Міністерство освіти і науки України

Національний технічний університет

«Дніпровська політехніка»

Інститут електроенергетики

(інститут)

Факультет інформаційних технологій

(факультет)

Кафедра системного аналізу і управління

(повна назва)

ПОЯСНЮВАЛЬНА ЗАПИСКА

кваліфікаційної роботи ступеня бакалавра

(бакалавра, магістра)

студента Сазанської Ірини Олегівни

(ПІБ)

академічної групи 124-19-2

(шифр)

спеціальності 124 Системний аналіз

(код і назва спеціальності)

на тему: «Оптимізація процесу формування маршрутів перевезення різнорідних вантажів торгівельної компанії»

(назва за наказом ректора)

Керівники	**Прізвище, ініціали**	**Оцінка за шкалою**		**Підпис**
		рейтинговою	**інституційною**	
кваліфікаційної роботи	*к.ф.-м.н., доц. Коряшкіна Л.С.*			
розділів:				
Інформаційно-аналітичний	*к.ф.-м.н., доц. Коряшкіна Л.С.*			
Спеціальний	*к.ф.-м.н., доц. Коряшкіна Л.С.*			

Рецензент	*к.ф.-м.н., доц. ФІО*			

Нормоконтролер	*к.ф.-м.н., доц. Хом'як Т.В.*			

Дніпро
2023

ЗАТВЕРДЖЕНО:

завідувач кафедри

Системного аналізу та управління
(повна назва)

_____ *к.т.н., доц. Т.А. Желдак*
(підпис) (прізвище, ініціали)

«_____»_____2023 року

ЗАВДАННЯ

на кваліфікаційну роботу

ступеня бакалавра
(бакалавра, магістра)

студенту Сазанській Ірині Олегівні **академічної групи** 124-19-2
(прізвище та ініціали) (шифр)

спеціальності 124 Системний аналіз

спеціалізації _____

на тему «Оптимізація процесу формування маршрутів перевезення різнорідних вантажів торгівельної компанії»
затверджену наказом ректора НТУ «Дніпровська політехніка» від 16.05.2023 № 350-с

Розділ	Зміст	Термін виконання
Інформаційно-аналітичний розділ	Визначення поняття транспортних задач і класифікації методів їх оптимізації, розгляд алгоритму імітації відпалу.	01.05.2023
Спеціальний розділ	Реалізація імітації відпалу для ефективного планування маршрутів перевезень продукції торгівельної компанії між складом та магазинами з попитом.	31.05.2023

Завдання видано _____ Л.С. Коряшкіна
(підпис керівника) (прізвище, ініціали)

Дата видачі: 18.03.2023 р.

Дата подання до екзаменаційної комісії: 13.06.2023 р.

Прийнято до виконання _____ І.О. Сазанська
(підпис студента) (прізвище, ініціали)

РЕФЕРАТ

Пояснювальна записка: 59 с., 22 рис., 2 табл., 3 додатків, 14 джерел.

Пошук кращого маршруту перевезень в умовах асиметричності транспортної задачі є дуже важливим для спеціалістів галузі оптимізації та управління. Застосування сучасних метаевристичних алгоритмів оптимізації є важливим аспектом цього процесу, оскільки допомагає забезпечити ефективне планування маршрутів з врахуванням нерівномірності витрат часу, відстаней та інших факторів на різних маршрутах.

Об'єкт дослідження: оптимізація маршрутів розвезення товарів торгівельної компанії зі складу до магазинів.

Предмет дослідження: метаевристичний алгорим локального пошуку, відомий як імітація відпалу.

Мета дослідження: отримання субоптимального маршруту перевезень в умовах асиметричної задачі маршрутизації ємного транспорту.

Методи дослідження та апаратура: пояснення методу застосованого до проблеми оптимізацій процесу формування маршрутів доставки; мова програмування Python.

В *інформаційно-аналітичному розділі* наведені означення транспортних задач і класифікація методів їх оптимізації, розглянуто алгоритм імітації відпалу.

У *спеціальному розділі* був обраний метаевристичний метод локального пошуку – імітація відпалу, для ефективного планування маршрутів перевезень продукції торгівельної компанії між складом та магазинами з попитом на товари.

Практична цінність отриманих у роботі результатів полягає у формуванні оптимального маршруту з урахуванням наданих умов, що сприятиме поліпшенню основних показників діяльності торгівельної компанії і приноситиме додатковий прибуток.

Ключові слова: ЛОКАЛЬНИЙ ПОШУК, АСИМЕТРИЧНА ТРАНСПОРТНА ЗАДАЧА, ІМІТАЦІЯ ВІДПАЛУ, ГРАФІК ОХОЛОДЖЕННЯ.

ABSTRACT

Explanatory note: 59 p., 22 pictures, 2 tables, 3 appendixes, 14 sources.

The search for an optimal transportation route in the context of asymmetric transportation problem is crucial for professionals in the field of optimization and management. The application of modern metaheuristic optimization algorithms is an important aspect of this process, as it facilitates efficient route planning considering uneven time costs, distances, and other factors across different routes.

Object of research: optimization of delivery routes for a trading company from the warehouse to the stores.

Subject of research: metaheuristic algorithm of local search known as simulated annealing.

The purpose of the research: obtaining a suboptimal transportation route in the conditions of an asymmetric capacitated vehicle routing problem.

Research methods and equipment: explanation of the applied method for optimizing the delivery route formation process; programming language Python.

The *information-analytical section* provides definitions of transportation problems and the classification of methods of their optimization, the concept of the simulated annealing algorithm.

In a *special section*, the metaheuristic method of local search, simulated annealing, was chosen for efficient planning of product transportation routes between the warehouse and the stores based on their respective demands.

The *practical value* of the obtained results in this study lies in the formation of an optimal route considering the given conditions, which will contribute to improving the key performance indicators of the trading company and generating additional profit.

Keywords: LOCAL SEARCH, ASYMMETRICAL VEHICLE ROUTING PROBLEM, SIMULATED ANNEALING, COOLING SCHEDULE.

ЗМІСТ

ВСТУП

Оптимізація маршрутів транспортних засобів являється ключовою складовою в організації ефективної транспортної логістики, оскільки саме транспортні перевезення мають значний вплив на використання ресурсів.

Актуальність оптимізації задач маршрутизації транспортних засобів відображається в її значущості для підвищення ефективності, зниження витрат та покращення якості обслуговування.

Вирішення задач маршрутизації є складним завданням, оскільки вимагає подолання технічних та організаційних викликів. Зокрема, оптимальна маршрутизація забезпечує зменшення витрат на паливо, підтримку транспортних засобів та робочу силу, що стає особливо актуальним в контексті зростаючих цін на паливо та обмежених фінансових ресурсів. Додатково, ефективна маршрутизація сприяє зменшенню часу на доставку товарів або перевезення пасажирів, впливаючи на загальну продуктивність та ефективність організації.

Задача маршрутизації має комбінаторний характер, вимагаючи використання ефективних алгоритмів та оптимізаційних методів. Оптимальна маршрутизація сприяє належному використанню інфраструктури, забезпечуючи оптимальне розподіл ресурсів та зменшення заторів.

Таким чином, оптимізація задач маршрутизації транспортних засобів виступає як важлива складова управління транспортними системами, сприяючи підвищенню ефективності та сталості в цій сфері.

Мета данної роботи полягає у вдосконаленні транспортного процесу та підвищенні ефективності використання рухомого складу шляхом розробки та реалізації оптимальних маршрутів для транспортних засобів з урахуванням асиметричності, тобто нерівномірності відстаней на різних маршрутах. Очікується, що використання методу імітації відпалу дозволить знайти близько до оптимального рішення, яке дозволить уникнути нераціональних перевезень та забезпечить покращення основних показників роботи торгівельної компанії і приноситиме додатковий прибуток.

1 ІНФОРМАЦІЙНО-АНАЛІТИЧНИЙ РОЗДІЛ

1.1 Загальні відомості задачі маршрутизації транспортних засобів

Проблема маршрутизації транспортного засобу (МТЗ) — це одна з ключових комбінаторних оптимізаційних задач [1] у сфері логістики та задач цілочисельного програмування, що базуються на побудові оптимальних маршрутів для транспортних засобів, які мають об’їхати усі пункти, починаючи й закінчувати одним й тим же місцем, для ефективного розподілу ресурсів існуючого автопарку та задоволення попиту клієнтів на доставку товарів або надання послуг.

Першими, хто почав вирішувати задачі у цій області вважаються Г. Данциг і Дж. Рамсер, які, у своїй статті в 1959 році запропонували математичну постановку задачі та алгоритмічний підхід до вирішення практичної задачі поставки бензину від кінцевої станції магістрального трубопроводу до великої кількості обслуговуючих терміналів (задача диспетчеризації вантажівок). Через кілька років інші дослідники включили більше одного транспортного засобу в постановку задачі, а також запропонували більш ефективну евристику на основі жадного алгоритму. З тих пір було запропоновано велику кількість моделей та алгоритмів, присвячених пошуку точного і наближеного рішення багатьох варіантів даної задачі [2]. Згодом, ці задачі були об'єднані в загальну групу задач маршрутизації транспорту, що отримали назву VRP (Vehicle Routing Problem).

Усі задачи, що стосуються проблеми маршрутизації транспортного засобу мають власну термінологію, яка відрізняється від задач, що відносяться до класу «комівояжер».

Оскільки МТЗ напряму пов’язано із транспортною логістикою, в основі використовуються наступні терміни: «транспортний засіб» (vehicle), замість терміну «шлях» – термін «маршрут» (route), замість терміну «місто» – термін «клієнт» (customer). Більш того, для початкового «міста» вводиться додатковий

термін – «депо», що позначає місце, де повинні починатися і закінчуватися маршрути транспортних засобів. У реальних умовах депо може являти собою транспортне підприємство, пункт виробництва, склад тощо.

Пізніше, у 1964 році Кларк і Райт провели удосконалення методу Данціга і Рамсера, використовуючи ефективний жадібний алгоритм, також відомий як алгоритмом заощаджень. Визначення оптимального рішення для задач МТЗ є NP-складною [1], тому саме через це розмір проблем, які можна оптимально вирішити за допомогою математичного програмування або комбінаторної оптимізації, може бути обмеженим, отже, виходячи з цього, можна зрозуміти чому, як правило, в основному використовують евристичні методи для розв’язування такого типу задачі.

Основна ідея проблеми маршрутизації транспортного засобу полягає в тому, щоб оптимально розподілити певну кількість транспортних засобів, які базуються в центральному депо, для обслуговування заданого набору клієнтських локацій з врахуванням обмежень щодо маршрутування та обсягу завантаження.

Основна мета транспортної задачі полягає у плануванні оптимального маршруту, який опирається від мінімізації витрат ресурсу часу, а також фінансових ресурсів.

Зазвичай проблема маршрутизації транспортного засобу стає актуальним питанням у будь-якої компанії, яка займається перевезенням вантажу. Поняття задачі МТЗ являє собою питання: «Як транспортувати товар з одного, або кількох основних складів до певної групи клієнтів, за умовою, що є певний автопарк машин із заданими характеристиками, а водії можуть пересуватися лише певною мережею доріг».

1.1.1 Задача маршрутизації транспортного засобу

Класична задача маршрутизації транспортного засобу VRP відноситься до комбінаторної оптимізації, в якій для парку однотипних транспортних засобів

необхідно сформувати оптимальний машрут, який надає водію можливість проїхати усіх клієнтів починаючи та закінчуючи свій шлях єдиним депо при умові, що даний маршрут задовольнит усі поставлені вимоги.

Зазвичай критерій оптимальності може виражатися будь-яким фактором, але найчастіше за все він відповідає довжині маршруту або часовими витратами.

Як правило, план маршруту можна описати за допомогою графа, що зображений на рис. 1.1, у якому зображено ілюстративний приклад побудови маршрутів для парку з чотирьох транспортних засобів з метою мінімізації сумарної довжини маршрутів. Даний приклад відображає класичну задачу маршрутизації транспорту. У центрі зображено депо, де знаходиться парк з декількох однотипних транспортних засобів. На певній відстані від депо розташовані клієнти, яких необхідно відвідати. Відстані між усіма пунктами вважаються відомими.

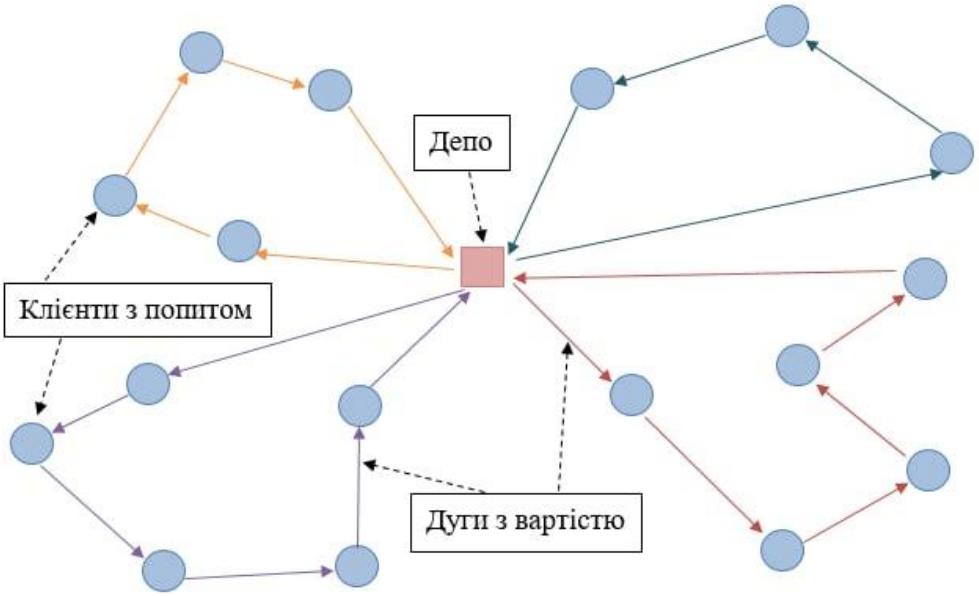

Рис. 1.1. Ілюстрація задачі маршруту транспорного засобу

Оптимальне рішення полягає у визначенні набору найкоротших маршрутів для кожного транспортного засобу, які пролягають через усіх клієнтів та завершуються поверненням у депо.

Класична VRP може бути представлена на графі $G = (V, E)$ з множиною вершин $V = \{0, n\}$ і множиною ребер E. Вершини $v \in V\backslash\{0\}$ відповідають клієнтам, тоді як вершина 0 відповідає депо. Також задається матриця n x n

невід'ємних вартостей c_{ij}, пов'язаних з кожним ребром (i, j) ϵ E і визначають витрати на пересування між вершинами i та j.

Отже, як наслідок, рішення класичної VRP зводиться до побудови декількох гамільтонових циклів мінімальної довжини на підграфі графу G з однією спільною вершиною–депо. При відсутності обмеження, що стосуються вантажопідйомності, а також довільній кількості транспортних засобів k завдання зводиться до побудови k циклів із загальною вершиною 0, які в сукупності містять всі вершини графа і мінімізують суму вартостей об'їзду клієнтів.

Тобто є маршрут, транспортний засіб, є депо, з якого починається й яким закінчується шлях, основна задача водія виконати усі вимоги клієнтів, задовольнити операційні обмеження, а також виконати усе протягом заданого часу за мінімальні витрати.

Це дозволяє забезпечити ефективне використання ресурсів, знизити витрати на транспортування та оптимізувати логістичні операції.

Дуги, можуть бути спрямовані через певний перелік доріг, оскільки слід при побудові маршруту пам’ятати, що кожна дорога, вулиця та інше мають свої обмеження. Також беремо до уваги те, кожна дуга має пов’язану вартість, якою зазвичай є її довжина або час у дорозі, який може залежати від типу та характеристик транспортного засобу. Для визначення загальної вартості кожного маршруту, необхідно знати вартість подорожі, також час у дорозі між кожним клієнтом і депо.

У результаті граф перетворюється на повний маршрут, де вершинами є клієнти та депо, а дугами є дороги між ними. Вартість на кожної дуги є найнижчою вартістю між двома точками початкової мережі доріг.

1.1.2 Задача маршрутизації ємного транспортного засобу

Задача маршрутизації ємного транспортного засобу (CVRP) є добре відомою задачею комбінаторної оптимізації в області дослідження операцій і

логістики, що відрізняється від класичної VRP задачі тим, що з’являється додаткове обмеження, що стосується введеням фіксованою вантажопідйомністю для кожного транспортного засобу. Зазвичай автопарк містить у собі однакові транспортні засоби, отже такого типу задачі називаються «CVRP однорідного парку».

Проблема передбачає визначення оптимального набору маршрутів для транспортних засобів для обслуговування набору клієнтів або місць, дотримуючись при цьому обмеження пропускної здатності кожного транспортного засобу. Мета полягає в тому, щоб мінімізувати загальну відстань або загальну вартість транспортування.

CVRP є надзвичайно актуальною проблемою в різних сценаріях реального світу, таких як системи доставки та розподілу, збір відходів, громадський транспорт і планування логістики. Це зазвичай зустрічається в галузях, де товари чи послуги потрібно постачати ефективно та результативно.

Відомо, що CVRP є NP-складною проблемою, а це означає, що пошук точного оптимального рішення для великих екземплярів стає обчислювально складним. Тому були розроблені різноманітні евристичні та метаевристичні алгоритми для пошуку якісних наближених розв’язків протягом розумного періоду часу. Ці алгоритми включають генетичні алгоритми, симуляцію відпалу, пошук табу та оптимізацію колонії мурашок, серед іншого.

Розв’язання CVRP вимагає врахування таких факторів, як обмеження місткості транспортного засобу, вимоги клієнтів, матриця відстані чи часу між місцями та цільова функція для оптимізації. Рішення зазвичай складається з набору маршрутів із зазначенням послідовності відвідування клієнтів кожним транспортним засобом разом із відповідним навантаженням на кожній зупинці.

Ефективне вирішення CVRP може призвести до значної економії коштів, кращого використання ресурсів, скорочення часу транспортування та підвищення задоволеності клієнтів. Це залишається активною сферою досліджень із постійними розробками алгоритмічних методів і варіантів конкретних проблем для вирішення складних умов реального світу та динамічних сценаріїв.

CVRP моделюється зв'язним графом $G = (V, E)$, тоді як $V = \{v_0, v_1, \ldots, v_n\}$ є набором вершин, а $E = \{(v_i, v_j) \mid v_i, v_j \in V, v_i \neq v_j\}$ – множина ребер. Вершина 0 зазвичай розглядається як депо, а вершини $1 \ldots n$ як клієнти. У депо є кілька транспортних засобів з однорідною місткістю Q. Кожне ребро (v_i, v_j), що з'єднує вершину i з вершиною j, має вагу, яка визначає відстань між ними. Кожен клієнт має q_i попит. CVRP визначає маршрут доставки товарів зі складу всім клієнтам і назад до депо, кожен клієнт відвідується транспортним засобом лише один раз, і кількість запитів клієнтів на маршруті не повинна перевищувати місткість транспортного засобу.

Якщо загальний попит клієнтів на певному маршруті перевищує пропускну здатність транспортного засобу, для клієнта, що залишився, необхідно встановити новий маршрут. Метою CVRP є мінімізація загальної довжини маршруту (також загального часу в дорозі) і кількості транспортних засобів (також кількості маршрутів, оскільки кожен маршрут стосується певного транспортного засобу).

1.1.3 Асиметрична транспортна задача

Проблема асиметричного маршруту транспортного засобу (AVRP) є варіантом класичної задачі маршруту транспортного засобу (VRP), де час подорожі або відстані між місцями є асиметричними, тобто час подорожі або відстань від місця A до B можуть відрізнятися від шляху $B \rightarrow A$ [3].

У загальному випадку, асиметрична задача відрізняється від симетричною тим, що вона моделюється орієнтованим графом (рис. 1.2). Відповідно, матриця вартостей є асиметричною. Вартість переходу від і-го клієнта до j-го клієнта відрізняється від переходу від j-го клієнту до і-го клієнта.

Також існує варіант задачі асиметричного маршруту транспортного засобу, у якій вводиться обмеження на ємність ТЗ (ACVRP). У результаті чого, основною особливістю даного типу задач являється те, що вводиться водночас обмеження

на максимально допустиму ємність ТЗ, а також об’єктивна оцінка часу та довжини побудованого маршруту.

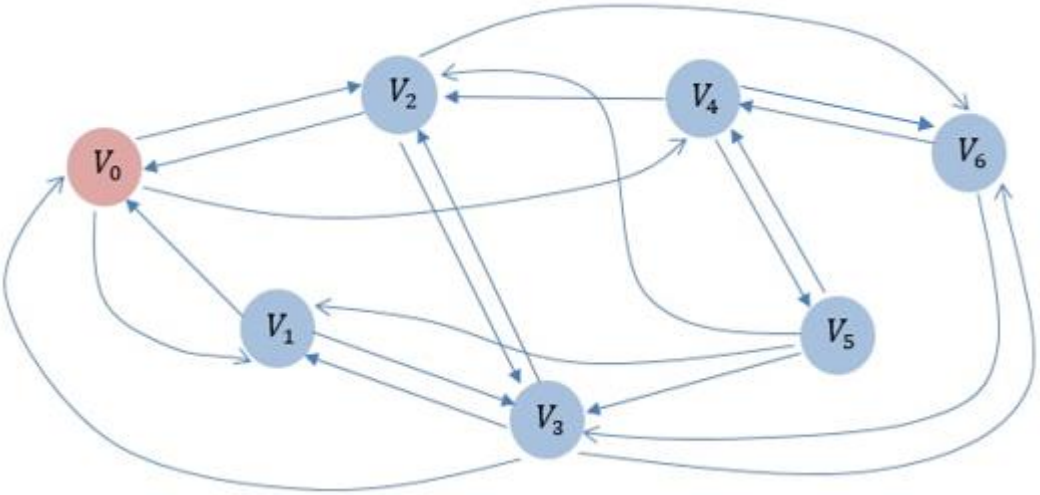

Рис. 1.2. Ілюстрація орієнтованого графу транспортної задачі

У більшості наукових робіт, присвячених дослідженню VRP, розглядається симетричний варіант завдання, що передбачає симетричність матриці вартостей. Таке припущення часто не співвідноситься з реальними умовами і веде до розриву між теорією і практикою: найкоротший шлях між двома точками дорожньої мережі, як правило, відрізняється в залежності від напрямку.

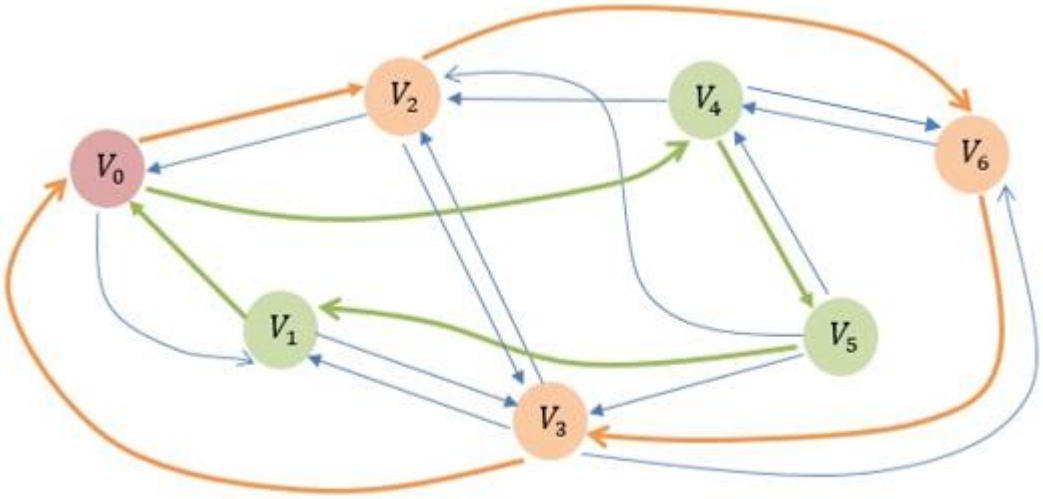

Рис. 1.3. Ілюстрація маршрутів транспорних засобів на орієнтованому графі

Асиметрія часу або відстані в дорозі в AVRP та ACVRP відображає реальні сценарії, коли транспортні мережі можуть мати різні дорожні умови, схеми руху або інші фактори, які спричиняють варіації часу в дорозі або відстані в різних напрямках. Це робить проблему складнішою, ніж симетричний VRP, оскільки

рішення щодо оптимального маршруту повинні враховувати ці асиметричні характеристики подорожі.

Розв’язання задач даного типу передбачає пошук ефективного розподілу клієнтів по транспортним засобам і визначення послідовності візитів для кожного транспортного засобу з урахуванням асиметричного часу подорожі або відстані (рис. 1.3). Різноманітні алгоритми та методики оптимізації, такі як евристичні підходи, метаевристики та точні алгоритми, використовуються для пошуку якісних рішень за розумний обчислювальний час.

AVRP та ACVRP має застосування в різних сферах, включаючи управління логістикою та транспортуванням, служби доставки, маршрутизацію автопарку та планування реагування на надзвичайні ситуації. Завдяки розв’язанню даних задач організації можуть оптимізувати маршрутизацію транспортних засобів, зменшити витрати на поїздки, покращити обслуговування клієнтів і підвищити загальну ефективність роботи.

1.2 Огляд та класифікація методів вирішення транспортних задач

Оскільки VRP відноситься до задач комбінаторної оптимізації, в якій число допустимих маршрутів експоненційно зростає, що у результаті при збільшенні числа клієнтів, і належить до класу складності NP. Для отримання розв’язку для таких задач, у яких розмірність більше 14, точні методи втрачають свою ефективність, оскільки перебір можливих розв’язків для знаходження оптимуму займає колосального часу обчислень навіть при відносно невеликій розмірності задачі. До точних методів відносяться такі методи, як метод гілок та меж, метод гілок та відсікань, динамічне програмування. Вони є розвитком методу повного перебору з відсівом підмножин допустимих розв’язків, що завідомо не містять оптимальних розв’язків. Однак час їх обчислень все одно зростає занадто швидко, а в гіршому випадку - аналогічно повному перебору.

В умовах реальних прикладів задач застосовуються наближені алгоритми. Взагалі, існує багато методів розв’язання VRP, значна частина яких це евристичні та метаевристичні методи, що пропонують кращий баланс між оптимальністю розв’язку та часом роботи. Мета метаевристике полягає у прискоринні розв’язання задачі, в обробці більшої кількості вузлів та отримати надійні алгоритми. Метаевристика забезпечує прийнятні розв’язки в адекватні терміни для розв’язання важких та комплексних задач.

Формування початкового розв'язку в задачі маршрутизації транспортного засобу (Vehicle Routing Problem, VRP) включає створення одного або кількох початкових розв'язків, які виступатимуть в якості початкових точок для подальшого покращення. Цей етап зазвичай використовує евристичні методи, які починають з порожнього розв'язку і поступово додають компоненти, поки не буде отримано допустимий розв'язок. Зокрема, часто застосовується жадібний алгоритм, який використовує різні евристичні функції.

Процес покращення розв'язку має на меті модифікувати початковий допустимий розв'язок, базуючись на певній метриці, досягнутий оптимальний розв'язок або закінчити роботу алгоритму відведеним часом. Крім того, використовується поняття "множина розв'язків", яка представляє собою множину допустимих розв'язків, досяжних з поточного стану. Метаевристичні алгоритми змінюють поточний стан на один зі станів з множини розв'язків, щоб знайти кращий розв'язок. Цей процес відповідає локальному пошуку в просторі станів. Локальний пошук використовує локальні операції, що змінюють один або кілька маршрутів, спрямовані на отримання кращого або ефективнішого розв'язку.

Таким чином, формування початкового розв'язку та його подальше покращення є важливими кроками в розв'язанні задачі маршрутизації транспортного засобу. Використовуючи евристичні методи та метаевристичні алгоритми, можна знаходити допустимі та оптимальні розв'язки, що дозволяє покращувати ефективність маршрутизації транспорту та знижувати витрати.

На рис. 1.4 зображена класифікаційна схема алгоритмів вирішення транспортних задач [4].

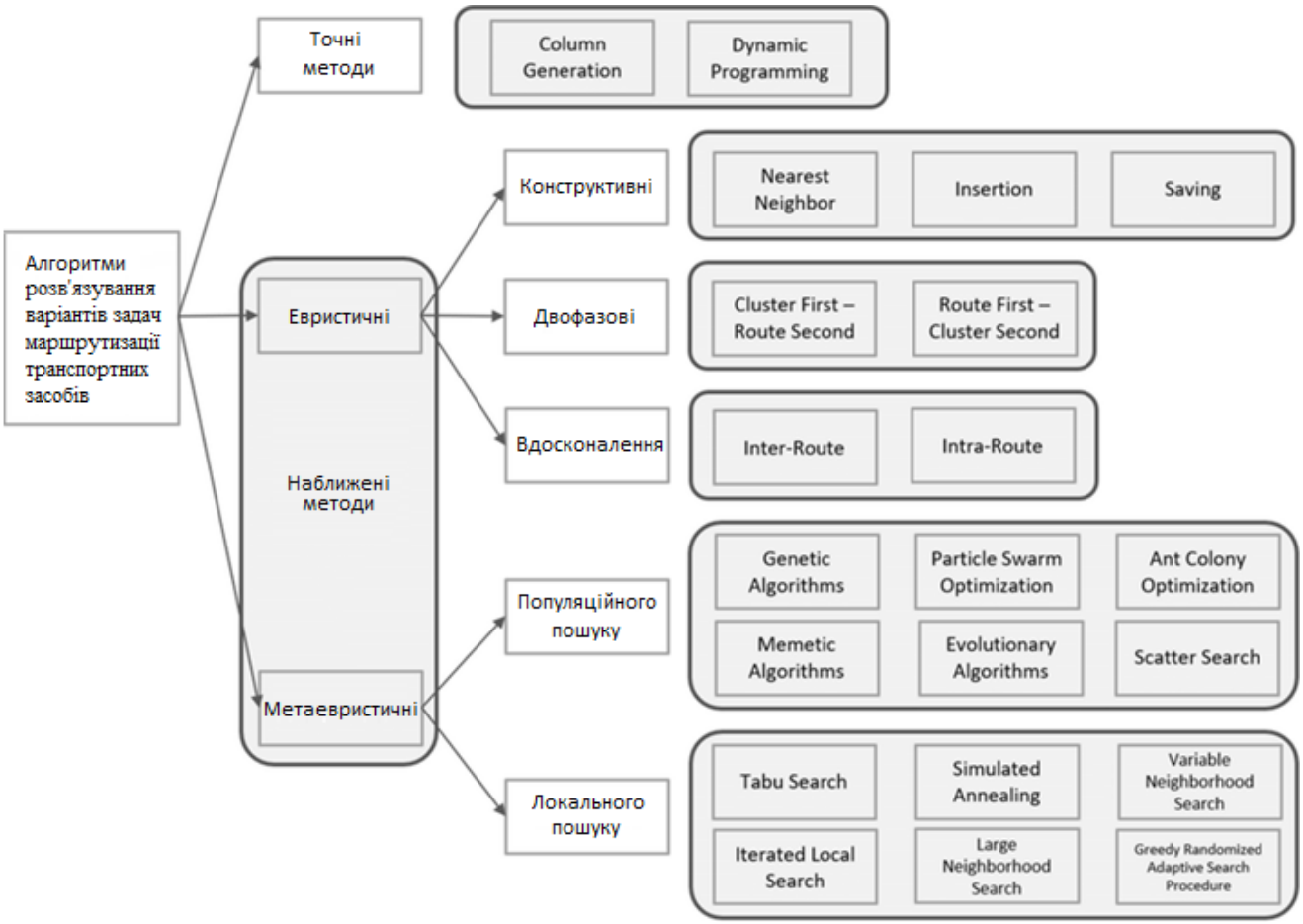

Рис. 1.4. Класифікаційна схема алгоритмів вирішення транспортних задач

1.2.1 Точні алгоритми рішення транспортних задач

Точні методи в оптимізації гарантують отримання оптимальних розв'язків шляхом математичного моделювання та використання точних алгоритмів. Ці методи ґрунтуються на математичних теоріях та алгоритмах, які дозволяють знайти розв'язки з найкращою можливою якістю з точки зору заданої метрики.

Один з основних точних методів - це повний перебір, який передбачає перевірку всіх можливих комбінацій розв'язків для вибору найкращого. Однак, час, необхідний для повного перебору, залежить від розмірності задачі, і для великих задач він стає неприйнятно великим. Часова складність точних методів зазвичай зростає експоненційно зі збільшенням розмірності задачі.

Інші точні методи включають метод гілок і меж (Branch and Bound), динамічне програмування (Dynamic Programming), лінійне програмування (Linear

Programming) та інші. Кожен з цих методів має свої особливості і використовується залежно від типу задачі та її характеристик.

Хоча точні методи гарантують оптимальні розв'язки, їх практичне застосування обмежене задачами невеликої розмірності через значні обчислювальні витрати. Великі задачі потребують великої обчислювальної потужності та часу, що часто виходить за межі практичних можливостей.

Точні методи широко використовуються для вирішення задач з малою розмірністю, де можливо виконати повний перебір або використати інші точні алгоритми для гарантованого отримання найкращих розв'язків. Для великих задач, де точні методи не ефективні, застосовуються наближені або метаевристичні методи, які забезпечують прийнятну якість розв'язків за прийнятний час.

1.2.2 Евристичні алгоритми рішення транспортних задач

Евристичні методи є набором прийомів у пошуку розв'язків задачі, які дозволяють обмежити перебір і знаходити прийнятні розв'язки в прийнятний час. Ці методи використовуються для вирішення задач, які вважаються важкими або нерозв'язними за допомогою точних методів.

Одним з головних обмежень евристичних методів є їхній приблизний характер. Отриманий розв'язок може бути далеким від оптимального, але водночас відповідати вимогам задачі або бути достатньо близьким до оптимального. Евристичні методи дозволяють провести обмежений пошук у просторі розв'язків, використовуючи різні стратегії та прийоми для виявлення перспективних розв'язків.

Перевагою евристичних методів є їхня здатність знаходити прийнятні розв'язки задач великої розмірності в прийнятний час. Це особливо важливо для задач, що належать до класу NP-повних, де точні методи можуть бути недоцільними через їхню велику часову складність. Евристичні методи

дозволяють здійснювати ефективний пошук розв'язків, що наближаються до оптимального, за допомогою різноманітних стратегій, які базуються на експертних знаннях, локальних операціях, метаевристикаh або інших евристичних принципах.

Однак, слід пам'ятати, що розв'язки, отримані евристичними методами, не є гарантовано оптимальними. В деяких випадках можуть існувати кращі розв'язки, які не були знайдені через обмеженості методу. Проте, евристичні методи є потужним інструментом для знаходження прийнятних розв'язків у великих і складних задачах, де точні методи є неефективними або непридатними.

Евристичні алгоритми поділяються на три основні категорії.

Конструктивні алгоритми. Виконують поступову побудову розв’язку, відстежуючи зростання його вартості, але не мають фази подальшого поліпшення: Nearest Neighbour, Greedy, Savings, Insertion.

Двофазні (кластерні) алгоритми. Задача розбивається на дві операції - угруповання вершин для кожного майбутнього маршруту (кластеризація) і розв’язання TSP для кожної отриманої групи: (алгоритм Фішера і Джайкумара, алгоритм замітання (Sweep Algorithm), алгоритм пелюсток (Petal Algorithm), алгоритм Османа). У двофазних алгоритмах можлива наявність зворотного зв'язку між етапами розв’язку. (Petal, Sweep, Cluster first-route second, Rout first – cluster second).

Покращуючі алгоритми. Спочатку ведеться пошук деякого розв’язку, а потім робляться спроби обміну вершин (ребер) всередині кожного маршруту або між маршрутами: (евристики поліпшення багатьох маршрутів). (Inner route, Intra route, k-opt, 2-opt, 3-opt).

1.2.3 Метаевристичні алгоритми рішення транспортних задач

Термін «метаевристика» був вперше введений Фредом Гловером в 1986 році для позначення алгоритмічних схем вищого рівня, спрямованих на ефективне вивчення простору пошуку складних оптимізаційних задач. Метаевристики

описують великий основний підрозділ в стохастичній оптимізації, що містить у собі великі класи алгоритмів і методів, які в тій чи іншій мірі використовується випадковість для пошуку оптимального або субоптимального розв’язку складних задач.

Отже, метаевристика - це метод розв’язання обчислювальних задач шляхом комбінування існуючих процедур з відкритим інтерфейсом та закритою реалізацією, яке призводить до максимально ефективного розв’язку. Метаевристичний підхід застосовується для розв’язання задач, які не мають, або не потребують спеціально підбраного або реалізованого алгоритиму розв’язку задачі. Найчастіше метаевристика використовуються не тільки для розв’язку задач комбінаторної оптимізації, але також вони можуть бути застосовані для будь-якої іншої задачі за умовою, що її можна звести до розв’язання логічних рівнянь. Порівнянно з ітераційними та іншими традиційниим методами оптимізації метаевристика не надає стовідсоткової гарантії, що в глобальному масштабі оптимальний розв’язок може бути знайдено для деякого класу задач в силу стохастичності. Однак під час пошуку основуючись на великому наборі допустимих розв’язків метаевристики часто дозволяють знайти вдалий розв’язок з меншими обчислювальними, а також часовими витратами.

Виділяють наступні властивості метаевристик:

- метаевристика є стратегією, яка управляє процесом пошуку;
- метою метаевристики є ефективне дослідження простору пошуку для знаходження (суб) оптимального розв’язку;
- методи, які реалізуються метаевристичним алгоритмом, варіюються від простого локального пошуку до складного процесу навчання;
- метаевристичний алгоритм є наближеним і зазвичай недетермінованим;
- у метаевристики закладається механізм, що запобігає застряганню в локальному оптимумі;
- основні положення метаевристик допускають абстрактний опис;
- метаевристика може використовувати проблемно-орієнтоване знання в формі евристик, керованих високорівневою стратегією;

- передові метаевристики використовують досвід, накопичений в процесі пошуку і представлений у вигляді пам'яті, для управління пошуком.

Основна частина літератури по темі метаевристик має експериментальний характер, описуючи емпіричні результати, засновані на імітаційному моделюванні і програмному експерименті з алгоритмами. Однак існують також деякі формальні теоретичні результати, що обґрунтовують можливість знаходження глобального оптимуму.

1.3 Алгоритм імітації відпалу

Імітація відпалу (Simulated annealing, SA) – це відомий метаевристичний метод локального пошуку, який знайшов широке застосування в оптимізації, особливо для задач зі складною локальною структурою та багатьма локальними мінімумами. Його стохастичний підхід дозволяє вийти з локальних оптимумів та здійснювати глобальний пошук кращих рішень. Крім того, метод моделювання відпалу має математичну основу, що дозволяє розглядати його з точки зору теорії марковських процесів та аналізу його статистичних властивостей [5].

Алгоритм імітації відпалу [6] базується на аналогії фізичного процесу відпалу металу, де мінімізують енергію системи, використовуючи нагрівання та подальше повільне охолодження для досягнення більш стабільної структури атомів речовини.

Техніка повільного охолодження дозволяє атомам металу вибудовуватися і утворювати правильну кристалічну структуру, що має високу щільність і низьку енергію, перехід атома з одного осередку в інший відбувається з певною ймовірністю, причому ймовірність зменшується з пониженням температури. Початкова температура і швидкість зниження температури називається графіком відпалу.

У контексті оптимізації, в основі алгоритму імітації відпалу лежить ідея прийняття найгірших рішень з певною ймовірністю на початковому етапі, щоб уникнути передчасного застрягання в локальному оптимумі. З часом ця ймовірність поступово зменшується, що призводить до більш жорсткого вибору кращих рішень у міру просування алгоритму.

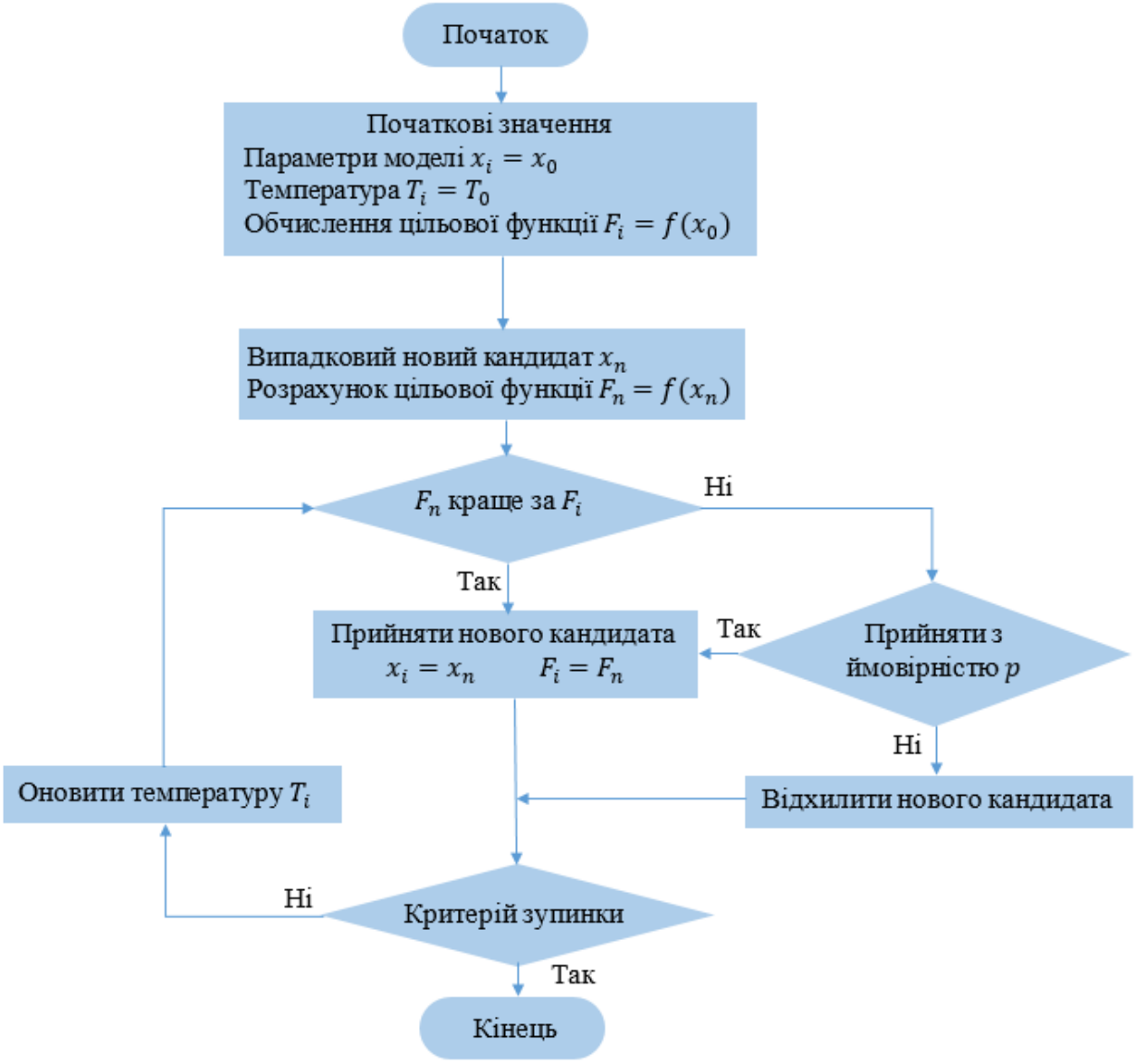

Рис. 1.5. Алгоритм «імітації відпалу»

SA є марковським процесом і характеризується властивістю відсутності пам'яті [7]. Це означає, що якщо процес запущений і працює достатньо довго, початкова конфігурація втрачається, і ймовірність знаходження певної конфігурації у заданому стані стає стаціонарною. Ця ймовірність залежить лише від значення функції, яка визначає якість рішення. Коли температура T дорівнює

нулю, ймовірність досягає свого максимуму тільки у глобально оптимальних конфігураціях. Цей основний результат, який випливає з оптимізаційного дослідження [8], викликав велике зацікавлення у вирішенні задач оптимізації за допомогою простого та універсального методу. Він знайшов застосування в різних галузях науки та технологій, і його потенціал досі вивчається та розширюється шляхом поєднання з іншими методами оптимізації та адаптації до специфічних вимог конкретних завдань.

Вирішуючи задачу комбінаторної оптимізації з використанням імітації відпалу, загальний алгоритм якого зображено на рис 1.5, слід починати з певного можливого розв'язку задачі. Наступний етап полягає в оптимізації цього розв'язку, використовуючи метод, що є аналогічний відпалу твердих речовин. Сусід цього розв'язку формується за допомогою відповідного методу, а також розраховується вартість або допустимість нового розв'язку.

У разі, якщо новий розв'язок виявляється більш оптимальним за поточний з точки зору зменшення витрат або підвищення придатності, то новий розв'язок приймається. Однак, у випадку, коли новий розв'язок не є кращим від поточного розв'язку, то новий розв'язок приймається лише з певною ймовірністю (1.1): експоненційного критерію прийняття рішень, що заснований на розподілі Гіббса [1].

$$p = e^{(-\Delta / T)}, \tag{1.1}$$

де $\Delta -$ це зміна вартості між поточним та новим розв'язком, T - поточна температура.

Таким чином, ймовірність прийняття нового рішення експоненційно зменшується в міру погіршення руху.

Процедура «імітації відпалу» має менше шансів застрягти в місцевому оптимумі через те, що погані шляхи все ще мають шанс бути здійсненими. Спочатку температуру відпалу слід виставити високою для того, щоб ймовірність прийняття також була високою, й тому приймаються майже всі нові розв`язки. Потім температуру поступово знижують, використовуючи графік охолодження [9], щоб у результаты ймовірність прийняття низькоякісних розв'язків буде дуже

малою: високі температури дозволяють краще досліджувати пошуковий простір, тоді як нижчі температури дозволяють регулювання кращого розв'язку. Процес повторюється до тих пір, поки температура не наблизиться до нуля або подальшого поліпшення не вдасться досягти (рис. 1.6).

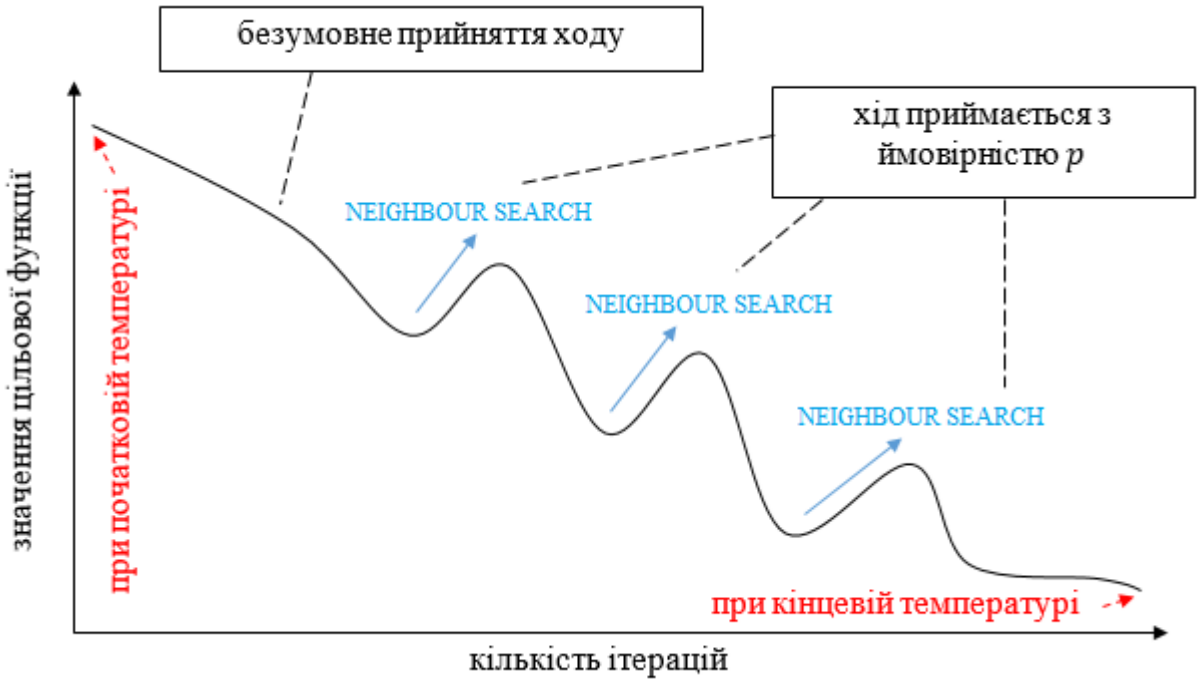

Рис. 1.6. Графік збіжності алгортиму імітації відпалу

1.4 Висновки

На сьогодні залишається актуальним питання оптимізації маршрутів транспортних засобів (VRP) для будь-якої компанії, яка пов'язана з перевезенням вантажу, товарів тощо. Основною метою оптимізації маршрутів полягає у тому, щоб певний транспортний засіб пройшов увесь маршрут за відповідний час, такий підхід також забезпечує мінімізацію витрат на амортизацію та знижує ризики псування стану вантажу з дотриманням усіх поставлених вимог. На основі даних принципів й будуються задачі маршрутизації транспортного засобу.

Для аналізу даних та знаходження оптимальних маршрутів використовують декілька різних методів, а саме: точні методи, евристичні та метаевристичні методи, які також включають у себе певний перелік алгоритмів для розв'язку

транспортних задач. Однак, саме такий різновид методів й вимагає від користувача детального вибору стосовно того, який метод слід обрати для вирішення поставленої задачі.

Точні методи, такі як математичне програмування або повний перебір, можуть бути ефективними для розв'язання CVRP на малих наборах даних або з обмеженими обмірковими ресурсами. Однак, для великих та складних наборів даних точні алгоритми не слід використовувати через їх високу обчислювальну складність, що може призвести до тривалого часу виконання та непрактичності в реальних умовах.

Метаевристичні алгоритми є ефективним вибором для вирішення завдань маршрутизації транспортних засобів завдяки своїй здатності шукати наближені оптимальні рішення у великих просторах пошуку, а також їх гнучкості та застосовності до різних умов та обмежень, що забезпечує швидкість та адаптивність при вирішенні реальних проблем маршрутизації.

Одним з таких методів є алгоритм «імітації відпалу», що використовується для ефективного розв’язання багатьох комбінаторних задач оптимізації. Переваги даного підходу включають здатність досліджувати простір рішень у широкому діапазоні, виявляти локально оптимальні рішення та мати гнучкі механізми для подолання локальних мінімумів при тому, що задачі маршрутизації трансопртних засобів мають велику кількість можливих рішень, і існує ризик застрягання в локальних мінімумах при використанні деяких точних або градієнтних методів.

2 СПЕЦІАЛЬНИЙ РОЗДІЛ

2.1 Постановка задачі

В основі даної кваліфікаційної роботи полягає вирішення наступної задачі:

Торгівельна компанія «Varus» має на меті провести оптимізацію загальної логістичної системи для підвищення загальних показників компанії, що у свою чергу надасть можливість збільшити прибутки, мінімізувати час на транспортування товарів, а також знизити відсоток псування вантажу.

Для досягання поставленої мети, «Varus» надав необхідні дані, що у свою чергу являють собою інформацію про загальну кількість магазинів, також координати кожного супермаркету та основного складу. Також була надана інформація про те, які машини використовуються та їхні характеристики.

У даній задачі було використано метод «імітації відпалу», який надає можливість провести повний аналіз вхідних даних. Результатом даного аналізу являється інформація стосовно того, яку необхідну кількість транспортних засобів слід долучити до транспортування, а також було побудовано оптимальний маршрут для кожного з них.

Інформація такого типу надає компанії можливість не тільки оптимізувати перевезення вантажу та постійного забезпечення супермаркетів необхідними товарами, а також дає змогу максимізувати загальний прибуток компанії та мінімізувати логістичні витрати.

2.2 Математична модель задачі

Нехай $G = (V, E) -$ повний орієнтований граф, де $V = \{0,1, \dots, n\} -$ множина вузлів, а $E -$ множина дуг. Вузли $i = 1,2, \dots, n$ відповідають клієнтам, кожен із детермінованим попитом $0 < d_i \leq Q$. Вузол $i = 0$ є складом з нульовим попитом

$d_0 = 0$. Кожен транспортний засіб має вантажлопідйомність $Q > 0$. Кількість транспортних засобів становить K.

Нехай $c_{ij} -$ невід'ємна вартість на основі відстані, що пов'язана з дугою $(i,j) \in E$ та $c_{ii} = +\infty, \forall i \in V$.

Нейхай $y_i^k -$ бінарна змінна, що вказує, чи відвідає транспортний засіб k вершину i (2.1), $x_{ijk} -$ бінарна змінна, що вказує, чи відвідується клієнт j одразу після i транспортним засобом k (2.2).

$$y_i^k = \begin{cases} 1, & \text{якщо вузол } i \text{ відвідується транспортним засобом } k \\ 0, & \text{в іншому випадку} \end{cases} \quad (2.1)$$

$$x_{ijk} = \begin{cases} 1, & \text{якщо вузол } j \text{ відвідується після } i \text{ транспортним засобом } k \\ 0, & \text{в іншому випадку} \end{cases} \quad (2.2)$$

Сформулюємо наступну математичну модель транспортної задачі, за якою у цільовій функції (2.3) мінімізується загальна залежна від відстані вартість дуг побудованих маршрутів усіх транспортних засобів:

$$\min \sum_{k=1}^{K} \sum_{i \in V} \sum_{j \in V} c_{ij} \cdot x_{ijk}, \#(2.3)$$

$$\sum_{k=1}^{K} \sum_{i \in V \setminus \{j\}} x_{ijk} = 1 \qquad \forall j \in V \setminus \{0\} \#(2.4)$$

$$\sum_{k=1}^{K} \sum_{j \in V \setminus \{i\}} x_{ijk} = 1 \qquad \forall i \in V \setminus \{0\} \#(2.5)$$

$$\sum_{i \in V \setminus \{0\}} x_{i0k} = \sum_{j \in V \setminus \{0\}} x_{0jk} \qquad \forall k = 1, \dots, K \#(2.6)$$

$$Q - \sum_{i \in V} d_i \cdot y_i^k \geq 0 \qquad \forall k = 1, \dots, K \#(2.7)$$

$$x_{ijk} \in \{0,1\} \qquad \forall (i,j) \in E \#(2.8)$$

$$y_i^k \in \{0,1\} \qquad \forall k, i \in V \#(2.9)$$

Обмеження (2.4) і (2.5) вимагають, щоб кожен клієнт був відвідан тільки одним транспортним рівно один раз.

Обмеження (2.6) передбачає, що кількість транспортних засобів, які залишають склад, така ж, як і кількість транспортних засобів, які повертаються до нього.

Обмеження (2.7) стверджує, що ємність транспортного засобу не повинна бути перевищена сумарному попиту клієнтів, яких відвідає ТЗ до того, як повернеться на склад.

Обмеження (2.8) і (2.9) вимагають, щоб невідомі змінні приймали лише бінарні значення.

2.3 Формування вхідних даних

Для формування вхідних даних для реалізованого алгоритму потрібно надати програмі два файли та ємність транспортних засобів, приклад зчитування яких для алгоритму відбувається за кодом у додатку В.

Перший файл містить матрицю відстаней, яка зчитується із файлу. Формат даних у файлі наступний: $n\ i_1\ j_1\ w_1\ i_2\ j_2\ w_2 \dots\ j_n\ i_n\ w_n$, де $n -$ кількість всіх вершин графа розв'язуваної задачі.

Далі перераховуються пари вершин (i, j) та їх відповідні відстані w_{ij}. Необхідно також описати пари (j, i) з відповідними відстанями. Якщо дороги між двома вершинами відсутні, слід упустити відповідну пару (i, j) або (j, i).

Другий файл містить послідовність нумерації вершин, описаних у матриці відстаней, а також кількість потреб магазину. Формат даних у файлі наступний: $0\ \ d_2\ \ d_3 \dots\ d_n$, де перелік починається з 0, який позначає склад, з якого вивозять продукцію до магазинів. За ним йдуть номери вершин, що відповідають магазинам, та попит кожного магазину: d_i, за послідовністю згадування його порядкового номеру у файлі з відстанями.

2.4 Оптимізація побудови маршрутів алгоритмом імітації відпалу

Для визначення початкового вирішення проблеми маршрутизації транспорту з місткістю транспортних засобів (CVRP) був реалізован евристичний конструктивний алгоритм найближчого сусіда (Nearest Neighbour). Цей метод полягає у виборі стартової вершини та послідовному додаванні найближчої доступної вершини до повного формування маршруту [10]. Алгоритм найближчого сусіда вимагає лише знаходження найближчої доступної вершини для кожної вершини маршруту і має низьку обчислювальну складність, що дає змогу швидко сформувати початковий маршрут без необхідності виконання складних оптимізацій.

Функцію сусідства – набір рухів, що застосовуються до поточного рішення для створення нового, було реалізовано 2-opt алгоритмом вдосконалення [11], щоб збалансувати дослідження простору пошуку з використанням хороших рішень. Нижче на рис. 2.1 наведено візуалізацію роботи алгоритму локального пошуку, який використовується для покращення поточного маршруту у задачах оптимізації маршрутів.

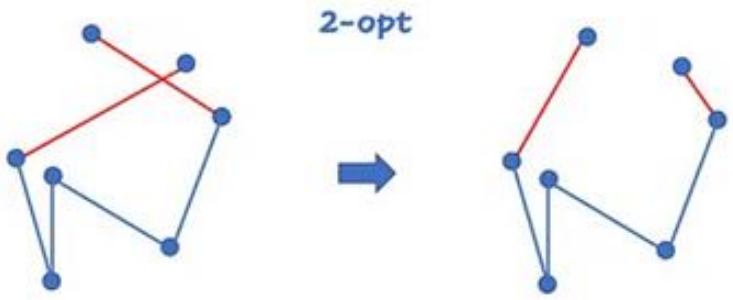

Рис. 2.1. Ілюстрація результату зміни рішення із залученням 2-opt алгоритму

Метод імітації відпалу є алгоритмічним підходом, що базується на математичній теорії марковських процесів. Він використовується для оптимізації і дослідження функцій шляхом стохастичної генерації траєкторій. Кожна наступна точка на траєкторії обирається залежно від поточної точки, і вибір виконується з використанням ймовірностей, що залежать тільки від поточної точки і не залежать від попередньої історії обчислень.

$$Y \leftarrow Neighbour\left(X^{(t)}\right)$$

$$X^{(t+1)} \leftarrow \begin{cases} Y, \text{ якщо } f(Y) \leq f(X^{(t)}) \\ Y, \text{ якщо } f(Y) > f\left(X^{(t)}\right), \text{з ймовірністю } p = e^{-\frac{f(Y)-f\left(X^{(t)}\right)}{T}} \\ X^{(t)}, \text{ якщо } f(Y) > f\left(X^{(t)}\right), \text{з ймовірністю } (1-p) \end{cases} \quad (2.10)$$

Алгоритм імітації відпалу використовує температурний параметр T для керування процесом прийняття рішень. Цей параметр визначає ймовірність прийняття рухів, що погіршують поточне рішення: висока значення T сприяє прийняттю більшого числа погіршень рухів, що в свою чергу призводить до більшої диверсифікації та пошуку рішень в широкому просторі. Правило (2.10) називається експоненційним правилом прийняття. Ймовірність прийняття стає рівною 1 при $T \to \infty$, незалежно від того, чи покращує цей хід рішення, що призводить до випадкового блукання. З іншого боку, при $T \to 0$, приймаються лише ходи, що покращують рішення, аналогічно до стандартного локального пошуку.

2.4.1 Порівняльний аналіз графіків охолодження

Як згадувалось раніше, локальний пошук зупиняється в локально оптимальній точці, що може призвести до пастки в строгому локальному мінімумі (рис. 2.2). У цій главі ми розглядаємо випадок з фіксованою структурою сусідства, але генерація та прийняття руху є стохастичними. Також, застосовується концепція "контрольованого погіршення" (2.10), яка дозволяє приймати рішення з гіршим значенням з метою виходу з локального атрактора.

В імітації відпалу графік охолодження (cooling schedule) визначає спосіб зміни температури системи у процесі пошуку оптимального рішення. Два основних типи графіків охолодження, які можуть бути використані в імітації відпалу, це монотонний (monotonic) та немонотонний (non-monotonic) графік охолодження [9, 12].

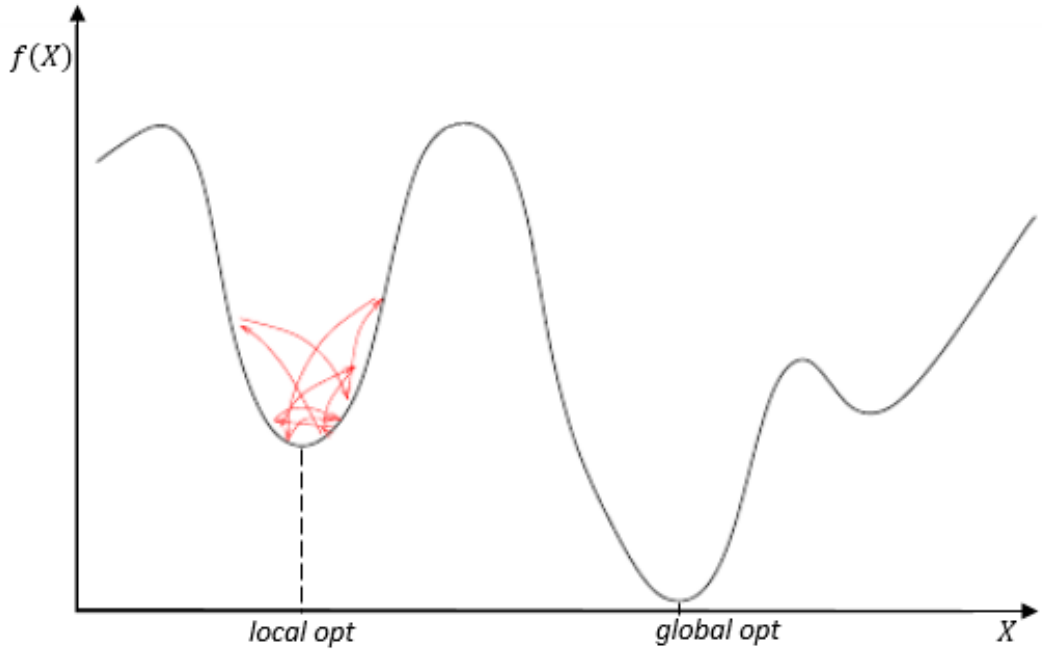

Рис. 2.2. Імітація відпалу: загроза захоплення локального мінімуму через низьку поточну T

· Монотонні графіки охолодження.

У монотонному графіку температура системи безперервно і монотонно зменшується з часом або ітерацій. При монотонному розкладі охолодження температура поступово знижується до досягнення деякого зумовленого критичного значення, при якому пошук зупиняється.

1. Лінійне мультиплікативне (LM) охолодження.

Лінійне зниження температури здійснюється відніманням від початкової температури величини температурного циклу з лінійно пропорційним до нього коефіцієнтом, що зі збільшенням значення призводить до швидшого охолодження.

$$T_k = T_0 - \alpha^k, \qquad \alpha > 0, \tag{2.11}$$

де $T_0 -$ початкова температура, $\alpha -$ параметр прискорення, $k -$ температурний цикл.

2. Експоненційне мультиплікативне (EM) охолодження.

Запропоноване Кіркпатріком, Гелаттом і Веккі (1983) і використане як еталон для порівняння різних критеріїв охолодження. Зниження температури

здійснюється множенням початкової температури на коефіцієнт, який експоненціально зменшується по температурному циклу.

$$T_k = T_0 \cdot \alpha^k, \qquad 0 < \alpha < 1, \tag{2.12}$$

де $T_0 -$ початкова температура, $\alpha -$ параметр прискорення, $k -$ температурний цикл.

3. Логарифмічне мультиплікативне (LGM) охолодження.

Засноване на асимптотичній умові конвергенції імітованого відпалу (Aarts, E.H.L. та Korst, J., 1989), але включає фактор прискорення охолодження, що робить можливим його використання на практиці. Зниження температури здійснюється множенням початкової температури на коефіцієнт, який зменшується обернено пропорційно натуральному логарифму температурного циклу.

$$T_k = \frac{T_0}{1+\alpha \cdot \log(1+k)}, \qquad \alpha \geq 1, \tag{2.13}$$

де $T_0 -$ початкова температура, $\alpha -$ параметр прискорення, $k -$ температурний цикл.

4. Квадратичне мультиплікативне (QM) охолодження.

Зниження температури здійснюється множенням початкової температури на коефіцієнт, який зменшується обернено пропорційно квадрату температурного циклу.

$$T_k = \frac{T_0}{1+\alpha \cdot k^2}, \qquad \alpha > 0, \tag{2.14}$$

де $T_0 -$ початкова температура, $\alpha -$ параметр прискорення, $k -$ температурний цикл.

Мультиплікативні монотонні графіки охолодження (2.11–2.14), залежать від наступних вхідних параметрів: початкової температури на величину, що залежить від поточного кроку та постійної α, значення якої встановлюється вручну. Проведемо дослідження з метою вивчення впливу обраної стратегії монотонного мультиплікативного охолодження на швидкість збігу алгоритму імітації відпалу (рис. 2.3, 2.4, 2.5, 2.6).

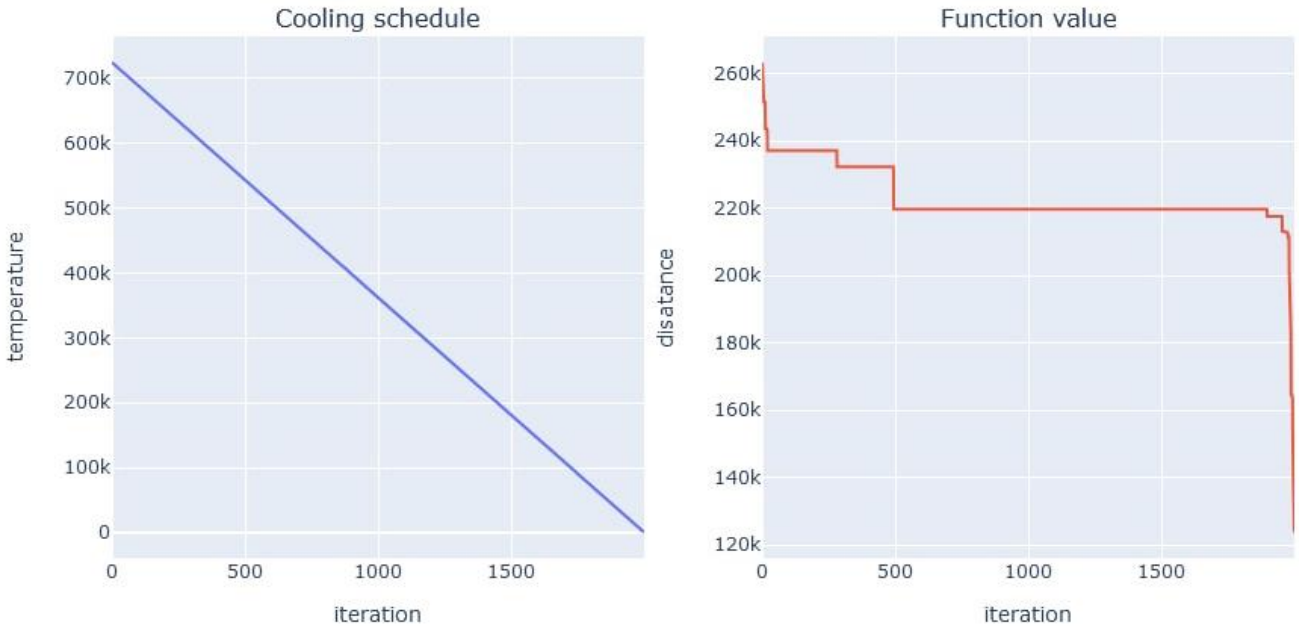

Рис. 2.3. Графіки а) LM охолодження та б) швидкості збігу алгоритму

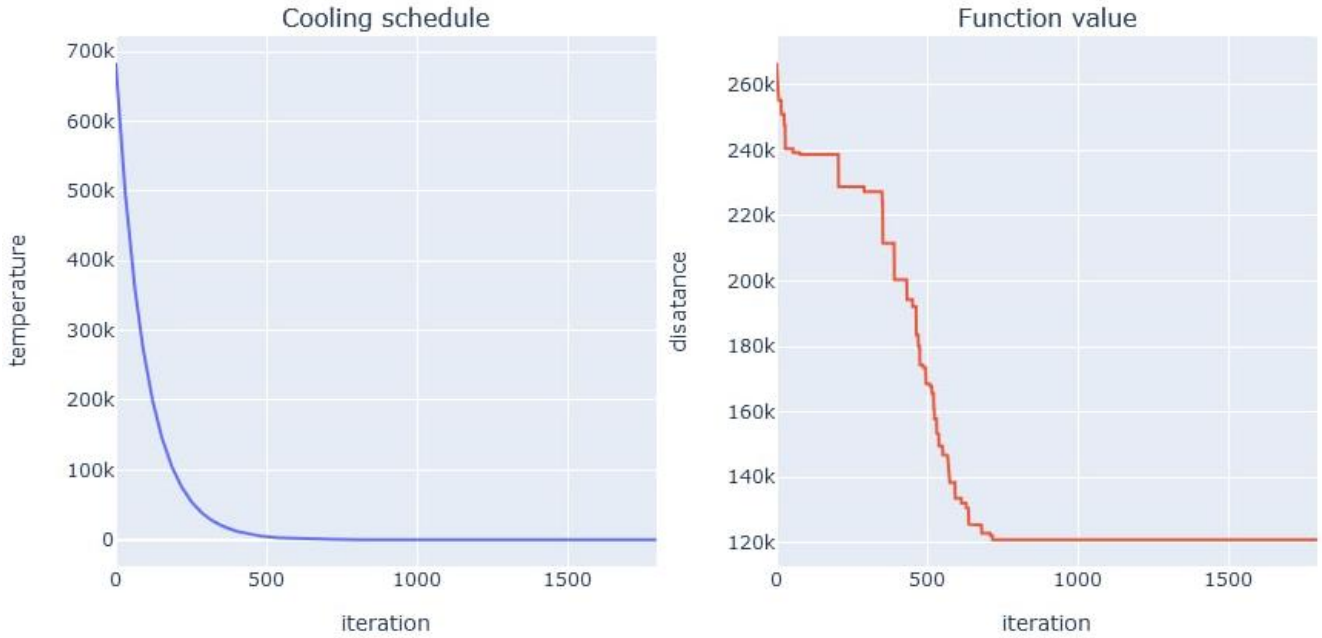

Рис. 2.4. Графіки а) ЕМ охолодження та б) швидкості збігу алгоритму

Перш за все слід зазначити, що лише саме логарифмічне та квадратичне мультиплікативне охолодження показали найшвидше зниження температури, що у свою чергу не надає можливість збільшити допустиму область дослідження функції (рис. 2.5 – 2.6). Підтвердження цього факту є швидка мінімізація значення функції за мінімальну кількість ітерацій, що на відміну від лінійного та експоненціального охолодження не досягає задовільного субоптимального рішення.

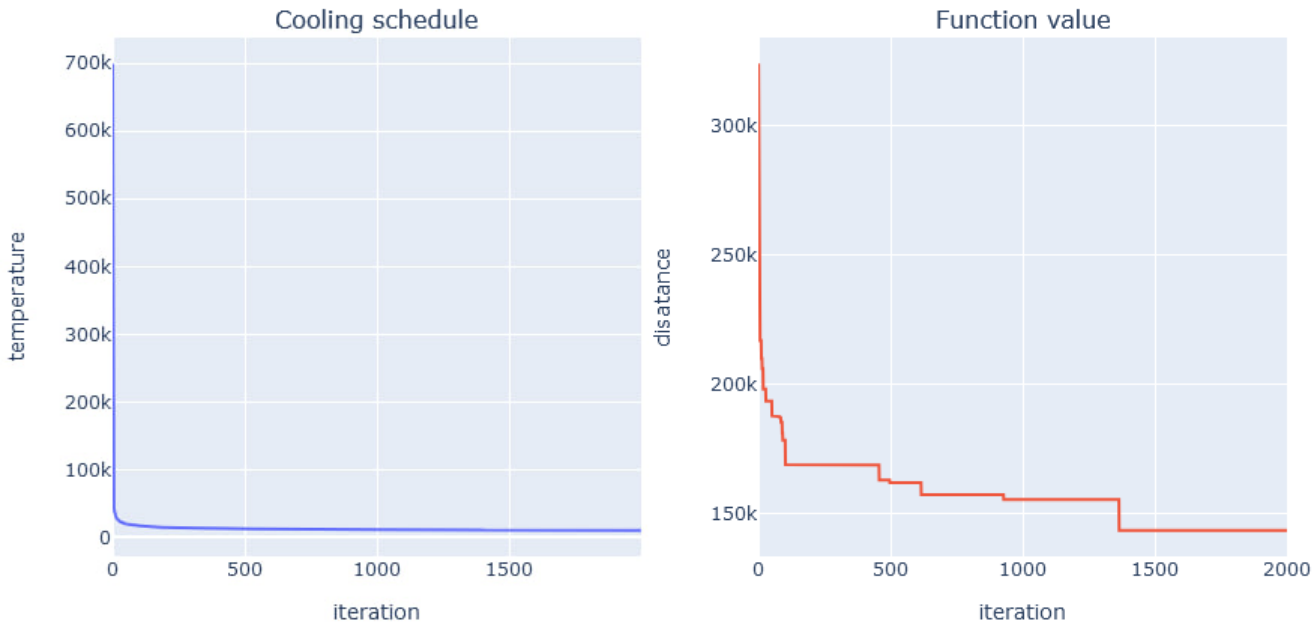

Рис. 2.5. Графіки а) LGM охолодження та б) швидкості збігу алгоритму

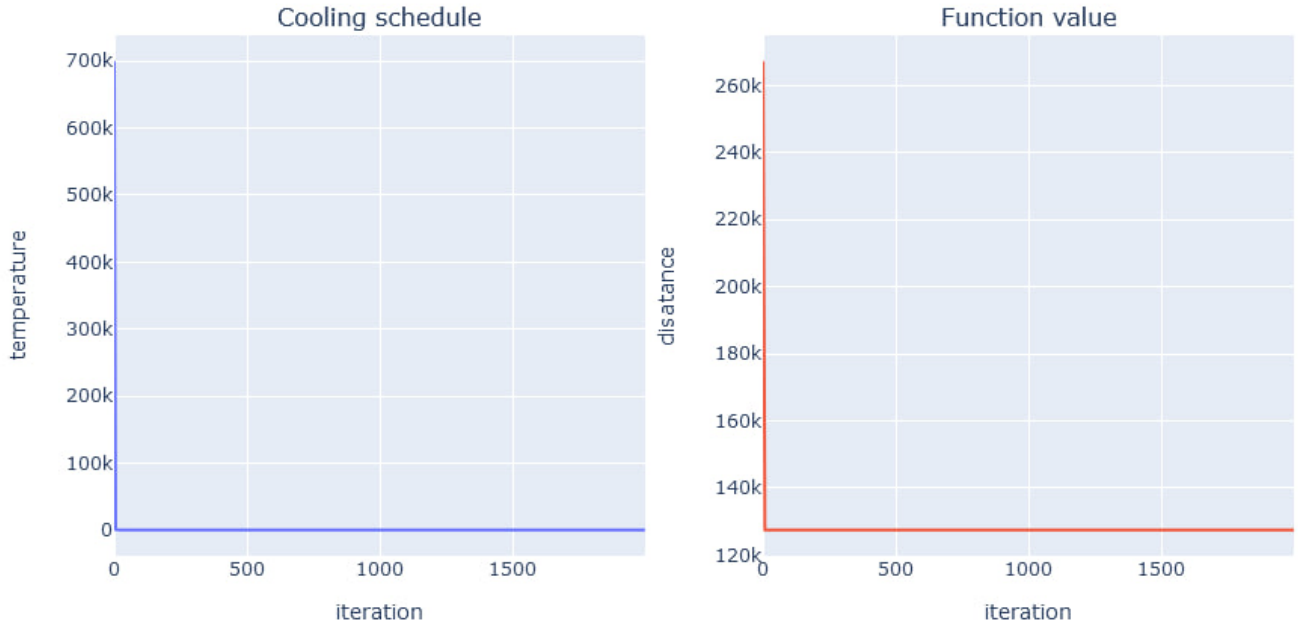

Рис. 2.6. Графіки а) QM охолодження та б) швидкості збігу алгоритму

При дослідженні лінійного мультиплікативного охолодження (рис. 2.3) можна прийти до висновку, що даний тип процесу охолодження передбачає поступове зменшення температури на постійне значення з часом, через що, не дивлячись на результат мінімізації функції, даний процес потребує найбільшу кількість ітерацій. Такий графік може бути корисним у ситуаціях, де потрібне рівномірне та передбачуване зниження температури, в контексті CVRP

мультиплікативне лінійне охолодження не дасть ефективних результатів на відміну від більш швидких графіків зміни температури системи.

Результат роботи експоненційного мультиплікативного охолодження (рис. 2.4) характеризують даний тип охолодження, що забезпечує швидкий початковий спад температури, а далі у міру поступового зменшення температури швидкість охолодження уповільнюється, що дозволяє системі ретельніше досліджувати простір рішень та надає перевагу над іншими швидкими графіками охолодження як логарифмічний та квадратичний.

Отже, проаналізувавши описані дослідження, можна прийти до висновку, що не тільки швидкість зниження температури, а також ступінь дослідження простору рішень являються факторами, на які слід опиратися при прийнятті рішення імплементації методу охолодження. За результатми дослідження виявилось, що експоненційний та квадратичний графіки охолодження зазвичай є більш ефективними у контексті імітації відпалу для пошук рішень транспортних задач, але слід зауважити, що остаточний вибір графіку охолодження залежить від конкретного контексту та всіх вимог завдання.

У випадку адитивних графіків монотонного охолодження (2.15–2.17), поточна температура має залежність від наступних величин: початкової температури, що залежить від поточного кроку, загальної кількості кроків та кінцевої температури.

5. Лінійне адитивне (LA) охолодження.

$$T_k = T_{end} + (T_0 - T_{end}) \cdot \left(\frac{n-k}{n}\right), \tag{2.15}$$

де $T_0 -$ початкова температура, $T_{end} -$ кінцева температура, $k -$ температурний цикл, $n -$ загальна кількість кроків.

6. Експоненційне адитивне (EA) охолодження.

$$T_k = T_{end} + (T_0 - T_{end}) \cdot \left(1 + e^{\frac{2\ln(T_0 - T_{end})}{n} \cdot \left(k - \frac{n}{2}\right)}\right)^{-1}, \tag{2.16}$$

де $T_0 -$ початкова температура, $T_{end} -$ кінцева температура, $k -$ температурний цикл, $n -$ загальна кількість кроків.

7. Квадратичне адитивне (QA) охолодження.

$$T_k = T_{end} + (T_0 - T_{end}) \cdot \left(\frac{n-k}{n}\right)^2, \quad (2.17)$$

де $T_0 -$ початкова температура, $T_{end} -$ кінцева температура, $k -$ температурний цикл, $n -$ загальна кількість кроків.

Розглянемо швидкість збігу алгоритму імітації відпалу залежно від монотонного адативного графіку охолодження (рис. 2.7, 2.8, 2.9).

Рис. 2.7. Графіки а) LA охолодження та б) швидкості збігу алгоритму

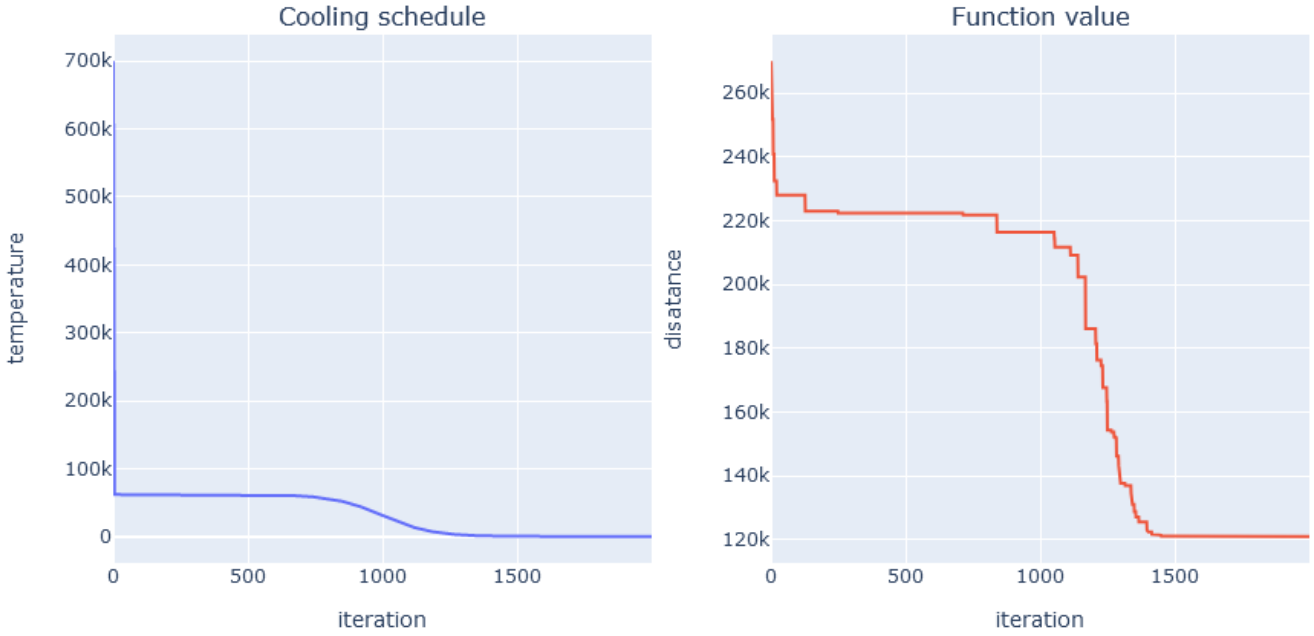

Рис. 2.8. Графіки а) EA охолодження та б) швидкості збігу алгоритму

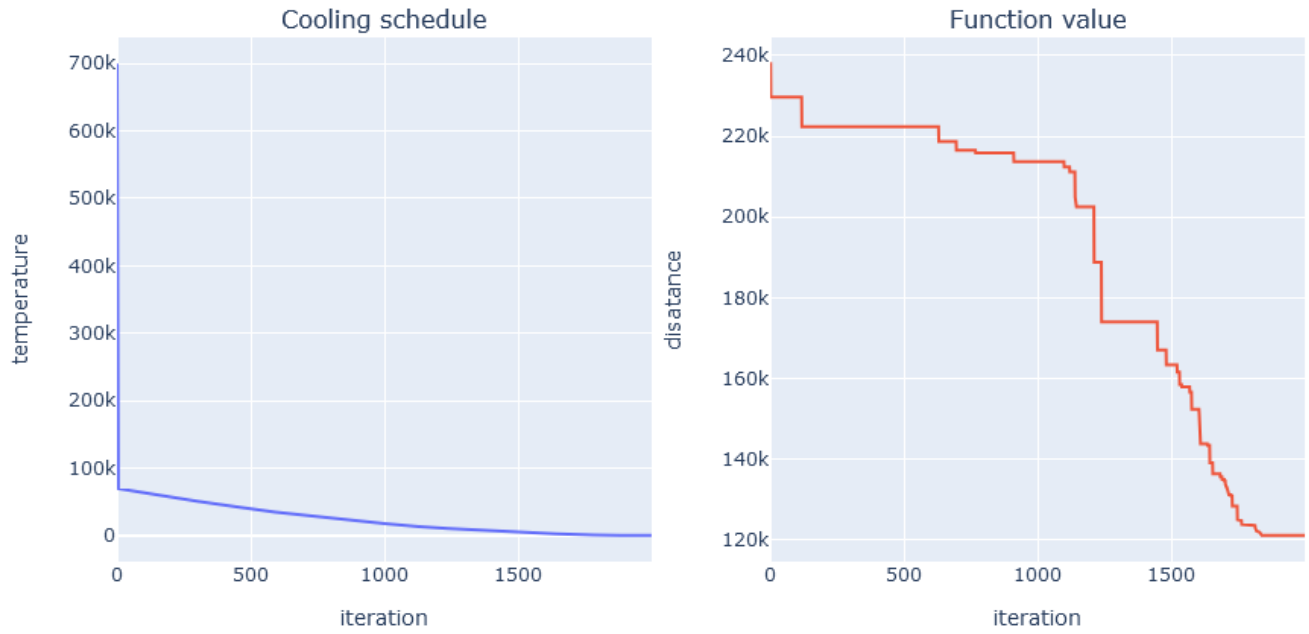

Рис. 2.9. Графіки а) QA охолодження та б) швидкості збігу алгоритму

Дослідивши результати роботи лінійного адитивного охолодження (рис. 2.7) можна прийти до висновку, що, не дивлячись на швидкий спад температури, порівняно з мультиплікативним типом, отриманий результат мінімізації виявився аналогічним. Отже, у даному випадку виявляється, що незалежно від вибору між, мультиплікативним чи адитивним типом, лінійне охолодження показує однакові результати, а також однакову кількість ітерацій. Виходячи з цього, маємо те, що адитивне лінійне охолодження не дасть ефективних результатів на відміну від більш швидких графіків зміни температури системи для задачи CVRP.

Графіки адитивного експоненційного та квадратичного охолодження (рис. 2.8 – 2.9) свідчать, що обидва досягають оптимального значення майже при однаковій швидкості зниження температури. Однак, за першим методом збіг цільової функції має більшу швидкість.

Основним показником відмінності між адитивним та мультиплікативним квадратичним методом охолодження стала ступінь дослідження області рішень: адитивний тип показав кращий результат.

Адитивне експоненційне охолодженням на відміну від квадратичного має швидший збіг функції за рис. 2.8-2.9 (б), однак, спостерігаючи за плавною частиною зміни температури (рис. 2.8-2.9, а), квадратичний спад має гладку зміну параметру T, це може позитивно вплинути на ступінь дослідження області

рішень, у випадку з експоненційним графіком зміни температури спостерігається протилежний результат.

На рис. 2.10 зображено ілюстрацію характеристик монотонних графіків пониження температурного параметру алгоритму імітації відпалу, що були отримані в дослідженнях (табл. 2.1). Дана візуалізація підтверджує винсновки щодо ознак швидкості збігу та ступеню дослідження області рішень: кращі результати розташовуються ближче до центру оптимальності. Однак, слід також пам’ятати про важливість поведінки графіків зміни температур.

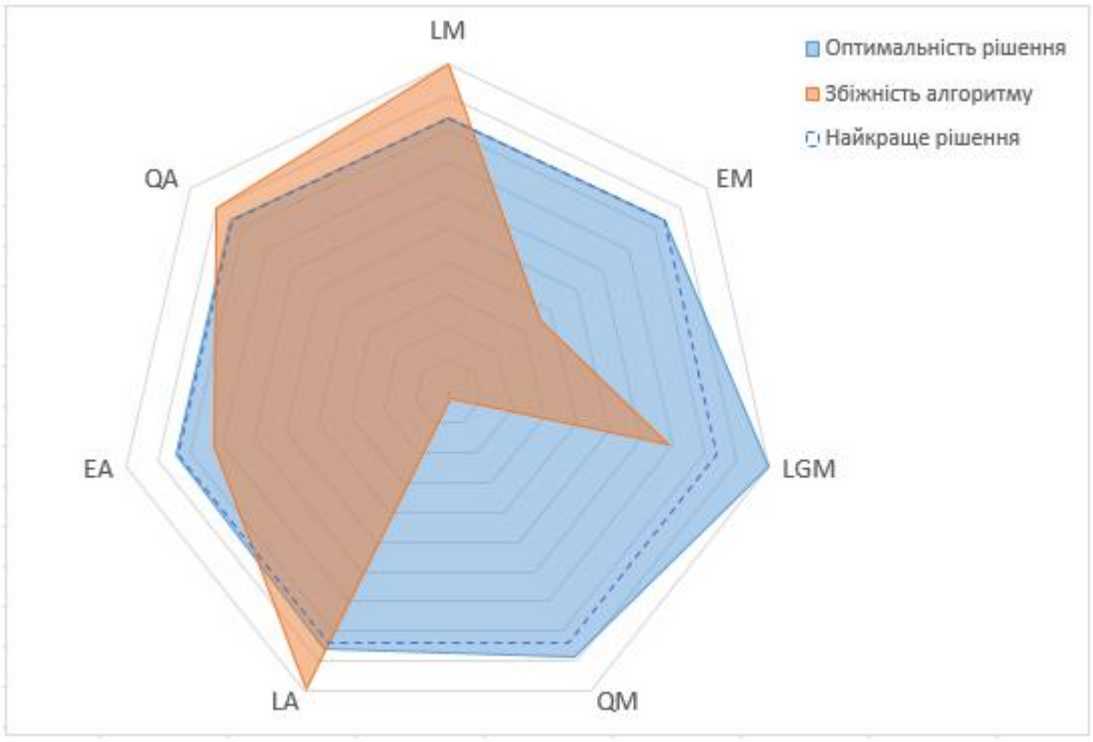

Рис. 2.10. Пелюсткова діаграма характеристик монотонних графіків охолоджень

Отже, виходячи з цього, можна прийти до висновку, що серед монотонних графіків охолодження експоненційна мультиплікативна зміна температури виявилась найефективнішоб стратегією охолодження при використанні алгоритму «імітації відпалу» для пошук рішень задач маршрутизації транспортних засобів.

· Немонотонні графіки охолодження.

У немонотонному охолодженні температура системи T при кожному переході стану обчислюється множенням значення поточної температури, отриманого за будь-яким із попередніх критеріїв, на адаптивний коефіцієнт, заснований на

різниці між поточною цільовою метою рішення та найкращим значенням цільової функції, досягнутим до цього моменту алгоритмом (2.18).

$$T = \mu \cdot T_k, \qquad \mu = \left[1 + \frac{f(s_i) - f^*}{f(s_i)}\right] \tag{2.18}$$

де $T_k -$ поточна температура, $\mu -$ адаптивний коефіцієнт, $f^* -$ найкраще значення функції до моменту поточного шагу, $f(s_i) -$ поточне значення цільової функції, $s_i -$ рішення з підмножини, що складається з наборів клієнтів або точок доставки, які мають бути включені у маршрутизацію транспортних засобів.

Коефіцієнт μ означає, що чим більша відстань між поточним рішенням і найкращим досягнутим рішенням, тим вища температура і, отже, дозволено прийняття рішення з гіршим значенням. Цей критерій є варіантом критерію, запропонованого М. Локателлі (2000), і його можна використовувати в комбінації з будь-яким із попередньо згаданих монотонних критеріїв обчислення графіків охолодження T_k.

Досліджуючи вплив немонотонних графіків охолодження на швидкість збігу алгоритму імітації відпалу та його ступеню дослідження області рішень для пошуку розв'язку задачі CVRP, зміни, що покращували б результати досліджень монотонних стратегій пониження температури, не були виявлені (табл. 2.1).

Таблиця 2.1

Результативні значення досліджень графіків охолодження

Графік охолодження			Цільова функція	Ітерація збігу
Монотонні	Мультиплікативні	LM	120997	2000
		EM	120997	717
		LGM	143493	1373
		QM	127502	39
	Адитивні	LA	123206	1990
		EA	120997	1455
		QA	121122	1802
Немонотонні	Мультиплікативні	LM	128842	1295
		EM	136857	635
		LGM	142109	993
		QM	139611	22
	Адитивні	LA	136879	1975
		EA	136857	1428
		QA	136857	1801

· Модифікація графіку охолодження: рестарт [13].

В імітації відпалу з рестартом (Simulated Annealing with Restart) вводиться додатковий механізм, який допомагає впоратися із проблемою застрягання у локальних оптимумах. Регулятор рестарту використовується для періодичного скидання поточного стану алгоритму та дослідження нових областей простору пошуку. Основна ідея регулятора рестарту полягає в тому, що замість продовження пошуку в локальній околиці, коли алгоритм досягає локального оптимуму, він перезапускається з новим випадковим початковим станом або попереднього кращого знайденого стану. Це дозволяє імітації відпалу досліджувати нові області простору пошуку та має потенціал для знаходження кращого глобального оптимуму.

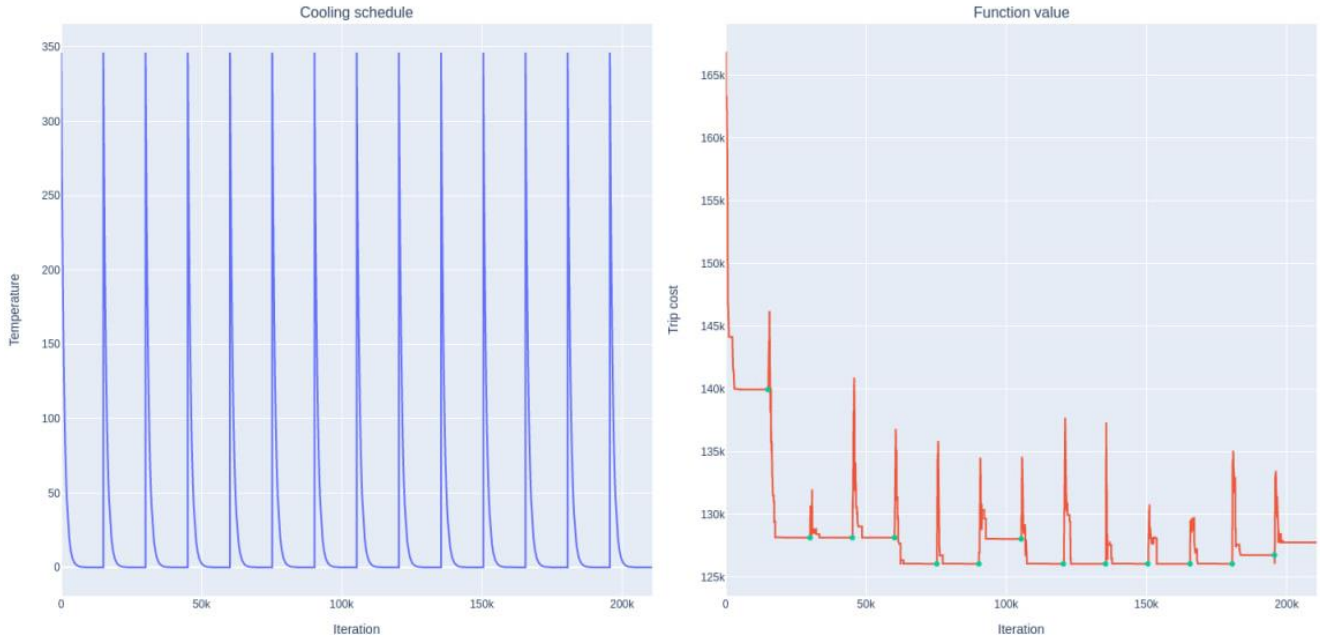

Рис. 2.11. Графіки а) ЕМ охолодження з рестартом та б) динаміки цільової функції

Регулятор рестарту зазвичай визначається певними критеріями або умовами, за якими відбувається рестарт. Наприклад, можна встановити максимальну кількість ітерацій без покращення, після якого відбувається рестарт, або встановити граничне значення для приросту вартості, при якому також відбувається рестарт. Регулятор рестарту може бути налаштований для оптимального балансу між дослідженням та використанням найкращих знайдених рішень.

За реалізацією агоритму імітації відпалу, ілстниг програми якого наведено у додатку В, control та fitness є метриками, що використовуються для контролю роботи алгоритму імітації відпалу. Вони служать адаптації регулятора (regulator) і визначення моменту, коли відбувається перегрів алгоритму і потрібно відновлення (reheat).

Параметр control відповідає за контроль інтенсивності пошуку у просторі рішень. Значення control розраховується з використанням формули 2.19.

$$control = n \cdot \ln(iterations) \tag{2.19}$$

де n – параметр, що визначає інтенсивність пошуку, а iterations – кількість виконаних ітерацій алгоритму.

Чим більше iterations, тим більше значення control, що призводить до збільшення інтенсивності пошуку нових рішень.

Метрика fitness використовується для порівняння поточного рішення з попереднім. Значення fitness розраховується у методі anneal() перед прийняттям чи відхиленням нового рішення. Воно служить визначення, чи є поточне рішення краще чи гірше попереднього. Якщо поточне рішення краще, значення fitness оновлюється, приймаючи значення цільової функції поточного рішення.

Адаптація регулятора (regulator) відбувається з урахуванням значень control і fitness. В алгоритмі імітації відпалу приймаються нові рішення із заданою ймовірністю, що базується на зміні вартості поточного та нового рішень та поточній температурі (T). Значення регулятора впливає на цю можливість. Якщо поточне рішення краще за попереднє (current_cost $<$ fitness), то значення регулятора зменшується (2.20).

$$regulator = \max\left(\frac{regulator}{control - fitness}, regularization\right) \tag{2.20}$$

де $regularization$ – нижня границя для значення регулятора.

Якщо поточне рішення гірше за попереднє (current_cost $>$ fitness), то значення регулятора збільшується (2.21).

$$regulator = \min\left(\frac{n}{best - current\_cost}, regularization\right) \tag{2.21}$$

де n – параметр, що визначає інтенсивність пошуку, $best$ – останнє найкраще значеня функції, $regularization$ – верхня границя для значення регулятора.

Таким чином, значення control та fitness використовуються для адаптації регулятора (regulator), який впливає на ймовірність прийняття нових рішень в алгоритмі імітації відпалу. Це дозволяє контролювати інтенсивність пошуку та відновлення алгоритму при перегріві (рис. 2.11).

2.4.2 Тестування алгоритму за наборами даних Augerat 1995 – Set A

Щоб знайти найкращий діапазон часу очікування, було проведено ряд тестів (табл. 2.2) із використанням екземплярів (instance) Augerat 1995 – Set A [14] для 31–33, 35–38, 43–45, 47, 52–54, 59–64, 68 і 79 клієнтів, кожен з яких вирішувався 5 разів. На основі результатів тестових екземплярів можна побачити, що у одній половині випадків (14 з 27) найкращі результати були отримані, коли діапазон часу очікування становив від 0 до 7 хвилин, в іншій половині (13 з 27) – від 7 до 12 хвилин.

Головний недолік імітованого відпалу полягає в асимптотичній збіжності алгоритму [7]. Властивість без пам’яті (поточний хід залежить лише від поточного стану, а не від попередньої історії) робить імітацію відпалу не найефективнішим алгоритмом для вирішення задач оптимізації з великими даними.

Асимптотична збіжність алгоритму імітації відпалу відноситься до властивості цього алгоритму сходитися до оптимального рішення при збільшенні числа ітерацій або часу виконання. Данна проблема виникає при практичному застосуванні SA: якщо локальна конфігурація близька до локального мінімізатора, а температура вже дуже мала в порівнянні зі стрибком вгору, який потрібно виконати, щоб вийти з атрактора, величезна кількість ітерацій може бути проведена навколо атрактора (рис 2.2). Враховуючи обмежений час дослідження

області рішень, усі майбутні ітерації можна витратити, намагаючись покинути область локального мінімуму замість подальшого дослідження області рішень. Дані твердження підкріплюються тестуванням алгоритму SA: час пошуку кінцевого рішення не має чіткої залежності від кількості вершин (рис. 2.13).

Табл. 2.2

Результати тестування за наборами даних Augerat 1995 – Set A

Instance	n	K	Q	Simulated annealing results			Function			Time
				Best	Worst	Avg	LB	AE	RE	
A-n32-k5	32	5	100	784	785	784,2	784	0,2	0,00026	00:00:33
A-n33-k5	33	5	100	661	663	661,4	661	0,4	0,00061	00:05:13
A-n33-k6	33	6	100	742	745	742,6	742	0,6	0,00081	00:01:19
A-n34-k5	34	5	100	778	780	778,4	778	0,4	0,00051	00:02:46
A-n36-k5	36	5	100	805	813	808,333	799	9,33333	0,01168	00:01:56
A-n37-k5	37	5	100	669	670	669,25	669	0,25	0,00037	00:01:48
A-n37-k6	37	6	100	953	963	957,333	949	8,33333	0,00878	00:01:55
A-n38-k5	38	5	100	731	735	732,4	730	2,4	0,00329	00:08:45
A-n39-k5	39	5	100	826	829	827	822	5	0,00608	00:01:30
A-n39-k6	39	6	100	833	837	835,333	831	4,33333	0,00521	00:09:06
A-n44-k6	44	6	100	967	970	968,333	937	31,3333	0,03344	00:09:24
A-n45-k6	45	6	100	980	1038	1016,2	944	72,2	0,07648	00:01:54
A-n45-k7	45	7	100	1171	1186	1180,33	1146	34,3333	0,02996	00:09:40
A-n46-k7	46	7	100	921	935	927,75	914	13,75	0,01504	00:04:38
A-n48-k7	48	7	100	1085	1094	1088,67	1073	15,6667	0,0146	00:10:27
A-n53-k7	53	7	100	1052	1067	1057,33	1010	47,3333	0,04686	00:06:25
A-n54-k7	54	7	100	1244	1263	1254	1167	87	0,07455	00:10:21
A-n55-k9	55	9	100	1111	1128	1117,33	1073	44,3333	0,04132	00:10:43
A-n60-k9	60	9	100	1420	1481	1442,33	1354	88,3333	0,06524	00:07:17
A-n61-k9	61	9	100	1102	1234	1163	1034	129	0,12476	00:11:06
A-n62-k8	62	8	100	1381	1384	1382,25	1288	94,25	0,07318	00:05:01
A-n63-k9	63	9	100	1785	1807	1790,5	1616	174,5	0,10798	00:11:17
A-n63-k10	63	10	100	1398	1421	1414,25	1314	100,25	0,07629	00:11:40
A-n64-k9	64	9	100	1503	1506	1504,5	1401	103,5	0,07388	00:06:50
A-n65-k9	65	9	100	1299	1349	1321,67	1174	147,667	0,12578	00:11:31
A-n69-k9	69	9	100	1265	1276	1270,5	1159	111,5	0,0962	00:07:24
A-n80-k10	80	10	100	1977	2008	1993,2	1763	230,2	0,13057	00:03:11

де $instance -$ екземпляр з набору Augerat 1995 – Set A, $n -$ кількість вершин, $K -$ кількість транспортних засобів, $Q -$ ємність транспортного засобу, $best -$ найкраще знайдене рішення, $worst -$ найгірше знайдене рішення, $avg -$ метматичне очікування рішень за екземпляром, $LB -$ оптимальне рішення, $AE -$

абсолютна похибка між LB та avg, RE – відносна похибка між LB та avg, $time$ – середній час роботи алгоритму в пошуку рішення за відповідним екземпляром.

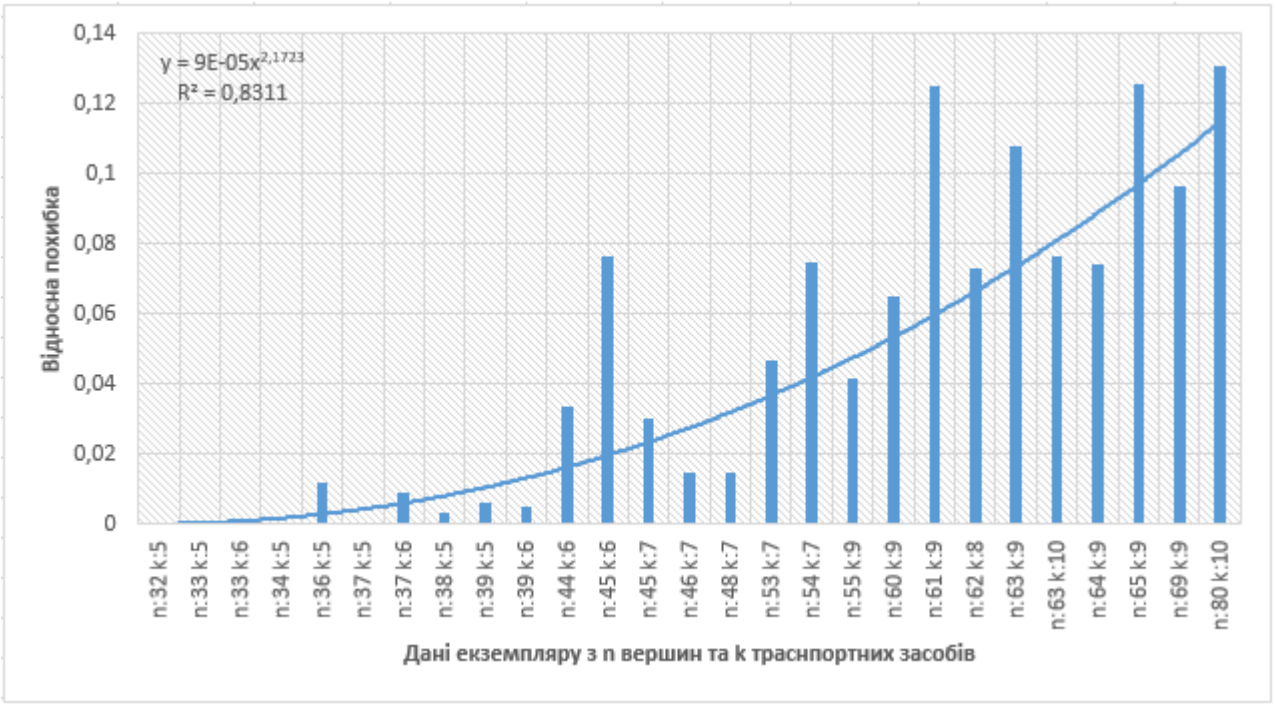

Рис. 2.12. Діаграма відносної похибки за даними Augerat 1995 – Set A

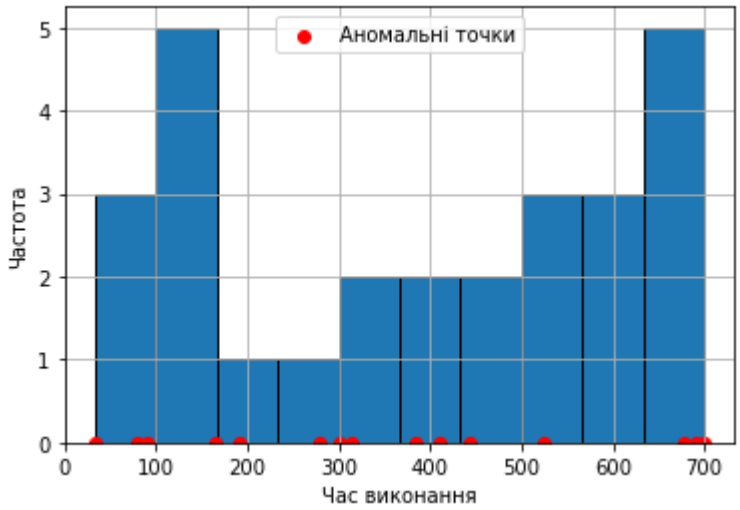

Рис. 2.13. Гістограма розподілу за даними часу досягнення критерію зупинки

За результатами отриманих рішень з даними, що містять невелику кількість вершин, алгоритм імітації відпалу відображає свою ефективність, знаходячи розв’язки, за якими відносна похибка має задовільні значення (рис. 2.12).

За даними середніх значень та дисперсії повторних відпрацювань алгоритму репродукуваність алгоритму значна, що надає імітації відпалу характеристики надійності та якості.

2.4.3 Результати задачі торгівельної компанії

За наданими координатами мережі магазинів, що складається з 31 супермаркету та одного складу, шукаємо в місті з дорогами, підходящими для експлуатації транспортних засобів із автопарку компанії, найкоротші шляхи між кожною вершиною за допомогою модулю smart mobility utilities (рис. 2.14), застосовуючи алгоритм Дейкстри (1959).

Довжина складених шляхів формує матрицю відстаней між всіма вузлами майбутнього графу для вирішення задачі CVRP.

Рис. 2.14. Ілюстрація отриманих шляхів між всіма вузлами мережі торгівельної компанії а) від скаду б) від одного з магазинів

```
Required vehicles amount: 3
OPTIMIZED COST: 124711
Route 1: 0 2 3 16 10 25 26 22 20 23 30 8 19 24 0
Route 2: 0 29 21 12 9 27 11 28 15 4 6 17 0
Route 3: 0 14 13 1 7 5 18 31 0
Час виконання програми становить:  0:00:36.749300
```

Рис. 2.15. Результат побудови маршруту транспортних засобів

Застосовуючи отриману матрицю довжин маршрутів та наданого попиту кожного супермаркету в найзавантаженіший день, отримуємо результат за алгоритмом імітації відпалу, що відображаєтсья на рис. 2.15.

За вхідними даними отримано інформацію про те, що продукцію слід розвозити трьома маршрутами, кожен з яких відображен на карті міста на рис. 2.16. Загальний шлях всіх трьох побудованих маршрутів склав 124 км 711 м.

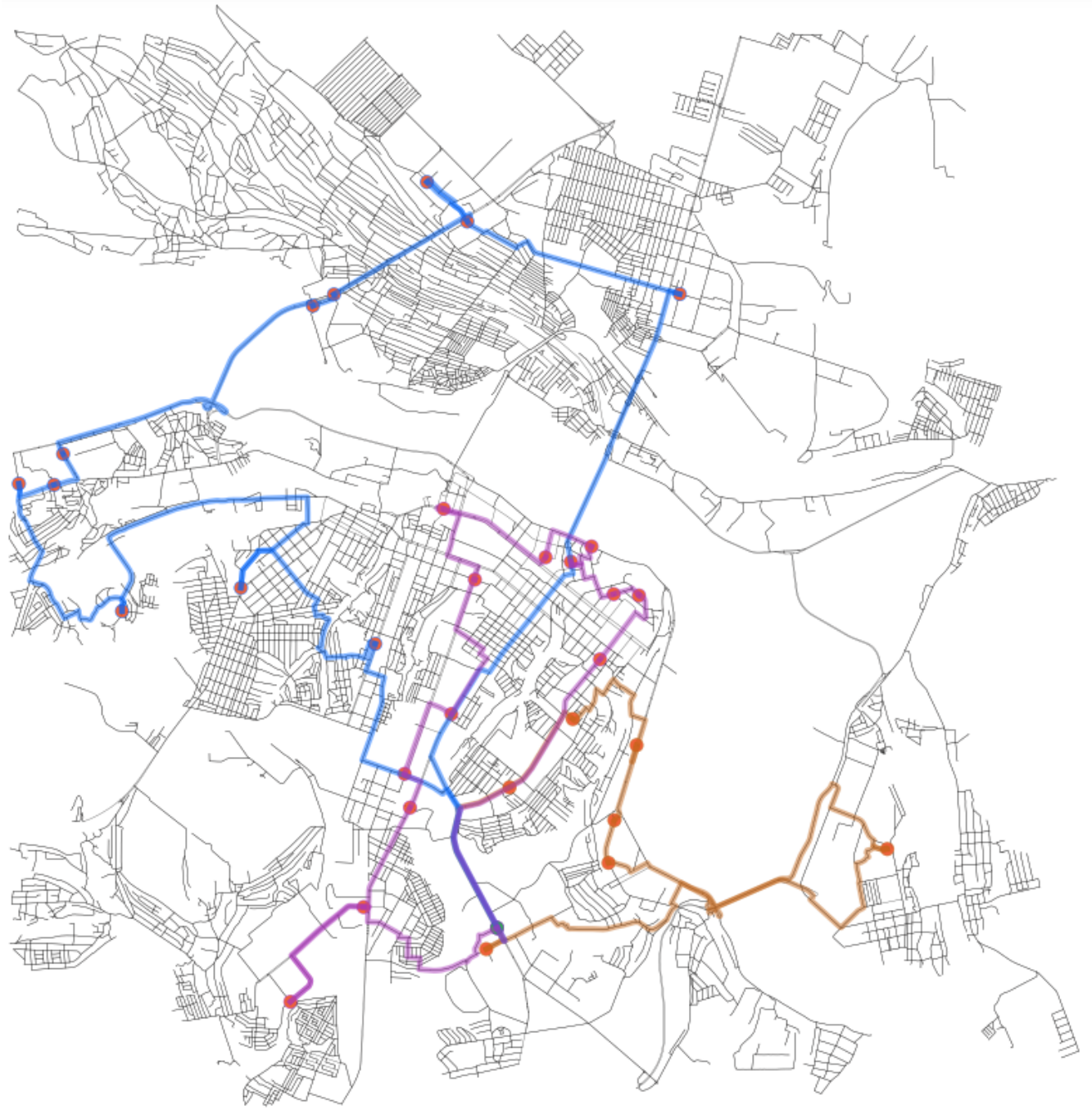

Рис. 2.16. Візуалізація шляхів містом для задачі торгівельної компанії

2.5 Висновки

У даному розділі були розглянуті різні стратегії монотонного мультиплікативного, монотонного адитивного та немонотонного охолодження, які визначають швидкість зміни температури під час процесу охолодження системи. Кожна з обраних стратегій характеризується своїм коефіцієнтом, що визначає швидкість охолодження на кожному кроці.

На практиці виявлено, що метод імітації відпалу представляє собою ефективний інструмент для глобальної оптимізації, дослідження функцій та вирішення складних задач, де інші алгоритми можуть застрягати в локальних мінімумах. Його стохастичний характер і здатність досліджувати широкий простір параметрів роблять його корисним інструментом для пошуку субоптимальних рішень для невеликого розміру вхідних даних.

Через виявлену проблему асимптотичної збіжності алгоритму було відтворено модифікацію поведінки графіку зміни температурного параметру системи.

У результаті реалізованої стратегії алгоритму імітації відпалу, його використання на невеликих наборах даних з меншею кількістю вершин (до 50 вершин) призводить до задовільних результатів у наближенні до оптимальних рішень, а також забезпечує відтворюваність алгоритму.

ВИСНОВКИ

У рамках цієї кваліфікаційної роботи було досліджено та застосовано метод імітації відпалу для вирішення комбінаторної проблеми маршрутизації транспортних засобів (CVRP).

Таким чином, у інформаційно-аналітичному розділі було визначено поняття маршрутизації транспортних задач та методи їх оптимального пошуку. Окремо було розглянуто алгоритм локального пошуку, відомого як імітація відпалу що буде реалізовуватись у спеціальному розділі. Були коротко описані різновиди транпортних задач.

У спеціальному розділі були обрані для реалізації оптимальні методи початквого пошуку – метод найближчого сусіда, та вибору сусіднього рішення – 2-opt алгоритм покращення. Був проведений аналіз різних графіків охолодження для пониження температури, обґрунтовано прийняте рішення реалізації монотонної експоненціної зміни тепмературного параметру на основі трьох факторів: швидкості збігу алгоритму, здатності досліджувати широкий простір рішень, динаміки зміни температури. Було проведено вдосконалення впливу температури модифікацією рестарту для ефективнішого використання алгоритмом імітації відпалу ресурсу часу з метою ширшого дослідження простору рішень.

Додатко для пошуку рішення задачі торгівельної компанії було використано алгоритм дейкстри для формування вхідних даних відстаней через пошук найкоротшого шляху між вершинами графу на мапі дозволених для експлуатації доріг. Здійснено пошуку ефективних перевезень вантажів у мережі магазинів торігвельної компанії створеним методом імтації відпалу.

Було виявлено, що при використанні імітації відпалу на невеликих наборах даних з малою кількістю вершин алгоритм демонструє хороші результати в досягненні наближених оптимальних рішень, а також своєї репродукуваності. Однак, провівши подальший аналіз, було виявлено, що в області CVRP, де потрібне ефективне рішення задачі оптимального маршрутизації з урахуванням

обмежень вантажопідйомності транспортних засобів, більш продуктивними можуть бути алгоритми, що мають властивість пам'яті. Ці алгоритми здатні враховувати попередні рішення та використовувати їх для прийняття оптимальних рішень при побудові маршрутів. Такі алгоритми можуть надати більш точні та ефективні рішення для великих наборів даних та складних комбінаторних завдань, таких як CVRP.

Таким чином, результати данної роботи мають практичне значення для застосування в індустрії логістичних перевезень з невеликою кількістю вхідних значень та надають рекомендації для практики в галузі маршрутизації транспортних засобів.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Гуляницький Л.Ф., Мулеса О.Ю. Прикладні методи комбінаторної оптимізації [Текст]: підручник / Овакімян А.С., Саркісян С.Г, Зіроян М.А., Тінякова В.И., 2016. – 142 с.
2. Oren Nahum. The Real-Time Multi-Objective Vehicle Routing Problem [Текст]: наукова праця / Israel: Bar-Ilan University, 2013. – 33 с.
3. Rosa Herrero Anton. Hybrid Methodologies for Symmetric and Asymmetric Vehicle Routing Problems [Текст]: наукова праця / Industrial Informatics: Advanced Production Techniques Department of Telecommunications and Systems Engineering Autonomous University of Barcelona, 2015. – 182 с.
4. Springer [Електронний ресурс]. – Режим доступу https://www.researchgate.net/publication/344273735_Vehicle_routing_problem_and_related_algorithms_for_logistics_distribution_a_literature_review_and_classification
5. Aarts E. H. L., Korst J. H. M. Simulated annealing and Boltzmann machines [Текст]: наукова праця / Chichester: Wiley, 1989. – 272 с.
6. Kirkpatrick S., Gelatt C. D., Vecchi M. P. Optimization by simulated annealing [Текст]: наукова праця / Science. Vol. 220, 1983. – с. 671–680.
7. Roberto Battiti, Mauro Brunato, Franco Mascia. Reactive Search and Intelligent Optimization [Текст]: підручник / Italy: Department of Information Engineering and Computer Science, University of Trento, 2007. – 92 с.
8. Debasis Mitra, Fabio Romeo, and Alberto Sangiovanni-Vincentelli, Convergence and finite-time behavior of simulated annealing [Текст]: наукова праця / Advances in Applied Probability. Vol. 18, 1986. – с. 747–771.
9. Ibrahim Hassan Osman, Metastrategy simulated annealing and tabu search algorithms for the vehicle routing problem [Текст]: наукова праця / Ann. Oper. Res. 41, 1993. – с. 37–52.

10. G. Gutin, A. Yeo, A. Zverovich. Traveling salesman should not be greedy: domination analysis of greedy-type heuristics for the TSP [Текст]: наукова праця / Discreate Applied Mathematics. Vol. 17, 2002. – с. 81–86.
11. Bazylevych R., Kutelmakh R., Tomchuk A. Solving large-scale traveling salesman problem by the “common edges” method [Текст]: наукова праця / Software department: Lviv Polytechnic National University, 2014. – 7 с.
12. Github [Електронний ресурс]. – Режим доступу https://nathanrooy.github.io/posts/2020-05-14/simulated-annealing-with-python/
13. Medium [Електронний ресурс]. – Режим доступу https://towardsdatascience.com/simulated-annealing-with-restart-a19a53d914c8
14. Cvrplib [Електронний ресурс]. – Режим доступу http://vrp.atd-lab.inf.puc-rio.br/index.php/en/

ДОДАТКИ

ДОДАТОК А

Відомість матеріалів кваліфікаційної роботи

№ з/п	Позначення	Назва	Кількість	Примітки
1				
2		Документація		
3				
4	САУ.КР.23.13.ПЗ	Пояснювальна записка	59	Формат А4
5				
6		Демонстраційні матеріали	12	Презентація на CD-R
7				
8		Копія роботи	1	Диск CD-R
9				
10				
11				
12				
13				
14				
15				
16				
17				
18				

					САУ.КР.23.13.ДА.ПЗ.			
Змін	Аркуш	№ докум.	Підпис	Дата				
Розроб.		Сазанська			**Матеріали кваліфікаційної роботи**	Літ.	Аркуш	Аркушів
К. розд.		Коряшкіна						
Керівн.		Коряшкіна				НТУ «ДП», 12; 124-19-2		
Н.контр.		Хом’як						
Зав. каф.		Желдак						

ДОДАТОК Б

Відгук

на кваліфікаційну роботу бакалавра за спеціальністю 124 Системний аналіз

студентки групи 124-19-2

Сазанської Ірини Олегівни

Керівник кваліфікаційної роботи бакалавра

Кандидат фізико-математичних наук

Доцент кафедри системного аналізу та управління Л.С. Коряшкіна

ДОДАТОК В

В1. Лістинг програми зчитування вхідних даних

```
import acvrp, sys, datetime
if len(sys.argv) == 3:
        start_time = datetime.datetime.now()
        with open('demand.txt', 'r') as fr:
                demand = fr.read().split()
                with open(sys.argv[1], 'r') as fp:
                        content = fp.read().split()
                        res = acvrp.SA(list(map(int, content)), list(map(int,
demand)), int(sys.argv[2]), learning_plot=False).solve()
                        if res[1]:
                            print('OPTIMIZED COST:', res[1])
                            print(' '.join(map(str, res[0])))
                            print(res[2])
                        else:
                            print('NO SOLUTION')
                        print('Час виконання програми становить : ',
datetime.datetime.now() - start_time)
```

В2. Лістинг програми алгоритму імітації відпалу

```
import math, random
import time, datetime
import numpy as np
import matplotlib.pyplot as plt, seaborn as sns, plotly.graph_objects as go
from plotly.subplots import make_subplots
from matplotlib.font_manager import FontProperties
from heapq import heappush, heappop
import itertools


class SA:
    def __init__(self, data, demand, VehicleQuantity, infinity=1<<17, initial_t=0,
rate=0.999, stopping_t=1e-4, initial_fitness=1, initial_solution=None,
iteration_bound=[1 << 17, 1 << 24], regularization_bound=(0.3, 3),
learning_plot=False, silent_mode=True):
        '''
        :param data: as full edge distance matrix or single line input
        :param infinity: a large integer number that is larger than the cost of
all possible solutions
        :param initial_t: initial temperature
        :param rate: the rate temperature decreases each iteration
        :param stopping_t: the final temperature accepted by algorithm
        :param initial_fitness: increase this number if algorithm failed to
descend in early stage
```

```
        :param iteration_bound: the (minimal, maximum) iteration number accepted
by algorithm; higher minimal iteration will increase the chance to find better
solution
        :param regularization_bound: as (lower_bound, upper_bound) to regularize
self.regulator
                - lower_bound affects how high each Tempering can go; the algorithm
might fail to descend if this number is too small or fail to jump out of local
minimal if this number is too large
                - upper_bound affects how flat the learning line can be; the Temper
might lose its magic if this number is too large
        :param learning_plot: setting this as True will also output the detail
info for each Temper
        :param silent_mode: setting this as True will only output one dot '.' when
finished; used in multi-threading mode
        '''

        self.capacity = VehicleQuantity
        self.demand = []
        while demand:
                self.demand.append(demand.pop(0)) # the need of customers, the
depot is responsible for 0, it does not have a need

        self.vehicles = math.ceil(float(sum(self.demand) / self.capacity))
        # required vehicles amount count
        print('Required vehicles amount:', self.vehicles)

        self.INF = infinity
        self.rate = rate
        if isinstance(data[0], list):
            self.n = len(data)
            self.M = []
            for row in data:
                self.M.append([self.INF if x == infinity else x for x in row])
        else:
            self.n = data.pop(0)
            self.M = [[self.INF] * self.n for _ in range(self.n)]
            while data:
                u = data.pop(0)
                v = data.pop(0)
                weight = data.pop(0)
                if u != v:
                    self.M[u][v] = weight
        self.T_initial = self.T = initial_t if initial_t else 100*math.log(self.n)
        self.T_stopping = stopping_t
        self.regularization_bound = regularization_bound
        self.iteration_bound = iteration_bound
        self.regulator = 1
        self.fitness = initial_fitness
        self.control = self.fitness + 1
        if initial_solution:
```

```
            initial_solution = initial_solution[:-1] if initial_solution[0] ==
initial_solution[-1] else initial_solution
            self.current_solution = initial_solution if len(initial_solution) ==
self.n else self.initialization()
        else:
            self.current_solution = self.initialization()
        self.best_solution = list(self.current_solution)
        self.initial_cost, self.best_cost, self.worst_cost, self.current_cost =
[self.trip_cost(self.current_solution)]*4
        self.cost_list = [self.current_cost]
        self.reheat_x = []
        self.reheat_y = []
        self.detailed_info = learning_plot
        self.silent_mode = silent_mode
        self.temperature_list = []

    def NewRoute_2opt(self, old_path):
        #алгоритм 2-opt для створення нового шляху
        #:param old_path: старий шлях
        #:return: новий шлях

        a, b = np.random.randint(1, len(old_path)), np.random.randint(1,
len(old_path))  # Згенерувати випадкове ціле число від 2 до довжини массиву,
гарантуючи, що перше й останнє дорівнюють 0
        random_left, random_right = min(a, b), max(a, b)  # Сортувати згенеровані
цілі числа
        rever = old_path[random_left:random_right]  # шлях середньої частини
old_path вставляєеться випадковим чином
        new_path = old_path[:random_left] + rever[::-1] + old_path[random_right:]
# 2-opt алгоритм, перевертання та з'єднання нових шляхів

        return new_path

    def trip_cost(self, candidate):
        res = [self.M[u][v] for u, v in zip(candidate[:-1], candidate[1:])]

        address_index = [i for i in range(len(candidate)) if candidate[i] == 0]
        quantity = penalty = [0]*self.vehicles

        return sum(res) + 1000*sum(self.quantity_resource(candidate))



    def quantity_check(self, route, node) -> bool:
        return sum([self.demand[n] for n in route]) + self.demand[node] <=
self.capacity

    def quantity_resource(self, candidate):
        address_index = [i for i in range(len(candidate)) if candidate[i] == 0]
        quantity = penalty = [0]*self.vehicles
```

```
        for i in range(len(address_index) - 1):  # Число між координатами складу
(0) - маршрут кожного автомобіля
                for j in range(address_index[i], address_index[i + 1], 1):
                        quantity[i] += self.demand[candidate[j]]
                # Обчислюється поточну ємність кожного транспортного засобу,
                # переконавшись, що максимальна ємність не може бути перевищена
                penalty[i] = max(0, quantity[i] - self.capacity)
        # Штрафний пункт, щоб запобігти перевищенню місткості транспортного засобу

        return penalty

    def initialization(self):
        node = 0
        res = [node]
        routes = [[] for _ in range(self.vehicles)]

        array = list(range(1, self.n))

        while array:
            _array = list(self.M[node])
            _h = []
            for idx, val in enumerate(_array):
                heappush(_h, (val, idx))
            node = None
            while node not in array:
                node = heappop(_h)[1]

            array.remove(node)
            res.append(node)

            # Додати вузол в маршрут одній з машин
            for i, route in enumerate(routes):
                if self.quantity_check(route, node):
                    route.append(node)
                    break
            else:
      #Якщо вузол не помістився в існуючи маршрути, створити новий маршрут
                routes.append([node])

        res.extend([0] + route for route in routes)

        arr = list(itertools.chain.from_iterable(res[-self.vehicles:])) + [0]
        return arr

    def accept(self, candidate, k, random_acceptance=True):
        res = False
        candidate_cost = self.trip_cost(candidate)
        if candidate_cost < self.current_cost:
            self.current_cost = candidate_cost
            self.current_solution = candidate
            if candidate_cost < self.best_cost:
```

```
                self.best_cost = candidate_cost
                self.best_solution = candidate
                res = True
            elif random_acceptance:
                random.seed(time.time())
                if random.random() < math.exp((self.current_cost -
candidate_cost)*self.regulator/self.T):  # probability function
                    self.current_cost = candidate_cost
                    self.current_solution = candidate
                    res = True
                else:
                    if self.current_cost > self.worst_cost:
                        self.worst_cost = self.current_cost

            self.T *= self.rate
            self.cost_list.append(self.current_cost)
            return res


        def anneal(self):
            k = 1
            while self.T > self.T_stopping:
                self.accept(self.NewRoute_2opt(self.current_solution),k)
                self.temperature_list.append(self.T)
                k += 1
            return []

        def solve(self):
            def sort_order(array):
                idx = array.index(0)
                return array[idx:] + array[0:idx + 1]

            def display(percentage, number):
                print('\r', end='')
                bar = [':']
                space = [' ']
                bar_n = math.ceil(percentage * 50)
                space_n = 50 - bar_n
                print(''.join(bar*bar_n + space*space_n) + '%s' % number, flush=True,
end='')

            if not self.silent_mode:
                if self.detailed_info:
                    print('Initialized: ', self.best_cost, '| Fitness:', self.fitness,
'/', self.control)
                else:
                    print(''.join(['FITNESS'] + [' ']*42), 'COST')

            last_best = self.current_cost + 1
            while self.current_cost < last_best and len(self.cost_list) <
self.iteration_bound[1] and self.fitness >= 0:
```

```
            if len(self.cost_list) > 1:
                self.T = self.T_initial
                self.regulator = max(self.regulator/(self.control - self.fitness),
self.regularization_bound[0])
                if not self.silent_mode:
                    if self.detailed_info:
                        print('Temper from', self.current_cost, 'at',
len(self.cost_list), '| Fitness:', self.fitness, '/', self.control)
                    else:
                        display(self.fitness/self.control, self.current_cost)
                self.reheat_x.append(len(self.cost_list))
                self.reheat_y.append(self.current_cost)
            last_best = self.current_cost
            last_worst = self.worst_cost
            self.anneal()
            if self.current_cost < last_best:
                self.fitness += 1
                self.control += 1
                if self.current_cost == self.best_cost:
                    self.fitness += 1
                    if self.control <= self.fitness:
                        self.control = self.fitness + 1
                self.regulator = min(self.n/(last_best - self.current_cost),
self.regularization_bound[1])
            else:
                self.fitness -= 1
                if self.worst_cost > last_worst:
                    self.fitness += 1
                    self.control = self.fitness + 1
                last_best = self.current_cost + 1
                if not self.fitness and len(self.cost_list) <
self.iteration_bound[0]:
                    self.fitness = 3
                    if self.control <= 3:
                        self.control = 4
        if self.best_cost > self.INF:
            return self.best_solution, 0  # indication that there might be NO
solution for the problem

        if self.detailed_info:
            fig = make_subplots(rows=1, cols=2, subplot_titles=("Cooling
schedule", "Function value"))

            fig.add_trace(go.Scatter(x=list(range(len(self.temperature_list))),
y=self.temperature_list), 1, 1)
            fig.add_trace(go.Scatter(x=list(range(len(self.cost_list))),
y=self.cost_list), 1, 2)
            fig.add_trace(go.Scatter(x=self.reheat_x, y=self.reheat_y,
mode='markers'), 1, 2)
```

```
            fig.update_xaxes(title='Iteration', col=1, row=1);
fig.update_yaxes(title='Temperature', col=1, row=1)
            fig.update_xaxes(title='Iteration', col=2, row=1);
fig.update_yaxes(title='Trip cost', col=2, row=1)

            fig.show()
        else:
            if not self.silent_mode:
                print(str(self.best_cost) + '.', end='')
            else:
                print()


        return self.best_solution, self.best_cost,
self.quantity_resource(self.best_solution)
```