Вінницький національний технічний університет
Факультет інтелектуальних інформаційних технологій та автоматизації
Кафедра автоматизації та інтелектуальних інформаційних технологій

МАГІСТЕРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

на тему:

«Розробка інтелектуального модуля для розв'язання задачі логістики останньої милі»

Виконав: студент 2-ого курсу групи 1АКІТ-22м
спеціальності 151 – Автоматизація та комп'ютерно-інтегровані технології
Назаренко В. О. _____

Керівник: к.т.н., доцент каф. АІІТ
Іванов Ю. Ю. _____
« 10 » 12 2023 р.

Опонент: к.т.н., професор каф. КСУ
Бицков М. М. _____
« 10 » 12 2023 р.

Допущено до захисту
зав. кафедри АІІТ
д.т.н., проф. Бісікало О. В.
« 11 » 12 2023 р.

Вінниця ВНТУ – 2023 рік

Вінницький національний технічний університет

Факультет інтелектуальних інформаційних технологій та автоматизації

Кафедра автоматизації та інтелектуальних інформаційних технологій

Рівень вищої освіти II-ий (магістерський)

Галузь знань – 15 – Автоматизація та приладобудування

Спеціальність – 151 – Автоматизація та комп'ютерно-інтегровані технології

Освітньо-професійна програма – Інтелектуальні комп'ютерні системи

ЗАТВЕРДЖУЮ

Завідувач кафедри АІІТ

д.т.н., проф. Бісікало О. В.

«20» 09 2023 р.

З А В Д А Н Н Я

НА МАГІСТЕРСЬКУ КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ

Назаренко Віктору Олександровичу

1. Тема роботи: Розробка інтелектуального модуля для розв'язання задачі логістики останньої милі.

   Керівник роботи: к.т.н., доцент каф. АІІТ Іванов Ю. Ю.

   Затверджені наказом ВНТУ від «18» 09 2023 року № 247.
2. Строк подання роботи студентом: до 19.12.2023 р.
3. Вихідні дані до роботи: кількість вузлових точок; координати точок на площині; параметри алгоритму (кількість бджіл, актуальність джерела нектару; кількість ітерацій); кількість запусків для збору статистики; набір тестових задач.
4. Зміст текстової частини: вступ; аналіз методів розв'язання задачі логістики останньої милі; розробка математичного апарату ройового алгоритму для розв'язання задачі логістики останньої милі; розробка інтелектуального програмного модуля та експериментальні дослідження; економічний розділ; висновки; список використаних джерел.
5. Перелік ілюстративного матеріалу: схема роботи рою; приклад поведінки бджіл у природі; приклад роботи алгоритму штучної бджолиної колонії; приклад геометричної постоптимізаційної евристики; результат роботи програми; схема програми.

6. Консультанти розділів магістерської кваліфікаційної роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
1–3	Іванов Ю. Ю. к.т.н., доцент каф. АІІТ	20.09.2023	04.12.2023
4	Козловський В. О., к.е.н., проф. каф. ЕПВМ	23.09.2023	26.11.2023

7. Дата видачі завдання: «20»____09____ 2023 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів магістерської кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Аналіз методів розв'язання задачі логістики останньої милі	до 10.10.23	Виконано
2	Розробка математичного апарату ройового алгоритму для розв'язання задачі логістики останньої милі	до 25.10.23	Виконано
3	Розробка інтелектуального програмного модуля та експериментальні дослідження	до 10.11.23	Виконано
4	Економічний розділ	до 26.11.23	Виконано
5	Оформлення пояснювальної записки та ілюстративного матеріалу	до 04.12.23	Виконано
6	Попередній захист роботи	до 05.12.23	Виконано
7	Остаточний захист роботи	до 19.12.23	Виконано

Студент _____ Назаренко В. О.
(підпис) (прізвище та ініціали)

Керівник роботи _____ Іванов Ю. Ю.
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

УДК 519.161 + 004.023

Назаренко В. О. Розробка інтелектуального модуля для розв'язання задачі логістики останньої милі. Магістерська кваліфікаційна робота зі спеціальності 151 – Автоматизація та комп'ютерно-інтегровані технології, освітньо-професійна програма – Інтелектуальні комп'ютерні системи управління. Вінниця: ВНТУ, 2023. 110 с.

На укр. мові. Бібліогр.: 35 назв; рис.: 33; табл.: 18.

У роботі проаналізовано та модифіковано алгоритм ройового інтелекту на основі поведінки бджіл для розв'язання задачі логістики останньої милі. Представлено його математичний апарат, а також наведено принципи роботи. Розроблено відповідне алгоритмічне та програмне забезпечення, яке дозволяє виконати експериментальні дослідження.

Ключові слова: оптимізація, задача логістики, остання миля, дискретність, ройовий інтелект, карта.

ABSTRACT

NazarenkoV. O. Development of an Intelligent Module for Solving the Last Mile Logistics Task. Master's thesis in specialty 151 – Automation and computer-integrated technologies, educational and professional program – Intelligent computer control systems. Vinnitsa: VNTU, 2023. 110 p.

In Ukrainian language. Bibliography: 35 titles; fig.: 33; tabl.: 18.

In this work was analyzed and modified an algorithm of swarm intelligence inspired by the bees behaviour for solving the last mile logistics task. It mathematical apparatus was presented, as well as the working principles. The algorithms and software, allows us to perform some experimental researches was developed.

Keywords: optimization, logistics task, last mile, discreteness, swarm intelligence, map.

ЗМІСТ

ВСТУП

Актуальність. Для досягнення успіху в технологічних перегонах підприємствам необхідно адаптуватися до динамічних зовнішніх умов. Масова автоматизація процесів створила стрімкий розвиток сфери роботів-кур'єрів. Аналітична компанія IDTechEx говорить про те, що до 2040 року обсяг ринку роботів-кур'єрів може досягти $290 млрд., поясненням чого є здешевлення подібних технологій та підвищення вантажопідйомності таких пристроїв [1, 2].

При цьому, слід зазначити, що існує так звана задача останньої милі, пов'язана з доставкою товару, наприклад, від розподільчого центру або маркетплейсу до дверей клієнта [1]. Цей етап є найважливішою і найдорожчою ланкою транспортної логістики. Витрати на останню милю можуть досягати понад 50% загальних витрат ланцюга постачання. Одним із можливих рішень може бути використання вуличних дронів-кур'єрів як наземних, так і повітряних. Багато компаній активно розробляють та тестують, а деякі вже ввели в експлуатацію таких роботів [3].

Основним завданням таких роботів є доставка товарів з точки відправки в точку замовлення за мінімальний час найкоротшим циклічним маршрутом. Оскільки роботи працюють автономно і без участі людини, самостійно розпізнаючи об'єкти зовнішнього світу та прокладаючи собі шлях, виникає задача оптимізації такого маршруту, яка на практиці зводиться до розв'язання варіацій задачі комівояжера [1, 4].

Останнім часом спостерігається збільшення кількості наукових робіт різних авторів, які досліджують задану задачу з використанням ройового

інтелекту. Наприклад, можна виділити наукові роботи М. Сахіна [5], Д. Карабоги [6], Дж. Ксі [7], І.А. Хижняка [8] та інших. Вони представляють алгоритм оптимізації, інспірований поведінкою бджіл у колонії, як один із найбільш ефективних та зручних алгоритмів, який можна використовувати для розв'язання різноманітних оптимізаційних задач. Тому актуальним є застосування даного алгоритму для розв'язання задачі логістики останньої милі та розробка модифікацій, які підвищать ефективність пошуку розв'язку.

Мета і задачі дослідження. Метою роботи є підвищення ефективності розв'язання задачі логістики останньої милі з використанням алгоритму штучної бджолиної колонії за рахунок застосування нової моделі поведінки бджіл.

Для досягнення мети необхідно розв'язати наступні *задачі*:

– розглянути постановку задачі логістики останньої милі та проаналізувати алгоритми її розв'язання;

– модифікувати математичну модель алгоритму штучної бджолиної колонії;

– розробити програмні засоби та оцінити ефективність розробки.

Об'єктом дослідження є оптимізація циклічного маршруту для даної задачі логістики.

Предметом дослідження є моделі, методи та інструментальні засоби для пошуку мінімального маршруту в заданих умовах.

Методи дослідження. У роботі використано поняття теорії оптимізації та ройового інтелекту під час дослідження роботи алгоритму штучної бджолиної колонії у ході розв'язання задачі оптимізації маршруту. Для аналізу

та перевірки достовірності отриманих теоретичних положень застосовано експериментальне дослідження.

Наукова новизна отриманих результатів полягає у наступному: запропоновано модифікацію алгоритму штучної бджолиної колонії, особливістю якої є математична модель поведінки бджіл, що дозволяє підвищити точність розв'язання задачі логістики останньої милі.

У роботі отримані результати, які мають *практичну цінність*, а саме: наведено експериментальні дослідження ефективності застосування представленого алгоритму розв'язання задачі логістики останньої милі.

Апробація результатів та публікації. За результатами роботи опубліковано 2 тези доповіді: науково-практична конференція з міжнародною участю "Актуальні задачі медичної, біологічної фізики та інформатики" (м. Вінниця, 2022) [9], де описано основні принципи роботи ройового інтелекту на прикладі задачі навчання нейромережі; Всеукраїнська науково-практична конференція "Молодь в науці: дослідження, проблеми, перспективи" (м. Вінниця, 2023), де розглянуто алгоритм штучної бджолиної колонії [10]. Основні результати можна використати на практиці, що підтверджено довідкою про впровадження розробки.

1 АНАЛІЗ МЕТОДІВ РОЗВ'ЯЗАННЯ ЗАДАЧІ ЛОГІСТИКИ ОСТАННЬОЇ МИЛІ

1.1 Постановка задачі

Транспортна логістика відіграє важливу роль у роботі практично будь-якого підприємства. Велике значення надається впровадженню методів оптимізації маршрутів, які дозволяють економити до кількох десятків відсотків від вартості товару. Крім доставки товарів, є ще безліч областей, в яких необхідно застосовувати методи оптимізації маршрутів: перевезення людей, туризм, управління супутниками, збирання генома, кластеризація масивів даних, проєктування комунікаційних мереж, з'єднання населених пунктів лініями енергопередавання і газопостачання, проєктування топології інтегральних схем тощо [1, 3, 4, 7, 8, 11, 12]. Оптимальні маршрути потрібні не тільки листоношам, а й ремонтним бригадам, а також тим, хто здійснює доставку різноманітної продукції, перевіряє мережі, тестує сайти і, навіть, роботів.

На сьогоднішній день у логістиці спостерігається тенденція до використання безпілотних технологій як одного з найінноваційних та перспективних напрямків, що обумовлено потенційним зниженням витрат та підвищенням прибутковості діяльності організацій. Часто такі технології використовуються у сфері складської та транспортної логістики, коли безпілотному апарату необхідно знайти наближений до оптимального

маршрут. Наприклад, схема мікрорайону, яким пересуватиметься робот-кур'єр наведена на рисунку 1.1.

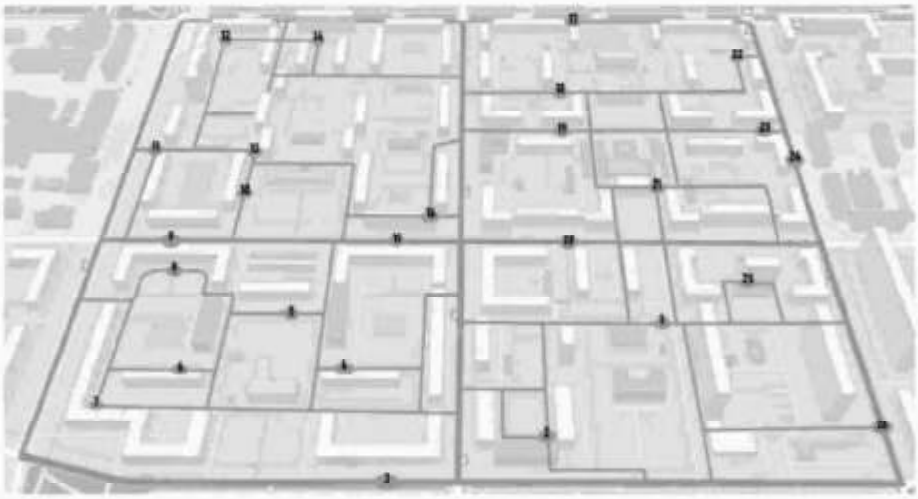


Рисунок 1.1 – Схема мікрорайону для доставки товарів

Загалом розглядається n вузлових пунктів, які пов'язані між собою транспортною мережею. Відома матриця відстаней від кожного пункта до інших [13]

$$L = \begin{pmatrix} \infty & c_{12} & \dots & c_{1n} \\ c_{21} & \infty & \dots & c_{2n} \\ \dots & \dots & \dots & \dots \\ c_{n1} & c_{n2} & \dots & \infty \end{pmatrix}, \tag{1.1}$$

причому в загальному випадку не завжди $c_{ij} = c_{ji}$. Умовно позначимо дороги

$$a_{ij} = \begin{cases} 1, \text{якщо маршрут з } i \text{ - ої точки до } j \text{ - ої існує;} \\ 0, \text{в іншому випадку.} \end{cases} \tag{1.2}$$

Отже, a_{ij} може набувати булевих значень (одиниця або нуль). У ході розв'язання такої оптимізаційної задачі необхідно знайти набір пунктів, при якому забезпечується екстремальне значення цільової функції [13]

$$f(X) \to extremum\ . \tag{1.3}$$

Цільовою функцією є мінімізація всього маршруту [14]

$$L = \min f(\{a_{ij}\}) = \sum_{i=1}^{n}\sum_{j=1}^{n} c_{ij}a_{ij}\ . \tag{1.4}$$

Обмеження щодо одноразового в'їзду в кожен пункт задається таким чином [14]

$$\sum_{i=1}^{n} a_{ij} = 1,\ j = 1,2,...,n; i \neq j\ . \tag{1.5}$$

Обмеження щодо одноразового виїзду з кожного пункта [14]

$$\sum_{j=1}^{n} a_{ij} = 1,\ i = 1,2,...,n; i \neq j\ . \tag{1.6}$$

Зазначені обмеження не повністю описують допустимі маршрути і не виключають можливості розриву маршруту. Щоб усунути цей недолік, введемо невід'ємні цілочисельні змінні $u_i, u_j (i = 2,...,n; j = 2,...,n; i \neq j)$, які в

процесі розв'язування задачі набуватимуть значень порядкових номерів вузлових пунктів за оптимальним маршрутом руху. Запишемо обмеження, які усувають можливість існування підциклів [13, 14]

$$u_i - u_j + na_{ij} \le n-1. \tag{1.7}$$

Тоді нехай робот переїжджає з міста 1 до міста 2 на p-му кроці і допустимо також, що, тоді з міста 2 робот вирушить на наступному $(p+1)$-му кроці, тобто $u_i = p; u_j = p+1$. Звідси випливає, що

$$u_i - u_j + na_{ij} < p-(p+1)+na_{ij} = na_{ij} - 1 \le n-1. \tag{1.8}$$

Отже, якщо вибрано маршрут пересування з i-го міста до j-го, то нерівність (1.8) фіксує два підряд порядкових номери цих міст.

З метою спрощення викладення матеріалу опускається взаємодія робота-кур'єра з людьми та будь-якими перешкодами, а також ігноруються обмеження, які накладаються правилами дорожнього руху. Також у цій роботі не розглядаються обмеження, пов'язані з ємністю акумулятора та запасом ходу робота, не враховуються габарити та вага товарів, які доставляють та вантажопідйомність робота.

Розглянемо наступний приклад. Робот, який виїжджає з контрольної точки 1, має побувати в кожній контрольній точці один раз і повернутися до початку. Знайти найкоротший маршрут, якщо відстані між контрольними точками відомі та наведені на рисунку 1.2.

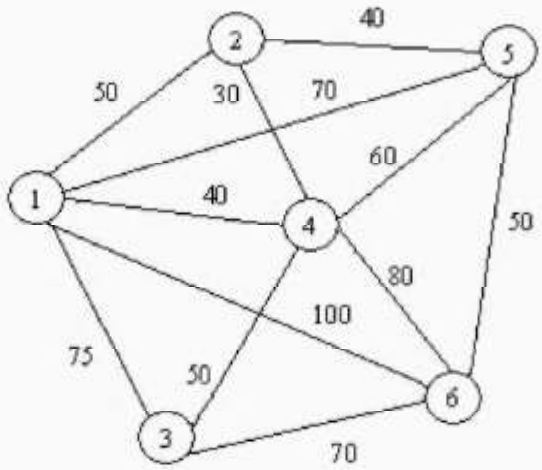


Рисунок 1.2 – Транспортна мережа

Загалом маємо 12 доріг та 6 вершин, в яких має побувати робот. З першої точки можна потрапити до кожної з п'яти інших $a_{12}, a_{13}, a_{14}, a_{15}, a_{16}$. Друга точка пов'язана з трьома (першою, четвертою та п'ятою), а отже, маємо такі три змінні: a_{21}, a_{24}, a_{25}. Аналогічно позначаємо змінні, що відповідають можливим маршрутам руху з третьої (a_{31}, a_{34}, a_{36}), четвертої $(a_{41}, a_{42}, a_{43}, a_{45}, a_{46})$, п'ятої $(a_{51}, a_{52}, a_{54}, a_{56})$ та шостої $(a_{61}, a_{63}, a_{64}, a_{65})$ точок. Отримали 24 булеві змінні (a_{ij} = {0; 1}). Однак деякі змінні, наприклад, та a_{12} та a_{21} описують один маршрут, довжина якого за умовою задачі не змінюється залежно від напрямку пересування. Матрицю найкоротших відстаней S між всіма вершинами можна представити таким чином:

$$C = \begin{pmatrix} \infty & 50 & 75 & 40 & 70 & 100 \\ 50 & \infty & 80 & 30 & 40 & 90 \\ 75 & 80 & \infty & 50 & 110 & 70 \\ 40 & 30 & 50 & \infty & 60 & 80 \\ 70 & 40 & 110 & 60 & \infty & 50 \\ 100 & 90 & 70 & 80 & 50 & \infty \end{pmatrix}.$$

Тоді запишемо критерій оптимальності – мінімізація довжини маршруту комівояжера, враховуючи описані вище умови:

$$L = \min f(\{a_{ij}\}) = 50a_{12} + 75a_{13} + 40a_{14} + 70a_{15} + 100a_{16} + 30a_{24} + \\ + 40a_{25} + 50a_{34} + 70a_{36} + 60a_{45} + 80a_{46} + 50a_{56}. \quad (1.9)$$

Обмеження щодо одноразового в'їзду в кожний пункт сформулюємо у вигляді

$$\begin{cases} a_{21} + a_{31} + a_{41} + a_{51} + a_{61} = 1; \\ a_{12} + a_{42} + a_{52} = 1; \\ a_{13} + a_{43} + a_{63} = 1; \\ a_{14} + a_{24} + a_{34} + a_{54} + a_{64} = 1; \\ a_{15} + a_{25} + a_{45} + a_{65} = 1; \\ a_{16} + a_{36} + a_{46} + a_{56} = 1. \end{cases} \quad (1.10)$$

Обмеження щодо одноразового виїзду з кожного пункту

$$\begin{cases} a_{12} + a_{13} + a_{14} + a_{15} + a_{16} = 1; \\ a_{21} + a_{24} + a_{25} = 1; \\ a_{31} + a_{34} + a_{36} = 1; \\ a_{41} + a_{42} + a_{43} + a_{45} + a_{46} = 1; \\ a_{51} + a_{52} + a_{54} + a_{56} = 1; \\ a_{61} + a_{63} + a_{64} + a_{65} = 1. \end{cases} \quad (1.11)$$

Записуємо умови щодо усунення підциклів

$$
\begin{cases}
u_2 - u_4 + 6a_{24} \le 5; \\
u_2 - u_5 + 6a_{25} \le 5; \\
u_3 - u_4 + 6a_{34} \le 5; \\
u_3 - u_6 + 6a_{36} \le 5; \\
u_4 - u_2 + 6a_{42} \le 5; \\
u_4 - u_3 + 6a_{43} \le 5; \\
u_4 - u_5 + 6a_{45} \le 5; \\
u_4 - u_6 + 6a_{46} \le 5; \\
u_5 - u_2 + 6a_{52} \le 5; \\
u_5 - u_4 + 6a_{54} \le 5; \\
u_5 - u_6 + 6a_{56} \le 5; \\
u_6 - u_3 + 6a_{63} \le 5; \\
u_6 - u_4 + 6a_{64} \le 5; \\
u_6 - u_3 + 6a_{65} \le 5; \\
i = 2,...,6;\ j = 2,...,6; i \ne j; \\
u_i, u_j - \textit{цілі числа}.
\end{cases}
\tag{1.12}
$$

У результаті, розв'язавши задачу, можна отримати оптимальний варіант маршруту, який відображено на рисунку 1.3. Тобто за оптимальним планом необхідно пройти маршрут [1–2–5–6–3–4–1], довжина якого мінімальна і дорівнює 300 одиниць.

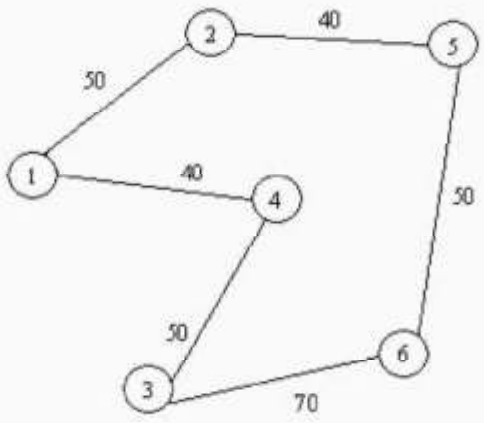


Рисунок 1.3 – Маршрут мінімальної довжини

1.2 Особливості комбінаторної оптимізації

Комбінаторна оптимізація полягає у пошуку деякого оптимального розв'язку на кінцевій множині заданих об'єктів, тобто формулюється екстремальна задача комбінаторної (дискретної) оптимізації. Інтерес до таких задач збільшився в XIX та XX століттях у зв'язку з розвитком теорії графів, програмування, криптографії та теорії ймовірності. Зараз комбінаторні методи застосовуються як і у самій математиці, так і поза нею – теорія кодування, планування експерименту, топологія, кінцева алгебра, математична логіка, теорія ігор, кристалографія, біологія, статистична фізика, економіка тощо.

Слід зазначити, що задачі, як і алгоритми, прийнято класифікувати за складністю. Задача із класу P – проста в обчислювальному плані задача, для якої існують ефективні методи розв'язання, тобто з ростом розмірності її можна розв'язати точно за прийнятний час. Клас NP – це множина задач, які можна розв'язати за поліноміальний час на недетермінованій машині Тюрінга [13, 14].

Для деяких задач виявили дивовижну властивість. Виявилося, що деякі з них універсальні в тому сенсі, що побудова поліноміального алгоритму для будь-якої такої задачі спричиняє можливість побудови такого ж алгоритму для решти задач класу NP. NP-повна задача – це задача із класу NP, до якої можна звести будь-яку іншу задачу із цього класу за поліноміальний час. Задача A поліноміально зводиться до задачі B, якщо існують поліноміальні алгоритми, які перетворюють вихідні дані A на вихідні дані B, і відповідно результат B на результат A [15, 16].

Таким чином, *NP*-повні задачі утворюють у певному сенсі підмножину "найскладніших" задач у класі *NP*; і якщо для якоїсь із них буде знайдено "швидкий" алгоритм розв'язання, то й будь-яка інша задача з класу *NP* може бути розв'язана так само "швидко".

Слід зазначити, що з погляду обчислювальної складності, більшість практичних задач комбінаторної оптимізації відноситься до класу *NP*, тобто їх складність не дозволяє знаходити точне рішення зі збільшенням розмірності задачі за прийнятний час навіть при введенні певних обмежень. Зі зростанням розмірності задачі кількість варіантів перебору зростає екстремально швидко (рисунок 1.4). Подібні задачі цікавлять дослідників і фахівців в області дискретної оптимізації, оскільки допускається застосування різноманітних методів.

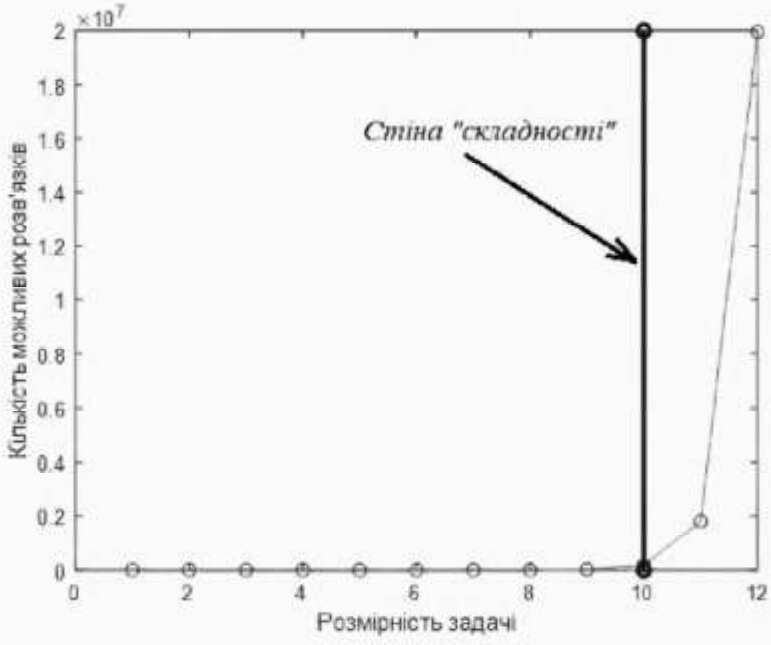


Рисунок 1.4 – Залежність кількості можливих розв'язків від розмірності задачі

Яскравим представником таких задач дискретної оптимізації є задача останньої милі, яку зводять до *NP*-повної задачі комівояжера. У 1859 році знаменитий математик В. Гамільтон запропонував дитячу головоломку, в якій пропонувалося зробити "кругосвітню подорож" по 20 містах, розташованих у різних частинах земної кулі. Кожне місто з'єднувалося дорогами з трьома сусідніми так, що дорожня мережа утворювала 30 ребер додекаедра, у вершинах якого знаходилися міста. Обов'язковою була умова: кожне місто, за винятком першого, можна відвідати один раз (рисунок 1.5). Дана головоломка представляє собою початкове формулювання задачі комівояжера. Математичне формулювання було сформульоване К. Менгером, а пізніше Г. Уітні запропоновано назву задачі [13].

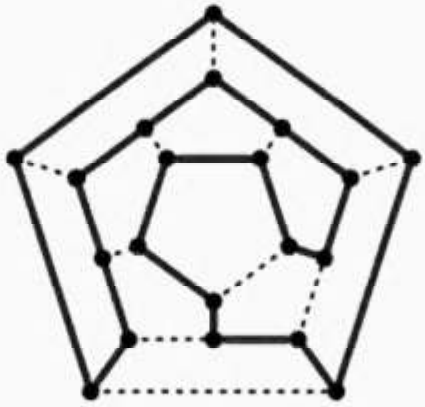


Рисунок 1.5 – Розв'язок головоломки Гамільтона

Тобто задано певну скінченну множину перестановок n елементів (пунктів) у формі $S_n = \{s_i\}$, де s_i – одна перестановка. Цільова функція $f(s_i)$ представляє собою довжину замкнутої ламаної, такої, що проходить через n заданих точок в порядку, який задає перестановка $s_i \in S_n$. Необхідно знайти

найменшу або оптимальну довжину цієї ламаної [2]. Можна виділити такі види задачі комівояжера:

1. Симетрична задача (*TSP*) – відстані задані між будь-якими двома містами однакові (матриця відстаней симетрична). Симетричну задачу називають метричною, якщо відносно довжин шляхів виконується нерівність трикутника, тобто обхідні шляхи довші за прямі (ребро від вершини i до вершини j ніколи не довше за шлях через проміжну вершину k), що можна представити правилом

$$c_{ij} \le c_{ik} + c_{kj}. \qquad (1.13)$$

2. Асиметрична задача (*ATSP*) допускає несиметричність матриці відстаней. У ще більш загальному випадку шляхи між певними містами відсутні.

3. Задача з частковим упорядкуванням (*SOP*) вимагає, щоб вузловий пункт i був відвіданий до пункта j (таких умов може бути декілька).

Вагомий внесок в дослідження даної задачі внесли Дж. Данціг та Д.Р. Фалкерсон, які сформулювали постановку задачі дискретної оптимізації, яку в подальшому інтерпретували до теорії графів та лінійного цілочисельного програмування. У 1972 році Р. Карп довів *NP*-повноту задачі комівояжера та представив теоретичне обгрунтування складності пошуку розв'язків даної задачі на практиці. Програміст-дослідник Г. Райнельт зібрав набір стандартизованих екземплярів задачі комівояжера різного ступеня складності *TSPLIB* для порівняння результатів роботи різних груп дослідників [17].

Отже, задача комівояжера пов'язана з комбінаторним аналізом, теорією графів та лінійним програмуванням. У даному розділі роботи розглянемо різні варіанти математичної постановки задачі комівояжера, суть яких зводиться до загальної задачі оптимізації за певним критерієм (відстань, час, кошти тощо).

Дана задача може бути сформульована як задача на графі в наступній постановці: побудувати граф $G(V, E)$, вершини якого відповідають містам, а дуги відображають комунікації, які сполучають пари міст. Цикл, який включає кожну вершину заданого графа G тільки один раз (за винятком стартової вершини циклу), називається маршрутом комівояжера (рисунок 1.6) [13].

Рисунок 1.6 – Приклад розв'язку задачі комівояжера

Геометрично граф G – це фігура на площині, яка задається множиною точок (вершин) $V = \{v_1, v_2, \ldots, v_n\}$ і множиною ліній (ребер) $E = \{e_1, e_2, \ldots, e_m\}$, що сполучають між собою вершини. Таким чином, граф G повністю задається парою (V, E). Якщо ребра з множини E орієнтовані, а це зазвичай показується

стрілкою, то вони називаються дугами, а граф називають орієнтованим, в протилежному випадку неорієнтованим (рисунок 1.7) [19].

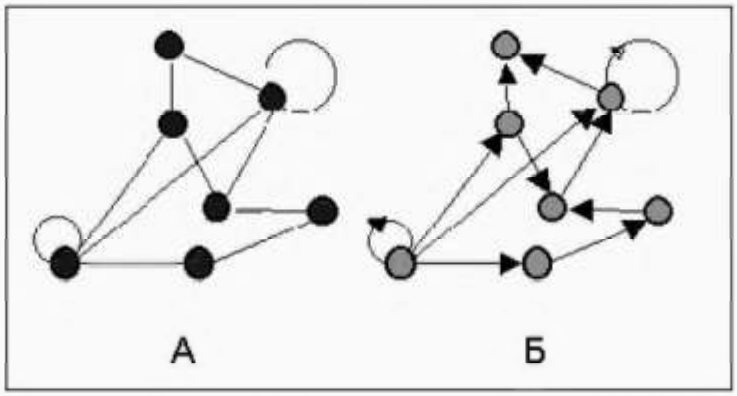


Рисунок 1.7 – Приклад неорієнтованого (А) і орієнтованого (Б) графів

Шляхом (орієнтованим маршрутом) орієнтованого графа називається послідовність дуг, в якій кінцева вершина будь-якої дуги, відмінної від останньої, є початковою вершиною наступної. Орієнтованим циклом (ейлеровим) називається такий шлях, в якому кожна дуга використовується не більше одного разу, а відповідний граф має назву ейлерового. Простим циклом (гамільтоновим) називається такий шлях, в якому кожна вершина використовується тільки раз, а відповідний граф є гамільтоновим [19].

Якщо G – неорієнтований граф з n вершинами, то остовним деревом (остовом, каркасом) даного графа називається будь-який остовний підграф G, який є деревом (графом, що не має циклів, в якому кожна пара вершин сполучена одним ребром, а вершини не використовуються більше разу). Наприклад, якщо G – граф, показаний на рисунку 1.8А, то графи на рисунках 1.8Б та 1.8В є остовом цього графа [19].

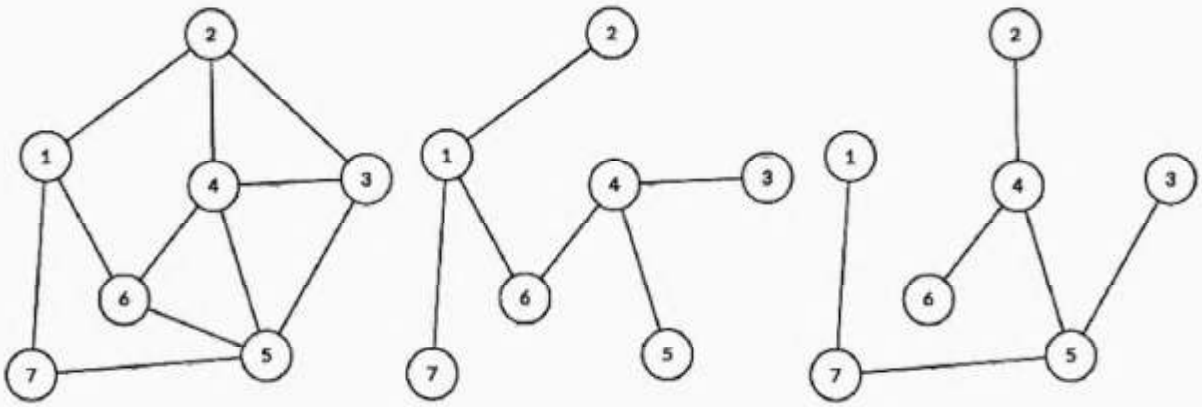


Рисунок 1.8 – Граф G (А) та його остови (Б, В)

Іноді ребрам графа G приписуються числа – ребру (v_i, v_j) ставиться у відповідність деяке число c_{ij}, назване вагою або довжиною. Тоді граф G називається графом зі зваженими ребрами. Іноді ваги приписуються вершинам v_i графа, і тоді виходить граф зі зваженими вершинами. Якщо в графі ваги приписані і ребрам, і вершинам, то він називається просто зваженим. При розгляді шляху, який представлено послідовністю ребер, за його вагу (довжину) береться число L, що дорівнює сумі ваг всіх ребер, що входять у даний маршрут (рисунок 1.9) [19].

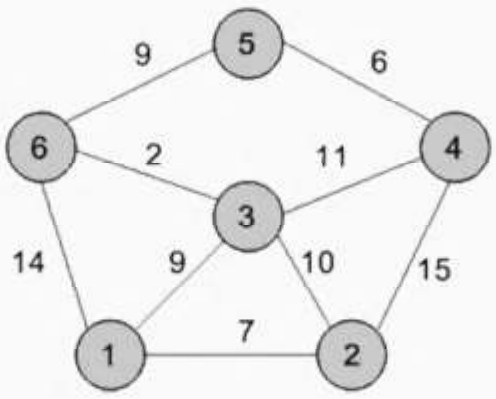


Рисунок 1.9 – Граф зі зваженими ребрами

Існує ряд способів представлення графа у пам'яті комп'ютера (матриці інциденції та суміжності, доступності, список ребер тощо). У даній роботі будемо використовувати найбільш поширений спосіб – матрицю суміжності, яка зручна для внутрішнього представлення даних [13].

Матриця суміжності для неорієнтованого графа буде симетричною відносно своєї головної діагоналі, а для орієнтованого – несиметричною. Матриця суміжності S представляє собою квадратну матрицю розміром m^2 (m – кількість вершин в графі), яка заповнена вагами за наступним правилом: якщо в графі є ребро e, яке сполучає вершини v_i і v_j, то $S(v_i, v_j) = weight(e)$, інакше $S(v_i, v_j) = 0$ [13]:

$$S(v_i, v_j) = \begin{cases} weight(e), & якщо\ e(v_i, v_j) \neq 0; \\ 0, & якщо\ e(v_i, v_j) = 0. \end{cases} \tag{1.14}$$

Отже, необхідно знайти цикл на графі G, який має найменшу довжину або є оптимальним маршрутом комівояжера. Але невідомо жодного простого критерію або ефективного алгебраїчного методу, що дозволяє відповісти на запитання, чи існує в довільному графі G гамільтоновий цикл.

Критерії існування гамільтонових графів, які подано в роботах Пошана, Неша-Вільямса, Оре і Дірака, є занадто загальними і не придатні для довільних графів, які зустрічаються на практиці (наприклад простий граф-каркас шестикутника). Можна привести відомі достатні умови (сильну – із теореми Оре, та більш слабку – із теореми Дірака) [13, 18].

Теорема О. Оре: Нехай G – кінцевий та простий (з кінцевою кількістю вершин та ребер та без кратних ребер) граф, який має $n > 2$ вершин. Позначимо через *deg* степінь певної вершини в G, тобто число інцидентних цій вершині ребер. Тоді, якщо

$$\deg(v) + \deg(w) \geq n \tag{1.15}$$

для будь-якої пари несуміжних (з'єднаних одним і тим же ребром) вершин v і w графа G, то граф є гамільтоновим.

Теорема П. Дірака. Нехай G – кінцевий та простий граф, який має $n > 2$ вершин. Тоді, якщо

$$\deg(v) \geq n/2 \tag{1.16}$$

для будь-якої вершини v графа G, то граф є гамільтоновим.

Узагальненням цих двох теорем є теорема Бонді-Хватала: граф G є гамільтоновим тоді і лише тоді, коли його замикання – гамільтоновий граф. Для графа G з n вершинами замикання будується додаванням до G ребра (v, w) для кожної пари несуміжних вершин v і w, сума ступенів яких не менша n.

Взагалі існує ряд методів для знаходження гамільтонового графа [18–20]:

– алгебраїчний метод заснований на роботах Йоу-Даніельсона-Дхавана включає в себе побудову шляхів за допомогою простого множення матриць.

– мультимаршрутний метод Селбі: крім побудови шляхів, що допомагають швидше знайти гамільтоновий цикл або вказують на його відсутність, значна увага приділяється частині графа, яка залишилася поза цими шляхами. За допомогою певної процедури деякі вершини можуть бути видалені, що дозволяє суттєво зменшити кількість кроків пошуку циклу;

– метод перебору Робертса-Флореса: на противагу алгебраїчним методам, за допомогою яких намагаються відразу знайти всі гамільтонові цикли, і при реалізації яких доводиться зберігати всі шляхи, які можуть виявитися частинами таких циклів, метод перебору має справу з одним шляхом, що безперервно подовжується аж до моменту, коли отримано гамільтоновий цикл, або стає ясно, що цей шлях не може привести до гамільтонових циклів. Тоді маршрут модифікується деяким систематичним способом (який гарантує, що врешті-решт будуть вичерпані всі можливості), після чого продовжується пошук гамільтонового циклу. У цьому методі для пошуку не потрібен великий об'єм пам'яті та за один раз знаходиться один гамільтоновий цикл.

Алгебраїчні методи знаходження гамільтонових циклів не можуть бути застосовані до задач з більш ніж кількома десятками вершин, оскільки вони потребують занадто великого часу роботи. Більш прийнятним є підхід Робертса-Флореса, який не має надмірних вимог до пам'яті комп'ютера, але час роботи якого експоненціально залежить від кількості вершин графа.

У реальних задачах цілочисельних значень набувають не всі, а одна або декілька змінних. Такі задачі називають частково цілочисельними. Сюди

можна віднести задачу про призначення, про найкоротший маршрут у графі, про комівояжера тощо [19].

1.3 Огляд методів розв'язання

Дана задача розв'язується за допомогою низки методів – точних та евристичних. Усі ефективні методи (що скорочують повний перебір) – евристичні. Евристика – це процедура, яка знаходить певний допустимий розв'язок $\widetilde{X} \in D$, який може не співпадати з оптимальним (бути квазіоптимальним). Звичайно, хотілося б, щоб $\widetilde{X}$ співпадав з оптимальним розв'язком X^*, а $f(\widetilde{X}) = f(X^*)$. Але для більшості евристик можна тільки сподіватися (і для деяких довести асимптотичну збіжність), що $f(\widetilde{X})$ наближається до оптимуму $f(X^*)$ [9, 11].

У залежності від того, обчислює евристика новий маршрут або намагається покращити вже існуючий, евристичні методи поділяють на конструктивні та постоптимізаційні. Найчастіше використовують any-time алгоритми, тобто ті, які поступово покращують деяке поточне наближене рішення.

Точні методи гарантовано знаходять оптимальний шлях, потребуючи значних часових витрат, тому на практиці їх застосовують до задач відносно малих розмірностей. Евристичні методи можуть знайти субоптимальний розв'язок, причому за малий час. Їх можна поділити на три групи:

конструктивні та стохастичні алгоритми (комбінують методи пошуку локальних та глобальних розв'язків), а також алгоритми локальної оптимізації.

Метод повного перебору дозволяє знайти розв'язок у будь-якому випадку, але, як правило, розмірність задачі настільки велика, що розв'язати її простим перебором варіантів не можна, оскільки гамільтонових циклів на графі може бути дуже багато, а потрібно визначити найменший із них. Виникає проблема "прокляття" розмірності. Більш вдало сказати, навіть "комбінаторний вибух" – зростання часової складності алгоритму при збільшенні кількості вхідних даних задачі [18]. Зі зростанням кількості вершин графа n кількість варіантів перебору z зростає екстремально швидко:

$$z = \begin{cases} \dfrac{(n-1)!}{2} \text{ для симетричної задачі комівояжера;} \\ (n-1)! \text{ для асиметричної задачі комівояжера,} \end{cases} \quad (1.17)$$

Можна запропонувати таку схему розв'язання задачі (рисунок 1.10):

Крок 1. Згенерувати всі можливі перестановки n вершин графа G.

Крок 2. Розрахувати для кожної перестановки довжину маршруту L.

Крок 3. Вибрати найкоротшу довжину L_{min}.

Метод гілок і меж – загальний алгоритмічний метод, який широко застосовується для розв'язання задач дискретної оптимізації. По суті, є вдосконаленим методом повного перебору з послідовним відсівом рішень, що здаються невигідними. У основі методу гілок і меж лежить ідея послідовного розбиття множини рішень (стратегія "розділяй і володарюй") шляхом розгалуження і знаходження оцінок (меж) для кожної множини рішень.

Внаслідок того, що в процесі роботи деякі рішення не розглядаються, метод гілок і меж не може гарантувати знаходження точного рішення задачі. Відкинуте на початковому етапі "невигідне" рішення може опинитися врешті-решт кращим. Зазвичай необхідно повторити алгоритм для n-2 рівнів дерева, де n – кількість пунктів [19].

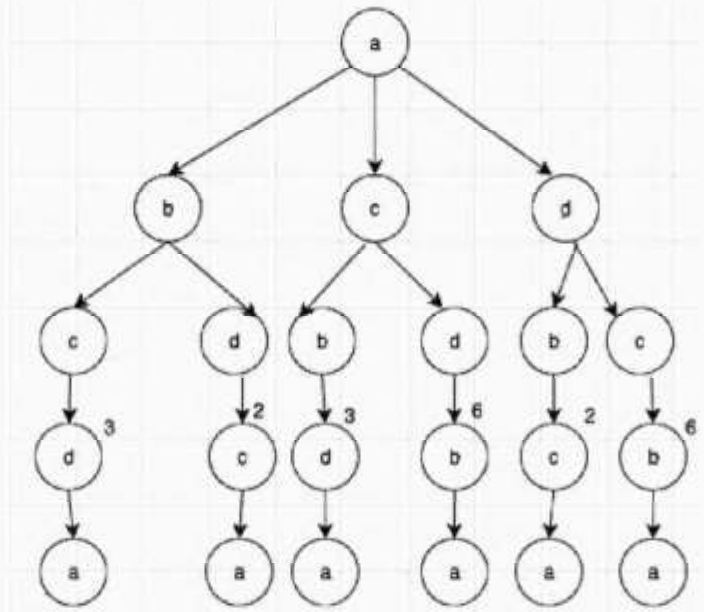


Рисунок 1.10 – Приклад дерева для повного перебору маршрутів

Вважається універсальним методом, оскільки підходить для рішення асиметричної задачі комівояжера, тоді як інші методи пристосовані в основному для вирішення симетричної задачі. Вперше метод гілок і меж був запропонований у 1960 р. стосовно задач лінійного цілочисельного програмування. У 1963 р. групою авторів була запропонована модифікація методу гілок і меж, розроблена спеціально для пошуку найменшого гамільтонового циклу на графі. У результаті з'являється багато робіт, які досліджують особливості використання методу гілок і меж та його

модифікацій для розв'язання різних прикладних задач. Такий великий успіх пояснюється тим, що автори першими звернули увагу на широту можливостей методу, відзначили важливість використання специфіки задачі.

Складність даного алгоритму оцінюють як $O(n^2 \cdot c^n)$, де c – деяка константа. Але задовільних теоретичних оцінок швидкодії алгоритму немає. Практика показує, що на сучасних ЕОМ він дозволяє знайти оптимальний розв'язок для графів з кількістю вершин $n \leq 100$. Слід зазначити, що маючи достатньо часу, можна обчислити найкоротший маршрут, оскільки зі збільшенням витрат часу алгоритм вироджується в повний перебір.

Алгоритм Крістофідеса (метод мінімального остовного дерева або "дерев'яний" алгоритм) базується на пошуці остовного дерева графа мінімальної ваги [13]:

Крок 1. Будується остовне дерево (каркас) мінімальної ваги. Для цього можна застосувати алгоритми Пріма, Краскала або Борувки.

Крок 2. На множині вершин остовного дерева, які мають непарну степінь, знаходиться парасполучення мінімальної ваги. Таких вершин в будь-якому остовному дереві парна кількість.

Крок 3. Остовне дерево перетворюється в ейлеровий граф шляхом приєднання ребер відповідно до знайденого парасполучення.

Крок 4. У отриманому графі знаходиться ейлеровий цикл (алгоритми Флері або Хіергольцера).

Крок 5. Ейлеровий цикл перетворюється в гамільтоновий за допомогою видалення повторів вершин.

Всі етапи реалізуються за поліноміальний час $O(n^3)$. Відомо, що довжина гамільтонового циклу, знайденого даним алгоритмом, не більша за $1{,}5 \cdot L_{min}$.

Жадібний алгоритм має складність $O(n^2 \cdot \log_2(n))$, а задати його можна такою послідовністю кроків [18, 21]:

Крок 1. Відсортувати всі ребра за їх довжиною у зростаючому порядку.

Крок 2. Вибрати найкоротше ребро і додати його до маршруту, якщо дана дія не порушує заданих обмежень.

Крок 3. Повторити крок 2, поки у маршруті не буде n ребер.

Алгоритм найближчого сусіда – один з найпростіших евристичних жадібних методів. Формулюється таким чином [19]:

Крок 1. Вибрати випадкову вузлову точку.

Крок 2. Послідовно включати до маршруту нові точки, причому, кожен черговий пункт повинен бути найближчим до останнього вибраного пункту.

Алгоритм простий в реалізації, швидко виконується, має складність $O(n^2)$, але, як і інші "жадібні" алгоритми, може видавати неоптимальні рішення. Для будь-якої кількості міст $n > 3$ можна підібрати таке їх розташування (значення відстаней між вершинами графа і початкову вершину), що даний алгоритм буде видавати найгірше рішення.

Методи лінійного програмування, які базуються на симплексному методі та методі Куна-Манкреса (угорський алгоритм) для задачі про призначення, використовують математичний апарат лінійного програмування. Спочатку математик Г. Кун дав йому ім'я "угорський метод" у зв'язку з тим, що алгоритм значною мірою заснований на ранніх роботах двох угорських математиків (Кеніга та Егерварі). Англієць Д. Манкрес у 1957 році зауважив,

що алгоритм є поліноміальним. Часова складність оригінального алгоритму була $O(n^4)$, проте Дж. Едмондс і Р. Карп показали, що його можна модифікувати так, щоб досягти часу виконання $O(n^3)$. Л.Р. Форд та Д.Р. Фалкерсон поширили метод на загальні транспортні задачі [19].

Ефективними є методи локальної оптимізації маршруту, коли у побудованому маршруті вибирається декілька близьких один до одного вузлів і змінюється конфігурація ребер між ними, поки не отримано найменшої довжини. Узагальнення цього простого принципу є основою для алгоритму Ліна-Кернігана, реалізованого в 1965 році. Кількість з'єднань між вузлами, які приймають участь в перестановці, визначають складність методу локальної оптимізації (коефіцієнт складності k). Його часова складність складає $O(n^k)$. Тому такі евристики отримали назву *k-opt* евристик. Правила перетворення наведені на рисунках 1.11 та 1.12 [22, 23].

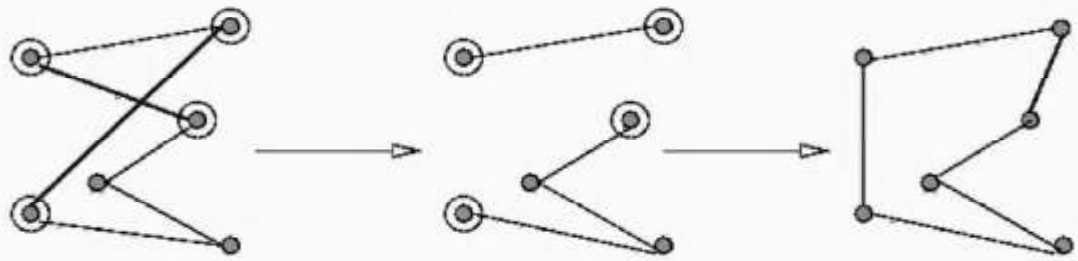


Рисунок 1.11 – Заміна ребер 2-*opt* евристикою

Метаевристичні методи включають сучасні стратегії пошуку, які інспіровані живою та неживою природою. До них прийнято відносити алгоритм імітації відпалу, пошук із заборонами, імовірнісні жадібні алгоритми, генетичні алгоритми, рою часток, зграї птахів, наслідування

поведінки мурашиної колонії, вовків, котів, кажанів, косяка риб, бактеріального рою, рою бджіл, імунних систем, нейромережі тощо [10, 11].

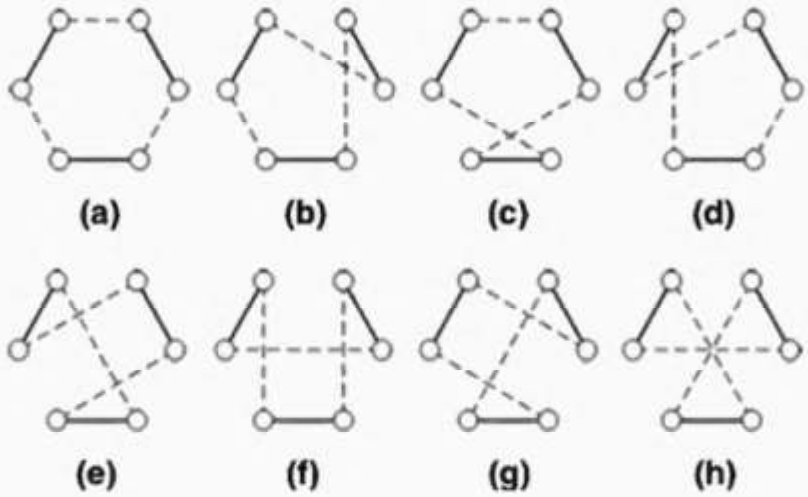


Рисунок 1.12 – Заміна ребер 3-*opt* евристикою

Ідея цих методів заснована на припущенні, що цільова функція $f(X)$ має багато локальних екстремумів, а аналіз усіх допустимих рішень неможливий, не дивлячись на їх кінцеву кількість. У такій ситуації потрібно зосередити пошук на найбільш перспективних частинах області допустимих рішень. Таким чином, задача зводиться до виявлення таких частин і швидкого їх перегляду. У принципі, цими методами можна знаходити як досить якісні розв'язки, так і досить віддалені від оптимального. Якість результатів та тривалість обчислень залежать від реалізації окремих елементів методу.

Алгоритм імітації відпалу, заснований на процесі кристалізації металів. Ідея в тому, щоб привести метал (цільова функція), нагрітий до високої температури (критерій зупинки), в стан з найменшою енергією (мінімальний гамільтоновий цикл), використовуючи контрольоване охолодження. Алгоритм

використовує функції стану (всі можливі маршрути), енергії (найбільш короткий маршрут) і температури [11].

Механізм роботи генетичного алгоритму полягає в пошуку рішення шляхом комбінування параметрів у ході комп'ютерної біологічної еволюції. Відбувається своєрідний природний відбір рішень. Відмінною здатністю алгоритму є використання ряду операторів схрещування і мутації, які проводять рекомбінацію рішень-кандидатів.

Системи з використанням колективу інтелектуальних агентів, як правило, складаються з множини агентів, які локально взаємодіють між собою і з навколишнім середовищем. Самі агенти поодинці, зазвичай, є достатньо простими, але всі разом, локально взаємодіючи, вони створюють так званий ройовий інтелект, який самостійно впродовж певного часу може самонавчатися та самоорганізовуватися (рисунок 1.13) [11, 15].

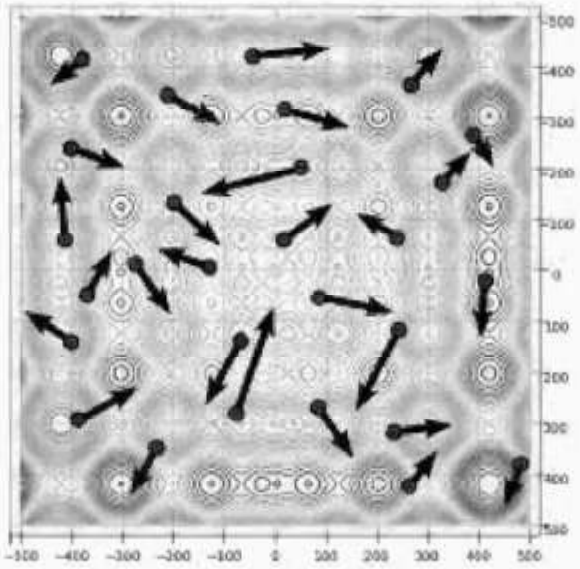


Рисунок 1.13 – Приклад роботи ройового алгоритму

Метод рою часток спочатку був розроблений для моделювання соціальної поведінки та заснований на поведінці зграї птахів. У 1995 році Дж. Кеннеді та Р. Еберхарт запропонували цей метод для розв'язання задач оптимізації. Основна ідея методу полягає у переміщенні часток у просторі рішень. При цьому частка "пам'ятає" найкращу точку в просторі рішень, в якій була, і прагне до неї повернутися. Як зв'язок між частками використовується так звана загальна пам'ять (кожна частка рою знає координати глобально найкращої точки, у якій був рій). Крім того, на рух часток впливають інерційність і випадкові відхилення [11].

Наприклад, алгоритм оптимізації на основі наслідування поведінки мурашиної колонії представляє собою імітацію поведінки колонії мурашок в природі (рисунок 1.14).

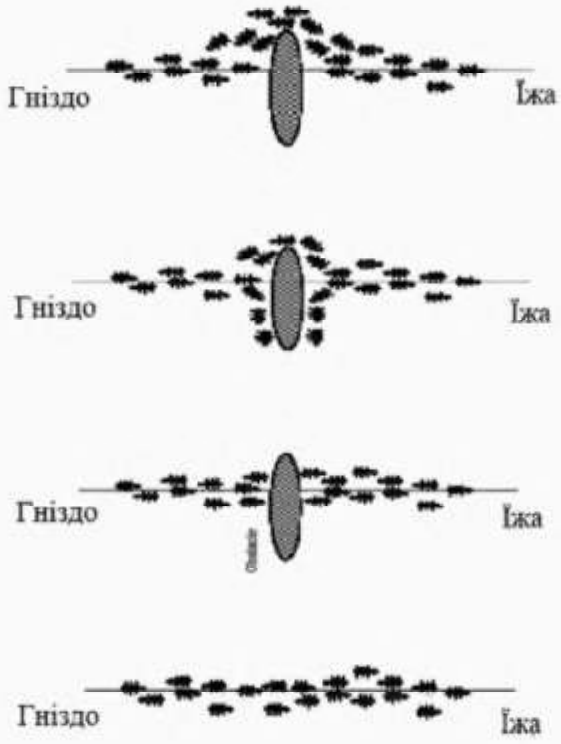


Рисунок 1.14 – Принцип роботи мурашиного алгоритму

Суть роботи полягає у використанні моделі поведінки реальних мурашок, які шукають дорогу від колонії до джерела живлення, відкладаючи феромон. Мурашка, який виявив помічений феромоном шлях, з певною ймовірністю піде ним, зміцнивши своїм власним феромоном. Таким чином, ймовірність того, що в майбутньому інші мурашки будуть рухатися даним шляхом, росте з кількістю мурашок, які раніше використали цей шлях. Це приводить до виникнення найкоротших шляхів, оскільки феромон прагне акумулюватися швидше [11].

Ще одним алгоритмом для розв'язання задачі комівояжера є алгоритм зграї сірих вовків, розроблений у 2013 році С. Мірджалілі. Алгоритм імітує полювання та ієрархічний розподіл зграї вовків. Для того, щоб математично змоделювати соціальну ієрархію вовків використовують найбільш пристосоване рішення – альфа (α), друге та третє найкращі рішення, які називають бета (β) та дельта (δ) відповідно, а решта кандидатів-рішень вважаються омегою (ω) (рисунок 1.15) [24].

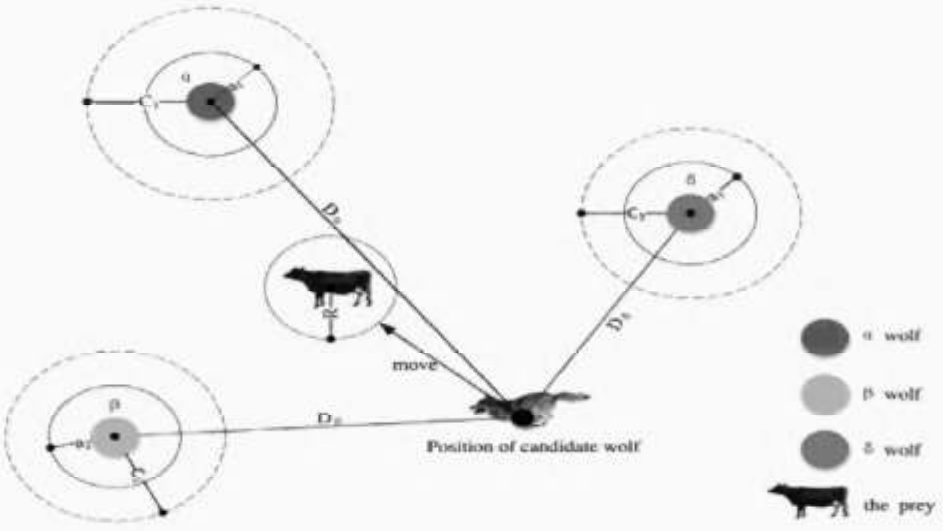


Рисунок 1.15 – Принцип роботи алгоритму зграї вовків

Перший рівень – альфа (α), який є лідером, не обов'язково повинен бути найсильнішим вовком, головне – перевершення інших вовків у керуванні зграєю. Таким чином, він відповідає за прийняття рішень. Другий рівень – бета (β), допомагає альфі приймати рішення. Отже, бета представляє собою радника альфи та вихователя зграї. Третій рівень – дельта (δ), його задача – контролювати омегу (ω); у цю категорію можуть входити скаути, старійшини, мисливці та доглядачі. Нарешті, четвертий рівень – омега (ω), особини звичаної зграї, які поступаються всім домінуючим вовкам.

Алгоритм штучної бджолиної колонії – це алгоритм ройового інтелекту, заснований на імітації поведінки колонії бджіл. Він розроблений турецьким вченим Д. Карабогою у 2005 році для розв'язання задач нелінійної оптимізації, а у 2019 році – задачі комівояжера. Штучна колонія використовує алгоритм подібний до видобутку нектару медоносними бджолами. Для збору нектару в бджолиній колонії застосовуються бджоли-розвідники та бджоли-робітники. Перші проводять дослідження території, що оточує вулик, щодо наявності нектару. Після повернення у вулик вони повідомляють інформацію про кількість нектару, напрям його розташування та відстань до нього. Далі, у найбільш відповідні області вилітають робітники, причому, чим більше нектару в цій галузі, тим більше бджіл вилітає в неї. Алгоритм показав себе дуже ефективним у ході розв'язання низки практичних задач [6, 25–27].

Використання нейромереж грунтується на аналогії з механізмами встановлення впорядкованих нейронних зв'язків. Наприклад, для розв'язання задачі комівояжера за допомогою нейронної мережі Хопфілда потрібно

закодувати маршрут активністю нейронів і так підібрати зв'язки між ними, щоб енергія мережі була пов'язаною з повною довжиною маршруту [18, 28].

Дещо інша ідея лежить в основі методу еластичної мережі, який представляє собою евристичну стратегію пошуку розв'язку задачі, засновану на роботі адаптивної фігури, що змінює форму під дією фізичних сил розтягу-стиску. Тобто у ході застосування даного методу створюється ефект "гумки" та формується "натяг" адаптивного кола, що допомагає знаходити розв'язок задачі на площині.

Алгоритм стартує з розміщення на площині невеликого адаптивного кола (мережі), яке нерівномірно розширюючись проходить практично біля всіх міст і, в результаті, визначає шуканий маршрут. Кожна точка кола рухається під дією двох сил: перша (розтягуюча) переміщує її в бік найближчого міста, а друга (стягуюча) зміщує в сторону її сусідів на колі так, щоб зменшити його довжину. У міру розширення такої мережі, кожне місто виявляється асоційованим з певною ділянкою кола (рисунок 1.16).

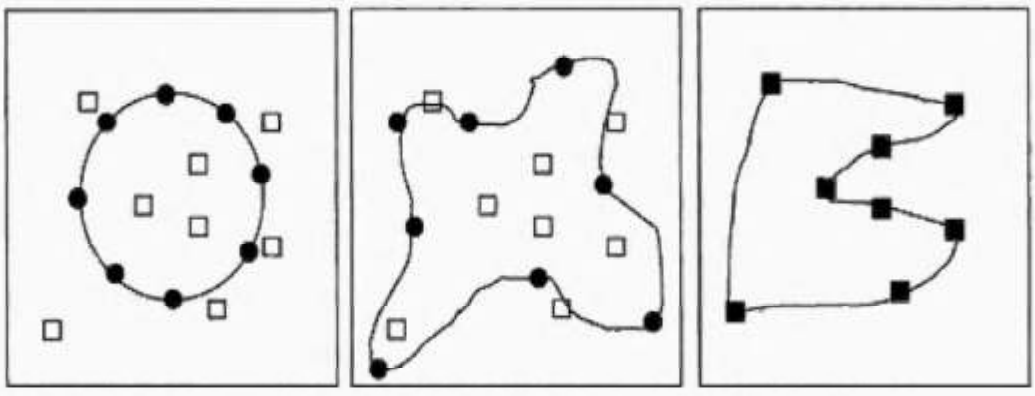


Рисунок 1.16 – Приклад роботи нейронної мережі

Спочатку всі міста мають приблизно однаковий вплив на кожну точку маршруту. В подальшому великі відстані стають менш впливовими, і кожне місто буде більш специфічним для найближчих до нього точок кола. Властивість розтягування кола в сторону концентрації міст особливо важлива на перших етапах роботи. Поступове збільшення специфічності нагадує метод навчання нейромережі Кохонена і контролюється значенням деякого ефективного радіуса мережі.

Проте у цього алгоритму є ряд недоліків, що особливо проявляється під час роботи з нерівномірно розподіленими на карті наборами даних — у такому випадку результат обчислень буде дуже неточним або умова проходження всіх вершин графа не виконається. Тому необхідно використовувати додаткові післяоптимізаційні евристики. Вдале налаштування вхідних параметрів та багаторазове повторення алгоритму дозволяє покращити результат обчислень.

Отже, методи ройового інтелекту намагаються об'єднати основні евристичні методи в рамках алгоритмічних схем більш високого рівня, направлених на ефективне вивчення простору пошуку. Дані методи використовують випадковість, елементи самонавчання, інтенсифікацію і диверсифікацію пошуку, адаптивні механізми управління, конструктивні евристики і методи локального пошуку. Хороша реалізація метаевристики може забезпечити знаходження оптимального рішення за розумний час.

1.4 Висновки

У даному розділі розглянуто комбінаторну природу задачі логістики останньої милі, наведено її математичну постановку. Проведений аналіз показав, що для розв'язання даної задачі при великій розмірності графа потрібно орієнтуватись на методи ройового інтелекту. У роботі розглядається алгоритм бджолиної колонії, математична модель якого буде розглянута у наступному розділі.

2 РОЗРОБКА МАТЕМАТИЧНОГО АПАРАТУ РОЙОВОГО АЛГОРИТМУ ДЛЯ РОЗВ'ЯЗАННЯ ЗАДАЧІ ЛОГІСТИКИ ОСТАННЬОЇ МИЛІ

2.1 Дослідження об'єкта автоматизації

Головною метою автоматизації є спрощення складних функцій діяльності людей у певній галузі. Відповідно об'єктом автоматизації є функції та процеси, які формалізовано виконуються з певною метою. Автоматизована система управління – збірний термін, що має відношення до всього різноманіття комп'ютерних пристроїв, які забезпечують управління різноманітними процесами. Спочатку такі системи розвивалися на виробництві, проте подібність технологічних процесів з процесами роботи різних механізмів дозволяє зараховувати до подібних систем ті, що використовуються в управлінні транспортом, різними інженерними системами, інформацією тощо [29].

У даній роботі розглядається автоматизована інформаційна система, спрямована на обробку інформації. Проєктування такої системи включає розробку структури та алгоритмів обробки інформації, а також загального алгоритму функціонування всієї системи. Для автоматизації управління процесами використовуються технічні засоби, які здатні збирати, обробляти та перетворювати інформацію про стан процесу. Вони також забезпечують передавання інформації через канали зв'язку, формування команд управління,

використання командної інформації для впливу на процес та зв'язок з оператором тощо.

Далі розглядається узагальнена структура, яка показує об'єднання низки програмних модулів, що відповідають за окремі функції, у програмному додатку. Необхідно дослідити процес обробки даних, організацію роботи з ними, дати рекомендації до вибору відповідних параметрів тощо.

Концептуально система з рисунку 2.1 складається з модуля керування та функціональних модулів. Модуль керування відповідає за базові функції для маніпуляції з даними та їх візуалізацією. Центральний модуль керування є ключовою компонентою системи. Він відповідає за координацію та керування роботою інших складових, виконуючи такі важливі функції, як прийняття рішень, розподіл ресурсів, збирання та аналіз даних, керування взаємодією з користувачем та іншими компонентами, введення журналу подій.

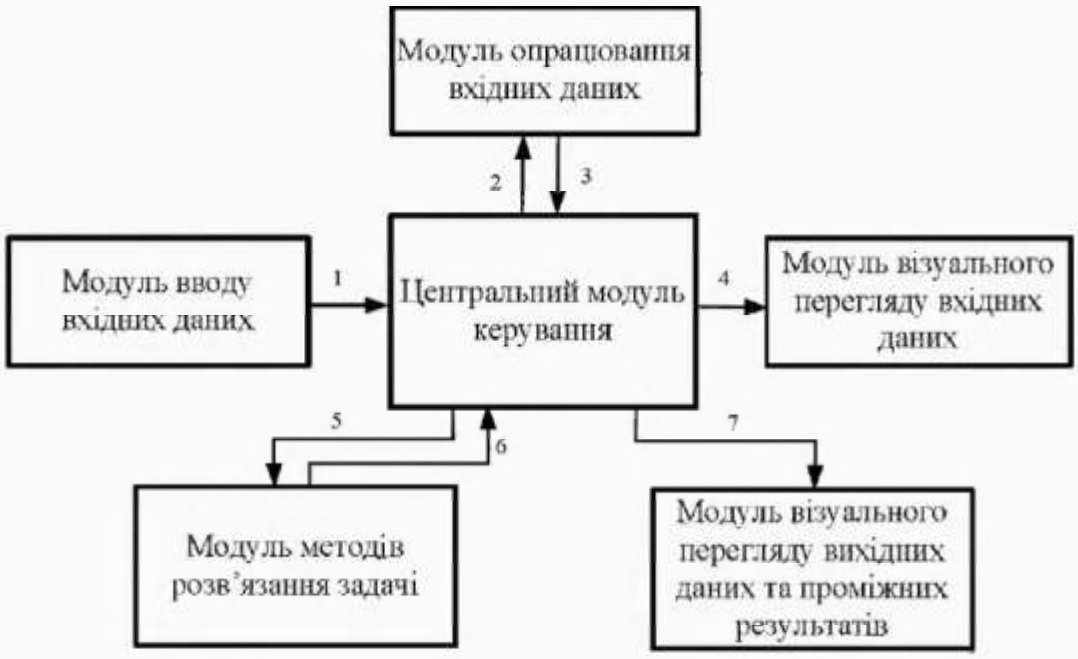


Рисунок 2.1 – Структура розробленої системи

Відповідно функціональні модулі реалізують обробники даних та візуалізатори. Модуль вводу вхідних даних відповідає за збирання та підготовку інформації, яка буде використовуватись в програмі. Його основна функція – отримання даних від користувача або з зовнішніх джерел і їх передавання до центрального модуля керування, який переправляє дані до модуля опрацювання вхідних даних для подальшої обробки. Відповідно далі можна переглянути вибрані зображення в інтерфейсі користувача.

У модулі вибору методів основна орієнтація направлена на методи ройового інтелекту. Вони є потужним і надзвичайно популярним класом оптимізаційних методів, які дозволяють знаходити розв'язок для широкого кола задач з різних предметних областей. Їх сила полягає в здатності розв'язувати найскладніші задачі без знання простору пошуку, поступово покращуючи деякий поточний наближений розв'язок (рисунок 2.2).

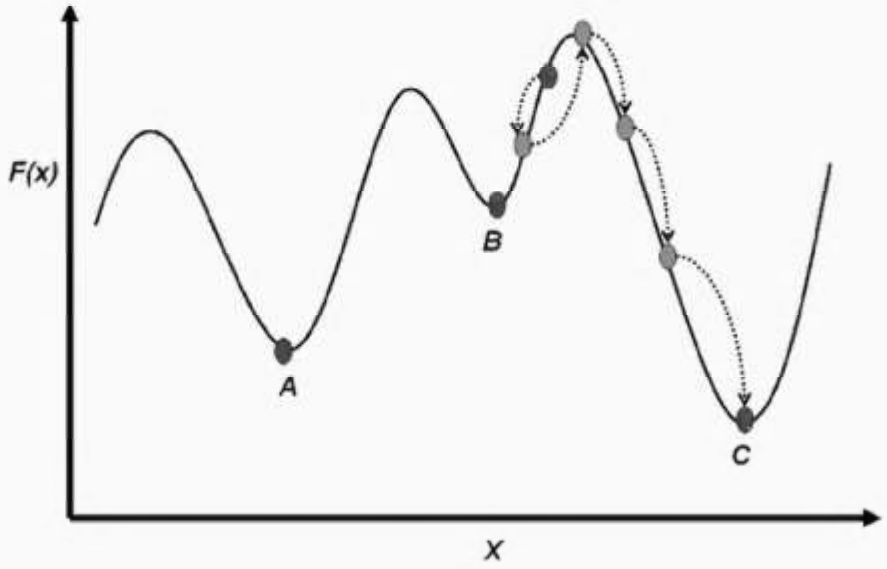


Рисунок 2.2 – Випадковий пошук оптимуму

Спрощено можна розглядати даний клас методів як таких, які реалізують прямий випадковий пошук можливих рішень задачі (оптимальних або близьких до них), поки не буде виконана певна умова або досягнуто граничну кількість ітерацій. На кожній ітерації знаходиться нове рішення задачі, яке базується вже не на одному, а на декількох рішеннях з популяції [10].

Алгоритм рою має свою власну поведінку і характеристики, але можна виділити наступні етапи роботи таких методів [11, 16, 30]:

1. Ініціалізація. Метод знаходження початкового рішення.

2. Окіл. Кожному розв'язку відповідає множина околу.

3. Критерій вибору околу визначається у разі наявності багатьох околів. Цей критерій повинен вказати вибраний окіл і умови його вибору.

4. Відбір кандидатів. Окіл може бути дуже великим. Тоді зазвичай розглядається тільки підмножина переходів на кожній ітерації. Критерій вибору визначає, яким чином можуть бути вибрані рішення для включення у список кандидатів.

5. Критерій прийняття. Переходи оцінюються за допомогою функції g яка залежить від значення цільової функції, штрафів за порушення обмежень тощо. Вибирається найкраще рішення залежно від цього критерію з урахуванням запобігання зацикленню

$$\widetilde{X} = \arg opt(g \mid y = f(X)). \qquad (2.1)$$

6. Критерії зупинки. Алгоритм може бути зупинений згідно різних критеріїв: час обчислень, кількість ітерацій, темп покращення рішення задачі

тощо. Це може бути більш ніж один критерій для управління різними фазами пошуку.

2.2 Узагальнена модель алгоритму штучної бджолиної колонії

Алгоритм штучної бджолиної колонії моделює особливості поведінки медоносних бджіл під час пошуку джерел їжі (нектару на квітці). Серед усіх бджіл виділяють особини, які виконують пошук нектару, а саме: робочих бджіл або фуражирів, бджіл-спостерігачів і бджіл-розвідників. У початковий момент часу існує апріорна інформація про місцезнаходження джерел їжі. Вона зберігається таким чином, що до неї мають доступ усі особини [10].

Робочі бджоли направляються до джерел нектару (існуючих рішень) і проводять пошук в їх околі. Кожному джерелу відповідає одна робоча бджола. Джерело нектару характеризується значущістю. Якщо нове джерело їжі (нова точка на множині допустимих рішень) за певними параметрами (величина фітнес-функції) краще, ніж попереднє джерело, яке збережене в пам'яті бджоли, то вона запам’ятовує саме його. Потім робочі бджоли повертаються у вулик і передають бджолам-спостерігачам інформацію про нові, більш привабливі джерела, вербуючи їх. Кожний вулик має майданчик для танцю, на якому бджоли, які виявили джерела нектару, виконують виляючий танець, намагаючись залучити інших незайнятих бджіл летіти за ними, ”рекламуючи” джерело їжі. Ті, в свою чергу, оцінюють всі знайдені джерела їжі й певним імовірнісним чином вибирають, з якого джерела починати свій пошук [31].

Після цього бджоли-спостерігачі проводять процес пошуку аналогічно тому, як це робили робочі бджоли. Нові рішення також проходять відбір і зберігаються тільки в тому випадку, якщо вони дозволяють покращити якість нектару (величину фітнес-функції). Якщо в процесі пошуку певне рішення не оновлюється протягом заданої кількості ітерацій, то воно вважається "забутим"; бджола, до якої приписане дане джерело нектару, звільняється від роботи і стає бджолою-розвідником, тобто вибирає довільну точку в якості нового джерела нектару (рисунок 2.3) [6].

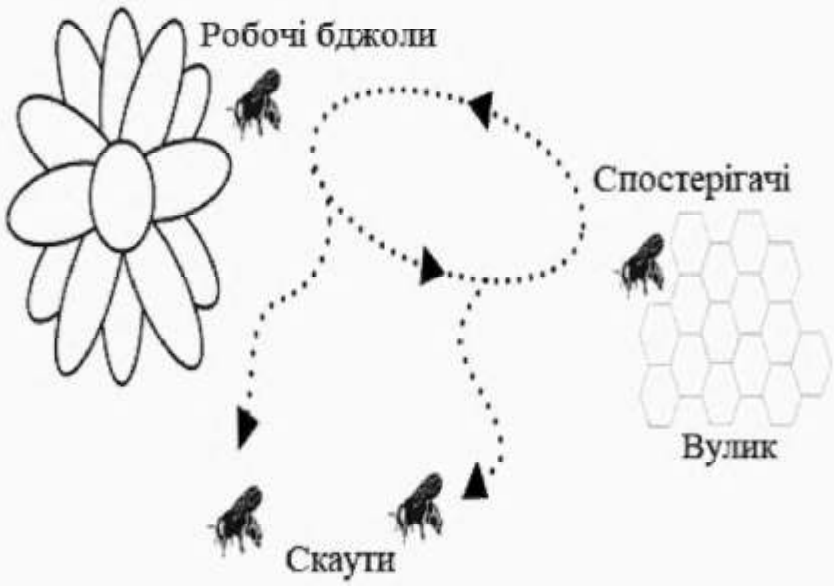


Рисунок 2.3 – Поведінка бджіл у природі

Вперше танець бджіл почав вивчати німецький ентомолог К. фон Фріш у 1920 році. Він отримав за своє відкриття Нобелівську премію. Танець бджоли складається з двох основних стадій: стадія похитування та стадія повернення. Під час першої стадії тіло бджоли здійснює коливальні рухи з боку на бік, рухаючись при цьому вперед прямою лінією. На другій

стадії бджола повертається до вихідної точки свого руху. При цьому повернення до початкової точки танцю відбувається по черзі з правого або з лівого боку.

Таким чином, бджола описує "вісімку" на прямій, якою вона рухається. Фігура танцю вказує розташування джерела нектару; темп танцю залежить від відстані і нектару в ньому, а напрям виляючого танцю на вертикальній соті відповідає напрямку польоту по відношенню до положення Сонця, ультрафіолетові промені якого бджоли розрізняють навіть, якщо небо затягнуте хмарами.

Якщо танець бджоли енергійний, то цим вона ніби радіє і повідомляє, що поруч повно медоносів, відповідно велике джерело нектару і пилок. Якщо ж танець повільний, то це означає, що до їжі доведеться летіти далеко. Сторону польоту бджола задає напрямком танцю. Якщо вона прагне вгору, то необхідно летіти у бік Сонця, якщо вниз – навпаки від нього, вправо або вліво – означає відповідний бік від Сонця. Причому бджола враховує і рух Сонця, відповідно повідомляючи координати. Крім цього, точно відомо, що під час танцю бджола передає ультразвуковий сигнал іншим бджолам.

Також бджоли можуть відчувати запах рослин медоносів на відстані кілометра, тому бджоли-збирачі не тільки спостерігають за танцем, але й запам'ятовують запах нектару і пилку, щоб безпомилково знаходити по наявних орієнтирах потрібне їм джерело їжі (рисунок 2.4).

Розглядаємо задачу мінімізації. Узагальнений алгоритм штучної бджолиної колонії включає в себе такі кроки [6, 8, 10, 12, 24–27, 31]:

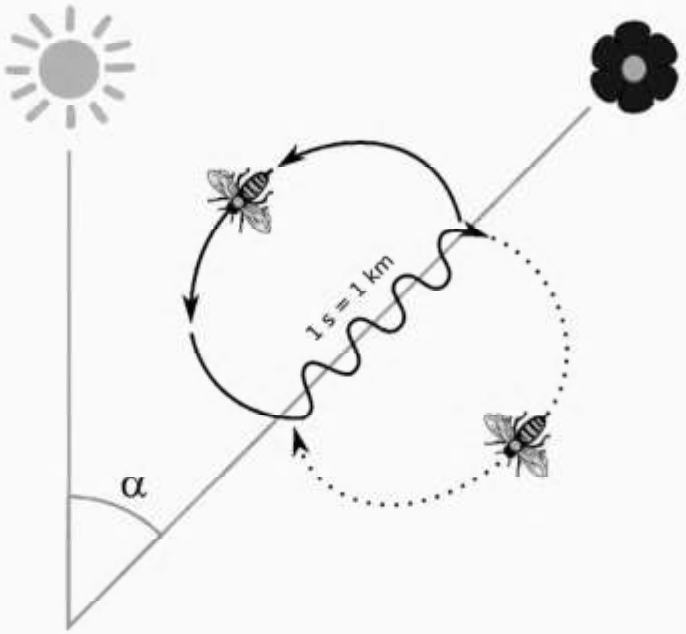


Рисунок 2.4 – Приклад виляючого танцю бджоли

Крок 1. Ініціалізація параметрів алгоритму: $X_i = (x_{i1},...,x_{in})^T$ – позиції джерел нектару, до яких летять E робочих бджіл; $O \geq E$, $S << E$ – кількість бджіл-спостерігачів та бджіл-розвідників; *limit* – рішення вважається забутим, якщо воно не змінювалося впродовж такої кількості ітерацій; K – максимальна кількість ітерацій алгоритму.

Для задачі неперервної оптимізації, якщо при генерації нового рішення його j-та компонента виходить за межі області D, то проводиться декілька спроб згенерувати її заново. Якщо після закінчення k спроб нове рішення постійно знаходиться за межами множини D, то дана координата замінюється на відповідне граничне значення:

$$x_{ij}^{new} = \begin{cases} LB_j, \text{ якщо } x_{ij} < LB_j; \\ UB_j, \text{ якщо } x_{ij} > UB_j, \end{cases} \tag{2.2}$$

де LB_j, UB_j – нижня та верхня межі множини допустимих рішень по j-ій координаті, $UB_j > LB_j$.

Подібну перевірку можна реалізувати програмно без циклу в один рядок:

$$x_{ij}^{new} = \max(\min(x_{ij}, UB_j), LB_j). \tag{2.3}$$

Крок 2. Для кожної робочої бджоли знаходяться координати її положення:

$$X_i = LB + rand \otimes (UB - LB), i = \overline{1, E}, \tag{2.4}$$

де $rand$ – $(n \times 1)$-вектор незалежних дійсних випадкових чисел, рівномірно розподілених на інтервалі [0, 1]; LB, UB – дійсні числа, межі інтервалу пошуку для кожної координати, $UB > LB$; $\otimes$ – поелементне множення.

Для задачі дискретної оптимізації необхідно згенерувати для кожної робочої бджоли перестановку цілих чисел.

Крок 3. Для всіх отриманих рішень (робочих бджіл E) оцінюються джерела їжі – значення цільової функції $f(X_i)$, $i = \overline{1, E}$.

Крок 4. Отримати нові рішення для робочих бджіл E:

$$X_{new\_i} = X_i + U(a,b) \otimes (X_i - X_r), i = \overline{1, E}, \tag{2.5}$$

де $U(a, b)$ – $(n\times 1)$-вектор незалежних дійсних випадкових чисел, рівномірно розподілених на інтервалі $[a, b]$; $a = -1$; $b = 1$; $b > a$; індекс r вибирається випадково з множини $\{1, ..., E\} / \{i\}$, тобто $r \neq i$; $\otimes$ – поелементне множення.

Для задачі дискретної оптимізації аналогічно необхідно згенерувати для кожної робочої бджоли перестановку цілих чисел, застосувавши певні перетворення уже відомих вхідних рішень.

Крок 5. Розрахувати нове значення цільової функції $f(X_{new\_i})$.

Крок 6. Провести селекцію серед нових X_{new_i} та існуючих X_i рішень ($i = \overline{1, E}$) жадібним способом:

$$X_i = \begin{cases} X_{new\_i}, \text{ якщо } f(X_{new\_i}) \le f(X_i); \\ X_i, \quad \text{інакше.} \end{cases} \tag{2.6}$$

Крок 7. Розрахувати ймовірність вибору p_i джерел їжі X_i бджолами-спостерігачами ($i = \overline{1, E}$). Зазвичай використовують колесо рулетки.

Пропорційний відбір. Імовірність вибору p_i бджолою-спостерігачем певного джерела їжі виявляється пропорційною значенню фітнес-функції, тобто кожному джерелу їжі відповідає сектор колеса рулетки:

$$p_i = \frac{F(X_i)}{\sum_{i=1}^{E} F(X_i)}, \tag{2.7}$$

де $F(.)$ – певна функція (залежно від задачі, на максимум або мінімум).

Наприклад, для задачі на мінімум та максимум відповідно можна використати такі функції [24]:

$$p_i = \frac{\exp\left(\frac{-f(X_i)}{f_{mean}}\right)}{\sum_{i=1}^{E}\exp\left(\frac{-f(X_i)}{f_{mean}}\right)}, \tag{2.8}$$

$$p_i = \frac{\exp\left(\frac{f(X_i)}{f_{mean}}\right)}{\sum_{i=1}^{E}\exp\left(\frac{f(X_i)}{f_{mean}}\right)}, \tag{2.9}$$

де $f_{mean} = \frac{\sum_{i=1}^{E} f(X_i)}{E}$ – середнє значення якості джерел їжі на даній ітерації.

Слабка сторона цього підходу у тому, що особини з дуже малим значенням функції пристосованості $f(X_i)$ мало досліджуються, що може призвести до передчасної збіжності алгоритму.

Лінійне ранжування, в якому використовується тільки номер джерела їжі в упорядкованому списку, тоді ймовірність вибору можна описати формулою

$$p_i = \frac{1}{E} \cdot \left(a - (a-b) \cdot \frac{i-1}{E-1}\right), \tag{2.10}$$

де $1 \le a \le 2$ – вибирається випадково; $b = 2 - a$; i – номер джерела їжі в упорядкованому списку [24].

Також можна застосовувати рівномірне ранжування, яке задається наступною формулою [24]

$$p_i = \begin{cases} \dfrac{1}{\mu}, \text{ якщо } 1 \le i \le \mu; \\ 0, \text{ якщо } \mu < i \le E, \end{cases} \tag{2.11}$$

де $\mu \le E$ — параметр (фактично кількість найкращих агентів).

Слід зазначити, що попередньо необхідно відсортувати бджіл за значенням цільової функції (за зростанням для задачі на мінімум або за спаданням для задачі на максимум), зберігши порядкові номера бджіл до сортування. При цьому у відсортованому списку найбільшу ймовірність відбору отримає бджола з номером 1.

Сігма-відсікання виконує масштабування цільової функції за формулою:

$$p_i = \frac{F_\sigma(X_i)}{\sum_{i=1}^{E} F_\sigma(X_i)}, \tag{2.12}$$

$$F_\sigma(X_i) = 1 + \frac{f(X_i) - f_{mean}}{2 \cdot RMSE}, \tag{2.13}$$

$$RMSE = \sqrt{\frac{\sum_{i=1}^{E} (f(X_i) - f_{mean})^2}{E}}, \tag{2.14}$$

де $f_{mean} = \dfrac{\sum_{i=1}^{E} f(X_i)}{E}$ – середнє значення якості джерел їжі на даній ітерації.

Вибір джерела їжі можна ототожнити з вибором числа з інтервалу $[A; B]$, де A та B позначають відповідно початок і кінець фрагмента кола. Отже, у цьому випадку вибір зводиться до вибору числа з інтервалу [0, 1] (відповідного сектору колеса рулетки), яке відповідає конкретній точці на колі (рисунок 2.5). Слід зазначити, що [24]

$$\sum_{i=1}^{E} p_i = 1. \qquad (2.15)$$

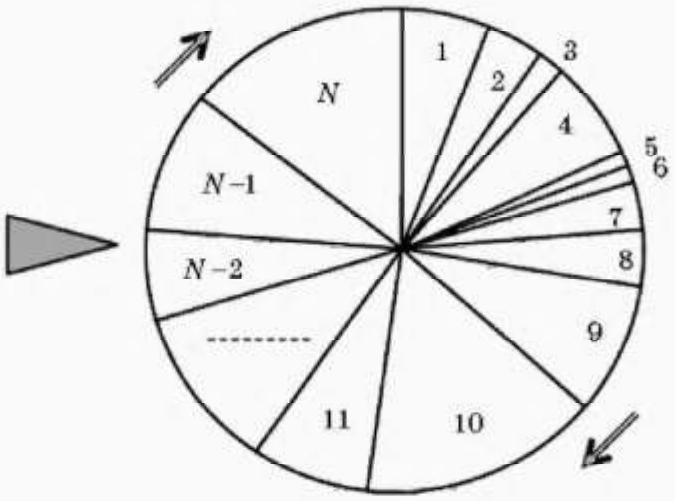


Рисунок 2.5 – Приклад роботи колеса рулетки

Крок 8. Отримати нові рішення для бджіл-спостерігачів O аналогічно рішенням робочих бджіл на кроці 4. Але індекс r вибирається на основі ймовірностей p з кроку 7. Для цього генерується випадкове число *rand* і перевіряється, в який сектор рулетки r воно потрапляє. Для задач дискретної оптимізації тут необхідно виконати низку перетворень, які нагадують процедуру схрещування в генетичному алгоритмі.

Крок 9. Повторити кроки 5 (оцінка джерел їжі) та 6 (селекція рішень).

Крок 10. Генерація нових джерел нектару замість "забутих". Для таких джерел випадково генерують нові рішення на множині допустимих рішень D аналогічно до кроку 2.

Крок 11. Видалити S найгірших рішень і згенерувати нові аналогічно до кроку 2.

Крок 12. Перевіряємо критерій зупинки пошуку (час, кількість ітерацій, стагнація пошуку тощо).

Крок 13. Якщо умова на кроці 12 не виконана, то визначити найкращу бджолу s^{best} та повернутися до кроку 4; інакше – перейти на крок 14.

Крок 14. Визначити та вивести глобально найкраще рішення $f^{best}(X^{best})$ для бджоли s^{best}.

Розглянемо даний алгоритм на прикладі задачі глобальної оптимізації певної функції. Нехай задано $f(x) = x_1^2 + x_2^2 \to \min$, $-5 \le x_1, x_2 \le 5$. Параметри алгоритму задано таким чином: $E = 3$, $O = 3$, $limit_i$ = 6. Нехай функція пристосованості представлена у такому вигляді

$$fit = \begin{cases} \dfrac{1}{1+f_i} & якщо\ f_i \ge 0; \\ 1 + abc(f_i), якщо\ f_i < 0. \end{cases} \qquad (2.16)$$

Ініціалізуємо випадковим чином позиції 3-ох джерел їжі для робочих бджіл, використовуючи рівномірний розподіл в межах [-5, 5] (таблиця 2.1).

Таблиця 2.1 – Ініціалізація колонії

№	x_1	x_2	$f(x_1, x_2)$	*fit*	*best*
1	1.4112	-2.5644	8.5678	0.1045	–
2	0.4756	1.4338	2.2820	0.3047	–
3	-0.1824	-1.0323	1.0990	0.4764	+

Перша робоча бджола (випадкові індекси $k = 2, j = 1$):

$$x_{new\_11} = 2.1644,\ x_{new\_12} = -2.5644.$$

Тоді $f(x_{new\_11}, x_{new\_12}) = 11.26$, застосувавши жадібну стратегію вибору рішення $fit_{new1} = 0{,}0816 < 0{,}1045$. Розв'язок не вдалося покращити, збільшуємо його лічильник "забутості": $limit_1 = 0 + 1 = 1$.

Друга робоча бджола (випадкові індекси $k = 3, j = 2$):

$$x_{new\_21} = 0.4756,\ x_{new\_22} = 1.6217.$$

Тоді $f(x_{new\_21}, x_{new\_22}) = 2.8560$, застосувавши жадібну стратегію вибору рішення $fit_{new2} = 0{,}2593 < 0{,}3047$. Розв'язок не вдалося покращити, збільшуємо його лічильник "забутості": $limit_2 = 0 + 1 = 1$.

Третя робоча бджола (випадкові індекси $k = 1, j = 1$):

$$x_{new\_31} = -0.0754,\ x_{new\_32} = -1.0323.$$

Тоді $f(x_{new\_31}, x_{new\_32}) = 1.0714$, застосувавши жадібну стратегію вибору рішення $fit_{new2} = 0,4828 > 0,4764$. Розв'язок вдалося покращити, скидуємо його лічильник "забутості" до $limit_3 = 0$, виконуємо заміну рішення X_3 на X_{new_3} (таблиця 2.2).

Таблиця 2.2 – Результати першої фази алгоритму

№	x_1	x_2	$f(x_1, x_2)$	*fit*	*best*
1	1.4112	-2.5644	8.5678	0.1045	–
2	0.4756	1.4338	2.2820	0.3047	–
3	-0.0754	-1.0323	1.0714	0.4828	+

Обчислюємо значення ймовірності для вибору рішень. Маємо:

$$p_1 = 0.1172,$$

$$p_2 = 0.3416,$$

$$p_3 = 0.5412,$$

$$\sum_{i=1}^{3} p_i = 0.1172 + 0.3416 + 0.5412 = 1.$$

Генеруємо нові рішення для бджіл-спостерігачів із вибраних рішень X_i залежно від p_i. Перша бджола-спостерігач летить до джерела $i = 3$, друга – до джерела $i = 2$, третя – до джерела $i = 3$. Результати обчислень наведено в таблиці 2.3.

Таблиця 2.3 – Результати другої фази алгоритму

№	x_1	x_2	$f(x_1, x_2)$	*fit*	*best*
1	1.4112	-2.5644	8.5678	0.1045	–
2	0.1722	1.4338	2.0855	0.3241	–
3	0.0348	-1.0323	1.0669	0.4838	+

Однією з основних проблем розробки метаевристик є забезпечення балансу між інтенсивністю і широтою пошуку. Інтенсифікація пошуку вимагає швидкої збіжності алгоритму, що означає швидке зменшення різноманітності популяції. А диверсифікація пошуку навпаки дозволяє забезпечити більш широкий огляд простору пошуку і більш високу ймовірність локалізувати глобальний екстремум задачі, тобто вона вимагає збереження різноманітності популяції якомога більшу кількість поколінь.

Найбільш розвиненими механізмами вирішення даної проблеми є механізми самоадаптації популяційних алгоритмів. У даний час нові високоефективні популяційні алгоритми розробляють за допомогою гібридизації різних метаевристик. Також важливою задачею, у зв'язку з доступністю потужних обчислювальних засобів, є розробка метаевристик з використанням паралелізації обчислень.

Отже, дослідник повинен налаштовувати даний метод до певної конкретної задачі. Не слід забувати, що пошук рішення завжди залишається мистецтвом, якому можна навчитися лише шляхом проб і помилок, застосовуючи різні методи для вирішення конкретних задач.

2.3 Опис удосконалень алгоритму

У даному розділі опишемо ключові особливості дискретної моделі алгоритму штучної бджолиної колонії. Першим кроком є генерація рішень бджолами-робітниками. Для цього запропоновано застосувати алгоритм найближчого сусіда (АНС) для низки вершин графа G.

Наприклад, розглянемо граф з рисунку 2.6. Вибираємо за допомогою генератора випадкових чисел перший вузол – 4. Далі шукаємо найкоротший шлях до іншого можливого вузла, тобто ребро (v, w) з найменшою вагою – це ребро (4, 2) з вагою 3. Ті ж самі дії виконуємо з вузла 2. Загалом сформується маршрут за списком ребер у таблиці 2.4 загальною довжиною L = 106 одиниць.

Рисунок 2.6 – Приклад графа для пошуку маршруту АНС

Наступним кроком є модифікація маршруту робочими-бджолами. Для цього використано операцію реорганізації, яка полягає в наступному [15]:

Крок 1. Випадково відібрати декілька позицій у маршруті P_1 (P_2).

Крок 2. Запам'ятати їх та видалити з маршруту P_2 (P_1).

Крок 3. Скопіювати збережені пункти в маршрут P_2 (P_1) в такому ж порядку, в якому вони розташовані у маршруті P_1 (P_2).

Таблиця 2.4 – Пошук маршруту АНС

№	Ребро (v, w)	Довжина, c
1	(4, 2)	3
2	(2, 7)	5
3	(7, 1)	15
4	(1, 3)	23
5	(3, 6)	4
6	(6, 5)	18
7	(5, 4)	38

Приклад стартових маршрутів наведено в таблиці 2.5.

Таблиця 2.5 – Два випадкові маршрути

P_1	2	3	4	5	6	1
P_2	4	5	2	1	6	3

Наприклад, візьмемо позиції 1, 3 та 4 для маршруту P_2, в яких знаходяться пункти 4, 2 та 1 відповідно. Новий маршрут ініціалізується всіма пунктами маршруту P_1 за виключенням вузлів 4, 2, 1. Замість них на відкриті

позиції копіюємо ці пункти в такому ж порядку, в якому вони розташовані у маршруті P_2 (таблиця 2.6).

Таблиця 2.6 – Приклад реорганізації на основі першого маршруту

A_1	–	3	–	5	6	–
A_1	4	3	2	5	6	1

Аналогічно можна згенерувати іншу перестановку пунктів, змінивши ролі P_1 та P_2. Візьмемо позиції 1, 3 та 4 для маршруту P_1, в яких знаходяться пункти 2, 4 та 5 відповідно. Новий маршрут ініціалізується всіма пунктами маршруту P_2 за виключенням вузлів 2, 4, 5. Замість них на відкриті позиції копіюємо ці пункти в такому ж порядку, в якому вони розташовані у маршруті P_1 (таблиця 2.7).

Таблиця 2.7 – Приклад реорганізації на основі другого маршруту

A_2	–	–	–	1	6	3
A_2	2	4	5	1	6	3

Щоб згенерувати маршрути-кандидати, для бджіл-наглядачів запропоновано використати операції перестановки, зсуву та подвійної інверсії. У запропонованому алгоритмі ці операції спочатку виконуються з однаковою ймовірністю вибору, а в наступних ітераціях вона змінюється відповідно до успіху кожної із них, що підвищує вибір успішної операції з використанням колеса рулетки, також задається лічильник успіху кожної операції [32].

Операція перестановки: два різних цілих числа x і y генеруються випадковим чином (від 1 до розмірності задачі n) та міняються місцями для нового маршруту (рисунок 2.7).

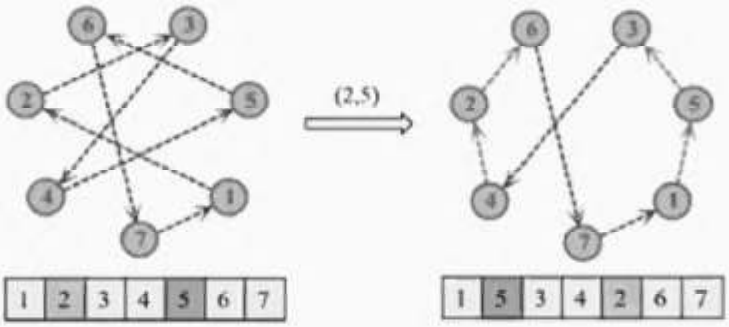


Рисунок 2.7 – Приклад операції перестановки

Операція зсуву: два різних цілих числа x і y генеруються випадковим чином (від 1 до розмірності задачі n). Потім відбувається циклічний зсув на одиницю вліво для їх позицій (рисунок 2.8).

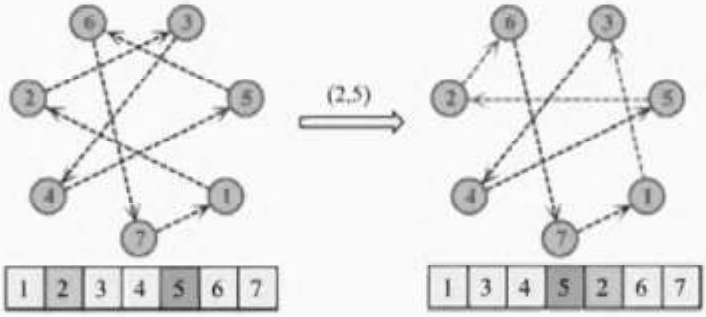


Рисунок 2.8 – Приклад операції зсуву

Операція подвійної інверсії формує значні зміни у заданому маршруті: генеруються дві підпослідовності $A = [x, y]$ та $B = [y + 1, z]$, які не пересікаються, з послідовними позиціями $[A, B]$. Кожна з підпослідовностей

інвертується – A^* =[y, x], B^* = [z, y + 1], а потім послідовності міняються місцями – [B^*, A^*] (рисунок 2.9).

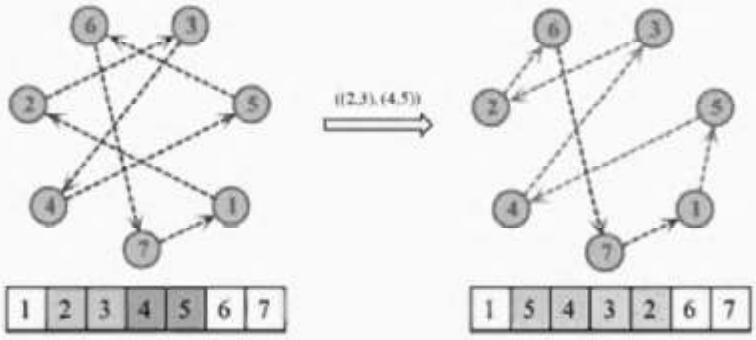


Рисунок 2.9 – Приклад операції подвійної інверсії

Бджіл-розвідників генеруємо для трьох найгірших маршрутів або для маршрутів, які вважаються "забутими", випадковим чином.

Після того, як умова зупинки виконана, для найкращого маршруту застосовується геометрична 2-opt евристика, ідея якої в тому, що з маршруту систематично видаляються групи з двох ребер $(v_k, v_t), (v_b, v_m)$, які замінюються іншими двома ребрами $(v_k, v_b), (v_t, v_m)$. Слід зазначити, що умова циклічності маршруту не повинна порушуватися. Якщо така заміна дозволяє зменшити довжину циклу [22, 23]

$$\Delta = c(v_k, v_t) + c(v_b, v_m) - (c(v_k, v_b) + c(v_t, v_m)) \geq 0, \qquad (2.17)$$

то ребра $(v_k, v_b), (v_t, v_m)$ зберігають, а даний алгоритм продовжує свою роботу.

2.4 Висновки

У розділі сформульовано основні принципи узагальненого алгоритму штучної бджолиної колонії. Представлено модифіковану математичну модель для розв'язання цілочисельних комбінаторних оптимізаційних задач з пошуку маршрутів. У наступному розділі необхідно виконати експериментальні дослідження ефективності роботи модифікації.

3 РОЗРОБКА ІНТЕЛЕКТУАЛЬНОГО ПРОГРАМНОГО МОДУЛЯ ТА ЕКСПЕРИМЕНТАЛЬНІ ДОСЛІДЖЕННЯ

3.1 Обгрунтування вибору мови програмування

Однією з найбільш популярних мов програмування для розв'язання інженерних задач є Python (рисунок 3.1). У зв'язку зі стрімким розвитком обчислювальної техніки відбувається поступова зміна вимог, які висуваються до мов програмування. Все більшу роль починають відігравати інтерпретовані мови (простий і гнучкий інструмент), оскільки потужність персональних комп'ютерів починає забезпечувати достатню швидкість виконання інтерпретованих програм. А єдиною суттєвою перевагою компільованих мов програмування є створений ними високошвидкісний код [33].

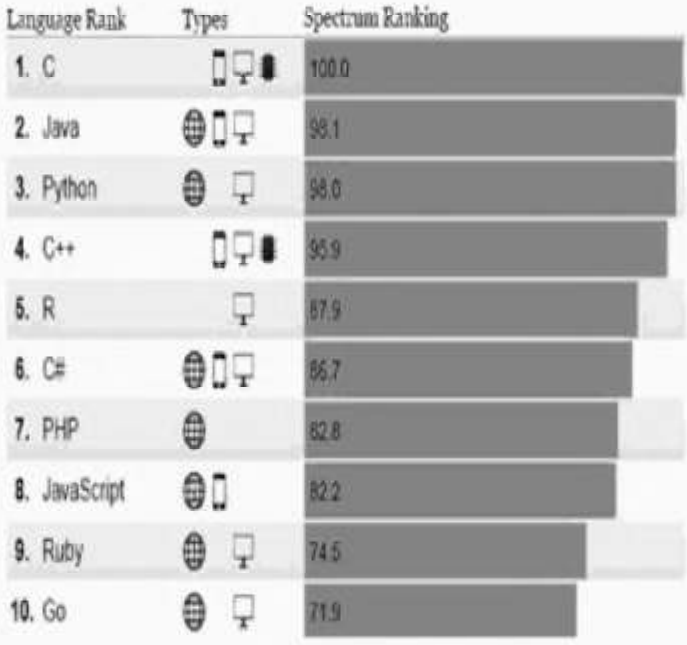


Рисунок 3.1 – Використання розробниками мов програмування

Однією із таких простих, функціональних і гнучких мов є Python. Його очевидною перевагою є те, що інтерпретатор Python реалізований практично на всіх платформах і операційних системах. Важлива риса – розширюваність мови, тобто можливість її вдосконалення зацікавленими програмістами.

Наявність великої кількості різноманітних програмних модулів забезпечує додаткові функції. Можна навести такі модулі: Numerical Python (розширені математичні можливості), Tkinter (розробка додатків з використанням графічного інтерфейсу користувача), OpenGL (використання бібліотеки графічного моделювання об'єктів). Недоліком даної мови є порівняно невисока швидкість виконання програм, але її можна нівелювати перевагами.

Дана мова програмування поєднує потужність і гнучкість з високою ефективністю виконавчого коду й можливістю доступу до апаратних ресурсів комп'ютера. Тому програмне забезпечення для розв'язання поставленої задачі буде розроблятися саме на цій мові.

Основні технічні та мінімальні системні вимоги до розробленої програми: операційна система *Windows* 7 або новіша; наявність клавіатури та миші; процесор *Core* i5 (*Ryzen* 5) або кращий; оперативна пам'ять – 8 Гб; відеопам'ять – 1 Гб; жорсткий диск – 500 Гб і більше.

Таким чином, оскільки дуже важливо для розробки нашої програми наявність як потужної математичної компоненти, так і бібліотек для відображення графа і мережі, то мовою програмування додатку вибрана Python.

3.2 Опис програмного модуля

Для роботи програми потрібно використовувати файли з розширенням *.*tsp*, внутрішній вміст яких можна побачити, наприклад в текстовому редакторі NotePad++. Слід пам'ятати, що не всі файли можуть бути прочитані. Це зумовлено тим, що вони не містять конкретних координат міст у читабельному для програми форматі. Для графів передбачені відображення вершин V і ребер E з вказівкою їх довжин. За замовчуванням створюється повний неорієнтований граф. Довжини ребер графа обчислюються автоматично, виходячи з координат вершин на площині. Результат роботи програми наведено на рисунку 3.2.

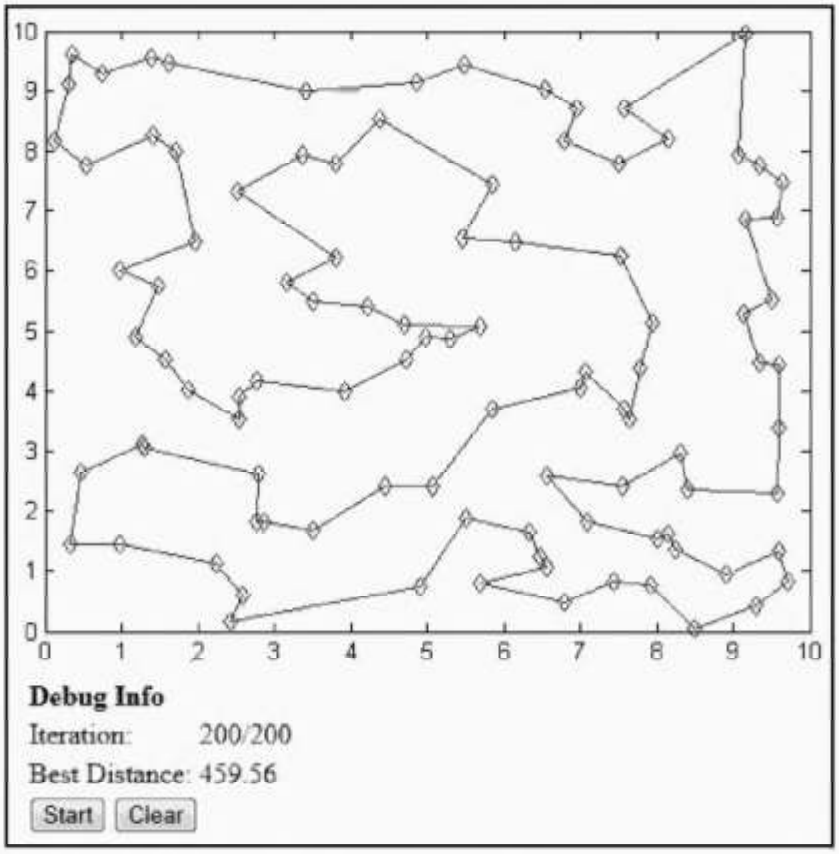


Рисунок 3.2 – Приклад роботи програми

У розробленому інтерфейсі користувача наявна низка пунктів [23, 24]:

Select file with data – дозволяє вибрати файл-основу поточної карти. Назва файлу створюється за шаблоном "generSource_кількість_пунктів". Для збереження створених карт їх слід перейменувати перед запуском наступного сеансу алгоритму, що використовує випадковий набір такої ж кількості міст на карті.

Generate cities randomly – при виборі цього параметру пункти будуть створені випадковим чином, а параметр Select file with data не буде враховано.

Дані карти будуть встановлені відповідно до параметрів Cities Quantity, Higher Bound X, Higher Bound Y. Якщо одне із цих полів не буде заповнене або формат даних не представлений цілочисельними невід'ємними числами, меншими за 10^6, то в роботі програми станеться збій.

Cities Quantity – визначає кількість міст на карті.

Higher Bound X – визначає максимально допустиме значення абсолютної координати міста по осі абсцис.

Higher Bound Y – визначає максимально допустиме значення абсолютної координати міста по осі ординат.

ComboBox – надає можливість вибрати яким чином нанести бджіл:

1. Ковдра – реалізація стандартного розміщення, у кожній вершині може перебувати по одній особині.

2. Дробовик – випадковий розподіл вершинами графа, причому необов'язково, щоб чисельність колонії і кількість вершин збігалися.

Screen Refresh Rate (разів за 10 секунд) – частота оновлення вікна алгоритму.

Write down intermediate data to console – цим параметром контролюється, чи слід виводити проміжні дані до консолі.

Write to file – записує результат до файлу (тека "output"), нумеруючи їх, починаючи з нуля. Кожен наступний запуск алгоритму з цим параметром генеруватиме новий файл.

Варто звернути увагу, що якщо користувач хоче повторно запустити карту, отриману автоматично (випадково), то потрібно забрати відмітку з чекбоксу Generate Cities Randomly та вибрати відповідний файл з теки "generatedSources". Назва такого файлу міститиме кількість міст, які були згенеровані у експерименті.

Виконання самого алгоритму може бути розбито на три основні блоки:

I. Надання початкового значення необхідним параметрам.

II. Обчислення маршруту ітеративно.

III. Перевірка критерію зупинки процесу.

Якщо блок 3 повертає значення true, то алгоритм припиняє роботу і виконує такі операції:

1. Виводить дані на екран у вигляді окремого вікна, створює два додаткових вікна із зображенням мережі та маршруту в абсолютних координатах.

2. Записує результат у файл, якщо було вибрано відповідні пункти меню.

За допомогою наочної демонстрації роботи методу можна забезпечити вищий рівень розуміння його роботи у ході розв'язання оптимізаційної задачі. Тому для забезпечення демонстрації поведінки алгоритму в різних умовах користувач програми має можливість регулювати параметри.

3.3 Результати експериментів

У ході першого експерименту визначимо оптимальну кількість бджіл-наглядачів O у колонії. Для цього її розмір буде змінюватися від 10 до 100 особин. Тестовою задачею буде задача tsp225 із бібліотеки TSPlib [17]. Відповідно до таблиці 3.1, коли $O = 50$ можна отримати кращі результати обчислень. Відносна похибка обчислена за формулою

$$\varepsilon = \frac{|L - L^{opt}|}{L^{opt}} \cdot 100\%, \tag{3.1}$$

де L – середня довжина знайденого маршруту за 25 запусків програми; $L^{opt} = 3916$ – найкращий відомий розв'язок задачі.

Таблиця 3.1 – Аналіз залежності якості розв'язку від кількості бджіл

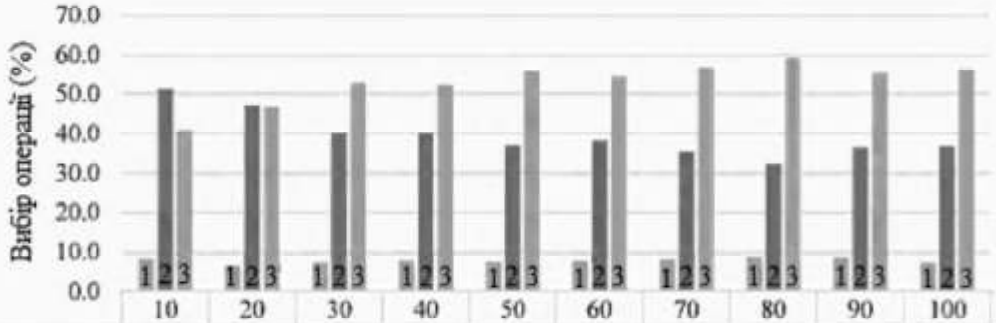
Кількість бджіл-наглядачів, O	Середня довжина маршруту, L	ε, %
10	4080,159	4,192
20	4051,219	3,453
30	4040,568	3,181
40	4023,181	2,737
50	4012,020	2,452
60	4034,733	3,032
70	4044,954	3,293
80	4052,120	3,476
90	4038,062	3,117
100	4040,216	3,172

Оскільки алгоритм ефективно використовує експлуатаційний і дослідницький процеси для забезпечення достатньої різноманітності в популяції колонії, то не потрібно задавати їй надвеликий розмір.

Значення *limit* не повинно бути занадто великим, щоб гарантувати оновлення популяції. Але при малій кількості джерел нектару значення *limit* все ж таки доцільно вибирати більшим, щоб не "загубилися" значення, які могли б бути покращені надалі.

Через те, що кількість параметрів у алгоритмі штучної бджолиної колонії досить мала, то до вибору значення максимальної кількості ітерацій K потрібно ставитися досить акуратно.

Крім того, слід зазначити, що на рисунку 3.3 показано ймовірність вибору операцій зміни маршруту для бджіл-наглядачів з використанням колеса рулетки. Можна побачити, що операція перестановки має низьку успішність та загалом нечасто дозволяє покращити маршрут, на другому місці розташувалась операція зсуву, а найбільш часто використовувалась операція подвійної інверсії.



	10	20	30	40	50	60	70	80	90	100
Перестановка	8.1	6.3	7.2	7.7	7.3	7.5	8.0	8.6	8.4	7.1
Зсув	51.1	47.0	40.1	40.0	36.9	38.2	35.3	32.2	36.4	36.7
Подвійна інверсія	40.8	46.7	52.6	52.2	55.8	54.3	56.7	59.2	55.2	56.2

Рисунок 3.3 – Ймовірність вибору операції залежно від її успішності

Графік збіжності процесу наведений на рисунку 3.4, з якого помітно, що операція перестановки покращує розв'язок, але дуже повільно; операції зсуву та подвійної інверсії поодинці доходять до плато насичення; однакова ймовірність вибору операції покращення маршруту бджолами-наглядачами (випадковий вибір) програє подібній процедурі, але з вибором складових операцій (з вибором), за їх успішністю у ході розв'язання задачі. Крім того, останні дві стратегії поступово продовжують покращувати розв'язок залежно від кількості ітерацій алгоритму. Найкраще себе проявила стратегія з вибором успішних операцій.

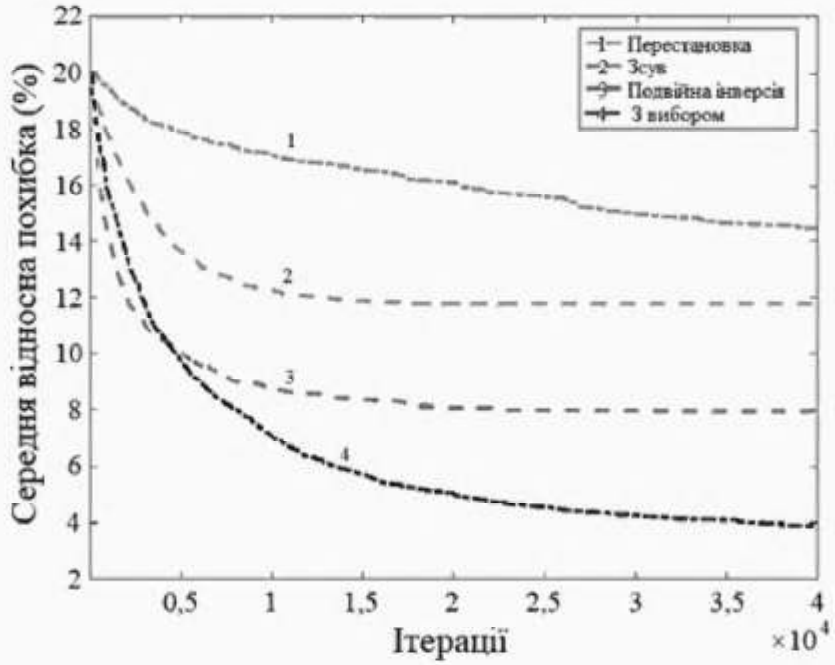


Рисунок 3.4 – Графік порівняння стратегій покращення розв'язку

Порівняння ефективності роботи стандартного алгоритму (Б) та модифікованого (М) наведено на 5-ти задачах з бібліотеки TSPlib [17]. Використано найкращі отримані параметри колонії: $E = 50$, $O = 50$, $S = 2$,

limit = 300, $K = 10000$, 25 запусків програми. Довжини маршрутів у таблиці 3.2 показано без стандартних одиниць вимірювання. Для даних задач наведено середню відносну похибку та середній час пошуку розв'язку (с), які розраховано за формулами

$$\varepsilon = \frac{1}{5} \cdot \sum_{j=1}^{5} \frac{| L_j - L^{\text{opt}}{}_j |}{L^{\text{opt}}{}_j} \cdot 100\%, \tag{3.2}$$

$$t = \sum_{j=1}^{5} t_j \Big/ 5, \tag{3.3}$$

де L_j – середня довжина маршруту для j-ої задачі; $L^{\text{opt}}{}_j$ – найкращий відомий розв'язок j-ої задачі; t_j – середній час розв'язання j-ої задачі.

Таблиця 3.2 – Результати експериментів

Назва задачі			*berlin*52	*kroa*100	*tsp*225	*pcb*442	*u*1060
Найкращий відомий результат			7542	21282	3916	50778	224094
Алгоритм	Б	Середня довжина	8390,26	25840,03	4455,12	56284,32	242304,33
		Середній час	18,17	35,17	76,68	234,86	936,76
	М	Середня довжина	7580,33	21735,31	4095,02	52643,62	230673,64
		Середній час	60,64	211,12	937,33	1640,35	3346,44

Приклад використання геометричної евристики для покращення маршруту наведено на рисунку 3.5.

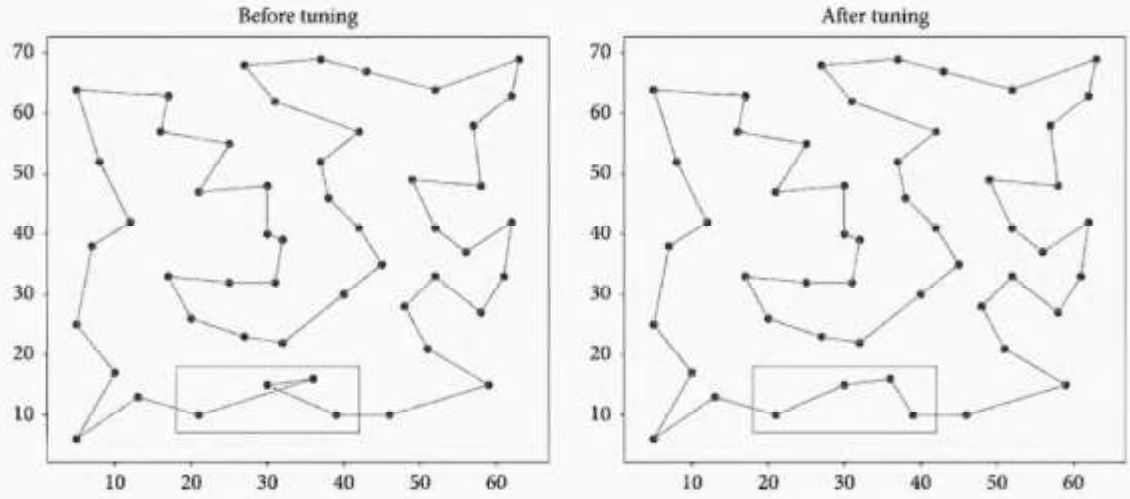


Рисунок 3.5 – Застосування 2-opt евристики

У таблиці 3.3 показано узагальнення результатів експериментів. Можна помітити, що базовий алгоритм без налаштувань менш точний, але швидше працює.

Таблиця 3.3 – Порівняння ефективності роботи алгоритмів на тестових задачах

#	Б	М	Результат порівняння
Середня відносна похибка, %	13,080	2,764	-10,316 (М)
Середній час пошуку, с	260,328	1239,176	+978,848 (Б)

На рисунках 3.6 та 3.7 наведено наглядне графічне відображення результатів роботи.

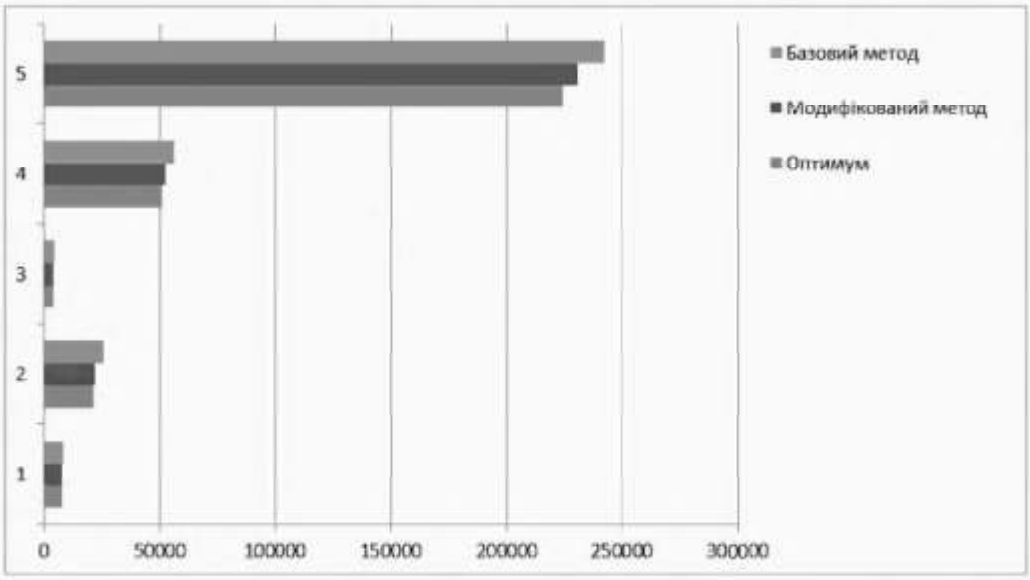


Рисунок 3.6 – Порівняння маршрутів

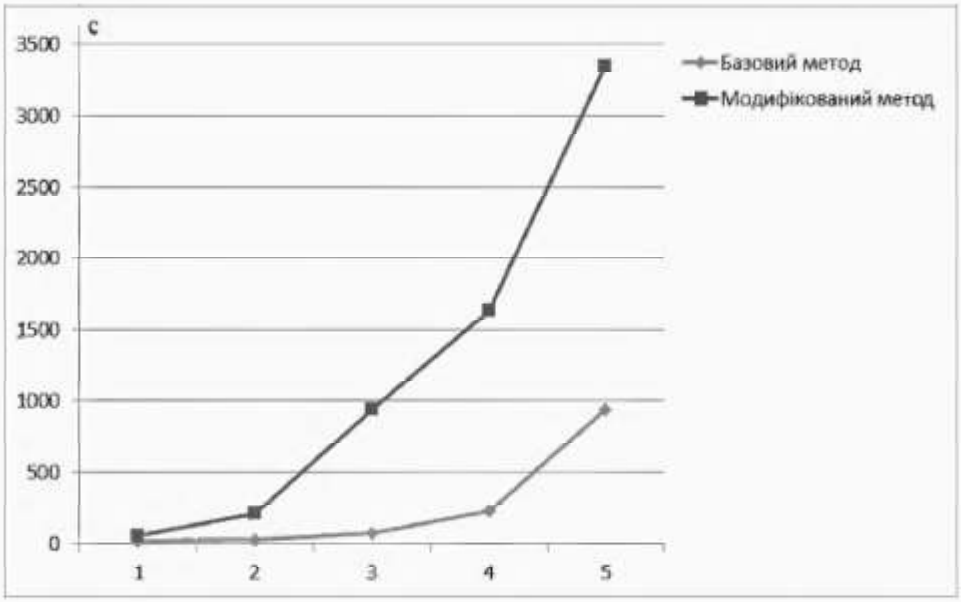


Рисунок 3.7 – Порівняння часу розв'язання

Дослідження роботи алгоритму на задачах великої розмірності проведено на згенерованій випадковим чином карті розмірності $n = 20000$ вершин за 10 тестових запусків (таблиця 3.4).

Таблиця 3.4 – Результати експерименту на масштабній задачі

Кількість вершин			20000
Алгоритм	Б	Середня довжина	773567,42
		Середній час	19638,15
	М	Середня довжина	715653,64
		Середній час	36213,87

У таблиці 3.5 для даної задачі показано порівняння ефективності роботи методів за довжиною маршруту та часом роботи.

Таблиця 3.5 – Результати розв'язання задачі великої розмірності

#	Результат порівняння	Переможець
Середня довжина маршруту	-57913,78	М
Середній час, с	+16575,72	Б

Визначено, що модифікований алгоритм працює на 4,604 год. довше, але результат на 57913,78 умовних одиниць менший. Можна побачити, що в даному досліді ситуація аналогічна до попередніх експериментів, тобто модифікація працює краще в плані точності, але програє по часу роботи. Якщо маршрут обчислюється заздалегідь, то даний недолік нівелюється.

3.4 Висновки

У даному розділі представлено розроблене програмне забезпечення, наведено опис його параметрів. Програма дозволяє керувати параметрами модифікованого алгоритму й аналізувати ефективність його роботи. Отримано, що алгоритм досить точний, а використання додаткових евристик дозволить покращити результат роботи, при цьому може збільшитись час обчислень.

4 ЕКОНОМІЧНИЙ РОЗДІЛ

4.1 Технологічний аудит розробленої модифікації

Для вирішення задачі у виконаній нами магістерській кваліфікаційній роботі було проведено експериментальне дослідження ефективності використання запропонованого модифікованого алгоритму поведінки штучної бджолиної колонії для розв'язання задачі логістики останньої милі.

Для проведення аналізу та встановлення комерційного потенціалу розробленого нами інтелектуального модуля було запрошено 3-ох відомих експертів – кандидата технічних наук, доцента Сторчака В.Г., кандидата технічних наук, доцентку Барабан М.В. і кандидата технічних наук, доцента Кулика Я.А.

Визначення комерційного потенціалу розробленого нами інтелектуального модуля для розв'язання задачі логістики останньої милі було здійснено за критеріями, наведеними в таблиці 4.1.

Таблиця 4.1 – Рекомендовані критерії оцінювання рівня комерційного потенціалу будь-якої розробки і їх бальна оцінка

Критерії оцінювання та бали (за 5-ти бальною шкалою)					
	0	1	2	3	4
Технічна здійсненність концепції:					
1	Достовірність концепції не підтверджена	Концепція підтверджена експертними висновками	Концепція підтверджена розрахунками	Концепція перевірена на практиці	Перевірено роботоздатність продукту в реальних умовах
Ринкові переваги (недоліки):					
2	Багато аналогів на малому ринку	Мало аналогів на малому ринку	Кілька аналогів на великому ринку	Один аналог на великому ринку	Продукт не має аналогів на великому ринку

Продовження таблиці 4.1

	0	1	2	3	4
3	Ціна продукту значно вища за ціни аналогів	Ціна продукту дещо вища за ціни аналогів	Ціна продукту приблизно дорівнює цінам аналогів	Ціна продукту дещо нижче за ціни аналогів	Ціна продукту значно нижче за ціни аналогів
4	Технічні та споживчі властивості продукту значно гірші, ніж в аналогів	Технічні та споживчі властивості продукту трохи гірші, ніж в аналогів	Технічні та споживчі властивості продукту на рівні аналогів	Технічні та споживчі властивості продукту трохи кращі, ніж в аналогів	Технічні та споживчі властивості продукту значно кращі, ніж в аналогів
Ринкові перспективи					
5	Експлуатаційні витрати значно вищі, ніж в аналогів	Експлуатаційні витрати дещо вищі, ніж в аналогів	Експлуатаційні витрати на рівні експлуатаційних витрат аналогів	Експлуатаційні витрати трохи нижчі, ніж в аналогів	Експлуатаційні витрати значно нижчі, ніж в аналогів
6	Ринок малий і не має позитивної динаміки	Ринок малий, але має позитивну динаміку	Середній ринок з позитивною динамікою	Великий стабільний ринок	Великий ринок з позитивною динамікою
7	Активна конкуренція великих компаній на ринку	Активна конкуренція	Помірна конкуренція	Незначна конкуренція	Конкурентів немає
Практична здійсненність					
8	Відсутні фахівці як з технічної, так і з комерційної реалізації ідеї	Необхідно наймати фахівців або витрачати значні кошти та час на навчання наявних фахівців	Необхідне незначне навчання фахівців та збільшення їх штату	Необхідне незначне навчання фахівців	Є фахівці з питань як з технічної, так і з комерційної реалізації ідеї
9	Потрібні значні фінансові ресурси, які відсутні. Джерела фінансування ідеї відсутні	Потрібні незначні фінансові ресурси. Джерела фінансування відсутні	Потрібні значні фінансові ресурси. Джерела фінансування є	Потрібні незначні фінансові ресурси. Джерела фінансування є	Не потребує додаткового фінансування

Продовження таблиці 4.1					
	0	1	2	3	4
10	Необхідна розробка нових матеріалів	Потрібні матеріали, що використовуються у військово-промислово-му комплексі	Потрібні дорогі матеріали	Потрібні досяжні та дешеві матеріали	Всі матеріали для реалізації ідеї відомі та давно використовуються у виробництві
11	Термін реалізації ідеї більший за 10 років	Термін реалізації ідеї більший за 5 років. Термін окупності інвестицій більше 10-ти років	Термін реалізації ідеї від 3-х до 5-ти років. Термін окупності інвестицій більше 5-ти років	Термін реалізації ідеї менше 3-х років. Термін окупності інвестицій від 3-х до 5-ти років	Термін реалізації ідеї менше 3-х років. Термін окупності інвестицій менше 3-х років
12	Необхідна розробка регламентних документів та отримання великої кількості дозвільних документів на виробництво та реалізацію продукту	Необхідно отримання великої кількості дозвільних документів на виробництво та реалізацію продукту, що вимагає значних коштів та часу	Процедура отримання дозвільних документів для виробництва та реалізації продукту вимагає незначних коштів та часу	Необхідно тільки повідомлення відповідним органам про виробництво та реалізацію продукту	Відсутні будь-які регламентні обмеження на виробництво та реалізацію продукту

Запрошені нами висококваліфіковані експерти оцінили розроблений інтелектуальний модуль для розв'язання задачі логістики останньої милі утаблиці 4.2. Для встановлення комерційного потенціалу скористаємося рекомендаціями, які наведено в таблиці 4.3 [34, 35]

Оскільки середньоарифметична сума балів, що їх виставили експерти, складає 43 бали, то це означає, що розроблений інтелектуальний модуль для розв'язання задачі логістики останньої милі має рівень комерційного потенціалу, який вважається "високим".

Таблиця 4.2 – Результати технологічного аудиту розробленого інтелектуального модуля (за шкалою оцінювання 0-1-2-3-4)

Критерії	Прізвище, ініціали експертів		
	Сторчак В. Г.	Барабан М. В.	Кулик Я. А.
	Бали, що їх виставили експерти:		
1	4	3	3
2	4	3	4
3	3	4	3
4	4	4	4
5	3	3	4
6	4	4	3
7	3	4	4
8	4	3	3
9	4	4	3
10	3	3	4
11	4	4	3
12	4	4	4
Сума балів	$СБ_1 = 44$	$СБ_2 = 43$	$СБ_3 = 42$
Середньоарифметична сума балів $\overline{СБ}$	$\overline{СБ} = \frac{\sum_{1}^{3} СБ_i}{3} = \frac{44+43+42}{3} = \frac{129}{3} = 43,0$		

Таблиця 4.3 – Рівні комерційного потенціалу будь-якої наукової розробки

Середньоарифметична сума балів $\overline{СБ}$, розрахована на основі висновків експертів	Рівень комерційного потенціалу розробки
0 – 10	Низький
11 – 20	Нижче середнього
21 – 30	Середній
31 – 40	Вище середнього
41 – 48	Високий

Це пояснюється тим, що розроблений інтелектуальний модуль використовує модифікацію, яка включає переваги сучасних наукових рішень (простота, гнучкість, ефективність).

4.2 Розрахунок витрат на розробку та проведення досліджень

При розробленні інтелектуального модуля нами було витрачено:

1) Основна заробітна плата $З_о$ визначається за формулою (таблиця 4.4):

$$З_о = \frac{М}{Т_р} \cdot t \text{ (грн.)}, \qquad (4.1)$$

де М – місячний посадовий оклад розробника, грн., (6700…25000) грн/місяць; $Т_р$ – кількість робочих днів в місяці, $Т_р$ = 21 день; t – кількість днів роботи.

Б) Додаткова заробітна плата $З_д$ розробників (виконавців) розраховується як (10…12)% від величини їх основної заробітної плати, тобто:

$$З_д = \alpha \cdot З_о = (0{,}1...0{,}12) \cdot З_о . \qquad (4.2)$$

Приймемо, що α = 0,114. Тоді для нашого випадку отримаємо:

$$З_д = 0{,}114 \times 29481 = 3360{,}83 \approx 3361 \text{ грн.}$$

В) Нарахування на заробітну плату $НЗП_{зп}$ розробників (дослідників) розраховуються за формулою:

$$НЗП_{зп} = (З_о + З_д) \cdot \frac{\beta}{100}, \qquad (4.3)$$

де β – ставка обов'язкового єдиного внеску на державне соціальне страхування у %, нехай β = 22%. Тоді:

$$НЗН_{зп} = (29481 + 3361) \times 0,22 = 7225,24 \approx 7226 \text{ грн.}$$

Таблиця 4.4 – Основна заробітна плата розробників (виконавців)

Найменування посади виконавця	Місячний посадовий оклад, грн.	Оплата за робочий день, грн.	Кількість днів роботи	Витрати на оплату праці, грн.
1. Науковий керівник магістерської роботи	19500	928,57	20 годин	≈ 3095,23
2. Магістрант-студент-виконавець	2000 (беремо 6700)	319,04	82	≈ 26161,28
3. Консультант з економічної частини	18800	895,24	1,5 години	≈ 223,81 (при 6-годинному робочому дні)
Загалом				$З_о$ = 29480,32 грн. ≈ 29481 грн.

Г) Амортизація основних засобів А, які використовувались під час виконання цієї роботи (таблиця 4.5):

$$А = \frac{Ц \cdot Н_а}{100} \cdot \frac{Т}{12} \text{ (грн.)}, \qquad (4.4)$$

де Ц – загальна балансова вартість основних засобів, грн.; $Н_а$ – річна норма амортизаційних відрахувань, $Н_а$ = (2,5...25)%; Т – термін використання основних засобів, місяці.

Таблиця 4.5 – Розрахунок амортизаційних відрахувань

Найменування обладнання, приміщень тощо	Балансова вартість, грн.	Норма амортизації, %	Термін використання, міс.	Величина амортизаційних відрахувань, грн.
1. Комп'ютерна техніка, обладнання тощо	82000	25	3,2 (при 75% використанні	3971,87 ≈ 3972
2. Приміщення університету, кафедри	40000	2,5	3,2 (при 45% використанні)	119,99 ≈ 120
Всього				А = 4092 грн.

Д) Витрати на матеріали М розраховуються за формулою:

$$М = \sum_{1}^{n} Н_i \cdot Ц_i \cdot К_i - \sum_{1}^{n} В_i \cdot Ц_в \text{ (грн.)}, \quad (4.5)$$

де $Н_i$ – витрати матеріалу *i*-го найменування, кг; $Ц_i$ – вартість матеріалу *i*-го найменування; $К_i$ – коефіцієнт транспортних витрат, $К_i$ = (1,1…1,15); $В_i$ – маса відходів матеріалу *i*-го найменування; $Ц_в$ – ціна відходів матеріалу *i*-го найменування; n – кількість видів матеріалів.

Е) Витрати на комплектуючі К розраховуються за формулою:

$$К = \sum_{1}^{n} Н_i \cdot Ц_i \cdot К_i \text{ (грн.)}, \quad (4.6)$$

де $Н_i$ – кількість комплектуючих *i*-го виду, шт.; $Ц_i$ – ціна комплектуючих *i*-го виду; $К_i$ – коефіцієнт транспортних витрат, $К_i$ = (1,1…1,15); n – кількість видів комплектуючих.

Під час виконання роботи загальні витрати на матеріали та комплектуючі склали приблизно 2500 грн.

Ж) Витрати на силову електроенергію $В_е$ розраховуються за формулою:

$$В_е = \frac{В \cdot П \cdot Ф \cdot К_п}{К_д}, \quad (4.7)$$

де В – вартість 1 кВт-год. електроенергії, в 2023 р. В ≈ 4,5 грн/кВт; П – установлена потужність обладнання, кВт, нехай П = 1,22 кВт; Ф – фактична кількість годин роботи обладнання, 290 годин; $К_п$ – коефіцієнт використання потужності, $К_п$ < 1 = 0,85; $К_д$ – коефіцієнт корисної дії, $К_д = 0,75$.

Тоді витрати на силову електроенергію будуть дорівнювати:

$$В_е = \frac{В \cdot П \cdot Ф \cdot К_п}{К_д} = \frac{4,5 \cdot 1,22 \cdot 290 \cdot 0,85}{0,75} = 1804,30 \approx 1805 \text{ грн.}$$

З) Інші витрати $В_{інш}$ можна прийняти як (50…300)% від основної заробітної плати розробників, тобто:

$$В_{інш} = (0,5....3) \times З_о. \quad (4.8)$$

Для нашого випадку отримаємо:

$$В_{інш} = 1,5 \times 29481 = 44221,5 \approx 44222 \text{ грн.}$$

К) Сума всіх попередніх статей витрат складає витрати на виконання роботи безпосередньо розробником-магістрантом:

$$В = 29481 + 3361 + 7226 + 4092 + 2500 + 1805 + 44222 = 92687 \text{ грн.}$$

Л) Загальні витрати на розроблення інтелектуального модуля для розв'язання задачі логістики останньої милі $В_{заг}$ розраховуються за формулою:

$$В_{заг} = \frac{В}{\beta}, \quad (4.9)$$

де β – коефіцієнт, який характеризує етап (стадію) виконання цієї роботи. Можна прийняти, що, $\beta \approx 0{,}85$ [34, 35], оскільки робота практично завершена:

$$В_{заг} = \frac{92687}{0{,}85} = 109043{,}53 \text{ грн. або приблизно 110 тисяч грн.}$$

Тобто прогнозовані загальні витрати на розроблення інтелектуального модуля для розв'язання задачі логістики останньої милі становлять приблизно 110 тисяч грн.

4.3 Прогнозування комерційного ефекту від можливої комерціалізації розробки

Економічний ефект від впровадження та можливої комерціалізації розробленого інтелектуального модуля пояснюється його значно кращими функціональними можливостями та ключовими перевагами, які можуть отримати підприємства, що будуть використовувати цей програмний продукт, а саме: його простоту, гнучкість, ефективність. Тому нашу розробку можна реалізовувати на ринку дещо дорожче, ніж аналогічні (але гірші) за функціями

подібні розробки. Так, якщо подібні, але гірші за функціями програмні продукти у 2022-2023 роках коштували на ринку приблизно 150 тисяч грн., то нашу розробку можна буде реалізовувати на ринку приблизно за 175 тисяч грн. або на 25 тисяч грн. дорожче.

Аналіз місткості ринку показує, що на сьогодні в Україні попит на подібний програмний продукт, особливо в умовах стрімкої діджиталізації, може бути великим. Тому можна очікувати стрімке зростання попиту на нашу розробку принаймні впродовж 3-ох років після її впровадження (до появи нових, більш ефективних розробок). Тобто, якщо наша розробка буде впроваджена з 1 січня 2024 року, то її результати будуть виявлятися впродовж 2024-го, 2025-го та 2026-го років. Прогноз зростання попиту на нашу розробку складає по роках: 2023 рік – базовий попит 17 шт.; 2024 р. – приблизно +10 шт. до базового року; 2025 р. – +20 шт. до базового року; 2026 р. – +30 шт. до базового року.

Можливе збільшення чистого прибутку $\Delta П_i$, що його може отримати потенційний інвестор від комерціалізації нашої розробки на ринок:

$$\Delta П_i = \sum_{1}^{n} (\Delta Ц_o \cdot N + Ц_o \cdot \Delta N)_i \cdot \lambda \cdot \rho \cdot (1 - \frac{\upsilon}{100}), \qquad (4.10)$$

де $\Delta Ц_o$ – покращення основного якісного показника від впровадження результатів нашої розробки у цьому році. Для нашого випадку це є збільшення ціни реалізації нашої розробки $\Delta Ц_o$ = (175 – 150) = + 25 тисяч грн; N – основний кількісний показник, який визначає обсяг діяльності у році до впровадження результатів розробки, N = 17 шт.; ΔN – покращення основного кількісного показника від впровадження результатів розробки; $Ц_o$ – основний якісний показник (тобто ціна), який визначає обсяг діяльності у році після впровадження результатів розробки, $Ц_o$= 175 тисяч грн.; n – кількість років, за які очікується отримання позитивних результатів від впровадження розробки;

для нашого випадку n = 3; λ – коефіцієнт, який враховує сплату податку на додану вартість; $\lambda = 0{,}8333$; ρ – коефіцієнт, який враховує рентабельність продукту, $\rho = (0{,}2...0{,}5)$, $\rho = 0{,}5$; υ – ставка податку на прибуток, $\upsilon = 18\%$.

Тоді можливе зростання чистого прибутку $\Delta П_i$ для потенційного інвестора складе:

$$\Delta П_1 = (25 \cdot 17 + 175 \cdot 10) \cdot 0{,}8333 \cdot 0{,}5 \cdot (1 - \frac{18}{100}) \approx 743 \text{ тис. грн.}$$

$$\Delta П_2 = (25 \cdot 17 + 175 \cdot 20) \cdot 0{,}8333 \cdot 0{,}5 \cdot (1 - \frac{18}{100}) \approx 1341 \text{ тис. грн.}$$

$$\Delta П_3 = (25 \cdot 17 + 175 \cdot 30) \cdot 0{,}8333 \cdot 0{,}5 \cdot (1 - \frac{18}{100}) \approx 1939 \text{ тис. грн.}$$

Приведена вартість зростання всіх чистих прибутків від можливого впровадження нашої розробки становитиме:

$$ПП = \sum_{1}^{\tau} \frac{\Delta П_i}{(1+\tau)^t}, \qquad (4.11)$$

де $\Delta П_i$ – збільшення чистого прибутку у кожному із років, впродовж яких виявляються результати виконаної та впровадженої роботи, грн.; τ – період часу, за який виявляються результати впровадженої роботи, τ = 3 роки; τ – ставка дисконтування, $\tau = 0{,}10$ (10%); t – період часу від моменту початку розробки до моменту отримання можливих чистих прибутків інвестором.

Тоді приведена вартість зростання всіх можливих чистих прибутків ПП, що їх може отримати потенційний інвестор від комерціалізації нашої розробки, складе:

$$ПП = \frac{743}{(1+0,1)^2} + \frac{1341}{(1+0,1)^3} + \frac{1939}{(1+0,1)^4} \approx 614 + 1008 + 1325 = 2947 \text{ тисяч грн.}$$

Теперішня вартість інвестицій PV, що повинні бути вкладені для реалізації нашої розробки:

$$PV = (1,0 \ldots 5,0) \times B_{заг.} \quad (4.12)$$

$$PV = (4,0 \ldots 5,0) \times 110 = 5 \times 110 = 550 \text{ тисяч грн.}$$

Абсолютний ефект від можливих вкладених інвестицій $E_{абс}$:

$$E_{абс} = ПП - PV, \quad (4.13)$$

де ПП – приведена вартість збільшення всіх чистих прибутків для інвестора від можливого впровадження нашої розробки, грн.; PV – теперішня вартість інвестицій PV = 550 тисяч грн.

Абсолютний ефект від можливого впровадження нашої розробки (при прогнозованому ринку збуту) за три роки складе:

$$E_{абс} = 2947 - 550 = 2397 \text{ тисяч грн.}$$

Оскільки $E_{абс} > 0$, то комерціалізація нашої розробки доцільна.

Далі розрахуємо внутрішню дохідність $E_в$ вкладених інвестицій:

$$E_в = \sqrt[T_ж]{1 + \frac{E_{абс}}{PV}} - 1, \quad (4.14)$$

де $T_ж$ – життєвий цикл розробки, 4 роки (2023…2026 рр.).

Для нашого випадку отримаємо:

$$E_в = \sqrt[4]{1+\frac{2397}{550}} - 1 = \sqrt[4]{1+4,3582} - 1 = \sqrt[4]{5,3582} - 1 = 1,52 - 1 = 0,52 = 52\%.$$

Далі визначимо ту мінімальну дохідність, нижче за яку потенційному інвестору не вигідно буде займатися комерціалізацією нашої розробки. Мінімальна дохідність або мінімальна (бар'єрна) ставка дисконтування $\tau_{мін}$ визначається за формулою:

$$\tau_{мін} = d + f, \qquad (4.15)$$

де d – середньозважена ставка за депозитними операціями в комерційних банках, в 2023 році в Україні d = (0,10...0,12); f – показник, що характеризує ризикованість вкладень, f = (0,05...0,30).

Для нашого випадку отримаємо:

$$\tau_{мін} = 0,12 + 0,30 = 0,42 \text{ або } \tau_{мін} = 42\%.$$

Оскільки величина $E_в = 52\% > \tau_{мін} = 42\%$, то потенційний інвестор у принципі може бути зацікавлений у комерціалізації розробленого нами інтелектуального модуля для розв'язання задачі логістики останньої милі.

Далі розраховуємо термін окупності коштів, вкладених у можливу комерціалізацію розробленого нами програмного продукту:

$$T_{ок} = \frac{1}{E_в}. \qquad (4.16)$$

$$T_{ок} = \frac{1}{0,52} = 1,92 \text{ років} < 3 \text{ років},$$

що свідчить про потенційну доцільність комерціалізації розробленого нами інтелектуального модуля.

Далі проведено моделювання залежності величини внутрішньої дохідності вкладених потенційним інвестором коштів в комерціалізацію розробленого модуля від рівня інфляції в країні. Так, якщо рівень інфляції в країні зросте до 20%, то:

$$ПП = \frac{743}{(1+0,2)^2} + \frac{1341}{(1+0,2)^3} + \frac{1939}{(1+0,2)^4} \approx 516 + 776 + 935 = 2227 \text{ тисяч грн.}$$

Тоді абсолютний ефект від можливого впровадження розробки складе:

$$E_{абс} = 2227 - 550 = 1677 \text{ тисяч грн.}$$

Внутрішня дохідність $E_в$ вкладених інвестицій становитиме:

$$E_в = \sqrt[4]{1 + \frac{1677}{550}} - 1 = \sqrt[4]{1 + 3,0491} - 1 = \sqrt[4]{4,0491} - 1 = 1,418 - 1 = 0,418 = 41,8\%.$$

Оскільки величина $E_в = 41,8\% \approx \tau_{мін} = 42\%$, то потенційний інвестор у принципі також може бути зацікавлений у комерціалізації нашої розробки.

Прийнявши рівень інфляції у 30%, отримаємо:

$$ПП = \frac{743}{(1+0,3)^2} + \frac{1341}{(1+0,3)^3} + \frac{1939}{(1+0,3)^4} \approx 440 + 610 + 679 = 1729 \text{ тисяч грн.}$$

Тоді абсолютний ефект від можливого впровадження розробки складе:

$$E_{абс} = 1729 - 550 = 1179 \text{ тисяч грн.}$$

Внутрішня дохідність $E_в$ вкладених інвестицій становитиме:

$$E_в = \sqrt[4]{1 + \frac{1179}{550}} - 1 = \sqrt[4]{1 + 2{,}1436} - 1 = \sqrt[4]{3{,}1436} - 1 = 1{,}33 - 1 = 0{,}33 = 33\%.$$

Оскільки величина $E_в = 33\% < \tau_{мін} = 42\%$, то потенційний інвестор може бути незацікавлений у комерціалізації нашої розробки, але остаточне рішення щодо цього питання буде прийматися при врахуванні інших обставин (наприклад, шляхом зниження рівня прийнятого ризику з f = 30% до меншої величини, або шляхом підняття ціни реалізації нашої розробки тощо). Зроблені розрахунки у вигляді графіків наведено на рисунку 4.1.

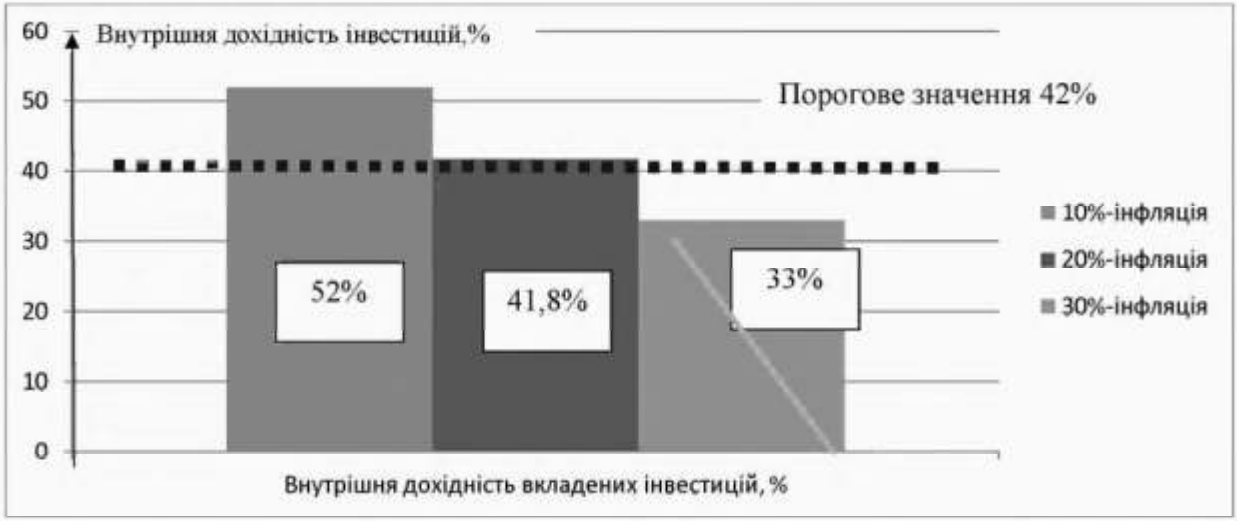


Рисунок 4.1 – Моделювання залежності величини внутрішньої дохідності потенційних інвестицій від рівня інфляції в країні (10%, 20%, 30%)

4.4 Висновки

Аналіз комерційного потенціалу розробки показав, що програмний продукт за своїми характеристиками випереджає можливі аналогічні реалізації і є перспективною розробкою. Він має кращі функціональні показники, а тому є конкурентоспроможним товаром на ринку. Результати виконаної економічної частини зведено у таблицю 4.6.

Таблиця 4.6 – Результати економічної частини

Показники	Задані у технічному завданні	Досягнуті у магістерській кваліфікаційній роботі	Висновок
1. Витрати на розробку	не більше 120 тис. грн.	110 тис. грн.	Виконано
2. Абсолютний ефект від впровадження розробки, тис. грн.	не менше 2300 тис. грн.	2397 тис. грн. (при 10%-інфляції)	Виконано
3. Внутрішня норма дохідності інвестицій, %	не менше 42%	52% (при 10%-інфляції); 41,8% (при 20%-інфляції);	Виконано
4. Термін окупності інвестицій, роки	до 3-ох років	1,92 роки	Виконано

Таким чином, основні техніко-економічні показники модифікованого алгоритму виконані.

ВИСНОВКИ

У даній роботі розв'язується задача логістики останньої милі. Визначено, що одним з найбільш ефективних та зручних для її розв'язання є алгоритм штучної бджолиної колонії. Тому розглянуто його математичну модель та представлено модифікації для розв'язання заданої задачі. Описано особливості розробленого програмного модуля. Тестування ефективності модифікованого алгоритму проведено на низці задач.

Визначено, що запропонована модифікація досить точна, але повільніша за базовий алгоритм. Крім того, враховуючи стохастичність алгоритму, необхідно вдало підбирати параметри під кожну конкретну задачу. Таким чином, застосування алгоритму штучної бджолиної колонії дозволяє отримати хороше квазіоптимальне рішення, не потребуючи громіздких обчислень.

Аналіз комерційного потенціалу програмного забезпечення показав, що його розробка є перспективною.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Automatic Multi-sensor Task Allocation Using Modified Distributed Bees Algorithm / I. Tkach, Y. Edan, A. Jevtic, S. Y. Nof. *IEEE International Conference on Systems, Man, and Cybernetics*, Manchester (United Kingdom), 2013. P. 1401–1406.

2. Кавун С.В., Ревак І.О. Застосування теорії графів у задачах комунікаційного менеджменту. *Науковий вісник Львівського державного університету внутрішніх справ*. 2015. № 2. P. 225–240.

3. Waligora L. Application of Hamilton's Graph Theory in New Technologies. *World Scientific News*. Vol. 89. 2017. P. 71–81.

4. Mollabakhshi N., Eshghi M. Combinational Circuit Design Using Bees Algorithm. *IEEE Conference Anthology*. Piscataway, 2014. P. 1–4.

5. Sahin M. Improvement of the Bees Algorithm for Solving the Traveling Salesman Problems. *Bilişim teknolojileri dergisi*. 2022. Vol. 15. P. 65–74.

6. Karaboga D., Gorkemli B. Solving Traveling Salesman Problem by Using Combinatorial Artificial Bee Colony Algorithms. *International Journal on Artificial Intelligence Tools*. 2019. Vol. 28. P. 1–28.

7. Xie J., Carrillo L.R.G., Jin L. An Integrated Traveling Salesman and Coverage Path Planning Problem for Unmanned Aircraft Systems. *IEEE Control Systems Letters*. 2019. № 3. P. 67–72.

8. Метод ройового інтелекту (штучної бджолиної колонії) тематичного сегментування оптико-електронного зображення / І.А. Хижняк,

О.М. Маковейчук, Р.Г. Худов, В.О. Подліпаєв та інші. *Системи управління, навігації та зв'язку*. № 2. 2018. С. 91-96.

9. Назаренко В.О., Іванов Ю.Ю., Кривогубченко С.Г. та інші. Модифікований алгоритм оптимізації функції втрат нейромережі. *Актуальні задачі медичної, біологічної фізики та інформатики*: матер. науково-практичної конференції з міжнародною участю. Вінниця: ВНМУ ім. М.І. Пирогова, 2022. С. 73-74.

10. Назаренко В.О., Іванов Ю.Ю., Кривогубченко С.Г. та інші. Деякі аспекти метаевристичних алгоритмів оптимізації. *Молодь в науці: дослідження, проблеми, перспективи*: матер. всеукраїнської науково-практичної конференції. Вінниця: ВНТУ, 2023. URL: https://conferences.vntu.edu.ua/index.php/mn/mn2023/paper/view/17145 (дата звернення 15.03.2023).

11. Субботін С.О., Олійник А.О., Олійник О.О. Неітеративні, еволюційні та мультиагентні методи синтезу нечіткологічних і нейромережних моделей. Запоріжжя: ЗНТУ, 2009. 375 с.

12. Novel Genetic Bees Algorithm Applied to Single Machine Scheduling Problem / M.S. Packianather, B. Yuce, E. Mastrocinque, F. Fruggiero et al. *World Automation Congress*. Big Island of Hawaii, 2014. 906–911.

13. Solution of a Large-Scale Traveling-Salesman Problem. In 50 Years of Integer Programming 1958-2008 From the Early Years to the State-of-the-Art / V. Chantal, W.J. Cook and others. Springer-Verlag Berlin, 2010. P. 7–28.

14. Тимофієва Н.К. Про деякі властивості множини розв'язків задачі комівояжера. *Control Systems and Computers. Фундаментальные и прикладные проблемы Computer Science*. К., 2018. № 5. С. 3–12.

15. Simon D. Evolutionary Optimization Algorithms: Biologically Inspired and Population-Based Approaches to Computer Intelligence. John Wiley & Sons, 2013. 776 p.

16. Погорілий С.Д., Потебня А.В. Розробка новітніх систем штучного інтелекту для розв'язання задачі комівояжера. *Вісник Київського національного університету імені Тараса Шевченка*. Київ, 2012. №4. С. 173–184.

17. Traveling Salesman Problem. University of Waterloo. URL: http://www.math.uwaterloo.ca/tsp/index.html (02.05.2022).

18. Laporte G. The Traveling Salesman Problem: An Overview of Exact and Approximate Algorithms. *European Journal of Operational Research*. Canada, Montreal, 1992. № 59. P. 231–247.

19. Гуляницький Л.Ф., Мулеса О.Ю. Прикладні методи комбінаторної оптимізації. К.: Видавничо-поліграфічний центр "Київський університет", 2016. 142 с.

20. Кравець П., Пасічник В. Проданюк М. Ігрова самоорганізація гамільтонового циклу графа. *Information Systems and Networks*. № 10. 2021. P. 13–32.

21. Muliarevych O. Cyber-Physical System for Solving Travelling Salesman Problem. *Advances in Cyber-Physical Systems*. 2017. Vol. 2. № 1. P. 22–28.

22. Helsgaun K. An Effective Implementation of k-opt Moves for the Lin-Kernighan TSP Heuristic. URL: http://akira.ruc.dk/~keld/research/LKH/KoptReport.pdf (23.09.2023).

23. Travelling Salesman Problem using 3OPT and 2OPT Perturbation. URL: https://github.com/source-nerd/tsp_3opt_2opt (дата звернення 27.11.2023).

24. Koc E. Bees Algorithm: Theory, Improvements and Applications. Cardiff University, 2010. 209 p.

25. Akay B., Karaboga D. Artificial Bee Colony Algorithm for Large-Scale Problems and Engineering Design Optimization. *Journal of intelligent manufacturing*. 2012. Vol. 23 (4). P. 1001–1014.

26. Karaboga D., Basturk B. A Powerful and Efficient Algorithm for Numerical Function Optimization: Artificial Bee Colony (ABC) Algorithm. *Journal of Global Optimization*. 2007. Vol. 39. P. 459–471.

27. Karaboga D., Basturk B. On the Performance of Artificial Bee Colony (ABC) Algorithm. *Applied Soft Computing*. 2008. Vol. 8. P. 687–697.

28. Базилевич Р., Кутельмах Р. Дослідження ефективності існуючих алгоритмів для розв'язання задачі комівояжера. *Вісник НУ "Львівська політехніка"*. Львів, 2009. С. 235–245.

29. Промислові мережі та інтеграційні технології в автоматизованих системах / Пупена О.М., Ельперін І.В., Луцька Н.М. та інші. К.: Ліра-К, 2021. 522 с.

30. Погорілий С.Д., Потебня А.В. Новітній метод розв'язання задач комбінаторної оптимізації великої розмірності. *Наукові праці ДонНТУ. Серія «Інформатика, кібернетика та обчислювальна техніка»*. 2014. №1. С. 114–125.

31. Liu J.-L., Li C.-C. An Improved Artificial Bee Colony Algorithm Applied to Engineering Optimization Problems. *Journal of information science and engineering*. 2016. Vol. 32. P. 863–886.

32. A Discrete JAYA Algorithm Based on Reinforcement Learning and Simulated Annealing for the Traveling Salesman Problem / J. Xu, W. Hu, W. Gu,

Y. Yu. *Mathematics*. 2023. Vol. 11. 23 p. URL: https://ideas.repec.org/a/gam/jmathe/v11y2023i14p3221-d1199888.html (режим доступу 29.11.2023).

33. Python documentation. URL: https://docs.python.org/3/index.html (дата звернення 04.11.2023).

34. Кавецький В.В., Козловський В.О., Причепа І.В. Економічне обґрунтування інноваційних рішень. Вінниця: ВНТУ, 2016. 113 с.

35. Козловський В.О., Лесько О.Й., Кавецький В.В. Методичні вказівки до виконання економічної частини магістерських кваліфікаційних робіт. Вінниця: ВНТУ, 2021. 42 с.

ДОДАТКИ

Додаток А

(обов'язковий)

Технічне завдання

ЗАТВЕРДЖУЮ

завідувач кафедри АІІТ

д.т.н., проф. Бісікало О. В.

«___»_____ 2023 р.

ТЕХНІЧНЕ ЗАВДАННЯ

на магістерську кваліфікаційну роботу

«Розробка інтелектуального модуля

для розв'язання задачі логістики останньої милі»

08-31.МКР.014.02.000 ТЗ

Керівник роботи:

к.т.н., доц. каф. АІІТ

Іванов Ю. Ю.

«___» _____ 2023 р.

Виконавець:

ст. гр. 1АКІТ-22м

Назаренко В. О.

«___» _____ 2023 р.

Вінниця ВНТУ – 2023

1. Назва та галузь застосування

Робота спрямована на розв'язання задачі логістики з використанням алгоритму штучної бджолиної колонії, який можна використовувати для розв'язання різноманітних задач дискретної оптимізації, особливо оптимізації маршрутів.

2. Підстави для розробки

Розробку здійснювати на підставі наказу по університету № 247 від 18.09.2023 та завдання до магістерської кваліфікаційної роботи, складеного та затвердженого кафедрою «Автоматизації та інтелектуальних інформаційних технологій».

3. Мета та призначення розробки

Метою роботи є підвищення ефективності розв'язання задачі логістики останньої милі з використанням алгоритму штучної бджолиної колонії за рахунок застосування нової моделі поведінки бджіл.

4. Джерела розробки

1. Otri S. Improving the Bees Algorithm for Complex Optimisation Problems. Cardiff University, 2011. 220 p.

2. Karaboga D., Gorkemli B. Solving Traveling Salesman Problem by Using Combinatorial Artificial Bee Colony Algorithms. *International Journal on Artificial Intelligence Tools*. 2019. Vol. 28. P. 1–28.

3. Sahin M. Improvement of the Bees Algorithm for Solving the Traveling Salesman Problems. *Bilişim teknolojileri dergisi*. 2022. Vol. 15. P. 65–74.

5. Показники призначення

Основні технічні вимоги та мінімальні системні вимоги до програми: операційна система *Windows* 7 або новіша; наявність клавіатури та миші;

процесор *Core* i5 (*Ryzen* 5) або кращий; оперативна пам'ять – 8 ГБ; відеопам'ять – 512 МБ; жорсткий диск – 200 ГБ і більше.

Вхідні дані: файл зі структурою карти на площині (кількість міст n, координати x та y); параметри алгоритму (кількість бджіл; актуальність джерела нектару; кількість ітерацій алгоритму); кількість запусків для збору статистики; набір тестових задач. Результати роботи програми: довжина маршруту (L); похибки розрахунків; графічне відображення розв'язку.

6. Економічні показники

До економічних показників входять:

- термін окупності – до 3-ох років;
- абсолютний ефект від впровадження розробки – не менше 2300 тис. грн.;
- мінімальна дохідність – не менше 42%;
- інші економічні переваги у порівнянні з аналогами.

7. Стадії розробки

1. Розділ 1 «Аналіз методів розв'язання задачі логістики останньої милі» має бути виконаний до 10.10.23.

2. Розділ 2 «Розробка математичного апарату ройового алгоритму для розв'язання задачі логістики останньої милі» має бути виконаний до 25.10.23.

3. Розділ 3 «Розробка інтелектуального програмного модуля та експериментальні дослідження» має бути виконаний до 10.11.23.

4. Економічний розділ має бути виконаний до 26.11.23.

8. Порядок контролю та приймання

1. Рубіжний контроль провести до 04.12.23.

2. Попередній захист магістерської кваліфікаційної роботи провести до 05.12.23.

3. Захист магістерської кваліфікаційної роботи провести до 19.12.23.

Додаток Б

(обов'язковий)

ІЛЮСТРАТИВНА ЧАСТИНА

РОЗРОБКА ІНТЕЛЕКТУАЛЬНОГО МОДУЛЯ ДЛЯ РОЗВ'ЯЗАННЯ ЗАДАЧІ ЛОГІСТИКИ ОСТАННЬОЇ МИЛІ

Зав. кафедри АІІТ _____ д-р техн. наук, професор каф. АІІТ Бісікало О. В.

(підпис) (науковий ступінь, вчене звання, ініціали та прізвище)

Керівник роботи _____ канд. техн. наук, доцент каф. АІІТ Іванов Ю. Ю.

(підпис) (науковий ступінь, вчене звання, ініціали та прізвище)

Тех. контроль _____ канд. техн. наук, доцент каф. АІІТ Іванов Ю. Ю.

(підпис) (науковий ступінь, вчене звання, ініціали та прізвище)

Нормоконтроль _____ канд. техн. наук, доцент каф. АІІТ Іванов Ю. Ю.

(підпис) (науковий ступінь, вчене звання, ініціали та прізвище)

Опонент _____ _____

(підпис) (науковий ступінь, вчене звання, ініціали та прізвище)

Студент гр. 1АКІТ-22м _____ Назаренко В. О.

(підпис) (ініціали та прізвище)

Рисунок Б.1 – Схема роботи рою

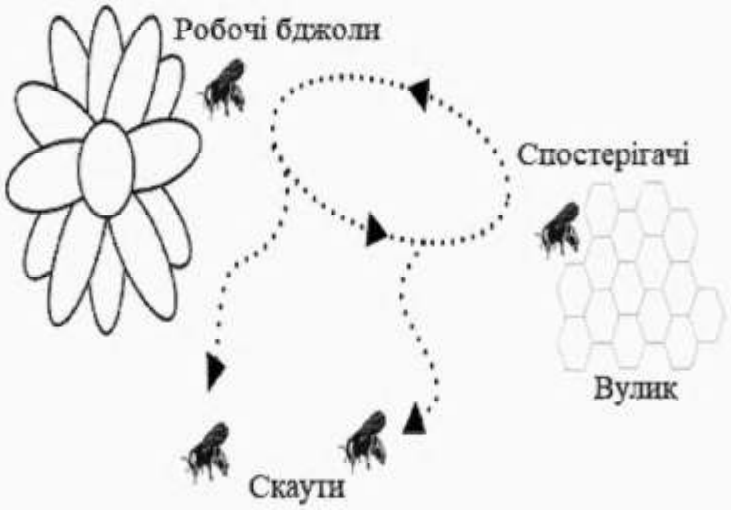


Рисунок Б.2 – Приклад поведінки бджіл у природі

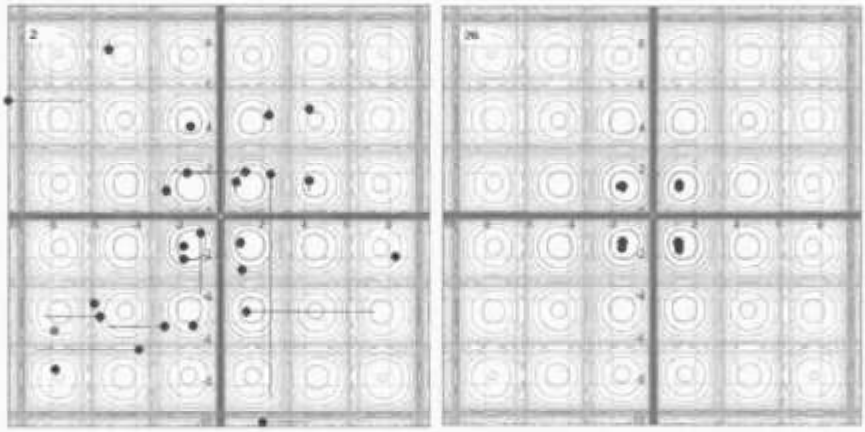


Рисунок Б.3 - Приклад роботи алгоритму штучної бджолиної колонії

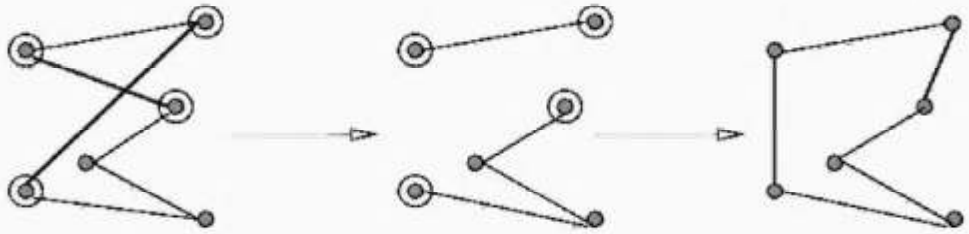


Рисунок Б.4 – Приклад геометричної постоптимізаційної евристики

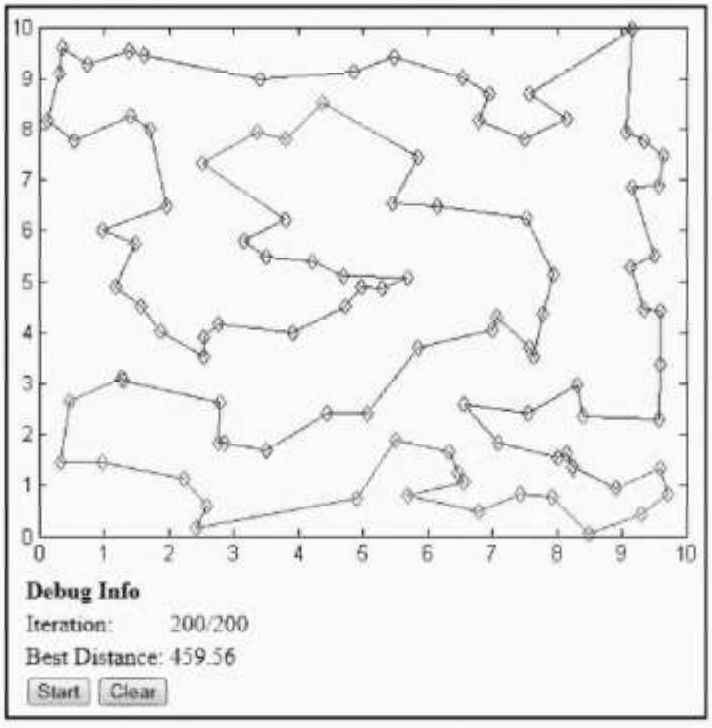


Рисунок Б.5 – Результат роботи програми

Рисунок Б.6 – Схема програми

Додаток В

(обов'язковий)

Лістинг програмного забезпечення

```
##########################################################################
import matplotlib.pyplot as plt
import numpy as np
import random as r

class Node:
    def __init__(self, idn, x, y):
        self.idn = idn
        self.pos = np.array((float(x), float(y)))

class Bee:
    def __init__(self):
        self.choosen_nodes = []
        self.recuiter = True
        self.distance = 0.0

    def choose_rand_move(self, move, nods):
        for i in range(move):
            if self.is_complete():
                break
            else:
                sel = nods[r.randint(0, len(nodes) - 1)]
                while sel in self.choosen_nodes:
                    sel = nods[r.randint(0, len(nodes) - 1)]
                self.choosen_nodes.append(sel)
            self.total_distance()

    def change_role(self, role):
        self.recuiter = role

    def replace_nodes(self, nods):
        self.choosen_nodes = nods
        self.total_distance()

    def total_distance(self):
        distance = 0.0
        for i in range(len(self.choosen_nodes) - 1):
            node1 = self.choosen_nodes[i]
            node2 = self.choosen_nodes[i + 1]
            distance += np.linalg.norm(node1.pos - node2.pos)
        distance += np.linalg.norm(self.choosen_nodes[-1].pos - self.choosen_nodes[0].pos)
        self.distance = distance
```

```
def is_complete(self):
    if len(self.choosen_nodes) >= len(nodes):
        return True
    else:
        return False

def load_nodes(filename):
    ret = []
    with open(filename) as f:
        nodes_s = f.readlines()
    nodes_s = [x.strip() for x in nodes_s]
    for n in nodes_s:
        node = n.split(' ')
        ret.append(Node(node[0], node[1], node[2]))
    return ret

nodes = load_nodes("data")

def main():
    epoch = 1000
    n_bee = 50
    n_move = 3
    bees = []
    best_bee = Bee()
    e = 0
    for i in range(n_bee):
        bees.append(Bee())
    while not best_bee.is_complete():
        print ("\nEpoch", e + 1)
        print (forward pass)
        for bee in bees:
            bee.choose_rand_move(n_move, nodes)
        print ("evaluating")
        bees = sorted(bees, key=lambda be: be.distance, reverse=False)
        best_bee = bees[0]
        print ("Best distance so far", best_bee.distance)
        print ("Best route so far", [n.idn for n in best_bee.choosen_nodes])
        print ("Bees are making decision to be recruiter or follower")
        Cmax = max(bees, key=lambda b: b.distance).distance
        Cmin = min(bees, key=lambda b: b.distance).distance
        recruiters = []
        for bee in bees:
            Ob = (Cmax - bee.distance) / (Cmax - Cmin)
            probs = np.e ** (-(1 - Ob) / (len(bee.choosen_nodes) * 0.01))
            rndm = r.uniform(0, 1)
            if rndm < probs:
                bee.change_role(True)
                recruiters.append(bee)
            else:
                bee.change_role(False)
        print ("number of recruiter", len(recruiters))
        print ("Bees are choosing their recruiter")
```

```
        divider = sum([(Cmax - bee.distance) / (Cmax - Cmin) for bee in recruiters])
        probs = [((Cmax - bee.distance) / (Cmax - Cmin)) / divider for bee in recruiters]
        cumulative_probs = [sum(probs[:x + 1]) for x in range(len(probs))]
        for bee in bees:
            if not bee.recuiter:
                rndm = r.uniform(0, 1)
                selected_bee = Bee()
                for i, cp in enumerate(cumulative_probs):
                    if rndm < cp:
                        selected_bee = recruiters[i]
                        break
                bee.replace_nodes(selected_bee.choosen_nodes[:])
        e += 1

def ABC_TSP():
    x = [node.pos[0] for node in nodes]
    y = [node.pos[1] for node in nodes]
    l = [node.idn for node in nodes]
    fig, ax = plt.subplots()
    ax.scatter(x, y)
    for i, lbl in enumerate(l):
        ax.annotate(lbl, (x[i], y[i]))
    plt.show()

main()
################################################################################
```

Додаток Г

(обов'язковий)

Довідка про впровадження

Додаток Д

(обов'язковий)

Протокол перевірки МКР

ПРОТОКОЛ
ПЕРЕВІРКИ КВАЛІФІКАЦІЙНОЇ РОБОТИ
НА НАЯВНІСТЬ ТЕКСТОВИХ ЗАПОЗИЧЕНЬ

Назва роботи: Розробка інтелектуального модуля для розв'язання задачі логістики останньої милі.

Тип роботи: магістерська кваліфікаційна робота

Підрозділ: кафедра автоматизації та інтелектуальних інформаційних технологій, факультет інтелектуальних інформаційних технологій та автоматизації

Показники звіту подібності Plagiat.pl (StrikePlagiarism)

Оригінальність _____% Схожість _____%

Аналіз звіту подібності (відмітити потрібне):

- ■ Запозичення, виявлені у роботі, оформлені коректно і не містять ознак плагіату
- □ Виявлені у роботі запозичення не мають ознак плагіату, але їх надмірна кількість викликає сумніви щодо цінності роботи і відсутності самостійності її автора. Роботу направити на розгляд експертної комісії кафедри.
- □ Виявлені у роботі запозичення є недобросовісними і мають ознаки плагіату та/або в ній містяться навмисні спотворення тексту, що вказують на спроби приховування недобросовісних запозичень

Особа, відповідальна за перевірку _____ Маслій Р. В.
(підпис) (прізвище, ініціали)

Ознайомлені з повним звітом подібності, який був згенерований системою Plagiat.pl (StrikePlagiarism) щодо роботи.

Автор роботи _____ Назаренко В. О.
(підпис) (прізвище, ініціали)

Керівник роботи _____ Іванов Ю. Ю.
(підпис) (прізвище, ініціали)