

Бакалаврська кваліфікаційна робота на тему:

**ДОСЛІДЖЕННЯ МОДИФІКАЦІЇ АЛГОРИТМУ ШТУЧНОЇ
МУРАШИНОЇ КОЛОНІЇ НА ОСНОВІ ПРИБЛИЗНОГО
НЕДЕТЕРМІНОВАНОГО ПОШУКУ ПО ДЕРЕВУ**

Виконала: студентка IV курсу, групи ІІСТ-216
Спеціальність 126 – Інформаційні системи та
технології

Витвицька К. В.

« 12 » 06 2025 р.

Керівник: к.т.н., доцент каф. АІТ

Севастьянов В. М.

Рецензент: д.т.н., професор каф. КСУ

Ковтун В. В.

« 13 » 06 2025 р.

Допущено до захисту
зав. кафедри АІТ

д.т.н., проф. Бісікало О. В.

« 16 » 06 2025 р.

Вінниця ВНТУ – 2025 рік

Вінницький національний технічний університет
Факультет інтелектуальних інформаційних технологій та автоматизації
Кафедра автоматизації та інтелектуальних інформаційних технологій
Рівень вищої освіти перший (бакалаврський)
Галузь знань 12 – «Інформаційні технології»
Спеціальність 126 – «Інформаційні системи та технології»
Освітньо-професійна програма Інтелектуальні інформаційні системи

ЗАТВЕРДЖУЮ

завідувач кафедри АІТ

д.т.н., проф. Бісікало О. В.

«16» 06 2025 р.

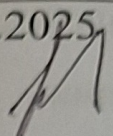
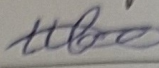
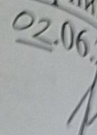
З А В Д А Н Н Я

НА БАКАЛАВРСЬКУ КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТЦІ

Витвицькій Катерині Володимирівні

1. Тема роботи: Дослідження модифікації алгоритму штучної мурашиної колонії на основі приблизного недетермінованого пошуку по дереву.
Керівник роботи: к.т.н., доцент каф. АІТ Севастьянов В. М.
Затверджені наказом ВНТУ від «20» 03 2025 року № 97.
2. Строк подання роботи студентом: «до 23» 06 2025 р.
3. Вихідні дані до роботи: кількість ітерацій N ; розмір популяції m ; коефіцієнти алгоритму α , r_0 та q ; координати пунктів x та y ; значення швидкості v та вантажності засобу w ; цінність товарів pr ; рента засобу r ; текстовий файл із даними задачі.
4. Зміст текстової частини: вступ; аналіз предметної області; розробка математичної моделі алгоритму оптимізації; розробка програмного забезпечення та експериментальні дослідження; висновки; список використаних джерел.
5. Перелік ілюстративного матеріалу (з точним зазначенням обов'язкових креслень): схема програми; графічне відображення кривої збіжності алгоритму; приклад пошуку маршруту.

6. Консультанти розділів роботи

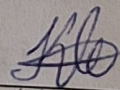
Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
1-3	Севастьянов В. М. к.т.н., доцент каф. АІТ Іванов Ю. Ю. к.т.н., доцент каф. АІТ	21.03.2025  	02.06.2025  

7. Дата видачі завдання: «21» 03 2025 р.

КАЛЕНДАРНИЙ ПЛАН

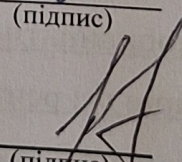
№ з/п	Назва та зміст етапу	Термін виконання		Прим.
		початок	закінчення	
1	Аналіз предметної області	21.03.2025	04.04.2025	Викон.
2	Розробка математичної моделі алгоритму оптимізації	05.04.2025	26.04.2025	Викон.
3	Розробка програмного забезпечення та експериментальні дослідження	27.04.2025	16.05.2025	Викон.
4	Оформлення пояснювальної записки і графічного матеріалу	17.05.2025	27.05.2025	Викон.
5	Попередній захист роботи	28.05.2025	02.06.2025	Викон.
6	Остаточний захист роботи	16.06.2025	23.06.2025	Викон.

Студентка


(підпис)

Витвицька К. В.
(прізвище та ініціали)

Керівник роботи


(підпис)

Севастьянов В. М.
(прізвище та ініціали)

АНОТАЦІЯ

Бакалаврська кваліфікаційна робота складається з 72 сторінок формату А4, на яких є 23 рисунка, 5 таблиць, список використаних джерел містить 25 найменувань.

У даній роботі розглянуто метаевристичний алгоритм оптимізації для пошуку маршрутів, який представляє собою приблизний недетермінований пошук по дереву з використанням особливостей взаємодії колонії програмних мурашок. Розроблено алгоритмічне та програмне забезпечення, яке дозволяє виконати відповідні експериментальні дослідження.

Ключові слова: оптимізація, метаевристика, маршрут, пошук по дереву, феромон.

ABSTRACT

The bachelor's thesis consists of 72 A4 pages with 23 figures, 5 tables, the list of sources used contains 25 items.

In the thesis has been analyzed metaheuristic optimization algorithm, which is presented as an approximate nondeterministic tree search using the features of the interaction of a colony of software ants. Also have been developed algorithms and software, that allows to perform corresponding experimental researches.

Keywords: optimization, metaheuristics, route, tree search, pheromone.

ЗМІСТ

ВСТУП	3
1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ	6
1.1 Постановка задачі	6
1.2 Аналіз алгоритмів розв’язання задачі	16
1.3 Висновки	24
2 РОЗРОБКА МАТЕМАТИЧНОЇ МОДЕЛІ АЛГОРИТМУ ОПТИМІЗАЦІЇ	25
2.1 Особливості роботи метаевристик	25
2.2 Біологічні основи алгоритму	29
2.3 Математична модель	33
2.4 Висновки	41
3 РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ	
ТА ЕКСПЕРИМЕНТАЛЬНІ ДОСЛІДЖЕННЯ	42
3.1 Вибір мови програмування	42
3.2 Опис програмного забезпечення	44
3.3 Експерименти	46
3.4 Висновки	53
ВИСНОВКИ	54
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ	55
ДОДАТКИ	59
Додаток А (обов’язковий). Технічне завдання	60
Додаток Б (обов’язковий). Ілюстративна частина	63
Додаток В (обов’язковий). Лістинг програми	66
Додаток Г (обов’язковий). Протокол перевірки БКР	70

ВСТУП

Актуальність. Логістика орієнтується на оптимізацію процесів зберігання, розподілу та переміщення товарів, використовуючи інформаційні технології для забезпечення злагодженої роботи всієї ланки постачання. Вона стикається із численними викликами, зокрема оптимізацією витрат на транспортування товарів. Наприклад, однією з головних проблем є доставка на останню милю, яка часто найдорожча ланка у ланцюзі постачання [1]. Для досягнення успіху в технологічних перегонах компаніям необхідно швидко реагувати на зміни зовнішнього середовища. Тому швидко розвивається галузь роботів-кур'єрів, що зумовлено здешевленням необхідних технологій та збільшенням вантажопідйомності пристроїв. Основне їх завдання — доставка товарів від пункту відправлення до пункту замовлення за мінімальний час, обираючи найкоротший маршрут. Оскільки роботи працюють автономно, самостійно розпізнаючи навколишнє середовище та прокладаючи свій шлях, постає задача оптимізації маршруту, що зводиться до задачі пошуку найкоротшого циклічного маршруту з умовами [2].

Останнім часом спостерігається збільшення кількості наукових робіт різних авторів, які досліджують задану задачу з використанням ройового інтелекту [3, 4], зокрема, Т. Шутле [5], П. Лі [6], С. Штовби [7], Т. Желдака [8]. Вони виділяють алгоритми засновані на взаємодії особин у мурашиній колонії, як ефективні пошукові процедури, здатні розв'язувати логістичні задачі. У даній роботі розглядається суттєва модифікація даного алгоритму, яка базується на застосуванні приблизного недетермінованого

пошуку по дереву. Актуальною задачею є дослідження даного алгоритму на практиці у ході розв'язання задач логістики.

Мета і задачі дослідження. Метою роботи є підвищення функціональних можливостей програмного забезпечення для експериментального дослідження розв'язання задач логістики за рахунок використання алгоритму штучної мурашиної колонії на основі приблизного недетермінованого пошуку по дереву.

Для досягнення мети необхідно розв'язати наступні задачі:

- розглянути постановку логістичних задач на графах, виділити тип задачі, яка буде тестовою, та проаналізувати методи її розв'язання;
- виконати теоретичне дослідження алгоритму штучної мурашиної колонії на основі приблизного недетермінованого пошуку по дереву;
- розробити програмні засоби та оцінити ефективність їх роботи.

Об'єктом дослідження є процес пошуку оптимального маршруту на карті.

Предметом дослідження є моделі, методи та інструментальні засоби для розв'язання заданої логістичної задачі.

Методи дослідження. В роботі використано положення теорії графів, дискретної оптимізації та ройового інтелекту під час дослідження розглянутого алгоритму оптимізації; експериментальне дослідження для аналізу та перевірки достовірності отриманих теоретичних положень. Обробка даних виконувалася за допомогою мови програмування Python.

Науково-технічний результат роботи полягає у тому, що на основі теоретичних положень розроблено математичне, алгоритмічне та програмне

забезпечення, яке дозволяє застосовувати алгоритм штучної мурашиної колонії на основі приблизного недетермінованого пошуку по дереву.

У роботі отримані результати, які мають *практичну цінність*, а саме: наведено експериментальні дослідження ефективності застосування представленого алгоритму на різних тестових задачах.

Апробація результатів та публікації. За результатами дослідження опубліковано тези доповіді [9] на Всеукраїнській науково-технічній конференції підрозділів ВНТУ (м. Вінниця, 2025).

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Постановка задачі

Транспортна логістика відіграє ключове значення в діяльності практично кожного підприємства. Особливу увагу приділяють впровадженню алгоритмів оптимізації маршрутів, які дозволяють знижувати витрати на транспортування товарів, економлячи до декількох десятків відсотків від їх вартості.

Крім доставки товарів, є ще низка галузей, в яких необхідно застосовувати методи оптимізації маршрутів: перевезення людей, туризм, управління супутниками, кластеризація масивів даних, проектування комунікаційних мереж, з'єднання населених пунктів лініями енергопередавання і газопостачання, проектування топології інтегральних схем тощо [1]. 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.

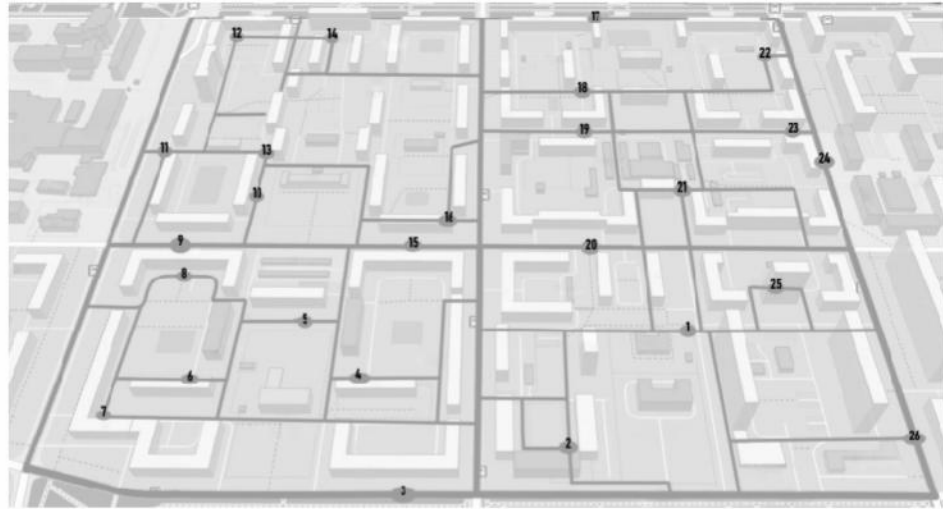


Рисунок 1.1 – Схема мікрорайону для доставки товарів

Загалом розглядається n вузлових пунктів, які пов'язані між собою транспортною мережею. Відома матриця відстаней від кожного пункта до інших [10, 11]:

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

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

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

Отже, a_{ij} може набувати булевих значень. У ході розв'язання такої оптимізаційної задачі необхідно знайти набір пунктів, при якому забезпечується екстремальне значення цільової функції [10, 11]:

$$f(X) \rightarrow \text{extremum} . \quad (1.3)$$

Цільовою функцією є мінімізація всього маршруту [10, 11]:

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

Обмеження щодо одноразового в'їзду в кожен пункт задається таким чином [10, 11]:

$$\sum_{i=1}^n a_{ij} = 1, \quad j = 1, 2, \dots, n, i \neq j . \quad (1.5)$$

Обмеження щодо одноразового виїзду з кожного пункту [11, 12]:

$$\sum_{j=1}^n a_{ij} = 1, \quad i = 1, 2, \dots, n, i \neq j . \quad (1.6)$$

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

процесі розв'язування задачі набуватимуть значень порядкових номерів вузлових пунктів за оптимальним маршрутом руху. Запишемо обмеження, які усувають можливість існування підциклів [12]:

$$u_i - u_j + na_{ij} \leq n - 1. \quad (1.7)$$

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

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

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.

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 [10].

Вагомий внесок в дослідження даної задачі внесли Дж. Данціг та Д.Р. Фалкерсон, які сформулювали постановку задачі оптимізації, яку в подальшому інтерпретували до теорії графів та лінійного цілочисельного програмування. У 1972 році Р. Карп довів *NP*-повноту задачі та представив теоретичне обґрунтування складності пошуку розв'язків даної задачі на практиці. Програміст-дослідник Г. Райнелт зібрав набір стандартизованих екземплярів задачі різного ступеня складності TSPLIB для порівняння результатів роботи різних груп дослідників [12].

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 кожен вершину заданого графа G тільки один раз (за винятком стартової вершини циклу), називається маршрутом (рисунок 1.2).

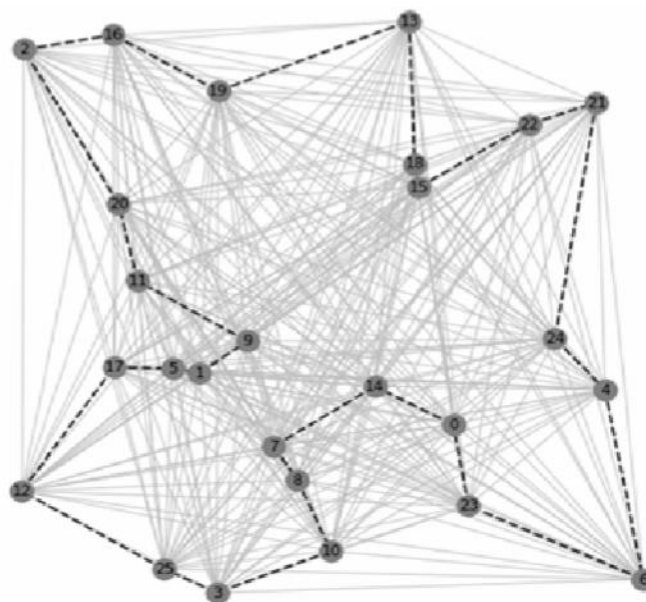


Рисунок 1.2 – Приклад розв'язку

Модифікацією цієї задачі є наступна комплексна комбінаторна задача перевезення товарів циклічним маршрутом із обмеженням на вантажність засобу перевезення. На карті або графі $G(V, E)$ задані вершини $V = \{v_i\}$ ($i = \overline{1, n}$), які відповідають вузловим точкам, а ребра $E = \{e_j\}$ відображають комунікації між ними з вагами c_j . У цих пунктах розташовані m предметів (різні кількість та вартість), причому для кожного з них задана вага w_k та вартість c_k . У засобу є вантажність W , також він не може більше збирати предмети, коли повернеться до початкового міста. Необхідно пройти мінімальний циклічний маршрут без повторень, взявши в пройдених пунктах предмети максимальної вартості таким чином, щоб їх сумарна вага не перевищувала W [5, 13, 14].

Дохід складається з вартості взятих предметів із вирахуванням вартості оренди засобу, яка визначається ціною за одиницю часу R і часом проходження по всьому маршруту T :

$$f(x, z) = \sum_{k \in A} c_k - R \cdot T(x, z) = \sum_{k \in A} c_k - R \cdot \sum_{i=1}^n t_i \rightarrow \max, \quad (1.9)$$

$$g(z) = \sum_{k \in A} w_k \leq W, \quad (1.10)$$

де x – послідовність пройдених пунктів;

z – набір товарів;

n – кількість пунктів на карті;

$T(x, z)$ – час, впродовж якого було пройдено весь маршрут x і взято z речей;

A – множина предметів у засобі (відповідні порядкові номери);

t – час проходження фрагмента маршруту, який залежить від відстані c між містами i та j і поточної швидкості засобу v_c :

$$t_{ij} = \frac{c_{ij}}{v_c}. \quad (1.11)$$

Швидкість переміщення засобу залежить від поточної ваги (знижується зі збільшенням його ваги):

$$v_c = v_{\max} - \left(\sum_{k \in A} w_k \right) \cdot \frac{v_{\max} - v_{\min}}{W}, \quad (1.12)$$

де $v_{\max}, v_{\min}$ – максимальна та мінімальна швидкість засобу відповідно.

Для гарантування того, що результат цільової функції задачі буде позитивним числом пропонується генерувати ціну оренди за рюкзак за формулою:

$$R = b \cdot \frac{E_p}{E_t}, \quad (1.13)$$

де b – незалежне дійсне випадкове число, рівномірно розподілене на інтервалі $[0,05; 0,25]$;

E_p, E_t – оцінки найбільшої вартості предметів та найменшого часу проходження маршруту відповідно.

Геометрично граф G – це фігура на площині, яка задається множиною точок (вершин) $V = \{v_1, v_2, \dots, v_n\}$ і множиною ліній (ребер) $E = \{e_1, e_2, \dots, e_m\}$, 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 1.3) [10].

Шляхом орієнтованого графа називається послідовність дуг, в якій кінцева вершина будь-якої дуги, відмінної від останньої, є початковою вершиною наступної. Орієнтованим циклом називається такий шлях, в якому кожна дуга використовується не більше одного разу, а відповідний граф має назву ейлерового. Простим циклом називається такий шлях, в якому кожна вершина використовується тільки раз, а відповідний граф є гамільтоновим [10].

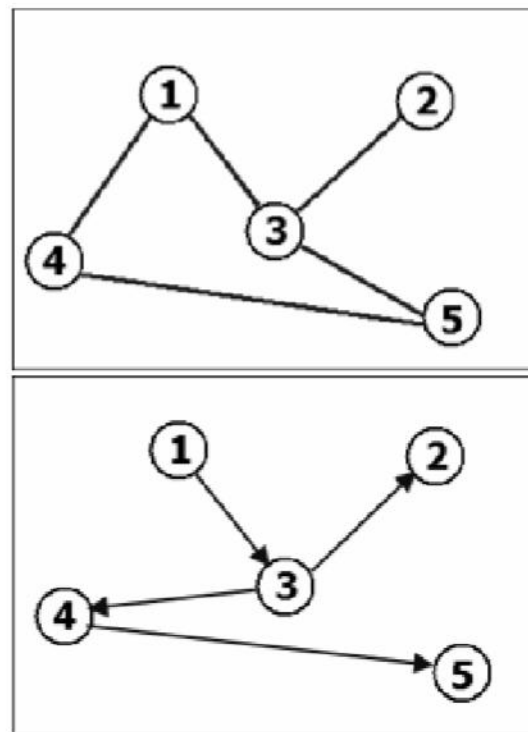


Рисунок 1.3 – Приклад неорієнтованого і орієнтованого графів

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 1.4А, то граfi на рисунках 1.4Б та 1.4В є остовом цього графа [10].

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.

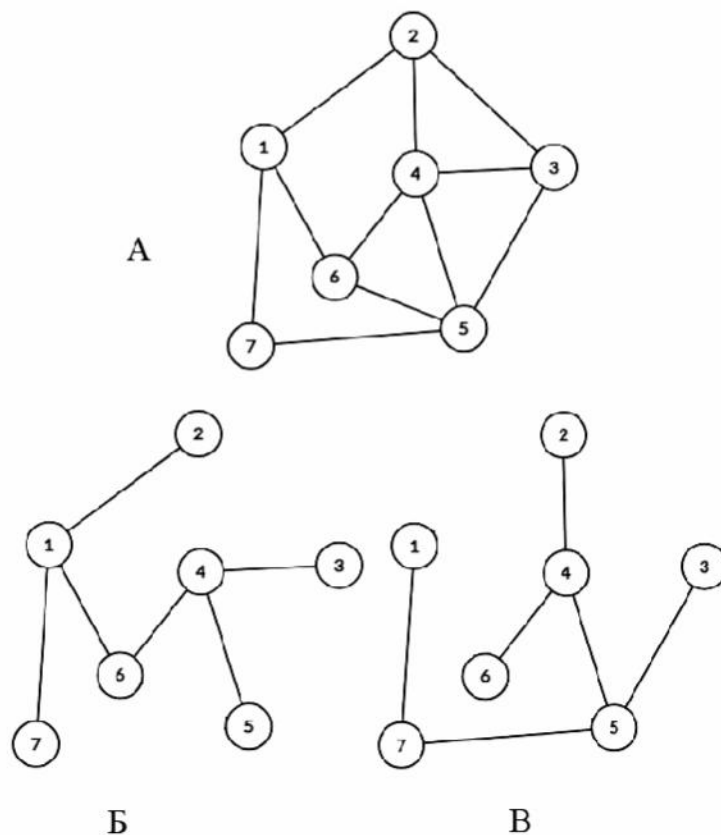


Рисунок 1.4 – Граф G (А) та його остови (Б, В)

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 1.5) [10].

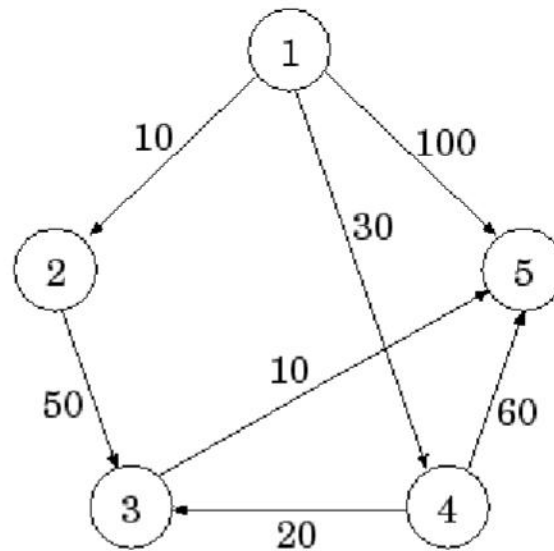


Рисунок 1.5 – Граф зі зваженими ребрами

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 [10].

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:

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

1.2 Аналіз алгоритмів розв'язання задачі

Класифікація алгоритмів розв'язання задач пошуку маршрутів наведена на рисунку 1.6 [3, 9].

Метод грубої сили є найпростішим, але водночас й найнеефективнішим. Він дозволяє знайти розв'язок у будь-якому випадку (якщо він існує), але зазвичай розмірність задачі така велика, що розв'язати її простим перебором варіантів стає неможливим, оскільки на графі може бути кілька гамільтонових циклів, а потрібно знайти найменший. Зі збільшенням кількості вершин графа кількість варіантів перебору зростає дуже швидко:

$$B = \begin{cases} \frac{(n-1)!}{2} & \text{для симетричної задачі;} \\ (n-1)! & \text{для асиметричної задачі,} \end{cases} \quad (1.15)$$

Виникає “прокляття” розмірності, а це означає, що навіть при незначному збільшенні розмірності простору задачі час на її вирішення може стати непрактично великим [3].

Дерев'яний алгоритм базується на пошуці остовного дерева графа мінімальної ваги. Для цього можна застосувати алгоритми Пріма, Краскала або Боруки. На множині вершин остовного дерева, які мають непарну степінь, знаходиться паросполучення мінімальної ваги. Таких вершин в будь-якому остовному дереві парна кількість. Остовне дерево перетворюється в ейлеровий граф шляхом приєднання ребер відповідно до знайденого паросполучення. У отриманому графі знаходиться ейлеровий цикл, який перетворюється в гамільтоновий за допомогою видалення повторів вершин.

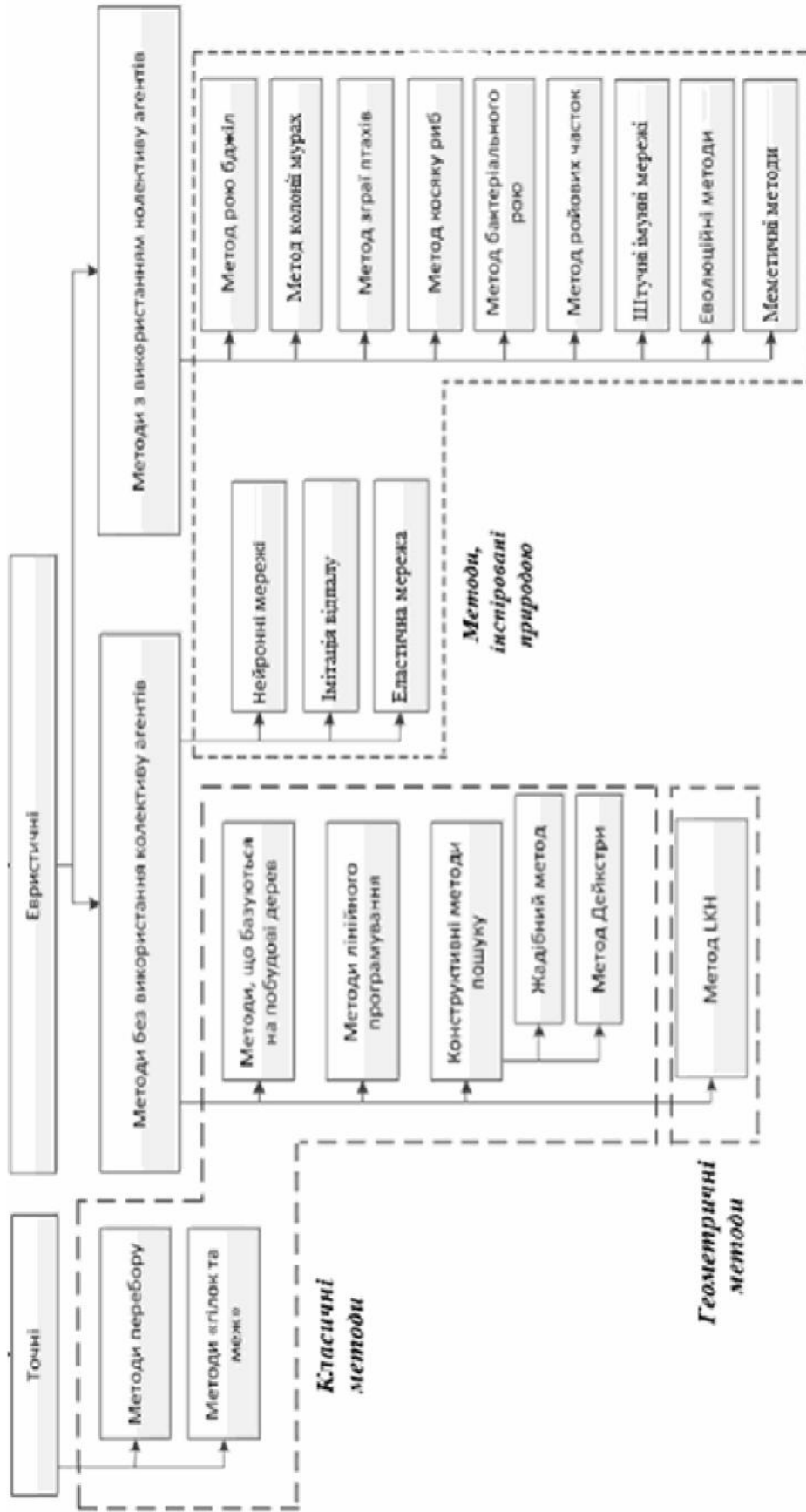


Рисунок 1.6 – Класифікація алгоритмів розв’язання задач пошуку маршрутів

Вважається універсальним методом, оскільки підходить для рішення асиметричної задачі комівояжера, тоді як інші методи пристосовані в основному для вирішення симетричної задачі.

Метод гілок і меж — узагальнена алгоритмічна стратегія, яка активно застосовується для розв’язання задач дискретної оптимізації. Представляє собою вдосконалений варіант повного перебору, де на кожному кроці відсіюються невігідні рішення. Основа методу — розбиття множини рішень за принципом “розділай і володарюй” через розгалуження та обчислення меж для кожної частини [3].

Проблема в тому, що через відсіювання деяких рішень на ранніх етапах, алгоритм не завжди забезпечує точність: відкинута на початку “невигідна” гілка може зрештою виявитися тією, що дає найкраще значення. Алгоритм вимагає повторення для $(n - 2)$ рівнів дерева, де n — кількість пунктів на карті (рисунок 1.7).

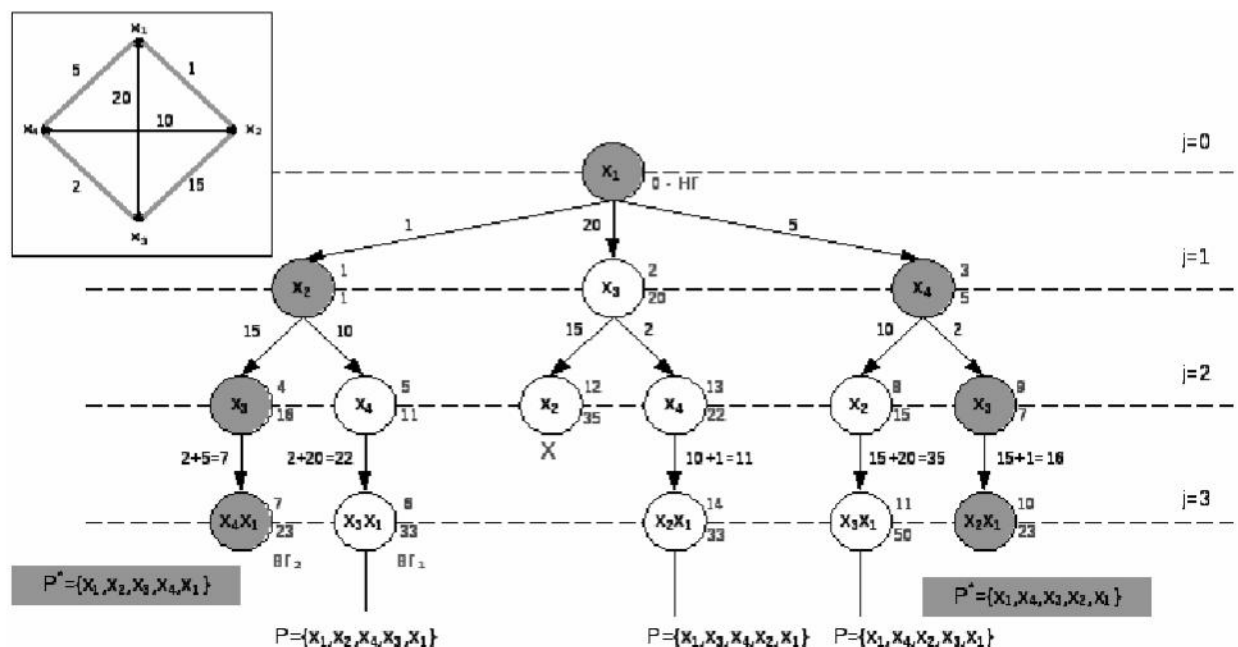


Рисунок 1.7 – Приклад роботи алгоритму гілок та меж

Евристичні алгоритми дозволяють знаходити квазіоптимальні розв'язки. Вони поділяються на два основних типи: алгоритми, які не використовують колектив інтелектуальних агентів, та алгоритми на основі взаємодії колективу програмних агентів. Залежно від того, чи евристика генерує новий маршрут, чи намагається покращити вже існуючий, їх можна класифікувати на конструктивні та ітеративні (післяоптимізаційні) евристики.

Алгоритм найближчого сусіда – один з найпростіших евристичних алгоритмів оптимізації маршрутів. Необхідно вибрати випадкову вузлову точку. Потім послідовно включати до маршруту нові точки, причому, кожен черговий пункт повинен бути найближчим до останнього вибраного пункту [3, 4].

Використання нейромереж ґрунтується на аналогії з механізмами встановлення впорядкованих нейронних зв'язків. Наприклад, еластична мережа 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 її застосування створюється ефект “гумки” та формується “натяг” адаптивного кільця, що допомагає шукати розв'язок задачі на площині (рисунок 1.8). Перевагами є мала кількість параметрів та швидкість роботи; недоліком є зациклювання алгоритму у поганих локальних оптимумах [11].

На практиці поширеними є ітеративні алгоритми локальної оптимізації. Для обчислення результату необхідно на вхід подати згенерований певним чином маршрут (для конструктивних евристик навпаки). Кількість з'єднань між вузлами, які приймають участь в перестановці, визначають складність

методу локальної оптимізації (коефіцієнт складності k). Тому такі евристики отримали назву k -opt-евристик.

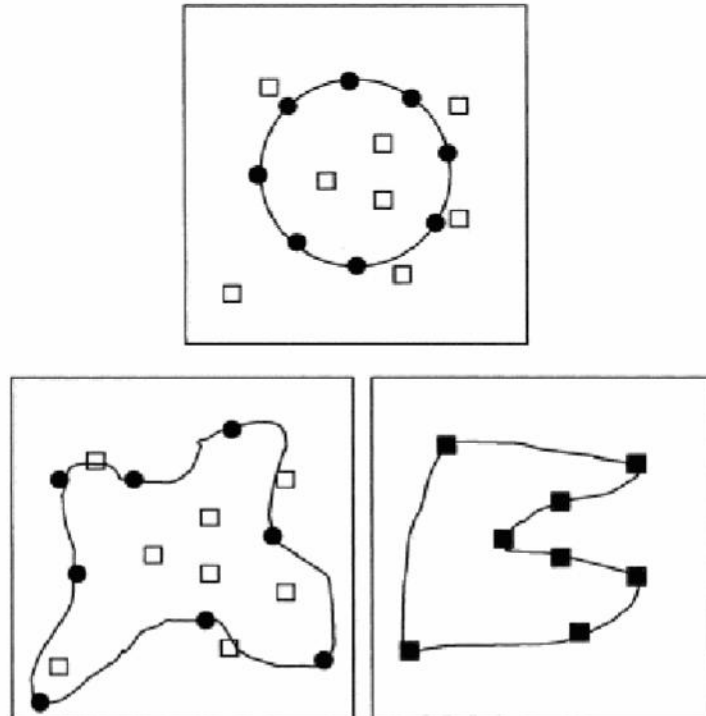


Рисунок 1.8 – Приклад роботи нейронної мережі

Однією з таких евристик є розроблена Керніганом, Ліном та Хельсгауном евристика LKH , яка застосовує методи локальної оптимізації низької складності 2 -opt, 3 -opt тощо [15]. Приклад 2 -opt та 3 -opt евристик наведений на рисунках 1.9 та 1.10.

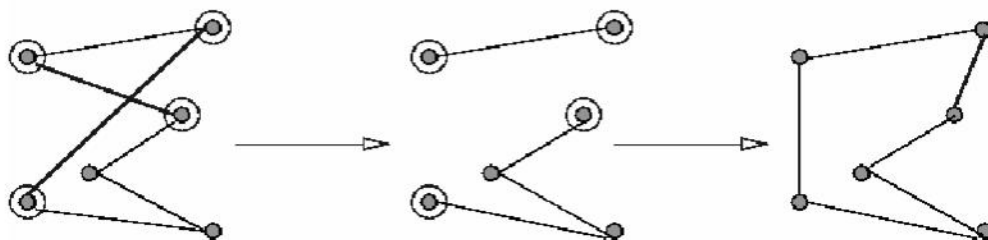


Рисунок 1.9 – Заміна ребер 2 -opt евристикою

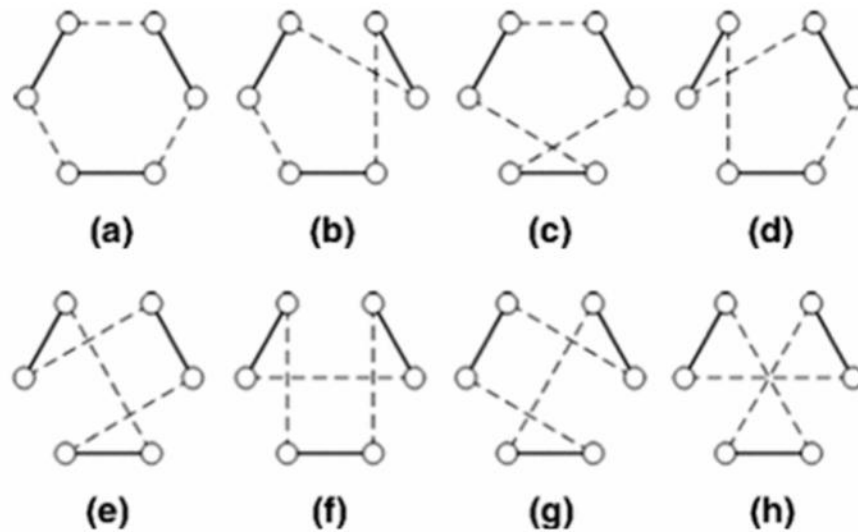


Рисунок 1.10 – Заміна ребер 3-opt евристикою

Метаевристичні алгоритми включають сучасні стратегії пошуку, які інспіровані живою та неживою природою [16, 17]. До них прийнято відносити алгоритм імітації відпалу, пошук із заборонами, алгоритм Ліна-Кернігана, імовірнісні жадібні алгоритми, генетичні алгоритми, наслідування поведінки мурашиної колонії, рою бджіл, зграї птахів, вовків, котів, кажанів, колонії мурах, косяку риб, рою часток, імунних систем тощо.

Ідея цих алгоритмів 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, ними можна знаходити як досить якісні розв'язки, так і досить віддалені від оптимального. Якість результатів та тривалість обчислень залежать від реалізації окремих елементів алгоритму.

Алгоритм 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 алгоритм для розв'язання задач оптимізації. Основна ідея 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 (рисунок 1.11) [3].

Алгоритм заснований на наслідуванні поведінки комах, а саме мурашиної колонії, може бути успішно застосований для розв'язання складних комплексних задач логістики. Результати перших експериментів із застосуванням мурашиних алгоритмів для вирішення задачі комівояжера були багатообіцяючими, проте далеко не кращими, порівняно з вже існуючими методами [7, 9].

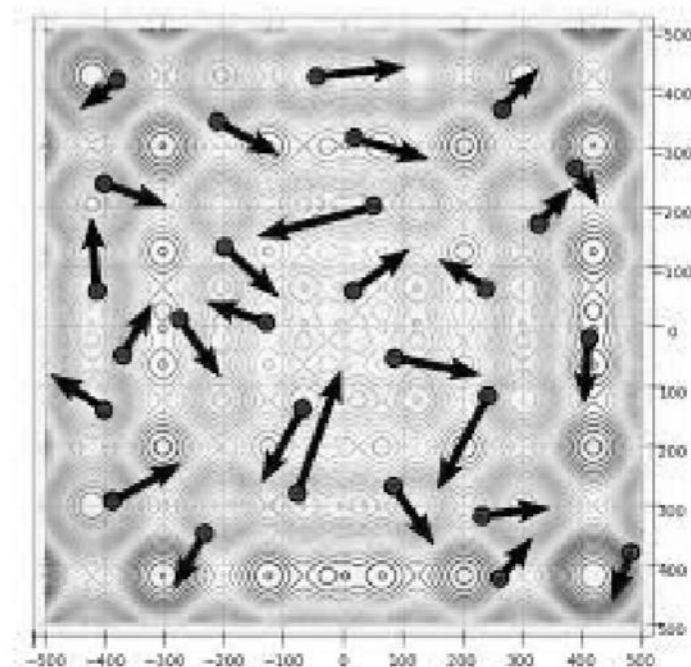


Рисунок 1.11 – Приклад роботи алгоритму рою часток

Але простота класичного мурашиного алгоритму залишала можливості для модифікацій, які стали предметом подальших досліджень фахівців у галузі дискретної оптимізації. Загалом це дозволило зробити дану евристику однією з найбільш поширених та ефективних саме для розв'язання задачі комівояжера та її модифікацій [5, 13, 18, 19].

За їх допомогою отримано хороші результати для багатьох складних комбінаторних задач: комівояжера, оптимізації маршрутів вантажівок, розмальовки графа, квадратичної задачі про призначення, задач календарного планування тощо. Особливо ефективними мурашині алгоритми виявилися при on-line оптимізації процесів у розподілених нестаціонарних системах, наприклад трафіків у телекомунікаційних мережах [16].

Мурашині алгоритми представляють собою ймовірнісну жадібну евристику. Ідея мурашиного алгоритму – моделювання поведінки мурах, пов'язаної з їхньою здатністю швидко знаходити найкоротший шлях від мурашника до джерела їжі і адаптуватися до нових умов навколишнього середовища.

У даній роботі розглядається суттєва модифікація алгоритму наслідування поведінки мурашиної колонії на основі приблизного випадкового пошуку по дереву [5]. Змінено метод обчислення ймовірності переходу до вершини, а також глобальне правило зміни концентрації феромона на ребрах графа, застосовано метод боротьби зі стагнацією обчислень. Крім того, ще одним шляхом покращення роботи методу є застосування геометричних евристик Ліна-Кернігана-Хельсгауна [15].

Таким чином, класифікація алгоритмів для задач пошуку маршрутів охоплює різноманітні стратегії, кожна з яких має свої переваги та обмеження

в залежності від специфіки задачі. Вибір алгоритму залежить від вимог до точності, швидкості та ресурсів, а також від розміру та складності конкретної задачі.

1.3 Висновки

У цьому розділі представлена математична постановка тестової задачі. Її обчислювальна складність ускладнює пошук точного рішення для великих графів за прийнятний час. Проведений аналіз показав, що необхідно використовувати алгоритми ройового інтелекту, серед яких перспективним є алгоритм штучної мурашиної колонії. Розглядається його модифікація, математичну модель якої представлено в наступному розділі.

2 РОЗРОБКА МАТЕМАТИЧНОЇ МОДЕЛІ АЛГОРИТМУ ОПТИМІЗАЦІЇ

2.1 Особливості роботи метаевристик

Метаевристики — це стратегії, які “керують” процесом пошуку рішення певної задачі. Мета метаевристики полягає у ефективному дослідженні простору пошуку для знаходження (майже) оптимальних рішень. Метаевристичні алгоритми варіюють від найпростіших процедур локального пошуку до складних процесів навчання. Загалом метаевристики об’єднують основні евристичні методи в рамках більш загальних алгоритмічних схем, які спрямовані на ефективне дослідження простору пошуку. Ці методи використовують випадковість, самонавчання, інтенсифікацію та диверсифікацію пошуку, адаптивні механізми управління, конструктивні еристики (наприклад, жадібні алгоритми) і методи локального пошуку. Добре реалізована метаевристика може дозволити знайти оптимальне рішення за розумний проміжок часу. Для більшості метаевристик можна тільки сподіватися (і для деяких довести асимптотичну збіжність), що певний допустимий розв’язок $f(\tilde{X})$, $\tilde{X} \in D$ наближається до оптимуму $f(X^*)$ [3].

Метаевристики є потужним і надзвичайно популярним класом оптимізаційних методів, які дозволяють знаходити розв’язок для широкого кола задач з різних предметних областей. Сила метаевристик полягає в їх здатності розв’язувати найскладніші задачі без знання простору пошуку.

Вони дозволяють поступово покращувати деякий поточний наближений розв'язок (рисунок 2.1). 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 [1, 4].

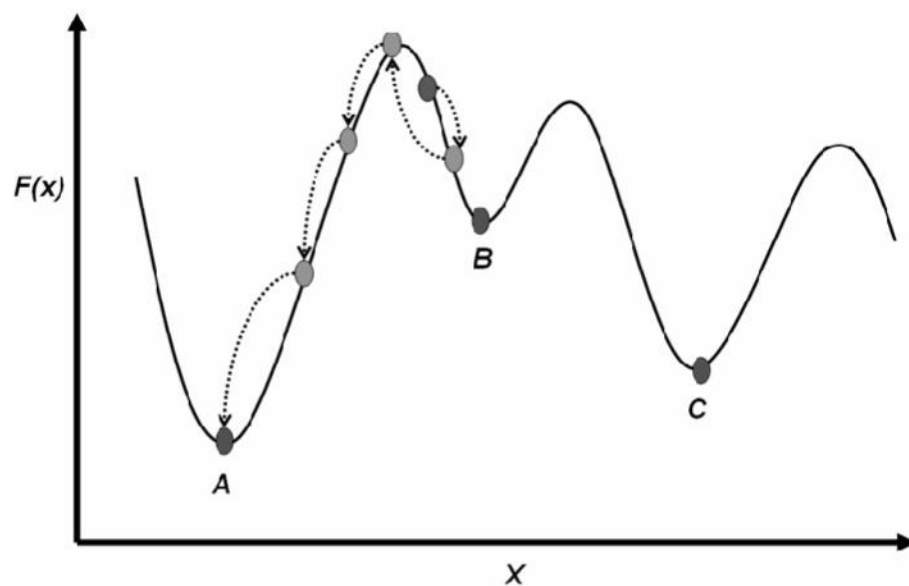


Рисунок 2.1 – Випадковий пошук оптимуму

Метаевристики прийнято ділити на методи локального пошуку (порогові, траєкторні), коли на кожній ітерації є одне допустиме рішення і відбувається перехід до наступного, і на методи, які працюють з сімейством (популяцією) рішень.

Траєкторні методи, наприклад алгоритм імітації відпалу, залишають в просторі пошуку траєкторію, послідовність рішень, де кожне рішення є сусіднім до попереднього щодо певного околу. Цільова функція уздовж траєкторії міняється немонотонно, що дозволяє “вибиратися” з локальних

екстремумів і знаходити кращі наближені рішення. Самоадаптація алгоритму дозволяє змінювати параметри, використовуючи передісторію пошуку.

До другої групи відноситься ройовий інтелект. На кожній ітерації знаходиться нове рішення задачі, яке базується вже не на одному, а на декількох рішеннях з популяції. У генетичних методах для цих цілей використовуються процедури схрещування і мутацій. У мурашиних алгоритмах застосовується інша ідея, заснована на збиранні статистичної інформації про найбільш вдалі рішення. Ця інформація враховується в імовірнісних жадібних алгоритмах і підказує, які саме компоненти рішень (ребра графа, підприємства, елементи системи технічних засобів тощо) найчастіше приводили до малої похибки. Ідея мурашиного алгоритму є спробою імітації поведінки мурашок, які майже не мають зору і орієнтуються за запахом залишеним попередниками. Сильний запах (феромон) є для них індикатором активності та діяльності попередників. Він акумулює в собі передісторію пошуку і підказує дорогу до мурашника [16].

Загалом системи з використанням колективу інтелектуальних агентів, як правило, складаються з множини агентів, які локально взаємодіють між собою і з навколишнім середовищем. Самі агенти поодиноці, зазвичай, є достатньо простими, але всі разом, локально взаємодіючи, вони створюють так званий ройовий інтелект. Модель ройового інтелекту представляє багатоагентну систему, яка складається з множини інтелектуальних програмних агентів, що здатні самостійно впродовж певного часу виконувати поставлене завдання, самонавчатися та самоорганізовуватися [20].

Алгоритм рою має свою власну поведінку і характеристики, але можна виділити наступні етапи роботи таких методів [3]:

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 не тільки вибраний окіл, але 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 $\tilde{X} = \arg \text{opt}(g \mid y = f(X))$ з урахуванням запобігання зацикленню.

6. Критерії зупинки. Метаевристика може бути зупинена згідно різних критеріїв: час обчислень, кількість ітерацій, темп покращення рішення задачі тощо. Може бути визначений більш ніж один критерій для управління різними фазами пошуку.

Далі буде представлено метаевристичний підхід до розв'язання задачі мінімізації замкнутого маршруту, заснований на роботі інтелектуальної мультиагентної системи. Доведено, що даний метод асимптотично збігається до оптимуму.

2.2 Біологічні основи алгоритму

Алгоритм приблизного недетермінованого пошуку по дереву є розширенням алгоритму, заснованого на ідеї мурашиної колонії, який був запропонований у роботі [3]. Внесено зміни в деякі елементи оригінального алгоритму, зокрема функцію пристосованості та розподілу феромону серед мурах. Ці модифікації дозволяють отриманому алгоритму наблизитися до структури алгоритмів пошуку по дереву з елементами стохастичності процесу, що робить алгоритм здатним до адаптації в реальному часі. Така властивість може бути корисною при роботі з великими або динамічними просторами пошуку.

Основу алгоритму складає поведінка мурах, які є соціальними комахами і живуть в колоніях. Приблизно 2% усіх комах є соціальними, половину з яких складають мурашки — маленькі істоти вагою від 1 до 5 мг. Кількість мурашок у колонії може варіюватися від 30 до кількох мільйонів. На планеті налічується близько 10^{16} мурашок, що за масою дорівнюють всьому людству. Поведінка мурах під час транспортування їжі, подолання перешкод, будівництва мурашників та інших завдань часто наближається до теоретично оптимальних рішень. Це явище називається колективним інтелектом — особливим видом кооперативної поведінки, що є стратегією виживання [19]. Як приклад на рисунку 2.2 приведена структура взаємозв'язаних гнізд суперколонії мурашок *Formica lugubris* в Швейцарії. Мережа мурашників близька до мінімального остовного дерева, яке сполучає всі гнізда колонії.

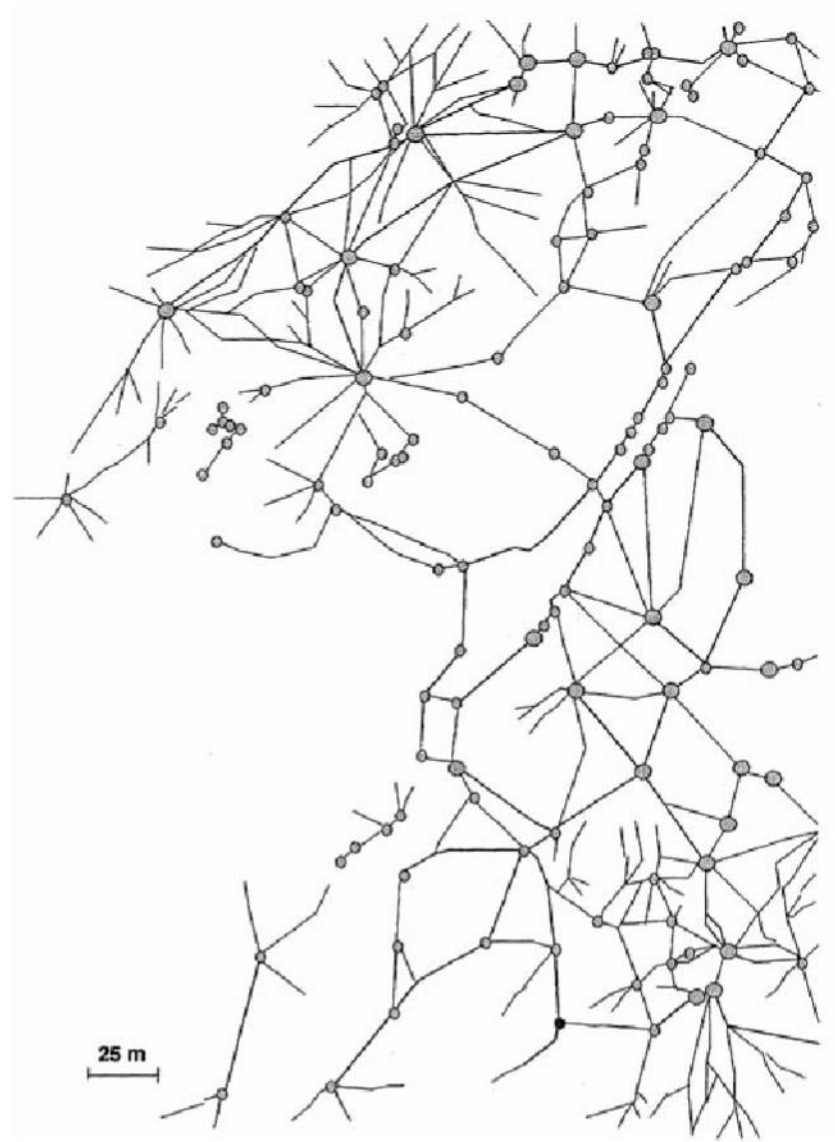


Рисунок 2.2 – Транспортна мережа мурах

Мурахи використовують два способи передавання інформації: прямий – обмін їжею, мандібулярний, візуальний і хімічний контакти; непрямий – стігмергія (stigmergy). Стігмергія – це рознесений в часі тип взаємодії, при якому один суб'єкт певним чином змінює частину навколишнього середовища, а решта суб'єктів використовує цю інформацію пізніше, коли знаходиться поряд. Було визначено 2 форми стігмереж: сематектонічна і сигнальна.

Сематектонічна стігмережа відноситься до комунікацій за допомогою змін у фізичних характеристиках оточення. Прикладами діяльності, яка здійснюється за допомогою сематектонічної стігмережі, є будівництво житла та сортування потомства.

Мурашки мають дуже посередній зір (а деякі види взагалі його не мають) і орієнтуються в основному по запаху. Тому сигнальна стігмережа сприяє комунікації через сигнальний механізм, який реалізується за допомогою ферменту (хімічної сполуки) – феромону. Він представляє собою особливу речовину, яка відкладається як слід під час переміщення мурашок. Феромон з часом випаровується, що дозволяє мурашкам підстроюватися під зміну зовнішнього середовища. Одним з перших типів поведінки, який був досліджений ентомологами, є здатність мурах знаходити найкоротші шляхи між гніздом і джерелом їжі. Мурашки рухаються слідами з феромонів, відкладених іншими мурашками (рисунок 2.3).

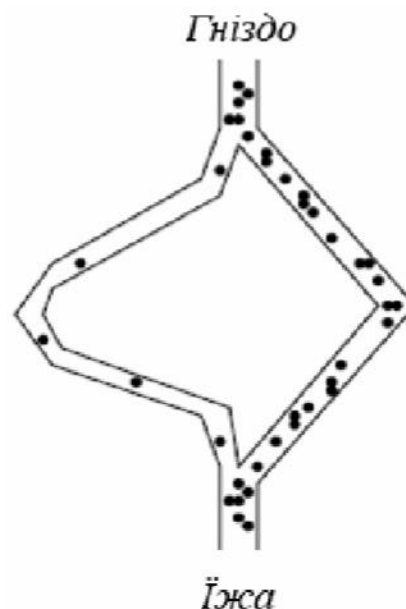


Рисунок 2.3 – Рух мурашок в пошуках їжі

Алгоритм оптимізації на основі наслідування поведінки мурашиної колонії – один з ефективних поліноміальних методів для пошуку наближених рішень різноманітних задач пошуку маршрутів. Ідея створення штучної колонії запропонована бельгійським дослідником М. Доріго [16]. У основі алгоритму лежать принципи самоорганізації в природі колонії мурах. Незважаючи на розрізнену поведінку кожного мурахи, колонія представляє високоорганізовану систему.

Мурахи, взаємодіючи у колонії між собою, здатні розв'язувати складні задачі. Ізольований мураха випадково рухається у просторі пошуку, а мурашка, який знайшов помічений феромоном шлях, може почати рухатися ним (з певною ймовірністю), зміцнивши своїм феромоном. Феромон на подібних шляхах прагне акумулюватися швидше, а, отже, маршрут може бути субоптимальним або оптимальним. Феромон представляє пам'ять системи, крім того, він пов'язаний з наявністю вибраного елемента в “хороших” рішеннях, які раніше були побудовані мурахами.

Перевагами алгоритму є:

- швидкість знаходження рішення;
- збіжність до оптимального рішення;
- можливість використання в динамічних додатках (адаптація до змін навколишнього середовища);
- можливість розпаралелювання.

Серед недоліків:

- складність теоретичного аналізу;
- експериментальність;

– невідомий час збіжності.

2.3 Математична модель

Мурашині алгоритми представляють собою ймовірнісну жадібну метаевристику, в якій імовірності встановлюються, виходячи з інформації про якість рішення, отриманої із попередніх рішень. В основі самоорганізації мурашок лежить їх взаємодія чотирьох складових [7]:

1. Багатократність полягає в одночасному ітераційному пошуку декількома мурашками оптимального маршруту. Кожен мураха виступає як незалежний агент і за одну ітерацію здійснює обхід усіх вершин графа.

2. Як позитивний зворотний зв'язок виступає залишений феромонний слід. Імовірність включення вершини в маршрут мурашки прямопропорційна кількості феромону. З цього виходить випадковість: чим коротший маршрут, тим більша кількість феромону буде залишена на ньому. Використання тільки позитивного зв'язку приводить до передчасної збіжності – випадок, коли всі мурашки рухаються одним і тим же субоптимальним маршрутом.

3. Для уникнення цього використовується від'ємний зворотний зв'язок – випаровування феромону: його час не повинен бути дуже великим, щоб пошук маршруту не звівся до єдиного варіанту, але дуже малий час випаровування приведе до некооперованої поведінки мурашок і значної похибки.

4. Випадковість виникає у кожної мурашки і вносить вклад у перехід із пункту i в пункт j . Для її реалізації необхідні такі складові:

1. Видимість є локальною інформацією, що характеризує бажання мурашки відвідати місто j з міста i . Чим вона вища, тим ближче місто i тим сильніше бажання його відвідати.

Видимість – це величина, яка зворотна відстані L між містами i та j

$$\eta_{ij} = \frac{1}{L_{ij}}. \quad (2.1)$$

2. Пам'ять мурашки (список табу) – це список відвіданих мурашкою міст, заходити в які ще раз не можна. При використанні списку табу мурашка гарантовано не потрапить в одне і те ж місто двічі. Цей список зростає при проходженні маршруту і обнуляється на початку кожної ітерації. Тоді M – список міст, які ще необхідно відвідати мурашці k , який знаходиться в місті i . Це є доповненням до пам'яті мурашки [21].

3. Феромонний слід – підтверджене досвідом бажання відвідати місто j з міста i . На відміну від видимості, феромонний слід більш глобальна і динамічна інформація. Вона оновлюється впродовж роботи алгоритму, відображаючи отриманий мурашками досвід. Кількість феромону на ребрі (i, j) на ітерації t позначимо як τ_{ij} .

Алгоритм складається з низки кроків, які виконуються циклічно до моменту виконання критерію зупинки обчислень:

1. Створення мурашок. Вибір початкових вершин, куди розміщуються мурахи, і спосіб їх розміщення залежить від обмежень за умовами задачі. Також задається первинний рівень феромону τ_0 між містами. Він, зазвичай, ініціалізується невеликим додатним числом / числами. Мурахи можуть розташовуватись в одному або в різних містах, з повтореннями або без них.

Збіжність алгоритму і якість рішення сильно залежать від початкового розташування колонії. Існують наступні стратегії розташування мурах [7]:

- ковдра – реалізація стандартного розміщення мурах, у кожній вершині може перебувати по одній мурашці.
- дробовик – випадковий розподіл мурах вершинами графа, причому необов'язково, щоб чисельність колонії і кількість вершин збігалися.
- фокусування – вся колонія знаходиться в одній вершині.
- блукаюча колонія – у кожен момент часу, тобто на кожній ітерації, вся колонія переміщується у випадково вибрану вершину.

2. Рух мурашок. Безпосередній вибір наступного міста здійснюється з використанням “колеса рулетки”. Вибір міста рівносильний вибору числа з інтервалу $[a; b] = [0, 100]$, де a та b позначають початок/кінець фрагмента кола, який відповідає сектору колеса рулетки (рисунок 2.4).

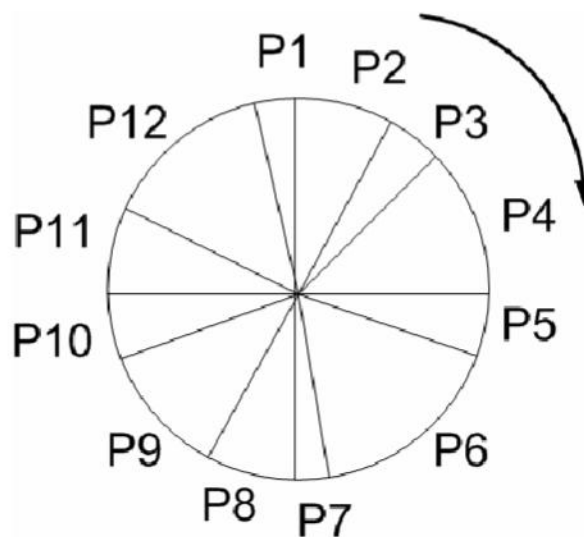


Рисунок 2.4 – Принцип рулетки

Якщо мураха ще не закінчив маршрут, тобто не відвідав всі вершини графа, то ймовірність переходу з вершини i у вершину j за списком міст M

визначається за допомогою ймовірнісного рівняння. Використовується модифіковане “псевдовипадково-пропорційне” правило переходу [5, 22]:

$$j = \begin{cases} \arg \max_{j \in M} \{\eta_{i,j} \cdot \tau_{i,j}\}, & \text{якщо } rand \leq r_0; \\ J, & \text{якщо } rand > r_0, \end{cases} \quad (2.2)$$

де M – список міст транспортної мережі, які ще невідвідані мурахою;

$rand$ – випадкове число, рівномірно розподілене на інтервалі $[0, 1]$;

$r_0 = [0, 1]$ – заданий користувачем параметр;

J – місто знайдене за використанням ймовірнісного переходу по колесу рулетки [23]:

$$p = \begin{cases} \frac{\alpha \cdot \tau_{i,j} + (1 - \alpha) \cdot \eta_{i,j}}{\sum_{j=1}^{length(M)} (\alpha \cdot \tau_{i,j} + (1 - \alpha) \cdot \eta_{i,j})} \cdot 100\%, & \text{якщо } j \in M; \\ 0\%, & \text{якщо } j \notin M, \end{cases} \quad (2.3)$$

де τ_{ij} – рівень феромону для ребра (i, j) на t -ій ітерації.

Даний варіант, порівняно з моделлю М. Доріго [16], не потребує використання параметра β . Крім того, слід зазначити, що мураха подорожує тільки невідвіданими містами (як вказано списком табу), тобто цикли заборонені. Параметр r_0 використовується для регулювання балансу між інтенсифікацією та диверсифікацією простору пошуку рішень: при $rand \leq r_0$ алгоритм експлуатує простір, вибираючи найкращий шлях; у разі $rand > r_0$ алгоритм розширює простір пошуку.

3. Обчислення цільової функції $f(x_k)$. Для кожного з маршрутів розраховується його загальна довжина (цільова функція).

4. Оновлення феромону. Після завершення маршруту мураха залишає деяку кількість феромону на ребрах між вузлами мережі, тобто концентрація феромону коригується [5]:

$$\Delta\tau_{ij,k} = \begin{cases} \tau_0 \cdot \left(1 - \frac{f_t(x_k) - \varepsilon}{\bar{f} - \varepsilon}\right), & \text{якщо } (i, j) \in x_k; \\ 0, & \text{якщо } (i, j) \notin x_k, \end{cases} \quad (2.4)$$

де τ_0 – стартовий рівень феромону на ребрі (i, j) ;

x_k – порядок вершин, пройдений мурашкою k на t -ій ітерації;

ε – нижня межа вартості оптимального рішення, яку знаходять довільним чином, наприклад, жадібним алгоритмом;

$\bar{f}$ – середня довжина z останніх найкращих рішень $f^{global_best}(x_k)$:

$$\bar{f} = \frac{\sum_{i=1}^l f_i^{global_best}(x_k)}{z}. \quad (2.5)$$

Графік функції для лінійного динамічного масштабування зміни феромону наведено на рисунку 2.5. Використання процедури динамічного масштабування дозволяє уникати невеликі досягнення у пошуці, уникаючи фокусування лише навколо хороших рішень на ранніх етапах розв'язання.

Дослідження простору рішень забезпечує випаровування феромону, тобто зменшення його кількості, враховуючи попередні ітерації алгоритму. Дане правило можна записати у формулі

$$\begin{aligned}
 \tau_{ij}(t+1) &= \underbrace{\tau_{ij}(t)}_{\text{новий рівень феромону}} + \underbrace{\Delta\tau_{ij}(t)}_{\text{випаровування}} - q \cdot \tau_{ij}(t) = \\
 &= (1-q) \cdot \tau_{ij}(t) + \Delta\tau_{ij}(t),
 \end{aligned}
 \tag{2.6}$$

$$\Delta\tau_{ij}(t) = \sum_{k=1}^m \Delta\tau_{ij,k}(t),
 \tag{2.7}$$

де m – кількість мурашок у колонії;

$q \in [0,1]$ – коефіцієнт випаровування.

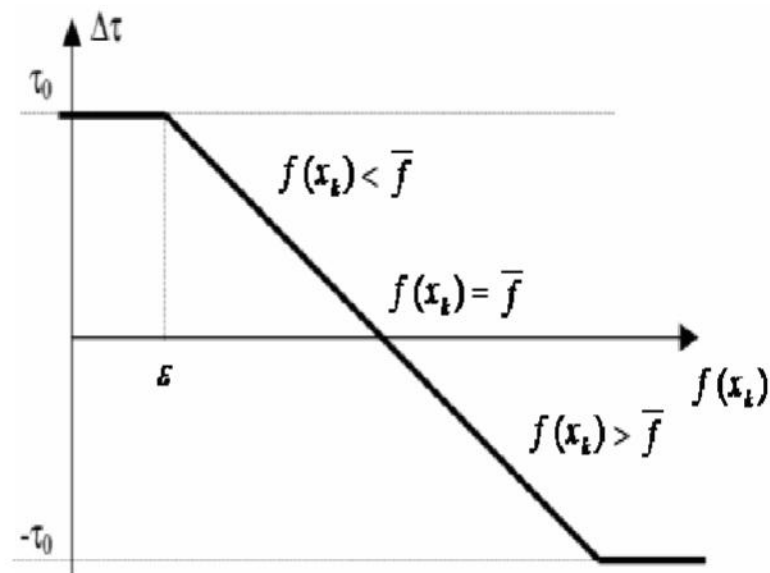


Рисунок 2.5 – Лінійне динамічне масштабування феромону

Очевидно, що при великих значеннях q феромон випаровується швидко, а відповідно малі значення сприяють повільному випаровуванню. Відзначимо, що чим більше випаровується феромон, тим більш випадковим стає пошук рішення. Так при $q = 1$ маємо звичайний випадковий пошук.

5. Додаткові дії. За необхідності можна виконати додаткові дії, використавши, наприклад, ітеративні k -opt-евристики (рисунок 2.6).

Основна ідея полягає в тому, щоб замінити k елементів розв'язку одночасно для зменшення загальної довжини шляху [15]:

а) 2-opt: базова форма евристики, де змінюється лише дві дуги маршруту;

б) 3-opt: розширення 2-opt, де замінюються одночасно три сегменти маршруту;

в) k -opt: загальний випадок, де замінюються k сегментів. Як і для 2-opt та 3-opt, це здійснюється шляхом пошуку всіх можливих перестановок для вибраних сегментів і вибору найбільш вигідної із них.

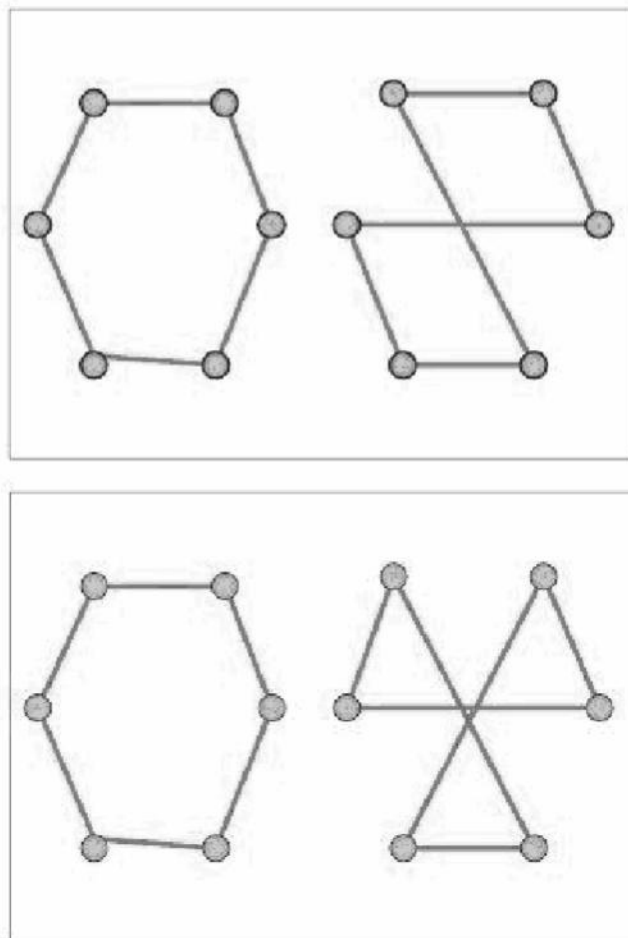


Рисунок 2.6 – Приклад геометричних евристик

На рисунку 2.7 показана поведінка алгоритму. Жирною лінією наведено оптимальний на даний момент маршрут, тонкою – представлено найкращі рішення на кожній з ітерацій. Помітно, що алгоритм продовжує знаходити все нові і нові рішення, тобто для безмежно великої кількості ітерацій (за безмежно великий час) він прямує до глобального оптимуму.

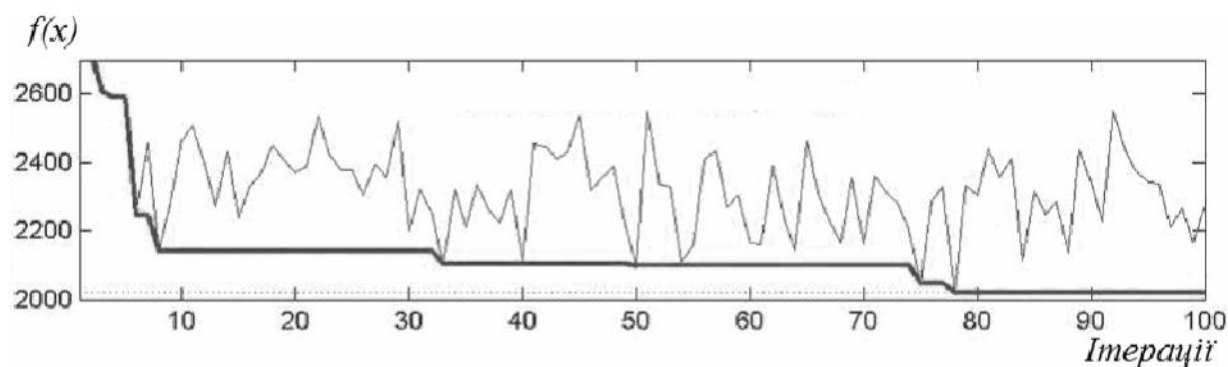


Рисунок 2.7 – Процес пошуку оптимуму

Для покращення часових характеристик в розрахунок вводять елітних мурах, що покращують тільки ребра найкращого маршруту. Інші мурахи починають досліджувати маршрути з ребрами “елітних” маршрутів. На перший погляд здається, що чим більше елітних мурах, тим краще працює алгоритм.

Дійсно, алгоритм із великою кількістю елітних мурашок дуже швидко знаходить “хороші” субоптимальні маршрути, проте після цього надовго залишається в локальному оптимумі, оскільки елітні мурахи сильно підсилюють дані субоптимальні рішення [19].

2.4 Висновки

У даному розділі розглянуто модифікацію алгоритму штучної мурашиної колонії. Під час розробки математичної моделі ройового алгоритму визначено основні принципи функціонування інтелектуальних програмних агентів, які застосовуються для пошуку оптимальних рішень. Подальшим етапом дослідження є оцінка ефективності роботи цієї метаевристики на тестових задачах.

3 РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ТА ЕКСПЕРИМЕНТАЛЬНІ ДОСЛІДЖЕННЯ

3.1 Вибір мови програмування

Python — високорівнева мова програмування, яка підтримує декілька стилів, можна вибрати найбільш підходящий залежно від поставлених задач. Наприклад, якщо необхідно моделювати складну систему, зручніше використовувати об'єктно-орієнтоване програмування, а якщо працюємо з даними, то кращим буде функціональне програмування.

За статистикою дана мова програмування є однією з найпопулярніших у світі та має активну спільноту. Офіційна документація велика і добре структурована, є посібники та довідкові матеріали. Часто члени спільноти самі ініціюють пропозиції щодо покращення Python, що робить дану мову не тільки потужним інструментом для програмістів, але й платформою, що постійно розвивається, готовою адаптуватися до нових викликів і вимог технологічного світу (рисунок 3.1) [24].

Однією з ключових переваг Python є його інтуїтивно зрозумілий синтаксис. Тому Python – відмінний вибір як для новачків, так і для досвідчених розробників. Код на ньому читається легко, що зменшує когнітивне навантаження та прискорює розробку.

Python пропонує велику стандартну бібліотеку, що охоплює різні області програмування, починаючи від веб-розробки та закінчуючи науковими обчисленнями. Найчастіше використовуються бібліотеки типу

NumPy, pandas, SciPy, Matplotlib, Seaborn та Scikit-learn для проведення різних обчислень, аналізу даних, візуалізації та машинного навчання. Ці інструменти разом з IPython і Jupyter Notebook забезпечують потужне середовище для досліджень, що скорочує необхідність використання зовнішніх бібліотек і прискорює розробку додатків.

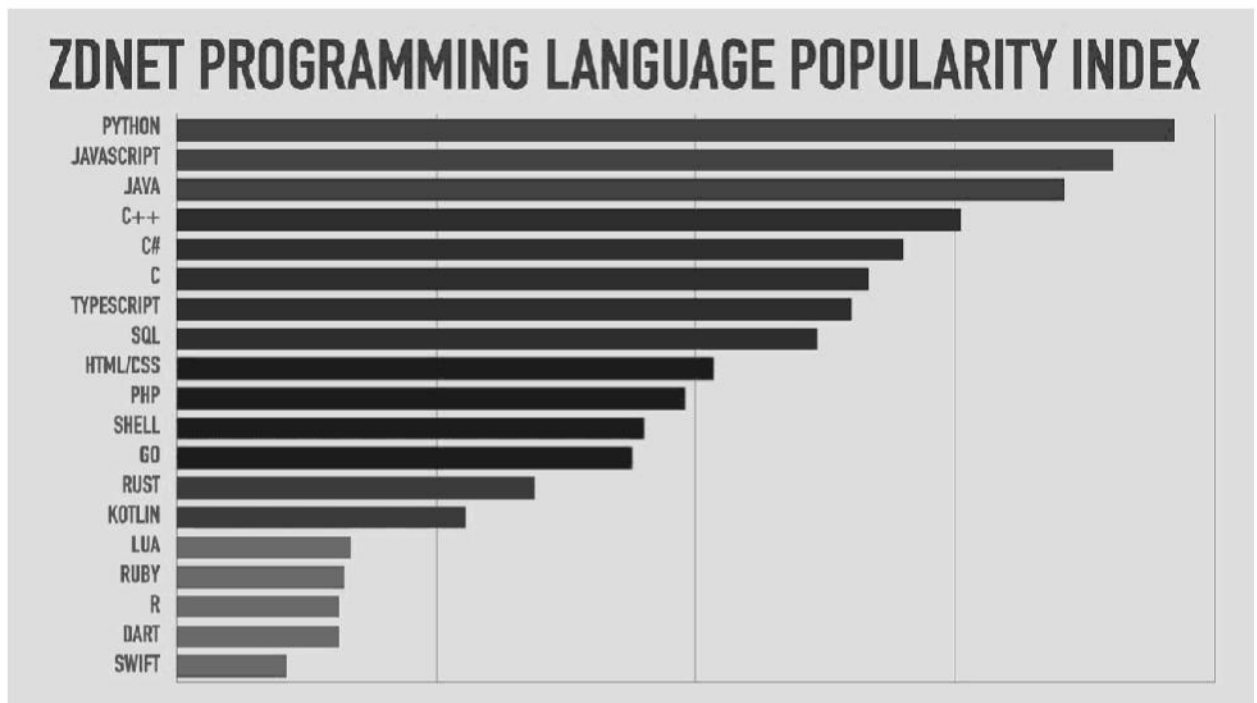


Рисунок 3.1 Рейтинг мов програмування (2024 рік)

Але Python критикують за повільну швидкість порівняно з деякими компільованими мовами, наприклад, C++ або Java. Це пов'язано з його природою і динамічною типізацією даних. Python використовує автоматичне управління пам'яттю, що іноді призводить до менш ефективного використання та складання “сміття”, особливо в додатках з інтенсивними обчисленнями або великим обсягом даних. Це потребує додаткових зусиль для оптимізації використання пам'яті.

Отже, Python також надає широкий спектр вбудованих інструментів, що дозволяє розробникам охоплювати різні аспекти програмування. Його можна використовувати для автоматизації задач, розробки веб-застосунків, аналізу даних. Таким чином, оскільки для нашої програми дуже важлива наявність для розробки вибрана мова програмування Python.

3.2 Опис програмного забезпечення

Розвиток інформаційних технологій та необхідність автоматизації різних логістичних процесів стимулюють створення нових програмних продуктів, що відповідають сучасним вимогам і стандартам. При цьому необхідно працювати з картою, що зводиться до роботи з графовими моделями.

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. Розроблена програма призначена для побудови графів $G(V, E)$, які зчитуються з файлів формату *.tsp та *.data та можуть бути згенеровані користувачем. Програма дозволяє відображати вершини V і ребра E з вказівкою їхніх довжин. За замовчуванням генерується повний неорієнтований граф, але також підтримується робота з неповними графами та графами з орієнтованими ребрами. Довжини ребер обчислюються автоматично на основі координат вершин на площині. Функція виведення графіку має наступний вигляд: *plot(x,*

$y, 'k^*)$, де перші два параметри вказують на координати вершин графа, а третій – це тип маркеру вершин та ребер графа, їхній колір.

Після закінчення роботи виводиться кінцеве значення цільової функції або довжина маршруту комівояжера, а також час пошуку. Користувач має можливість проглядати всі кроки виконання алгоритму у графічному режимі. Програма показує весь процес виконання алгоритму після кожної ітерації у вигляді зображень маршруту (рисунок 3.2).

Довжини ребер E графа G обчислюються, використовуючи евклідову відстань між точками на площині

$$c_{i,j+1} = \sum_{l=1}^{n-1} \sqrt{(x_{i+1} - x_i)^2 + (y_{i+1} - y_i)^2}, \quad (3.1)$$

де x та y – координати міст на площині;

i та $(i+1)$ – опорний та наступний вузли.

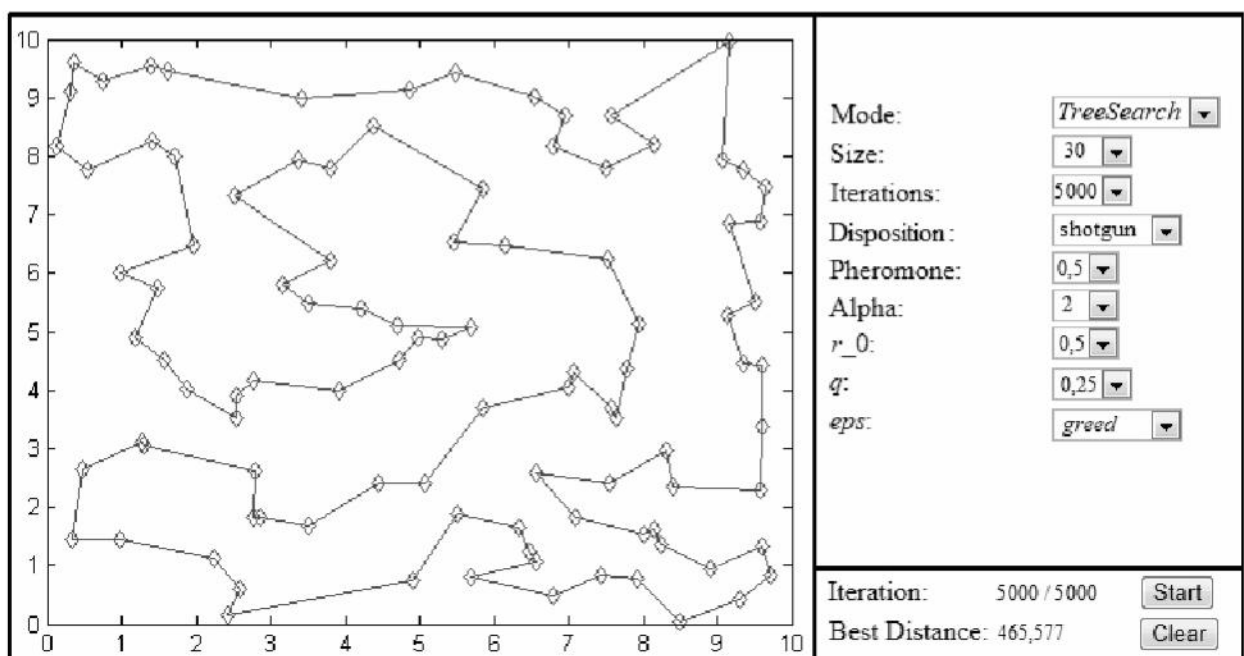


Рисунок 3.2 – Віконний інтерфейс користувача

Для забезпечення демонстрації поведінки алгоритму в різних умовах користувач програми має можливість регулювати деякі параметри, що впливають на роботу, наприклад матрицю відстаней; кількість ітерацій та мурашок; стратегію розташування агентів на графі; стартовий рівень феромону; степінь впливу феромону; параметр; коефіцієнт випаровування феромону; метод пошуку нижньої межі вартості маршруту. Використання наочної демонстрації роботи алгоритму дозволяє досягти більш глибокого розуміння його принципів при розв'язанні оптимізаційної задачі.

3.3 Експерименти

Тестування роботи алгоритму на задачах з бібліотеки TSPLib [25] наведено у таблиці 3.1, де МК – мурашина колонія, МК+ПНПД – розглянутий у роботі алгоритм. Результати довжини маршруту представлені в абстрактних величинах без одиниць вимірювання. В першому стовпці в дужках наведено найкращий відомий результат.

Використано такі параметри алгоритму:

- $N = 5000$ ітерацій;
- $m = 25$ агентів;
- стратегія розміщення мурах – дробовик;
- стартовий рівень феромону $\tau_0 = rand$;
- важливість концентрації феромону $\alpha = 0,7$;
- параметр вибору $r_0 = 0,45$;
- рівень випаровування феромону $q = 0,25$;

- пошук нижньої межі вартості маршруту – простий жадібний алгоритм, тобто на кожному кроці процедура робить локально найкращий вибір, сподіваючись, що отриманий результат буде оптимальним;
- збір статистики виконувався на 30 запусках програми;
- $k = 4$ задачі;
- кількості вузлів $n = \{101, 318, 535, 1577\}$.

Таблиця 3.1 – Результати розв’язання тестових задач

Назва задачі та її розмірність, найкращий результат	Середня довжина маршруту / середній час пошуку в секундах			
	Алгоритм МК+ПНПД		Алгоритм МК	
	Довжина	Час	Довжина	Час
eil101 (629)	639,243	52,643	649,311	43,212
lin318 (42029)	42251,145	206,861	43512,357	178,812
ali535 (202339)	204122,493	344,032	206932,412	293,558
fl1577 (22249)	23295,262	637,735	24577,424	543,012

Результати у графічному вигляді представлені на рисунках 3.3 та 3.4. Маршрути порівнюються за допомогою стовпчикових діаграм (вертикальні стовпці) на рисунку 3.3. Кожен стовпчик представляє окрему задачу. Використано функцію `plt.bar(x, y)`, яка приймає список позицій за осями x

та у. На рисунку 3.4 показано графік, що відображає, як змінюється час пошуку розв’язку залежно від номера задачі.

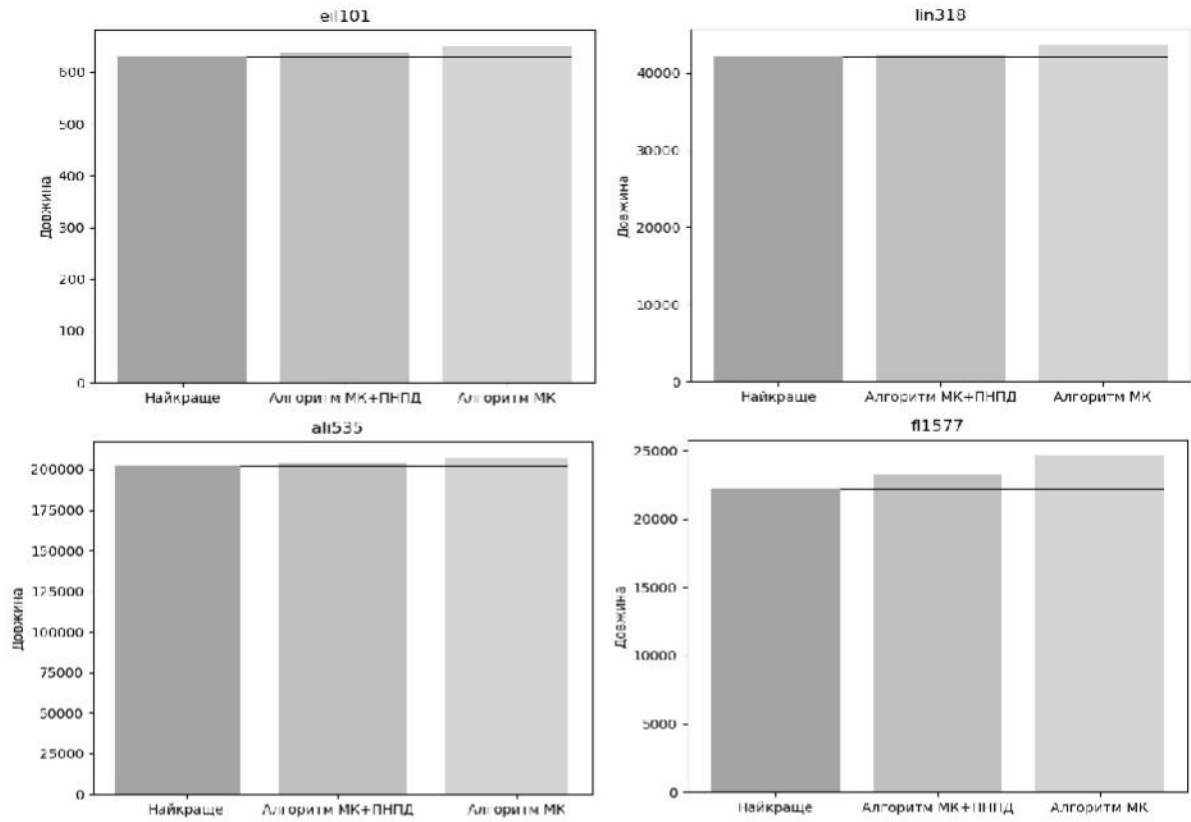


Рисунок 3.3 — Порівняння довжин маршрутів

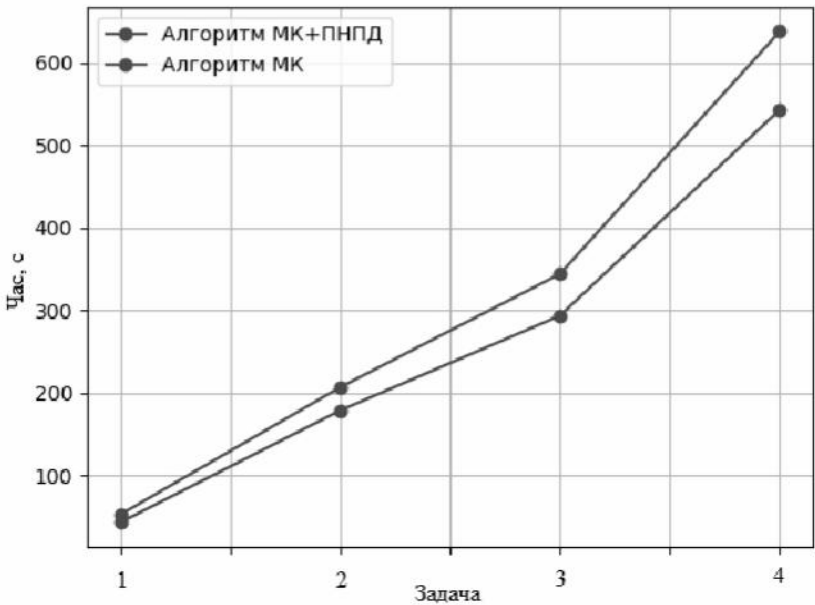


Рисунок 3.4 — Порівняння часу пошуку

У таблиці 3.2 для всіх $k=4$ задач із таблиці 3.1 наведено оціночні параметри (середню відносну похибку δ_{all} та середній час t_{all} пошуку розв'язку у секундах), які розраховано за формулами:

$$\delta_{all} = \frac{1}{k} \cdot \sum_{i=1}^k \frac{|f_i - f_{opt_i}|}{f_{opt_i}} \cdot 100\%, \quad (3.2)$$

$$t_{all} = \frac{1}{k} \cdot \sum_{i=1}^k t_i, \quad (3.3)$$

де f_i – середня довжина маршруту для i -ої задачі;

f_{opt} – найкращий відомий розв'язок;

t_i – середній час розв'язання i -ої задачі.

Таблиця 3.2 – Порівняння ефективності роботи алгоритмів
на тестових задачах

#	Алгоритм МК+ПНПД	Алгоритм МК	Результат порівняння
Похибка, %	1,935	4,873	-2,983
Час, с	310,318	264,649	+45,669

Складність розглянутої метаевристики оцінимо, знайшовши експериментальну залежність часу виконання від кількості вузлів задачі комівояжера (рисунок 3.5). Дослідження проведено на:

- згенерованій задачі;
- без розпаралелювання обчислень;
- $N = 20$ ітерацій;

- $m=10$ агентів;
- інші параметри аналогічні попереднім;
- збір статистики виконувався на кількості вузлів $n = \{50, \dots, 1000\}$.

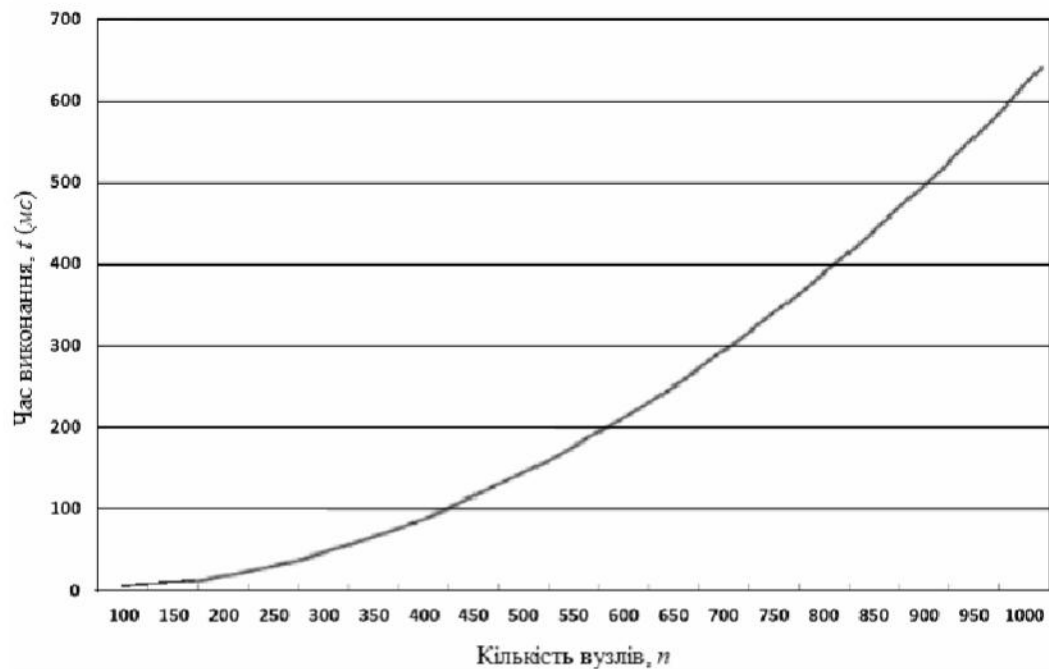


Рисунок 3.5 – Оцінювання часу роботи алгоритму

Залежність часу виконання обчислень від розмірності задачі має вигляд параболи. Отже, час розв'язання задачі у загальному випадку зростає за закономірністю $O(n^2)$, тобто має квадратичний характер.

Далі розглянемо алгоритм-переможець на комплексній задачі перевезення товарів циклічним маршрутом із обмеженням на вантажність засобу перевезення. Параметри алгоритму аналогічні попереднім; 30 запусків програми по 20 хв кожний. Цільова функція задачі максимізує дохід, враховуючи цінність завантажених товарів (у кожному пункті різні їх кількість та цінність) із вирахуванням вартості оренди засобу перевезення. Найкращі відомі розв'язки задачі наведено в таблиці 3.3 [5].

Таблиця 3.3 – Відомі результати

#	Цінність товарів	Довжина маршруту	Час	Дохід
Відомий результат eil51	53368	467,00	652,11	11763,38
Відомий результат dsj1000	595519	19290271,00	26699765,02	61523,70

Результати виконаних експериментів представлено у таблиці 3.4 для двох задач eil51 та dsj1000. Наведено ключові показники, які демонструють ефективність розв'язання, зокрема загальну довжину маршруту та час його проходження, цінність товарів у засобі перевезення та значення цільової функції (дохід).

Таблиця 3.4 – Результати роботи алгоритму на комплексній задачі

#	Цінність товарів	Довжина маршруту	Час	Дохід
Задача eil51	52145	459,00	659,35	10078,47
Задача dsj1000	587164	18750975,00	26376923,01	59625,54

Алгоритм знаходить менші довжини маршрутів та час руху, але значення доходу теж менше (таблиця 3.5). Отже, алгоритм намагається зменшити час і відстань, тобто він вибирає маршрути, що проходять через менш вигідні пункти, де менша цінність предметів.

Таблиця 3.5 – Порівняння результатів із відомими розв'язками

#	Цінність товарів	Довжина маршруту	Час	Дохід
Порівняння eil51	-1223	-8,00	+7,24	-1684,91
Порівняння dsj1000	-8355	-539296,00	-322842,01	-1898,16

Можливими шляхами його покращення для даної задачі є:

- використання багатокритеріальної цільової функції;
- орієнтація моделі на вартість замість довжин маршрутів та часу їх проходження;
- застосування локальної оптимізації та геометричних евристик;
- кластеризація пунктів за цінністю товарів;
- зняття обмеження на час роботи алгоритму.

Загалом таким чином можна підвищити ефективність розв'язання задачі логістики. Також необхідно оптимізувати код програмного забезпечення, що дозволить зменшити ресурси, необхідні для виконання алгоритмів, що важливо при роботі з великими обсягами даних.

3.4 Висновки

У розділі пояснено вибір мови програмування для розробки програмного додатку. Проведено його тестування на різних складних задачах. Розглянута метаевристика дає точніші результати (середня відносна похибка на 2,983% менша), але при цьому працює повільніше (час виконання зріс на 45,669 с) для задач комівояжера; для задачі перевезення товарів із обмеженням теж отримано хороший результат за довжиною маршруту.

ВИСНОВКИ

У сучасному світі задачі оптимізації займають важливе місце в багатьох галузях науки та техніки, оскільки вони сприяють підвищенню ефективності виконання низки процесів. У даній роботі розглядаються логістичні задачі оптимізації. Більшість із них належать до класу NP. Оскільки повний перебір можливих рішень є надзвичайно витратним, це змушує дослідників шукати компромісні рішення для забезпечення балансу між швидкістю та точністю пошуку розв'язку. Одним із таких підходів є використання низки спеціальних метаевристичних алгоритмів розв'язання подібних задач.

У ході дослідження представлено основні принципи та математична модель метаевристичного алгоритму, що використовує наближений недетермінований пошук по дереву з наслідуванням поведінки колонії мурах. Розроблено програмне забезпечення та проведено експериментальне дослідження ефективності роботи алгоритму на тестових задачах. Встановлено, що розглянутий алгоритм можна застосовувати, враховуючи його здатність знаходити оптимальні рішення навіть для складних задач.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Кавун С.В., Ревак І.О. Застосування теорії графів у задачах комунікаційного менеджменту. *Науковий вісник Львівського державного університету внутрішніх справ*. 2015. № 2. pp. 225–240.
2. Optimization of Swarm Robotics Algorithms / R.P. Kukharchuk, T.A. Vakaliuk, O.V. Zaika, A.V. Riabko. *Radio Electronics, Computer Science, Control*. 2022. Vol. 3. pp. 66–76.
3. Blum C., Roli A. Metaheuristics in Combinatorial Optimization: Overview and Conceptual Comparison. *ACM Computing Surveys*. 2003. Vol. 35. pp. 268–308.
4. Simon D. Evolutionary Optimization Algorithms: Biologically Inspired and Population-Based Approaches to Computer Intelligence. John Wiley & Sons, 2013. 776 p.
5. Stützle T., Maniezzo V., Boschetti M.A. Matheuristics. Algorithms and Implementations. Springer, 2021. 212 p.
6. Li P., Wei L., Wu D. An Intelligently Enhanced Ant Colony Optimization Algorithm for Global Path Planning of Mobile Robots in Engineering Applications. *Sensors*. 2025. Vol. 25. 32 p. URL: <https://www.mdpi.com/1424-8220/25/5/1326>. (дата зверення 26.04.2025)
7. Shtovba S. Ant Algorithms: Theory and Applications. *Programming and Computer Software*. 2005. Vol. 31 (4). pp. 167–178.
8. Желдак Т.А. Застосування методу моделювання колонії мурах до розв'язання комбінаторних задач планування виконання замовлень

металургійними підприємствами. *Математичні машини і системи*. К., 2013. № 4. С. 95–106.

9. Адаптивні ройові алгоритми оптимізації / К.В. Витвицька, Ю.Ю. Іванов, С.Г. Кривогубченко та інші. *LIV Всеукраїнська науково-технічна конференція підрозділів Вінницького національного технічного університету*. Вінниця, 24-27.03.2025. URL: <https://conferences.vntu.edu.ua/index.php/all-fksa/all-fksa-2025/paper/view/23529/19765>. (дата звернення 21.03.2025)

10. Гуляницький Л.Ф., Мулеса О.Ю. Прикладні методи комбінаторної оптимізації: навчальний посібник. К.: Видавничо-поліграфічний центр “Київський університет”, 2016. 142 с.

11. Waligora L. Application of Hamilton’s Graph Theory in New Technologies. *World Scientific News*. 2017. Vol. 89. pp. 71–81.

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

13. Maniezzo V., Carbonaro A. An ANTS Heuristic for the Frequency Assignment Problem. *Future Generation Computer Systems*. 2000. Vol. 16 (8). pp. 927–935.

14. Maniezzo V., Carbonaro A., Hildmann H. An ANTS Heuristic for the Long-Term Car Pooling Problem. *New Optimization Techniques in Engineering. Studies in Fuzziness and Soft Computing*. 2004. Vol. 141. pp. 411–430.

15. 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>. (23.04.2025)

16. Dorigo M., Stützle T. Ant Colony Optimization: Overview and Recent Advances. *Handbook of Metaheuristics*. 2010. pp. 311–351.
17. Dorigo M., Gambardella L.M. Ant Colony System: A Cooperative Learning Approach to the Traveling Salesman Problem. *IEEE Transactions of Evolutionary Computing*. 1997. Vol. 1 (1). pp. 53–66.
18. A Novel Ant Colony Optimization Based on Game for Traveling Salesman Problem / S. Liu, K. Yang, X. You, H. Pan. *Applications of Intelligence*. 2020. № 50. pp. 4529–4542.
19. Муляревич О.В. Переваги застосування колективної поведінки агентів для розв'язку задачі комівояжера динамічного характеру. *IX міжнародна науково-технічна конференція "Методи і засоби вимірювань фізичних величин"*. Львів, 2012. С. 165–168.
20. Wang Y., Han Z. Ant Colony Optimization for Traveling Salesman Problem Based on Parameters Optimization. *Applications of Soft Computing*. 2021. № 107. 11 p. URL: <https://www.sciencedirect.com/science/article/abs/pii/S1568494621003628?via%3Dihub>. (дата звернення 23.03.2025)
21. An Adaptive Ant Colony Optimization for Solving Large-Scale Traveling Salesman Problem / K. Tang, X.-F. Wei, Y.-H. Jiang et al. *Mathematics*. 2023. Vol. 11. 26 p. URL: <https://www.mdpi.com/2227-7390/11/21/4439>. (дата звернення 23.04.2025)
22. Shahadat A.S.B., Akhand M., Kamal M.A.S. Visibility Adaptation in Ant Colony Optimization for Solving Traveling Salesman Problem. *Mathematics*. 2022. Vol. 10. 24 p. URL: <https://www.mdpi.com/2227-7390/10/14/2448>. (дата звернення 05.04.2025)
23. A Scalable Model for Capacitated Vehicle Routing Problem with Pickup and Delivery Under Dynamic Constraints Using Adaptive Heuristic-Based Ant

Colony Optimization / R.I. Muslem, M.K.M. Nasution, S. Sutarman, S. Suherman. *Eastern-European Journal of Enterprise Technologies*. 2025. Vol. 133. pp. 57–65. URL: <https://journals.uran.ua/eejet/article/view/319733/313583>. (дата звернення 26.04.2025)

24. The most popular programming languages in 2024 (and what that even means). URL: <https://www.zdnet.com/article/the-most-popular-programming-languages-in-2024-and-what-that-even-means/>. (дата звернення 04.04.25)

25. Traveling Salesman Problem. University of Waterloo. URL: <http://www.math.uwaterloo.ca/tsp/index.html>. (дата звернення 02.04.2025)

ДОДАТКИ

Додаток А
(обов'язковий)
Технічне завдання

60

ЗАТВЕРДЖУЮ

завідувач кафедри АІТ

д.т.н., проф. Бісікало О. В.

«23» 03 2025 р.

ТЕХНІЧНЕ ЗАВДАННЯ

на бакалаврську кваліфікаційну роботу

«Дослідження модифікації алгоритму штучної мурашиної колонії
на основі приблизного недетермінованого пошуку по дереву»

08-02.БКР.003.02.000 ТЗ

Керівник роботи:

к.т.н., доц. каф. АІТ

Севастьянов В. М.

«24» 03 2025 р.

Виконавець:

ст. гр. ІІСТ-216

Витвицька К. В.

«24» 03 2025 р.

Вінниця ВНТУ – 2025

1. Назва та галузь застосування

Робота спрямована на дослідження модифікації алгоритму штучної мурашиної колонії. Її можна використовувати для розв'язання різноманітних задач дискретної оптимізації, особливо пошуку шляхів.

2. Підстави для розробки

Розробку здійснювати на підставі наказу по університету №97 від 20.03.2025 та завдання до бакалаврської кваліфікаційної роботи, складеного та затвердженого кафедрою "Автоматизації та інтелектуальних інформаційних технологій".

3. Мета та призначення розробки

Метою роботи є підвищення функціональних можливостей програмного забезпечення для експериментального дослідження розв'язання задач логістики за рахунок використання алгоритму штучної мурашиної колонії на основі приблизного недетермінованого пошуку по дереву.

4. Етапи виконання робіт

Виконавцем перерахованих у даному розділі етапів є студентка групи ІСТ-216 Витвицька Катерина Володимирівна факультету Інтелектуальних інформаційних технологій та автоматизації Вінницького національного технічного університету, а замовником є кафедра автоматизації та інтелектуальних інформаційних технологій.

№	Назва та зміст етапу	Термін виконання	
		початок	закінчення
1	Аналіз предметної області	21.03.2025	04.04.2025
2	Розробка математичної моделі алгоритму оптимізації	05.04.2025	26.04.2025
3	Розробка програмного забезпечення та експериментальні дослідження	27.04.2025	16.05.2025
4	Оформлення пояснювальної записки і графічного матеріалу	17.05.2025	27.05.2025

5. Джерела розробки

1. Maniezzo V., Boschetti M.A., Stützle T. *Matheuristics. Algorithms and Implementations*. Springer, 2021. 212 p.
2. A Scalable Model for Capacitated Vehicle Routing Problem with Pickup and Delivery Under Dynamic Constraints Using Adaptive Heuristic-Based Ant Colony Optimization / R.I. Muslem, M.K.M. Nasution, S. Sutarman, S. Suherman. *Eastern-European Journal of Enterprise Technologies*. 2025. Vol. 133. pp. 57–65.
3. Li P., Wei L., Wu D. An Intelligently Enhanced Ant Colony Optimization Algorithm for Global Path Planning of Mobile Robots in Engineering Applications. *Sensors*. 2025. Vol. 25. 32 p.

6. Показники призначення

Основні технічні вимоги та мінімальні системні вимоги до програми: операційна система *Windows* 7, 8 або 10; підключення клавіатури та миші; процесор *Core i5 (Ryzen 5)* або кращий; оперативна пам'ять – не менше 8 ГБ; відеопам'ять – 512 Mb; жорсткий диск – 500 ГБ і більше.

Вхідні дані: кількість ітерацій; розмір популяції; коефіцієнти алгоритму; координати пунктів; значення швидкості та вантажності засобу; цінність товарів; рента засобу; текстовий файл із даними задачі. Результати роботи програми: відповідна перестановка пунктів.

Студентка гр. ІІСТ-216

(підпис)

Витвицька К. В.
(ініціали та прізвище)

Додаток Б
(обов'язковий)

ІЛЮСТРАТИВНА ЧАСТИНА

ДОСЛІДЖЕННЯ МОДИФІКАЦІЇ АЛГОРИТМУ ШТУЧНОЇ
МУРАШИНОЇ КОЛОНІЇ НА ОСНОВІ ПРИБЛИЗНОГО
НЕДЕТЕРМІНОВАНОГО ПОШУКУ ПО ДЕРЕВУ



Рисунок Б.1 – Схема програми

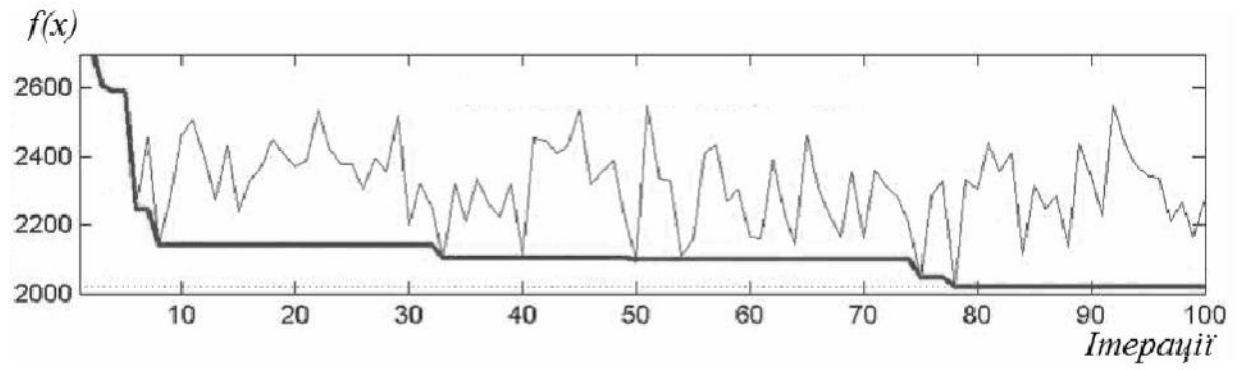


Рисунок Б.2 – Графічне відображення кривої збіжності алгоритму

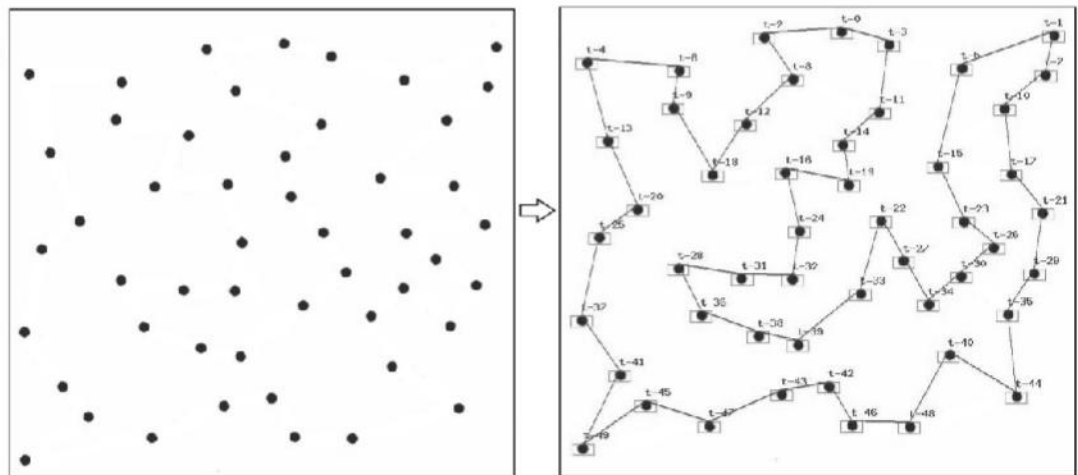


Рисунок Б.3 – Приклад пошуку маршруту

Додаток В

(обов'язковий)

Лістинг програми

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
from dataclasses import dataclass, field
import random
from typing import List, Tuple
import networkx as nx

from aco_routing.ant import Ant
from aco_routing.graph_api import GraphApi

@dataclass
class ACO:
    graph: nx.DiGraph
    ant_max_steps: int
    num_iterations: int
    ant_random_spawn: bool = True
    evaporation_rate: float = 0.1
    alpha: float = 0.7
    r0: float = 0.45
    search_ants: List[Ant] = field(default_factory=list)

    def __post_init__(self):
        self.graph_api = GraphApi(self.graph, self.evaporation_rate)
        for edge in self.graph.edges:
            self.graph_api.set_edge_pheromones(edge[0], edge[1], 1.0)

    def _deploy_forward_search_ants(self) -> None:
        for ant in self.search_ants:
            for _ in range(self.ant_max_steps):
                if ant.reached_destination():
                    ant.is_fit = True
                    break
            ant.take_step()

    def _deploy_backward_search_ants(self) -> None:
        for ant in self.search_ants:
            if ant.is_fit:
                ant.deposit_pheromones_on_path()
```

```

def _deploy_search_ants(
    self,
    source: str,
    destination: str,
    num_ants: int,
) -> None:

    for _ in range(self.num_iterations):
        self.search_ants.clear()

        for _ in range(num_ants):
            spawn_point = (
                random.choice(self.graph_api.get_all_nodes())
                if self.ant_random_spawn
                else source
            )

            ant = Ant(
                self.graph_api,
                spawn_point,
                destination,
                alpha=self.alpha,
                beta=self.beta,
            )
            self.search_ants.append(ant)

        self._deploy_forward_search_ants()
        self._deploy_backward_search_ants()

def _deploy_solution_ant(self, source: str, destination: str) -> Ant:
    ant = Ant(
        self.graph_api,
        source,
        destination,
        is_solution_ant=True,
    )
    while not ant.reached_destination():
        ant.take_step()
    return ant

def find_shortest_path(
    self,
    source: str,
    destination: str,
    num_ants: int,
) -> Tuple[List[str], float]:

```



```

        self._deploy_search_ants(
            source,
            destination,
            num_ants,
        )
        solution_ant = self._deploy_solution_ant(source, destination)
        return solution_ant.path, solution_ant.path_cost

from dataclasses import dataclass
from typing import List
import networkx as nx
import matplotlib.pyplot as plt

@dataclass
class GraphApi:
    graph: nx.DiGraph
    evaporation_rate: float

    def set_edge_pheromones(self, u: str, v: str, pheromone_value: float) ->
None:
        self.graph[u][v]["pheromones"] = pheromone_value

    def get_edge_pheromones(self, u: str, v: str) -> float:
        return self.graph[u][v]["pheromones"]

    def deposit_pheromones(self, u: str, v: str, pheromone_amount: float) -
> None:
        self.graph[u][v]["pheromones"] += max(
            (1 - self.evaporation_rate) * pheromone_amount, 1e-13
        )

    def get_edge_cost(self, u: str, v: str) -> float:
        return self.graph[u][v]["cost"]

    def get_all_nodes(self) -> List[str]:
        return list(self.graph.nodes)

    def get_neighbors(self, node: str) -> List[str]:
        return list(self.graph.neighbors(node))

    def visualize_graph(self, shortest_path: List[str]) -> None:
        for edge in self.graph.edges:
            source, destination = edge[0], edge[1]
            self.graph[source][destination]["pheromones"] = round(
                self.graph[source][destination]["pheromones"]
            )

```


Додаток Г
(обов'язковий)
Протокол перевірки БКР

ПРОТОКОЛ ПЕРЕВІРКИ КВАЛІФІКАЦІЙНОЇ РОБОТИ

Назва роботи: Дослідження модифікації алгоритму штучної мурашиної колонії на основі приблизного недетермінованого пошуку по дереву.

Тип роботи: бакалаврська кваліфікаційна робота (БКР)
(бакалаврська кваліфікаційна робота / магістерська кваліфікаційна робота)

Підрозділ: кафедра автоматизації та інтелектуальних інформаційних технологій, факультет інтелектуальних інформаційних технологій та автоматизації, ІІСТ-216

(кафедра, факультет, навчальна група)

Коефіцієнт подібності текстових запозичень, виявлених у роботі системою StrikePlagiarism 13.96 %

Висновок щодо перевірки кваліфікаційної роботи (відмітити потрібне):

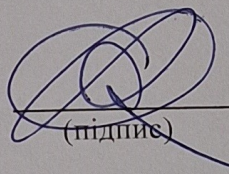
- ☒ Запозичення, виявлені у роботі, є законними і не містять ознак плагіату, фабрикації, фальсифікації. Роботу прийняти до захисту.
- ☐ У роботі не виявлено ознак плагіату, фабрикації, фальсифікації, але надмірна кількість текстових запозичень та/або наявність типових розрахунків не дозволяють прийняти рішення про оригінальність та самостійність її виконання. Роботу направити на доопрацювання.
- ☐ У роботі виявлено ознаки плагіату та/або текстових маніпуляцій як спроб укриття плагіату, фабрикації, фальсифікації, що суперечить вимогам законодавства та нормам академічної доброчесності. Робота до захисту не приймається.

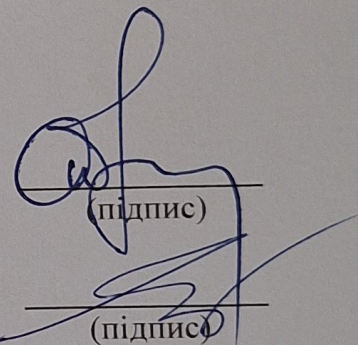
Експертна комісія:

Зав. каф. АІТ, д.т.н., професор Бісікало О. В.
(прізвище, ініціали, посада)

Секретар кафедри АІТ Янчук Л. М.
(прізвище, ініціали, посада)

Особа, відповідальна за перевірку

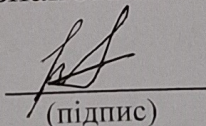

(підпис)


(підпис)

Овчинников К. В.
(прізвище, ініціали)

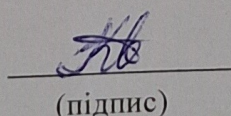
З висновком експертної комісії ознайомлена.

Керівник


(підпис)

доцент Севастьянов В. М.
(прізвище, ініціали, посада)

Здобувач


(підпис)

Витвицька К. В.
(прізвище, ініціали)