Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет Інформаційно-аналітичних технологій та менеджменту
(повна назва)
Кафедра Інформатики
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти другий (магістерський)

ДОСЛІДЖЕННЯ МЕТОДІВ ОПТИМІЗАЦІЇ МАРШРУТІВ У ЛОГІСТИЧНИХ СИСТЕМАХ
(тема)

Виконав:
студент 2 курсу, групи ІНФм-22-2
Євтушенко В.Р.
(прізвище, ініціали)

Спеціальності 122 Комп'ютерні науки
(код і повна назва спеціальності)

Тип програми освітньо-професійна

Освітня програма Інформатика
(повна назва освітньої програми)

Керівник доц. Творошенко І.С.
(посада, прізвище, ініціали)

Допускається до захисту

Зав. кафедри _____ Кобилін О.А.
(підпис) (прізвище, ініціали)

2024 р.

Харківський національний університет радіоелектроніки

Факультет Інформаційно-аналітичних технологій та менеджменту
(повна назва)

Кафедра Інформатики
(повна назва)

Рівень вищої освіти другий (магістерський)

Спеціальність 122 Комп'ютерні науки
(код і повна назва)

Тип програми освітньо-професійна

Освітня програма Інформатика
(повна назва освітньої програми)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
«____»_____ 2024 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

студентові Євтушенку Владиславу Руслановичу
(прізвище, ім'я, по батькові)

1. Тема роботи Дослідження методів оптимізації маршрутів у логістичних системах

затверджена наказом по університету від 3 листопада 2023 року № 1280Ст

2. Термін подання студентом роботи до екзаменаційної комісії 23 грудня 2023 р.

3. Вихідні дані до роботи науково-методична література, матеріали конференцій, дані інтернет-мережі, математичні моделі алгоритмів оптимізації, теоретичні відомості особливостей застосування методів оптимізації, інспірованих тваринними технологіями, для оптимізації маршрутів у логістичних системах, використані програмні засоби: мова програмування Python, середовище розроблення Google Colab.

4. Перелік питань, що потрібно опрацювати в роботі
1. Огляд методів оптимізації маршрутів у логістичних системах.
2. Вивчення математичних моделей алгоритмів ACO, ABC та CS.
3. Програмна реалізація методів оптимізації маршрутів у логістичних системах.
4. Порівняльний аналіз ефективності розглянутих методів.

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (п.5 включається до завдання за рішенням випускової кафедри) актуальність проблеми, об'єкт та мета дослідження, постановка задачі, аналіз предметної області, вихідні дані для дослідження, етапи реалізації поставленої задачі, аналіз результатів.

6. Консультанти розділів роботи (п.6 включається до завдання за наявності консультантів згідно з наказом, зазначеним у п.1)

Найменування розділу	Консультант (посада, прізвище, ім'я, по батькові)	Позначка консультанта про виконання розділу	
		підпис	дата

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів роботи	Терміни виконання етапів роботи	Примітка
1	Отримання завдання на кваліфікаційну роботу	03.11.2023	
2	Аналіз завдання, підбір літератури	03.11.23-05.11.23	
3	Аналіз літератури з досліджуваної проблеми	05.11.23-07.11.23	
4	Розробка математичних моделей	07.11.23-09.11.23	
5	Аналіз технічних засобів	09.11.23-11.11.23	
6	Програмна реалізація	11.11.23-17.11.23	
7	Оформлення пояснювальної записки	17.12.23-25.11.23	
8	Перевірка на плагіат	05.12.2023	
9	Рецензування	10.12.2023	
10	Підготовка презентації та доповіді	15.12.2023	
11	Занесення роботи в електронний архів	02.01.2024	
12	Попередній захист кваліфікаційної роботи	02.01.2024	

Дата видачі завдання 3 листопада 2023 р.

Студент _____
(підпис)

Керівник роботи _____ доц. Творошенко І.С.
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ/ABSTRACT

Пояснювальна записка до кваліфікаційної роботи: 76 с., 8 табл., 23 рис., 41 джерело.

ЛОГІСТИЧНА СИСТЕМА, ОПТИМІЗАЦІЯ ТРАНСПОРТУВАННЯ, ТВАРИННІ ТЕХНОЛОГІЇ, МЕТОД МУРАШИНОЇ КОЛОНІЇ, МЕТОД ШТУЧНОЇ БДЖОЛИНОЇ КОЛОНІЇ, МЕТОД ЗОЗУЛИНОГО ПОШУКУ.

Об’єктом дослідження є оптимізація маршрутів у логістичних системах.

Метою дослідження є вивчення та порівняння методів оптимізації маршрутів у логістичних системах.

Проведено дослідження методів мурашиної колонії, бджолиної колонії, зозулиного пошуку, здійснено порівняльний аналіз їх ефективності в задачах оптимізації маршрутів у логістичних системах.

У результаті дослідження здійснена програмна реалізація системи для пошуку оптимальних шляхів в логістичних системах.

LOGISTICS SYSTEM, TRANSPORTATION OPTIMIZATION, ANIMAL TECHNOLOGIES, ANT COLONY OPTIMIZATION, ARTIFICIAL BEE COLONY OPTIMIZATION, CUCKOO SEARCH ALGORITHM.

The object of research is the optimization of routes in logistics systems.

The research aims to study and compare methods of route optimization in logistics systems.

Research on the methods of ant colony, bee colony, and cuckoo search was carried out, and a comparative analysis of their effectiveness in the tasks of route optimization in logistics systems was carried out.

As a result of the research, the software implementation of the system for finding optimal paths in logistics systems was carried out.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

VRP – Vehicle Routing Problem (маршрутизація транспортного засобу)

UPS – United Parcel Service (американська компанія)

ТЗ – транспортний засіб

TSP – Travelling Salesman Problem (задача комівояжера)

NP – Non-deterministic Polynomial time

GA – Genetic Algorithm (генетичний алгоритм)

ACO – Ant Colony Optimization (алгоритм мурашиних колоній)

ABC – Artificial Bee Colony optimization (алгоритм бджолиного рою)

CS – Cuckoo Search (зозулин пошук)

ПНП – персональна найкраща позиція

ГНП – глобальна найкраща позиція

ВСТУП

У сучасному швидкоплинному світі ефективне управління логістикою стало стрижнем успішного ведення бізнесу. Оптимізація маршрутів транспортування вантажів відіграє ключову роль у зниженні витрат, мінімізації впливу на навколишнє середовище та забезпеченні своєчасних поставок. Оптимізація маршрутів, як основний компонент логістичних систем, привернула значну увагу як дослідників, так і практиків. Ця кваліфікаційна робота розпочинає комплексне дослідження методів оптимізації маршруту, зосереджуючись на порівняльному аналізі трьох видатних природних методів: алгоритму мурашиної колонії, алгоритму бджолиного рою та алгоритму зозулі.

Ефективний розподіл товарів має важливе значення для підприємств, які прагнуть отримати конкурентну перевагу на світовому ринку. Традиційно логістичними операціями керували за допомогою евристичних підходів, які часто призводили до неоптимальних рішень, що призводило до збільшення операційних витрат і втрати ресурсів. Створені природою алгоритми стали потужними інструментами для вирішення складних задач оптимізації. Ці алгоритми черпають натхнення з поведінки самоорганізації, яка спостерігається в природних системах, пропонуючи новий підхід до вирішення проблем оптимізації маршрутів у логістиці.

Досліджувані алгоритми отримали визнання за їхній потенціал щодо надання майже оптимальних рішень проблеми маршрутизації транспортного засобу та її різноманітних розширень. Кожен із цих алгоритмів містить унікальний набір функцій і механізмів, що робить їх придатними кандидатами для порівняльного аналізу.

Мурашиний алгоритм, натхненний поведінкою мурах у пошуках їжі, використовує сліди феромонів для пошуку оптимальних маршрутів. Алгоритм бджолиного рою, натхненний поведінкою бджіл у колективному пошуку їжі, використовує принципи ройового інтелекту для ефективного

дослідження та використання пошукових просторів. З іншого боку, алгоритм зозулі, натхненний паразитичною поведінкою птахів зозулі, представляє концепцію польоту Леві для глобальних досліджень.

Це дослідження має на меті пролити світло на сильні та слабкі сторони цих алгоритмів у контексті оптимізації маршруту. Проводячи порівняльний аналіз, прагнемо визначити, який алгоритм найкраще підходить для різних сценаріїв логістики. Цей аналіз враховуватиме такі фактори, як обчислювальна ефективність, якість рішення, масштабованість і адаптованість до реальних логістичних обмежень.

У даному дослідженні описано теоретичні основи кожного алгоритму, окреслено методологію для проведення порівняльного дослідження, представлено експериментальні результати та запропоновано уявлення про практичне значення висновків. Зрештою, це дослідження прагне зробити свій внесок у покращення методів оптимізації маршрутів у логістичних системах, сприяючи більшій ефективності, стійкості та конкурентоспроможності в галузі.

1 АНАЛІЗ ІСНУЮЧИХ ЗАСТОСУНКІВ ДЛЯ ОПТИМІЗАЦІЇ МАРШРУТІВ У ЛОГІСТИЧНИХ СИСТЕМАХ

1.1 Аналіз сучасних застосунків для оптимізації маршрутів у логістичних системах

Знаходження найкоротшого маршруту ще не означає його оптимізацію. Це багатогранний процес, який враховує час у дорозі, втому водія, затори, дорожні аварії, роботи з технічного обслуговування та навіть кількість поворотів і світлофорів.

То нащо ж потрібна автоматизація маршрутів? Допомогти компаніям зменшити витрати, збільшити доходи та оптимізувати роботу, одночасно надаючи першокласні послуги клієнтам. Керування маршрутом вручну є складним завданням. На сьогодні, є доступним програмне забезпечення, яке автоматизує процес логістики.

Оптимізувати маршрут вручну майже неможливо. Якщо вантажівка має десять зупинок під час своєї подорожі, то можливі сотні, якщо не тисячі, можливих змін маршруту. А якщо точок доставки куди більше десяти?

Ось тут і вступає в дію спеціальне програмне забезпечення. Воно дозволяє тестувати різні сценарії та вибирати найбільш прийнятне рішення для конкретних випадків.

Типовим прикладом переваг автоматизованої оптимізації маршрутів є міжнародна вантажна компанія «United Parcel Service» (UPS). Завдяки вагомим дослідженням UPS визначила, що повороти ліворуч є причиною заторів, збільшення споживання палива та затримок доставки. Вони почали перебудовувати маршрути вантажівок, щоб мінімізувати повороти ліворуч, досягнувши неймовірного скорочення на 90%. Це дозволило їм доставляти додатково 350000 пакунків на рік і заощадити майже 38 мільйонів літрів палива.

Це служить яскравою ілюстрацією того, як оптимізація маршруту дає результати.

Переваги для компанії від застосування оптимізації маршруту:

– економія палива;

– зниження витрат на технічне обслуговування автомобіля;

– збільшує швидкість доставки вантажу;

– прискорюється процес доставки, можна обробляти більше замовлень, без розширення автопарку;

– оптимізація навантаження на співробітників;

– значне скорочення ризиків, пов’язаних з аваріями, заторами і проблемними зонами;

– сприяє лояльності клієнтів завдяки якісному обслуговуванню;

– допомагає значно знизити кількість шкідливих викидів в атмосферу;

– покращується процес планування маршруту;

– підвищується безпека роботи співробітників.

Ці переваги роблять впровадження програм ефективним та вигідним для оптимізації маршрутів в логістичних системах.

Існує чимало готових рішень оптимізації, які можна запровадити. Нижче наведено одні з найбільш актуальних програм для оптимізації маршруту у 2023 році.

Samsara – це хмарна платформа, що здатна будувати маршрути та оптимізовувати їх. Сервіс може похвалитися безліччю цінних функцій, призначених для захисту подорожей від збоїв. Це програмне забезпечення легко інтегрується з відеореєстратором автомобіля та оснащене передовими функціями ШІ.

Завдяки відеореєстратору система відстежує стан водія, розраховує перерви на відпочинок і сповіщає про порушення правил дорожнього руху, обмеження швидкості або маршруту. Штучний інтелект, окрім того, стежить за дорогою, допомагає запобігати порушенням, визначає стан доріг, ситуацію з трафіком, зменшує ризики аварій та поломок (рис. 1.1).

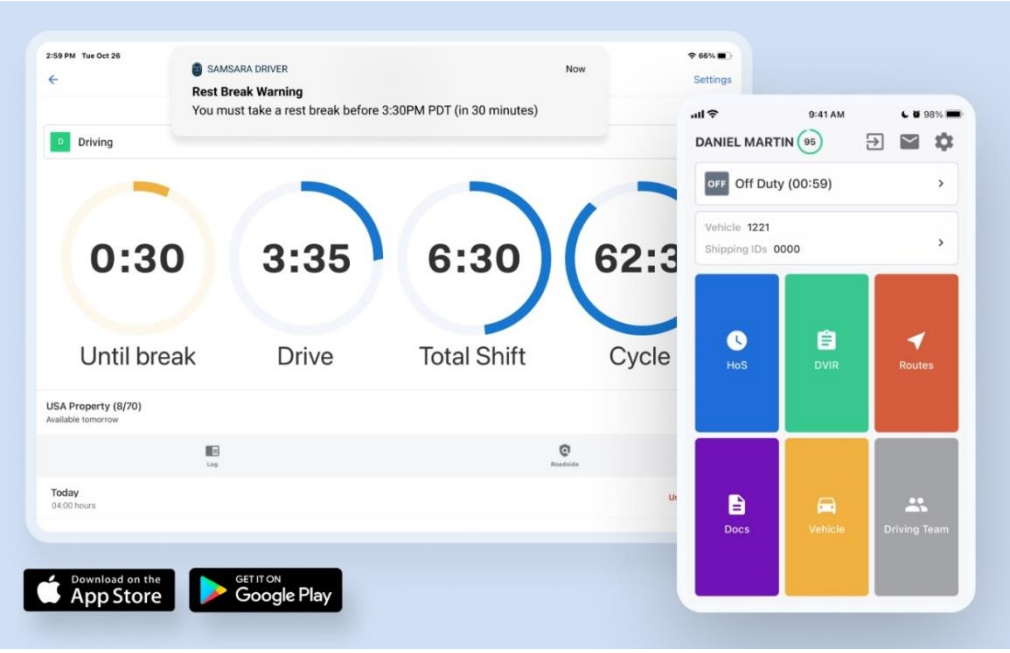


Рисунок 1.1 – Зразок інтерфейсу Samsara

Samsara надає інструменти диспетчеризації, можливості планування маршруту та моніторингу у реальному часі. Можна відстежувати свої діючі перевезення, розподіляти завдання та динамічно коригувати маршрути відповідно до реальних обставин. Система ретельно збирає вичерпні звіти та пропонує глибоку аналітичну інформацію.

Вартість послуг пристосована до конкретних функцій, які замовник бажає впровадити. Компанія публічно не розголошує ціни, замість цього замовник вказуєте свої вимоги та розмір свого автопарку. Потім представники зв’яжуться з ним, щоб обговорити дрібніші деталі та витрати.

Недоліком Samsara можна вважати навігацію в застосунку, яка не є інтуїтивно зрозумілою. Це спричинено великою кількістю функцій, що можуть здаватися складними для новачків. Окрім того, іноді під час оптимізації маршрутів сервіс будував найкоротші маршрути, які при цьому не були найоптимальнішими.

OptimoRoute вважається одним із найкращих готових рішень для оптимізації маршруту, що обслуговує понад 30 країн. Ця система може похвалитися понад 50 різноманітними функціями, які допомагають побудувати маршрут і зробити його найефективнішим.

Програма автоматично відстежує дії водія, спостерігає за його станом і оцінює поведінку на дорозі. Це допомагає розраховувати робочий час, щоб запобігти виснаженню співробітників, позбавляючи компанії від переплати за понаднормову роботу. Тисячі замовлень можна безперешкодно імпортувати з електронних таблиць, і всі вони відстежуються в режимі реального часу (рис. 1.2).

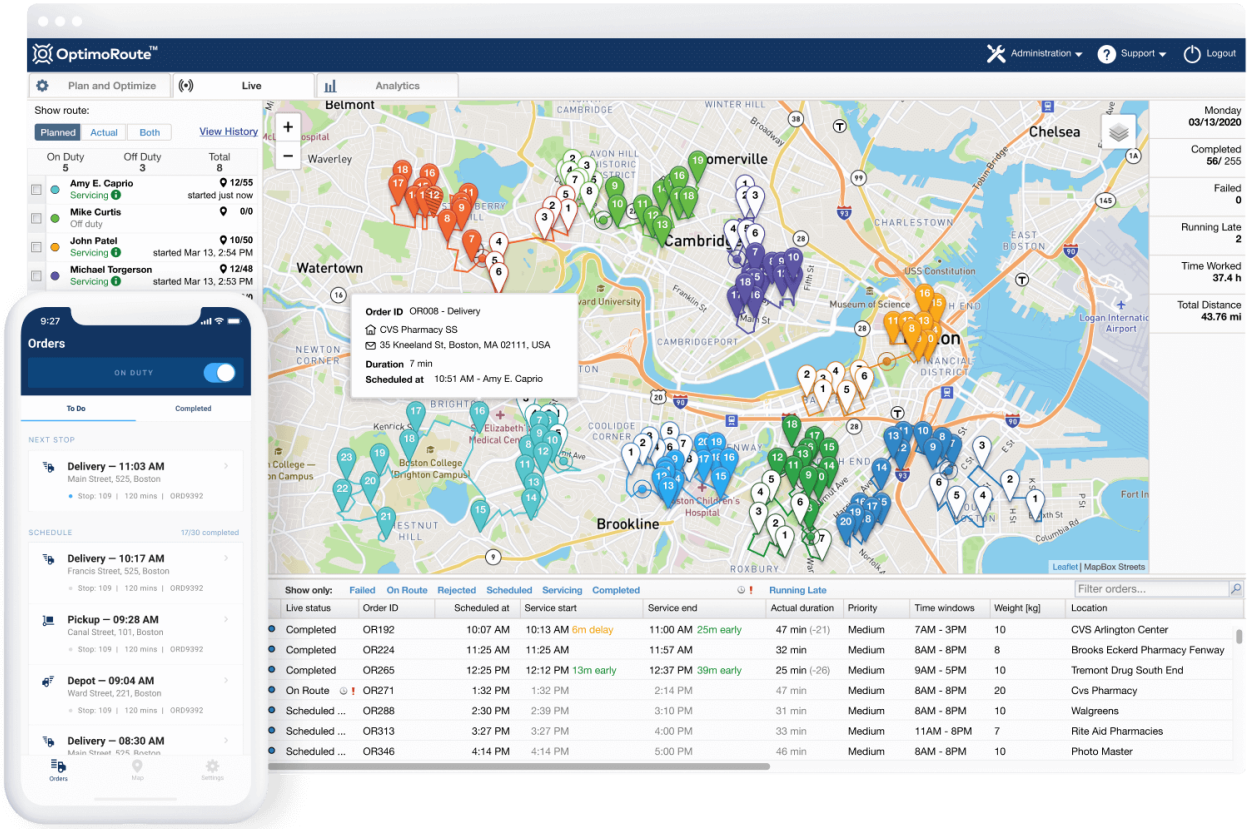


Рисунок 1.2 – Зразок інтерфейсу OptimoRoute

Програмне забезпечення відмінно справляється з оптимізацією маршруту, враховуючи пріоритет, дні тижня, часові вікна, замовлення на повернення та багато інших факторів. Крім того, доступний мобільний додаток для доступу в дорозі.

Недоліками сервісу є неможливість отримування сповіщень через e-mail (лише смс-повідомлення), потреба в ручному введенні обмежень на вантаж, маршрут та ін. Окрім цього при розрахунку маршрутів не враховуються погодні умови.

Verizon Connect виступає як надійний інструмент керування автопарком із персоналізованими функціями, що відповідають унікальним потребам підприємств. Повна інтеграція зі сторонніми службами розширює можливості продукту, спрощуючи управління ланцюгом поставок, фінансове відстеження та управління запасами транспортних засобів (рис. 1.3).

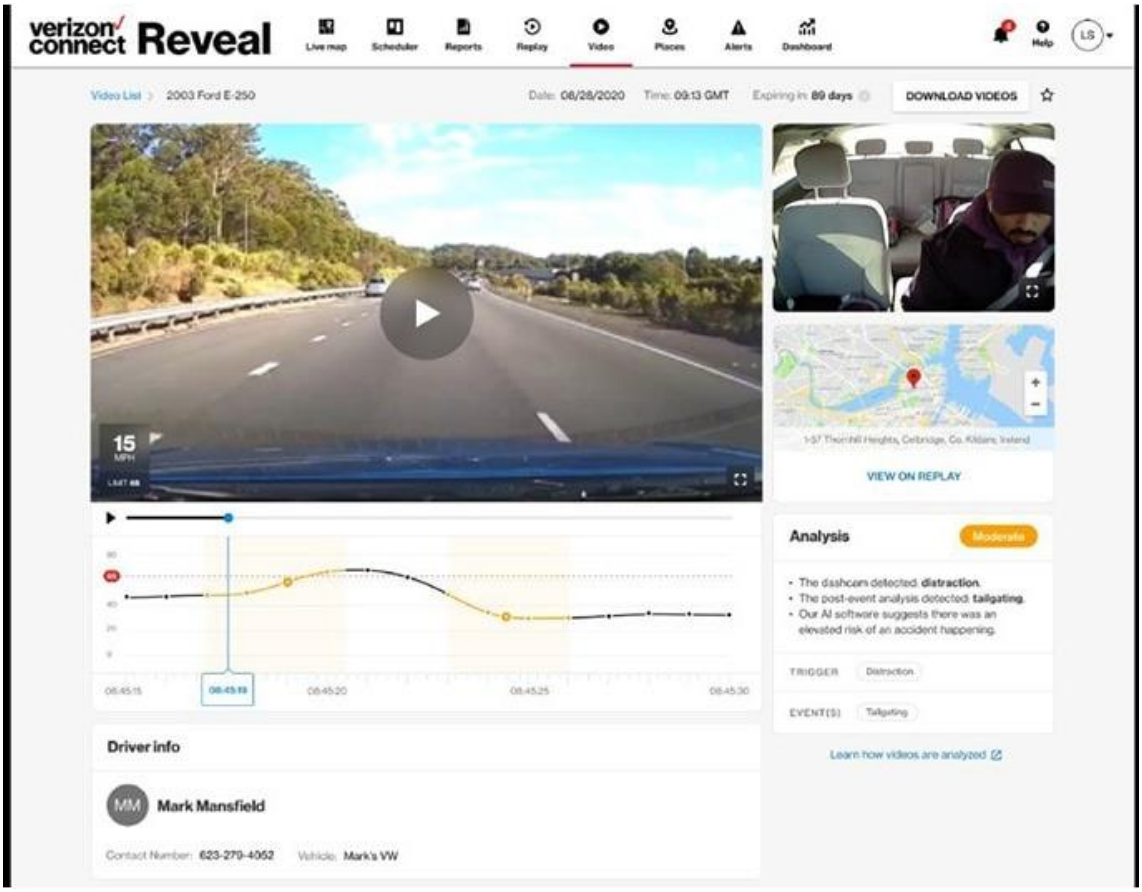


Рисунок 1.3 – Зразок інтерфейсу Verizon Connect

Система легко інтегрується з програмами управління паливом, дозволяючи точно відстежувати витрати палива та виявляти несправності в реальному часі. Інформаційна панель надає інформацію про поведінку водія, виявляючи раптове гальмування, перевищення швидкості, порушення правил тощо. І таким чином оптимізує продуктивність співробітників, запобігає додатковим витратам, нещасним випадкам та несправностям.

Відстеження в режимі реального часу дозволяє контролювати транспортні засоби, переглядати їхні маршрути та адаптувати їх до мінливих дорожніх умов, що робить сервіс безцінним рішенням для компаній.

Основним недоліком може бути висока вартість, що може стати серйозною перепоною для невеликих підприємств чи підприємств із обмеженим бюджетом.

Fleetio – це хмарне рішення, яке надає можливості побудови та оптимізації логістики маршрутів. Він складається з двох великих модулів: системи управління автопарком і системи управління запасами. Краса Fleetio полягає в його можливостях повної інтеграції з різними службами та системами, що значно розширює його функціональні можливості.

Спеціалізований декодер дозволяє користувачам зберігати понад 90 специфікацій автомобіля та контролювати їх стан. Інтеграція паливної картки та автоматична система нагадувань про технічне обслуговування на основі оцінки стану автомобіля є одними з основних функцій (рис. 1.4).

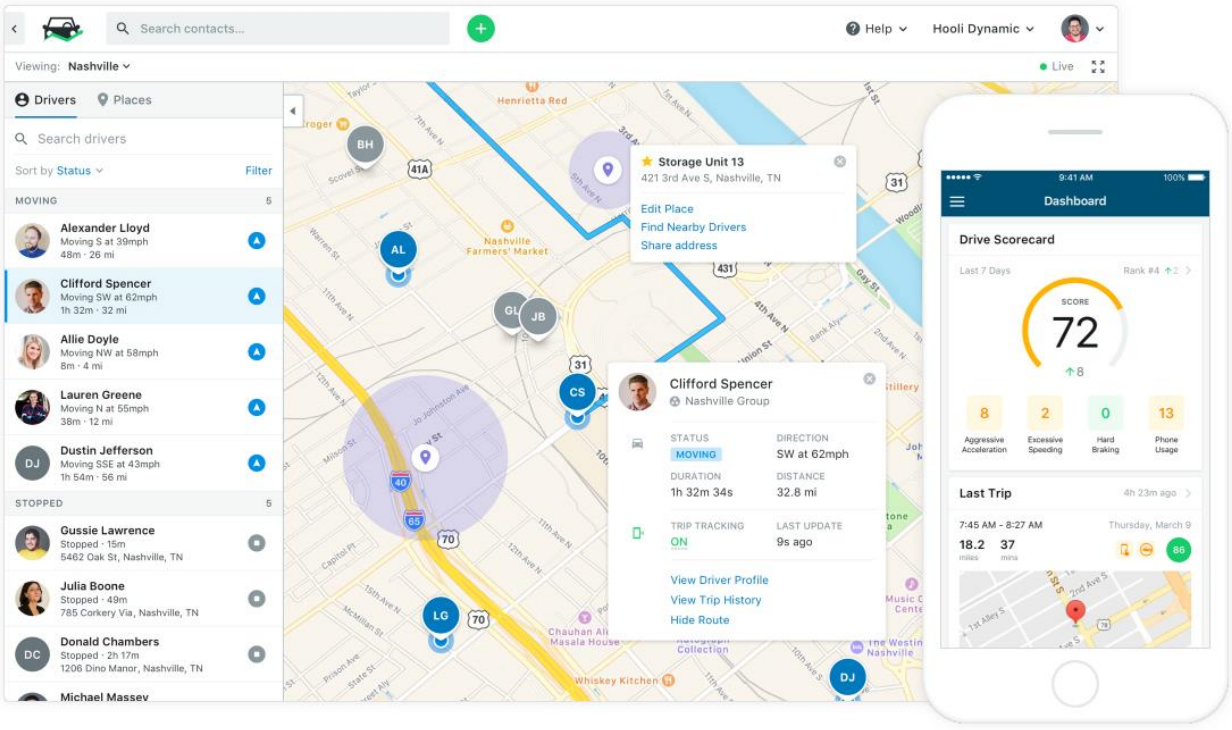


Рисунок 1.4 – Зразок інтерфейсу Fleetio

Панель моніторингу подана у вигляді зручних діаграм та графіків, що дає змогу збирати статистику, проводити аналітику та покращувати оптимізацію маршрутів.

Те, що Fleetio є хмарним сервісом, спричиняє деякі обмеження. Серед недоліків сервісу можна зазначити епізодичні збої та помилки серверів, а також необхідність високошвидкісного підключення до Інтернету.

Всі перераховані програми є широко використовуваними в даний час. Задача оптимізації маршрутів у них вирішується завдяки використанню специфічних методів оптимізації, таких як різноманітні тваринні технології, генетичний алгоритм та його модифікації, методи, засновані на алгоритмах Дейкстри, Флойда-Уоршелла, Левіта та інших. Також часто використовується комбінування декількох методів для досягнення кращого результату оптимізації.

1.2 Опис класичної задачі Vehicle Routing Problem

Проблема маршрутизації транспортних засобів – це загальний термін, що охоплює ряд проблем, коли набір географічно розосереджених клієнтів повинен мати набір маршрутів, визначених для парку транспортних засобів, розташованих на одному або кількох депо [1]. По суті, проблема полягає в пошуку оптимальної стратегії доставки, яка може мінімізувати загальні транспортні витрати для транспортування продуктів від джерел до споживачів.

Метою VRP є забезпечення доставки до набору клієнтів із відомими потребами через транспортні маршрути з мінімальними витратами, як починаючи, так і закінчуючи в депо.

На рисунку 1.5 зображено задачу VRP та один з можливих результатів.

У класичній VRP проблема визначається набором клієнтів (їхні вимоги та місця розташування), розташуванням депо, кількістю транспортних засобів, доступних для транспортування продукції, та їх пропускною здатністю.

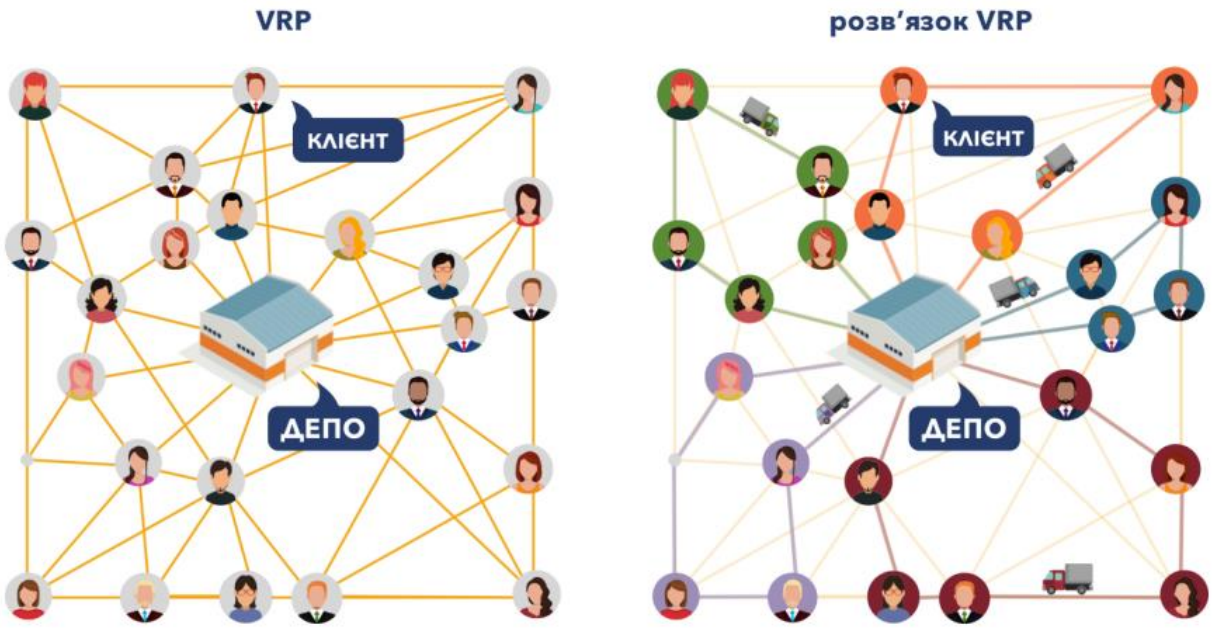


Рисунок 1.5 – Приклад VRP та одного з варіантів її розв'язання

Зі складу продукція повинна бути доставлена клієнтам у кількості, яка відповідає їхнім потребам. Кожен транспортний засіб повинен починати і закінчувати свій маршрут в депо і відвідати принаймні одного клієнта.

Проблема полягає у визначенні розподілу клієнтів і транспортних засобів по маршрутах і послідовності обслуговування клієнтів на кожному маршруті. Мета полягає в тому, щоб знайти рішення, яке мінімізує загальну вартість транспортування. Крім того, рішення повинно відповідати обмеженням, які вимагають, щоб кожен клієнт обслуговувався точно один раз і в повному обсязі, а загальний попит на кожному маршруті не повинен перевищувати пропускну здатність транспортного засобу. Транспортні витрати визначаються як вартість переміщення з будь-якого пункту в будь-який інший пункт.

Класична задача оптимізації маршрутизації транспортних засобів належить до сфери комбінаторної оптимізації [2]. Формально класичну VRP можна представити у вигляді графа $G = (N, A)$, де N – множина вузлів, що відповідають клієнтам і позначаються $1, 2, \ldots, n$, а вузол 0 представляє центральне депо, де всі транспортні засоби починають і закінчують свій маршрут; A позначає набір дуг, що з'єднують відповідні вузли графа

(клієнтів), так, що якщо i позначає одного клієнта, а j – іншого, то дуга, що з'єднує їх позначається $(i,j)\ \epsilon\ A$.

Кожен клієнт характеризується певним попитом d_i , $i \in N$. Для кожної дуги існує відповідний час у дорозі t_{ij} – час, необхідний транспортному засобу для подорожі від клієнта i до клієнта j, включаючи час обслуговування у клієнта i: s_i. Вартість переміщення транспортного засобу з i в j складає c_{ij}.

Індекс $k \in V$ згодом використовується для позначення відповідного транспортного засобу (де V означає кількість ідентичних ТЗ вантажопідйомністю q). Змінні x_{ijk} приймають значення {0, 1}, де 1 означає, що транспортний засіб рухається від вузла i до вузла j, 0 – зворотне [3].

Базуючись на наведених вище позначеннях, математична модель для VRP є наступною, де потрібно мінімізувати цільову функцію (1.1):

$$\sum_{k\in V}\sum_{(i,j)\in A} c_{ij}x_{ijk} \to \min, \tag{1.1}$$

$$\sum_{k\in V}\sum_{j\in N} x_{ijk} = 1, \text{ для } \forall i \in N, \tag{1.2}$$

$$\sum_{k\in C} d_i \sum_{j\in N} x_{ijk} \le q, \text{ для } \forall k \in V, \tag{1.3}$$

$$\sum_{j\in N} x_{0jk} = 1, \text{ для } \forall k \in V, \tag{1.4}$$

$$\sum_{i\in N}^{K} x_{ihk} - \sum_{j\in N}^{N} x_{hjk} = 0, \text{ для } \forall h \in N,\ \forall k \in V, \tag{1.5}$$

$$\sum_{j\in N} x_{i,n+1,k} = 1, \text{ для } \forall k \in V, \tag{1.6}$$

$$x_{ijk} \in \{0,1\},\ \forall (i,j) \in A,\ \forall k \in V. \quad (1.7)$$

Загальна вартість всіх маршрутів всіх транспортних засобів визначається цільовою функцією (1.1).

Обмеження (1.2) передбачає, що кожен клієнт обслуговується одним транспортним засобом і тільки один раз.

Обмеження (1.3) позначає, що ТЗ не може обслуговувати більше клієнтів, ніж дозволяє його вантажопідйомність.

Обмеження (1.4) гарантує, що кожен ТЗ виїжджає з депо лише один раз.

Обмеження (1.5) означає, що транспортний засіб може вийти з вершини *h*, тільки якщо він увійшов у цю вершину.

Обмеження (1.6) показує, що всі автомобілі обов'язково повертаються в депо, і лише раз. Це обмеження є похідним від обмежень (1.4) і (1.5).

Рішення проблеми передбачає:

– поділ множини вузлів на підмножини (маршрути);

– визначення порядку проходження для кожної підмножини.

Рішення вважається прийнятним (можливим), якщо всі маршрути відповідають обмеженням проблеми. VRP є узагальненням задачі комівояжера (TSP), і є NP-повною. TSP – це VRP з одним транспортним засобом, без обмежень, немає депо та клієнти не мають попиту.

1.3 Класифікація та аналіз існуючих методів для оптимізації маршрутів у логістичних системах

Оптимізація маршрутів у логістичних системах є складною задачею, яка вимагає вивчення та використання різних методів. Вибір методу залежить від конкретних умов та вимог задачі. Комбінація різних методів та використання нових технологій, таких як штучний інтелект та машинне

навчання, може покращити результати оптимізації маршрутів у логістичних системах.

Методи оптимізації маршрутів можна класифікувати за різними ознаками.

Точні методи: до цієї категорії входять метод гілок і меж, метод гілок та відсікань та інші.

Евристичні методи: ці методи базуються на експертних знаннях та правилах. До них відносяться метод найближчого сусіда, кластерні алгоритми та інші.

Метаевристичні методи: сюди належать генетичні алгоритми, та «тваринні» технології, такі як мурашині алгоритми, алгоритми бджолиного рою та інші, які моделюють природний пошук рішень.

Методи навчання з підкріпленням: вони використовують нейронні мережі та навчання з підкріпленням для вирішення задачі оптимізації маршрутів.

Точні методи: є математичними алгоритмами, спрямованими на пошук точних розв’язків задач оптимізації. Вони покладаються на математичні моделі проблеми та потребують значної обчислювальної потужності для вирішення.

Ці методи гарантують досягнення оптимальних рішень. Проте їхнє практичне застосування можливе лише для задач обмеженої розмірності [4]. Ці методи розв’язання обраного класу задач базуються на вичерпному переборі всіх можливих розв’язків, і є неефективними при вирішенні задач великої розмірності через значні часові затрати.

До точних методів відносяться такі методи, як метод гілок та меж, метод гілок та відсікань, динамічне програмування та інші.

Метод гілок та меж (Branch-and-Bound) – це метод, заснований на інтелектуальному переборі можливих розв’язків комбінаторної задачі оптимізації. Ця методика розгалужує простір потенційних рішень і використовує верхню та нижню межі для обмеження пошуку. Цей підхід

передбачає поділ простору рішень на підмножини, що не перекриваються та представлені у вигляді вузлів дерева розгалужень. Потім алгоритм досліджує гілки дерева відповідно до стратегії маршруту [5]. Щоб уникнути дослідження всього дерева, алгоритм оцінює значення вузла перед створенням нового вузла в дереві, порівнюючи найкраще можливе значення рішення, яке можна знайти у відповідному піддереві, з поточним значенням найкращого рішення [4]. Якщо краще рішення не може належати до піддерева, що починається у розглянутому вузлі, піддерево відсікається, в іншому випадку процес дослідження продовжується.

Метод гілок та відсікань (Branch-and-Cut). Основна ідея методу гілок та відсікань полягає в тому, щоб зменшити простір пошуку кандидатів шляхом введення нових обмежень. Загалом, алгоритм працює подібно до методу гілок та меж, але включає додатковий крок розбиття обмежень [5] – це мистецтво спрощення складних проблем шляхом рекурсивного розбиття їх на менші підпроблеми. Проблеми, які розв’язуються за допомогою динамічного програмування, зазвичай поділяються на етапи, на кожному з яких приймається рішення. Кожен етап також має набір пов’язаних з ним станів. Рішення на одному етапі перетворює поточний стан у стан наступного етапу. Рішення про перехід до наступного стану залежить виключно від поточного стану, а не від попередніх станів чи рішень [6].

Евристичні методи – це набір прийомів в пошуку розв’язання задачі, які мають на меті обмежити перебір, виконуючи відносно обмежений пошук у просторі рішень, що зазвичай веде до знаходження розв’язку в межах прийнятного часу. Недолік цих методів полягає в тому, що вони є наближеними, отримані за допомогою них розв’язки можуть значно відрізнятися від оптимальних. Тим не менш, їхня перевага полягає в їхній здатності знаходити прийнятні рішення для NP-повних задач великої розмірності в розумні часові рамки [7]. Евристичні методи часто протиставляються формальним методам, які опираються на точні математичні алгоритми.

Евристичні алгоритми можна розділити на три основні категорії.

Конструктивні алгоритми. Ці методи передбачають поступове створення рішення з відстеженням приросту його вартості. Їм не вистачає подальшої фази вдосконалення. Прикладами є «Найближчий сусід», «Жадібний алгоритм», «Заощадливий алгоритм» та «Insertion».

Двофазні (кластерні) алгоритми. У цих підходах проблема ділиться на дві операції: групування вершин для кожного майбутнього маршруту (кластеризація) і вирішення проблеми комівояжера (TSP) для кожної отриманої групи. Приклади охоплюють алгоритм Фішера і Джайкумара, алгоритм замітання (Sweep Algorithm), алгоритм пелюстки (Petal Algorithm) та алгоритм Османа. Двофазні алгоритми можуть демонструвати зворотний зв’язок між етапами вирішення (Petal, Sweep, Cluster-first-route-second, Rout first – cluster second) [7].

Покращуючі алгоритми. Ці алгоритми спочатку шукають рішення, а потім намагаються обмінюватися вершинами (ребрами) в межах кожного маршруту або між маршрутами для покращення рішення. Приклади включають різноманітні багатомаршрутні евристики покращення, такі як Inner Route, Intra Route, *k*-opt, 2-opt і 3-opt.

Евристичні методи оптимізації маршрутів є потужними інструментами в логістиці. Вони дозволяють знаходити прийнятні рішення для складних завдань оптимізації та планування маршруту. Однак їх ефективне використання потребує належної конфігурації та налаштування параметрів для досягнення бажаних результатів.

Метаевристичні методи – це клас оптимізаційних алгоритмів, що використовують стохастичні процеси та механізми пошуку для знаходження найліпших рішень. Їх застосування дає можливість знаходити оптимальні рішення та підвищувати ефективність логістичних операцій. Однак для досягнення найкращих результатів вони потребують належної конфігурації та врахування конкретних характеристик проблеми [3, 8].

Перевагою метаевристичних методів є їх здатність вирішувати складні задачі за відсутності безпосереднього знання простору пошуку. Це робить їх придатними для вирішення складних завдань оптимізації [8]. Метаевристичні методи опираються на ретельне дослідження найбільш перспективних областей простору рішень. Такі рішення часто перевершують ті, що отримані за допомогою класичної евристики. Відмінною рисою метаевристичних алгоритмів є те, що вони не забезпечують точної послідовності дій для вирішення задачі; кожен алгоритм додатково уточнюється шляхом налаштування його контрольних параметрів.

Іншими словами, метаевристика – це метод вирішення обчислювальних задач за рахунок поєднання існуючих процедур з відкритим інтерфейсом і закритою реалізацією, що призводить до високоефективних рішень. Практично всі метаевристики базуються на основі ідей, запозичених з процесів живої і неживої природи.

Генетичний алгоритм (Genetic Algorithm) – це добре відомий тип метаевристичного алгоритму, який привернув значну увагу в усьому світі. Генетичні алгоритми – це алгоритми еволюційного пошуку, які використовують механіку природного відбору та генетику для пошуку рішень проблем [8].

Принцип GA передбачає моделювання еволюції популяції особин шляхом створення нових поколінь нащадків за допомогою ітераційного процесу. Еволюція триває до тих пір, поки не будуть виконані певні критерії конвергенції, такі як максимальна кількість поколінь, конвергенція до однорідної популяції подібних особин або отримання оптимального рішення. Згодом найкраще сформована особина декодується, щоб отримати відповідне рішення.

Генетичні алгоритми працюють із популяцією рішень-особин, а не лише з одним рішенням, що дозволяє їм виконувати багатосторонній пошук одночасно. Кожна окрема особина представляє потенційне рішення проблеми.

Алгоритм мурашиних колоній (Ant Colony Optimization) є метаевристичним методом, що створений на основі імітації поведінки справжніх мурах.

При вирішенні комбінаторної задачі оптимізації за допомогою ACO проблему можна візуалізувати як пошук найкоротшого шляху у зваженому графі. Штучні мурахи (програмні агенти) співпрацюють, щоб знайти найліпший шлях, поступово будуючи маршрути, що складаються з вузлів графа. На кожному етапі вибраний вузол стохастичним способом додається до часткового рішення, але з залежністю від кількості феромону, доступного на вузлах або ребрах, які їх поєднують [9].

Слід феромону оновлюється мурахами під час кожної ітерації, що дозволяє їм співпрацювати у пошуку хороших рішень. Значення феромонів також поступово зменшуються з часом, імітуючи випаровування справжнього феромону. Отже, неякісні рішення поступово усуваються з розгляду в міру просування пошуку.

Завдяки гнучкості та надійності алгоритму, ACO є дуже привабливим для застосування до складних комбінаторних задач оптимізації. Методологія може бути адаптована для вирішення широкого кола завдань з мінімальними змінами основного алгоритму.

Алгоритм бджолиного рою (Artificial Bee Colony Optimization) належить до категорії алгоритмів ройового інтелекту, які описують поведінку систем, що самоорганізуються. Подібно до оптимізації мурашиної колонії, ABC ґрунтується на поведінці бджіл під час пошуку та збору їжі (нектару), але має свої особливості.

Бджоли-розвідники вирушають на пошуки найбагатших квітами місць у певній близькості від вулика. Під час пошуку бджола опирається на власні спостереження та спостереження інших бджіл. Деякі території переглядаються кілька разів і бджоли поступово звужують свій вибір до найбільш багатих квітами місць. Згодом бджола зупиняється в найбільш підходящому місці, і частина рою збирається до нього для збору нектару.

У ABC кожна штучна бджола, перебуваючи в русі, орієнтується на найкраще положення, визначене функцією придатності [10]. Кожна бджола підтримує свої індивідуальні найкращі позиції, які уточнюються та оновлюються кілька разів під час наступних пошуків. По суті, під час польоту бджола порівнює отриманий результат з новими даними й обирає кращий. В результаті рій сходиться в найкраще положення.

Зозулин пошук (Cuckoo Search) був представлений Сінь-Ше Янгом і Суашем Дебом у 2009 році. Алгоритм інспірований поведінкою зозуль, які підкладають свої яйця в гнізда інших птахів.

В алгоритмі CS кожне яйце в гнізді представляє рішення, а яйце зозулі відповідає новому розв'язку. Мета полягає в тому, щоб використовувати нові та потенційно кращі (зозулині) рішення для заміни менш перспективних рішень у гніздах. У найпростішому варіанті цього алгоритму кожне гніздо містить лише одне яйце.

Припускаючи, що йдеться про задачу глобальної безумовної оптимізації, яка спрямована на максимізацію функції придатності, алгоритм базується на наступних трьох правилах [11]:

– кожна зозуля відкладає по одному яйцю в навмання вибране гніздо;

– гнізда з яйцями високої якості (високі значення придатності) переходять до наступного покоління;

– яйце зозулі, відкладене у гніздо, з певною ймовірністю $\xi_a \in (0; 1)$ може бути виявлено господарем і вилучено з гнізда.

Останнім часом стала популярною практика комбінування різних алгоритмів, евристик та метаевристик, щоб використати переваги кожної процедури та знайти якомога краще вирішення задачі оптимізації маршрутів. Проте в даній роботі ця практика не досліджуватиметься.

1.4 Аналіз літературних джерел щодо апробації результатів стосовно оптимізації маршрутів у логістичних системах

Робота у джерелі [1] присвячена розробці інтелектуального алгоритму оптимізації для задачі маршрутизації автотранспорту. Вона поєднує переваги алгоритмів мурашиної колонії та оптимізації за допомогою Алгоритму пташиної зграї. В різних етапах алгоритму застосовуються різні стратегії пошуку, і використовується адаптивне налаштування для отримання зворотної інформації від середовища. Це прискорює швидкість збіжності, покращує здатність до навчання та допомагає уникнути локального оптимуму, забезпечуючи найкращий результат та підвищуючи ефективність.

У джерелі [2] розглядається багатокрокова проблема поширення ресурсів у логістичній мережі. Оскільки традиційні алгоритми мають недоліки, такі як повільна збіжність, слабкий локальний пошук і вразливість до раннього виходу з ладу, то в цій статті розроблено вдосконалений багатоцільовий алгоритм штучного бджолиного рою з адаптивними правилами сусідства. Процес вдосконалення алгоритму включає в себе: адаптивний розмір кроку при стратегії оновлення популяції замість випадкового розміру кроку та адаптивний метод оновлення ваги з кількома правилами пошуку в локальній оптимальності.

У джерелі [3] проаналізовано процес формування маршрутів доставки товарів для реалізації у міських магазинах за допомогою сервісу Ant Logistics на основі алгоритму оптимізації Ant Colony. Порівнюючи два варіанти формування маршрутів обслуговування однієї з найбільших торговельних мереж Харкова із застосуванням сервісу Ant Logistics, з’ясовано, що застосування алгоритму Ant Colony є більш оптимальним, ніж алгоритм Кларка-Райта на основі показників маршрутів доставки.

У джерелі [4] описується вирішення проблеми гонитви за найкоротшою відстанню, що спричинює ігнорування туристичного досвіду в процесі планування туристичного маршруту. Запропоновано вдосконалений

алгоритм оптимізації мурашиної колонії для планування туристичного маршруту. Його особливість полягає в тому, що контекстна інформація про мальовничі місця значно впливає на вибір людьми туристичного напрямку, тому стратегія оновлення феромонів поєднується з контекстною інформацією, такою як погода та ступінь комфорту мальовничого місця.

У джерелі [5] наведено статтю, в якій проводиться дослідження розвитку та впровадження різних алгоритмів для оптимізації логістичних систем в період з 2010 рік по 2020 рік. Дослідження проводилось для 5 категорій: алгоритм мурашиної колонії, алгоритм рою частинок, меметичний алгоритм, генетичний алгоритм і алгоритм штучної бджолиної колонія. Результати показали, що протягом 10 років біоінспіровані процедури зазнали швидкого прогресу. Вони успішно застосовуються для оптимізації та проєктування особливо складних систем, таких як логістичні системи розподілу. В той же час інші типи алгоритмів зазнали меншого вдосконалення і часто є менш ефективними.

У джерелі [6] досліджуються проблеми екологічної маршрутизації транспортних засобів у логістиці холодового ланцюга з урахуванням повного набору викидів парникових газів. Розроблена відповідна оптимізаційна модель екологічного маршруту використовує стандартну оптимізацію роєм частинок і модифіковану оптимізацію роєм частинок. Метою цього дослідження є мінімізація загальних витрат, які включають експлуатаційні витрати транспортного засобу, витрати на втрату якості, вартість свіжості продукту, вартість штрафу, вартість енергії та вартість викидів парникових газів.

У джерелі [7] було запропоновано та реалізовано евристичний підхід, заснований на оптимізації роєм частинок, який називається ePSO, для вирішення проблем маршрутизації транспортного засобу з одночасним перехресним докінгом і скороченням викидів вуглецю. В роботі проводиться порівняння ефективності ePSO та генетичного алгоритму. Експериментальні

результати показали, що запропонований підхід ePSO був кращим, ніж GA, у більшості випадків за результатами перевірки статистичних гіпотез.

Робота у джерелі [8] присвячена підвищенню ефективності планування і оптимізації транспортних маршрутів із урахуванням часових обмежень, розробці програмного забезпечення з використанням сучасного мета-евристичного метода покращеної мурашиної колонії 3-opt для вирішення задачі комівояжера. Описується аналіз особливостей та характеристик сучасних систем оптимізації логістичних операцій підприємств малого бізнесу; аналіз та узагальнення існуючих методів оптимізації різнотипних логістичних операцій; дослідження різних модифікацій мурашиного алгоритму та особливостей їх застосування для вирішення задачі комівояжера.

У джерелі [9] наведено критичний аналіз існуючих на сьогоднішній день науково-методичних уявлень та інформаційних технологій щодо оптимізації процесів доставки вантажів, визначені відповідні проблеми та шляхи їх вирішення. Визначено, що використання інформаційних технологій в організації, оптимізації та управлінні логістичними процесами здійснення вантажних і пасажирських перевезень, а також управлінні дорожнім рухом у містах в умовах великої завантаженості в теперішній час є епізодичним і недосконалим. Також досліджено проблему подальшого розвитку інформаційних технологій вирішення задач з динамічної маршрутизації перевезень в реальному часі.

У джерелі [10] описуються порівняння різних методів оптимізації руху вантажного транспорту в логістичних мережах. Аналізується вплив різних чинників на результати оптимізації. Об’єктом порівняння є результати застосування алгоритмів для вирішення задачі комівояжера. В підсумках роботи зазначено, що найкращі показники мають модифіковані тваринні алгоритми, зокрема мурашиний алгоритм, алгоритм бджолиного рою та алгоритм коника.

В роботі [11] досліджено проблеми, пов’язані зі використанням інформаційних технологій для управлінні логістичною системою підприємства. Автори аналізують основні виклики, з якими стикаються компанії при впровадженні інформаційних технологій в логістику, такі як високі витрати, недостатня кваліфікація персоналу, неповна системна інтеграція тощо. Крім того, у статті пропонуються конкретні рекомендації та рішення щодо подолання цих проблем для успішного впровадження інформаційних технологій у логістичну систему компанії.

Автори роботи [12] розглядають сучасні інформаційні технології, що застосовуються в транспортній логістиці, описуючи їх ключові характеристики та вплив на управління транспортними процесами, оптимізацію маршрутів, контроль за рухом транспорту, зменшення витрат та підвищення ефективності логістичних операцій. Крім того, у статті наводяться приклади застосування конкретних інформаційних технологій у транспортній логістиці та висвітлюються переваги, які вони можуть надати компаніям у цій галузі.

У статті [13] автори аналізують фактори, які впливають на успішне впровадження та використання інформаційних логістичних систем, серед яких: недостатня інформаційна культура, фінансові обмеження, брак кваліфікованого персоналу та інші. Вони пропонують шляхи вирішення цих проблем з метою покращення логістичних процесів та ефективності бізнесу в Україні.

Огляд сучасних інформаційних технологій, що використовуються в транспортно-логістичній галузі, їх вплив на ефективність, розвиток систем управління наведено в роботі [14]. У статті зосереджено увагу на тому, як інформаційні технології можуть поліпшити планування, координацію, моніторинг та управління транспортними процесами і ресурсами, зрештою сприяючи підвищенню ефективності та якості логістичних операцій.

1.5 Постановка задачі дослідження

Створені природою алгоритми стали потужними інструментами для вирішення складних задач оптимізації.

Досліджувані алгоритми отримали визнання за їхній потенціал щодо надання майже оптимальних рішень проблеми маршрутизації транспортного засобу та її різноманітних розширень. Кожен із цих алгоритмів містить унікальний набір функцій і механізмів, що робить їх придатними кандидатами для порівняльного аналізу.

Об’єктом дослідження є оптимізація маршрутів у логістичних системах.

Метою дослідження є вивчення та порівняння методів оптимізації маршрутів у логістичних системах.

Для досягнення мети необхідно вирішити такі завдання:

– провести аналіз особливостей існуючих логістичних систем;

– провести аналіз перспективних оптимізаційних алгоритмів;

– розробити математичні моделі для кожного алгоритму;

– реалізувати алгоритми ACO, ABC та CS;

– провести експериментальне дослідження впливу кожного з алгоритмів на оптимізацію маршрутів у логістичних системах;

– провести порівняльний аналіз ефективності методів;

– сформулювати висновки та рекомендації.

2 ОСОБЛИВОСТІ МЕТОДІВ ОПТИМІЗАЦІЇ МАРШРУТІВ У ЛОГІСТИЧНИХ СИСТЕМАХ

2.1 Оптимізаційна логістична задача

Загалом оптимізаційна логістична задача має виглядає так [12]:

$$\left.\begin{array}{l} y = f(x) \to \max(\min) \\ x \in X \end{array}\right\}, \tag{2.1}$$

де X – множина допустимих розв'язків (альтернатив, дій, можливих варіантів логістичних рішень);

f – числова функція, визначена на множині X, котра разом з вимогою максимізації чи мінімізації називається цільовою функцією.

Розв'язок оптимізаційної логістичної задачі (2.1) являє собою пару X^*, y^*, де X^* – множина оптимальних рішень, y^* – оптимальне (максимальне, чи мінімальне – залежить від напрямку оптимізації) значення цільової функції, яке вона досягає на множині можливих рішень X. Зазвичай ці задачі вирішуються частково, обмежуючись не загальним розв'язанням і визначаючи лише одне рішення серед множини оптимальних вирішень, а не всю цю множину.

Розв'язання оптимізаційної логістичної задачі передбачає використання спеціалізованих математичних методів оптимізації, комп'ютерних програм та обчислювальних засобів на основі відповідної вхідної інформації [13].

Будь-яка оптимізаційна логістична задача складається з двох компонентів: цільової функції та обмежень. Цільова функція формалізує критерій оптимальності, визначаючи найкращий серед альтернативних варіантів логістичного рішення. З іншого боку, обмеження визначають набір

можливих альтернатив. Обмеження представлені у формі нерівностей та/або рівнянь.

Найчастіше оптимізаційні логістичні задачі є багатовимірними і зазвичай приймають наступну узагальнену форму:

$$\left.\begin{array}{l} y = f(x_1, \ldots, x_n) \to \max(\min) \\ g_i(x_1, \ldots, x_n) \le 0, \quad i = \overline{1, m} \\ h_k(x_1, \ldots, x_n) = 0, \quad k = \overline{1, p} \end{array}\right\}, \qquad (2.2)$$

де x_1, ..., x_n та y – дійсні змінні (керовані параметри), перші n з яких є основними та утворюють план $x = (x_1, \ldots, x_n)$ задачі, а остання вказує відповідне значення цільової функції;

f, g_i, $i = \overline{1, m}$, h_k, $k = \overline{1, p}$ – числові функції змінних x_1, ..., x_n (перша функція є цільовою, а решта використовуються з метою відбиття множини допустимих рішень).

Якщо в рівнянні (2.2) кожна з функцій f, g_i, $i = \overline{1, m}$ та h_k, $k = \overline{1, p}$ є лінійною, то передбачається задача лінійного програмування, інакше це проблема нелінійного програмування. Зазвичай логістичні задачі оптимізації є лінійними. Нелінійні цільові функції або окремі обмеження виникають у випадках, коли залежності між певними змінними демонструють нелінійні характеристики.

Серед обмежень оптимізаційної логістичної задачі можна зустріти спеціальні обмеження, такі як обмеження на знаки певних змінних або вимоги щодо їх цілісності. Такі обмеження виокремлюють, іменуючи невиокремлені обмеження як основні, а виокремлені – додаткові. У випадку, коли серед додаткових обмежень відсутні вимоги до цілочисельності, маємо задачу математичного програмування (лінійну або нелінійну) з неперервними змінними. У випадку ж коли одна чи декілька змінних повинні набувати лише цілих (у загальнішому випадку – дискретних) значень – задачу цілочисельного (дискретного) математичного програмування.

Наприклад, задача про оптимізацію плану перевезень вантажів може бути задачею математичного програмування з неперервними змінними, що визначають обсяги перевезень по відповідним маршрутам, в свою чергу, задача щодо визначення оптимальної локації розміщення нового розподільчого центру логістичної мережі це задача дискретного математичного програмування (точніше з логічними змінними 0 та 1), які за допомогою значень 1 або 0 характеризують підтвердження вибору чи, навпаки, відмову від вибору деякого пункту для розташування в ньому нового розподільчого центру.

Тип задачі (лінійна, нелінійна, дискретна) визначає методи її вирішення, зокрема:

– лінійне програмування (симплексний метод, подвійний симплексний метод та інші);

– цілочисельне програмування (методи розгалужень і відсікань, розгалужений пошук, комбінаторні, евристичні та методи випадкового пошуку);

– нелінійне програмування (прямі та непрямі методі, проєкція, лінеаризація та інші);

– інші (залежать від особливостей проблеми, що розглядається).

Детальне пояснення методів лінійного, цілочисельного та нелінійного програмування наведено у спеціальній літературі з математичного програмування, методів оптимізації та дослідження операцій [12–15].

Реалізація оптимізаційних методів для вирішення завдань логістичної оптимізації ефективно здійснюється за допомогою обчислювальних засобів та спеціалізованого програмного забезпечення. Слід зазначити, що прогрес у галузі економіко-математичного моделювання та методів оптимізації розвивається паралельно з розвитком комп’ютеризації.

Оптимізація логістичних рішень здійснюється на основі всебічного аналізу комплексу взаємопов’язаних факторів, визначення та порівняльної оцінки можливих альтернатив і можливих варіантів дій.

Для пошуку рішень оптимізаційної логістичної задачі використовується поєднання економіко-математичних методів моделювання й оптимізації, обчислювальної техніки та необхідних програмних засобів.

Процес оптимізації логістичних рішень із застосуванням економіко-математичних інструментів складається з наступних головних етапів:

– формулювання проблемної ситуації, постановка цілей та визначення обмежень;

– розробка економіко-математичної моделі;

– обрання методів та програмних засобів для проведення розрахунків;

– підготовка вхідних даних;

– пошук та аналіз альтернатив рішень;

– затвердження рішення та узгодження плану його реалізації;

– контроль виконання рішення та оцінка результатів;

– проведення підсумкового аналізу проблемної ситуації з подальшим її переосмисленням (з можливим поверненням до попередньої чи початкової фази).

Таким чином, краще розглядати логістичну оптимізацію як циклічний процес, який постійно оновлюється, а не як ізольований акт у цьому процесі.

Слід зазначити, що економіко-математичне моделювання, яке передбачає переведення цілей і обмежень, що відповідають проблемній ситуації, у математичні форми, є ключовим етапом оптимізації логістичних рішень.

Приклади задач логістичної оптимізації включають наступне:

– створення найбільш економічно ефективного плану транспортування продукції, сировини чи інших виробничих ресурсів від постачальників до споживачів безпосередньо, чи через центри розподілу;

– визначення максимальної пропускної здатності транспортної мережі;

– розрахунок найбільш економічно ефективного транспортного маршруту між двома визначеними точками в межах транспортної мережі;

– вибір найкращої локації розміщення нового розподільчого центру в логістичній мережі;

– формування оптимальної стратегії керування постачаннями виробничих ресурсів та готової продукції;

– створення оптимального плану реальних інвестицій для забезпечення необхідного зростання виробничих потужностей логістичних систем;

– розробка систем оптимального керування запасами сировини, інших виробничих ресурсів, готової продукції тощо.

2.2 Механізм оптимізації маршрутів на основі методу Мурашиних колоній

У даний час у галузі «Природних обчислень» (Natural Computing) активно розробляються та вдосконалюються методи глобальної оптимізації, черпаючи натхнення з природних систем. Ці методи штучного інтелекту включають механізми прийняття рішень і адаптації, які природним чином розвивалися протягом мільйонів років [7]. Подібним чином мурахи, які існують понад 100 мільйонів років, пристосувалися виживати в різних умовах по всьому світу, утворюючи популяції, загальна вага яких становить від 10% до 25% біомаси наземних тварин [7]. Незважаючи на відносно просту поведінку окремих мурах, мурашині колонії являють собою складні соціальні структури, здатні вирішувати складні проблеми, такі як пошук найкоротшого шляху за допомогою хімічної регуляції.

Рисунок 2.1 яскраво ілюструє харчову поведінку справжніх мурах у пошуках джерел їжі.

Феромонний слід дозволяє мурахам відкривати нові маршрути, коли перешкоди загороджують їхні існуючі шляхи. Якщо найкоротший шлях стає недоступним через перешкоду, мурахи повинні знайти новий найкоротший шлях.

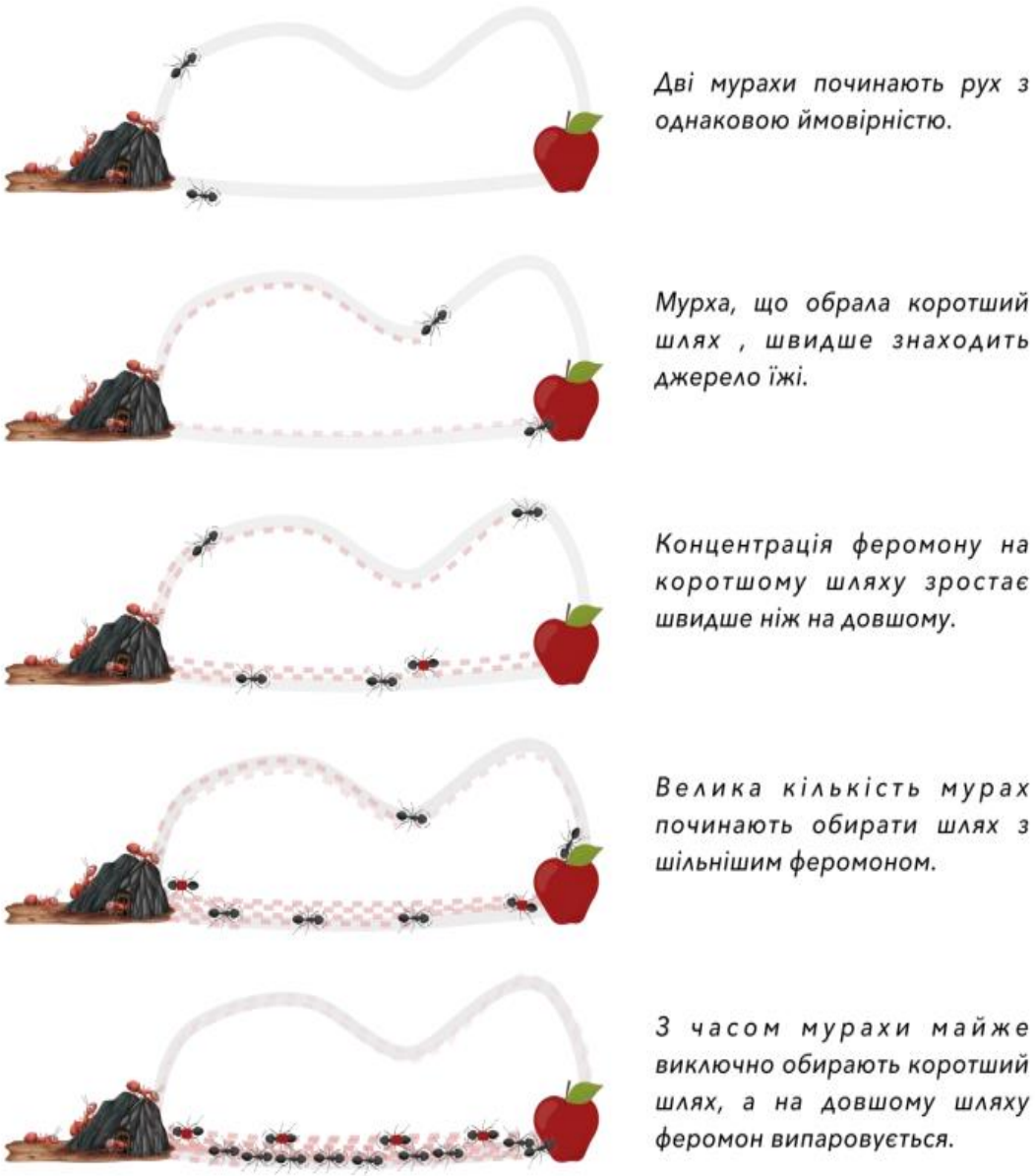


Рисунок 2.1 – Поведінка мурах в процесі пошуку їжі

Спочатку вони з однаковою ймовірністю обходять перешкоду з обох сторін, як на шляху до їжі, так і на зворотному шляху. Мурахи, які вибирають коротший шлях, пройдуть його швидше в обох напрямках, таким чином збагачуючи його більшою кількістю феромонів. Оскільки вища концентрація феромону на шляху збільшує ймовірність того, що мурахи пройдуть цей шлях, наступні мурахи з більшою ймовірністю виберуть цей шлях, насичуючи шлях феромоном, поки він залишається доступним (рис. 2.2).

Така поведінка мурах привернула увагу багатьох дослідників, які вивчають механізми взаємодії окремих особин в колонії.

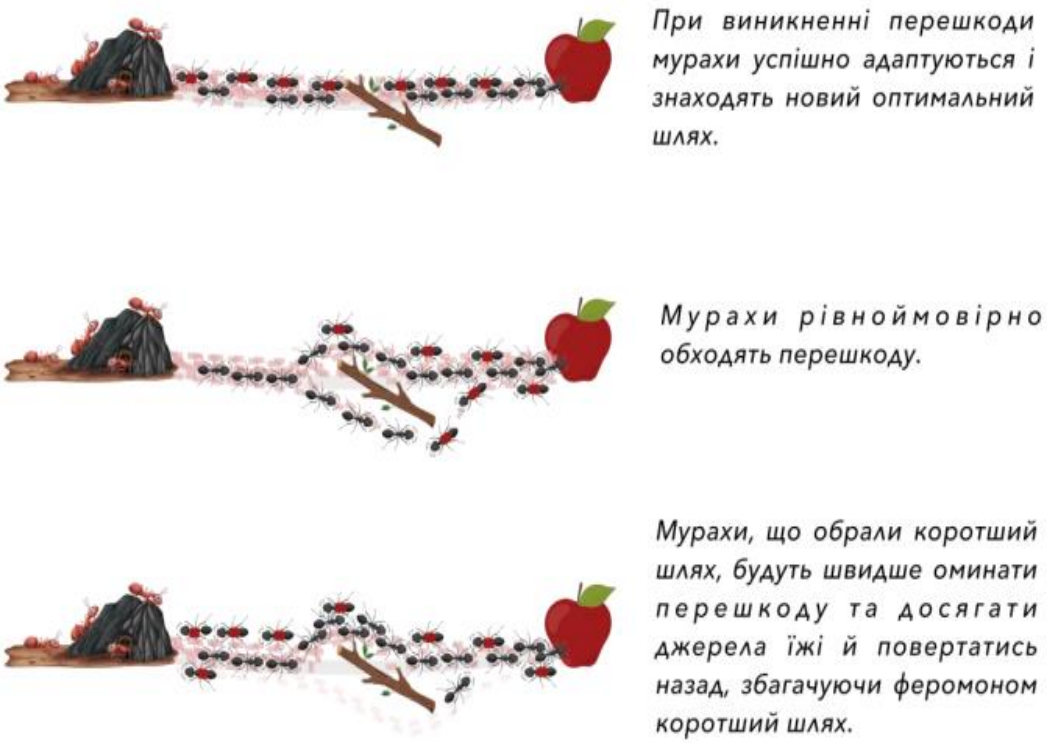


Рисунок 2.2 – Поведінка мурах під час виникнення перешкоди

Проводячи експерименти з вибором між двома шляхами різної довжини, що ведуть від мурашника до джерела їжі, біологи помітили, що, як правило, мурахи врешті-решт використовують більш короткий шлях [9].

Модель такої поведінки наступна:

– спочатку мурахи безладно рухаються від гнізда до джерела їжі;

– коли одна мураха знаходить їжу, вона повертається до гнізда, залишаючи по собі слід з особливого ферменту – феромону;

– інші мурахи виявляють «аромат» відкладеного феромону і намагаються слідувати по позначеному шляху;

– сліди феромонів створюють асинхронну та непряму комунікаційну схему, що дозволяє мурахам обмінюватися інформацією один з одним у міру виявлення альтернативних шляхів до їжі: чим вища концентрація феромонів на шляху, тим більша ймовірність, що інші мурахи підуть ним;

– коротші шляхи потребують менше часу і, отже, частіше ніж інші позначаються феромоном, а висока концентрація феромонів, у свою чергу,

робить коротший шлях більш привабливим, залучаючи все більшу кількість мурах;

– довші шляхи, що рідко використовуються, зрештою, зникають через зниження кількості феромону в процесі випаровування.

Ця непряма комунікація в колонії мурах, відома як стигмергія, активована описаними механізмами позитивного та негативного зворотного зв'язку, забезпечує відкриття та підкріплення найкоротшого шляху. Основні принципи поведінки мурашиної колонії [9]:

– простота дій кожної мурахи;

– відсутність централізованого контролю;

– непрямий обмін інформацією через відкладання феромонів;

– випаровування феромонів з часом, що забезпечує адаптивну поведінку.

Мурашиний алгоритм, що також відомий як алгоритм оптимізації мурашиними колоніями (Ant Colony Optimization algorithm) – це добре відома метаевристика, яка належить до сімейства методів ройового інтелекту. Його суть полягає в аналізі та застосуванні описаної поведінкової моделі мурашиної колонії ля вирішення різноманітних задач маршрутизації на графах. Перший алгоритм мурашиної колонії був запропонований у докторській дисертації Марко Доріго в 1992 році для вирішення проблеми пошуку оптимального шляху на графі [9].

Основні етапи роботи ACO в загальній формі показані на структурній блок-схемі на рисунку 2.3.

Віртуальні мурахи послідовно будують рішення задач, рухаючись по ребрах графа. При цьому для руху з найбільшою ймовірністю будуть обрані ребра з малою довжиною і великою кількістю феромону, який задається для кожного ребра певним умовним позначенням. Феромон оновлюється під час кожної ітерації: концентрація збільшується на побудованих маршрутах пропорційно до їх довжини та зменшується по всьому графу з урахуванням коефіцієнта випаровування.

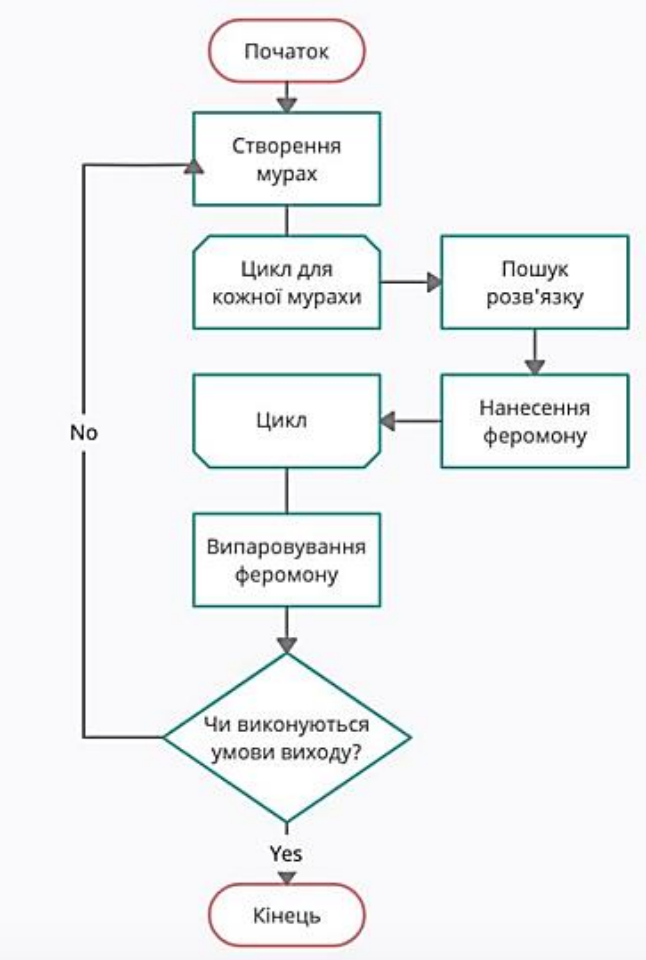


Рисунок 2.3 – Структурна схема алгоритму мурашиних колоній у загальному вигляді

Згодом більшість мурах вибере найкоротший шлях, завдяки накопиченню на ньому значної кількості феромону, що є формою позитивного зворотного зв’язку. Щоб уникнути застрягання в локальному оптимумі, також моделюється негативний зворотній зв’язок у вигляді випаровування феромонів. При моделюванні мурашиної поведінки на графі, де ребра представляють можливі шляхи руху мурах, шлях із найбільшим накопиченням феромонів на ребрах графа є вирішенням задачі, що виникає в результаті роботи ACO.

Теоретичний інтерес для наукових досліджень представляє питання конвергенції алгоритмів мурашиних колоній. На даний момент є докази гарантованої збіжності алгоритму мурашиної колонії, але через його стохастичну природу час збіжності є невизначеним [15].

Алгоритм мурашиної колонії, незалежно від модифікацій, можна представити як [15].

Поки умови виходу лишаються невиконаними, повторювати кроки:

Крок 1. Створення мурах.

Крок 2. Знаходження рішення.

Крок 3. Оновлення феромону.

Крок 4. Додаткові дії (за потреби).

Створення мурах. Початкове розміщення мурах визначається умовами задачі, що є визначальним для кожного завдання. Існують різні варіанти розміщення особин: вони можуть починати з одної точки або з різних точок, які, в свою чергу, можуть повторюватись або не повторюватись.

При створенні мурах необхідно ввести невелике позитивне число, яке позначатиме початковий рівень феромону, щоб запобігти нульовій ймовірності переходу на початковому етапі [9].

Знаходження рішення. У кожен момент часу ймовірність переходу між вершинами визначається наступною формулою:

$$p_{ij}(t) = \frac{\tau_{ij}(t)^{\alpha}\left(\frac{1}{d_{ij}}\right)^{\beta}}{\sum_{j \in J_i} \tau_{ij}(t)^{\alpha}\left(\frac{1}{d_{ij}}\right)^{\beta}}, \qquad (2.3)$$

де J_i – множина вершин, в які можна перейти з вершини i;

$\tau_{ij}(t)$ – рівень феромону на дузі (i, j) на кроці t;

d_{ij} – відстань між вершинами i та j;

α, β – константні параметри.

При $\alpha = 0$ вибір найближчої вершини є найбільш імовірним, що робить алгоритм жадібним. При $\beta = 0$ феромон є єдиним фактором у відборі, що призводить до локальних оптимумів (субоптимальних рішень). Саме тому

виникає потреба в експериментальному пошуку компромісу між цими значеннями [15].

Оновлення феромону. Значення рівня феромону оновлюється за формулою:

$$\tau_{ij}(t+1) = (1-\rho)\tau_{ij}(t) + \sum_{k \in M_{ij}} \frac{L_{\min}}{L_k}, \tag{2.4}$$

де ρ – рівень інтенсивності випаровування;

M_{ij} – множина мурашок, які пройшли по дузі (i, j);

L_k – вартість поточного рішення (довжина маршруту) для k-ї мурахи;

L_{min} – параметр, що приймає значення порядку вартості оптимального рішення (передбачувана ідеальна вартість рішення в задачі мінімізації);

$\frac{L_{\min}}{L_k}$ – феромон, що відклався k-ю мурахою на ребрі (i, j).

Таким чином, чим гіршим є рішення (чим вище відповідне L_k), до якого входить ребро (i, j) , тим менше феромону мураха відкладе на цьому ребру [15].

Додаткові дії. Зазвичай на цьому кроці застосовується алгоритм локального пошуку, але він також може бути застосований на фінальному етапі після знаходження всіх рішень.

Алгоритм мурашиної колонії не гарантує досягнення оптимального рішення, але має поліноміальну складність, що дозволяє швидко отримувати приблизні рішення для складних проблем, таких як проблема маршрутизації транспортного засобу (VRP). Точні методи мають факторіальну складність і зазвичай неефективні для понад 10 точок.

Переваги ACO. Для VRP алгоритми мурашиних колоній є відносно ефективними, вони працюють краще, ніж інші глобальні оптимізації для VRP, такі як нейронні мережі чи генетичні алгоритми [7, 13–15]:

– алгоритм орієнтується на пам’ять всієї колонії а не окремої особини чи на пам’ять попереднього покоління;

– випадковий вибір шляху і пам’ять колонії мінімізують схильність до неоптимальних початкових рішень;

– АСО адаптований до динамічних змін;

– може застосовуватися для багатьох різних задач.

До недоліків АСО можна віднести:

– теоретичний аналіз ускладнюється тим, що розподіл ймовірностей змінюється ітеративно;

– дослідження є більш експериментальними, аніж теоретичними;

– збіжність гарантована, але час збіжності не визначений;

– зазвичай потрібні додаткові методи, наприклад локальний пошук;

– алгоритм залежить від параметрів, які можна підібрати лише через експерименти.

2.3 Механізм оптимізації маршрутів на основі методу Бджолиної колонії

Щоб знайти глобальний екстремум, необхідно виконати багаторазові обчислення або використовувати більш складні методи пошуку, що містять імовірнісні елементи. Були запропоновані різні алгоритми для вирішення проблем такого роду, включаючи алгоритми випадкового пошуку (неорієнтовані, спрямовані та самонавчальні), генетичні та еволюційні алгоритми та алгоритми моделювання відпалу.

Одним із останніх варіантів генетичного алгоритму пошуку є алгоритм Бджолиного рою (також відомий як Particle Swarm Optimization, Artificial Bee Colony Algorithm і Bees Algorithm).

Використання метода Бджолиного рою базується на концепціях моделювання багатоагентних систем, які використовуються для дослідження

динаміки децентралізованих систем. Агент працює автономно, дотримуючись набору простих правил, і може взаємодіяти з середовищем та іншими агентами [8, 12].

Кожна бджола в роєві вважається частинкою або агентом. Усі частинки рою діють окремо відповідно до єдиного принципу управління: рухаються до найкращих персональних та глобальних позицій, постійно оцінюючи значення поточної позиції (рис. 2.4).

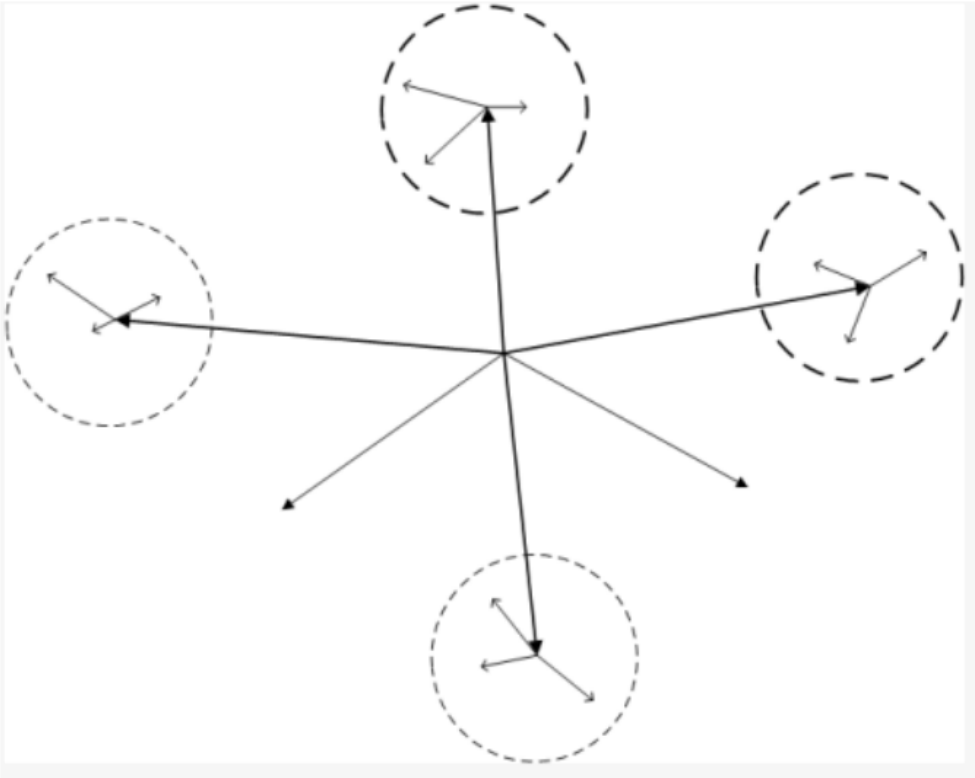


Рисунок 2.4 – Схематичне зображення стратегії алгоритму

Положення бджоли представляє координати в досліджуваному N-вимірному просторі.

Персональна найкраща позиція (ПНП) – є позиція з найвищим значенням цільової функції, знайденої бджолою. У кожної бджоли своя ПНП. У кожній точці свого шляху бджола порівнює значення цільової функції в поточній позиції зі значенням своєї ПНП. Якщо поточне положення має вище значення функції придатності, то ПНП оновлюється до значення поточної позиції.

Глобальна найкраща позиція (ГНП) визначається як позиція з найвищим значенням цільової функції, яку виявляє весь рій. Інформація про ГНП доступна кожній окремій бджолі. Якщо під час руху одна з бджіл виявляє позицію з вищим значенням придатності, ГНП замінюється поточною позицією цієї бджоли (рис. 2.5).

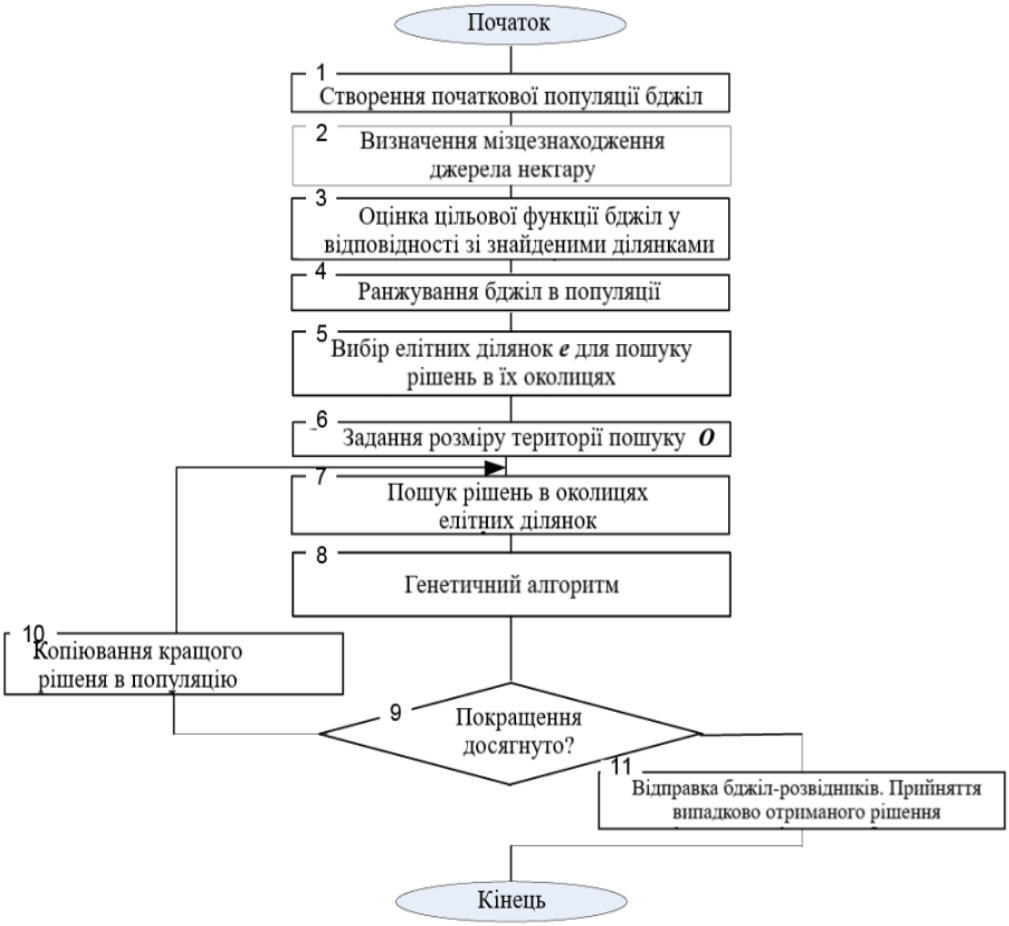


Рисунок 2.5 – Загальна схема алгоритму АВС

Нижче наведено опис алгоритму пошуку з застосуванням методу Бджолиного рою [16–19]:

Крок 1. Визначення області пошуку оптимальних значень та встановлення параметрів алгоритму, наприклад, розмір рою.

Крок 2. Розміщення певним чином бджіл в області пошуку та призначення початкових швидкостей їх руху, у найпростішому випадку використовується метод випадкового перебору:

$$X_i = rand\left(G\left(X\right)\right),\ i = 1, \ldots, n. \tag{2.5}$$

Крок 3. Для кожної бджоли в рої виконується переміщення перейдіть на нову позицію на основі її поточної позиції та швидкості. Проводиться перевірка чи не виходить бджола за межі досліджуваної території та виконуються необхідні обмежувальні дії.

Крок 4. Обчислюється значення цільової функції для кожної бджоли в її новій позиції. Це значення порівнюється зі значенням ПНП бджоли і якщо це необхідно, оновлюється ПНП. Порівнюємо це значення з глобальною найкращою позицією рою та оновлюємо ГНП, якщо необхідно.

Крок 5. Обчислюється нова швидкість руху для кожної бджоли на основі рівняння:

$$v_n^{i+1} = w \cdot v_n^i + c_1 \cdot \psi_1 \cdot \left(p_n - x_n\right) + c_2 \cdot \psi_2 \cdot \left(g_n - x_n\right), \tag{2.6}$$

де v_n^i – швидкість бджоли по виміру n на i-й ітерації;

w – ітераційна вага, це число знаходиться в інтервалі [0, 1] і відображає, в якій мірі частинка зберігає свою початкову швидкість;

p_n, g_n – значення координати n для ПНП бджоли та для ГНП всього рою відповідно;

Ψ_1, Ψ_2 – випадкова величина в діапазоні [-1, 1];

c_1, c_2 – статичні вагові коефіцієнти, які визначають тягу до власної ПНП та до ГНП рою відповідно;

c_1 визначає, який вплив на агента чинить її пам’ять про ПНП, а c_2 – який вплив на окрему бджолу мають інші члени рою [4, 16, 20]. Збільшення c_1 передбачає дослідження простору рішень шляхом руху кожної бджоли в напрямку своєї ПНП; збільшення значення c_2 передбачає дослідження ймовірного глобального максимуму.

Дані коефіцієнти розглядаються як пізнавальний і соціальний фактори.

Крок 6. Проводиться перевірка критерію зупинки, якщо пошук ще не завершено, повертаємося до Кроку 3 (рис. 2.6).

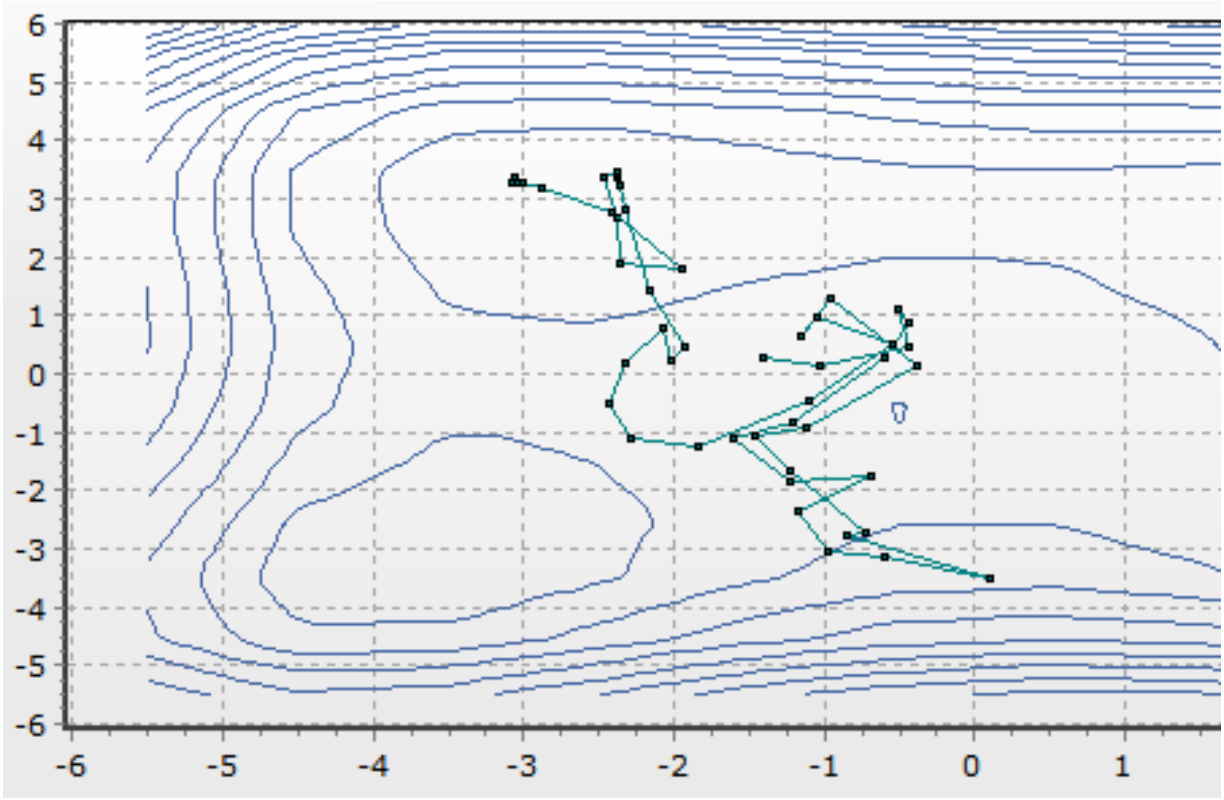


Рисунок 2.6 – Приклад траєкторії польоту бджоли в процесі пошуку максимуму функції Гіммельблау

Підводячи підсумок, рух кожної бджоли представляє собою компроміс між рухом до передбачуваного глобального максимуму та рухом до виявленого локального максимуму.

На рисунку 2.6 показаний приклад траєкторії пошуку однією бджолою максимуму функції Гіммельблау:

$$z = 500 - (x^2+y-11)^2+ (x+ y^2-7)^2. \quad (2.7)$$

Траєкторія виглядає як складний шлях, що нагадує рух частинки в броунівському русі.

2.4 Механізм оптимізації маршрутів на основі методу Зозулиного пошуку

Алгоритм «Зозулин пошук» (Cuckoo Search) – це евристичний метод оптимізації, інспірований поведінкою зозуль при пошуку місця для гнізда. Він заснований на рандомізації та співпраці між окремими «зозулями» (агентами алгоритму). Основна ідея полягає в тому, щоб кожен «зозуля» (агент) ніс у собі можливе рішення задачі (яке може бути представлене у вигляді вектора параметрів), і рухався в просторі можливих рішень, намагаючись знайти оптимальний максимум (або мінімум) функції від цих параметрів [8, 21] (рис. 2.7 [21]).

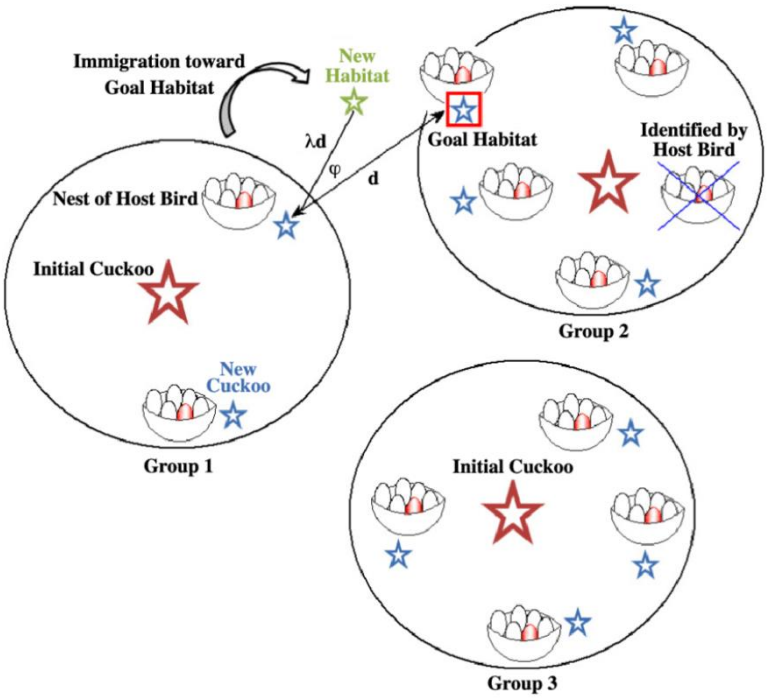


Рисунок 2.7 – Схематичне зображення процесу виконання алгоритму CS

Алгоритм CS продемонстрував свою універсальність, знайшовши застосування в різноманітних задачах оптимізації. Він продемонстрував надзвичайний досвід у оптимізації функцій, точному налаштуванні параметрів, покращенні обробки зображень, підтримці машинного навчання та оптимізації процесів вибору функцій.

Унікальна здатність цього алгоритму підтримувати гармонійну рівновагу між розвідкою та експлуатацією в поєднанні з застосуванням польоту Леві та тактикою стратегічного залишення призводить до високоефективного обходу простору пошуку, що зрештою призводить до знаходження оптимальних або майже оптимальних рішень.

Крім того, метод надає низку переконливих переваг. Його впровадження надзвичайно просте, а його адаптивна природа дозволяє легко інтегруватись у різні проблемні області. Притаманний алгоритму дослідницький характер наділяє його вражаючими можливостями глобального пошуку, тоді як аспект локального пошуку максимізує використання перспективних регіонів. Більше того, конфігурації параметрів алгоритму є гнучкими, що дозволяє проводити тонке налаштування для досягнення ще більшої продуктивності, тим самим підвищуючи його загальну ефективність (рис. 2.8 [22]).

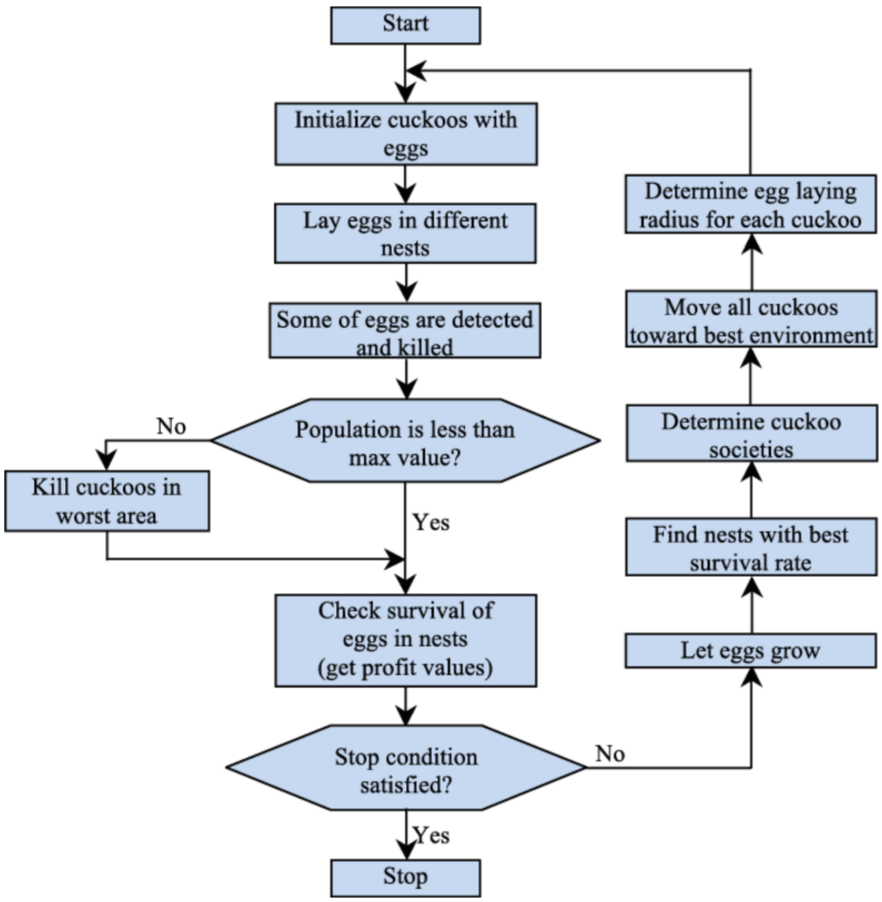


Рисунок 2.8 – Блок-схема структури алгоритму CS

Алгоритм Зозулиного пошуку ретельно дотримується структурованої процедури, спрямованої на поступове підвищення якості рішення. Нижче наведено ключові етапи, які керують алгоритмом.

Ініціалізація: початкова популяція «зозуль» генерується випадковим чином або за допомогою певного правила. Початкова популяція «зозуль» створюється з обраною кількістю агентів, кожен з яких представляє можливе рішення задачі. Їх початкові позиції можуть генеруватися випадково або за допомогою спеціальних методів [5, 23].

Пошук та оцінка: кожна «зозуля» обирає деяке рішення (позицію) та обчислює відповідне значення цільової функції для цього рішення (функції, яку потрібно оптимізувати). Ця функція може визначати, наскільки гарним є дане рішення. Результати оцінки зберігаються для подальшого використання.

Політ Леві: черпаючи натхнення з феномена польоту Леві, який спостерігався у різних видів тварин, алгоритм генерує нові варіанти рішення для кожної зозулі. Польоти Леві охоплюють стохастичну траєкторію, що характеризується важкими кроками, що дозволяє глибоко досліджувати широкий простір пошуку [9, 14, 24].

Репродукція: потім деякі «зозулі» (агенти) вибираються для репродукції. Вибір може залежати від значення функції (чим краще рішення, тим більша ймовірність вибору), інших критеріїв або випадкового вибору.

Пошук вижившиx: зозулі-нащадки можуть замінити частину початкової популяції. Це дозволяє впроваджувати нові рішення в популяцію та зберігати найкращі знайдені рішення.

Оновлення популяції: періодично популяція «зозуль» оновлюється, включаючи покращені або нові рішення, знайдені на попередніх етапах алгоритму. Це може включати в себе видалення менш ефективних рішень. Ця операція може залежати від конкретних правил або стратегій.

Умова завершення: алгоритм продовжує виконуватися доти, доки не виконується певна умова завершення. Ці звичайні критерії зупинки можуть включати досягнення максимального порогу ітерації, досягнення попередньо

визначеного контрольного показника придатності або перевищення попередньо встановленого обмеження часу.

Схема алгоритму CS:

– проводиться ініціалізація популяції $S = s_i$, $i \in [1: |S|]$ з $|S|$ гнізд та зозулі, таким чином, визначаємо вектори X_0^i і вектор початкової позиції зозулі X_c;

– виконуючи польоти Леві, визначаємо нове положення зозулі:

$$X_c^{i+1} = X_c^i + V \otimes L_{|X|}(\lambda), (i=1,2,\ldots,n), \quad (2.8)$$

де $V = (v_j , j \in [1: |X|])$ – вектор розміру кроків для відповідних компонентів вектора X;

$L_{|X|}(\lambda) - (|X| \times 1)$ – випадковий вектор незалежних дійсних випадкових чисел, розподілених відповідно до закону Леві [25–28];

– випадковим чином обирається гніздо s_j , $j \in [1: |S|]$ і, якщо $\varphi(X_c) > \varphi(X_j)$, замінюємо яйце в цьому гнізді на яйце зозулі, тобто вважаємо $X_j = X_c$;

– з ймовірністю ξ_a видаляється певна кількість випадково вибраних гірших гнізд з популяції (включаючи, можливо, гніздо s_j) і дотримуючись правил кроку 1 будується така ж кількість нових гнізд.

Початкові положення гнізд і зозулі вважаємо випадковими, рівномірно розподіленими в деякому гіперпаралелепіпеді. Як правило, усі компоненти вектора V вважаються однаковими і рівними v, де величина v пов’язана з масштабами області пошуку. Основними вільними параметрами алгоритму є константи, що визначають траєкторію польотів Леві, та імовірність вилучення гнізда ξ_a [25–28].

3 ДОСЛІДЖЕННЯ МЕТОДІВ ОПТИМІЗАЦІЇ МАРШРУТІВ У ЛОГІСТИЧНИХ СИСТЕМАХ

3.1 Вибір інструментальних засобів для реалізації методів оптимізації маршрутів у логістичних системах

Для реалізації досліджуваних алгоритмів для задачі оптимізації маршрутизації транспортних засобів було розроблено програмне забезпечення з використанням мови програмування Python.

Python – це високорівнева, об’єктно-орієнтовна, крос-платформна, інтерпретована мова програмування, відома своєю інтерактивністю, модульністю, динамічністю, портативністю. Вибір цієї мови виправданий перевагами, описаними нижче.

Масштабні бібліотеки – в Python інтегровані великі стандартні бібліотеки, які охоплюють більшість поширених завдань програмування, що дає можливість значно зменшити довжину коду та полегшити використання мови.

Гнучкість – Python легко взаємодіє з іншими популярними мовами програмування, такими як C, C++, Java. Він також може бути вбудований в інші мови програмування, дозволяючи включати функції Python за допомогою інших мов програмування.

Портативність – якщо код не містить унікальних системних функцій, код Python можна легко перенести на іншу платформу.

Продуктивність – простота Python, бібліотечні функції, інтеграція процесів, модульні системи тестування та розширені можливості керування забезпечують значні переваги для ефективності розробника, збільшуючи швидкість програми та продуктивність порівняно з іншими мовами. Розробники можуть зосередитися на унікальному коді для конкретного проєкту, використовуючи наявні бібліотеки [29].

Простота – оскільки більшість функцій інкапсюльовано в бібліотеках, програмування, читання та розуміння коду Python є дуже простими. Відсутність фігурних дужок дещо спрощує читання коду та процес дебагу.

Інтерпретовність – у Python оператори виконуються один за одним, що спрощує налагодження порівняно зі скомпільованими мовами.

IoT – Python використовується для Інтернету речей, який стає все більш актуальним.

Мультифункціональність та *інтегрованість* – Python можна застосовувати на різних платформах для web, mobile, front-end та back-end розробки. Python дозволяє розробку скриптів для конкретних сценаріїв, уникаючи розробки глобального програмного забезпечення. Окрім цього це найпопулярніша на даний час мова для data science та machine learning [29].

Велике комьюніті – велика спільнота програмістів, що використовують Python, підтримка програмістів на етапах розробки, наявність експертних тематичних джерел та постійне вдосконалення й розробка додаткових бібліотек.

Довіра у великих проєктах – останнім часом Python обирають такі гіганти як Google, YouTube, Dropbox, Yahoo. Він використовується в освіті, науці, охороні здоров'я, управління фінансами та інших важливих галузях.

Бібліотеки, що використовуються у проєкті, зображені на рисунку 3.1.

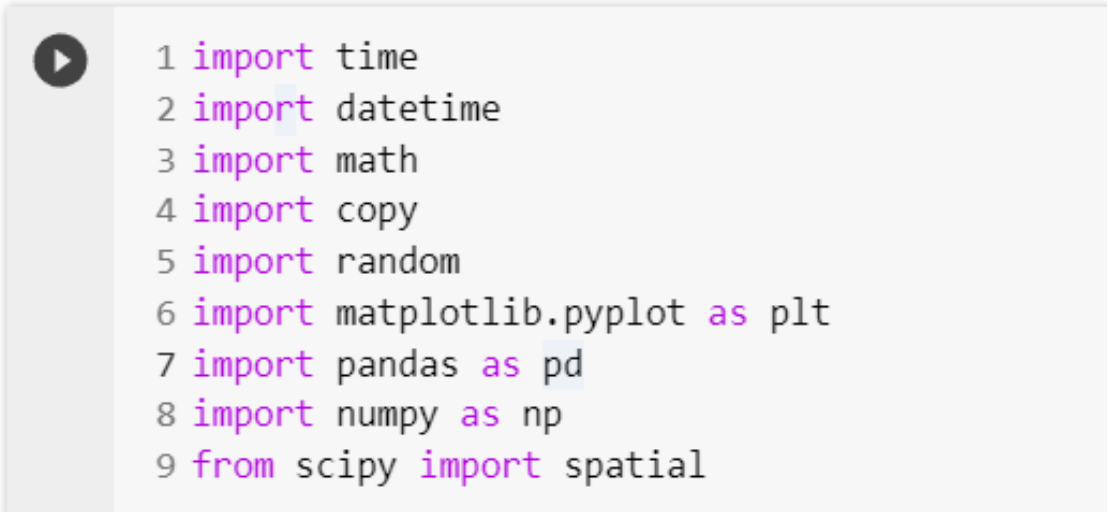


Рисунок 3.1 – Бібліотеки, що використовуються у проєкті

Призначення використаних бібліотек:

– *matplotlib* – бібліотека для візуалізації даних, як правило, у формі графіків, графів і діаграм;

– *pandas* – бібліотека для маніпулювання даними та їхнього аналізу;

– *numpy* – бібліотека для роботи з великими багатовимірними масивами та матрицями;

– *scipy* – бібліотека призначена для виконання наукових та інженерних розрахунків;

– *random* – модуль, що дозволяє використовувати випадкові значення;

– *math* – модуль із вбудованими математичними операціями;

– *copy* – модуль, що дозволяє копіювати об’єкти (за замовчуванням в Python при копіюванні створюється не копія, а посилання на цільовий об’єкт);

– *time* – модуль, що дозволяє використовувати різні функції, пов’язанні з часом;

– *datetime* – модуль, що забезпечує класи для маніпуляцій з датами та часом.

В якості середовища розробки програми було обрано Google Colaboratory. Це безкоштовна хмарна платформа, запропонована Google, яка надає дослідникам, студентам і всім бажаючим зручний спосіб проводити експерименти з машинного та глибокого навчання. Він створений на основі Jupyter Notebook.

Основний інтерфейс середовища зображено на рисунку 3.2. Вибір припав на даний сервіс через його особливості та переваги, описані нижче.

Google Colab пропонує безкоштовний доступ до ресурсів хмарних обчислень, користувачі Colab можуть отримати доступ до Google Cloud Services і API, які можуть бути корисними для таких завдань, як зберігання й обробка даних у хмарі. Він надає віртуальну машину з прискоренням GPU, що особливо цінно для завдань машинного навчання, які потребують значної обчислювальної потужності.

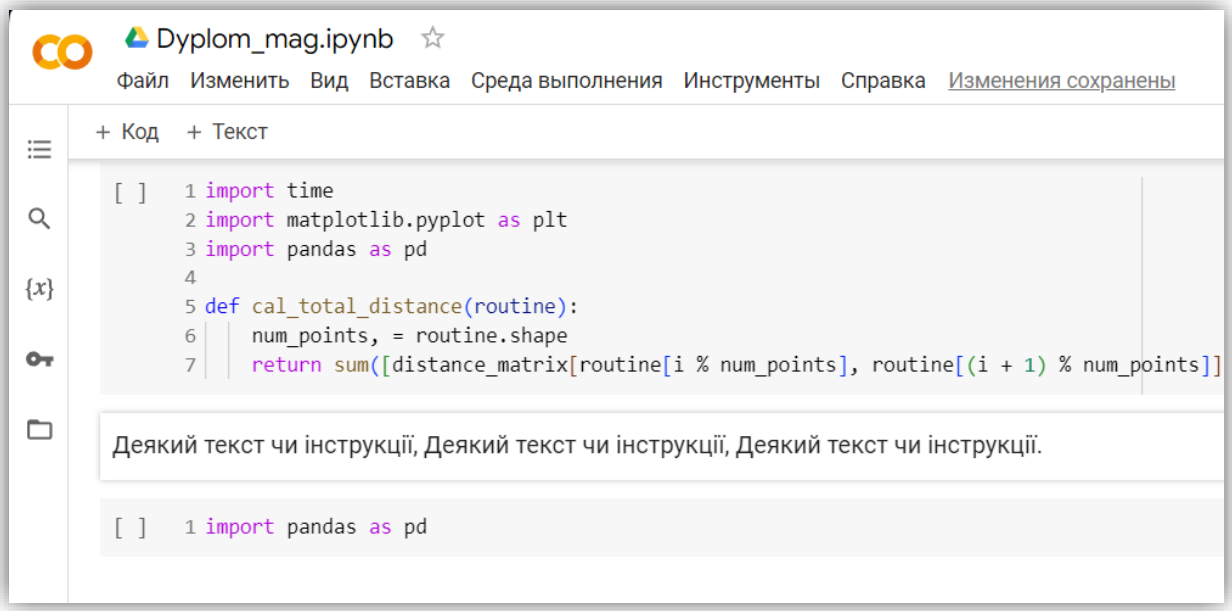


Рисунок 3.2 – Інтерфейс Google Colaboratory

Також Colab інтегрується з Jupyter Notebooks, що полегшує створення, редагування та обмін інтерактивними документами, які поєднують код, візуалізації та пояснювальний текст. Блокнотами, створеними в Colab, можна легко ділитися з колегами або ширшою спільнотою.

Користувачі можуть зберігати свої блокноти та набори даних безпосередньо на Google Drive, забезпечуючи надійний доступ до файлів і легку співпрацю з іншими.

Colab постачається з попередньо встановленими бібліотеками для аналізу даних, машинного та глибокого навчання, такими як TensorFlow, Keras, PyTorch, scikit-learn та інші.

Окрім того, Colab дозволяє користувачам зберігати різні версії блокнота та відстежувати зміни за допомогою інтеграції з Git, що значно полегшує співпрацю та керування кодом.

Однак, як і будь яке середовище, Google Colaboratory має свої недоліки:

– *обмежена тривалість сеансу*: сеанси Colab мають максимальний час роботи (зазвичай близько 12 годин), якого може бути недостатньо для тривалих, ресурсномістких завдань;

– *обмеження ресурсів*: хоча він надає безкоштовні ресурси GPU, вони не такі потужні, як виділені GPU або TPU, доступні на платних хмарних платформах;

– *залежність від Інтернету*: Colab вимагає стабільного підключення до Інтернету, що робить його непридатним для роботи в автономному режимі;

– *перемінна продуктивність*: оскільки Colab є безкоштовною службою, продуктивність може змінюватися залежно від попиту, а іноді доступні ресурси можуть бути обмежені.

На щастя, подібні недоліки майже не мають впливу на результати дослідження за темою кваліфікаційної роботи.

Google Colab – це потужний інструмент, який надає доступ до високопродуктивних обчислювальних ресурсів для завдань Data Science та Machine Learning. Він пропонує широкий спектр функцій і особливо привабливий для тих, кому потрібне безкоштовне хмарне рішення. Незважаючи на певні обмеження, Colab залишається цінним інструментом як для початківців, так і для експертів у цій галузі, а його особливості дозволяють значно спростити візуалізацію результатів дослідження.

3.2 Етапи програмної реалізації методів оптимізації маршрутів у логістичних системах

Програмна реалізація методів оптимізації маршрутів у логістичних системах включає створення комп’ютерних моделей та розробку алгоритмів для мурашиного алгоритму, алгоритму бджолиної колонії та алгоритму зозулиного пошуку. Написання коду для кожного методу, валідація та тестування реалізації в реальних або симульованих умовах дозволяють перевірити їхню ефективність та адаптувати їх для оптимізації маршрутів у логістичних системах.

Етап програмної реалізації є важливим етапом у дослідженні та порівняльному аналізі мурашиного алгоритму, алгоритму бджолиної колонії та алгоритму зозулиного пошуку. Перетворення математичних моделей цих алгоритмів у програмний код дозволяє реалізувати їх у вигляді функціональних інструментів для оптимізації транспортних маршрутів. Під час програмної реалізації увага зосереджується на реалізації ключових аспектів кожного алгоритму, таких як механізми прийняття рішень, пошукові стратегії та адаптивність до змін у вхідних даних.

Кожен з цих алгоритмів має унікальні особливості та параметри, що вимагають детальної обробки та уточнення під час програмної імплементації. Приділяється особлива увага оптимізації реалізації для забезпечення ефективності та швидкодії в обчисленнях при рішенні задачі оптимізації маршрутів.

Після написання коду проводиться ретельне тестування реалізації кожного алгоритму за допомогою різноманітних тестових випадків, щоб оцінити його точність, здатність до знаходження оптимальних рішень та поведінку в різних умовах.

Детальна програмна реалізація та наступний її аналіз допомагають отримати унікальний погляд на взаємодію кожного з алгоритмів з вихідними даними, розв’язуванням конкретних завдань логістики та їхнім потенціалом для оптимізації маршрутів у логістичних системах.

Для початку, необхідно створити площину вузлів, на яких будуть тестуватися алгоритми. Граф, на якому буде відбуватися пошук оптимальних маршрутів, буде генеруватися випадковим чином. Першочергово генеруються координати точок. Наступний крок передбачає обчислення матриці відстані на основі цих згенерованих координат. Це обчислення виконується за допомогою модуля «spatial» із бібліотеки «scipy». Ця матриця обчислює відстань від кожної вершини до кожної іншої вершини на графі.

Фрагмент коду, який відповідає за створення графа та матриці відстаней показано на рисунку 3.3.

```python
import numpy as np
import random
from scipy import spatial

num_points = 30 # кількість вершин
points_coordinate = np.random.rand(num_points, 2)  # генерація рандомних вершин
print("Координати вершин:\n", points_coordinate[:10], "\n")

# розрахунок матриці відстаней між вершинами
distance_matrix = spatial.distance.cdist(points_coordinate, points_coordinate, metric='euclidean')
print("Матриця відстаней:\n", distance_matrix[:2])
```

```
Координати вершин:
 [[0.06021492 0.5843609 ]
 [0.13898    0.8841521 ]
 [0.72637304 0.16604735]
 [0.49284902 0.57058883]
 [0.4709509  0.5907544 ]
 [0.163576   0.46994574]
 [0.25722282 0.18834063]
 [0.84920104 0.46174266]
 [0.03188854 0.49012598]
 [0.92162655 0.57567853]]

Матриця відстаней:
 [[0.         0.30996564 0.78660847 0.43285325 0.41078575 0.15418931
  0.44231683 0.79845748 0.09840023 0.86145539 0.49475085 0.27523929
  0.657135   0.9514092  0.82018105 0.31370309 0.91339502 0.5122782
  0.55901704 0.72591655 0.19675123 1.03120968 0.13312561 0.41481816
  0.68999279 0.51814061 0.58740204 0.12771849 0.59413613 0.71679399]
 [0.30996564 0.         0.92774189 0.47280568 0.44304276 0.41493599
  0.70578678 0.82634355 0.40831993 0.84124406 0.33116811 0.13866985
  0.60550735 0.84470018 0.6593842  0.55844949 0.75256711 0.47889745
  0.6934005  0.53038851 0.25763963 1.15716106 0.17704174 0.68378195
  0.87310337 0.44999898 0.8342367  0.36828631 0.48254954 0.60149685]]
```

Рисунок 3.3 – Генерація графа

Нижче перераховано створені класи, в межах яких реалізовані основні частини досліджуваних алгоритмів, а також дано загальний опис основних функцій, що їм належать та етапи їх роботи.

Клас *AntColonyOptimization*:

– *initialize_pheromone_matrix* – функція, що ініціалізує початкову матрицю феромонів на ребрах графу з випадковими значеннями або початковими значеннями;

– *update_pheromone_matrix* – функція, що оновлює значення феромонів на ребрах графу після кожної ітерації відповідно до значень цільової функції та враховуючи внесок кожної мурашки, а також використовує механізм випаровування феромонів;

– *construct_solutions* – створює рішення (маршрути) для кожної мурашки, керуючись вибором ребра на основі значень феромонів та інших параметрів.

Клас *ArtificialBeeColony*:

– *initialize_population* – функція для ініціалізації початкової популяції бджіл з випадковими або заданими значеннями параметрів;

– *employed_bees_phase* – функція, де кожна зайнята бджола досліджує свою зону та оновлює свої позиції;

– *onlooker_bees_phase* – функція, де бджоли аналізують інформацію, отриману від зайнятих бджіл та оцінюють їхні позиції;

– *scout_bees_phase* – функція, яка дозволяє популяції виходити з локальних мінімумів шляхом дослідження нових областей, в ній реалізовано відкидання старих гірших позиції та оновлення їх новими кращими.

Клас *CuckooSearch*:

– *initialize_nests* – функція, що ініціалізує початкові позиції зозуль (гнізда) з випадковими або початковими значеннями;

– *levy_flight* – функція, в якій реалізована генерація випадкових кроків для зміни позицій зозуль з використанням розподілу Леві;

– *get_best_nests* – функція в якій реалізовано процес обміну гніздами між зозулями для покращення рішення;

– *abandon_nests* – функція, яка застосовується для видалення гнізда з низькими значеннями фітнес-функції.

В алгоритмі ACO є два вирішальні фактори: феромон τ і видимість η. Феромон τ відноситься до інформації, що залишається на кожному шляху, яким проходять мурахи. Видимість η позначає зворотну відстань між вузлами. Імовірність P, що мурахи оберуть певний маршрут складається з добутку феромону τ, видимості η та їх суми. При найкоротшому шляху, високому рівні феромону τ, а також високій видимості η ймовірність P стає вищою.

Отже, у рядках 26-38 відбувається підрахунок ймовірності P (рис. 3.4). Потім рядок 45 вибирає послідовність найкоротших шляхів за допомогою ймовірності P. Нарешті, рядки 50-60 оновлюють феромон з коефіцієнтом випаровування ρ (рис. 3.5).

```
24        for i in range(self.max_iter):
25            # ймовірність переходу без нормалізації
26            prob_matrix = (self.Tau ** self.alpha) * (self.prob_matrix_distance) ** self.beta
27            for j in range(self.size_pop):  # для кожної мурахи
28                # точка початку шляху
29                self.Table[j, 0] = 0
30                for k in range(self.n_dim - 1):  # кожна вершина, яку проходять мурахи
31                    # точка, яка була пройдена і не може бути пройдена повторно
32                    taboo_set = set(self.Table[j, :k + 1])
33                    # список дозволених вершин, з яких буде робитися вибір
34                    allow_list = list(set(range(self.n_dim)) - taboo_set)
35                    prob = prob_matrix[self.Table[j, k], allow_list]
36                    prob = prob / prob.sum() # нормалізація ймовірності
37                    next_point = np.random.choice(allow_list, size=1, p=prob)[0]
38                    self.Table[j, k + 1] = next_point
39
```

Рисунок 3.4 – Фрагмент програмної реалізації алгоритму ACO

```
43                # фіксація кращого рішення
44                index_best = y.argmin()
45                x_best, y_best = self.Table[index_best, :].copy(), y[index_best].copy()
46                self.generation_best_X.append(x_best)
47                self.generation_best_Y.append(y_best)
48
49                # підрахунок феромону, який буде додано до ребра
50                delta_tau = np.zeros((self.n_dim, self.n_dim))
51                for j in range(self.size_pop):  # для кожного мурахи
52                    for k in range(self.n_dim - 1):  # для кожної вершини
53                        # мурахи перебираються з вершини n1 в вершину n2
54                        n1, n2 = self.Table[j, k], self.Table[j, k + 1]
55                        delta_tau[n1, n2] += 1 / y[j]  # нанесення феромону
56                    # мурахи повзуть від останньої вершини назад до першої
57                    n1, n2 = self.Table[j, self.n_dim - 1], self.Table[j, 0]
58                    delta_tau[n1, n2] += 1 / y[j]  # нанесення феромону
59
60                self.Tau = (1 - self.rho) * self.Tau + delta_tau
61
62            best_generation = np.array(self.generation_best_Y).argmin()
63            self.best_x = self.generation_best_X[best_generation]
64            self.best_y = self.generation_best_Y[best_generation]
65            return self.best_x, self.best_y
```

Рисунок 3.5 – Фрагмент програмної реалізації алгоритму ACO

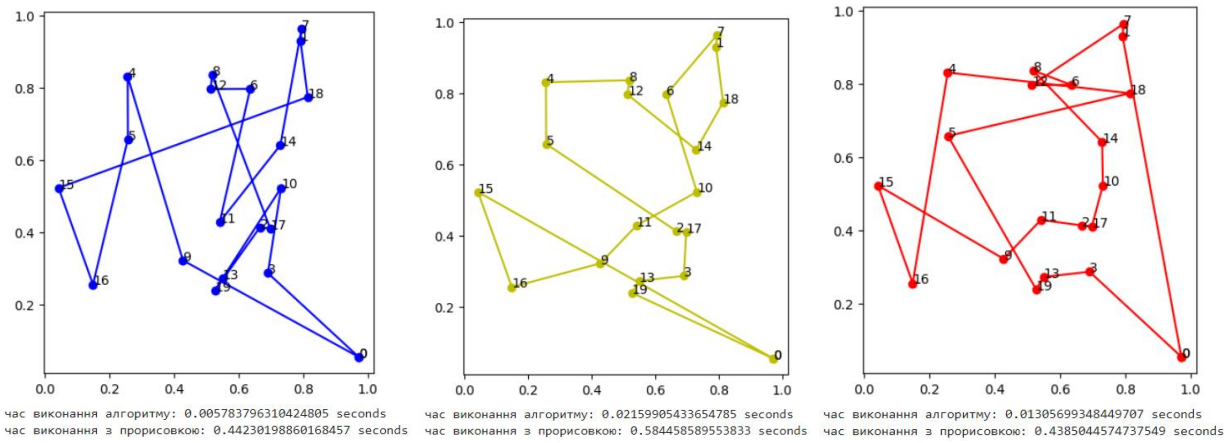
Також одним з вкрай необхідних елементів реалізації є проста, але дуже важлива функція розрахунку загальної довжини шляху, яка адаптована та використовується в усіх трьох алгоритмах (рис. 3.6).

```
4
5 # розрахунок довжини шляху
6 def cal_total_distance(routine):
7     num_points, = routine.shape
8     return sum([distance_matrix[routine[i % num_points], routine[(i + 1) % num_points]] for i in range(num_points)])
9
```

Рисунок 3.6 – Функція розрахунку загальної довжини шляху

Щоб переконатися у коректній роботі реалізованих алгоритмів проведено запуск програми на графі з 20 вузлів.

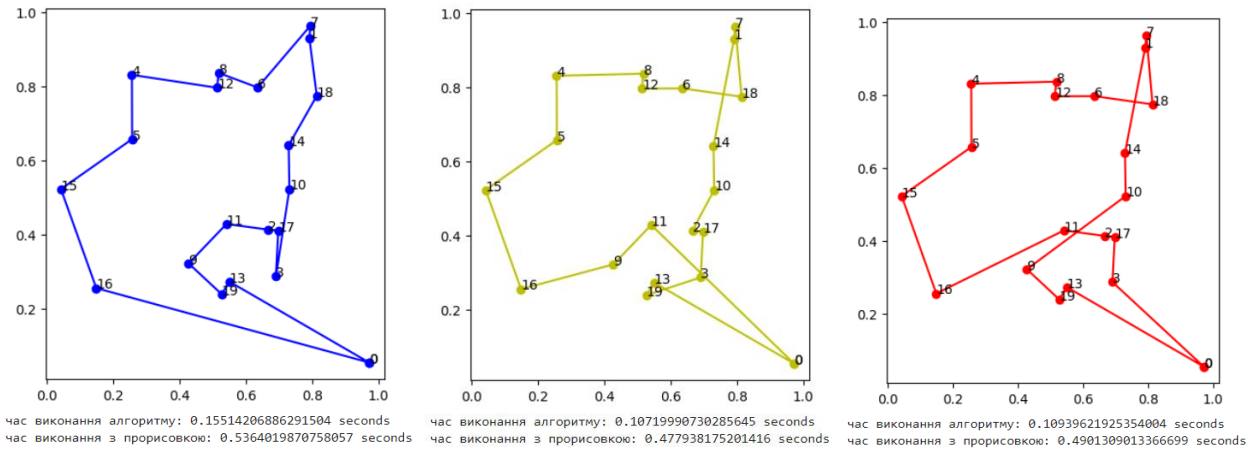
Для наглядності на рисунку 3.7 показано шляхи, отримані в результаті проходження алгоритмів після однієї ітерації, а отже вони є цілком випадковими.



а) б) в)

Рисунок 3.7 – Рішення, знайдені на першій ітерації:
а) алгоритм ACO; б) алгоритм ABC; в) алгоритм CS

Тепер, щоб пересвідчитися в тому, що алгоритми реалізовано правильно і вони працюють, збільшимо кількість ітерацій до 10 і запустимо програму на тому самому графі. Маршрути знайдені в результаті проходження показані на рисунку 3.8.



а) б) в)

Рисунок 3.8 – Рішення, знайдені на десятій ітерації:

а) алгоритм ACO; б) алгоритм ABC; в) алгоритм CS

На рисунку 3.8 можна побачити, що після 10 ітерацій маршрути далеко не оптимальні, проте значно кращі ніж були спочатку.

Таким чином, можна впевнено зазначити, що реалізація алгоритмів ACO, ABC та CS пройшла успішно і вони коректно виконують поставлену перед ними задачу оптимізації маршрутів у логістичних системах.

3.3 Тестування розроблених застосунків та аналіз результатів

У рамках цього підрозділу потрібно провести порівняльну характеристику розроблених алгоритмів, дослідити ефективність роботи кожного з них на прикладі задачі глобальної оптимізації.

Для початку визначимо оптимальний розмір популяції агентів ACO, знайшовши найменший час виконання алгоритму на графі з 30 вершин за 50 ітерацій. Вплив розміру популяції на час виконання алгоритму мурашиної колонії на графі з 30 вершин наведений у таблиці 3.1.

Таблиця 3.1 – Вплив розміру популяції на час виконання алгоритму ACO

Розмір популяції	**Збіжність алгоритму, (%)**	**Час виконання, (с)**
5	62	0,44
10	69	0,87
20	72	1,81
30	80	3,25
40	98	4,89
50	100	5,14
75	98	8,06
100	100	10,41
120	100	13,01
150	100	16,01
200	100	20,13

Дослідження здійснювалося на прикладі пошуку найкращого співвідношення часу виконання та збіжності алгоритму, найбільш ефективні значення представлені в таблиці 3.1. З неї видно, що оптимальний розмір популяції – 50 особин. особин. Тепер дослідимо залежність часу виконання та збіжність алгоритму ACO від кількості ітерацій, при популяції у 50 агентів та 30 вузлах (табл. 3.2).

Таблиця 3.2 – Вплив кількості ітерацій на збіжність і час виконання алгоритму ACO

Кількість ітерацій	**Збіжність алгоритму, (%)**	**Час виконання, (с)**
1	**2**	**3**
5	40	0,51

Продовження таблиці 3.2

1	2	3
10	73	0,73
20	67	1,87
30	78	2,90
50	97	4,34
75	99	6,51
100	99	7,57
120	100	9,98
150	100	12,31

Як видно з таблиці 3.2, збіжність алгоритму, як і час виконання зростають з кількістю ітерацій.

Дослідження роботи алгоритму ACO показали, що ефективність алгоритму залежить від обраних параметрів, чим краще вони підібрані, тим точнішим є результат і меншим є час виконання алгоритму [30–40].

Тепер необхідно провести аналогічні експерименти з реалізованими алгоритмами ABC та CS. Для коректних результатів порівняльного аналізу всі розрахунку проводитимуться на графі в 30 вузлів.

Вплив розміру популяції на час виконання ABC наведений у таблиці 3.3.

Таблиця 3.3 – Вплив розміру популяції на час виконання алгоритму ABC

Розмір популяції	Збіжність алгоритму, (%)	Час виконання, (с)
1	2	3
5	64	0,41
10	71	0,88
20	76	1,83

Продовження таблиці 3.3

1	**2**	**3**
30	84	2,66
40	100	3,79
50	100	4,34
75	100	6,88
100	97	8,64
120	99	11,13
150	100	14,75
200	100	19,34

З таблиці 3.3 видно, що оптимальний розмір популяції при даних умовах – 40 особин.

Тепер дослідимо залежність часу виконання та збіжність алгоритму ABC від кількості ітерацій, при популяції у 40 агентів та 30 вузлах (табл. 3.4).

Таблиця 3.4 – Вплив кількості ітерацій на збіжність і час виконання алгоритму ABC

Кількість ітерацій	**Збіжність алгоритму, (%)**	**Час виконання, (с)**
5	51	0,37
10	64	0,69
20	74	1,54
30	84	2,24
50	99	3,59
75	99	6,11
100	100	7,13
120	100	8,84
150	100	11,56

Як видно з таблиці 3.4, збіжність алгоритму, як і час виконання зростають з кількістю ітерацій.

Дослідження роботи алгоритму ABC показали, що ефективність алгоритму залежить від обраних параметрів, чим краще вони підібрані, тим точнішим є результат і меншим є час виконання алгоритму.

Вплив розміру популяції на час виконання алгоритму CS наведений у таблиці 3.5.

Таблиця 3.5 – Вплив розміру популяції на час виконання алгоритму CS

Розмір популяції	**Збіжність алгоритму, (%)**	**Час виконання, (с)**
5	58	0,42
10	64	0,90
20	72	1,82
30	81	2,65
40	100	3,60
50	100	4,42
75	100	8,15
100	100	9,18
120	96	12,02
150	100	14,68
200	100	19,98

З таблиці 3.5 видно, що оптимальний розмір популяції – 40 особин.

Тепер проведемо дослідження залежності часу виконання та збіжність алгоритму CS від кількості ітерацій, при популяції у 40 агентів на графі з 30 вузлів (табл. 3.6).

Таблиця 3.6 – Вплив кількості ітерацій на збіжність і час виконання алгоритму CS

Кількість ітерацій	**Збіжність алгоритму, (%)**	**Час виконання, (с)**
5	44	0,41
10	49	0,74
20	68	1,57
30	74	2,49
50	81	4,02
75	93	6,63
100	98	7,36
120	99	9,35
150	100	12,16

Як видно з таблиці 3.6, збіжність алгоритму, як і час виконання зростають з кількістю ітерацій.

Дослідження роботи алгоритму CS показали, що ефективність алгоритму залежить від обраних параметрів, чим краще вони підібрані, тим точнішим є результат і меншим є час виконання алгоритму [30–40].

Для порівняльного аналізу ефективності роботи алгоритмів проведено порівняння зведених найкращих результатів часу виконання та кількості ітерацій необхідних для досягнення алгоритмами повної збіжності.

Аналіз ефективності досліджуваних алгоритмів проведено на 2 ситуаціях, у першому випадку дослідження проводилось на 30 вузлах (що є відносно невеликою кількістю), а в другому – на 300 (такий граф можна вважати досить великим). Зведені результати досліджень представлені в таблицях 3.7 і 3.8. З їх допомогою нарешті можна провести порівняльний аналіз та з отриманих результатів сформулювати висновки. На рисунку 3.9 показано оптимальний маршрут, знайдений на 30-вузловому графі, на якому проводилися експерименти, за допомогою алгоритму ABC.

Таблиця 3.7 – Результати роботи алгоритмів на 30 вузлах

Алгоритм	**Кількість ітерацій**	**Час виконання, (с)**
ACO	150	12,31
ABC	120	8,85
CS	150	12,06

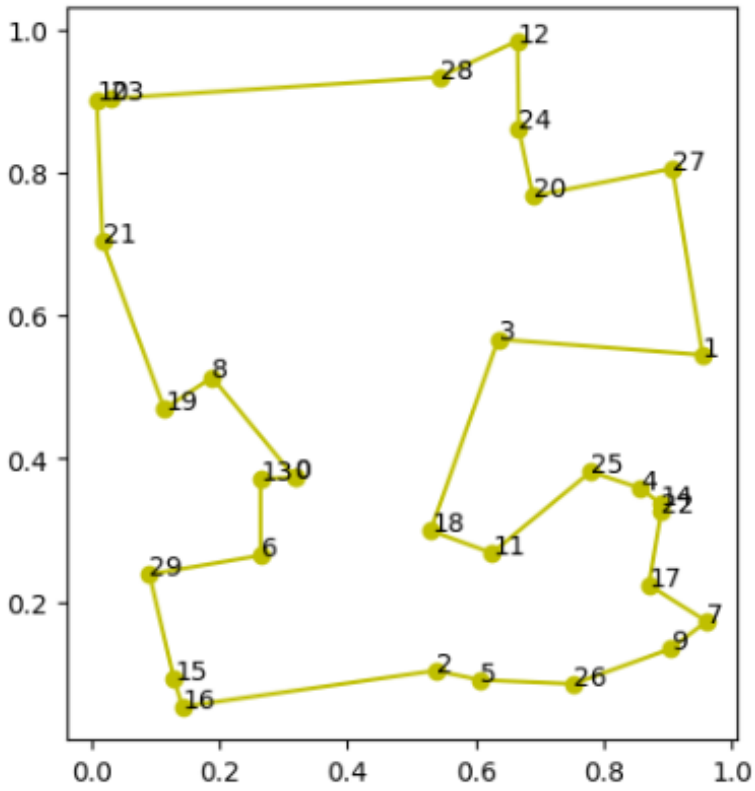


Рисунок 3.9 – Оптимальний маршрут для 30 вузлів

На рисунку 3.10 показано оптимальний маршрут, знайдений на 300-вузловому графі, на якому проводилися дослідження, за допомогою алгоритму ACO.

Таблиця 3.8 – Результати роботи алгоритмів на 300 вузлах

Алгоритм	**Кількість ітерацій**	**Час виконання, (хв.)**
ACO	500	21,99
ABC	450	27,78
CS	500	29,26

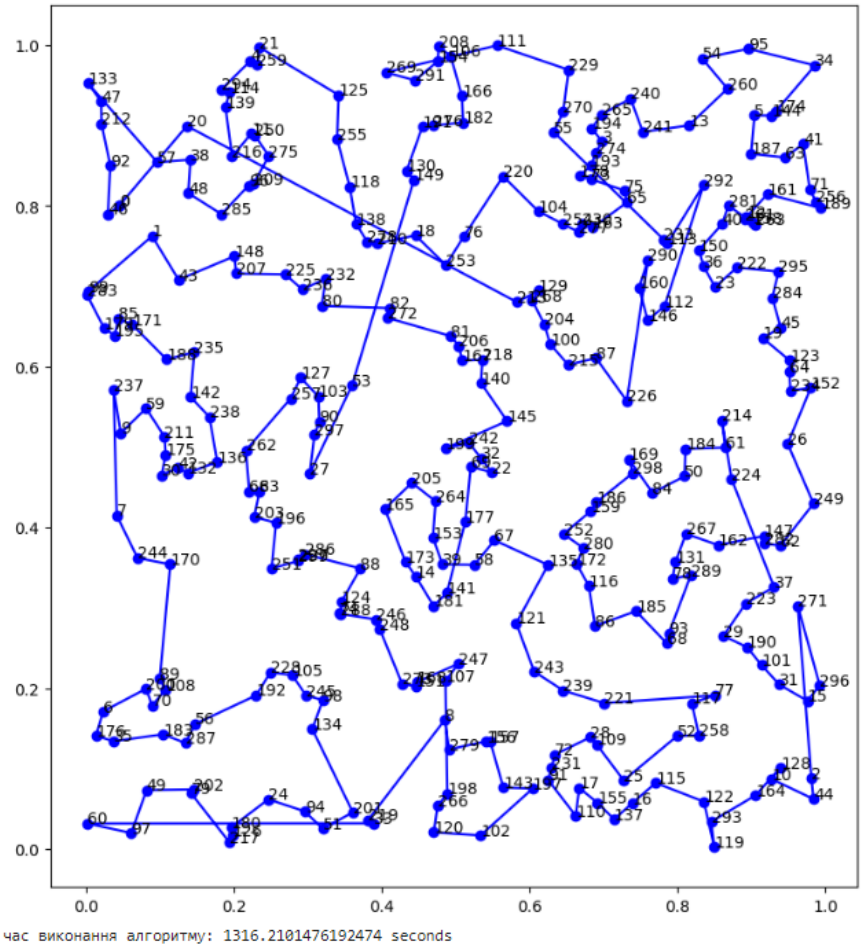


Рисунок 3.10 – Оптимальний маршрут для 300 вузлів

На основі даних таблиці 3.7 можна зробити висновок, що при невеликій кількості вузлів всі три досліджуваних алгоритми витрачають приблизно однаковий час на досягнення необхідної збіжності. При цьому алгоритм ACO краще працює при трохи більшій популяції та потребує виконання більшої кількості ітерацій.

За даними таблиці 3.8 спостерігаємо, що при великій кількості вузлів алгоритм ACO є дещо швидшим за ABC та CS і йому необхідно менше ітерацій, ніж іншим двом. Алгоритм ABC впорався за трохи менший час, ніж CS. У свою чергу, слід зазначити, що алгоритму CS при пошуку розв’язку даної задачі вдалося досягти стовідсоткової збіжності лише з другої спроби, тобто при першій генерації вершин алгоритм не знаходив найбільш короткий маршрут. Це є не помилкою програми, а недоліком алгоритму, який полягає в

тому, що в рідкісних випадках алгоритми ABC та CS не досягають повної збіжності через особливості конкретної задачі [37].

При аналізі роботи алгоритмів видно, що ACO, ABC та CS є універсальними методами для пошуку оптимальних маршрутів у логістичних системах. Для пошуку оптимуму в досить простих графах (невелика кількість вузлів) краще використовувати метод CS чи ABC, так як вони в даному випадку працюють трохи швидше за ACO.

При вирішенні задачі оптимізації для великих графів рекомендується використовувати ACO, так в даних умовах він працює швидше і потребує меншу кількість ітерацій для знаходження оптимального рішення. До того ж цей метод завжди гарантує стовідсоткову збіжність алгоритму.

Точність обчислення усіх алгоритмів є дуже високою. Варто зазначити що на час виконання алгоритмів та точність знайденого рішення впливають обрані параметри, які потрібно намагатись підібрати якомога оптимальніше [30–40], опираючись на проведені вище дослідження.

3.4Перспективи подальшої роботи

Дана сфера не є новою чи малодослідженою, проте досі лишається перспективною, через вдосконалення вже існуючих методів, та появи нових.

Перспектива розвитку може бути пов’язана із дослідженням модифікації розібраних методів, проведення їх порівняльного аналізу, визначення доцільності спрямування зусиль на спроби модифікації тих чи інших технологій. Оцінювання їх прикладної результативності стосовно об’ємних даних, де присутні додаткові динамічні коефіцієнти та параметри.

Під час виконання цього дослідження було виявлено ряд перспективних напрямків, які можуть бути важливими для подальших досліджень у галузі застосування тваринних технологій в логістичних системах. Ось деякі роздуми про ці можливі перспективи:

– поглиблення аналізу впливу параметрів алгоритмів (подальше дослідження може включати в себе докладний аналіз впливу параметрів тваринних алгоритмів на їх продуктивність. Визначення оптимальних значень параметрів може покращити результати оптимізації);

– порівняльний аналіз з іншими методами (проведення більш детального порівняльного аналізу тваринних алгоритмів з іншими методами оптимізації, такими як методи точної оптимізації та інші евристичні підходи). Це допоможе краще зрозуміти переваги і недоліки тваринних алгоритмів;

– адаптація до змінних умов (дослідження може бути спрямоване на розробку та апробацію тваринних методів, які здатні адаптуватися до змінних умов, таких як перемінні обмеження, перерви в постачанні чи змінні цільові функції).

Ці напрямки подальших досліджень в області використання тваринних алгоритмів у логістиці вказують на значущий потенціал цієї галузі та необхідність подальших наукових та практичних досліджень.

Основною перспективою ж є практичне використання тваринних технологій у реальних логістичних системах. Розробка та імплементація систем, які використовують ці алгоритми, може сприяти покращенню логістичних процесів.

ВИСНОВКИ

У рамках кваліфікаційної роботи було реалізовано та проведено порівняльний аналіз між трьома методами оптимізації перевезень у логістичних системах, інспірованих механізмами живої природи, а саме методи Мурашиної колонії (ACO), Штучної бджолиної колонії (ABC) та Зозулиного пошуку (CS).

Дослідження та порівняльний аналіз мурашиного алгоритму, алгоритму бджолиної колонії, та алгоритму зозулиного пошуку у контексті оптимізації логістичних систем відкривають перспективи для подальших досліджень у сфері розв’язання складних проблем логістики та оптимізації маршрутів.

Одним із ключових висновків роботи є те, що кожен з вивчених алгоритмів має свої унікальні переваги та обмеження. ACO, використовуючи ідеї феромонового сліду та ймовірнісний підхід, дозволяє знаходити оптимальні маршрути завдяки колективному інтелекту мурах. ABC, імітуючи поведінку бджолиного рою, забезпечує ефективний обмін інформацією та здатність адаптуватися до змін у середовищі. CS, інспірований поведінкою зозулі при виборі гнізда, вирізняється здатністю швидко перебирати різні варіанти для знаходження оптимального рішення.

Під час порівняльного аналізу алгоритмів було виявлено, що вибір оптимального методу залежить від конкретних умов задачі. Наприклад, ACO може виявитися ефективним у випадках, коли важлива експлуатація існуючих маршрутів, тоді як ABC та CS добре підходять для задач із змінними умовами, оскільки вони забезпечують більшу гнучкість та адаптивність.

Усі три алгоритми показали свою ефективність у вирішенні завдань оптимізації перевезень у логістичних системах, але також виявили певні обмеження.

Після проведеного експерименту стало відомо, що при середній (50-100) та великій (100+) кількості вузлів алгоритм бджолиної колонії досягає середньої точності у 96%, в той час як алгоритм зозулиного пошуку має 93% точності. Алгоритм мурашиної колонії, зі своєю середньою точністю на рівні 98%, перевершує обидва інші методи. З урахуванням таких показників ефективності, можна зробити висновок про відносну перевагу мурашиного алгоритму у вирішенні завдань оптимізації великих логістичних систем. Окрім того, час виконання алгоритму ACO на великих графах у середньому на 12-19% менший ніж у ABC та CS, однак, на 5-10% більший при пошуку маршрутів при малій (до 50) кількості вузлів.

До обмежень застосування даних методів слід віднести необхідність врахування динамічних умов середовища та інших реальних обмежень, що може потребувати додаткових модифікацій для кращої адаптації алгоритмів.

У цілому, подальше вдосконалення та розширення досліджень в цій області можуть призвести до створення нових, більш ефективних методів оптимізації, що знайдуть широке застосування у вирішенні завдань логістики та управління ланцюгом постачання.

Результати дослідження апробовано у вигляді тез доповідей під час XXXVI Міжнародної науково-практичної конференції «Modern problems and the latest theories of development» [41].

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Худов, Г. В., Хижняк, І. А., Марченко, В. П., & Горошко, О. О. (2022). Метод визначення маршруту руху транспортних засобів з використанням модифікованого алгоритму мурашиної колонії. *Системи обробки інформації*, (3 (170)), 58-66.

2. Данчук, В. Д. (2021). ІНТЕЛЕКТУАЛЬНІ МЕТОДИ ТА ІНФОРМАЦІЙНА ТЕХНОЛОГІЯ ОПТИМІЗАЦІЇ ПРОЦЕСІВ ДОСТАВКИ ВАНТАЖІВ У ВЕЛИКИХ МІСТАХ (Doctoral dissertation, НАЦІОНАЛЬНИЙ ТРАНСПОРТНИЙ УНІВЕРСИТЕТ).

3. Бабійчук, О. Ю. (2022). Система реалізації задачі дискретної оптимізації з використанням еволюційного алгоритму.

4. Трахановська, М. Р. (2020). Методи оптимізації маршрутів перевезення.

5. Бондаренко, О., Устиненко, О., & Сєриков, В. (2023). Метаевристичні алгоритми. Метафори-стратегії (оглядова стаття). *Вісник Національного технічного університету «ХПІ». Серія: Машинознавство та САПР*, (1), 3-18.

6. Ясько, О. С. (2023). Алгоритм мурашиної колонії для вирішення транспортних задач із виходом з локальних мінімумів. 27-ий міжнародний молодіжний форум «РАДІОЕЛЕКТРОНІКА І МОЛОДЬ У XXI СТОЛІТТІ». *РАДІОЕЛЕКТРОНІКА ТА МОЛОДЬ У XXI СТОЛІТТІ*, 222.

7. Москальський, Б. А. (2023). Оптимізація логістичних операцій підприємств малого бізнесу на основі інтелектуальних технологій.

8. Tvoroshenko I.S., and Kramarenko O.O. (2019) Software determination of the optimal route by geoinformation technologies, *Radio Electronics Computer Science Control*, 3, pp. 131-142.

9. Баранова, А. Д. (2021). *Маршрутизація транспортних засобів з часовими вікнами* (Master's thesis, КПІ ім. Ігоря Сікорського).

10. Тимчук, О. С., Проценко, Я. А., & Парамонов, А. І. (2019). Застосування алгоритму мурашиної колонії до вирішення задачі декількох комівояжерів без депо. *Системи обробки інформації*, (3), 73-78.

11. Гуляницький, Л. Ф., & Коткова, А. А. (2020). До класифікації задач маршрутизації транспортних засобів.

12. Герега, Б. Д. (2021). *Еволюційні алгоритми глобальної пошукової оптимізації* (Bachelor's thesis, КПІ ім. Ігоря Сікорського).

13. Xu, X., Hao, J., & Zheng, Y. (2020). Multi-objective artificial bee colony algorithm for multi-stage resource leveling problem in sharing logistics network. *Computers & Industrial Engineering*, 142, 106338.

14. Olkhova, M., Roslavtsev, D., Matviichuk, O., & Mykhalenko, A. (2020). City delivery routes planning based on the ant colony algorithm.

15. Препелиця, Б. Ю., & Михальчук, Г. Й. (2023, January). МЕТОД РОЗВ'ЯЗАННЯ ВЕЛИКОМАСШТАБНОЇ ЗАДАЧІ МАРШРУТИЗАЦІЇ ТРАНСПОРТНИХ ЗАСОБІВ. In *The 1th International scientific and practical conference "Current issues of science and integrated technologies"(January 10-13, 2023) Milan, Italy. International Science Group. 2023. 799 p.* (p. 722).

16. Найдьонов, І. М. (2015). Топологічна евристика в розв'язанні проблеми маршрутизації транспортних засобів (VRP). *ScienceRise*, *6*(2), 52.

17. Liang, S., Jiao, T., Du, W., & Qu, S. (2021). An improved ant colony optimization algorithm based on context for tourism route planning. PLoS One, 16(9), e0257317.

18. КОРОТКА, Є. (2019). ЕВРИСТИЧНІ АЛГОРИТМИ В ЗАДАЧАХ МАРШРУТИЗАЦІЇ ТРАНСПОРТНИХ ЗАСОБІВ. *МІНІСТЕРСТВО ОСВІТИ ТА НАУКИ УКРАЇНИ ХЕРСОНСЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ*, 59.

19. Jiang, C., Fu, J., & Liu, W. (2021). Research on vehicle routing planning based on adaptive ant colony and particle swarm optimization algorithm. *International Journal of Intelligent Transportation Systems Research*, 19(1), 83-91.

20. Wang, C., Qian, Y., & Shaic, S. (2021). The applications of nature-inspired algorithms in logistic domains: a comprehensive and systematic review. *Arabian Journal for Science and Engineering*, 46, 3443-3464.

21. Шишацький, А. В., Налапко, О. Л., & Одарущенко, О. Б. (2021). Основні біоінспіровані алгоритми обробки різнотипних даних. Інтеграція інформаційних систем і інтелектуальних технологій в умовах трансформації інформаційного суспільства: тези доповідей IV Міжнародної науково-практичної конференції, що присвячена 50-ій річниці кафедри інформаційних систем та технологій. Полтава: ПДАУ, 2021. 109-114. In *Integration of information systems and intelligent technologies in the conditions of information society transformation. Abstracts of the IVth International scientific-practical conference dedicated to the 50th anniversary of the Department of Information Systems and Technologies. Poltava, Ukraine. 2021. 144 p.* (p. 109).

22. Li, Y., Lim, M. K., & Tseng, M. L. (2019). A green vehicle routing model based on modified particle swarm optimization for cold chain logistics. *Industrial Management & Data Systems*, 119(3), 473-494.

23. Daradkeh Y.I., Gorokhovatskyi V., Tvoroshenko I., Gadetska S., and Al-Dhaifallah M. (2023) Statistical data analysis models for determining the relevance of structural image descriptions, *IEEE Access*, 11, pp. 126938-126949.

24. Tvoroshenko, I. S. (2004) Structure and functions of intelligent decision-making tools in complex systems. *Artificial Intelligence*, *4*, 462-470.

25. M. Ayaz Ahmad, Irina Tvoroshenko, Jalal Hasan Baker, Liubov Kochura, Vyacheslav Lyashenko (2020) Interactive Geoinformation Three-Dimensional Model of a Landscape Park Using Geoinformatics Tools, *International Journal on Advanced Science, Engineering and Information Technology*, 10(5), pp. 2005-2013.

26. Гороховатський В.О., Творошенко І.С. (2021). Методи інтелектуального аналізу та оброблення даних: навч. посібник.

27. Творошенко, І. С., & Подласенко, Є. П. (2019). Дослідження методу розпізнавання геоінформаційних ситуацій в системах моніторингу територій.

28. Гороховатський В., Творошенко І., Сидоренко Д. (2021) Класифікація зображень із використанням кластерного подання, *Міжн. наук. симпозіум Інтелектуальні рішення-С. Обчислювальний інтелект. Теорія прийняття рішень: праці міжн. наук. симп. (Вересень 29, 2021)*. Київ-Ужгород, С. 44-45.

29. Ramos, T. R. P., Gomes, M. I., & Póvoa, A. P. B. (2020). Multi-depot vehicle routing problem: a comparative study of alternative formulations. *International Journal of Logistics Research and Applications*, *23*(2), 103-120.

30. Daradkeh Y.I., Gorokhovatskyi V., Tvoroshenko I., and Zeghid M. (2022) Tools for fast metric data search in structural methods for image classification, *IEEE Access*, 10, pp. 124738-124746.

31. Gorokhovatskyi V., Tvoroshenko I., Kobylin O., and Vlasenko N. (2023) Search for visual objects by request in the form of a cluster representation for the structural image description, *Advances in Electrical and Electronic Engineering*, 21(1), pp. 19-27.

32. Гороховатський В.О., Творошенко І.С., Чмутов Ю.В. (2022) Застосування систем ортогональних функцій для формування простору ознак у методах класифікації зображень, *Сучасні інформаційні системи*, 6(3), С. 5-12.

33. Гороховатський В., Передрій О., Творошенко І., Марков Т. (2023) Матриця відстаней для множини компонентів структурного опису як інструмент для створення класифікатора зображень, *Сучасні інформаційні системи*, 7(1), С. 5-13.

34. Pomazan V., Tvoroshenko I., and Gorokhovatskyi V. (2023) Development of an application for recognizing emotions using convolutional neural networks, *International Journal of Academic Information Systems Research*, 7(7), pp. 25-36.

35. Vladyslav, Y. (2022). ANALYSIS OF EXISTING METHODS OF SEARCHING IMAGES IN DATABASES. *EDITORIAL BOARD*, 801.

36. Pomazan V., Tvoroshenko I., and Gorokhovatskyi V. (2023) Handwritten character recognition models based on convolutional neural networks, *International Journal of Academic Engineering Research*, 7(9), pp. 64-72.

37. Євтушенко, В., & Тарасенко, Д. (2023, January). ІНФОРМАЦІЙНІ ТЕХНОЛОГІЇ ПЛАНУВАННЯ ТА РОЗРОБЛЕННЯ ІТ-СТАРТАПІВ. In *The 1th International scientific and practical conference "Current issues of science and integrated technologies"(January 10-13, 2023) Milan, Italy. International Science Group. 2023. 799 p.* (p. 645).

38. Tvoroshenko I., Gorokhovatskyi V., Kobylin O., and Tvoroshenko A. (2023) Application of deep learning methods for recognizing and classifying culinary dishes in images, *International Journal of Academic and Applied Research*, 7(9), pp. 57-70.

39. Gorokhovatskyi V., Tvoroshenko I. (2023) Identification of visual objects by the search request. *International scientific symposium «INTELLIGENT SOLUTIONS-S». Computational intelligence (results, problems and perspectives). Decision making theory: proceedings of the international symposium*, September 28, 2023, Kyiv-Uzhorod, Ukraine, pp. 25-27.

40. Yakovleva O., Kovač M., Ardasov V. & Yeremenko I. (2023). Study on adding functionality to the Zoom online conference system for monitoring the participant activities, *Public Administration and Regional Development*, 19(1), pp. 158-184.

41. Євтушенко В. (2023) Порівняльний аналіз методів оптимізації маршрутів на основі «тваринних» технологій, *Abstracts of XXXVI International Scientific and Practical Conference «Modern problems and the latest theories of development», (September 11 – 13, 2023). Munich, Germany*, pp. 260-262.