне по усім напрямкам розсіювання від точкового джерела, функція $P(x,y,z)$ має бути $P(d)$, де d – відстань до центру розсіювання.

Для вивчення закономірностей розподілів $P(x,y,z)$ та $P(x,y)$ було виконано серію дослідів для різноманітних кубів та матриць розсіювання при кількості ітерацій від 1 до 100. Розміри кубів та матриць варіювалися від 3×3 до 21×21. У роботі [16] вже були викладені деякі результати таких досліджень. Одним з нетривіальних, на думку автора, результатів є те, що після вже невеличкої кількості ітерацій моделювання розсіювання – починаючи з 4-5 ітерацій, розподіл $P(d)$ стає дуже схожим на нормальний (гаусовий), і з кожною наступною ітерацією наближається до ідеального вигляду функції $P(d) = exp(-d^2)$. Більше того, загальний характер такого розподілу спостерігається майже при будь-якому співвідношенні коефіцієнтів симетричного кубу $F_{3d}(x, y, z)$ або матриці розсіювання $F_{2d}(x, y)$ – важливо тільки, щоб виконувалася умова: $F(d_2) \le F(d_1)$, якщо $d_2 \ge d_1$. Таке спостерігається навіть у випадку однакових коефіцієнтів: $F(d) = const$.

Оскільки з кожною ітерацією має місце наближення $P(d) \rightarrow exp(-d^2)$, то доцільно зробити так, щоб гаусовий розподіл був би і на початкових ітераціях. Задля цього можна використовувати куби (матриці) розсіювання, коефіцієнти яких визначаються формулою $F(d) = c\cdot exp(-a\cdot d^2)$.

Враховуючи наведене вище, апроксимуємо $P(d)$ наступним чином:

$$P(d) = K\times G(d), \tag{1}$$

де: K – коефіцієнт, $G(d)$ – функція Гаусса (нормального розподілу)

$$G(d) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{d^2}{2\sigma^2}} \tag{2}$$

Потрібно обчислити параметри апроксимації K та σ. Це можна виконати обробкою результатів моделювання методом найменших квадратів для значень $P(d_i)$. А далі виконати оцінку точності такої апроксимації. Нижче наведені деякі результати такої обробки.

Матриця 3×3. Сімейство графіків $\sigma(I)$, $K(I)$ та похибки $E(I)$ для значень параметра a від 0.1 до 2.0 при кількості ітерацій I = 1..100.

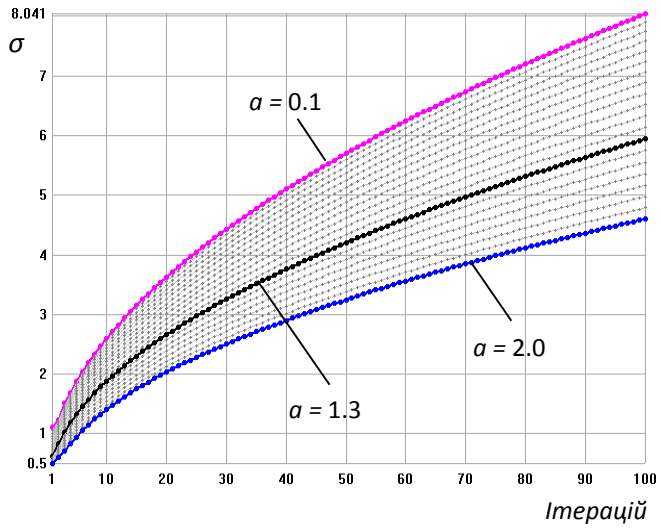


Рис. 3. Залежність σ від кількості ітерацій для матриці 3×3

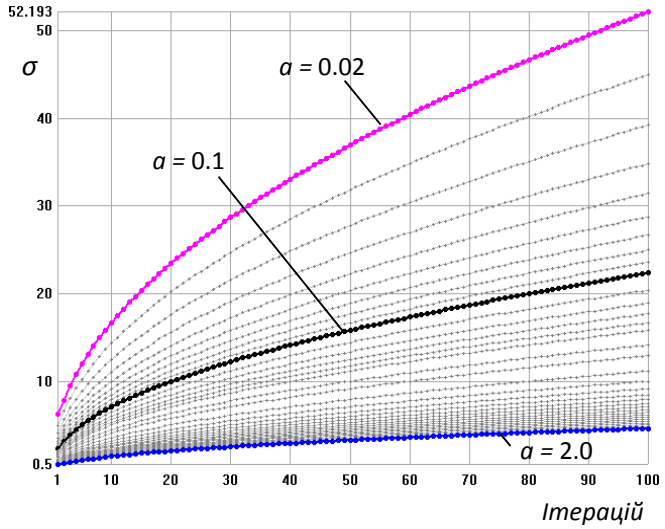


Рис. 6. Залежність σ від кількості ітерацій для матриці 21×21

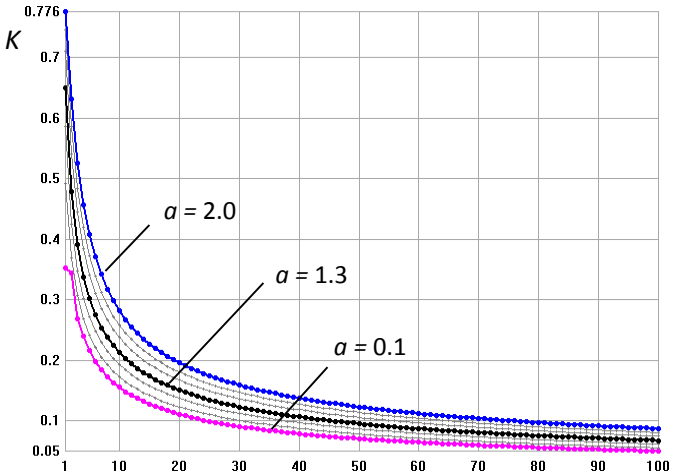


Рис. 4. Залежність K від кількості ітерацій для матриці 3×3

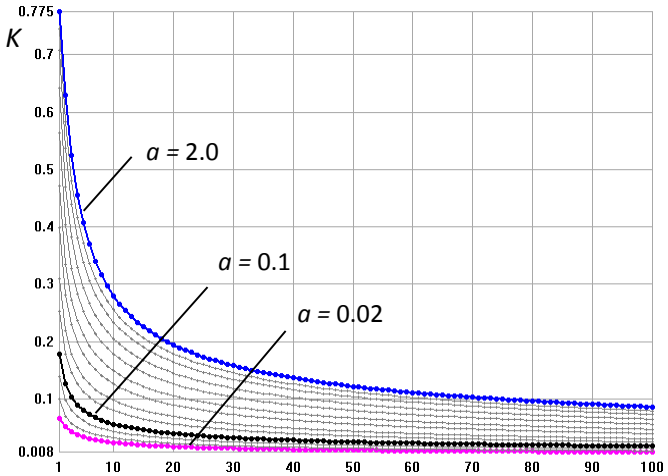


Рис. 7. Залежність K від кількості ітерацій для матриці 21×21

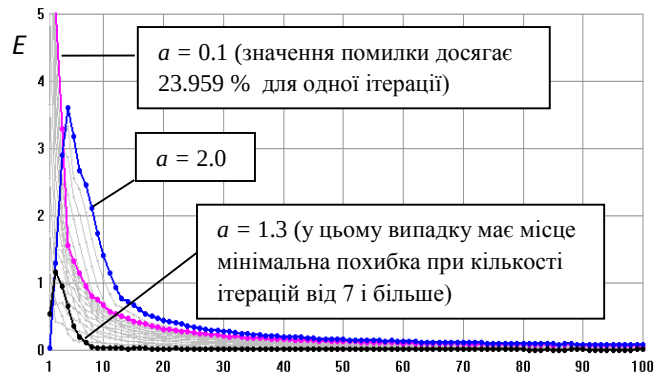


Рис. 5. Залежність відносної помилки E (%) від кількості ітерацій для матриці 3×3

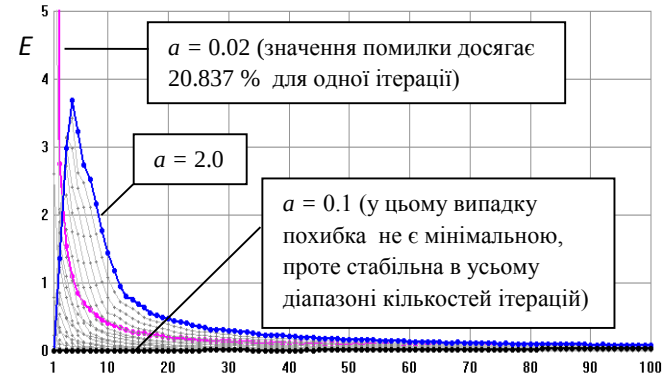


Рис. 8. Залежність відносної помилки E (%) від кількості ітерацій для матриці 21×21

Матриця 21×21. Сімейство графіків *σ(I)*, *K(I)* та похибки *E(I)* для значень параметра *a* від 0.02 до 2.0 при кількості ітерацій *I* = 1..100.

Далі проілюстровано залежність *σ(I)* для різних розмірів матриць *N×N* при фіксованих значеннях параметру *a*

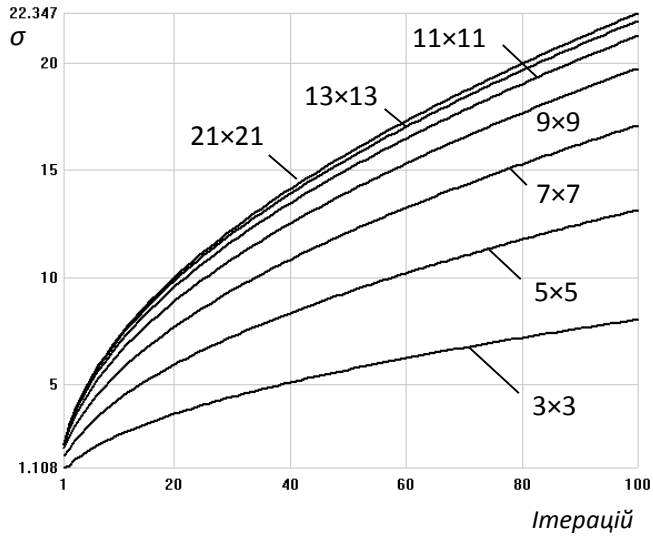


Рис. 9. Сімейство кривих σ(I) при a=0.1

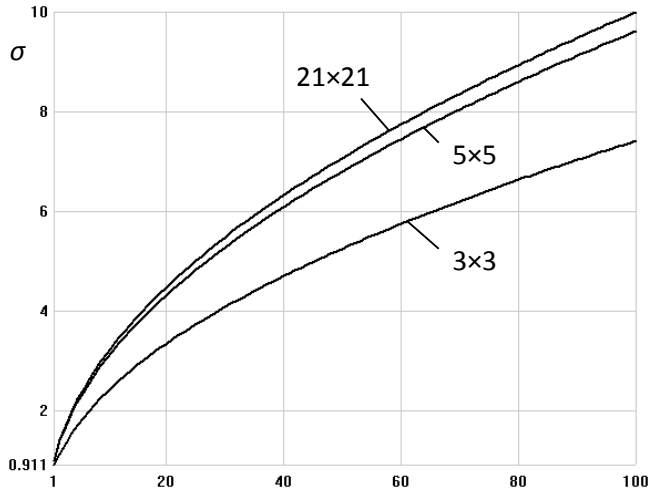


Рис. 10. Сімейство кривих σ(I) при a=1.0

Розгляд графіків на рис. 10 призводить до висновку, що при *a* = 1.0 недоцільно використовувати матриці розсіювання розмірами більше 5×5.

Для зручності вибору оптимальних параметрів моделі потрібно деяке узагальнення отриманих результатів. Для кубів та матриць розсіювання залежність основних параметрів (*σ* та *K*) можна записати як

σ = Функція(N, a, I)

K = Функція(N, a, I),

де:

N – розмір матриці *N×N* або кубу *N×N×N;*

a – параметр ініціалізації матриці*;*

I – кількість ітерацій моделювання

При фіксованому значенні параметра *a* значення сигми оцінюється як корінь квадратний від кількості ітерацій $\sigma \sim I^{0.5}$.

4.2. Калібрування моделі

Нехай відомо, наприклад, у результаті обробки вимірів на місцевості, що розподіл концентрації забруднення має вигляд нормального закону (гаусова модель):

$$P(x) = \frac{K}{\sigma\sqrt{2\pi}} e^{-\frac{x^2}{2\sigma^2}}, \qquad (3)$$

де: *x* – відстань у метрах до центра джерела забруднення, *K* – константа, яка означає загальний рівень забруднення. Результатом обробки отриманих даних будуть числові значення *σ* та *K* (якщо гаусова модель адекватна ситуації).

Тепер потрібно знайти відповідність значень натуральних значенням модельним. Як встановлено вище, результат растрового моделювання розсіювання можна з певною точністю також представити функцією Гаусса:

$$P(d) = \frac{K_{моделі}}{\sigma_{моделі}\sqrt{2\pi}} e^{-\frac{d^2}{2\sigma^2_{моделі}}}, \qquad (4)$$

де: $K_{моделі}$ та $\sigma_{моделі}$ – параметри моделі, які обумовлюються матрицею розсіювання та кількістю ітерацій, *d* – відстань у дискретному просторі растрової моделі.

Якщо прийняти, що розмір квадратних (кубічних) елементів растру становить *m* метрів, тоді *d* елементів по осі виміру будуть репрезентувати відстань у метрах: *x = m·d*. Підставимо *d = x/m* у (4) і помножимо вираз для *P(d)* на масштабний коефіцієнт *M*

$$P_{масшт}(x) = M\frac{K_{моделі}}{\sigma_{моделі}\sqrt{2\pi}} e^{-\frac{x^2}{2m^2\sigma^2_{моделі}}} \qquad (5)$$

Значення масштабованої растрової моделі повинні дорівнювати натуральним значенням

$$P(x) = P_{масшт}(x)$$

Для цього масштабний коефіцієнт має бути

$$M = \frac{K}{K_{моделі}\cdot m} \qquad (6)$$

Підставимо це у (5) й отримаємо

$$P_{масшт}(x) = \frac{K}{(m\sigma_{моделі})\sqrt{2\pi}} e^{-\frac{x^2}{2(m\sigma_{моделі})^2}},$$

Повна тотожність $P_{масшт}(x) \equiv P(x)$ буде, якщо

$$m\cdot\sigma_{моделі} = \sigma \qquad (7)$$

Це означає, що зменшення роздільної здатності моделі (тобто, збільшення m), призводить до пропорційного збільшення σ для натуральних значень. Якщо задане σ для натуральних значень, то $\sigma_{моделі}$ потрібно вибирати згідно співвідношення

$$\sigma_{моделі} = \sigma / m \qquad (8)$$

Наприклад, у джерелі [2] вказується, що розподіл концентрації забруднення навколо деякого автошляху по відстані від осі в метрах оцінюється нормальним законом при $\sigma = 60$. Якщо моделювати це при роздільній здатності растру $m = 10$ метрів то потрібно забезпечити $\sigma_{моделі} = 60/m = 6$. Відповідну кількість ітерацій та коефіцієнт $K_{моделі}$ можна знайти з графіків, наведених вище. Наприклад, для матриці 3×3 згідно рис. 3 результат $\sigma_{моделі} \approx 6$ досягається при таких параметрах: (a=0.1, I=56), або (a=0.2, I=58), або (a=0.3, I=60), і так далі до (a=1.2, I=96). Який набір параметрів вибрати? Найменший час моделювання – 56 ітерацій при $a = 0.1$. Оцінка похибки – 0.114%. Якщо вибрати варіант $a = 1.2$ при 96 ітераціях, то похибка буде 0.018%. А якщо вибрати матрицю розсіювання розмірами 7×7, то потрібного результату можна досягти при a=0.5 за 37 ітерацій з похибкою 0.008%. І так далі. Оптимальний вибір варіанту згідно якогось критерію повинна виконувати ГІС.

Можна поставити таке запитання: а навіщо виконувати растрове моделювання для рівномірного розсіювання? Може краще безпосередньо обчислювати результати згідно гаусової моделі? Дійсно, це справедливо, якщо є тільки одне точкове джерело – центр розсіювання. Растрове моделювання спрощує отримання результату, якщо точкових джерел багато. А коли потрібно виконувати моделювання розсіювання від просторово розподілених джерел, наприклад, множини лінійних (автодороги) або площинних (палаючі ліси або торфовища) джерел, то, можливо, тільки растрове моделювання принципово вирішує таке завдання.

4.3. Моделювання вітру

Вітер загалом можна представити як впорядкований рух часток чогось у певному напрямку. Основними параметрами будуть напрямок та швидкість.

Найпростішу дискретну растрову модель руху на площині можна описати матрицею 3×3

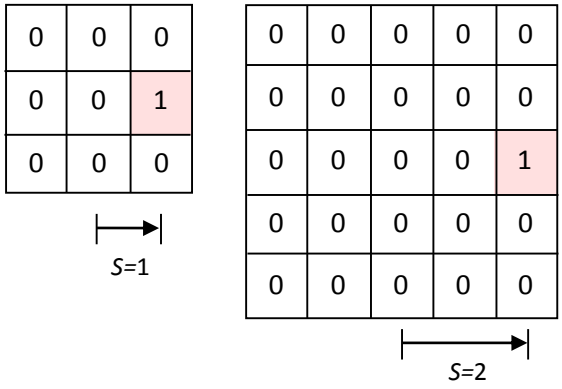


Рис. 11. Матриці зсуву 3×3 та 5×5

Якщо позначити через S відстань одиниці від центру матриці, то кожна ітерація моделювання з використанням таких матриць буде означати зсув вмісту усіх елементів растру на S дискретних позицій праворуч (на картах – це на схід). Якщо роздільна здатність растрової моделі m метрів на елемент площини (або об'єму), тоді буде моделюватися швидкість $V = m{\cdot}S$ метрів за ітерацію. При виконанні I ітерацій буде зсув на $D = I{\cdot}S$ дискретних позицій, що відповідатиме відстані у метрах $L = m{\cdot}D$.

У реальних процесах не всі частки рухаються з однаковою швидкістю. Тоді слід враховувати деяку функцію розподілу часток, яка може мати різноманітний вигляд – як мінімум, трьох основних типів:

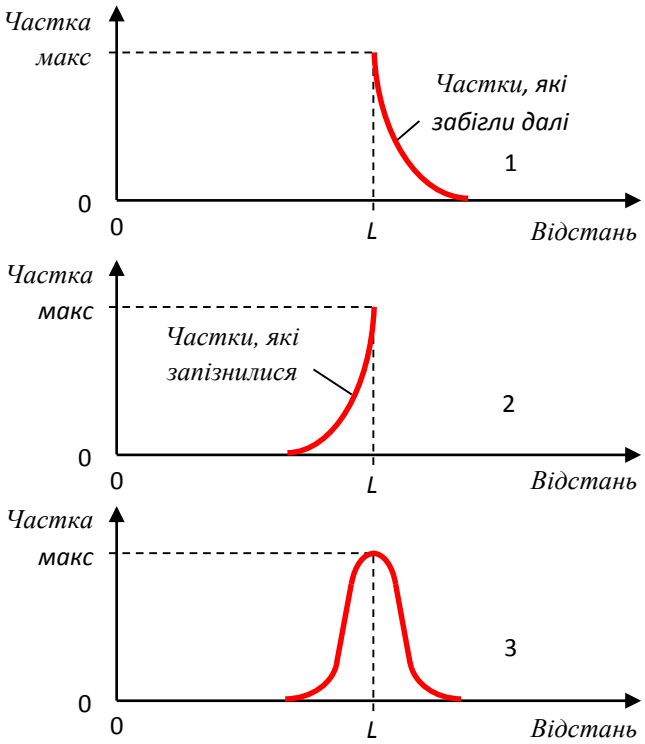


Рис. 12. Типи розподілу часток по відстані

Розглянемо моделювання переносу для функції розподілу першого типу. Позначимо частку того, що забігає далі, через *a*. Тоді основна частка, яка рухається зі швидкістю *S* на ітерацію, буде 1-*a*. Це за умови, що сума коефіцієнтів матриці дорівнює 1 (умова збереження кількості в результаті переносу). Крім того, повинно бути $a < 0.5$. Найпростішою моделлю такого процесу є матриця 5×5 із двома значеннями: *a* та 1-*a*. Нижче наведені деякі результати моделювання зсуву (переносу) при $a = 0.1$.

0	0	0	0	0
0	0	0	0	0
0	0	0	1-*a*	*a*
0	0	0	0	0
0	0	0	0	0

S=1

Рис. 13. Матриця зсуву, яка враховує часткове випередження

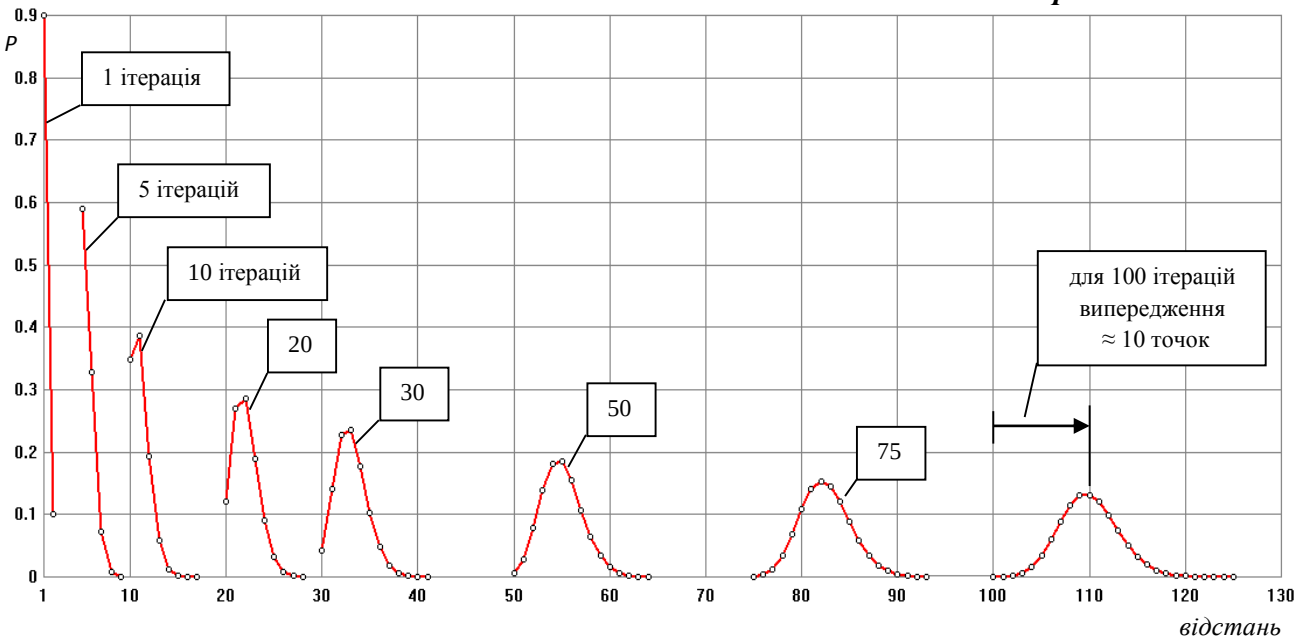


Рис. 14. Результати моделювання переносу при частці випередження a = 0.1 для різної кількості ітерацій

Тепер розглянемо зсув із частковим запізненням. Яку матрицю зсуву вибрати?

0	0	0	0	0
0	0	0	0	0
0	0	0	*b*	1-*b*
0	0	0	0	0
0	0	0	0	0

S=2

0	0	0
0	*b*	1-*b*
0	0	0

S=1

Рис. 15. Варіанти матриці зсуву із запізненням

Обидва варіанти мають недоліки. Матриця при *S*=2 описує зсув одразу на дві дискретні позиції при кожній ітерації. Матриця розмірами 3×3 при *S*=1 має ненульовий центральний елемент (*b*), що означає фактично не зсув, а розтягування. Проте, оскільки кожна ітерація залишає у точці джерела частку b^I, то при $b < 0.5$ це значення стрімко зменшується до нуля.

Чим менше *b*, тим більшою є подібність результатів роботи моделей на основі вказаних двох матриць (за винятком шкали відстаней). Моделювання для різної кількості ітерації це великою мірою підтверджує. Проте, звісно, вказаними двома матрицям не вичерпується різноманіття матриць (кубів) зсуву.

Нижче на графіку рис. 16 наведені результати моделювання зсуву із використанням матриці 3×3 при *S*=1.

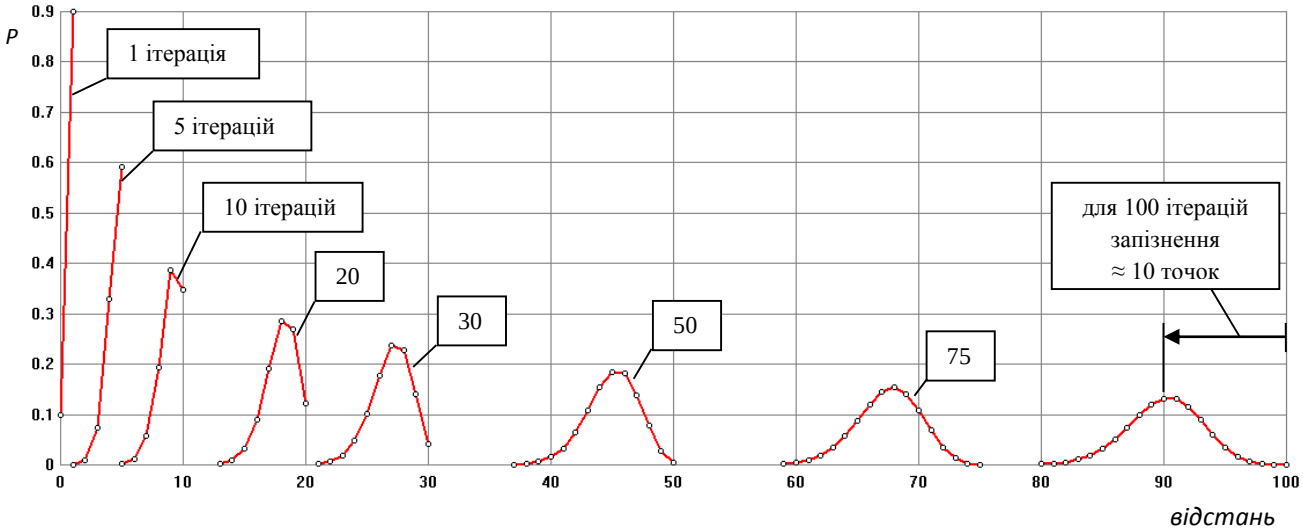


Рис. 16. Результати моделювання переносу при частці запізнення b = 0.1

Якщо порівнювати результати моделювання часткового випередження та запізнення, то можна помітити, що починаючи з 10-ї ітерації зсув фактично не на $D = I{\cdot}S$, а на одну або декілька позицій більше (рис. 14) або менше (рис. 16). Для 100 ітерацій різниця сягає 10 дискретних позицій.

Іншим, як здається, важливим висновком є те, що при зростанні кількості ітерації розподіл значень наближається до функції Гаусса (2)

$$P(x) = G(x\text{-}x_c) \qquad (9)$$

де x_c – відстань до точки, що репрезентує максимальне значення. При 100 ітераціях центр симетрії сигмоїди розташовується приблизно у точці $x_c = 90$. Максимальне відхилення значень становить 0.0088.

Наближення до нормального симетричного розподілу з кожною наступною ітерацію має місце, незважаючи на те, що значення матриць зсуву несиметричні. Ланцюжки ітерацій обумовлюють взаємодію усе зростаючої кількості дискретних елементів, певним чином перемішуючі їхні значення.

4.4. Зсув із розсіюванням

Для розглянутих вище прикладів моделей зсуву із випередженням та запізненням можна констатувати поступове розсіювання вздовж осі відстані. У багатьох реальних процесах розповсюдження зсув поєднується із розсіюванням в усіх напрямках (можливо, несиметричним). Це можна змоделювати, як мінімум, двома способами.

Перший спосіб. Кожна ітерація складається з двох фаз – зсув та розсіювання. Фактично виконується вже по дві ітерації:

B *= 0*
B *= Зсув(**A**)* //1-*а ітерація*
A *= 0*
A *= Розсіювання(**B**)*
B *= 0*
B *= Зсув(**A**)* //2-*а ітерація*
A *= 0*
A *= Розсіювання(**B**)*

Що краще: спочатку зсув, потім розсіювання, чи навпаки? Іншими словами, чи виконується співвідношення

*Розсіювання(Зсув(**P**)) = Зсув(Розсіювання(**P**))*

Це питання залишається відкритим.

Другий спосіб. Виконувати звичайний загальний цикл ітерацій моделювання по одній ітерації на зсув з одночасним розсіюванням

B *= 0*
B *= ЗсувТаРозсіювання(**A**)* //1-*а ітерація*
A *= 0*
A *= ЗсувТаРозсіювання (**B**)* //2-*а ітерація*
B *= 0*
B *= ЗсувТаРозсіювання (**A**)* //3-*а ітерація*

Для забезпечення цього потрібно сформувати відповідну матрицю зсуву-розсіювання.

Нехай стався вибух і було викинуто у атмосферу деяку кількість забруднюючих речовин. Відомі координати точки викиду а також швид-

кість та напрямок вітру у цій місцевості. Потрібно змоделювати розповсюдження забруднення та відобразити на карті відповідні зони небезпеки.

Для моделювання розповсюдження речовини потрібно визначити відповідний куб (матрицю) розсіювання. Розглянемо це на прикладі матриці розмірами 7×7 (рис. 17).

Одна ітерація розсіювання для такої матриці дає зсув межі забруднення на три точки праворуч – на схід. Зсув центру зони (там де значення максимальне) – на дві точки (S=2). Із кожною ітерацією зсувається та розширюється межа зони забруднення.

0	0	0	0	0	0	0
0	0	0	0	0	0	0
0	0	0	0	0.006	0.029	0.042
0	0	0	0.006	0.06	0.6	0.18
0	0	0	0	0.006	0.029	0.042
0	0	0	0	0	0	0
0	0	0	0	0	0	0

Рис. 17. Матриця зсуву та розсіювання

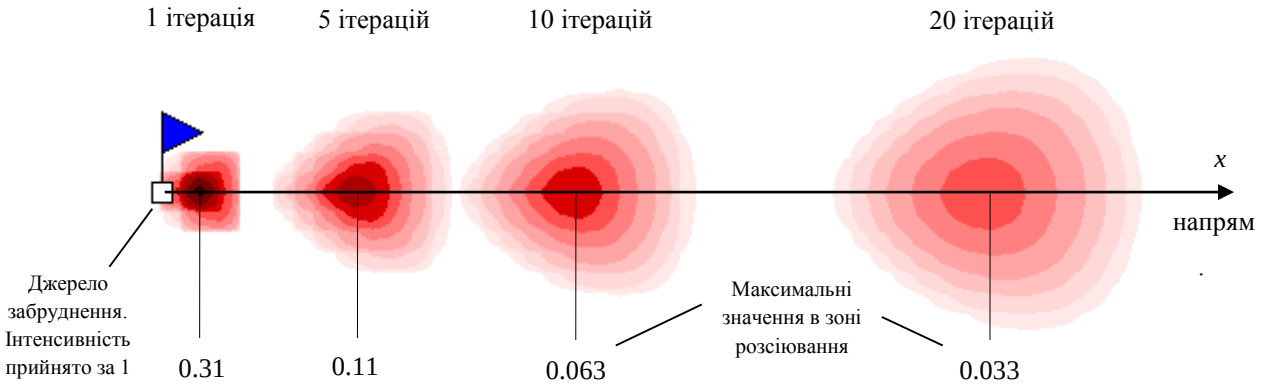


Рис. 18. Пересування та розсіювання

Якщо роздільна здатність шару моделювання становить 1 метр на піксел, тоді одна ітерація, яка дає зсув на два піксела, означає здування хмари на два метра. Якщо вважати, що кожна ітерація моделює проміжок часу в одну секунду, то виходить, що моделюється вітер швидкістю 2 метри в секунду. Якщо кожна ітерація моделює деякий проміжок часу ***t***, то для I ітерацій загальна тривалість процесу, що моделюється, становить $T = I \cdot t$.

Середня швидкість вітру (V) визначається як відстань від джерела до центру зони розсіювання (L) поділену на загальну тривалість (T). Тоді можна записати таке співвідношення:

$$V = \frac{L}{T} = \frac{m \cdot I \cdot S}{I \cdot t} = \frac{m \cdot S}{t} \qquad (10)$$

Як вибирати параметри моделювання? Нехай потрібно змоделювати швидкість вітру V = 20 м/сек. Якщо маємо шар моделювання роздільною здатністю m = 1 м/піксел, то для матриці розсіювання із S = 2 піксели кожна ітерація буде означати проміжок часу в 0.1 секунду:

$$t = \frac{m \cdot S}{V} = \frac{1 \cdot 2}{20} = 0.1$$

Якщо потрібно змоделювати рух хмари впродовж однієї години (T = 3600 сек), то доведеться виконувати 36000 ітерацій:

$$I = \frac{T}{t} = \frac{3600}{0.1} = 36000$$

Це може бути неприйнятним з декількох причин. По-перше, така кількість ітерацій може означати надмірний час роботи комп’ютера. По-друге, мабуть, не потрібно вести розрахунки для кожної одної десятої долі секунди (t=0.1) –

можливо, достатньо відстежувати розташування хмари забруднення кожну хвилину, а то й рідше. По-третє, роздільна здатність шару моделювання m = 1 метр на піксел може бути завеликою – може бути достатньо роздільності вдесятеро гірше, що дозволить суттєво заощадити пам'ять. Так, чи інакше, повинно виконуватися співвідношення

$$V \cdot t = m \cdot S \qquad (11)$$

Наприклад, вихідними даними є V = 20 м/сек, T = 3600 сек. Далі формулюємо такі обмеження: $I < 1000$; $m \leq 20$; $S < 20$; $t \leq 60$. Вибираємо I = 360, тоді $t = T/I$ = 10 сек. Значення m та S обчислюються із співвідношення: $V \cdot t = 20 \cdot 10 = 200 = m \cdot S$

Цим умовам задовольняє m = 20, S = 10. Значенню S=10 відповідає матриця розмірами не менше 21×21.

4.5 Моделювання розповсюдження із урахуванням динаміки інтенсивності джерела, змін напрямку та швидкості вітру

Це можна розглянути на наступному прикладі. Нехай потрібно змоделювати процес розповсюдження впродовж 24 годин

для хмари, яка утворюється після вибуху і рухається відповідно напрямку та швидкості вітру.

Спочатку потрібно визначити динаміку змін основних параметрів – напряму та швидкості вітру та інтенсивність джерела. У графічному вигляді – це рис. 19.

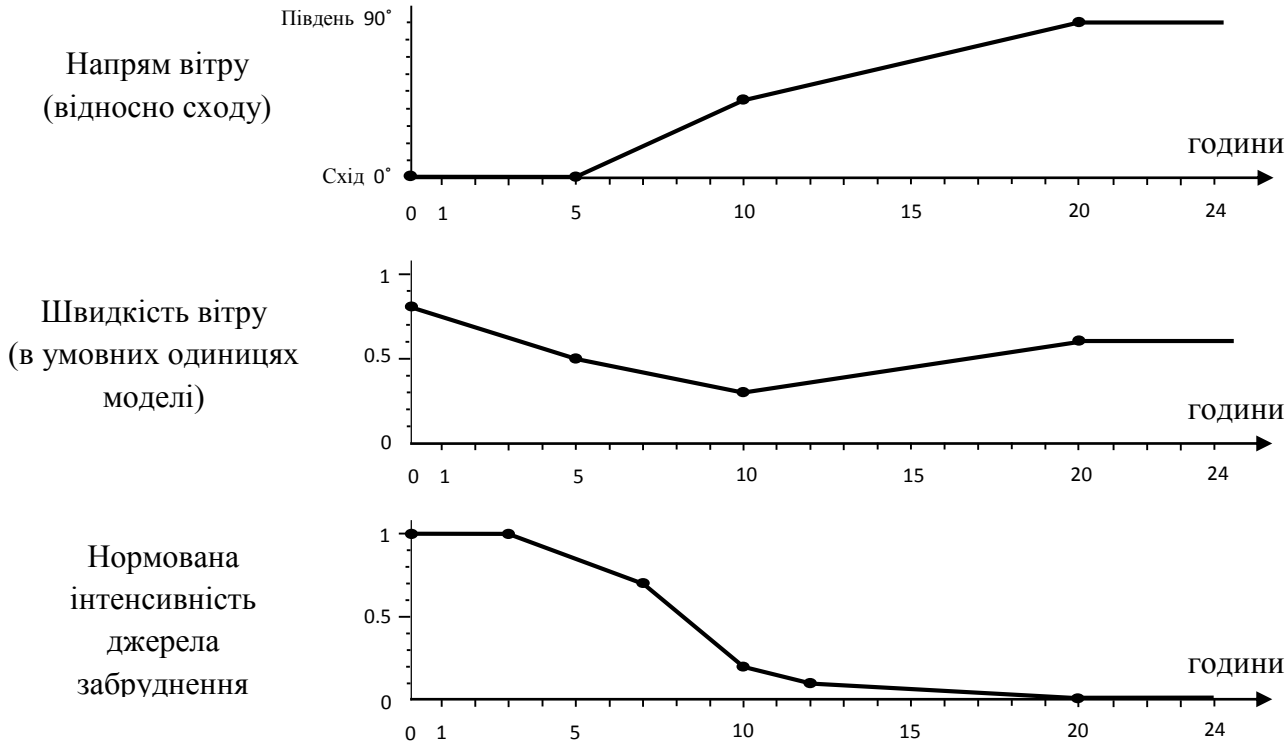


Рис. 19. Опис основних параметрів процесу

Для моделювання таких процесів зручно використовувати мову програмування геоінформаційної системи ГІС "ОКО", яка містить багато операторів, створених спеціально для моделювання процесів розповсюдження. Моделювання зсуву-розсіювання виконується викликом оператора

DispersionLayerWindEx mA aW aV 1 aS vI

Параметр mA – це матриця або куб розсіювання. Для опису конкретної ситуації потрібно сформувати робочі масиви:

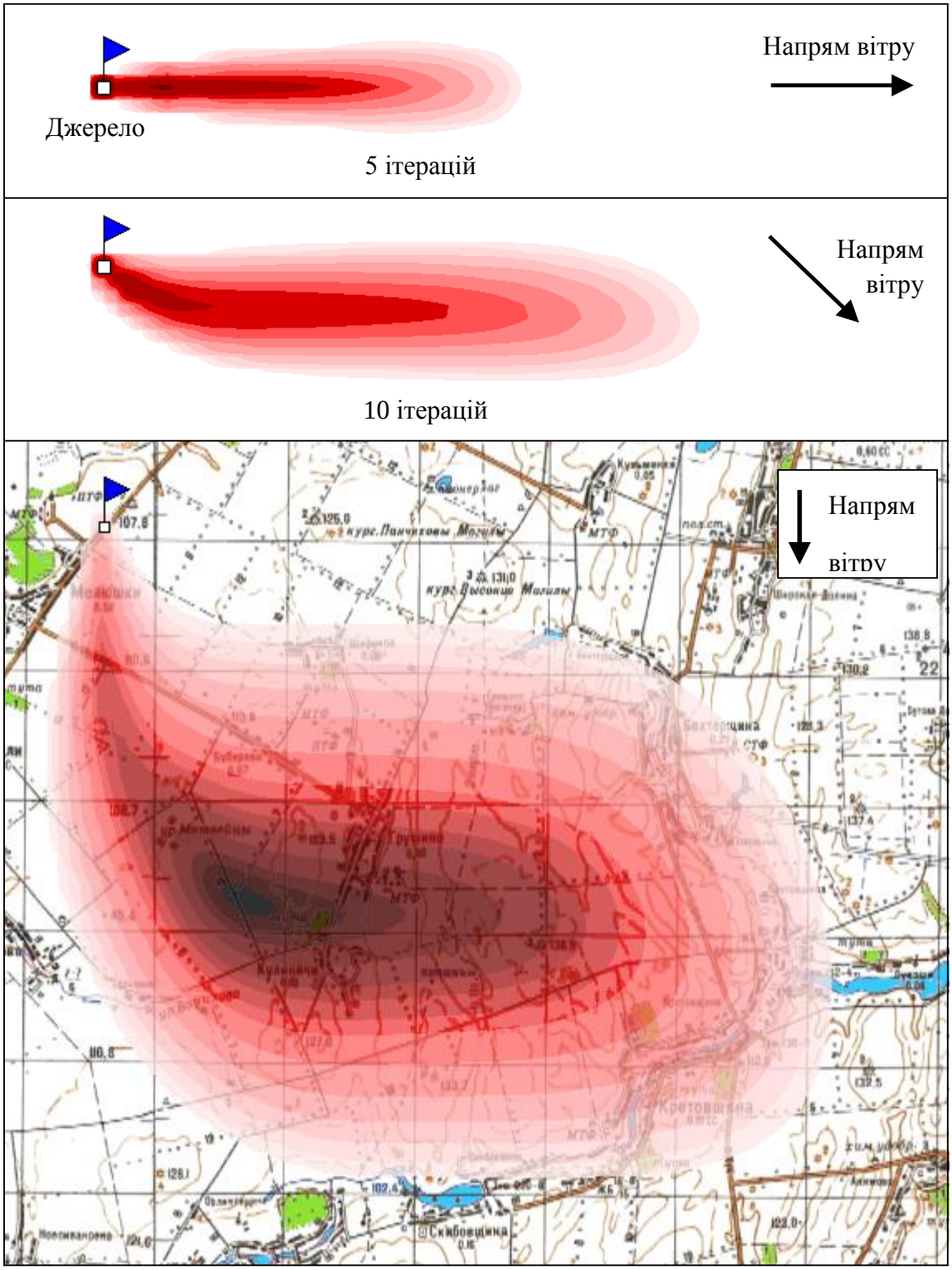
– масив aW для опису динаміки напрямків вітру;

– масив aV для опису динаміки швидкостей вітру;

– масив aS для опису динаміки джерела забруднення.

Для моделювання створюється шар потрібної роздільної здатності та матриця зсуву-розсіювання. Значення параметрів растрової моделі знаходяться з рівнянь (10) та (11).

У середовищі ГІС виконується необхідна кількість ітерацій. Якщо, наприклад, t=1 година, то 24 ітерації відповідатимуть 24 годинам реального часу (рис. 20)



Виконано 24 ітерації

Рис. 20. Ілюстрація моделювання розповсюдження хмари забруднення та відображення на карті результатів заключної ітерації

5. Висновки

Запропоновані евристичні алгоритми на основі метода кінцевих елементів для моделювання процесів розповсюдження, зокрема, розсіювання та зсуву. Як виявилося, ці алгоритми підтримують гаусову модель, яка загальноприйнята для опису таких процесів. Це може бути свідченням загальної адекватності запропонованого підходу.

Впровадження таких алгоритмів у геоінформаційних системах дозволяє ефективно вирішувати засобами растрового моделювання достатньо складні завдання аналізу процесів розповсюдження.

Нарощування можливостей комп'ютерних систем, зокрема, обсягів оперативної пам'яті, є корисним для подальшого розвитку растрового моделювання.

Список посилань

1. Бутенко О.С., Охарєв В.О. Механізм визначення кількісних характеристик рівня концентрації забруднюючих речовин викидами автомобільного транспорту // "Екологічна безпека та природокористування": Зб. наук. праць. М-во освіти і науки України, НАНУ. – К. – 2009, Вип. 3. – С14-33.
2. Левченко Н.О., Євстратьев О.А., Гогунський В.Д. Конкурс расчетных методик рассеяния загрязняющих веществ в атмосферном воздухе // Труды Одесского политехнического университета. – 1997, Выпуск 2. – С. 136-139.
3. Briggs G.A. A plume rise model compared with observations, JAPCA, 1965, pp.433-438.
4. Методика прогнозирования масштабов заражения сильдействующими ядовитыми веществами при авариях (разрушениях) на химически опасных объектах и транспорте РД 52.04.253-90. [Електронний ресурс]. – Режим доступу: http://libgost.ru/rd/67060-Tekst_RD_52_04_253_90.
5. Методика расчёта концентраций в атмосферном воздухе вредных веществ, содержащихся в выбросах предприятий. ОНД-86 / Под ред. М. Е Берлянда. – Л.: Гидрометеоиздат, 1987. – 94 с.
6. Hanna S.R., Briggs G.A. and Hosker R.P. Handbook on Atmospheric Diffusion. DOE/TIC-11223, Department of Energy, 1982, 102 pp.
7. Turner D.B. Workbook Of Atmospheric Dispersion Estimates. U.S. Department Of Health, Education, And Welfare. [Електронний ресурс]. – Режим доступу: http://www.archive.org/download/workbookofatmosp026353mbp.pdf
8. Горячев Г.В., Гаврилюк М.А. Моделирование распространения загрязняющих веществ в атмосферном воздухе с использованием ГИС-технологий по методике ОНД-86 // Наукові праці ВНТУ. Електронний журнал. – 2009, №3. [Електронний ресурс]. – Режим доступу: http://praci.vntu.edu.ua/article/view/1069/175.
9. ArcGIS Spatial Analyst. ESRI. [Електронний ресурс]. – Режим доступу: http://www.esri.com/software/arcgis/extensions/spatialanalyst
10. Порєв В.М. Можливості та перспективи ГІС "ОКО" // Матеріали VI Міжнародної науково-практичної конференції ГІС-Форум. – К. – 2006. С.206-208.
11. Тоффоли Т., Марголус Н. Машины клеточных автоматов. – М.:Мир,1991.– 280 с.
12. Бандман О.Л. Клеточно-автоматные модели пространственной динамики // Системная информатика. – 2006. – №10. – С. 59-113
13. Сергеев А.П., Берг Д.Б., Губарев С.В. «Цифровые» реалии аналитических моделей // Вестник УрО РАН. Наука. Общество. Человек. – 2009. – 2 (28). – С. 38-49.
14. Титов В. П. К вопросу о моделировании диффузии газов в потоке воздуха // Сб. трудов № 144 «Теплогазоснабжение и вентиляция». – М.: МИСИ, 1977. – С.45-50.
15. Wolfram S. A new kind of science. Champaign, Ill., USA: Wolfram Media Inc., 2002. – 1197 pp.
16. Порєв В.М. Деякі оцінки моделі розповсюдження на основі клітинних автоматів. Вісник Криворізького технічного університету – 2012. – № 30. – С.7-10.

УДК 004.421.5; 681.3

ГАПАК О.М.

ОЦІНКА ЯКОСТІ ГЕНЕРАТОРА ГОЛЛМАННА, РЕАЛІЗОВАНОГО НА ОСНОВІ FCSR

У статті представлено результати дослідження генератора Голлманна на основі FCSR. Проведено тестування генератора за методикою NIST STS при різній кількості та порядку базових компонент. Показано, що за результатами тестів NIST STS ці генератори не відповідають вимогам, які ставляться до генераторів псевдовипадкових послідовностей.

The results of the research of Hollmann generator on the basis of FCSR are presented in this work. Generator was tested by the technique NIST STS at different quantity and order of the basic components. On the results of NIST STS testing these generators don't meet the requirements that apply to generators of pseudorandom sequences.

Вступ

У сучасних умовах зростання продуктивності обчислювальних систем та можливостей об'єднання значної кількості комп'ютерів для порушення безпеки даних, актуальною є проблема підвищення надійності захисту інформації, в тому числі засобами, в основі яких лежить використання псевдовипадкових послідовностей (ПВП) [1].

ПВП у криптографії використовуються для генерування ключів симетричних та асиметричних систем; потокового шифрування; генерування електронного цифрового підпису тощо. Розробники генераторів ПВП (ГПВП) повинні дотримуватись ряду вимог: висока швидкодія; якомога більший період повторюваності; простота програмної та апаратної реалізації; можливість керування вхідними параметрами та інші [2].

Найпоширенішими у програмно-апаратній реалізації є ГПВП на основі регістрів зсуву з лінійним зворотним зв'язком – LFSR (Linear Feedback Shift Register). Ці генератори знайшли широке використання у різних галузях науки і техніки. Існує багато методів щодо підвищення їх криптографічних властивостей. До них можна віднести генератор Голлманна, який реалізується на кількох регістрах LFSR. Властивості цих генераторів при правильній реалізації є кращими в порівнянні із генератором LFSR. Результати досліджень таких генераторів Голлманна широко висвітлені у літературі, приведені оцінки їх якості та визначені оптимальні параметри [1; 2; 3]. Проведений аналіз літературних джерел у цій галузі вказує на перспективу використання ГПВП на основі регістрів зсуву зі зворотним зв'язком та перенесення – FCSR (Feedback with Carry Shift Register), які є малодослідженими. В основному всі роботи, присвячені цьому типу генераторів, зводяться до рекомендацій щодо побудови та формального опису принципів їх функціонування [4; 5]. Є декілька досліджень періоду генератора FCSR [6; 7]. У статті [8] ми описали роботу ГПВП на основі FCSR, розглянули його модифікації та прийшли до висновку, що доцільно використати принцип "stop-and-go" для побудови генератора Голлманна. Великий інтерес становить оцінка його якості при різних початкових параметрах.

Метою нашої роботи є проведення оцінки якості генераторів Голлманна на основі FCSR за допомогою статистичних тестів NIST STS, визначення впливу вхідних параметрів базових FCSR на якість генератора та порівняння його із генератором Голлманна, побудованого на основі LFSR.

Каскад Голлманна на основі FCSR

FCSR схожий на LFSR, в обох є регістр зсуву та функція зворотного зв'язку, різниця полягає в тому, що у FCSR є також регістр перенесення. На рисунку 1 представлена конфігурація Фібоначчі. У порівнянні з LFSR, замість *xor* над усіма бітами відвідної послідовності, ці біти додаються один з одним і вмістом регістру перенесення. Результат *mod 2* стає новим бітом, результат *div 2* стає новим вмістом регістру перенесення.

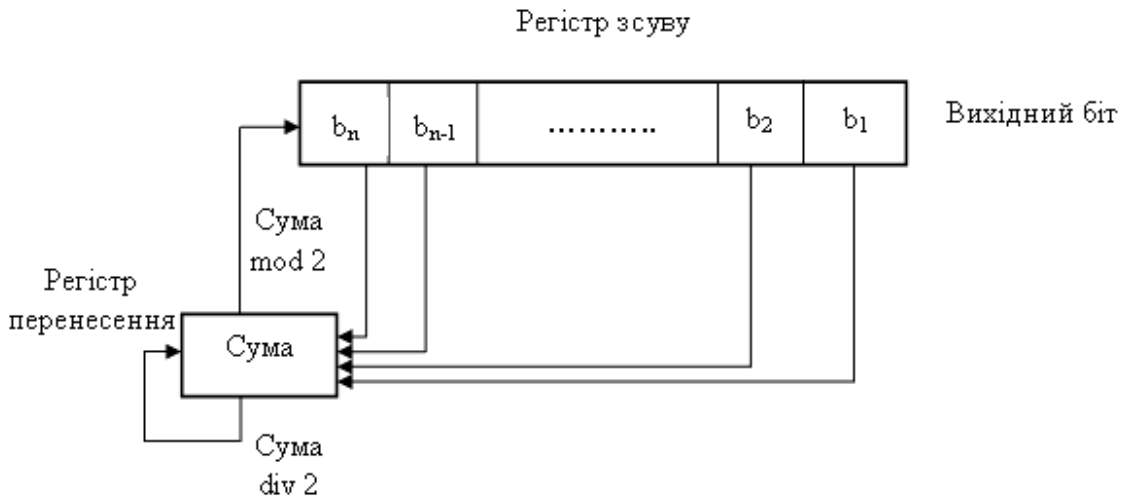


Рис. 1. FCSR

Максимальний період FCSR дорівнює *q*-1, де *q* – ціле число зв'язку, що задає відвідну послідовність. Якщо для *n*-розрядного регістра відвідна послідовність задана у вигляді (*n,m,k,p*), то період становитиме $T = 2^n + 2^m + 2^k + 2^p - 2$. Зауважимо, що при використанні *n*-розрядного регістра LFSR максимальний період становить $T = 2^n - 1$.

Проаналізувавши літературу [2; 3; 4; 6], можемо розширити використання генераторів на основі FCSR, аналогічно до генераторів на основі LFSR.

Каскад Голлманна являє собою підсилену версію генератора «stop-and-go». Він складається із деякої послідовності генераторів FCSR, тактування кожного з яких керується попереднім FCSR. Якщо виходом FCSR-1 в момент часу *t* є 1, то тактується FCSR-2, інакше повторюється попереднє значення FCSR-2. Якщо виходом FCSR-2 в момент часу *t* є 1, то тактується FCSR-3 і т.д. Вихід останнього FCSR є виходом генератора Голлманна (рис.2).

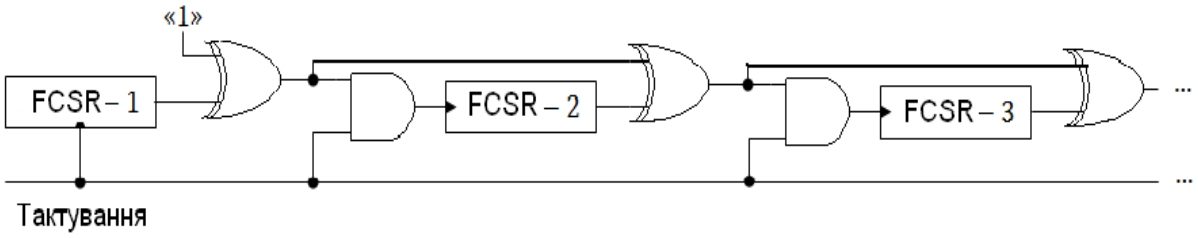


Рис. 2. Каскад Голлманна

Для дослідження ГПВП нами розроблений програмний засіб на мові C#. Цей засіб дозволяє згенерувати ПВП за алгоритмом Голлманна із різною кількістю, порядком та довжиною базових FCSR.

Оскільки будь-які послідовності, які породжені ГПВП безпосередньо для криптографічних цілей, підлягають обов'язковому тестуванню, то для проведення оцінки якості генератора Голлманна ми обрали набір статистичних тестів NIST STS.

Статистичні тести NIST STS (National Institute of Standards and Technology Statistical Test Suite) використовуються для визначення якісних та кількісних ознак випадковості послідовності чисел. Сьогодні ця методика є найбільш поширеною серед розробників криптографічних засобів захисту інформації. Пакет NIST STS містить 16 статистичних тестів, які розроблені для перевірки гіпотези про випадковість двійкових послідовностей довільної довжини, породжуваних ГПВП. Усі тести спрямовані на виявлення різних дефектів випадковості. Кожен тест подається у розрізі мети, позначення, статистик, опису та правил інтерпретації результатів [9].

Для заданого генератора Голлманна, відповідно до вимог NIST STS, ми сформували двійкові послідовності довжиною 10^8 біт, для того

щоб із них можна було б отримати 100 підпослідовностей довжиною в 1000000 біт.

Для початку ми визначили, як впливає порядок вибору базових FCSR на статистичні властивості ПВП. Ми обрали три регістри FCSR-1(2,1); FCSR-2(3,2); FCSR-3(3,2,1), сформували шість можливих варіантів генератора Голлманна та провели оцінки якості таких генераторів. Під час вибору базових компонент, що дають максимальний період, ми опиралися на працю Б. Шнаєра [4]. Результати досліджень наведено в таблиці 1.

Табл. 1. Результати тестування 1

Номер з/п	Послідовність базових регістрів	Кількість пройдених тестів
1	FCSR-1; FCSR-2; FCSR-3	3
2	FCSR-1; FCSR-3; FCSR-2	0
3	FCSR-2; FCSR-1; FCSR-3	1
4	FCSR-2; FCSR-3; FCSR-1	0
5	FCSR-3; FCSR-1; FCSR-2	0
6	FCSR-3; FCSR-2; FCSR-1	3

Аналогічні дослідження проведено на регістрах більшої розрядності FCSR-4(6,2); FCSR-5(7,2); FCSR-6(8, 3,2,1). Результати досліджень наведено у таблиці 2.

Табл. 2. Результати тестування 2

Номер з/п	Послідовність базових регістрів	Кількість пройдених тестів
1	FCSR-4; FCSR-5; FCSR-6	5
2	FCSR-4; FCSR-6; FCSR-5	4
3	FCSR-5; FCSR-4; FCSR-6	4
4	FCSR-5; FCSR-6; FCSR-4	4
5	FCSR-6; FCSR-4; FCSR-5	4
6	FCSR-6; FCSR-5; FCSR-4	5

Отже, із одержаних результатів можемо зробити висновки, що на якість генератора впливає порядок вибору базових регістрів. Як бачимо, обирати регістри потрібно у порядку спадання чи зростання їх розрядності та періоду базових FCSR. Ці висновки підтверджують аналогічні дослідження щодо генератора Голлманна на основі LFSR [3].

Далі визначено, як впливає кількість базових FCSR на статистичні властивості ПВП. Для цього ми обрали однакові 3-бітні FCSR(3,2,1) та 7–бітні FCSR(7,4,2,1) із різним початковим вмістом регістрів. У таблиці 3 вказано результати експерименту.

Із результатів проведених експериментів можемо стверджувати, що кількість базових регістрів впливає на статистичні властивості генератора Голлманна, при цьому необхідно використовувати більш як 15 регістрів, що підтверджується рекомендаціями криптологів [3; 4].

Табл. 3. Результати тестування 3

Кількість регістрів	FCSR(3,2,1)	FCSR(7,4,2,1)
	Кількість пройдених тестів	Кількість пройдених тестів
3	1	1
5	0	2
7	1	3
9	1	4
15	4	4
19	4	4

Насправсді є один суттєвий недолік: використання однакових регістрів FCSR у каскаді Голлманна є неефективним. Так, у дослідженнях генератора Голлманна на основі LFSR із твірним поліномом 7-го степеня кількість пройдених тестів значно більша – уже при використанні п’яти регістрів вона становить 11 [2]. Тому наступні наші дослідження ми проводили із використанням різних базових FCSR у каскаді Голлманна. Для аналізу ми обрали 13-бітні базові регістри із різними відвідними послідовностями та початковим вмістом регістрів. Під час тестувань виявили, що генератор побудований на основі одного FCSR, має кращі статистичні властивості (7 пройдених тестів), а ніж генератор Голлманна (не більше 5 пройдених тестів). Такі дані виявлені й для інших регістрів FCSR.

Висновки

Аналіз проведених тестувань ПВП побудованої за каскадом Голлманна на основі FCSR, показав, що використання великої кількості базових регістрів приводить до збільшення періоду ПВП. Але, в той же час, згенеровані послідовності не проходять значну частину тестів NIST STS у порівнянні з такими ж генераторами на основі LFSR. Таким чином, на нашу думку, використовувати генератор Голлманна на основі FCSR у криптографії недоцільно.

Перспективами подальших досліджень у цьому напрямку є побудова генератора Голлманна із комбінацією FCSR/LFSR при різних функціях об’єднання; проведення оцінки якості ГПВП з використанням статистичних тестів.

Список посилань

1. Зохре Карім Заде. Програмно-апаратні засоби генерації псевдовипадкових послідовностей для підвищення ефективності захисту інформації в ЕОМ та мережах : автореф. дис. на здобуття наук, ступеня канд. техн. наук : спец. 05.13.13 «Обчислювальні машини, системи та мережі» / Зохре Карім Заде. – Київ, 2007. – 18 с.
2. Костів Ю. М. Визначення оптимальних параметрів генератора Голлманна за допомогою статичних тестів NIST / Ю. М. Костів, В. М. Максимович, О. І. Герасимчук, Я. Р. Совин, М. М. Мандрона // Вісник Національного університету «Львівська політехніка», серія «Автоматика, вимірювання та керування». – 2013. – № 753. – С. 57–67.
3. Гарасимчук О. І. Оцінка якості генератора Голлманна, реалізованого на основі модифікованих генераторів М-послідовностей / О. І. Гарасимчук, Ю. М. Костів, Т. Г. Паршенко // Системи обробки інформації. – 2010. – №6(87). – С. 35– 38.
4. Шнайер Б. Прикладная криптография: Протоколы, алгоритмы и исходные тексты на языке С / Б. Шнайер.− М. : Триумф, 2002. – 816 с.
5. Klapper A. Fibonacci and Galois Representations of Feedback with Carry Shift Registers / A. Klapper, M. Goresky, – IEEE Trans – 2004, pp. 56–71.
6. Mittelbach M. Investigation of FCSR-based Pseudorandom Sequence Generators for Stream Ciphers / M. Mittelbach, A. Finger // Proc. of International Conference on Networking (ICN), Gosier, Guadeloupe, France, Mar. 2004 . [Електронний ресурс]. – режим доступу: http://www.researchgate.net/publication/228853356_Investigation_of_FCSR-based_pseudorandom_sequence_generators_for_stream_ciphers/.
7. Shyrochin V.P. Investigations of the basic component of FCSR –generator / V.P. Shyrochin, I.V. Vasyltsov, B.Z. Karpinskij // Computing, 2003, Vol. 2, Issue 3, pp. 77-81. [Електронний ресурс]. – режим доступу: http://computingonline.net/index.php/computing/article/viewFile/234/209.
8. Гапак О. М. Визначення довжини періоду генераторів псевдовипадкових послідовностей на основі регістрів зсуву зі зворотним зв’язком та перенесення / О. М. Гапак // Моделювання та інформаційні технології – 2014. – №73. – С. 92– 97.
9. NIST SP 800-22rev1a. A Statistical Test Suite for Random and Pseudorandom Number Generators for Cryptographic Applications//National Institute of Standards and Technology Special Publication 800-22rev1a, 2010, 131 p.

УДК 519.254

ГОТОВИЧ В.А.,
НАЗАРЕВИЧ О.Б.

ЗАСТОСУВАННЯ МЕТОДУ “ГУСЕНИЦЯ-SSA” ДЛЯ АНАЛІЗУ РІЧНОГО ЧАСОВОГО РЯДУ ЕЛЕКТРОНАВАНТАЖЕННЯ ОРГАНІЗАЦІЇ

Запропоновано застосування методу “Гусениця-SSA” для обробки часових рядів електронавантаження. Досліджено часовий ряд електронавантаження організації на річному інтервалі спостереження, в результаті якого часовий ряд розділено на окремі компоненти: річний тренд, періодичні складові та стохастичний залишок.

Proposed application of the method "Caterpillar-SSA" for processing time series of electricity load. Researched the time series of electricity load of organization during the year, in which the time series is divided into separate components: an annual trend, periodic and stochastic components.

1. Вступ

Необхідність ощадливого споживання енергоресурсів унаслідок невпинного скорочення світових запасів енергоносіїв, постійного їх подорожчання, зумовлюють актуальність задач економії та моніторингу споживання енергоресурсів. У цьому контексті важливе місце посідає електроенергія як універсальний та один із найрозповсюдженіших видів споживаної населенням та промисловістю енергії. Тому актуальним є моніторинг споживання енергоносіїв, зокрема, електроенергії. Моніторинг потрібен на різних рівнях споживання, починаючи від окремо взятих підприємств, установ та організацій, житлових будинків, закінчуючи рівнями галузей народного господарства і країни в цілому.

Для ефективного розв’язання завдань моніторингу споживання електроенергії необхідна побудова адекватних інформаційних технологій, які в свою чергу базуються на відповідних моделях, методах, алгоритмах та їх реалізації апаратно-програмними засобами.

2. Постановка завдання

У цій роботі ставимо завдання дослідити процес електронавантаження організації (навчального корпусу №1 університету ТНТУ ім. Івана Пулюя) на річному інтервалі спостереження за допомогою методу “Гусениця-SSA” з метою виявлення характеру динаміки даного процесу, а також оцінки виділених цим методом компонент: річного тренду, періодичних складових та стохастичного залишку.

3. Аналіз літератури

В загальному випадку на формування процесу електронавантаження впливає багато випадкових факторів. Наприклад, випадкові моменти часу, в які відбувається ввімкнення користувачами електроприладів різної потужності. Окрім того, на динаміку процесу електронавантаження конкретної організації (установи) впливають такі чинники (фактори): режими роботи установи, день тижня, температура повітря навколишнього середовища, тривалість світлового дня і т.п.. На формування динаміки процесу електронавантаження також впливає періодичність людської діяльності. З огляду на це процесу електронавантаження взагалі і навчального корпусу університету зокрема властива стохастична періодичність.

У результаті огляду публікацій, у яких описуються дослідження стохастично періодичних процесів виявлено, що автори пропонують використовувати такі математичні моделі: адитивну та мультиплікативну моделі, періодично-корельований випадковий процес, лінійний періодичний випадковий процес [1, 2, 3], періодичний процес за Слуцьким [4], кусково-стаціонарний випадковий процес [5]. Запропоновані моделі адекватно описують випадкові процеси на відносно коротких інтервалах спостереження. У нашому випадку для вирішення поставленого завдання необхідно побудувати математичну модель на річному інтервалі спостереження.

На сьогоднішній день дослідники широко використовують такі чисельні методи аналізу реалізацій (часових рядів) нестаціонарних випадкових процесів як: авторегресії та проінтегрованого ковзного середнього (ARIMA) [6],

групового врахування аргументів (МГУА) [7], головних компонент (МГК) [8], емпірична модова декомпозиція (EMD) [9], сингулярного спектрального аналізу "Гусениця-SSA" [10] тощо. Для цих методів розроблено алгоритмічно-програмне забезпечення, зокрема, доступне для використання в математичних пакетах (Mathcad, Matlab, R).

У даній роботі для дослідження реалізації процесу електронавантаження на річному інтервалі спостереження авторами пропонується застосувати метод "Гусениця-SSA". Його перевагами є відносна простота (метод є чисельним і здебільшого полягає в роботі з матрицями) а також те, що не потрібно попередньо задавати модель досліджуваного часового ряду. Завдяки цьому можна застосовувати метод для аналізу широкого кола часових рядів, у тому числі й нестаціонарних.

4. Виклад основного матеріалу дослідження

Метод "Гусениця-SSA" полягає у перетворенні одновимірного часового ряду в багатовимірний (т. зв. траєкторну матрицю), сингулярного розкладу матриці, визначення власних функцій, групування і подальшого застосування методу головних компонент. Це дає можливість розділити часовий ряд на окремі компоненти: річний тренд, періодичну складову (суму періодичних складових) і стохастичний залишок. Таким чином математична модель випадкового процесу подається у вигляді суми:

$$f(x) = f_T(t) + f_n(t) + f_r(t) + \varepsilon(t), t \in [0, T], \quad (1)$$

де $f_T(t)$ – річний тренд (оцінка динаміки процесу впродовж року),

$f_n(t)$ – періодична складова або сума періодичних складових,

$f_r(t)$ – швидкі нерегулярні коливання малої інтенсивності (можуть бути і не відтворені в конкретному випадковому процесі),

$\varepsilon(t)$ – випадкова складова, шум.

Відповідно до поставленого в даній роботі завдання, маємо на меті розкласти досліджуваний річний часовий ряд електронавантаження на тренд, періодичні складові та випадковий процес (шум), а також знайти обґрунтування фізичної природи виявлених у результаті розкладу компонент.

Метод "Гусениця"-SSA є альтернативою застосуванню відомих підходів. Так моделі типу "тренд – шум" або "авто регресія – ковзне середнє" добре підходять лише для часових рядів відносно простої структури, тоді як аналіз Фур'є, регресійний аналіз чи вейвлет-перетворення призводять до строгої періодичності внаслідок розкладання вихідної функції по фіксованій системі базисних функцій.

На практиці використовують різні модифікації методу. Констатуємо, що метод розвивається до сьогодні. У зв'язку з цим, виділяють два основних напрямки застосування методу: розв'язання задач загального типу (виділення тренду, згладжування, виявлення періодичностей, прогнозування) і спектральний аналіз стаціонарних часових рядів з метою дослідження характеру їх динаміки.

Переваги методу було викладено вище. Основним його недоліком є необхідність наявності в дослідника певного досвіду. Метод є інтерактивним і може давати результати, які дещо відрізняються між собою:

− по-перше, необхідно коректно підібрати параметр методу – довжину вікна. Чим більша довжина вікна, тим на більшу кількість складових буде розкладено вихідний ряд. При цьому можливе змішування із шумом періодичних компонент, які спричиняють малий вклад у вихідний ряд. Окрім того, можливі обмеження в обчислювальній потужності обладнання, яке використовується дослідником, адже із зростанням довжини вікна зростає час виконання необхідних обчислень. З іншого боку, занадто мала довжина вікна призведе до змішування складових компонент ряду, які чинять порівняно великий внесок в ряд;

− по-друге, складно піддається формалізації етап групування, на якому необхідно коректно відділити компоненти розкладеної траєкторної матриці, що відповідають періодичним складовим ряду, від компонент які відповідають стохастичному залишку. В цьому полягає інтерактивність методу – групування компонент проводить дослідник, виходячи з власного досвіду.

Суть методу та рекомендації щодо його застосування детально розглянуто в наукових працях, наприклад у [10]. З огляду на це не будемо додатково описувати математичний апарат методу. Перейдемо до отриманих результатів. На рис. 1 зображено графік досліджуваної реалізації процесу електронавантаження. Відліки ряду зафіксовано з інтервалом в одну годину.

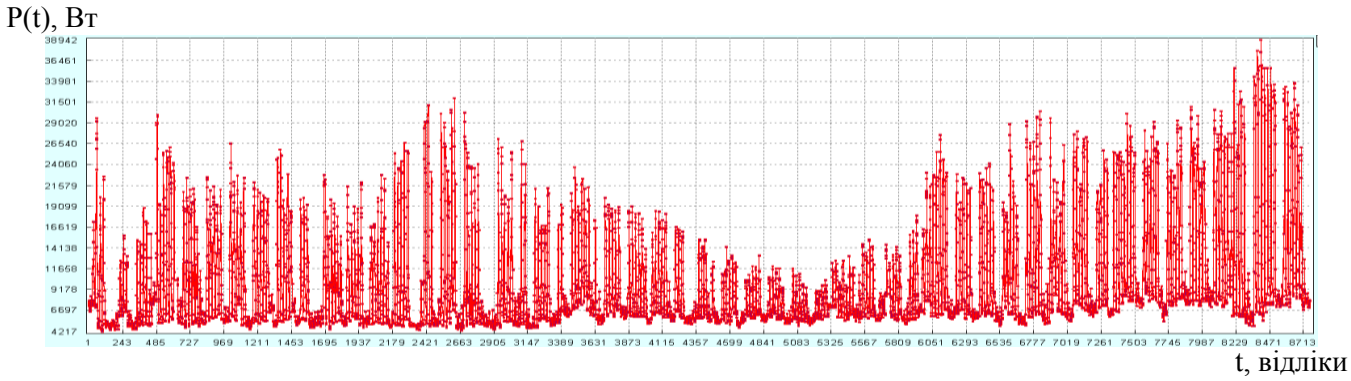


Рис. 1. Графік ряду електронавантаження по фазі А корпусу №1 ТНТУ ім. Івана Пулюя впродовж 2007 року

Експериментальним шляхом для аналізу часового ряду було вибрано довжину вікна (параметр гусениці) $L = 672$ відліки. Довжину вікна підібрано з урахуванням того, що вона повинна бути кратною періоду основної складової часового ряду. Прогнозуємо, що у зв’язку із періодичною повторюваністю людської діяльності, потенційними періодами досліджуваного часового ряду є доба (24 відліки) і тиждень (168 відліків). Запропонована довжина гусениці (число 672) кратне і 168 і 24.

На рис. 2 і 3 представлено графіки перших 18 рангованих по спаданню компонент, отриманих у результаті розкладу річного часового ряду. Спостерігаємо основну негармонічну компоненту, яку можна віднести до річного тренду (під №1 із часткою вкладу в ряд 77%), а також декілька гармонічних компонент сезонного характеру. Всі наступні складові ряду, починаючи із 16-ї або мають негармонічний характер, або характеризуються мізерним вкладом у вихідний ряд (менше 0,2%), вважатимемо їх шумом.

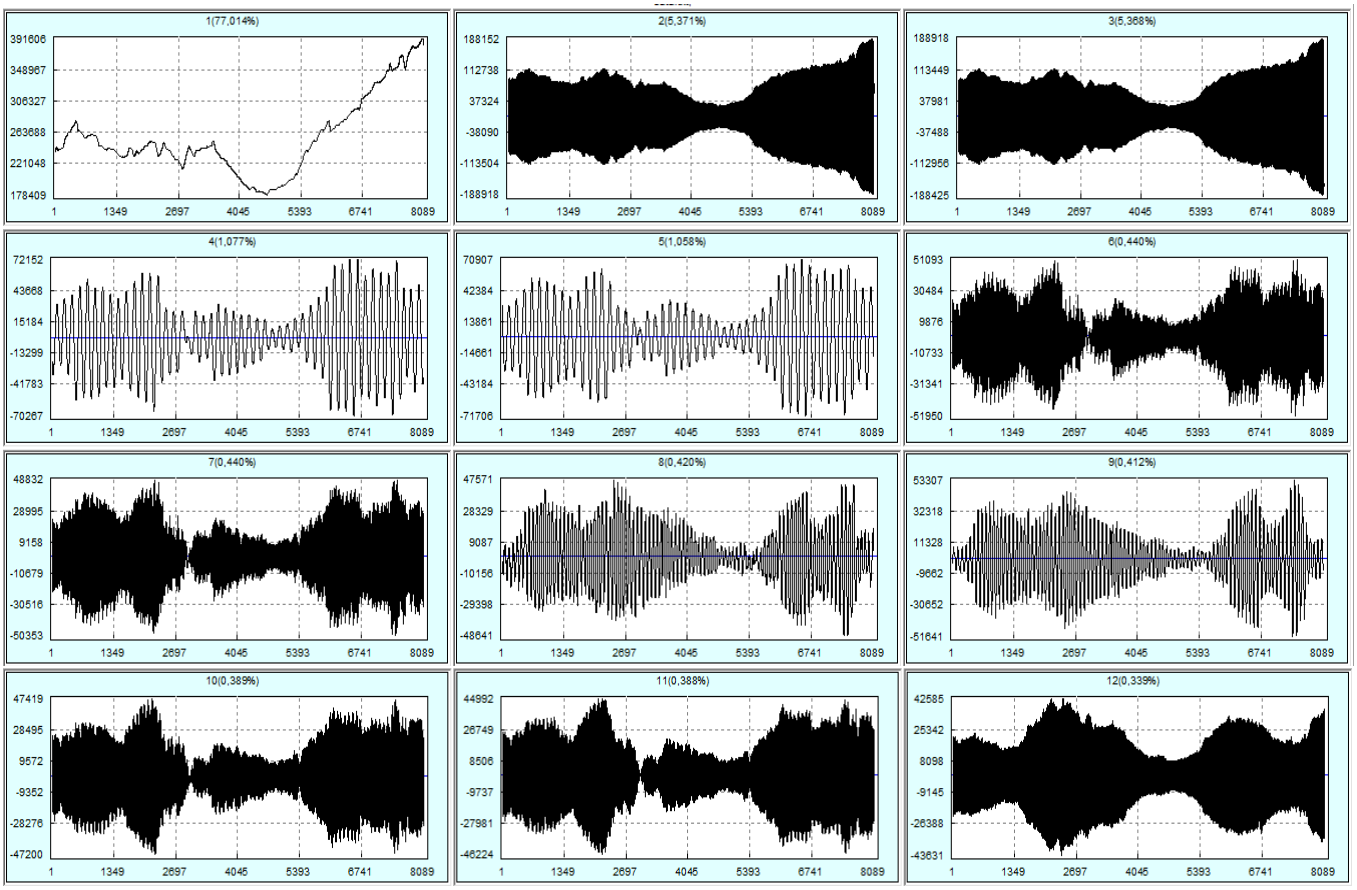


Рис. 2. Графіки перших 12 компонент, отриманих після розкладу часового ряду методом “Гусениця-SSA”

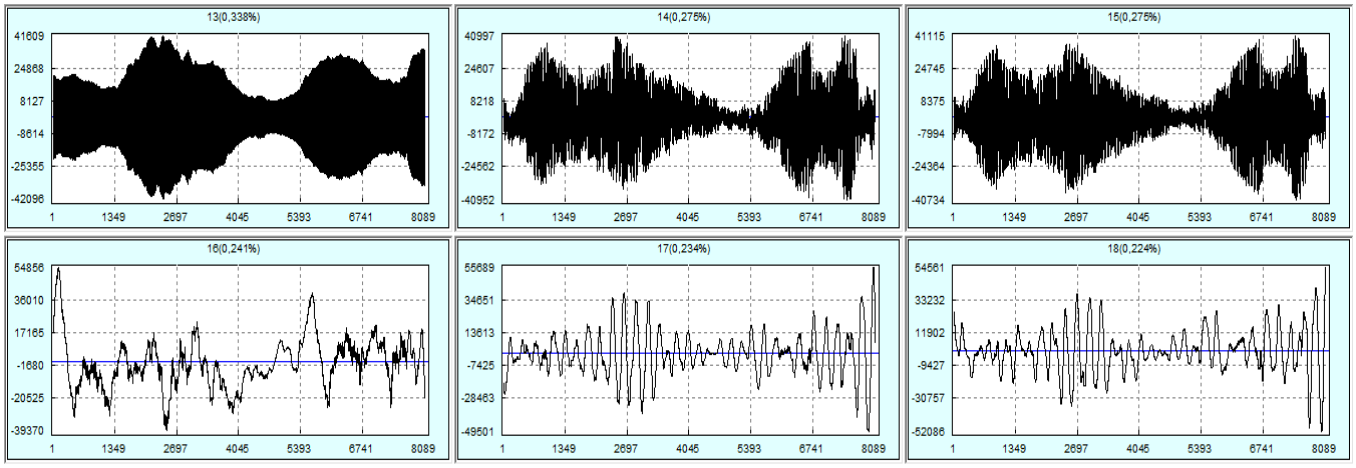


Рис. 3. Графіки компонент №№ 13-18, отриманих після розкладу часового ряду методом "Гусениця-SSA"

Наступний етап методу "Гусениця-SSA" – групування компонент, виділених після розкладу. В першу групу виділимо лише першу компоненту (неперіодичний, повільно змінний характер графіка говорить про те, що це річний тренд ряду). На основі схожості графіків окремих компонент ряду, отриманих в результаті етапу розкладу, та відсотка їх внеску у вихідний ряд згрупуємо разом 2 та 3 компоненти, 4 та 5, 6 та 7, 8 та 9, 10 та 11, 12 та 13, 14 та 15 компоненти. Всі компоненти під номерами 16 - 672 зведемо в одну групу. Графіки компонент ряду, отримані після етапу групування, подано на рис 4.

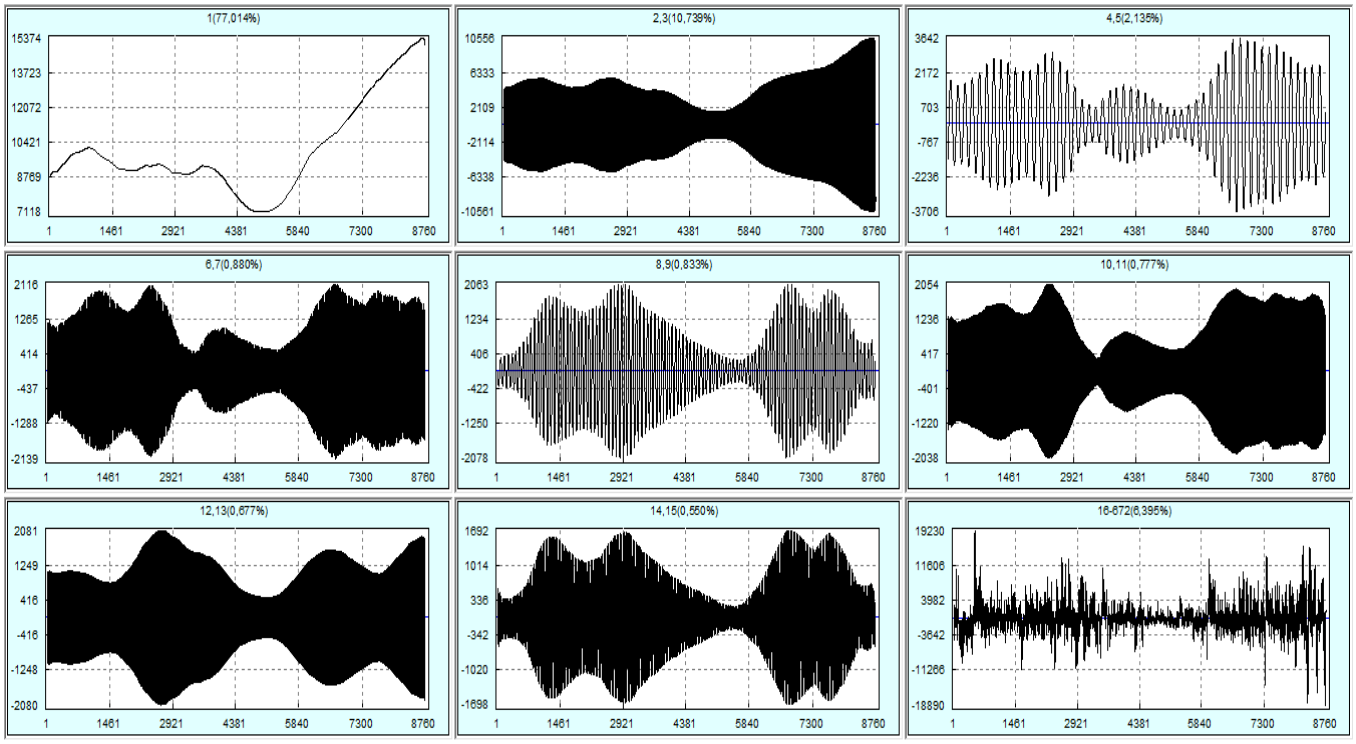


Рис. 4. Графіки компонент вихідного часового ряду після етапу групування методу "Гусениця-SSA"

Періодограми компонент ряду, отриманих після етапу групування (рис. 5) свідчать про те, що ми отримали в результаті окрім тренду (рис. 6) та шуму (рис. 5ж, 7) складові вихідного часового ряду з періодами 24 години (доба), 168 годин (тиждень), 28, 84 годин (півтижня), 21, 12 годин (півдоби) та 34 години.

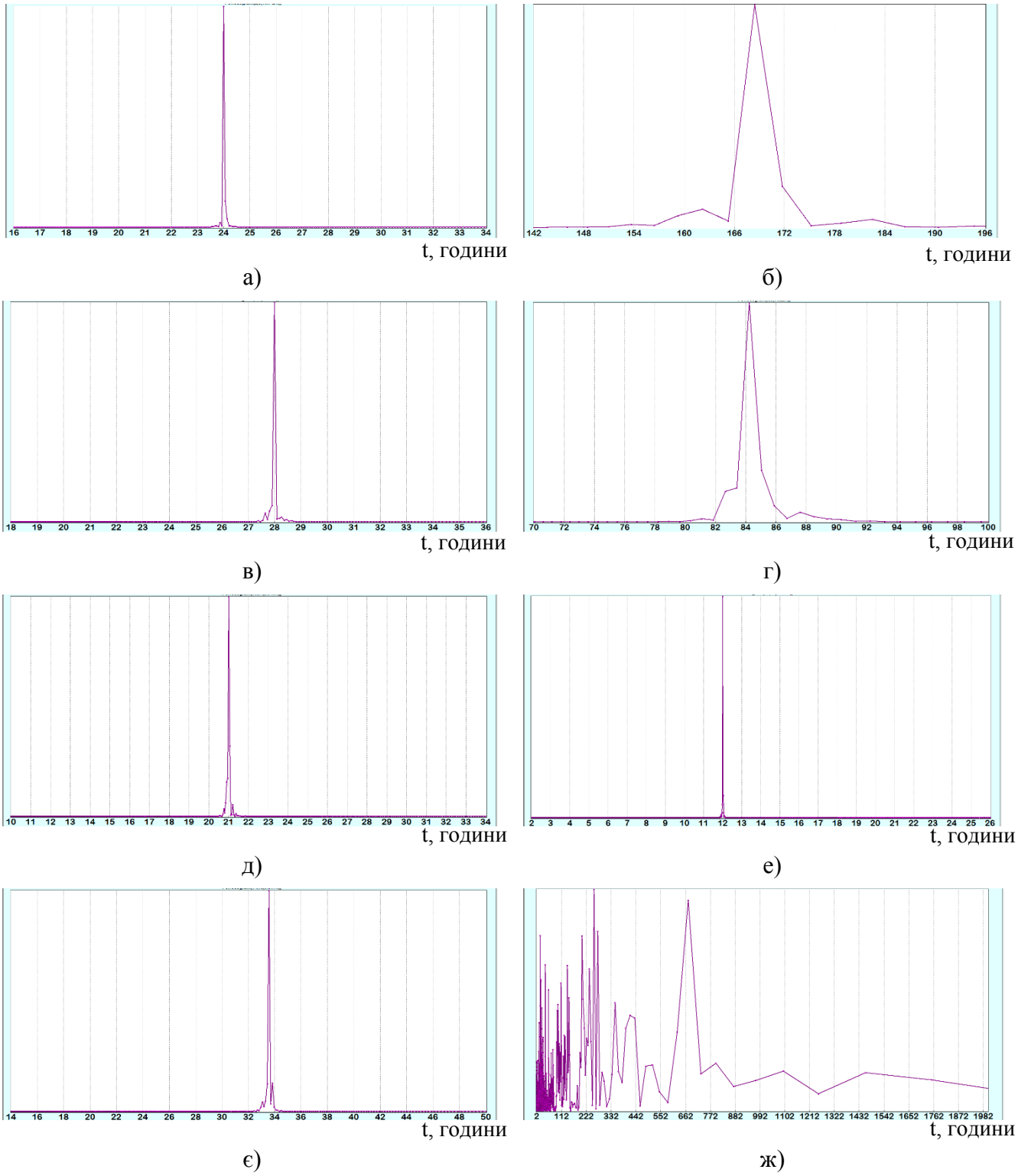


Рис. 5. Періодограми компонент часового ряду електронавантаження з періодами а) 24 години, б) 168 годин, в) 28 годин, г) 84 години, д) 21 година, е) 12 годин, є) 34 години, ж) з різними періодами (шум)

Останнім етапом методу “Гусениця-SSA” є відновлення вихідного ряду шляхом сумування його складових. Розглянемо декілька графіків. Після відновлення ряду лише на основі першої компоненти (річного тренду) отримаємо графіки, наведені на рис. 8. Якщо ж для відновлення ряду використати всі компоненти, отримані після групування, крім шумової компоненти, то отримаємо графіки, наведені на рис. 9 та 10.

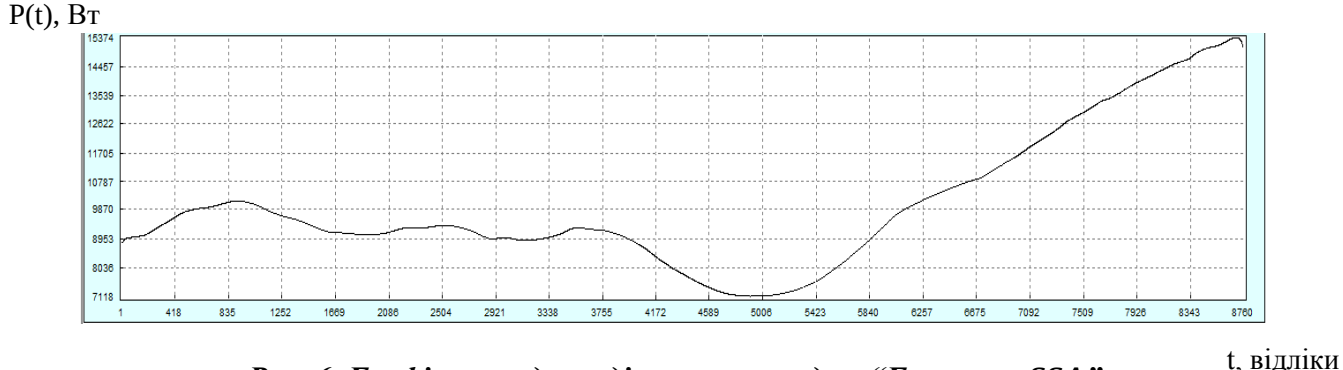


Рис. 6. Графік тренду, виділеного методом "Гусениця-SSA"

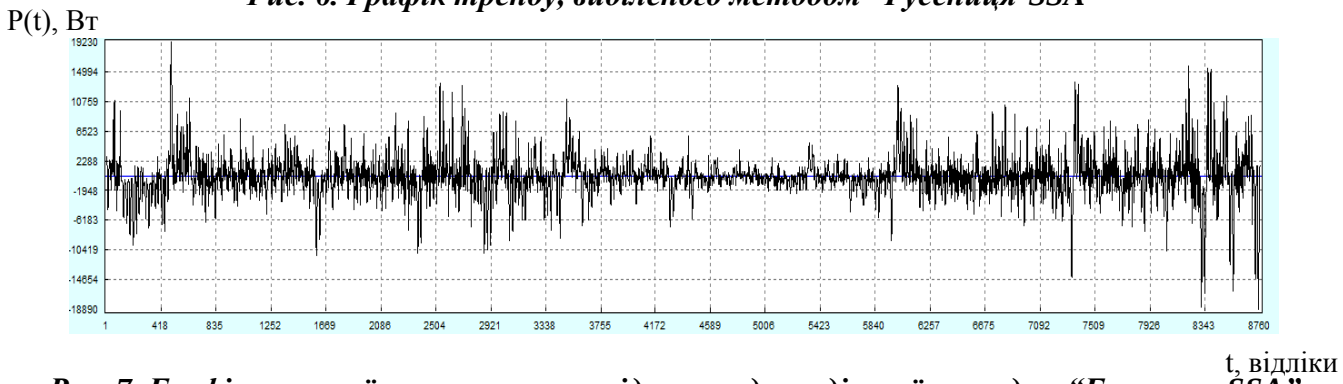


Рис. 7. Графік шумової компоненти вихідного ряду, виділеної методом "Гусениця-SSA"

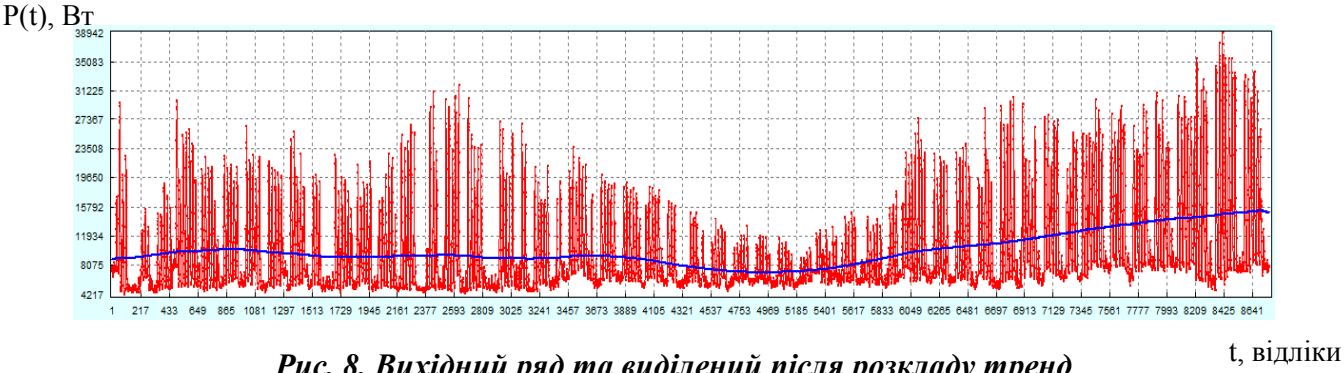


Рис. 8. Вихідний ряд та виділений після розкладу тренд

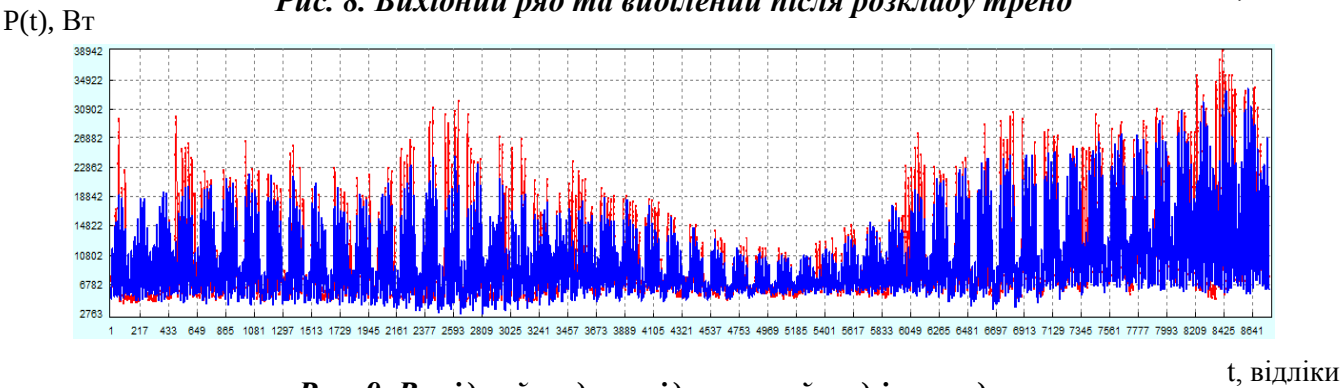


Рис. 9. Вихідний ряд та відновлений ряд із складових

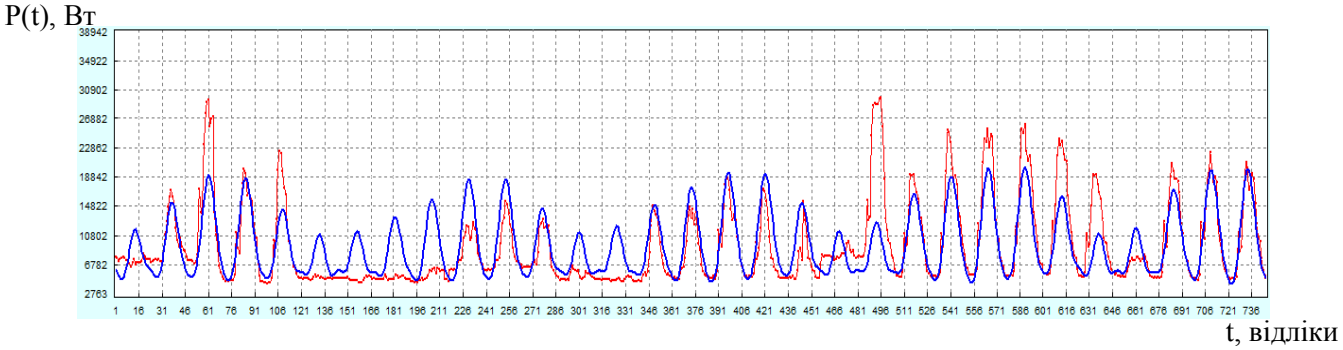


Рис. 10. Фрагмент графіків вихідного ряду та відновленогоряду (впродовж січня 2007 року)

Таким чином, у результаті розкладу вихідного часового ряду було отримано коливні складові з періодами доба і тиждень, що відповідає періодичності графіку роботи університету. Річний тренд демонструє загальну сумарну динаміку розвитку процесу протягом року, а шумова компонента – сукупність багатьох випадкових факторів, які впливають на досліджуваний процес електронавантаження.

Висновки

За допомогою методу “Гусениця-SSA” у роботі здійснено розклад часового ряду, який відповідає електронавантаженню корпусу №1 ТНТУ ім. Івана Пулюя впродовж 2007 року. Виділено та проаналізовано компоненти часового ряду а саме: річний тренд (неперіодичну і негармонічну складову, яка повільно змінюється в часі й характеризує динаміку зміни процесу в часі протягом року), а також коливні гармонічні компоненти, які відповідають складовим процесу з різними періодами (24 години і 168 годин), що дають оцінку графіку роботи установи у зрізі електронавантаження корпусу. Також виділено шумову компоненту, яка відповідає сукупності випадкових факторів, що впливають на формування процесу електронавантаження.

Щодо виокремлених компонент із періодами 28, 84, 21, 12, 34 години, варто зауважити, що для пояснення величини цих періодів необхідно провести додаткові дослідження особливостей роботи університету, топології користувачів електромережі корпусу №1 тощо. Власне, це надежить до перспектив подальших досліджень, започаткованих у цій роботі.

Перелік посилань

1. Драган Я. П. Линейные периодически коррелированые случайные процессы / Я. П. Драган, Н. В. Приймак. – Львов, 1986. – 30 с. – (Препринт / АН УССР, Физико-механический ин.-т, № 120).
2. Марченко Б. Г. Побудова моделі та аналіз стохастично періодичних навантажень енергосистем / Б. Г. Марченко, М. В. Приймак // Праці ін-ту електродинаміки. – К. : ІЕД НАН України, 1999. – Вип. 1. – С. 129-153.
3. Приймак М. В., Мацюк О. В. Моделі газонавантажень з врахуванням стохастичної періодичності та можливості їх статистичного аналізу // Розвідка та розробка нафтових і газових родовищ. Всеукраїнський щоквартальний науково-технічний журнал. Івано-Франківськ:. – 2003. – № 2(7).
4. Слуцкий Е. Е. Избранные труды / Е. Е. Слуцкий. – М.: Наука, 1960. – 292 с.
5. Марченко Б. Г. Визначення основних ймовірнісних характеристик газоспоживання / Б. Г. Марченко, Н. В. Мулик, М. Є. Фриз // Вимірювальна та обчислювальна техніка в технологічних процесах. – 2005. – №2.
6. Asteriou, Dimitros; Hall, Stephen G. (2011). "ARIMA Models and the Box–Jenkins Methodology". Applied Econometrics (Second ed.). Palgrave MacMillan. pp. 265–286.
7. Ивахненко А. Г. Принятие решений на основе самоорганизации / А. Г. Ивахненко, Ю. П. Зайченко, В. Д. Димитров. – М. : «Сов. радио», 1976. – 280 с.
8. Abdi H., Williams L. J. (2010). Principal component analysis.. Wiley Interdisciplinary Reviews: Computational Statistics, 2. с. 433–459.
9. Hilbert-Huang transform: [Електронний. ресурс]. – Режим доступу: http://www.scholarpedia.org/article/Hilbert-Huang_transform – Назва з екрану. – Дата звернення: 26.07.2015
10. Голяндина Н. Э. Метод “Гусеница”-SSA: анализ временных рядов: Учеб. пособие. СПб: Изд-во СПбГУ, 2004. 76 с.

Відомості про авторів

Амонс О.А.	к.т.н., доцент кафедри АУТС НТУУ "КПІ"
Барна В.В.	студент кафедри АУТС НТУУ "КПІ"
Білорус Д.С.	програміст, ТОВ "Неткрекер"
Болдак А.О.	к.т.н., доцент кафедри ОТ НТУУ "КПІ"
Букасов М.М.	к.т.н., доцент кафедри АУТС НТУУ "КПІ"
Великий М.М.	студент кафедри ОТ НТУУ "КПІ"
Воробйова А. Л.	студент кафедри ОТ НТУУ "КПІ"
Габинет А. В.	аспірант кафедри ОТ НТУУ "КПІ"
Галушко Д.А.	асистент кафедри АУТС НТУУ "КПІ"
Гапак О.М.	к.пед.н., доцент кафедри комп'ютерних систем та мереж Ужгородського національного університету
Готович В.А.	інженер 1-ої категорії Інженерно-технічної групи з обслуговування комп'ютерної техніки ТНТУ ім. Івана Пулюя
Гой А.А.	студент кафедри АУТС НТУУ "КПІ"
Діброва М.А.	аспірант кафедри ОТ НТУУ "КПІ"
Дорога-Іванюк О.О.	асистент кафедри АУТС НТУУ "КПІ"
Дорогий Я.Ю.	к.т.н., доцент кафедри АУТС НТУУ "КПІ"
Карнаухов О.К.	інженер, Тернопільський національний технічний університет ім. І.Пулюя
Клименко І.А.	к.т.н., доцент кафедри ОТ НТУУ "КПІ"
Коган А.В.	асистент кафедри АСОІУ НТУУ "КПІ"
Кравець П.І.	к.т.н., доцент кафедри АУТС НТУУ "КПІ"
Лисецький Т.М.	пошукач кафедри АСОІУ НТУУ "КПІ"
Мазуревич Ю.В.	студент кафедри ОТ НТУУ "КПІ"
Марковський О.П.	к.т.н., доцент кафедри ОТ НТУУ "КПІ"
Мельников О.В.	к.т.н., н.с. НДІ Інформаційних процесів НТУУ "КПІ"
Мінухін С.В.	к.т.н., професор кафедри інформаційних систем Харківського національного економічного університету імені Семена Кузнеця
Місюра О.Б.	к.т.н., с.н.с. НДІ Інформаційних процесів НТУУ "КПІ"
Моргун М.В.	студент кафедри АУТС НТУУ "КПІ"
Муха І.П.	к.т.н., доцент кафедри АСОІУ НТУУ "КПІ"
Назаревич О.Б.	асистент кафедри Комп'ютерних наук ТНТУ ім. Івана Пулюя
Павлов О.А.	д.т.н., професор кафедри АСОІУ НТУУ "КПІ"
Порєв В.М.	к.т.н., ст. викладач кафедри ОТ НТУУ "КПІ"
Ролік О.І.	д.т.н. професор кафедри АУТС НТУУ "КПІ"
Сперкач М.О.	асистент кафедри АСОІУ НТУУ "КПІ"
Теленик С.Ф.	д.т.н., проф., завідувач кафедри АУТС НТУУ "КПІ"
Терещенко С.Ю.	студент кафедри ОТ НТУУ "КПІ"
Томащук А.В.	студент кафедри АУТС НТУУ "КПІ"
Троцький С.О.	аспірант кафедри АУТС НТУУ "КПІ"
Федорчук В.В.	студент кафедри АУТС НТУУ "КПІ"
Ференс Д.А.	студент кафедри АУТС НТУУ "КПІ"
Філімонов В.Ф.	студент кафедри АУТС НТУУ "КПІ"
Шимкович В.М.	аспірант кафедри АУТС НТУУ "КПІ"
Ясочка М.В.	к.т.н., асистент кафедры АУТС НТУУ "КПІ"

SUMMARY

Kravets P.I., Shymkovych V.M., Fedorchuk V.V., Goy A.A. Neural controller stability of moving object with the hardware and software realization on FPGA…….……..... 4

Sperkach M. The problem of defining the maximum start moment of execution of tasks with a common tough due date on parallel machines with different speeds…….…. 12

Markovskyi O.P.,Tereshchenko S.Y. Method for double errors correction in digital transmission channel with spectral modulation……..……... 19

Rolik O., Halushko D., Barna V., Tomashchuk A., Yasochka M. Evaluation of the quality of the multimedia services by using neural network classifier……. 25

Gabinet A. Analysis of the protocols effectiveness to access video data storages…….……..… 31

Telenyk S., Amons O., Trotskiy O., Bilorus D. Automation of supporting business processes in the web environment……..……...…….. 35

Boldak A., Mazurevych Y. Optimization method of display of graphic information in a client-server information system……..……... 43

Telenyk S., Bukasov M., Karnauhov O., Filimonov V., Morhun N. Optimization models for multi-tier data storage……....……... 48

Minukhin S.V. Methods of distribution of tasks between processors of a computing cluster using frequency and voltage scaling…….…..……...……. 54

Dibrova M.A., Kogan A.V., Vorobyova A.L. Method of forming a plurality of paths in the network data center…….……...……..….... 60

Markovskyi O.P.,Velikiy M.M. Method for backup and shortcut restoring data in system remote information storage……..……. 65

Pavlov A.A., Misura E.B., Melnikov O.V., Mukha I.P., Lisetsky T.N. Algorithmization of the third and fourth levels of the four-level model of scheduling, operational planning and decision making in networked systems with limited resources……... 72

Klymenko I.A. The method of optimization reconfiguration for the dynamic reconfigurable computer……..…. 93

Dorohiy I.I., Doroha-Ivaniuk O.O., Ferens D.A. Implementation of neural network structural optimization algorithm…….. 101

Porev V.M. Raster heuristic algorithm modeling the dynamics of the spread for GIS…… 107

Hapak O.M. Evaluation of the quality of the hollmann generator, realized on the basis of FCSR…….. 119

Gotovych V.A., Nazarevych O.B. Application of the method "Caterpillar-SSA" to analyze time series of annual electricity load of organization……. 123

Information about authors……...……...….... 130

Summary…….…...……..……. 131