НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО

Факультет інформатики та обчислювальної техніки
(назва факультету, інституту)

Кафедра автоматизованих систем обробки інформації і управління
(назва кафедри)

"На правах рукопису"
УДК *519.854.2*

«До захисту допущено»
Завідувач кафедри

_____ О.А.Павлов
(підпис) (ініціали, прізвище)

“ ” _____ 20 *18* р.

МАГІСТЕРСЬКА ДИСЕРТАЦІЯ

на здобуття ступеня магістра

за спеціальністю *122 Комп'ютерні науки та інформаційні технології*
(код та назва спеціальності)

спеціалізацією *Інформаційні управляючі системи та технології*
(код та назва спеціалізації)

на тему: *Задача мінімізації сумарного відхилення моментів завершення від директивних строків при виконанні завдань паралельними пристроями*

Виконав: студент *VI* курсу групи ІС-61м
(шифр групи)

Година Анастасія Вікторівна _____
(прізвище, ім'я, по батькові) (підпис)

Науковий керівник *доц., к.т.н., доц. Жданова О.Г.* _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Консультант *к.т.н., доц. Жданова О.Г.* _____
(науковий ступінь, вчене звання, прізвище, ініціали) (підпис)

Рецензент _____ _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Засвідчую, що у цій магістерській дисертації немає запозичень з праць інших авторів без відповідних посилань.
Студент _____
(підпис)

Київ – 2018

РЕФЕРАТ

Магістерська дисертація: 117 с., 32 рис., 23 табл., 1 додаток, 125 джерел.

Актуальність. Теорія розкладів і календарне планування утворюють одне з важливих і цікавих напрямків в області оптимізації та в даний час переживають період бурхливого розвитку. Це пов'язано, перш за все, з появою принципово нових видів продукції, технологій, інтенсифікацією виробництва, його безперервним оновленням і вдосконаленням. Стрімкий розвиток систем зв'язку, інтернету, логістичних структур ставить перед математиками нові завдання, в тому числі в області теорії розкладів. На практиці виникає безліч різноманітних завдань календарного планування виробництва і збуту продукції, ефективного використання обладнання та інших ресурсів, узгодження роботи різних служб і так далі.

Різноманітність математичних моделей і методів складання розкладів зазвичай ставить перед прикладними математиками й програмістами неминучу проблему побудови швидких алгоритмів і їх ефективної програмної реалізації з урахуванням особливостей вирішуваної задачі.

Більшість задач теорії розкладів і календарного планування є NP-складними, і виникають серйозні труднощі при їх вирішенні, так як побудова оптимального розкладу вимагає великих затрат часу навіть при порівняно невеликих розмірностях вхідних даних. У такій ситуації необхідно проводити більш глибокі дослідження задач в рамках теорії складності.

У зв’язку з цим актуальною є розробка програмного продукту для складання календарних планів виконання завдань паралельними пристроями, який допоможе мінімізувати сумарне відхилення від директивних строків.

Зв'язок роботи з науковими програмами, планами, темами. Робота виконувалась на кафедрі автоматизованих систем обробки інформації та управління Національного технічного університету України «Київський політехнічний інститут ім. Ігоря Сікорського» в рамках теми «Ефективні методи розв’язання задач теорії розкладів» (номер держреєстрації 0117U000919).

Мета дослідження – підвищення якості розв’язку задач календарного планування за рахунок побудови оптимального чи близького до оптимального розкладу, що дозволяє мінімізувати сумарний час відхилення від директивних строків.

Для досягнення мети необхідно виконати наступні **завдання**:

- провести аналіз відомих результатів з розв’язання поставленої в рамках роботи задачі;
- розробити алгоритм створення календарного плану виконання завдань паралельними пристроями, що мінімізує сумарне відхилення моментів завершення завдань від директивних строків;
- виконати програмну реалізацію розробленого алгоритму;
- дослідити ефективність алгоритму при різних вхідних даних шляхом проведення обчислювальних експериментів;

Об’єкт дослідження – процес календарного планування виконання завдань.

Предмет дослідження – моделі та методи розв’язання задач календарного планування з метою мінімізації сумарного відхилення виконання завдань від директивних строків паралельними пристроями.

Методи дослідження. Для виконання поставлених завдань у роботі було використано методи: теорії розкладів, дослідження операцій та теорії складності (при розробленні методів розв’язання задач складання розкладів).

Наукова новизна отриманих результатів.

Розроблено трьохетапний евристичний алгоритм розв‘язання задачі мінімізації сумарного відхилення моментів завершення від директивних строків при виконанні завдань паралельними пристроями.

Набув подальшого розвитку метод розв’язання задачі задачі мінімізації сумарного відхилення моментів завершення від директивних строків при виконанні завдань паралельними пристроями та має статистично сталі високі показники роботи.

Публікації. Матеріали роботи представлено у науковій статті міжнародного наукового журналу «Науковий огляд» [123] (свідоцтво про державну реєстрацію КВ № 20878-10678Р), опубліковані в тезах наукової конференції студентів, магістрантів та аспірантів «Інформатика та обчислювальна техніка» – ІОТ-2018 [124] та прийнято до публікації у вигляді тез 20-ї міжнародної конференції SAIT 2018 [125].

ПАРАЛЕЛЬНІ ПРИСТРОЇ, ДИРЕКТИВНИЙ СТРОК, СКЛАДАННЯ РОЗКЛАДІВ, КАЛЕНДАРНЕ ПЛАНУВАННЯ, МІНІМІЗАЦІЯ СУМАРНОГО ВІДХИЛЕННЯ

ABSTRACT

Master's thesis: 117 pages, 32 figures, 23 tables, 1 appendix, 125 references.

Relevance. The theory of schedules and operational scheduling are important and interesting directions in the field of optimization and are currently experiencing a period of rapid development. This is connected, first of all, to the emergence of fundamentally new types of products, technologies, intensification of production, its continuous updating and improvement. The rapid development of communication systems, the Internet, logistics structures puts for mathematicians new tasks, including in the field of scheduling theory. In practice, there are many diverse tasks of calendar planning of production and sales of products, the efficient use of equipment and other resources, the coordination of the work of various services, and so on.

A variety of mathematical models and scheduling methods usually puts for mathematicians and programmers the inevitable problem of constructing fast algorithms and their effective program implementation, taking into account the features of a solvable problem.

Most tasks in the theory of scheduling and operational scheduling are NP-hard, and there are serious difficulties in solving them, since building an optimal schedule requires a great deal of time even with relatively small dimensions of the input data. In such a situation, it is necessary to conduct more in-depth research of problems within the framework of complexity theory.

In this regard, it is important to develop a software product for drawing up schedules for the execution of tasks with parallel devices, which will help minimize the total deviation from the policy terms.

Relationship of work with scientific programs, plans, themes. The work was carried out at National Technical University of Ukraine «Igor Sikorsky Kyiv Polytechnic Institute» the department of Computer-Aided Management and Data Processing Systems within the theme "Effective methods for solving the problems of the theory of schedules" (state registration number 0117U000919).

Purpose and objectives of the study. Improving the efficiency of scheduling by building optimal or near to optimal schedules and minimizing the total deviation time from the due dates.

The following **tasks**:

- performing the known scheduling results review;
- developing an algorithm for minimizing total deviation time from the due dates for parallel machine scheduling;
- developing a software implementation of the algorithm in a form that can be used for schedule optimizing;
- performing an analysis of the results.

The object of study is the process of operational scheduling.

Subject of research: models and methods for solving scheduling problems in order to minimize the total deviation of task`s execution from due dates by parallel machines.

Research methods: theory of schedules, operations research and complexity theory.

Scientific novelty of the research. The approach is developed to solve the problem of minimizing the total deviation of completion times from due dates by parallel machines. The method for solving the problem of minimizing the total deviation of completion times from due dates by parallel machines and has statistically stable high performance indicators.

Publications. The materials are presented in the scientific article of the international scientific journal "Scientific Review" [123] (certificate of state registration КВ № 20878-10678P), published in the abstracts a scientific conference of students, undergraduates and graduate students "Informatics and Computer Science" - ICT-2018 [124] and accepted for publication at 20th International scientific-technical conference SAIT 2018 "System analysis and information technologies" [125].

PARALLEL MACHINES, DUE DATE, SCHEDULING, MINIMIZING TOTAL DEVIATION

ЗМІСТ

ВСТУП

Задачі теорії розкладів та задачі календарного планування пов’язані з розподілом обмежених ресурсів для виконання множини завдань та пошуком розкладу з найкращим значенням цільової функції. Обидва напрями виникли в 50-х роках минулого століття і зумовлені як зростанням складності комбінаторних задач по управлінню виробничими процесами, та і появою перших швидкодіючих обчислювальних пристроїв, що дали надію вирішити ці задачі за прийнятний час.

Хоча перші задачі теорії розкладів моделювали виробничі процеси, незабаром з’ясувалось, що схожі задачі виникають і в інших областях людської діяльності, зокрема:

- на виробництві, коли необхідно впорядкувати окремі операції по виконавцям (цеху, верстатам) і за часом;
- на транспорті при складанні розкладу руху поїздів, літаків, громадського міського транспорту;
- при плануванні занять в навчальних закладах;
- при плануванні зайнятості персоналу, наприклад, чергування лікарів;
- при виконанні складних тривалих проектів будівництва будівель, кораблів і т.п.;
- при плануванні проведення спортивних заходів;
- в комп'ютерних мережах при плануванні послідовності передачі пакетів інформації і т.д.

В класичних задачах теорії розкладів фактично не розглядаються ресурси, відмінні від ресурсу «машина», а основний акцент робиться на різноманітні технології виконання множини робіт і на різноманіття цільових функцій. До середини 80-х років минулого століття були виділені основні моделі класичної теорії розкладів та встановлена комбінаторна складність більшості задач, що виникають в рамках цих моделей. Не дивно, що приблизно у той самий час були зроблені перші спроби розширити класичні моделі теорії розкладів впровадженням в них таких додаткових

ресурсів як енергія, гроші, машинна пам’ять і багато іншого. Новий напрям досліджень отримав назву «Задачі теорії розкладів з обмеженнями на ресурси» і став одним з найбільш популярних напрямів в теорії розкладів в останні два десятиліття.

Теорія розкладів як наука стала формуватися в середині 1950-х років, після робіт Беллмана і Джонсона, які розглянули найпростішу задачу теорії – задачу двох верстатів [1, 2], і сформулювали основи математичного апарату, що лежать в основі вирішення цієї та інших подібних задач [3]. Потім протягом 30 років, аж до середини 1980-х років, розвиток теорії йшов в рамках декількох конкретних наукових напрямків.

Перший напрямок – використання для аналізу та синтезу розкладів комбінаторних методів. В рамках цього напрямку було отримано ряд важливих теоретичних результатів: розроблено перестановочний прийом, що дозволив знаходити аналітичні умови локальної оптимальності розкладу малої розмірності [4], розроблений неперервнологічний апарат (включаючи теорію логічних визначників), що дозволив отримати аналітичні умови локальної оптимальності розкладів довільно високої розмірності [5, 6], розвинений матричний метод синтезу розкладів, що використовує специфіку матриці часу виконання робіт в блоках системи [7], виділені випадки зводимості складання розкладу високої розмірності до складання розкладів малої розмірності [8].

Другий напрямок досліджень був пов'язаним з використанням для побудови оптимальних розкладів методів гілок і меж [5, 9, 10 та ін.]. Ці дослідження підтвердили, що методи гілок і меж дозволяють знаходити оптимальні розклади, подібно до того, як вони дозволяють знаходити оптимальні рішення в інших областях. При цьому, не було отримано яких-небудь досить загальних теоретичних результатів. Було тільки з'ясовано, що використання локальних умов оптимальності розкладу призводить до більш ефективного варіанту алгоритму – так званого методу гілок, меж та умов.

Третій напрям – застосування для аналізу і синтезу розкладів методів статистичного моделювання [11, 12 та ін.]. Ці методи засновані на випадковому послідовному поліпшенні наявного розкладу, шляхом вибору в кожному з послідовно

розігруваних множин можливих розкладів кращого розкладу і запам'ятовування рекордного з них. При цьому повинно бути сформульовано певне правило зупинки процесу. Було з'ясовано, що дані методи дозволяють отримувати розклади, близькі до оптимальних, однак оцінити точність отриманого рішення проблематично. Як і в другому напрямку досліджень, тут не вдалося отримати якихось досить загальних теоретичних результатів.

Четвертий напрямок примітний тим, що для синтезу оптимальних розкладів тут намагалися використовувати методи математичного програмування [13 – 15 та ін.]. При цьому було встановлено, що будь-яке завдання синтезу оптимального розкладу можна сформулювати як деяку задачу математичного програмування. Однак розмірність останньої зазвичай виявляється занадто великою. У зв'язку з цим практичне застосування методів математичного програмування для відшукання оптимальних розкладів виявляється проблематичним.

До середини 1980-х років відбулося деяке «насичення» числа отриманих науково-практичних результатів, що стосуються теорії розкладів. З іншого боку, з'ясувалося, що задача складання оптимального розкладу в загальному випадку є NP-складною, тобто не розв'язаною алгоритмами з поліноміальною складністю [16]. У зв'язку з цим інтерес дослідників став все більше переміщатися на нові постановки задач теорії розкладів, а також на нові методи вирішення різних постановок зазначених задач. Так, стали вивчатися задачі складання оптимальних розкладів при надходженні робіт до системи в режимі реального часу [17, 18], а також при заданих обмеженнях на допустимий порядок виконання робіт [19, 20], з'явилися дослідження по наближеним алгоритмам пошуку оптимальних розкладів з гарантованою точністю [21] і алгоритмам локального пошуку [22]. Значний інтерес придбали завдання синтезу оптимальних розкладів виконання робіт, при допустимості зміни порядку проходження робіт через систему , а також завдання побудови оптимальних розкладів в системах з неточно відомими часом виконання робіт [23]. У всіх зазначених напрямках працювало чимало вчених і були отримані цікаві наукові результати

Хоча перші задачі теорії розкладів моделювали виробничі процеси, незабаром стало відомо, що схожі задачі виникають і в інших областях людської діяльності,

особливо в організації оптимальної роботи сучасних обчислювальних пристроїв. Швидкий розвиток комп'ютерних технологій породжує велику кількість нових задач в теорії розкладів.

Зокрема, один із популярних напрямків розвитку комп'ютерних обчислень – це паралельні обчислення. При цьому основна складність при проектуванні паралельних програм полягає в тому, щоб забезпечити правильну послідовність взаємодій між різними обчислювальними процесами, а також координацію ресурсів, розподілених між процесами. В 1987 році Рейвайрд-Сміт [24] запропонував розглянути однорідну комунікаційну модель виконання множини завдань паралельної програми як задачу теорії розкладів з комунікаційними затримками. Ідея, запропонована Рйвардом-Смітом, виявилась дуже плодотворною, і задачі з комунікаційними затримками інтенсивно вивчаються вже більше двох десятиріч. Результатам в даному напрямку присвячені глави в монографіях [25] та [26].

Іншим важливим досягненням у розвитку комп'ютерних технологій стала можливість сучасних багатопроцесорних комп'ютерів варіювати свою швидкість. Високі швидкості обчислень призводять до більш швидкого і якісного обслуговування, але при цьому і до великих затрат енергії. Пошук золотої середини між економією енергії та збереженням якості обслуговування породив новий клас задач, в яких потребується знайти енергетично ефективні розклади.

Більшість задач теорії розкладів є NP-складними. Не дивлячись на це, практика потребує вирішення таких задач. Для цього існує декілька підходів.

Першим підходом є розробка поліноміальних евристичних алгоритмів. Для деяких евристичних алгоритмів відомі оцінки похибки одержуваного рішення. Такі алгоритми називаються наближеними [27]. Існують наближені алгоритми, що гарантують як відносну похибку, та абсолютну похибку [28, 29]. Деякі NP-складні задачі допускають існування так званої апроксимаційної (спрощувальної) схеми. В рамках даної схеми можна знайти наближене рішення з відносною похибкою не більше будь-якого заданого значення $\varepsilon > 0$ за час, що поліноміально залежить від від $1/\varepsilon$ та від розміру вхідної інформації задачі. Для задач, що не мають апроксимаційної

схеми, велике значення має встановлення граничного значення ε, для якого є можливим знаходження ε-наближеного розв'язку за поліноміальний час.

На даний момент широкого поширення мають метаевристичні алгоритми, які знаходять "гарне" рішення, близьке до оптимального, за прийнятний час. Недоліком таких алгоритмів є відсутність оцінок якості отриманого рішення. Невідомо, наскільки рішення відрізняється від оптимального в найгіршому випадку.

Точним методам рішення NP-складних задач також приділено чималу увагу в роботах з теорії розкладів. Найбільшого поширення набули методи скороченого перебору, звані методами гілок і меж [30-32]. Для скорочення перебору обчислюються нижні оцінки цільової функції (у разі її мінімізації) і використовуються комбінаторні властивості задач. Також для вирішення завдань теорії розкладів широко застосовується метод динамічного програмування [33, 34].

Часто задачі теорії розкладів можуть бути сформульовані як задачі цілочисельного лінійного програмування.

Останнім часом широкого розповсюдження отримав метод програмування в обмеженнях (ПвО, в анлгомовній літературі – Constraint Programming). Однією з областей його успішного застосування є теорія розкладів.

Деякі складні задачі теорії розкладів можуть бути оптимально вирішені за допомогою алгоритмів, що використовують елементи відразу декількох методів. Однією з їх назв являється назва – «Гібридні алгоритми» [35].

В наш час побудова та аналіз наближених алгоритмів з гарнатованою оцінкою точності для NP-складних задач є одним із важливих напрямів досліджень. Цей напрям завоював велику кількість прихильників у середовищі дослідників.

Перерахуємо основні причини:

- існує величезна кількість оптимізаційних задач, які потребують вирішення, і більшість з них NP-складні. Для багатьох з них необов'язково знаходити точні відповіді, а досить побудувати гарне розумне рішення за короткий час, з чим успішно справляються наближені алгоритми;
- на практиці більшість оптимізаційних задач дуже складні, оскільки включають в себе багато додаткових обмежень, які не дозволяють знайти

гарне гарантоване наближене рішення. Але найчастіше наближені алгоритми для більш простих варіантів тих же задач підказують нові ідеї для евристик, які потім успішно застосовуються і для практичних завдань;

- побудова наближених алгоритмів з гарантованою оцінкою точності вимагає аналізу поведінки алгоритмів і виявлення властивостей оптимальних рішень, тобто виявлення структурних властивостей розглянутих задач. У свою чергу краще розуміння структури задачі веде до утворення нових алгоритмічних ідей;
- з точки зору знаходження точних рішень майже все NP-складні задачі еквівалентні один одному. Побудова наближених алгоритмів або доказ неможливості їх побудови за умови розбіжності класів P і NP дає можливість порівняти між собою NP-складні задачі по тому, наскільки добре вони можуть бути спрощені;
- побудова наближених алгоритмів з гарантованою оцінкою точності часто заснована на нових глибоких і цікавих результатах в різних областях математики і сприяє розвитку математики в цілому.

Таким чином, встановлення обчислювальної складності задач теорії розкладів і розробка для NP-складних задач теорії розкладів ефективних наближених алгоритмів з гарантованою оцінкою точності є актуальним напрямком в сучасних наукових дослідженнях.

Мета дослідження – підвищення якості розв’язку задач календарного планування за рахунок побудови оптимального чи близького до оптимального розкладу, що дозволяє мінімізувати сумарний час відхилення від директивних строків.

Для досягнення мети необхідно виконати наступні **задачі**:

- провести аналіз відомих результатів, з розв’язання поставленої в рамках роботи задачі;

- розробити алгоритм створення календарного плану виконання завдань паралельними пристроями, що мінімізує сумарне відхилення моментів завершення завдань від директивних строків;
- виконати програмну реалізацію розробленого алгоритму;
- дослідити ефективність алгоритму при різних вхідних даних шляхом проведення обчислювальних експериментів;

Об’єкт дослідження – процес календарного планування виконання завдань.

Предмет дослідження – моделі та методи розв’язання задач календарного планування з метою мінімізації сумарного відхилення виконання завдань від директивних строків паралельними пристроями.

1 ОГЛЯД ПІДХОДІВ ДО РОЗВ'ЯЗАННЯ ЗАДАЧ З ВИПЕРЕДЖЕННЯМ-ЗАПІЗЕННЯМ ДЛЯ ПАРАЛЕЛЬНИХ ПРИЛАДІВ

Планування роботи завжди було складним завданням у різних сферах людської діяльності. Для вирішення задач випередження-запізнення в теорії розкладів розроблено велику кількість методів дискретної оптимізації, які умовно можна розділити на дві групи: точні та наближені. Умовність цього поділу витікає із того, що багато точних методів можуть застосовуватися як наближені, а наближені методи при певних умовах можуть застосовуватися як точні або їх складова частина [36].

До наближених методів рішення відносяться методи глобального випадкового пошуку і локальних варіацій [37-40]. Характерною особливістю цього методу є те, що на кожній ітерації процес вибору нового рішення виконується не детерміновано, а в результаті реалізації деякого випадкового процесу, при чому стратегія вибору може залежати від передісторії пошуку. Одним із розвитком методів локальної варіації є Tabu-Search-аналіз [41].

До наближених методів також відносяться генетичні алгоритми та еволюційні стратегії. Генетичні алгоритми детально описані у публікаціях [42-50] та є евристичними алгоритмами пошуку, які використовуються для рішення задач оптимізації та моделювання послідовності підбору, комбінації та варіації шуканих параметрів на основі механізмів, які нагадують біологічну еволюцію. Оскільки, алгоритм в процесі пошуку використовує деяке кодування множини параметрів замість самих параметрів, то він може ефективно застосовуватися для рішення задач теорії розкладів, дискретної оптимізації, визначених як на числових множинах, так і на кінцевих множинах вільної природи.

Достатньо широке розповсюдження в наш час отримали різні евристичні методи рішення задачі [51, 52-55]. Основні підходи при побудові цих методів базуються на прийомах зниження вимог та використанні різних видів вирішальних правил, обґрунтованих при рішенні близьких за постановкою задачі, але які не містять цілої множини додаткових складностей та обмежень, характеристик для реально розглянутої проблеми. Евристичні алгоритми використовують різні підходи без

суворих обґрунтувань [56]. Евристичні методи дозволяють знаходити прийнятні рішення навіть у дуже складних випадках: при неповноті, випадковості вихідних даних, відсутності адекватної математичної моделі, NP-складності вирішуваної задачі та відсутності точних методів їх рішення.

Також часто для отримання наближеного рішення задачі використовується «лівостороння» схема розгалуження у методі гілок та меж [57, 58] із зупинкою алгоритму у випадку отримання першого допустимого рішення задачі.

Більшість задач теорії розкладів можуть бути сформульовані як задачі цілочисельного математичного програмування. Частковим випадком яких є задача лінійного цілочисельного програмування [59]. Динамічне програмування [60-62] зазвичай застосовується до вирішенню задач, які можна розбити на послідовність задач меншої розмірності та більш простої структури.

1.1 Огляд наукових досліджень

Задачі випередження-запізнення зазвичай є NP-складними, окремий випадок таких задач розглянуто в публікації [63]. Таким чином знаходження їх оптимального розв’язку є достатньо складним. Проте, докладаються зусилля по розробці ефективних точних методів для вирішення даного типу завдань. Робота над цим питанням проведена у публікаціях [64-71]. У деяких конкретних ситуаціях, часто за участю однієї машини без послідовності, що залежить від часу переналадки, методи здаються дуже ефективними, але більшість з них використовуються для малого/середнього розміру задач.

Перші публікації для задач з випередженням/запізенням почали з’являтись в 1980-х роках, вони були розглянуті Бейкером і Скудером (1990) [72]. Вони приводять огляд відкритої літератури по задачі складання розкладу з випередження/запізнення, описуючи різні моделі і різні штрафи за впередження і запізнення завдань. Пізніше ще одне дослідження було проведено Лауфом і Вернером (2004) [73] для задач випередження-запізнення в середовищі з великою кількістю машин.

Коли директивні терміни завдань визначено, але вони є різними, задачі випередження/запізнення є NP-повними. У роботі [74] надано ефективний алгоритм, що будує розклад з мінімальними витратами, у випадку коли всі завдання мають однакову тривалість.

В теорії розкладів особливе значення мають задачі для одного приладу. Результати, одержані при дослідженні даних завдань, можуть бути використані для побудови алгоритмів вирішення складніших багатоприладних і багатостадійний задач, що виникають на практиці. При цьому, задачі для одного приладу можуть бути використані як для побудови евристичних алгоритмів рішення багатоприладних і багатостадійний завдань, так і для отримання оцінок оптимального значення.

У роботі [64] дослідники розглядають задачу на одному пристрої, у якій кожне завдання має чітко визначений директивний строк та штрафи за випередження і запізнення. Для визначення мінімальних штрафів, для даної задачі пропонується нова нижня межа. Вона заснована на декомпозиції кожної роботи на унарні операції, які потім призначаються за часовими інтервалами (слотами), що дає попередній розклад. Витрати на присвоєння визначаються таким чином, що мінімальна вартість призначення – дійсна нижня межа. Експериментально перевірений метод гілок та меж, що базується на основі цієї нижньої межі та деяких нових правил домінування.

У статті [65] розглянуто задачу складання розкладу для одного пристрою, що враховує штрафи за випередження/запізнення та час на переналадку. Доведено ефективність алгоритму гілок та меж, що спирається на правиа домінування та на евристику, для отримання гарних розкладів.

У роботі [67] запропоновано точний алгоритм для задачі складання розкладу з випередженням-запізненням для одного пристрою, де є дозвіл на час простою для пристрою. Алгоритм є розширенням попереднього алгоритму авторів для задачі без простою машини, який базується на методі SSDP (Successive Sublimation Dynamic Programming). У цьому алгоритмі нижня межа обчислюється шляхом застосування методу динамічного програмування до лагранжевої релаксації вихідної задачі, а потім послідовно покращується шляхом накладання додаткових обмежень на релаксацію, доки розрив між нижньою та верхньою межами не зменшиться. У процесі

застосування алгоритму усуваються непотрібні стани, щоб скоротити обчислювальні зусилля та використання пам'яті. Експериментальні дослідження показують, що запропонований алгоритм може вирішити задачу на 200 завдань для однієї машини.

У роботі [72] надано розширений алгоритм, що застосовується до наступних типів задач: загальна задача мінімізації зважених штрафів за випередження-запізнення, де момент надходження є нульовим, а також з конкретно заданими моментами надходження завдань; загальна задача мінімізації часу завершення виконання робіт з заданими моментами надходження робіт, а також загальна задача зведення до мінімуму запізнення з заданими моментами надходження робіт. Обчислювальні експерименти показують, що запропонований дослідниками алгоритм перевершує існуючі точні алгоритми і може вирішити поставлені задачі на множині до 200 завдань.

Яно і Кім [75] розглянули задачу складання розкладу для статичних задач планування на одній машини, де метою є мінімізація суми зваженого випередження та запізнення. Вони розробили оптимальний та евристичний алгоритм для часткового випадку зважених штрафів, коли ваги є пропорційними до тривалостей виконання завдань. Оптимальний алгоритм використовує властивості домінування для зменшення кількості послідовностей, які необхідно розглянути. Деякі евристичні підходи використовують ці властивості як основу побудови хороших початкових послідовностей. Процедура попарної перестановки, використовується щоб поліпшити початкові евристичні рішення.

Оу і Мортон [76] досліджують задачу складання розкладу для заданого набору завдань на одній машині з метою мінімізації штрафів сумарного випередження та запізнення. Запропоновано та перевірено два правила першочергового пріоритету для цієї NP-повної задачі. Дослідники зробили висновок, що якість рішення значно погіршується, якщо витрати на випередження завдань ігноруються на користь розгляду тільки вартості запізнення.

Девіс і Кейнт [77] розглядають задачу складання розкладу для однієї машини з призначенням штрафів за випередження та запізнення. Представлено та продемонстровано ефективну процедуру складання розкладу, що може бути

включена в перелік алгоритмів, що дозволяють проводити пошук в області перестановок робіт. Планувальник також може бути вбудований в існуюче евристичне рішення для поліпшення якості розкладу випередження/запізнення при низьких обчислювальних витратах.

Шварц і Мухопадьяй [78] розробили підхід групування (кластеризації) для визначення оптимального моменту запуску завдань на прикладі задачі випередження/запізення при припущенні, що послідовність завдань відома апріорно. Підхід групування здійснює оптимальний вибір часу для 500 завдань за кілька секунд обчислювального часу.

Срідхаран і Жоу [79] використовують евристику диспетчеризації для задач випередження/запізнення, що дозволяє вставку допустимого резерву приладу. Евристичний алгоритм, що розроблений для складання розкладів, заснований на теорії прийняття рішень. Ефективність алгоритму була перевірена для 116 опублікованих раніше задач, для яких відомо оптимальне рішення. Також ефективність алгоритму була перевірена шляхом порівняння його з двона добре відомими процедурами диспетчеризації, адаптованих до задач випередження/запізнення.

Алгоритм, описаний в статті [80], заснований методі на табу-пошуку. Запропоновано оптимальний алгоритм синхронізації для визначення часу завершення кожного завдання в заданій послідовності. Процедура табу-пошуку використовується разом з оптимальним алгоритмом синхронізації для створення послідовностей завдань та остаточного розкладу. Обчислювальні експерименти показують, що продуктивність запропонованого підходу досить добра, особливо для задач великого розміру.

Алгоритми для визначення оптимальної послідовності відносно слабко досліджені. Шварцем [81] був запропонований алгоритм гілок і меж, заснований на сортуванні суміжних завдань з різними вагами штрафів. У статті показано, що розташування суміжних завдань у оптимальному розкладі залежить від критичного значення часу початку їх виконання. Виходячи з відношення переваги, у статті

розроблено критерії, за яких задача може бути розбита на менші підзадачі. Алгоритм гілок та меж був випробуваний на 70-ти прикладах розміру n=10.

Лії і Чой [82] розглянули задачу складання розкладу з визначеними директивними строками на одному пристрої. Представлено оптимальний алгоритм синхронізації, який визначає оптимальний час запуску кожної роботи в заданій послідовності робіт. Час простою вставляється між блоками завдань таким чином, щоб функція витрат кожного створюваного робочого блоку була мінімізована. Визначні оптимальні послідовності створюються за допомогою генетичного алгоритму, використовуються оператори кросовера та оператори внутрішньої блокової мутації на основі послідовності. Доведено, що запропонований генетичний алгоритм перевершує існуючі евристичні методи. В середньому, ефект зниження витрат становить 12-33% за еволюційним алгоритмом порівняно з рішенням евристичної процедури. Приблизна оптимальність рішень ГА також ілюструється порівнянням з точним алгоритмом.

Яно і Кім [75] вивчили деякі домінантні властивості послідовностей завдань зі штрафами, пропорційними часу обробки.

Найбільш значущі результати для розкладів із заборонами і дозволами на порушеннями директивних термінів представлені в статтях Гордона [83], Валенте, Альвеса [84], Фельдмана і Біскапа [85], а також Тунг-ай Тсая [86].

У роботі [84] автори представляють алгоритм променевого пошуку для задачі складання розкладу з випередженням/запізненням для одного пристрою. Були розглянуті процедури фільтрації, що використовують як функцію оцінювання пріоритету, так і специфічні властивості. Результати обчислень показують, що алгоритми пошуку відновлюваного пучка перевершують їх аналоги, тоді як процедура фільтрування на основі пріоритетів перевершує альтернативу на основі правил. Найкращі рішення даються алгоритмом пошуку сусідства, але ця процедура є обчислювально-витратною і може застосовуватися тільки для малого або середнього об’єму даних. Пошукова евристика з відновлювальним променем забезпечує результати, які є близькими до якості рішень і значно швидше, тому їх можна використовувати для вирішення навіть великих задач.

У роботі [86] розроблена розроблено генетичний алгоритм, модифікована оптимальна процедура синхронізації, яка починає пошук за допомогою прийнятного рішення, отриманого шляхом застосування правила EXP-ET, розробленого Оу та Мортоном, для проблеми SET – задачі випередження/запізнення для одного пристрою. Обчислювальні результати в наданому експерименті показують, що розроблена ГА покращує рішення SET як у якості, так і в ефективності.

Хугевен і Ван де Вельд [87] запропонували алгоритм, заснований на методі гілок і меж. У зв'язку з тим, що алгоритм має експоненціальне зростання часу обчислення, він може використовуватися тільки для задач з кількістю завдань, не перевершує 20. Складність вирішення завдання великих розмірностей зробила необхідним на практиці майже завжди використовувати наближені алгоритми.

Нещодавно, Ратлі та ін. (2013) [88] представили огляд математичних та евристичних методів, запропонованих для вирішення задач випередження-запізнення на одній машині. У цих трьох роботах, автори розглянули задачі, пов’язані із загальними термінами виконання для всіх робіт. Ціла книга, присвячена плануванню випередження-запізнення була написана Йозефовською (2007) [89], де автор узагальнила основні моделі та алгоритми для цього класу задач, в тому числі з загальними та індивідуальними директивними строками, відповідно. Випадок з індивідуальними директивними строками має не тільки більш загальний характер, але і вважається більш складним, так як випадок з загальним директивним строком має деякі властивості, які полегшують задачу планування рішень. Наприклад, відомо, що час простою не з’являється в оптимальному вирішенні деяких варіантів, де розглядаються загальні директивні строки, що робить задачу більш легкою для вирішення.

У роботі [90] розглянуто метод мінімізації сумарного запізнювання робіт на одиночному пристрої на основі визначення найкоротшого гамільтонового шляху в довільному повнозв’язному графі з використанням рангового підходу і правил домінування. Запропоновано метрики оцінювання поліпшення результатів роботи алгоритму при використанні правил домінування — відносного зменшення сумарного запізнювання і кількості одержуваних локально-оптимальних рішень.

Наведено результати обчислювальних експериментів для зваженого і незваженого запізнювання робіт з довільними директивними термінами. Визначено умови, при яких досліджуваний алгоритм дозволяє покращити розклад робіт, і знайдено найбільш ефективні правила домінування.

Результати розрахунків сумарного зваженого запізнення в діапазоні тривалостей робіт, розподілених по рівномірному закону в інтервалі [1, 10], для різних типів директивних строків показали, що час роботи алгоритму відповідає отриманій у роботі оцінці часової складності. У роботі також приведено результати проведених експериментів і відносне зменшення сумарного зваженого запізнення при різних тривалостях робіт. З них слідує, що у випадку зваженого запізнення кращий розклад виконання робіт отримано при використанні правил домінування WMDD (зважений модифікований директивний строк), ATC (очевидна вартість запізнення) та WMS (зважений мінімальний резерв), які дозволяють зменшити сумарне запізнення для м’яких та середніх директивних строків.

Праця [91] присвячена розгляду методів вирішення задачі мінімізації сумарного запізнення для одного пристрою. Отримано нові властивості задачі, використання яких дозволяє скоротити перебір при побудові розв’язку за допомогою одного відомого алгориму декомпозиційного типу. На основі отриманих властивостей запропоновано поліноміальний алгоритм розв’язку у випадку, коли при будь-якому можливому розкладі кількість робіт, що запізнюються є константою.

Отримані властивості розкладів, вимоги при яких впорядковані в порядку зростання тривалостей обслуговування, на їх основі дано дві оцінки оптимального значення цільової функції. Отримано необхідну і достатню умови того, що один псевдо-поліноміальний алгоритм знаходить оптимальне рішення для деякого прикладу завдання. Показано, що процес побудови розкладу за допомогою алгоритму можна звести до процесу побудови кусково-лінійної функції спеціального виду. Запропоновано модифікацію цього алгоритму, яка дозволяє вирішувати завдання розбиття з нецілочисельними параметрами. Недоліком є відсутність дослідження алгоритму на задачах з паралельними пристроями.

Стаття [92] присвячена задачі впорядкування множини незалежних робіт на для виконання на одному пристрої з метою мінімізації сумарного випередження та запізнення від спільного директивного строку. В даному дослідженні пропонуються методології рішення та властивості оптимального розкладу з метою виявлення знань, які в кінцевому підсумку можуть бути корисними при дослідженні більш складних моделей. Для вирішення цієї задачі запропоновано два методи: метод змішаного цілочисленого лінійного програмування та метод динамічного програмування. Обчислювальні експерименти показали, що метод динамічного програмування є ефективним в отриманні оптимальних рішень і та не має пробем, пов`язаних з вимогами до пам`яті.

ПДС-алгоритми [93] – це алгоритми з поліноміальною та експоненціальною складовими, при цьому в деяких випадках можлива декомпозиція на підзадачі меншої розмірності, кожна з яких вирішується незалежно. Наприклад, в задачі мінімізації сумарного зваженого моменту закінчення виконання завдань [94, p. 2] здійснюється декомпозиція на множини максимального пріоритету, в кожному з яких оптимізація здійснюється незалежно. На відміну від відомих алгоритмів, ПДС-алгоритми відносять вирішувану індивідуальну задачу до підкласу поліноміально вирішуваних в процесі аналізу її вирішення. З цією метою для кожної важкорозв’язуваної задачі строго виводяться логіко-аналітичні умови, що перевіряються в процесі її рішення, в разі виконання яких дана довільна індивідуальна задача точно вирішується поліноміальних алгоритмом. Логіко-аналітичні умови і поліноміальний алгоритм їх перевірки складають поліноміальних складову ПДС-алгоритму.

Вчені Павлов О.А. та Місюра О.Б. у своїй публікації [94] запропонували ефективні ПДС-алгоритми вирішення NP-складних задач випередження-запізнення для випадків, коли момент початку виконання роботи фіксований або знаходиться в інтервалі часу [t_1, t_k]. Рішення базується на ПДС-алгоритмі розв'язання задачі мінімізації сумарного запізнення робіт. Розробили евристичний алгоритм визначення найпізнішого моменту початку виконання робіт, при якому досягається мінімальне значення функціоналу. Навели приклади розв'язання задач і результати експериментальних досліджень. Ефективність розроблених алгоритмів

підтверджується статистичними дослідженнями з великою кількістю прикладів. Оптимальні розклади і самі пізні моменти початку виконання завдань, отримані розробленим і точним алгоритмом, збігаються, що дозволяє зробити висновок про необхідність подальших теоретичних і статистичних досліджень.

У статті [95] розглядається задача мінімізації сумарного зваженого запізнення виконання множини робіт на одному приладі в рамках теорії календарного планування. В даній статті приділяється увага результатам, які можуть бути використані у алгоритмі, що побудований за схемою гілок та меж. Для того, щоб повністю описати алгоритм гілок та меж, потрібно визначитись у виборі наступних елементарних процедур (що разом складають стратегію алгоритму гілок та меж – для задачі мінімізації): процедури галуження конструктивного пошуку, тобто способу розбиття множини допустимих розв'язків у вершині дерева пошуку; процедури оцінювання знизу множин допустимих розв'язків у вузлах дерева пошуку. В деяких випадках виділяють окрему процедуру оцінювання зверху множин допустимих розв'язків. В даній роботі ця процедура стандартна та полягає у фіксуванні рекорду – найкращого значення критерію листового вузла дерева пошуку.

Дана стаття носить методологічний характер, та описує алгоритм гілок та меж, що використовує нові правила домінування та модифіковані процедури оцінювання. В якості нижніх оцінок оптимального значення критерію множин допустимих розв'язків використовуються два варіанти процедур. Процедура Поттса та Ван Васенгова будується на локально оптимальній послідовності відносно попарних перестановок робіт. Для процедури Гелдерса та Клайндорфера визначений найефективніший алгоритм обчислення (результати обчислювальних експериментів до статті не увійшли).

У статті [96] зроблено огляд останніх досягнень в області дослідження задачі мінімізації сумарного зваженого запізнення виконання робіт одним приладом, та був сформульований на їх основі ефективний алгоритм гілок та меж, який призначений для статистичних досліджень характеристик нових алгоритмів, таких, як запропонований ПДС-алгоритм для цієї задачі. Даний алгоритм широко використовує 10 відомих глобальних правил домінування, більшість з яких були встановлені

нещодавно та не згадувались в емпіричних дослідженнях, дві процедури нижньої оцінки оптимального значення критерію, одна з яких є найбільш строгою серед відомих із регульованою складністю обчислень, а інша – обчислюється за лінійний час та дає приріст ефективності на задачах великої розмірності, процедуру отримання локально оптимальної послідовності з метою одержання строгої верхньої оцінки оптимального значення критерію та удосконалення нижньої оцінки на основі лагранжевої релаксації.

У роботі [97] розглянуто задачу складання розкладу виконання одним приладом незалежних робіт з різними тривалостями та директивними термінами за критеріями максимізації моменту запуску робіт і мінімізації сумарного випередження, в якому всі роботи не запізнюються. Для встановленого моменту запуску представлено алгоритм побудови допустимого розкладу з мінімальним сумарним випередженням. Наведено доведення того, що задача побудови допустимого розкладу оптимального одночасно за критеріями максимізації моменту запуску і мінімізації сумарного випередження робіт, заданих у лексиграфічному порядку є P-вирішеною. Запропоновано точний поліноміальний алгоритм визначення допустимого розкладу, оптимального за критерієм мінімізації сумарного випередження для заданого моменту запуску в системі, яка складається з множини незалежних робіт, виконаних на одному приладі.

У роботі [98] розглядається інформаційне забезпечення алгоритмів розв’язання задач складання розкладів за критерієм мінімізації сумарного випередження і запізнення відносно директивних строків: при виконанні незалежних завдань одним приладом при наявності налагоджень залежних від послідовності (МВЗН) та при виконанні груп (МВЗГ) одним приладом із налагодженнями незалежними від послідовності. Описується загальна структура інформаційної системи ієрархічної моделі планування з мережевим представленням технологічних процесів. В основу системи покладена ієрархічна модель планування. Ця модель планування і управління складними системами, яка враховує мережеве представлення технологічних процесів і об- межені ресурси, складається з трьох рівнів: агрегованого, погоджувального і точного планування. Поставлена задача формування планів виконання робіт з

прив’язкою до ресурсів вимагала створити розподілену систему побудови планів для кожного рівня управління. При цьому забезпечується чіткий взаємозв’язок розв’язків, прийнятий на кожному рівні. Розв’язання задачі мінімізації цільової функції на верхньому рівні управління є вхідною інформацією для ефективного розв’язання задач планування по визначеному критерію оптимальності на нижчих рівнях ієрархії. Це дозволило створити систему взаємопов’язаних алгоритмів, що дозволило розв’язувати задачі планування в комплексі.

Для демонстрації ефективності алгоритму в інформаційній системі ієрархічного планування було проведено дослідження залежності часу розв’язування задач від факторів запізнення та діапазону директивних строків, впливу розмірності задач на середній час розв’язування для задачі МВЗН і задачі МВЗГ. У роботі також визначені залежності між параметрами генерованих задач та часом розв’язування допоможуть сформулювати нові обмеження і умови, які можуть прискорити роботу алгоритму.

У [99] розглядається структура системи моделювання для дослідження та оцінки ефективності алгоритмів розв’язання задач складання розкладів за критерієм мінімізації сумарного випередження і запізнення відносно директивних строків: при виконанні незалежних завдань одним приладом за наявності налагоджень, залежних від послідовності, та при виконанні груп завдань одним приладом із налагодженнями, незалежними від послідовності. Для дослідження та оцінки ефективності алгоритмів розв’язання задач складання розкладів за критерієм мінімізації сумарного випередження і запізнення відносно директивних строків, розроблено методику тестування алгоритмів на задачах різної складності. Вибір значень параметрів вхідих задач і їх розмірностей ґрунтується на методиці проведення наукового дослідження для підтвердження результатів, викладених у [100].

Система моделювання, представлена у [99], одночасно з генеруванням, розв’язанням і звітуванням, надає можливість проводити переналаштування параметрів генератора задач залежно від статистичних результатів розв’язання тих задач, які викликатимуть найбільший інтерес. Отже, запропонована схема є дослідницьким середовищем, що дає можливість оперативно отримувати інформацію

про якість роботи алгоритмів на певних класах задач і спрямовувати дослідження у тих напрямках, які виявлятимуть нові особливості розкладів.

Робота [101] присвячена розгляду задачі складання розкладу для однієї машини із обмеженим періодом обслуговування з метою мінімізації суми абсолютних відхилень моментів закінчення робіт від спільного директивного строку. Дана задача сформульована математично як модель змішаного цілочисленного лінійного програмування. У роботі представлено деякі умови оптимальності та описано окремі випадки задачі, що вирішуються поліноміально. Оскільки обчислювальна складність, пов’язана з формулюванням, ускладнює розв’язок стандартних великих задач за прийнятний час, для оптимізації запропоновано алгоритм мурашиних колоній, що базується на результатах дослідження. Обчислювальні результати для різних розмірів задач показують, що застосування запропонованого у [85] алгоритму є ефективним для отримання рішень, близьких до оптимальних.

У [102] розглядається задача складання розкладів за критерієм мінімізації сумарного випередження і запізнення відносно директивних строків при виконанні незалежних завдань одним приладом (МВЗ) при наявності налагоджень. Запропоновано евристичний алгоритм пошуку локального оптимального розв’язку задачі, що розглядається. Експериментальні дослідження показали ефективність розробленого алгоритму, що дозволяє розв‘язувати задачі великої розмірності за короткий час.

Алгоритм, що представляється, складається із двох етапів. На першому етапі (блоки 1 і 2) налагодження не враховуються. У блоці 1 розв‘язується задача мінімізації сумарного запізнення при виконанні незалежних завдань одним приладом (МСЗ) [103]. Алгоритм побудовано на перестановках і полягає в оптимальному використанні завданнями, що запізнюються, резервів завдань, що не запізнюються. Таким чином, реалізується зменшення сумарного запізнення за рахунок зменшення сумарного випередження. У блоці 2 здійснюється зменшення значення сумарного випередження і запізнення за допомогою послідовного збільшення моментів початку виконання завдань.

Для визначення ефективності алгоритму були проведені дослідження залежності часу розв‘язання задачі від загального числа завдань і кількості завдань, що вимагають налагодження приладу, заданого у відсотковому відношенні до загального числа завдань.

У роботі [104] розглядається NP-складна задача у сильному сенсі детермінована задача побудови оптимального розкладу обслуговування множини робіт на одному пристрої за критерієм мінімізації сумарного зваженого запізнення. Розроблено метаевристичний мурашиний алгоритм з комбінованою схемою локального пошуку в двох реалізаціях: послідовний з однією мурашиною колонією (ACO) та на його основі паралельний з декількома незалежними колоніями (PACO). ACO засновується на моделюванні взаємодії деяких штучних аналогів мурав’їв, що програмно представлені у вигляді агентів колонії, шляхом застосування непрямого зв’язку – зв’язків феромона. В задачі складання розкладу за критерієм мінімізації сумарного запізнення, на кожній ітерації агенти складають свої рішення за n кроків, на кожному з яких застосовується правило вибору вимоги j на позицію k в розкладі. Сліди феромону слугують розподіленою чисельно інформацією, котра враховується мурав’ями для конструювання розв’язків задачі і котру мурав’ї адаптивно змінюють для відображення досвіду, накопиченого у процесі пошуку рішення. При цьому кількість феромону, що залишається агентами, пропорціональна якості рішення, що складене відповідним агентом: чим менше значення цільово функції, тим більше буде залишено феромона.

В алгоритмі мурашиної колонії використовуються два правила поновлення слідів феромону: глобальне і локальне. В роботі [105] було запропоновано ефективний спосіб глобального оновлення, який застосовується після побудови рішення агентом і застосування алгоритмів локального пошуку. Оновлення феромона безпосередньо після того, як мурашка додав нову вимогу j до часткової послідовності на позицію k, є локальним.

В роботі [104] реалізовано паралельний метод PACO, в якому одночасно кожна з p мурашиних колоній здійснює незалежний від інших послідовний пошук оптимального рішення, використовуючи свою локальну матрицю сліду феромону.

При цьому відсутній обмін інформацією про сліди феромону, поточні наближені рішення в колоніях і інших тому подібних проміжних даних процесу пошуку рішення. Однак як тільки одна з колоній знайде рішення, про це повідомляється іншим і всі колонії завершують свою роботу.

Запропонований в роботі паралельний метаевристичний гібридний метод вирішення SMTWTP, в якому мурашиний алгоритм поєднується з локальним пошуком, показав свою ефективність – для всіх тестових завдань були отримані оптимальні рішення з невеликим розкидом часу рішення. Даний підхід до побудови алгоритмів може бути використаний при вирішенні інших складних задач комбінаторної оптимізації великої розмірності.

Оскільки, у роботі будуть проводитися дослідження задач складання розкладів саме для паралельних пристроїв (як ідентичних, так і різної продуктивності), то проаналізуємо сучасні досягнення наукової спільноти щодо вирішення задач цього класу із застосуванням різних математичних методів.

Стаття [106] присвячена задачі складання розкладу для паралельних машин, у якій роботи мають директивні терміни та штрафи за випередження/запізнення. Для розв’язання задачі запропоновано нові нижні межі. Також представлено простий локальний алгоритм пошуку для отримання верхньої межі. Обчислювальні експерименти порівнюють ці межі як з задачами для однієї машини, так і для задач з паралельними машинами, і показують, що розрив між верхніми та нижніми межами становить близько 1,5%.

У статті [107] розглядається задача планування, де кожне з n завдань повинне оброблятися без переривань одному з m не пов'язаних паралельних пристроїв. Для кожного завдання встановлюється дається момент запуску та час обробки на кожному пристрої, а загальний директивний строк задано для всіх завдань. Мета полягає в тому, щоб розподілити завдання за таким чином, щоб зважена сума штрафів випередження і запізнення була мінімальною. Для цієї задачі одержано деякі структурні властивості, корисні у зв'язку з пошуком наближеного рішення. Крім того, представлено різні конструктивні та ітераційні евристичні алгоритми, які порівнюються на задачах до 500 завдань та 20 пристроїв.

У роботі [108] запропоновано евристичний алгоритм для широкого класу задач складання розкладу з випередженням/запізненням. Розглянуто задачі для паралельних машин з врахуванням таких особливостей як час простою, час наладки та моменти надходження робіт. Також розглянуто задачі, цільовою функцією яких є мінімізація сумарного (середнього) зваженого часу завершення виконання робіт чи сумарного (середнього) зваженого часу виконання, які виникають як частковий випадок, коли директивні строки виконання всіх завдань встановлюються в нуль чи відповідно до моментів надходження робіт. Розроблено доволі простий, заснований на локальному пошуку, метаевристичний алгоритм, що покладається на складні процедури для ефективного виконання локального пошуку, відповідно до особливостей вхідної задачі. Представлено ефективні підходи для оцінки результативності задач для паралельних пристроїв, що узагальнюють існуючі задачі для одного пристрою.

Робота [109] присвячена дослідженню точних та наближених алгоритмів складання розкладу для мінімаксної задачі оптимального по швидкодії без переривань у багатопроцесорній обчислювальній системі. Для вирішення цієї задачі розроблено нові точні псевдополіноміальні алгоритми (з пошуком в ширину та в глибину) при фіксованому числі процесорів і наближені алгоритми (евристичний та ймовірнісний). Для випадку, коли процесори ідентичні, запропоновано алгоритм, заснований на методі агрегування. Дані алгоритми дозволяють також знаходити розв’язки з гарантованою точністю. У статті приводяться результати обчислювальних експериментів, на основі яких проводиться порівняльний аналіз цих алгоритмів.

У праці [110] розглядається багатогранник розкладів обслуговування частково впорядкованої множини робіт, що мають однакові тривалості обслуговування, паралельними ідентичними приладами. Побудована багатогранна релаксація цього багатогранника, описаний клас правильних нерівностей. Отримані достатні умови опорності побудованих нерівностей до багатогранника. Наводяться приклади, що демонструють певні умови, при яких отримані нерівності можуть служити відсіканнями у відповідних алгоритмах. Також показана поліноміальна розв`язність задачі ідентифікації нерівностей описаного класу.

У роботі [111] дано порівняльний аналіз наближених алгоритмів розв’язку мінімаксної задачі для однорідних пристроїв. Найкращий списків алгоритм порівнюється з генетичним алгоритмом, результати якого є близькими до оптимального розв`язку.

У статті [112] пропонується генетичний алгоритм для розв`язку задачі мінімізації сумарного запізнення робіт для паралельних пристроїв. Проаналізовано дві схожі задачі, що були раніше запропоновані Ху (Hu) [113] для аналогічного дослідження. Було визначено, що продуктивність генетичного алгоритму перевершує підхід, який використовує Ху і дуже близький до процедури вичерпного пошуку. Показано, що реалізація електронних таблиць генетичного алгоритму дозволяє дуже легко пристосувати задачу до будь-якого набору показників цільової функції без зміни фактичної моделі. Було проведено емпіричний аналіз для вивчення впливу параметрів генетичного алгоритму, а саме кросоверу, швидкості мутацій та розміру популяцій.

У статті [114] розглянуто задачу впорядкування множини робіт до множини ідентичних паралельних пристроїв з метою зведення до мінімуму сумарно зваженого випередження та запізнення відносно спільного директивного терміну. Запропоновано гібридний евристичний алгоритм для побудови прийнятного розв’язку, поєднуючи правила пріоритетів призначення робіт до пристроїв та локальний пошук з точними процедурами для вирішення підзадачі для одного пристрою.

Робота [115] присвячена дослідженню задачі складання розкладу за критерієм мінімізації сумарних зважених штрафів випередження та запізнення на ідентичних паралельних машинах згідно загального директивного терміну. Запропоновано алгоритм руйнування та відтворення (ruin-and-recreate – FR&R) для отримання прийнятних рішень цієї задачі.

1.2 Методи вирішення задач складання розкладів на паралельних пристроях

Для вирішення задач теорії розкладів розроблено велику кількість методів дискретної оптимізації, які умовно можна розділити на дві групи: точні та наближені. Умовність цього поділу витікає із того, що багато точних методів можуть застосовуватися як наближені, а наближені методи при певних умовах можуть застосовуватися як точні або їх складова частина [36].

Найбільшого розповсюдження отримали наступні підходи та методи [116]:

– точні методи:
  1) методи цілочисельного програмування:
     a) лінійне цілочисельне програмування;
     b) нелінійне цілочисельне програмування; булеве програмування;
  2) послідовні алгоритми оптимізації:
     a) метод гілок та меж;
     b) динамічне програмування та методи аналізу і відсіву варіантів;
     c) методи теорії графів;
– наближені методи:
  1) обмеження об’єму розрахунків в послідовних алгоритмах оптимізації;
  2) методи випадкового пошуку:
     a) методи глобального випадкового пошуку,
     b) методи локальної варіації;
  3) генетичні алгоритми та еволюційні стратегії;
  4) евристичні методи;
  5) гибридні алгоритми;
  6) метаевристичні методи;
  7) імітаційне моделювання.

Логічне програмування в обмеженнях. Складання розкладу можна уявити як завдання задоволення обмежень. Для вирішення таких завдань розроблено безліч алгоритмів, починаючи з класичного методу Гаусса і закінчуючи складними

методами, застосовуваними в системах доведення теорем і в системах символьних обчислень. Виник навіть цілий напрям в програмуванні – програмування в обмеженнях (constraint programming).

Програмування в обмеженнях тісно пов'язане з традиційним логічним програмуванням, в рамках якого воно і сформувалося. Більшість систем програмування в обмеженнях є звичайним інтерпретатор Прологу з вбудованим механізмом для вирішення певного класу задач задоволення обмеженнь. Програмування в таких системах називають логічним програмуванням в обмеженнях (Constraint Logic Programming або CLP).

Ідея вирішення завдань така, що програміст визначає деяку множину змінних $x_1,...,x_n$ і області їх значень $X_1,...,X_n$, описує додаткові обмеження, яким повинні задовольняти змінні, а система знаходить відповідні значення змінних, які одночасно задовольнятимуть всім заданим обмеженням.

Результатом роботи алгоритму буде множина значень кожної змінної, що не суперечать зазначеним обмеженням. При цьому область визначення змінних, що беруть участь в строгих обмеженнях, може істотно скоротитися або навіть містити єдине значення.

Основна перевага, що отримується при використанні CLP, – це скорочення простору пошуку, що досягається не шляхом оцінки кожного варіанти розкладу, а за рахунок того, що система сама виключає з розгляду «дороги, що свідомо ведуть в тупик».

Використання CLP, в комбінації з інтелектуальним пошуком, для вирішення задачі складання розкладу описано в роботі [117].

Генетичні алгоритми. Розглянуті методи в своїй основі використовують ітераційну техніку поліпшення результатів. Протягом однієї ітерації вони шукають рішення, краще в околицях даного. Якщо таке рішення знайдено, воно стає поточним і починається нова ітерація. Це продовжується до тих пір, поки приріст цільової функції не зменшиться практично до нуля або не виконається задану кількість ітерацій. Очевидно, що такі методи орієнтовані на пошук тільки локальних

оптимумів, причому положення знайденого оптимуму залежить від стартової точки. Глобальний же оптимум може бути знайдений тільки випадково. Для підвищення ймовірності знаходження глобального оптимуму використовується множинний експеримент з різними початковими точками, що істотно збільшує час пошуку.

У зв'язку з цим становить інтерес розробка алгоритмів, які зберігали переваги описаних методів і вільних від зазначеного недоліку. До таких алгоритмів відносяться генетичні алгоритми.

Генетичні алгоритми – це стохастичні евристичні оптимізаційні методи, основна ідея яких була взята з теорії еволюційного розвитку видів. Основним механізмом еволюції є природний відбір, суть якого полягає в тому, що більш пристосовані особини мають більше шансів на виживання і розмноження і, отже, приносять більше потомства, ніж менш пристосовані особини. При цьому завдяки передачі генетичної інформації нащадки успадковують від батьків основні їх якості. Носіями генетичної інформації індивідуума виступають молекули ДНК. При розмноженні тварин відбувається злиття двох батьківських статевих клітин. Їх ДНК взаємодіють, утворюючи ДНК нащадка. Основний спосіб взаємодії – кросинговер. При кросинговері ДНК предків діляться на дві частини, а потім обмінюються своїми половинками. При успадкування можливі мутації через радіоактивності або інших впливів, в результаті яких можуть змінитися деякі гени в статевих клітинах одного з батьків. Змінені гени передаються нащадку і надають йому нові властивості. Якщо ці нові властивості корисні, вони, швидше за все, збережуться в даному виді і при цьому відбудеться стрибкоподібне підвищення пристосованості виду.

Першим кроком при розробці математичної моделі, побудованої на генетичному алгоритмі, є розробка структури хромосоми, в якій буде зберігатися розв’язок. У нашому випадку такою «хромосомою» є розклад. Обрана структура повинна враховувати всі особливості і обмеження, що пред’являються до шуканого розкладу, а також те, що від її вибору безпосередньо залежать реалізації алгоритмів кросинговеру і мутації. В остаточному підсумку, вибір «хромосоми» впливає не тільки на швидкість, але і на збіжність алгоритму взагалі.

Одним з найбільш зручних уявлень вирішення даної задачі є [118] тривимірна матриця, по осях i, j, k якої відкладаються відповідно роботи, тривалості робіт і машини [119]. Елементом матриці є запит на виконання певної роботи з певною тривалістю на певній машині.

Така структура хромосоми зручна тим, що вже на етапі зазначення початкових даних можна виключити свідомо невдалі рішення, заблокувавши відповідні варіанти. На наступному кроці алгоритму створюється початкова популяція, розмір якої залежить від розмірності задачі і становить зазвичай кілька сотень рішень.

Для організації оптимізуючого процесу необхідно створити спрямовуючу силу розвитку популяції. В якості такої сили виступає вимога мінімізації цільової функції або, в термінах генетичних алгоритмів, фітнес-функції. Зазвичай в якості її використовується адитивний показник оптимальності, заснований на штрафах, встановлюваних кожному рішенню за будь-якої незручний момент в розкладі. Перевагою такого вибору є можливість налаштування алгоритму під конкретну задачу шляхом варіювання коефіцієнтів і, тим самим, зміни пріоритетів при пошуку оптимального розкладу.

Таким чином, помістивши початкову популяцію в створену нами штучне середовище і реалізувавши процеси селекції, кросинговеру і мутації, ми отримаємо ітераційний алгоритм пошуку оптимального рішення, на кожній ітерації якого виконуються наступні дії:

Етап 1. Кожна особа популяції оцінюється за допомогою фітнес-функції.

Етап 2. Кращі рішення (зазвичай близько 5%) копіюються в нову популяцію без зміни. Такий принцип (принцип елітизму) попереджує втрати кращих рішень і забезпечує підвищену збіжність алгоритму.

Етап 3. На основі пропорційного відбору з поточної популяції вибираються два рішення, які піддаються рекомбінації. Для цього хромосоми батьків обмінюються відповідними сегментами.

Етап 4. Отриманий в попередньому пункті розклад може виявитися некоректним, наприклад, може не відповідати певним обмеженням. В

цьому випадку можна повторювати операцію рекомбінації до тих пір, поки не буде отримано коректний розклад, але бажаніше передбачити евристичні механізми виправлення розкладу.

Етап 5. Якщо нова популяція сформована, то стара видаляється, після чого переходимо до етапу 1. В іншому випадку переходимо до етапу 3.

Розглянутий алгоритм є не тільки стійким до локальних мінімумів, але і завдяки внутрішньому паралелізму, вираженого в роботі не з окремими рішеннями, а з цілими класами рішень, забезпечує відносно швидкий пошук оптимального рішення.

В статті [120] розглянута задача складання розкладу для ідентичних паралельних машин для мінімізації загального часу виконання та присвоєння робіт на машинах. Для вирішення поставленої задачі використовується генетичний алгоритм.

Алгоритм розфарбування графа. Завдання складання розкладу можна розглядати як задачу розфарбування графа. Нагадаємо, що завданням розфарбування графа називають пошук хроматичного числа графа або, іншими словами, пошук мінімального числа кольорів, необхідних для розфарбування вершин деякого графа з використанням для кожної пари сусідніх вершин різних кольорів. Сама задача пошуку хроматичного числа є NP-повною задачею, для вирішення якої в більшості випадків використовуються різні жадібні алгоритми.

Для постановки задачі складання розкладу як завдання розфарбування графа будується граф, в якому кожна вершина являє собою заплановане навчальним планом заняття. У тому випадку, якщо між якимись двома вершинами можливі конфлікти, наприклад, обидва заняття проводяться в одній аудиторії або з одним викладачем, то вони з'єднуються ребром. Це еквівалентно забороні одночасного проведення цих занять. Тоді задача складання розкладу представляється як мінімізація числа кольорів, необхідних для розфарбування графа. Кожен колір відповідає одному періоду розкладу.

Застосування цього підходу для вирішення реальних завдань, малоефективне. У той же час, задача розфарбування графа при складанні розкладів може виявитися

корисною в разі її комбінації з іншими алгоритмами, як це показано в роботі [121] на прикладі складання розкладу іспитів для Ноттінгемського університету.

Висновки до розділу

В даному розділі було наведено огляд існуючих підходів до вирішення задач на паралельних пристроях. Найбільш розповсюдженими методами для розв'язання цього класу задач є наближені методи, а саме: методи випадкового пошуку, генетичні алгоритми, гібридні алгоритми.

В процесі аналізу було з'ясовно, що більшість робіт попередників були направлені на дослідження задач, які полягають у мінімізації випередження чи запізнення для одного пристрою. Ті, що розраховані на декілька пристроїв, або розглядають випадки тільки з ідентичними пристроями, або ж не розглядають випадки, коли є декілька директивних строків та момент початку складання розкладу є фіксованим. Тому доцільність проведення досліджень у даному напрямку є очевидною.

2 ПОЛІНОМІАЛЬНО РОЗВ'ЯЗУВАНІ ЗАДАЧІ МІНІМІЗАЦІЇ СУМАРНОГО ВІДХИЛЕННЯ ВІД ДИРЕКТИВНОГО СТРОКУ

Зачасту ефективні наближені алгоритми розв'язання задач класу NP базуються на ідеях та підходах, використаних при розв'язанні задач класу P. Тому спочатку розглянемо частковий випадок, який є поліноміально розв'язним для одного пристрою та декількох ідентичних пристроях з одним директивним строком.

2.1 Сумарне відхилення від директивного строку для одного пристрою

2.1.1 Формалізація задачі

Нехай маємо деякий розклад виконання завдань одним пристроєм. Введемо наступні позначення:

Q – множина завдань, що запізнюються: $Q = \{i \mid C_i > d\}$, $|Q| = q$;

$\lfloor 1 \rfloor$ – номер завдання, що є першим за порядком в послідовності завдань, що запізнюються;

$\lfloor i \rfloor$ – номер завдання, що є i-тим за порядком в послідовності завдань, що запізнюються;

W – множина завдань, що виконуються з випередженням: $W = \{i \mid C_i \le d\}$), $|W| = w$ (завдання, у якого випередження дорівнює 0, будемо відносити до множини W);

$\lceil 1 \rceil$ – номер завдання, що є першим в послідовності завдань множини W, якщо рахувати від директивного строку справа наліво по часової осі (є першим з кінця в упорядкуванні завдань множини W);

$\lceil i \rceil$ – номер завдання, що є i-тим в послідовності завдань множини W, якщо рахувати від директивного строку справа наліво по часової осі (є i-тим з кінця в упорядкуванні завдань множини W).

Будемо вважати, що немає завдання, яке «розривається» директивним строком (тобто починається до директивного строку, а закінчується після нього).

Передбачається також, що момент запуску $u = d - \sum_{i \in W} p_i$ може бути більше 0 (тобто початок розкладу може бути зсунений вправо від нульового моменту часу) і директивний строк має таке значення, що є достатній люфт для можливих значень моменту запуску.

Рис. 2.1 ілюструє введені позначення.

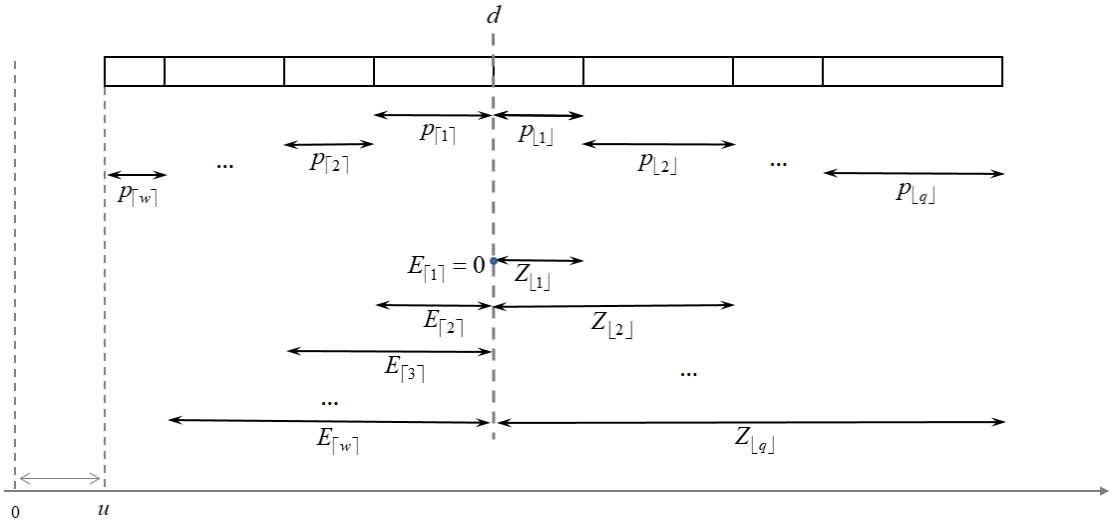

Рисунок. 2.1. Основні позначення та нумерація завдань у розкладі

2.1.2 Властивості задачі

Сумарне випередження завдань у розкладі:

$$E = \sum_{i \in W} E_i = \sum_{k=1}^{w} E_{\lceil k \rceil} = E_{\lceil 1 \rceil} + E_{\lceil 2 \rceil} + E_{\lceil 3 \rceil} + \ldots E_{\lceil w \rceil} = 0 + p_{\lceil 1 \rceil} + (p_{\lceil 1 \rceil} + p_{\lceil 2 \rceil}) + \ldots + (p_{\lceil 1 \rceil} + p_{\lceil 2 \rceil} + \ldots + p_{\lceil w-1 \rceil}) = \\ = \sum_{k=1}^{w-1} \sum_{i=1}^{k} p_{\lceil k \rceil} = \sum_{i=1}^{w-1} (w-i) p_{\lceil i \rceil}. \quad (2.1)$$

Сума $\sum_{i=1}^{w-1} (w-i) p_{\lceil i \rceil}$ не містить доданка, що відповідає $p_{\lceil w \rceil}$ (оскільки ця величина не впливає на значення $E_{\lceil 1 \rceil}, E_{\lceil 2 \rceil}, \ldots, E_{\lceil w \rceil}$), введемо доданок, що відповідає $p_{\lceil w \rceil}$, в цю суму з коефіцієнтом 0: $E = \sum_{i=1}^{w} (w-i) p_{\lceil i \rceil}$.

Сумарне запізнення завдань:

$$Z = \sum_{i \in Q} Z_i = \sum_{k=1}^{q} Z_{\lfloor k \rfloor} = \sum_{k=1}^{q} \sum_{i=1}^{k} p_{\lfloor k \rfloor} = \sum_{i=1}^{q} (q - i + 1) p_{\lfloor i \rfloor} \quad (2.2)$$

Отже, цільова функція (ЦФ) визначається так:

$$E + Z = \sum_{i=1}^{w} (w - i) p_{\lceil i \rceil} + \sum_{i=1}^{q} (q - i + 1) p_{\lfloor i \rfloor} \quad (2.3)$$

Як бачимо, $E + Z$ являє собою суму n доданків, кожний з яких є добутком цілого числа на тривалість завдання. Кожна з n тривалостей бере участь у сумуванні один раз, а цілі числа (коефіцієнти цільової функції при тривалостях) мають такі значення:

$$\begin{aligned} &w-1, w-2, \ldots, 2, 1, 0 \\ &q, q-1, \ q-2, \ldots, 2, 1. \end{aligned} \quad (2.4)$$

На рисунку 2.2 показано відповідність цих чисел позиціям завдань у розкладі (позицією будемо вважати порядковий номер завдання у розкладі).

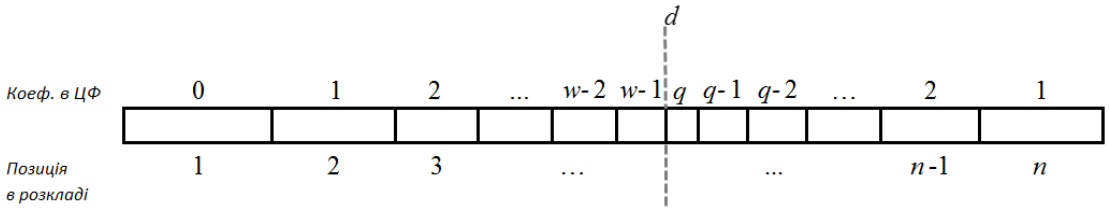

Рисунок 2.2. Відповідність коефіцієнтів ЦФ та позицій завдань у розкладі

Відомо, що сума попарних добутків двох числових послідовностей приймає мінімальне значення, коли одна з послідовностей не спадає, а інша – не зростає [10]. Отже, величина сумарного відхилення $E + Z$ буде мінімальною, якщо послідовність тривалостей не спадатиме, а послідовність цілих чисел не зростатиме. З урахуванням цього можна виділити такі властивості оптимального упорядкування завдань:

1) існує тільки один коефіцієнт зі значенням 0 і він повинен бути при найбільшій тривалості p_i;

2) існує два коефіцієнти зі значеннями 1 і вони повинні бути при двох наступних найбільших значеннях p_i; це означає, що:

– одне з цих завдань повинно належати множині Q (запізнюватись), тобто воно повинно бути останнім в упорядкуванні;

– інше завдання повинно належати множині W (випереджати директивний строк), тобто воно в оберненому упорядкуванні повинно бути передостаннім (якщо рахувати від директивного строку справа наліво), тобто фактично бути другим в загальній послідовності;

– байдуже, яке з цих завдань буде запізнюватись, а яке буде виконуватись з випередженням;

3) існує два коефіцієнти зі значеннями 2 і вони повинні бути при двох наступних найбільших значеннях p_i і так далі;

4) якщо n є непарним числом, то $w-1=q=\frac{n-1}{2}$ (або $w=q+1$), тобто кількість завдань, що виконуються з ненульовим випередженням, дорівнює кількості завдань, що запізнюються (рим. 2.3); якщо n є парним числом, то $w=q$ (рисунок 2.4);

5) є одне завдання $i=\lceil 1 \rceil$ у якого $E_i+Z_i=0$;

З урахування властивостей 2) - 3) кількість альтернативних оптимальних розкладів становить $2^{(n-2)/2}$, якщо n є парним числом, і $2^{(n-1)/2}$, якщо n є непарним.

Згідно [122] існує процедура поліноміального розв'язання за умови:

$$d \geq \Delta = p_n + p_{n-2} + \ldots + p_{n-2\lceil (n-2)/2 \rceil}. \qquad (2.5)$$

Для будь якого розкладу справедливо наступна умова:

$$C_j \geq p_j, j = 1,2\ldots,n, \qquad (2.6)$$

яка означає те, що жодне завдання не може розпочатись до нульового моменту часу.

При виконанні умови поліноміальної розв'язуваності (2.6), для мінімізації сумарного відхилення необхідно розподіляти завдання на пристрої за правилом «найдовшому завданню повинен відповідати найменший коефіцієнт ЦФ».

Нехай завдання пронумеровані за незростанням тривалостей: $p_1 \geq p_2 \geq \ldots \geq p_n$.

На рисунку 3.1 показані варіанти оптимальних розкладів виконання завдань для непарного та парного n (тут і далі на діаграмах Ганта у прямокутниках вказані номери завдань, а над ними – відповідні їм коефіцієнти цільової функції). Так у розкладі для непарного n (рис. 2.3) завдання, що включаються до множини Q (запізнюються), мають парні номери від 2 до n-1, а завдання, що включаються до множини W (випереджають), мають непарні номери від 1 до n. Якщо поміняти місцями завдання, яким відповідають однакові значення коефіцієнтів ЦФ, сумарне значення відхилень не зміниться (розклад залишиться оптимальним).

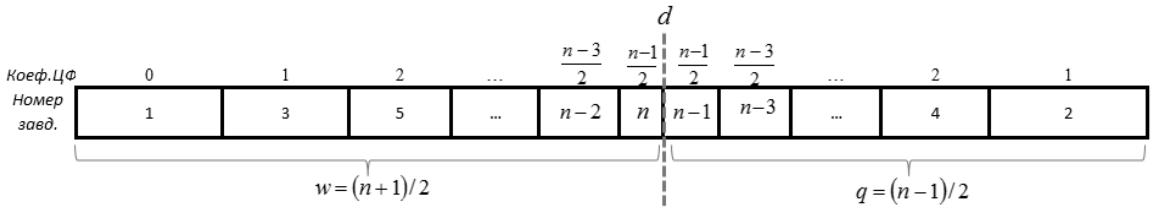

Рисунок 2.3 – n є непарним числом

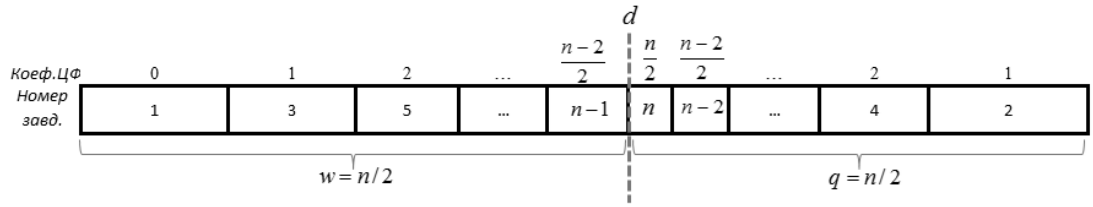

Рисунок 2.4 – n є парним числом (варіант 1)

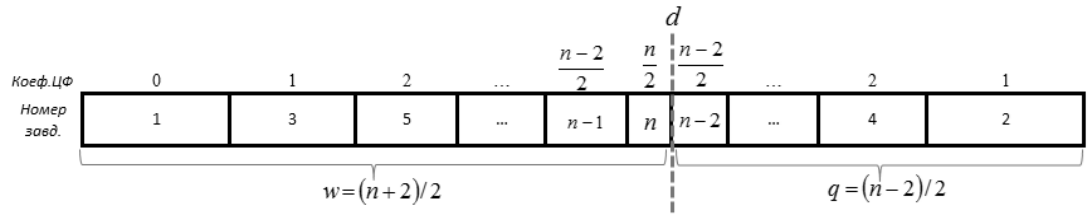

Рисунок 2.5 – n є парним числом (варіант 2)

Зсув розкладу вліво/вправо на величину Δ ($\Delta < p_{\lceil 1 \rceil}$ і $\Delta < p_{\lfloor 1 \rfloor}$), який призводить до «розриву» одного з завдань директивним строком, не призводить до покращення значення цільової функції. Тому, достатньо розглядати лише ті розклади, в яких завдання «не розриваються» директивним строком, тим самим зменшуючи можливу кількість альтернативних розкладів.

2.1.3 Алгоритм розв'язання задачі

Оптимальний розклад отримуємо за наступним алгоритмом:

КРОК 1**.** Впорядкувати завдання за незростанням тривалостей: $p_1 \geq p_2 \geq \ldots \geq p_n$**.**

КРОК 2. $l_w = 1$ – ініціалізація порядкового номеру позиції для завдань, що виконуються з випередженням.

$l_q = n$ - ініціалізація порядкового номеру позиції для робіт, що запізнюються.

КРОК 2. Визначення кількості завдань множин *Q* та *W*.

ЯКЩО n – непарне, ТО $q = \frac{n-1}{2}$, а $w = q + 1 = \frac{n+1}{2}$

ІНАКШЕ $q = w = \frac{n}{2}$

КРОК 3.

ЦИКЛ по завданням *i* від 1 до *n*

ЯКЩО i – непарне ($i \bmod 2 != 0$)

ТО включити завдання *i* до множини *W*, на позицію l_w.

$$l_w = l_w + 1$$

ІНАКШЕ поставити завдання *i* до множини *Q*, на позицію l_q.

$$l_q = l_q - 1$$

2.2 Сумарне відхилення від директивного строку для декількох ідентичних пристроїв

2.2.1 Формалізаця задачі

Розглянемо тепер випадок, коли в системі є $m > 1$ ідентичних пристроїв. Будемо розглядати розклади, в яких немає завдань, що починаються до директивного строку і закінчуються після нього.

Введемо позначення:

Q_j – множина завдань, що запізнюються на пристрої j, $|Q_j| = q_j$, $j = \overline{1,m}$;

$j\lfloor i \rfloor$ – номер завдання, що виконується на пристрої j та є i-тим за порядком в послідовності завдань, що запізнюються на цьому пристрої;

W_j – множина завдань, що на пристрої j виконуються з випередженням, $|W_j| = w_j$, $j = \overline{1,m}$ (завдання, у якого на пристрої j випередження дорівнює 0, будемо відносити до множини W_j);

$j\lceil i \rceil$ – номер завдання, що є i-тим з кінця в упорядкуванні завдань з множини W_j (на пристрої j є i-тим в послідовності завдань множини W_j, якщо рахувати від директивного строку справа наліво по часовій осі).

З урахуванням введених позначень визначимо сумарне відхилення моментів закінчення від директивного строку (суму значень випереджень та запізнень):

$$E + Z = \sum_{j=1}^{m} \sum_{i=1}^{w_j} E_{j\lceil i \rceil} + \sum_{j=1}^{m} \sum_{i=1}^{q_j} Z_{j\lfloor i \rfloor}, \tag{2.7}$$

Сумарне випередження:

$$E = \sum_{j=1}^{m} \sum_{i=1}^{w_j} E_{j\lceil i \rceil} = \sum_{j=1}^{m} \sum_{k=1}^{w_j - 1} \sum_{i=1}^{k} p_{\lceil k \rceil} = \sum_{j=1}^{m} \sum_{i=1}^{w_j - 1} (w_j - i) p_{j\lceil i \rceil}, \tag{2.8}$$

Сума $\sum_{j=1}^{m} \sum_{i=1}^{w_j - 1} (w_j - i) p_{j\lceil i \rceil}$ не містить доданків, що відповідають завданням $p_{j\lceil w_j \rceil}$ (оскільки їх тривалості не впливають на значення $E_{j\lceil 1 \rceil}, E_{j\lceil 2 \rceil}, \ldots, E_{j\lceil w_j \rceil}$), введемо в ці суми доданки, що відповідають $p_{j\lceil w_j \rceil}$, з коефіцієнтами 0: $\sum_{j=1}^{m} \sum_{i=1}^{w_j} (w_j - i) p_{j\lceil i \rceil}$

Сумарне запізнення:

$$Z = \sum_{j=1}^{m} \sum_{i=1}^{q_j} Z_{j\lfloor i \rfloor} = \sum_{j=1}^{m} \sum_{k=1}^{q_j} \sum_{i=1}^{k} p_{j\lfloor k \rfloor} = \sum_{j=1}^{m} \sum_{i=1}^{q_j} (q_j - i + 1) p_{j\lceil i \rceil}., \tag{2.9}$$

Сумарне відхилення:

$$E+Z=\sum_{j=1}^{m}\sum_{i=1}^{w_j}(w_j-i)p_{j\lceil i\rceil}+\sum_{j=1}^{m}\sum_{i=1}^{q_j}(q_j-i+1)p_{j\lceil i\rceil}, \tag{2.10}$$

2.2.2 Властивості задачі

$E+Z$ являє собою суму n доданків $\left(n=\sum_{j=1}^{m}\left(w_j+q_j\right)\right)$, кожний з яких є добутком цілого числа на тривалість завдання. Кожна з n тривалостей бере участь у сумуванні один раз, а цілі числа, на які множаться тривалості (коефіцієнти цільової функції), мають значення, що показані в таблиці 2.1.

Таблиця 2.1 – Значення коефіцієнтів ЦФ (множників при тривалостях завдань)

Номер пристрою	Коефіцієнти ЦФ	
	Завдання, що виконуються з випередженням (W)	Завдання, що запізнюються (Q)
1	$w_1-1, w_1-2,\ldots,2,1,0$	$q_1, q_1-1,\ldots,2,1$
2	$w_2-1, w_2-2,\ldots,2,1,0$	$q_2, q_2-1,\ldots,2,1$
…	…	…
m	$w_m-1, w_m-2,\ldots,2,1,0$	$q_m, q_m-1,\ldots,2,1$

Як бачимо, кількість завдань з коефіцієнтами 0 становить m, з коефіцієнтами 1 - становить $2m$, з коефіцієнтами 2 – становить $2m$ і так далі. На рисунку 2.6 показано відповідність цих коефіцієнтів позиціям завдань у розкладі.

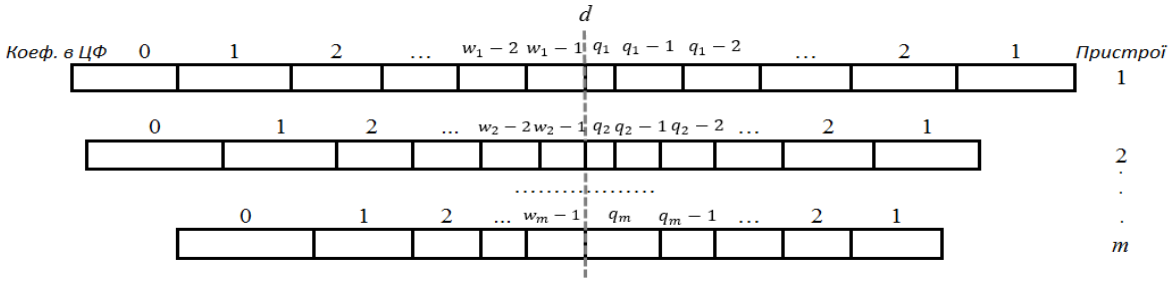

Рисунок 2.6 – Відповідність коефіцієнтів ЦФ позиціям завдань у розкладі

Встановимо значення w_j, q_j, $j=\overline{1,m}$. Ці значення залежать від цілочисельності/нецілочисельності та парності/непарності частки ділення $\frac{n}{m}$.

В таблиці 2.2 наведені значення кількостей завдань, що запізнюються та виконуються з випередженням на пристроях. В таблиці через $[\alpha]$ позначено цілу частину числа α - найбільше ціле, що задовольняє нерівності $[\alpha] \le \alpha$.

Таблиця 2.2 – Цілочисельність/нецілочисельність, парність/непарність частки ділення $\frac{n}{m}$

<table>
<tr><th></th><th>Парність/ непарність</th><th>Значення w_i</th><th>Значення q_i</th></tr>
<tr><td rowspan="2">$n \bmod m = 0$</td><td>$\frac{n}{m}$ - не парне</td><td>$w_1 = w_2 = ... = w_m = \frac{\frac{n}{m}+1}{2}$</td><td>$q_1 = q_2 = ... = q_m = \frac{\frac{n}{m}-1}{2}$</td></tr>
<tr><td>$\frac{n}{m}$ - парне</td><td>$w_1 = w_2 = ... = w_m = \frac{n}{2m}$</td><td>$q_1 = q_2 = ... = q_m = \frac{n}{2m}$</td></tr>
<tr><td rowspan="6">$n \bmod m = s$</td><td rowspan="3">$\frac{n}{m}$ - не парне</td><td colspan="2">для s пристроїв</td></tr>
<tr><td>$w_1 = w_2 = ... = w_s = \frac{\left[\frac{n}{m}\right]+1}{2}$</td><td>$q_1 = q_2 = ... = q_s = \frac{\left[\frac{n}{m}\right]-1}{2}+1$</td></tr>
<tr><td colspan="2">для $m-s$ пристроїв</td></tr>
<tr><td>$w_{s+1} = ... = w_m = \frac{\left[\frac{n}{m}\right]+1}{2}$</td><td>$q_{s+1} = ... = q_m = \frac{\left[\frac{n}{m}\right]-1}{2}$</td></tr>
<tr><td rowspan="3">$\frac{n}{m}$ - парне</td><td colspan="2">для s пристроїв</td></tr>
<tr><td>$w_1 = w_2 = ... = w_s = \frac{\left[\frac{n}{m}\right]}{2}+1$</td><td>$q_1 = q_2 = ... = q_s = \frac{\left[\frac{n}{m}\right]}{2}$</td></tr>
<tr><td colspan="2">для $m-s$ пристроїв</td></tr>
<tr><td></td><td>$w_{s+1} = ... = w_m = \frac{\left[\frac{n}{m}\right]}{2}$</td><td>$q_{s+1} = ... = q_m = \frac{\left[\frac{n}{m}\right]}{2}$</td></tr>
</table>

Для мінімізації сумарного відхилення необхідно розподіляти завдання між пристроями за правилом «найдовшому завданню повинен відповідати найменший коефіцієнт ЦФ».

2.2.3 Алгоритм розв'язання задачі

Алгоритм розподілу завдань між паралельними пристроями полягає у наступному:

КРОК 1. Впорядкувати завдання за незростанням тривалостей ($p_1 \ge p_2 \ge ... \ge p_n$).

КРОК 2. Впорядкувати значення коефіцієнтів ЦФ за неспаданням.

КРОК 3. ЦИКЛ по завданням

Завдання i (завдання з поточної множини непризначених завдань, яке має максимальну тривалість) призначити на пристрій з найменшим вільним коефіцієнтом ЦФ (який визначає позицію завдання в упорядкуванні відповідного пристрою).

Згідно алгоритму, спочатку заповнюються перші позиції всіх пристроїв (їм відповідають коефіцієнти 0), потім останні позиції всіх пристроїв (їм відповідають коефіцієнти 1), після цього – другі позиції всіх пристроїв (їм відповідають коефіцієнти 1) і т.д.

Згідно алгоритму, його трудомісткість визначається такими процедурами: сортування тривалостей завдань (трудомісткість якої $n \cdot \log n$), сортування коефіцієнтів ЦФ (трудомісткість якої також $n \cdot \log n$). Отже трудомісткість алгоритму така: $O(n \cdot \log n)$.

Приклад 2.1

Задано $n = 25$ завдань та $m = 3$ ідентичних пристроїв. Тривалість виконання завдань задано вектором:

$$p = \{20,18,16,13,12,12,11,11,10,10,10,9,8,8,8,7,6,6,5,5,4,3,3,3,2\}.$$

На рис.6 показано оптимальний розклад виконання цих завдань (у прямокутниках вказано номер завдання та тривалість його виконання на пристрої, над прямокутниками вказані значення коефіцієнтів ЦФ).

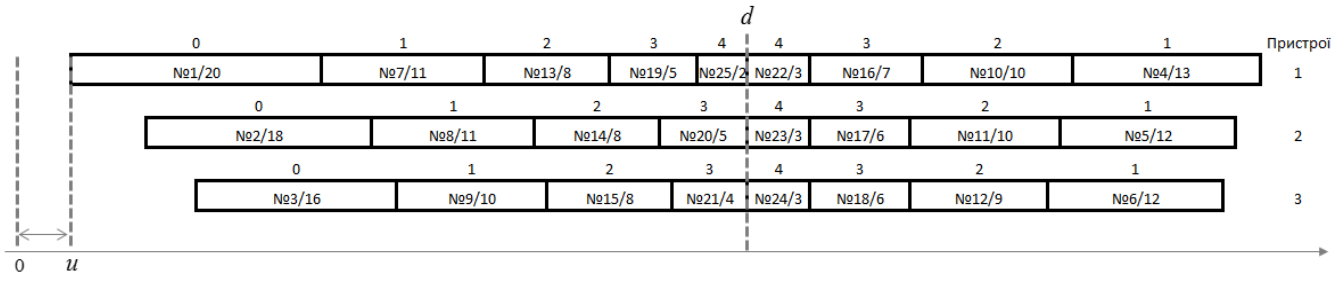

Рисунок 2.7 – Оптимальний розклад для системи з прикладу 1

Висновки до розділу

У даному розділі представлено розв'язання часткових випадків задачі складання розкладу з метою мінімізації сумарного відхилення від директивних строків.

Передбачалось, що початок розкладу (момент запуску) може бути зсунений вправо від нульового моменту часу на будь-який термін, тобто директивний строк має достатньо велике значення і допускає будь-які можливі значення моменту запуску розкладів. Це допущення дозволило розробити для задач, що розглядались, поліноміальні алгоритми побудови оптимальних розкладів.

Представлено алгоритм розв’язання задчі для одного та декількох ідентичних приладів відносно одного директивного строку. Отримані у цьому розділі результати будуть використані для розв’язання загальної задачі, що розглядається у дисертаційній роботі.

3 СКЛАДАННЯ РОЗКЛАДУ ВИКОНАННЯ ЗАВДАНЬ ПАРАЛЕЛЬНИМИ ПРИСТРОЯМИ З МЕТОЮ МІНІМІЗАЦІЇ СУМАРНОГО ВІДХИЛЕННЯ МОМЕНТІВ ЗАВЕРШЕННЯ ВІД ДИРЕКТИВНИХ СТРОКІВ

3.1 Постановка задачі

Задано дві множини завдань $I_1 = \{1,2,..,i_1,.,n_1\}$ та $I_2 = \{1,2,..,i_2,.,n_2\}$ та кількість пристроїв m $(m>1)$. Множина завдань I_1 має спільний директивний строк d_1, а множина I_2 – директивний строк d_2. Пристрої працюють паралельно, є взаємозамінними і відрізняються один від одного продуктивністю виконання завдань. Пристрої можна впорядкувати за швидкістю виконання завдання і цей порядок однаковий для всіх завдань. Для кожного пристрою j, $j=\overline{1,m}$ задано коефіцієнт продуктивності h_j такий, що тривалість виконання завдання i на пристрої j дорівнює $h_j p_i$. «Еталонним» будемо називати пристрій з коефіцієнтом продуктивності $h=1$. У цьому випадку величина p_i є тривалістю виконання завдання i на еталонному пристрої. Величина h_j – коефіцієнт продуктивності пристрою j (якщо $h_j>1$, то пристрій j менш продуктивний, ніж еталонний, якщо $h_j<1$ – більш продуктивний).

Передбачається, що всі завдання множин I_1 та I_2 надходять одночасно в нульовий момент часу, процес обслуговування кожного завдання проходить без переривань до завершення обслуговування. Пристрої можуть починати свою роботу в ненульовий момент часу.

Необхідно побудувати розклад, що мінімізує сумарне відхилення моментів закінчення виконання завдань від директивних строків.

Введемо позначення:

Q_j – множина завдань, що запізнюються на пристрої j, $|Q_j| = q_j$, $j=\overline{1,m}$;

$j\lfloor i \rfloor$ – номер завдання, що виконується на пристрої j та є i-тим за порядком в послідовності завдань, що запізнюються на цьому пристрої;

W_j – множина завдань, що на пристрої j виконуються з випередженням, $|W_j| = w_j$, $j = \overline{1,m}$ (завдання, у якого на пристрої j випередження дорівнює 0, будемо відносити до множини W_j);

$j\lceil i \rceil$ – номер завдання, що є i-тим з кінця в упорядкуванні завдань з множини W_j (на пристрої j є i-тим в послідовності завдань множини W_j, якщо рахувати від директивного строку справа наліво по часовій осі).

3.2 Узагальнена схема алгоритму складання розкладу

Алгоритм розв'язання задачі складається з трьох етапів.

ЕТАП I. *Попереднє закріплення пристроїв між підмножинами завдань*

I.1 Розв'язання допоміжної оптимізаційної задачі – отримання підмножин M_1 (пристрої, на які будуть призначатися завдання множини I_1) та M_2 (пристрої, на які будуть призначатися завдання множини I_2).

ЕТАП II *Побудова початкового розкладу*

II.1 Побудова допустимого розкладу S_1: за алгоритмом А1 скласти розклад виконання завдань множини I_1 пристроями множини M_1) ($u(S_1)$ момент початку розкладу S_1).

II.2 Побудова допустимого розкладу S_2: за алгоритмом А1 скласти розклад виконання завдань множини I_2 пристроями множини M_2) ($u(S_2)$ момент початку розкладу S_2).

ЕТАП III. *Покращення допустимого розкладу.*

3.3 Задача закріплення пристроїв між підмножинами завдань

Ця оптимізаційна задача є допоміжною, вона розв'язується на ЕТАПІ 1 алгоритму і результати її розв'язання є вхідними для наступного етапу.

Вхідними даними цієї задачі є:

$$I,\ n \tag{3.1}$$

$$I_1,\ n_1,\ d_1,\ P_1 = \sum_{i \in I_1} p_i \tag{3.2}$$

$$I_2,\ n_2,\ d_2,\ P_2 = \sum_{i \in I_2} p_i \tag{3.3}$$

$$M,\ m,\ h_1, h_2, ..., h_m \tag{3.4}$$

Необхідно закріпити пристрої за підмножинами завдань I_1 та I_2, тобто розділити множину пристроїв M на дві підмножини M_1 та M_2, де M_1 – множина пристроїв, на які будуть призначатися завдання множини I_1, M_2 – множина пристроїв, на які будуть призначатися завдання множини I_2.

Можливі два підходи до розв'язання цієї допоміжної задачі:

– розподіл, який базується тільки на кількісних характеристиках множин (умовно назвемо його *кількісним розподілом*);
– розподіл, що враховує значення тривалостей та коефіцієнтів продуктивності (умовно назвемо його *об'ємним розподілом*).

3.3.1 Кількісний розподіл

Знайдемо такі групи пристроїв m_1 та m_2 ($m_1 + m_2 = m$), щоб середня кількість завдань, що виконується різними групами пристроїв, була максимально близькою:

$$\frac{n_1}{m_1} \approx \frac{n_2}{m_2} \tag{3.4}$$

$$\frac{n_1}{m_1} = \frac{n_2}{m - m_1} \tag{3.5}$$

$$(m - m_1) \cdot n_1 - m_1 n_2 = 0 \tag{3.6}$$

$$n_1 m - m_1 n = 0 \tag{3.7}$$

$$m_1 = \frac{m \cdot n_1}{n} \tag{3.8}$$

Приклад 3.1.

Нехай задано $m=10$ пристроїв, та множини робіт $n_1=78$, $n_2=12$. Директивні строки d_1 та d_2.

Розділимо пристрої на дві підмножини наступним чином:

$$\frac{n_1}{n_2}\approx\frac{m_1}{m_2}$$

$$m_1\approx m\cdot\frac{n_1}{n} \qquad m_1\approx 10\cdot\frac{78}{100}=]7{,}8[=8$$

$$m_2\approx m\cdot\frac{n_2}{n} \qquad m_2\approx 10\cdot\frac{12}{100}=]1{,}2[=2$$

Нехай $\frac{n_1}{m_1}$ (та $\frac{n_2}{m_2}$) є цілими числами, тоді наша задача розбивається на дві незалежні підзадачі:

– у першій підзадачі необхідно побудувати розклад розподілу $n_1=78$ завдань за $m_1=8$ пристроями, що мінімізує відхилення виконання завдань від директивного строку d_1;
– у другій підзадачі необхідно побудувати розклад розподілу $n_2=12$ завдань за $m_2=2$ пристроями, що мінімізує відхилення виконання завдань від директивного строку d_2.

Таблиця 3.1 – Коефіцієнти цільової функції

Множина пристроїв	Номер пристрою	Коефіцієнти ЦФ	
		Завдання, що виконуються з випередженням (W)	Завдання, що запізнюються (Q)
m_1	1	$w_1-1, w_1-2,...,2,1,0$	$q_1, q_1-1,...,2,1$
	2	$w_2-1, w_2-2,...,2,1,0$	$q_2, q_2-1,...,2,1$
	…	…	…
	m_1	$w_{m_1}-1, w_{m_1}-2,...,2,1,0$	$q_{m_1}, q_{m_1}-1,...,2,1$
m_2	1	$w_1-1, w_1-2,...,2,1,0$	$q_1, q_1-1,...,2,1$
	…	…	…
	m_2	$w_{m_2}-1, w_{m_2}-2,...,2,1,0$	$q_{m_2}, q_{m_2}-1,...,2,1$

Кількісний розподіл найкраще підходить до задачі складання розкладу для ідентичних пристроїв.

3.3.2 Об’ємний розподіл

Введемо додаткову характеристику машин: $s_j = \frac{1}{h_j}$, $j = 1,\dots, m$ – швидкість роботи машини j, тоді:

$\sum_{j \in M_1} s_j$ – сумарна швидкість машин множини M_1;

$\sum_{j \in M_2} s_j$ – сумарна швидкість машин множини M_2.

Представляється логічним розбивати множину M на дві підмножини за наступними правилом: сумарна швидкість відповідної підмножини машин повинна бути прямо пропорційна сумарному обсягу робіт або\та обернено пропорційна до відповідного директивного строку.

Формалізуємо ці твердження. Так «сумарна швидкість відповідної підмножини машин є прямо пропорційною сумарному обсягу робіт» означає, що :

$$\frac{P_1}{\sum_{j \in M_1} s_j} = \frac{P_2}{\sum_{j \in M_2} s_j} \tag{3.9}$$

(з урахуванням того, що P_1 та P_2 є відомими цілими числами, а s_j, $j = \overline{1,m}$ є заданими дійсними числами, тому рівність в (3.9) може бути не досяжною, тому постає задача таким чином розбити множину M на дві підмножини, щоб різниця між правою та лівою частинами рівняння в (3.9) була мінімальною.

Таким чином, можуть бути сформульовані наступні критерії розподілу:

$$\left| \frac{\sum_{j \in M_1} s_j}{\sum_{j \in M_2} s_j} - \frac{P_1}{P_2} \right| \to \min \tag{3.10}$$

$$\left| \frac{\sum_{j \in M_1} s_j}{\sum_{j \in M_2} s_j} - \frac{d_2}{d_1} \right| \to \min \tag{3.11}$$

$$\left| \frac{\sum_{j \in M_1} s_j}{\sum_{j \in M_2} s_j} - \frac{P_1 d_2}{P_2 d_1} \right| \to \min \tag{3.12}$$

Математична модель допоміжної оптимізаційної задачі

Введемо наступні змінні:

$$x_j = \begin{cases} 1, \text{ якщо } j \in M_1 \\ 0, \text{ якщо } j \in M_2 \end{cases} \quad j = \overline{1,m} \tag{3.13}$$

З урахуванням змістовного значення змінних сумарна швидкість машин множини M_1 становить $\sum_{j \in M_1} s_j = \sum_{j=1}^{m} s_j x_j$, множини M_2 становить $\sum_{j \in M_2} s_j = \sum_{j=1}^{m} s_j (1 - x_j)$.

Тоді критерії оптимізації будуть мати вигляд:

$$\left| \frac{\sum_{j=1}^{m} s_j x_j}{\sum_{j=1}^{m} s_j (1 - x_j)} - \frac{P_1}{P_2} \right| \to \min \tag{3.14}$$

$$\left| \frac{\sum_{j=1}^{m} s_j x_j}{\sum_{j=1}^{m} s_j (1 - x_j)} - \frac{d_1}{d_2} \right| \to \min \tag{3.15}$$

$$\left| \frac{\sum_{j=1}^{m} s_j x_j}{\sum_{j=1}^{m} s_j (1 - x_j)} - \frac{P_1 d_1}{P_2 d_2} \right| \to \min \tag{3.16}$$

Обмежень, окрім $x_j \in \{0; 1\}$, $j = \overline{1,m}$, немає.

Виконаємо спрощення виразів цільових функцій:

$$\left| \frac{\sum_{j=1}^{m} s_j x_j}{\sum_{j=1}^{m} s_j (1 - x_j)} - \frac{P_1}{P_2} \right| = \left| P_2 \sum_{j=1}^{m} s_j x_j - P_1 \sum_{j=1}^{m} s_j (1 - x_j) \right| = \tag{3.17}$$

$$= \left| P_2 \sum_{j=1}^{m} s_j x_j - P_1 \sum_{j=1}^{m} s_j + P_1 \sum_{j=1}^{m} s_j x_j \right| = \left| (P_2 + P_1) \sum_{j=1}^{m} s_j x_j - P_1 \sum_{j=1}^{m} s_j \right|$$

$$\left|\frac{\sum_{j=1}^{m} s_j x_j}{\sum_{j=1}^{m} s_j (1-x_j)} - \frac{d_2}{d_1}\right| = \left|d_1 \sum_{j=1}^{m} s_j x_j - d_2 \sum_{j=1}^{m} s_j (1-x_j)\right| =$$

$$= \left|d_1 \sum_{j=1}^{m} s_j x_j - d_2 \sum_{j=1}^{m} s_j + d_2 \sum_{j=1}^{m} s_j x_j\right| = \left|(d_1 + d_2) \sum_{j=1}^{m} s_j x_j - d_2 \sum_{j=1}^{m} s_j\right| \quad (3.18)$$

$$\left|\frac{\sum_{j=1}^{m} s_j x_j}{\sum_{j=1}^{m} s_j (1-x_j)} - \frac{P_1 d_2}{P_2 d_1}\right| = \left|P_2 d_1 \sum_{j=1}^{m} s_j x_j - P_1 d_2 \sum_{j=1}^{m} s_j (1-x_j)\right| =$$

$$= \left|P_2 d_1 \sum_{j=1}^{m} s_j x_j - P_1 d_2 \sum_{j=1}^{m} s_j + P_1 d_2 \sum_{j=1}^{m} s_j x_j\right| = \left|(P_2 d_1 + P_1 d_2) \sum_{j=1}^{m} s_j x_j - P_1 d_2 \sum_{j=1}^{m} s_j\right| \quad (3.19)$$

3.4 Побудова допустимого розкладу

Введемо позначення:

Q_j – множина завдань, що запізнюються на пристрої j, $|Q_j| = q_j$, $j = \overline{1,m}$;

$j\lfloor i \rfloor$ – номер завдання, що виконується на пристрої j та є i-тим за порядком в послідовності завдань, що запізнюються на цьому пристрої;

W_j – множина завдань, що на пристрої j виконуються з випередженням, $|W_j| = w_j$ $j = \overline{1,m}$ (завдання, у якого на пристрої j випередження дорівнює 0, будемо відносити до множини W_j);

$j\lceil i \rceil$ – номер завдання, що є i-тим з кінця в упорядкуванні завдань з множини W_j (на пристрої j є i-тим в послідовності завдань множини W_j, якщо рахувати від директивного строку справа наліво по часовій осі).

Рисунок 3.1 ілюструє ці позначення.

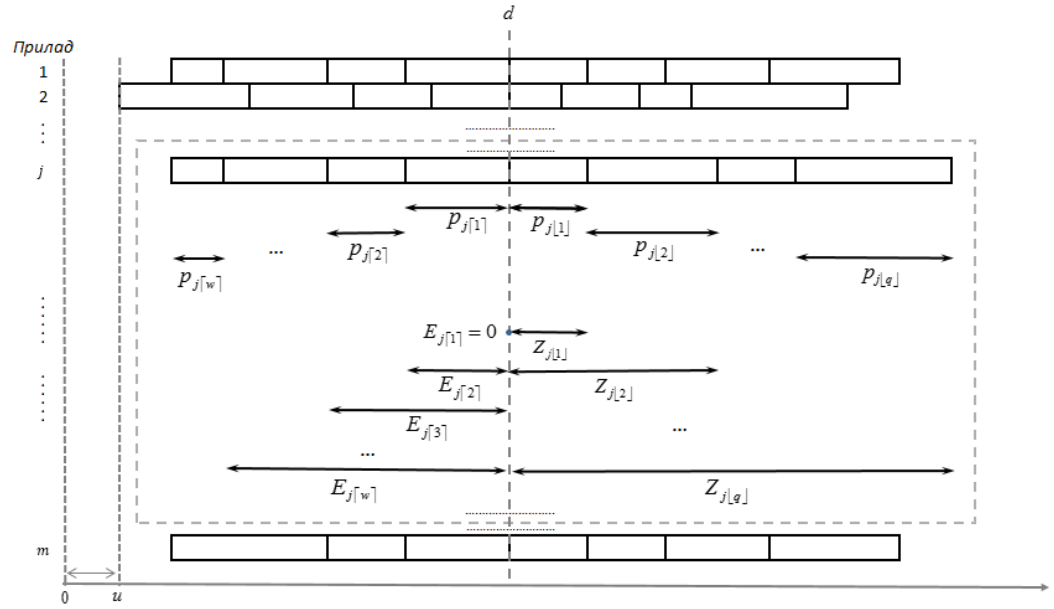

Рисунок 3.1 – Позначення в розкладах з декількома пристроями

Визначимо для цього випадку сумарне відхилення моментів закінчення від директивного строку:

Сумарне випередження:

$$E=\sum_{j=1}^{m}\sum_{i=1}^{w_j}E_{j\lceil i\rceil}=\sum_{j=1}^{m}h_j\sum_{k=1}^{w_j-1}\sum_{i=1}^{k}p_{\lceil k\rceil}=\sum_{j=1}^{m}h_j\sum_{i=1}^{w_j-1}(w_j-i)p_{j\lceil i\rceil} \tag{3.19}$$

Сумарне запізнення:

$$Z=\sum_{j=1}^{m}\sum_{i=1}^{q_j}Z_{j\lfloor i\rfloor}=\sum_{j=1}^{m}h_j\sum_{k=1}^{q_j}\sum_{i=1}^{k}p_{j\lfloor k\rfloor}=\sum_{j=1}^{m}h_j\sum_{i=1}^{q_j}(q_j-i+1)p_{j\lfloor i\rfloor}. \tag{3.20}$$

Сумарне відхилення:

$$E+Z=\sum_{j=1}^{m}h_j\sum_{i=1}^{w_j}(w_j-i)p_{j\lceil i\rceil}+\sum_{j=1}^{m}h_j\sum_{i=1}^{q_j}(q_j-i+1)p_{j\lfloor i\rfloor} \tag{3.21}$$

Отже, $E+Z$ являє собою суму n доданків ($n=\sum_{j=1}^{m}\left(w_j+q_j\right)$), кожний з яких є добутком коефіцієнту продуктивності j-го пристрою, цілого числа (що відповідає позиції завдання у розкладі відповідного пристрою) і тривалості завдання. Назвемо *складеним коефіцієнтом* добуток коефіцієнта продуктивності пристрою на ціле

число. Кожна з n тривалостей бере участь у сумуванні один раз, а складені коефіцієнти мають значення, що показані в таблиці 3.1.

Таблиця 3.2 – Значення складених коефіцієнтів

Номер пристрою	Складені коефіцієнти	
	Завдання, що виконуються з випередженням (W)	Завдання, що запізнюються (Q)
1	$h_1(w_1-1), h_1(w_1-2), \ldots, 2h_1, 1h_1, 0$	$h_1q_1, h_1(q_1-1), \ldots, 2h_1, 1h_1$
2	$h_2(w_2-1), h_2(w_2-2), \ldots, 2h_2, 1h_2, 0$	$h_2q_2, h_2(q_2-1), \ldots, 2h_2, 1h_2$
…	…	…
m	$h_m(w_m-1), h_m(w_m-2), \ldots, 2h_m, 1h_m, 0$	$h_mq_m, h_m(q_m-1), \ldots, 2h_m, 1h_m$

Для мінімізації сумарного відхилення необхідно розподіляти завдання між пристроями за правилом «найдовшому завданню повинен відповідати найменший складений коефіцієнт ЦФ».

Побудований за таким правилом розклад може виявитись неодопустимим (момент початку виконання розкладу на одному чи декількох пристроях – від'ємний). Тому для уникнення цієї проблеми, необхідно модифікувати алгоритм побудови розкладу.

Модифікований алгоритм розподілу завдань між пропорційними пристроями (алгоритм А1)

КРОК 1. Впорядкувати завдання за незростанням тривалостей ($p_1 \geq p_2 \geq \ldots \geq p_n$).

КРОК 2. Впорядкувати значення складених коефіцієнтів ЦФ за неспаданням, що належать множині W та множині Q.

КРОК 3. ЦИКЛ по завданням

Знайти для завдання позицію у розкладі, якій відповідає *найменший* вільний складений коефіцієнт

ЯКЩО складений коефіцієнт належить множині W та для завдання виконується умова $h_j p_{j\lceil i \rceil} \leq d - h_j \sum_{i \in W_j} p_{j\lceil i \rceil}$:

Завдання i призначити на поточну позицію у розкладі.

ІНАКШЕ

Завдання i призначити до множини Q з найменшим вільним складеним коефіцієнтом

3.5 Перестановочний алгоритм покращення допустимого розкладу

Для покращення розкладу пропонується здійснювати перестановку завдань між двома множинами пристроїв: M_1 – множиною пристроїв, на які призначено завдання множини I_1 (з директивним строком d_1) та M_2 – множиною пристроїв, на які призначено завдання множини I_2 (з директивним строком d_2).

Нехай множини I_1 та I_2 нумеровані таким чином, що $d_2 > d_1$.

Пропонуються наступні операції перестановки:

Перестановка типу 1. *Обмін завдань між пристроями множин* M_1 та M_2

Деяка підмножина завдань з множини I_2 може призначатися на виконання на пристрої множини M_1.

Перестановка типу 2. *Обмін завдань між пристроями всередині множини* M_2

Деяка підмножина завдань з пристрою l може призначатися на виконання до пристрою v, $l \neq v, l, v \in M_2$.

Введемо наступні позначення:

$Q_j^{M_k}$ – множина завдань, що запізнюються на пристрої $j \in M_k$, $k = \{1, 2\}$, $\left|Q_j^{M_k}\right| = q_j^{M_k}$, $j = \overline{1, m}$;

$W_j^{M_k}$ – множина завдань, що на пристрої $j \in M_k$, $k = \{1, 2\}$ виконуються з випередженням, $\left|W_j^{M_k}\right| = w_j^{M_k}$, $j = \overline{1, m}$;

T_j^Q – сумарна тривалість виконання пристроєм j завдань що запізнюються;

T_j^W – сумарна тривалість виконання пристроєм j завдань що виконуються з випередженням;

$\Delta_j^{M_1} = d_1 + T_j^Q$ – момент закінчення виконання завдань на пристрої $j \in M_1$;

$\delta_j^{M_1} = d_2 - \Delta_j^{M_1}$ – часовий проміжок між моментом закінчення виконання завдань на пристрої j з множини M_1 та директивним строком d_2.

$\gamma_j^{M_2} = d_2 - T_j^W$ – часовий проміжок до початку виконання завдань на пристрої j з множини M_2.

Для спрощення запису, введемо наступне позначення коефіцієнтів цільової функції:

$z_i^{new} = \begin{cases} (w_j^{M_k} - i) \\ (q_j^{M_k} - i + 1) \end{cases}$ – коефіцієнт ЦФ для завдання, що переставляється на позицію i відносно директивного строку та буде виконуватись на пристрої $j \in M_k$, $k = \{1, 2\}$;

$z_i^{old} = \begin{cases} (w_j^{M_k} - i) \\ (q_j^{M_k} - i + 1) \end{cases}$ – попередній коефіцієнт ЦФ для завдання, що знаходилося на позиції i відносно відносно директивного строку та виконувалось на пристрої $j \in M_k$, $k = \{1, 2\}$.

Для ілюстрації цих перестановок розглянемо деякий допустимий розв'язок (рис.1).

Примітка: для наглядності на рисунках вказана «реальна» тривалість виконання завдання з урахуванням коефіцієнтів продуктивності пристроїв (тривалість виконання завдання на пристрої позначено у дужках).

Приклад 3.2.

Задано $n_1 = 10$ та $n_2 = 14$ завдань і $m = 8$ пристроїв. Директивний строк для множини завдань I_1 становить $d_1 = 50$, а для множиин завдань I_2 – $d_2 = 60$.

Коефіцієнти продуктивності пристроїв мають значення: $h_1 = 1,\ h_2 = 1{,}2,\ h_3 = 1{,}4,\ h_4 = 1{,}8,\ h_5 = 2{,}1,\ h_6 = 3,\ h_7 = 3{,}5,\ h_8 = 4$.

Тривалість виконання завдань задано вектором: $p^{I_1} = \{20,15,14,12,10,9,7,6,4,3\}$, $p^{I_2} = \{24,23,20,14,13,12,9,8,7,6,5,4,3,2\}$.

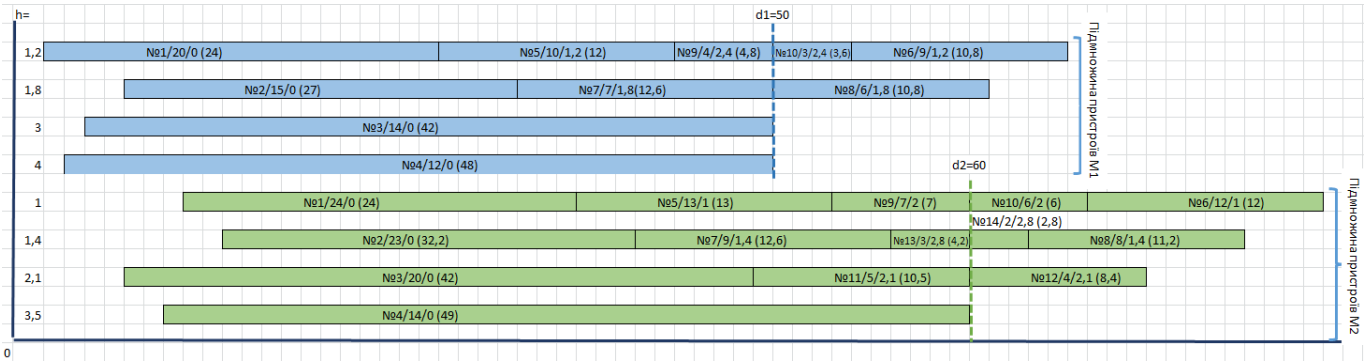

Рисунок 3.2 – Розклад отриманий за алгоритмом А1

Сумарне випередження:

$E = 37.4 + 46.6 = 84$.

Сумарне запізнення:

$Z = 39.6 + 50.4 = 90$.

Сумарне відхилення: $L = E + Z = 174$.

Для покращення допустимого розкладу виконаємо наступні кроки:

1. Для кожного пристрою M_1 розраховуємо величину $\Delta_j^{M_1}$.

Для ілюстрації позначень, для прикладу візьмемо частину розкладу для пристроїв з множини M_1 (рис.3.3).

Приклад 3.3

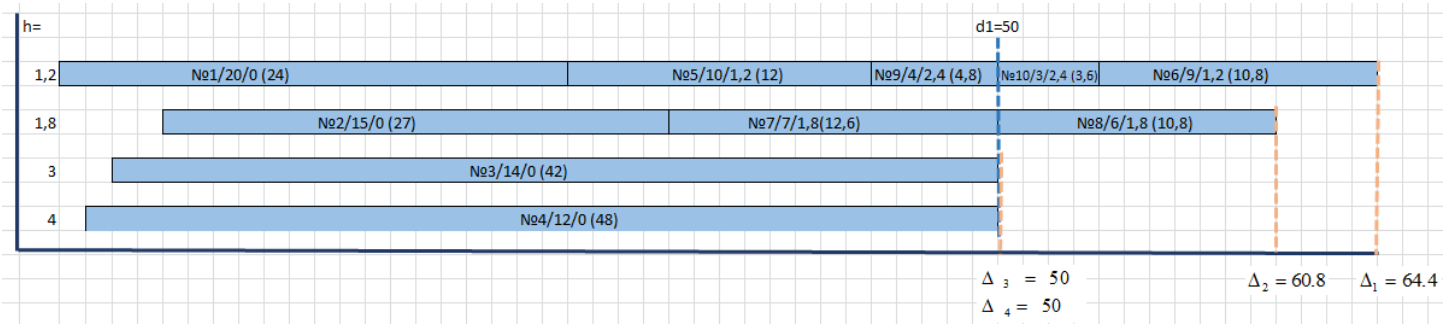

Рисунок 3.3 – Розрахунок величини $\Delta_j^{M_1}$

$\Delta_1^{M_1} = 50 + 3.6 + 10.8 = 64.4$

$\Delta_2^{M_1} = 50 + 10.8 = 60.8$

$\Delta_3^{M_1} = 50$

$\Delta_4^{M_1} = 50$

2. Для кожного пристрою з M_1 розраховуємо величину $\delta_j^{M_1} = d_2 - \Delta_j^{M_1}$.

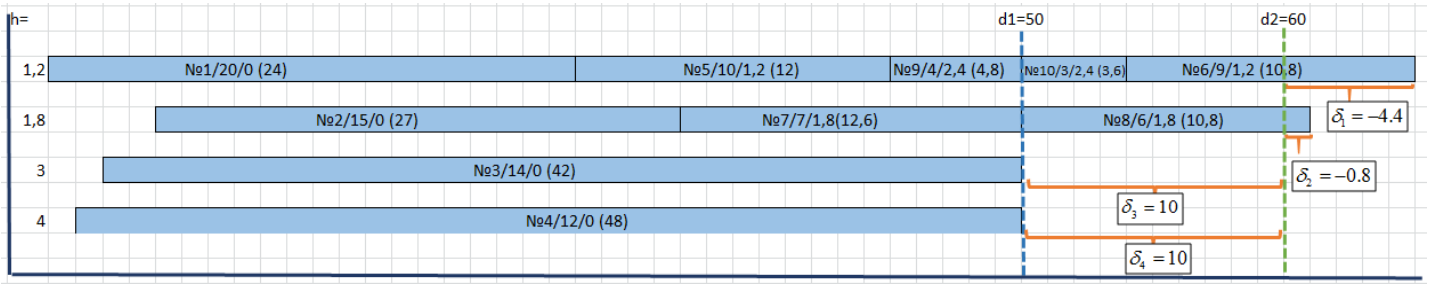

Рисунок 3.4 – Розрахунок величини $\delta_j^{M_1}$

Визначимо умови перестановки завдань з множини I_2 на виконання до пристрою множини M_1.

Умова 1

$p_i^{I_2} \cdot h_j^{M_1} \le \delta_j$

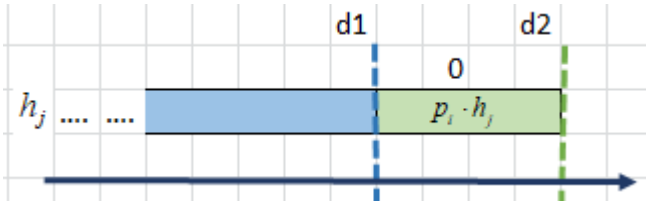

Рисунок 3.5 – Приклад перестановки завдання згідно умови 1.

Умова 2

$\delta_j > 0, (p_i^{I_2} \cdot h_j^{M_1} - \delta_j) \cdot z_i^{new} < p_i^{I_2} \cdot h_j^{M_2} \cdot z_i^{old}$

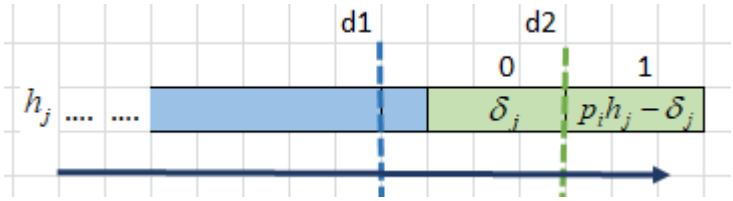

Рисунок 3.6 – Приклад перестановки завдання згідно умови 2.

Умова 3

$\delta_j \ge 0, \left|\delta_j\right| + p_i^{I_2} \cdot h_j^{M_1} z_i^{new} < p_i^{I_2} \cdot h_j^{M_2} \cdot z_i^{old}$

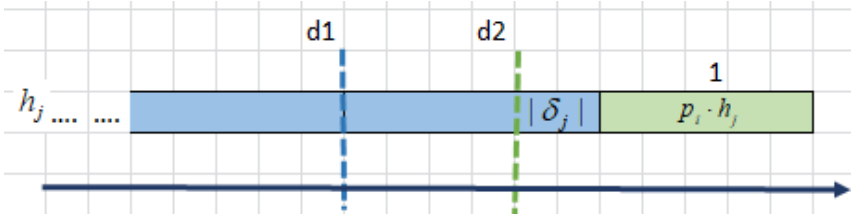

Рисунок 3.7 – Приклад перестановки завдання згідно умови 3.

Аналогічно визначимо умови перестановки завдань з множини I_1 на виконання до пристрою множини M_2.

Умова 1

$\gamma_j = d_1,\ h_j p_i \le \gamma_j$

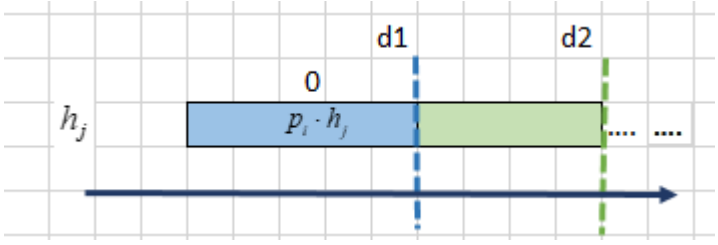

Рисунок 3.8 – Приклад перестановки завдання згідно умови 1.

Умова 2

$\gamma_j > d_1, p_i^{I_1} \cdot h_j^{M_2} z_i^{new} < p_i^{I_1} \cdot h_j^{M_1} \cdot z_i^{old}$

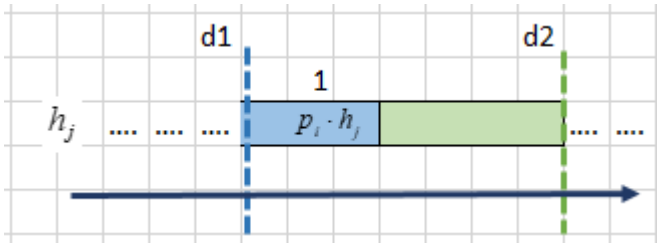

Рисунок 3.9 – Приклад перестановки завдання згідно умови 2.

Умова 3

$\gamma_j < d_1,\ p_i^{I_1} \cdot h_j^{M_2} \cdot z_i^{new} + (d_1 - \gamma_j) < p_i^{I_1} \cdot h_j^{M_1} \cdot z_i^{old}$

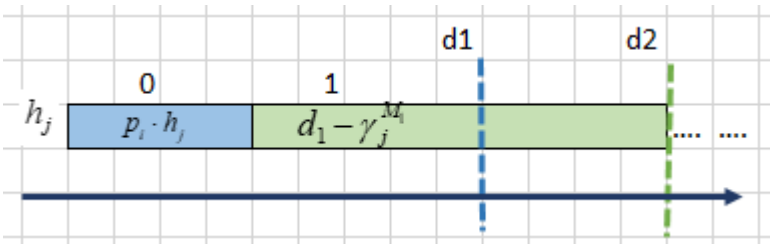

Рисунок 3.10 – Приклад перестановки завдання згідно умови 3.

Після того, як відбулися всі можливі перестановки робіт між множинами пристроїв M_1 та M_2, перевіряємо чи можна покращити значення ЦФ шляхом перестановки робіт всередині цих множин використовуючи «місця що звільнилися».

У результаті проведення перестановок для покращення допустимого розкладу отримаємо альтернативний розклад, що показаний на рис.3.11.

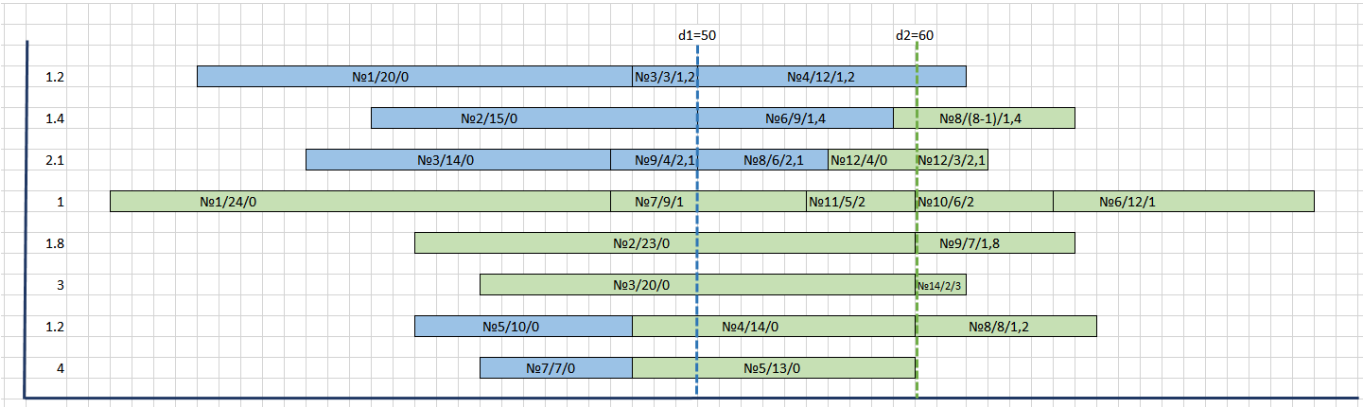

Рисунок 3.11 – Альтернативний розклад після проведення перестановок

Сумарне відхилення: $L = E + Z = 154{,}7$.

Значення цільової функції покращилось.

Алгоритм перестановок завдань для покращення допустимого розкладу (алгоритм А2)

КРОК 1. Створити масив завдань I_t, що будуть переставлятися на виконання до множини пристроїв.

КРОК 2. Створити масив пристроїв M_k, на які будуть призначатися завдання з множини I_t.

КРОК 3. Впорядкувати завдання з множини I_t за незростанням добутку тривалості завдання на складений коефіцієнт цільової функції.

КРОК 2. ЦИКЛ по завданням з I_t

2.1. Розрахувати значення ЦФ для пристрою, на якому виконується завдання.

2.2. Розрахувати значення ЦФ для пристрою без завдання.

КРОК 3. ЦИКЛ по пристроям з множини M_k

3.1. Розрахувати значення ЦФ для пристрою з урахуванням нового завдання.

3.2. Розрахувати значення ЦФ для пристрою без урахування нового завдання.

ЯКЩО перестановка завдання на новий пристрій покращує значення ЦФ

ЯКЩО знайдена перестановка краща за попередню

Зберегти поточну перестановку та позначити її як найкращу.

ЯКЩО краща перестановка була знайдена

Видалити завдання з попередньої позиції.

Призначити завдання на виконання до пристрою з M_k

Висновки до розділу

У розділі представлено узагальнену схему алгоритму складання розкладу для задачі мінімізації сумарного відхилення моментів завершення від директивних термінів для паралельних пристроїв з різною продуктивністю, що складається з трьох етапів. На першому етапі пропонується розв’язати допоміжну оптимізаційну задачу закріплення пристроїв між підмножинами завдань. Для пристроїв з різною продуктивністю найкращим є об’ємний розподіл, що враховує значення тривалостей робіт та коефіцієнтів продуктивності пристроїв. Другий етап схеми складання розкладу передбачає побудову допустимого розкладу. Для допустимого розкладу запропонований модифікований алгоритм розподілу завдань між пропорційними пристроями. На останньому етапі пропонується здійснювати перестановку завдань для покращення допустимого розкладу.

4 ОПИС ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

4.1 Засоби розробки

Для розробки програмного забезпечення було використано мову програмування C++, крос-платформенний інструментарій для розробки Qt та розподілену систему контролю версій GIT.

C++ – надзвичайно потужна мова, що містить засоби створення ефективних програм практично будь-якого призначення, від низькорівневих утиліт і драйверів до складних програмних комплексів самого різного призначення.

Підтримуються різні стилі і технології програмування, включаючи традиційне директивне програмування, ООП. Є можливість роботи на низькому рівні з пам'яттю, адресами, портами. Існує також можливість створення узагальнених алгоритмів для різних типів даних, їх спеціалізація і обчислення на етапі компіляції, використовуючи шаблони.

Доступні компілятори для великої кількості платформ, на мові C ++ розробляють програми для найрізноманітніших платформ і систем.

Мова спроектована так, щоб дати програмісту максимальний контроль над усіма аспектами структури і порядку виконання програми. Жодна з мовних можливостей, що призводить до додаткових накладних витрат, не є обов'язковою для використання. Мова дозволяє забезпечити максимальну ефективність програми.

Qt – багатоплатформовий фреймворк для розробки програмного забезпечення на мові програмування C ++.

Qt дозволяє запускати написане з його допомогою програмне забезпечення в більшості сучасних операційних систем шляхом простої компіляції програми для кожної системи без зміни вихідного коду. Включає в себе всі основні класи, які можуть знадобитися при розробці прикладного програмного забезпечення, починаючи від елементів графічного інтерфейсу і закінчуючи класами для роботи з мережею, базами даних і XML. Є повністю об'єктно-орієнтованим, розширюваним і підтримує техніку компонентного програмування.

GIT – це одна з найвідоміших систем контролю версій з відкритим вихідним кодом, на яку покладаються мільйони проектів по всьому світу (включаючи як комерційні, так і безкоштовні проекти). Він використовує модель даних, яка допомагає забезпечити криптографічну цілісність всього, що є присутнім в репозиторії. Кожен раз коли файли додаються, генеруються їх контрольні суми; аналогічний процес відбувається при їх витяганні.

4.2 Вимоги до технічного забезпечення

Вимоги до технічних характеристик робочих місць користувачів:

- операційна система: Windows 7 з пакетом оновлення 3+, Vista, 7, 8, 10; Mac OS X 10.6 або більш пізньої версії; Ubuntu 10.04 +, Debian 6 +, OpenSuSE 11.3 +, Fedora Linux 14;
- процесор: Intel Pentium 4 / Athlon 64 або пізнішої версії з підтримкою SSE2;
- вільне місце на HDD: 350 Мб;
- оперативна пам’ять: 512 Мб.

4.3 Архітектура програмного забезпечення

4.3.1 Опис класів програмного забезпечення

У частині графічного матеріалу представлена схема структурна класів програми для побудови розкладу для задачі мінімізації сумарного відхилення моментів завершення від директивних строків.

Схема містить наступні класи:

Machine – клас для роботи з приладами.

Scheduler – клас для складання розкладу.

Distribution – клас для розподілу машин за множинами.

Task – клас для роботи з завданнями.

Logger – клас для виведення данних на головне вікно програми.

Generator – клас для генерування вхідної задачі

Criteria – клас для роботи з критеріями оптимізації розбиття машин по множинам.

MainWindow – клас для роботи з головним вікном програми.

Таблиця 4.1 – Опис атрибутів класу Machine

Атрибут	Тип значення	Опис
m_id	int	Ідентифікатор машини
m_coef	float	Коефіцієнт продуктивності машини
m_revert_coef	float	Обернений коефіцієнт продуктивності машини
m_deadline	float	Основний директивний строк
m_tasks_before_deadline	std::set<QSharedPointer<Task>>	Набір завдань, що виконуватимуться до основного директивного строку
m_tasks_after_deadline	std::set<QSharedPointer<Task>>	Набір завдань, що виконуються після основного директивного строку
m_aditional_deadline	float	Додатковий директивний строк
m_time_left	double	Залишок часу від додаткового директивного строку до моменту виконання усіх завдань для основного директивного строку
m_tasks_aditional_before	std::set<QSharedPointer<Task>>	Набір завдань, що будуть виконуватися до додаткого директивного строку
m_tasks_adtional_after	std::set<QSharedPointer<Task>>	Набір завдань, що виконуватимуться після додаткового директивного строку

Таблиця 4.2 – Опис атрибутів класу Logger

Атрибут	Тип значення	Опис
m_console	QTextEdit	Текстове поле для виводу усіх повідомлень в програмі
s_logger	Logger	Об’єкт що використовується для виведення усіх повідомлень в програмі

Таблиця 4.3 – Опис атрибутів класу Task

Атрибут	Тип значення	Опис
m_id	int	Унікальний ідентифікатор задачі
m_duration	float	Тривалість задачі

Продовження таблиці 4.3

Атрибут	Тип значення	Опис
m_coef	double	Коефіцієнт продуктивності задачі (залежить від машини)
m_machine	QSharedPointer<Machine>	Машина на якій виконуватиметься задача

Таблиця 4.4 – Опис атрибутів класу Scheduler

Атрибут	Тип значення	Опис
m_deadlines	std::set<QPair<float, std::set< QSharedPointer<Task>>>>	Набір дедлайнів. Один дедлайн складається з двох компонентів – значення дедлайнів, та масивів завдань для конкретного дедлайна
m_criterias	std::vector<Criteria>	Набір критеріїв оптимізації для розподілу машин по множинам
m_machines	std::set< QSharedPointer<Machine>>	Набір машин
m_schedules	QVector< QPair<double, std::set< QSharedPointer<Machine>>>>	Набір розкладів. Кожен елемент складається з двох компонентів – значення цільової функції для розкладу та набору машин
m_coefficient	double	Значення цільової функції для поточного розкладу
m_is_experement_mode	bool	Індикатор режиму роботи програми

Таблиця 4.5 – Опис атрибутів класу Distribution

Атрибут	Тип значення	Опис
m_size	size_t	Розмірність задачі (кількість машин)
m_min	size_t	Мінімальна кількість машин в одній множині
m_max	size_t	Максимальна кількість машин в одній множині
m_curr_step	size_t	Поточна ітерація роботи алгоритму

Таблиця 4.6 – Опис атрибутів класу Generator

Атрибут	Тип значення	Опис
m_device	std::random_device	Об’єкт для роботи з генератором рандомних значень
m_generator	std::mt19937	Генератор рандомних значень
m_document	QSharedPointer<QXlsx::Document>	Документ з вихідними даними
m_is_multiple	bool	Індикатор режиму роботи генератора
m_runs_amount	int	Кількість запусків генератора даних
m_curr_run	int	Поточний номер запуску генератора
m_file_proto	QString	Шаблон назви файлу із вихідними даними
m_generator_type	int	Тип генератора рандомних даних
ui	Ui::Generator*	Об’єкт для роботи з графічними інтерфейсом користувача

Таблиця 4.7 – Опис атрибутів класу MainWindow

Атрибут	Тип значення	Опис
ui	Ui::MainWindow*	Об’єкт для роботи з графічними інтерфейсом користувача
m_generator	Generator*	Об’єкт для роботи з генератором вхідних даних
m_scheduler	Scheduler*	Об’єкт для роботи з об’єктом побудови розкладів
m_schedules	ScheduleArray	Набір розкладів

Таблиця 4.8 – Опис атрибутів класу ScheduleChart

Атрибут	Тип значення	Опис
ui	Ui::ScheduleChart*	Об’єкт для роботи з графічними інтерфейсом користувача
m_is_optimized	bool	Ідентифікатор режиму роботи класу
m_machines	std::set< QSharedPointer<Machine>>	Набір машин розкладу
m_deadlines	QPair<float, float>	Пара дедлайнів
m_deadline_series	QVector<QtCharts::QLineSeries*>	Масив даних для відображення дедлайнів на графіку
m_chart	QtCharts::QChart *	Об’єкт для відображення графіків

Продовження таблиці 4.8

Атрибут	Тип значення	Опис
m_x_axis	QtCharts::QValueAxis *	Вісь x
m_y_axis	QtCharts::QValueAxis *	Вісь y
m_max_x	int	Максимальне значення осі x
m_max_y	int	Максимальне значення осі y

Таблиця 4.9 – Опис атрибутів класу Criteria

Атрибут	Тип значення	Опис
min_value	double	Мінімальне значення крітерію оптимізації
distribution	std::vector<short>	Розподіл машин по множинам
evaluator	std::function<double(double a, double b, double c, double d)>	Функція що розраховує значення критерія для розподілу

4.3.2 Специфікація функцій алгоритмічного забезпечення

В цьому підрозділі у таблицях 4.10–4.18 буде описано специфікацію функцій класів алгоритмічного забезпечення

Таблиця 4.10 – Опис специфікації функцій класу Machine

Метод	Призначення	Повертає результат	Список параметрів	Семантика параметрів
GetID	Отримання ідентифікатора пристрою	int	-	-
GetCoefficient	Отримання коефіцієнту продуктивності пристрою	float	-	-
GetRevertedCoefficient	Отримання оберненого коефіцієнту продуктивності пристрою	float	-	-

Продовження таблиці 4.10

GetDeadline	Отримання директивного терміну	float	-	-
GetTasksBeforeDeadline	Отримання масиву робіт до директивного строку	std::set<QSharedPointer<Task>>	-	-
GetTasksAfterDeadline	Отримання масиву робіт після директивного строку	std::set<QSharedPointer<Task>>	-	-
GetAditionalTasks	Отримати масив завдань для додаткового директивного строку	QVector<TaskPointer>	-	-
GetAditionalTasksBeforeDeadline	Отримання масиву робіт до додаткового директивного строку	std::set<QSharedPointer<Task>>	-	-
GetAditionalTasksAfterDeadline	Отримання масиву робіт після додаткового директивного строку	std::set<QSharedPointer<Task>>	-	-
Reset	Очищення полів класу	-	-	-
SetDeadline	Встановлення директивного терміну для пристроїв	-	deadline	Директивний термін виконання завдань
SetAditionalDeadline	Встановлення значення додаткового дейдлайна		deadline	Директивний термін виконання завдань
GetLowestCoefficientForTask	Отримати найменший коефіцієнт позиції для завдання	float	-	-
AddTask	Поставити завдання на виконання до пристрою	-	TaskPointer task	Завдання

Продовження таблиці 4.10

RemoveTask	Видалити завдання з пристрою	-	TaskPointer task	Завдання
GetTasks	Отримати масив завдань для основного директивного строку	QVector<TaskPair>	-	-
GetAllTasks	Отримати масив усіх завдань на пристрої	QVector<TaskPointer>	-	-
GetFinalCoefficient	Отримати значення цільової функції для машини	double	-	-
GetFinalCoefficientWithTask	Отримати значення цільової функції для машини з додатковим завданям	double	TaskPointer task	Завдання
GetFinalCoefficientWitoutTask	Отримати значення цільової функції для машини без одного із завдань	double	TaskPointer task	Завдання
GetTimeLeft	Залишок часу між додатковим директивним строком та останнім завданням основго директивного строку	double	-	-
GetLeftCoef	Отримати коефіцієнт позиції для завдань що запізнюються	float	-	-
GetRightCoef	Отримати коефіцієнт позиції для завдань що випереджають	float	-	-
GetLeftDuration	Отримати тривалість усіх завдань до основного директивного строку	double	-	-

Продовження таблиці 4.10

GetRightDuration	Отримати тривалість усіх завдань після основного директивного строку	double		
GetArrayDuration	Отримати тривалість усіх завдань в масиві	double	std::set<QSharedPointer<Task>>tasks	Масив завдань
LowestCoefForTaskRegularDeadline	Отримати найменший коефіцієнт для завдання з першим директивним строком	float	TaskPointer task	Завдання
LowestCoefForTaskAditionalDeadline	Отримати найменший коефіцієнт для завдання з другим директивним строком	float	TaskPointer task	Завдання
AddTaskForRegularDeadline	Додати завдання на машину для першого директивного строку	-	TaskPointer task	Завдання
AddTaskForAditionalDeadline	Додати завдання на машину для другого директивного строку	-	TaskPointer task	Завдання

Таблиця 4.11 – Опис специфікації функцій класу Logger

Метод	Призначення	Повертає результат	Список параметрів	Семантика параметрів
CreateLogger	Створити об’єкт для роботи з логом	-	QTextEdit *console	
GetLogger	Отримати об’єкт для роботи з логом	-	-	-
Write	Вивести повідомлення на консоль	-	QString msg	Текст повідомлення
Write	Вивести повідомлення на консоль	-	QVariant msg	Повідомлення (будь який об’єкт)
Info	Вивести інформаційне повідомлення	-	QString msg	Текст повідомлення
Error	Вивести повідомлення про помилку	-	QString msg	Текст повідомлення

Таблиця 4.12 – Опис специфікації функцій класу Scheduler

Метод	Призначення	Повертає результат	Список параметрів	Семантика параметрів
ReadData	Зчитування вхідних даних з файлу	-	QString file_name	Шлях до файлу
RunExperements	Запуск експерементів	-	QString file_proto, int amount	file_proto – шаблон для файлу з даними для експеремента amount – кількість експериментів
Clear	Очистити всі поля класу	-	-	-
BuildSchedule	Побудувати розклад	-	-	-
GetCoefficient	Отримати значення цільової функції	double	-	-
GetDeadlines	Отримати значення директивних строків	QPair<float, float>	-	-
shedules_built	Сигналізує про завершення алгоритму побудови допустимих розкладів та оптимального розкладу	-	const ScheduleArray& schedules, const std::vector<Criteria>& criterias	Масив знайдених розкладів
ReadMachines	Зчитування машин з файлу	-	const QXlsx::Document& doc, int start_row, int start_column	-файл для зчитування даних; - рядок; - колонка.

Продовження таблиці 4.12

ReadDeadline	Зчитування директивних строків з файлу	-	const QXlsx::Document& doc, int start_row, int start_column	-файл для зчитування даних; - рядок; - колонка.
ReadConfiguration	Зчитування розбиття машин за директивними строками з файлу	-	const QXlsx::Document& doc, int start_row, int start_column	файл для зчитування даних; - рядок; - колонка.
WriteMachines	Виведення пристроїв на головну форму програми	-	-	-
WriteDeadlines	Виведення директивних строків на головну форму програми	-	-	-
WriteSchedule	Виведення побудованого розкладу	-	-	-
GenerateConfiguration	Знайти розбитта машин за множинами	-	-	-
CreateSchedule	Побулова допустимого розкладу			
OptimizeSchedule	Знайти оптимальний розклад		Schedule schedule	Допустимий розклад
GetMachineByID	Знайти машину за унікальним ідентифікатором	MachinePointer	int id	Ідентифікатор пристрою
CheckTasks	Перевірити чи існують в розкладах задачі без пристроїв	-	-	-

Таблиця 4.14 – Опис специфікації функцій класу Task

Метод	Призначення	Повертає результат	Список параметрів	Семантика параметрів
GetCopy	Отримати копію завдання	QSharedPointer<Task>	-	-
GetID	Отримати ідентифікатор завдання	int	-	-
GetDuration	Отримати тривалість виконання завдання	float		
GetCoefficient	Отримати складений коефіцієнт цільової функції	double		
GetMachine	Отримати пристрій	QSharedPointer<Machine>	-	-
SetCoefficient	Встановити коефіцієнт	-	float coef	Складений коефіцієнт цільової функції
SetMachine	Встановити пристрій		QSharedPointer<Machine> m	Вказівник на пристрій

Таблиця 4.15 – Опис специфікації функцій класу Generator

Метод	Призначення	Повертає результат	Список параметрів	Семантика параметрів
Generated	Повідомляє про завершення генерування вхідного файлу	-	QString filename	Шлях до файлу
GeneratedWithRuns	Повідомляє про завершення генерування вхідних файлів для експерименту	-	QString filename, int runs	Шлях до файлу, кількість дослідів
ErrorOccurred	Повідомляє про помилку що сталась в процесі генерації вхідних даних		QString msg	Повідомлення

Продовження таблиці 4.15

RunGenerator	Запуск генератора задач	-	-	-
GenerateMachines	Генерує масив машин та зберігає його у файл	-	-	-
GenerateDeadlines	Генерує масив директивних строків та зберігає його у файл	-	-	-
GenerateTasks	Генерує масив завдань та зберігає його у файл	-	int column, int amount, float start, float end	Початкова колонка у таблиці, кількість завдань, найменше значення тривалості завдання та найбільше
GetDefaultFormat	Повертає формат оформлення даних для таблиці	QXlsx::Format	-	-
SaveFile	Збереження файлу	-	-	-
GetFileProto	Відображає форму вибору назви файлу для користувача	-	-	-
Error	Відображає повідомлення про помилку		QString msg	Повідомлення
GetIntFromUI	Отримати значення int з елемента графічного інтерфейса користувача	int	QLineEdit *text, bool* is_ok	Вказівник на графічний елемент користувача
GetFloatFromUI	Отримати значення float з елемента графічного інтерфейса користувача	int	QLineEdit *text, bool* is_ok	Вказівник на графічний елемент користувача
GetRandomInt	Отримати випадкове значення int	int	int from, int to	-початок проміжку; -кінець проміжку.

GetRandomFloat	Отримати випадкове значення float	float	int from, int to	-початок проміжку; -кінець проміжку.

Таблиця 4.16 – Опис специфікації функцій класу MainWindow

Метод	Призначення	Повертає результат	Список параметрів	Семантика параметрів
SetScheduleButtonsState	Ініціалізує графічний інтерфейс користувача перед побудовою нового розкладу	-	bool state	Значення стану елементів графічного інтерфейсу
BuildTable	Будує шапку для таблиці для допустимого розкладу	-	-	-
BuildOptimizedTable	Будує шапку для таблиці покращеного розкладу	-	-	-
BuildCellData	Заповнює чарунку таблі даними	QTextEdit	QString data	Дані для відображення
ShowSchedule	Відкриває форму що візуалізує обраний графік	-	int schedule	Номер розкладу
OpenFile	Відкриває діалогове вікно вибору вхідного файлу	-	-	-
OpenFile	Відкриває обраний користувачем файл із вхідними даними	-	QString filename	Шлях до файлу
RunScheduler	Запускає алгоритм пошуку оптимального розкладу	-	-	-
SchedulesBuilt	Викликається алгоритмом побудови розкладу для подальшого його відображення		const ScheduleArray& schedules, const std::vector<Criteria>& criterias	Масив усіх знайдених рокладів

Продовження таблиці 4.16

ShowGenerator	Відкриває форму для генерування вхідних даних	-	-	-
ShowGeneratorWithRuns	Відкриває форму для генерування вхідних даних для експериментів	-	-	-
GenerationError	Відображає помилки що трапились під час генерації		QString msg	Повідомлення про помилку
GenerationEnded	Сигналізує про завершення генерації вхідних даних та візуалізує результат генерації		QString filename	Шлях до згенерованого файлу
GenerationWithRunsEnded	Сигналізує про завершення генерації експериментів та візуалізує результат генерації		QString filename, int runs	Шаблон вхідних файлів для експериментів, кількість експериментів
ShowFirstSchedule	Відкриває форму з візуалізацією розкладу побудованого за першим критерієм	-	-	-
ShowSecondSchedule	Відкриває форму з візуалізацією розкладу побудованого за другим критерієм	-	-	-
ShowThirdSchedule	Відкриває форму з візуалізацією розкладу побудованого за третім критерієм	-	-	-
ShowOptimizedSchedlule	Відкриває форму з візуалізацією розкладу побудованого за покращеним алгоритмом	-	-	-

Таблиця 4.17 – Опис специфікації функцій класу ScheduleChart

Метод	Призначення	Повертає результат	Список параметрів	Семантика параметрів
DrawDeadline	Побудувати лінії директивних строків	int	int start, float deadline	Порядковий номер першої машини для директивного строку, значення директивного строку
BuildAxes	Побудувати осі графіку	-	-	-
BuildData	Побудувати роботи та машини	-	-	-
AddData	Побудувати графік однієї роботи на машині	float	TaskPointer task, MachinePointer machine, float start_time, short direction, int y	Робота для відображення, машина для роботи, час початку виконання, коефіцієнт виконання, порядковий номер машини
AddAditionalData	Побудувати графік однієї роботи на машині для додаткового директивного строку	float	TaskPointer task, MachinePointer machine, float start_time, short direction, int y	Робота для відображення, машина для роботи, час початку виконання, коефіцієнт виконання, порядковий номер машини

Таблиця 4.18 – Опис специфікації функцій класу Criteria

Метод	Призначення	Повертає результат	Список параметрів	Семантика параметрів
Check	Знайти значення критерія оптимізації для розподілу робіт	-	double a, double b, double c, double d, std::vector< short> dist	Параметри критерію оптимізації, розподіл машин по множинам

4.4 Опис роботи з програмним забезпеченням

Після запуску програми відкриється головне вікно програми, що показано на рис.4.1.

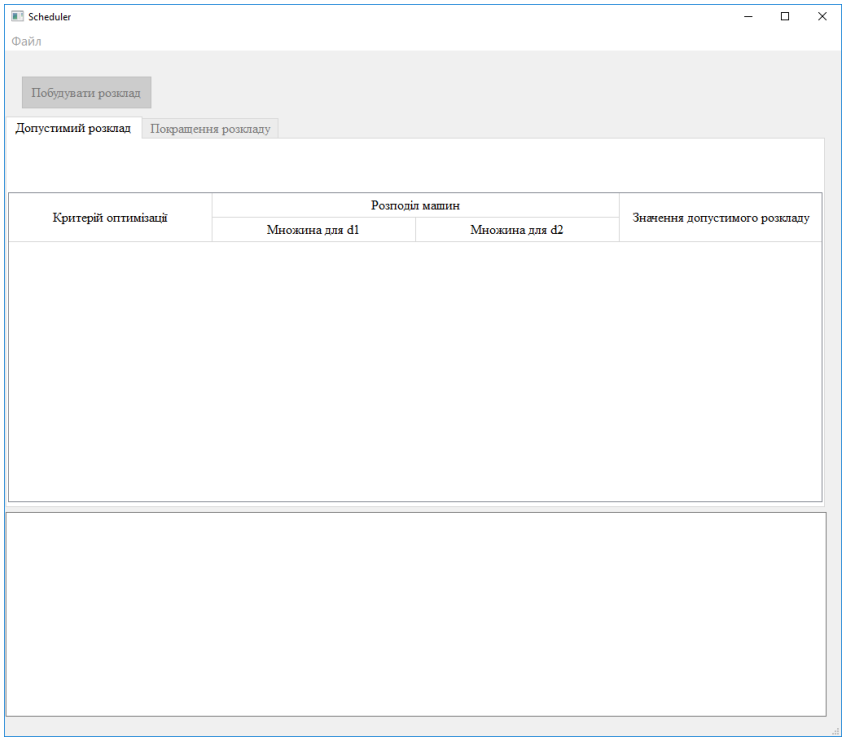

Рисунок 4.1 – Головне вікно роботи з програмою

4.4.1 Зчитування даних з файлу

Для того, щоб зчитати вхідні дані з зовнішнього файлу, у меню програми «Файл», що знаходиться у верхньому лівому куті, необхідно натиснути «Відкрити»

(рис.4.2). Також для зчитування даних з файлу можна скористатися комбінацією клавіш Ctrl+O.

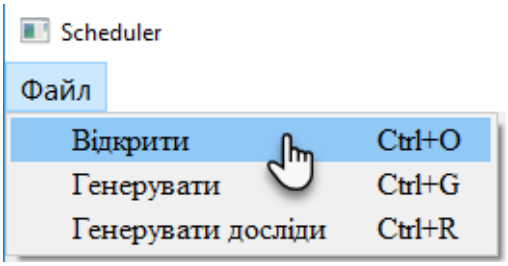

Рисунок 4.2 – Зчитування зовнішнього файлу з вхідними даними

Після цього буде запропоновано вказати шлях до файлу формату xls з вхідними даними.

Після зчитування з файлу, вхідні дані буде виведено у консолі програми (рис.4.3).

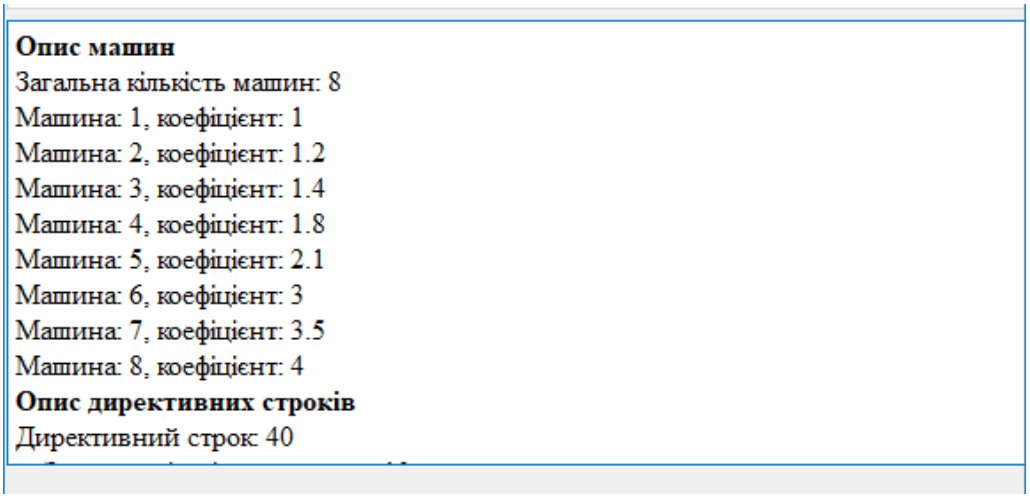

Рисунок 4.3 – Консоль програми

4.4.2 Генерація вхідних даних

Для того, щоб згенерувати дані в меню «Файл» головного вікна необхідно обрати пункт «Генерувати» (рис.4.4), або скористатися комбінацією клавіш Ctrl+G.

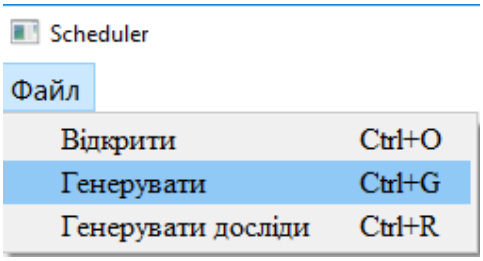

Рисунок 4.4 – Генерація вхідних даних

Користувачу відкриється форма, що представлена на рис. 4.5.

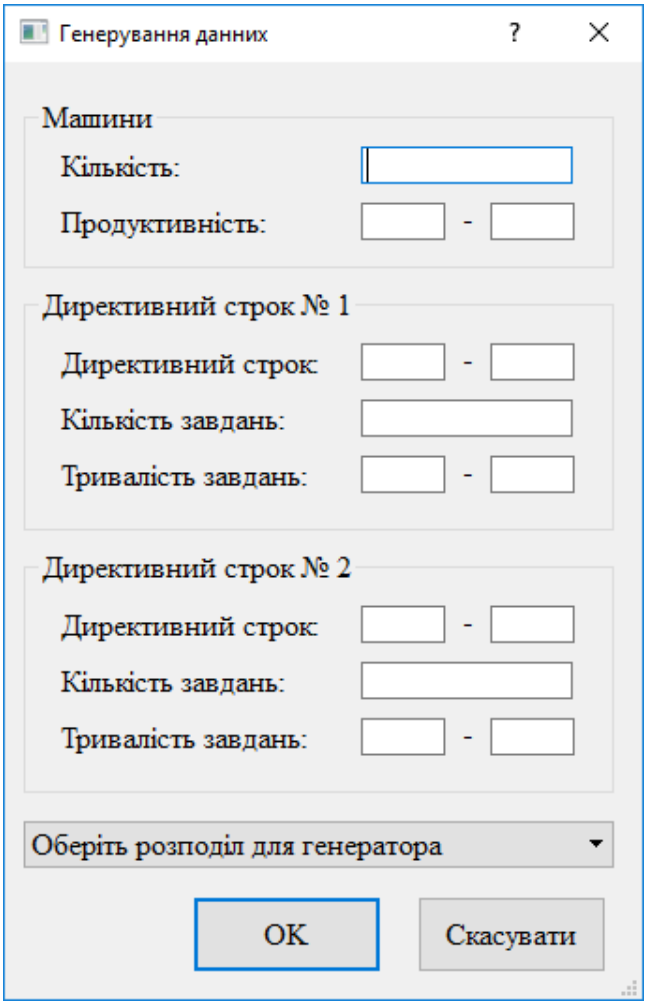

Рисунок 4.5 – Форма генерації вхідних даних

Необхідно заповнити поля форми та натиснути кнопку «ОК». Після чого програмою буде запропоновано зберегти згенерований файл.

4.4.3 Побудова розкладу

Для побудови розкладу, після зчитування файлу з вхідними даними, користувачу на головній формі програми необхідно натиснути кнопку «Побудувати розклад».

На головній формі програми стануть активними 2 вкладки: вкладка «Допустимий розклад» та вкладка «Покращення розкладу».

4.4.3.1 Допустимий розклад

Форма побудови допустимого розкладу для кожного з критеріїв оптимізації представлена на рис. 4.6.

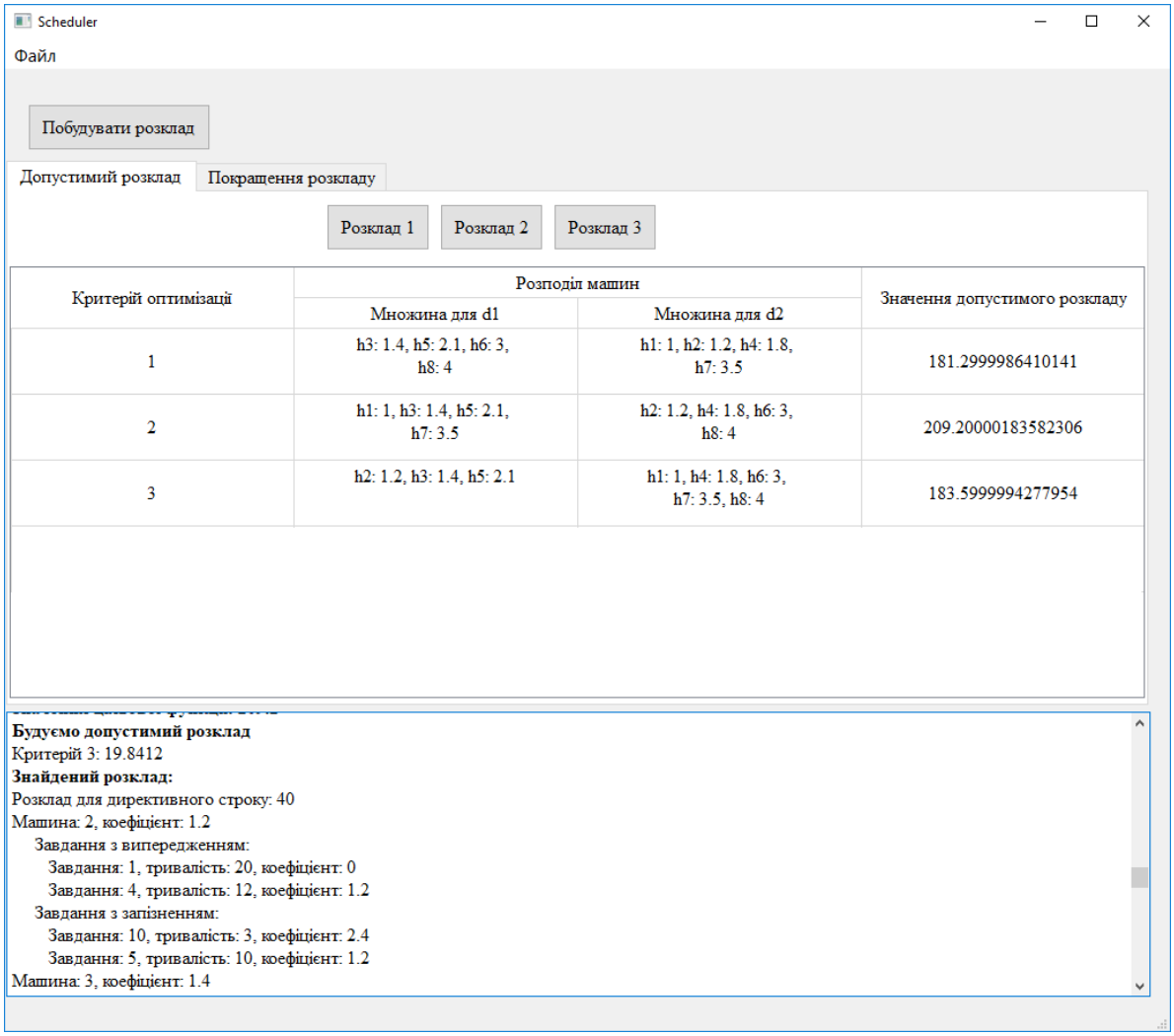

Рисунок 4.6 – Форма побудови допустимого розкладу

На формі, у таблиці представлено розбиття пристроїв за множинами завдань, а також значення допустимого розкладу для кожного з критеріїв оптимізації.

Для того, щоб переглянути візуалізацію побудованого допустимого розкладу, над таблицею необхідно натиснути кнопку «Розклад 1», «Розклад 2» чи «Розклад 3». Візуалізований розклад буде мати вигляд, що представлений на рис. 4.7.

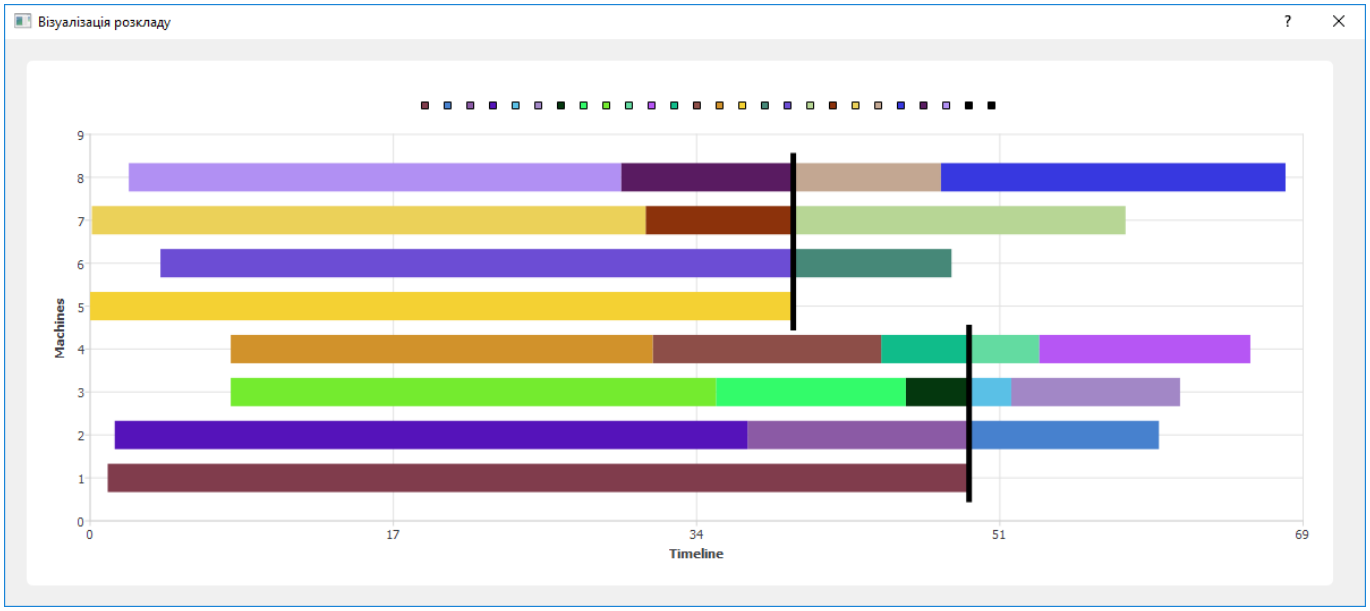

Рисунок 4.7 – Візуалізація побудованого допустимого розкладу

4.4.3.2 Покращений розклад

Для перегляду покращеного розкладу, згідно алгоритму перестановок, потрібно перейти на вкладу «Покращення розкладу» головної форми програми (рис.4.8).

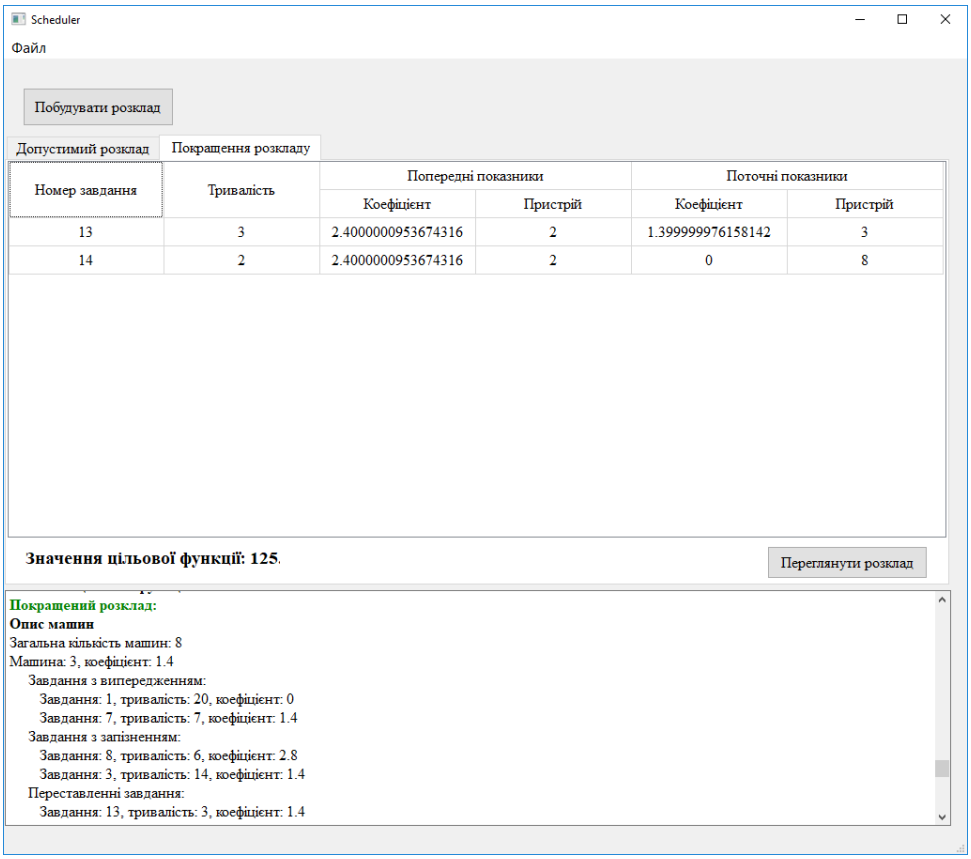

Рисунок 4.8 – Форма побудови покращеного розкладу

На формі, у таблиці представлені номери завдань, що були переставлені на виконання до пристроїв множини M_1, та їх попередні та поточні значення коефіцієнтів цільової функції.

Для перегляду покращеного розкладу потрібно натиснути кнопку «Переглянути розклад». Візуалізований розклад буде мати вигляд, що представлений на рис. 4.9.

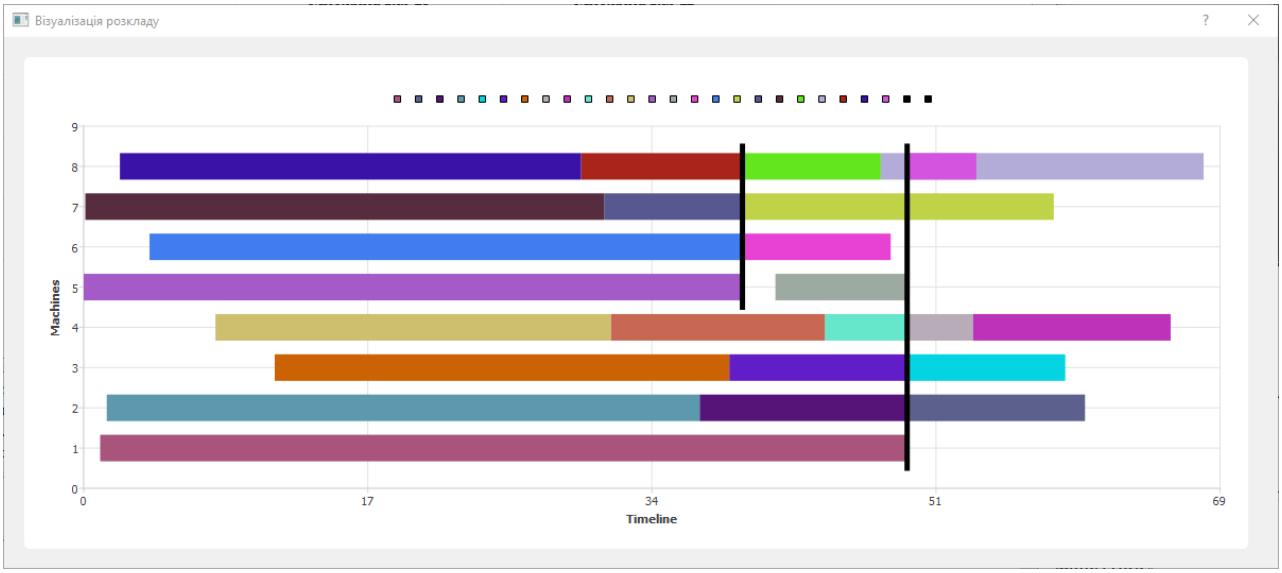

Рисунок 4.9 – Візуалізація покращеного розкладу

Висновки до розділу

У розділі наведено опис засобів розробки програмного забезпечення для складання розкладу для задачі мінімізації сумарного відхилення моментів завершення від директивних термінів для паралельних пристроїв з різною продуктивністю. Здійснено опис класів програмного забезпечення. Основними класами є: Machine, Scheduler, Distribution, Task, Logger, Generator, Criteria, MainWindow. У розділі здійснено опис атрибутів класів., та описано специфікацію функцій класів алгоритмічного забезпечення. Описано керівництво роботи з програмним забезпеченням для побудови розкладу виконання завдань з метою мінімізації сумарного відхилення моментів завершення від директивних строків на паралельних пристроях.

5 ЕКСПЕРИМЕНТАЛЬНЕ ДОСЛІДЖЕННЯ ЕФЕКТИВНОСТІ АЛГОРИТМІВ

5.1 Перевірка ефективності критеріїв оптимізації

Для перевірки ефективності розроблених критеріїв оптимізації допоміжної задачі необхідно провести серію експериментів.

Для проведення дослідження критеріїв оптимізації, порівняння їх ефективності за різних початкових умов, пропонується виділити наступні ознаки, за якими задачі можна класифікувати:

- *задачі класу 1*: $P_1 < P_2$, $d_1 < d_2$;
- *задачі класу 2*: $P_1 \approx P_2$ (сумарні тривалості робіт з множин I_1 та I_2 є приблизно однаковими), $d_1 < d_2$;
- *задачі класу 3*: директивні строки d_1 та d_2 достатньо великі, що існує часовий люфт перед початком виконання розкладу.

В таблиці 5.1 представлено параметри генерації індивідуальних задач. При дослідженні генеруємо по 500 індивідуальних задач з кожного класу. Усі величини, що задані у проміжку, розподілені рівномірно.

Таблиця 5.1 – Параметри генерації індивідуальних задач

Клас задачі	**m**	**h**	**d_1**	**d_2**	**n_1**	**n_2**	**p_i ($i \in I_1$)**	**p_i ($i \in I_2$)**
1	10	1-4	600	700	100	150	1-80	1-80
2	10	1-4	600	700	100	150	1-40	40-80
3	10	1-4	2000	4000	100	150	1-80	1-80

На рисунках 5.1–5.3 представлені представлені результати розрахунків: значення цільової функції (ЦФ) в початкових розкладах згенерованих індивідуальних задач в залежності від критерію оптимізації ДОЗ.

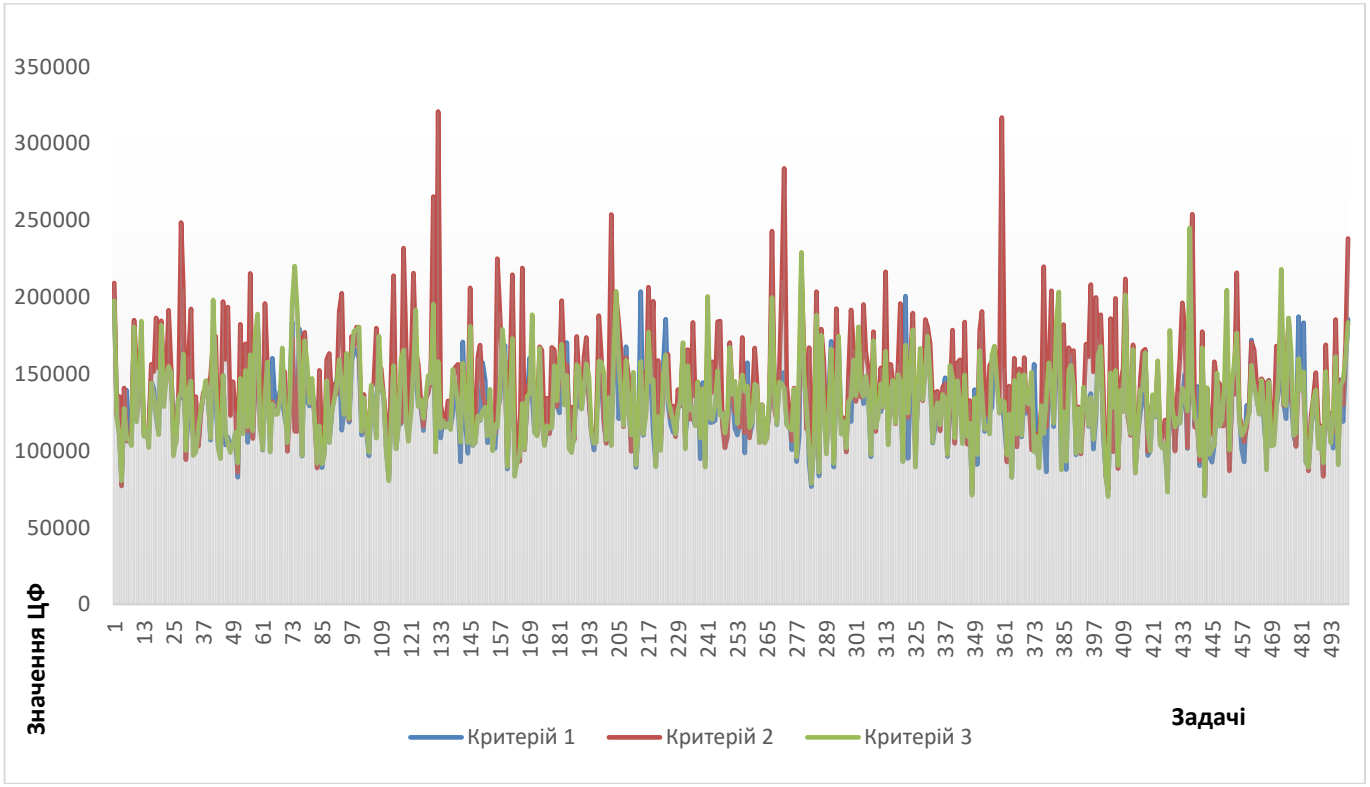

Рисунок 5.1 – Залежність значення ЦФ початкового розкладу від критерію ДОЗ для задач класу 1

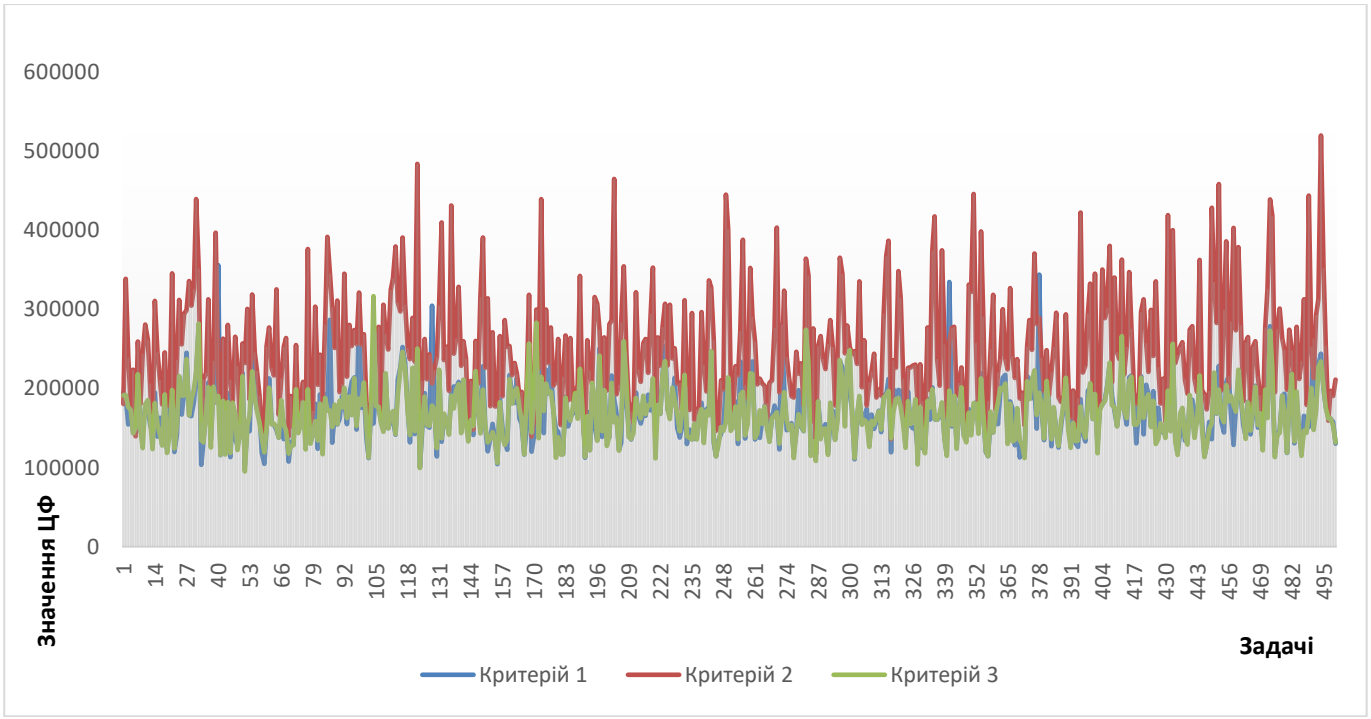

Рис. 5.2 – Залежність значення ЦФ початкового розкладу від критерію ДОЗ для задач класу 2

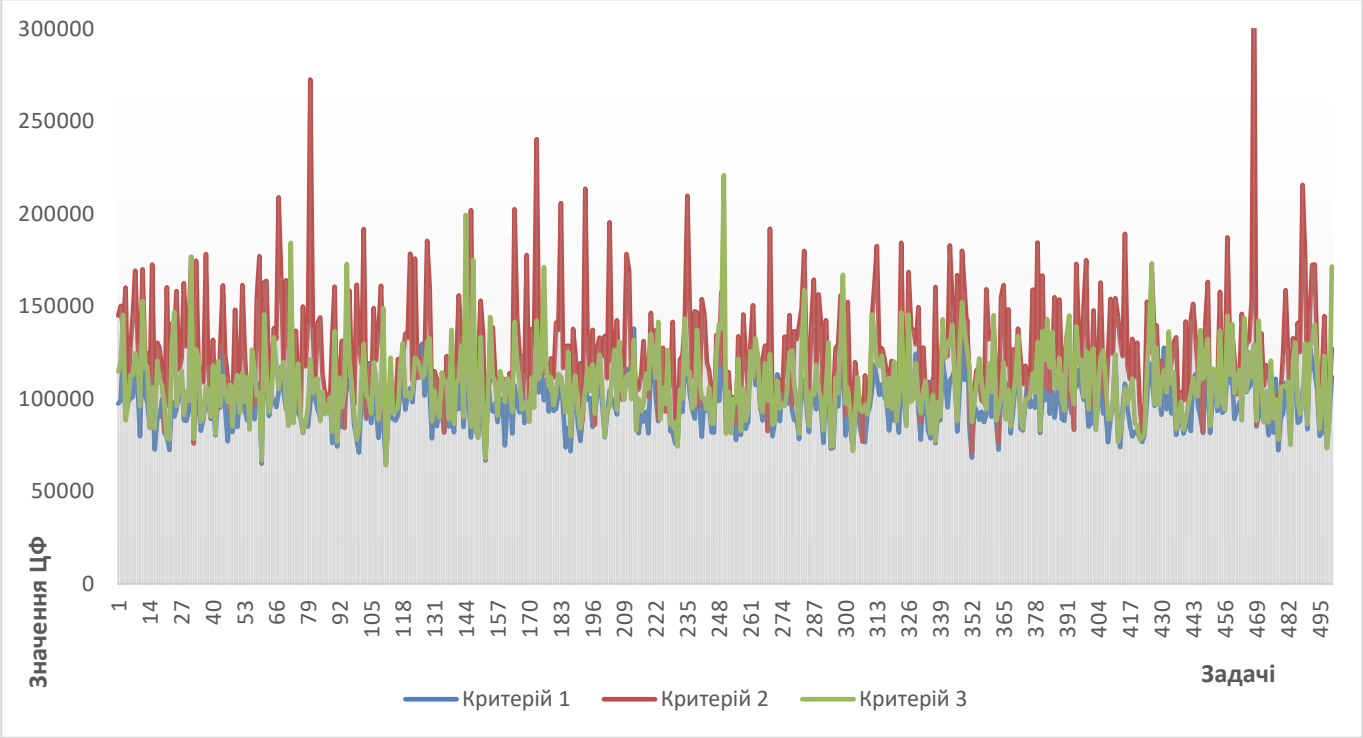

Рис. 5.3 – Залежність значення ЦФ початкового розкладу від критерію ДОЗ для задач класу 3

На рисунках 5.4 – 5.6 представлено значення кращого початкового розкладу у відсотках в залежності від критерію ДОЗ для задач кожного класу.

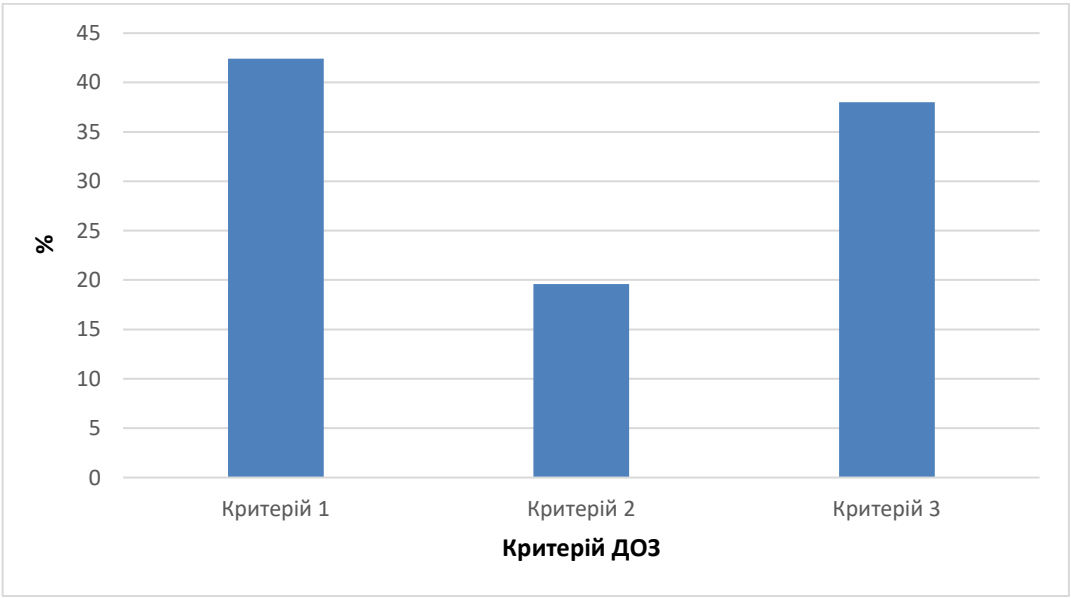

Рисунок 5.4 – Відсоток знаходження кращого початкового розкладу в залежності від критерію ДОЗ для задач класу 1

Для задач класу 1 відносна кількість випадків (у %) побудови початкового розкладу з кращим значенням ЦФ залежно від використаного в ДОЗ критерію закріплення пристроїв за множинами завдань становила:

– критерій 1: 42%;
– критерій 2: 20%;
– критерій 3: 38.

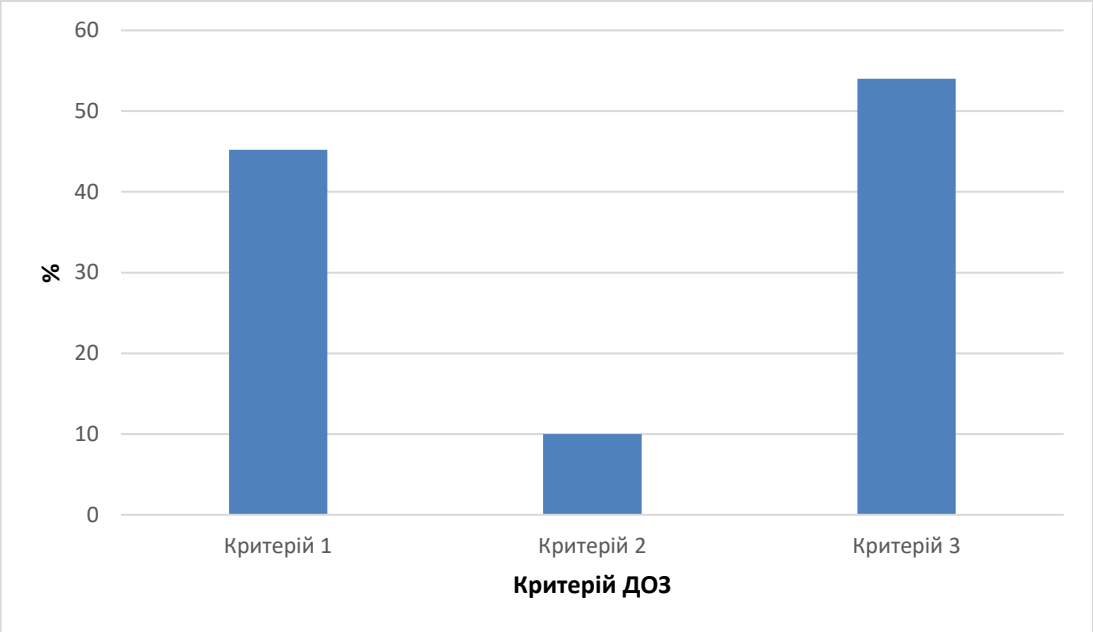

Рис.5.5 – Відсоток знаходження кращого початкового розкладу в залежності від критерію ДОЗ для задач класу 2

Для задач класу 2 відносна кількість випадків (у %) побудови початкового розкладу з кращим значенням ЦФ залежно від використаного в ДОЗ критерію закріплення пристроїв за множинами завдань становила:

– критерій 1: 36%;
– критерій 2: 10%;
– критерій 3: 54%.

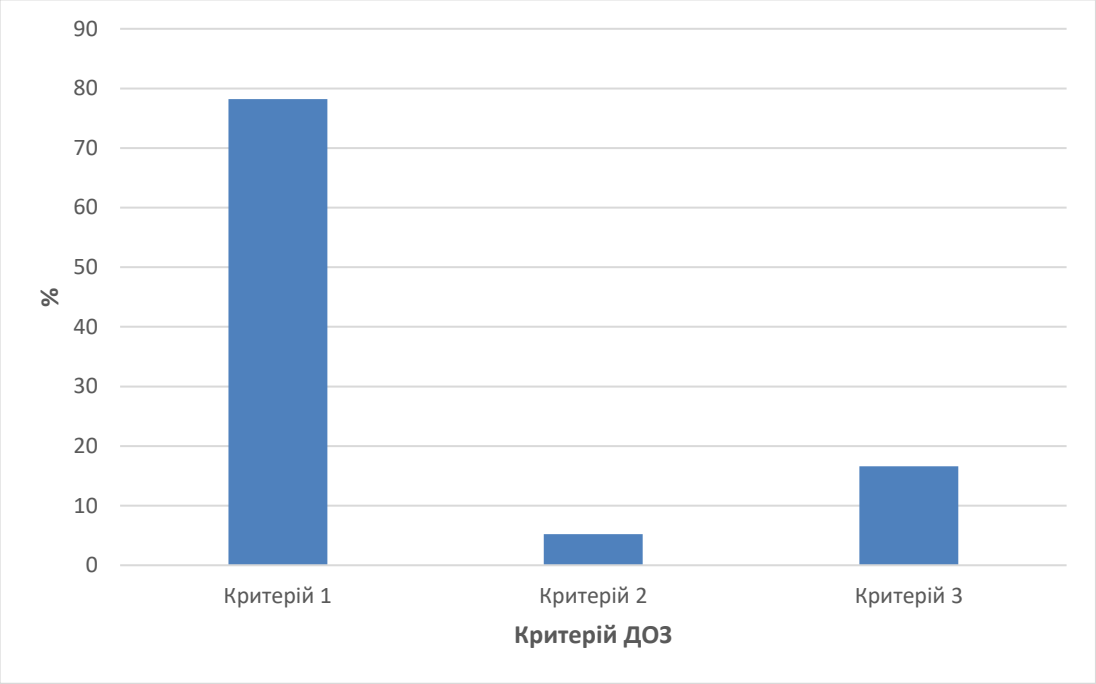

Рис.5.6 – Відсоток знаходження кращого початкового розкладу в залежності від критерію ДОЗ для задач класу 3

Для задач класу 3 відносна кількість випадків (у %) побудови початкового розкладу з кращим значенням ЦФ залежно від використаного в ДОЗ критерію закріплення пристроїв за множинами завдань становила:

- критерій 1: 78,2%;
- критерій 2: 5,2%;
- критерій 3: 16,6%.

5.2 Перевірка ефективності алгоритму перестановок для покращення допустимого розкладу

Для перевірки алгоритму перестановок, пропонується збільшувати розрив між директивними строками d_1 та d_2 для збільшення кількості можливих перестановок для покращення допустимого розкладу. На рис.5.7 представлено графік залежності значень цільової функції від розриву між директивними строками.

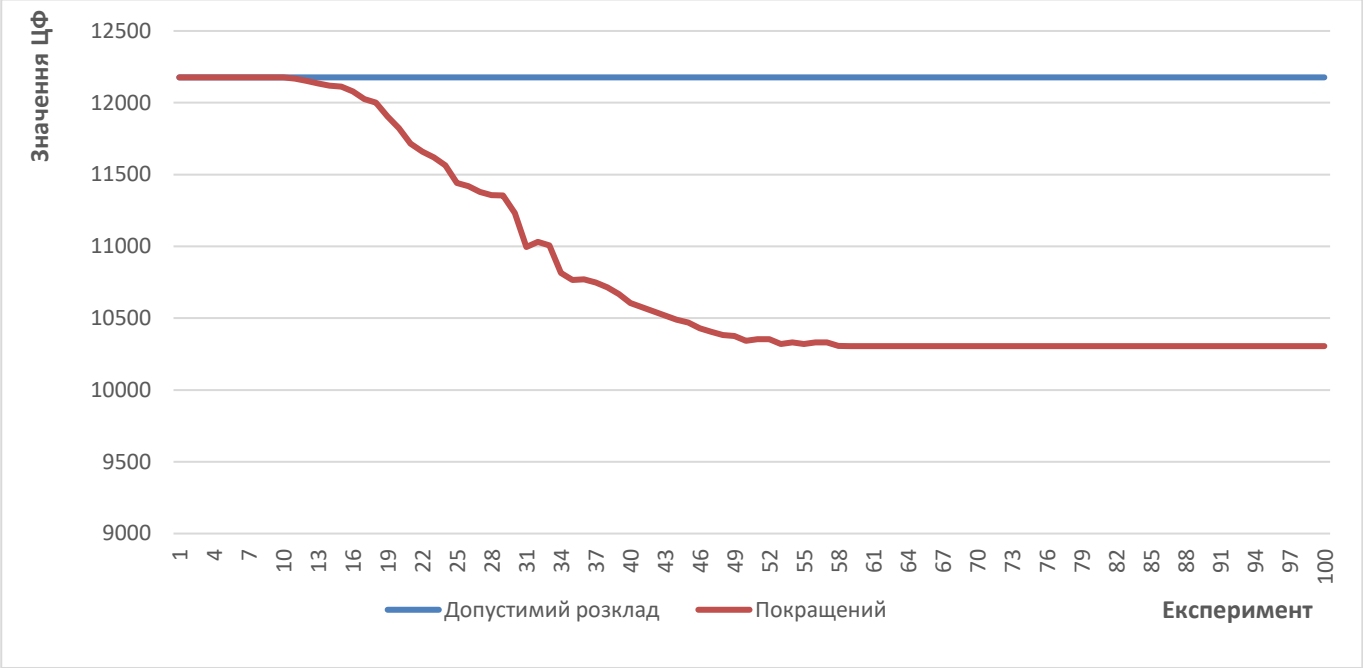

Рисунок 5.7 – Залежність значення ЦФ від розриву між директивними строками

На рисунку 5.8 представлено залежність часу роботи алгоритму від розмірності вхідної задачі (кількості завдань).

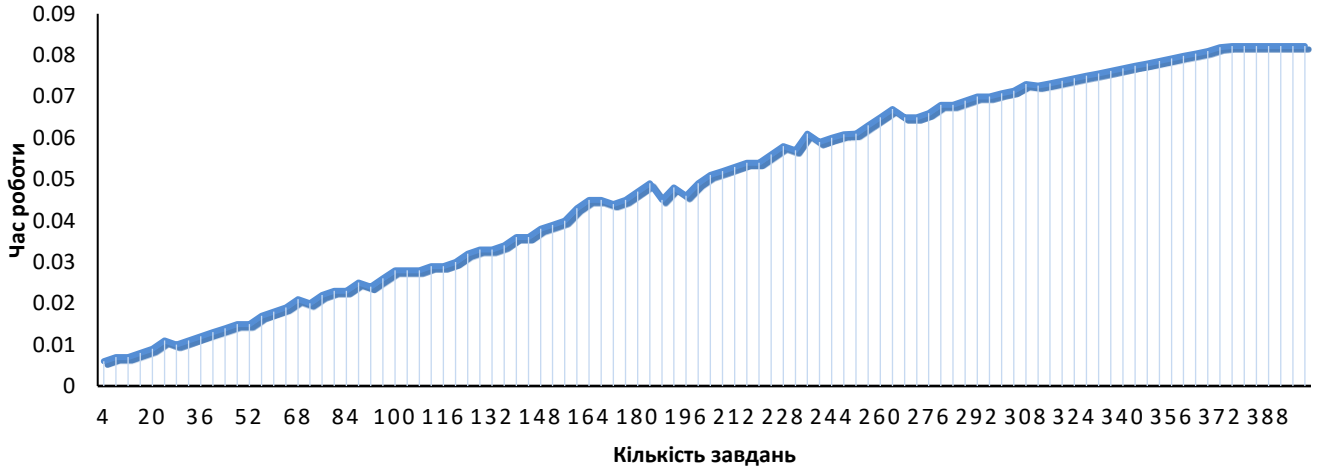

Рисунок 5.8 – Залежність часу роботи алгоритму від кількості завдань

Висновки до розділу

Для перевірки ефективності розробленого алгоритму була проведена серія експериментів. Для перевірки критеріїв допоміжної оптимізаційної задачі, було виділено 4 класи задач. Для досліджені було згенеровано по 500 індивідуальних задач кожного класу. На підставі проведених експериментів з критеріями оптимізації можна зробити висновки:

- для задач, у яких сумарна тривалість завдань множини I_1 є меншою за тривалість виконання завдань множини I_2 при розв'язанні ДОЗ доцільно використовувати 1-й критерій оптимізації;
- для задач, у яких сумарні тривалості робіт з множин I_1 та I_2 є приблизно однаковими доцільно використовувати 3-й критерій оптимізації;
- для задач, у яких директивні строки d_1 та d_2 достатньо великі, що існує часовий люфт перед початком виконання розкладу доцільно використовувати 3-й критерій оптимізації;
- при збільшенні розриву між директивними строками збільшується кількість можливих перестановок, що призводить до знаходження кращих розв'язків у порівнянні з допустимим рішенням.

ВИСНОВКИ

Під час виконання роботи були розглянуті питання календарного планування на виробництві. Було розглянуто сучасні методології управління виробництвом та існуючі методи розв’язку деяких видів задач складання розкладів для паралельних пристроїв.

Була розглянута задача мінімізації сумарного відхилення моментів завершення від директивних строків при виконанні завдань паралельними пристроями. Була наведена постановка задачі, її властивості, а також наведено алгоритм побудови допустимого розкладу та перестановок для покращення допустимого розкладу.

За результатами роботи можна зробити висновки, що всі поставлені завдання були виконані, а саме:

- проведено огляд відомих результатів з розв’язання поставленої в рамках роботи задачі;
- розроблено алгоритм створення календарного плану виконання завдань паралельними пристроями, що мінімізує сумарне відхилення моментів завершення від директивних строків;
- виконана програмна реалізація алгоритму. Для створення програмного продукту було використано мову програмування C++ та платформу QT, в якості середовища розробки використовувалось IDE QT Creator;
- було проведено серію експериментів, що дозволили зробити висновки про ефективність алгоритму при різноманітних вхідних даних.

За матеріалами дисертації було опубліковано три наукові роботи: стаття у фаховому виданні та двоє тез доповідей на наукових конференціях [123-125].

ПЕРЕЛІК ПОСИЛАНЬ

1. Bellman, R., Gross, O. Some combinatorial problems arising in the theory of multistage processes // Journ. Soc. industr. and appl. mathematics. 1945. Vol. 2. No. 3.

2. Johnson, S.M. Optimal two- and three-stage production schedules with setup times included // Nav. res. log. quart. 1954. Vol. 1. No. 1.

3. Bellman, R. Mathematical aspects of scheduling theory // Journ. Soc. industr. and appl. mathematics. 1956. Vol. 4. No. 3

4. Танаев, В.С. Введение в теорию расписаний / В.С. Танаев, В.В. Шкурба. – М.: Наука, 1975.

5. Левин, В.И. К планированию работы вычислительных систем. I. II, III (Математический аппарат, анализ плана, синтез плана) / В.И. Левин // Автоматика и вычисл. техника. 1982. № 5; 1983. №№ 2, 3.

6. Левин, В.И. Оптимизация расписания обработки деталей с помощью смешанных условий оптимальности / В.И. Левин // Math. Operationsforsch. und Statist., Ser. Optimization. 1987. Vol. 18. No. 5.

7. Szwarc, W. Elimination Methods in the m× n Sequencing Problem // Nav. Res. Log. Quart. 1971. Vol. 18. No. 3.

8. Бурдюк, В.Я. О задаче m станков ($m \geq 2$) / В.Я. Бурдюк // Кибернетика. 1969. № 3.

9. McMahon, G.B., Burton, P.G. Flow-Shop Scheduling with the Branch-andBound Method // Oper. Res. 1967. Vol. 15. No. 3.

10. Lomnicki, Z.A. A «Branch-Bound» Algorithm for the Exact Solution of the Three-Machine Scheduling Problem // Oper. Res. Quart. 1965. Vol. 16. No. 1.

11. Heller, J. Some Numerical Experiments for an M J × Flow Shop and Its Decision-Theoretical Aspects // Oper. Res. 1960. Vol. 8. No. 2.

12. Elmaghraby, S.E. The One Machine Sequencing Problem with Delay Costs // Journ. Ind. Eng. 1968. Vol. 19. No. 2.

13. Bowman, E.H. The Schedule-Sequencing Problem // Oper. Res. 1959. Vol. 7. No. 5.

14. Giglio, R.J. Wagner H.M. Approximate Solutions to the Three-Machine Scheduling Problems // Oper. Res. 1964. Vol. 12. No. 2.

15. Лурье, А.Л. О некоторых задачах календарного планирования // Сб. научн. трудов. М.: Наука. 1962. Вып. 7.

16. Гэри, М. Вычислительные машины и труднорешаемые задачи / М. Гэри, Д. Джонсон. – М.: Мир, 1982.

17. Левин, В.И. Логические методы исследования вычислительных систем реального времени / В.И. Левин // Автоматика и вычисл. техника. 1985. № 3.

18. Левин, В.И. Задача М станков при поступлении деталей в режиме реального времени / В.И. Левин // Автоматика и телемеханика. 1989. № 1.

19. Серик, А.Е. Использование интервалов очередности для решения задач очередности с ограничениями / А.Е. Серик // Кибернетика. 1980. № 4.

20. Левин, В.И. Задача m станков при ограничениях на порядок следования деталей / В.И. Левин // Автоматика и телемеханика. 1987. № 3.

21. Севастьянов, С.В. Эффективное построение расписаний в системах откры- того типа / С.В. Севастьянов // Сиб. журн. исследования операций. 1994. Т. 1. № 1.

22. Local Search in Combinatorial Optimization. 1997. Chichester: John Wiley

23. Левин, В.И. Задача трех станков с неопределенными временами обработки / В.И. Левин // Автоматика и телемеханика. 1996. № 1.

24. Rayward-Smith V.J. UET scheduling with unit interprocessor communication delays // Discrete Applayd Mathematics 1987. Vol. 18 P. 55–71.

25. Drozdowski M. Scheduling for parallel processing.London: Springer-Verlag, 2009.

26. Chr´etienne P., Coffman Jr. E.J., Lenstra J.K., and Liu. Z. Scheduling Theory and Its Applications. – New-York: Wiley, 1995.

27. Корбут А.А., Сигал И.Х., Финкельштейн Ю.Ю. Гибридные методы в дискретном программировании // Изв. АН СССР. Техн.кибернет. 1988. № 1. С. 65–77.

28. Каширских К.Н., Поттс К.Н., Севастьянов С.В. Улучшенный алгоритм решения двухмашинной задачи flow shop с неодновремен- ным поступлением работ // Дискретный анализ и исследование операций. 1997.– Т. 4, № 1.– С. 13 – 32.

29. Sevastianov S.V., Tchernykh I.D. Computer-Aided Way to Prove Theorems in Scheduling // Bilardi G., Italiano G.F., Pietracaptina Литература 222 A., Pucci G. (eds.) Proceedings of Sixth Annual European Symposium on Algorithms ESA'98, 24 – 26 august.– Venice, Italy: 1998.– SpringerVerlag, LNCS, V. 1461, 1998.– P. 502 – 513.

30. Сигал И.Х., Иванова А.П. Введение в прикладное дискретное программирование: теория и вычислительные алгоритмы. М.: Физмат- лит, 2002. 240 с

31. Brooks G.N., White C.R. An algorithm for finding optimal or near – optimal solutions to the production scheduling problem // J. Ind. Eng.– 1965.– V. 16, N 1.– P. 34 – 40.

32. Carlier J. The one-machine sequencing problem // European J. of Oper. Res.– 1982.– V. 11, N 1.– P. 42 – 47

33. Беллман Р. Динамическое программирование М.: ИЛ, 1960. 400 с

34. Гэри М., Джонсон Д. Вычислительные машины и труднорешаемые задачи. Пер. с англ. М.: Мир, 1982. 416 с

35. Jain V., Grossmann I.E. Algorithms for hybrid MILP/CLP models for a class of optimization problems // INFORMS J. Computing.– 2001.– V. 13.– P. 258 – 276.

36. Сергиенко И.В. Задачи дискретной оптимизации. Проблемы, методы решения, исследования / И.В. Сергиенко, В.П. Шило. – К.: Наукова думка, 2003. – 260 с.

37. Жиглявский А.А. Методы поиска глобального экстремума / А.А. Жиглявский, А.Г. Жилинскас. – М.: Наука, 1991. – 205 с.

38. Гурин Л.С. Задачи и методы оптимального распределения ресурсов / Л.С. Гурин, Я.С. Дымарский, А.Д. Меркулов. – М.: Сов. Радио, 1968. – 464 с.

39. Зак Ю.А. Методы оптимизации и их применение в целлюлозно-бумажной промышленности / Ю.А. Зак, Р.М. Рейдман, А.А. Рувинский. – М.: Лесная промышленость, 1973. – 248 с.

40. Zack Yu. A. Methods of Multiextremal Optimization under Constraints for Separably Quasimonotone Functions / Yu Zack // Journal of Computer and Systems Sciences International, 2011. – vol.50, №3. – P. 37 – 391.

41. Glover F. Tabu Search, Part II / F. Glover // ORSA Journal on Computing, 1990. – vol.2, №1. – P. 4-32.

42. Емельянов В.В. Теория и практика эволюционного моделирования / В.В. Емельянов, В. В. Курейчик. – М.: Физматлит, Наука, 2003. – 432 с.

43. Курейчик В.М. Генетические алгоритмы: Монография / В.М. Курейчик. – Таганрог: Изд.ТРТУ, 1998. – 242 с.

44. Cleveland G.A. Using genetic algorithms to schedule flow shop releases / G.A. Cleveland Smith S.F.// In. Proceeding of theThird International Conference on Genetic Algorithms. Morgan Kaufmann Publishers. – San Mateo,California. – 1989. – P. 160-169.

45. Glover F. Tabu Search, Part I / F. Glover // ORSA Journal on Computing. – 1989 – Vol. 1, No 3. – P. 190-206.

46. Glover F. Tabu Search, Part II / F. Glover // ORSA Journal on Computing. – 1990 – Vol. 2, No 1. – P. 4-32.

47. Nissen Volker. Einfuhrung in Evolutionare Algorithmen. / Volker Nissen // Optimierung nach dem Vorbild der Evolution. – Vieweg, Munchen, 1997. –P. 345.

48. Goldberg David E. Genetic Algorithms in Search / David E. Goldberg // Optimization, and Machine Learning. – Adison-Wesley, 1998. – 403 s.

49. Michalewicz Z. Heuristic methods for evolutionary computation tech-niques / Z. Michalewicz //Journal of Heuristics. – 1995. - №1. – P. 177-206.

50. Michalewicz Z. Genetic Algorithms + Data Structures = Evolution Programs / Z. Michalewicz. – Springer, Berlin, 1999. – P. 67.

51. Подчасова Т.П., Португал В.М. и др. Эвристические методы календарного планирования / Т.П.Подчасова, В.М. Португал и др. – Киев.: Техника, 1980. – 140 с.

52. Метод комбинаторных евристик для решения комбинаторных задач упорядочения и распределения ресурсов / Д.И. Батищев, Э.Д. Гудман, И.П. Норенков, М.Х. Прилуцкий. – М.: Информационные технологи, 1997. – С. 29-32.

53. Hundal T.S. An extension of Palmer’s heuristic for the flow-shop scheduling problem / T.S. Hundal, J. Rajgopal // International Journal of Produktion Reasearch. – 1988. – №26ю – P. 1119 – 1124.

54. Gupta J.N.D. A functional heuristic algorithm for the flop-shop scheduling problem / J.N.D. Gupta // Operational Research Quartrrly. – 1971. – №2. – P. 39-47.

55. Cambell H.G. A heuristic algorithm for the n job, m machine sequencing problem / H.G. Cambell, R.A. Dudek, M.L. Smith // Management Science. – 1970 – №16. – P. 630-637.

56. Monma C.L. Analysis of heuristics for preemptive parallel machine scheduling with batch setup times [Текст] / C.L. Monma, C.N. Potts // Operations Research. – 1993. – Vol. 41. – P. 981–993.

57. Lawler E.L. Branch-and-bound methods: A Survey / E.L. Lawler, D.E. Wood // Oper. Res. – 1966. – Vol.4, §14. – P. 252 – 260.

58. Land A.H. An automatic method of solving discrete programming problems / A.H. Land, A.G. Doig // Econometrica. – 1960. – Vol.28. – P. 497–520.

59. Корбут А.А. Дискретное программирование / А.А. Корбут, Ю.Ю. Финкельштейн. – М.: Наука, Физматгиз, 1969. – 368 с.

60. Беллман Р. Динамическое программирование и современная теория управления = Dynamic Programming and Modern Control Theory / Р. Беллман, Р.Калаба. – Пер. с англ. – М.: Наука, 1969. – 119 с.

61. Лежнев А.В. Динамическое программирование в экономических задачах / А.В. Лежнев. – Бином. Лаборатория знаний, 2010. – 176 с.

62. Webster S. Dynamic programming algorithms for scheduling parallel machines with family setup times [Текст] / S. Webster, M. Azizoglu // Computers & Operations Research. – 2001. – Vol. 28. – P. 127–137.

63. Lawler, E. L. A ”pseudopolynomial” algorithm for sequencing jobs to minimize total tardiness. P.L. Hammer, E.L. Johnson, B. K. and Nemhauser, G. (Eds.), Studies in Integer Programming, volume 1 of Annals of Discrete Mathematics, p. 331–342. Elsevier, 1977.

64. Sourd, F. and Kedad-Sidhoum, S. (2003), The one-machine problem with earliness and tardiness penalties. Journal of Scheduling, v. 6, n. 6, p. 533–549

65. Sourd, F. (2005), Earliness-tardiness scheduling with setup considerations. Computers & Operations Research, v. 32, n. 7, p. 1849 – 1865.

66. Sourd, F. and Kedad-Sidhoum, S. (2008), A faster branch-and-bound algorithm for the earliness-tardiness scheduling problem. Journal of Scheduling, v. 11, n. 1, p. 49–58.

67. Tanaka, S. and Fujikuma, S. An efficient exact algorithm for general single-machine scheduling with machine idle time. Automation Science and Engineering, 2008. CASE 2008. IEEE International Conference on, p. 371–376, 2008.

68. Sourd, F. (2009), New exact algorithms for one-machine earliness-tardiness scheduling. INFORMS Journal on Computing, v. 21, n. 1, p. 167–175.

69. Tanaka, S., Fujikuma, S. and Araki, M. (2009), An exact algorithm for single-machine scheduling without machine idle time. Journal of Scheduling, v. 12, n. 6, p. 575–593.

70. Pessoa, A., Uchoa, E., de Arago, M. and Rodrigues, R. (2010), Exact algorithm over an arc-time-indexed formulation for parallel machine scheduling problems. Mathematical Programming Computation, v. 2, n. 3-4, p. 259–290.

71. Tanaka, S. and Fujikuma, S. (2012), A dynamic-programming-based exact algorithm for general single-machine scheduling with machine idle time. Journal of Scheduling, v. 15, n. 3, p. 347–361.

72. Baker K.R. Sequencing with earliness and tardiness penalties: a review / Baker K.R., Scudder G.D. // Operations Research. – 1990. – № 38 (1). – P. 22–36.

73. Lauff V. Scheduling with common due date, earliness and tardiness penalties for multimachine problems: a survey / V. Lauff, F. Werner. // Mathematical and Computer Modelling. – 2004. – №40(5-6). – P. 637–655.

74. Garey M.R. One–processor scheduling with symmetric earliness and tardiness penalties / M.R. Garey, R.E. Tarjan, G.T. Wilfong // Mathematics of Operations Research. – 1988. – №13. – P.330–348

75. Yano C.A. Algorithms for a class of single–machine weighted tardiness and earliness problems [Текст] / C.A. Yano, Y.D. Kim // European Journal of Operations Research. – 1991. – Vol.52. – P.167–178

76. Ow P.S. The single machine early/tardy problem / P.S. Ow, T.E. Morton // Management Science. – 1989. – Vol.2. – №35. – P.177–191

77. Davis J.S. Single–machine scheduling with early and tardy completion costs / J.S. Davis, J.J. Kanet // Naval Research Logistics. – 1993. – №40.– P. 85–101

78. Szwarc W. Optimal timing scheduling in earliness–tardiness single machine sequencing / W. Szwarc, S.K. Mukhopadhyay // Naval Research Logistics. – 1995. – №42. – P.1109–1114

79. Sridharan V. A decision theory based scheduling procedure for single–machine weighted earliness and tardiness problem [Текст] // V. Sridharan, Z. Zhou // European Journal of Operations Research. – 1996. – Vol.94. – P.292–301

80. Wan G., Yen B.P.C. Tabu search for single–machine scheduling with distinct due windows and weighted earliness/tardiness penalties // European Journal of Operations Research. – №142. – 2002. – P.271–281

81. Szwarc W. Adjacent ordering in single–machine scheduling with earliness and tardiness penalties // Naval Research Logistics. – 1993. – №40. – P.229–24

82. Lee C.Y., Choi J.Y. A generic algorithm for job sequencing problem with distinct due dates and general early–tardy penalty weights // Computers & Operational Research. – 1995. – №22. – P.857–869

83. Gordon V., Proth J.P., Chu C. A survey of the state–of–art of common due date assignment and scheduling research // European Journal of Operations Research. – 2002. – №139. – P.1–25

84. Valente J.M.S., Alves R.A.F.S. Filtered and recovering beam search algorithms for the early/tardy scheduling problem with no idle time // Computers & Industrial Engineering. – 2005. – №48(2) . – P.363–375

85. Feldmann M., Biskup D. Single–machine scheduling for minimizing earliness and tardiness penalties by meta–heuristic approaches // Computers & Industrial Engineering.– 2003.– №44.– P.307–323

86. Tung–I Tsai. A genetic algorithm for solving the single machine earliness/tardiness problem with distinct due dates and ready times // International Journal of Advanced Manufacturing Technologies. – 2007. – №32. –P.994–1000

87. Hoogeveen J.A., Van de Velde L.S. A branch–and–bound algorithm for single–machine earliness–tardiness scheduling with idle time // INFORMS Journal of Computing. – 1996. – №8. – P.402–412.

88. Ratli, M., Benmansour, R., Macedo, R., Hanafi, S. and Wilbaut, C. Mathematical Programming and Heuristics for Scheduling Problems with Early and Tardy Penalties, p. 183–223. John Wiley & Sons, Inc., 2013.

89. J´ozefowska, J. Just-in-Time Scheduling: Models and Algorithms for Computer and Manufacturing Systems. International Series in Operations Research & Management Science. Springer, 2007.

90. Минухин С. В. Метод минимизации суммарного запаздывания работ на одиночном устройстве рангового подхода и правил доминирования / С. В. Минухин, Д. С. Ленько. // Электронное моделирование. – 2014. – №36. – С. 57–79.

91. Кварацхелія А.Г. Методи вирішення задачі мінімізації сумарного запізнення для одного пристрою і задачі розбиття [Електронний ресурс] / А.Г. Кварацхелія // Наукова бібліотека дисертацій і авторефератів disserCat. – 2007. Режим доступу: http://www.dissercat.com/content/metody-resheniya-zadachi-minimizatsii-summarnogo-zapazdyvaniya-dlya-odnogo-pribora-i-zadachi#ixzz4Umeb7h8m

92. Chinyao Low, Rong-Kwei Li, and Guan-He Wu, “Minimizing Total Earliness and Tardiness for Common Due Date Single-Machine Scheduling with an Unavailability Interval,” Mathematical Problems in Engineering, vol. 2016, Article ID 6124734, 12 pages, 2016. doi:10.1155/2016/6124734

93. Згуровский М.З., Павлов А.А. Принятие решений в сетевых системах с ограниченными ресурсами: Монография. – К.: Наукова думка, – 2010. – 573 с.

94. Павлов А.А. ПДС-алгоритмы решения задач составления расписаний по критерию опережения/запаздывания на одном приборе / А. А. Павлов, Е. Б. Мисюра // Вісник Національного технічного університету України "КПІ". Інформатика,

управління та обчислювальна техніка. - 2014. - Вип. 60. - С. 4-19. - Режим доступу: http://nbuv.gov.ua/UJRN/Vkpi_iuot_2014_60_3.

95. Лазарев А. А. Теория расписаний. Задачи суммарного запаздывания для одного прибора / А. А. Лазарев, Е. Р. Гафаров., 2011. – 85 с. – (Ламберт).

96. Шевченко К. Ю. Алгоритм гілок та меж для статистичних досліджень нового ПДС-алгоритму розв’язання задачі мінімізації сумарного зваженого запізнення виконання робіт на одному приладі / К. Ю. Шевченко. // вісник НТУУ «КПІ» інформатика, управління та обчислювальна техніка. – №55. – с. 56–69.

97. 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. – №2. – С. 7–15.

98. Ващук Ф. Г. Інформаційне забезпечення алгоритмів мінімізації сумарного випередження і запізнення із налагодженнями / Ф. Г. Ващук, О. Б. Мельник, О. О. Місюра. // Системи обробки інформації. – 2012. – №2. – С. 250–259.

99. Мельник О. О. Система моделювання для дослідження ефективності алгоритмів розв'язання задач планування / О. О. Мельник. // Науковий вісник Ужгородського національного уніврситету. – 2015. – №1. – С. 84–88.

100. Шейко В., Кушнаренко Н. Організація та методика науково-дослідницької діяльності. –К.: Знання-Прес, 2002–295 с.

101. Low C. Minimizing the sum of absolute deviations under a common due date for a single-machine scheduling problem with availability constraint // Journal of Industrial and Production Engineering. – 2015. – P.204-2017

102. Складання розкладів сумарного випередження і запізнення із налагодженнями, що залежать від послідовності / Ф. Г.Ващук, О. А. Павлов, О. Б. Місюра, О. О. Мельник. // Вісник НТУУ «КПІ» Інформатика, управління та обчислювальна техніка. – №53. – С. 192–194.

103. Згуровский М.З., Павлов А.А. Принятие решений в сетевых системах с ограниченными ресурсами: Монография.– К.: Наукова думка, – 2010.– 573 с.

104. Долгова О. Э. Составление расписаний с минимизацией суммарного запаздывания на одном приборе методом параллельных муравьиных колоний / О. Э. Долгова, В. В. Пересветов. // Информатика, вычислительная техника и управление. – 2012. – №2. – С. 45–52.

105. Holthaus O., Rajendran C.A fast ant-colony algorithm for single-machine scheduling to minimize the sum of weighted tardiness of jobs // Journal of the Operational Research Society. 2005. No. 56

106. Kedad-Sidhoum, S., Solis, Y. R. and Sourd, F. (2008), Lower bounds for the earliness-tardiness scheduling problem on parallel machines with distinct due dates. European Journal of Operational Research, v. 189, n. 3, p. 1305 – 1316.

107. Bank J., Werner F. Heuristic algorithm for unrelated parallel machine scheduling with a common due date, release dates, and linear earliness and tardiness penalties // Math Comput Model. – 2001. – №33. – P.363–383

108. Kramer A. A unified heuristic and an annotated bibliography for a large class of earliness-tardiness scheduling problems [Електронний ресурс]/ A. Kramer, A. Subramanian. – Brazil, Working Paper UFPB, 2015. – Режим доступу до ресурсу: journals/corr/KramerS15.

109. Красовский Д.В. Алгоритмы решения минимаксной задачи составления расписания / Д.В. Красовский, М.Г. Фуругян // Известия РАН. Теория и системы управления. – 2008. – №5. – С. 69–74.

110. Симанчёв Р. Ю. Многогранник расписаний обслуживания идентичных требований параллельными приборами / Р. Ю. Симанчёв, И. В. Уразова. // Дискретный анализ и исследование операций. – 2011. – №1. – С. 85–97.

111. Кобак В. Г. Сравнительный анализ приближенных алгоритмов решения минимаксной задачи для однородных приборов / В. Г. Кобак, Д. М. Будиловский. – 2006.

112. Chaudhry, I.A. & Drake, P.R. Int J Adv Manuf Technol (2009) 42: 581. https://doi.org/10.1007/s00170-008-1617-z

113. Hu P-C (2004) Minimising total tardiness for the worker assignment scheduling problem in identical parallel-machine models. Int J Adv Manuf Technol 23:383–388

114. Alvarez-Valdes R. Minimizing weighted earliness–tardiness on parallel machines using hybrid metaheuristics // Computers & Operations Research. – 2014. – С. 1–11.

115. Kuo-Ching Ying. Minimising total weighted earliness and tardiness penalties on identical parallel machines using a fast ruin-and-recreate algorithm // International Journal of Production Research. – 2015.

116. Зак Ю.А. Прикладные задачи теории расписаний и маршрутизации перевозок / Ю.А. Зак. – М.: Книжный дом «ЛИБРОКОМ», 2012. – 394 с.

117. Stamatopoulos, P., Viglas, E., Karaboyas, S. (1998). Nearly optimum timetable construction through clp and intelligent search. International Journal on Artificial Intelligence Tools, 07(04), pp.415-442.

118. Логоша Б.А., Петропавловская А.В. Комплекс моделей и методов оптимизации расписания занятий в вузе// Экономика и математические методы, Т 29, 1993, №4.Гохман О. Г. Экспертное оценивание/О.Г.

119. Безгинов А. Н. Комплекс алгоритмов построения расписания вуза. структура представления данных и алгоритм построение опорного решения / А. Н. Безгинов, С. Ю. Трегубов. // Вестник Балтийского федерального университета им. Канта. – 2011. – №10. – С. 93–102.

120. Chaudhry, I.A. Int J Adv Manuf Technol (2010) 48: 747. https://doi.org/10.1007/s00170-009-2323-1

121. Burke, E., Elliman, D. and Weare, R. (1995). The Automation of the Timetabling Process in Higher Education. Journal of Educational Technology Systems, 23(4), pp.353-362.

122. Ventura, J. and Weng, M. (1995). An improved dynamic programming algorithm for the single-machine mean absolute deviation problem with a restrictive common due date. Operations Research Letters, 17(3), pp.149-152.

123. Годна А.В. Задача мінімізації сумарного відхилення від спільного директивного строку при виконанні завдань паралельними пристроями / А. В. Годна, О. Г. Жданова, А. О. Маленко, М. О. Сперкач // Науковий огляд. - 2017. -№9(14). - С. 14–32.

124. Година А.В. Задача мінімізації сумарного відхилення моментів завершення від директивних строків при виконанні завдань паралельними пристроями / А. В. Година, О. Г. Жданова, М. О. Сперкач / Матеріали науково-практичної конференції «Інформатика та обчислювальна техніка-ІОТ –2018». – м. Київ.: НТУУ «КПІ ім. Ігоря Сікорського», 23-24 квітня 2018 р.

125. Година А.В. Про один підхід складання розкладу виконання завдань паралельними пристроями з метою мінімізації сумарного відхилення моментів завершення від директивних строків/ А. В. Година, О. Г. Жданова, М. О. Сперкач / Матеріали 20-ї Міжнародної науково-технічної конференції SAIT 2018. – м. Київ.: НТУУ «КПІ ім. Ігоря Сікорського», 23-24 травня 2018 р.

ДОДАТОК А Графічний матеріал

ПЛАКАТ 1 Блок-схема алгоритму побудови допустимого розкладу

ПЛАКАТ 2 Блок-схема перестановочного алгоритму покращення допустимого розкладу

ПЛАКАТ 3 Схема структурна класів програми для побудови розкладу виконання завдань паралельними пристроями

ПЛАКАТ 4 Експериментальне дослідження ефективності критеріїв допоміжної оптимізаційної задачі (частина 1)

ПЛАКАТ 5 Експериментальне дослідження ефективності критеріїв допоміжної оптимізаційної задачі (частина 2)

ПЛАКАТ 6 Експериментальне дослідження ефективності алгоритму перестановок для покращення допустимого розкладу

ПЛАКАТ 7 Екранні форми програмного застосування