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

МАГІСТЕРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

на тему:

Автоматизована система оптимізації розкладу руху маршрутного транспорту з урахуванням часу простою

Виконав: студент 2 курсу, групи
ЗАКІТ-22м спеціальності 151 –
Автоматизація та комп'ютерно-інтегровані технології

Роман БАРАНЕЦЬ
Ім'я ПРІЗВИЩЕ

Керівник:
к.т.н., доцент, доцент кафедри КСУ
степінь, звання, посада
Олена НИКИТЕНКО
Ім'я ПРІЗВИЩЕ
«15» грудня 2023 р.

Опонент:
к.т.н., професор, професор кафедри КН
степінь, звання, посада
Олег КОЛЕСНИЦЬКИЙ
Ім'я ПРІЗВИЩЕ
«15» грудня 2023 р.

Допущено до захисту
Зав. кафедри АІІТ
Олег БІСІКАЛО
«15» жовтня 2023

Вінниця ВНТУ – 2023 рік

Вінницький національний технічний університет
Факультет інтелектуальних інформаційних технологій та автоматизації
Кафедра автоматизації та інтелектуальних інформаційних технологій
Рівень вищої освіти другий (магістерський)
Галузь знань – 15 – Автоматизація та приладобудування
Спеціальність – 151 – Автоматизація та комп'ютерно-інтегровані технології
Освітня програма - Інформаційні системи і Інтернет речей

ЗАТВЕРДЖУЮ
Завідувач кафедри АІІТ
Олег БІСІКАЛО
" 20 " вересня 2023 року

ЗАВДАННЯ
НА МАГІСТЕРСЬКУ КВАЛІФІКАЦІЙНУ РОБОТУ

студенту Баранцю Роману Вікторовичу
(прізвище, ім'я, по батькові)

1. Тема роботи Автоматизована система оптимізації розкладу руху маршрутного транспорту з урахуванням часу простою
керівник роботи Никитенко Олена Дмитрівна
затверджені наказом ВНТУ від "18" вересня 2023 року №247
2. Термін подання студентом роботи "12" грудня 2023 року
3. Вихідні дані до роботи: сформований план транспортування продукції, який враховує всі потреби споживачів, часові вікна обслуговування та мінімізує витрати на транспортування, розроблений алгоритм не меньш ніж на 10% ефективніший.
4. Зміст текстової частини: вступ, аналіз предметної області та огляд аналогів, удосконалення алгоритму мурашиної колонії та генетичного алгоритму, розробка програмного забезпечення, порівняння результатів.

5. Перелік ілюстративного матеріалу (з точним зазначенням обов'язкових креслень)
Класифікація характеристик VRP, приклад VRP, Ієрархічна схема узагальнень і розширень VRP, Клисифікація найпоширеніших алгоритмів розв`язання VRP, Схема зв`язку узагальнюючих різновидів VRP, Структурна схема алгоритму мурашиних колоній в загальному виді, Вікно програми після запуску, Отримані результати, Графік залежності часу роботи алгоритмів від класу задач, , Порівняння з іншими алгоритмами, Порівняння розв`язків алгоритмів.

1. Консультанти розділів роботи

Розділ	Ім`я, прізвище та посада консультанта	Підпис, дата	
		завдання видав	виконання прийняв
1-4	Олена НИКИТЕНКО к.т.н., доцент, доцент кафедри КСУ	01.10.2023	01.12.2023

2. Дата видачі завдання "20" вересня 2023 року

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва та зміст етапу	Термін виконання		Примітка
		початок	закінчення	
1	Проведення літературного огляду щодо існуючих систем оптимізації розкладу транспорту	20.09.2023	30.09.2023	вик.
2	Огляд та удосконалення методу мурашиних колоній	01.10.2023	10.10.2023	вик.
3	Написання коду для удосконаленого методу мурашиних колоній	11.10.2023	09.11.2023	вик.
4	Здійснення повного тестування системи	10.11.2023	19.11.2023	вик.
5	Порівняння результатів з існуючими методами оптимізації розкладів транспорту	20.11.2023	30.11.2023	вик.
6	Складання тексту магістерської роботи. Редагування та форматування	01.12.2023	12.12.2023	вик.

Студент (підпис) Роман БАРАНЕЦЬ (*Ім'я ПРІЗВИЩЕ*)

Керівник роботи (підпис) Олена НИКИТЕНКО (*Ім'я ПРІЗВИЩЕ*)

АНОТАЦІЯ

УДК 004.42

Роман Баранець Автоматизована система оптимізації розкладу руху маршрутного транспорту з урахуванням часу простою. Магістерська кваліфікаційна робота зі спеціальності 151 – Автоматизація та комп'ютерно-інтегровані технології, освітня програма – Інформаційні системи і Інтернет речей. Вінниця: ВНТУ, 2023 105с.

На укр. мові. Бібліогр.: 67 назв; рис.:27; табл.:14.

Ця магістерська робота присвячена оптимізації маршрутизації транспортних засобів, враховуючи часові вікна для мінімізації вартості доставки.

У першій частині аналізується проблема маршрутизації та існуючі рішення. У другому розділі пропонується модифікація алгоритму оптимізації мурашиної колонії та гібридний підхід до задачі маршрутизації з урахуванням часових вікон. Третій розділ описує програмне забезпечення розроблене на мові програмування Python, та надає інструкцію користування. У розділі 4 представлено мету та результати дослідження, ща підтверджує ефективність запропонованих алгоритмів порівняно з іншими рішеннями.

Ключові слова: маршрутизація, часові вікна, оптимізація мурашиної колонії, гібридні алгоритми.

ABSTRACT

Roman Baranets Automated optimization system for the schedule of route transportation considering downtime. Master's qualification work in the specialty 151 – Automation and Computer-Integrated Technologies, educational program – Information Systems and Internet of Things. Vinnytsia: VNTU, 2023. 105p.

In Ukrainian. Bibliography: 67 titles; fig.:27; tab.:14

This master's thesis is dedicated to the optimization of vehicle routing, taking into account time windows to minimize delivery costs. The first part analyzes the routing problem and existing solutions. The second section proposes a modification of the ant colony optimization algorithm and a hybrid approach to the routing problem with consideration of time windows. The third section describes the software developed in the Python programming language and provides user instructions. Section 4 presents the purpose and results of the research, confirming the effectiveness of the proposed algorithms compared to other solutions.

Keywords: routing, time windows, ant colony optimization, hybrid algorithms.

ЗМІСТ

1 АНАЛІЗ ЗАДАЧ МАРШРУТИЗАЦІЇ ТРАНСПОРТНИХ ЗАСОБІВ ТА ІСНУЮЧИХ МЕТОДІВ РОЗВ'ЯЗАННЯ 10
1.1 Класифікація задач оптимізації маршрутів транспортних засобів 10
1.2 Опис класичної задачі Vehicle Routing Problem (VRP 16
1.2.1 Постановка задачі 17
1.2.2 Математична постановка задачі 18
1.3 Огляд різновидів задач маршрутизації транспортних засобів 20
1.4 Аналіз класичних алгоритмів розв'язання задач маршрутизації транспорту 26
1.4.1 Точні методи 27
1.4.2 Евристичні методи 29
1.4.3 Метаевристичні методи 30
1.5 Огляд існуючих програмних продуктів 38
1.6 Постановка задачі дослідження 38
2 РОЗРОБКА МОДИФІКАЦІЙ АЛГОРИТМУ МУРАШИНИХ КОЛОНІЙ ТА ГЕНЕТИЧНОГО АЛГОРИТМУ ДЛЯ ЗАДАЧІ VRPTW 40
2.1 Математична постановка задачі VRPTW 41
2.2 Алгоритми Ant Colony Optimization для розв'язання VRPTW 44
2.2.1 Опис класичного алгоритму Ant Colony Optimization 44
2.3 Переваги та недоліки мурашиного алгоритму 50
2.4 Модифікації алгоритму мурашиних колоній 51
2.5 Розробка гібридного алгоритму 53
3 ОПИС ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ 55
3.1 Засоби розробки 55
3.2 Архітектура програмного забезпечення 57
3.2.1 Структура класів програмного забезпечення 57
3.2.2 Специфікація функцій програмного забезпечення 58
3.3 Вхідні данні 61
3.4 Вихідні данні 62

3.5 Інструкція користувача 62
РЕЗУЛЬТАТИ ДОСЛІДЖЕНЬ 67
4.1 Мета досліджень 67
4.2 Тестові данні 67
4.3 Результати досліджень ефективності запропонованих модифікацій алгоритму мурашиних колоній 68
4.4 Аналіз результатів досліджень 73
ВИСНОВКИ 76
ПЕРЕЛІК ПОСИЛАНЬ 78
ДОДАТКИ 82
Додаток А (обов`язковий) Технічне завдання 83
Додаток Б (обов`язковий) Ілюстративна частина 86
Додаток В (обов`язковий) Лістинг програми 93
Додаток Г (обов`язковий) Перевірка на плагіат 101

ВСТУП

Актуальність. Згідно з 30м щорічним звітом з логістики від The Council of Supply Chain Management Professionals (CSCMP) [1] лише в США за рік було витрачено понад 1.64 трильйонів доларів на логістичні та транспортні операції, а кінцева вартість продукту, що потрапляє до споживача, може складатись до 70% з транспортних витрат. Проблема маршрутизації транспортних засобів (VRP) передбачає визначення оптимального набору маршрутів, необхідних парку транспортних засобів (VRP) для обслуговування даного набору клієнтів, і є однією з найважливіших і добре вивчених задач комбінаторної оптимізації. Тож перед галуззю транспортної логістики однією з важливіших задач постає економія ресурсів та мінімізація екологічного сліду при транспортуванні вантажів. Для вирішення цієї проблеми стоїть питання розробки алгоритмів та програмних продуктів, що будуть скорочувати маршрути транспортних засобів. В останні роки значно збільшилась частка персональних доставок, які прив’язанні до зайнятості клієнтів, актуальним є питання врахування часових вподобань одержувачів вантажу. Математичне формулювання цієї задачі відоме як задача маршрутизації транспортних засобів з урахуванням часових вікон (далі VRPTW), яка накладає певні часові обмеження на обслуговування клієнтів транспортної мережі. Робота присвячена дослідженню та удосконаленню розв’язання задачі VRPTW.

З того часу, як Данциг і Рамзер ввели задачу в 1959 році, минуло понад 40 років [3]. Вони описали реальне застосування щодо доставки бензину до СТО та запропонували перше формулювання математичного програмування та алгоритмічний підхід. Кілька років потому, в 1964 році, Кларк і Райт запропонували ефективну жадібну евристику, яка вдосконалила підхід Данцига-Рамзера [4]. Після цих двох основних робіт було запропоновано сотні моделей та алгоритмів для оптимального та наближеного розв’язання різних версій VRP [11]. Значний вклад у дослідження задач VRP внесли також Фред

Гловер, який вперше описав Табу пошук та ввів термін «метаевристика» [5]; Маріус Соломон, який досліджував підходи чутливих часових аспектів ланцюга постачань [6], часові обмеження при плануванні [7], маршрутизацію та управління складськими системами [8]; Марко Доріго, що запропонував метаевристичний “Мурашиний алгоритм” [9], [10].

Математичне формулювання цієї задачі відоме як задача маршрутизації транспортних засобів з урахуванням часових вікон (VRPTW), яка накладає певні часові обмеження на обслуговування клієнтів транспортної мережі. Робота присвячена дослідженню та удосконаленню розв’язання задачі VRPTW.

Мета роботи. Мета – підвищити ефективність методів розв’язування задач маршрутизації транспортних засобів із часовими вікнами.

Загальна постановка задач дослідження:

- Аналізувати відомі результати при розв'язанні задач маршрутизації транспортних засобів;

- Удосконалення існуючих алгоритмів розв'язування задач маршрутизації транспортних засобів з урахуванням часових вікон за рахунок модифікацій та комбінацій метаевристик;

- Розробляти програмну реалізацію розроблених алгоритмів;

- Провести дослідження ефективності розробленого алгоритму.

Об'єктом дослідження є процес організації перевезень.

Предмет дослідження – задача маршрутизації транспортних засобів з часовими вінками.

Методи дослідження, застосовані в роботі, базуються на методах дослідження операцій, зокрема на метаевристичних алгоритмах.

Новизна отриманих результатів. Розроблені модифікований
та гібридний алгоритми розв’язання задачі VRPTW.

1 АНАЛІЗ ЗАДАЧ МАРШРУТИЗАЦІЇ ТРАНСПОРТНИХ ЗАСОБІВ ТА ІСНУЮЧИХ МЕТОДІВ РОЗВ'ЯЗАННЯ

1.1 Класифікація задач оптимізації маршрутів транспортних засобів

Дослідження Eksioglu, Vural і Reisman [11] виявило 1638 журнальних статей, опублікованих між 1959 і 2014 роками, головною темою яких є VRP. Багато книг і численні наукові статті також зробили внесок у сучасну літературу про VRP. Згідно Eksioglu та ін., література, що вивчає VRP, експоненціально зростає зі швидкістю 6% на рік [11]. Кількість публікацій на тему ВРП наведено на рисунках 1.1 та 1.2.

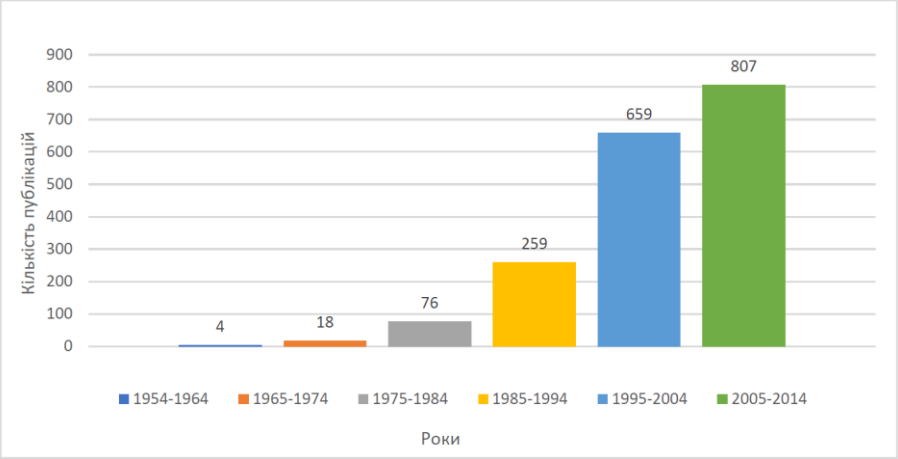

Рисунок 1.1 - Кількість публікацій на тему VRP, опублікованих у проміжок з 1954 по 2014 років

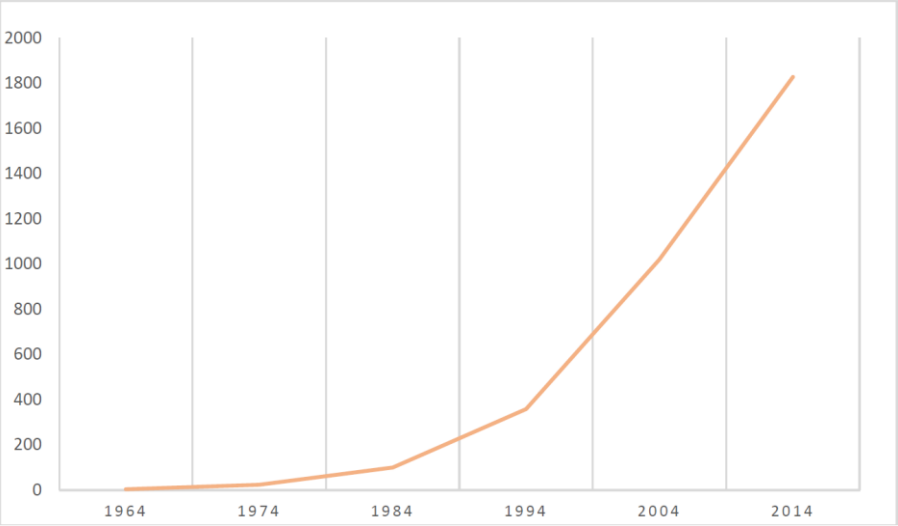

Рисунок 1.2 – Кумулятивний розподіл кількості публікацій на тему VRP за проміжок з 1954 по 2014 років

Наступні дослідники опублікували найбільше робіт у галузі VRP: Гілбер Лапорт (66 разів), Мішель Жандро (42 рази), Жан-Ів Потвін (41 раз), Брюс Л. Голден (34 рази), Маріус М. Соломон (22 рази) разів) і Христос Д. Тарантіліс (19 разів) [12].

Велика кількість публікацій на тему ВРП зумовлена її різноманітністю. Основною спільною рисою різних VRP є орієнтація на врахування одного або кількох обмежень (характеристик) математичної моделі.

Виділено такі основні характеристики VRP [13]:

а) Кількість точок живлення:

1) постачання в одній точці (наприклад, постачання або обслуговування зі складу різним клієнтам);

2) Кілька точок доставки (наприклад, доставка від кількох виробників).

б) Бали витрат:

1) споживання в одній точці (наприклад, продукція з різних місць відвантажується на склади або в пункти сертифікації);

2) Кілька точок споживання.

в) Стандартна кількість:

1) Єдиний критерій (тільки один критерій - відстань, час або інші характеристики - враховується при оцінці оптимальності маршруту);

2) Багатокритеріальний (часто з використанням кількох конфліктуючих критеріїв).

г) Параметри дороги:

1) Дорога характеризується параметром;

2)Дороги характеризуються кількома параметрами (відстань, пропускна здатність, розмір, плата за проїзд або інші фіксовані збори, обмеження швидкості тощо).

д) кількість товару:

1) Втрата товарів (наприклад, виконання сервісних робіт);

2) Кількість товару представлено дійсними числами (неперервна задача):

- один;

- дещо;

3) Кількість товару характеризується цілими числами (дискретна задача):

- один;

- Дещо.

е) Тип вантажу:

1) Однотипний товар;

2) Кілька одиниць товару:

- Дозволений будь-який вантаж;

-Забороняється використання окремих транспортних засобів для перевезення вантажів;

- Кожен транспортний засіб має обмеження на різні види вантажів, що перевозяться;

- Спільне перевезення окремих видів вантажів в одному транспортному засобі заборонено.

є) Кількість транспортних компаній:

1) Одна (стоянки зосереджені в одному місці);

2) Кілька (флот розподілений територіально);

3) Збігається з точкою постачання.

ж) Кількість традиційних знань:

1) частина традиційних знань;

2) фіксований обсяг наявних традиційних знань;

3) Обсяг доступних традиційних знань не обмежений.

з) Однорідність стоянки:

1) Усі традиційні знання однакові;

2) Різні типи ТЗ мають різні характеристики:

- Різниця в певній характеристиці (наприклад, ємність);

- Багато характеристик відрізняються (вантажопідйомність, вартість використання, швидкість і т.д.).

и) Обмеження місткості автомобіля:

1) Без обмежень (кількість товару можна не враховувати);

2) Обмежте лише один параметр;

3) Надає обмеження на кілька параметрів.

і) Вимоги до транспортування:

1) При транспортуванні товарів від кожного пункту постачання до кожного пункту споживання може використовуватися один транспортний засіб незалежно від товару, що перевозиться;

2) При транспортуванні товарів з різних пунктів видачі один транспортний засіб може доставляти товари в різні точки споживання;

3) Вантажівка доставляє товари з різних точок постачання до кожної точки споживання;

4) ТЗ може збирати товари в різних точках виробництва та транспортувати їх до різних точок споживання.

ї) Додаткові умови:

1) кожен вид товарів повинен транспортуватися від певної точки виробництва до певної точки споживання (наприклад, пасажирський транспорт);

2) порожній біг не допускається (за винятком першої та останньої ділянок маршруту);

3) Кожен транспортний засіб може подорожувати лише один раз;

4) Деякі точки споживання також є точками постачання:

- Вивіз з точки споживання відбувається одночасно з доставкою;

- Вилучено з точки споживання після доставки

й) Строк:

1) Максимальна тривалість транспортних маршрутів обмежена;

2) час безперервного перебування транспортного засобу на дорозі обмежений;

3) Часове вікно:

- Доставка в точки споживання через визначені проміжки часу;

- Вихід з виробничої точки здійснюється через задані проміжки часу;

4) Крос-докінг.

к) Обмеження відстані:

1) Загальна довжина всіх маршрутів обмежена;

2) Максимальна відстань, яку транспортний засіб може проїхати, обмежена.

л) Обмеження щодо місць виробництва та споживання:

1) Деякі ТЗ не можуть отримати доступ до певних точок;

2) Предмети розподіляються між ТЗ.

м) Балансові обмеження:

1) Тривалість збалансованої маршрутизації;

2) Збалансувати довжину маршруту;

3) Збалансувати кількість точок обслуговування;

4) Збалансувати кількість поставленого товару.

н) Послідовність транспортування:

1) Наведіть відповідне співвідношення між точками виробництва та точками споживання;

2) Розділення етапу транспортування та етапу видалення;

3) Обмеження порядку завантаження та розвантаження вантажів - принцип штабелювання (ЛІФО);

4) Висунути інші вимоги до послідовності транспортування.

о) Періодичність:

1) Запланований маршрут виконується один раз;

2) Планування довгострокове;

3) Сплануйте регулярний маршрут.

п) Фактори, що визначають транспортні витрати:

1) Транспортні витрати залежать лише від самого маршруту;

2) Залежність транспортних витрат від маршруту та завантаження автомобіля:

- Пропорційна залежність;
- Складні функціональні залежності.

р) Умови повернення традиційних знань:

1) Транспортний засіб необхідно повернути на початкову стоянку;

2) транспортний засіб має завершити маршрут на певній стоянці;

3) Транспортний засіб може закінчити свій маршрут у будь-якому місці.

с) Топологія:

1) Задача симетрії (характеристики шляху не залежать від вибору напрямку):

2) Метрична (відстань описується евклідовою метрикою);

3) Неметрична (відстань не описується евклідовою метрикою);

4) Проблема асиметрії (характеристики шляху залежать від вибору напрямку);

5) Комплексна топологія.

т) Випадковість:

1) детерміновані дані;

2) Випадкові дані (компоненти задачі мають випадкову поведінку):

- Випадкові параметри дороги;
- Випадкові запити;
- Випадкові умови (обмеження);
- Випадкові особливості ТЗ.

у) Впевненість:

1) Статичні завдання, тобто замовлення не змінюється в процесі доставки;

2) Динамічні завдання, тобто замовлення можна додавати або скасовувати.

Ця класифікація в першу чергу стосується автомобільного транспорту, але більшість наведених характеристик стосується й інших видів. Наведене сімейство атрибутів дозволяє сформулювати набір задач транспортної логістики, в тому числі маршрутних. Систематичний опис класифікації характеристик VRP показано на рисунку Додаток Г рис1.

1.2 Опис класичної задачі Vehicle Routing Problem (VRP)

Проблема маршрутизації транспортного засобу (VRP) — це загальний термін для класу задач, у яких необхідно визначити набір маршрутів для парку транспортних засобів, розташованих на одній або кількох стоянках, для певної кількості територіально розподілених клієнтів. Таким чином, завдання полягає у визначенні найкращої стратегії доставки, щоб мінімізувати загальну вартість транспортування продукту від джерела до клієнта.

Мета VRP полягає в тому, щоб забезпечити доставку набору клієнтів із відомими потребами вздовж маршруту транспортування з найменшою вартістю, починаючи і закінчуючи на складі.

На малюнку 1.3 показано проблему VRP і один із можливих результатів.

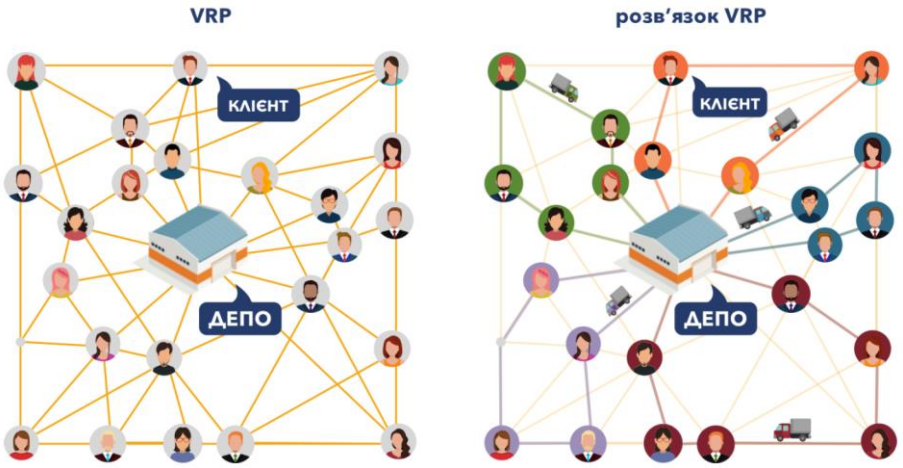

Рисунок 1.3 – Приклад VRP та один з варіантів її розв'язання

1.2.1 Постановка задачі

У класичному VRP визначається набір клієнтів (їх потреби, місце розташування), місця розташування складів, кількість транспортних засобів, доступних для транспортування продукту, і їх місткість.

Зі складу товари повинні бути доставлені замовнику в кількості, яка відповідає потребам замовника. Кожен транспортний засіб має починати та закінчувати свій маршрут у гаражі та бути доставленим принаймні одному клієнту.

Завдання полягає у визначенні розподілу клієнтів і транспортних засобів між маршрутами та порядку обслуговування клієнтів на маршрутах. Мета полягає в тому, щоб знайти рішення, яке мінімізує загальну вартість транспортування. Крім того, трансфери повинні враховувати обмеження, що кожен клієнт може отримати лише одну повну послугу, а загальний попит на кожен маршрут має бути в межах місткості транспортного засобу. Вартість транспортування визначається як вартість переміщення з будь-якої точки в будь-яку іншу точку.

1.2.2 Математична постановка задачі

Класична задача оптимізації транспортних засобів належить до задач комбінаторної оптимізації [14]. Тому формально класичну задачу VRP можна представити у вигляді графа G = (N, A), де N – множина вершин, що відповідає множині клієнтів, позначається 1,2,... . . . , n, вершина 0 відповідає центральному депо (депо) початкової та кінцевої точок маршруту для всіх транспортних засобів; A – набір дуг, що з'єднують відповідні вершини (клієнти) графа, якщо i – клієнт, а j – інший client , то дуга, що з'єднує їх, представлена (i, j) ∈ A.

Кожен клієнт має певні вимоги di, i∈N. Кожна дуга відповідає часу tij - часу руху автомобіля від клієнта i до клієнта j. Цей час включає час обслуговування клієнта i: si; вартість переміщення автомобіля від i до j дорівнює cij.

Індекс k ∈ V далі представлятиме відповідну вантажівку (де V – кількість однакових вантажівок вантажопідйомністю q). Змінна t приймає значення {0, 1}, де 1 означає, що ТК переміщується з вершини I у вершину j, а 0 означає навпаки.

Базуючись на наведених вище позначеннях, математична модель задачі VRP виглядає наступним чином (обмеження 1.2 - 1.7), де цільова функція (1.1) повинна бути мінімізована:

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

$$\sum_{k \epsilon V} \sum_{j \in N} x_{ijk} = 1, \quad \text{для } \forall i \in N, \qquad (1.2)$$

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

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

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

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

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

Загальна вартість усіх маршрутів для всіх ТЗ визначається цільовою функцією (1.1). Обмеження (1.2) означає, що кожен клієнт обслуговується ТЗ лише один раз. Обмеження (1.3) стверджує, що транспортний засіб не може обслуговувати більше клієнтів, ніж дозволяє його вантажопідйомність. Обмеження (1.4) визначає, що кожен транспортний засіб виїжджає зі стоянки тільки один раз. Обмеження (1.5) означає, що транспортний засіб може покинути вершину h тільки після входу в вершину. Обмеження (1.6) стверджує, що всі транспортні засоби повертаються на стоянку, але тільки один раз. Це обмеження випливає з обмежень (1.4) і (1.5).

Рішення проблеми:

- Розбити множину вершин на підмножини (маршрути);

- Встановіть порядок обходу кожної підмножини.

Рішення є прийнятним (прийнятним), якщо всі шляхи задовольняють обмеження задачі. VRP є узагальненням проблеми комівояжера (TSP), тому вона є NP-складною. TSP — це єдиний транспортний засіб VRP без обмежень, без гаражів і без попиту клієнтів.

1.3 Огляд різновидів задач маршрутизації транспортних засобів

Різноманітний характер створює велику кількість самостійних галузей наукової лінії VRP. Кожна галузь дослідження визначається скороченням обмежень або властивостей, які розглядаються задачею оптимізації. Багато з цих незалежних гілок об'єднуються для створення нових «головних» гілок. Розглянемо найпоширеніші ВРП, які зустрічаються в науковій літературі:

Asymmetric Cost Matrix VRP (AVRP) – асиметрична проблема маршрутизації. Взагалі кажучи, асиметрична задача відрізняється від симетричної тим, що вона моделюється орієнтованим графом. Вартість переходу від клієнта i до клієнта j відрізняється від вартості переходу від клієнта j до клієнта i. [14]

Проблема спокійного маршрутизації транспортного засобу (CVRP) — це завдання маршрутизації транспортного засобу, яка враховує навантаження транспортного засобу. У випадку CVRP до класичної проблеми VRP додаються обмеження на місця споживання, і пропускна здатність ТЗ стає обмеженою [16].

обмеження

$$\sum_{i \in N} d_i \sum_{j \in N} x_{ijk} \leq q, \text{ для } \forall\, k \in V$$

описано, що обсяг вантажу на кожному маршруті не повинен перевищувати вказане значення q (однакове для всіх ТЗ) [17], [18].

Мета полягає в тому, щоб мінімізувати кількість автопарків, задіяних у доставці, і загальний час.

VRP з обмеженням відстані (DCVRP) — це завдання маршрутизації з обмеженнями відстані. Загальна довжина дуг на шляху не може перевищувати максимальну довжину шляху. Це обмеження може замінити або доповнити обмеження потужності [22].

Мета полягає в тому, щоб врахувати обмежену місткість паливного баку.

Гетерогенний флот VRP (HVRP) — це завдання використання гетерогенного парку для маршрутизації транспорту. Одночасно

використовуються різні типи транспортних засобів, а маршрути плануються з урахуванням місткості кожного автомобіля.

Коли кількість транспортних засобів необмежена, проблема називається розміром автопарку та змішаним VRP (FSMVRP). Якщо певний тип транспортного засобу з будь-якої причини не може дістатися до певних клієнтів, завдання стає залежним від сайту VRP (SVRP). Крім того, якщо транспортному засобу дозволено здійснювати кілька поїздок, тоді ми можемо вирішити HVRP через багаторазове використання транспортного засобу (HVRPM) [15].

Багатосайтовий VRP (MDVRP) - завдання транспортної маршрутизації з кількома сайтами. Ми не маємо одного центрального складу, а маємо кілька складів, де ми можемо надавати клієнтам обслуговування. Таким чином, кожному ТК призначається станція, де він починає і закінчує свій маршрут, і різні ТК можуть бути прикріплені до різних станцій. Мета полягає в тому, щоб мінімізувати кількість флоту та загальний час доставки. [21 – 23] Проблема масштабування MDVRP з маршрутом від депо до депо (MDVRPI) передбачає перевантаження транспортних засобів у будь-якому депо на маршруті [24–26].

Open VRP (OVRP) — це відкрита проблема маршрутизації трафіку. Запланований маршрут може не закінчуватися на станції. Тому віддаленість останнього клієнта від станції не впливає на загальну вартість трансферу. Використовується, коли компанія не має власного автопарку, але використовує служби експрес-доставки [27].

VRP періодичної передачі (PVRP) — це завдання маршрутизації періодичної передачі. Оптимізація відбувається протягом кількох днів (зазвичай раз на день). Клієнти можуть обслуговуватися кілька разів з різною частотою протягом обраного періоду часу [28]. Існує також розширена версія PVRP з вибором послуги - PVRP з вибором послуги (PVRP-SC), в якій частота відвідувань клієнта встановлюється в міру вирішення проблем. Це дає вам доступ до більш оптимальних маршрутів і перевагу в обслуговуванні клієнтів [29].

VRP із самовивозом і доставкою (PDVRP) — це завдання маршрутизації транспортування через прийом і доставку. Кожен клієнт має дві цінності

- 1 – необхідно доставити, 2 – потрібно забрати у замовника і доставити на склад.

Клієнти можуть як отримати, так і відправити товар. Тобто від споживача на склад повертається певна кількість товару. За винятком того, що загальне навантаження та загальний обсяг доставки вздовж маршруту не можуть перевищувати ліміт вантажопідйомності транспортного засобу, ця ємність не повинна бути перевищена в будь-якій точці маршруту [15].

Варіант завдання PDVRP полягає в тому, що вивезені товари не повертаються на склад, а повинні бути доставлені іншому клієнту, наприклад, транспорту людей. У деяких випадках транспортний засіб має забрати та доставити одному клієнту за один прийом (одночасний прийом і доставка VRP), наприклад, нові та повернуті пляшки. Також важливий варіант VRP - 1 − M − 1 («один-до-багатьох-до-одного»), коли всі товари, що підлягають доставці, забираються лише зі складу, а всі товари, які підлягають експорту, доставляються лише на склад. [30]. Варіанти PDVRP також включають Deliver First, Second Pick VRP або VRP with Backhaul (VRPB) — усі товари мають бути доставлені замовнику перед видаленням від клієнта-постачальника [31] та Mixed Pick and Deliver VRP (VRPMD) — видалення та доставка у будь-якому порядку[32]. Варто відзначити варіант VRP, що використовує принцип стекування «останнім прийшов, першим вийшов» - VRP з LIFO. Подібно до VRRPD, але з додатковим обмеженням: розвантаження вантажу [33] може відбуватися лише в порядку, зворотному завантаженню, тобто останній завантажений продукт завжди вивантажується першим. Часто використовується для багатопозиційних вантажів, щоб скоротити час навантаження і розвантаження транспортних засобів, оскільки немає необхідності переставляти вантаж. Проте в літературі інтенсивно досліджувався лише випадок одного ТЗ, ТСП із прийомом і доставкою та ЛІФО (ЦППДЛ).

VRP розділеної доставки (SDVRP) — це проблема маршрутизації розділеної доставки. Різні транспортні засоби можуть використовуватися одночасно для обслуговування клієнтів, якщо можна зменшити загальні витрати. Послаблення базової лінії є важливим у ситуаціях, коли замовлення клієнтів перевищують місткість автомобіля.

Random VRP – VRP із випадковими даними. Деякі компоненти проблеми можуть мати довільну поведінку, наприклад: час у дорозі (VRP із випадковим часом у дорозі (VRPST)), набір клієнтів (VRP із випадковими клієнтами (VRPSC)) [34], час обслуговування клієнта (VRP із випадковий час обслуговування (VRPSST)), попит споживачів (VRP with random demand (VRPSD)) [34]. Існує два поширених підходи до вирішення SVRP. Перший етап дає рішення без урахування випадкових величин. Наступний етап полягає в коригуванні існуючого рішення після розрахунку випадкових значень.

Нечітка VRP (FVRP) — це проблема нечіткої маршрутизації. У практичних застосуваннях дуже важко отримати чіткі значення параметрів. Проблеми, що містять елементи невизначеності та суб’єктивності, можна успішно моделювати за допомогою методів теорії нечітких множин.

Динамічний VRP (DVRP) — це динамічне завдання маршрутизації транспорту. Передбачається, що в процесі вирішення проблеми деякі параметри будуть змінені, а маршрут буде динамічно перераховуватися з урахуванням нової інформації. Зазвичай це поломки автомобіля, пробки, нові замовлення, несподівані дзвінки тощо. Найчастіше розглядається така ситуація, коли нові клієнти можуть з’явитися протягом дня, тобто після виходу ТЗ зі складу [27]. Тому в залежності від рівня обслуговування клієнта змінюються умови виконання завдань і потрібно динамічно перераховувати маршрути з урахуванням нової інформації, що надходить. Зазвичай статистика появи нових замовлень відома заздалегідь, і на цій основі створюються імітаційні моделі для розробки та тестування алгоритму.

Проблема маршрутизації транспортних засобів на супутниковому об’єкті (VRPSF) – проблема маршрутизації транспорту, що супроводжує склади.

Особливістю умов місії є можливість перезавантаження деяких транспортних засобів на маршруті без повернення на склад. Типовим випадком є проблема маршрутизації транспортних засобів на супутникових об’єктах, де багато споживачів очікують регулярних поставок від центрального постачальника [30].

Зелений ВРП (GVRP). Цей варіант VRP має на меті включити в процес оптимізації різні екологічні проблеми, такі як викиди парникових газів, забруднення, відходи та наслідки використання «зеленішої» конфігурації автопарку.

VRP with time window (VRPTW) - VRP з часовим вікном. Кожен клієнт пов’язаний з певним періодом часу та може обслуговуватися лише протягом цього періоду часу. Задіяні параметри: вікно часу (початок і кінець), тривалість обслуговування, тривалість руху між точками, час очікування. Якщо часове вікно не розпочалося до прибуття транспортного засобу, автомобіль чекатиме, поки відкриється пункт обслуговування. Якщо часове вікно закінчується до прибуття транспортного засобу, пункт більше не обслуговується [19]. Ви також можете розглянути можливість налаштування набору часових вікон для кожного клієнта (VRP із кількома часовими вікнами). Крім того, ці часові вікна можна гнучко регулювати з деякими додатковими витратами (VRP з м’якими часовими вікнами) [20].

В останнє десятиліття з’явився новий варіант VRP – Consistent VRP (ConVRP) – послідовний VRP, у якому довіра між Дистриб'юторські компанії та клієнти. Тому компанія встановлює безперервні маршрути протягом певного періоду часу, що зменшує кількість водіїв, з якими доводиться мати справу кожному клієнту, тим самим зменшуючи тертя між ними.

Перелік існуючих типів завдань маршрутизації можна представити у вигляді ієрархічної системи (рисунок А.2). З цього випливає, що основою для більшості типів VRP є базова межа несучої здатності (CVRP), яка може бути нечітко вказана в описі. Очевидно, що за рахунок поєднання розглянутих вище різновидів ВРП можна отримати безліч варіацій їх продукції. Найцікавіші

дослідження стосуються конкретних комбінацій умов і обмежень для визначення нових сфер застосування VRP, таким чином наближаючись до реальних умов.

Приклад на рисунку 1.4 показує зв'язок кількох відомих варіантів VRP, узагальнених у п'ять базових розширень: відкритий (OVRP); з декількома складами (MDVRP); з обмеженнями вантажопідйомності (CVRP); з часовими вікнами (VRPTW), асиметричний (AVRP). У літературі їх назви зазвичай утворюють за принципом додавання букв до абревіатури, що характеризує розширені компоненти. Залежно від загальноприйнятої системи назва проблеми та її абревіатура можуть бути дуже довгими, наприклад, Multi-Depot Capacited Vehicle Routing Problem with Time Window (MDCVRPTW) [13] .

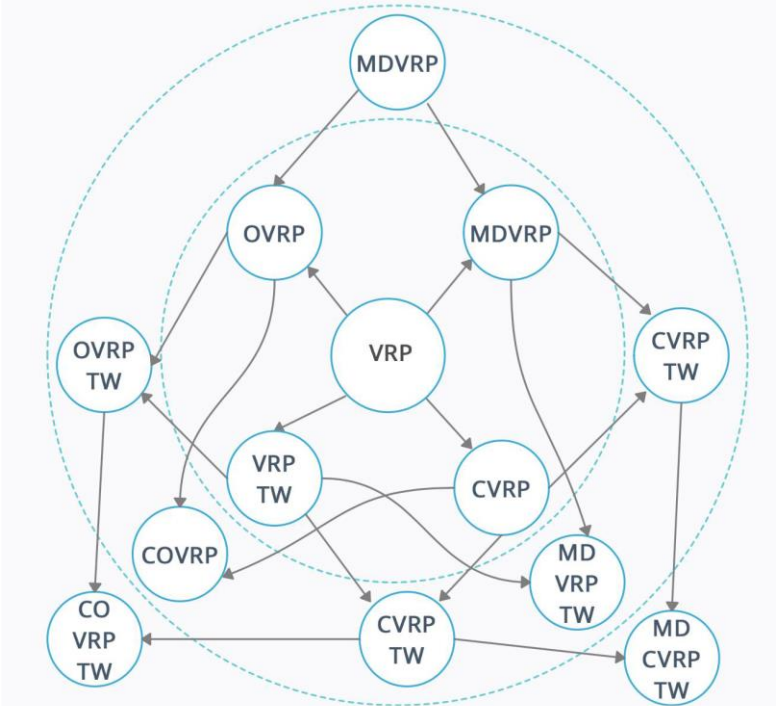

Рисунок 1.4-Схема підключення для узагальнених різновидів ВРП

Однак, незважаючи на те, що інтерес до проблем типу VRP зростає з кожним роком, залишається багато проблем, пов'язаних із:

- Кількість обмежень, які можуть бути застосовані до задачі, оскільки додавання хоча б одного нового обмеження змінює математичну модель задачі, а отже, і алгоритм її вирішення;

- Практична реалізація алгоритму вирішення задачі ВРП, оскільки на практиці при впровадженні ПЗ виникають нові ситуації, які потребують перепроектування.

Тому проблема маршрутизації транспортних засобів актуальна на сьогоднішній день і потребує нового методу розв’язання, який дозволить використовувати відомі алгоритми розв’язання, але буде більш ефективним за певними критеріями.

1.4 Аналіз класичних алгоритмів розв'язання задач маршрутизації транспорту

VRP - це комбінаторна задача оптимізації, де кількість дозволених маршрутів експоненціально зростає зі збільшенням кількості клієнтів і належить до класу складності NP. Точні методи не можуть ефективно розв’язувати задачі з розмірністю більше 14, тому що навіть якщо розмірність задачі відносно мала, сортування можливих рішень для пошуку оптимального рішення потребує значного обчислювального часу. До точних методів відносяться метод розгалужень і меж, метод розгалужень і меж, динамічне програмування та інші методи. Вони є розвитком повних методів пошуку шляхом фільтрації підмножини прийнятних рішень, які, як відомо, не містять оптимального рішення. Однак час їх обчислення все одно зростає занадто швидко і в гіршому випадку – нагадує повний пошук.

На реальних прикладах задач використовуються наближені алгоритми [39]. Існує багато методів вирішення VRP, багато з яких є евристичними та метаевристичними методами, які забезпечують кращий баланс між якістю рішення та часом обчислення. Метаевристика має три основні цілі: прискорити вирішення проблем, обробляти більше вузлів і отримати більш надійні алгоритми. Метаевристика забезпечує прийнятні рішення важких і складних проблем за достатній час.

Початкове створення рішення передбачає створення одного або кількох прийнятних рішень, які стануть відправною точкою для наступного етапу вдосконалення. У звичайному випадку використовується евристичний процес побудови, починаючи з пустого рішення та поступово додаючи компоненти рішення, поки не буде отримано можливе рішення. При цьому найбільш поширеними є жадібні алгоритми з використанням різноманітних евристичних функцій. Процес удосконалення рішення намагається модифікувати початкове можливе рішення на основі визначеної метрики, доки не буде отримано оптимальне рішення або поки алгоритм не закінчить час. Використовуйте поняття «набір рішень» – набір прийнятних рішень/станів, яких можна досягти з поточного стану. Щоб знайти кращі рішення, метаевристики використовують стан із набору рішень для зміни поточного стану. Цей процес відповідає локальному пошуку в просторі станів. Локальний пошук використовує так звані локальні операції, які змінюють один або кілька маршрутів, щоб спробувати отримати краще (дешевше) рішення.

1.4.1 Точні методи

Точний підхід гарантує оптимальні рішення. Однак практичне застосування таких алгоритмів придатне лише для задач малої розмірності [34]. Точні методи базуються на точних математичних алгоритмах. Цей метод розв'язування виділеного класу задач заснований на повному перерахуванні всіх можливих розв'язків і неефективний при розв'язуванні задач великої розмірності через високі витрати часу (час розв'язання експоненціально залежить від розмірності задачі).

Розгалуження і межа (ВМВ) — це метод, заснований на інтелектуальному виборі можливих рішень задачі комбінаторної оптимізації. Принцип полягає в розбитті простору рішень на непересічні підмножини, які представлені вузлами розгалуженого дерева. Потім алгоритм досліджує гілки дерева відповідно до

стратегії маршруту [40]. Щоб уникнути дослідження всього дерева, перед створенням нового вузла в дереві алгоритм оцінює вузол, порівнюючи значення найкращого можливого рішення, яке можна знайти у відповідному піддереві, з поточним найкращим рішенням [13]. Якщо краще рішення не може належати до піддерева, яке має корінь у даному вузлі, піддерево відкидається. В іншому випадку процес розвідки продовжиться. Багато задач комбінаторної оптимізації можна розв'язати більш-менш ефективно за допомогою стратегій MGM.

Метод обрізання гілок

Основна ідея методу розрізу гілки (BCT) полягає в тому, щоб зменшити простір пошуку кандидатів шляхом додавання нових обмежень. Загалом, алгоритм MGV працює подібно до MGM, але з додатковим кроком порушення обмежень.

Динамічне програмування (Dynamic Programming)

Динамічне програмування (DPP) відноситься до процесу спрощення складних проблем шляхом рекурсивного розкладання їх на менші підпроблеми. Як правило, проблеми, які розв’язуються за допомогою динамічного програмування, можна розділити на кілька етапів, на кожному з яких приймаються рішення. Кожен етап також має багато станів, пов'язаних з ним [35]. Рішення на одному етапі перетворюють поточний стан у стан наступного етапу. Рішення про перехід до наступного стану залежить лише від поточного стану, а не від попередніх станів чи рішень. Низхідне динамічне програмування засноване на збереженні результатів певних обчислень для подальшого використання [36]. Динамічне програмування «знизу вгору» рекурсивно перетворює складні обчислення на серію простіших обчислень.

1.4.2 Евристичні методи

Евристика — це набір методів пошуку рішень проблеми, які обмежують пошук, тобто виконують відносно обмежений пошук у просторі рішень і зазвичай знаходять рішення в межах діапазону. прийнятний час. Недоліком цих методів є те, що вони є наближеними і отримані цими методами рішення можуть бути далекими від оптимальних. Перевага полягає в тому, що вони можуть знайти прийнятні рішення для великовимірних NP-повних проблем за прийнятний час [37]. Евристичні методи часто протиставляються формальним методам вирішення, Це залежить від точних математичних алгоритмів.

Евристика поділяється на три основні категорії.

Конструктивний алгоритм. Будуйте вузли поступово, відстежуючи зростання їх вартості, але без подальших етапів уточнення: найближчий сусід, жадібний, збереження, вставка.

Двоступеневий (кластерний) алгоритм. Завдання розбивається на дві операції – групування (кластеризація) вершин кожного майбутнього маршруту та виконання рішення TSP для кожної отриманої групи: (алгоритм Фішера та Джайкумара, алгоритм Sweep, алгоритм Petal, алгоритм Османа). У двоетапному алгоритмі можливий зворотний зв'язок між етапами розв'язання. (пелюстка, розмах, кластер перший, маршрут другий, маршрут перший, кластер другий).

вдосконалити алгоритм. Спочатку знайдіть деякі рішення, а потім спробуйте поміняти місцями вершини (ребра) в межах кожного маршруту або між маршрутами: (покращена евристика для багатьох маршрутів). (внутрішня траса, внутрішня траса, к-опт, 2-опт, 3-опт).

1.4.3 Метаевристичні методи

Слово «метаевристика» походить від грецького прийменника «мета» (що означає «вищий рівень») і «евристика» (від грец. ευρίσκειν, «пошук»), вперше введені в 1986 році Фредом Гловером [6], позначають алгоритмічні схеми, спрямовані на досягнення вищих рівнів Ефективне дослідження складних задач оптимізації у просторах пошуку [40]. Метаевристика описує великий (і основний) підрозділ стохастичної оптимізації — великий клас алгоритмів і методів, які певною мірою використовують стохастичність для пошуку оптимальних (майже оптимальних або субоптимальних) рішень складних проблем. Метаевристика — це сучасний, потужний і надзвичайно популярний клас методів оптимізації, який може знаходити рішення для різноманітних проблем. Перевагою метаевристик є їх здатність розв’язувати складні задачі без знання простору пошуку, тому ці методи дають змогу розв’язувати складні задачі оптимізації [13]. Метаевристичні методи базуються на ретельному дослідженні найбільш перспективних ділянок простору рішень. Ці розв'язки кращі, ніж отримані за допомогою класичної евристики. Характерною особливістю метаевристичних алгоритмів є те, що вони не забезпечують точного опису послідовності дій для вирішення задачі, а кожна послідовність дій додатково специфікується шляхом вибору значень керуючих параметрів.

Іншими словами, метаевристика — це метод вирішення обчислювальних проблем шляхом поєднання існуючих програм із відкритими інтерфейсами та закритими реалізаціями для отримання найбільш ефективного рішення. Метаевристика часто використовується для вирішення проблем, для яких немає задовільного алгоритму для конкретної проблеми, або коли немає практичної необхідності реалізувати такий алгоритм. Найчастіше метаевристики використовуються для вирішення задач комбінаторної оптимізації, але їх також можна використовувати для будь-якої іншої задачі, яку можна звести до вирішення логістичного рівняння. На відміну від традиційних алгоритмів оптимізації та ітераційних методів, метаевристики не можуть гарантувати, що

певні типи проблем можуть знайти оптимальні рішення глобально через випадковість. Однак при пошуку великої кількості прийнятних рішень метаевристики часто можуть знайти успішні рішення за низьких обчислювальних і часових витрат.

Метаевристики мають такі характеристики :

- Метаевристика - це стратегія управління процесом пошуку;

- Метою метаевристики є ефективне дослідження простору пошуку для пошуку (не)оптимальних рішень;

-Методи реалізації метаевристичних алгоритмів варіюються від простого локального пошуку до складних процесів навчання;

- Метаевристики є наближеними і часто недетермінованими;

- Метаевристики включають механізм запобігання потраплянню в локальні оптимуми;

- Основні положення метаевристики допускають абстрактний опис;

- Метаевристика може використовувати проблемно-орієнтоване знання у формі евристики, що керується стратегіями високого рівня;

- Розширена метаевристика використовує досвід, накопичений під час процесу пошуку, і представляє його у вигляді пам'яті для керування пошуком.

Основна частина літератури з метаевристики має експериментальний характер і описує емпіричні результати, засновані на імітаційному моделюванні та експериментах з алгоритмічним програмним забезпеченням. Однак існують також деякі формальні теоретичні результати, що демонструють можливість знаходження глобального оптимуму

.

Майже всі метаевристики засновані на спостереженнях ідей, що виникають у живих і неживих природних процесах.

Імітований відпал (SA) є добре відомим метаевристичним методом пошуку, який успішно застосовувався до багатьох задач комбінаторної оптимізації. Це алгоритм підйому на гору, який збільшує можливість пробиття локального оптимуму в просторі пошуку. Однак, незважаючи на те, що це

чудове рішення, воно дуже повільне порівняно з простим процесом сходження на пагорб.

Термін «імітований відпал» базується на фізичному процесі твердого відпалу, коли відбувається кристалізація речовини, і ми намагаємося мінімізувати енергію системи шляхом повільного охолодження, доки атоми не досягнуть стабільного стану. Техніка повільного охолодження дозволяє атомам металу розташуватися і сформувати правильну кристалічну структуру, яка має високу щільність і низьку енергію.Перехід атомів з однієї комірки в іншу відбувається з певною ймовірністю, і ймовірність зменшується при меншій кількості атомів. температура. Початкова температура і швидкість охолодження називаються графіком відпалу.

Використовуючи SA для вирішення задачі комбінаторної оптимізації (CO), ми починаємо з можливого вирішення проблеми. Потім ми спробували оптимізувати це рішення за допомогою методу, схожого на твердотільний відпал. Використовуйте відповідний метод, щоб створити сусідів з'єднання та обчислити вартість (або прийнятність) нового з'єднання. Нові рішення приймаються, якщо вони кращі за поточне рішення з точки зору зниження вартості (або покращеної адаптивності). Якщо нове рішення не краще поточного рішення, нове рішення приймається з певною ймовірністю. Імовірність прийняття зазвичай встановлюється як $\exp(-\Delta/T)$, де Δ — зміна вартості між старим і новим рішеннями, а T — поточна температура. Тому, коли трафік погіршується, ймовірність експоненціально зменшується.

Порівняно з простим підйомом на пагорб, процес SA має меншу ймовірність застрягти в локальному оптимумі, оскільки погані шляхи все ще мають шанс бути здійсненними. Спочатку, якщо обрано високу температуру відпалу, ймовірність прийняття буде високою, і майже всі нові розчини будуть прийняті. Потім температуру поступово знижують, тому ймовірність прийняття рішення низької якості дуже мала. Алгоритм працює більш-менш як підйом на пагорб, тобто вищі температури дозволяють краще досліджувати простір пошуку, а нижчі — краще досліджувати простір пошуку простір. Відкоригуйте

розчин. Цей процес повторюється до тих пір, поки температура не наблизиться до нуля або поки неможливо досягти подальшого поліпшення. Це подібно до того, як тверді атоми досягають кристалічного стану. На малюнку 1.5 показано, як випадкові зсуви вгору дозволяють алгоритму SA уникнути локальних оптимумів у задачах мінімізації.

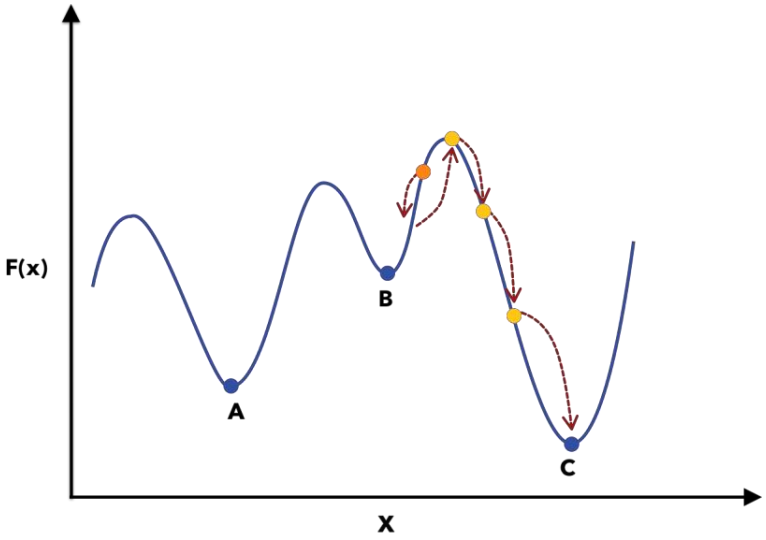

Рисунок 1.5 - Імітація відпалу: випадковий підйом і відхилення від локального оптимуму

Застосування SA до проблем CO також вимагає вибору загальних параметрів SA та конкретних рішень. Вибір параметрів SA має вирішальне значення для продуктивності алгоритму. Ці параметри: значення початкової температури T, функція температури, яка визначає, як температура змінюється з часом, кількість ітерацій N(t), виконаних при кожній температурі, і критерій зупинки для завершення алгоритму.

Генетичний алгоритм (Genetic Algorithm)

Генетичний алгоритм (GA) — добре відомий метаевристичний алгоритм, який наразі привертає все більше уваги у всьому світі. Генетичні алгоритми — це еволюційні алгоритми пошуку, які використовують механізми природного відбору та природної генетики для пошуку рішень проблем.

Генетичні алгоритми створюють нові покоління нащадків за допомогою ітераційного процесу, розвиваючи популяцію особин, закодованих у хромосомах, доки не будуть виконані певні критерії конвергенції. Такі критерії можуть включати, наприклад, максимальну кількість поколінь, конвергенцію до однорідної групи подібних індивідів або отримання оптимального рішення. Отриману оптимальну хромосому потім декодують, щоб забезпечити відповідне рішення.

Генетичні алгоритми працюють з популяціями індивідуумів рішення, а не лише з одним рішенням, тому вони виконують багатосторонній пошук одночасно. Кожна людина представляє потенційне рішення проблеми.

Створення нового покоління особин складається з трьох основних етапів.

Етап відбору включає випадковий вибір двох батьків із популяції для спаровування. Імовірність вибору представника популяції, як правило, пропорційна його придатності для підкреслення генетичної якості при збереженні генетичного різноманіття. Тут придатність відноситься до міри прибутку, корисності або блага, які слід максимізувати під час дослідження простору рішення.

На етапі рекомбінації або розведення використовуються гени від обраних батьків для отримання потомства, яке сформує наступне покоління.

Етап мутації — це одночасна випадкова модифікація певних генів індивідуума для подальшого вивчення простору рішення та забезпечення або збереження генетичного різноманіття. Як правило, мутації відбуваються рідше. Тому для забезпечення різноманітності популяції використовуються процедури мутації.

Нове покоління створюється шляхом повторення процесу відбору, розмноження та мутації, поки всі хромосоми в новій популяції не замінять хромосоми в старій популяції. Тому для забезпечення ефективного пошуку необхідний відповідний баланс між генетичною якістю та різноманітністю в популяції.

Меметичний алгоритм (MA) є «покращеною» версією GA, у якій локальні методи вдосконалення застосовуються до кожного нащадка. Обидва алгоритми належать до вищої категорії еволюційних алгоритмів.

Табу Пошук (Табу Пошук)

Основні поняття Табу-пошук (TS) — це один із найпоширеніших алгоритмів, який базується на принципі продовження пошуку, навіть якщо досягнуто локального оптимуму. TS досліджує простір рішень, переходячи від рішення до найкращого рішення в підмножині його сусідства на кожній ітерації. На відміну від класичних методів регресії, поточне рішення може погіршуватися від однієї ітерації до наступної. Тому, щоб уникнути петель, рішення з певними властивостями нещодавно досліджених розчинів тимчасово пригнічуються або пригнічуються. Тривалість часу, протягом якого атрибут залишається табуйованим, називається його директивою табу, і вона може змінюватися через різні проміжки часу. Статус табу можна змінити, коли виконуються певні умови; це називається критерієм бажаності, який виникає, наприклад, коли рішення табу є кращим за будь-яке раніше знайдене рішення.

Тому схильність відхилятися від очікуваного шляху може мати як негативний, так і позитивний вплив на пошук. Так працює метод Табу, за винятком того, що новий шлях вибирається не випадково. Навпаки, пошук табу працює на основі припущення, що немає сенсу приймати нові рішення, якщо не уникнути шляхів, які вже були досліджені. Це гарантує, що нові регіони простору вирішення проблеми досліджуються, щоб уникнути локальних мінімумів і в кінцевому підсумку знайти бажане рішення.

Початкове рішення зазвичай створюється з використанням найдешевших доступних евристик. Після створення початкового рішення спробуйте вдосконалити рішення за допомогою локального пошуку з однією або кількома структурами сусідства та оптимальною стратегією.

Пошук у змінному околиці (Variable Neighborhood Search)

Variable Neighborhood Search (VNS) є відносно «молодим» метаевристичним алгоритмом, запропонованим Гансеном і Младеновичем. Основна ідея базується на дослідженні простору пошуку шляхом поступового збільшення розміру околиці, генерування нових рішень у околиці до досягнення певної умови зупинки. Крім того, щоразу, коли в поточному районі створюється нове рішення, до рішення застосовується локальний пошук для його оптимізації, перш ніж буде прийнято рішення про заміну.

Фази евристики пошуку змінної околиці включають фазу уточнення (для можливого покращення даного рішення) і так звану фазу струшування (для вирішення локальних мінімумів). Фаза уточнення, процес струшування та фаза зміни сусідства виконуються послідовно, доки не будуть виконані попередньо визначені критерії зупинки. Іншими словами, наступні три кроки повторюються, доки не буде виконано певний критерій зупинки:

a) Процедура струшування; b) Процедура покращення; c) Етапи зміни сусідства.

Метаевристика VNS пропонує багато привабливих функцій, які спонукають дослідників використовувати її для вирішення проблем CO. По-перше, алгоритм простий, легкий для розуміння та впровадження, і він значною мірою не залежить від параметрів. По-друге, добре відомі методи обходу сусідства та локального пошуку можна легко інтегрувати та застосовувати у VNS різними способами. По-третє, це надійна техніка, яку можна використовувати для різних типів задач без необхідності змінювати основну структуру алгоритму. Нарешті, її можна легко поєднувати з іншими евристичними та метаевристичними методами для вирішення складних проблем оптимізації.

Оптимізація колонії мурашок

Оптимізація колонії мурашок (ACO) — це метаевристичний метод, натхненний поведінкою справжніх мурах.

Коли ACO використовується для вирішення задачі комбінаторної оптимізації, проблему можна візуалізувати як пошук найкоротшого шляху у

зваженому графі. Штучні мурахи (програмні агенти) співпрацюють, щоб знайти найкращий шлях, створюючи рішення, що складається з вузлів графа поетапно. Кожен крок додає вибрані вузли до часткового рішення випадковим чином, але зміщено кількістю феромону, доступного у вузлі або на краях, що з'єднують вузли. Шлях феромонів оновлюється мурахами на кожній ітерації, дозволяючи мурахам співпрацювати у пошуку хорошого вирішення проблеми. Значення феромонів також зменшуються з часом, подібно до випаровування справжніх феромонів. Тому в ході пошуку неякісні рішення надалі розглядатися не будуть.

Застосування алгоритму мурашиної колонії в складних задачах комбінаторної оптимізації є привабливим завдяки гнучкості та надійності алгоритму. Фактично, методика може бути застосована для вирішення широкого кола завдань з мінімальними змінами в основному алгоритмі.

Останнім часом запропоновано багато алгоритмів для вирішення різноманітних задач задач комбінаторної оптимізації. У той же час, алгоритм ACO в поєднанні зі спеціалізованими локальними процесами виявився дуже ефективним для пошуку рішень симетричних і асиметричних задач комівояжера (TSP/ATSP), задач послідовного ранжирування. (SOP), квадратична проблема розподілу (QAP]), біквадратична задача розподілу. Одним із найефективніших ACO є Ant Colony System (ACS), яка містить спеціальну програму для оновлення слідів феромонів, що активує пошук прилеглих територій для кращих рішень.

Дослідники намагаються вирішити VRP найкращим чином, поєднуючи алгоритми та евристики, щоб використовувати сильні сторони кожного процесу. Особливо за останні кілька років для досягнення кращих результатів були розроблені гібридні алгоритми, що включають метаевристики.

1.5 Огляд існуючих програмних продуктів

Програмні продукти, які вирішують задачі маршрутизації транспортних засобів, можна розділити на такі категорії:

– Sterling Transportation Management System (IBM) – оптимізує ланцюг поставок транспортних мереж шляхом планування та аналізу вхідних і вихідних процесів транспортування;

– ArcLogistics – інструмент для планування та оптимізації роботи автопарку: імпорт замовлень, розрахунок оптимальних маршрутів, створення маршрутних таблиць, звітність та аналіз ефективності роботи;

– ANTOR LogisticsMaster – це програмний продукт, який дозволяє автоматично планувати доставку ваших товарів за різними критеріями маршруту, враховуючи модель транспортної мережі, транспортну доступність певних міських районів, реальне «вікно доставки», об'єм і вага продукту тощо;

Тому задача маршрутизації транспортних засобів є актуальною на сьогоднішній день, враховуючи наявність певної кількості алгоритмів і програмних продуктів. Однак проблема VRP все ще вимагає нового підходу, який поєднує відомі алгоритми вирішення та їх модифікації та використовує їх переваги для підвищення ефективності.

1.6 Постановка задачі дослідження

Метою магістерської роботи є підвищення ефективності методів вирішення задач маршрутизації транспортних засобів з часовими вікнами.

Для досягнення поставлених цілей необхідно вирішити наступні питання:

- Аналізувати відомі результати при розв'язанні задач маршрутизації транспортних засобів;

- Удосконалення існуючих алгоритмів розв'язування задач маршрутизації транспортних засобів з урахуванням часових вікон за рахунок модифікацій та комбінацій метаевристик;

- Розробляти програмну реалізацію розроблених алгоритмів;

- Провести дослідження ефективності розробленого алгоритму.

Об'єктом дослідження є процес організації перевезень.

Предметом даного дослідження є завдання маршрутизації транспортного засобу з часовими вікнами.

Таким чином, надається огляд існуючих типів проблем маршрутизації транспортних засобів та їх вирішення. Підвищення ефективності транспортного планування за рахунок мінімізації загальних транспортних витрат є одним із найважливіших критеріїв оптимізації організаційної роботи у сфері транспортної логістики, тому можна зробити такі висновки: Важливість завдань курсу ВРП, практичний інтерес до завдання курсу. З метою підвищення ефективності відомих алгоритмів розв’язування таких задач сформульовано відповідну постановку задачі та мету дослідження.

2 РОЗРОБКА МОДИФІКАЦІЙ АЛГОРИТМУ МУРАШИНИХ КОЛОНІЙ ТА ГЕНЕТИЧНОГО АЛГОРИТМУ ДЛЯ ЗАДАЧІ VRPTW

Проблема маршрутизації транспортного засобу з часовими вікнами (VRPTW) є розширенням проблеми маршрутизації транспортного засобу з обмеженою пропускною спроможністю (CVRP), де кожного клієнта необхідно обслуговувати протягом відповідного періоду часу, який називається часовим вікном. Часові вікна можуть бути жорсткими або м’якими. У випадку жорстких вікон транспортні засоби, які прибувають до клієнта передчасно, повинні чекати, доки клієнт не буде готовий розпочати обслуговування. Очікування до початку часового вікна не має жодних витрат, крім часу. У випадку з м’якими часовими вікнами кожне часове вікно може бути порушено та збільшити штрафні витрати. Часові вікна можуть бути односторонніми, наприклад із зазначенням кінцевого часу доставки.

Часові проміжки в реальному житті виникають через проблеми, з якими стикаються бізнес-організації, які працюють за гнучким графіком. Конкретні застосування жорстких вікон включають патрулювання безпеки, банківську пошту, доставку пошти, збір промислових відходів, доставку продуктів, маршрути шкільних автобусів і розповсюдження міських газет. У цьому розділі використовується найбільш вивчений варіант жорстких часових вікон.

Проблема маршрутизації транспортного засобу з часовими вікнами (VRPTW) передбачає пошук транспортних маршрутів, які відповідають потребам клієнта в продуктах, враховуючи часові рамки обслуговування клієнта та мінімізуючи загальну вартість транспортування. Якщо автомобіль прибуває в пункт обслуговування до початку інтервалу часу, встановленого клієнтом, транспортний засіб повинен чекати, поки клієнт буде доступний для обслуговування.Якщо транспортний засіб прибуває в пункт обслуговування після інтервалу часу, встановленого клієнтом, клієнт не може відремонтувати.

Припустимо, що є K транспортних засобів, що перевозять вантажі, і початкова та кінцева точки всіх транспортних засобів знаходяться на місці.

В існуючій літературі VRPTW зазвичай передбачається, що кількість транспортних засобів, доступних для обслуговування клієнтів, є необмеженою, а цільова функція залежить від обраного методу вирішення. Для точного методу метою є мінімізація загальної пройденої відстані. Для евристичного підходу основною метою також є мінімізація кількості використовуваних транспортних засобів, а вторинною метою є мінімізація загальної пройденої відстані. З цього загального твердження можуть бути винятки.

2.1 Математична постановка задачі VRPTW

Математична модель задачі маршрутизації транспортного засобу з урахуванням часових вікон представлена такою формулою:

$$\sum_{i=0}^{n}\sum_{j=0,j\neq i}^{n}\sum_{k=1}^{K} c_{ij}x_{ijk} \rightarrow min, \tag{2.1}$$

$$\sum_{k=1}^{K}\sum_{j=1}^{n} x_{0jk} \leq 1, \tag{2.2}$$

$$\sum_{j=1}^{n} x_{0jk} = \sum_{j=1}^{n} x_{j0k} \leq 1, \text{ для } k\epsilon\overline{1,K}, \tag{2.3}$$

$$\sum_{k=1}^{K}\sum_{j=0,j\neq i}^{n} x_{ijk} = 1, \text{ для } i\epsilon\overline{1,n}, \tag{2.4}$$

$$\sum_{k=1}^{K}\sum_{i=0,i\neq j}^{n} x_{ijk} = 1, \text{ для } j\epsilon\overline{1,n}, \tag{2.5}$$

$$\sum_{i=1}^{N} q_i \sum_{j=0, j\neq i}^{N} x_{ijk} \leq Q, \qquad \text{для } k\epsilon\overline{1,K}, \tag{2.6}$$

$$\sum_{i=1}^{n} \sum_{j=0, j\neq i}^{n} x_{ijk}\left(w_i + s_i + t_{ij}\right) \leq T_k, \quad \text{для } k\epsilon\overline{1,K}, \tag{2.7}$$

$$t_0 = w_0 = s_0 = 0, \tag{2.8}$$

$$x_{ijk}\left(t_i + w_i + s_i + t_{ij}\right) \leq t_j, \quad \text{для } i\epsilon\overline{1,n}, j\epsilon\overline{1,n}, (i \neq j), k\epsilon\overline{1,K}, \tag{2.9}$$

$$e_i \leq (t_i + w_i) \leq l_i, \quad \text{для } i \in \overline{1,n}, \tag{2.10}$$

Серед відомих місць:

cij – вартість переміщення від i до j, (i,j) ∈ N;

tij – тривалість руху від i до j, (i,j) ∈ N;

qi – попит на позицію i, i ∈ N;

si – тривалість обслуговування в точці i, i ∈ N;

ei – початок часового вікна в точці i, i ∈ N;

li – кінцева точка часового вікна в точці i, i ∈ N;

K – Загальна кількість транспортних засобів;

Tk – максимальна тривалість (час) маршруту k-го автомобіля, k ∈ K;

Q – місткість автомобіля;

N, n+1 – кількість точок, включаючи станції.

Змінні такі:

ti – час досягнення точки i, i ∈ N;

wi – час очікування в точці i, i ∈ N.

Xijk = {1, якщо k-й транспортний засіб обслуговує маршрут (i, j); 0 інакше}.

Отже, компоненти моделі такі:

(2.1) – цільова функція мінімізації транспортних витрат (2.2) – k-й автомобіль може виїхати зі стоянки тільки один раз;

(2.3) – Якщо k-й автомобіль виїжджає зі стоянки, він повинен туди повернутися (2.4) – i-та точка k-го автомобіля може виїхати лише один раз;

(2.5) – В j-ту точку k-го транспортного засобу можна в’їхати лише один раз (2.6) – Місткість транспортного засобу;

(2.7) – загальний час перебування k-ої вантажівки на маршруті не може перевищувати максимальної тривалості маршруту k-ої вантажівки;

(2.8) – початкові умови;

(2.9) – Добуток часу прибуття і-го пункту, часу очікування і-го пункту, часу обслуговування і-го пункту та часу руху від і-го пункту до і -та точка не може бути більшою за j-ту точку Час прибуття до j точок;

(2.10) – Час прибуття точки і та час очікування точки і повинні входити в часове вікно точки і.

У літературі запропоновано різні підходи до вирішення задачі VRPTW: точні та наближені.

Через свою складність проблема VRPTW є NP-повною. Для розв’язування таких задач із розмірністю більше 20 вузлів точні методи можна використовувати лише в тому випадку, якщо час на пошук розв’язку є незначним, оскільки вибір можливих розв’язків для пошуку оптимального розв’язку займає багато обчислювального часу, навіть якщо проблема розмір Відносно невеликий.

Для вирішення практичних завдань використовуються евристичні та метаевристичні методи. Евристичні методи можна використовувати для знаходження наближених рішень за розумний час, але хоча вони можуть бути виявлені емпірично, вони не гарантують високоякісних рішень. Метаевристики базуються на ретельному вивченні найбільш перспективних частин простору рішень, що допомагає підвищити якість порівняно з класичними евристиками, щоб за допомогою метаевристик можна було досягти в розумний час Знайти прийнятне рішення.

Гібридні алгоритми поєднують різні ідеї евристики або метаевристики та широко використовуються для вирішення різних задач комбінаторної оптимізації. У цьому документі пропонується вдосконалений алгоритм оптимізації колонії гібридних мурах у поєднанні з алгоритмом пошуку табу.

Цей алгоритм показує хорошу продуктивність і результати на відомому наборі даних Solomon [9].

2.2 Алгоритми Ant Colony Optimization для розв’язання VRPTW

2.2.1 Опис класичного алгоритму Ant Colony Optimization

В даний час в рамках напрямку «природні обчислення» активно створюються і вдосконалюються методи глобальної оптимізації, навіяні природними системами. Ці методи штучного інтелекту відображають механізми прийняття рішень та адаптації та є природним результатом мільйонів років еволюції. Подібним чином мурахи, які існують понад 100 мільйонів років, пристосувалися виживати в різних умовах майже скрізь у світі, утворюючи популяцію з 1016 особин, загальна вага яких становить від 10% до 25% біомаси наземних тварин [62]. Поведінка кожної особини в колонії мурашок є відносно простою, представляє складну соціальну структуру та здатна вирішувати складні завдання, такі як пошук найкоротшого шляху за допомогою хімічних коригувань. На рисунку 2.1 графічно зображено поведінку справжніх мурашок у пошуках джерел їжі.

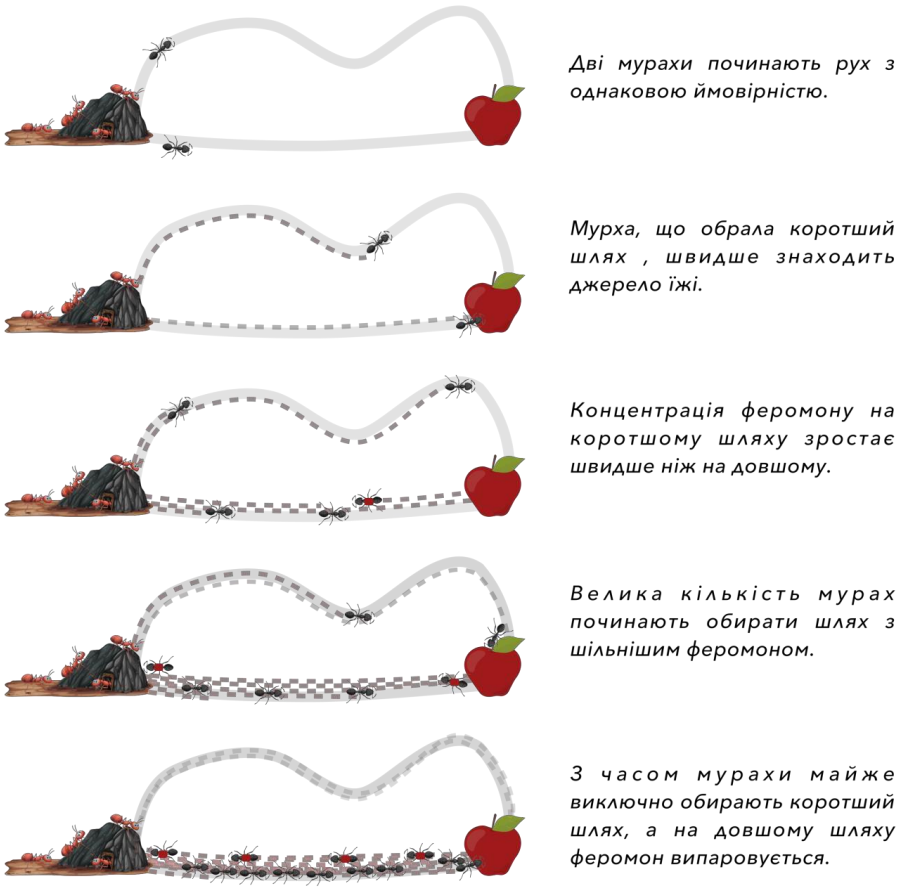

Рисунок 2.1 – Поведінка мурах під час пошуку джерел їжі

Сліди феромонів дозволяють мурахам знаходити нові маршрути, коли наявні маршрути стикаються з перешкодами. Якщо найкоротший шлях недоступний через перешкоди, мурахи повинні знайти новий найкоротший шлях. Спочатку вони з однаковою ймовірністю обходили перешкоди з обох боків на шляху до їжі та на зворотному шляху. Ті мурахи, які оберуть коротший шлях, рухатимуться по ньому швидше в обох напрямках, тому він буде багатий феромонами. Оскільки чим вища концентрація феромону на шляху, тим більша ймовірність, що мурахи підуть цим шляхом, і наступні мурахи з більшою ймовірністю виберуть цей шлях, збагачуючи шлях феромоном, поки феромон ще доступний. (Малюнок 2.2).

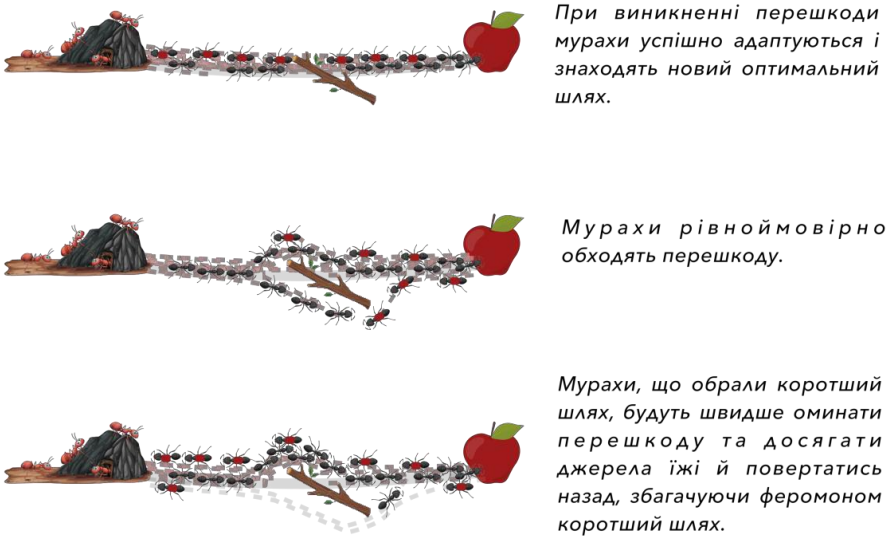

Рисунок 2.2 – Поведінка мурах при виникненні перешкод

Така поведінка мурах привернула увагу багатьох дослідників, які вивчають механізми взаємодії між особинами в мурашиних колоніях. Експериментуючи з вибором між двома шляхами різної довжини від гнізда до джерела їжі, біологи помітили, що найчастіше мурахи обирали коротший шлях.

Модель такої поведінки така:

- Спочатку мурахи безладно рухаються від гнізда до джерела їжі;

- Коли мураха знаходить їжу, вона повертається в гніздо, залишаючи слід особливого ферменту - феромону;

- Інші мурахи сприймають «запах» відкладених феромонів і намагаються йти по вказаному шляху;

- Сліди феромонів генерують асинхронну та непряму комунікаційну схему, за допомогою якої мурахи можуть опосередковано обмінюватися інформацією один з одним, коли виявлено альтернативний шлях до їжі: чим вища концентрація феромону на шляху, тим більша ймовірність, що інші мурахи підуть ним;

- Коротші шляхи займають менше часу, і тому легше позначаються феромонами, ніж інші шляхи, а висока концентрація феромонів, у свою чергу,

робить короткі шляхи більш привабливими, залучаючи все більше і більше мурах;

- Довгі шляхи, які рідко використовуються, з часом зникнуть через зменшення кількості феромону під час випаровування.

Завдяки описаним механізмам позитивного та негативного зворотного зв’язку ця непряма комунікація в колонії мурах називається «стигмергією» та пропонує можливість ідентифікувати та ремонтувати найкоротші шляхи. При цьому можна виділити наступні основні принципи поведінки колонії мурашок [61]:

- Рухи кожної мурашки прості;

- Відсутність централізованого контролю;

- Використовуйте осадження феромонів для непрямого обміну інформацією;

- Феромони з часом випаровуються, що забезпечує адаптивну поведінку.

Мурашиний алгоритм, також відомий як алгоритм оптимізації мурашиної колонії, є відомим метаевристичним алгоритмом, пов’язаним із методами розвідки роїв. Його суть полягає в аналізі та використанні моделі, описаної поведінкою мурашиної колонії. Мурашині колонії пов'язані з вирішенням різноманітних задач пошуку маршрутів на графах. Перший мурашиний алгоритм був запропонований Марко Доріго в його докторській дисертації в 1992 році для завдання пошуку найкращого шляху на графі [9]. Основні етапи мурашиного алгоритму загалом показані на структурній схемі на малюнку 2.3. Віртуальні мурахи пересуваються по краях графіка, постійно будуючи рішення проблеми. У той же час ребра з малою довжиною та великою кількістю феромонів (заданих певним значенням умови для кожного ребра) вибираються так, щоб мати найбільшу ймовірність руху.

Феромон оновлюється на кожній ітерації: він збільшується на побудованому маршруті пропорційно його довжині та зменшується на всьому графіку з урахуванням коефіцієнта випаровування.

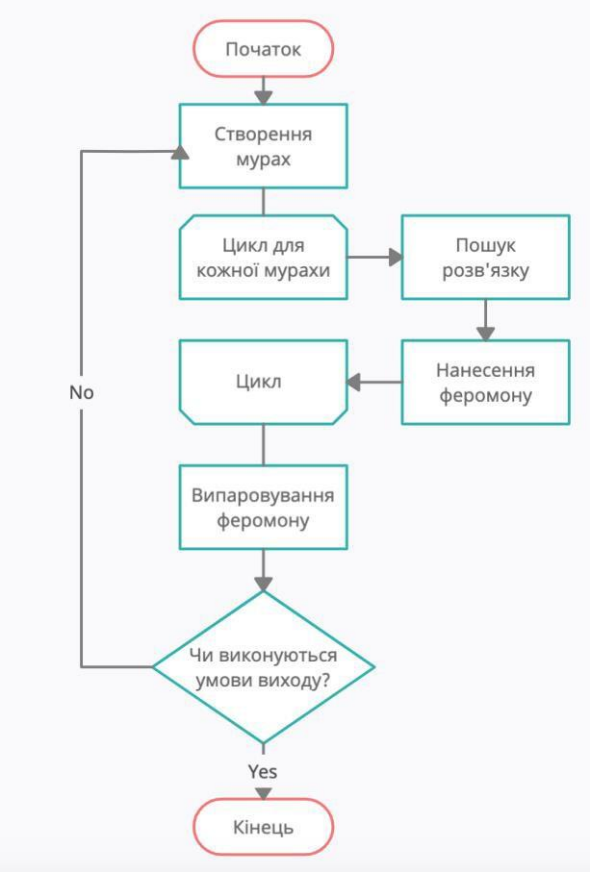

Рисунок 2.3 – Схема структури загального вигляду алгоритму мурашиної колонії

Згодом більшість мурах вибере найкоротший шлях, оскільки на ньому накопичилася велика частка феромонів, що є своєрідним позитивним зворотним зв'язком. Щоб уникнути попадання в локальний оптимум, був також змодельований негативний зворотний зв'язок у вигляді випаровування феромонів. Коли поведінка мурах моделюється на графіку, де краї — це можливі шляхи, якими мурахи можуть пройти, найбільш багаті феромонами шляхи вздовж країв графіка — це рішення проблеми, тобто результати роботи ACO.

Проблема конвергенції мурашиного алгоритму має теоретичне значення для наукових досліджень. На даний момент мурашиний алгоритм має докази гарантованої збіжності, але через його випадковий час збіжність невизначена [61].

Узагальнена схема алгоритму

Алгоритм мурашки, незалежно від його модифікації, може бути виражений у такій формі:

Крок 1, доки умови виходу не виконуються, повторюйте кроки:

Крок 1.1 Створіть Ant

Крок 1.2 Знайдіть рішення

Крок 1.3 Оновіть феромони

Крок 1.4 Додаткові операції (необов'язково) Створення мурах

Умови завдання визначають вихідну точку для розміщення мурашок, яка є визначальною для кожного завдання. У вас є такі варіанти розміщення осіб: почати з однієї точки, почати з іншої точки (повторити чи ні).

При створенні мурах невелике позитивне число визначає початковий рівень феромону, щоб запобігти нульовій ймовірності переходу на початкових етапах.

Знайдіть рішення

У кожен момент ймовірність переходу між вершинами визначається наступною формулою:

$$p_{ij}(t) = \frac{\tau_{ij}(t)^{\alpha}\left(\frac{1}{d_{ij}}\right)^{\beta}}{\sum_{j \in J_i} \tau_{ij}(t)^{\alpha}\left(\frac{1}{d_{ij}}\right)^{\beta}}, \qquad (2.11)$$

Ji — множина вершин, яким дозволено перехід від вершини i;

□ij (t) – рівень феромону на дузі i - j на кроці t;

dij – відстань між вершинами i та j;

альфа і бета є постійними параметрами.

Коли α=0, найімовірніше буде обрано найближче місто, тобто алгоритм стає жадібним. Коли □=0, феромон є єдиним параметром вибору, що призводить до локального оптимуму (субоптимального рішення). Тому компроміс між цими значеннями необхідно знайти експериментально [62].

Оновити феромони

Рівні феромонів оновлюються за такою формулою:

$$\tau_{ij}(t+1) = (1-\rho)\tau_{ij}(t) + \sum_{k \in M_{ij}} \frac{L_{\min}}{L_k}, \qquad (2.12)$$

ро - інтенсивність випаровування;

Mij - безліч мурах, що проходять по дузі (i, j);

Lk - вартість поточного перетину k-ої мурахи (довжина маршруту);

Lmin – параметр, значення якого є величиною вартості оптимального рішення (прогнозована ідеальна ціна рішення в мінімальній задачі);

Lmin/Lk – це феромон, який виділяє k-та мураха на краю (i, j).

Отже, чим гірший розчин потрапляє в край (i, j) (відповідно більший Lk), тим менше феромону мурахи відкладають на цьому краю [62].

Додаткові операції

Як правило, на цьому етапі використовуються алгоритми локального пошуку, але вони також можуть бути використані на завершальному етапі після пошуку всіх рішень.

2.3 Переваги та недоліки мурашиного алгоритму

Оскільки VRP є NP-жорстким, відповідно до обмежень, VRPTW також є NP-жорстким. Фактично, навіть знайти можливе рішення VRPTW для фіксованої кількості транспортних засобів само по собі є NP-складною проблемою.

Мурашиний алгоритм не гарантує оптимального рішення, але він має поліноміальну складність і дозволяє отримати наближені рішення складних задач, таких як VRP, за короткий час. Точні методи мають факторну складність, тому вони, як правило, неефективні, коли кількість балів перевищує 10.

Переваги ACO

Для VRP мурашині алгоритми є відносно ефективними та працює краще, ніж інші глобальні оптимізації VRP (нейронні мережі, генетичні алгоритми).

Порівняння між мурашиним алгоритмом і генетичним алгоритмом:

- Алгоритм фокусується на пам'яті всієї групи (на відміну від пам'яті попереднього покоління);

- Випадковий вибір шляху та групова пам'ять мінімізують тенденцію до неоптимальних вихідних рішень;

- Адаптуватися до динамічних змін;

- Використовується для багатьох різних завдань.

недолік

До недоліків можна віднести:

а) Теоретичний аналіз ускладнюється наступними фактами:

- Ітераційні зміни розподілу ймовірностей;

- Дослідження більше експериментальні, ніж теоретичні;

б) Збіжність гарантована, але час збіжності невизначений;

в) Часто потрібні додаткові методи, такі як локальний пошук;

г) Алгоритм залежить від параметрів, обраних виключно на основі експериментів.

2.4 Модифікації алгоритму мурашиних колоній

У цьому розділі пропонується модифікація ACO, яка додає правила для побудови прийнятних рішень [1].

Основною складовою алгоритму ACO є знаходження ймовірності переходу з i-ї точки в j-ту точку в момент часу t [62]. Обмеження вікна часу безпосередньо впливає на ймовірність переходу до тієї чи іншої вершини, оскільки чим менше значення в кінці вікна часу для вершини, тим більша

ймовірність вибору цієї вершини. Враховуючи існування часового вікна, ймовірність переходу буде визначатися за такою формулою:

$$p_{ij}(t) = \frac{\tau_{ij}(t)^{\alpha} \cdot \left(\frac{1}{d_{ij}}\right)^{\beta} \cdot \left(\frac{1}{r_j}\right)^{\gamma}}{\sum_{j \in J_i} \tau_{ij}(t)^{\alpha} \cdot \left(\frac{1}{d_{ij}}\right)^{\beta} \cdot \left(\frac{1}{r_j}\right)^{\gamma}}, \qquad (2.13)$$

Де:

Ji — множина вершин, які допускають перехід, починаючи з вершини i;

tau ij (t) – рівень феромону на межі i - j на кроці t;

Dij – відстань між вершинами i та j;

rj = lj – t – час, що залишився від j-ї точки до кінця вікна часу [ej, lj];

альфа, бета, гамма = {0, 1} – постійні параметри.

Для прискорення розв’язання та покращення отриманого розв’язку ACO пропонуються такі правила:

- Якщо поточне значення цільової функції більше кращого значення, яке було знайдено, відрізати безперспективні маршрути (зменшить кількість неефективних ітерацій);

- Якщо в поточній ітерації знайдено m послідовних ідентичних рішень (m послідовних поліпшень знайдених рішень не спостерігається), перехід до наступної ітерації якомога раніше;

- Оновити феромон після того, як кожна мураха побудувала мураху, чиє значення цільової функції нижче за найкращий знайдений шлях;

- Скидання феромонів на початку кожної ітерації, щоб запобігти повторюванню алгоритмом найкращого рішення, знайденого в попередній ітерації.

2.5 Розробка гібридного алгоритму

Останнім часом було запропоновано багато алгоритмів ACO для вирішення різних задач комбінаторної оптимізації. Однак дослідження показують, що достовірність результатів підвищується, якщо поєднати кілька метаевристик. Розвиток гібридних алгоритмів є перспективним напрямком, оскільки гібридні алгоритми поєднують у собі переваги використовуваних алгоритмів.

З метою підвищення ефективності модифікованого алгоритму мурашиної колонії рекомендовано розробити гібридний алгоритм, який поєднує в собі переваги двох метаевристик, що дозволить покращити результати роботи алгоритму в цілому [1], а саме мурашиної колонії. алгоритм оптимізації та пошук табу.

Табу-пошук є метаевристичним алгоритмом, який може значно підвищити продуктивність локального пошуку за рахунок використання структур пам'яті. Ідеї пошукових алгоритмів ACO і Tabu досить різні. Пошук Tabu не покладається на інформацію попередніх ітерацій, як ACO, а випадковим чином вибирає напрямок, який може збільшити простір пошуку. У кожній ітерації околиці поточного рішення вивчаються за допомогою методу локального пошуку, і найкраще рішення вибирається як нове поточне рішення. Однак, на відміну від класичних методів локального пошуку, процес не зупиняється на першому локальному оптимумі. Якщо потенційне рішення було знайдено раніше або якщо воно порушує дане правило, воно позначається як «табу» (заборонено), щоб алгоритм не розглядав рішення повторно, оскільки воно безнадійне. Найкраще перехрестя в околицях буде обрано як поточне перехрестя, навіть якщо воно гірше поточного перехрестя. Такий підхід дозволяє методу відійти від локальних оптимумів і дослідити більшу частину простору пошуку [62].

Схема гібридного алгоритму.

Запропонована схема гібридного алгоритму базується на схемі ACO та додає ітерацію пошуку табу [1]:

Крок 1. Повторюйте наступні кроки, поки записане рішення не покращиться за 1 ітерацію:

Крок 1.1 Створіть мураху.

Крок 1.2 Знайдіть рішення.

Крок 1.3 Для кожного з b найкращих рішень, отриманих у поточній ітерації, застосуйте пошук табу:

Крок 1.3.1 Створіть список табу.

Крок 1.3.2, поки записане рішення не буде покращено після □ ітерацій:

Крок 1.3.3 Визначте околиці поточного записаного перехрестя, за винятком перехресть, доданих до списку табу.

Крок 1.3.4. Знайти найкраще рішення по сусідству.

Крок 1.3.5 Оновлення списку табу.

Крок 1.4 Оновлення феромонів.

Запропонована схема гібридного алгоритму додає ітерації пошуку табу на основі схеми ACO, що дозволяє якісно покращити показники обох метаевристик, оскільки розроблений гібридний алгоритм поєднує в собі переваги обох метаевристик, тому можна досягти значного покращення результатів пошуку розв’язків задач [1].

Тому запропоновано математичну модель задачі VRPTW та метод її розв’язання. Обґрунтовано вибір гібридних алгоритмів та їх метаевристик. Наведено загальну схему алгоритму ACO, перераховано його основні недоліки та переваги. Запропоновано вдосконалення ACO та гібридний алгоритм для його вирішення, а також розроблено гібридний алгоритм, що використовує вдосконалений ACO та пошук табу.

3 ОПИС ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1 Засоби розробки

Для реалізації гібридного алгоритму для задачі маршрутизації транспортного засобу з часовим вікном з використанням модифікованого алгоритму оптимізації мурашиної колонії та пошуку табу було розроблено програмне забезпечення з використанням мови програмування Python.

Python — це передова об’єктно-орієнтована, крос-платформна, інтерпретована мова програмування з характеристиками інтерактивності, модульності, динамічності та портативності.

Вибір цієї мови має такі переваги.

Розширена бібліотека - Python інтегрує велику стандартну бібліотеку, яка охоплює переважну більшість поширених завдань програмування, що дозволяє значно скоротити довжину коду та спростити використання мови.

Гнучкість. Python можна легко розширити до інших популярних мов програмування, таких як C, C++ і Java, а також легко вбудувати в інші мови програмування, дозволяючи вам використовувати функції Python за допомогою інших мов програмування.

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

Продуктивність – простота, інтегровані в процес бібліотеки, модульні системи тестування та розширені можливості керування Python дають розробникам переваги в ефективності, які підвищують швидкість і продуктивність програми порівняно з використанням інших мов. Розробники можуть використовувати існуючі бібліотеки, зосереджені на унікальному коді для конкретного проекту.

Простота. Оскільки більшість функцій інкапсульовано в бібліотеках, код Python легко програмувати, читати та розуміти. Відсутність фігурних дужок полегшує читання та налагодження коду.

Можливість інтерпретації – у Python оператори виконуються по одному, що спрощує налагодження відносно скомпільованих мов.

Універсальність та можливість інтеграції – python працює на багатьох платформах – для веб-сайтів, мобільних пристроїв, інженерних інженерних систем, бек-енд розробки. Ви також можете використовувати python для розробки сценаріїв для конкретних сценаріїв, щоб уникнути розробки глобального програмного забезпечення. Python також є популярною мовою для науки про дані та машинного навчання.

Велика інклюзивна спільнота програмістів, які використовують Python, - надання підтримки програмістам на етапі розробки, наявність експертних предметних ресурсів і розробка додаткових бібліотек.

Довіра до масштабних проектів. Останнім часом Python обрали такі компанії, як Google, YouTube, Dropbox, Yahoo, і використовували його в освіті, науці, охороні здоров’я та фінансах.

Бібліотеки, які використовуються в проекті, показані на рисунку 3.1

```
from PyQt5 import QtWidgets, QtCore, QtGui,Qt
import random
import math
import copy
import os
import io
import time
import datetime
```

Рисунок 3.1 - Бібліотеки, що використовуються у проекті

Мета використання бібліотеки:

- PyQt5 – модуль для роботи з додатками GUI;

- random – модуль, що дозволяє працювати з випадковими значеннями;

- Math - модуль з вбудованими математичними операціями;

- copy – модуль, який дозволяє копіювати об’єкти (за замовчуванням у Python при копіюванні створюється посилання на цільовий об’єкт);

- os – модуль для автоматизації завдань операційної системи (створення/видалення папок, отримання їх вмісту, відкриття файлів по шляхах, зміна каталогів);

- Time – модуль, який дозволяє використовувати різні функції, пов'язані з часом;

- datetime – цей модуль містить класи для роботи з датами та часом.

Як IDE для розробки програмного продукту використовується Atom (рис. 3.2) – кросплатформна IDE з відкритим кодом, розроблена GitHub.

Рисунок 3.2 - IDE Atom.

3.2 Архітектура програмного забезпечення

3.2.1 Структура класів програмного забезпечення

Клас POINT відповідає за ручне введення даних завдання: він отримує координати доданих на діаграму точок (клієнтів), дані про клієнтів (попит, часові вікна, час обслуговування) і парк ТЗ: кількість ТЗ і його місткість.

Клас WINDOW відповідає за запуск програми, створення та запуск інтерфейсу та візуалізацію роботи алгоритму: він відповідає за отримання та збереження даних проблеми з файлів або з класу POINT, обчислення матриці

відстані, запуск гібридного алгоритму, моніторинг подій роботи курсору та збереження отриманих даних задачі, візуалізація етапів роботи та результатів алгоритму.

За реалізацію алгоритму відповідає клас WORKER.

3.2.2 Специфікація функцій програмного забезпечення

Таблиця 3.1 містить опис функціональності програмної реалізації гібридного алгоритму.

Таблиця 3.1 – Специфікація функцій програмної реалізації гібридного алгоритму

Назва функції	Вхідні параметри	Опис функції
POINT.__init__	x, y – координати x, y; type_point – тип точки (клієнт/депо).	Зберігає інформацію про клієнта чи депо.
POINT. send_data	-	Відправляє в поток отриманні дані задачі
CALC_WORKER. __init__	count_ants – кількість мурах; max_vehicle_number – макс. кількість ТЗ, distance_matrix – матриця відстаней, data – дані задачі, vehicle_capacity – місткість ТЗ, alpha, beta, gamma – коеф. ймовірностей, count_final_iterations – к-сть ітерацій для виходу з алгоритму, start_pheromone – початковий феромон, rate_minimum_distance, evaporation_pheromone_rate – коефіцієнт випаровування феромону, count_best_ants – кількість кращих мурах для ТАБУ, count_tabu_iterations –кількість ітерацій табу пошуку.	Створює поток для роботи алгоритму.
CALC_WORKER. run	-	Реалізовує гібридний алгоритм

Назва функції	Вхідні параметри	Опис функції
CALC_WORKER. calc_ant	Ant – масив даних про мураху	Рахує загальну відстань та шлях, що пройшли мурахи
CALC_WORKER. tabu_search	result – масив мурах популяції	Реалізовує табу пошук
CALC_WORKER. return_transition	current_point – поточний клієнт, car - ТЗ, avaliable_points – доступні клієнти, pheromone_matrix – матриця феромонів	Шукає наступного клієнта для ТЗ
CALC_WORKER. return_transition_asa	current_point – поточний клієнт, car - ТЗ, avaliable_points – доступні клієнти, pheromone_matrix – матриця феромонів	Шукає наступного клієнта для ТЗ з урахуванням часових вікон
CALC_WORKER. is_avaliable	start_point – початковий клієнт, finish_point – наступний клієнт, car- ТЗ	Розрахунок можливості відвідування клієнта
CALC_WORKER. return_time	start_point – початковий клієнт, finish_point – инаступний клієнт, car- ТЗ	Розрахунок часу пересування
CALC_WORKER. is_finish	result – масив мурах популяції	Перевірка умов виходу з алгоритму
CALC_WORKER. update_pheromone_ matrix	pheromone_matrix - матриця феромонів, ants – масив мурах, best_distance – кращий маршрут	Оновлення феромонів у матриці
CALC_WORKER. return_values	result – масив мурах популяції	Повертає ребра, по яким пройшовся мураха та відстань
CALC_WORKER. return_best	result – масив мурах популяції	Шукає кращу мураху

Назва функції	Вхідні параметри	Опис функції
CALC_WORKER. terminate_worker	-	Зупинка розрахунків
WINDOW. __init__	-	Створює інтерфейс програми
WINDOW.resizeEvent	event - подія	Масштабує вікно
WINDOW. read_file	path – шлях до файлу	Зчитує данні задачі із файлу
WINDOW. add_point	-	Запускає очікування кліку для додавання клієнта
WINDOW.add_depart	-	Запускає очікування кліку для додавання депо
WINDOW. start_calc	-	Створює екземпляри класу CALK_WORKER, запускає алгоритми
WINDOW. calc_distance_matrix	-	Рахує матрицю відстаней
WINDOW. return_distance	i, j - клієнти	Рахує відстань між двома клієнтами
WINDOW. mousePressEventFrame	event - подія	Реагує на клік по полю відображення клієнтів
WINDOW. mouseScrollEventFrame	event - подія	Реагує на прокрутку
WINDOW. accept_data	values – масив даних про отриману точку	Додає до розрахунків отриманого клієнта
WINDOW. draw_point	number – номер точки	Малює точку(клієнта)
WINDOW. draw_depart	-	Малює депо

Назва функції	Вхідні параметри	Опис функції
WINDOW. draw_lines	signal - сигнал	Малює та зберігає маршрути
WINDOW. clear_scene	-	Очищує сцену
WINDOW. stop_work	-	Зупиняє роботу алгоритму
WINDOW. clear_data	-	Очищує всю отриману та розраховану інформацію
WINDOW. init_car_colors	-	Генерує кольори маршрутів для кожної машини

3.3 Вхідні данні

Вхідні дані - це дані, які надходять в систему і необхідні для подальшої роботи системи. Введіть дані, зареєстровані в програмі, за допомогою форм або шляхом імпорту з файлу.

Дані для гаражів і стоянок:

- Кількість транспортних засобів, доступних для доставки;
- Місткість автомобіля.

Інформація споживача:

- ідентифікатор (номер);
- координати;
- Попит на продукцію;
- Початок і кінець вікна часу;
- години роботи.

Дані алгоритму:

- коефіцієнти альфа, бета, гамма.

3.4 Вихідні данні

Вихідними даними є сформований план транспортування продукції, який враховує всі потреби споживачів, часові вікна обслуговування та мінімізує витрати на транспортування.

3.5 Інструкція користувача

Цей розділ містить інструкції щодо запуску та використання розробленої програмної реалізації гібридного алгоритму пошукової оптимізації мурашиної колонії та табу.

Щоб почати використовувати програму, помістіть програмні файли в папку поточного користувача на жорсткому диску, запустіть термінал і введіть команду «python3 app.py» (рис. 3.3).

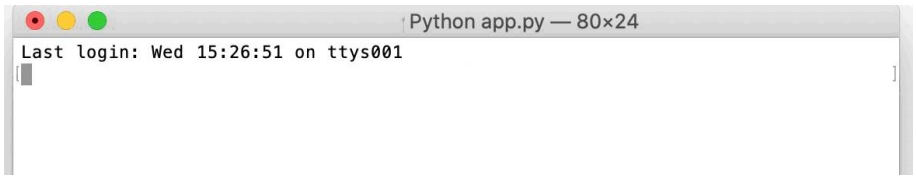

Рисунок 3.3 – Команда “python3 app.py” у терміналі

Після чого відбудеться запуск програми (рисунок 3.4).

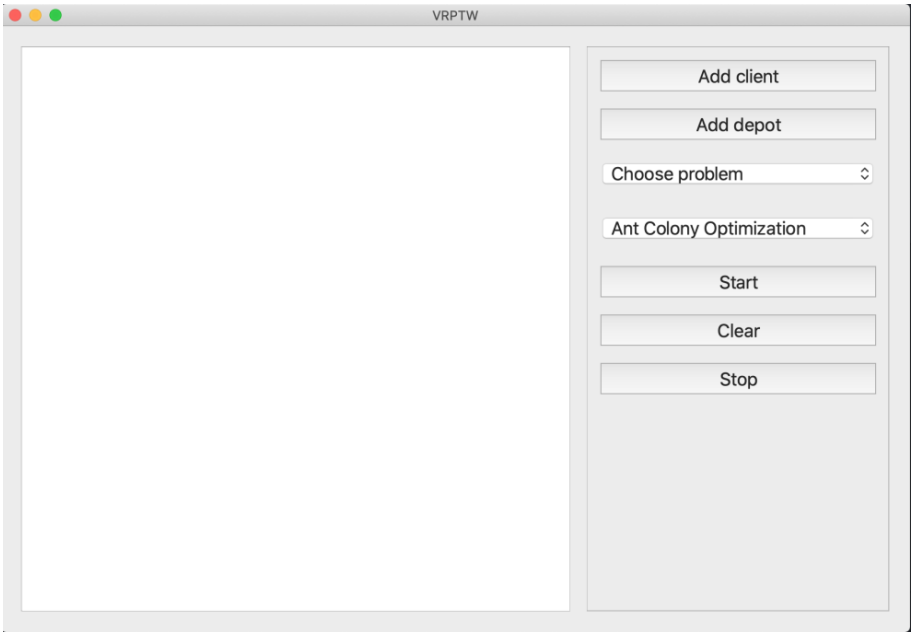

Рисунок 3.4 – Вікно програми після запуску

На сторінці відображаються кнопки для ручного додавання клієнтів і складів або читання завдань з файлу. Щоб ввести дані завдання вручну, необхідно додати точки, що відповідають клієнтам і місцезнаходженням складу, і ввести всю необхідну інформацію.

Додати процес клієнта:

а) Натисніть кнопку «Додати клієнта»;

б) Накресліть точку в полі таблиці (рисунок 3.5);

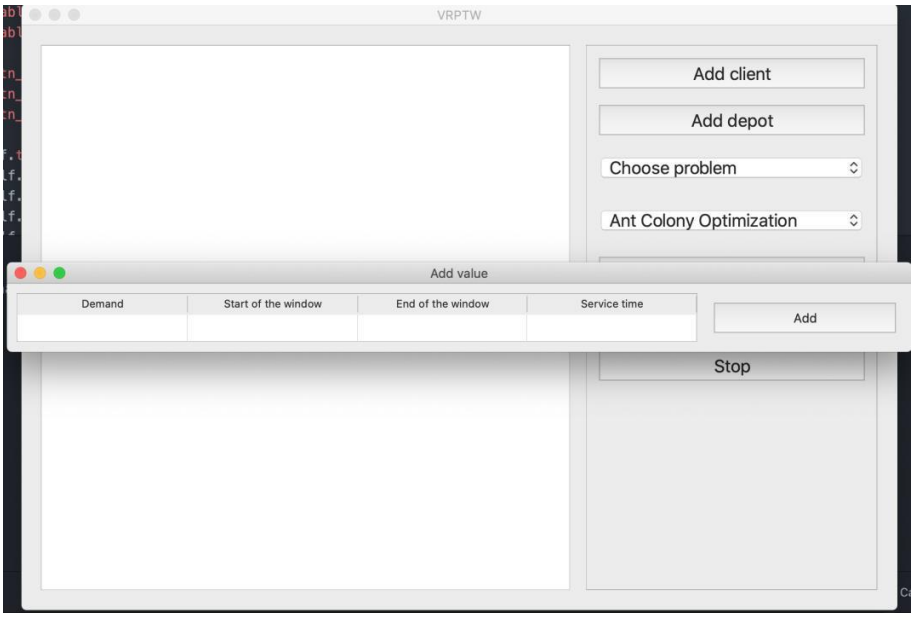

Рисунок 3.5 – Додання клієнта

в) Введіть параметри у відповідні поля (рис. 3.6);

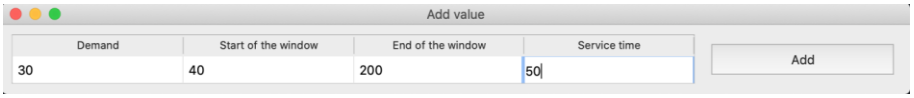

Рисунок 3.6 – Додайте дані клієнта

г) Натисніть кнопку «Додати».

Процес додавання складу:

а) Натисніть кнопку «Додати склад»;

б) Намалюйте крапку на полі форми (рис. 3.7);

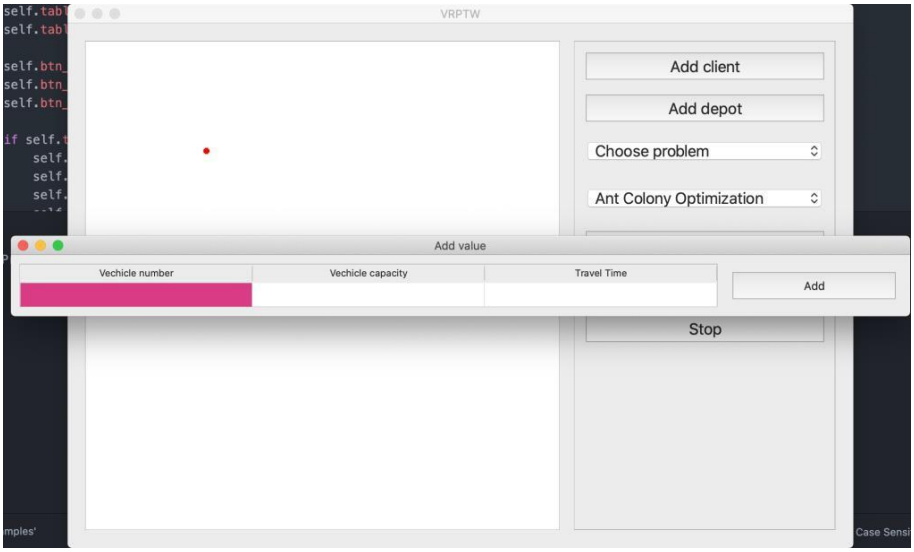

Рисунок 3.7 – додання депо

в) Ввести параметри у відповідні поля (рис. 3.8);

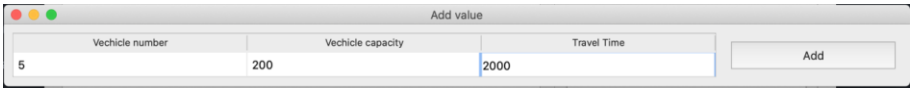

Рисунок 3.8 – Додати дані про склад

г) Натисніть кнопку «Додати».

Процес запуску завдання з файлу:

Виберіть завдання зі списку (рис. 3.9).

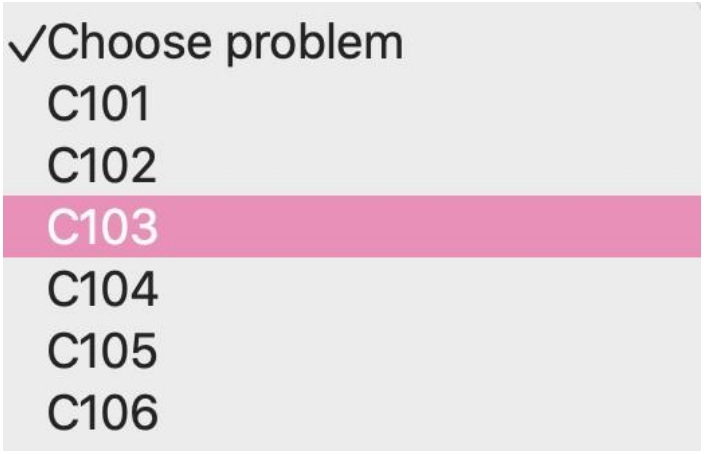

Рисунок 3.9 – список задач Соломона

почати розрахунок:

а) Вибрати алгоритм (рис. 3.10);

Рисунок 3.10 – Список алгоритмів розрахунку

b) Натисніть кнопку Пуск.

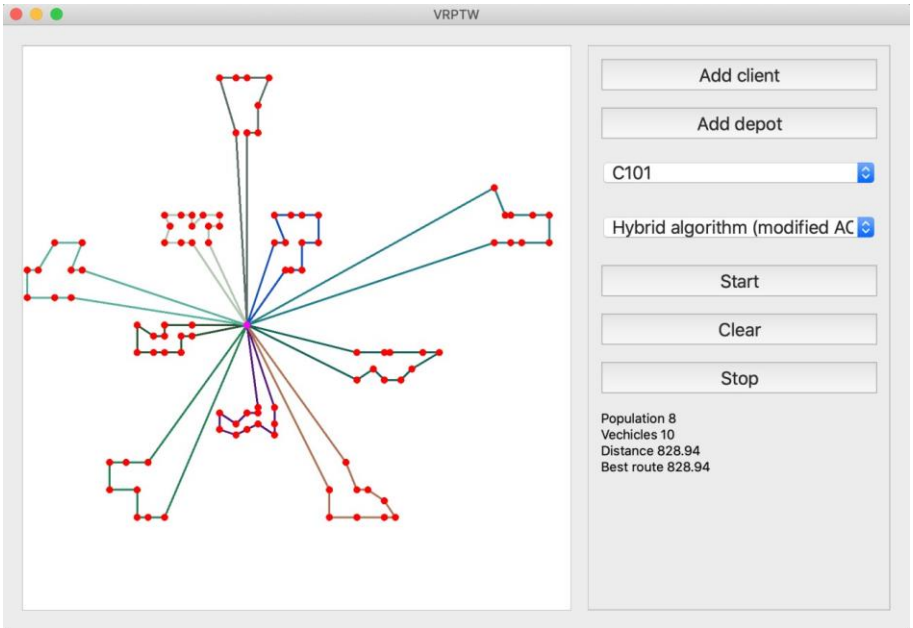

Рисунок 3.11 – Були досягнуті результати.

Маршрут відображається в терміналі у вигляді тексту (рис. 3.12):

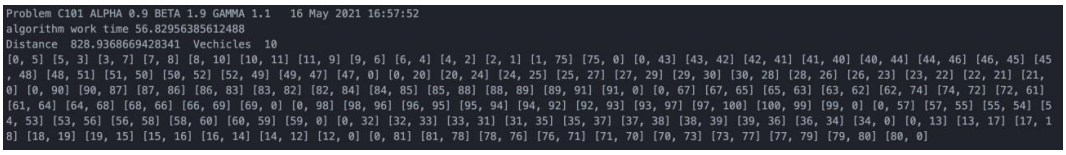

Рисунок 3.12 – Маршрути

У цьому розділі описано програмну реалізацію гібридного алгоритму для оптимізації вдосконалених алгоритмів пошуку мурашиної колонії та табу, обґрунтовано обрану мову програмування, описано бібліотеки, класи та специфікації функцій, що використовуються у розробці. Наведено опис, що містить покрокову інструкцію щодо взаємодії з програмним продуктом.

РЕЗУЛЬТАТИ ДОСЛІДЖЕНЬ

4.1 Мета досліджень

Метою даного дослідження є оцінка ефективності розробленої модифікації алгоритму Ant і розробленого гібридного алгоритму для вирішення задачі VRPTW.

4.2 Тестові данні

Для дослідження результатів розробленого гібридного алгоритму використовувався класичний набір із 56 задач, розроблений М. Соломоном у 1987 році. Тест складається з шести різних типів завдань (C1, C2, R1, R2, RC1, RC2). Кожен набір даних містить від 8 до 12 запитань із розмірністю 25, 50 і 100 вузлів. Проблеми класу С характеризуються кластерними клієнтами, часові вікна яких створюються на основі відомих рішень. Проблема R-типу характеризується рівномірно випадковою генерацією клієнтів у межах регіону. Проблеми типу RC поєднують випадково розміщених і кластерних клієнтів. Місія типу 1 має вузьке часове вікно та низьку потужність автомобіля. Місія типу 2 має широке часове вікно та високу потужність автомобіля. Тому розв’язок задачі 2-го типу вміщує меншу кількість маршрутів, але більшу кількість клієнтів на маршрут порівняно з задачею 1-го типу [6].

Експерименти проводяться шляхом виконання трьох методів для кожної задачі, які припиняються після встановленого часу обчислення. Розв’язки кожного типу задач усереднюються і результати додаються до таблиці. На малюнку 4.1 показано приклад документа із запитанням C101 із використанням чисел, координат, вимог до продукту, часу обслуговування та часових вікон клієнта.

C101

VEHICLE
NUMBER CAPACITY
25 200

CUSTOMER

CUST NO.	XCOORD.	YCOORD.	DEMAND	READY TIME	DUE DATE	SERVICE TIME
0	40	50	0	0	1236	0
1	45	68	10	912	967	90
2	45	70	30	825	870	90
3	42	66	10	65	146	90
4	42	68	10	727	782	90
5	42	65	10	15	67	90
6	40	69	20	621	702	90
7	40	66	20	170	225	90
8	38	68	20	255	324	90

Рисунок 4.1 – Задача 101 з бенчмарку Соломона

Цей метод тестування працює для розробників у всьому світі. Це дозволяє порівняти результати з відомим найкращим рішенням, оцінити швидкість алгоритму та наскільки він відхиляється від найкращого рішення.

4.3 Результати досліджень ефективності запропонованих модифікацій алгоритму мурашиних колоній

Щоб перевірити роботу алгоритму, спочатку необхідно визначити ефективні значення коефіцієнтів, що відповідають за ступінь випаровування феромону, залежність маршруту від феромону, відстань до наступного клієнта і часове вікно. Для кожної задачі експериментально підбираються найбільш вдалі коефіцієнти, і на основі цього дослідження вибирається середнє значення для кожної категорії задач Соломона.

У процесі дослідження було згенеровано 3312 варіантів коефіцієнтів α, β та β для кожної 25-вимірної задачі в діапазоні від 0,7 до 3 з кроком 0,1, 0,2 та 0,3 відповідно. Кожен варіант перевіряється, і найкраще рішення визначається експериментально разом із середнім значенням коефіцієнтів для всіх категорій.

(0,9, 1,9, 1,1). Aso — це рандомізований алгоритм, а середня квадратична похибка результатів становить 12,6.

Результати експериментального дослідження показують, що для задачі Соломона покращений ACO знаходить рішення, які в середньому на 15% кращі, ніж ACO, тоді як гібридний алгоритм у середньому на 23% кращий.

Дослідження швидкості алгоритму показують, що швидкість алгоритму залежить від категорії проблеми (рис. 4.2), розмірності проблеми (рис. 4.3) і постійних параметрів алгоритму, таких як кількість окремих мурашиних колоній (рис. 4.3 - 4.4) і кількість ітерацій.

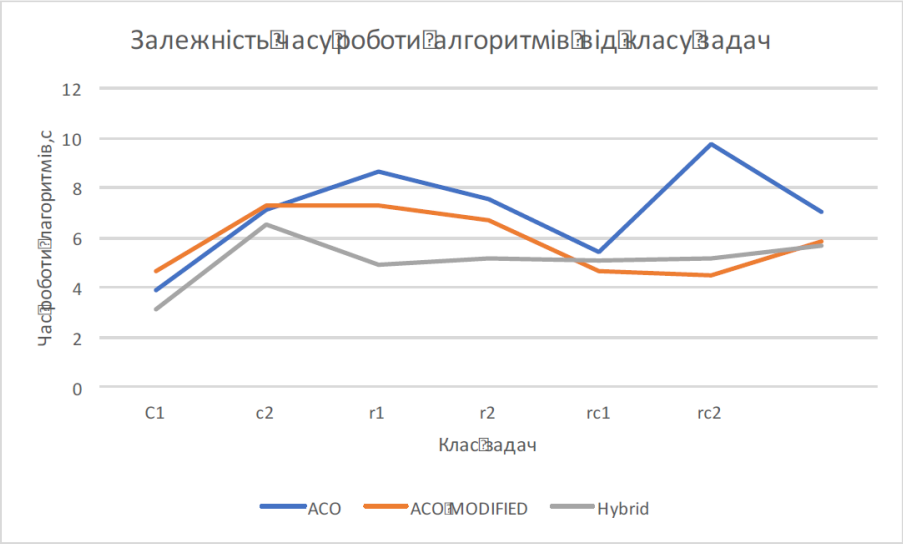

Рисунок 4.2 – Графік залежності часу роботи алгоритмів від класу задач на прикладі розмірності 25.

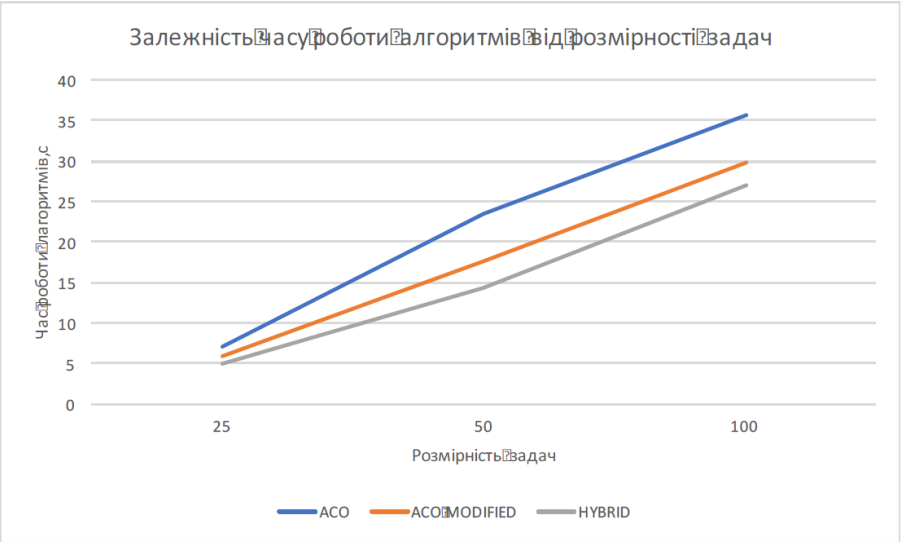

Рисунок 4.3 – Графік середнього часу знаходження розв’язку алгоритмами для задач розмірності 25, 50, 100

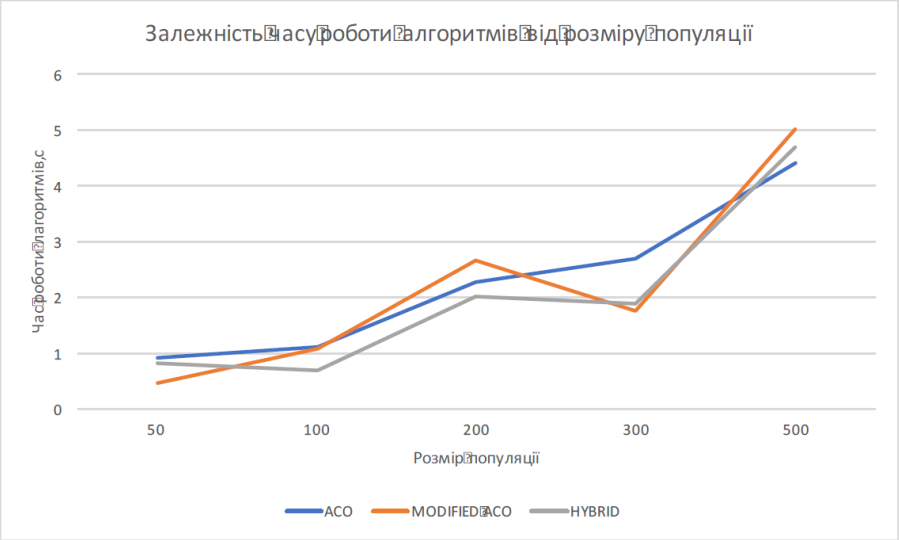

Рисунок 4.4 – На прикладі задачі C102 граф залежності часу алгоритму від кількості особин у популяції (50, 100, 200, 300, 500).

Наступним етапом експериментального дослідження є порівняння результатів гібридного алгоритму з найкращими всесвітньо відомими результатами щодо загальної довжини побудованого маршруту та кількості задіяних транспортних засобів. Результати досліджень наведені в таблиці 4.1.

Таблиця 4.1 – Результати експериментів

	Оптим. Розв'язок		**ACO + евристики**		**Відхилення**
Клас тестових задач	Загальна довжина маршруту	Кількість задіяних ТЗ	Загальна довжина маршруту	Кількість задіяних ТЗ	
R101.25	617,1	8	653,2	8	5,85
R102.25	547,1	7	578,3	8	5,70
R103.25	454,6	5	486,7	5	7,06
R104.25	416,9	4	443,2	4	6,31
R105.25	530,5	6	565,2	6	6,54
R106.25	465,4	3	498,9	5	7,20
R107.25	424,3	4	458,2	4	7,99
R108.25	397,3	4	428,8	4	7,93
R109.25	441,3	5	481,7	5	9,15
R110.25	444,1	4	480,9	5	8,29
R111.25	428,8	5	466,9	4	8,89
R112.25	393	4	423,9	4	7,86
R201.25	463,3	4	531,4	2	14,70
R202.25	410,5	4	456,2	4	11,13
R203.25	391,4	3	449,3	2	14,79

Клас тестових задач	Оптим. розв'язок		АСО + евристики		Відхилення
	Загальна довжина маршруту	Кількість задіяних ТЗ	Загальна довжина маршруту	Кількість задіяних ТЗ	
R204.25	355	2	399,7	2	12,59
R205.25	393	3	453,2	2	15,32
R206.25	374,4	3	421,8	2	12,66
R207.25	361,6	3	416,7	2	15,24
R208.25	328,2	1	353,7	1	7,77
R209.25	370,7	2	417,5	2	12,62
R210.25	404,6	3	472,6	2	16,81
R211.25	350,9	2	413,1	1	17,73
C101.25	191,3	3	191,8	3	0,26
C102.25	190,3	3	196,5	3	3,26
C103.25	190,3	3	199,5	3	4,83
C104.25	186,9	3	193,9	3	3,75
C105.25	191,3	3	191,8	3	0,26
C106.25	191,3	3	191,8	3	0,26
C107.25	191,3	3	192,6	3	0,68
C108.25	191,3	3	191,8	3	0,26
C109.25	191,3	3	193,2	3	0,99
C201.25	214,7	2	215,5	2	0,37
C202.25	214,7	2	223,1	1	3,91
C203.25	214,7	2	221,2	2	3,03
C204.25	213,1	2	223,5	2	4,88
C205.25	214,7	2	215,5	2	0,37
C206.25	214,7	2	231,3	2	7,73
C207.25	214,5	2	232,2	2	8,25
C208.25	214,5	2	216,4	2	0,89
RC101.25	461,1	4	462,6	4	0,33
RC102.25	351,8	3	371,3	4	5,54
RC103.25	332,8	3	353,8	4	6,31
RC104.25	306,6	3	324,9	3	5,97
RC105.25	411,3	4	439,1	5	6,76
RC106.25	345,5	3	361,3	3	4,57
RC107.25	298,3	3	308,2	3	3,32
RC108.25	294,5	3	310,6	3	5,47
RC201.25	360,2	3	391,6	3	8,72
RC202.25	338	3	367,3	2	8,67
RC203.25	326,9	3	363,4	2	11,17
RC204.25	299,7	3	331,9	2	10,74
RC205.25	338	3	361,9	3	7,07

Клас тестових задач	Оптим. розв'язок		АСО + евристики		Відхилення
	Загальна довжина маршруту	Кількість задіяних ТЗ	Загальна довжина маршруту	Кількість задіяних ТЗ	
RC206.25	324	3	370,3	2	14,29
RC207.25	298,3	3	334,3	2	12,07
RC208.25	269,1	2	313,6	1	16,54

Ми провели порівняння ефективності з іншими алгоритмами, зокрема з методом PCP (Priority Constraint Publishing), та гібридним генетичним алгоритмом. Результати порівняння наведені в таблиці 4.2

Таблиця 4.2 – Порівняння з іншими алгоритмами

	Кращий відомий розв'язок		Розроблений гібридний алгоритм		pcp		GA-PCP algorithm	
Клас тестових задач	Загальна довжина маршруту	Кількість задіяних ТЗ	Загальна довжина маршруту	Кількість задіяних ТЗ	Загальна довжина маршруту	Кількість задіяних ТЗ	Загальна довжина маршруту	Кількість задіяних ТЗ
C104.25	186,9	3	193,9	3	189,8	2	186,9	3
R104.25	416,9	4	443,2	4	467,8	3	449,3	5
RC108.25	294,5	3	310,6	3	300,4	5	294,7	5
C204.25	213,1	2	223,5	2	290,9	2	282,05	2
R208.25	328,2	1	353,7	1	332	2	328,5	2
RC208.25	269,1	1	313,6	1	285	2	270,9	2

Запропонований гібридний алгоритм (див. Розділ 2.5.3) експериментально порівнюється з відомими результатами алгоритмів пошуку ASO і Tabu на тестовій задачі Соломона [6], як показано в Таблиці 4.3.

Таблиця 4.3 – Порівняння розв’язків алгоритмів

	Кращий відомий розв’язок	**Алгоритм мурашиних колоній**	**Табу пошук**	**Розроблений гібридний алгоритм**
	Загальна довжина маршруту			
C1.100	826,7	851,47	844,22	840,55
C2.100	589,24	628,47	609,11	604,12
R1.100	1156,36	1396,55	1233,65	1201,24
R2.100	932,35	1007,58	960,25	958,11
RC1.100	1368,23	1167,35	1422,18	1408,14
RC2.100	1075,67	1165,87	1121,39	1112,24

4.4 Аналіз результатів досліджень

Хоча час пошуку та прийнятність рішення безпосередньо залежать від кількості та ітерацій мурах, а також від коефіцієнта, що відповідає за рівень випаровування феромону, залежність маршруту від феромону, відстані до

наступного клієнта, часового вікна , зрозуміло, що На закінчення можна скоротити цей час за допомогою модифікацій алгоритму ASO та гібридного алгоритму, розробленого на його основі.

Таким чином, розробка гібридного алгоритму вдосконаленого ASO і табу-пошуку дозволяє знаходити рішення задачі за менший час порівняно з класичним алгоритмом ASO та іншими запропонованими евристичними алгоритмами VRPTW.

Основний висновок, зроблений у результаті дослідження гібридного алгоритму для побудови найкращих і всесвітньо відомих найкращих результатів наближення для загальної довжини маршруту та кількості залучених ТК, полягає в тому, що алгоритм має різну ефективність для різних ситуацій. Категорія тестового питання. Отже, визначається середній рівень відхилення розв’язку гібридного алгоритму від найкращого відомого розв’язку по загальній довжині маршруту:

- 7,4% покращення для задач R1;
- Для проблем R2 покращення на 13,76%;
- Для питань C1 збільшення на 1,62%;
- Для питань C2 збільшення на 3,68%;
- збільшення на 4,78% для категорії питань RS1;
- Категорія питань RS2 покращена на 11,16%.

Розроблений гібридний алгоритм дає такі результати порівняно з найвідомішими рішеннями в залежності від кількості транспортних засобів, задіяних на маршруті:

- Для класу завдань R1 67% результатів однакові та 8% результатів кращі;
- Для класу задач R2 такий самий результат становить 36,4%, а кращий – 63,6%;
- 100% ідентичний для питання категорії C1;
- Для задач C2 88% результатів однакові і 12% результатів кращі;
- Ті самі 63% для категорії питань RS1;

- Для класу запитань RS2 25% результатів однакові та 75% результатів кращі.

Ефективність запропонованого гібридного алгоритму підтверджується дослідженнями, які показують, що ефективність результатів алгоритму підвищується, якщо кілька метаевристик комбінувати та модифікувати. Розвиток гібридних алгоритмів є перспективним напрямком, оскільки вони поєднують переваги використовуваних евристик. Отже, за результатами експериментальних досліджень можна визначити, що за допомогою гібридних алгоритмів можна отримати розв’язки, близькі до відомих оптимальних розв’язків для всіх категорій задач.

Тому було проаналізовано розроблений вдосконалений алгоритм оптимізації мурашиної колонії та гібридний алгоритм на основі вдосконаленого пошуку ASO та табу. За результатами експериментального дослідження можна визначити, що модифікація ACO позитивно впливає на роботу ACO, а запропонований на цій основі гібридний алгоритм є дуже ефективним і дає порівняльні результати порівняно з відомими кращими рішеннями. s рішення. У результаті було оцінено ефективність розробленого гібридного алгоритму та модифікацій ASO у розв’язанні задачі VRPTW, що дозволило досягти поставлених завдань дослідження.

ВИСНОВКИ

Магістерська робота присвячена задачі маршрутизації транспортного засобу з урахуванням часових вікон. Мета — мінімізувати загальну вартість доставки продукції споживачам з урахуванням часового вікна. У першій частині наведено аналіз проблеми маршрутизації транспортного засобу, описано існуючі типи проблем та їх вирішення. Наведено перелік програмних продуктів, які вирішують задачі, близькі до задачі маршрутизації транспортного засобу. З метою підвищення ефективності відомих алгоритмів розв'язування таких задач сформульовано відповідну постановку задачі та мету дослідження.

Друга частина присвячена розробці модифікованої версії алгоритму оптимізації мурашиної колонії та розробці гібридного алгоритму для задачі VRPTW. Наведено математичну модель задачі VRPTW та метод її розв'язання. Для алгоритму мурашиної колонії наведено узагальнену схему алгоритму, наведено його основні недоліки та переваги. Запропоновано модифікацію алгоритму ACO. Крім того, гібридний алгоритм, заснований на покращеному ACO та пошуку табу, розроблено для завдання маршрутизації потоку трафіку з урахуванням часових вікон.

Третій розділ містить опис програмного забезпечення. Крім того, написана покрокова інструкція користування програмою, яка вирішує задачу маршрутизації транспортного засобу з урахуванням часових вікон.

У розділі 4 представлено мету, результати та аналіз дослідження ефективності запропонованого алгоритму. На основі результатів експериментального дослідження можна визначити, що запропоновані модифіковані та гібридні алгоритми є дуже ефективними та забезпечують порівнювані рішення порівняно з найвідомішими рішеннями. У результаті було оцінено ефективність розробленого гібридного алгоритму у розв'язанні задачі VRPTW, що дозволило досягти поставлених завдань дослідження. Дослідження

підтверджує ефективність запропонованих вдосконалених ACO та гібридних алгоритмів. Модифікації алгоритму ACO підвищують ефективність класичного алгоритму, а гібридні алгоритми дозволяють отримати рішення, близькі до відомих оптимальних рішень для різних типів задач.

Результатами магістерської роботи є:

- Аналіз відомих результатів для вирішення задач маршрутизації транспортних засобів;

- Розроблено вдосконалений алгоритм оптимізації мурашиної колонії та гібридний алгоритм, який поєднує покращений пошук ACO та табу;

- Розроблена програмна реалізація алгоритму;

- Досліджено ефективність розробленого алгоритму.

ПЕРЕЛІК ПОСИЛАНЬ

1. Баранова А. Д., Жданова Е. Г. Задача маршрутизації транспортних засобів з часовими вікнами. VI Всеукраїнська науково-практична конференція молодих вчених та студентів «Інформаційні системи та технології управління»(ІСТУ-2021). – 2021. – Kyiv, Ukraine.

2. CSCMP’s 30 th Annual State of Logistics Report and Presentation. CSCMP’s Annual State of Logistics Report ® and Presentation. – 2019. – №30.

3. Dantzig G. B., Ramser J. H. The Truck Dispatching Problem. INFORMS. – 1959. – p. 12.

4. Clarke G., Wright J. W. Scheduling of wehicles from a central depot to a number of delivery points. Operations Research. –1962. – pp. 568–581.

5. Glover F., Laguna M. Tabu Search. Springer. –1997. – p. 408.

6. Kallehauge B., Larsen J., Madsen O. B. G., Solomon M. M. Chapter 3 Vehicle routing problem with time windows. Column Generation Springer. –2005. – pp. 67–98.

7. Burak Eksioglu. The vehicle routing problem: A taxonomic review [Електронний ресурс] / Burak Eksioglu, Arif Volkan Vural, Arnold Reisman – Режим доступу до ресурсу: https://staff.fmi.uvt.ro/~daniela.zaharie/ma2018/projects/biblio/applications/VehicleRouting/VRP_taxonomicReview.pdf. (дата звернення 20.10.2023)

8. Solomon M. M., Desrosiers J. Time Window Constrained Routing and Scheduling Problems. –1988. – pp. 1–13.

9. Solomon M. M. Algorithms for the VR and SP rith TM Constraints // vol. 35 .–1987. – no. 2. –pp. 254–265.

10. Dorigo M., Stutzle T. Ant Colony Optimization : Overview and Recent Advances. no. May. –2009. – pp. 1–34.

11. Dorigo M., Stützle T. Ant colony Optimization. – 2004. – p. 321.

12. Eksioglu B.,. Volkan A, Reisman A. The vehicle routing problem : A taxonomic review. Computers & Industrial Engineering. – 2009. – vol. 57. – no. 4. – pp. 1472–1483.

13. Кубил В. Н. ИССЛЕДОВАНИЕ И РАЗРАБОТКА МЕТОДОВ РЕШЕНИЯ МОГОКРИТЕРИАЛЬНЫХ ЗАДАЧ МАРШРУТИЗАЦИИ ТРАНСПОРТА НА ОСНОВЕ МУРАВЬИНОГО АЛГОРИТМА : дис. канд. техн. наук : 05.13.01 / Кубил Виктор Николаевич. – Новочеркасск, 2019. – 184 с.

14. Volkan A.,. Eksioglu B, Reisman A. The vehicle routing problem : A taxonomic review. Computers & Industrial Engineering. – 2009. – pp. 1472–1483

15. Liong C.-Y. Vehicle routing problem: Models and solutions. Journal of Quality Measurement and Analysis. – 2008.

16. Braekers K., Ramaekers K., & Nieuwenh, I. V. (2016). The Vehicle Routing Problem: State of the Art Classification and Review. Computers & Industrial Engineering.

17. Eduardo U., Diego P. (2017). New benchmark instances for the Capacitated Vehicle Routing Problem. European Journal of Operational Research.

18. Soysal M., Bloemhof J. M., & Bektas T. (2015). The time-dependent two-echelon capaci- tated vehicle routing problem with environmental considerations. International Journal of Production Economics.

19. Ehsan T., & Vahid K. (2016). Enhanced intelligent water drops and cuckoo search algorithms for solving the capacitated vehicle routing problem. Information Sciences.

20. Maurizio B., & Ferdinando P. (2015). A Variable Neighborhood Search Branching for the Electric Vehicle Routing Problem with Time Windows. Electronic Notes in Discrete Mathematics.

21. Ilker K., & Nursel O. (2015). An advanced hybrid meta-heuristic algorithm for the vehicle routing problem with backhauls and time windows. Computers & Industrial Engineering.

22. Bin Y., Zhi-Hua H. (2015). Routing with time-windows for multiple environmental vehicle types. Computers & Industrial Engineering.

23. Li J.,Li Y.,Pardalos P.M.(2016).Multi-depot vehicle routing problem with time windows under shared depot resources. Journal of Combinatorial Optimization.

24. Montoya-Torres J. R., Franco J. L., Isaza S. N., Jimenez H. F., Herazo-Padilla N. (2015). A literature review on the vehicle routing problem with multiple depots. Computers & Industrial Engineering.

25. De Oliveira F. B., Enayatifar R., Sadaei H. J., Guimara es F. G., & Potvin J.Y. (2016). A cooperative coevolutionary algorithm for the Multi-Depot Vehicle Routing Problem.Expert Systems with Applications.

26. Jairo R. M.-T., Julian L. F., & Santiago N. I. (2015). A literature review on the vehicle routing problem with multiple depots. Computers & Industrial Engineering.

27. Li J., Li Y., Panos M. P. (2016). Multi-depot vehicle routing problem with time windows under shared depot resources. Journal of Combinatorial Optimization.

28. Tania R.P.R., Maria I.G., Ana B.P. (2019). Multi-depot vehicle routing problem: a comparative study of alternative formulations. International Journal of Logistics Research and Applications.

29. Cha ́vez J. J. S., Escobar J. W., Echeverri M. G. (2016). A Multi-objectivePareto and Colony Algorithm for the Multi-depot Vehicle Routing Problem with Backhauls. International Journal of Industrial Engineering Computations.

30. Braekers K., Ramaekers K. , Nieuwenh I. V. (2016). The Vehicle Routing Problem: State the Art Classification and Review. Computers & Industrial Engineering.

31. Baldacci R., Battarra M., & Vigo D. (2008). Routing a heterogeneous fleet of vehicles. In B. L. Golden, S. Raghavan, & E. Wasil (Eds.), The vehicle routing problem: latest advances and new challenges (pp. 3–27). Berlin: Springer.

32. Saso K., Vili P. (2015). A survey of genetic algorithms for solving multi depot vehicle routing problem. Applied Soft Computing.

33. Mustaf A., Seyda T. (2015). An adaptive local search algorithm for vehicle routing problem with simultaneous and mixed pickups and deliveries. Computers & Industrial Engi- neering.

34. Baozhen Y., Bin Y. (2016). An improved particle swarm optimization for carton heterogeneous vehicle routing problem with a collection depot. Annals of Operations Research.

35. Meryem B., Abdelmadjid B. (2016). Quantum Inspired Algorithm for a VRP with Heterogeneous Fleet Mixed Backhauls and Time Windows. International Journal of Appli- ed Metaheuristic Computing.

36. Lijun W., Zhenzhen Z. (2015). A variable neighborhood search for the capacitated vehicle routing problem with two-dimensional loading constraints. European Journal of Operational Research.

37. Emmanouil E., Christos D. (2016). The Vehicle Routing Problem with Simultaneous Pick- ups and Deliveries and Two-Dimensional Loading Constraints. European Journal of Operati- onal Research

38. Cagri K., Gilbert L. (2018). Vehicle routing with backhauls: Review and research perspecti- ves. Computers & Operations Research.

39. Sebastian R., Andreas B. (2018). Heuristics for vehicle routing problems with backhauls, time windows, and 3D loading constraints. European Journal of Operational Research, 266, 3, 877–894.

40. El-Sherbeny N. A. Vehicle routing with time windows: An overview of exact, heuristic and metaheuristic methods. Journal of King Saud University - Science. – 2010, Jul. – vol. 22. – no. 3 . – pp. 123–131.

ДОДАТКИ

Додаток А

(обов'язковий)

ВНТУ

ЗАТВЕРДЖЕНО
Зав. кафедри АІІТ ВНТУ,
д.т.н., професор
_____Олег БІСІКАЛО

" ___ " _____ 2023 р.

ТЕХНІЧНЕ ЗАВДАННЯ

на виконання магістерської кваліфікаційної роботи

«АВТОМАТИЗОВАНА СИСТЕМА ОПТИМІЗАЦІЇ РОЗКЛАДУ РУХУ МАРШРУТНОГО ТРАНСПОРТУ З УРАХУВАННЯМ ЧАСУ ПРОСТОЮ»

08-33.МКР.01.02.000 ТЗ

Студент групи ЗАКІТ-22м

_____Роман Баранець___
Підпис *Ім'я ПРІЗВИЩЕ*

Керівник к.т.н., доцент, доцент кафедри КСУ

_____Олена НИКИТЕНКО
Підпис *Ім'я ПРІЗВИЩЕ*

Вінниця 2023

1. Назва та галузь застосування

1.1. Назва – Автоматизована система оптимізації розкладу руху маршрутного транспорту з урахуванням часу простою

1.2. Галузь застосування – автомобільні перевезення.

2. Підстава для проведення розробки.

Тема магістерської кваліфікаційної роботи затверджена наказом по ВНТУ від "18" вересня 2023 року №247

3. Мета та призначення розробки.

Метою магістерської кваліфікаційної роботи є розробка та вдосконалення алгоритму мурашиних колоній для оптимізації маршрутів транспортних засобів з урахуванням часових вікон.

4. Джерела розробки.

Магістерська кваліфікаційна робота виконується вперше. В ході проведення розробки повинні використовуватись такі документи:

1. А. Д. Баранова, Е. Г. Жданова. Задача маршрутизації транспортних засобів з часовими вікнами // VI Всеукраїнська науково-практична конференція молодих вчених та студентів «Інформаційні системи та технології управління»(ІСТУ-2021). – 2021. – Kyiv, Ukraine.
2. CSCMP's 30 th Annual State of Logistics Report and Presentation. // CSCMP's Annual State of Logistics Report ® and Presentation. – 2019. – №30.

5. Вимоги до розробки.

5.1. Перелік головних функцій:

- аналіз вхідних даних за допомогою різних методів;
- оптимізація розкладу руху маршрутних транспортних засобів;

5.2. Основні технічні вимоги до розробки:

- WINDOWS 10;
- Серидовище розробки(IDE) з підтримкою Python
- 4 гб ОЗУ
- 500 мб вільної пам`яті на диску

6. Стадії та етапи розробки.

6.1 Пояснювальна записка:

1. Аналіз методів, принципів, підходів і засобів реалізації задачі розробка та вдосконалення алгоритмів для оптимізації маршрутів транспортних засобів з урахуванням часових вікон відповідно до теми кваліфікаційної роботи. Постановка задач дослідження «20»_09_2023 р.
2. Удосконалення алгоритму мурашиних колоній «01»_10__ 2023 р.
3. Розробка програмного забезпечення та аналіз результатів «11»_10__ 2023 р.

7. Порядок контролю і приймання.

7.1. Хід виконання роботи контролюється керівником роботи. Рубіжний контроль провести до «19»__11_ 2023 р.

7.2. Атестація МКР здійснюється на попередньому захисті. Попередній захист магістерської кваліфікаційної роботи провести до «21»__11_ 2023 р.

7.3. Підсумкове рішення щодо оцінки якості виконання роботи приймається на засіданні ЕК. Захист магістерської кваліфікаційної роботи провести до «19»__12_ 2023 р.

Додаток Б

(обов`язковий)

ІЛЮСТРАТИВНА ЧАСТИНА

Автоматизована система оптимізації розкладу руху маршрутного транспорту з урахуванням часу простою

Рисунок Б.1 - Класифікація характеристик VRP

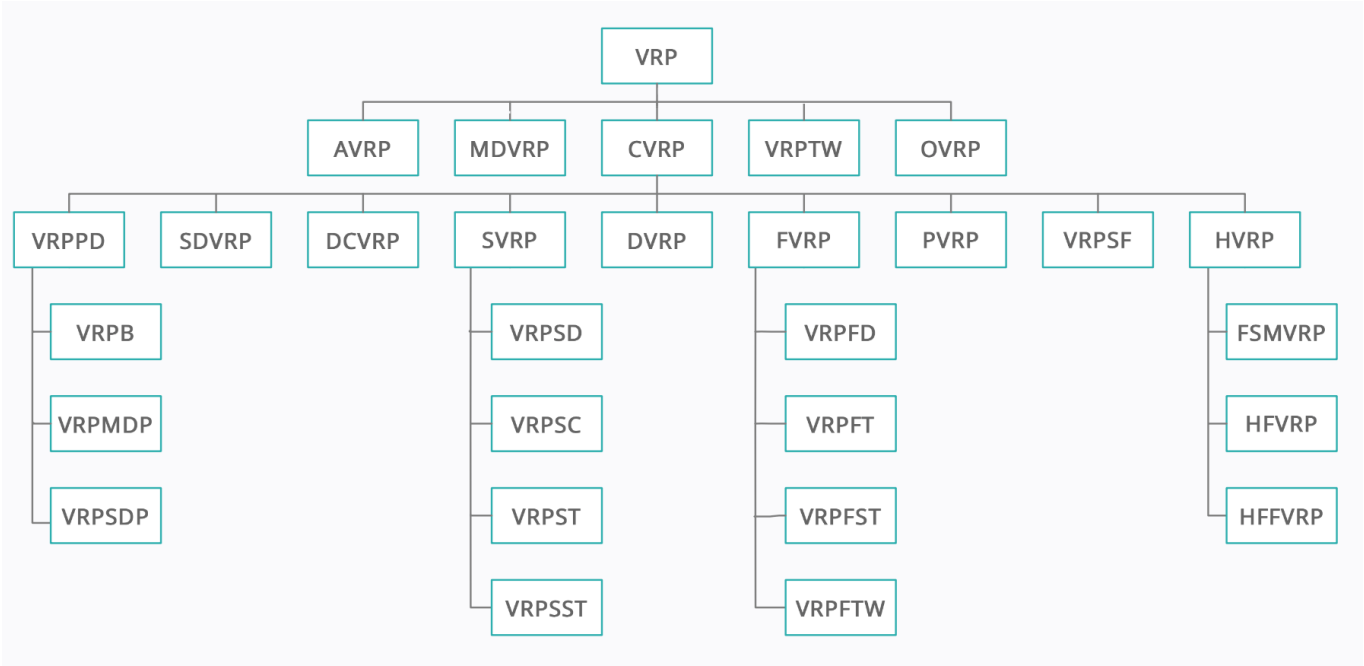

Рисунок Б.2 - Ієрархічна схема узагальнень і розширень VRP

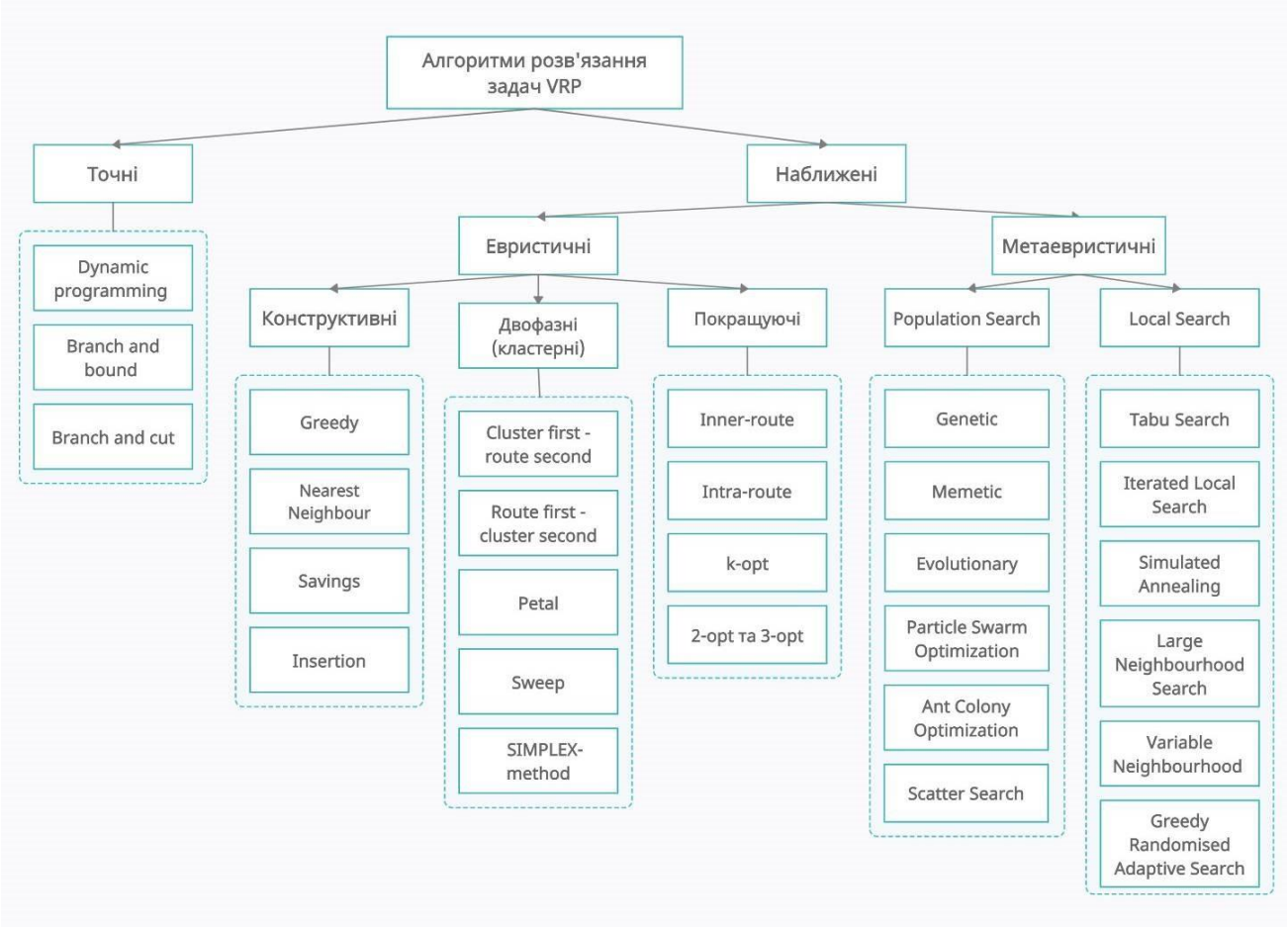

Рисунок Б.3 - Класифікація найпоширеніших алгоритмів розв’язання VRP

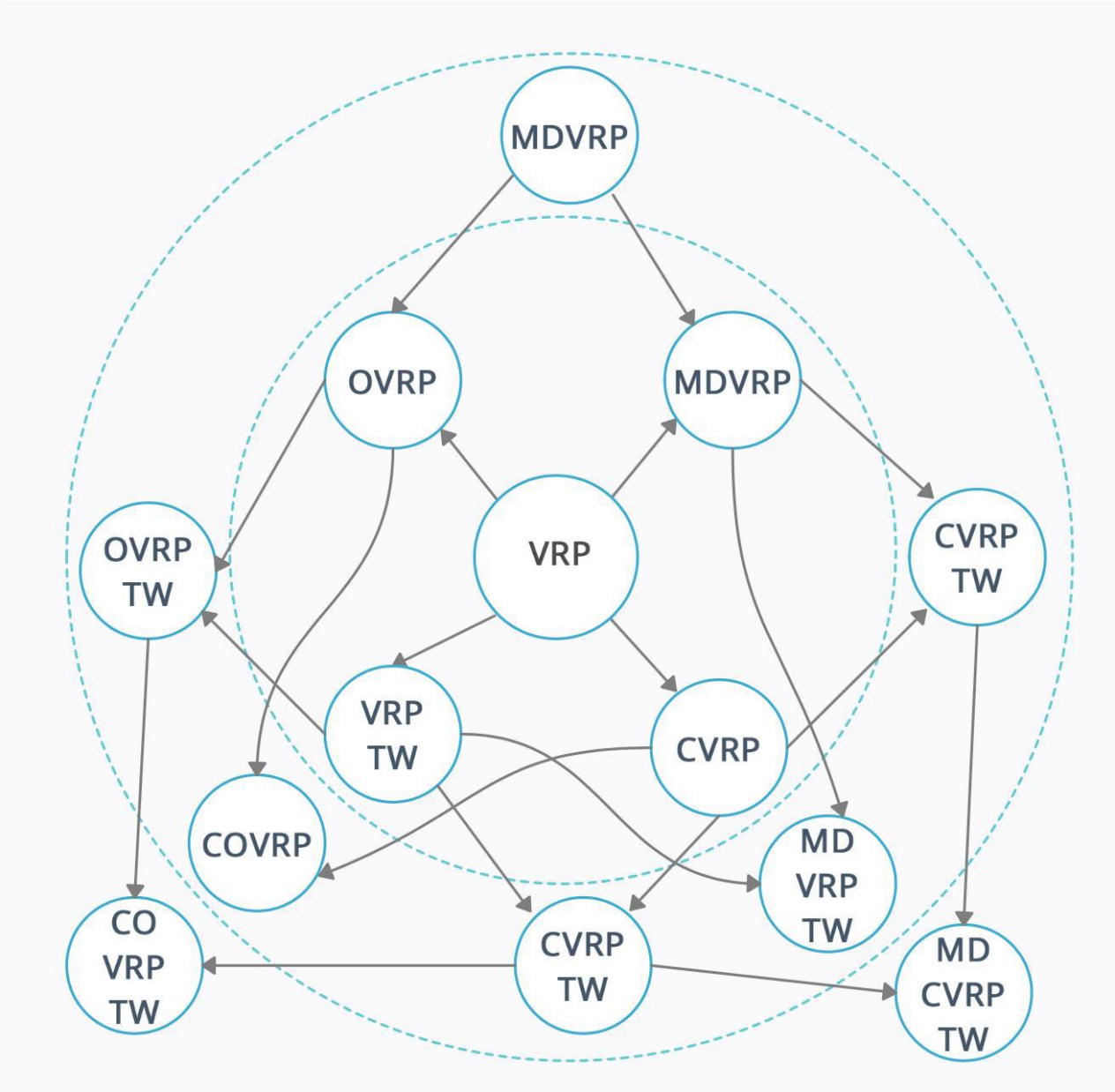

Рисунок Б.4 - Схема зв'язку узагальнюючих різновидів VRP

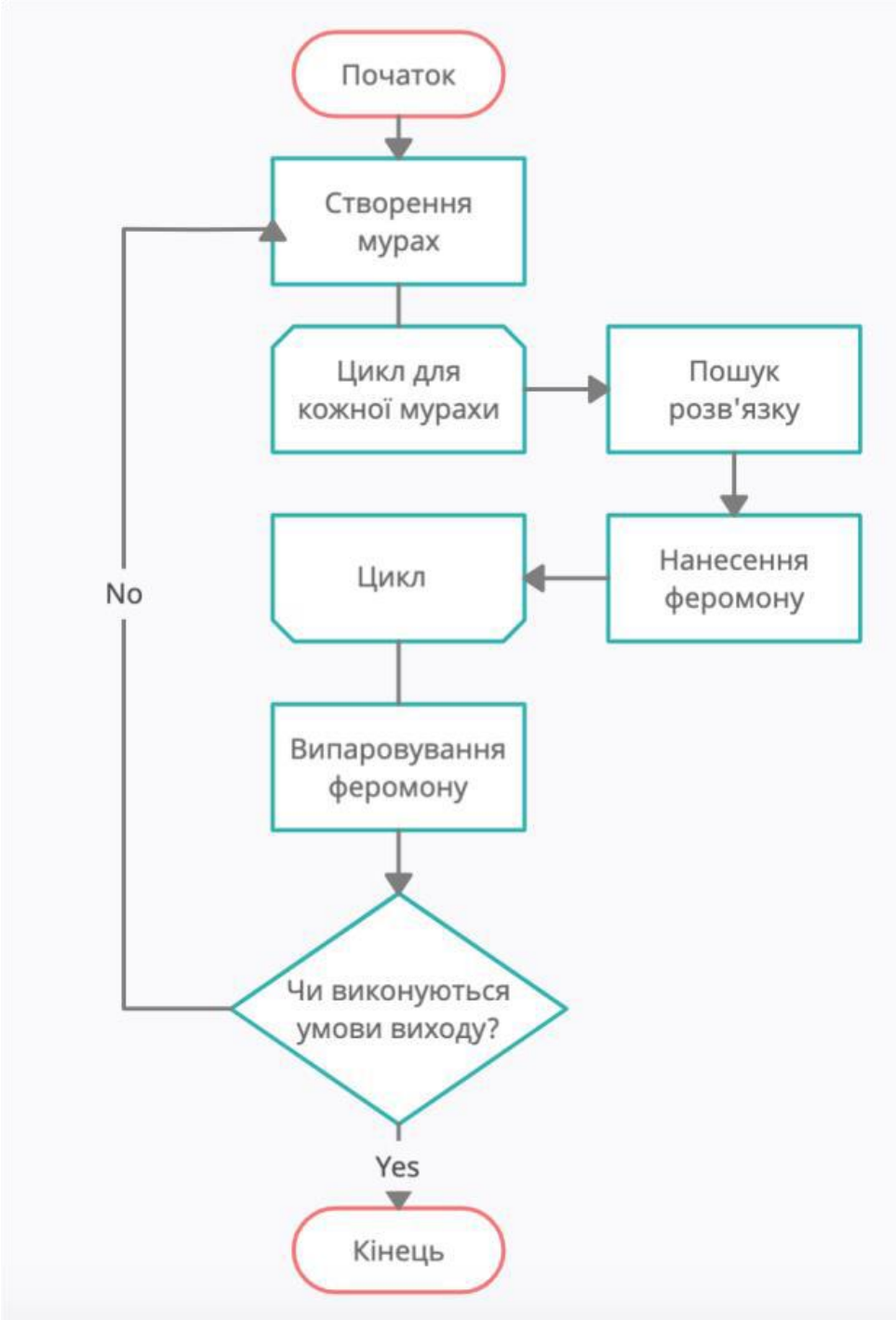

Рисунок Б.5 - Структурна схема алгоритму мурашиних колоній в загальному вигляді

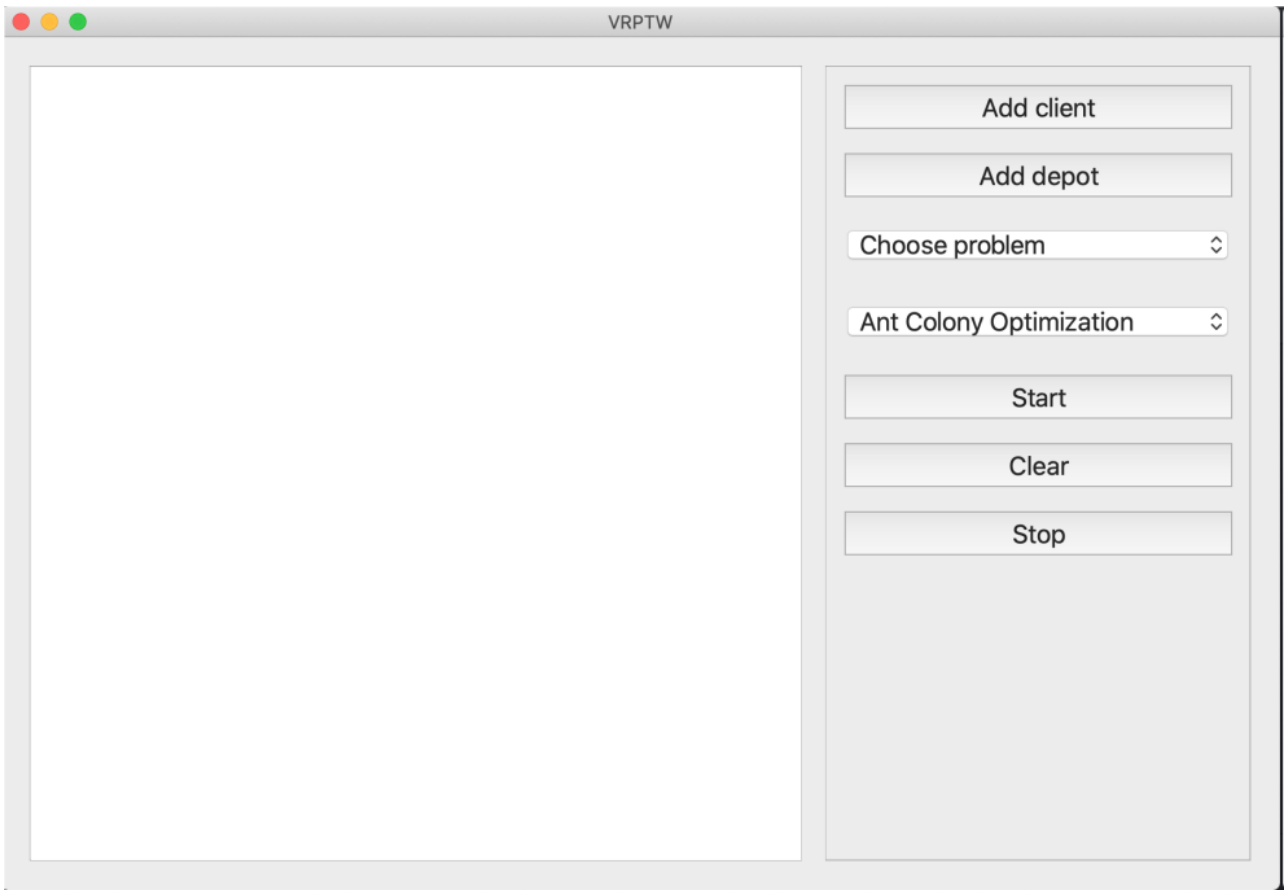

Рисунок Б.6 - Вікно програми після запуску

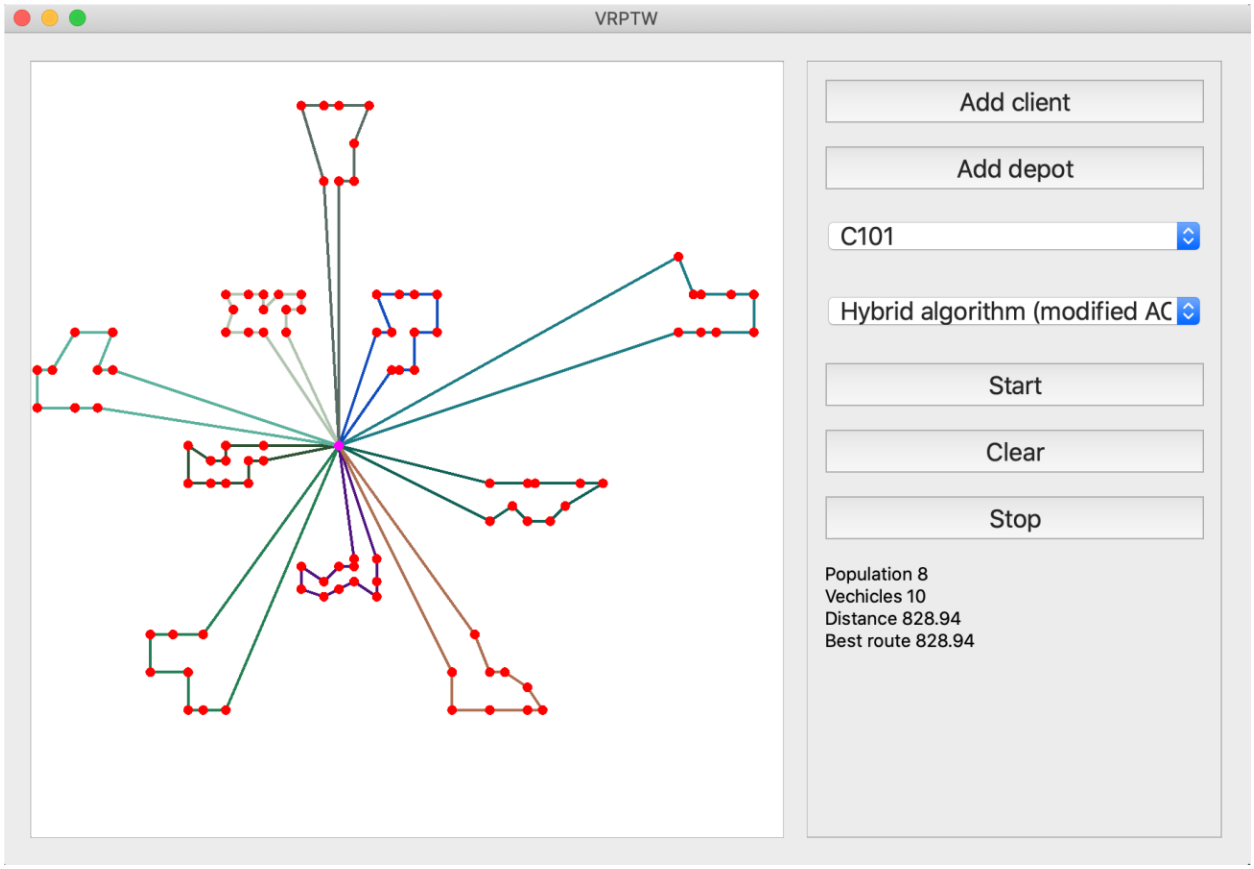

Рисунок Б.7 - Отримані результати

Додаток В

Лістинг програми

Створення даних

```
def create_data_model():
    """Stores the data for the problem."""
    data = {}
    data["distance_matrix"] = [
        # fmt: off
      [0, 548, 776, 696, 582, 274, 502, 194, 308, 194, 536, 502, 388, 354, 468, 776, 662],
      [548, 0, 684, 308, 194, 502, 730, 354, 696, 742, 1084, 594, 480, 674, 1016, 868, 1210],
      [776, 684, 0, 992, 878, 502, 274, 810, 468, 742, 400, 1278, 1164, 1130, 788, 1552, 754],
      [696, 308, 992, 0, 114, 650, 878, 502, 844, 890, 1232, 514, 628, 822, 1164, 560, 1358],
      [582, 194, 878, 114, 0, 536, 764, 388, 730, 776, 1118, 400, 514, 708, 1050, 674, 1244],
      [274, 502, 502, 650, 536, 0, 228, 308, 194, 240, 582, 776, 662, 628, 514, 1050, 708],
      [502, 730, 274, 878, 764, 228, 0, 536, 194, 468, 354, 1004, 890, 856, 514, 1278, 480],
      [194, 354, 810, 502, 388, 308, 536, 0, 342, 388, 730, 468, 354, 320, 662, 742, 856],
      [308, 696, 468, 844, 730, 194, 194, 342, 0, 274, 388, 810, 696, 662, 320, 1084, 514],
      [194, 742, 742, 890, 776, 240, 468, 388, 274, 0, 342, 536, 422, 388, 274, 810, 468],
      [536, 1084, 400, 1232, 1118, 582, 354, 730, 388, 342, 0, 878, 764, 730, 388, 1152, 354],
      [502, 594, 1278, 514, 400, 776, 1004, 468, 810, 536, 878, 0, 114, 308, 650, 274, 844],
      [388, 480, 1164, 628, 514, 662, 890, 354, 696, 422, 764, 114, 0, 194, 536, 388, 730],
      [354, 674, 1130, 822, 708, 628, 856, 320, 662, 388, 730, 308, 194, 0, 342, 422, 536],
      [468, 1016, 788, 1164, 1050, 514, 514, 662, 320, 274, 388, 650, 536, 342, 0, 764, 194],
      [776, 868, 1552, 560, 674, 1050, 1278, 742, 1084, 810, 1152, 274, 388, 422, 764, 0, 798],
      [662, 1210, 754, 1358, 1244, 708, 480, 856, 514, 468, 354, 844, 730, 536, 194, 798, 0],
        # fmt: on
    ]
    data["num_vehicles"] = 4
    data["depot"] = 0
    return data
```

Визначаємо координати

```
[(456, 320), # location 0 - the depot
(228, 0),    # location 1
(912, 0),    # location 2
(0, 80),     # location 3
(114, 80),   # location 4
(570, 160),  # location 5
(798, 160),  # location 6
```

```
(342, 240),  # location 7
(684, 240),  # location 8
(570, 400),  # location 9
(912, 400),  # location 10
(114, 480),  # location 11
(228, 480),  # location 12
(342, 560),  # location 13
(684, 560),  # location 14
(0, 640),    # location 15
(798, 640)]  # location 16

def distance_callback(from_index, to_index):
    """Returns the distance between the two nodes."""
    # Convert from routing variable Index to distance matrix NodeIndex.
    from_node = manager.IndexToNode(from_index)
    to_node = manager.IndexToNode(to_index)
    return data["distance_matrix"][from_node][to_node]

transit_callback_index = routing.RegisterTransitCallback(distance_callback)
routing.SetArcCostEvaluatorOfAllVehicles(transit_callback_index)
dimension_name = "Distance"
routing.AddDimension(
    transit_callback_index,
    0,  # no slack
    3000,  # vehicle maximum travel distance
    True,  # start cumul to zero
    dimension_name,
)
distance_dimension = routing.GetDimensionOrDie(dimension_name)
distance_dimension.SetGlobalSpanCostCoefficient(100)
def create_data_model():
    """Stores the data for the problem."""
    data = {}
    data["distance_matrix"] = [
        # fmt: off
      [0, 548, 776, 696, 582, 274, 502, 194, 308, 194, 536, 502, 388, 354, 468, 776, 662],
      [548, 0, 684, 308, 194, 502, 730, 354, 696, 742, 1084, 594, 480, 674, 1016, 868, 1210],
      [776, 684, 0, 992, 878, 502, 274, 810, 468, 742, 400, 1278, 1164, 1130, 788, 1552, 754],
      [696, 308, 992, 0, 114, 650, 878, 502, 844, 890, 1232, 514, 628, 822, 1164, 560, 1358],
      [582, 194, 878, 114, 0, 536, 764, 388, 730, 776, 1118, 400, 514, 708, 1050, 674, 1244],
      [274, 502, 502, 650, 536, 0, 228, 308, 194, 240, 582, 776, 662, 628, 514, 1050, 708],
      [502, 730, 274, 878, 764, 228, 0, 536, 194, 468, 354, 1004, 890, 856, 514, 1278, 480],
```

```
    [194, 354, 810, 502, 388, 308, 536, 0, 342, 388, 730, 468, 354, 320, 662, 742, 856],
    [308, 696, 468, 844, 730, 194, 194, 342, 0, 274, 388, 810, 696, 662, 320, 1084, 514],
    [194, 742, 742, 890, 776, 240, 468, 388, 274, 0, 342, 536, 422, 388, 274, 810, 468],
    [536, 1084, 400, 1232, 1118, 582, 354, 730, 388, 342, 0, 878, 764, 730, 388, 1152, 354],
    [502, 594, 1278, 514, 400, 776, 1004, 468, 810, 536, 878, 0, 114, 308, 650, 274, 844],
    [388, 480, 1164, 628, 514, 662, 890, 354, 696, 422, 764, 114, 0, 194, 536, 388, 730],
    [354, 674, 1130, 822, 708, 628, 856, 320, 662, 388, 730, 308, 194, 0, 342, 422, 536],
    [468, 1016, 788, 1164, 1050, 514, 514, 662, 320, 274, 388, 650, 536, 342, 0, 764, 194],
    [776, 868, 1552, 560, 674, 1050, 1278, 742, 1084, 810, 1152, 274, 388, 422, 764, 0, 798],
    [662, 1210, 754, 1358, 1244, 708, 480, 856, 514, 468, 354, 844, 730, 536, 194, 798, 0],
    # fmt: on
  ]
  data["num_vehicles"] = 4
  data["depot"] = 0
  return data


[(456, 320), # location 0 - the depot
(228, 0),    # location 1
(912, 0),    # location 2
(0, 80),     # location 3
(114, 80),   # location 4
(570, 160),  # location 5
(798, 160),  # location 6
(342, 240),  # location 7
(684, 240),  # location 8
(570, 400),  # location 9
(912, 400),  # location 10
(114, 480),  # location 11
(228, 480),  # location 12
(342, 560),  # location 13
(684, 560),  # location 14
(0, 640),    # location 15
(798, 640)]  # location 16

def distance_callback(from_index, to_index):
  """Returns the distance between the two nodes."""
  # Convert from routing variable Index to distance matrix NodeIndex.
  from_node = manager.IndexToNode(from_index)
  to_node = manager.IndexToNode(to_index)
  return data["distance_matrix"][from_node][to_node]

transit_callback_index = routing.RegisterTransitCallback(distance_callback)
```

```
routing.SetArcCostEvaluatorOfAllVehicles(transit_callback_index)
```

Наступний код створює вимірювання відстані за допомогою методу AddDimension рішення. Аргументи: transit_callback_index —це індекс для Distance_Callback .

```
dimension_name = "Distance"
routing.AddDimension(
    transit_callback_index,
    0,  # no slack
    3000,  # vehicle maximum travel distance
    True,  # start cumul to zero
    dimension_name,
)
distance_dimension = routing.GetDimensionOrDie(dimension_name)
distance_dimension.SetGlobalSpanCostCoefficient(100)
```

Метод SetGlobalSpanCostCoefficient цей фрагмент тексту описує налаштування великого коефіцієнта (100) для глобального інтервалу маршрутів, який у цьому прикладі є максимальною відстанню між маршрутами. Це робить глобальний відрізок визначальним фактором у цільовій функції, тому програма мінімізує довжину самого довгого маршруту.

```
def print_solution(data, manager, routing, solution):
    """Prints solution on console."""
    print(f"Objective: {solution.ObjectiveValue()}")
    max_route_distance = 0
    for vehicle_id in range(data["num_vehicles"]):
        index = routing.Start(vehicle_id)
        plan_output = f"Route for vehicle {vehicle_id}:\n"
        route_distance = 0
        while not routing.IsEnd(index):
            plan_output += f" {manager.IndexToNode(index)} -> "
            previous_index = index
            index = solution.Value(routing.NextVar(index))
            route_distance += routing.GetArcCostForVehicle(
                previous_index, index, vehicle_id
            )
```

```
        plan_output += f"{manager.IndexToNode(index)}\n"
        plan_output += f"Distance of the route: {route_distance}m\n"
        print(plan_output)
        max_route_distance = max(route_distance, max_route_distance)
    print(f"Maximum of the route distances: {max_route_distance}m")

"""Simple Vehicles Routing Problem (VRP).

  This is a sample using the routing library python wrapper to solve a VRP
  problem.
  A description of the problem can be found here:
  http://en.wikipedia.org/wiki/Vehicle_routing_problem.

  Distances are in meters.
"""

from ortools.constraint_solver import routing_enums_pb2
from ortools.constraint_solver import pywrapcp


def create_data_model():
    """Stores the data for the problem."""
    data = {}
    data["distance_matrix"] = [
        # fmt: off
      [0, 548, 776, 696, 582, 274, 502, 194, 308, 194, 536, 502, 388, 354, 468, 776, 662],
      [548, 0, 684, 308, 194, 502, 730, 354, 696, 742, 1084, 594, 480, 674, 1016, 868, 1210],
      [776, 684, 0, 992, 878, 502, 274, 810, 468, 742, 400, 1278, 1164, 1130, 788, 1552, 754],
      [696, 308, 992, 0, 114, 650, 878, 502, 844, 890, 1232, 514, 628, 822, 1164, 560, 1358],
      [582, 194, 878, 114, 0, 536, 764, 388, 730, 776, 1118, 400, 514, 708, 1050, 674, 1244],
      [274, 502, 502, 650, 536, 0, 228, 308, 194, 240, 582, 776, 662, 628, 514, 1050, 708],
      [502, 730, 274, 878, 764, 228, 0, 536, 194, 468, 354, 1004, 890, 856, 514, 1278, 480],
      [194, 354, 810, 502, 388, 308, 536, 0, 342, 388, 730, 468, 354, 320, 662, 742, 856],
      [308, 696, 468, 844, 730, 194, 194, 342, 0, 274, 388, 810, 696, 662, 320, 1084, 514],
      [194, 742, 742, 890, 776, 240, 468, 388, 274, 0, 342, 536, 422, 388, 274, 810, 468],
      [536, 1084, 400, 1232, 1118, 582, 354, 730, 388, 342, 0, 878, 764, 730, 388, 1152, 354],
      [502, 594, 1278, 514, 400, 776, 1004, 468, 810, 536, 878, 0, 114, 308, 650, 274, 844],
      [388, 480, 1164, 628, 514, 662, 890, 354, 696, 422, 764, 114, 0, 194, 536, 388, 730],
      [354, 674, 1130, 822, 708, 628, 856, 320, 662, 388, 730, 308, 194, 0, 342, 422, 536],
      [468, 1016, 788, 1164, 1050, 514, 514, 662, 320, 274, 388, 650, 536, 342, 0, 764, 194],
      [776, 868, 1552, 560, 674, 1050, 1278, 742, 1084, 810, 1152, 274, 388, 422, 764, 0, 798],
      [662, 1210, 754, 1358, 1244, 708, 480, 856, 514, 468, 354, 844, 730, 536, 194, 798, 0],
```

```
        # fmt: on
    ]
    data["num_vehicles"] = 4
    data["depot"] = 0
    return data


def print_solution(data, manager, routing, solution):
    """Prints solution on console."""
    print(f"Objective: {solution.ObjectiveValue()}")
    max_route_distance = 0
    for vehicle_id in range(data["num_vehicles"]):
        index = routing.Start(vehicle_id)
        plan_output = f"Route for vehicle {vehicle_id}:\n"
        route_distance = 0
        while not routing.IsEnd(index):
            plan_output += f" {manager.IndexToNode(index)} -> "
            previous_index = index
            index = solution.Value(routing.NextVar(index))
            route_distance += routing.GetArcCostForVehicle(
                previous_index, index, vehicle_id
            )
        plan_output += f"{manager.IndexToNode(index)}\n"
        plan_output += f"Distance of the route: {route_distance}m\n"
        print(plan_output)
        max_route_distance = max(route_distance, max_route_distance)
    print(f"Maximum of the route distances: {max_route_distance}m")



def main():
    """Entry point of the program."""
    # Instantiate the data problem.
    data = create_data_model()

    # Create the routing index manager.
    manager = pywrapcp.RoutingIndexManager(
        len(data["distance_matrix"]), data["num_vehicles"], data["depot"]
    )

    # Create Routing Model.
    routing = pywrapcp.RoutingModel(manager)
```

```
# Create and register a transit callback.
def distance_callback(from_index, to_index):
    """Returns the distance between the two nodes."""
    # Convert from routing variable Index to distance matrix NodeIndex.
    from_node = manager.IndexToNode(from_index)
    to_node = manager.IndexToNode(to_index)
    return data["distance_matrix"][from_node][to_node]

transit_callback_index = routing.RegisterTransitCallback(distance_callback)

# Define cost of each arc.
routing.SetArcCostEvaluatorOfAllVehicles(transit_callback_index)

# Add Distance constraint.
dimension_name = "Distance"
routing.AddDimension(
    transit_callback_index,
    0,  # no slack
    3000,  # vehicle maximum travel distance
    True,  # start cumul to zero
    dimension_name,
)
distance_dimension = routing.GetDimensionOrDie(dimension_name)
distance_dimension.SetGlobalSpanCostCoefficient(100)

# Setting first solution heuristic.
search_parameters = pywrapcp.DefaultRoutingSearchParameters()
search_parameters.first_solution_strategy = (
    routing_enums_pb2.FirstSolutionStrategy.PATH_CHEAPEST_ARC
)

# Solve the problem.
solution = routing.SolveWithParameters(search_parameters)

# Print solution on console.
if solution:
    print_solution(data, manager, routing, solution)
else:
    print("No solution found !")
```

```
if __name__ == "__main__":
    main()
def create_distance_matrix(data):
  addresses = data["addresses"]
  API_key = data["API_key"]
  # Distance Matrix API only accepts 100 elements per request, so get rows in multiple requests.
  max_elements = 100
  num_addresses = len(addresses) # 16 in this example.
  # Maximum number of rows that can be computed per request (6 in this example).
  max_rows = max_elements // num_addresses
  # num_addresses = q * max_rows + r (q = 2 and r = 4 in this example).
  q, r = divmod(num_addresses, max_rows)
  dest_addresses = addresses
  distance_matrix = []
  # Send q requests, returning max_rows rows per request.
  for i in range(q):
    origin_addresses = addresses[i * max_rows: (i + 1) * max_rows]
    response = send_request(origin_addresses, dest_addresses, API_key)
    distance_matrix += build_distance_matrix(response)

  # Get the remaining remaining r rows, if necessary.
  if r > 0:
    origin_addresses = addresses[q * max_rows: q * max_rows + r]
    response = send_request(origin_addresses, dest_addresses, API_key)
    distance_matrix += build_distance_matrix(response)
  return distance_matrix
```

Додаток Г

(обов`язковий)

ПРОТОКОЛ
ПЕРЕВІРКИ КВАЛІФІКАЦІЙНОЇ РОБОТИ
НА НАЯВНІСТЬ ТЕКСТОВИХ ЗАПОЗИЧЕНЬ

Назва роботи: «Автоматизована система оптимізації розкладу руху маршрутного транспорту з урахуванням часу простою»

Тип роботи: _____Магістерська кваліфікаційна робота_____
(БДР, МКР)

Підрозділ _____АІІТ, ФІІТА_____
(кафедра, факультет)

Показники звіту подібності Unicheck

Оригінальність _91,3_%_____ Схожість_8,7_%_____

Аналіз звіту подібності (відмітити потрібне)

- Запозичення, виявлені у роботі, оформлені коректно і не містять ознак плагіату.
- Виявлені у роботі запозичення не мають ознак плагіату, але їх надмірна кількість викликає сумніви щодо цінності роботи і відсутності самостійності її автора. Роботу направити на розгляд експертної комісії кафедри.
- Виявлені у роботі запозичення є недобросовісними і мають ознаки плагіату

та/або в ній містяться навмисні спотворення тексту, що вказують на спроби приховування недобросовісних запозичень.

Особа, відповідальна за перевірку_____ _____Роман МАСЛІЙ_____
(підпис) (ім`я, прізвище)

Ознайомлені з повним звітом подібності, який був згенерований системою

Unichek щодо роботи.

Автор роботи _____ _____Роман БАРАНЕЦЬ_____
(підпис) (ім`я, прізвище)

Керівник роботи _____ _____Олена НИКИТЕНКО
(підпис) (ім`я, прізвище)