**НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ**
**«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ**
**імені ІГОРЯ СІКОРСЬКОГО»**

**Факультет інформатики та обчислювальної техніки**

**Кафедра інформаційних систем та технологій**

До захисту допущено:

Завідувач кафедри

_____ Олександр РОЛІК

«___»_____20__ р.

# Дипломний проєкт

**на здобуття ступеня бакалавра**
**за освітньо-професійною програмою «Інформаційні управляючі системи та технології»**
**спеціальності 126 «Інформаційні системи та технології»**
**на тему: «Інформаційна система маршрутизації БПЛА. Підсистема складання маршрутів за допомогою ройового інтелекту»**

Виконала:
студентка IV курсу, групи ІС-93
Карпельова Ірина Олександрівна _____

Керівник:
доцент, к.т.н., доцент
Жданова Олена Григорівна _____

Рецензент:
доцент каф. ОТ, с.н.с., доцент
Антонюк Андрій Іванович _____

Засвідчую, що у цьому дипломному проєкті немає запозичень з праць інших авторів без відповідних посилань.
Студентка _____

Київ – 2023 року

**Національний технічний університет України**
**«Київський політехнічний інститут імені Ігоря Сікорського»**
**Факультет інформатики та обчислювальної техніки**
**Кафедра інформаційних систем та технологій**

Рівень вищої освіти – перший (бакалаврський)

Спеціальність – 126 «Інформаційні системи та технології»

Освітньо-професійна програма «Інформаційні управляючі системи та технології»

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Олександр РОЛІК
«___»_____20__ р.

**ЗАВДАННЯ**
**на дипломний проєкт студенту**
**Карпельовій Ірині Олександрівні**

1. Тема проєкту «Інформаційна система маршрутизації БПЛА. Підсистема складання маршрутів за допомогою ройового інтелекту», керівник проєкту Жданова Олена Григорівна, к.т.н., доцент, затверджені наказом по університету від 31 травня 2023 р. № 2101-с

2. Термін подання студентом проєкту: 12 червня 2023 року

3. Вихідні дані до проєкту: мова програмування TypeScript, редактор вихідного коду Visual Studio Code, бібліотека React, бібліотека MobX

4. Зміст пояснювальної записки: загальні положення, інформаційне забезпечення, математичне забезпечення, програмне і технічне забезпечення, технологічний розділ

5. Перелік графічного матеріалу (із зазначенням обов’язкових креслеників, плакатів, презентацій тощо): діаграма діяльності, діаграма варіантів використання, діаграма компонентів, схема структурна алгоритму.

6. Дата видачі завдання 27 лютого 2023 року

Календарний план

| № з/п | Назва етапів виконання дипломного проєкту | Термін виконання етапів проєкту | Примітка |
|---|---|---|---|
| 1 | Аналіз та уточнення технічного завдання | 24.04.2023 | |
| 2 | Аналіз предметної області, ознайомлення з аналогами, розробка функціональної моделі | 26.04.2023 | |
| 3 | Аналіз та проєктування математичного забезпечення | 01.05.2023 | |
| 4 | Проєктування та опис інформаційного забезпечення, реалізація програмного забезпечення | 04.05.2023 | |
| 5 | Реалізація програмного забезпечення | 08.05.2023 | |
| 6 | Опис програмного та технічного забезпечення | 10.05.2023 | |
| 7 | Оформлення звіту дипломного проєкту | 17.05.2023 | |
| 8 | Подання ДП на попередній захист | 05.06.2023 | |
| 9 | Подання ДП на основний захист | 12.06.2023 | |

Студент Ірина КАРПЕЛЬОВА

Керівник Олена ЖДАНОВА

# АНОТАЦІЯ

Карпельова І.О. Інформаційна система маршрутизації БПЛА. Підсистема складання маршрутів за допомогою ройового інтелекту. КПІ ім. Ігоря Сікорського, Київ, 2023.

У даному проєкті міститься текст, який складається з 72 сторінок, 5 розділів, 19 рисунків, 14 таблиць, 4 графічних матеріалів та посилання на 22 літературні джерела.

Ключові слова: задача маршрутизації транспортних засобів, безпілотний літальний апарат, оптимізація маршрутів, алгоритм мурашиних колоній.

Предметом розробки є інформаційна система маршрутизації БПЛА.

Метою розробки даної системи є створення маршрутів, які дозволяють досягнути мінімуму загального час, враховуючи час за який потрібно перезарядити безпілотний літальний апарат.

У дипломному проєкті розроблена клієнтська сторона інформаційної системи маршрутизації, яка включає інтерфейс та функціональність, які спрямовані на забезпечення взаємодії операторів з системою. Також розроблений та описаний алгоритм мурашиних колоній за допомогою якого відбувається побудова ефективного маршрутів. В даній роботі було проведено підбір параметрів для алгоритму та проведений аналіз його ефективності.

Отримані результати можуть бути використані з метою досягнення подібних цілей у процесі автоматизованої маршрутизації для аналогічних або схожих об'єктах.

SUMMARY

Karpelova I.O. Information System for UAV Routing. Route Planning Subsystem using Ant Colony Optimization. Igor Sikorsky Kyiv Polytechnic Institute, Kyiv, 2023.

This project contains a text composed of 72 pages, 5 chapters, 19 figures, 14 tables, 4 graphical materials, and references to 22 literary sources.

Keywords: vehicle routing problem, unmanned aerial vehicle, route optimization, ant colony algorithm.

The subject of development is an information system for UAV routing. The aim of developing this system is to create routes that minimize the total time, taking into account the time required for UAV recharging.

In the diploma project, the client-side of the routing information system was developed, which includes the interface and functionality aimed at facilitating user interaction with the system. Additionally, the ant colony algorithm was developed and described, which is used for constructing efficient routes. In this work, parameter tuning for the algorithm was performed, and its effectiveness was analyzed.

The obtained results can be utilized to achieve similar objectives in the process of automated routing for similar or related objects.

| Номер рядка | Формат | Позначення | Найменування | Кільк. аркушів | Номер екзем. | Примітка |
|---|---|---|---|---|---|---|
| 1 | | | Документація загальна | | | |
| 2 | | | | | | |
| 3 | | | Знову розроблена | | | |
| 4 | | | | | | |
| 5 | А4 | ІС93.090БАК.004 ПЗ | Пояснювальна записка | 72 | | |
| 6 | А3 | ІС93.090БАК.004 Д1 | Діаграма діяльності | 1 | | |
| 7 | А3 | ІС93.090БАК.004 Д2 | Діаграма варіантів | 1 | | |
| 8 | | | використання | | | |
| 9 | А3 | ІС93.090БАК.004 Д3 | Схема структурна алгоритму | 1 | | |
| 10 | А3 | ІС93.090БАК.004 Д4 | Діаграма компонентів | 1 | | |
| 11 | | | | | | |
| 12 | | | | | | |
| 13 | | | | | | |
| 14 | | | | | | |
| 15 | | | | | | |
| 16 | | | | | | |
| 17 | | | | | | |
| 18 | | | | | | |
| 19 | | | | | | |
| 20 | | | | | | |
| 21 | | | | | | |
| 22 | | | | | | |
| 23 | | | | | | |
| 24 | | | | | | |
| 25 | | | | | | |
| 26 | | | | | | |
| 27 | | | | | | |
| 28 | | | | | | |

| Зм. | Аркуш | № докум. | Підпис | Дата | ІС93.090БАК.004 ТП | | |
|---|---|---|---|---|---|---|---|
| Розроб. | | Карпельова І.О.. | | | Інформаційна система маршрутизації БПЛА. Підсистема складання маршрутів за допомогою ройового інтелекту. Відомість проєкту | Літ. | Аркуш | Аркушів |
| Керівн. | | Жданова О.Г. | | | | | 1 | 1 |
| | | | | | | КПІ ім. Ігоря Сікорського | | |
| | | | | | | Група ІС-93 | | |
| Затв. | | | | | | | | |

# Пояснювальна записка
## до дипломного проєкту
## на тему: «Інформаційна система маршрутизації БПЛА. Підсистема складання маршрутів за допомогою ройового інтелекту»

Київ – 2023 року

ЗМІСТ

# ВСТУП

Україна, як і багато інших країн, знаходиться в епоху стрімкого розвитку технологій, де нові інноваційні рішення стають ключем до вирішення складних проблем.

У сучасному світі, безпілотні літальні апарати (БПЛА) знайшли широке застосування для розв'язання різноманітних проблем.

БПЛА можуть бути використані для виявлення та моніторингу лісових пожеж, контролю над розповсюдженням вогню та надання даних для ефективної боротьби з ним. Вони також можуть здійснювати пошук та рятування в аварійних ситуаціях, зокрема при зсувах ґрунту, повенях або природних катастрофах. БПЛА забезпечують швидкий доступ до важкодоступних місць і надають реальний часовий звіт для координації рятувальних операцій.

Моніторинг міста за допомогою БПЛА дозволяє виявляти потенційні проблеми та ризики, такі як: транспортні затори, несправності в енергетичній мережі, забруднення довкілля. Це дозволяє владі міста та іншим зацікавленим сторонам приймати обґрунтовані рішення щодо планування містобудування, управління транспортом, екологічної політики та безпеки.

БПЛА можуть використовуватися для огляду інфраструктурних об'єктів, таких як: мости, магістралі, електроенергетичні лінії тощо. Вони забезпечують можливість виявлення потенційних проблем або пошкоджень, що дозволяє здійснювати своєчасні ремонти та підтримувати безпеку і надійність інфраструктури.

Для автоматизації та вдосконалення управління запасами безпілотні літальні апарати дозволяють проводити моніторинг запасів, виявляти ідентифікаційні мітки, а також контролювати стан і розташування товарів на складах. Це сприяє точному та ефективному управлінню запасами, зниженню втрат і оптимізації процесів замовлення та постачання.

Використання безпілотних літальних апаратів у моніторингу вуглеводневих родовищ стає все більш актуальним і ефективним рішенням. БПЛА можуть бути

оснащені різноманітними сенсорами, які здатні збирати дані про стан родовищ, включаючи температуру, тиск, склад ґрунту та атмосфери, концентрацію газів тощо. Завдяки можливостям безперешкодного проникнення у важкодоступні місця, вони можуть охоплювати значну площу родовища та забезпечувати швидкий та точний моніторинг.

Також використання БПЛА актуальне в сільському господарстві, адже воно відкриває широкі можливості для удосконалення та оптимізації сільськогосподарських процесів. БПЛА вносять значний вклад у підвищення ефективності, точності та економічної рентабельності сільського господарства.

Одним з основних застосувань БПЛА у сільському господарстві є здійснення аерофотозйомки та відеозапису. Завдяки високому розташуванню в повітряному просторі, БПЛА можуть здійснювати зйомку з різних кутів та висот, що дозволяє отримувати детальні та об'єктивні зображення сільськогосподарських угідь, полів і насаджень. Це допомагає аналізувати стан рослин, виявляти хвороби, шкідників, здоровість рослин, а також контролювати врожайність.

Тоді постає питання про складання маршрутів та час, який буде витрачений на обліт БПЛА при виконанні поставлених задач. Для цього спочатку необхідно розглянути задачі маршрутизації транспортних засобів.

Метою розробки даної системи є створення маршрутів, які дозволяють досягнути мінімуму загального часу обльоту об’єктів, враховуючи час, за який необхідно перезарядити безпілотний літальний апарат.

У даному дипломному проєкті розробляється частина комплексної роботи, а саме клієнтська частина.

Для виконання поставленої мети потрібно виконати два блока, а саме:

а) блок розв’язання задачі:
   1) зчитування вхідних даних;
   2) ведення вхідних даних вручну;
   3) складання маршрутів обльоту;
   4) візуалізація складеного маршруту;

5) візуалізація результатів розв’язаної задачі;

б) блок формування маршрутів алгоритмом мурашиних колоній.

Основні завдання дипломного проєктування включають наступне:

– аналіз задач маршрутизації та алгоритмів їх розв’язання;

– розробка алгоритму складання оптимальних маршрутів з урахуванням різних обмежень, таких як: швидкість БПЛА, час перезарядки, часовий ресурс БПЛА;

– програмна реалізація оптимізації маршруту для забезпечення максимальної ефективності польоту;

– визначення параметрів алгоритму, які забезпечують його ефективне функціонування;

– розробка інтерфейсу оператора для зручного налаштування параметрів маршруту та візуалізації складених маршрутів;

– підтримка візуалізації маршруту на карті та надання детальної інформації оператору.

# 1 ЗАГАЛЬНІ ПОЛОЖЕННЯ

## 1.1 Опис предметного середовища

Задача маршрутизації безпілотних літальних апаратів (БПЛА) – це складна задача оптимізації, яка передбачає визначення оптимальних маршрутів для групи безпілотних літальних апаратів, якими слід скористатись для виконання ряду завдань.

Безпілотні літальні апарати відкривають нові перспективи для вдосконалення логістики в різних умовах. За допомогою встановлення різноманітних датчиків зображення, БПЛА можуть здійснювати захоплення зображень об'єктів. Відстежені зображення можуть передаватися у режимі реального часу на контрольну станцію за допомогою бездротової передачі даних. Особливо варто відзначити їх здатність до переміщень без необхідності використання основної мережі. Останні технологічні прогреси, а саме: збільшення часу автономної роботи батареї, покращені комунікаційні пристрої та зниження виробничих витрат – призвели до зростання використання БПЛА у різних сферах.

На противагу традиційним земним транспортним засобах, які обов'язково слідують визначеним маршрутам, безпілотні апарати можуть прямувати безпосередньо між пунктами призначення, використовуючи повітряні простори для своїх переміщень. Однак, під час планування маршрутів потрібно пам’ятати про вплив метеорологічних умов, такі як сильний вітер, туман, дощ або сніг, які можуть обмежувати можливість безпечного польоту. Також БПЛА має лімітований запас палива, що обмежує їх здатність до виконання тривалих місій або довгих маршрутів [1].

Предметне середовище може включати наступні складові:

– безпілотний повітряний апарат (БПЛА) – це фізичний апарат, що використовується для збору даних і виконання місій. БПЛА може бути у різних розмірах та типах, включаючи дрони, літальні апарати великої витривалості та інші;

– об’єкти – це місця, які потрібно відвідати або облетіти, використовуючи безпілотні літальні апарати. Ці об'єкти мають різні функціональні призначення. Наприклад, зйомка відео та фото, місця збору інформації, території для пошуку та рятування;

– станції для перезарядки – це пункти, які призначені для перезарядки енергетичних джерел безпілотних літальних апаратів, такі як: акумулятор або паливні баки. Ці станції надають змогу БПЛА підключатись до зарядки або заправки, для того щоб продовжити свій політ.

### 1.1.1 Опис процесу діяльності

В даному дипломному проєкті ми розглядаємо діяльність одного актора, а саме: оператора БПЛА. На початку оператор, повинен ввійти в систему та надати інформацію про об’єкти, які БПЛА повинен відвідати для виконання певних завдань. Система отримує інформацію та відправляє її на сервер. Отримавши дані, система будує маршрут, включаючи станції для перезарядки БПЛА. Далі клієнтська частина його отримує і виводить результат на екран оператора. В кінцевому результаті можна побачити візуалізацію плану маршруту.

Діаграму процесу діяльності наведене на кресленику ІС93.090БАК.004 Д1.

### 1.1.2 Опис функціональної моделі

Система складається з двох акторів, а саме: оператор та експериментатора. У даному дипломі розроблено частина функціонала оператора.

Оператор може виконувати три операції, а саме: захід в особистий акаунт, складання маршруту відвідання об’єктів з використанням безпілотних літальних апаратів, переглядання результатів складання маршруту.

Для початку роботи у системі оператор повинен зайти у особистий акаунт.

Для того щоб скласти маршрут, оператор повинен спочатку вести координати об’єктів для відвідання та станцій для перезарядки. В нього є

можливість ввести координати вручну або зчитати координати об’єктів та станцій з JSON – файлу. Оператор також може додавати нові об’єкти або видаляти вже створені.

Далі оператор повинен виконати ведення даних про характеристику БПЛА, а саме: обмеження на довжину маршруту, середню швидкість переміщення та час, який необхідний для перезарядки на станціях.

Наступний крок це складання маршрутів. Під час перегляду результатів оператор може переглянути також час та довжину проходження маршруту, впорядковані об’єкти при відвіданні. У випадку необхідності оператор має можливість зберегти результати у JSON – файл.

Діаграму варіантів використання наведено на кресленику ІС93.090БАК.004 Д2.

## 1.2 Огляд наявних аналогів

Розглянемо пару наявних аналогів платформ, які можуть бути використані для планування маршрутів з використанням БПЛА.

DJI Terra – це програмне забезпечення, яке створила компанія DJI, призначене для обробки аерофотознімків та планування маршрутів безпілотних літальних апаратів (БПЛА). Воно було випущене у квітні 2018 року й розроблене спеціально для використання з БПЛА виробництва DJI.

DJI Terra має розширений функціонал, який дозволяє створювати точні 3D-моделі, картографічні дані й ортофотоплани на основі зроблених аерофотознімків з використанням БПЛА DJI. Програма також пропонує можливості оптимізації маршрутів з метою мінімізації часу.

Для оптимізації маршрутів DJI Terra використовує різноманітні алгоритми, які враховують фактори, такі як швидкість БПЛА, обмеження польоту, рельєф місцевості й інші параметри. Можна налаштувати різні параметри маршруту, такі як висота польоту, перекриття знімків й інтервал між ними, й програма

автоматично розрахує оптимальний маршрут для збору даних з мінімальним часом.

Оптимізований маршрут дозволяє ефективно збирати необхідні дані й максимально зменшити час, необхідний для виконання місії. Крім того, DJI Terra забезпечує можливості візуалізації та аналізу отриманих даних, що дозволяє їх використання у геопросторових додатках та рішеннях[2].

Altitude Angel – це платформа для управління повітряним простором та маршрутизації дронів. Вона пропонує різноманітні рішення, включаючи платформу для реєстрації та ідентифікації дронів, маршрутизацію з урахуванням обмежень повітряного простору та навколишнього середовища, моніторинг та контроль польотів, а також інтеграцію з іншими системами та сервісами, такими як системи навігації та погодні сервіси.

Однією з ключових переваг платформи Altitude Angel є її здатність автоматично оптимізувати маршрути дронів, забезпечуючи найбільш ефективні траєкторії перельоту з мінімальними витратами часу та енергії. Вона враховує різні фактори, такі як обмеження повітряного простору, розташування інших повітряних суден, зони з обмеженим доступом та інші параметри, щоб забезпечити безпеку та ефективність дронових польотів.

Платформа Altitude Angel також дозволяє інтегруватися з іншими системами та сервісами, створюючи комплексні рішення для керування місіями дронів. Вона дозволяє операторам дронів планувати маршрути, встановлювати обмеження та вимоги польоту, контролювати безпеку польоту та взаємодіяти з іншими учасниками повітряного простору [3].

## 1.3 Постановка задачі

### 1.3.1 Призначення розробки

Призначення розробки інформаційної системи є складання маршрутів для безпілотних літальних апаратів. Ця система забезпечує автоматизацію процесу побудови маршрутів, враховуючи обмеження, які впливають на планування

маршруту, а саме: середню швидкість переміщення безпілотного літального апарату, час, за який необхідно перезарядити БПЛА та часовий обмежений потенціал БПЛА.

### 1.3.2 Цілі та задачі розробки

Ціллю розробки даної системи є створення маршрутів, які дозволяють досягнути мінімуму загального часу обльоту об’єктів, враховуючи час, за який необхідно перезарядити безпілотний літальний апарат.

Для виконання поставленої мети потрібно виконати два блока, а саме:

а) блок розв’язання задачі:

1) зчитування вхідних даних;
2) ведення вхідних даних вручну;
3) складання маршрутів обльоту;
4) візуалізація складеного маршруту;
5) візуалізація результатів розв’язаної задачі;

б) блок формування маршрутів алгоритмом мурашиних колоній.

Для виконання поставлених задач та цілі необхідно виконати наступні завдання, а саме:

– аналіз задач маршрутизації та алгоритмів їх розв’язання;

– розробка алгоритму складання ефективних маршрутів з урахуванням обмежень, таких як: швидкість БПЛА, час перезарядки, часовий ресурс БПЛА;

– програмна реалізація оптимізації маршруту для забезпечення максимальної ефективності польоту;

– визначення параметрів алгоритму, які забезпечують його ефективну функціонування;

– розробка інтерфейсу оператора для зручного налаштування параметрів маршруту та візуалізації складених маршрутів;

– підтримка візуалізації маршруту на карті та надання детальної інформації оператору.

Висновки до розділу

У даному розділі було розглянуто загальні положення проєкту, які визначають його основу для подальшої розробки. Було детально описано предметне середовище, включаючи процес діяльності та функціональну модель. Процес діяльності та функціональну модель було описано як і для оператор, так і для експериментатора. Також було проведено огляд наявних аналогів з метою визначення унікальності та особливостей запропонованого проєкту.

Була сформульована постановка задачі, включаючи призначення розробки, визначення цілей, завдань та задач проєкту.

# 2 ІНФОРМАЦІЙНЕ ЗАБЕЗПЕЧЕННЯ

## 2.1 Опис вхідних даних

Вхідні дані системи є інформацією, яка надається до системи з зовнішніх джерел або внутрішніх компонентів системи. Вони є початковими даними, на основі яких система виконує свої функції та обробляє інформацію.

Опишемо вхідні дані, які надаються до нашої системи.

Дані про безпілотний літальний апарат:

– дані про середню швидкість переміщення;

– дані про часовий обмежений потенціал (можливий час обльотів без перезарядки);

– дані про час, за який необхідно перезарядити БПЛА.

Дані про об’єкти:

– дані про географічне положення (градуси довготи та широти).

Дані про станції для перезарядки:

– дані про географічне положення (градуси довготи та широти).

## 2.2 Опис вихідних даних

В нашій системі вихідні дані це схема маршруту, яка складається з кількох підмаршрутів, які включають всі об’єкти, та станції для перезарядки. Також на виході оператор отримує значення сумарної відстані обльоту та сумарний час маршруту БПЛА. Вихідні дані можуть бути використані для передачі інформації іншим операторам. На рисунку 2.1 зображено приклад маршруту, який система буде відображати оператору після розв’язання задачі.

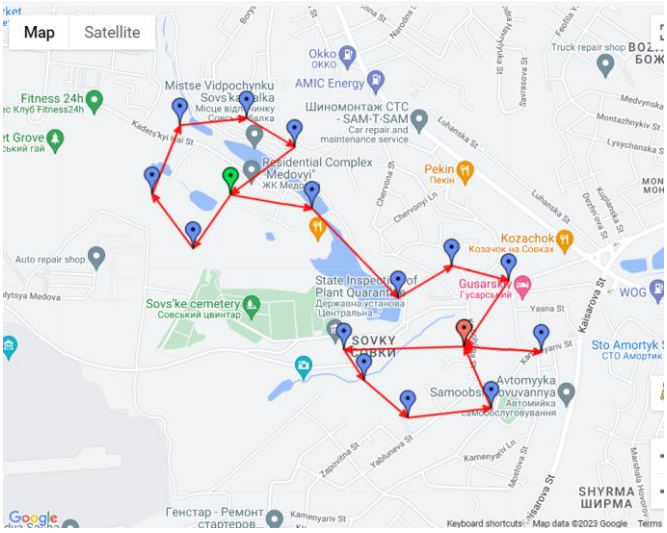


Рисунок 2.1 – Приклад побудованого маршруту

## 2.3 Опис JSON-файлу

Для ведення даних, щоб розв’язати задачу, оператор може завантажити файл. Система зчитує JSON-файл, який має наступну структуру:

– масив з даними про географічне положення об’єктів обльоту;

– координати станції з якої БПЛА починає маршрут;

– координати станції, на якій БПЛА здійснює процес перезарядки або завершує маршрут;

– середня швидкість переміщення БПЛА;

– часовий обмежений потенціал БПЛА;

– час, за який необхідно перезарядити БПЛА.

## Висновки по розділу

У даному розділі були описані вхідні дані, які надходять до системи для задач маршрутизації БПЛА, а саме дані про безпілотний літальний апарат, об’єкти та станції для перезарядки. Описані вихідні дані та зображений приклад вихідного маршруту. Також система може зчитувати JSON-файл, структуру якого було описано в даному розділі.

# 3 МАТЕМАТИЧНЕ ЗАБЕЗПЕЧЕННЯ

## 3.1 Класифікація задач оптимізації маршрутів транспортних засобів

Однією з найбільш вивчених та складних областей комбінаторної оптимізації є розробка маршрутів для транспортних засобів. У певних сегментах ринку транспортні витрати можуть складати значну частку в загальних витратах на товари, тому використання ефективних інструментів оптимізації може призвести до значного зниження загальних витрат. За останні десятиліття було проведено багато опитувань і досліджень щодо цієї проблеми та її різновидів.

В реальних життєвих ситуаціях існує безліч різних обмежень, які потрібно враховувати під час планування маршрутів, тому на даний момент існують різні види задач оптимізації маршрутів транспортних засобів. Наприклад, можуть бути обмеження на час руху, вантажопідйомність транспортних засобів, кількість клієнтів, яких необхідно обслуговувати, а також їх вимоги до часу доставки та інші. В залежності від того, які обмеження потрібно враховувати, створюються різні види задач маршрутизації транспортних засобів. Крім того, різні види задач можуть мати різні цілі оптимізації, такі як мінімізація часу, витрат на паливо або кількості транспортних засобів, що використовуються.

Задачі оптимізації маршрутів транспортних засобів можна класифікувати за різними ознаками. Основні з них:

– кількість транспортних засобів: задачі маршрутизації можуть вирішуватися для одного або більше транспортних засобів;

– кількість пунктів доставки: можуть існувати задачі з одним або більше депо;

– обмеження на кількість пунктів, які може відвідати транспортний засіб: в деяких випадках кількість пунктів, які може відвідати транспортний засіб, може бути обмеженою;

– часові обмеження: можуть бути задачі маршрутизації з обмеженнями часу, наприклад, якщо потрібно доставити товари за обмежений час;

– симетричність: задачі можуть бути симетричними, якщо відстань між будь-якими двома точками на маршруті є однаковою у двох напрямках;

– вид руху: задачі маршрутизації можуть вирішуватися для транспортних засобів, що рухаються від точки до точки або з точки до точки і назад [4].

Розглянемо найбільш поширені види задач оптимізації маршрутів транспортних засобів.

Capacitated Vehicle Routing Problem – задача оптимізації маршрутів транспортних засобів, враховуючи їх завантаженість. Ці задачі мають додаткове обмеження, яке полягає в обмеженні максимальної маси товарів, що можуть бути перевезені на кожній маршрутній схемі до певного значення, при цьому увесь транспорт має однакову місткість.

В міській логістиці часто використовується тактика багатоетапного розподілу, за якої вантажі доставляються клієнтам через проміжні склади, замість безпосередніх поставок [5].

VRP with Time Windows – задача оптимізації маршрутів транспортних засобів з урахуванням часових обмежень, також їх ще називають «часові вікна». Ця задача має додаткову умову, а саме: у кінцеву точку або адресу доставки потрібно транспортувати вантаж в заданий інтервал часу. У даних задачах додатково встановлюються також і інші умови. Якщо вантаж був відвезений до місця призначення після закінчення максимально допустимого часу доставки, то розв’язок недопустимий. Транспорт, який прибув до точки призначення до часу, який зазначений нижньою межею інтервалу, знаходиться в очікуванні. Після отримання розв’язку, ми можемо визначити момент виїзду транспорту з пункту розподілу, одночасно з цим уникнути непотрібного очікування та оптимізувати робочий час [5].

Multiple Depot VRP – задача оптимізації маршрутів транспортних засобів з кількома депо. У цьому випадку, оператори послуг обслуговуються з декількох різних баз в протилежність класичної задачі VRP, де шлях доставки починається та завершується в одній базі. Для вирішення цієї задачі необхідно розподілити клієнтів на різні бази, які мають власний автопарк. Транспортні засоби

розпочинають свій маршрут з відповідної бази, обслуговують клієнтів, закріплених за цією базою, а потім повертаються назад. Розв'язання задачі Multiple Depot VRP має велике значення для компаній та організацій, які мають розподілену базу клієнтів або різні логістичні центри. Multiple Depot VRP знайшла широке застосування в таких сферах, як логістика, дистрибуція, кур'єрські послуги, електронна комерція [5].

Задача маршрутизації безпілотних літальних апаратів часто відносять у сучасній літературі до задач оптимізації маршрутів транспортних засобів з кількома депо. Метою даних задач є прокладання оптимального маршруту через заданий набір об'єктів з мінімізацією загальної відстані або тривалості польотів. При цьому необхідно дотримуватися умов, що кожен об'єкт буде відвіданим лише одним БПЛА, а всі об'єкти повинні бути включеним до маршруту і лише один раз. Так як об’єкти лише відвідуються без необхідності доставки вантажу, це означає, що відсутнє обмеження на вантажопідйомність [6].

## 3.2 Змістовна постановка задачі

Маємо множину об’єктів для дослідження, які повинні бути обстежені одним БПЛА. Оскільки БПЛА має обмежений час автономної роботи батареї, то маємо дві станції для перезарядки c1 та c2. Перший маршрут БПЛА починає з станції c1, а завершує у будь-якій з двох, наступний маршрут БПЛА починає з тієї станції, на якій він перезаряджався.

Потрібно визначити множину підмаршрутів з метою проведення обстеження всіх об’єктів, які потрібно облетіти за мінімальний час, з урахуванням необхідності перезарядки батареї БПЛА.

## 3.3 Математична постановка задачі

Сформулюємо математичну постановку задачі маршрутизації БПЛА, яка описана в попередньому розділі.

Вхідними даними задаються:

— $s$ – середня швидкість переміщення безпілотного літального апарату;

— $t$ – час за який необіхдно перезарядити БПЛА;

— $T$ – часовий обмежений потенціал БПЛА (м ожливий час обльотів без перезарядки).

Задана множина, яка складається з $n$ об'єктів обльоту БПЛА $A = \{0,1,\dots,(n-1)\}$ та множину станцій для перезарядки $\mathrm{C} = \{\mathrm{c}_1, \mathrm{c}_2\}$.

Маємо також матрицю відстаней між об'єктами, включаючи станції для перезарядки $d_{lm} = \{l, m\, \epsilon\{0,1,\dots,(n+1)\}\}$. При цьому об'єкти $n$ та *(n+1)* відповідають станціям для перезарядки $c_1$ та $c_2$ відповідно. Значення матриці часу відвідання об'єктів розраховані відповідно до матриці відстаней $\{d_{lm}\}$ та середньої швидкості переміщення БПЛА *s:* $r_{lm} = \{l, m\, \epsilon\{0,1,\dots,(n+1)\}\}$.

Потрібно визначити план відвідання об'єктів, який складається з $B$ маршрутів, при цьому $B$ – це величина, що невідома спочатку, але визначається під час розв'язання. Перший маршрут БПЛА починає з станції с1, а завершує у будь-якій з двох станцій С, наступний маршрут БПЛА починає з тієї станції, на якій він перезаряджався.

Таким чином слід розбити множину об'єктів обльоту $A$, на $B$ впорядкованих підмножин:

$$\left\{A_1 = \left\{a_0^1, a_1^1 \dots a_{n_1-1}^1\right\}, \dots, A_B = \left\{a_0^B, a_1^B \dots a_{n_B-1}^B\right\}\right\}, \tag{3.1}$$

таких що:

$$\cup_{b=1}^{B} A_b = A \quad (\textstyle\sum_{b=1}^{B}(n-1)_b = (n-1), \tag{3.2}$$

$$A_i \cap A_j = \emptyset, \text{де } i \neq j. \tag{3.3}$$

Зазначені вище умови гарантують, що кожен об'єкт буде врахований у складі маршруту, проте тільки в одному маршруті плану обльотів і лише один раз,

при цьому:

$A_1 = \{a_0^1, a_1^1 \dots a_{n_1-1}^1\}$ – об’єкти, що включені у перший маршрут ;

…

$A_B = \{a_0^B, a_1^B \dots a_{n_B-1}^B\}$ – об’єкти, що включені в останній маршрут B.

За умов, що жодна з підмножин об'єктів не містить станцій перезарядки. Підмножини об’єктів розділяються відвідуванням станціями перезарядки. Спочатку БПЛА вилітає з першої станції. Після того як він відвідає кожної з підмножин об’єктів, БПЛА летить на перезарядку до першої або другої станції. Закінчує БПЛА проходження усіх підмножин об’єктів на другій або першій станції.

Позначимо через $R_b$ – сумарний час проходження маршруту $b$ ($b = 1, \dots, B$):

$$R_1 = r_{c_1,a_0^1} + \sum_{i=0}^{n_1-1} r_{a_i^1,a_{i+1}^1} + r_{a_{n_1-1,C}^1}, C \in \{c_1, c_2\}, \tag{3.4}$$

$$R_b = r_{C,a_0^b} + \sum_{i=0}^{n_b-1} r_{a_i^b,a_{i+1}^b} + r_{a_{n_b-1,C}^b}, b = 1, \dots, B, C \in \{c_1, c_2\}. \tag{3.5}$$

На основі змістовного опису задачі було визначено цільову функцію – мінімізувати загальний час , який необхідний для прольоту всіх об’єктів, включно з часом, який потрібно витратити для перезарядки на станціях:

$$z = (B-1)t + \sum_{b=1}^{B} R_b \to min \tag{3.6}$$

Ця цільова функція мінімізує кількість маршрутів, включених до плану польотів, з урахуванням часу, необхідного для перезарядки, кількості перезарядки і, таким чином, кількості маршрутів, включених до розв'язку.

Загальний час, витрачений на відвідування всіх об’єктів на кожному маршруті, включаючи закінчення на одній з станцій , повинно не перевищувати часовий обмежений потенціал БПЛА:

$$R_b \leq T, \text{де } b = 1, .., B \ , \tag{3.7}$$

де

$$R_b = \sum_{l=0}^{n+1_b} \sum_{m=0}^{n+1_b} r_{lm}. \tag{3.8}$$

Обмеження, яке гарантує, що план відвідання об’єктів складається з одного або більше маршрутів, виглядає наступним чином :

$$B \geq 1 \tag{3.9}$$

Структура розв’язку буде мати наступний вигляд:

$\boldsymbol{c_1}, a_0^1, a_1^1, \dots, a_{n_1-1}^1 \boldsymbol{C}, \dots, \boldsymbol{C}, a_0^B, a_1^B, \dots, a_{n_B-1}^B \boldsymbol{C},$

де $C = \{c_1, c_2\}$.

Також на рисунку 3.1 зображено графічний вигляд структури розв’язку.

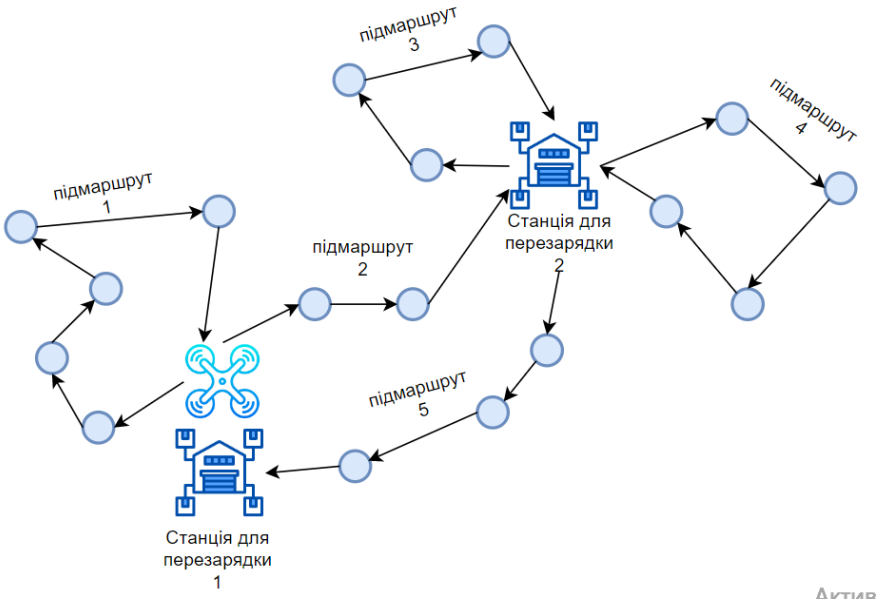


Рисунок 3.1 – Приклад структури розв'язку задачі

## 3.4 Аналіз методів розв’язання задач маршрутизації

Проблема задач маршрутизації безпілотних літальних апаратів є відомою складною задачею комбінаторної оптимізації. Для її вирішення потрібно знайти оптимальні маршрути. У практичних застосуваннях виникають складності при проєктуванні та управлінні розподілом системи, так як на побудову маршруту накладається кілька операційних обмежень.

На рисунку 3.2 зображено класифікація методів , якими можна розв’язати задачі маршрутизації [7].

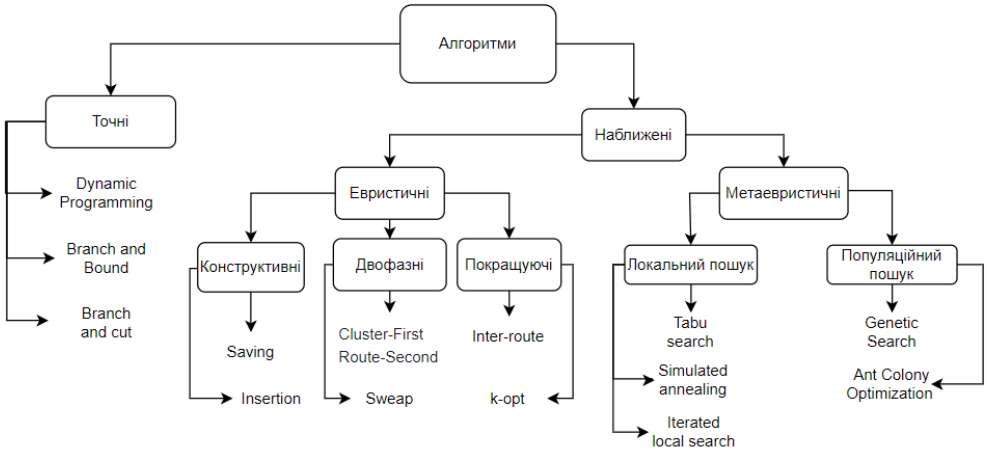


Рисунок 3.2 – Класифікація методів для розв’язання задач маршрутизації

### 3.4.1 Точні методи

Точні методи розв'язання задач маршрутизації базуються на математичних моделях та алгоритмах, які дозволяють знайти оптимальний розв'язок задачі. Ці методи є найбільш точними та надійними, оскільки вони гарантують знайдення оптимального розв'язку. Однак, вони мають обмеження в застосуванні до великих задач, оскільки вони можуть вимагати значних обчислювальних ресурсів і займати багато часу для розв'язання складних задач. Тому точні методи найчастіше використовуються при розв'язанні задач меншого масштабу, де їхні

переваги в точності та гарантованому оптимальному розв'язку переважають над обчислювальною складністю [8].

### 3.4.2 Наближені методи

Наближені методи (англ. approximate methods) – це алгоритми, які швидко надають розв'язок задачі маршрутизації, але не можуть гарантувати отримання оптимального розв’язку. Ці методи використовують евристичні або метаевристичні підходи для розв'язання задачі. Вони можуть бути корисні в тих випадках, коли точний розв'язок є занадто витратним з точки зору часу або ресурсів [7].

#### 3.4.2.1 Евристичні методи

Евристичні методи в контексті задач маршрутизації відносяться до алгоритмів, які використовують експертні знання та емпіричні правила для знаходження приблизних або оптимальних рішень. Евристичні методи поділяються на конструктивні, двофазні та покращуючі.

Конструктивні – це тип алгоритмів для розв'язання задач маршрутизації транспорту. Ці методи використовуються для створення початкового розв’язку для задачі VRP, яке потім може бути вдосконалено з використанням інших алгоритмів, наприклад, покращуючих методів. Основна ідея методів конструювання полягає в тому, щоб почати з порожнього маршруту та послідовно додавати клієнтів до маршруту до тих пір, поки не буде задоволене обмеження на кількість клієнтів, яких можна обслуговувати одним транспортним засобом.

Двофазні (кластерні) методи зазвичай складаються з двох основних етапів:

– Фаза розбиття на кластери: на цьому етапі вхідні дані розбиваються на підмножини (кластери) змінних та обмежень. Розбиття може здійснюватися за різними критеріями, наприклад, згрупувати змінні, які мають подібні властивості, або згрупувати обмеження, які мають схожу структуру.

– Фаза оптимізації: на цьому етапі кожен кластер розв'язується окремо, і знаходяться розв'язки для кожного кластера. Для цього можуть використовуватися різні методи розв'язання.

Методи покращення – це алгоритми, які спрямовані на поліпшення вже існуючого розв’язку, знаходячи краще розв’язку з меншою вартістю. Основна ідея методів покращення полягає в тому, щоб внести невеликі зміни в існуючий розв’язок, щоб отримати оптимальне. Ці невеликі зміни часто полягають у зміні порядку клієнтів між маршрутами, зміні порядку клієнтів у межах одного маршруту або зміні послідовності маршрутів. Мета полягає в тому, щоб знайти оптимальну послідовність маршрутів, яка відповідає всім обмеженням і мінімізує загальну вартість [9].

3.4.2.2 Метаевристичні методи

Метаевристичні методи використовують комбінацію локального та глобального пошуку рішень і об'єднують їх у абстрактні стратегії евристичної оптимізації задач.

Алгоритми локального пошуку є ітераційними методами, які базуються на переборі часткових варіантів серед точок, які знаходяться в околі поточної точки на кожній ітерації. Замість повного перебору використовується локальний перебір у підмножинах варіантів, які називаються околами. У задачах маршрутизації транспорту локальний пошук може використовуватись для покращення розв’язків, знайдених різними евристичними алгоритмами.

Популяційні методи черпають натхнення з природних концепцій, наприклад, еволюції видів і поведінки соціальних комах, які шукають їжу. Ці методи реалізують керівну стратегію високого рівня, засновану на різних структурах пам’яті, таких як нейронні мережі, пули розчинів, представлені у вигляді хромосом, або феромонних матриць. Крім того, усі відомі успішні евристики задач маршрутизації цього типу також покладаються на компоненти локального пошуку, щоб спрямувати пошук до перспективних рішень. Таким

чином, більшість популяційних методів у літературі є гібридними, що означає, що вони поєднують у собі різні підходи та компоненти з інших методів. Це дозволяє їм використовувати переваги кожного окремого підходу та забезпечує ефективну та збалансовану роботу в різних задачах. [8].

### 3.5 Різновиди мурашиних алгоритмів

Колонії мурашок та інші соціальні суспільства комах є розподіленими системами, що характеризуються високоорганізованою соціальною структурою, незважаючи на простоту окремих особин. Вони можуть виконувати складні завдання, які перевищують індивідуальні можливості окремих мурашок.

Сфера "мурашиних алгоритмів" досліджує моделі, отримані на основі спостережень за реальною поведінкою мурах, та використовує їх як джерело натхнення для розробки нових алгоритмів оптимізації та розподіленого керування. Основна ідея полягає в тому, що принципи самоорганізації, що забезпечуються високоорганізованою поведінкою реальних мурашок, можна використовувати для координації штучних агентів у вирішенні обчислювальних проблем.

Мурашині колонії надихнули створення різних видів мурашиних алгоритмів, які включають пошук їжі, розподіл праці та кооперативне транспортування. У всіх цих прикладах мурашки координують свою діяльність через стигмергію – непрямий спосіб комунікації, що здійснюється через модифікації середовища. Наприклад, мураха-пошуковик використовує хімічні речовини, щоб залишити сліди на землі, збільшуючи ймовірність, що інші мурахи підуть тим же шляхом. Біологи показали, що багато аспектів соціальної поведінки мурашиних колоній можна пояснити за допомогою простих моделей, в яких присутня лише стигмергічна комунікація. Іншими словами, дослідники показали, що часто стигмергічне, непряме спілкування вистачає для пояснення, як соціальні комахи досягають самоорганізації.

Ця ідея лежить в основі мурашиних алгоритмів і використовується для координації штучних суспільств агентів. Існує багато успішних прикладів мурашиних алгоритмів, одним з них є "мурашиний колоніальний оптимізатор" (ACO), який спеціалізується на розв'язанні дискретних проблем оптимізації. У цьому підході неочікувані відкриття, отримані з поведінки реальних мурашок у пошуку їжі, були застосовані для створення штучних мурах, які можуть вирішувати складні оптимізаційні задачі.

Таким чином, мурашині алгоритми використовують принципи самоорганізації та стигмергії для координації дії суспільств штучних агентів. Вони демонструють, що прості правила взаємодії між агентами можуть призвести до ефективного розв'язання складних завдань оптимізації. Ці алгоритми відкривають нові можливості для розробки імітаційних моделей та алгоритмів, які використовують природні принципи координації та колективного інтелекту для вирішення реальних проблем[10].

Розглянемо найбільш поширені різновиди алгоритмів мурашиних колоній.

Ant Colony System – покращує класичний алгоритм мурашиних систем AS шляхом використання інформації, накопиченої попередніми поколіннями, для удосконалення ймовірнісної моделі пошукового процесу. Це забезпечується на основі двох процесів.

В першу чергу це використання виключно стратегії пріоритетного вибору при оновленні феромонів, які мурахи залишають на ребрах, тобто феромон міняється лише на тих ребрах, що належать до найкращого розв’язку, який знайшли мурахи.

В другу чергу мурахи обирають наступний вузол для включення в частковий розв'язок, за використанням правила псевдовипадкового пропорційного вибору.

На етапі оновлення феромону, після закінчення активності кожного покоління мурах, він зберігається виключно у тій мурахі, яка виявила найкращий розв’язок.

Також у системі мурашиних колоній використовується онлайнове покрокове оновлення хімічних речовин мурах, що сприяє зниженню ймовірності вибору однакових маршрутів всіма учасниками популяції [8].

Max-Min Ant System

У MMAS обмежується кількість феромону, яку мурахи відкладають на ребрах, максимальним та мінімальним значеннями.

Основна мета MMAS полягає в уникненні надмірно великих або дуже малих значень феромону, що може призвести до передчасної збіжності або обмеженого дослідження. Шляхом обмеження діапазону значень феромону, MMAS спонукає мурах досліджувати різні шляхи та запобігає домінуванню одного шляху на початкових етапах алгоритму.

У MMAS спочатку всі значення феромону встановлюються на максимальне значення. Під час проходження мурахами ребер, вони відкладають феромон залежно від якості розв’язку. Проте, відкладений феромон обмежується максимальним та мінімальним обмеженнями, визначеними алгоритмом.

На кожній ітерації здійснюється глобальне оновлення, щоб налаштувати рівні феромону на ребрах. Найкраща мураха ітерації відкладає додатковий феромон на ребрах найкращого знайденого до того моменту розв’язку. Це посилення феромону на найкращому рішенні допомагає спрямовувати пошук в перспективні регіони простору розв’язків [11].

Elitist Ant System

В даному алгоритмі вводяться певна кількість «елітних» мурах у популяцію. Ці елітні мурахи мають здатність відкладати більшу кількість феромону на ребрах, які вони проходять, порівняно зі звичайними мурахами.

Основним завданням включення елітних мурів є підвищення ефективності використання потенційно перспективних рішень, які були виявлені до цього моменту. Шляхом активного відкладання більшої кількості феромону на ребрах найкращих шляхів, алгоритм спрямовує свої зусилля на пошук і концентрацію навколо цих високоякісних рішень.

Присутність елітних мурів у популяції покращує швидкість збіжності алгоритму, оскільки вони надають перевагу експлуатації добре зарекомендованих рішень. Проте важливо досягти рівноваги між дослідженням та експлуатацією. Якщо ввести занадто багато елітних мурів, це може спричинити передчасну збіжність, і алгоритм може застрягти в локальних оптимумах. З іншого боку, замала кількість елітних мурів може призвести до недостатнього дослідження простору пошуку [10].

Ant-Q

Л. Гамбарделла і М. Доріго написали дослідження, яке опублікували у 1995 році. В даній роботі вони представили вдосконалення алгоритму мурашиних колоній, запропонувавши його інтерпретацію як систему, що здатна до навчання. Даний метод отримав свою назву на основі методу машинного навчання і включив багато ідей з нього.

У цьому алгоритмі кожному ребру призначається значення, яке вказує на його придатність для переходу. Величина корисності переходу по ребру обчислюється на основі значень корисностей переходу через наступні ребра, які були визначені під час попереднього ідентифікація можливих наступних станів. Ці значення зберігаються в Q-таблиці, іншими слова дані модифікуються та оновлюються протягом роботи алгоритму [12].

## 3.6 Обґрунтування методу розв’язання

Для розв’язання нашої задачі було обрано алгоритм мурашиних колоній (ACO) , оскільки даний алгоритм здатний до глобальної оптимізації, тобто шукає оптимальний маршрут в усьому просторі можливих рішень. Також ACO використовує локальну інформацію, що дозволяє знаходити більш точні та адаптивні маршрути, враховуючи обмеження задачі.

Вперше алгоритм мурашиних колоній був запропонований Марком Доріго в 1992 році в рамках його докторської дисертації [8].

Розглянемо чотири основних етапи в поведінці мурах під час пошуку їжі.

На рисунку 3.3 зображено перший етап. Спочатку мурахи знаходяться в мурашнику і для знаходження їжі в них є два шляхи, один з яких коротший по відношенню до іншого. На двох шляхах відсутній вміст феромонів, оскільки мурахи ще не починали свій шлях.

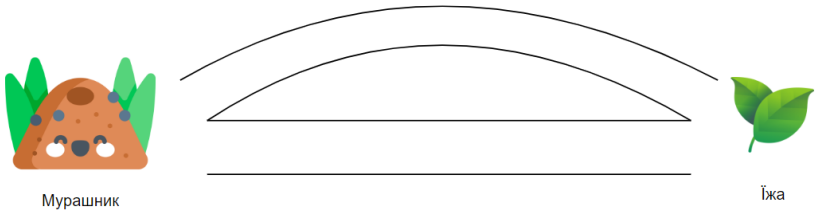


Рисунок 3.3 – Перший етап в поведінці мурах.

На другому етапі мурахи починають свій шлях з однаковою ймовірністю ($p = 0.5$) вибору маршруту. Прямий шлях до їжі коротший, тому мурахи будуть добиратись до їжі швидше по ньому. Другий етап зображений на рисунку 3.4.

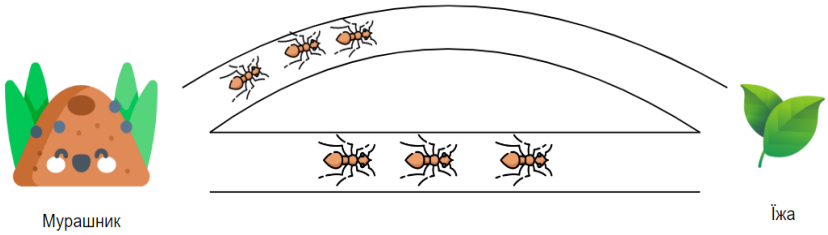


Рисунок 3.4 – Другий етап в поведінці мурах.

Третій етап зображено на рисунку 3.5. Мурахи ефективно досягають джерела їжі шляхом скорочення відстані, враховуючи коротший шлях. Далі вони будуть стикатись з аналогічною ситуацією вибору, але завдяки феромонному сліду, який вже пролягає по коротшому шляху, ймовірність вибору проходженням коротшим шляхом стає вищою.

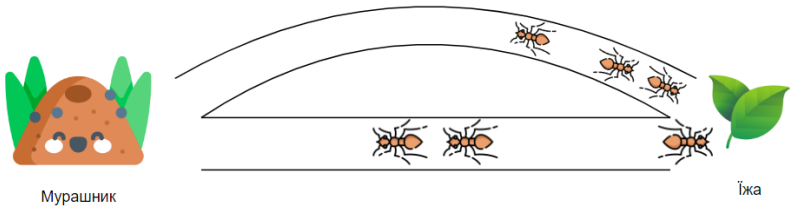


Рисунок 3.5 – Третій етап в поведінці мурах.

Більша кількість мурах повертається коротшим шляхом, що призводить до зростання концентрації феромонів. За допомогою процесу випаровування, концентрація феромонів на довшому шляху зменшується, що зменшує ймовірність вибору цього шляху на наступних етапах. В результаті, всі колонії поступово використовують коротший шлях з вищою ймовірністю. Таким чином, досягається оптимізація шляху [13]. Даний етап зображено на рисунку 3.6.

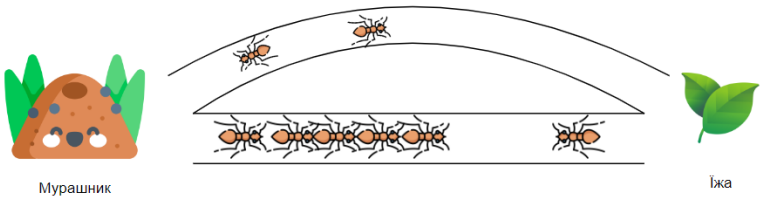


Рисунок 3.6 – Четвертий етап в поведінці мурах.

Структурна схема роботи алгоритму мурашиних колоній представлено на кресленику ІС93.090БАК.004 ДЗ.

## 3.7 Опис методу розв’язання

Вхідними параметрами алгоритму задаються:

– $q$ – кількість мурах;

– $I$ – кількість повторень з результатами без покращення;

– $\beta$ – коефіцієнт випаровування;

– $\gamma$ – ступінь значущості феромону;

– $\delta$ – ступінь значущості евристичної інформації.

Схема алгоритму мурашиних колоній

Крок 1 ПОКИ рекордний розв’язок не змінюється на I повторень повторювати наступні кроки

Крок 1.1 ПОКИ q мурах не відвідають всі об’єкти

Крок 1.1.1 Сформувати множину можливих підмаршрутів

Крок 1.1.2 Оновити та випарувати феромон

Крок 1.1.3 Оновити рекорд

Правило переходу до нового об’єкту

Ймовірність переходу мурах на кроці $I$ від поточного об’єкту $l \in V$, до нового об’єкту $m \in M_l$ розраховується за наступною формулою:

$$p_{lm}(I) = \frac{\alpha_{lm}(I)^{\gamma}\left(\frac{1}{d_{lm}}\right)^{\delta}}{\sum_{m \in M_l} \alpha_{lm}(I)^{\gamma}\left(\frac{1}{d_{lm}}\right)^{\delta}}, \quad (3.10)$$

де

$\alpha_{lm}(I)$ – рівень феромону на дузі $l - m$ на кроці $I$;

$\gamma, \delta$ – параметри алгоритму, які позначають ступінь значущості феромону та евристичної інформації відповідно, при цьому $\gamma, \delta > 0$.

Евристична інформація – це числове значення, яке не залежить від розв'язків, знайдених на попередніх кроках, і показує ступінь пріоритета включення в частину побудованого розв'язку нового ребра схеми моделі [8].

Оновлення феромону.

Так як немає оцінки значення цільової функції на основі тільки фрагмента розв'язку, тому зазвичай використовується покрокове оновлення феромону на дузі $(l, m)$ графа задачі за рахунок додавання фіксованої величини [8]:

$$\alpha_{lm}(I+1) = (1-\beta)\alpha_{lm}(I) + \Delta\alpha_{lm}, \quad (3.11)$$

де $\Delta\alpha_{lm}$ – сума всіх внесків, зроблених кожною мурашкою.

Внесок кожної мурахи $Q$, $Q = 1,2,\dots,q$ розраховується за наступною формулою:

$$\Delta\alpha_{lm} = \frac{1}{q}\sum_{Q=1}^{q}\Delta\alpha_{lm}{}^{Q}, \tag{3.12}$$

де $\Delta\alpha_{lm}{}^{Q} = \frac{1}{f(x^{Q})}$, якщо ребро (l, m) входить в розв’язок,

$f(x^{Q})$ – значення цільової функції задачі, отримане за допомогою розв'язку $x^{o}$знайденої мурахою.

3.7.1 Визначення параметрів алгоритму мурашиних колоній

Підбір параметрів для алгоритму мурашиних колоній в задачі маршрутизації БПЛА є важливим етапом розробки і впливає на його ефективність та якість результуючих маршрутів.

Спочатку розглянемо найбільш поширені методи підбору параметрів.

Ручне налаштування – це метод, при якому людина вручну вибирає значення параметрів для моделі на підставі свого досвіду, знань та інтуїції. Ручне налаштування може включати пробування різних значень параметрів та спостереження їх впливу на продуктивність моделі [21].

Випадковий пошук є одним із методів налаштування параметрів, де значення параметрів вибираються випадковим чином з певного діапазону. Цей метод не використовує жодних попередніх знань або структури даних для визначення оптимальних значень параметрів, але дозволяє провести пошук по широкому просторі можливих значень [21].

Пошук у випадковому порядку з розділенням навпіл – цей метод поєднує випадковий пошук параметрів з процедурою розділення навпіл для збільшення ефективності пошуку.

Процес починається з випадкового вибору початкових значень параметрів. Потім модель навчається з цими значеннями, і метрика продуктивності оцінюється на валідаційному наборі даних. Після цього відбувається процедура розділення навпіл, де кращі половини моделей за критерієм продуктивності зберігаються, а інші викидаються.

Залишені моделі розділяються на нові пари, і для кожної пари випадково вибираються нові значення параметрів, що дозволяє моделям експериментувати з новими комбінаціями. Цей процес повторюється до тих пір, поки не буде досягнуто заданої кількості ітерацій або не буде досягнуто достатньо точного розв'язку [22].

Пошук за сіткою – це метод, у якому визначається сітка параметрів, яка складається з заданих значень для кожного параметра. Далі, для кожної комбінації параметрів з сітки, модель тренується і оцінюється за допомогою певної метрики ефективності, наприклад, середня квадратична помилка.

Пошук за сіткою перебирає всі можливі комбінації параметрів на сітці, проводячи повний пошук у всьому просторі параметрів. Це дозволяє оцінити кожну комбінацію параметрів і знайти ту, яка дає найкращий результат [21].

Для підбору параметрів алгоритму мурашиних колоній, визначаємо, які параметри необхідно підбирати, а саме:

- $q$ – кількість мурах;
- $I$ – кількість повторень з результатами без покращення;
- $\beta$ – коефіцієнт випаровування;
- $\gamma$ – ступінь значущості феромону;
- $\delta$ – ступінь значущості евристичної інформації.

Спочатку підбираємо кількість мурах вручну, а саме аналізуючи цільову функцію на різних значеннях мурах на одній і тій ж задачі. Після підбору, було зафіксоване значення $q = 100$.

Відповідно до дослідження, проведеного Джеймсом Бергстрою та Йошуа Бенгіо в 2012 році, виявлено, що випадковий пошук параметрів насправді є більш ефективним в порівнянні з ручним пошуком або пошуком за допомогою сітки.

Замість систематичного тестування для охоплення "перспективних областей" у просторі параметрів, ефективніше випробовувати випадкові значення, які охоплюють весь простір параметрів [22]. Тому наступні чотири параметри, обираємо випадковим пошуком.

Встановлюємо діапазони можливих значень параметрів:

– $I \in [1,\ 50),\ I \in Z$;

– $\beta \in [0,\ 1),\ \beta \in R$;

– $\gamma \in [0, 5),\ \gamma \in R$;

– $\delta \in [0,\ 5),\ \delta \in R$.

На задачах розмірності 20 проводиться випадковий пошук параметрів. Система генерує 15 випадкових задач зі встановленою розмірністю. Дані задачі розв'язуються 100 разів, при цьому кожного разу використовуються нові значення параметрів, згенеровані системою. Для кожного набору параметрів розраховується середнє значення цільової функції, враховуючи результати розв'язання 15 задач. Далі вибирається найкраще середнє значення цільової функції (ЦФ), і параметри, що йому відповідають, вважаються ефективними. Результати проведення випадкового пошуку параметрів наведено в таблиці 3.1.

Таблиця 3.1 – Результати визначення параметрів

| Номер набору згенерованих параметрів | Середнє значення ЦФ | $q$ | $\beta$ | $\delta$ | $\gamma$ | $I$ |
|---|---|---|---|---|---|---|
| 1 | 74.56 | 100 | 0.28 | 0.17 | 1.58 | 24 |
| 2 | 76.56 | 100 | 0.55 | 0.35 | 2.79 | 40 |
| 3 | 77.65 | 100 | 0.65 | 0.42 | 4.42 | 13 |
| 4 | 79.07 | 100 | 0.81 | 4.07 | 4.72 | 2 |
| 5 | 78.31 | 100 | 0.14 | 2.87 | 3.25 | 13 |
| 6 | 77.28 | 100 | 0.19 | 3.45 | 4.79 | 12 |
| 7 | 76.23 | 100 | 0.47 | 2.45 | 1.95 | 30 |
| 8 | 76.13 | 100 | 0.23 | 3.86 | 1.35 | 33 |

| | | | | | | |
|---|---|---|---|---|---|---|
| 9 | 76.24 | 100 | 0.71 | 1.15 | 3.93 | 32 |
| 10 | 81.05 | 100 | 0.06 | 0.66 | 1.95 | 43 |
| 11 | 77.23 | 100 | 0.25 | 4.87 | 3.45 | 41 |
| 12 | 76.06 | 100 | 0.27 | 1.56 | 4.21 | 27 |
| 13 | 76.65 | 100 | 0.4 | 3.69 | 1.78 | 23 |
| 14 | 78.62 | 100 | 0.02 | 4.57 | 4.42 | 8 |
| 15 | 75.24 | 100 | 0.65 | 4.54 | 0.27 | 46 |
| 16 | 76.12 | 100 | 0.32 | 1.21 | 2.83 | 34 |
| 17 | 93.57 | 100 | 0.22 | 0.33 | 0.36 | 44 |
| 18 | 81.04 | 100 | 0.5 | 2.52 | 1.44 | 4 |
| 19 | 77.01 | 100 | 0.67 | 2.68 | 1.87 | 45 |
| 20 | 76.94 | 100 | 0.33 | 4.69 | 2.87 | 31 |
| 21 | 75.58 | 100 | 0.69 | 4.87 | 0.23 | 49 |
| 22 | 75.25 | 100 | 0.23 | 4.3 | 0.88 | 21 |
| 23 | 75.24 | 100 | 0.24 | 2.16 | 0.47 | 25 |
| 24 | 76.65 | 100 | 0.42 | 4.67 | 0.91 | 9 |
| 25 | 76.73 | 100 | 0.77 | 2.99 | 2.23 | 48 |
| 26 | 74.21 | 100 | 0.29 | 2.27 | 0.51 | 48 |
| 27 | 76.32 | 100 | 0.57 | 3.52 | 1.43 | 7 |
| 28 | 76.95 | 100 | 0.65 | 2.13 | 4.76 | 40 |
| 29 | 77.13 | 100 | 0.44 | 1.37 | 3.93 | 25 |
| 30 | 76.3 | 100 | 0.17 | 3.04 | 4.72 | 39 |
| 31 | 75.04 | 100 | 0.75 | 3.03 | 0.36 | 45 |
| 32 | 76.62 | 100 | 0.98 | 3.78 | 1.52 | 33 |
| 33 | 74.8 | 100 | 0.75 | 0.23 | 0.95 | 26 |
| 34 | 79.82 | 100 | 0.54 | 3.83 | 4.79 | 2 |
| 35 | 76 | 100 | 0.28 | 0.46 | 1.11 | 15 |
| 36 | 82.24 | 100 | 0.13 | 2.3 | 1.96 | 9 |
| 37 | 107.64 | 100 | 0.08 | 0.19 | 0.72 | 28 |

| | | | | | | |
|---|---|---|---|---|---|---|
| 38 | 75.51 | 100 | 0.87 | 1.97 | 1.47 | 13 |
| 39 | 74.72 | 100 | 0.9 | 1.13 | 1.27 | 14 |
| 40 | 74.83 | 100 | 0.74 | 2.69 | 0.4 | 48 |
| 41 | 77.13 | 100 | 0.42 | 4.6 | 4.9 | 34 |
| 42 | 74.34 | 100 | 0.43 | 2.35 | 0.64 | 30 |
| 43 | 115.22 | 100 | 0.4 | 0.19 | 0.11 | 17 |
| 44 | 100.94 | 100 | 0.92 | 0.8 | 0.04 | 17 |
| 45 | 87.46 | 100 | 0.81 | 1.99 | 0.08 | 4 |
| 46 | 78.36 | 100 | 0.09 | 2.77 | 0.77 | 24 |
| 47 | 75.52 | 100 | 0.8 | 4.2 | 0.28 | 44 |
| 48 | 75.38 | 100 | 0.47 | 1.45 | 1.83 | 49 |
| 49 | 77.56 | 100 | 0.82 | 1.2 | 4.65 | 8 |
| 50 | 75.96 | 100 | 0.06 | 3.66 | 3.25 | 49 |
| 51 | 76.51 | 100 | 0.37 | 2.95 | 2.18 | 44 |
| 52 | 83.62 | 100 | 0.92 | 1.93 | 0.02 | 21 |
| 53 | 113.9 | 100 | 0.03 | 0.37 | 0.53 | 36 |
| 54 | 79.74 | 100 | 0.05 | 1.77 | 4.68 | 36 |
| 55 | 77.2 | 100 | 0.12 | 4.81 | 4.97 | 10 |
| 56 | 123.45 | 100 | 0.85 | 0.11 | 1.14 | 1 |
| 57 | 74.89 | 100 | 0.62 | 0.69 | 1.48 | 6 |
| 58 | 82.35 | 100 | 0.03 | 2.49 | 2.01 | 13 |
| 59 | 76.78 | 100 | 0.77 | 4.62 | 0.6 | 7 |
| 60 | 76.15 | 100 | 0.09 | 0.67 | 2.04 | 31 |
| 61 | 74.97 | 100 | 0.98 | 2.64 | 1.23 | 48 |
| 62 | 76.96 | 100 | 0.57 | 4.17 | 1.44 | 25 |
| 63 | 76.28 | 100 | 0.09 | 4.09 | 4.26 | 28 |
| 64 | 93.1 | 100 | 0.52 | 0.75 | 0.22 | 20 |
| 65 | 77.94 | 100 | 0.68 | 4.17 | 1.91 | 6 |
| 66 | 77.1 | 100 | 0.3 | 4.49 | 4.06 | 31 |

| | | | | | | |
|---|---|---|---|---|---|---|
| 67 | 77.34 | 100 | 0.92 | 3.7 | 2.92 | 3 |
| 68 | 75.68 | 100 | 0.84 | 1.88 | 3.12 | 6 |
| 69 | 84.46 | 100 | 0.55 | 3.34 | 2.61 | 1 |
| 70 | 75.58 | 100 | 0.73 | 2.63 | 1.05 | 47 |
| 71 | 76.85 | 100 | 0.4 | 1.31 | 2.71 | 9 |
| 72 | 77.07 | 100 | 0.62 | 0.39 | 4.6 | 37 |
| 73 | 87.76 | 100 | 0.26 | 0.15 | 1.87 | 13 |
| 74 | 78.72 | 100 | 0.47 | 4.86 | 0.99 | 4 |
| 75 | 77.78 | 100 | 0.56 | 4.4 | 2.3 | 23 |
| 76 | 76.36 | 100 | 0.49 | 3.97 | 1.29 | 7 |
| 77 | 77.79 | 100 | 0.8 | 4.57 | 1.53 | 31 |
| 78 | 75.98 | 100 | 0.98 | 2.77 | 2.27 | 17 |
| 79 | 75.81 | 100 | 0.26 | 1.28 | 3.6 | 28 |
| 80 | 77.87 | 100 | 0.83 | 4.54 | 4.72 | 20 |
| 81 | 91.52 | 100 | 0.8 | 2.14 | 0.96 | 1 |
| 82 | 75.27 | 100 | 0.31 | 0.82 | 2.25 | 39 |
| 83 | 74.54 | 100 | 0.94 | 1.81 | 0.85 | 47 |
| 84 | 76.68 | 100 | 0.88 | 2.32 | 2.24 | 6 |
| 85 | 74.86 | 100 | 0.24 | 1.13 | 1.74 | 16 |
| 86 | 75.64 | 100 | 0.33 | 1.73 | 2.78 | 33 |
| 87 | 78.78 | 100 | 0.37 | 4.07 | 4.53 | 4 |
| 88 | 77.65 | 100 | 0.32 | 4.71 | 4.67 | 16 |
| 89 | 77.54 | 100 | 0.41 | 4.9 | 2.3 | 28 |
| 90 | 74.89 | 100 | 0.5 | 0.5 | 1.7 | 33 |
| 91 | 74.57 | 100 | 0.17 | 1.15 | 1.9 | 49 |
| 92 | 82.81 | 100 | 0.13 | 0.5 | 2.56 | 19 |
| 93 | 77.04 | 100 | 0.48 | 2.19 | 4.03 | 46 |
| 94 | 77.05 | 100 | 0.35 | 4.1 | 3.08 | 31 |
| 95 | 77.84 | 100 | 0.76 | 3.26 | 3.72 | 50 |

| | | | | | | |
|---|---|---|---|---|---|---|
| 96 | 86.21 | 100 | 0.06 | 2.96 | 2.65 | 1 |
| 97 | 75.9 | 100 | 0.42 | 1.17 | 2.86 | 42 |
| 98 | 76.87 | 100 | 0.95 | 4.26 | 1.43 | 37 |
| 99 | 77.75 | 100 | 0.83 | 2.79 | 4.58 | 23 |
| 100 | 76.47 | 100 | 0.64 | 2.02 | 0.47 | 8 |

Після підбору отримуємо наступні діапазони значення параметрів:

– $I \in [47, 49]$;

– $\beta \in [0.25, 0.3]$;

– $\gamma \in [0.45, 0.55]$;

– $\delta \in [2.25, 2.3]$.

3.7.1.1 Результати аналізу впливу кількості прогонів

Для перевірки роботи алгоритму з підібраними параметрами проведемо тестування. В таблиці 3.2 наведені результати роботи алгоритму на трьох різних значеннях прогонів, а саме: 10, 25, 50. На кожному зі значень прогонів генеруються задачі п’яти різних розмірностей, а саме: 10, 15, 20, 25, 30. Для кожної задачі рахується краще значення цільової функцій, середнє значення цільової функції та середнє квадратичне відхилення. На результатах цих значень проводиться аналіз роботи алгоритму.

Таблиця 3.2 – Результати тестування роботи алгоритму

| Кількість прогонів | Розмірність задачі | Краще значення цільової функції | Середнє значення цільової функції | Середньоквадратичне відхилення |
|---|---|---|---|---|
| 10 | 10 | 40.54 | 51.49 | 6.28 |
| | 15 | 44.52 | 64.45 | 7.54 |

| | | | | |
|---|---|---|---|---|
| | 20 | 71.82 | 77.39 | 4.16 |
| | 25 | 80.72 | 88.49 | 6.13 |
| | 30 | 87.1 | 99.19 | 7.96 |
| 25 | 10 | 37.33 | 52.44 | 7.9 |
| | 15 | 48.03 | 65.77 | 7.52 |
| | 20 | 61.07 | 74.72 | 6.7 |
| | 25 | 81.11 | 89.92 | 6.88 |
| | 30 | 83.62 | 96.97 | 9.44 |
| 50 | 10 | 35.14 | 53.56 | 7.57 |
| | 15 | 52.14 | 64.7 | 6.61 |
| | 20 | 61.86 | 75.5 | 6.02 |
| | 25 | 69.04 | 86.29 | 7.49 |
| | 30 | 81.74 | 101.6 | 9.29 |

Для кращого аналізу роботи алгоритму зображено графічно залежність кращих значень цільової функції (ЦФ) від розмірності задачі. На рисунку 3.8 можна побачити вплив кількості прогонів на точність результату роботи. Тобто, зі збільшенням прогонів значення цільової функції зменшується, що свідчить про ефективну роботу. Значення середнього значення цільових функцій та середнє квадратичне відхилення не залежить від кількості прогонів, проте зі збільшенням розмірності збільшується, це свідчить про коректність роботи алгоритму.

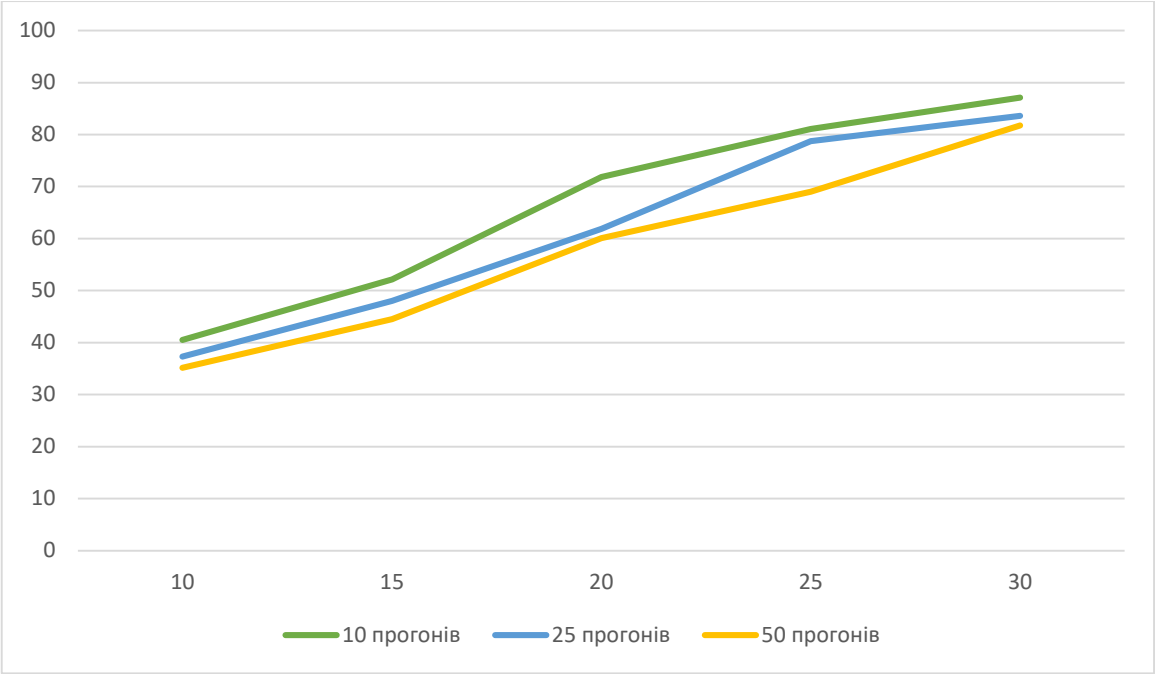


Рисунок 3.8 – Вплив кількості прогонів на точність результату роботи алгоритму

Висновки до розділу

У даному розділі були розглянуті основні аспекти, пов'язані з оптимізацією маршрутів транспортних засобів. Проаналізовано класифікацію задач, що надає загальний огляд та систематизацію цих задач.

Була сформульована змістовна постановка задачі. На основі змістовної постановки була побудована математична постановка задачі.

Проведений аналіз методів розв'язання задач маршрутизації, включаючи розгляд точних та наближених методів. Розглянуто евристичні та метаевристичні методи розв’язання даних задач. Було висвітлено різновиди мурашиних алгоритмів, які є одними з популярних метаевристичних методів розв'язання таких задач.

Обґрунтовано метод розв’язання та детально описано схему розв’язання обраного алгоритму.

Проведено підбір параметрів алгоритму, а саме: кількості мурах, кількості повторень з результатами без покращення, коефіцієнт випаровування, ступінь

значущості феромону та ступінь значущості евристичної інформації. Після проведення випадковим та ручним пошуком параметрів на основі 100 наборів параметрів, які наведені у таблиці 3.1, було отримано наступні результати:

- $q = 100$;
- $I \in [47, 49]$;
- $\beta \in [0.25, 0.3]$;
- $\gamma \in [0.45, 0.55]$;
- $\delta \in [2.25, 2.3]$.

Після визначення параметрів було проведено тестування роботи алгоритму та проаналізовано вплив кількості прогонів на точність результату роботи алгоритму. Результати роботи мурашиного алгоритму свідчать про ефективність запропонованого методу розв'язання задач маршрутизації.

# 4 ПРОГРАМНЕ ТА ТЕХНІЧНЕ ЗАБЕЗПЕЧЕННЯ

## 4.1 Засоби розробки системи

### 4.1.1 Програмне середовище розробки системи

Середовище розробки програмного забезпечення – це набір апаратних та програмних ресурсів, що використовуються розробником для створення програмних систем. Це місце, де розробники проводять більшу частину своєї роботи над програмою. Зазвичай середовище розробки налаштовується на локальних комп'ютерах, а роботу полегшує використання репозиторію Git. Оператори та клієнти не можуть отримати доступ до роботи, проведеної в середовищі розробки, поки хтось не поділиться необхідною інформацією з ними[14].

Було обрано програмне середовище для розробки інформаційної системи – Visual Studio Code.

Visual Studio Code – це безкоштовний редактор вихідного коду, який поєднує в собі легкість та потужність. Він доступний для різних операційних систем, включаючи Windows, macOS, Linux і Raspberry Pi, і може працювати як на робочому столі, так і в Інтернеті.

Visual Studio Code надає вбудовану підтримку для JavaScript, TypeScript і Node.js, а також має розширену екосистему, що дозволяє розширити підтримку інших мов програмування, таких як C++, C#, Java, Python, PHP і Go. Він також підтримує середовища виконання, такі як .NET і Unity, і хмарні платформи, включаючи Amazon Web Services, Microsoft Azure і Google Cloud Platform.

Серед основних функцій Visual Studio Code можна виділити розширений IntelliSense для автоматичного заповнення коду, графічний дебагер, можливість редагування кількома курсорами, автопідказки параметрів і багато інших потужних функцій. Редактор також пропонує швидку навігацію та можливості рефакторингу коду, а також вбудовану підтримку системи контролю версій Git. Багато з цих функцій були адаптовані з технологій Visual Studio.

Visual Studio Code розроблено з використанням технологій, таких як оболонка Electron, Node.js, TypeScript і протокол Language Server Protocol. Він постійно оновлюється, а багато розширень також часто отримують оновлення. Рівень підтримки різних мов програмування та їх розширень може варіюватися, від простого підсвічування синтаксису до функцій налагодження та рефакторингу. У випадку, коли мовний сервер не доступний, ви можете додати базову підтримку для своєї улюбленої мови за допомогою розфарбовувачів TextMate [15].

### 4.1.2 Мова розробки системи

Мова програмування системи – це система правил, символів, синтаксису та семантики, яка використовується для написання програмного коду, що контролює поведінку системи. Це інструмент, що дозволяє розробникам спілкуватися з комп'ютером та описувати логіку та функціональність програм або системи.

Мови програмування системи можуть мати різні рівні абстракції та призначення. Деякі з них спеціалізовані для конкретних областей застосування, наприклад, для розробки веб-сайтів, мобільних додатків або наукових обчислень. Інші мови є загальнопризначеними і можуть використовуватися для розробки різних типів програм.

Мови програмування системи надають розробникам можливість виражати свої ідеї та алгоритми у вигляді програмного коду, який комп'ютер може розуміти та виконувати. Вони визначають правила для створення змінних, функцій, класів, структур даних та інших елементів програми. Крім того, мови програмування системи можуть включати бібліотеки та фреймворки, які надають готові компоненти та інструменти для полегшення розробки .

Було обрано мовою розробки інформаційної системи – TypeScript.

TypeScript – це об'єктно-орієнтована мова програмування з відкритим вихідним кодом, заснована на JavaScript, розроблена і підтримувана компанією Microsoft. На практиці це надмножина потужної мови JavaScript, що складається

з усіх її сегментів. Іншими словам, весь код JavaScript є дійсним кодом TypeScript, а це означає, що можна використовувати TypeScript для додавання будь-якої розширеної функціональності, якщо вона реалізована на JavaScript.

TypeScript був створений Андерсом Гейлсбергом у Microsoft у 2010 році, і його перша версія була випущена у 2012 році під назвою TypeScript 0.8.

Незважаючи на те, що TypeScript отримав підтримку від багатьох розробників програмного забезпечення, багато спільноти розробників JavaScript ще не повністю перейшли на нього, оскільки відсутність інтегрованого середовища розробки (IDE) є його найбільшою недоліком.

Завдяки зростанню спільноти розробників, регулярно випускаються нові версії TypeScript з розширеними можливостями, що полегшує початок роботи з ним.

TypeScript надає підтримку для різних концепцій, таких як інтерфейси, загальні коди, успадкування і модифікатори доступу до методів. Використання інтерфейсів дозволяє визначити контракт, за допомогою Generics можна забезпечити перевірку типів під час компіляції. Успадкування дозволяє новим об'єктам отримати властивості від існуючих об'єктів. Модифікатори доступу до методів контролюють доступність членів класу.

У TypeScript існують два типи модифікаторів доступу: публічні і приватні. За замовчуванням, члени класу вважаються публічними, але це можна змінити відповідно до потреб [16]. https://invedus.com/blog/what-is-typescript-definition-history-features-and-uses-of-typescript/

### 4.1.3 Бібліотеки використані для розробки системи

Бібліотеки – це збірки коду, які містять готові функції, класи, модулі та інші компоненти, які можна використовувати у розробці програмного забезпечення. Вони створюються з метою повторного використання коду та надання готових рішень для різних задач.

Бібліотеки надають додаткові функції та компоненти, які розширюють можливості мови програмування або платформи. Вони дозволяють виконувати складні операції швидше та ефективніше.

Використання бібліотек дозволяє розробникам фокусуватися на основних завданнях і не витрачати час на реалізацію загальних функцій. Готові рішення забезпечують швидкий старт і полегшують розробку складних систем.

Багато бібліотек пропонують оптимізовані алгоритми та структури даних, що дозволяє ефективно використовувати обчислювальні та пам'ятові ресурси. Це особливо корисно при роботі з великими обсягами даних чи вимогливими обчисленнями.

MobX – це бібліотека управління станом для JavaScript та TypeScript додатків, що має велику популярність у розробці інтерфейсів. Вона забезпечує простий і масштабований спосіб керування станом програми та його синхронізацію з користувацьким інтерфейсом. MobX базується на принципах реактивного програмування, коли зміни в стані автоматично викликають оновлення в відповідних частинах програми.

За допомогою MobX можна оголошувати спостережувані змінні стану, які автоматично відстежуються на зміни. Коли відбувається зміна, MobX ефективно оновлює відповідні компоненти або обчислення, що забезпечує синхронізацію користувацького інтерфейсу з актуальним станом. Також MobX підтримує обчислені значення, які автоматично оновлюються при зміні залежностей.

Одним з головних переваг MobX є його простота використання. MobX можна легко інтегрувати в існуючі проєкти без великих зусиль або складних налаштувань. Крім того, MobX дозволяє писати декларативний і лаконічний код, що полегшує розуміння та підтримку стану вашої програми.

MobX часто використовується разом з популярними фреймворками, такими як React або Angular. Він надає гнучке та ефективне рішення для керування складним станом і дозволяє створювати реактивні додатки [17].

React, також відомий як ReactJS або React.js, представляє собою відому та безкоштовну бібліотеку JavaScript з відкритим вихідним кодом, яка

застосовується для створення компонентів інтерфейсу оператора та користувацьких інтерфейсів. Ця інтерфейсна бібліотека була розроблена компанією Facebook та продовжується підтримуватися ними разом із розростаючою спільнотою компаній та окремих розробників. React часто використовується як основа для розробки односторінкових додатків (SPA).

У 2011 році, коли рекламний додаток Facebook почав набувати популярності, його код почав отримувати безліч нових оновлень та нових членів команди, що з часом призвело до їх перевантаження та ускладнило управління ними. Спробуючи вирішити цю проблему, Джордан Волке створив прототип FaxJS, який зробив цей процес більш ефективним.

У 2012 році Facebook придбав Instagram з метою отримати доступ до їхніх нових технологій. Це створило певний тиск на материнську компанію, щоб відокремити React від Facebook та зробити його відкритим вихідним кодом. На кінець травня 2013 року Джордан Волке нарешті презентував те, що тепер відоме як "React" і відкрив його вихідний код. Протягом решти року React став предметом використання та експериментів у всій галузі. У 2014 році React почав набирати популярність, оскільки були додані нові важливі функції, такі як інструменти розробника React, як розширення інструментів розробника Chrome. Також у 2014 році був представлений React Hot Loader, плагін, що дозволяє компонентам React перезавантажуватись без втрати стану.

Ключові особливості React:

– архітектура на основі компонентів: React сприяє модульному підходу до побудови інтерфейсів, розбиваючи їх на перевикористовувані компоненти; кожен компонент керує власним станом і може бути поєднаний разом для створення складних інтерфейсів оператора;

– віртуальний DOM: React використовує віртуальне представлення реального DOM (Document Object Model) для ефективного оновлення та відображення компонентів; замість прямого втручання у DOM, React порівнює віртуальний DOM з реальним DOM і застосовує лише необхідні оновлення, що покращує продуктивність;

– декларативний синтаксис: React використовує декларативний синтаксис, що дозволяє розробникам описувати, як повинен виглядати інтерфейс на основі поточного стану додатка; це спрощує розуміння та підтримку коду, оскільки розробники можуть фокусуватись на тому, що має бути відображено в інтерфейсі, а не на тому, як цього досягти;

– односторонній потік даних: React використовує односторонній потік даних, де дані передаються від батьківських компонентів до дочірніх компонентів через властивості (props); це допомагає зберігати передбачуваний стан та спрощує відстеження та відлагодження змін даних [18].

Бібліотеки, використані для розробки клієнтської частини:

– d3 (версія 7.8.4) – це бібліотека для маніпулювання даними та створення візуалізацій; вона надає потужні інструменти для роботи з даними, маніпулювання DOM та створення складних графіків, діаграм та інших візуальних елементів;

– mobx (версія 6.9.0) – це станова управляюча бібліотека, яка допомагає організувати та відстежувати зміни в стані додатка; вона забезпечує простий та зрозумілий спосіб оголошення та використання спостережуваних об'єктів, реагуючи на зміни в цих об'єктах та автоматично оновлюючи відповідні компоненти;

– mobx-react-lite (версія 3.4.3)– це версія бібліотеки mobx-react, яка надає прив'язку між спостережуваними об'єктами mobx і компонентами React; вона дозволяє реагувати на зміни в спостережуваних об'єктах та автоматично оновлювати відповідні компоненти.

– ramda (версія 0.29.0) – це функціональна бібліотека для роботи з даними і функціями; вона надає набір утиліт для зручного трансформування та обробки даних, зокрема функції вищих порядків, композицію функцій.

– react (версія 18.2.0) – це основна бібліотека для розробки інтерфейсів оператора в React-додатках; вона надає можливості для створення компонентів, керування станом додатка та взаємодії з DOM.

– react-dom (версія 18.2.0) – це пакет, який містить утиліти для взаємодії React з DOM; він дозволяє рендерити React-компоненти в реальний DOM, керувати подіями та оновлювати інтерфейс на основі змін стану.

– react-router-dom (версія 6.11.0) – це набір розширень для маршрутизації в React-додатках; він дозволяє організувати навігацію між різними сторінками та компонентами, забезпечуючи зміну вмісту в залежності від поточного URL.

– react-scripts (версія 5.0.1) – це набір скриптів для розробки та побудови React-додатків; він автоматизує процеси, такі як розробка в режимі розробки, збірка продукційної версії та інші рутинні завдання, що пов'язані з розробкою React-додатків.

– typescript (версія 4.4.2) – це мова програмування, яка розширює синтаксис JavaScript.

## 4.2 Вимоги до технічного забезпечення

Для того щоб інформаційна система працювала комп’ютер оператора повинен відповідати наступним вимогам:

– ємність оперативної пам’яті повинен бути 2 ГБ або більше;

– об’єм твердотільного накопичувача або жорсткого диску повина бути 128 ГБ або більше;

– процесор Core i3-5xx, AMD A4/A6;

– необхідно встановити останню версію браузера.

## 4.3 Архітектура програмного забезпечення

### 4.3.1 Діаграма компонентів

У складі інформаційної системи присутні три складові: клієнтська складова, серверна складова і база даних.

У дипломній роботі була розроблена основною складовою саме клієнтська частина. Розглянемо компоненти клієнтської частини. Visualization component

відповідає за візуалізацію маршрутів на користувацькому інтерфейсі. SolverService відповідає за взаємодію з сервером шляхом надсилання запитів до контролера. Він виконує роль посередника між клієнтом і сервером, передаючи дані і отримуючи відповіді. SolutionsStore виконує функцію зберігання і оновлення рішень. Він зберігає дані про розв'язки, отримані від сервера, і забезпечує доступ до цих даних для подальшого використання в системі. Solver UI component відповідає за надання користувацького інтерфейсу, через який оператор може взаємодіяти з функціоналом розв'язування. Даний компонент дозволяє оператору вводити параметри задачі, якщо увійти у систему через оператора, або параметри дослідження, якщо увійти у систему через експериментатора.

Діаграма компонентів представлена на кресленику ІС93.090БАК.004 Д4.

### 4.3.2 Опис класів, функцій та методів програмного забезпечення

Клієнтська частина система – це та частина програмного продукту, яка взаємодіє з оператором. Вона забезпечує інтерфейс і способи взаємодії з системою, наприклад, веб-інтерфейс. Клієнтська частина включає різні компоненти, такі як стори, сервіси та компоненти бізнес-логіки(BLoC).

BLoC ( англ. Business Logic Component) містить стан та хендлери сторінок.

Хендлери у BLoC призначені для опрацювання подій, виконання певних дій та зміни стану. Вони обробляють вхідні події або запити і виконують необхідні дії, такі як обробка даних, взаємодія з іншими компонентами системи, проведення валідації, збереження даних. Хендлери відповідають за виконання бізнес-логіки з урахуванням отриманих вхідних даних і поточного стану.

Стан в BLoC відображає поточну інформацію та значення компонента. Він включає всі необхідні дані, які потрібні для виконання бізнес-логіки, а саме: значення полів, властивостей, списки даних тощо. Стан може зазнавати змін під час виконання операцій або відповідно до зовнішніх подій [19].

Сервіси взаємодіють з сервером, роблять запити і отримують оновлені дані, які потім передаються сторам. Одна з головних функцій сервісів – забезпечити оновлення сторів з актуальними даними з сервера. Це включає в себе отримання даних, обробку відповідей сервера і передачу цих даних до відповідних сторів [20].

Стори, з свого боку, відповідають за збереження цих отриманих даних і надання їх компонентам бізнес-логіки. Вони діють як контейнери для даних і забезпечують доступ до них з інших компонентів системи. Стори зберігають дані у своєму внутрішньому стані і надають методи для отримання цих даних та виконання операцій над ними. Це дозволяє компонентам бізнес-логіки працювати з актуальними даними і виконувати свої функції на основі цих даних [20].

У таблиці 4.1 описані методи, які використовує SolverService. Це сервіс, який відповідає за надсилання запитів до контроллера “Solver” на сервері і оновлення стору SolutionsStore.

Таблиця 4.1 – Опис методів програмної реалізації SolverService

| Метод | Вхідні параметри | Вихідні параметри | Опис |
|---|---|---|---|
| calculateRoute | data – об’єкт, який містить масив точок, які потрібно облетіти, стартову базу, другу базу, час обслуговування між польотами, максимальний польотний час без обслуговування і швидкість польоту | - | Даний метод робить запит до ендпоінту сервера, який відповідає за вирішення заданої задачі. Після цього він оновлює в сторі SolutionsStore останній отриманий результат |
| performExperiment | data – об’єкт, який містить назву алгоритму, яким потрібно вирішити | - | Даний метод робить запит до ендпоінту, що відповідає за |

| | | | |
|---|---|---|---|
| | задачу, кількість запусків алгоритму і розмірність задачі (кількість випадково згенерованих точок, що потрібно облетіти) | | проведення експерименту. Після цього він оновлює дані останнього проведеного експерименту в сторі SolutionsStore |
| downloadLastResult | - | Blob | Робить запит до на скачування останнього результату та повертає Blob у BLoC |

SolutionsStore – це стор, який зберігає дані останньої вирішеної заданої оператором задачі та масив результатів останнього експерименту. Не має методів

SolverScreenBLoC – це BLoC, який відповідає за управління сторінкою вирішення задачі, що була задана оператором. Він містить в собі стан форми, який зберігає дані, введені оператором на цій сторінці. Крім того, він має ряд методів, описаних у таблиці 4.2, які виконують різні операції.

Цей BLoC також надає ряд геттерів, які обчислюють дані, необхідні для відображення на екрані. Ці геттери отримують доступ до поточного стану форми та інших даних, які зберігаються у BLoC, і обчислюють необхідні значення для відображення оператору.

Таблиця 4.2 – Опис методів програмної реалізації SolverScreenBLoC

| Метод | Вхідні параметри | Вихідні параметри | Опис |
|---|---|---|---|
| createPointSetter | index – індекс точки у масиві точок fieldName – назва поля точки, для | Повертає функцію, яка є хендлером заданого | Даний метод створює і повертає хендлер заданого поля для і-ої точки |

| | якого потрібно створити хендлер | поля для і-ої точки | |
|---|---|---|---|
| addPoint | - | - | Даний метод додає нову точку із пустими полями до стану форми |
| setStartBaseLat | value – нове значення поля форми | - | Оновлює значення широти стартової бази |
| setStartBaseLng | value – нове значення поля форми | - | Оновлює значення довготи стартової бази |
| setAnotherBaseLat | value – нове значення поля форми | - | Оновлює значення широти другої бази |
| setAnotherBaseLng | value – нове значення поля форми | - | Оновлює значення довготи другої бази |
| setChargeTime | value – нове значення поля форми | - | Оновлює значення часу обслуговування |
| setMaxFlightTime | value – нове значення поля форми | - | Оновлює значення максимального польотного часу без обслуговування |
| setSpeed | value – нове значення поля форми | - | Оновлює значення швидкості польоту |
| submitForm | - | - | Приводить типи даних до необхідної форми і викликає метод сервісу SolverService, що відповідає за запит |

| | | | |
|---|---|---|---|
| | | | до відповідного ендпоінту сервера |
| downloadLastResult | - | - | Завантажує останній результат оператора у його файлову систему |

Програмна реалізація алгоритму мурашиних колоній використовує один клас під назвою PheromoneState. Цей клас призначений для зберігання інформації про поточний стан феромонів та надає ряд методів для маніпулювання цим станом. Конструктор класу PheromoneState приймає наступні параметри:

– points – масив об’єктів, які потрібно облетіти; цей масив містить координати індивідуальних точок або місць, які мурахи повинні відвідати під час своїх маршрутів.

– bases – масив станцій для перезарядки; станції визначають початкові та кінцеві точки маршрутів мурах.

– evaporationRate – швидкість випаровування феромонів.; це числове значення від 0 до 1, яке визначає, як швидко феромони випаровуються з маршрутів мурах; більше значення означає швидше випаровування.

– initialPheromoneVal – початковий рівень феромонів; це числове значення, яке встановлює початкову концентрацію феромонів на всіх маршрутах мурах.

У таблиці 4.3 описані методи програмної реалізації класу PheromoneState.

Таблиця 4.3 – опис методів програмної реалізації класу PheromoneState

| Метод | Вхідні параметри | Вихідні параметри | Опис |
|---|---|---|---|
| get | point1 – перша точка<br>point2 – друга точка | Число, яке відображає кількість феромонів на ребрі графа, що | Даний метод зчитує з хеш-таблиці і повертає кількість феромонів на ребрі графа, що з’єднує дві задані точки |

| | | | |
|---|---|---|---|
| | | з’єднує дві задані точки | |
| evaporateAll | - | - | Даний метод меншує феромони, що відповідають усім ребрам графа між усіма точками на величину, що залежить від швидкості випаровування феромонів |
| add | point1 – перша точка<br>point2 – друга точка<br>estimation – значення функції оцінки маршруту | - | Даний метод збільшує кількість феромонів на на ребрі графу, що з’єднує дві задані точки на величину, що зележить від значення функції оцінки |

У таблиці 4.4 описано функції, які використовуються для алгоритму мурашиних колоній

Таблиця 4.4 – Опис функцій програмної реалізації алгоритму мурашиних колоній

| Функція | Вхідні параметри | Вихідні параметри | Опис |
|---|---|---|---|
| createAntColonySolver | params – об’єкт, що містить максимальну кількість ітерацій без покращення, кількість мурах, швидкість випаровування феромонів, коефіцієнт важливості феромонів і коефіцієнт важливості евристичних даних | Повертає функцію, яка і є розв’язувачем задачі | Дана функція створює розв’язувач задачі з переданими параметрами |

| | | | |
|---|---|---|---|
| Функція (розв’язувач), яку повернув createAntColonySolver | pointsToObserve – масив точок, які потрібно облетіти<br>startBase – початкова база<br>anotherBase – друга база<br>chargeTime – час обслуговування між польотами (у хвилинах)<br>maxFlightTime – максимальний польотний час без обслуговування (у хвилинах)<br>speed – швидкість польоту | Повертає об’єкт, який містить поля:<br>route – побудований маршрут<br>fitness – значення цільової функції | Дана функція отримує вхідні дані, розв’язую задачу мурашиним алгоритмом і повертає результат розв’язання |
| createCalculateProbability | pherState – об’єкт класу PheromoneState, що мітсить дані про феромони<br>pheromoneImportance – коефіцієнт важливості феромонів,<br>heurInfoImportance – коефіцієнт важливості евристичних даних<br>calTimeBetweenTwoPoints – функція, яка обраховує час польоту між двома точками | Повертає функцію, яка обчислює ймовірність переходу до заданої наступної точки маршруту | Дана функція створює і повертає функцію, яка обчислює ймовірність переходу до заданої наступної точки маршруту |
| Функція, яку повернув createCalculateProbability | currentRoute – поточний (незавершений) маршрут<br>nextPoint – потенційна наступна точка | Число, яке відображає ймовірність перезоду до заданої точки | Дана функція обчислює ймовірність переходу до заданої |

| | | | |
|---|---|---|---|
| | | | наступної точки маршруту |
| getAvailableMoves | currentPoint – поточна точка<br>restOfPointToObserve – масив точок, які ще не були відвідані<br>bases – масив баз<br>restOfFlightTime – залишок часу польоту без обслуговування<br>speed – швидкість польоту | Масив точок, у які можна перейти з заданої точки з урахуванням обмежень задачі | Дана функція знаходить і повертає масив точок або баз, у які можна перейти з поточної точки. При цьому ці точки є таким при переході в які можливо буде повернутися на одну із баз без перевищення залишку польотного часу |

Висновки до розділу

У даному розділі було детально розглянуте програмне та технічне забезпечення системи. Були описані використані технології розробки, а саме: програмне середовище, в якому було розроблене програмне забезпечення, мова розробки, а також фреймворки та бібліотеки, які використовувалися для розробки клієнтської частини системи. Аргументовано вибір описаних технологій.

Також були сформульовані вимоги до технічного забезпечення операторів системи, враховуючи їх потреби та вимоги. За допомогою діаграми компонентів

та діаграми послідовності була детально описана архітектура системи зі сторони клієнта. Це допомагає зрозуміти взаємодію між компонентами та логіку взаємодії в рамках системи.

Крім того, було описано класи, функції та методи конкретних функціональних можливостей, які повинна виконувати клієнтська частина системи.

# 5 ТЕХНОЛОГІЧНИЙ РОЗДІЛ

## 5.1 Керівництво оператора

Розглянемо головну сторінку оператора, яку він бачить після входу в систему. На рисунку 5.1 зображено початкова сторінка системи.

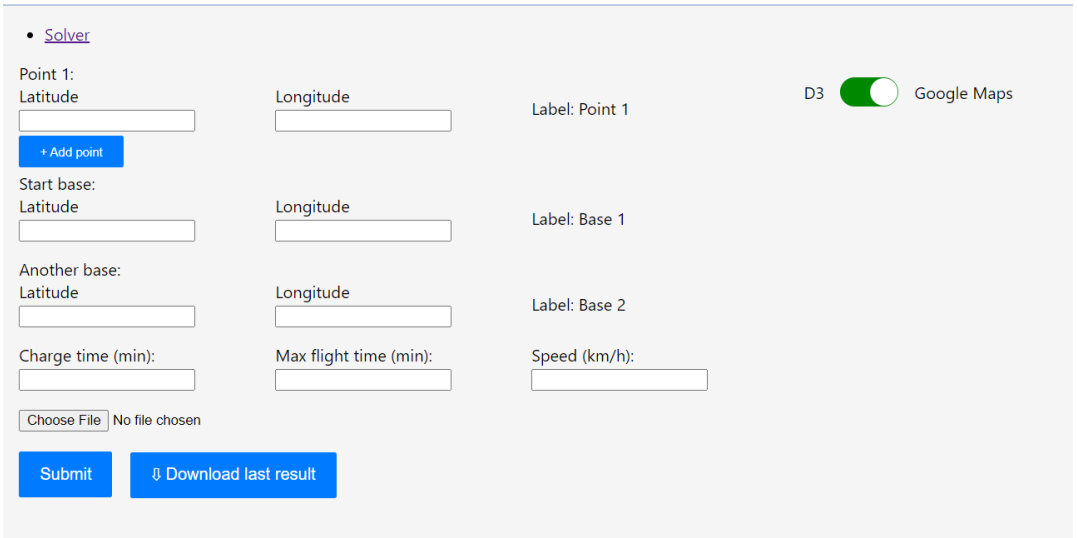


Рисунок 5.1 – Початкова сторінка оператора

Для того, щоб розв’язати задачу, нам потрібно ввести дані, або завантажити файл з даними.

Розглянемо варіант завантаження файлу з даними. Нам необхідно натиснути на кнопку «Choose file» та обрати JSON – файл. На рисунку 5.2 зображено приклад JSON-файла, для зчитування даних, а на рисунку 5.3 зображено стан сторінки після завантаження файлу.


```
{"points": [{"id": 0, "label": "Point 1","isStartBase": false, "isBase": false, "type": "point", "lat": 50.40918486208558, "lng": 30.4715177701855},
{"id": 1, "label": "Point 2", "isStartBase": false, "isBase": false, "type": "point", "lat": 50.41089745330126, "lng": 30.469492553258 05},
{"id": 2, "label": "Point 3", "isStartBase": false, "isBase": false, "type": "point", "lat": 50.41305671921 5206, "lng": 30.470894626515513},
{"id": 3, "label": "Point 4", "isStartBase": false, "isBase": false, "type": "point", "lat": 50.41328008593458, "lng": 30.474166130782937},
{"id": 4, "label": "Point 5", "isStartBase": false, "isBase": false, "type": "point", "lat": 50.41238661273848, "lng": 30.476541866024753},
{"id": 5, "label": "Point 6", "isStartBase": false, "isBase": false, "type": "point", "lat": 50.41042587638858, "lng": 30.477398688570982},
{"id": 6, "label": "Point 7", "isStartBase": false, "isBase": false, "type": "point", "lat": 50.40762113778469, "lng": 30.481643854822 76},
{"id": 7, "label": "Point 8", "isStartBase": false, "isBase": false, "type": "point", "lat": 50.408638805495045, "lng": 30.484331161899572},
{"id": 8, "label": "Point 9", "isStartBase": false, "isBase": false, "type": "point", "lat": 50.40816720610272, "lng": 30.487096361935137},
{"id": 9, "label": "Point 10", "isStartBase": false, "isBase": false, "type": "point", "lat": 50.40590842815011, "lng": 30.48873211406885},
{"id": 10, "label": "Point 11", "isStartBase": false, "isBase": false, "type": "point", "lat": 50.40414601013882, "lng": 30.486239539388908},
{"id": 11, "label": "Point 12", "isStartBase": false, "isBase": false, "type": "point", "lat": 50.40384813019712, "lng": 30.482111212575248},
{"id": 12, "label": "Point 13", "isStartBase": false, "isBase": false, "type": "point", "lat": 50.40501481594123, "lng": 30.479969156209677},
{"id": 13, "label": "Point 14", "isStartBase": false, "isBase": false, "type": "point", "lat": 50.405982895073585, "lng": 30.478995494225323}],
"startBase": {"id": 14, "label": "Base 1", "isStartBase": true, "isBase": true, "type": "base", "lat": 50.410872633580745, "lng": 30.473387201195454},
"anotherBase": {"id": 15, "label": "Base 2", "isStartBase": false, "isBase": true, "type": "base", "lat": 50.406082184122845, "lng": 30.48491535909018},
"speed": 30,
"maxFlightTime": 2,
"chargeTime": 1
}
```

Рисунок 5.2 – Приклад JSON-файлу

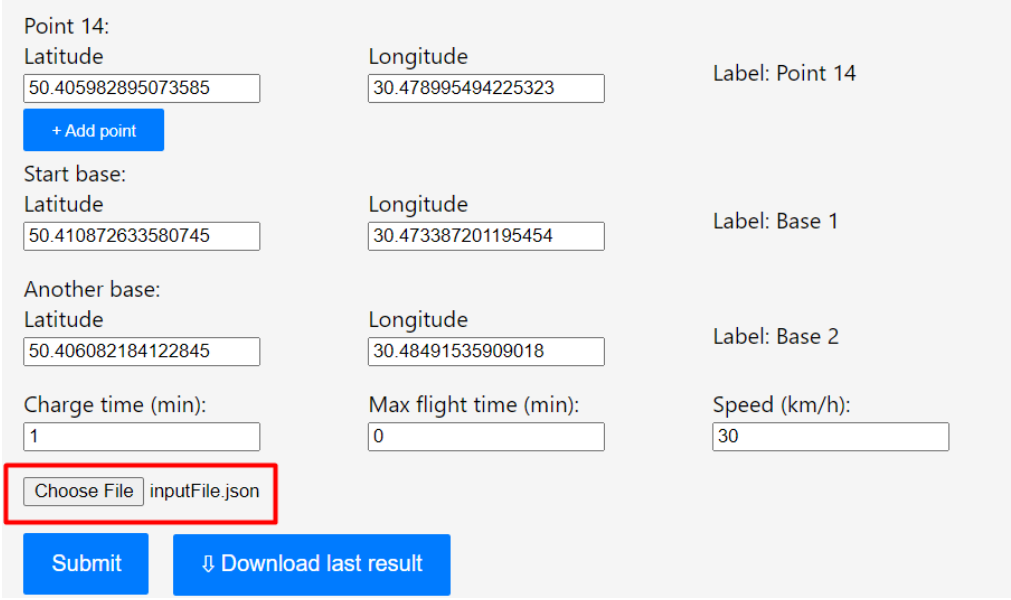


Рисунок 5.3 – Сторінка після завантаження файлу

Також для введення вручну даних, необхідно заповнити всі поля, а саме: дані про безпілотний літальний апарат (середня швидкість переміщення, часовий обмежений потенціал, час за який необхідно перезарядити БПЛА), дані про географічне положення об’єктів та станцій для перезарядки. На рисунку 5.4 зображений приклад із даними у заповнених полях.

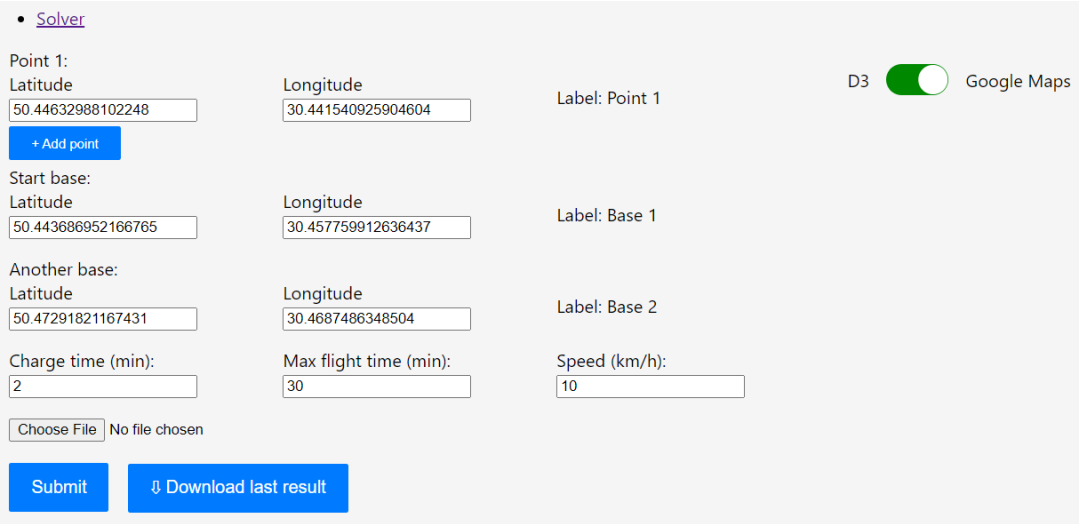


Рисунок 5.4 – Сторінка оператора із заповненими полями

При некоректному введені даних, оператор отримує інформацію про це, як зображено на рисунку 5.5.

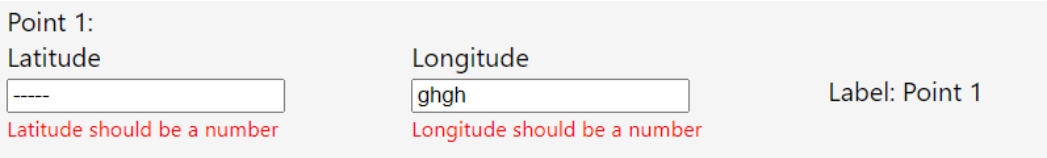


Рисунок 5.5 – Сторінка оператора з валідацією полей

Якщо оператор хоче додати нового об’єкта для відвідання, він може натиснути кнопку «Add point». На рисунку 5.6 зображена сторінка після додавання нового об’єкту.

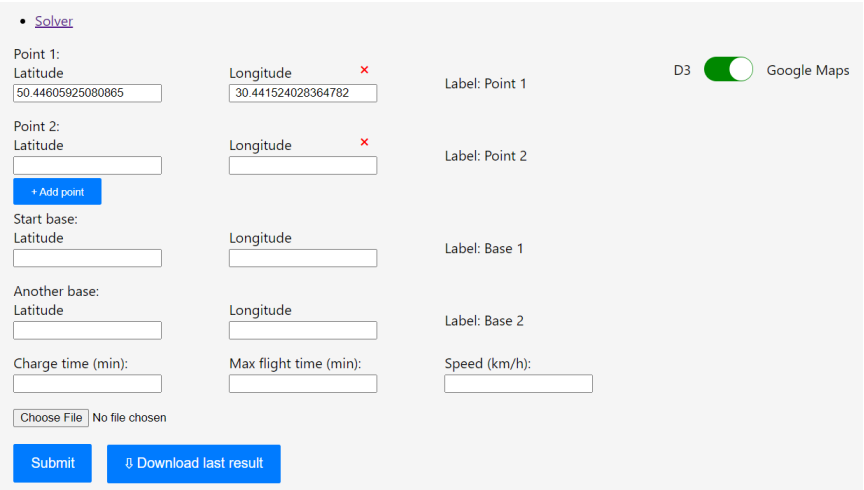


Рисунок 5.6 – Сторінка оператора після додавання об’єкта

Для того щоб розв’язати задачу спочатку потрібно додати всі необхідні об’єкти та ввести дані про їх географічне положення. Для прикладу розглянемо 5 об’єктів. Натискаємо кнопку «Submit» для розв’язання задачі. На рисунку 5.7 зображено сторінку після розв’язання задачі.

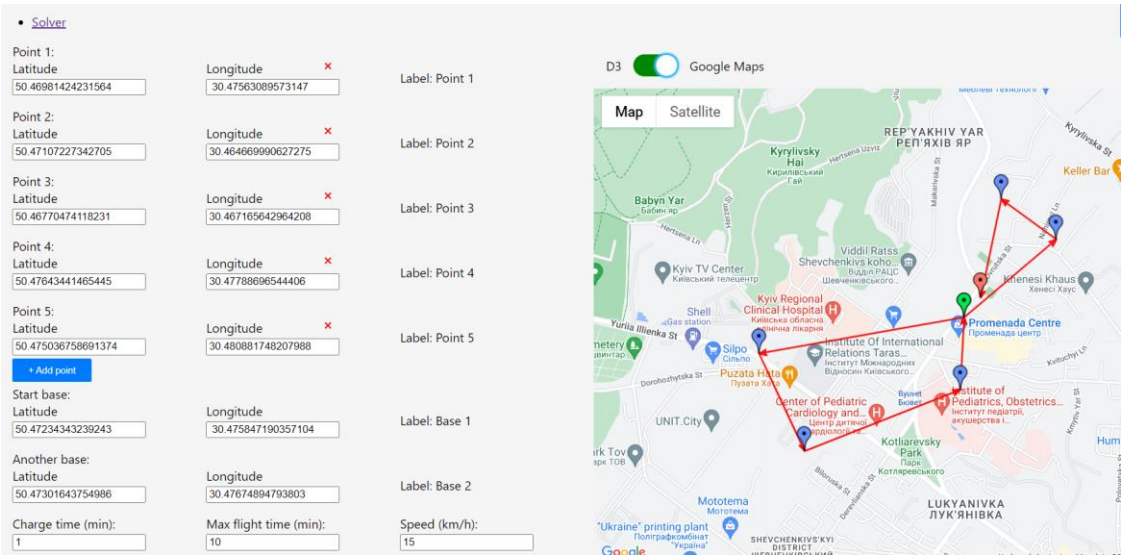


Рисунок 5.7 – Сторінка оператора після розв’язання задачі

Після розв’язання задачі оператор може переглянути результат в текстовому вигляді, який включає в себе інформацію про стартову станцію, об’єкти для обльоту, проміжні станції, кінцеву станцію. Також оператор на сторінці може побачити значення сумарної відстані маршруту обльоту БПЛА та час, за який БПЛА виконав завдання. На рисунку 5.8 зображено результат в текстовому вигляді.

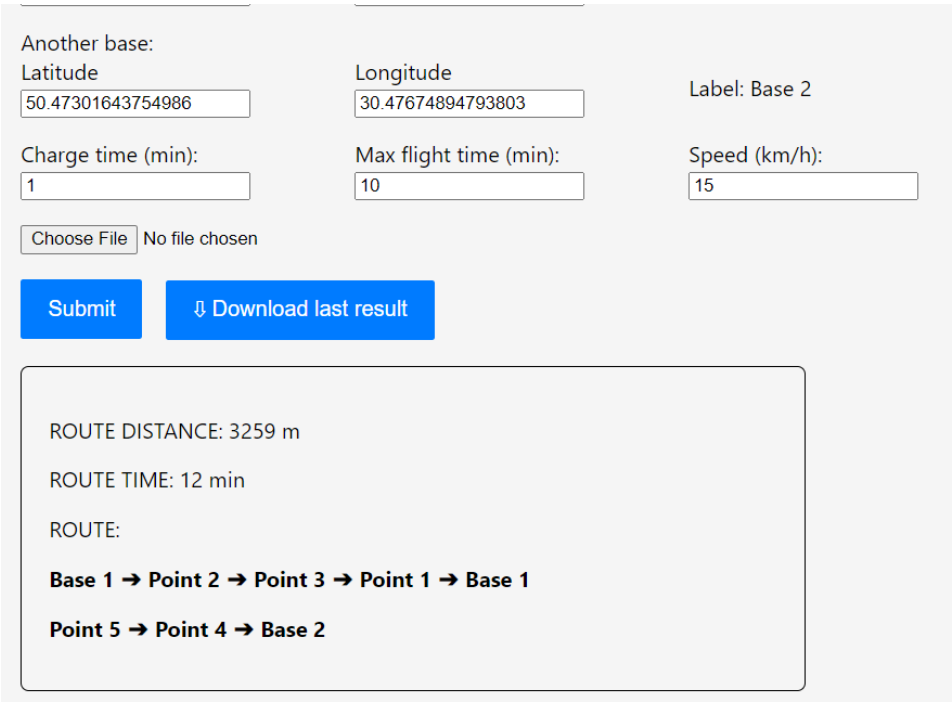


Рисунок 5.8 – Сторінка оператора з результатами задачі

Також при розв’язанні задачі оператор може переглянути побудований маршрут на Google Maps, як на рисунку 5.9 або графічно зображений маршрут, вимкнувши Google Maps, як на рисунку 5.10.

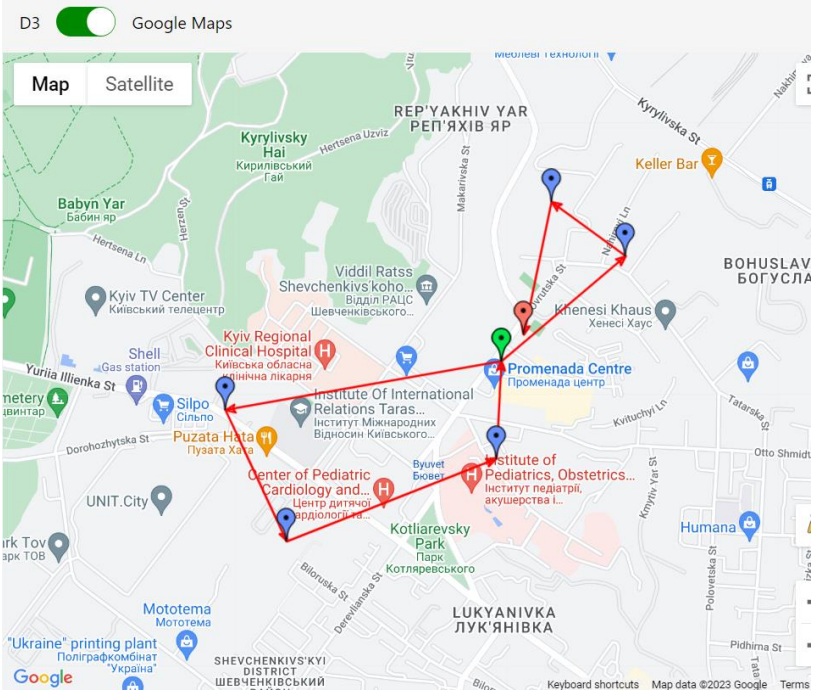


Рисунок 5.9 – Сторінка оператора з побудованим маршрутом на Google Maps

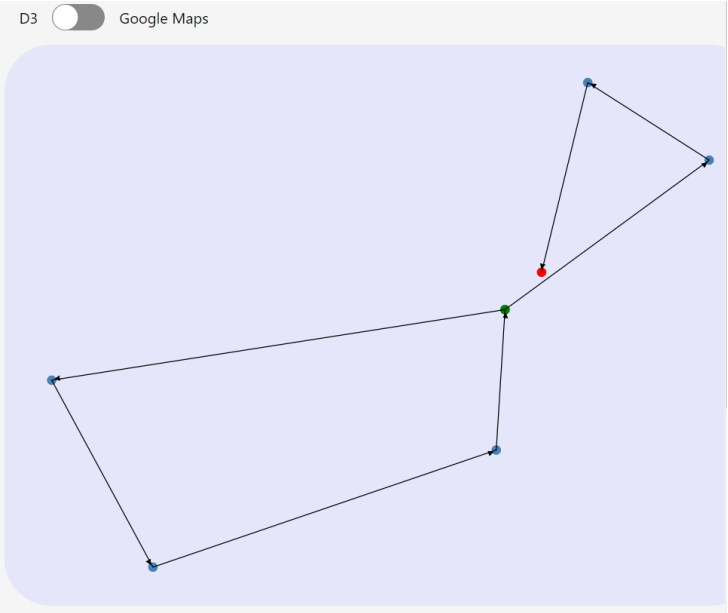


Рисунок 5.10 – Сторінка оператора з побудованим маршрутом

Оператор має право зберегти дані про результат побудованого маршруту у JSON-файл, натиснувши кнопку «Download last result». На рисунку 5.11 зображений збережений JSON-файл.


```
1 {"fitness":730544.8525797207,"route":
2 [{"id":null,"isBase":true,"isStartBase":true,"label":"Base 1","lat":50.47234343239243,"lng":30.475847190357104},
3 {"id":0,"isBase":false,"isStartBase":false,"label":"Point 2","lat":50.47107227342705,"lng":30.464669990627275},
4 {"id":1,"isBase":false,"isStartBase":false,"label":"Point 3","lat":50.46770474118231,"lng":30.467165642964208},
5 {"id":null,"isBase":false,"isStartBase":false,"label":"Point 1","lat":50.46981424231564,"lng":30.47563089573147},
6 {"id":null,"isBase":true,"isStartBase":true,"label":"Base 1","lat":50.47234343239243,"lng":30.475847190357104},
7 {"id":3,"isBase":false,"isStartBase":false,"label":"Point 5","lat":50.475036758691374,"lng":30.480881748207988},
8 {"id":2,"isBase":false,"isStartBase":false,"label":"Point 4","lat":50.47643441465445,"lng":30.47788696544406},
9 {"id":null,"isBase":true,"isStartBase":false,"label":"Base 2","lat":50.47301643754986,"lng":30.47674894793803}]}
```


Рисунок 5.11 – JSON-файл з результатами побудованого маршруту.

## 5.2 Випробування програмного продукту

### 5.2.1 Мета випробувань

Мета випробувань програмного продукту полягає у перевірці його функціональності, надійності та придатності для виконання запланованих завдань. Ці випробування спрямовані на виявлення помилок, проблем та недоліків продукту, що дозволяє забезпечити його якість та працездатність перед його впровадженням. Вони допомагають продемонструвати, що програмний продукт виконує заплановані функції, працює стабільно, забезпечує очікувану продуктивність та зручний інтерфейс оператора.

### 5.2.2 Результати випробувань

Випробування продукту та використання тестів "sunny-day" сценаріїв є важливою частиною процесу розробки та контролю якості програмного продукту. Ці сценарії допомагають перевірити, як продукт працює у найкращому випадку.

Основна мета випробування продукту полягає в перевірці його функціональності. Випробування дозволяють ідентифікувати й виправити помилки, недоліки або непередбачені ситуації, що можуть виникнути при роботі продукту в реальних умовах.

Тести "sunny-day" сценаріїв, зазвичай, виконуються умисно-створеними позитивними ситуаціями, коли все працює належним чином і немає ніяких проблем. Ці тести переконуються, що продукт працює як очікується, та дозволяє операторам досягти своїх цілей без перешкод.

Розглянемо наступний функціонал, який необхідно протестувати:

– зчитування з файлу даних для розв’язання задачі;
– розв’язання задачі;
– введення даних вручну;
– додавання нових об’єктів;
– відображення маршрутів на Google Maps;
– відображення маршрутів
– відображення результатів побудови маршрутів в текстовому вигляді;
– валідація полей розв’язання задачі.

У таблицях 5.1 – 5.8 наведено тестові сценарії основних функцій клієнтської сторони.

Таблиця 5.1 – Тестовий сценарій функції зчитування з файлу даних для розв’язання задачі

| | |
|---|---|
| Мета тесту | Перевірка коректності функціоналу зчитування з файлу даних для розв’язання задачі |
| Початковий стан системи | Завантажена сторінка оператора Solver |
| Вхідні дані | Файл з необхідними даними для розв’язання задачі |
| Кроки тестового сценарію | 1. Натиснути кнопку «Choose file»<br>2. Обрати необхідний файл |
| Очікуваний результат | Зчитані дані з’являються на головній сторінці |
| Стан системи після проходження тестового сценарію | Дані з’явились на головній сторінці |

| | |
|---|---|
| Результат проходження тестового сценарію | Успішно пройдено |

Таблиця 5.2 – Тестовий сценарій функції розв’язання задачі

| | |
|---|---|
| Мета тесту | Перевірка коректності функціоналу розв’язання задачі |
| Початковий стан системи | Завантажена сторінка оператора Solver |
| Вхідні дані | Дані, які необхідні для розв’язання задачі, а саме: дані про безпілотний літальний апарат(середня швидкість переміщення, часовий обмежений потенціал, час, за який необхідно перезарядити БПЛА), географічне положення об’єктів та станцій для перезарядки |
| Кроки тестового сценарію | 1. Ввести дані вручну або завантажити з файлу<br>2. Натиснути кнопку «Submit» |
| Очікуваний результат | Відображення результату розв’язання задачі |
| Стан системи після проходження тестового сценарію | Відображений результат розв’язання задачі |
| Результат проходження тестового сценарію | Успішно пройдено |

Таблиця 5.3 – Тестовий сценарій функції введення даних вручну

| | |
|---|---|
| Мета тесту | Перевірка коректності функціоналу ведення даних вручну |
| Початковий стан системи | Завантажена сторінка оператора Solver |
| Вхідні дані | – |

| | |
|---|---|
| Кроки тестового сценарію | Вести дані, які необхідні для обльоту |
| Очікуваний результат | Збереження даних у полях введення |
| Стан системи після проходження тестового сценарію | Дані, які вручну були введені, збережені у необхідних полях |
| Результат проходження тестового сценарію | Успішно пройдено |

Таблиця 5.4 – Тестовий сценарій функції додавання нових об’єктів

| | |
|---|---|
| Мета тесту | Перевірка коректності функціоналу додавання нових об’єктів |
| Початковий стан системи | Завантажена сторінка оператора Solver |
| Вхідні дані | – |
| Кроки тестового сценарію | Натиснути кнопку «Add point» |
| Очікуваний результат | З’являється нове поле для введення додаткових даних про новий об’єкт |
| Стан системи після проходження тестового сценарію | З’явилось додаткове поле для об’єкта |
| Результат проходження тестового сценарію | Успішно пройдено |

Таблиця 5.5 – Тестовий сценарій функції відображення маршрутів на Google Maps

| | |
|---|---|
| Мета тесту | Перевірка коректності функціоналу відображення маршрутів на Google Maps |

| | |
|---|---|
| Початковий стан системи | Завантажена сторінка оператора Solver |
| Вхідні дані | Завантажений файл або введені дані для задачі |
| Кроки тестового сценарію | Натиснути кнопку «Submit» |
| Очікуваний результат | Відображення прокладеного маршруту на Google Maps |
| Стан системи після проходження тестового сценарію | Відображений прокладений маршрут задачі на Google Maps |
| Результат проходження тестового сценарію | Успішно пройдений |

Таблиця 5.6 – Тестовий сценарій функції відображення маршрутів

| | |
|---|---|
| Мета тесту | Перевірка коректності функціоналу відображення маршрутів |
| Початковий стан системи | Завантажена сторінка оператора Solver |
| Вхідні дані | Завантажений файл або введені дані для задачі |
| Кроки тестового сценарію | 1. Натиснути кнопку «Submit»<br>2. Вимкнути Google Maps |
| Очікуваний результат | Відображення прокладеного маршруту |
| Стан системи після проходження тестового сценарію | Відображений прокладений маршрут |
| Результат проходження тестового сценарію | Успішно пройдений |

Таблиця 5.7 – Тестовий сценарій функції відображення результатів побудови маршрутів в текстовому вигляді

| | |
|---|---|
| Мета тесту | Перевірка коректності функціоналу відображення результатів побудови маршрутів в текстовому вигляді |
| Початковий стан системи | Завантажена сторінка оператора Solver |
| Вхідні дані | Завантажений файл або введені дані для задачі |
| Кроки тестового сценарію | Натиснути кнопку «Submit» |
| Очікуваний результат | Відображення результату в текстовому вигляді, який включає маршрут вильоту БПЛА з першої станції, відвідані об’єкти, станції для перезарядки та станцію, на яку повертається БПЛА |
| Стан системи після проходження тестового сценарію | Відображений детальний результат побудованого маршруту в текстовому вигляді |
| Результат проходження тестового сценарію | Успішно пройдений |

Таблиця 5.8 – Тестовий сценарій функції валідація полей розв’язання задачі

| | |
|---|---|
| Мета тесту | Перевірка коректності функціоналу додавання нових об’єктів |
| Початковий стан системи | Завантажена сторінка оператора Solver |
| Вхідні дані | – |
| Кроки тестового сценарію | Ввести дані, які недостовірні, наприклад текст чи символи, які не відповідають числам |

| | |
|---|---|
| Очікуваний результат | Інформування про некоректне введення даних |
| Стан системи після проходження тестового сценарію | Оператор проінформований про некоректне введення даних |
| Результат проходження тестового сценарію | Успішно пройдений |

Висновки до розділу

У даному розділі було розглянуто технологічний аспект проєкту, включаючи керівництво оператора та випробування програмного продукту.

Керівництво оператора надає покрокову інформацію як працює додаток зі сторони оператора.

Щодо випробувань програмного продукту, їх метою було перевірити функціональність продукту для виконання запланованих завдань.

Результати випробувань надають оцінку якості та працездатності програмного продукту. Згідно з сформульованою метою випробувань було розроблено тестові сценарії. Також продемонстровано результати проведених випробувань. Під час тестування системи було підтверджено, що система працює відповідно до поставлених вимог і відповідає очікуванням.

# ВИСНОВКИ

У даному дипломному проєкті було розроблена клієнтська частина інформаційної системи маршрутизації БПЛА з використанням алгоритму мурашиних колоній.

Було вивчено предметне середовище та проведено огляд існуючих аналогів, що дало можливість визначити мету та сформулювати завдання.

Розглянуто класифікацію задач маршрутизації транспортних засобів та обґрунтовано до якого виду належить наша задача. Проаналізовано методи розв’язання задач маршрутизації та розроблено алгоритм мурашиних колоній для планування та оптимізації маршрутів. Також було проведено підбір параметрів для побудови ефективних маршрутів. Спочатку було проведено встановлення параметрів, які було необхідно визначити. Ці параметри включали: кількість мурах, кількість повторень з результатами без покращення, коефіцієнт випаровування, ступінь значущості феромону та ступінь значущості евристичної інформації. Далі було проведено визначення можливих діапазонів значень для кожного з цих параметрів. За допомогою ручного пошуку було встановлено значення параметру кількості мурах, тоді як інші параметри були випадковим чином визначені шляхом використання 100 різних згенерованих наборів параметрів.

Обґрунтовані використані засоби розробки системи, а саме: програмне середовище, мова та бібліотеки , які використані для розробки клієнтської частини системи.

Описано тести сценарії для основного функціоналу клієнтської сторони. На основі отриманих результатів випробувань можна зробити висновок, що програмний продукт відповідає поставленим вимогам і демонструє задовільну працездатність.

Існує потенціал для подальшого розвитку та вдосконалення системи. Напрями досліджень можуть включати розширення функціональності, такі як підтримка додаткових типів БПЛА, покращення інтерфейсу оператора та

впровадження більш розширених опцій для конфігурації маршрутів. Також можлива інтеграція з додатковими джерелами даних, такими як датчики або системи моніторингу, а також дослідження нових методів оптимізації та автоматизації процесів маршрутизації.

У сфері моніторингу навколишнього середовища, система може використовуватись для виявлення та моніторингу забруднень, визначення якості повітря та води, а також для спостереження за екосистемами. В рятувальних операціях система може допомагати в пошуку та рятуванні загублених або поранених осіб, забезпечуючи точне визначення їх місцезнаходження та шляхи досягнення.

Застосування розробленої системи в цих галузях може призвести до покращення продуктивності, ефективності та безпеки роботи, а також до зниження ризиків, пов'язаних з людським фактором. Подальше дослідження і розвиток системи можуть сприяти її оптимізації та вдосконаленню для забезпечення ще більш широкого спектру застосувань і задоволення потреб різних галузей.

ПЕРЕЛІК ПОСИЛАНЬ


1. Bastian, C. and Rinnooy Kan, A.H.G."The stochastic vehicle routing problem revisited", European Journal of Operational Research 56. – 1992. -pp.407-412.

2. DJI Terra [Електронний ресурс]. – 2023. – Режим доступу до ресурсу: https://enterprise.dji.com/dji-terra .

3. Altitude Angel [Електронний ресурс]. – 2023. – Режим доступу до ресурсу: https://www.altitudeangel.com/ .

4. Ni Q., Tang Y. A Bibliometric Visualized Analysis and Classification of Vehicle Routing Problem Research. *Sustainability*. 2023. Vol. 15, no. 9. P. 7394.

5. Гуляницький Л. Ф., Коткова А. А. До класифікації задач маршрутизації транспортних засобів. Науковий вісник Ужгородського університету. Серія: Математика і інформатика. 2020. № 1(36). С. 73–84.

6. Гуляницький Л. Ф., Сторчевий В. В. Одна спеціальна задача маршрутизації БПЛА. Науковий вісник Ужгородського університету. Серія: Математика і інформатика. 2019. № 1(34). С. 69–78.

7. Konstantakopoulos G. D., Gayialis S. P., Kechagias E. P. Vehicle routing problem and related algorithms for logistics distribution: a literature review and classification. Operational Research. 2020. URL: https://doi.org/10.1007/s12351-020-00600-7.

8. Гуляницький Л.Ф., Мулеса О. Ю. Прикладні методики комбінаторної оптимізації: навч. посіб.: Видавничо-поліграфічний центр " Київський університет", 2016. – 142 с.

9. Martí R., Reinelt G. Heuristic Methods. Exact and Heuristic Methods in Combinatorial Optimization. Berlin, Heidelberg, 2022. P. 27–57.

10. Зіньков Р.В., Марчук Г.В. Державний університет «Житомирська політехніка». Принцип дії мурашиного алгориму при вирішенні задачі ковіміяжера. URL: https://conf.ztu.edu.ua/wp-content/uploads/2019/12/17-1.pdf

11. Skinderowicz R. Implementing a GPU-based parallel MAX–MIN Ant System. Future Generation Computer Systems. 2020. Vol. 106. P. 277–295.

12. Bonabeau, E. Dorigo, M.. Theraulaz, G. Swarm intelligence: from natural to artificial systems, Oxford University Press. – 2012.

13. Kollin F., Bavey A. Ant Colony Optimization Algorithms: Pheromone Techniques for TSP: thesis. 2017.

14. Pressman R. S., Maxim B. Software Engineering: A Practitioner's Approach. McGraw-Hill Education, 2019. 704 p.

15. Bruce J. Visual Studio Code: End-To-End Editing and Debugging Tools for Web Developers. Wiley & Sons, Incorporated, John, 2019. 312 p.

16. Ritika. What is TypeScript? Definition, History, Features and Uses [Електронний ресурс] –2022. – Режим доступу до ресурсу: https://invedus.com/blog/what-is-typescript-definition-history-features-and-uses-of-typescript/

17. Podila P., Weststrate M. MobX Quick Start Guide: Supercharge the client state in your React apps with MobX. Packt Publishing, 2018. 236 p.

18. Arancio S. ReactJS: A brief history [Електронний ресурс] / Stephen Arancio –2021. – Режим доступу до ресурсу: https://medium.com/@sjarancio/reactjs-a-brief-history-3c1e969a477f.

19. Windmill E. Flutter in Action. Manning Publications Co. LLC, 2020.

20. Brown E. Web Development with Node and Express: Leveraging the JavaScript Stack. O'Reilly Media, 2014. 332 p.

21. Satyam Kumar. 7 Hyperparameter Optimization Techniques Every Data Scientist Should Know. [Електронний ресурс] /–2021. – Режим доступу до ресурсу: https://medium.com/@sjarancio/reactjs-a-brief-history-3c1e969a477f.

22. A Complete Guide on Hyperparameters: Optimization Methods. [Електронний ресурс] – Режим доступу до ресурсу: https://indiantechwarrior.com/a-complete-guide-on-hyperparameters-optimization-methods/.