**МІНІСТЕРСТВО ОСВІТИ І НАУКИ, МОЛОДІ ТА СПОРТУ УКРАЇНИ**

**НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ**

**„КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ”**

**Факультет прикладної математики**

**Кафедра програмного забезпечення комп’ютерних систем**

|  |  |
| --- | --- |
| "На правах рукопису"  УДК 004.02 | До захисту допущено  **Завідувач кафедри**  **\_\_\_\_\_\_\_\_\_\_\_\_\_\_І.А. Дичка**  **“\_\_\_” \_\_\_\_\_\_\_\_\_\_\_\_2012 р.** |

**МАГІСТЕРСЬКА ДИСЕРТАЦІЯ**

зі спеціальності 8.080403 “Програмне забезпечення автоматизованих систем”

на тему:

**Табу-пошук для квадратичної задачі про призначення**

|  |  |  |
| --- | --- | --- |
| **Студент групи** КВ-64М | Подольський Сергій Валентинович | \_\_\_\_\_\_\_\_\_\_ |
| **Науковий керівник** | к.т.н., доцентЗорін Ю.М. | \_\_\_\_\_\_\_\_\_\_ |
| **Консультант** з охорони праці | к.т.н., ст. викл.Кружилко О.Є. | \_\_\_\_\_\_\_\_\_\_ |

Київ – 2012

**Національний технічний університет України**

**“Київський політехнічний інститут”**

Факультет прикладної математики

Кафедра програмного забезпечення комп’ютерних систем

Спеціальність 8.080403

“Програмне забезпечення автоматизованих систем”

ЗАТВЕРДЖУЮ

Завідувач кафедри

\_\_\_\_\_\_\_\_\_\_ І.А.Дичка

“\_\_” \_\_\_\_\_\_\_\_\_ 2010 р.

**ЗАВДАННЯ**

**на магістерську дисертацію**

студенту групи КВ-64М Подольському Сергію Валентиновичу

1. **Тема дисертації** ТАБУ-ПОШУК ДЛЯ КВАДРАТИЧНОЇ ЗАДАЧІ ПРО ПРИЗНАЧЕННЯ затверджена наказом по університету № 600-С від “2” березня 2012 р.

2. **Термін подання** студентом завершеної дисертації: “5” травня 2012 р.

3. **Об’єкт дослідження**: процес комбінаторної оптимізації при розв'язанні квадратичної задачі про призначення.

4. **Предмет дослідження**: евристичні методи розв’язання квадратичної задачі про призначення.

5. **Перелік задач, які мають бути вирішені**:

* провести аналіз точних та евристичних методів розв’язання квадратичної задачі про призначення;
* дослідити процеси перестановок пар елементів у векторі-розв’язку квадратичної задачі про призначення;
* розробити дельта-функцію для оновлення значення вартості розв’язку під час перестановок;
* розробити математичне підґрунтя та програмне забезпечення табу-пошуку для квадратичної задачі про призначення;
* обґрунтувати та довести ефективність розроблених методів;
* сформулювати остаточний алгоритм табу-пошуку для квадратичної задачі про призначення;
* навести чисельні результати експериментального порівняння швидкодії та якості розв’язків для розробленого табу-пошуку та відомих евристичних методів;
* дослідити умови праці в приміщенні, де планується впровадження та надати рекомендації щодо усунення шкідливих та небезпечних виробничих факторів.

6. **Перелік обов’язкового ілюстративного матеріалу**:

* ілюстрація відповідності потоків відстаням до та після перестановки;
* опис процесу отримання формули корекції зі складністю ;
* таблиці порівняння середнього часу пошуку глобального оптимуму та середнього значення вартості розв’язку;
* діаграми залежності вартості розв’язку від часу роботи табу-пошуку;
* графіки процесу зміни вартості розв’язку;
* профайлінг табу-пошуку до та після застосування розробленого математичного апарату.

7. **Консультант** з охорони праці – к.т.н., ст.викл. Кружилко О.Є.

8. **Дата видачі завдання**: “15” жовтня 2010 р.

**Науковий керівник** \_\_\_\_\_\_\_\_\_ Ю.М. Зорін

**Завдання прийняв до виконання \_\_\_\_\_\_\_\_\_** С.В. Подольський

**РЕФЕРАТ**

**Актуальність теми**. Квадратична задача про призначення є математичною моделлю багатьох практичних задач. До них належать оптимізація розташування пунктів виробництва, розташування електронних компонентів на печатній платі або мікросхемі, проводка ліній електроживлення, проектування розташування кампусів чи корпусів лікарень, проектування клавіатури друкарської машинки тощо. Крім того, задача комівояжера, актуальність якої залишається незмінною по сей день, є частковим випадком квадратичної задачі про призначення. По сьогодні не існує відомих методів розв’язання квадратичної задачі про призначення за поліноміальний час, у зв’язку з чим використовуються евристичні методи. Відомі метаевристичні алгоритми потребують задання апріорних параметрів та характеризуються різною ефективністю і швидкодією при різних типах вхідних даних задачі, тому розробка оптимізованих методів розв’язання квадратичної задачі про призначення є актуальною і важливою задачею як з наукової, так і з практичної точки зору.

**Об'єктом дослідження** є процес комбінаторної оптимізації при розв'язанні квадратичної задачі про призначення.

**Предметом дослідження** є евристичні методи розв’язання квадратичної задачі про призначення.

**Метою роботи** є розробка нового табу-пошуку для розв’язання квадратичної задачі про призначення, що характеризується вищою швидкодією та якістю розв’язків, ніж відомі методи табу-пошуку, такі як Robust Tabu Search.

**Методи дослідження**. В роботі використовуються методи евристичних алгоритмів, дискретної математики, дослідження операцій, комбінаторної оптимізації.

**Наукова новизна** роботи полягає в наступному:

1. Визначено нову математичну властивість квадратичної задачі про призначення, що відображає взаємозв’язок всіх величин приросту значення цільової функції внаслідок здійснення у векторі-розв’язку перестановок пар елементів із однієї трійки індексів.
2. Вперше виведено формулу обчислення половини нових значень ∆ зі складністю замість на кожній основній ітерації, що в результаті призводить до підвищення швидкодії табу-пошуку близько 25% та може бути використана також в інших евристичних алгоритмах, що виконують повне сканування околу.
3. Розроблено новий алгоритм табу-пошуку, який має до двох разів вищу швидкодію, ніж відомі евристичні алгоритми, за рахунок введення нового домену пошуку та використання розробленого математичного апарату.

**Практична цінність** отриманих в роботі результатів полягає в тому, що запропонований метод дає можливість ефективніше за швидкодією та якістю розв’язків в порівнянні з відомими методами розв’язувати квадратичну задачу про призначення та забезпечує приріст швидкодії табу-пошуку до двох разів. Також розроблене математичне підґрунтя може бути застосоване у відомих евристичних методах та призведе до приросту швидкодії ітерації близько 25%.

**Апробація роботи**. Основні положення і результати роботи були представлені та обговорювались на ІІІ та IV наукових конференціях магістрантів та аспірантів «Прикладна математика та комп’ютинг» ПМК-2011 (Київ, 13-15 квітня 2011 р.) та ПМК-2012 (Київ, 11-13 квітня 2012 р.)

**Структура й обсяг роботи**. Магістерська дисертація складається з вступу, п’яти розділів, висновків та додатків.

У вступі сформульовано задачу, яка вирішується, виконано оцінку сучасного стану проблеми, обґрунтовано актуальність напрямку досліджень, сформульовано мету і задачі досліджень.

У першому розділі розглянуто теоретичні засади формулювання квадратичної задачі про призначення, проаналізована обчислювальна складність задачі; наведені найбільш поширені споріднені задачі до квадратичної задачі про призначення; представлені відомості щодо практичного застосування та актуальності задачі.

У другому розділі проаналізовано основні методи розв’язання квадратичної задачі про призначення; досліджено точні методи розв’язання, які дозволяють отримати оптимальні розв’язки, та їх недоліки, а також відомі евристичні методи, що дозволяють отримати наближені квазіоптимальні розв’язки; розглянуто способи генерації екземплярів задач з відомими оптимальними розв’язками для проведення експериментів.

У третьому розділі сформульовано основні підходи та міркування процесу розробки алгоритму табу-пошуку для квадратичної задачі про призначення; описано процес розробки математичного підґрунтя, виведення основних формул та знаходження нових математичних властивостей квадратичної задачі про призначення; сформульовано остаточний алгоритм табу-пошуку.

У четвертому розділі експериментально доведена практична цінність отриманих результатів досліджень; проведено порівняння ефективності розроблених математичного підґрунтя та алгоритму із відомими евристичними методами.

У п’ятому розділі проведено аналіз праці в робочому середовищі та надано рекомендації щодо усунення шкідливих та небезпечних виробничих факторів відповідно до діючих нормативних документів.

У висновках проаналізовано та узагальнено отримані результати.

У додатках наведено скрипт виведення основної математичної формули та програмну реалізацію розробленого табу-пошуку.

Робота виконана на 105 аркушах, містить 3 додатки та посилання на список використаних літературних джерел з 65 найменувань. У роботі наведено 12 рисунків та 3 таблиці.

**Ключові слова:** табу-пошук, квадратична задача про призначення, задача комівояжера, розташування об’єктів, розміщення, комбінаторна оптимізація, дискретна оптимізація, евристичні методи.

**РЕФЕРАТ**

**Актуальность темы**. Квадратичная задача о назначениях является математическою моделью многих практических задач. Среди них оптимизация размещения пунктов производства, размещение электронных компонентов на печатной плате или микросхеме, проводка линий электропитания, проектирование размещения кампусов или корпусов больниц, проектирование клавиатуры печатной машинки и другие. Кроме того, задача коммивояжёра, актуальность которой остаётся неизменной по сей день, является частным случаем квадратичной задачи о назначениях. До сих пор не существует известных методов решения квадратичной задачи о назначениях за полиномиальное время, в связи с чем используются эвристические методы. Известные метаэвристические алгоритмы требуют задания априорных параметров и характеризируются различной эффективностью и быстродействием для разных типов входных данных задачи, поэтому разработка оптимизированных методов решения квадратичной задачи о назначениях является актуальной и важной задачей как с научной, так и с практической точки зрения.

**Объектом исследования** является процесс комбинаторной оптимизации при решении квадратичной задачи о назначениях.

**Предметом исследования** являются эвристические методы решения квадратичной задачи о назначениях.

**Целью работы** является разработка нового табу-поиска для решения квадратичной задачи о назначениях, который характеризируется более высоким быстродействием и качеством решений, чем известные методы табу-поиска, в частности, Robust Tabu Search.

**Методы исследования**. В работе используются методы эвристических алгоритмов, дискретной математики, исследования операций, комбинаторной оптимизации.

**Научная новизна** работы состоит в следующем:

1. Определено новое математическое свойство квадратичной задачи о назначениях, которое отображает взаимосвязь всех величин прироста значения целевой функции в результате осуществления над вектором-решением перестановок пар элементов из одной тройки индексов.
2. Впервые выведено формулу вычисления новых значений ∆ со сложностью вместо на каждой основной итерации, которая в конечном итоге приводит к повышению быстродействия около 25% и может быть использована также в других эвристических алгоритмах, которые выполняют полное сканирование окрестности.
3. Разработан новый алгоритм табу-поиска, который обладает до двух раз большей производительностью, чем известные эвристические алгоритмы, за счет введения нового домена поиска и использования разработанного математического аппарата.

**Практическая ценность** полученных в работе результатов заключается в том, что предложенный метод позволяет эффективнее по сравнению с известными методами решать квадратичную задачу о назначении и обеспечивает прирост быстродействия до двух раз. Также разработанное математическое основание применимо к известным эвристическим методам и приведет к приросту быстродействия около 25%.

**Апробация работы**. Основные положения и результаты работы были представлены и обсуждались на III и IV научных конференциях магистрантов и аспирантов «Прикладная математика и компьютинг» ПМК-2011 (Киев, 13-15 апреля 2011 г.) и ПМК-2012 (Киев, 11-13 апреля 2012).

**Структура и объем работы**. Магистерская диссертация состоит из введения, пяти глав, заключения и приложений.

Во введении сформулирована задача, которая решается, выполнена оценка современного состояния проблемы, обоснована актуальность направления исследований, сформулированы цель и задачи исследований.

В первом разделе рассмотрены теоретические основы формулировки квадратичной задачи о назначениях, проанализирована вычислительная сложность задачи; приведены наиболее распространенные родственные задачи к квадратичной задачи о назначениях; представлены сведения относительно практического применения и актуальности задачи.

Во втором разделе проанализированы основные методы решения квадратичной задачи о назначениях; исследованы точные методы решения, которые позволяют получить оптимальные решения, и их недостатки, а также известные эвристические методы, позволяющие получить приближенные квазиоптимальные решения; рассмотрены способы генерации экземпляров задач с известными оптимальными решениями для проведения экспериментов.

В третьем разделе сформулированы основные подходы и рассуждения касательно процесса разработки алгоритма табу-поиска для квадратичной задачи о назначениях, описаны процесс разработки математического аппарата, вывод основных формул и нахождение новых математических свойств квадратичной задачи о назначениях; сформулирован окончательный алгоритм табу-поиска.

В четвертом разделе экспериментально доказана практическая ценность полученных результатов исследований; проведено сравнение эффективности разработанного математического обеспечения и алгоритма с известными эвристическими методами.

В пятом разделе проведен анализ условий труда в рабочей среде и даны рекомендации по устранению вредных и опасных производственных факторов согласно действующих нормативных документов.

В выводах проанализированы и обобщены полученные результаты работы.

В приложениях приведены скрипт вывода основной математической формулы и программной реализации разработанного табу-поиска.

Работа выполнена на 105 листах, содержит 3 приложения и ссылки на список использованных литературных источников из 65 наименований. В работе приведены 12 рисунков и 3 таблицы.

**Ключевые слова**: табу-поиск, квадратичная задача о назначениях, задача коммивояжёра, расположение объектов, размещение, комбинаторная оптимизация, дискретная оптимизация, эвристические методы.

**ABSTRACT**

**Topic importance**. Quadratic assignment problem is a mathematical model for a variety of application problems. Among them are plants and factories location problem, the problem of placement of interconnected electronic components onto a printed circuit board or on a microchip, wiring problem, campus planning and hospital layout problems, design of control panels and typewriter keyboards and others. Furthermore, the travelling salesman problem may be seen as a special case of the quadratic assignment problem which is still urgent nowadays. There is no known algorithm for solving this problem in polynomial time, hence the heuristic methods are used for the global search of exact or approximate solutions to this and other NP-hard optimization problems. Known metaheuristics require adjusting several apriori parameters and can be characterized with ambiguous performance and speed on different input problem instances; therefore efficient quadratic assignment problem optimization methods development is urgent and significant scientific task as well as applied.

**Object of research** is the process of combinatorial optimization applied to the quadratic assignment problem optimization.

**Subject of research** is the heuristic methods for the quadratic assignment problem optimization.

**Research objective** is a new tabu searchdevelopment for the quadratic assignment problem optimization that can lead to a higher performance rates or quality of the solutions, than other known methods, such as Robust Tabu Search.

**Research methods.** Methods of heuristics, discrete mathematics, and operations research along with combinatorial optimization methods are used in the research.

**Scientific novelty** includes as follows:

1. A new mathematical feature of the quadratic assignment problem has been obtained which reveals a corellation between all the deltas of the objective function value change caused by an exchange of two facilities from the same three locations.
2. A new formula has been introduced that allows evaluating a half of delta values with time complexity instead of on each main iteration. This leads as a result to tabu search performance increase of about 25%. The obtained formula could also be used successfully in other heuristics that use the whole neighborhood scanning.
3. A new tabu search algorithm has been developed which is up to two times faster than well-known heuristic algorithms. An improvement is based on a new domain involvement as well as on the developed mathematical background.

**Application value** of the results obtained in the work relies on the fact that the proposed method allows to solve the quadratic assignment problems more efficiently as compared to known methods and provides a speed increase of up to two times. Also the developed mathematical basis appears to be applicable to the well-known heuristic methods and will lead to a speed increase of about 25%.

**Approbation.** The basic points and outcomes of the research have been presented and discussed at the 3rd and 4th scientific conferences for master students and graduates «Applied mathematics and computing» PMK-2011 (Kyiv, April 13-15, 2011) and PMK-2012 (Kyiv, April 11-12, 2012).

**Structure and content of the thesis.** The master thesis consists of the introduction, five chapters, conclusions and appendices.

The introduction presents the general description of the research problem, gives an overview on a current state of the scientific problem, explains the research importance and formulates the objective and tasks of the research.

In the first chapter the theoretical backgrounds of the quadratic assignment problem formulations are described; the problem computational complexity is analyzed; the most prevalent related problems to the quadratic assignment problem are given; the statements about the quadratic assignment problem applications and importance are presented.

In the second chapter the major solution methods for the quadratic assignment problem are analyzed; the exact methods which allow obtaining optimal solutions, their pros and cons are investigated as well as heuristic methods which allow obtaining approximate quasi-optimal solutions; generators of the quadratic assignment problem instances with known optimal solution are reviewed in order for the various methods performance comparison.

The third chapter formulates the basic approaches and considerations regarding the development process of the tabu search algorithm for the quadratic assignment problem and describes the process of the mathematical background development. The output of basic formulas and the discovery of a new mathematical feature of the quadratic assignment problem also obtained along with the ultimate tabu search algorithm formulated.

The fourth chapter experimentally proves the application value of the research results obtained via performance comparison of the developed algorithms and software with known heuristic methods.

In the fifth chapter the labor conditions in the working environment are analyzed and the recommendations in order to eliminate the harmful and dangerous production influence under the existing regulations are provided.

In the conclusions the general achievements on the presented thesis are summarized.

In the appendices the major formula inference script and the software implementation for the developed tabu search are provided.

The thesis is presented in 105 pages; it contains 3 appendices and 65 references to the used information sources. 12 figures and 3 tables are given in the thesis.

**Keywords**: tabu search, quadratic assignment problem, travelling salesman problem, facilities location, arranging objects, combinatorial optimization, discrete optimization, heuristic methods, metaheuristics.

ЗМІСТ

[Список термінів, скорочень та позначень 3](#_Toc324451790)

[ВСТУП 4](#_Toc324451791)

[1. КВАДРАТИЧНА ЗАДАЧА ПРО ПРИЗНАЧЕННЯ 6](#_Toc324451792)

[1.1 Формулювання квадратичної задачі про призначення 6](#_Toc324451793)

[1.2 Обчислювальна складність 11](#_Toc324451798)

[1.3 Споріднені задачі 13](#_Toc324451799)

[1.4 Актуальність задачі та практичне застосування 23](#_Toc324451804)

[1.5 Висновки 25](#_Toc324451808)

[2. АНАЛІЗ ВІДОМИХ МЕТОДІВ РОЗВ’ЯЗАННЯ 27](#_Toc324451809)

[2.1 Точні методи розв’язання 27](#_Toc324451810)

[2.2 Евристичні методи 31](#_Toc324451814)

[2.3 Квадратичні задачі про призначення з відомими оптимальними розв’язками 42](#_Toc324451822)

[2.4 Висновки 44](#_Toc324451823)

[3. РОЗРОБКА ТАБУ-ПОШУКУ ДЛЯ КВАДРАТИЧНОЇ ЗАДАЧІ ПРО ПРИЗНАЧЕННЯ 45](#_Toc324451824)

[3.1 Загальні положення 45](#_Toc324451825)

[3.2 Математичне підґрунтя 55](#_Toc324451826)

[3.3 Формулювання алгоритму табу-пошуку для квадратичної задачі про призначення 63](#_Toc324451827)

[3.4 Висновок 68](#_Toc324451828)

[4. ПОРІВНЯННЯ ЕФЕКТИВНОСТІ ТА АНАЛІЗ РЕЗУЛЬТАТІВ 69](#_Toc324451829)

[4.1 Порівняння ресурсоємкості 70](#_Toc324451830)

[4.2 Порівняння швидкодії 74](#_Toc324451831)

[4.3 Порівняння якості розв’язків 77](#_Toc324451832)

[4.4 Висновок 83](#_Toc324451833)

[5. ОХОРОНА ПРАЦІ ТА БЕЗПЕКА У НАДЗВИЧАЙНИХ СИТУАЦІЯХ 85](#_Toc324451834)

[5.1 Вступ 85](#_Toc324451835)

[5.2 Опис приміщення 85](#_Toc324451836)

[5.3 Аналіз умов праці 86](#_Toc324451837)

[5.4 Аналіз пожежної безпеки приміщення 92](#_Toc324451842)

[5.5 Ергономіка робочого місця 93](#_Toc324451843)

[5.6 Висновки 94](#_Toc324451844)

[ВИСНОВКИ 95](#_Toc324451845)

[СПИСОК ВИКОРИСТАНИХ ЛІТЕРАТУРНИХ ДЖЕРЕЛ 97](#_Toc324451846)

[ДОДАТКИ 104](#_Toc324451847)

[Додаток А. Скрипт MATLAB спрощення формули (16)](#_Toc324451848)

[Додаток Б. Програмна реалізація табу-пошуку](#_Toc324451849)

[Додаток В. Копії графічних матеріалів](#_Toc324451850)

# Список термінів, скорочень та позначень

**Profiling (профілювання)** – заміри характеристик роботи програми, таких як час виконання окремих фрагментів та підпрограм

**QAP** (Quadratic Assignment Problem) – квадратична задача про призначення

**QAP Instance** – екземпляр квадратичної задачі про призначення

**Апроксимація розв’язку** – отримання розв’язку, наближеного до оптимального із заданим співвідношенням вартості

**Вартість розв’язку** – значення цільової функції для розв’язку задачі

**Диверсифікація** – стратегія розширення області пошуку шляхом введення нових комбінацій атрибутів розв’язку

**Інтенсифікація** – стратегія стимулювання пошуку серед комбінацій атрибутів найякісніших отриманих раніше розв’язків

**Квазіоптимальний розв’язок** – розв’язок, близький до оптимального, який може бути використаний у якості оптимального в практичних цілях

**Кліка** – повний підграф неорієнтованого графа

**Лінеаризація** – метод заміни аналізу нелінійних систем аналізом лінійних, в деякій мірі еквівалентних вихідним

**Окіл (домен) пошуку** – множина сусідніх розв’язків

**Релаксація** – техніка перетворення жорстких обмежень у менш жорсткі з метою апроксимації складної задачі більш простою

**Сусідній розв’язок** – розв’язок, що відрізняється від поточного значенням лише одного з атрибутів

**Табу (Tabu, Taboo)** – заборона деякої дії

**Якість розв’язку** – значення, обернене до вартості розв’язку

# ВСТУП

Квадратична задача про призначення була вперше сформульована Купменсом та Бекманом у 1957 році як математична модель для розміщення множини окремих господарських об’єктів. Розглянемо задачу розміщення множини об’єктів у множину місцеположень, де вартість визначається як функція від відстаней та потоків між об’єктами, а також деякої вартості розміщення [[1](#113)]. Метою є призначення кожного об’єкта місцеположенню таким чином, щоб мінімізувати вартість. Якщо конкретніше, то дано три вхідні матриці *n × n* дійсних чисел *F = (fij)*, *D = (dkl)* та *B = (bik)*, де *fij* – потік між об’єктом *i* та *j*, *dkl* – відстань між місцеположенням *k* та місцеположенням *l*, а *bik* – вартість розміщення об’єкта *i* у місцеположенні *k*. Варіант Купменса-Бекмана квадратичної задачі про призначення може бути сформульовано таким чином: нехай n – це число об’єктів та місцеположень, а *N* визначається як множина *N = {1,2,…,n}*. Необхідно мінімізувати

де *Sn* – множина всіх перестановок *φ : N → N*. Кожний окремий добуток – це вартість призначення об’єкта *i* місцеположенню і об’єкта *j* місцеположенню . У контексті розміщення об’єктів у місцеположеннях матриці *F* та *D* симетричні та мають нулі на основних діагоналях, усі елементи кожної матриці невід’ємні. Конкретна квадратична задача про призначення з вхідними матрицями *F*, *D* та *B* буде далі позначатися як *QAP(F, D, B)* або як *QAP(F, D)*, якщо немає лінійної складової, тобто якщо *B = 0*.

Більш загальний варіант квадратичної задачі про призначення було введено Ловлером [[2](#118)]. А саме, замість двох матриць *F* та *D* дано чотиривимірний масив коефіцієнтів*.* Задача може бути сформульована як

|  |  |
| --- | --- |
|  | (1) |

Очевидно, що задача *QAP(F, D, B)* Купменса-Бекмана може бути сформульована як задача Ловлера шляхом присвоєння для всіх *i*, *j*, *k*, *l*, де *i ≠ j* або *k ≠ l*, або в противному випадку.

Незважаючи на всеохоплюючі дослідження, які здійснювались більше, ніж протягом трьох десятиліть, квадратична задача про призначення, на відміну від лінійної задачі про призначення (LAP), залишається однією з найскладніших задач комбінаторної оптимізації. Жоден точний алгоритм не може розв’язати задачі розмірності *n > 20* за прийнятний обчислювальний час.

Насправді було показано [[3](#164)], що квадратична задача про призначення NP-складна і навіть знаходження наближеного розв’язку як добутку оптимального розв’язку на деякий постійний множник не може бути здійснено за поліноміальний час, допоки .

Ці результати стримують навіть розв’язання задачі Купменса-Бекмана з матрицями, що задовольняють нерівність трикутника. До сих пір лише для дуже часткових випадків задачі Купменса-Бекмана, а саме для щільної задачі лінійного розташування (*dense linear arrangement problem*), була розроблена схема наближення до поліноміального часу [[4](#7)].

Метою даної роботи є розробка ефективнішого за існуючі метаевристичного алгоритму для розв’язання квадратичної задачі про призначення, де під ефективністю розуміється швидкодія та/або якість розв’язків, які отримуються внаслідок розв’язання задачі.

Вибір теми ґрунтується на тому, що наразі безперспективними є вдосконалення або модифікація точних методів, оскільки задача є  
NP-складною, і, за наявних на сьогоднішній день обчислювальних ресурсів, не представляється можливим знаходження розв’язків задач великої розмірності. Евристичні алгоритми в цьому випадку є єдиним і найбільш перспективним напрямком застосування в практичних цілях.

# КВАДРАТИЧНА ЗАДАЧА ПРО ПРИЗНАЧЕННЯ

## Формулювання квадратичної задачі про призначення

Для багатьох задач комбінаторної оптимізації існують різні, але еквівалентні математичні формулювання, що підкреслюють різні структурні характеристики задачі, які можуть привести до різних підходів розв’язання. Почнемо з того факту, що кожна перестановка *φ* множини *N = {1, 2, …, n}* може бути представлена у вигляді матриці *X = (xij)* розміру *n × n*, такої, що

Матриця X називається матрицею перестановок і характеризується такими обмеженнями на призначення:

Множина всіх матриць перестановок позначається як *Xn*. Згідно з теоремою Біркгофа матриці перестановок однозначно відповідають вершинам політопа призначення (*Birkhoff polytope, the perfect matching polytope of* *Kn,n*). Це призводить до опису квадратичної задачі про призначення у вигляді квадратичного цілочисельного програмування.

### Формулювання у вигляді квадратичного цілочисельного програмування

Використовуючи матриці перестановок замість самих перестановок, квадратична задача про призначення (1) може бути сформульована у вигляді наступної задачі цілочисельного програмування з квадратичною цільовою функцією

|  |  |
| --- | --- |
|  | (2) |
|  | (3) |
|  | (4) |
|  | (5) |

Відтепер, кожного разу, коли зазначається , мається на увазі, що задовольняє обмеженням призначення (3), (4) та (5). Багато авторів запропонували методи лінеаризації квадратичної форми цільової функції (2) шляхом введення додаткових змінних.

Квадратична задача про призначення у формі Купменса-Бекмана може бути сформульована у більш компактному вигляді, якщо ми визначимо скалярний (внутрішній) добуток між матрицями. Нехай скалярний (внутрішній) добуток двох матриць *A*, *B* дійсних чисел розміром *n × n* визначається як

Якщо дано деяку матрицю розміром , перестановку та відповідну їй матрицю перестановок , то та переставляють стовпці та рядки A відповідно до перестановки φ і тоді

Таким чином, можна сформулювати квадратичну задачу про призначення Купменса-Бекмана в альтернативній формі

|  |  |
| --- | --- |
|  | (6) |

### Увігнуте квадратичне формулювання

Нехай у цільовій функції (2) коефіцієнти є такими елементами матриці *S* розміром *n2 × n2*, що розташовано у рядку і стопці . Тепер нехай , де – одинична матриця розміром , а більше, ніж максимальна норма матриці *S*. Віднімання константи від елементів основної діагоналі *S* не змінює оптимальні розв’язки відповідної квадратичної задачі про призначення, а просто додає константу до цільової функції. Отже, квадратичну задачу про призначення можна розглядати з масивом коефіцієнтів *Q* замість *S*. Нехай

Тоді можемо переписати цільову функцію квадратичної задачі про призначення з масивом коефіцієнтів *Q* як квадратичну форму , де

Оскільки , то ми можемо припустити, що матриця *Q* є симетричною від’ємно визначеною неермітовою матрицею. Внаслідок цього ми отримаємо увігнуту задачу мінімізації і можемо сформулювати квадратичну задачу про призначення як

|  |  |
| --- | --- |
|  | (7) |

У [[5](#16)] було введено формулювання (7) і використано для отримання процедур зрізу площин. І хоча розроблені точні методи були неефективні з обчислювальної точки зору, евристики, побудовані на цих процедурах, призвели до субоптимальних розв’язків високої якості.

При додаванні складової до матриці *Q* замість її віднімання ми можемо завжди вважати, що цільова функція квадратичної задачі про призначення є увігнутою. Це призводить до формулювання квадратичної задачі про призначення як квадратичної увігнутої задачі мінімізації.

### Формулювання через слід матриці

Слід матриці *B* розміру *n × n* визначається як сума всіх елементів її основної діагоналі:

Нехай дано квадратичну задачу при призначення Купменса-Бекмана з вхідними матрицями *F*, *D* та *B*. Якщо , то

припускаючи, що ,, де є матрицю перестановок, пов’язаною з *X*. Оскільки , квадратична задача про призначення (6) може бути сформульована як

|  |  |
| --- | --- |
|  | (8) |

Формулювання через слід матриці вперше введено у [[6](#61)] та [[7](#62)] і використано у [[8](#67)] для запровадження техніки знаходження нижньої границі через власні значення для симетричних квадратичних задач про призначення. Для будь-яких двох матриць *A*, *B* дійсних чисел розміром згадаємо загальновідомі властивості , та . Для ми можемо записати квадратичну складову виразу (8) у вигляді

де *D* необов’язково симетрична. Отже, якщо дано квадратичну задачу про призначення, де лише одна з матриць *F* симетрична, то ми можемо привести її до квадратичної задачі про призначення, де обидві матриці симетричні. Це досягається за рахунок введення нової симетричної матриці :

### Добуток Кронекера

Нехай *A* – матриця дійсних чисел розміром , а *B* – матриця дійсних чисел розміром . Тоді добуток Кронекера матриць *A* та *B* визначається як

Тобто є матрицею розміром , сформованою із усіх можливих попарних добутків елементів *A* та *B*. Якщо є вектором, сформованим стовпцями матриці перестановок *X*, то квадратична задача про призначення може бути сформульована як

Операції використання добутку Кронекера і їх властивості були детально вивчені у [[9](#84)]. Проте, описане вище формулювання рідко використовується у дослідженнях квадратичної задачі про призначення. Виходячи з даного формулювання, Ловлер [[2](#118)] дав альтернативне визначення квадратичної задачі про призначення як лінійної задачі про призначення розміру *n* з тим додатковим обмеженням, що допустимі лише ті матриці перестановок розміром , які є добутками Кронекера матриць перестановок розмірів .

Якщо, як і раніше, матриця вартостей *C* розміром містить *n4* вартостей , таких, що (*ijkl*)-ий елемент відповідає елементу у (*(i−1)n+k*)-му рядку та (*(j−1)n+l*)-му стовпцю *C*, то квадратична задача про призначення може бути записана у вигляді

Оскільки допустимі розв’язки мають задовольняти додаткове обмеження, результуюча лінійна задача про призначення не може бути розв’язана ефективно.

## Обчислювальна складність

Результати, описані в даному підрозділі, доводять той факт, що квадратична задача про призначення є дуже складною задачею з теоретичної точки зору. Не лише тому, що квадратична задача про призначення не може бути точно розв’язана ефективно, а й тому, що розв’язок навіть не може бути апроксимований ефективно до точного з деяким постійним співвідношенням. Крім того, знаходження локального оптимуму також не є тривіальною задачею навіть для спрощених структурованих околів, таких як окіл обміну місцями двох елементів.

Два результати, отримані ще раніше в [[3](#164)] у 1976 р, довели складність розв’язання та апроксимації розв’язків квадратичної задачі про призначення. Було показано, що квадратична задача про призначення є NP-складною і що знаходження навіть ∆-апроксимації розв’язку є складною задачею у тому сенсі, що існування поліноміального алгоритму ϵ-апроксимації рівносильно до . Нехай надалі *Z(F, D, φ)* означає значення цільової функції розв’язку *φ* квадратичної задачі про призначення з матрицею потоків *F* та матрицею відстаней *D*.

**Визначення.** Дано дійсне число . Тоді кажуть, що алгоритм ϒ для розв’язання квадратичної задачі про призначення є алгоритмом  
ϵ-апроксимації, якщо умова

виконується для кожної квадратичної задачі про призначення *QAP(F, D)*, де – розв’язок *QAP(F, D)*, обчислений алгоритмом ϒ, а πopt – оптимальний розв’язок *QAP(F, D)*. Розв’язок *QAP(F, D)*, отриманий за допомогою алгоритму ϵ-апроксимації, називається ϵ-апроксимованим розв’язком.

**Теорема.** Квадратична задача про призначення є NP-складною в сильному сенсі. Для довільного ϵ > 0 існування алгоритму ϵ-апроксимації квадратичної задачі про призначення за поліноміальний час рівносильне .

Доведення здійснено шляхом зведення від задачі про цикл Гамільтона [[10](#73)]: дано граф *G*; чи містить *G* цикл, який проходить через кожну вершино рівно один раз?

У [[11](#152)] приводиться навіть більш сильний результат, який ще раз підтверджує широко поширену думку властивої складності квадратичної задачі про призначення у порівнянні з іншими складними задачами комбінаторної оптимізації. Добре відомо і легко показати, що задача комівояжера є частковим випадком квадратичної задачі про призначення. Задача комівояжера для *n* міст може бути сформульована як *QAP(F, D)*, де *D* – матриця відстаней задачі комівояжера, а *F* – матриця суміжності циклу Гамільтона для *n* вершин. У випадку, якщо матриця відстаней симетрична і задовольняє нерівність трикутника, задача комівояжера апроксимується за поліноміальний час у межах 3/2, як показано у [[12](#46)]. У [[11](#152)] було показано, що лише у випадку  *QAP(A, B)* апроксимується за поліноміальний час в межах деякого скінченного співвідношення апроксимації, навіть якщо *A* – матриця відстаней деякої множини точок на лінії, а *B* – симетрична блочно-діагональна матриця.

Врешті-решт, зазначимо також, що немає локального критерію для вирішення того, наскільки якісним є локальний оптимум у порівнянні з глобальним оптимумом. З точки зору обчислювальної складності, вирішення, чи є даний локальний оптимум глобальним розв’язком квадратичної задачі про призначення, є складною задачею.

## Споріднені задачі

Однією з можливостей отримати узагальнення квадратичної задачі про призначення є врахування цільових функцій вищого порядку і отримання в такому випадку кубічної, біквадратичної та, у загальному випадку, *N-ичної* задачі про призначення [[2](#118)]. Наприклад, для кубічної задачі про призначення ми маємо *n6* коефіцієнтів вартості , де *i*, *j*, *k*, *l*, *m*, *p* *=* 1, …, *n*, а задача формулюється у вигляді

Як зазначається у [[2](#118)], ми можемо сконструювати матрицю *S* розміром *n3 × n3*, що містить коефіцієнти вартості, такі, що кубічна задача про призначення еквівалентна лінійній задачі про призначення

Аналогічним чином лінійна задача про призначення може бути розширеною до будь-якої *N-ичної* задачі про призначення, враховуючи матрицю розв’язку *Y* як піднесення до *N-го* степеня Кронекера матриці перестановок із *Xn*.

Інша модифікація цільової функції, яка породжує задачу, споріднену до квадратичної задачі про призначення, – квадратична задача про призначення з вузьким місцем (*bottleneck quadratic assignment problem*), яка є підстановкою замість суми операції *max*. Вперше квадратична задача про призначення з вузьким місцем згадується у [[13](#168)] і виникає внаслідок практичного застосування при проводці електроживлення таким чином, щоб мінімізувати максимальні довжини дротів, що використовуються.

У даному підрозділі розглядається декілька узагальнень та задач, споріднених до квадратичної задачі про призначення, для яких були знайдені реальні практичні застосування, що викликало інтерес до їх аналізу та відшукання методик їх розв’язання.

### Квадратична задачі про призначення з вузьким місцем

У квадратичній задачі про призначення з вузьким місцем розміру *n* дано матрицю потоків *F* розміру *n × n* та матриці відстаней *D* розміру *n × n*. Необхідно знайти перестановку , яка мінімізує значення цільової функції

Більш загальна квадратична задача про призначення з вузьким місцем, аналогічна до квадратичної задачі про призначення (1), отримується, якщо коефіцієнти задачі мають вигляд :

Окрім застосування при проводці електроживлення, як зазначено раніше, квадратична задача про призначення з вузьким місцем має багато інших застосувань. В основному всі застосування квадратичної задачі про призначення породжують застосування квадратичної задачі про призначення з вузьким місцем, оскільки часто має сенс мінімізації найбільшої вартості замість загальних витрат, понесених внаслідок деякого рішення. Достатньо вивченою є задача з теорії графів, яка може бути змодельована як квадратична задача про призначення з вузьким місцем, а саме задача пропускної спроможності (*graph bandwidth problem*). Дано неорієнтований граф *G = (V, E)* з множиною вершин *V* та множиною ребер. Необхідно присвоїти вершинам графу *G* мітки *1*,*2*,*…*,*n*, де , таким чином, щоб максимальне абсолютне значення різниці міток вершин, які з’єднані ребром, було мінімізоване. Іншими словами, ми шукаємо таке позначення вершин, щоб максимальна відстань від одиничних елементів результуючої матриці суміжності до діагоналі була мінімізованою, тобто, щоб пропускна спроможність матриці суміжності була мінімізованою. Неважко помітити, що ця задача може бути змодельована як частковий випадок квадратичної задачі про призначення з вузьким місцем з матрицею потоків, рівною матриці суміжності *G* для довільного позначення вершин, та матрицею відстаней .

Квадратична задача про призначення з вузьким місцем є NP-складною, оскільки включає задачу комівояжера з вузьким місцем як частковий випадок (це є аналогічним до факту, що квадратична задача про призначення містить задачу комівояжера як частковий випадок). Деякі алгоритми для розв’язання квадратичної задачі про призначення з вузьким місцем були запропоновані у [[14](#22)]. Ці алгоритми працюють з границями Гілмора-Ловлера для квадратичної задачі про призначення з вузьким місцем, які включають у себе в свою чергу розв’язок лінійної задачі про призначення з вузьким місцем. Алгоритм для квадратичної задачі про призначення з вузьким місцем загального вигляду також включає порогову процедуру для зведення до 0 якомога більшої кількості коефіцієнтів.

У [[15](#30)] досліджено асимптотику квадратичної задачі про призначення з вузьким місцем та доведено результати аналогічно до тих, що отримано для квадратичної задачі про призначення: якщо коефіцієнти є незалежними випадковими змінними з рівномірним розподілом в межах , то відносна різниця між найгіршим та найкращим значенням цільової функції наближається до 0 з ймовірністю, що прямує до 0 з наближенням розмірності задачі до нескінченності.

Квадратична задача про призначення та квадратична задача про призначення з вузьким місцем є частковими випадками більш узагальненої квадратичної задачі про призначення, яка має назву алгебраїчна квадратична задача про призначення (*algebraic QAP*) аналогічно до алгебраїчної лінійної задачі про призначення [[16](#33)]. Якщо (*H*, *∗*, ≺) є повністю впорядкованою комутативною напівгрупою з композицією ∗ та відношенням порядку ≺, то алгебраїчна квадратична задача про призначення з коефіцієнтами вартості формулюється як

Дослідження квадратичної задачі про призначення з вузьким місцем, та у більш загальному випадку алгебраїчної квадратичної задачі про призначення, були відправною точкою для дослідження числа алгебраїчних задач комбінаторної оптимізації з коефіцієнтами, взятими з лінійно впорядкованих напівтруп, тобто лінійних задач про призначення, задач транспортування, задач про потоки та ін.

### Біквадратична задача про призначення

Узагальненням квадратичної задачі про призначення є біквадратична задача про призначення, яка по суті є квадратичною задачею про призначення з коефіцієнтами, сформульованими як добутки двох чотиривимірних масивів. Нехай дано два масиви та  
 розмірами . Біквадратична задача про призначення в такому випадку формулюється як

Основне практичне застосування біквадратичної задачі про призначення виникає при проектуванні надвеликих інтегральних схем (НВІС). В основі кіл НВІС лежать послідовні кола, а процес їх проектування складається з двох кроків: на першому кроці специфікації кіл переводяться в таблицю переходів станів системи, що моделюється за допомогою цифрових автоматів, а на другому кроці виконується пошук кодування станів таким чином, щоб фактична реалізація була мінімального розміру. Еквівалентно біквадратична задача про призначення може бути сформульована як

де означає множину всіх перестановок *N = {1, 2, …, n}*. Усі різні формулювання квадратичної задачі про призначення можуть бути розширені для біквадратичної задачі про призначення.

У [[17](#27)] обчислюються нижні границі для біквадратичної задачі про призначення, отримані з нижніх границь квадратичної задачі про призначення. Результати обчислень показали, що ці границі є слабкими і погіршуються зі збільшенням розмірності задачі. Дане спостереження припускає, що методи гілок та границь будуть ефективними лише для задач малої розмірності. Для задач більшої розмірності необхідні ефективні метаевристики, які знаходять апроксимовані розв’язки високої якості. З цією метою у [[18](#25)] були розроблені евристичні алгоритми – різновиди методу відпалу та табу-пошуку.

Результати обчислень для тестових задач розмірності *n = 32* з відомими оптимальними розв’язками (генератор тестових задач представлений у [[17](#27)]) показали, що одна з версій методу імітації відпалу є найефективнішою з усіх протестованих. Алгоритм GRASP також був застосований до задач з [[19](#132)] і знайшов оптимальні розв’язки для всіх згенерованих тестових задач у [[17](#27)].

### Квадратична задача про напівпризначення

У квадратичній задачі про напівпризначення (*quadratic semi-assignment problem*) також дано дві матриці коефіцієнтів – матрицю потоків та матрицю відстаней , але в даному випадку ми маємо *n* об’єктів та *m* місцеположень, . Необхідно призначити усі об’єкти місцеположенням і, принаймні, один об’єкт кожному місцеположенню таким чином, щоб мінімізувати загальну відстань, помножену на потік матеріалів (сировини, людей), що рухаються між різними об’єктами. Таким чином, цільова функція така ж сама, як і для квадратичної задачі про призначення, але за тією лише відмінністю, що допустимими є розв’язки, які не є відображенням один-до-одного (бієкціями) між множиною об’єктів та місцеположень, а є довільними відображеннями множини об’єктів в множину місцеположень. Таким чином, квадратична задача про напівпризначення в такому випадку формулюється як

Квадратична задача про напівпризначення об’єднує деякі цікаві задачі комбінаторної оптимізації, такі як кластеризація та k-розфарбовування графа. У задачі кластеризації дано *n* об’єктів та матрицю розбіжностей . Необхідно знайти розподіл цих об’єктів у *m* класів таким чином, щоб мінімізувати суму розбіжностей об’єктів, що належать до одного класу. Очевидно, що це квадратична задача про напівпризначення з матрицями коефіцієнтів *F* та *D*, де *D* – одинична матриця розміру *m × m*. У задачі k-розфарбовування графа дано граф із *n* вершин, необхідно перевірити, чи можуть його вершини бути розфарбовані у *k* різних кольорів таким чином, щоб кожні дві вершини, які з’єднані ребром, отримали різні кольори. Ця задача може бути змодельована як квадратична задача про напівпризначення, де *F* рівна матриці суміжності даного графу, а *D* – одинична матриця розміру *k × k*. Задача про k-розфарбовування має відповідь «так» тоді і тільки тоді, якщо описана вище квадратична задача про напівпризначення має оптимальне значення, рівне 0. Практичні застосування квадратичної задачі про напівпризначення також включають розподілені обчислення [[20](#169)] та планування [[21](#45)].

Квадратична задача про напівпризначення була вперше введена у [[22](#85)] і відзначена як NP-складна. У [[23](#134)] запропоновано алгоритм релаксації Лагранжа для цієї задачі та показано, що аналогічно до квадратичної задачі про призначення пошук оптимальних розв’язків навіть для задач малої розмірності є дуже складним. Нижні границі для квадратичної задачі про напівпризначення були запроваджені у [[24](#129)], а у [[25](#128)] були обговорені часткові випадки, придатні до розв’язання за поліноміальний час.

### Інші задачі, які можуть бути сформульовані як квадратичні задачі про призначення

Існує набір інших загальновідомих задач комбінаторної оптимізації, які можуть бути сформульовані як квадратичні задачі про призначення зі специфічними матрицями коефіцієнтів. Звісно, оскільки квадратична задача про призначення не є простою задачею, то немає сенсу використовувати алгоритми, розроблені для квадратичної задачі про призначення, для цих інших задач. Хоча, взаємозв’язок між квадратичною задачею про призначення і цими задачами має бути корисним для кращого розуміння квадратичної задачі про призначення та її успадкованої складності.

#### Розбиття графа та максимальна кліка

Дві достатньо досліджені NP-складні задачі комбінаторної оптимізації, які є частковими випадками квадратичної задачі про призначення, – задача розбиття графа (*graph partitioning problem*) та задача про максимальну кліку (*maximum clique problem*). У задачі про розбиття графа дано зважений граф із *n* вершинами та числом *k*, на яке ділиться *n*. Необхідно розбити множину *V* на *k* підмножин рівних потужностей таким чином, щоб сумарна вага ребер, відрізаних розбиттям, була мінімізованою. Дана задача може бути сформульована як квадратична задача про призначення з матрицею відстаней *D*, рівною матриці зваженої суміжності *G*, а матриця потоків *F* отримується шляхом множення ­ на матрицю суміжності об’єднання *k* повних підграфів, що не перетинаються та мають по вершин.

У задачі про максимальну кліку також дано граф із *n* вершин і необхідно знайти максимальне значення таке, що існує підмножина , яка включає кліку в *G*, тобто всі вершини *V1* з’єднані ребрами графа *G*. В цьому випадку розглядається квадратична задача про призначення з матрицею відстаней *D*, рівною матриці суміжності *G*, та матрицею потоків *F* суміжності графу, що складається з кліки розміру *k* та ізольованих вершин, помноженою на . Кліка розміру *k* існує в *G* тільки тоді, коли оптимальне значення відповідної квадратичної задачі про призначення дорівнює .

#### Задача комівояжера

У задачі комівояжера (*traveling salesman problem*) дано множину міст та попарні відстані між ними. Метою є знаходження подорожі, в якій кожне місто відвідується рівно один раз та має мінімальну довжину. Нехай множина цілих чисел представляє *n* міст і нехай симетрична матриця розміру представляє відстані між містами, де – відстань між містами *i* та *j* (). Задача комівояжера може бути сформульована як

Задача комівояжера може бути сформульована як квадратична задача про призначення з даною матрицею відстаней *D* та матрицею потоків *F*, яка рівна матриці суміжності циклу із *n* вершин. Задача комівояжера є загальновідомою NP-складною задачею комбінаторної оптимізації.

#### Задача лінійного розташування

У задачі лінійного розташування дано граф і необхідно розмістити його вершини у точках на лінії таким чином, щоб мінімізувати суму попарних відстаней між вершинами графа *G*, які з’єднані деяким ребром. Якщо розглянути більш узагальнений варіант зважених графів, то буде отримано задачу проводки електроживлення. Ця задача є NP-складною [[10](#73)]. Вона може бути сформульована як квадратична задача про призначення з матрицею відстаней, рівною матриці суміжності даного графу, а також з матрицею потоків , елементи якої для всіх *i* та *j*.

#### Задача мінімально зваженого графа Гамільтона

У задачі мінімально зваженого графа Гамільтона (*minimum weight feedback arc set problem*) дано зважений орієнтований граф з множиною вершин *V* та множиною дуг *E*. Метою є видалення множини дуг із *E* з мінімальною загальною вагою таким чином, щоб усі орієнтовані цикли у *G* були зруйновані і ациклічний орієнтований підграф залишився. Очевидно, що задача мінімального зваженого графа Гамільтона еквівалентна задачі знаходження ациклічного підграфа з максимальною вагою в графі *G*. Незважений варіант задачі, в якому ваги ребер графу, що лежить в основі, рівні 0 або 1, називається задачею про ациклічний орієнтований підграф (*acyclic subdigraph problem*) і достатньо широко розглядається у [[26](#99)].

Цікавим застосуванням даної задачі є так звана тріангуляція вхідних-вихідних таблиць, які виникають також при вхідному-вихідному аналізі в економіці для прогнозування розвитку промисловості.

Оскільки вершини ациклічного підграфа можуть бути позначені топологічно, тобто так, щоб в кожній дузі мітка її початку була більшою, ніж кінця, то дана задача може бути сформульована як квадратична задача про призначення. Матриця відстаней квадратичної задачі про призначення є зваженою матрицею суміжності графу *G*, а матриця потоків є нижньою трикутною матрицею, тобто , якщо , і в противному випадку.

Задача мінімально зваженого графа Гамільтона добре відома як NP-складна задача [[10](#73)].

#### Задачі упаковки в графах

Інша загальновідома NP-складна задача, яка може бути сформульована як квадратична задача про призначення, – задача упаковки графа (*graph packing problem*) [[27](#18)]. У задачі упаковки графа дано графи , , кожен із *n* вершин та множини ребер *E1* та *E2*. Перестановка π множини цілих чисел називається упаковкою G2 в G1, якщо задовольняє для . Іншими словами, пакування *G2* в *G1* – це вкладення вершин *G2* у вершини *G1* таким чином, щоб жодна пара ребер не збігалася. Задача упаковки графа полягає у відшуканні упаковки *G2* в *G1*, якщо така існує, або доведення, що такої упаковки не існує.

Задача упаковки графа може бути сформульована як квадратична задача про призначення з матрицею відстаней, рівною матриці суміжності *G2*, а також матрицею потоків, рівною матриці суміжності *G1*. Упаковка *G2* в *G1* існує тоді і тільки тоді, якщо оптимальне значення вартості розв’язку квадратичної задачі про призначення дорівнює нулю. У такому випадку оптимальний розв’язок квадратичної задачі про призначення визначає пакування.

## Актуальність задачі та практичне застосування

Перші спогади про застосування квадратичної задачі про призначення датуються 1957 роком, коли Купменс та Бекман [[1](#113)] отримали її як математичну модель призначення множини економічних об’єктів множині місцеположень. Таким чином, квадратична задача про призначення з’явилася вперше у контексті задач про призначення об’єктів, що й надалі залишається одним з основних її практичних застосувань. На сьогоднішній день відоме велике розмаїття інших застосувань квадратичної задачі про призначення, включаючи такі області як планування, задача проводки в електроніці, паралельні та розподілені обчислення, статистичний аналіз даних, проектування клавіатур друкарських машинок, спорт, хімія, археологія, балансування запуску турбін та виробництво комп’ютерів. Нещодавно були запропоновані деякі практичні застосування квадратичної задачі про призначення та споріднених задачі у сфері транспортування.

### Розташування об’єктів

У даному контексті *n* об’єктів мають бути призначені *n* місцеположенням. *A* = (*aij*) є матрицею потоків, тобто *aij* – це кількість матеріалів або сировини, що рухається від об’єкта *i* до об’єкта *j* за одиницю часу, а *B* = (*bij*) – матриця відстаней, тобто *bij* представляє відстань від місцеположення *i* до місцеположення *j*. Вартість одночасного призначення об’єкту π(*i*) місцеположенню *i* та об’єкта π(*j*) місцеположенню *j* становить . Очевидно, що призначення всіх об’єктів усім місцеположенням може бути представлено математично у вигляді перестановки *π* *∈ Sn*. У цій моделі загальна вартість призначення *π* усіх об’єктів місцеположенням дорівнює *Z*(*A*, *B*, *π*). Метою є знаходження такого призначення π об’єктів місцеположенням, щоб загальна вартість *Z*(*A*, *B*, *π*) була мінімальною. Це потребує розв’язання квадратичної задачі про призначення *QAP*(*A*, *B*).

Окрім задач про розташування заводів та фабрик, квадратична задача про призначення знаходить практичне застосування при вирішенні таких задач як проводка електроживлення, розташування електронних компонентів на печатній платі або мікросхемі, виробництво обчислювальної техніки, планування, комунікація процесів, балансування турбін, розташування лікарень, проектування клавіатури друкарської машинки, проектування виробничих ліній та ін.

### Проводка електроживлення

Одне із найбільш ранніх практичних застосувань квадратичної задачі про призначення забезпечує проводку електроживлення. Різні пристрої, такі як вимикачі та дисплеї, повинні бути розміщені на панелі, де вони мають бути з’єднані один з одним дротами. Суть задачі полягає у відшуканні способу розміщення пристроїв, що мінімізує загальну довжину дроту. Нехай *n* – кількість пристроїв, що будуть розміщуватись, а – довжина дроту від позиції *k* до позиції *l*. Матриця потоків визначається як

У цьому випадку розв’язок відповідної квадратичної задачі про призначення мінімізує загальну довжину дроту. Інше практичне застосування у контексті розміщень – планування кампусів. Задачі полягає в плануванні місць забудови *n* будівель кампусу, де – відстань від місця забудови *k* до *l*, а – інтенсивність трафіку між будівлею *i* та будівлею *j*. Метою є мінімізація сумарної відстані ходьби між усіма будівлями.

### Проектування клавіатури

Квадратична задача про призначення може бути застосована для проектування клавіатури друкарської машинки. Задачею є розміщення клавіш у клавіатурі таким чином, щоб мінімізувати час, необхідний для набору деякого тексту. Нехай множина цілих чисел *N = {1, 2, …, n}* позначає множину символів, які необхідно розмістити. Тоді означає частоту появи пари символів *i* та *j*. Елементи матриці описують час, необхідний для натискання клавіші на позиції *l* після натискання клавіші на позиції *k* для всіх клавіш, які мають бути призначені. В такому випадку перестановка *φ ∈ Sn* описує призначення символів клавішам. Оптимальний розв’язок *φ∗* для квадратичної задачі про призначення мінімізує середній час, необхідний для набору тексту.

Схоже практичне застосування, пов’язане з проектуванням ергономіки, – розробка панелей керування з метою мінімізації втоми очей. Існує також розмаїття інших практичних застосувань квадратичної задачі про призначення у різних сферах, таких як розташування будівель лікарні, рангування археологічних даних, розстановка команди в естафеті, планування паралельних виробничих ліній, аналіз хімічних реакцій для органічних сполук тощо.

## Висновки

Квадратична задача про призначення є однією з фундаментальних задач комбінаторної оптимізації у сфері оптимізації чи дослідження операцій в математиці з категорії задач про розташування об’єктів. Застосування квадратичної задачі про призначення включає широкий спектр напрямків у науці, виробництві, маршрутизації і т.д. Окрім початкового формулювання про розташування заводів і фабрик, квадратична задача про призначення є математичною моделлю задачі розташування електронних компонентів на печатній платі або мікросхемі, що є частиною етапу розміщення та маршрутизації автоматизованого проектування в індустрії комп’ютерної електроніки.

Постановка задачі нагадує задачу про призначення, але за тією відмінністю, що цільова функція вартості виражається у термінах квадратичних нерівностей, звідки й походить назва квадратичної задачі про призначення.

Задача є NP-складною, тому не існує алгоритму її розв’язання за поліноміальний час. Навіть задачі малого розміру можуть призводити до тривалого обчислювального часу. Задача комівояжера може розглядатися як частковий випадок квадратичної задачі про призначення, якщо потоки з’єднують об’єкти лише вздовж кола і мають однакове значення (константу). Багато інших задач з низки стандартних задач комбінаторної оптимізації можуть бути записані у даній формі.

# АНАЛІЗ ВІДОМИХ МЕТОДІВ РОЗВ’ЯЗАННЯ

## Точні методи розв’язання

Точний алгоритм для задачі комбінаторної оптимізації забезпечує глобальний оптимальний розв’язок задачі. У даному підрозділі буде коротко розглянуто декілька точних алгоритмів, таких як метод гілок та границь (*branch and bound*), січних площин (*cutting plane*) і гілок та січних (*branch and cut*), які використовуються для розв’язання квадратичної задачі про призначення.

### Метод гілок та границь

Методи гілок та меж були успішно застосовані до багатьох складних задач комбінаторної оптимізації і проявили себе як найбільш ефективні точні методи для розв’язання квадратичної задачі про призначення.

Основні елементи методів гілок та границь – це обмеження, розгалуження та правило відбору. Хоча для квадратичної задачі про призначення було розроблено багато технік обмеження, найбільш ефективні методи гілок та границь для цієї задачі використовують границю Глімора-Ловлера (*Gilmore-Lawler bound*). Справа в тому, що інші границі, які перевершують границю Глімора-Ловлера відносно якості границі, насправді є занадто ресурсоємкими у сенсі обчислювального часу. Хоча останнім часом були здійснені спроби використання інших границь, подібних до Глімора-Ловлера: границя [[28](#90)] була використана у методі гілок та границь [[29](#91)], внаслідок чого були отримані суттєві результати.

Для квадратичної задачі про призначення в основному використовується три види стратегій розгалуження:

1. розгалуження одного призначення [[2](#118)];
2. розгалуження пари призначень [[30](#141)];
3. розгалуження на основі відносного позиціонування [[31](#135)].

Розгалуження одного призначення, яке найбільш ефективне, призначає об’єкт місцеположенню на кожному кроці розгалуження, тобто кожна задача розділяється на підзадачі шляхом закріплення місцеположення одному з об’єктів, які ще не призначені. Декілька правил відбору пари об’єкт-місцеположення для визначення підзадач нового рівня дерева пошуку були запропоновані різними авторами. Відповідне правило зазвичай залежить від техніки обмеження. Якщо використовується границя Глімора-Ловлера, то зазначене вище правило часто формулюється у термінах задачі знижених вартостей останнього призначення, що розв’язується для обмеження підзадачі, яка наразі обмежується.

Методи парних призначень призначають пару об’єктів парі місцеположень на кроці обмеження, в той час, як в методах відносного позиціонування рівні дерева пошуку не відповідають номерам вже призначених місцеположенням об’єктів. У цьому випадку визначаються фіксовані призначення у кожній підзадачі у термінах відстаней між об’єктами, тобто їх відносних позицій. Чисельні результати показують, що методи розгалуження одного призначення перевершують методи парних призначень та методи відносного позиціонування .

У [[32](#163)] розроблено інше правило обмеження, що не належить до жодної з вищеописаних груп, а саме так зване політомічне (*polytomic*) або k-часткове (*k-partite*) правило обмеження. Дерево пошуку, що породжується цим методом, не є бінарним як у інших підходах. У цьому випадку використовується границя Глімора-Ловлера і правило обмеження базується на розв’язку φ останньої розв’язаної задачі лінійного призначення для обчислення нижньої границі поточного вузла дерева розв’язку. Нехай є підмножиною (множиною перестановок ), що складається з таких перестановок π, для яких . Аналогічно є множиною таких перестановок , для яких . Поточний вузол розгалужується на нових вузлів з множинами допустимих розв’язків, отриманих як

.

Інша проблема в реалізації методу гілок та границь пов’язана з так званим правилом відбору, яке визначає вибір підзадачі для галуження, тобто вузла дерева пошуку для розгалуження. Декілька стратегій, починаючи з першого пошуку, незалежного від глибини та ширини задачі, та закінчуючи критерієм, що залежить від конкретної задачі, пов’язаним з максимізацією нижніх границь чи зменшених вартостей, були запропоновані та протестовані різними авторами. Серед протестованих стратегій не виявлено явного лідера.

Кращі результати під час розв’язання задач великих розмірностей були отримані пізніше за допомогою паралельних реалізацій [[33](#19)].

### Методи січних площин

Традиційні методи січних площин для квадратичних задач про призначення були розроблені у [[5](#16)] та ін. Ці методи використовують формулювання змішаного цілочисельного лінійного програмування (ЗЦЛП) для квадратичної задачі про призначення, яке придатне для декомпозиції Бендера (*Benders’ decomposition*). У декомпозиції Бендера формулювання ЗЦЛП розбивається на основну задачу (*master problem*) та підзадачу (*slave problem*), де основна задача містить початкові змінні призначення та обмеження. Для фіксованого призначення основна задача зазвичай має формулювання лінійного програмування з точки зору початкових змінних призначення, а також двоїстих змінних підзадачі, та розв’язується за поліноміальний час для фіксованих значень цих двоїстих змінних. Алгоритми зазвичай працюють таким чином. Спочатку евристичні методи застосовуються для генерації початкових призначень. Далі підзадача розв’язується для фіксованих значень змінних призначення та обчислюються оптимальні значення прямих та двоїстих змінних. Якщо двоїстий розв’язок підзадачі задовольняє усі обмеження основної задачі, то отримано оптимальний розв’язок початкового формулювання ЗЦЛП квадратичної задачі про призначення. В противному випадку принаймні одне з обмежень основної задачі порушується. В цьому випадку основна задача розв’язується з фіксованими значеннями для двоїстих змінних підзадачі і отриманий розв’язок передається як вхідні дані до підзадачі. Описана процедура далі повторюється до тих пір, поки розв’язок підзадачі не буде задовольняти усі обмеження основної задачі.

Очевидно, що розв’язок основної задачі, отриманий шляхом фіксації двоїстих змінних підзадачі деякими допустимими значеннями, є нижньою границею квадратичної задачі про призначення, що розглядається. З іншого боку, значення цільової функції квадратичної задачі про призначення, що відповідає будь-якому допустимому присвоєнню змінних призначення, є верхньою границею. Алгоритм завершується, коли нижня та верхня границі співпадають. В цілому, час, необхідний для сходження нижніх та верхніх границь до спільного значення, дуже великий і тому ці методи можуть розв’язувати до оптимального розв’язку лише квадратичні задачі про призначення дуже малої розмірності. Проте, евристичні методи, успадковані від підходів січних площин, призводять до якісних квазіоптимальних розв’язків на ранніх стадіях пошуку [[5](#16)].

### Многогранні січні площини

Аналогічним чином до традиційних методів січних площин многогранні січні площини (*polyhedral cutting planes*), або методи гілок та січних (*branch and cut algorithms*), використовують релаксацію лінійного програмування (ЛП) або ЗЦЛП для задачі комбінаторної оптимізації, що розв’язується, а саме для квадратичної задачі про призначення в нашому випадку. Крім того, методи многогранних січних площин використовують клас нетривіальних чинних нерівностей, що задають грані, які мають задовольнятися усіма допустимими розв’язками початкової задачі. Якщо розв’язок релаксації є допустимим для початкової задачі, процес завершено. В противному випадку деякі зі згаданих вище чинних нерівностей, можливо, порушуються. У цьому випадку здійснюється «відсікання». Таким чином, одна або більше з порушених нерівностей додаються до ЛП або ЗЦЛП релаксації нашої задачі. Після вирішення процес повторюється. У випадку, якщо жодна з чинних нерівностей не порушується, але деяке цілісне обмеження порушуються, алгоритм виконує крок галуження шляхом фіксації допустимих цілих значень відповідній змінній. Кроки галуження породжують дерево пошуку як і у методах гілок та границь. Кожен вузол цього дерева оброблюється описаним вище чином шляхом виконання «відсікань» та подальшого галуження, якщо необхідно. Очевидно, що такі елементи методів гілок та границь як верхні границі, правила відбору та галуження відіграють аналогічну роль у методах січних площин. Отже, такий підхід поєднує в собі елементи як методів січних площин, так і методів гілок та границь.

Основна перевага методів многогранних січних площин у порівнянні з традиційними методами січних площин полягає у використанні відсікань, які не є чинними для всього політопу допустимих розв’язків. У цьому випадку повне обчислення має бути здійснене з нуля для різних фіксованих значень змінних. Це потребує додаткового обчислювального часу та додаткового об’єму пам’яті. Інший не менш важливий недолік традиційних методів січних площин пов’язаний зі «слабкістю» відсікань, які вони виконують. У порівнянні з відсіканнями, що породжені нерівностями, які описують грані, слабкі відсікання не можуть уникнути повільної збіжності.

## Евристичні методи

Хоча були здійснені значні вдосконалення у розробці точних алгоритмів для квадратичної задачі про призначення, задачі розмірності , як і раніше, практично неможливо розв’язати у зв’язку з дуже великим часом обчислень. Це робить незамінною розробку евристичних алгоритмів як таких, що здатні забезпечити розв’язки високої якості за прийнятний час. Досить багато досліджень було присвячено розробці таких підходів. Розрізняють наступні види евристичних алгоритмів:

* Методи конструювання (*Construction methods*)
* Методи обмеженого переліку (*Limited enumeration methods*)
* Методи поліпшення (*Improvement methods*)
* Табу-пошук (*Tabu search*)
* Метод імітації відпалу (*Simulated annealing* )
* Генетичні алгоритми (*Genetic algorithms*)
* Рандомізовані адаптивні процедури жадібного пошуку

(*Greedy randomized adaptive search procedures, GRASP*)

* Мурашині колонії (*Ant systems*)

Далі розглянемо основні з перерахованих вище евристичних алгоритмів більш детально у наступних підрозділах, а також проаналізуємо їх відомі реалізації та досягнення для квадратичної задачі про призначення.

### Методи конструювання

Методи конструювання були вперше введені у [[34](#77)]. Вони є ітеративними алгоритмами, які зазвичай стартують з порожньої перестановки, яку ітеративно заповнюють частковими перестановками шляхом призначення деякого об’єкту, який ще не був призначений, деякому вільному місцеположенню. Псевдокод роботи такого алгоритму зображено на рис. 1.

В даному випадку є частковими перестановками, а – деяка евристична процедура, що призначає об’єкт деякому місцеположенню та повертає . – це множина пар вже призначених об’єктів місцеположенням. Процедура конструює перестановку шляхом додавання призначення до . Евристака , що використовується в , може бути будь-якою евристикою, яка обирає місцеположення для об’єкта , , жадібно, або ж з застосуванням локального пошуку.

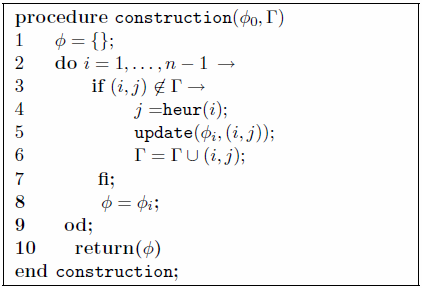


Рис. . Псевдокод для методів конструювання

Одна з найбільш ранніх евристик, використаних на практиці, а саме CRAFT, є методом конструювання [[35](#20)].

### Методи обмеженого переліку

Методи обмеженого переліку ґрунтуються на тому спостереженні, що часто методи переліку (наприклад, методи гілок та границь) знаходять якісні розв’язки на ранніх стадіях пошуку і далі використовують багато часу для незначного поліпшення цього розв’язку або доведення його оптимальності. Така поведінка методів переліку припускає спосіб збереження часу у випадку, якщо ми зацікавлені у якісному, але необов’язково оптимальному розв’язку, а саме накладання деякого ліміту на процес переліку. Цей ліміт може бути лімітом часу або лімітом кількості ітерацій, яку алгоритм має здійснити.

Інша стратегія, що слугує з тією ж метою, полягає в маніпуляціях з нижньою границею. Вона може бути здійснена шляхом збільшення нижньої границі, якщо не було жодного вдосконалення розв’язку протягом значної кількості ітерацій, і дасть глибші відсікання у дереві пошуку для пришвидшення процесу. Очевидно, що такий підхід може відсікти оптимальний розв’язок і, отже, має застосовуватись з обережністю, можливо, разом з деякими евристиками, які використовують ускладнений пошук у просторі допустимих розв’язків.

### Методи поліпшення

Дані методи належать до ширшого класу алгоритмів локального пошуку. Процедура локального пошуку стартує з початкового допустимого розв’язку та ітеративно намагається поліпшити поточний результат. Це здійснюється шляхом підстановки замість поточного кращий допустимий розв’язок з його околу. Цей ітеративний крок повторюється до тих пір, поки не може бути знайдено подальше поліпшення. Методи поліпшення є алгоритмами локального пошуку, які допускають лише вдосконалення поточного розв’язку на кожній ітерації [[36](#2)].

Базовими інгредієнтами методів поліпшення (та локального пошуку взагалі) є *окіл* та *порядок*, в якому сканується окіл. Часто використовуваними околами для квадратичних задач про призначення є окіл парного обміну та окіл потрійного циклічного обміну. У випадку парного обміну окіл даного розв’язку (перестановки) складається з усіх перестановок, які можуть бути отримані з даного шляхом застосування процедури переставляння двох елементів місцями (обміну). У цьому випадку сканування всього околу, тобто розрахунок значень цільової функції для всіх сусідніх розв’язків до даного, займає обчислювального часу. Розмір околу становить і необхідно кроків для обчислення різниці значень цільової функції розв’язку та розв’язку з околу . Якщо окіл вже просканований і є сусіднім до , то тоді окіл розв’язку може бути проскановано за [[37](#71)].

У випадку циклічного потрійного обміну окіл розв’язку складається з усіх перестановок, отриманих з циклічним обміном деяких трьох елементів з деякими різними індексами. Розмір цього околу . Циклічний потрійний обмін насправді не призводить до значного покращення результатів у порівнянні з парним обміном.

Іншим важливим інгредієнтом методів поліпшення є порядок (послідовність), в якому окіл сканується. Цей порядок можу бути або фіксованим заздалегідь, або випадковим. Якщо дано структуру околу та порядок сканування, повинно бути задане правило оновлення поточного розв’язку при переході від поточної ітерації до наступної. Найчастіше використовуються такі правила оновлення:

* перше поліпшення;
* найкраще поліпшення;
* правило Хайдера [[38](#94)].

У випадку першого покращення поточний розв’язок оновлюється, як тільки перший якісніший сусідній розв’язок знайдений. Найкраще поліпшення сканує весь окіл та обирає найякісніший сусідній розв’язок (якщо такий існує взагалі). Правило Хайдера починає сканування околу початкового розв’язку у заданому заздалегідь циклічному порядку. Поточний розв’язок оновлюється, як тільки знайдено якісніший сусідній розв’язок. Сканування околу нового розв’язку стартує з того місяця, де було перервано сканування попереднього у заданому циклічному порядку.

Для того, щоб отримати кращі результати, методи поліпшення та алгоритми локального пошуку загалом застосовуються декілька разів, починаючи з різних початкових розв’язків.

### Табу-пошук

Табу-пошук є локальним пошуком, запровадженим Гловлером [[39](#79)] як техніка подолання локальної оптимальності. Однією з технік подолання локальної оптимальності може бути також дозвіл погіршення поточного розв’язку під час руху від однієї ітерації до наступної у порівнянні з методами поліпшення. У випадку табу-пошуку основна ідея полягає в тому, щоб «запам’ятовувати», які розв’язки були вже відвідані в ході роботи алгоритму, для того, щоб отримати перспективні напрямки для подальшого пошуку. Таким чином, пам’ять, а не лише локальне дослідження околу поточного розв’язку, керує пошуком.

Основними інгредієнтами табу-пошуку є структура околу, рухи (кроки), табу-список та критерій аспірації. Рух (крок) – це операція, яка при застосуванні до деякого розв’язку , генерує окіл . У випадку квадратичної задачі про призначення околом є окіл попарного обміну, а рухами (кроками) зазвичай є перестановки. Табу-список – це список заборонених рухів (табу-рухів), тобто рухів які не є дозволеними для застосування у поточному розв’язку. Статус «табу» рухів змінюється також, а табу-список оновлюється протягом пошуку. Критерій *аспірації* – це одна з умов, задоволення якої табу-рухом скасовує його статус «табу».

Узагальнена процедура табу-пошуку починається з початкового допустимого розв’язку *S* і знаходить розв’язок найвищої якості серед частини околів *S*, отриманих з незаборонених рухів, які не є «табу». Слід відзначити, що цей сусідній розв’язок необов’язково вдосконалює значення цільової функції. Після цього оновлюється поточний розв’язок, тобто він замінюється обраним новим розв’язком. Очевидно, що процедура може мати цикли, тобто може відвідувати деякий розв’язок більше, ніж один раз. У спробі уникнути цього вводиться табу-критерій з метою визначення кроків, які, як очікується, призведуть до циклів. Далі такі рухи оголошуються як «табу» і додаються до табу-списку. Втім, заборона декількох кроків може призвести до заборони відвідування «цікавих» розв’язків. Критерій аспірації розрізняє потенційно цікаві розв’язки серед заборонених. Пошук зупиняється, коли виконується критерій зупинки (ліміт часу пошуку, обмеження кількості ітерацій тощо).

Існує багато свободи в реалізації різних елементів алгоритмів табу-пошуку, таких як табу-список (довжина та обробка), критерій аспірації, критерій табу і т.д. Ефективність алгоритмів табу-пошуку багато в чому залежить від обраної реалізації елементів і немає загальної згоди щодо найкращої реалізації жодного з них.

Для квадратичної задачі про призначення було запропоновано різні реалізації табу-пошуку, такі як табу-пошук з фіксованим табу-списком [[40](#166)], стійкий табу-пошук (*Robust Tabu Search*) [[41](#171)], де довжина табу-списку обирається випадковим чином у межах мінімального та максимального значення, та реактивний табу-пошук [[42](#13)], в якому вводиться механізм адаптації довжини табу-списку. Метою реактивного табу-пошуку є вдосконалення стійкості алгоритму. Алгоритм розпізнає, коли виникає цикл, тобто коли деякий розв’язок відвіданий знову, та збільшує довжину табу-списку відповідно до довжини виявленого циклу.

Пізніше також були запропоновані паралельні реалізації табу-пошуку. Алгоритми табу-пошуку дають можливість природної паралельної реалізації шляхом розподілення об’єму пошуку у околі серед декількох процесорів.

### Метод імітації відпалу

Метод імітації відпалу є методом локального пошуку, який використовує аналогію між задачами комбінаторної оптимізації та задачами статистичної механіки. Цю аналогію було вперше зафіксовано у [[43](#42)] та показано, що алгоритм, описаний у [[44](#133)], який використовується для імітації фізичних систем багатьох часток, може бути застосований як евристика для задачі комівояжера.

Аналогія між комбінаторною оптимізацією та фізичною системою багатьох часток ґрунтується на двох фактах:

* допустимі розв’язки задачі комбінаторної оптимізації відповідають станам фізичної системи;
* значення цільової функції відповідають енергії станів фізичної системи.

У фізиці конденсованих середовищ відпалом називається тепловий процес отримання низьких енергетичних станів тіла в термостаті. Метою є досягнення так званого основного стану, який характеризується мінімальною енергією. У [[45](#37)] було показано, що процес імітації охолодження дає узагальнену евристику, яка може бути застосована до будь-якої задачі комбінаторної оптимізації, як тільки структура околу буде введена до множини допустимих розв’язків.

Усі алгоритми імітації відпалу для квадратичної задачі про призначення використовують окіл попарного обміну. Вони відрізняються тим, яким шляхом процес охолодження чи термальна рівновага реалізовані. Чисельні результати показують, що продуктивність алгоритмів імітації відпалу сильно залежать від значень керуючих параметрів, а також особливо від вибору планування охолодження.

Імітація відпалу може бути змодельована математично неоднорідними ергодичними ланцюгами маркова і ця модель була використана для ймовірнісного аналізу збіжності алгоритмів імітації відпалу.

За природніх умов у введеній структурі околу і не дуже жорстких обмежень повільності процесу охолодження може бути показано, що імітація відпалу асимптотично сходиться до оптимального розв’язку задачі, яка розглядається.

### Генетичні алгоритми

Так звані генетичні алгоритми є підходом до вирішення задач комбінаторної оптимізації, що ґрунтується на законах живої природи. Основна ідея полягає в адаптації механізмів еволюції, що задіяні у процесах селекції в природі, до задач комбінаторної оптимізації. Генетичний алгоритм для розв’язання задач комбінаторної оптимізації було запропоновано у 1975 році [[46](#95)].

Генетичний алгоритм починає роботу з множини початкових допустимих розв’язків, яка називається *початковою популяцією* і генерується випадковим чином або з використанням деякої евристики. Елементи популяції зазвичай називаються *індивідами*. Алгоритм обирає число пар індивідів-батьків з поточної популяції та використовує так звані правила схрещування (*кросовер*) для отримання деякого допустимого розв’язку-нащадка для кожної пари індивідів. Далі деяка кількість «поганих» розв’язків, тобто розв’язків, що дають великі значення цільової функції, відкидається з поточної популяції. Даний процес повторюється до тих пір, поки не спрацює критерій зупинки, тобто ліміт часу, ліміт кількості ітерації, міра збіжності тощо. В процесі роботи алгоритму до поточної популяції періодично застосовуються *мутації* або *імміграції* для вдосконалення її загальної якості шляхом модифікації деяких індивідів або заміни їх відповідно на кращі. Часто всередині генетичних алгоритмів періодично застосовуються інструменти локальної оптимізації, в результаті чого отримуються так звані *гібридні алгоритми*. Пошук урізноманітнюється засобами так званих *турнірів*. Турнір полягає в застосуванні декількох запусків генетичного алгоритму, починаючи з різних початкових популяцій та закінчуючи їх до того, як вони зійдуться. «Краща» популяція отримується як об’єднання кінцевих популяцій цих різних запусків, а далі новий запуск генетичного алгоритму стартує для цієї популяції.

Ряд авторів запропонували генетичні алгоритми для квадратичної задачі про призначення. Стандартні алгоритми мають труднощі генерації найкращих відомих розв’язків навіть для квадратичних задач про призначення малого або середнього розміру. Гібридні підходи, тобто комбінації технік генетичних алгоритмів, показали кращі результати [[47](#68)]. Пізніше у [[48](#6)] було запропоновано інший гібридний алгоритм – так званий жадібний генетичний алгоритм, який показав дуже якісні результати на квадратичних задачах про призначення великого розміру з бібліотеки QAPLIB [[49](#QAPLIB)].

### Мурашині колонії

Алгоритми мурашиних колоній – розроблені нещодавно метаевристичні алгоритми для задач комбінаторної оптимізації, які імітують поведінку мурашиних колоній при пошуку їжі. Спочатку мурахи шукають їжу у околиці свого мурашнику у випадкових напрямках. Як тільки мураха знаходить джерело їжі, вона бере деяку їжу з цього джерела та приносить її назад до мурашника. Протягом мандрівки назад мураха залишає на землі слід деякої субстанції, що зветься *феромоном*. Слід феромону слугує орієнтиром для подальшого пошуку мурахам у напрямку вже знайденого джерела харчів. Інтенсивність феромону в слідах пропорційна до кількості їжі, знайденої у джерелі. Таким чином, шляхи досягнення джерела їжі, що відвідується часто великою кількістю мурах, будуть відмічені більшою інтенсивністю слідів феромону. При спробі імітації поведінки мурах для отримання алгоритмів комбінаторної оптимізації можуть бути використані такі аналогії:

1. область пошуку мурахами схожа на множину допустимих розв’язків;
2. кількість їжі у джерелах їжі нагадує значення цільової функції;
3. слід феромону нагадує компонент адаптивної пам’яті.

Мурашині колонії були вперше введені Марко Доріго [[50](#51)] та призвели до добрих результатів для загальновідомих задач, таких як задача комівояжера та квадратична задача про призначення [[51](#52)].

У випадку квадратичної задачі про призначення слід феромону, який також є ключовим елементом мурашиних колоній, реалізовано у формі матриці . є мірою бажаності розташування об’єкта у місцеположенні у розв’язку, згенерованому мурахами.

Алгоритм ітеративний і конструює фіксовану кількість розв’язків на кожній ітерації. Число є керуючим параметром алгоритму і описує кількість мурах. На першій ітерації ці розв’язки генеруються випадковим чином, тоді як в подальших ітераціях вони оновлюються, використовуючи матрицю феромонів . Спочатку матриця феромонів є матрицею констант, константи обернено пропорційні до найкращого значення цільової функції, знайденого до цього. Це узгоджується з поведінкою мурах, напрямки пошуку яких спочатку вибираються випадковим чином. Позначимо найкращий відомий на даний момент розв’язок як , а його значення цільової функції через . У подальших ітераціях елементи матриці збільшуються на одне й те ж саме значення, яке обернено пропорційне до . Оновлення розв’язків на кожній ітерації здійснюється спочатку на основі матриці феромонів, а потім шляхом застосування деякого методу поліпшення. У обох випадках оновлення складається з обміну об’єктів пари місцеположень, вибраних таким чином, щоб максимізувати (нормалізовану) суму відповідних елементів матриці феромонів. Далі розв’язок, отриманий після цього оновлення, вдосконалюється шляхом застосування деяких методів поліпшення, тобто першого або найкращого вдосконалення. Як тільки виявлено поліпшення найкращого відомого розв’язку, компонент *інтенсифікації* «змушує» мурах в подальшому використовувати частину простору розв’язку, де було знайдено вдосконалення. Якщо після великої кількості ітерацій не було поліпшення найкращого відомого розв’язку, використовується *диверсифікація*, яка в основному полягає в новому випадковому запуску спочатку.

Чисельні результати, представлені у [[51](#52)], показують, що мурашині колонії є конкурентоспроможними метаевристиками особливо для квадратичних задач про призначення з реального життя, де дуже мало якісних розв’язків зібрано разом. Для згенерованих випадковим чином задач, які мають багато якісних розв’язків, розподілених якимось чином рівномірно у просторі пошуку, мурашині колонії працюють гірше, ніж інші евристики, такі як генетичні алгоритми або табу-пошук.

## Квадратичні задачі про призначення з відомими оптимальними розв’язками

Оскільки квадратична задача про призначення є дуже складною з практичної точки зору, дуже часто евристичні методи є єдиним доцільним підходом для її вирішення, і до сих пір немає жодних гарантій щодо продуктивності для жодного з методів, розроблених для квадратичної задачі про призначення. Однією з можливостей для вимірювання та порівняння продуктивності різних метаевристичних методів є використання квадратичних задач про призначення з відомими оптимальними розв’язками. Метаевристичні методи застосовуються для цих задач і отримані розв’язки порівнюються з заздалегідь відомими оптимальними. Задачі з відомими оптимальними розв’язками в ідеалі повинні мати дві властивості: по-перше, вони мають бути репрезентативними у сенсі їх складності, і, по-друге, вони не мають бути особливо легкими для жодного з метаевристичних методів.

По сьогодні два генератори квадратичних задач про призначення з відомими оптимальними розв’язками були запропоновані у [[52](#144)] та [[53](#122)].

Перший метод для генерації квадратичних задач про призначення з відомими оптимальними розв’язками був запропонований в [[52](#144)] у 1988 р. Вхідні дані даного алгоритму складаються з розмірності задачі, що буде генеруватися, оптимального розв’язку (перестановки) вихідної задачі, двох керуючих параметрів та , де , а також матриці відстаней сітки розміру , де . містить прямолінійні відстані, які також називаються Манхеттенськими відстанями, тобто відстані між двома заданими вузлами , що лежать у рядках , та у стовпцях , відповідно, і обчислюються як . Вихідними даними алгоритму є друга матриця , така, що π є оптимальним розв’язком . Ідея полягає в тому, щоб починати з матриці такої, що є тривіальною задачею з оптимальним розв’язком . Далі перетворюється таким чином, що більше не є тривіальною, але продовжує бути її оптимальним розв’язком.

Алгоритм починається з константної матриці , де . є тривіальною задачею, оскільки усі перестановки призводять до однакового значення цільової функції і, таким чином, оптимального розв’язку. Потім матриця ітеративно перетворюється таким чином, щоб вона вже не була константною матрицею, а оптимальний розв’язок залишився тим же. На останній ітерації алгоритм конструює задачі з оптимальним розв’язком за допомогою задачі з її оптимальним розв’язком шляхом присвоєння . Оптимальне значення цільової функції розв’язку становить .

Інший генератор квадратичних задач про призначення з відомими оптимальними розв’язками було запропоновано у [[53](#122)]. Як і попередній, даний генератор починає з тривіальної задачі та ітеративно перетворює і таким чином, що результуюча квадратична задача про призначення все ще має той самий оптимальний розв’язок, але вже нетривіальний. Перетворення такі, що для всіх нерівність  
 еквівалентна наприкінці кожної ітерації.

Якщо матриці коефіцієнтів розглядаються як зважені матриці суміжності графів, то кожна ітерація перетворює елементи, що відповідають деякому підграфу зі знаками на ребрах (знаковизначений підграф). Застосування даного алгоритму з різними знаковизначеними графами призводить до різних генераторів квадратичних задач про призначення. Було протестовано число генераторів за участю різних знаковизначених підграфів, тобто підграфів, що складаються з єдиного ребра, знаковизначених трикутників та знаковизначених остовних дерев. Цікавим та несподіваним є той факт, що використання більш складних знаковизначених підграфів призводить до генерації більш «легких» квадратичних задач про призначення, ніж тих, які генеруються внаслідок використання підграфів, що складаються з одного ребра. Квадратична задача про призначення у даному контексті вважається «легкою», якщо більшість з евристичних методів, застосованих відносно неї, знаходять розв’язки, близькі до оптимальних, за досить короткий час. Немає жодних даних щодо обчислювальної складності цих квадратичних задач про призначення.

## Висновки

Існує декілька типів точних алгоритмів для знаходження глобальних оптимальних розв’язків квадратичної задачі про призначення. Навіть досі, у зв’язку з величезною складністю квадратичної задачі про призначення, задачі з розмірністю, більшою, ніж , залишаються майже непридатними для розв’язання будь-якими з точних методів. В таких випадках часто використовуються метаевристичні алгоритми для оцінки розв’язків квадратичних задач про призначення. Такі методи є досить актуальними, оскільки, хоча і не завжди забезпечують глобальні оптимуми, проте призводять до якісних результатів у допустимих рамках обчислювального часу. Відкриття нових метаевристичних методів, які забезпечують отримання якісних результатів досить швидко, високо цінується. Метаевристичні алгоритми є потужним інструментом для наближеного розв’язання складних задач комбінаторної оптимізації. Однак якість результатів може суттєво відрізнятися для різних вхідних даних. Розуміння такої поведінки є важливими з теоретичної точки зору, але також має свої практичні застосування, такі як генерація даних протягом обчислювального етапу метаевристичного алгоритму.

# РОЗРОБКА ТАБУ-ПОШУКУ ДЛЯ КВАДРАТИЧНОЇ ЗАДАЧІ ПРО ПРИЗНАЧЕННЯ

## Загальні положення

По сьогодні одним з найбільш ефективних за швидкістю знаходження розв’язків та їх якістю серед усіх відомих метаевристичних алгоритмів показав себе алгоритм Robust Tabu Search Еріка Таілларда [[41](#171)], розроблений вперше у 1991 році та модифікований востаннє у 2006 році. З моменту першої публікації алгоритму Robust Tabu Search сторонніми авторами були проведені дослідження щодо вдосконалення зазначеного алгоритму та розробки його паралельних реалізацій, адаптацій для масивно-паралельних обчислень на графічних процесорах [[54](#43)]. Також були розроблені різноманітні модифікації алгоритму Robust Tabu Search, спрямовані на підвищення швидкодії для знаходження розв’язків квадратичних задач про призначення особливого вигляду, наприклад на розріджених матрицях [[55](#Ane10)] тощо, а також постійно здійснюються спроби вдосконалення швидкодії алгоритму для розв’язання квадратичних задач про призначення загального вигляду. На даний момент у мережі Інтернет у розділі “Software for QAP” порталу “QAPLIB – A Quadratic Assignment Problem Library”, присвяченого квадратичній задачі про призначення, загальнодоступною є авторська реалізація з відкритим вихідним кодом алгоритму Robust Tabu Search мовою C++ [[49](#QAPLIB)], що дозволяє проводити різноманітні дослідження та об’єктивні порівняння швидкодії.

У даній роботі проведений детальний опис процесу розробки ефективнішого за швидкодією (або якістю знайдених розв’язків за аналогічний час) табу-пошуку, ніж Robust Tabu Search [[56](#ПМК)]. Проте, перед тим, як перейти до загальних міркувань та обґрунтування евристичних припущень, на яких базується процес розробки ефективнішого нового табу-пошуку, доцільно провести детальний аналіз алгоритму Robust Tabu Search з метою пояснення принципу роботи використаних в ньому прийомів, які увійшли чи не увійшли до розробленого табу-пошуку, основних недоліків алгоритму Robust Tabu Search та потенціалу для вдосконалень, який з’являється завдяки новим властивостям квадратичної задачі про призначення, виявленим в даній роботі.

Розв’язком квадратичної задачі про призначення є призначення кожному місцеположенню одного об’єкта. Якщо ідентифікувати всі місцеположення та всі об’єкти за номером, то розв’язок можна представити у формі вектора невід’ємних цілих чисел, кожен елемент якого містить номер об’єкта, що призначений місцеположенню, номер якого у свою чергу відповідає індексу даного елемента у векторі. Кожний такий вектор є допустимим кандидатом на розв’язок квадратичної задачі про призначення. Вектор-розв’язок є допустимим у тому випадку, якщо всі його елементи різні і відповідають номерам всіх наявних об’єктів.

Перед початком основної ітерації алгоритму Robust Tabu Search конструюється вектор-розв’язок задачі як випадкова перестановка всіх елементів, а також обчислюється його вартість (значення цільової функції). Після цього для кожної можливої перестановки двох елементів з індексами *i* та *j* у сконструйованому векторі обчислюється приріст значення вартості , до якого призведе перестановка. Дане значення приросту може бути як додатним, так і від’ємним числом: якщо , то перестановка призводить до отримання якіснішого розв’язку; якщо , то якість розв’язку погіршується; якщо , то якість не змінюється.

Далі на кожній ітерації алгоритму здійснюється модифікація розв’язку: особливим чином обираються два елементи вектора і переставляються місцями, при цьому модифікується значення вартості розв’язку. Якщо на деякій ітерації отримується значення вартості менше, ніж найменше відоме, то воно запам’ятовується разом з відповідним вектором, отриманим внаслідок перестановки, як найкращий (найякісніший) відомий розв’язок. Далі розглянемо більш детально правила вибору на кожній ітерації пари елементів, над якими здійснюється перестановка.

Основною складовою алгоритму Robust Tabu Search, яка власне перетворює його зі звичайного стохастичного чи жадібного (greedy) пошуку в табу-пошук, є табу-список. В алгоритмі Robust Tabu Search у якості домену (околу) пошуку слугує призначення місцеположенню об’єкта. Таким чином, табу-список в алгоритмі Robust Tabu Search містить інформацію щодо заборони призначення кожного об’єкта кожному місцеположенню. В авторській програмній реалізації [[41](#171)] Robust Tabu Search описаний табу-список реалізований у вигляді матриці розміру , кожний елемент якої є номером основної ітерації, після якої стає дозволеним призначення *i*–му місцеположенню *j*­–го об’єкта, де *i* – номер рядка, *j*­ – номер стовпця матриці. Очевидно, що розмір такої структури даних в пам’яті становить цілих чисел. Критерієм визначення того, чи є призначення забороненим, слугує порівняння номера поточної основної ітерації з номером, що міститься у відповідній клітинці табу-матриці призначень: якщо номер поточної ітерації більший, то призначення дозволене.

У розробленому в даній роботі табу-пошуку використано інший домен (окіл), в якому ведеться табу-список. Замість того, щоб забороняти призначення місцеположенню об’єкта, більш природнім для задачі, що розв’язується, є заборона здійснення перестановки двох елементів з індексами *i* та *j* у векторі-розв’язку. Оскільки атомарною операцією над вектором в табу-пошуку є перестановка пари елементів, то доцільнішим є забороняти цю саму операцію, а не призначення. У алгоритмі Robust Tabu Search при перевірці того, чи є забороненою перестановка, перевіряється, чи є забороненим хоча б одне з двох нових призначень, що з’являться після поточної перестановки, замість того, щоб перевіряти на предмет заборони саму перестановку. Такий підхід у алгоритмі Robust Tabu Search звужує потенційний окіл пошуку та потребує збиткових операцій. Крім того, при забороні самих перестановок замість заборони призначень розмір структури даних, що реалізує табу-список, зменшується до , оскільки операція перестановки пари елементів з індексами *i* та *j* у векторі є комутативною відносно індексів *i* та *j*. При реалізації запропонованого евристичного підходу у тому ж табу-пошуку було відзначено зниження середнього часу знаходження глобальних оптимумів до 13%.

Після здійснення будь-якої перестановки в алгоритмі Robust Tabu Search стають забороненими призначення, що виникають внаслідок зворотної перестановки протягом наступних *t* ітерацій, де *t* – псевдовипадкове число у діапазоні від нуля до деякої верхньої границі, яка визначає максимальну довжину табу-списку. Для кожного з двох призначень число *t* генерується окремо. Дана верхня границя є параметром алгоритму і задається апріорі. За замовчанням у авторській програмній реалізації цей параметр дорівнює . Очевидно, що значення даного параметру не має перевищувати , оскільки в противному випадку можлива ситуація, коли всі призначення виявляться забороненими на деякій ітерації. Саме цей псевдовипадковий вибір довжини табу-списку вносить елементи стохастичного пошуку в алгоритм Robust Tabu Search, завдяки чому пошук і стає стійким (robust). Справа в тому, що кожна перестановка пари елементів у векторі на кожній ітерації, як буде показано далі, вибирається з мінімальним значенням приросту вартості розв’язку. При постійній довжині табу-списку можливе виникнення явища періодичності послідовного відбору одних і тим самих перестановок. У разі виникнення подібної ситуації матиме місце зациклення пошуку в одному й тому ж околі і неможливість виходу за його межі, внаслідок чого пошук вже не дасть підвищення якості розв’язку протягом як завгодно тривалого часу. Втім, слід зазначити, що в авторській реалізації не враховано ситуації, коли обидва призначення внаслідок здійснення перестановки забороняються на нуль наступних ітерацій, тобто фактично щойно здійснена перестановка не додається до табу-списку і не забороняється взагалі. Крім того, враховуючи відбір перестановки з мінімальним значенням приросту вартості розв’язку, як буде показано далі, з високою ймовірністю після здійснення такої перестановки одразу буде обрана зворотна перестановка, оскільки приріст вартості розв’язку при зворотній перестановці має те ж саме значення, але з від’ємним знаком. За такого збігу обставин буде змарновано дві ітерації алгоритму, коли спочатку здійснюється перестановка, проте розв’язок одразу ж повертається в попередній стан. У табу-пошуку, що розроблено в даній роботі, цей незначний недолік пропонується усунути шляхом обмеження знизу довжини табу-списку принаймні одиницею. Враховуючи також те, що в розробленому табу-пошуку забороняються перестановки, а не призначення, експериментально було встановлено інше оптимальне значення довжини верхньої границі табу-списку, яке дорівнює розмірності квадратичної задачі про призначення, що розв’язується.

При відборі на кожній основній ітерації пари елементів з індексами *i* та *j* для здійснення перестановки у векторі-розв’язку використовується складний критерій, проте у будь-якому випадку серед допустимих пар індексів *i* та *j* вибирається пара *i* та *j* з мінімальним значенням приросту вартості розв’язку. Такий підхід забезпечує не просто «сліпий» повний перебір усіх можливих модифікацій вектора-розв’язку, а вносить елементи жадібного (greedy) пошуку. Розглянемо далі більш детально критерій вибору наступної перестановки у векторі-розв’язку елементів з індексами *i* та *j* на кожній ітерації. Нехай – значення вартості поточного вектора-розв’язку ; – найменше відоме значення вартості найякіснішого відомого розв’язку . Тоді, якщо серед усіх можливих перестановок пар елементів з індексами *i* та *j* у поточному векторі-розв’язку знайдеться така перестановка *i* та *j*, що , тобто якщо серед усіх можливих перестановок є перестановки, здійснення яких дає новий якісніший розв’язок, ніж найкращий відомий, то серед усіх таких перестановок елементів з індексами *i* та *j* у поточному векторі-розв’язку обирається перестановка з мінімальним значенням . Очевидно, що в даному випадку . Іншими словами, обирається така перестановка, яка дає якомога якісніший розв’язок серед усіх якісніших розв’язків, ніж найякісніший відомий, які можна отримати. У випадку, коли серед усіх можливих перестановок пар елементів у поточному векторі-розв’язку немає жодної такої перестановки, яка могла б покращити найякісніший відомий розв’язок, переходять до другого критерію вибору перестановки.

У алгоритмі Robust Tabu Search другий критерій вибору перестановки, який застосовується, якщо не дав результату описаний вище перший критерій, приймає до розгляду перестановки пар елементів з індексами *i* та *j* у векторі-розв’язку, що призводять до двох нових призначень місцеположенням об’єктів, хоча б одне з яких не було здійснено вже досить давно, тобто протягом великої кількості ітерацій. Кількість ітерацій, протягом якої не здійснювалось призначення, є другим апріорним параметром алгоритму Robust Tabu Search і за замовчанням в авторській реалізації дорівнює . Серед усіх таких перестановок пар елементів, якщо такі перестановки наявні, обирається також перестановка з мінімальним значенням приросту вартості поточного розв’язку. Цей прийом, застосований у алгоритмі Robust Tabu Search, є другою особливістю, яка робить пошук стійким (robust). Описаний підхід також можна розглядати як диверсифікацію, яка виконується над розв’язком, якщо деякі призначення перестають аналізуватися алгоритмом у зв’язку з характерним для них великим значенням приросту вартості розв’язку після однієї перестановки, незважаючи на яке дане призначення може насправді входити в глобальний оптимум. У розробленому в даній роботі табу-пошуку запропоновано більш природній підхід диверсифікації, пов’язаний зі зміною домену (околу) пошуку: замість того, щоб обирати перестановки, що призводять до призначень, які не здійснювались протягом тривалої кількості ітерацій, в розробленому табу-пошуку обираються самі перестановки елементів з індексами *i* та *j* у векторі-розв’язку, які не виконувались протягом тривалої кількості ітерацій. У зв’язку з цим експериментальним шляхом також встановлена інша нова оптимальна кількість ітерацій, рівна , після якої слід виконати диверсифікацію. Якщо ж серед усіх можливих перестановок елементів з індексами *i* та *j* у векторі-розв’язку немає таких, які не виконувались протягом заданої тривалої кількості ітерацій, то перестановка обирається за третім критерієм.

У алгоритмі Robust Tabu Search третій критерій вибору перестановки, який застосовується, якщо не дав результату описаний вище другий критерій, зводиться до вибору на даній ітерації перестановки елементів з індексами *i* та *j* у вектор-розв’язку з мінімальним значенням приросту вартості поточного розв’язку, здійснення якої не призводить до забороненого на даній ітерації призначення місцеположенню об’єкта. У розробленому табу-пошуку, як вже відзначалося раніше, забороняються не призначення, а самі перестановки, тому вибір перестановки елементів з індексами *i* та *j* у вектор-розв’язку з мінімальним значенням приросту ведеться серед незаборонених табу-списком перестановок.

Таким чином, перший критерій, за яким обирається перестановка, має найвищий пріоритет, другий критерій – середній, а третій – найнижчий. Якщо не знайдено жодної перестановки за першим критерієм, виконується пошук за другим. Якщо за другим не знайдено також – перестановка обирається за третім критерієм. В будь-якому випадку серед допустимих перестановок завжди обирається перестановка елементів з індексами *i* та *j* у векторі-розв’язку, яка призводить до мінімального значення приросту вартості розв’язку . При цьому, якщо значення від’ємне, то обирається перестановка, яка якомога більше покращує якість поточного розв’язку; якщо всі допустимі значення невід’ємні, то серед них обирається перестановка, яка якомога менше погіршує якість поточного розв’язку. Таким чином автоматично забезпечується пошук серед локальних оптимумів в околі перестановок пар елементів, а також забезпечується запобігання збіжності процесу пошуку до одного локального оптимуму. Характер зміни вартості розв’язку з кожною ітерацією зображено на рис. 2.



Рис. . Залежність вартості розв’язку від номера ітерації

Після того, як за першим, другим або третім критерієм було обрано перестановку пари лементів з індексами *i* та *j*, вона виконується над поточним вектором-розв’язком, а до значення вартості поточного розв’язку додається значення приросту , тобто . При цьому, як вже було відзначено раніше, здійснена перестановка забороняється на деяку кількість ітерацій у табу-списку. Далі для всіх можливих перестановок пар елементів з індексами *i* та *j* оновлюються значення , на чому зупинимось далі біль детально.

Однією з основних особливостей алгоритму Robust Tabu Search, яка є запорукою його високої швидкодії, є позбавлення від повного обчислення вартості кожного нового вектора-розв’язку на кожній ітерації. Замість того, щоб обчислювати повністю зі складністю вартість кожного чергового потенційного модифікованого розв’язку внаслідок перестановок, дана вартість обчислюється повністю лише один раз перед запуском основної ітерації алгоритму разом з усіма можливими значеннями приросту цієї вартості внаслідок здійснення перестановок пар елементів з індексами *i* та *j*. Далі вартість модифікованого розв’язку внаслідок здійснення перестановки обчислюється як сума зі складністю . Проте після здійснення деякої обраної перестановки пари елементів всі значення стають недійсними і потребують корекції. У алгоритмі Robust Tabu Search корекція усіх значень приросту вартості поточного розв’язку внаслідок здійснення наступної перестановки пари елементів з індексами *i* та *j* після того, як безпосередньо перед нею була здійснена деяка перестановка елементів з індексами *p* та *q*, обчислюється за формулою

у випадку, якщо всі індекси *i*, *j*, *p*, *q* є різними, тобто , а також за формулою

у тому випадку, якщо пари індексів *i*, *j* та *p*, *q* елементів наступної перестановки та здійсненої перед цим перестановки відповідно мають один спільний індекс *k*. Тобто, якщо наступна перестановка елементів з індексами *i* та *j* буде виконуватись над іншою парою елементів, відмінних від тих, які переставлялись безпосередньо перед нею, то всі відповідні значення оновлюються зі складністю . Проте, якщо наступна перестановка стосується одного з елементів, який був переставлений безпосередньо перед цим, то замість корекції значення зі складністю виконується його повне обчислення зі складністю за тією ж самою формулою, за якою вперше обчислювались всі початкові значення на етапі ініціалізації перед початком основних ітерацій алгоритму Robust Tabu Search. Очевидно, що якщо обидва індекси *i*, *j* пари елементів наступної перестановки співпадають з індексами *p*, *q* здійсненої перед цим перестановки, то матиме місце зворотна перестановка і тоді .

Всього кількість усіх можливих різних перестановок пар елементів, один з індексів яких співпадає з одним з індексів пари елементів, які були переставлені місцями безпосередньо перед цим, завжди становить  
. Дана кількість випливає з тих міркувань, що кожен з двох щойно переставлених місцями елементів *i* та *j* може бути далі переставлений з кожним елементом *k* серед інших елементів, як зображено на рис. 3.

…

1

2

Рис. . Ілюстрація перестановки одного з переставлених елементів

Тоді загальна кількість перестановок елементів з індексами *i* та *j*, для яких нові значення обчислюються зі складністю на кожній ітерації алгоритму Robust Tabu Search, дорівнює різниці кількості всіх можливих перестановок і кількості перестановок, значення яких обчислююється повністю зі складністю :

На відміну від алгоритму Robust Tabu Seacrh та інших метаевристичних алгоритмів, у яких значень обчислюються на кожній ітерації зі складністю , у розробленому в даній роботі табу-пошуку лише значення обчислюються повністю зі складністю , в той час, коли інші значення обчислюються зі складністю лише завдяки знайденій в даній роботі новій математичній властивості квадратичної задачі про призначення, детально дослідженої та описаної в наступному підрозділі.

## Математичне підґрунтя

Розглянемо більш детально формулу (9), за якою обчислюється приріст вартості поточного розв’язку внаслідок здійснення в ньому перестановки двох елементів з індексами *i* та *j*.

|  |  |
| --- | --- |
|  | (9) |

Очевидно, що складність обчислення за формулою (9) зумовлена знаком суми, що до неї входить.

Для того, щоб обчислити значення , на яке зміниться вартість розв’язку внаслідок здійснення в ньому перестановки елементів з індексами *i* та *j*, необхідно відняти від вартості добутки відстаней на відповідні потоки, які зумовлені наявними в розв’язку призначеннями місцеположенню *i* об’єкта та місцеположенню *j* об’єкта , а також додати до вартості добутки відстаней на відповідні потоки, які з’являться в модифікованому розв’язку внаслідок призначень місцеположенню *i* об’єкта та місцеположенню *j* об’єкта .

Розглянемо перший доданок формули (9), а саме

Розкривши дужки, отримаємо

Даний вираз означає, що від вартості розв’язку необхідно відняти зумовлені попередніми призначеннями місцеположенню об'єкта та місцеположенню об'єкта добутки:

1. ­– відстані від місцеположення до самого себе, якщо вона не дорівнює нулю, на потік від об’єкта до самого себе, якщо він не дорівнює нулю;
2. ­– відстані від місцеположення до самого себе, якщо вона не дорівнює нулю, на потік від об’єкта до самого себе, якщо він не дорівнює нулю;

а також додати до вартості розв’язку зумовлені новими призначеннями місцеположенню об'єкта та місцеположенню об'єкта добутки:

1. ­– відстані від місцеположення до самого себе, якщо вона не дорівнює нулю, на потік від об’єкта до самого себе, якщо він не дорівнює нулю;
2. ­– відстані від місцеположення до самого себе, якщо вона не дорівнює нулю, на потік від об’єкта до самого себе, якщо він не дорівнює нулю.

Аналогічно розглянемо другий доданок формули (9), а саме

Розкривши дужки, отримаємо

Даний вираз означає, що від вартості розв’язку необхідно відняти зумовлені попередніми призначеннями місцеположенню об'єкта та місцеположенню об'єкта добутки:

1. – відстані від місцеположення до місцеположення на потік від об’єкта до об’єкта ;
2. – відстані від місцеположення до місцеположення на потік від об’єкта до об’єкта ;

а також додати до вартості розв’язку зумовлені новими призначеннями місцеположенню об'єкта та місцеположенню об'єкта добутки:

1. – відстані від місцеположення до місцеположення на потік від об’єкта до об’єкта ;
2. – відстані від місцеположення до місцеположення на потік від об’єкта до об’єкта .

Нарешті, розглянемо третій доданок формули (9) для обчислення , що є сумою, яка і призводить до складності , а саме

Вираз під знаком суми складається з двох доданків. Розглянемо перший доданок . Розкривши в ньому дужки, отримаємо

Даний вираз означає, що від вартості розв’язку необхідно відняти зумовлені попередніми призначеннями місцеположенню об'єкта та місцеположенню об'єкта добутки:

1. – відстані від кожного іншого з решти, відмінного від *i* та *j*, місцеположення до місцеположення *i* на потік від об’єкта до об’єкта ;
2. – відстані від кожного іншого із решти, відмінного від *i* та *j*, місцеположення до місцеположення на потік від об’єкта до об’єкта ;

а також додати до вартості розв’язку зумовлені новими призначеннями місцеположенню об'єкта та місцеположенню об'єкта добутки:

1. – відстані від кожного місцеположення із решти інших місцеположень до місцеположення *i* на потік від об’єкта до об’єкта ;
2. – відстані від кожного місцеположення із решти інших місцеположень до місцеположення на потік від об’єкта до об’єкта .

Другий доданок під знаком суми має зміст, аналогічний до першого доданку, та враховує відстані між всіма тими ж самими місцеположеннями і потоки між всіма тими ж самими об’єктами у протилежних напрямках, якщо вони відрізняються, тобто якщо матриці відстаней та потоків між усіма парами місцеположень та об’єктів не є симетричними відносно основної діагоналі.

Далі, для спрощення, будемо розглядати відстані та потоки лише в одному напрямку, а саме в напрямку до кожного з решти місцеположень та об’єктів . Як було показано вище, складність обчислення досягається за рахунок того, що від вартості необхідно відняти та додати по різні добутки відстаней на потоки.

Розглянемо ускладнений випадок, коли щойно були переставлені місцями елементи та у векторі і необхідно обчислити нові значення та після перестановки пари елементів і та після перестановки пари елементів і відповідно. Припускаємо, що всі індекси місцеположень *i*, *j*, *k* різні. Тоді можемо розглянути добутки відстаней від місцеположень *i*, *j*, *k* до кожного місцеположення із решти місцеположень на відповідні потоки до та після всіх можливих перестановок об’єктів , , .

Розглянемо спочатку відповідність відстаней , ,  потокам , ,  до здійснення будь-яких перестановок (рис. 4). При цьому припускаємо, що відомі значення приростів , ,  вартості розв’язку внаслідок здійснення перестановок пар елементів  та ;  та ;  та  відповідно. До даних відомих значень , ,  входять такі складові, що стосуються решти статичних призначень:

Рис. . Відповідність потоків відстаням до перестановок

Тепер розглянемо відповідність відстаней , ,  потокам , ,  після здійснення перестановки елементів  та  (рис. 5).

Рис. . Відповідність потоків відстаням після перестановки та

Перед нами постає задача обчислити нові значення , , , до яких входять такі складові, що стосуються решти статичних призначень:

Слід відзначити, що саме ці складові та зумовлюють складність обчислення нових значень , .

Після проведення дослідження правих частин складених вище виразів для , , , ,  була здійснена спроба пошуку залежностей між ними, в результаті чого була знайдена нова властивість розглянутих парних перестановок елементів вектора-розв’язку квадратичної задачі про призначення, яка описується формулою

|  |  |
| --- | --- |
|  | (10) |

Цінність відкриття даної залежності полягає в тому, що, використавши її, можна обчислювати лише одне значення або зі складністю , а друге – зі складністю за формулою або відповідно.

Однак, до , , , ,  окрім досліджених вище складових , , , ,  входять також інші складові , , , , , кожна з яких враховує також нерозглянуті відстані та потоки між самими місцеположеннями *i*, *j*, *k* та об’єктами , , :

Кожне з описаних вище значень обчислюється за однаковою схемою і містить у свою чергу такі складові:

1. корекція добутків на відповідні потоки відстаней від обох місцеположень та до місцеположення , де – третє місцеположення серед *i*, *j*, *k*;
2. корекція добутків на відповідні потоки відстаней від місцеположення до обох місцеположень та , тобто у зворотному напрямку, де – третє місцеположення серед *i*, *j*, *k*;
3. корекція добутків на відповідні потоки відстаней від місцеположень та до самих себе, якщо вони не дорівнюють нулю (корекція вартості призначень);
4. корекція добутків на відповідні потоки відстаней між самими місцеположеннями та в обох напрямках.

За цією схемою розпишемо всі значення , , , , , припускаючи, що об’єкти та вже переставлені місцями:

|  |  |
| --- | --- |
|  | (11) |
|  | (12) |
|  | (13) |
|  | (14) |
|  | (15) |

Використавши відомі значення , ,  та виведені вище формули (11), (12), (13), (14), (15) для , , , , , можемо подати формулу (10) для взаємозв’язку шуканих та у вигляді

Тоді формула для обчислення нового значення зі складністю через вже обчислене нове значення матиме вигляд

Якщо обчислювати часткову суму з використанням наведених початкових визначень для кожного , то матимемо досить велику кількість арифметичних операцій, яка може виявитись навіть більшою, ніж для квадратичних задач про призначення невеликої розмірності . Спробуємо зменшити кількість обчислювальних операцій, що спричинені даною частковою сумою. Для цього виконаємо спрощення формули

|  |  |
| --- | --- |
|  | (16) |

за допомогою пакету MATLAB, скориставшись стандартною функцією simplify, яка відшукує найкоротшу форму представлення символьного виразу (див. [Додаток А](#_Додаток_А._Скрипт)). Формула для обчислення нового значення зі складністю за відомим обчисленим новим значенням остаточно набуде вигляду

## Формулювання алгоритму табу-пошуку для квадратичної задачі про призначення

Розроблений алгоритм, як і переважна більшість метаевристичних алгоритмів, є ітеративним. На кожній ітерації алгоритму відбувається спроба отримати кращий розв’язок. Основна ітерація алгоритму виконується доти, доки не буде задоволений деякий критерій зупинки роботи алгоритму. Критерієм зупинки може бути одна з перерахованих нижче умов, або їх комбінація (диз’юнкція):

1. вичерпано час, відведений на пошук розв’язку задачі;
2. досягнуто максимальної кількості ітерацій, заданих перед запуском пошуку розв’язку задачі;
3. досягнуто деякого порогового значення цільової функції (вартості) розв’язку задачі. Тобто заздалегідь задається прийнятне значення вартості розв’язку, якого достатньо в рамках проблеми, що вирішується. Таким чином, знаходиться компроміс між часом пошуку розв’язку та його вартістю, коли незначні покращення розв’язку протягом дуже тривалого часу пошуку вже не є критичними для контексту прикладної проблеми;
4. покращення розв’язку не спостерігалося протягом відносно великого часу або кількості ітерацій, тобто період часу до кожного наступного покращення розв’язку зростає на порядок;
5. покращення розв’язку в одиницю часу протягом деякого фіксованого періоду часу менше, ніж деяке мінімальне порогове значення покращення розв’язку;
6. інші, більш складні критерії, що визначаються обмеженнями та характерними рисами проблеми, процес вирішення якої передбачає розв’язання квадратичної задачі про призначення.

Перед початком основної ітерації алгоритму здійснюється етап ініціалізації. На етапі ініціалізації виконуються такі дії.

1. Задається критерій зупинки пошуку, або комбінація декількох критеріїв зупинки.
2. Вводиться розмірність задачі .
3. Вводиться матриця відстаней розмірністю між усіма парами місцеположень.
4. Вводиться матриця потоків розмірністю між усіма парами об’єктів.
5. Задається початковий допустимий вектор-розв’язок задачі. Зазвичай даний вектор конструюється як псевдовипадкова перестановка всіх його елементів, наприклад, за допомогою алгоритму Фішера-Йєтса (*Fisher–Yates shuffle*).
6. Обчислюється вартість сконструйованого початкового допустимого вектора-розв’язку задачі за формулою
7. Ініціалізується найкращий (найякісніший) відомий квазіоптимальний вектор-розв’язок задачі значенням .
8. Ініціалізується вартість найкращого (найякіснішого) відомого квазіоптимального вектора-розв’язку значенням .
9. Для кожної можливої перестановки двох елементів з індексами *i* та *j* у початковому допустимому векторі-розв’язку обчислюється значення , на яке змінюється величина вартості розв’язку при перестановці в ньому місцями цих елементів, за формулою

Загальна кількість таких перестановок у векторі довжини пар елементів з індексами *i* та *j* становить .

1. Для всіх перестановок у векторі розв’язку пар елементів з індексами *i* та *j* ініціалізується табу-список, наприклад, програмно у вигляді рваної (*jagged*) матриці, визначеної лише в області або , кожна комірка якої містить номер ітерації, починаючи з якої дана перестановка не є забороненою (табу), тобто є дозволеною для виконання над вектором-розв’язком . Початкові значення даного табу-списку мають містити номер першої ітерації, або деяку від’ємну ітерацію, оскільки під час першої ітерації алгоритму всі перестановки мають бути дозволеними.

Після етапу ініціалізації починається основна ітерація табу-пошуку, яка передбачає виконання таких дій.

1. Якщо серед усіх можливих перестановок у векторі-розв’язку пар елементів з індексами *i* та *j* знайдені перестановки *i* та *j*, внаслідок здійснення кожної з яких отримується розв’язок, вартість якого менша, ніж вартість найкращого відомого розв’язку , тобто , то серед них обирається перестановка *i* та *j* з мінімальним значенням , тобто така перестановка, яка якомога більше зменшує вартість розв’язку і тим самим підвищує його якість. Якщо серед усіх перестановок елементів з індексами *i* та *j* немає жодної перестановки, внаслідок виконання якої отримується якісніший розв’язок, ніж найкращий відомий, то здійснюється перехід до наступного пункту основної ітерації алгоритму табу-пошуку.
2. Якщо серед усіх можливих перестановок у векторі-розв’язку пар елементів з індексами *i* та *j*, , знайдені перестановки *i* та *j*, кожна з яких не виконувалася вже протягом останніх ітерацій і більше, то обрати серед них перестановку *i* та *j* з мінімальним значенням . В противному випадку, якщо немає жодної перестановки елементів з індексами *i* та *j*, які не виконувалися протягом останніх ітерацій і більше, то здійснюється перехід до наступного пункту основної ітерації алгоритму табу-пошуку.
3. Серед усіх можливих перестановок у векторі-розв’язку пар елементів з індексами *i* та *j*, які не є забороненими на даній ітерації за табу-списком, обирається перестановка *i* та *j* з мінімальним значенням .
4. Виконати обрану на одному з попередніх трьох пунктів перестановку елементів з індексами *i* та *j* у векторі-розв’язку , отримавши новий модифікований вектор-розв’язок .
5. Оновити значення вартості модифікованого внаслідок перестановки елементів з індексами *i* та *j* вектора-розв’язку за формулою .
6. Порівняти отримане на попередньому кроці значення вартості модифікованого розв’язку поточної ітерації з найменшим відомим значенням вартості найкращого (найякіснішого) відомого вектора-розв’язку . Якщо , то запам’ятати поточний вектор-розв’язок у якості найкращого відомого вектора-розв’язку , а також вартість поточного розв’язку у якості найменшої відомої вартості :

Також, за необхідності, запам’ятовується поточна ітерація, на якій було знайдено новий найкращий відомий розв’язок, час роботи алгоритму від моменту запуску, поточний час тощо.

1. Заборонити обрану перестановку елементів з індексами *i* та *j* у векторі-розв’язку протягом наступних *n* ітерацій, де *n* ­– псевдовипадкове натуральне число з рівномірним розподілом, згенероване у діапазоні . Заборона виконується шляхом запису в табу-список (або матрицю) номера наступної ітерації, починаючи з якої перестановка стає дозволеною, у вигляді суми номера поточної ітерації та згенерованого числа *n*.
2. Для кожної можливої перестановки елементів з індексами *p* та *q* у модифікованому векторі-розв’язку внаслідок перестановки елементів з індексами *i* та *j*, , , скоректувати значення за формулою
3. Для кожної можливої перестановки елементів з індексами *k* та *i* у модифікованому векторі-розв’язку внаслідок перестановки елементів з індексами *i* та *j*, , , обчислити нове значення за формулою
4. Для кожної можливої перестановки елементів з індексами *k* та *j* у модифікованому векторі-розв’язку внаслідок перестановки елементів з індексами *i* та *j*, , , обчислити нове значення , модифікувавши його, за формулою
5. Оновити змінну, що містить старе значення , новим обчисленим значенням : .
6. Скоректувати старе значення змінної для обраної перестановки елементів з індексами *i* та *j* у векторі-розв’язку , взявши його зі знаком мінус: .
7. Перевірити, чи виконується хоча б один заданий критерій зупинки роботи алгоритму. Якщо виконується, то основна ітерація завершується, повертається найякісніший знайдений квазіоптимальний вектор-розв’язок та його вартість . В противному випадку, тобто якщо критерій зупинки не виконується, здійснюється перехід до першого пункту основної ітерацій алгоритму.

## Висновок

В даному розділі було детально описано процес розробки табу-пошуку для квадратичної задачі про призначення загального виду та обґрунтовано його очікувану ефективність. Було проведено дослідження складності кожної операції, що виконується розробленим табу-пошуком, та зосереджено увагу на одній з найбільш ресурсоємких операцій складності . Під час розробки математичного підґрунтя, що стосується способу реалізації даної операції, було знайдено нову математичну властивість квадратичної задачі про призначення. Виявлену властивість було теоретично сформульовано у вигляді закону, що описується формулою

взаємозв’язку основних обчислювальних величин процесу табу-пошуку.

На основі знайденої властивості, вперше було виведено формулу, яка дозволяє виконувати зі складністю половину від кількості операцій, які раніше у всіх відомих метаевристичних алгоритмах виконувались зі складністю . Таким чином, теоретично було обґрунтовано та математично доведено очікуваний приріст швидкодії розробленого табу-пошуку у порівнянні з такими відомими алгоритмами, як, наприклад, Robust Tabu Search Еріка Таілларда.

Результатом досліджень, проведених у даному розділі, є формулювання остаточного алгоритму розробленого табу-пошуку для квадратичної задачі про призначення, який може бути реалізований на ЕОМ. У наступному розділі буде експериментально проведено практичне підтвердження вищої ефективності розробленого табу-пошуку у порівнянні з відомими метаевристичними методами.

# ПОРІВНЯННЯ ЕФЕКТИВНОСТІ ТА АНАЛІЗ РЕЗУЛЬТАТІВ

Перед тим, як перейти безпосередньо до конкретних методик тестування та порівняння результатів, для уявлення обчислювальних можливостей, які були наявні на момент розробки табу-пошуку в даній роботі, слід дати характеристику апаратним та програмним засобам, які були в ній використані. У зв’язку з законом Мура, який, принаймні, діяв до моменту виконання даної роботи, результати з замірами часу та швидкодії в інших сторонніх публікаціях за попередні роки можуть суттєво відрізнятись від тих, які отримані в даній роботі.

Під час розробки та порівняння ефективності роботи табу-пошуку було використано таке апаратне та програмне забезпечення:

* Процесор AMD Phenom II X6 1075T 3.0 GHz
* Пам’ять DDR3 1600 MHz 4096 MB
* Операційна система Windows 7 64-bit
* Середовище розробки Visual Studio 10 / 11 Developer Preview

Алгоритм розробленого табу-пошуку було реалізовано мовою програмування С++ 11 (див. [Додаток Б](#_Додаток_Б._Програмна)). Для порівняння ефективності роботи реалізованого алгоритму з іншими відомими метаевристичними алгоритмами було використано їх авторські реалізації на мовах програмування ANSI C/C++. Для проведення різного роду експериментів з метою створення спільного програмного інтерфейсу (API) для запуску як розробленого алгоритму, так і інших авторських реалізацій алгоритмів, для кожного з них було створено wrapper за шаблоном проектування Adapter.

Всі вихідні коди компілювалися у середовищі Visual Studio з конфігурацією Release та усіма увімкненими опціями оптимізації компілятора, спрямованими на надання переваги швидкості.

З метою досягнення максимальної об’єктивності отриманих результатів всі коди програм, що реалізують використані при тестуванні алгоритми, не є паралельними і не використовують багатопоточність, можливості мультиядерної архітектури та акселерацію на GPU.

У якості вхідних даних та прикладів квадратичних задач про призначення був використаний стандартний набір квадратичних задач про призначення, загальнодоступний на сторінці «Problem Instances and Solutions» порталу «QAPLIB - A Quadratic Assignment Problem Library» в мережі Internet [[49](#QAPLIB)]. Дані приклади задач є досить дослідженими і для більшості з них вже знайдено глобальні оптимуми за допомогою точних алгоритмів на високопродуктивних обчислювальних системах. Також ці приклади прийнято застосовувати при виконанні та публікації досліджень, присвячених квадратичній задачі про призначення. Всі ці задачі розбито на групи, що характеризуються певними характерними особливостями (розріджені матриці, симетричні матриці, манхеттенські відстані тощо). Кожна задача має назву, що складається з перших літер прізвища автора, та розмірності задачі .

## Порівняння ресурсоємкості

У даному підрозділі експериментально перевіряється практичний приріст швидкодії, очікуваний внаслідок застосування розробленого математичного підґрунтя, що дозволяє під час табу-пошуку обчислювати половину значень зі складністю замість складності . Для того, щоб дати оцінку суттєвості розробленого математичного підґрунтя у рамках даної роботи, необхідно експериментально з’ясувати, на скільки ресурсоємкою є операція обчислення значень до та після його застосування на фоні загального часу всіх інших обчислювальних операцій табу-пошуку.

Методика з’ясування загальної частки обчислень від усіх інших обчислень табу-пошуку передбачає використання засобів Visual Studio 2010 Profiling Tools. При цьому передбачається запуск табу-пошуку для розв’язання квадратичної задачі про призначення Tai100a на фіксований сталий період часу. Оскільки обчислення , що аналізуються, виконуються на кожній ітерації алгоритму в однаковому об’єму, то загальний час табу-пошуку визначає лише точність їх відносної частки, а не абсолютне значення. У якості критерію зупинки табу-пошуку взято ліміт часу роботи в 10 секунд.

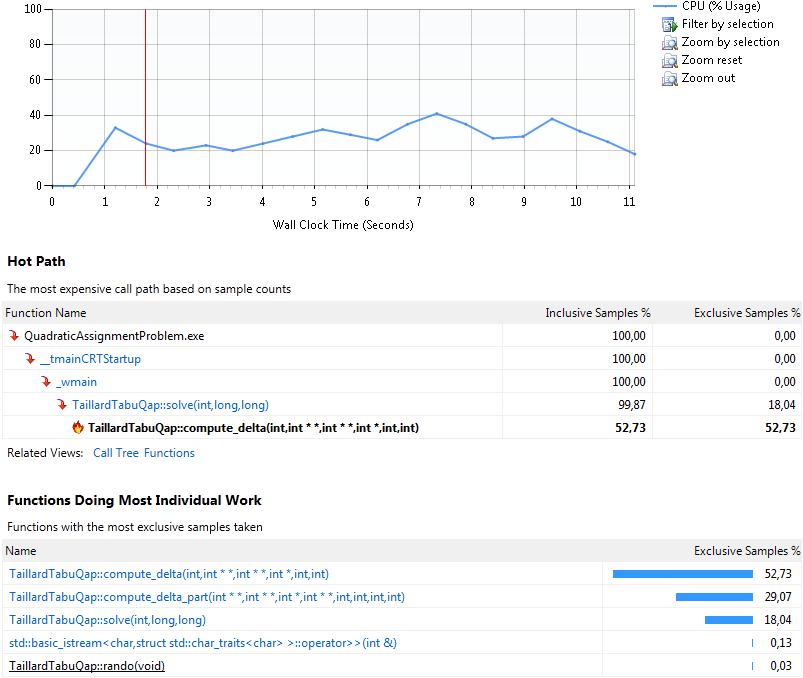


Рис. . Sample Profiling Report for Robust Tabu Search

На рис. 6 зображено звіт утилізації обчислювальних ресурсів, що містить графік використання процесорного часу, а також загальні частки часу обчислень у відсотках для найбільш ресурсоємких методів авторської програмної реалізації алгоритму Robust Tabu Search Еріка Таілларда.

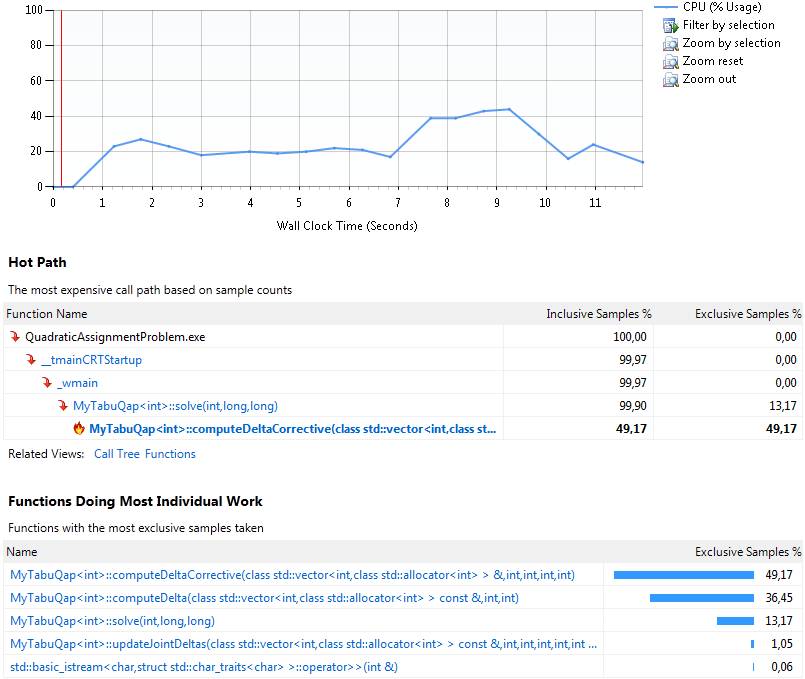
Із таблиці на рис. 6 видно, що найбільше обчислювального часу табу-пошуку, а саме 52.73 %, утилізується методом compute\_delta. Даний метод викликається разів на кожній основній ітерації алгоритму Robust Tabu Search і виконує повне обчислення значення за формулою, складність якої становить .

На другому місці за часткою використання обчислювального часу, а саме 29.07 %, знаходиться метод compute\_delta\_part. Даний метод викликається разів на кожній основній ітерації алгоритму Robust Tabu Search і виконує обчислення нового значення , коректуючи старе , за формулою, складність якої становить .

Решта обчислювального часу, а саме до 18.3 %, йде на всі інші операції, в тому числі пошук дозволених перестановок з мінімальним значенням та інше.

Даний звіт наочно демонструє, на скільки суттєвою за часом є операція повного обчислення нових значень зі складністю . Більше половини всього часу табу-пошуку витрачається лише на дану операцію. Разом з тим, обчислення значень зі складністю , кількість яких на порядок більша, ніж попередніх, займає значно менше часу. Все це підкреслює практичну важливість зменшення кількості викликів операції обчислення нових значень зі складністю та цінність розробленого з цією метою математичного підґрунтя. Далі ж для порівняння продемонстровано звіт утилізації обчислювальних ресурсів розробленим в даній роботі табу-пошуком.

На рис. 7 зображено звіт утилізації обчислювальних ресурсів, що містить графік використання процесорного часу, а також загальні частки часу обчислень у відсотках для найбільш ресурсоємких методів програмної реалізації розробленого табу-пошуку.

Рис. . Sample Profiling Report для розробленого табу-пошуку

Із таблиці на рис. 7 видно, що за часткою використання загального обчислювального часу, а саме 49.17 %, у розробленому табу-пошуку перше місце посідає тепер метод computeDeltaCorrective, який на кожній основній ітерації обчислює нових значень зі складністю за тією ж самою формулою, що й метод compute\_delta\_part у алгоритмі Robust Tabu Search Еріка Таілларда.

Метод computeDelta у розробленому табу-пошуку, який виконує ті ж самі дії, що й метод compute\_delta у реалізації Robust Tabu Search, а саме обчислення нового значення зі складністю , тепер посідає друге місце і використовує лише 36.45 % всього обчислювального часу. Це зумовлено тим, що тепер метод computeDelta у розробленому табу-пошуку викликається рази, а не , як у Robust Tabu Search. Інші значення , які у реалізації Robust Tabu Search обчислювалися за допомогою того ж методу compute\_delta, тепер обчислюються у даному табу-пошуку зі складністю методом updateJointDeltas, котрий використовує лише 1.05 % всього обчислювального часу. Ще одне нове значення для зворотної перестановки елементів з індексами *i* та *j* у векторі-розв’язку обчислюється також зі складністю шляхом простого множення старого значення на .

Якщо врахувати той факт, що метод computeDeltaCorrective у розробленому табу-пошуку фактично виконує ті ж самі дії, що й метод compute\_delta\_part у реалізації Robust Tabu Search, то очевидним є зменшення загально часу обчислення значень зі складністю до двох разів. Крім того, за один і той самий заданий час в 10 секунд розробленим табу-пошуком було виконано 46115 основних ітерацій, в той час як алгоритмом Robust Tabu Search було виконано лише 32461 ітерація, тобто майже в півтора рази менше, що ще раз свідчить про суттєвість застосування розробленого математичного підґрунтя.

Таким чином, експериментально було доведено, що розроблене в даній роботі математичне підґрунтя внаслідок теоретичного зменшення складності до для лише однієї операції має суттєвий вплив на підвищення швидкодії всього алгоритму табу-пошуку в цілому.

## Порівняння швидкодії

Як показали дослідження, табу-пошук дозволяє знаходити заздалегідь відомі глобальні оптимуми квадратичних задач про призначення розмірністю за дуже швидкий час (в середньому менше однієї секунди), на відміну від точних методів, для яких знаходження глобального оптимуму задач такої розмірності зайняло б неприйнятно великий час. Це дає можливість провести експериментальні дослідження з порівняння швидкодії знаходження різних глобальних оптимумів розробленим табу-пошуком та іншими метаевристичними алгоритмами. Звісно, під час розв’язання нестандартних квадратичних задач про призначення неможливо з’ясувати, чи був досягнутий глобальний оптимум, оскільки невідомим є значення його вартості. Однак експериментальні дані дослідження швидкості отримання відомих глобальних оптимумів можна екстраполювати на інші задачі, щоб мати уявлення про порівняння швидкодії роботи різних алгоритмів і приймати вибір щодо застосування того чи іншого алгоритму.

Таблиця

Порівняння середнього часу знаходження глобального оптимуму

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **QAP Instance** | **Середній час пошуку глобального оптимуму (мс)** | | | | |
| **Simulated Annealing** | **Fast Ant System** | **Алгоритм перетину локальних оптимумів** | **Robust Tabu Search** | **Розроблений табу-пошук** |
| Tai20a | 3283 | 32238 | 2589 | 767 | **419** |
| Tai20b | 3565 | 56 | **18** | 46 | 23 |
| Tai25a | 9169 | — | 80825 | 1862 | **1409** |
| Tai25b | 12268 | 174 | 391 | 189 | **93** |
| Tai30b | 14421 | **816** | 4817 | 1905 | 1189 |
| Chr20a | 17976 | 12798 | **601** | 2270 | 2002 |
| Chr25a | 15378 | 2534 | **780** | 16014 | 13172 |
| Bur26a | — | 327 | **104** | 302 | 174 |
| Bur26b | — | **78** | 112 | 529 | 308 |

У табл. 1 наведений усереднений за 1000 запусків час знаходження відомого глобального оптимуму у мілісекундах. У випадку, якщо в клітинці замість числа вказано прочерк, протестованому алгоритму не вдалося знайти глобальний оптимум відповідної квадратичної задачі про призначення за прийнятний час взагалі. Слід зауважити, що через наявність елементів стохастичного пошуку один і той самий алгоритм може витрачати дуже різний час для розв’язання однієї і тієї ж задачі. У зв’язку з цим необхідно виконувати статистичний аналіз на основі досить великої кількості експериментів.

Для порівняння швидкості знаходження глобальних оптимумів разом з розробленим табу-пошуком були використані найбільш відомі та ефективні метаевристичні алгоритми Simulated Annealing, Fast Ant System, та Robust Tabu Search Еріка Таілларда, а також розроблений окрім табу-пошуку під час проведення досліджень в даній роботі Алгоритм перетину локальних оптимумів [[57](#ПМК1)], в якому було також використане розроблене в рамках даної роботи математичне підґрунтя.

Для тестування всіх алгоритмів були використані стандартні квадратичні задачі про призначення, запропоновані Еріком Таіллардом (*Tai\*\*\**), Крістофайдсом та Беневейнтом (*Chr\*\*\**) , Бьоркардом та Оферменом (*Bur\*\*\**). Усі задачі мають розмірності від 20 до 30, з якими точні методи за наявних обчислювальних ресурсів не можуть впоратись взагалі. Задачі *Tai\*\*\**, згенеровані за допомогою генератора псевдовипадкових чисел з рівномірним розподілом, мають нулі на основних діагоналях матриць відстаней та потоків, а самі ж матриці симетричні відносно основних діагоналей. У задачах *Bur\*\*\** одна матриця є матрицею суміжності зваженого дерева другої, що є повним графом. У задачах *Bur\*\*\** дані першої матриці описують середній час послідовного друку кожної пари літер друкарської машинки, а дані другої – частоту зустрічання цих пар літер підряд у різних мовах.

Із результатів, наведених у табл. 1, видно, що розроблений табу-пошук проявив найвищу швидкодію для задач *Tai20a*, *Tai25a*, *Tai25b*. Решту задач найшвидше в середньому розв’язав алгоритм перетину локальних оптимумів, окрім задач *Tai30b* та *Bur26b*, які найшвидше в середньому розв’язав алгоритм Fast Tabu Search. Проте, алгоритм Fast Tabu Search виявився неспроможним знайти глобальний оптимум задачі *Tai25a* за прийнятний час взагалі, що вже робить його непридатним для розв’язання класу подібних задачі. Що стосується алгоритму Simulated Annealing, то він виявився аутсайдером в усіх тестах. Втім, слід відзначити, що для всіх без виключення задач розроблений табу-пошук перевершив за швидкодією алгоритм Robust Tabu Search.

Важливим є той факт, що серед усіх протестованих алгоритмів лише три алгоритми впоралися взагалі зі знаходженням глобальних оптимумів, а саме алгоритм перетину локальних оптимумів, Robust Tabu Search та розроблений табу-пошук. Оскільки розроблений табу-пошук у всіх без виключення випадках перевершив за швидкодією Robust Tabu Search, то лише алгоритм перетину локальних оптимумів та розроблений табу-пошук найкраще можуть конкурувати між собою для розв’язання всіх використаних квадратичних задач про призначення.

Отже, на основі отриманих результатів, можемо зробити висновок, що, якщо ми не маємо апріорних даних щодо специфіки квадратичної задачі про призначення, що розв’язується, то найдоцільнішим є використання розробленого табу-пошуку або розробленого в рамках даної роботи алгоритму перетину локальних оптимумів [[57](#ПМК1)].

## Порівняння якості розв’язків

Переважна більшість метаевристичних алгоритмів використовує у якості початкового допустимого розв’язку задачі обраний або сконструйований випадковим чином розв’язок. При цьому, як показують дослідження, продуктивність пошуку у сенсі підвищення його якості за одиницю часу швидко падає з часом. Як видно із рис. 8, значення вартості розв’язку на кожній ітерації стрімко зменшується на початку пошуку, а далі виходить на плато. Це означає, що переважна більшість часу роботи метаевристичних алгоритмів витрачається на незначне вдосконалення якості розв’язку, в той час як розв’язок, близький за якістю до оптимального, отримується на початку роботи алгоритму.

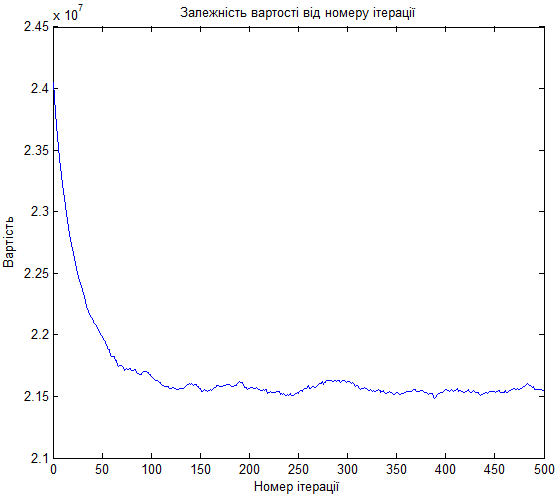


Рис. . Динаміка зміни вартості розв’язку в залежності від номера ітерації при розв’язанні задачі *Tai100a* розробленим табу-пошуком

Таким чином за умов, коли постановка задачі не потребує знаходження глобального оптимуму, а лише близького до нього значення, а час пошуку повинен бути передбачуваним та стабільним або постійним, актуальним є не час пошуку глобального оптимуму, який може зайняти години, дні або тижні, а якість знайденого розв’язку за короткий обмежений період часу. Крім того, неможливо з’ясувати, чи є знайдений розв’язок глобальним оптимумом, тому процес відшукання глобального оптимуму фактично має продовжуватись нескінченно або зупинятись примусово, якщо протягом дуже великого часу не вдається отримати покращення якості розв’язку.

В даному підрозділі досліджується якість розв’язків тих же самих квадратичних задач про призначення, що й в попередньому підрозділі. Методика тестування передбачає визначення усередненого за 100 запусків значення вартості найякіснішого знайденого розв’язку задачі, отриманого після 100 мілісекунд роботи алгоритму.

Таблиця

Порівняння середньої якості розв’язків

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **QAP Instance** | **Середнє значення вартості розв’язку** | | | | |
| **Simulated Annealing** | **Fast Ant System** | **Алгоритм перетину локальних оптимумів** | **Robust Tabu Search** | **Розроблений табу-пошук** |
| Tai20a | 719700 | 711814 | 710996 | 706297 | **705832** |
| Tai20b | 127963268 | 122571699 | **122455319** | 122480160 | **122455319** |
| Tai25a | 1211821 | 1191203 | 1191286 | 1177224 | **1175948** |
| Tai25b | 365033430 | 344532957 | 344626409 | 347301492 | **344424731** |
| Tai30b | 664031835 | **638359264** | 638876610 | 668590808 | 644680592 |
| Chr20a | 2850 | 2344 | 2262 | 2318 | **2261** |
| Chr25a | 6879 | 4283 | **4026** | 4331 | 4182 |
| Bur26a | 5475432 | 5428630 | **5427177** | 5431488 | 5428648 |
| Bur26b | 3848289 | 3818723 | **3817906** | 3822009 | 3819655 |

Як видно із табл. 2, найякісніші розв’язки для усіх задач, окрім однієї задачі *Tai30b*, були знайдені за допомогою двох розроблених в рамках даної роботи алгоритмів, а саме алгоритму перетину локальних оптимумів [[57](#ПМК1)] та розробленого табу-пошуку. Лише одна задача була розв’язана якісніше за відведений час стороннім алгоритмом Fast Ant System. Слід також відзначити, що в даному тесті для всіх задач розроблений табу-пошук показав кращі результати, ніж Robust Tabu Search. Що стосується задачі *Chr25a*, то розроблений табу-пошук показав найкращий результат, поступившись місцем лише алгоритму перетину локальних оптимумів. Аутсайдером виявився алгоритм Simulated Annealing, який показав найгірші результати у всіх випадках.

Таким чином, розроблені в рамках даної роботи алгоритми дали найякісніші розв’язки для всіх крім одного тестів. Отже у випадках, коли необхідно отримувати квазіоптимальні розв’язки заздалегідь невідомих квадратичних задач про призначення у реальному часі за обмежений період, то найдоцільнішим є вибір алгоритму перетину локальних оптимумів або розробленого табу-пошуку.

Далі, після того, як з’ясовано, яка середня якість розв’язків досягається всіма випробуваними алгоритмами за сталий відведений період часу, розглянемо детальніше, як змінюється якість розв’язку у часі протягом відведеного періоду. Динаміку процесу підвищення якості розв’язку у кожен момент часу найзручніше досліджувати, представивши графічно залежність вартості найменшого відомого розв’язку від часу у діапазоні від початку відліку часу до ліміту часу, відведеного на пошук. При цьому можна визначати відрізки часу, протягом яких алгоритми працюють з найвищою ефективністю і виконують найзначніше покращення результату. Дана методика дозволить визначити оптимальний ліміт часу, якого достатньо для досягнення значень, близьких до глобальних оптимумів. При цьому можна визначити моменти часу, після яких робота алгоритму вже стає спрямована на «мікрооптимізацію». Це може стати корисним особливо в тих випадках, коли якість розв’язку перестає виправдовувати час обчислень, що витрачається на підвищення цієї якості розв’язку.

На рис. 9 зображена динаміка зміни якості розв’язку у часі протягом перших 100 мс розв’язання задачі *Tai100a*. Ефективність роботи алгоритму Simulated Annealing падає вже після перших 20 мс процесу пошуку і якість отриманого результату є найгіршою. Найдовше спостерігається вдосконалення якості при роботі алгоритму Fast Ant System, а саме протягом 80 мс. Алгоритм перетину локальних оптимумів припиняє покращувати результат після 25 мс. Розроблений табу-пошук демонструє найбільш стрімке зниження вартості розв’язку і виходить на плато після 70 мс пошуку, конкуруючи далі з невеликим випередженням з алгоритмом Robust Tabu Search.

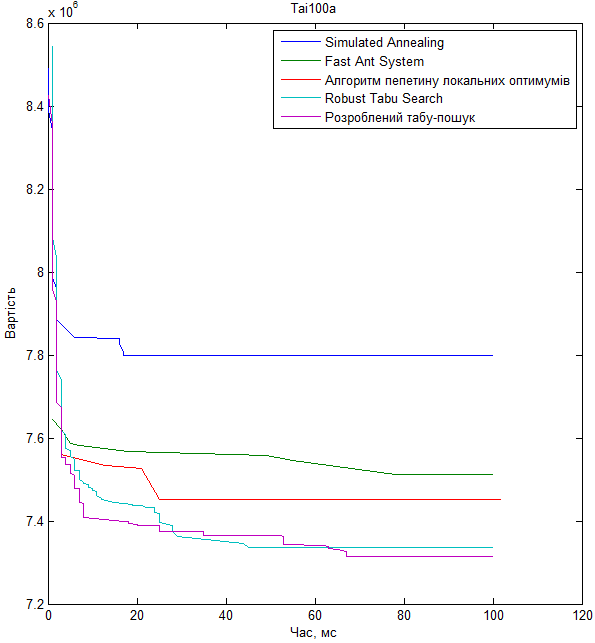


Рис. . Залежність вартості розв’язку від часу для задачі *Tai100a*

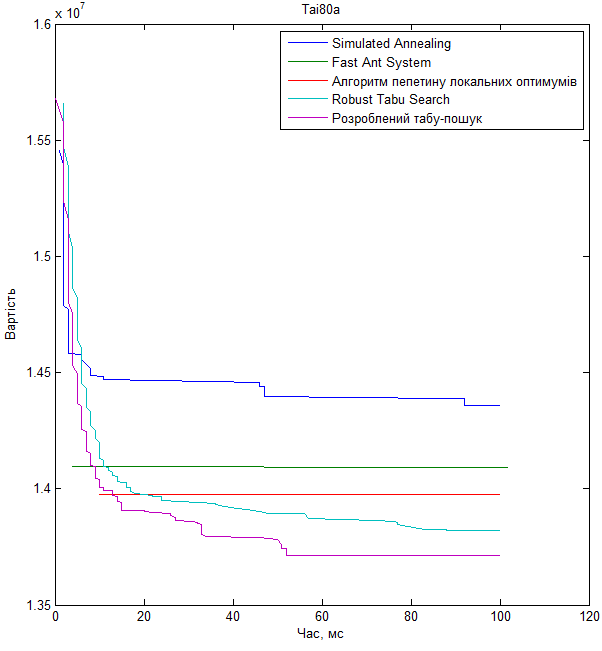


Рис. . Залежність вартості розв’язку від часу для задачі *Tai80a*

Аналогічна ситуація спостерігається на рис. 10 та рис. 11 при зменшенні розмірності квадратичної задачі до та до відповідно. Приблизно тими ж залишаються моменти часу виходу якості розв’язків на плато. Найстрімкіше зниження вартості розв’язку відбувається під час роботи розробленого табу-пошуку.

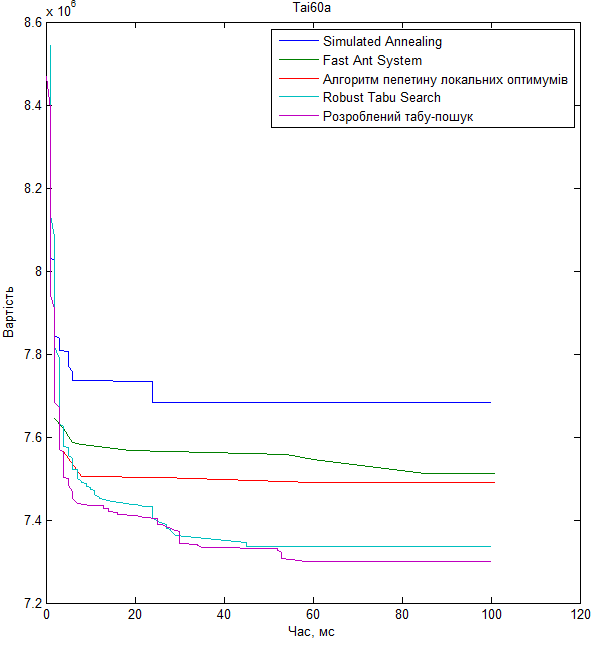


Рис. . Залежність вартості розв’язку від часу для задачі *Tai60a*

## Висновок

У даному підрозділі було проаналізовано та експериментально доведено суттєвість впливу розробленого математичного підґрунтя на швидкодію, а також ефективність роботи розробленого табу-пошуку під час розв’язання різних квадратичних задач про призначення. Було проведено порівняння швидкодії та якості розв’язків розробленого табу-пошуку з такими відомими метаевристичними алгоритмами як Simulated Annealing, Fast Ant System та Robust Tabu Search Еріка Таілларда. У всіх проведених тестах розроблений табу-пошук показав вищу швидкодію та якість розв’язків, ніж алгоритм Robust Tabu Search. Якісні результати та високу швидкодію показав також розроблений у рамках досліджень, що проводились у даній роботі, алгоритм перетину локальних оптимумів, який побудовано з використанням того ж математичного підґрунтя, що розроблено для використання у табу-пошуку.

Досліджено ефективність роботи алгоритмів в залежності від часу, що дозволяє визначати найбільш продуктивні періоди часу підвищення якості розв’язків. Проведено порівняння залежності якості розв’язку деяких квадратичних задач про призначення від часу пошуку для всіх алгоритмів, що тестувалися. Розроблений табу-пошук у всіх випадках проявив найбільш стрімке підвищення якості розв’язку від самого початку, що робить його найбільш ефективним при застосуванні за умов, коли час пошуку обмежений коротким проміжком.

# ОХОРОНА ПРАЦІ ТА БЕЗПЕКА У НАДЗВИЧАЙНИХ СИТУАЦІЯХ

## Вступ

Програмне забезпечення, реалізоване на основі даної дипломної роботи, призначене для використання на підприємстві в приміщенні офісного типу. Програмне забезпечення призначене для обчислення наближеного розв’язку квадратичної задачі про призначення, яка лежить в основі практичної економічної задачі, що вирішується. Апаратним середовищем, на якому виконуватиметься робота з програмним забезпеченням, є персональна ЕОМ, обладнана рідкокристалічним дисплеєм, принтером, сканером та іншими офісними периферійними пристроями.

Робочі операції, в яких буде використано програмне забезпечення, включають введення інформації, налаштування, контроль та проведення обчислень, складання звітів щодо отриманих результатів роботи. Основними вимогами з точки зору охорони праці з урахуванням виробничого середовища є попередження виникнення таких загроз для здоров’я людини як фізичні та нервово-психічні перенавантаження, погіршення зору, займання, ураження електричним струмом та ін.

## Опис приміщення

Розглянемо фактори, які характеризують приміщення, в якому використовується програмне забезпечення. Вид приміщення – офіс. План приміщення зображено на рис.12. Приміщення має природне та штучне освітлення. Природне освітлення здійснюється через світловий проріз, орієнтований на північ, віконний проріз обладнано жалюзями. Приміщення обладнано системами опалення та кондиціонування повітря. Приміщення також обладнується шафою для зберігання документів, магнітних дисків, а також тумбою. Приміщення оснащено аптечкою першої медичної допомоги та вогнегасником.

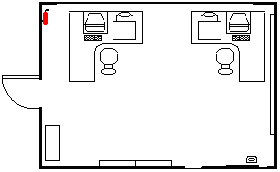


Рис. . План приміщення

Робоча площа приміщення становить 14 м2; висота – 3 м. Об’єм приміщення становить . Кількість працюючих – 2. Площа приміщення на одного працівника становить , об’єм приміщення на одного працівника .

Згідно з [[58](#Охр1)] площа на одне робоче місце має становити не менше ніж 6.0 м2, а об’єм не менш ніж 20 м3.

Отже, об'ємно-планувальні рішення приміщення для роботи з ВДТ ЕОМ і ПЕОМ відповідають вимогам [[58](#Охр1)].

## Аналіз умов праці

### Напруженість праці користувача ПЕОМ

Виходячи з характеру розробленого у ДП програмного продукту та згідно з [[59](#Охр3)] робота користувача ПЕОМ за показниками напруженості трудового процесу відноситься до наступних класів.

1. За показником інтелектуального навантаження
   1. Зміст роботи – рішення простих альтернативних завдань згідно з інструкцією (**допустимий**).
   2. Сприймання сигналів (інформації) та їх оцінка – сприймання сигналів з наступною корекцією дій та операцій (**допустимий**).
   3. Розподіл функцій за ступенем складності завдання – обробка та виконання завдання (**оптимальний**).
   4. Розподіл функцій за ступенем складності завдання – робота за індивідуальним планом (**оптимальний**).
2. За сенсорним навантаженням
   1. Тривалість зосередженого спостереження – до 25 % від часу зміни (**оптимальний**).
   2. Щільність сигналів (світлових, звукових) та повідомлень в середньому за 1 годину роботи – до 75 (**оптимальний**).
   3. Кількість виробничих об'єктів одночасного спостереження – до 5 (**оптимальний**).
   4. Навантаження на зоровий аналізатор
      1. Розмір об'єкта розрізнення (при відстані від очей працюючого до об'єкта розрізнення не більше 0,5 м), при тривалості зосередженого спостереження (% часу зміни) – більше 5 мм 100% часу (**оптимальний**).
      2. Робота з оптичними приладами – **відсутня**.
      3. Спостереження за екранами відеотерміналів (годин на зміну) – 3-4 (**шкідливий, 1 ступінь**)
   5. Навантаження на слуховий аналізатор (при виробничій необхідності сприйняття мови чи диференційованих сигналів) – **відсутнє**.
   6. Навантаження на голосовий апарат – **відсутнє**.
3. Емоційне навантаження
   1. Ступінь відповідальності за результат своєї діяльності. Значущість помилки – несе відповідальність за виконання окремих елементів завдання. Вимагає додаткових зусиль в роботі з боку працівника (**допустимий**).
   2. Ступінь ризику для власного життя – виключений (**оптимальний**).
   3. Ступінь відповідальності за безпеку інших осіб життя – виключений (**оптимальний**).
4. Монотонність навантажень
   1. Кількість елементів (прийомів), необхідних для реалізації простого завдання або в операціях, які повторюються багаторазово ­– більше 10 (**оптимальний**).
   2. Тривалість виконання простих виробничих завдань чи операцій, що повторюються, – 100-25 сек. (**допустимий**).
   3. Час активних дій (в % до тривалості зміни, решту часу ­– спостереження за технологічним процесом) – 20 та більше (**оптимальний**).
   4. Монотонність виробничої обстановки (час пасивного спостереження за технологічним процесом в % від часу зміни) – менше 75 (**оптимальний**).
5. Режим праці
   1. Фактична тривалість робочого дня – 6-7 год. (**оптимальний**).
   2. Змінність роботи – однозмінна робота (без нічної зміни) (**оптимальний**).
   3. Наявність регламентованих перерв, та їх тривалість – перерви регламентовані, достатньої тривалості від 3% до 7% часу зміни (**допустимий**).

Оскільки оцінка напруженості праці встановлюється за показником, який має найвищий ступінь напруженості, то характер робіт має шкідливий клас першого ступеню. Нервово-психічні шкідливі виробничі фактори у даному випадку пов’язані з перевантаженням аналізаторів, а саме великим напруженням зору, тому необхідно при розробці програмного забезпечення врахувати можливість звукового оповіщення за допомогою низки звукових індикаторів інформації замість візуальних. Даний захід дозволяє розподілити напруження між слуховим та зоровим аналізатором.

### Повітряне середовище

Відповідно до [[60](#Охр4)] проведемо порівняння фактичних значень параметрів мікроклімату виробничих приміщень з оптимальними значеннями. Результати порівняння зведено у табл. 3.

Таблиця

Значення параметрів мікроклімату

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Сезон року** | **Параметри мікроклімату** | | | | | |
| **Оптимальні** | | | **Фактичні** | | |
| Температура повітря, °С | Відносна вологість,% | Швидкість руху, м/с | Температура повітря, °С | Відносна вологість,% | Швидкість руху, м/с |
| Холодний | 22-24 | 60-40 | 0,1 | 22-23 | 55-45 | 0,1 |
| Теплий | 23-25 | 60-40 | 0,1 | 24-25 | 60-50 | 0,1 |

Температура внутрішніх поверхонь робочої зони (стіни, підлога, стеля), не виходить більше, ніж на 2 град. °С за межі оптимальних величин температури повітря. Перепад температури повітря по висоті робочої зони не перевищує 2 град. °С, а по горизонталі робочої зони та протягом робочої зміни – не виходить за межі допустимих температур для даної категорії роботи.

Таким чином, роботи, які виконуються в приміщенні, відносяться до категорії Iа. Це досягається за допомогою використання змішаної вентиляції в приміщенні та системи кондиціонування повітря з індивідуальним регулюванням температури та об’єму повітря, що подається.

### Шум

Основними джерелами шуму в приміщенні є системні блоки персональних ЕОМ, а саме вентилятори та жорсткі диски, що містяться в них, шум яких за [[61](#Охр5)] можна класифікувати як постійний. Також можливе виникнення непостійного переривчастого шуму внаслідок роботи принтерів, сканерів, а також переривчастого мінливого шуму внаслідок дії оптичних приводів, дисководів.

Середній рівень шуму одного системного блоку ПЕОМ не перевищує 30 дБА. Оскільки в приміщенні передбачено два робочі місця для двох працюючих, то постійний шум створюється двома системними блоками ПЕОМ. Розрахуємо середній рівень постійного шуму LАсер, дБА, використовуючи додаток 1 нормативного документу [[61](#Охр5)]. Оскільки передбачається використання ПЕОМ однакової конфігурації, то різниця між найбільшим і найменшим виміряними рівнями не перевищуватиме 5 дБ. Тоді середнє *LА*сер дорівнює середньому арифметичному значенню виміряних рівнів, тобто рівню шуму 30 дБА одного системного блока ПЕОМ.

Обчислимо рівень еквівалентного рівня переривчастого шуму принтеру, сканеру та оптичного приводу, використовуючи додаток 2 нормативного документу [[61](#Охр5)].

Принтер: *L*1*А* = 50 дБА протягом 15 хв.

Сканер: *L*2*А* = 30 дБА протягом 5 хв.

Дисковод: *L*3*А* = 40 дБА протягом 15 хв.

За таблицею Д.2.1 визначаємо Li для кожного найближчого значення рівня:

*L*1 = 15,1 дБА

*L*2 = 19,1 дБА

*L*3= 15,1 дБА

Визначаємо величини LАі – Lі для кожного рівня:

50 дБА – 15,1 дБА = 34,9 дБА

30 дБА – 19,1 дБА = 10,9 дБА

40 дБА – 15,1 дБА = 24,9 дБА

Визначаємо енергетичну суму рівнів за таблицею Д.1.1 [[61](#Охр5)]:

34,9 дБА ­– 10,9 дБА = 24 дБА

При різниці рівнів 24 дБА додаток дорівнює нулю, який додаємо до більшого рівня:

34,9 дБА + 0 дБА = 34,9 дБА

Визначаємо різницю між сумою двох перших рівнів та третім рівнем:

34,9 дБА – 24,9 дБА = 10,0 дБА

При різниці рівнів 10,0 дБА додаток LА = 0,4 дБА, який додаємо до більшого рівня:

34,9 дБА + 0,4 дБА = 35,3 дБА

Еквівалентний рівень – 35,3 дБА.

Розрахуємо середній рівень шуму системних блоків, принтеру, сканеру та оптичних приводів. Оскільки різниця між середнім рівнем шуму системних блоків та еквівалентним рівнем не перевищує 5 дБ, то середнє значення дорівнює середньому арифметичному (17).

|  |  |
| --- | --- |
|  | (17) |

Отже, фактичний обмірюваний рівень шуму склав дБА, що (згідно з [[61](#Охр5)]) задовольняє вимогам (до 50 дБА для програмістів ПЕОМ).

### Освітлення

Приміщення має суміщене освітлення, що складається з природного одностороннього бокового освітлення та штучного робочого.

При роботі з ПЕОМ необхідно розрізняти символи на клавішах клавіатури, розмір яких становить в середньому від 1 мм до 5 мм. Отже, згідно з [[62](#Охр7)], характеристику зорової роботи можна класифікувати як малої точності, підрозряд зорової роботи – V. При використанні стандартної матової білої клавіатури з чорними символами характеристика фону класифікується як світла, а контраст об’єкта з фоном можна класифікувати як великий. Отже, при системі загального штучного освітлення освітленість має становити 200 лк, сукупність нормованих величин показника осліпленості і коефіцієнта пульсації коефіцієнт суміщеного освітлення при боковому освітленні .

Необхідний нормований рівень освітленості досягається за допомогою використання газорозрядних ламп ЛДЦ80, що містяться у кількості по 4 штуки у світильниках, які рівномірно розподіляються по площі стелі, та за рахунок наявності бічного світлового прорізу розміром .

Отже, до заходів, які необхідно приймати для збереження виробничих умов освітленості у рамках нормованих, є своєчасне очищення скла вікон та своєчасна заміна ламп у світильниках у випадках їх виходу з ладу.

## Аналіз пожежної безпеки приміщення

Запобігання пожежі досягається виключенням утворення горючого середовища і джерел загорянь. До легкозаймистих предметів відносяться дерев’яні меблі (шафа з документами, тумба, робочі столи). Оскільки перераховані речі можна віднести до твердих горючих речовин і матеріалів, здатних при взаємодії з повітрям тільки горіти, то розглянуте приміщення згідно з [[63](#Охр10)] можна віднести до категорії В (згідно з НАПБ Б.07.005-86).

Приміщення оснащене системою автоматичної пожежної сигналізації відповідно до вимог [[63](#Охр10)], має димові пожежні сповіщувачі, тобто виконано норматив [[64](#НАПБ_Б060042007)].

З урахуванням площі приміщення 14 м2 та наявності двох ПЕОМ, нормативні вимоги [[65](#Охр11)] передбачають наявність двох вуглекислотних вогнегасників ВВК-1,4 (старе позначення – ОУ-2) чи ВВК-2 (старе позначення – ОУ-3) або один ВВПА-400 (згідно з вимогами, 2 шт. на кожні 20 м2). У даному приміщенні є лише 1 вогнегасник, тому необхідно придбати ще 1 шт.

Шляхи евакуації відповідають нормативним вимогам [[63](#Охр10)].

Всі співробітники пройшли інструктаж.

## Ергономіка робочого місця

Обладнання і організація робочого місця працюючих з ПЕОМ мають забезпечувати відповідність конструкцій всіх елементів робочого місця.

Висота робочої поверхні складає 700 мм, має простір для ніг заввишки 680 мм, завширшки 700 мм, завглибшки (на рівні колін) 500 мм, на рівні простягнутої ноги – 700 мм. Довжина столу 1400 мм, ширина – 1400 мм.

Робочий стілець є підйомно-поворотним, регульованим за висотою. За відстанню від спинки до переднього краю сидіння поверхня сидіння є плоскою, передній край – заокругленим. Регулювання за кожним із параметрів здійснюється незалежно, легко і надійно фіксується. Шаг регулювання елементів стільця становить: для лінійних розмірів – 15 мм, для кутових – 2 град. Зусилля регулювання становить 15 Н.

Висота поверхні сидіння регулюється в межах 400...500 мм, а ширина і глибина становить 450 мм. Кут нахилу сидіння – 14 град. вперед і 4 град. назад.

Висота спинки стільця становить 300 мм, ширина 400 мм, радіус кривизни горизонтальної площини 400 мм. Кут нахилу спинки регулюється в межах 1...30 град. від вертикального положення. Відстань від спинки до переднього краю сидіння регулюється в межах 260...400 мм.

Для зниження статичного напруження м'язів верхніх кінцівок використовуються стаціонарні підлокітники завдовжки 300 мм, завширшки 60 мм, що регулюються за висотою над сидінням у межах 230...260 мм і відстанню між підлокітниками в межах 350...500 мм.

Поверхня сидіння і спинки стільця є напівм'якою з нековзним, повітронепроникним покриттям, що легко чиститься і не електризується.

Робоче місце відповідає вимогам [[58](#Охр1)].

## Висновки

Аналіз умов праці в розглянутому робочому приміщенні показав, що умови праці з ПЕОМ відповідають вимогам [[58](#Охр1)], оскільки:

* площа та об’єм приміщення на одного працюючого відповідають нормативним значенням;
* мікроклімат, що підтримується в приміщенні, має оптимальні значення температури, відносної вологості та швидкості руху повітря;
* фактичний рівень шуму в приміщенні не перевищує максимального допустимого;
* рівень освітленості робочих поверхонь є достатнім;
* обладнання і організація робочого місця працюючих забезпечують відповідність конструкцій всіх елементів робочого місця.

Рекомендації:

* знизити нервово-психічний шкідливий виробничий фактор, пов’язаний з перевантаженням зорового аналізатора, шляхом введення звукових інформаційних оповіщень;
* контролювати технічний стан вентиляторів ПЕОМ, що є основними джерелами шуму в приміщенні;
* своєчасно очищувати скло вікон та своєчасно замінювати лампи у світильниках у випадках їх виходу з ладу;
* оздобити приміщення додатковим вуглекислотним вогнегасником.

# ВИСНОВКИ

Результати, отримані внаслідок проведених в рамках даної роботи досліджень, полягають в наступному:

1. знайдено нову математичну властивість квадратичної задачі про призначення і сформульовано у вигляді формули виду  
   яка відображає взаємозв’язок всіх величин приросту значення цільової функції внаслідок здійснення перестановок пар елементів із однієї трійки індексів у векторі-розв’язку (див. підрозділ 3.2);
2. на основі знайденої властивості квадратичної задачі про призначення вперше запропоновано нову формулу

яка дозволяє зі складністю замість виконувати половину від кількості найбільш ресурсоємких операцій з оновлення приростів вартості розв’язку внаслідок перестановки елементів, що в результаті призводить до підвищення швидкодії близько 25%;

1. з використанням розробленого математичного підґрунтя, розроблено новий алгоритм табу-пошуку, який має до двох разів вищу швидкодію, ніж відомі метаевристичні алгоритми, в тому числі Robust Tabu Search, а також дозволяє отримувати розв’язки вищої якості за один і той самий час.

Розроблене математичне підґрунтя може бути застосоване не лише для використання табу-пошуком, але й іншими метаевристичними алгоритмами, наприклад, мурашиними системами. Дане твердження було доведено шляхом розробки алгоритму перетину локальних оптимумів [[57](#ПМК1)], який разом з розробленим табу-пошуком під час порівняння ефективності з іншими метаевристичними алгоритмами відзначили себе одними із найбільш швидкодіючих та таких, що дозволяють отримати найякісніші розв’язки за сталий період часу.

Розроблений табу-пошук для квадратичної задачі про призначення дуже просто розпаралелюється шляхом одночасного запуску основних ітерацій на багатьох вузлах високопродуктивних обчислювальних систем.

Перспективною є також реалізація розробленого табу-пошуку для масивно-паралельних обчислень на графічних процесорах за допомогою мови програмування CUDA C або технології Microsoft Accelerated Massive Parallelism.

Також проведено аналіз праці в розглянутому робочому середовищі та надано рекомендації щодо усунення шкідливих та небезпечних виробничих факторів відповідно до діючих нормативних документів.

# СПИСОК ВИКОРИСТАНИХ ЛІТЕРАТУРНИХ ДЖЕРЕЛ

x

|  |  |
| --- | --- |
| 1. | Assignment problems and the location of economic activities [Text] / T. C. Koopmans, M. J. Beckman // Econometrica. — 1957. — № 25. — P. 53-76. |
| 2. | The quadratic assignment problem [Text] / E. L. Lawler // Management Science. — 1963. — № 9. — P. 586-599. |
| 3. | P-complete approximation problems [Text] / T. Gonzalez, S. Sahni // Journal of the Association of Computing Machinery. — 1976. — № 23. — P. 555–565. |
| 4. | A new rounding procedure for the assignment problem with applications to dense graph arrangement problems [Text] / S. Arora, A. Frieze, H. Kaplan // 37-th Annual IEEE Symposium on Foundations of Computer Science (FOCS) : proceedings. — 1996. — P. 21–30. |
| 5. | On the use of exact and heuristic cutting plane methods for the quadratic assignment problem [Text] / M. S. Bazaraa, H. D. Sherali // Journal of Operations Research Society. — 1982. — № 33. — P. 991–1003. |
| 6. | The derivation of a greedy approximator for the Koopmans-Beckmann quadratic assignment problem [Text] / C. S. Edwards // Combinatorial Programming : proc. of 77-th conf. — 1977. — P. 55-86. |
| 7. | A branch and bound algorithm for the Koopmans-Beckmann quadratic assignment problem [Text] / C. S. Edwards // Mathematical Programming Study. — 1980. — № 13. — P. 35-52. |
| 8. | Quadratic assignment problems [Text] / G. Finke, R. E. Burkard, F. Rendl // Annals of Discrete Mathematics. — 1987. — № 31. — P. 61–82. |
| 9. | Kronecker Products and Matrix Calculus with Applications [Text] / A. Graham. — Toronto : Halsted Press, 1981. |
| 10. | Computers and intractability: A guide to the theory of NP-completeness [Text] / M. R. Garey, D. S. Johnson. — New York : W. H. Freeman and Company, 1979. |
| 11. | Performance ratio of heuristics for triangle inequality quadratic assignment problems [Text] / M. Queyranne // Operations Research Letters. — 1986. — № 4. — P. 231–234. |
| 12. | Worst case analysis of a new heuristic for the traveling salesman problem / N. Christofides // Graduate School of Industrial Administration, Carnegie-Mellon University, Pittsburgh, PA, 1976 : tech. report 338. |
| 13. | The backboard wiring problem: A placement algorithm [Text] / L. Steinberg // SIAM Review. — 1961. — № 3. — P. 37–50. |
| 14. | Quadratische Bottleneckprobleme [Text] / R. E. Burkard // Operations Research Verfahren. — 1974. — № 18. — P. 26–41. |
| 15. | On random quadratic bottleneck assignment problems [Text] / R. E. Burkard, U. Fincke // Mathematical Programming. — 1982. — № 23. — P. 227–232. |
| 16. | An algebraic approach to assignment problems [Text] / R. E. Burkard, W. Hann, U. Zimmermann // Mathematical Programming. — 1977. — № 12. — P. 318–327. |
| 17. | On the biquadratic assignment problem [Text] / R. E. Burkard, E. Cela, B. Klinz // DIMACS Series on Discrete Mathematics and Theoretical Computer Science. — 1994. — № 16. — P. 117–146. |
| 18. | Heuristics for biquadratic assignment problems and their computational comparison [Text] / R. E. Burkard, E. Cela // European Journal of Operational Research. — 1995. — № 83. — P. 283–300. |
| 19. | A GRASP for the biquadratic assignment problem [Text] / T. Mavridou, P. M. Pardalos, L. S. Pitsoulis, M. G. Resende // European Journal of Operations Research. — 1998. — № 105. — P. 613–621. |
| 20. | Multiprocessor scheduling with the aid of network flow algorithms [Text] / H. S. Stone // IEEE Transactions on Software Engineering. — 1977. — № 3. — P. 85–93. |
| 21. | A polynomial algorithm to optimally schedule tasks on a virtual distributed system under tree-like precedence constraints [Text] / P. Chretienne // European Journal of Operational Research. — 1989. — № 43. — P. 225–230. |
| 22. | A quadratic assignment problem without column constraints [Text] / H. Greenberg // Naval Research Logistic Quarterly. — 1969. — № 16. — P. 417–422. |
| 23. | A Lagrangean relaxation algorithm for sparse quadratic assignment problems [Text] / I. Z. Milis, V. F. Magirou // Operations Research Letters. — 1995. — № 17. — P. 69–76. |
| 24. | Lower bounds for the quadratic semi-assignment problem / F. Malucelli, D. Pretolani // Centre des Recherches sur les Transports, Universite de Montreal, Canada, 1993 : tech. report 955. |
| 25. | Quadratic Assignment Problems: Solution Methods and Applications : Ph.D. Thesis / F. Malucelli ; Dipartimento di Informatica, Universita di Pisa, Italy, 1993. |
| 26. | Polyhedral Combinatorics and the Acyclic Subdigraph Problem [Text] / M. Junger. — Berlin, Germany : Heldermann, 1985. |
| 27. | Extremal Graph Theory [Text] / B. Bollobas. — London : Academic Press, 1978. |
| 28. | Lower bounds for the quadratic assignment problem based upon a dual formulation [Text] / P. Hahn, T. Grant // to appear in Operations Research. |
| 29. | Solution of the quadratic assignment problem using the Hungarian method [Text] / P. Hahn, T. Grant, N. Hall // to appear in European Journal of Operational Research. |
| 30. | An experimental comparison of techniques for the assignment of facilities to locations [Text] / C. E. Nugent, T. E. Vollmann, J. Ruml // Journal of Operations Research. — 1969. — № 16. — P. 150–173. |
| 31. | Locational decisions with interactions between facilities: the quadratic assignment problem a review / P. B. Mirchandani, T. Obata // Rensselaer Polytechnic Institute, Troy, New York, May 1979 : working paper Ps-79-1. |
| 32. | A parallel branch and bound algorithm for the quadratic assignment problem [Text] / C. Roucairol // Discrete Applied Mathematics. — 1987. — № 18. — P. 221–225. |
| 33. | Joining forces in solving large-scale quadratic assignment problems in parallel [Text] / A. Bruengger, J. Clausen, A. Marzetta, M. Perregaard // 11-th IEEE International Parallel Processing Symposium (IPPS) : proceedings. — 1997. — P. 418–427. |
| 34. | Optimal and suboptimal algorithms for the quadratic assignment problem [Text] / P. C. Glimore // SIAM Journal on Applied Mathematics. — 1962. — № 10. — P. 305–313. |
| 35. | Allocating facilities with CRAFT [Text] / E. S. Buffa, G. C. Armour, T. E. Vollmann // Harvard Business Review. — 1962. — № 42. — P. 136–158. |
| 36. | Local Search in Combinatorial Optimization [Text] / E. Aarts, J. K. Lenstra. — Chichester : Wiley, 1997. |
| 37. | Algorithms for assignment problems on an array processor [Text] / A. M. Frieze, J. Yadegar, S. El-Horbaty, D. Parkinson // Parallel Computing. — 1989. — № 11. — P. 151–162. |
| 38. | A computationally simplified pair exchange algorithm for the quadratic assignment problem [Text] / C. H. Heider // Center for Naval Analysis, Arlington, Virginia, 1972 : paper No. 101. |
| 39. | Tabu search - Part I [Text] / F. Glover // ORSA Journal on Computing. — 1989. — № 1. — P. 190–206. |
| 40. | Tabu search applied to the quadratic assignment problem [Text] / J. Skorin-Kapov // ORSA Journal on Computing. — 1990. — № 2. — P. 33–45. |
| 41. | Robust tabu search for the quadratic assignment problem [Text] / E. Taillard // Parallel Computing. — 1991. — № 17. — P. 443–455. |
| 42. | The reactive tabu search [Text] / R. Battiti, G. Tecchiolli // ORSA Journal on Computing. — 1994. — № 6. — P. 126–140. |
| 43. | Thermodynamical approach to the traveling salesman problem: An efficient simulation algorithm [Text] / V. Cerny // Journal of Optimization Theory and Applications. — 1985. — № 45. — P. 41–51. |
| 44. | Equations of state calculations by fast computing machines [Text] / N. Metropolis, A. Rosenbluth, M. Rosenbluth, A. Teller, E. Teller // Journal of Chemical Physics. — 1953. — № 21. — P. 1087–1092. |
| 45. | A thermodynamically motivated simulation procedure for combinatorial optimization problems [Text] / R. E. Burkard, F. Rendl // European Journal Operational Research. — 1984. — № 17. — P. 169–174. |
| 46. | Adaptation in Natural and Artificial Systems [Text] / J. H. Holland. — Ann Arbor : University of Michigan Press, 1975. |
| 47. | Genetic hybrids for the quadratic assignment problem [Text] // DIMACS Series in Discrete Mathematics and Theoretical Computer Science. — 1994. — № 16. — P. 173–187. |
| 48. | A greedy genetic algorithm for the quadratic assignment problem [Text] : working paper 3826-95 / R. K. Ahuja, J. B. Orlin, and A. Tivari. — Sloan School of Management, MIT, 1995. |
| 49. | QAPLIB - A Quadratic Assignment Problem Library [Електронний ресурс] / R.E. Burkard, E. Çela, S.E. Karisch, F. Rendl. — Режим доступу : http://www.seas.upenn.edu/qaplib/. — Загол. з екрану. — Мова : англ. |
| 50. | The ant system: optimization by a colony of cooperating agents [Text] / A. Colorni, M. Dorigo, V. Maniezzo // IEEE Transactions on Systems, Man, and Cybernetics. — 1996. — Vol. B, № 26. — P. 29–41. |
| 51. | The ant system applied to the quadratic assignment problem [Text] / A. Colorni, V. Maniezzo // IEEE Transactions on Knowledge and Data Engineering. — 1998. |
| 52. | of quadratic assignment test problems with known optimal solutions [Text] / G. S. Palubeckis // U.S.S.R. Comput. Maths. Math. Phys. — 1988. — № 28. — P. 97–98. |
| 53. | Generating quadratic assignment test problems with known optimal permutations [Text] / Y. Li, P. M. Pardalos // Computational Optimization and Applications. — 1992. — № 1. — P. 163–184. |
| 54. | Massively parallel tabu search for the quadratic assignment problem [Text] / J. Chakrapani, J. Skorin-Kapov // Annals of Operations Research. — 1993. — № 41. — P. 327–342. |
| 55. | An efficient implementation of the robust tabu search heuristic for sparse quadratic assignment problems [Text] / G. Paul // European Journal of Operational Research. — 2011. — № 209. — P. 215-218. |
| 56. | Зорін Ю.М., Подольський С.В. Табу-пошук для квадратичної задачі про призначення // IV наук.-техн. конф. «Прикладна математика та комп’ютинг». Тези доповідей. — К.: НТУУ «КПІ», 12 квітня 2012. |
| 57. | Зорін Ю.М., Подольський С.В. Алгоритм перетину локальних оптимумів для оптимізації задачі про квадратичні призначення // ІІІ наук.-техн. конф. «Прикладна математика та комп’ютинг». Тези доповідей. — К.: НТУУ «КПІ», 13-15 квітня 2011. — С. 353 – 356. |
| 58. | Державні санітарні правила і норми роботи з візуальними дисплейними терміналами електронно-обчислювальних машин [Текст] : ДСанПіН 3.3.2.007-98 : затв. Постановою Головного державного санітарного лікаря України від 10.12.98 № 7. |
| 59. | Гігієнічна класифікація праці за показниками шкідливості та небезпечності факторів виробничого середовища, важкості та напруженості трудового процесу [Текст] : затв. наказом МОЗ України від 27.12.01 № 528). |
| 60. | Санiтарнi норми мiкроклiмату виробничих примiщень [Текст] : ДСН 3.3.6.042-99 : затв. Постановою Головного державного санітарного лікаря України від 01.12.99 № 42. |
| 61. | Санiтарнi норми виробничого шуму, ультразвуку та iнфразвуку [Текст] : ДСН 3.3.6.037.99 : затв. Постанова Головного Державного санітарного лікаря України від 01.12.99 № 37). |
| 62. | Державні будівельні норми. Інженерне обладнання будинків і споруд. Природне і штучне освітлення [Текст] : ДБН В.2.5-28-2006 : затв. наказом Міністерства будівництва, архітектури та житлово-комунального господарства України від 15.05.06 № 168. |
| 63. | Пожежна безпека на об’єктах будівництва [Текст] : ДБН В.1.1.7–2002 : затв. наказом Держбуду України від 03.12.02 № 88. |
| 64. | Перелік однотипних за призначенням об'єктів, які підлягають обладнанню автоматичними установками пожежогасіння та пожежної сигналізації [Текст] : НАПБ Б.06.004-2007 : затв. наказом Міністерства України з питань надзвичайних ситуацій та у справах захисту населення від наслідків Чорнобильської катастрофи від 22.08.05 № 161. |
| 65. | Типові норми належності вогнегасників [Текст] : затв. наказом Міністерства України з питань надзвичайних ситуацій та у справах захисту населення від наслідків Чорнобильської катастрофи від 02.04.04 № 151. |

x

# ДОДАТКИ

## Додаток А. Скрипт MATLAB спрощення формули (16)

|  |
| --- |
| clc  clear all  echo off    syms fii fjj fkk fij fji fik fki fjk fkj  syms dii djj dkk dij dji dik dki djk dkj      Rij = ...  (dik - djk) \* (fik - fjk) + ... % Missed g = k in delta ij  (dki - dkj) \* (fki - fkj) + ... % Missed g = k in delta ij  (dii - djj) \* (fii - fjj) + ... % Loopback  (dij - dji) \* (fij - fji); % Reverse flows direction    Rik = ...  (dij - dkj) \* (fki - fji) + ... % Missed g = j in delta ik  (dji - djk) \* (fik - fij) + ... % Missed g = j in delta ik  (dii - dkk) \* (fkk - fjj) + ... % Loopback  (dki - dik) \* (fjk - fkj); % Reverse flows direction    Rjk = ...  (dki - dji) \* (fij - fkj) + ... % Missed g = i in delta jk  (dik - dij) \* (fji - fjk) + ... % Missed g = i in delta jk  (djj - dkk) \* (fkk - fii) + ... % Loopback  (dkj - djk) \* (fik - fki); % Reverse flows direction    R\_ik = ...  (dij - dkj) \* (fkj - fij) + ... % Missed g = j in delta\* ik  (dji - djk) \* (fjk - fji) + ... % Missed g = j in delta\* ik  (dii - dkk) \* (fkk - fii) + ... % Loopback  (dik - dki) \* (fki - fik); % Reverse flows direction    R\_jk = ...  (dji - dki) \* (fki - fji) + ... % Missed g = i in delta\* jk  (dij - dik) \* (fik - fij) + ... % Missed g = i in delta\* jk  (djj - dkk) \* (fkk - fjj) + ... % Loopback  (djk - dkj) \* (fkj - fjk); % Reverse flows direction      x = - Rik - Rjk + Rij + R\_ik + R\_jk;    simplify(x) |

Після запуску даного скрипта на виході в пакеті MATLAB R2011b буде отримано такий результат:

|  |
| --- |
| ans =  -(dij - dik - dji + djk + dki - dkj)\*(fij - fik - fji + fjk + fki - fkj) |

## Додаток Б. Програмна реалізація табу-пошуку

|  |
| --- |
| #pragma once    #include "abstract\_qap.h"      template <typename T>  struct MyTabuQap : Qap<T>  {  //////////////////////////////////////////////////////////////////////////  // Compute cost delta caused by two elements swap  //////////////////////////////////////////////////////////////////////////  inline T computeDelta(const vector<int> & solution, const int i, const int j)  {  const auto  si = solution[i],  sj = solution[j];    T delta =  (distances[i][i] - distances[j][j]) \* (flows[sj][sj] - flows[si][si]) + // Loopback  (distances[i][j] - distances[j][i]) \* (flows[sj][si] - flows[si][sj]); // Reverse flows directions    for (size\_t k = 0; k < size; k++)  if (k != i && k != j)  {  const register auto sk = solution[k];  delta +=  (distances[k][i] - distances[k][j]) \* (flows[sk][sj] - flows[sk][si]) +  (distances[i][k] - distances[j][k]) \* (flows[sj][sk] - flows[si][sk]);  }  return delta;  }        //////////////////////////////////////////////////////////////////////////  // ij-swap corrective after pq-swap performed, or  // pq-swap corrective after ij-swap performed,  // i, j, p, q  are distinct indices  //////////////////////////////////////////////////////////////////////////  inline T computeDeltaCorrective(vector<int> & solution, const int i, const int j, const int p, const int q)  {  const auto  si = solution[i],  sj = solution[j],  sp = solution[p],  sq = solution[q];    return  (distances[p][i] - distances[p][j] + distances[q][j] - distances[q][i]) \*  (flows[sp][sj] - flows[sp][si] + flows[sq][si] - flows[sq][sj])  +  (distances[i][p] - distances[j][p] + distances[j][q] - distances[i][q]) \*  (flows[sj][sp] - flows[si][sp] + flows[si][sq] - flows[sj][sq]);  }        //////////////////////////////////////////////////////////////////////////  // Update deltas of ik-swap and jk-swap after some ij swap is performed  // Assuming delta of ij-swap is not updated yet  //////////////////////////////////////////////////////////////////////////  inline void updateJointDeltas(  const vector<int> & solution, // Solution vector  const int i, const int j, const int k, // Three distinct indices  const T delta\_ij, T & delta\_ik, T & delta\_jk) // References to delta matrix elements to be updated  {  const auto  si = solution[i],  sj = solution[j],  sk = solution[k];    const T new\_delta\_ik = computeDelta(solution, i, k);    delta\_jk +=  delta\_ik - delta\_ij - new\_delta\_ik -  (distances[i][j] - distances[i][k] - distances[j][i] + distances[j][k] + distances[k][i] - distances[k][j]) \*  (flows[si][sj] - flows[si][sk] - flows[sj][si] + flows[sj][sk] + flows[sk][si] - flows[sk][sj]);    delta\_ik = new\_delta\_ik;  }        //////////////////////////////////////////////////////////////////////////  // Solve QAP  //////////////////////////////////////////////////////////////////////////  Result<T> solve(  const T costThreshold = 0, // Min solution cost threshold as stop criterion  const long numberOfIterations = LONG\_MAX, // Max number of algorithm iterations as stop criterion  const clock\_t timeLimit = numeric\_limits<clock\_t>::max()) // Max execution time limit (ms) after start as stop criterion  {  const long tabu\_duration = size;  const long aspiration = 2 \* size \* size;    clock\_t start\_clock = clock();  Result<T> best;  best.cost = numeric\_limits<T>::max();    // Construct initial random solution  vector<int> solution(size);  for (size\_t i = 0; i < size; i++)  solution[i] = i;  random\_shuffle(solution.begin(), solution.end());  best.solution = solution;    // Compute cost  T cost = 0;  for (size\_t i = 0; i < size; i++)  for (size\_t j = 0; j < size; j++)  cost += distances[i][j] \* flows[solution[i]][solution[j]];  best.cost = cost;    // Initialize deltas and tabu  vector< vector<T> > deltas(size, vector<T>(size));  vector< vector<long> > tabu(size, vector<long>(size));  for (size\_t i = 0; i < size - 1; i++)  for (size\_t j = i + 1; j < size; j++)  {  deltas[i][j] = computeDelta(solution, i, j);  tabu[i][j] = -long(size \* i + j);  }    /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  /\* Main loop \*/  /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  for (long iteration = 0; iteration < numberOfIterations && clock() - start\_clock < timeLimit; iteration++)  {  size\_t iMin, jMin;  T minDelta = numeric\_limits<T>::max();  bool already\_aspired = false;  const auto iterationMinusAspiration = iteration - aspiration;  const T bestCostMinusCost = best.cost - cost;    for (size\_t i = 0; i < size - 1; i++)  for (size\_t j = i + 1; j < size; j++)  {  bool aspired =  tabu[i][j] < iterationMinusAspiration ||  deltas[i][j] < bestCostMinusCost;    if (aspired && (!already\_aspired || already\_aspired && deltas[i][j] < minDelta) ||  !aspired && !already\_aspired && tabu[i][j] < iteration && deltas[i][j] < minDelta)  {  minDelta = deltas[iMin = i][jMin = j];  already\_aspired |= aspired;  }  }    // Perform swap  swap(solution[iMin], solution[jMin]);    // Update cost  cost += minDelta;    #pragma region Check if solution better than the best known  /\* best solution improved ?\*/  if (cost < best.cost)  {  best.cost = cost;  best.solution = solution;  best.iteration = iteration;  best.duration = clock() - start\_clock;  time(&best.timeStamp);    // Optimum reached  if (cost <= costThreshold)  break;    // Print result if there is no stop criterion specified  if (numberOfIterations == LONG\_MAX && costThreshold == 0 && timeLimit == numeric\_limits<clock\_t>::max())  cout << best << endl;  }  #pragma endregion    // Forbid reverse move for a random number of iterations but at least 1  tabu[iMin][jMin] = iteration + 1 + tabu\_duration \* rand() / RAND\_MAX;      // Update deltas  for (size\_t i = 0; i < size - 1; i++)  if (i != iMin && i != jMin)  for (size\_t j = i + 1; j < size; j++)  if (j != iMin && j != jMin)  deltas[i][j] += computeDeltaCorrective(solution, i, j, iMin, jMin);    for (size\_t k = 0; k < iMin; k++)  updateJointDeltas(solution, iMin, jMin, k, minDelta, deltas[k][iMin], deltas[k][jMin]);  for (size\_t k = iMin + 1; k < jMin; k++)  updateJointDeltas(solution, iMin, jMin, k, minDelta, deltas[iMin][k], deltas[k][jMin]);  for (size\_t k = jMin + 1; k < size; k++)  updateJointDeltas(solution, iMin, jMin, k, minDelta, deltas[iMin][k], deltas[jMin][k]);    deltas[iMin][jMin] \*= -1;  }  return best;  }  }; |

## Додаток В. Копії графічних матеріалів

Відповідність потоків відстаням до перестановки

|  |  |
| --- | --- |
|  |  |

Відповідність потоків відстаням після перестановки

|  |  |
| --- | --- |
|  |  |

Отримання формули корекції складності

|  |  |  |
| --- | --- | --- |
|  |  |  |

…

1

2

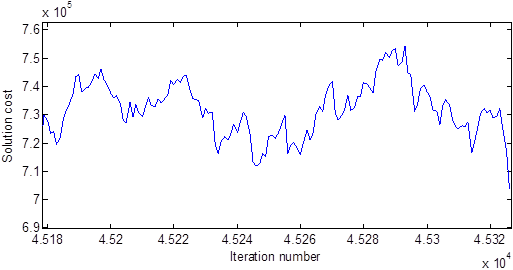
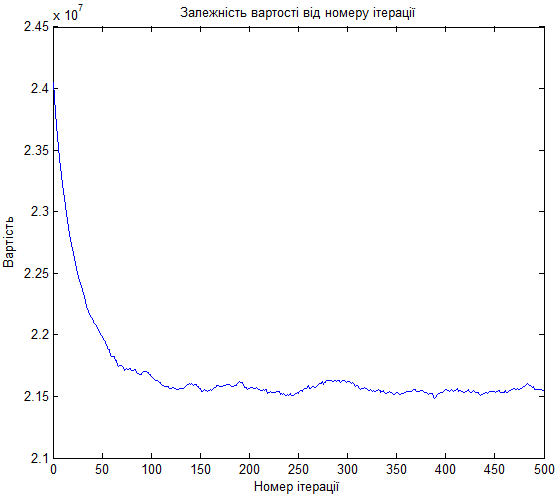
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **QAP Instance** | **Середній час пошуку глобального оптимуму (мс)** | | | | |
| **Simulated Annealing** | **Fast Ant System** | **Алгоритм перетину локальних оптимумів** | **Robust Tabu Search** | **Розроблений табу-пошук** |
| Tai20a | 3283 | 32238 | 2589 | 767 | **419** |
| Tai20b | 3565 | 56 | **18** | 46 | 23 |
| Tai25a | 9169 | — | 80825 | 1862 | **1409** |
| Tai25b | 12268 | 174 | 391 | 189 | **93** |
| Tai30b | 14421 | **816** | 4817 | 1905 | 1189 |
| Chr20a | 17976 | 12798 | **601** | 2270 | 2002 |
| Chr25a | 15378 | 2534 | **780** | 16014 | 13172 |
| Bur26a | — | 327 | **104** | 302 | 174 |
| Bur26b | — | **78** | 112 | 529 | 308 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **QAP Instance** | **Середнє значення вартості розв’язку** | | | | |
| **Simulated Annealing** | **Fast Ant System** | **Алгоритм перетину локальних оптимумів** | **Robust Tabu Search** | **Розроблений табу-пошук** |
| Tai20a | 719700 | 711814 | 710996 | 706297 | **705832** |
| Tai20b | 127963268 | 122571699 | **122455319** | 122480160 | **122455319** |
| Tai25a | 1211821 | 1191203 | 1191286 | 1177224 | **1175948** |
| Tai25b | 365033430 | 344532957 | 344626409 | 347301492 | **344424731** |
| Tai30b | 664031835 | **638359264** | 638876610 | 668590808 | 644680592 |
| Chr20a | 2850 | 2344 | 2262 | 2318 | **2261** |
| Chr25a | 6879 | 4283 | **4026** | 4331 | 4182 |
| Bur26a | 5475432 | 5428630 | **5427177** | 5431488 | 5428648 |
| Bur26b | 3848289 | 3818723 | **3817906** | 3822009 | 3819655 |

Залежність вартості розв’язку від часу роботи алгоритму табу-пошуку

|  |  |
| --- | --- |
|  |  |

Процес зміни вартості розв’язку



Visual Studio Profiling

|  |  |
| --- | --- |
| До застосування розробленого математичного апарату | Після застосування математичного розробленого апарату |
|  |  |