Національна академія наук України
Інститут проблем моделювання в енергетиці ім. Г.Є. Пухова

**Сушко Сергій Володимирович**

УДК 004.89:004.4

# МЕТОДИ ОПТИМАЛЬНОГО РОЗПАРАЛЕЛЮВАННЯ ПРОГРАМ МІКРОПРОЦЕСОРНИХ СИСТЕМ ДЛЯ ПІДВИЩЕННЯ ЇХ ЕФЕКТИВНОСТІ

05.13.05 – комп’ютерні системи та компоненти

**Автореферат**
дисертації на здобуття наукового ступеня
кандидата технічних наук

**Київ – 2021**

Дисертацією є рукопис.

Робота виконана в Інституті проблем моделювання в енергетиці ім. Г.Є. Пухова Національної академії наук України

**Науковий керівник:** доктор технічних наук,
старший науковий співробітник
**Чемерис Олександр Анатолійович**,
Інститут проблем моделювання в енергетиці
ім. Г.Є. Пухова НАН України,
заступник директора з наукової роботи.

**Офіційні опоненти:** доктор технічних наук, професор
**Новотарський Михайло Анатолійович,**
Національний технічний університет України «Київський політехнічний інститут імені Ігоря Сікорського», професор кафедри обчислювальної техніки;

кандидат технічних наук, доцент
**Кудерметов Равіль Камілович,**
Національний університет «Запорізька політехніка», завідувач кафедри комп'ютерних систем та мереж.

Захист відбудеться "14" травня 2021 року о 12 годині на засіданні спеціалізованої вченої ради Д 26.185.01 Інституту проблем моделювання в енергетиці ім. Г.Є. Пухова НАН України за адресою: 03164, м. Київ, вул. Генерала Наумова, 15.

З дисертацією можна ознайомитись в бібліотеці Інституту проблем моделювання в енергетиці ім. Г.Є. Пухова НАН України за адресою: 03164, м. Київ, вул. Генерала Наумова, 15.

Автореферат розісланий "10" квітня 2021 р.

Вчений секретар
спеціалізованої вченої ради В.В. Душеба.

## ЗАГАЛЬНА ХАРАКТЕРИСТИКА РОБОТИ

**Актуальність теми.** Потужність обчислювальних систем стрімко розвивається використовуючи новітні розробки у галузі апаратних компонентів. Водночас, також розвиваються обчислювальні алгоритми та методи обчислень.

Задля максимально ефективного використання апаратних можливостей постійно вдосконалюються компілятори, створюються нові мови програмування та збагачуються новим синтаксисом існуючи мови програмування.

Аналіз літературних джерел показав, що існує дуже багато засобів і методів оптимізації, які використовуються на різних рівнях оптимізації. Найбільше методів та алгоритмів оптимізації використовується в компіляторах. Недоліком вдосконалення оптимізаційних можливостей компіляторів є обмеженість налаштувань оптимізації компілятора та прихована логіка його роботи. В той же час окремим напрямком досліджень щодо оптимізації програмного коду є перетворення вихідного коду в інший вихідний код (S2S перетворення). Такий підхід дозволяє зосередитися на одному або декількох методах оптимізації та водночас може використовуватися сумісно з іншими засобами оптимізації, наприклад він доповнює оптимізаційні методи компіляторів.

Проблемою оптимізації програмного забезпечення займались і займаються як науково-академічні організації так і науково-дослідні відділи виробників програмного забезпечення, як в Україні, так і в світі. Варто виділити великий внесок в теорію і практику розпаралелювання програм таких науковців, як, зокрема, П. Фотріє, А. Фрабле, В. П'ю, М. Лем, А. Дарта, Вл.В. Воєводіна, Н.А. Лиходіда, В.А. Вальковського, І.А. Каляєва, В.П. Гергеля, В.Г. Хорошевського, Ю.В. Капітонової, О.А. Летичєвського, В.М. Бєлецького, А.Ю. Дорошенка та інших.

Різноманіття методів оптимізації, можливість або неможливість використання одночасно декількох методів оптимізації складає множину методів оптимізації. Деякі з методів оптимізації мають власні параметри, які також впливають на ефективність оптимізації. Правильний підбір набору методів оптимізацій та значень їх параметрів може додатково підвищити ефективність оптимізації. Деякі параметри оптимізації явно задаються компілятору, деякі встановлюються на етапі попереднього аналізу коду або евристично. Незважаючи на вагомі результати досліджень, в них не завжди приділяється достатньо уваги саме вибору найбільш ефективного методу та його параметрів для кожного конкретного прикладу. Саме тому підбір параметрів оптимізаційного методу для конкретної частини програмного коду є актуальною задачею для пошуку найбільш ефективного результату оптимізації. В даній роботі розглядається підвищення ефективності методів розбиття на блоки та розпаралелювання та пропонуються підходи щодо підбору найкращих розмірів блоків розбиття для методу розбиття на блоки.

Дана робота досліджує метод розбиття на блоки, що виконує S2S перетворення універсального програмного коду мовами програмування C або C++, який може виконуватися швидше та/або більш енергоефективно. Розглядається ефективність даного методу на різноманітних тестових даних. Пропонується спосіб підбору параметрів методу, що призводить до максимальної ефективності методу оптимізації з точки зору часу обчислень. Запропоновано використання дискретного

методу рою часток в задачі пошуку кращих розмірів блоків розбиття. Отриманий метод дістав назву метод інтелектуального блочного розбиття.

**Зв'язок роботи з науковими програмами, планами, темами.** Дисертаційну роботу виконано у межах відомчих науково-дослідних тем НАН України, а саме: «Розвиток методів зниження енергоспоживання обчислювальних систем за рахунок оптимізації обробки масивів даних» (шифр – ФРІСК)» (номер державної реєстрації 0114U000879) – проведено дослідження зниження енергоспоживання при проектуванні мікропроцесорних систем за рахунок ефективного розпаралелювання циклів; «Розвиток теорії, розробка новітніх інформаційних технологій в задачах комплексного моделювання та управління процесами перетворення та використання енергії (шифр: НОВІНТЕХ)» (номер державної реєстрації 0117U004347) – розроблено метод визначення оптимальних розмірів прямокутних блоків розбиття ітераційного простору операторів циклу мікропроцесорних програм на основі дискретного методу рою часток; «Розвиток теорії, розробка методів та засобів реалізації гібридних експертно-моделюючих комп'ютерних систем в задачах комплексного управління перетворенням енергії (шифр – ГІБРИД)» (номер державної реєстрації 0112U000050) – проведено дослідження щодо комбінування різних методів оптимізації програм з метою вибору найкращої в плані часу виконання програми комбінації.

Дисертаційна робота виконана в Інституті проблем моделювання в енергетиці ім. Г.Є. Пухова Національної академії наук України.

**Метою** роботи є підвищення ефективності функціонування мікропроцесорних систем за рахунок розпаралелювання та оптимізації програмного забезпечення.

Задача роботи полягає в доведенні ефективності методів та впровадження їх у програмні додатки.

**Основні задачі дослідження** відповідно до поставленої мети полягають у наступному:

1. Виконати аналіз методів оптимізації комп'ютерних програм, що спрямовані на підвищення ефективності обчислень (швидкодію, енергоефективність, використання пам'яті, тощо);
2. Вдосконалити метод оцінки часу виконання тестових програм при використанні різноманітних методів розбиття на блоки;
3. Розробити алгоритм та програму, що виконує в автоматичному режимі вимірювання часу виконання тестових програм;
4. Розробити алгоритм та програму, що виконує пошук мінімального часу виконання тестових програм для різних розмірів блоків розбиття використовуючи дискретний метод рою часток;
5. На основі результатів дослідження розробити метод пошуку найкращих розмірів блоків розбиття на базі дискретного методу рою часток;
6. Впровадити результати роботи.

**Об'єктом дослідження** є процес оптимізації програмного коду операторів циклів мікропроцесорних програм.

**Предметом дослідження** є комплекси автоматичної розробки програмного забезпечення мікропроцесорних систем.

**Методи дослідження** базуються на використанні теорії множин, методах трансформації обчислювальних циклів та еволюційних оптимізаційних методах.

**Наукова новизна одержаних результатів** полягає в наступному:

*Вперше:*

1. Запропоновано ітераційний процес розпаралелювання операторів циклів мікропроцесорних програм, який відрізняється від відомих методів визначенням оптимального рішення щодо розбиття ітераційного простору оператора циклу на окремі блоки.

2. Запропоновано оцінку доцільності оптимізації коду програм і отримано експериментальні дані, що засвідчують покращення ефективності обчислень (за часом або енергоспоживанням), в більшості випадків при застосуванні методу розбиття на блоки.

3. Розроблено метод інтелектуального блочного розбиття, що виконує пошук оптимальних розмірів прямокутних блоків розбиття ітераційного простору операторів циклу мікропроцесорних програм на основі дискретного методу рою часток, який може бути застосовано до довільних обчислювальних циклів написаних мовами програмування С або С++.

*Удосконалено:*

4. Метод оцінки часу виконання тестових програм при використанні різноманітних методів розбиття на блоки, який базується на автоматичному послідовному багаторазовому запуску тестових програм і фіксації отриманих результатів швидкодії, що можуть бути в подальшому проаналізовані.

5. Дискретний метод рою часток шляхом визначення коефіцієнтів методу, а саме – початкового коефіцієнту інерції, індивідуального та соціального коефіцієнтів, при яких пошук розмірів блоків розбиття в задачі прискорення швидкодії виконується швидше класичного методу.

6. Використання дискретного методу рою часток шляхом визначення кращих початкових даних для розташування часток рою, що зменшує число ітерацій для знаходження оптимального рішення.

**Практичне значення одержаних результатів** полягає в тому, що розроблений в дисертаційній роботі метод інтелектуального блочного розбиття може бути використаний для оптимізації програмного забезпечення написаного мовами програмування С або С++ широкого спектру застосування. Оскільки вказаний метод оптимізації використовує S2S перетворення, то безпосередньо алгоритм оптимізованої програми та використовуваний компілятор не мають значення. Запропонований метод оптимізації є універсальним відносно типу задач. Метод може бути використаний з будь-яким компілятором та апаратною платформою. Експериментальні дослідження виявили, що найбільш вагоме пришвидшення інструкцій досягається на апаратній платформі з великою кількістю кешу. На мікроконтролері, що не має кешу, прискорення не спостерігалось.

Алгоритм, розроблений у дисертаційній роботі був використаний для практичного удосконалення програмного забезпечення в ТОВ «Дельта СПЕ».

**Особистий внесок здобувача.** Усі основні положення та результати дисертаційної роботи отримані автором самостійно. У роботах, виконаних у

співавторстві, автору належать такі результати: у роботі [1] запропоновано методи вимірювання та виконано виміри часу виконання та енергоспоживання тестових програм на платформі Raspberry Pi 3 для різних методів розбиття; у статті [2] виконано виміри часу виконання та енергоспоживання тестових програм на платформі x64 для різних методів розбиття, введено поняття коефіцієнту енергоефективності та отримано значення коефіцієнтів енергоефективності; у статті [3] запропоновано використання різноманітних розмірів блоків розбиття з метою визначення їх впливу на швидкодію тестових програм; в [4] проведено аналіз використання різноманітних блоків розбиття; в публікації [5] визначено складний характер впливу параметрів розбиття та запропоновано напрямки для визначення таких параметрів; в праці [6] виконано дослідження та аналіз енергоефективності обчислень на різних апаратних платформах.

**Апробація результатів роботи**. Основні положення та результати дисертаційної роботи доповідалися і обговорювалися:

на науково-технічній конференції молодих вчених та спеціалістів ІПМЕ ім. Г.Є. Пухова НАН України (м. Київ, 2016), на Міжнародній конференції Моделювання-2016 (м. Київ, 2016), на Міжнародній науково-практичній конференції Summer InfoCom Advanced Solutions 2016 (м. Київ, 2016), на Міжнародній науково-практичній конференції Winter InfoCom Advanced Solutions 2016 (м. Київ, 2016), на міжнародній конференції IEEE International Conference on Electronics and Nanotechnology (ELNANO) (м. Київ, 2017), на Зимовій школі ALIOT (м. Чернівці, 2018), на науково-технічній конференції молодих вчених та спеціалістів ІПМЕ ім. Г.Є. Пухова НАН України (м. Київ, 2018), на Міжнародній конференції IEEE International Conference on Dependable Systems, Services and Technologies (DESSERT-2018) (м. Київ, 2018), на Міжнародній конференції Моделювання-2018 (м. Київ, 2018), на Міжнародній конференції Reconfigurable Ubiquitous Computing 2018 (м. Дзівнув, Польща, 2018), на Міжнародній науково-практичній конференції комп'ютерні системи та мережні технології (м. Київ, 2019), на Міжнародній конференції IEEE International Scientific-Practical Conference Problems of Infocommunications, Science and Technology (PIC S&T-2019) (м. Київ, 2019), на Науково-практичній конференції «Безпека енергетики в епоху цифрової трансформації» (м. Київ, 2019).

**Публікації**. За результатами виконаних досліджень опубліковано 16 наукових праць, серед яких: 1 стаття у періодичних наукових виданнях інших держав, 1 розділ в колективній монографії, що індексується міжнародною наукометричною базою Scopus, 4 наукових статті (у тому числі 3 статті у наукових фахових виданнях), 8 – у збірниках матеріалів міжнародних конференцій, з яких 3 індексуються міжнародною наукометричною базою Scopus та 2 у збірниках матеріалів науково-практичних конференцій.

**Структура та обсяг дисертаційної роботи.** Дисертація складається з анотації, вступу, чотирьох розділів, висновків, списку використаних джерел, що містить 155 найменувань і 6 додатків. Дисертація має 126 сторінок основного тексту, 74 рисунки, 15 таблиць, 35 сторінок додатків. Загальний обсяг дисертації становить 161 сторінку.

## ОСНОВНИЙ ЗМІСТ РОБОТИ

У **вступі** обґрунтовано актуальність роботи, визначено об’єкт і предмет дослідження, сформульовано мету і завдання, визначено наукову новизну та практичну цінність отриманих результатів.

У **першому розділі** виконано загальний аналіз проблеми енергоспоживання обчислювальних пристроїв. Наведені основні складові частини в структурі енергоспоживання, фактори, що впливають на енергоспоживання обчислювальних систем.

Сформульовано проблему енергоспоживання електронних пристроїв на рівні транзисторів як функцію від багатьох змінних, що включає як один із параметрів частоту перемикання транзистора. Аналіз формул залежності енергоспоживання дав змогу стверджувати, що зменшення кількості операцій, а отже і кількості перемикань транзисторів зменшить енергоспоживання під час обчислень.

Проведено порівняльний аналіз загальних підходів, що використовуються для зменшення енергоспоживання апаратно-програмних комплексів.

В даному розділі розглянуто загальні підходи щодо оптимізації комп’ютерних програм (КП).

Розглянуто загальні підходи та сучасні методи оптимізації обчислювальних циклів. Визначено основні проблеми оптимізації КП такі як:

- Глибока оптимізація програмного забезпечення потребує високої кваліфікації розробника програмного забезпечення;
- Більшість алгоритмів задана для однопотокового процесу, в той час як сучасні мікропроцесори багатоядерні, і адаптація одноядерних програм для ефективного використання на багатоядерному мікропроцесорі потребує складної перебудови алгоритму;
- Наявність великої кількості методів оптимізації та їх параметрів;
- Деякі програми, такі як декодування звуку, відео, обробка пакетів даних згідно мережевих протоколів, бездротової передачі даних, супутникового зв'язку, мають бути обчислені за визначений короткий проміжок часу;
- Наявність задач, в яких виникає необхідність виконання великої кількості операцій над великою кількістю вхідних даних;
- Використання малопотужного апаратного забезпечення, що має дуже обмежені обчислювальні ресурси.

Наведена трирівнева ієрархія методів оптимізації КП дозволяє найбільш повно використовувати всі наявні ресурси для покращення обраних параметрів програм. Рівні методів оптимізації:

1. оптимізація алгоритму;
2. оптимізація послідовності програм;
3. мікрокомандна оптимізація.

Обґрунтовується важливість оптимізації на кожному рівні. Особлива увага приділяється оптимізації послідовності команд, що дозволяє не змінюючи алгоритм програми та компілятор досягати зниження використаних ресурсів, необхідних для роботи програми.

Автором наводяться методи оптимізації обчислювальних циклів КП, що спрямовані на прискорення швидкодії. Пояснюються засоби, за рахунок яких досягається пришвидшення виконання КП.

Наведено дані, що показують значний та складнопрогнозований ефект від оптимізації програмного забезпечення (ПЗ) при використанні різних опцій компілятора.

У **другому розділі** розглядаються практичні методи оптимізації ПЗ. В якості підходу до реалізації оптимізації, обґрунтовується доцільність застосування класу оптимізацій по типу S2S перетворення, що дає незалежність від апаратної платформи. Порівняння традиційного підходу та обраного для реалізації наводиться на Рисунку 1.

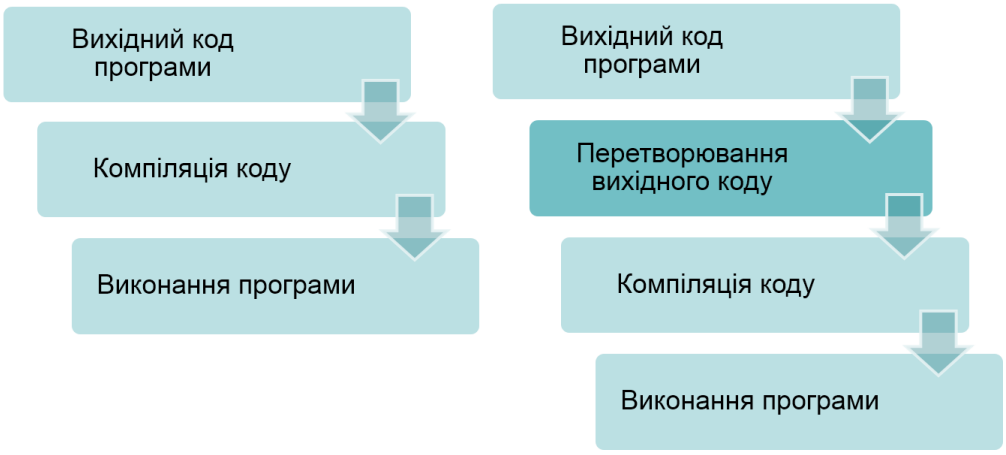


Рисунок 1 – Порівняння традиційної компіляції програм та компіляції з оптимізацією вихідного коду

Зазначається, що такий підхід розширює можливості по оптимізації та дозволяє використовувати різноманітні методи оптимізації без створення або змінення компіляторів. Також наводяться аргументи, що спосіб оптимізації, що базується на S2S перетворенні є перспективним з точку зору апаратної незалежності. Обґрунтовується можливість отримання синергетичного ефекту від одночасного використання оптимізаційних методів S2S перетворення та оптимізаційних методів компілятора, що призведе до покращення цільової функції відносно використання тільки оптимізаційних методів компілятора. Наводяться переваги та недоліки S2S перетворення.

Переваги:

- апаратна незалежність;
- доповнюють оптимізаційні методи компілятора;
- розробка одного конкретного методу потребує значно менше зусиль ніж розробка компілятора з нуля;
- наявність наукових досліджень та напрацювань у відкритому доступі;
- можуть бути застосовані незалежно від використаного компілятора.

Недоліки:

- реалізація оптимізаційного методу в існуючому компіляторі швидша за створення реалізації того ж самого методу шляхом S2S перетворення.

В розділі розглядається поліедральна модель – математична модель комп'ютерних програм, що забезпечує компактне відображення великої кількості операцій. Вкладені обчислювальні цикли програм є типовим прикладом використання моделі. Найпоширенішим використанням моделі є модифікація вкладених циклів при оптимізації програм. Пояснюються математичні методи, що використовуються для опису поліедральної моделі, приклади застосуванні при оптимізації циклів КП. Обґрунтовується можливий ефект від застосування.

В розділі вводиться поняття ітераційного простору, що є способом представлення обчислювальних циклів у вигляді *n*-мірного опуклого багатогранника, вузли якого являють собою кожну обчислювальну ітерацію циклу. Ітераційний простір є складовою частиною поліедральної моделі. Наводяться приклади обчислювальних циклів КП та їх ітераційний простір у математичному та графічному вигляді (див. Рисунок 2).

```
for(i1 = 1; i1 < N; i1++)
  for(i2 = 1; i2 < i1; i2++)
    for(i3 = 1; i3 < N; i3++)
    {operations}
```

*Ітераційний простір*

$$D = \begin{bmatrix} 1 & 0 & 0 & 0 & -1 \\ -1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & -1 \\ 1 & -1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & -1 \\ 0 & 0 & -1 & 1 & 0 \end{bmatrix} \begin{pmatrix} i1 \\ i2 \\ i3 \\ N \\ 1 \end{pmatrix} \geq 0 \begin{vmatrix} i1 \geq 1 \\ i1 \leq N \\ i2 \geq 1 \\ i2 \leq i1 \\ i3 \geq 1 \\ i3 \leq N \end{vmatrix}$$

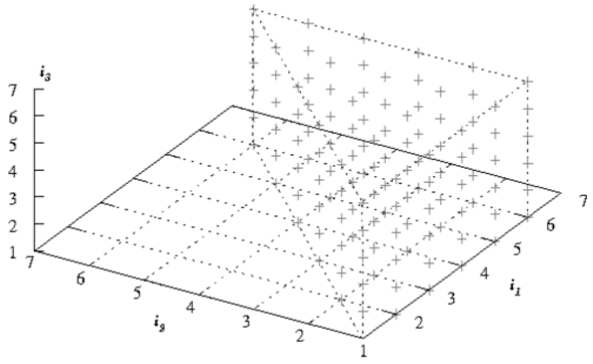


Рисунок 2 – Приклад ітераційного простору для потрійного обчислювального циклу

З метою прискорення швидкодії та енергоефективності обчислень, пропонується використання методу розбиття обчислювального циклу на блоки (МРОЦНБ, tiling method) та метод розпаралелювання. Суть МРОЦНБ в розбитті великого ітераційного простору на один або декілька блоків розбиття (tile) меншого розміру. Приклад такого розбиття наведено на Рисунку 3.

МРОЦНБ покращує локальність даних, а також надає можливість виконання нових блоків на різних ядрах мікропроцесора, що сприятиме ефективності розпаралелювання КП. Однак ефективне розпаралелювання можливе тільки при відсутності залежностей між операціями різних блоків.

Автор звертає увагу, що МРОЦНБ останнім часом отримав достатній розвиток в дослідницькій сфері і для задачі верифікації методу автор використовує пакет програм Pluto. Цей пакет включає низку програмних модулів, що почергово спочатку створюють поліедральну модель на основі вихідного коду мовою С, потім власне сам пакет Pluto модифікує цю поліедральну модель згідно заданих методів МРОЦНБ та паралелізму, далі інші програмні модулі спрощують отриману модель, і на останньому етапі програмні модулі створюють вихідний код мовою програмування С по отриманій моделі.

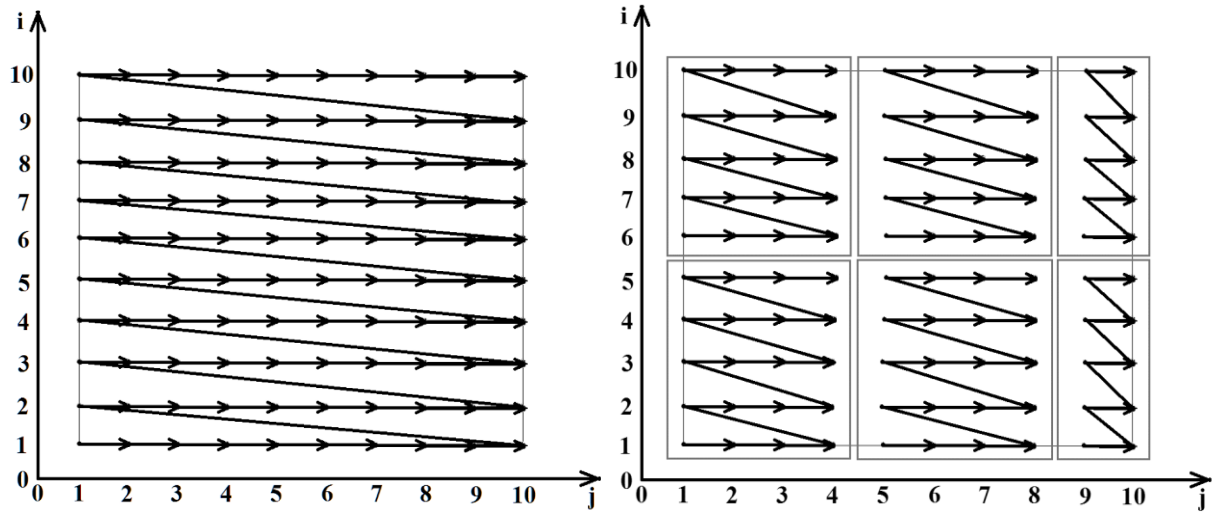


Рисунок 3 – Приклад початкового ітераційного простору та модифікованого ітераційного простору із розбиттям на блоки розміром 5x4

Таким чином, програмний пакет Pluto є прикладом використання S2S перетворення. Структурна схема послідовності програмних модулів пакету Pluto представлена на Рисунку 4.

Для оцінки впливу методу розбиття на блоки на час виконання програм та на енергоспоживання було виконано чотири серії експериментів. Перші дві серії експериментів було виконано на компактному одноплатному комп'ютері Raspberry Pi 3 (чотириядерний центральний процесор ARM Cortex-A53, 1.2 ГГц, кеш L1 32KB, L2 512KB, пам'ять 1ГБ LPDDR2 900 МГц). Третя та четверта серії експериментів було виконано на сучасному ПК з процесором Intel (чотириядерний центральний процесор Intel® Core™ i5-4670K, 3.4 ГГц, кеш L1 4*64KB, L2 4*25.26KB, L3 6МБ, пам'ять 16ГБ DDR3 1333 МГц).

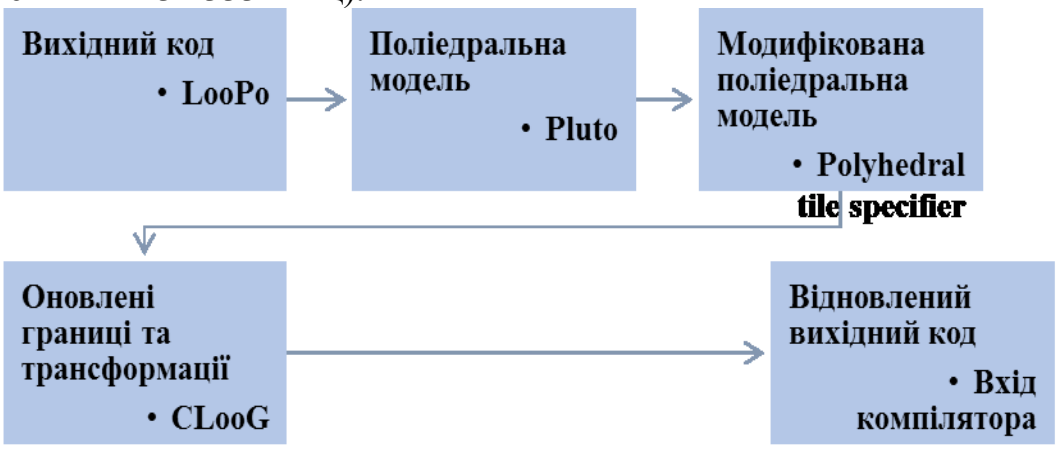


Рисунок 4 – Послідовність перетворення програми з використанням поліедральної моделі пакету Pluto

Задля прискорення часу обробки всіх тестових програм створено скрипт, що виконує послідовний запуск всіх тестових програм, обробку отриманих результатів та представлення часу виконання у зручному вигляді.

Результати прискорення часу виконання тестових програм для плати Raspberry Pi 3 та для ПК на базі чотириядерного процесора Intel® Core™ i5 4670K наведено в Таблицях 1 та 2 відповідно.

Таблиця 1 – Відносне прискорення часу виконання тестових програм для плати Raspberry Pi 3

| **Тестова програма** | **Опції оптимізації програми Pluto** | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | **Tile** | **Tile Parallel** | **Tile L2Tile Parallel** | **Innerpar** | **Innerpar Tile Parallel** | **Tile Multipar Parallel** | **Diamond-tile** | **Diamond-tile Parallel** |
| correlation | 1,71 | 3,44 | 1,80 | 1,96 | 3,57 | 2,30 | 1,95 | 3,62 |
| covariance | 1,81 | 3,71 | 1,72 | 1,83 | 3,74 | 2,26 | 1,75 | 3,78 |
| gemm | 1,09 | 2,15 | 2,14 | 1,03 | 2,15 | 1,69 | 1,02 | 2,22 |
| gemver | 3,56 | 7,14 | 7,78 | 3,75 | 7,40 | 4,67 | 2,76 | 7,45 |
| gesummv | 1,05 | 2,45 | 2,16 | 1,20 | 2,47 | 1,34 | 1,02 | 2,75 |
| symm | 1,02 | 0,99 | 1,07 | 1,03 | 1,07 | 1,01 | 1,04 | 0,98 |
| syr2k | 2,05 | 5,53 | 1,58 | 1,86 | 5,03 | 2,84 | 1,95 | 5,62 |
| syrk | 1,26 | 2,36 | 0,85 | 1,26 | 2,33 | 1,19 | 1,17 | 2,29 |
| trmm | 1,28 | 4,49 | 1,25 | 1,43 | 4,52 | 2,30 | 1,28 | 5,41 |
| 2mm | 1,11 | 2,44 | 0,89 | 0,90 | 2,44 | 1,65 | 1,09 | 2,48 |
| 3mm | 1,16 | 2,39 | 0,85 | 1,00 | 2,40 | 1,67 | 1,16 | 2,42 |
| atax | 0,64 | 1,41 | 1,46 | 0,97 | 1,42 | 1,04 | 0,64 | 1,46 |
| bicg | 0,66 | 1,47 | 1,46 | 1,02 | 1,48 | 1,01 | 0,66 | 1,51 |
| doitgen | 1,15 | 1,72 | 0,74 | 1,28 | 1,72 | 1,31 | 1,08 | 1,74 |
| mvt | 1,33 | 4,13 | 4,14 | 0,84 | 4,13 | 2,21 | 1,29 | 4,21 |
| cholesky | 0,88 | 1,95 | 0,77 | 0,90 | 2,25 | 1,26 | 0,95 | 2,08 |
| durbin | 0,98 | 0,98 | 1,01 | 1,07 | 0,98 | 1,00 | 1,01 | 0,99 |
| gramschmidt | 1,28 | 2,69 | 1,05 | 1,00 | 2,66 | 1,74 | 1,25 | 2,69 |
| lu | 1,76 | 4,49 | 1,82 | 1,84 | 4,62 | 2,78 | 1,65 | 3,95 |
| ludcmp | 1,11 | 1,10 | 1,05 | 1,04 | 1,16 | 1,03 | 1,04 | 1,05 |
| trisolv | 0,86 | 1,18 | 1,83 | 1,01 | 2,32 | 0,88 | 0,86 | 1,21 |
| deriche | 0,98 | 1,05 | 0,96 | 1,01 | 1,01 | 0,95 | 0,98 | 1,02 |
| floyd-warshall | 0,62 | 0,89 | 0,39 | 1,01 | 1,02 | 0,59 | 0,63 | 0,90 |
| nussinov | 1,03 | 3,13 | 1,12 | 1,00 | 2,51 | 1,85 | 1,01 | 2,96 |
| fdtd-2d | 1,16 | 2,24 | 0,98 | 1,11 | 3,28 | 1,77 | 1,00 | 1,00 |
| heat-3d | 0,96 | 0,95 | 1,00 | 1,07 | 1,85 | 1,09 | 1,00 | 1,00 |
| jacobi-2d | 1,04 | 1,73 | 1,00 | 0,99 | 2,60 | 1,56 | 1,04 | 1,00 |
| seidel-2d | 0,99 | 2,05 | 1,00 | 0,99 | 2,72 | 1,52 | 1,02 | 2,06 |

Друга та четверта серії експериментів присвячена виміру енергоспоживання тих самих тестових програм при тих самих методах розбиття на блоки та розпаралелюванні на вказаних апаратних платформах.

Методика виміру енергоспоживання відрізняється від методики виміру часу виконання. Оскільки час роботи програм часто становить менше секунди, ватметр не в змозі виконати вимір за такий короткий час. Тому для виміру енергоспоживання використовуються модифіковані тестові програми у яких обчислювальний цикл, енергоспоживання якого вимірюється, додано в середину нескінченого циклу. Такий підхід дозволяє отримати стале енергоспоживання тестової програми, обчислювальний цикл якої може виконуватися навіть за дуже короткий час.

Таблиця 2 – Відносне прискорення часу виконання тестових програм для ПК на базі процесора Intel® Core™ i5 4670K

| **Тестова програма** | **Опції оптимізації програми Pluto** | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | **tile** | **tile parallel** | **tile l2tile parallel** | **innerpar** | **innerpar parallel** | **innerpar tile parallel** | **tile multipar parallel** | **diamond -tile** | **diamond -tile parallel** |
| correlation | 4,91 | 11,18 | 3,93 | 4,63 | 7,78 | 11,43 | 6,32 | 5,01 | 10,85 |
| covariance | 4,99 | 10,86 | 3,85 | 4,70 | 9,63 | 10,87 | 6,47 | 4,92 | 10,75 |
| gemm | 2,31 | 9,70 | 11,62 | 1,44 | 3,88 | 9,51 | 4,97 | 2,29 | 9,39 |
| gemver | 5,17 | 16,10 | 16,08 | 14,39 | 22,07 | 15,90 | 9,12 | 5,03 | 14,37 |
| gesummv | 0,89 | 2,83 | 2,40 | 1,68 | 4,63 | 2,98 | 1,64 | 0,89 | 2,78 |
| symm | 1,05 | 1,03 | 1,05 | 1,02 | 1,02 | 1,05 | 1,05 | 1,05 | 1,05 |
| syr2k | 1,41 | 5,26 | 1,44 | 1,42 | 5,63 | 5,22 | 2,64 | 1,32 | 5,20 |
| syrk | 0,99 | 2,85 | 1,08 | 0,99 | 2,24 | 2,14 | 1,43 | 1,00 | 2,85 |
| trmm | 3,43 | 17,52 | 3,62 | 5,23 | 1,22 | 17,38 | 9,06 | 3,46 | 16,43 |
| 2mm | 1,94 | 5,61 | 1,86 | 1,61 | 6,14 | 5,56 | 3,75 | 1,95 | 5,54 |
| 3mm | 1,63 | 4,71 | 1,57 | 1,31 | 5,03 | 4,75 | 3,14 | 1,63 | 4,73 |
| atax | 0,40 | 1,26 | 1,27 | 0,99 | 1,45 | 1,24 | 0,69 | 0,39 | 1,25 |
| bicg | 0,82 | 2,64 | 2,52 | 2,12 | 2,85 | 2,56 | 1,46 | 0,77 | 2,00 |
| doitgen | 5,02 | 4,00 | 3,43 | 6,00 | 0,25 | 4,03 | 5,22 | 5,03 | 3,90 |
| mvt | 3,78 | 11,32 | 12,84 | 1,16 | 3,68 | 11,48 | 7,16 | 3,58 | 9,99 |
| cholesky | 1,26 | 2,98 | 1,16 | 1,00 | 1,80 | 3,09 | 1,62 | 1,27 | 3,02 |
| durbin | 1,00 | 1,00 | 0,99 | 0,99 | 1,00 | 1,00 | 1,00 | 1,00 | 1,00 |
| gramschmidt | 1,77 | 3,13 | 1,40 | 1,04 | 2,85 | 3,13 | 2,58 | 1,77 | 3,06 |
| lu | 1,49 | 3,80 | 1,74 | 1,56 | 1,40 | 4,45 | 2,53 | 1,50 | 3,73 |
| ludcmp | 1,00 | 1,00 | 1,00 | 0,99 | 1,00 | 1,00 | 0,98 | 0,96 | 0,99 |
| trisolv | 0,71 | 1,79 | 1,57 | 1,22 | 1,54 | 1,95 | 1,02 | 0,74 | 1,31 |
| deriche | 1,28 | 1,98 | 1,58 | 1,00 | 1,91 | 2,22 | 1,25 | 1,19 | 1,40 |
| floyd-warshall | 0,83 | 1,62 | 1,02 | 0,99 | 0,42 | 1,63 | 1,12 | 0,83 | 1,61 |
| nussinov | 1,05 | 3,07 | 1,20 | 1,01 | 2,21 | 2,62 | 1,93 | 1,06 | 3,09 |

Варто зазначити, що експерименти 1 та 2 виконувались для тих самих тестових програм з такою ж розмірністю даних, проте на різних апаратних платформах, що дає можливість порівнювати енергоефективність обчислень. Порівняння сумарної електричної енергії, необхідної для обчислень тестових програм на платформах на базі Raspberry Pi 3 та Intel® Core™ i5 4670K дає можливість стверджувати, що в 26 випадках з 28 для обчислення тієї ж програми на Raspberry Pi 3 потрібно використати менше електричної енергії. Варто зазначити, що тривалість обчислень на Raspberry Pi 3 більше якнайменше в 6 разів ніж на платформі x64.

Виконуючи аналіз результатів обчислень, отриманих в експериментах 1 та 2 автор доводить можливість використання МРОЦНБ та розпаралелювання на більшості задач з метою прискорення швидкодії.

Аналізуючи результати вимірів двох експериментів автор звертає увагу на те, що метод розбиття на блоки параметричний, тобто має параметри роботи, власне це розміри блоків розбиття. Зважаючи на те, що в загальному випадку вони можуть бути довільними цілими числами, а безпосередньо в тесті використовувалась за

замовчуванням одна пара значень 32 та 32, то у автора виникло запитання, чи вплине на швидкість обчислень зміна розмірів блоків у деяких діапазонах значень.

Для перевірки цієї гіпотези автор провів експеримент 5, в якому для кожної з тестових програм на настільному ПК на базі чотириядерного процесора Intel® Core™ i5 4670K з 16ГБ оперативної пам'яті виміряно час виконання для багатьох різних розмірів блоків розбиття. Експерименти проводилися для двох комбінацій – методу розбиття на блоки та для методів розпаралелювання і розбиття на блоки одночасно. Отримані результати значно відрізняються в залежності від розмірів блоків та тестової програми. Приклади отриманих залежностей часу виконання тестових програм від розмірів двох блоків розбиття зображені на Рисунках 5 та 6.

Враховуючи складну залежність часу виконання від розміру блоків автор вводить дві оцінки для кожної з характеристик. Перша оцінка – відношення середнього часу виконання до мінімального, друга – відношення максимального часу виконання до мінімального. Такі оцінки дозволяють оцінити потенційне можливе покращення для випадково вибраного розміру блоку в середньому та максимально можливе покращення. Отримані дані зведені в таблицю 3.

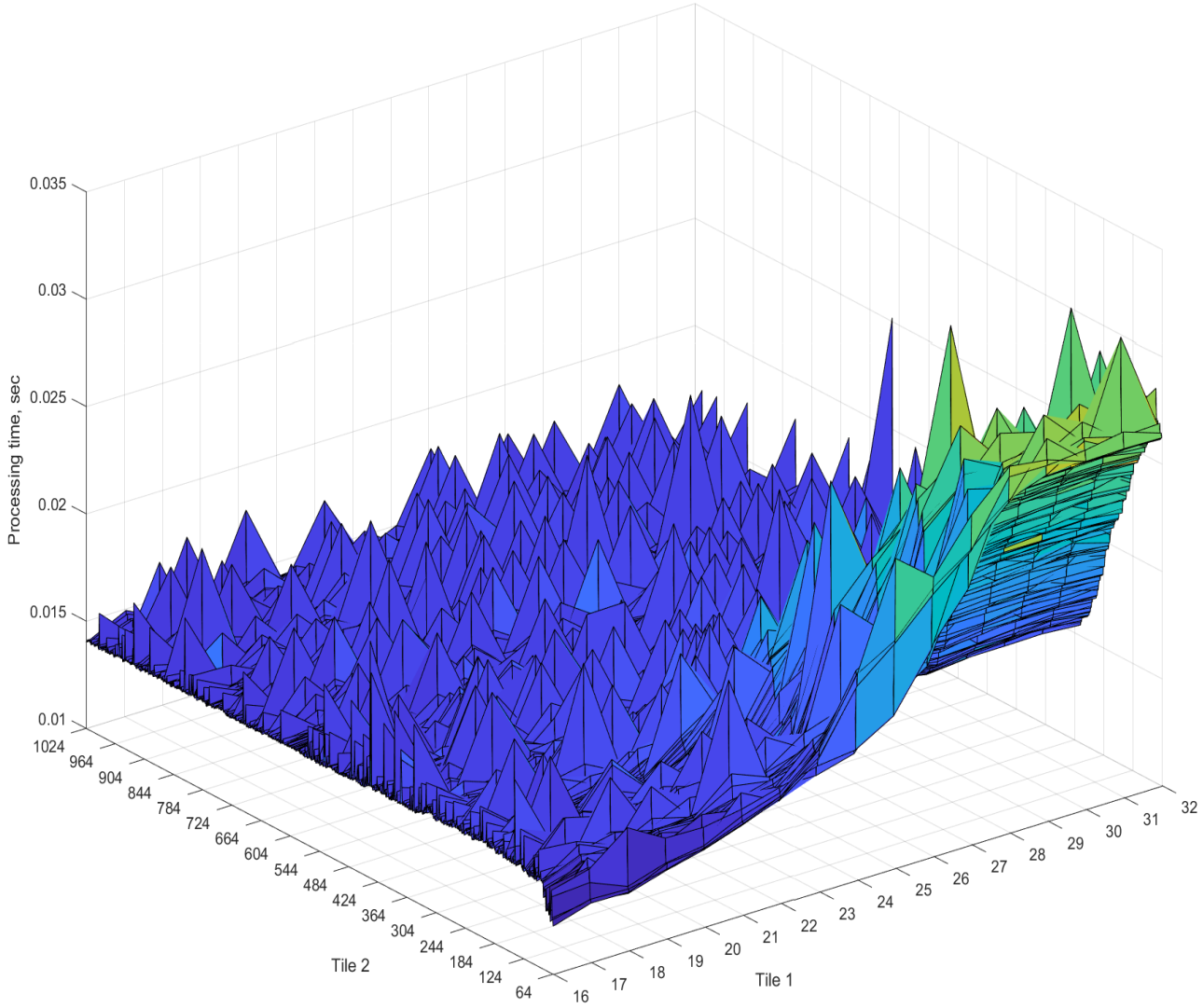


Рисунок 5 – Приклад 1 залежності часу виконання від розмірів блоків розбиття

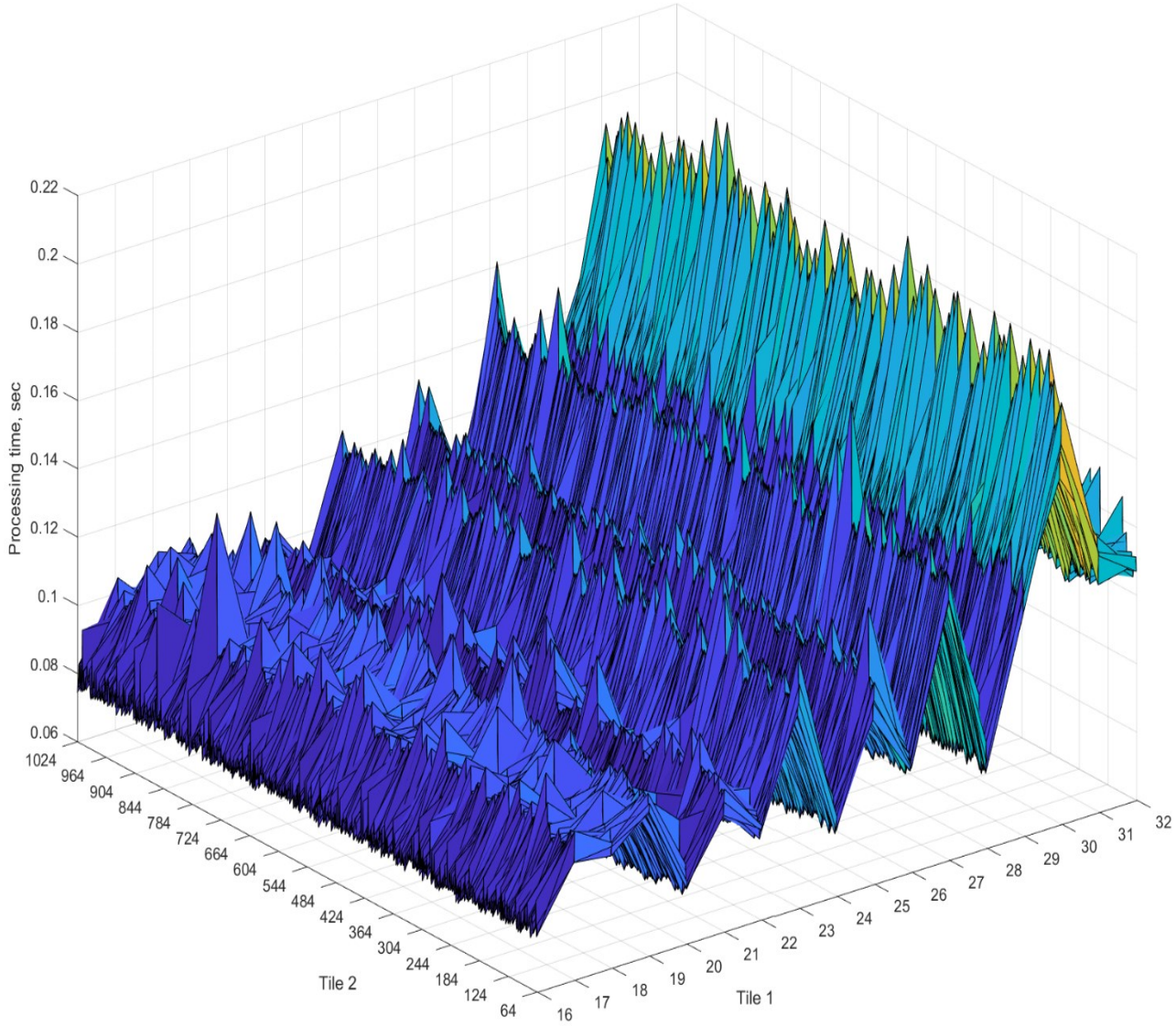


Рисунок 6 – Приклад 2 залежності часу виконання від розмірів блоків розбиття

Таким чином, в розділі показано, що час виконання складним чином залежить від розміру блоків розбиття. Ця функція може мати чітко виражені пікові значення, в яких виконання програми має значно менший час виконання чим при інших значеннях параметрів блоків розбиття ітераційного простору оператора циклу програми. Зважаючи на зазначене, пошук кращих розмірів блоків розбиття може забезпечити мінімальний час виконання програми.

В **третьому розділі**, запропоновано використання дискретного методу рою часток для задачі пошуку кращих розмірів блоків розбиття. Наявність великої розбіжності між часом виконання для різних розмірів блоків розбиття відзначається автором як можливість досягти прискорення часу виконання програм за рахунок правильного підбору оптимальних розмірів блоків.

Формула для методу рою часток наведена нижче:

$$v_{i,t+1} = wv_{i,t} + c_1 r_1 (m_i - x_{i,t}) + c_2 r_2 (g - x_{i,t}), \quad (1)$$

де $v_{i,t}$ – швидкості $i$ частки на $t$ ітерації, $x_{i,t}$ – координати $i$ частки на $t$ ітерації, $c_1$, $c_2$ – коефіцієнти відповідно індивідуальної та соціальної поведінки часток, $r_1$, $r_2$ – випадкові величини у діапазоні від 0 до 1, що обчислюються для кожної частки на

кожній ітерації, *m, g* – локальний і глобальний мінімуми, *w* – інерція швидкості; *i=1..N* – кількість часток (розмір популяції), *t=1..M* – кількість ітерацій.

Таблиця 3 – Відношення часу виконання тестових програм для різних розмірів блоків розбиття

| **Тестова програма** | **Tile** | | **Tile + Innerpar + Parallel** | |
|---|---|---|---|---|
| | **Середній/ Мінімальний** | **Максимальний/ Мінімальний** | **Середній/ Мінімальний** | **Максимальний/ Мінімальний** |
| 2mm | 1.02 | 2.49 | 1.51 | 43.77 |
| 3mm | 1.02 | 1.46 | 1.39 | 13.74 |
| atax | 1.13 | 1.78 | 1.46 | 32.85 |
| bicg | 1.17 | 2.30 | 1.42 | 46.51 |
| cholesky | 1.05 | 1.55 | 2.74 | 28.62 |
| covariance | 1.08 | 1.87 | 2.03 | 254.57 |
| doitgen | 1.64 | 2.29 | 1.52 | 2.99 |
| durbin | 1.01 | 1.58 | 1.01 | 2.03 |
| gemm | 1.13 | 1.74 | 1.22 | 2.41 |
| gemver | 1.21 | 1.54 | 1.24 | 5.86 |
| gesummv | 1.25 | 2.75 | 1.29 | 13.75 |
| gramschmidt | 1.65 | 2.63 | 2.05 | 16.28 |
| lu | 1.11 | 2.96 | 2.22 | 48.82 |
| mvt | 1.20 | 2.43 | 1.33 | 8.23 |
| symm | 1.02 | 1.67 | 1.02 | 1.57 |
| syr2k | 1.20 | 1.97 | 1.53 | 15.84 |
| syrk | 1.04 | 2.28 | 1.62 | 28.84 |

Суть методу зводиться до ітеративного пошуку мінімуму функції кожної з часток, що інформаційно пов’язані. Використовуючи різні коефіцієнти методу, а саме: коефіцієнти індивідуальної та соціальної поведінок, інерцію швидкості, можна корегувати поведінку часток, визначаючи їх збіжність до глобального мінімуму або пошук в околі власного локального мінімуму.

В даній роботі використано дискретний метод рою часток (ДМРЧ), оскільки вузлами ітераційного простору можуть бути тільки цілі числа. Для такого методу використовується та сама формула (1), що і для класичного методу рою часток, проте додаються обмеження – $x_{i,t} \in Z$ та $v_{i,t} \in Z$. Для виконання цих умов початкові координати часток обираються цілими і на кожній ітерації значення швидкості округлюється до цілого.

Виходячи зі складної залежності часу виконання КП від розмірів блоків розбиття, що випливає з Рис. 5 та 6, автором запропоновано використання дискретного методу рою часток як алгоритму пошуку мінімуму часу виконання. Головна перевага даного методу для вирішення поставленої задачі – наявність багатьох агентів пошуку, що дозволяє охоплювати пошуковий діапазон найкращим чином, уникаючи пошуку тільки навколо локального мінімуму.

Алгоритм запропонованого методу наведено на Рисунку 7.

З метою визначення швидкості та ефективності пошуку проведено серію експериментів щодо використання дискретного методу рою часток. Автор послідовно виконував пошук кращого часу виконання тестової програми для різної кількості ітерацій та кількості часток.

Отримані дані засвідчують покращення результатів при збільшенні кількості ітерацій та часток. В той же час найкращі результати отримуються при значній кількості ітерацій та часток. Це означає, що в практичному використанні необхідно виконати багато вимірів, і як наслідок, це вимагатиме більше часу, щоб отримати кращий результат пошуку. У розділі наведено результати підбору параметрів дискретного методу рою часток.

**Четвертий розділ** дисертації присвячено верифікації запропонованого методу інтелектуального блочного розбиття і показано, що запропонований метод дозволяє автоматично обрати оптимальні параметри розпаралелювання послідовних програм мовою високого рівня. Для оцінки енергоефективності обчислень автором пропонується метод, що порівнює час виконання та енергоспоживання для початкової та оптимізованої програм.

Сумарна електрична енергія, що необхідна для обчислення програми дорівнює інтегралу електричної потужності по часу виконання цієї програми:

$$E = \int_0^T p(t)dt. \quad (2)$$

Приймаючи до уваги постійне значення напруги живлення, значення електричної потужності в часі можна прийняти у вигляді:

$$p(t) = U \cdot i(t). \quad (3)$$

Враховуючи, що середнє значення сили струму може бути представлено згідно вказаної формули:

$$I_{mean} = \frac{1}{T}\int_0^T i(t)dt. \quad (4)$$

З наведених формул отримаємо загальну формулу розрахунку споживаної електричної енергії:

$$E = U \cdot I_{mean} \cdot T. \quad (5)$$

Для визначення кількісної оцінки покращення енергетичної ефективності при використанні оптимізації введемо коефіцієнт енергетичної ефективності розрахунків, як відношення початкової електричної енергії, що необхідна для розрахунку до електричної енергії, що необхідна до розрахунку оптимізованої версії:

$$k_E = \frac{E_{original}}{E_{optimized}} = \frac{U \cdot I_{mean\_original} \cdot T_{original}}{U \cdot I_{mean\_optimized} \cdot T_{optimized}} = \left(\frac{T_{original}}{T_{optimized}}\right) \cdot \left(\frac{I_{mean_{original}}}{I_{mean_{optimized}}}\right). \quad (6)$$

Використовуючи результати часу виконання тестових програм та їх енергоспоживання, автор використовує формулу (6) для оцінки коефіцієнту енергоефективності обчислень для ПК на базі Intel® Core™ i5-4670K та для плати Raspberry Pi 3. Таблиця 4 містить результати вимірювання енергоефективності для платформи на базі процесора Intel. Значення більші за 1 свідчать про покращення енергоефективності, значення менші 1 – про погіршення.

Результати такого експерименту дозволяють стверджувати, що в більшості випадків метод розбиття на блоки не тільки сприяє зниженню часу виконання, але ще покращує енергоефективність обчислень.

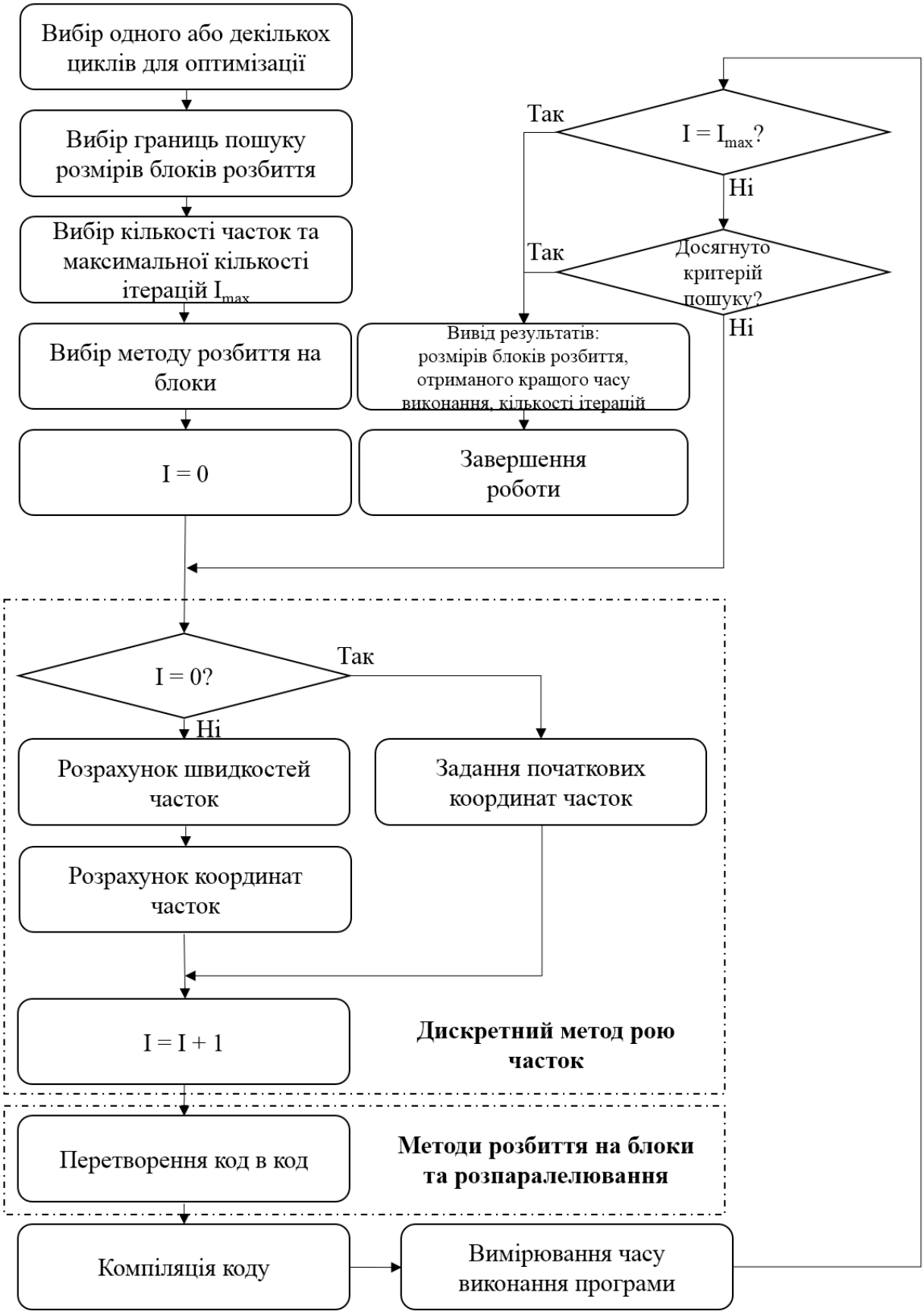


Рисунок 7 – Алгоритм методу інтелектуального блочного розбиття

Таблиця 4 – Коефіцієнти енергоефективності обчислень для тестових програм для ПК на базі Intel® Core™ i5-4670K при використанні різних методів розбиття на блоки та розпаралелюванні

| **Тестова програма** | **Опції оптимізації програми Pluto** | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | **Tile** | **Tile Parallel** | **Tile L2Tile Parallel** | **Innerpar** | **innerpar parallel** | **Innerpar Tile Parallel** | **Tile Multipar Parallel** | **Diamond-tile** | **Diamond-tile Parallel** |
| correlation | 4,86 | 8,07 | 5,01 | 4,80 | 10,71 | 15,73 | 7,13 | 5,10 | 15,03 |
| covariance | 4,81 | 7,54 | 5,00 | 4,99 | 14,48 | 16,06 | 7,52 | 5,09 | 15,50 |
| gemm | 2,35 | 6,28 | 18,93 | 1,45 | 5,89 | 14,76 | 5,79 | 2,25 | 14,57 |
| gemver | 5,12 | 12,43 | 21,14 | 16,04 | 31,95 | 20,90 | 10,08 | 5,12 | 18,61 |
| gesummv | 0,88 | 2,16 | 3,10 | 1,77 | 6,51 | 3,91 | 1,84 | 0,90 | 3,62 |
| symm | 1,04 | 1,02 | 1,06 | 1,01 | 1,03 | 1,05 | 1,05 | 1,06 | 1,05 |
| syr2k | 1,41 | 3,68 | 1,81 | 1,44 | 8,39 | 7,46 | 2,97 | 1,32 | 7,43 |
| syrk | 1,00 | 2,12 | 1,33 | 0,96 | 2,91 | 2,87 | 1,56 | 0,99 | 3,83 |
| trmm | 3,40 | 11,10 | 4,61 | 5,47 | 1,70 | 27,26 | 10,97 | 3,49 | 25,93 |
| 2mm | 1,91 | 3,92 | 2,37 | 1,64 | 9,24 | 7,90 | 4,34 | 1,97 | 7,94 |
| 3mm | 1,59 | 3,28 | 2,01 | 1,34 | 7,41 | 6,78 | 3,70 | 1,68 | 6,79 |
| atax | 0,42 | 1,04 | 1,56 | 1,00 | 1,89 | 1,50 | 0,72 | 0,36 | 1,52 |
| bicg | 0,82 | 2,08 | 3,28 | 2,26 | 3,93 | 3,27 | 1,61 | 0,76 | 2,57 |
| doitgen | 4,73 | 2,84 | 4,42 | 6,42 | 0,30 | 5,65 | 6,20 | 5,39 | 5,50 |
| mvt | 3,67 | 8,53 | 17,28 | 1,16 | 4,64 | 15,23 | 8,19 | 3,69 | 13,35 |
| cholesky | 1,24 | 2,20 | 1,48 | 1,01 | 2,33 | 4,19 | 1,82 | 1,28 | 4,07 |
| durbin | 1,00 | 1,00 | 1,01 | 0,99 | 1,00 | 1,00 | 1,00 | 1,00 | 1,00 |
| gramschmidt | 1,79 | 2,41 | 1,74 | 1,04 | 3,80 | 4,07 | 2,84 | 1,75 | 3,97 |
| lu | 1,46 | 2,74 | 2,25 | 1,60 | 1,90 | 6,35 | 2,90 | 1,53 | 5,18 |
| ludcmp | 1,01 | 1,00 | 1,00 | 0,99 | 1,00 | 1,00 | 0,97 | 0,96 | 0,99 |
| trisolv | 0,73 | 1,43 | 1,93 | 1,22 | 1,96 | 2,47 | 1,09 | 0,70 | 1,64 |
| deriche | 1,28 | 1,57 | 2,00 | 1,00 | 2,43 | 2,83 | 1,32 | 1,19 | 1,75 |
| floyd-warshall | 0,81 | 1,21 | 1,33 | 0,99 | 0,54 | 2,22 | 1,27 | 0,84 | 2,18 |
| nussinov | 1,04 | 2,30 | 1,52 | 1,02 | 2,86 | 3,49 | 2,17 | 1,07 | 4,15 |
| fdtd-2d | 0,93 | 1,18 | 0,46 | 0,92 | 4,22 | 3,02 | 1,46 | 1,00 | 1,00 |
| heat-3d | 2,02 | 1,64 | 1,00 | 2,62 | 11,79 | 4,37 | 2,51 | 1,00 | 1,00 |
| jacobi-2d | 0,97 | 1,35 | 1,00 | 0,95 | 5,05 | 2,99 | 1,61 | 1,00 | 1,00 |
| seidel-2d | 1,19 | 2,22 | 1,00 | 1,02 | 4,75 | 4,65 | 2,14 | 1,00 | 1,00 |

Таким чином, як видно з таблиці 4, отримані дані засвідчують покращення енергоефективності обчислень в більшості випадків. Варто звернути увагу, що кількість необхідної електричної енергії для розрахунку може бути зменшена в декілька разів у найкращому випадку. Аналогічно було отримано дані для плати Raspberry Pi 3, що також засвідчують покращення енергоефективності обчислень.

В другому підрозділі четвертого розділу показано зменшення кількості ітерацій і часток пошуку при запропонованих коефіцієнтах методу дискретного методу розбиття на блоки. Порівнюючи мінімум часу виконання програм, отриманий при підібраних коефіцієнтах ДМРЧ з мінімумом часу виконання програм, можна стверджувати, що пошук кращих розмірів блоків розбиття став виконуватись набагато швидше. А саме, в даному випадку мінімум часу виконання був знайдений за 64 ітерації для 4 часток або 4 ітерації для 8 часток. Раніше, щоб

отримати такі дані треба було виконати 64 ітерації для 32 часток або 256 ітерацій для 4 часток.

В третьому підрозділі четвертого розділу показана ефективність розбиття для цільової функції часу, коли дії спрямовані на мінімізацію часу виконання програми користувача. На рисунках 8 та 9 наводяться порівняння часу виконання початкових тестових програм, час виконання тестових програм з використанням методів розбиття на блоки окремо та сумісно з розпаралелюванням, а також час виконання тестових програм з використанням запропонованого методу інтелектуального блочного розбиття.

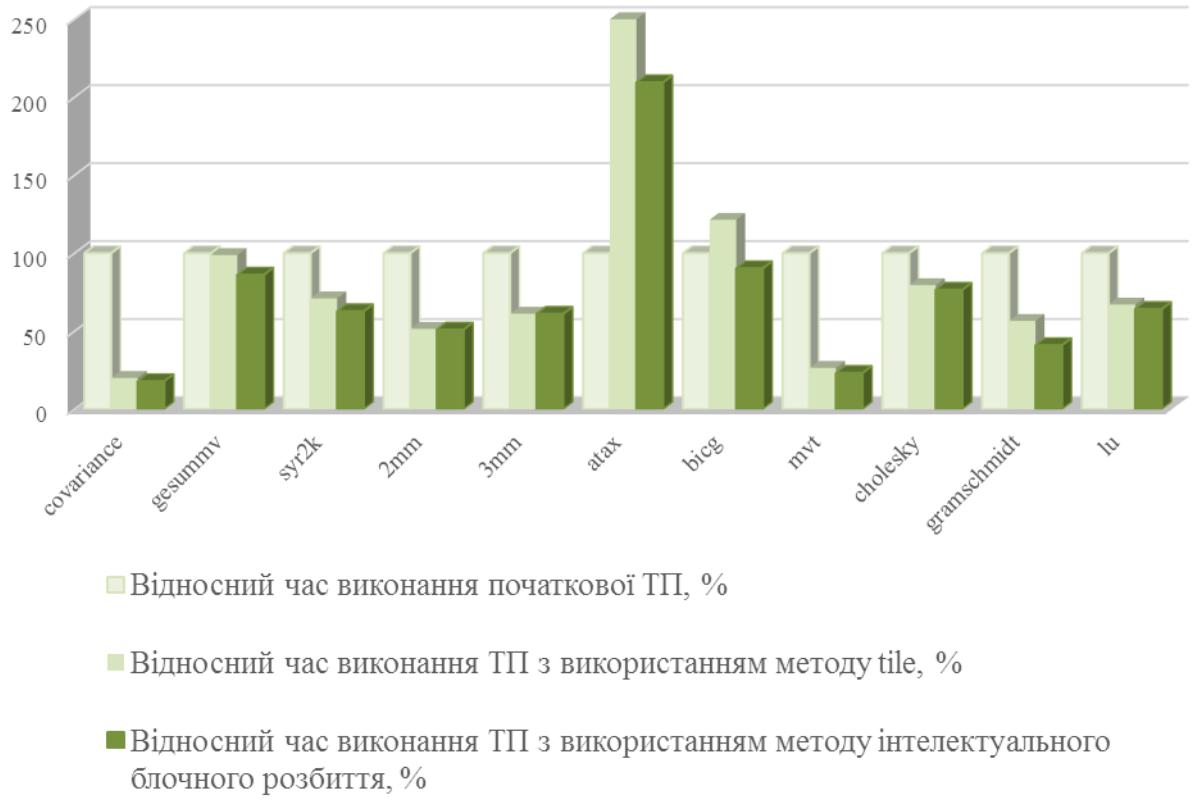


Рисунок 8 – Відносний час виконання тестових програм при використанні методу tile та методу інтелектуального блочного розбиття

Отримані результати свідчать, що запропонований метод інтелектуального блочного розбиття додатково прискорює час виконання тестових програм, що використовують метод розбиття на блоки від 1% до 45%.

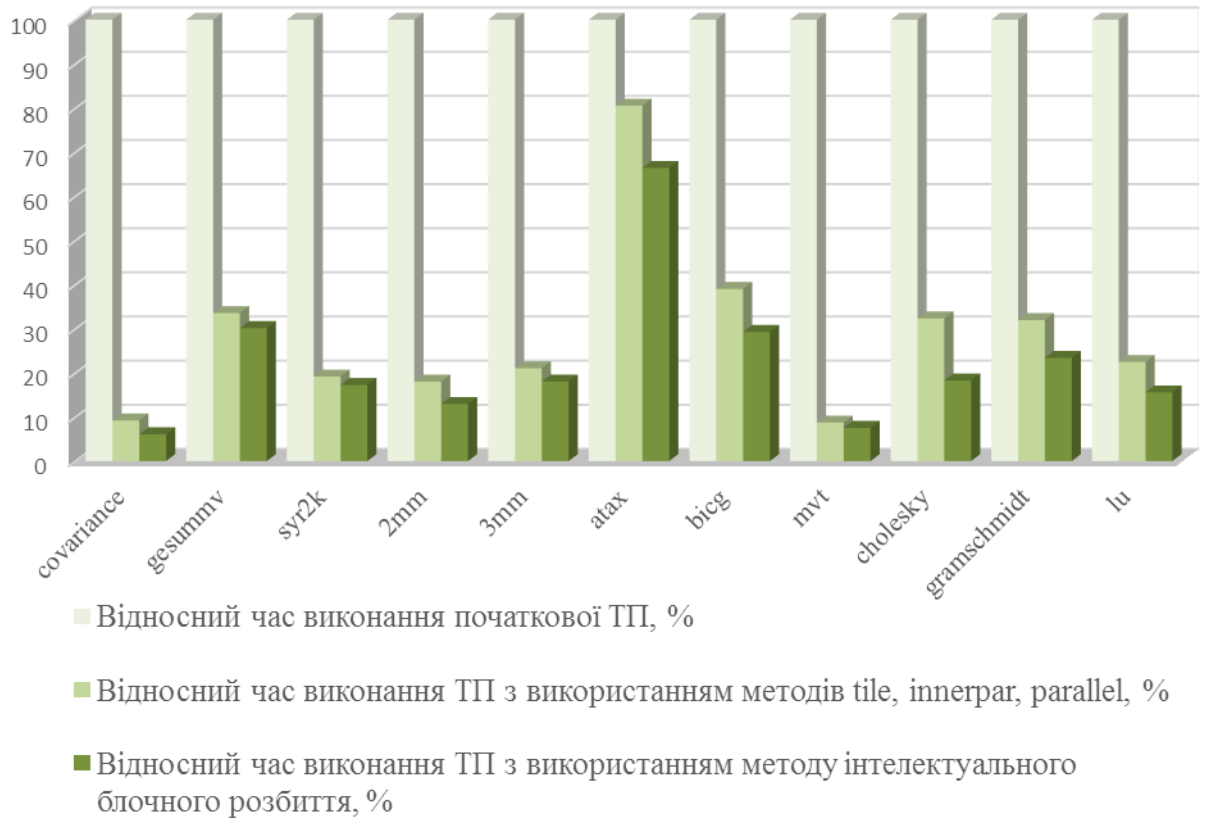


Рисунок 9 – Відносний час виконання тестових програм при використанні методів tile, innerpar, parallel та методу інтелектуального блочного розбиття

## ВИСНОВКИ

У дисертаційній роботі розв'язано актуальну науково-практичну задачу з удосконалення методів оптимізації ПЗ. В роботі запропоновано метод інтелектуального блочного розбиття, який використовує дискретний метод рою часток з метою підбору найкращих розмірів блоків розбиття в оптимізаційному методі розбиття на блоки для прискорення виконання комп’ютерних програм. Ефективність застосування запропонованого методу доводиться отриманими експериментальними даними, які засвідчують покращення отриманих результатів відносно класичного методу розбиття на блоки.

При цьому отримані наступні основні результати:

1. Вперше запропоновано ітераційний процес розпаралелювання операторів циклів мікропроцесорних програм, який відрізняється від відомих методів визначенням оптимального рішення щодо розбиття ітераційного простору оператора циклу на окремі блоки.

2. Вперше запропоновано оцінку доцільності оптимізації коду програм і отримано експериментальні дані, що засвідчують покращення ефективності обчислень (за часом або енергоспоживанням), в більшості випадків при застосуванні методу розбиття на блоки.

3. Розроблено метод інтелектуального блочного розбиття та його програмну реалізацію, що виконує пошук оптимальних розмірів прямокутних блоків розбиття ітераційного простору операторів циклу мікропроцесорних програм на основі

дискретного методу рою часток, який може бути застосовано до довільних обчислювальних циклів написаних мовами програмування С або С++.

4. Вдосконалено метод оцінки часу виконання тестових програм при використанні різноманітних методів розбиття на блоки, який базується на автоматичному послідовному багаторазовому запуску тестових програм і фіксації отриманих результатів швидкодії, що можуть бути в подальшому проаналізовані.

5. Вдосконалено дискретний метод рою часток шляхом визначення коефіцієнтів методу, а саме: початкового коефіцієнту інерції, індивідуального та соціального коефіцієнтів, при яких пошук розмірів блоків розбиття в задачі прискорення швидкодії виконується швидше класичного методу.

6. Вдосконалено використання дискретного методу рою часток шляхом визначення кращих початкових даних для розташування часток рою, що зменшує число ітерацій для знаходження оптимального рішення.

В роботі наводяться дані вимірювань, що свідчать не тільки про прискорення виконання більшості тестових програм при використанні методів розбиття на блоки, але й про покращенні енергоефективності обчислень. Цей факт свідчить про доцільність використання методу в «зелених» обчисленнях.

Отримані у рамках дисертаційної роботи результати підтверджують ефективність запропонованого методу та його програмної реалізації. Запропонований метод та його програмна реалізація можуть бути використані для оптимізації комп'ютерних програм написаних мовою програмування С або С++ будь-якого призначення, апаратної платформи та використовуємого компілятора.

## СПИСОК ОПУБЛІКОВАНИХ ПРАЦЬ ЗА ТЕМОЮ ДИСЕРТАЦІЇ

1. Chemeris A. Influence of Software Optimization on Energy Consumption of Embedded Systems / Chemeris A., Lazorenko D., Sushko S. // Green IT Engineering: Components, Networks and Systems Implementation. Studies in Systems / Kharchenko V., Kondratenko Y., Kacprzyk J. (eds) – Cham.: Springer, 2017. – Vol. 105. – С. 111-133.

2. Чемерис А.А. Исследование быстродействия и энергопотребления при автоматической оптимизации методами разбиения на блоки и распараллеливания для вычислений на платформе X64 / А.А. Чемерис, С.В. Сушко // Моделювання та інформаційні технології. – Київ, 2017. – Вип. 80. – С. 52-60.

3. Сушко С.В. Вплив розмірів блоків розбиття операторів циклів на час виконання комп'ютерних програм / С.В. Сушко, О.А. Чемерис // Моделювання та інформаційні технології. – Київ, 2018. – Вип. 82. – С. 110-117.

4. Sushko S. Dependency between Tiles' Sizes and Program Execution Time / Sushko S., Chemerys A. // Measurement Automation Monitoring. – Gliwice, 2018. – Vol. 64, No.2, P. 28-30, Wydawnictwo PAK, ISSN 2450-2855.

5. Chemerys A. Analysis and Optimization of the Sizes of the Iteration Space Tiles During the Parallelization of Program Loop Operators / Chemerys A., Sushko S. // Advances in Cyber-Physical Systems. – Київ, 2018. – Випуск 3, Номер 1, С. 1-6.

6. Сушко С.В. Визначення енергоефективності обчислень на різних обчислювальних системах / С.В. Сушко, О.А. Чемерис // Моделювання та інформаційні технології. – Київ, 2018. – Вип. 85. – С. 34-39.

7. Чемерис А.А. Исследование эффективности выполнения программ, оптимизированных на основе полиэдральной модели / А.А. Чемерис, С.В. Сушко // Сборник трудов 5й международной конференции SIMULATION-2016. – Киев, 2016. – С. 241-244.

8. Чемерис А.А. Сравнение эффективности автоматической оптимизации на основе полиэдральной модели / А.А. Чемерис, С. В. Сушко // Summer InfoCom 2016: Матеріали II Міжнародної науково-практичної конференції, м. Київ, 1-3 червня 2016 р. – К.:Вид-во "Інжиніринг", 2016. – С. 74-76.

9. Чемерис А.А. Оценка снижения времени выполнения тестовых программ при применении автоматической оптимизации на основе полиэдральной модели на Raspberry Pi 3 / А.А. Чемерис, С.В. Сушко // Winter InfoCom 2016: Матеріали III Міжнародної науково-практичної конференції, м. Київ, 1-2 грудня 2016 р. – К.:Вид-во «Інжиніринг», 2016. – С. 63-65.

10. Chemeris A. The Dependence of Microprocessor System Energy Consumption on Software Optimization / A. Chemeris, S. Sushko // 2017 IEEE 37th International Conference on Electronics and Nanotechnology (ELNANO), April 18-20, 2017, Kyiv, ISBN: 978-1-5386-1700-7, P. 451-454.

11. Сушко С.В. Вплив розмірів блоків розбиття на час виконання програм. / С.В. Сушко // Науково-технічна конференція молодих вчених і спеціалістів ІПМЕ ім. Г. Є. Пухова НАН України, – Київ, 12 травня 2018, – С. 34-37.

12. Sushko S. Increasing An Energy Efficiency Of Computational System By Using Automatic Software Optimization / S. Sushko, A. Chemeris // 2018 IEEE 9th International Conference on Dependable Systems, Services and Technologies (DESSERT-2018), May 24-27 2018, Kyiv, ISBN 978-1-5386-5902-1, P. 613-617.

13. Сушко С. Вплив розмірів блоків розбиття операторів циклів на час виконання комп'ютерних програм / Сушко С. Чемерис О. // Збірник праць 6-ї міжнародної конференції Моделювання-2018, 12-14 вересня 2018, Київ, С. 234-238.

14. Чемерис О.А. Методи штучного інтелекту при оптимізації роботи мікропроцесорних систем. / О.А. Чемерис, С.В. Сушко // XII Міжнародна Науково-Практична Конференція Комп'ютерні Системи та Мережні Технології 28-30 березня 2019, Київ, С. 127-129.

15. Chemeris A. Usage of Discrete Particle Swarm Optimization Method for the Searching of Optimal Tile Size / A. Chemeris, S. Sushko // 2019 IEEE International Scientific-Practical Conference Problems of Infocommunications, Science and Technology (PIC S&T-2019), October 8-11, 2019, Kyiv, P.202-206. ISBN: 978-1-7281-4183-1.

16. Сушко С.В. Універсальні оптимізаційні методи програмного забезпечення / Сушко С.В. // Науково-практична конференція Безпека енергетики в епоху цифрової трансформації, 20 грудня 2019, Київ, С. 89-91.

## АНОТАЦІЯ

Сушко С.В. Методи оптимального розпаралелювання програм мікропроцесорних систем для підвищення їх ефективності. – На правах рукопису.

Дисертація на здобуття наукового ступеня кандидата технічних наук за спеціальністю 05.13.05 – комп'ютерні системи та компоненти. – Інститут проблем моделювання в енергетиці ім. Г.Є. Пухова НАН України, Київ, 2021.

У дисертаційній роботі отримано нові науково-прикладні результати щодо прискорення швидкодії комп'ютерних програм за рахунок використання методу розбиття обчислювальних циклів на блоки з підбором кращих розмірів блоків розбиття. Отримані експериментальні дані свідчать про прискорення швидкодії та енергоефективності обчислень для більшості тестових програм при використанні методу. Розробка знайде використання у комп'ютерних програмах, що написані мовами програмування C та C++. Перевагою даного методу є його універсальність; він може використовуватись для комп'ютерних програм різноманітного призначення, апаратної платформи та використаного компілятора.

Для пошуку кращих розмірів блоків розбиття автором використано дискретний метод рою часток, що ітеративно підбирає різні варіанти розмірів блоків і, тим самим виконує пошук найкращих розмірів блоків, що забезпечить мінімальний час виконання програм. Отже, час витрачений на пошук найкращих параметрів оптимізаційного методу дозволить знайти саме такі параметри, що прискорять швидкодію комп'ютерної програми максимально можливим способом. Знайдені параметри оптимізації в подальшому можуть бути використані в оптимізованій програмі на будь-якій кількості пристроїв. Таким чином, досягається доцільність тривалого пошуку кращих параметрів оптимізації.

Автором вперше застосовано дискретний метод рою часток для пошуку кращих розмірів блоків розбиття в методі оптимізації розбиття на блоки комп'ютерних програм.

Розробка знайде використання у комп'ютерних програмах, що написані мовами програмування C та C++. Перевагою запропонованого методу інтелектуального блочного розбиття є його універсальність; він може використовуватись для комп'ютерних програм будь-якого призначення, апаратної платформи та використаного компілятора. Недоліком методу є його ітеративна природа – необхідність вимірювання часу виконання програм для кожної запропонованої пари розмірів блоків розбиття.

*Ключові слова:* оптимізація комп'ютерних програм, енергоефективні обчислення, прискорення швидкодії комп'ютерних програм, метод розбиття на блоки, дискретний метод рою часток, метод інтелектуального блочного розбиття.

## ABSTRACT

Sushko S.V. Methods of optimal parallelization of programs of microprocessor systems to increase its efficiency. – As the manuscript.

Thesis for candidate's degree in technical science by speciality 05.13.05 – computer systems and components. – Pukhov Institute for Modeling in Energy Engineering, National Academy of Sciences of Ukraine, Kyiv, 2021.

New scientific and applied results for accelerating of execution of computer programs by using of tiling method with choice of best tiles' sizes were obtained in the thesis. Obtained experimental data show an improvement in processing speed and energy efficiency of calculations for most test programs with using of proposed Smart Tiling

Method.

Usage of tiling method improves locality of data and, thus, provides the prerequisites for accelerating of execution of computer programs. At the same time, when using this method, attention is not always focused directly to values of tile sizes themselves. In this work the author investigates an influence of tile sizes on execution time of test computer programs. The author provides data and graphs that confirm a complicated dependence between tile sizes and execution time of test programs.

To find best tile sizes the author used Discrete Particle Swarm Optimization Method, which iteratively selects different tile sizes based on execution time of computer program. It searches for best tile sizes, which will provide minimum program's execution time. Thus, time spent on iterative search for best parameters of the optimization method will allow to find that parameters that will improve performance of computer program in best possible way. Once optimization parameters had been found, they will be used in optimized program in any number of computing devices.

Usage of Discrete Particle Swarm Optimization Method in the problem of finding best tile sizes was proposed. The resulting method was called Smart Tiling Method.

*The scientific novelty* of the obtained results is as follow.

*In first time:*

– An iterative process of parallelization of loop operators of microprocessor programs is proposed, which differs from known methods by determining of optimal solution for dividing of iterative space of loop operators into separate blocks.

– The estimation of expediency of code optimization of programs was proposed. Experimental data which confirm improvement of efficiency of calculations for execution time and power consumption in most cases for tiling method were obtained.

– Smart Tiling Method was developed. The Method searches for optimal size of rectangular blocks of iterative space partitioning of microprocessor program loop operators. The Method is based on Discrete Particle Swarm Optimization Method and can be applied to arbitrary computational loops written with C or C++ programming languages.

*Improved:*

– The method of estimation of execution time of test programs for using different tiling methods. The method of estimation is based on automatic sequential multiple run of test programs and collecting of obtained results of performance, which can be further analyzed.

– Discrete Particle Swarm Optimization Method by determining of its coefficients – social, individual, and initial velocity inertia. With found coefficients a process of searching of tile size for problem of speed improvement is performed faster than classical method.

– Using Discrete Particle Swarm Optimization Method by determining best initial locations of particles which reduces number of iterations to find optimal solution.

As can be seen from the provided data, Smart Tiling Method increases an efficiency of the method it uses. Level of efficiency increase varies depending for the different test programs. In cases where tiling method gave a slowdown for test program, Smart Tiling Method also accelerated it.

The thesis presents measurement data which indicate not only acceleration of most

test programs by using of tiling method, but also an improvement of energy efficiency of calculations. This fact indicates of prospects of using the method in "green" calculations.

In the thesis also energy efficiency of computations for the same test programs and dame data dimension for two different hardware platforms was compared. It was made for Raspberry Pi 3 board and PC based on CPU Intel® Core ™ i5-4670K. The obtained data show that in 92.8% of cases Raspberry Pi 3 board is more energy efficient.

This research can be used for computer programs written in C and C ++ programming languages. The advantage of proposed Smart Tiling Method is its versatility; it can be used for computer programs of any purpose, hardware platform and used compiler. The disadvantage of the method is its iterative nature – a requirement to measure of execution time of programs for each proposed pair of tile sizes. Selection of best tile sizes can take a long time if large number of particles and iteration is used, also if execution time of computer program is long enough.

*Keywords:* optimization of computer programs, energy efficient calculations, accelerating of computer programs, tiling method, Discrete Particle Swarm Optimization Method, Smart Tiling Method.

## АННОТАЦИЯ

Сушко С.В. Методы оптимального распараллеливания программ микропроцессорных систем для повышения их эффективности. – На правах рукописи.

Диссертация на соискание ученой степени кандидата технических наук по специальности 05.13.05 – компьютерные системы и компоненты. – Институт проблем моделирования в энергетике им. Г.Е. Пухова НАН Украины, Киев, 2021.

В диссертационной работе получены новые научно-прикладные результаты по улучшению быстродействия компьютерных программ за счет использования метода разбиения вычислительных циклов на блоки с подбором лучших размеров блоков разбиения. Полученные экспериментальные данные свидетельствуют об улучшении производительности и энергоэффективности вычислений для большинства тестовых программ при использовании метода. Разработка найдет применение в компьютерных программах, написанных на языке программирования С и С ++. Преимуществом данного метода является его универсальность; он может использоваться для компьютерных программ любого назначения, аппаратной платформы и использованного компилятора.

Для поиска лучших размеров блоков разбиения автором использовался дискретный метод роя частиц, который итеративно подбирает различные варианты размеров блоков и таким образом выполняет поиск лучших размеров блоков, что обеспечит минимальное время выполнения программ. Таким образом, время, потраченное на поиск наилучших параметров оптимизационного метода, позволит найти именно такие параметры, которые улучшат быстродействие компьютерной программы максимально возможным способом. Найденные параметры оптимизации в дальнейшем используются в оптимизированной программе на любом количестве устройств. Таким образом, достигается целесообразность длительного поиска лучших параметров оптимизации.

Автором впервые применен дискретный метод роя частиц для поиска лучших размеров блоков разбиения в методе оптимизации разбиения на блоки компьютерных программ.

Разработка найдет применение в компьютерных программах, написанных на языке программирования C и C++. Преимуществом предложенного интеллектуального блочного разбиения является его универсальность; он может использоваться для компьютерных программ любого назначения, аппаратной платформы и использованного компилятора. Недостатком метода является его итеративная природа – необходимость измерения времени выполнения программ для каждой предложенной пары размеров блоков разбиения.

*Ключевые слова*: оптимизация компьютерных программ, энергоэффективные вычисления, улучшение быстродействия компьютерных программ, метод разбиения на блоки, дискретный метод роя частиц, метод интеллектуального блочного разбиения.