

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Факультет комп'ютерних наук і технологій

(повне найменування факультету)

Кафедра програмних засобів

(повне найменування кафедри)

Пояснювальна записка

до дипломного проекту (роботи)

магістр

(ступінь вищої освіти)

на тему ДОСЛІДЖЕННЯ ТА ПРОГРАМНА РЕАЛІЗАЦІЯ ЗАСОБІВ
ПОШУКУ НАЙКОРОТШОГО ШЛЯХУ НА ОСНОВІ
МУЛЬТИАГЕНТНОГО ПІДХОДУ
RESEARCH AND SOFTWARE IMPLEMENTATION OF
TOOLS FOR FINDING THE SHORTEST PATH
BASED ON A MULTI-AGENT APPROACH

Виконав(ла): студент(ка) 2 курсу, групи КНТ-122м

Спеціальності 121 Інженерія програмного

(код і найменування спеціальності)

забезпечення

Освітня програма (спеціалізація)

Інженерія програмного забезпечення

ЖЕЛНІН М.С.

(ПРІЗВИЩЕ та ініціали)

Керівник КОЦУР М.І.

(ПРІЗВИЩЕ та ініціали)

Рецензент СКРУПСЬКИЙ С.Ю.

(ПРІЗВИЩЕ та ініціали)

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Факультет КНТ
Кафедра програмних засобів
Ступінь вищої освіти магістр
Спеціальність 121 Інженерія програмного забезпечення
(код і найменування)
Освітня програма (спеціалізація) Інженерія програмного забезпечення
(назва освітньої програми (спеціалізації))

ЗАТВЕРДЖУЮ

Завідувач кафедри ПЗ, д.т.н, проф.
Сергій СУББОТІН
“ ” 2023 року

З А В Д А Н Н Я

НА ДИПЛОМНИЙ ПРОЄКТ (РОБОТУ) СТУДЕНТА(КИ)

ЖЕЛІНІНА Михайла Сергійовича

(ПРИЗВИЩЕ, ім'я, по батькові)

1. Тема проєкту (роботи) Дослідження та програмна реалізація засобів пошуку найкоротшого шляху на основі мультиагентного підходу. Research and Software Implementation of Tools for Finding the Shortest Path Based on a Multi-Agent Approach

керівник проєкту (роботи) к.т.н., доцент КОЦУР Михайло Ігорович,
(науковий ступінь, вчене звання, ПРИЗВИЩЕ, ім'я, по батькові)

затверджені наказом закладу вищої освіти від “14” листопада 2023 року № 437

2. Строк подання студентом проєкту (роботи) 01 грудня 2023 року

3. Вихідні дані до проєкту (роботи) рекомендована література, технічне завдання

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) 1. Аналіз предметної області. 2. Матеріали і методи. 3. Опис програми. 4. Експлуатація, тестування та експериментальне дослідження програми.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень, кількість слайдів, плакатів)

Слайди презентації

6. Консультанти розділів проєкту (роботи)

Розділ	ПРИЗВИЩЕ, ініціали та посада консультанта	Підпис, дата	
		завдання видав	прийняв виконане завдання
1-4 Основна частина	КОЦУР М.І., доцент		
Нормоконтроль	ЛИПОВЕЦЬ М.В., асистент		

7. Дата видачі завдання “ 08 ” вересня 2023 року.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проєкту (роботи)	Строк виконання етапів проєкту (роботи)	Примітка
1	Постановка завдання роботи.	1 тиждень	Завдання, ТЗ
2	Аналіз предметної області.	1 тиждень	Розділ 1
3	Вибір мови програмування та інших технологій	2 тиждень	Розділ 2
	розробки.		
4	Розробка структури програми.	3 тиждень	Розділ 3
5	Розробка програми.	4-7 тижні	Розділи 3, 4
6	Тестування та експериментальне дослідження	8 тиждень	Розділ 4
	програмного забезпечення.		
7	Оформлення пояснювальної записки та документів	9-10 тижні	Додатки
	до неї.		
8	Нормоконтроль та рецензування.	11 тиждень	
9	Захист роботи.	12 тиждень	

Студент(ка)

(підпис) Михайло ЖЕЛНІН
(Імя ПРИЗВИЩЕ)

Керівник проєкту (роботи)

(підпис) Михайло КОЦУР
(Імя ПРИЗВИЩЕ)

РЕФЕРАТ

Пояснювальна записка до дипломної кваліфікаційної роботи магістра:
96 с., 3 табл., 33 рис., 3 дод., 23 джерела.

JAVA, INTELLIJ IDEA, МУЛЬТИАГЕНТНИЙ МЕТОД,
НАЙКОРОТШИЙ ШЛЯХ, ОПТИМІЗАЦІЯ, ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ.

Об'єкт дослідження – процес розробки програмного забезпечення для багатовимірної оптимізації.

Предмет дослідження – методи та програмні засоби для розв'язання завдань пошуку найкоротшого шляху.

Метою роботи є дослідження та розробка програмного забезпечення для пошуку найкоротшого шляху на основі мультиагентного підходу.

Матеріали, методи та технічні засоби: мова програмування Java, середовище розробки IntelliJ IDEA.

Результати. Розроблено програмне забезпечення для розв'язання завдань пошуку найкоротшого шляху за допомогою мови програмування Java та середовища розробки IntelliJ IDEA.

Практична цінність роботи полягає у розробці програмного забезпечення, що реалізує методи мультиагентної оптимізації та дозволяє шукати найкоротший шлях.

Висновки. Виконано проєктування програмного забезпечення для пошуку найкоротшого шляху на основі мультиагентного підходу. Розроблено програмне забезпечення для розв'язання завдань пошуку найкоротшого шляху. Здійснено тестування розробленого програмного забезпечення для розв'язання завдань пошуку найкоротшого шляху.

Галузь використання – програмні засоби для оптимізації транспортних потоків.

ABSTRACT

Explanatory note to the diploma qualifying work of the master: 96 pages, 3 tables, 33 figures, 3 appendixes, 23 sources.

JAVA, INTELLIJ IDEA, MULTI-AGENT METHOD, SHORTEST PATH, OPTIMIZATION, SOFTWARE.

The object of research is the process of developing software for multidimensional optimization.

The subject of the research is software for solving the tasks of finding the shortest path.

The purpose of this work is to develop the software for finding the shortest path based on a multi-agent approach.

Materials, methods and technical tools: Java programming language, IntelliJ IDEA development environment.

Results. The software tools for solving problems of finding the shortest path using Java programming language and IntelliJ IDEA development environment have been developed.

The practical value of the work lies in the development of software that implements multi-agent approach and allows you to search for the shortest path.

Conclusions. The software for finding the shortest path based on the multi-agent approach has been designed. The software for finding the shortest path has been developed. The developed software for finding the shortest path has been tested.

Scope of use – software tools for traffic flow optimization.

ЗМІСТ

	С.
Перелік скорочень та умовних позначок	8
Вступ.....	9
1 Аналіз предметної області.....	11
1.1 Задача пошуку найкоротшого шляху	11
1.2 Аналіз програмного забезпечення для пошуку найкоротшого шляху....	22
1.3 Висновки за розділом 1	35
2 Матеріали і методи	37
2.1 Вибір мови програмування	37
2.2 Вибір середовища розробки для пошуку найкоротшого шляху.....	39
2.3 Інтелектуальні методи для пошуку оптимального шляху.....	41
2.4 Розробка моделі взаємодії користувача з програмним забезпеченням пошуку найкоротшого шляху.....	46
2.5 Висновки за розділом 2	48
3 Опис програми	49
3.1 Структура програмного забезпечення для пошуку найкоротшого шляху.....	49
3.2 Функціонування програмного забезпечення для пошуку найкоротшого шляху.....	50
3.3 Основні рішення щодо розробки програмного забезпечення для генерації текстової інформації.....	52
3.4 Проектування інтерфейсу програмного забезпечення для пошуку найкоротшого шляху.....	55
3.5 Висновки за розділом 3	59
4 Експлуатація, тестування та експериментальне дослідження програми.....	60
4.1 Призначення й умови застосування програми	60
4.2 Характеристики програми для розв'язання задачі пошуку найкоротшого шляху.....	61
4.3 Інструкція по експлуатації програми.....	62

4.3.1 Звернення до програми	62
4.3.2 Вхідні й вихідні дані	62
4.3.3 Повідомлення	62
4.4 Виконання програмного забезпечення для розв'язання задачі пошуку найкоротшого шляху	63
4.5 Тестування програмного забезпечення для пошуку оптимального шляху	67
4.6 Висновки за розділом 4	68
Висновки	69
Перелік джерел посилань	72
Додаток А Технічне завдання	75
Додаток Б Фрагмент тексту програми	79
Додаток В Слайди презентації	89

ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАК

ГА – генетичний алгоритм;

МП – мультиагентний пошук;

НМ – нейронні мережі;

ПЗ – програмне забезпечення;

ШІ – штучний інтелект.

ВСТУП

Програмні засоби для пошуку найкоротших маршрутів відіграють важливу роль у різних галузях, таких як транспорт, логістика, мережеве планування, інформаційні технології і багато інших. Вони набувають особливої важливості в сучасному світі, де управління маршрутами, оптимізація доставки та аналіз мережі стають критичними [1]-[7].

Ці програмні рішення спираються на алгоритми багатовимірної оптимізації, які дозволяють знаходити найкоротший шлях між двома точками у графах та мережах в умовах наявних обмежень [1]-[3].

Використання відповідного програмного забезпечення у транспортній галузі допомагає оптимізувати маршрути та знизити витрати на паливе, а в логістиці – планувати ефективні та швидкі доставки товарів різного призначення. У сфері інформаційних систем ці алгоритми використовуються для маршрутизації пакетів даних у комп'ютерних мережах та оптимізації планування маршрутів роботів у робототехніці [2]-[6].

З урахуванням широкого спектру застосувань, ці програмні рішення для пошуку найкоротших маршрутів є важливими для різноманітних сфер, що вимагають оптимізації маршрутів та найбільш ефективного використання ресурсів [1]-[7].

Однак відомі програмні засоби для пошуку оптимального шляху можуть використовувати неефективні алгоритми оптимального пошуку, які базуються на пошуку по всіх можливих шляхах та, відповідно, є досить складними для реалізації та використання (вимагають великої кількості часу та апаратних ресурсів). Деякі алгоритми, що використовуються у відповідному програмному забезпеченні, можуть не мати можливості ураховувати деякі особливості, такі як зміна умов транспортного потоку або різних обмежень, що можуть виникати у реальних умовах. Інші програмні засоби можуть вимагати значної кількості пам'яті для зберігання великих транспортних мереж або матриць шляхів, особливо у складних системах [1]-[7].

Тому актуальною є розробка програмного забезпечення для пошуку найкоротшого шляху на основі мультиагентного підходу.

У дипломній кваліфікаційній роботі магістра розв'язується актуальне завдання розробки програмного забезпечення для пошуку найкоротшого шляху на основі мультиагентного підходу.

Для досягнення поставленої мети у кваліфікаційній роботі магістра необхідно розв'язати такі задачі:

- виконати аналіз предметної області та програмних засобів для пошуку найкоротшого шляху;
- розробити модель взаємодії користувача з програмним забезпеченням для пошуку найкоротшого шляху;
- здійснити проектування програмного забезпечення для пошуку найкоротшого шляху на основі мультиагентного підходу;
- створити програмне забезпечення для пошуку найкоротшого шляху на основі мультиагентного підходу;
- виконати тестування розробленого програмного забезпечення для пошуку найкоротшого шляху.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Задача пошуку найкоротшого шляху

Мета пошуку найкоротшого шляху полягає в визначенні найбільш оптимального шляху між двома точками в графі, де сума ваг ребер, утворюючи цей шлях, є найменшою [1]-[8]. Це завдання має велике значення в багатьох галузях науки та техніки, і вирішення цієї проблеми потребує розробки ефективних методів та програмних засобів. Багато задач оптимізації можна сформулювати у вигляді математичних моделей, які можна візуалізувати за допомогою графів. Такий підхід дозволяє використовувати алгоритми теорії графів для пошуку найкращих рішень та зменшення критеріїв оптимальності аналітичних моделей. Отже, розв'язання таких задач зводиться до знаходження найкоротшого шляху між вершинами графа. Проблема пошуку найкоротшого шляху є однією з найважливіших у класичній теорії графів [5]-[9].

Ця задача виникає під час вирішення оптимізаційних завдань у різних галузях (таких як задачі комівояжера, транспортні проблеми і багато інших), де необхідно знати відстані між усіма точками графа. Крім того, ця проблема є вельми досить актуальною, оскільки її розв'язання надає можливість розробки програмного забезпечення для різних прикладних завдань, зокрема, для завдання пошуку оптимального маршруту з однієї точки до іншої, завдання пошуку найближчої та найдалшої точки (пункту призначення) від поточної точки, а також розрахунку відповідних відстаней [3]-[7].

Проблема знаходження найкоротшого шляху в графі є важливою для багатьох областей застосування. Існують різні алгоритми для її вирішення. Останнім часом вона активно вивчається для графів зі складною структурою. Це важливо, оскільки в реальних задачах потрібно знайти найкоротший шлях через залізничний маршрут, місто або складний вузол транспорту [6]-[9].

Проблема пошуку шляхів є ключовою у теорії графів. Тимчасовий графік відображає зміни в зв'язках ребер упродовж часу. Мінімізація стрибків

у часовому графі полягає у виявленні маршрутів від початкової до кожної цільової вершини з мінімальною кількістю переходів через ребра за час. Проблема шляхів у часових графах стоїть перед викликом знаходження оптимального часу для досягнення пункту призначення [2]-[7].

Пошук оптимального маршруту або маршрутизація – це процес, за допомогою якого комп'ютерна програма визначає найбільш оптимальний шлях між двома точками. Це практичний підхід до вирішення завдань маршрутизації. Ця область досліджень широко використовує алгоритм Дейкстри для пошуку найкоротшого шляху у зважених графах [1]-[9].

У теорії графів та відповідному програмному забезпеченні проблема пошуку оптимального шляху в тісному співвідношенні з проблемою найкоротшого шляху встановлює відповідність між двома точками у великій мережі за певними критеріями, такими як найкоротший, найбільш вигідний, найшвидший і т. д. [1]-[9].

При розв'язанні задачі пошуку найкоротшого шляху використовуються поняття часових графів. Під часовим графом розуміють граф, де ребра мають позначки, пов'язані з часом. Наприклад, в графі, де вершини відображають аеропорти, рейс з одного міста до іншого може бути представлений ребром, яке відповідає відправленню відповідно з одного міста і прибуттям до іншого у певний момент часу. На цій діаграмі вершини означають перехрестя, лінії між ними позначають маршрути від одного місця до іншого. Комбінація різних моделей інтервалів чергування та періодів, коли ділянка дороги недоступна через технічне обслуговування, можуть бути відображені на цій діаграмі. Також на ній можуть бути показані послідовність дотиків до країв аеропорту та інтервали часу у дорожній мережі [1]-[9].

Якщо час вимірюється в дискретних одиницях, то кожний часовий граф, що відображає послідовність подій, має однаковий інтервальний граф, і навпаки. Тимчасові графи, відомі як динамічні за часом, використовуються для розуміння поширення захворювань, передачі інформації через фізичний або віртуальний контакт між людьми, а також для аналізу поведінки в онлайн

соціальних мережах. Ці графи моделюють передачу даних в телефонних мережах, рух в дорожніх мережах та досліджують біологічні мережі на молекулярному рівні [1]-[9].

Проблема інтелектуального аналізу часових графів та досяжності розглядаються у різних роботах [7]-[9]. Такі графи математично еквівалентні випадково впорядкованим графам. Одна з проблем полягає у пошуку маршруту, який починається з певної точки, відвідує всі вершини графа та займає найменший час. У роботах [7]-[9] розглядається проблема комівояжера для часових графів, наведено алгоритм апроксимації для знаходження найкоротшого шляху, який відвідує всі вершини графа у певний час. Також досліджуються інші часові варіації комбінаторних задач оптимізації [1]-[9].

Нехай на часовому графіку послідовності контактів кожне з'єднання відображається як кортеж, що містить час t , необхідний для початку подорожі від u до v через ребро e . Цей час t вказує, коли можлива подорож від u до v стартує через ребро e . У разі, якщо є кілька доступних часових інтервалів для переміщення від u до v , такі різні часові варіанти між u і v формують послідовність часових ребер. Кількість таких часових переходів між вершинами u і v , які також відповідають дозволеним моментам початку подорожі від u до v , визначає кількість шляхів у з'єднанні (u, v) [1]-[9].

Граф інтервалів у часі подамо так, що кожне з'єднання в інтервальному часовому графі представлене множиною (u, v, intvls) . Ця множина визначає з'єднання від u до v , а intvls – це набір інтервалів часу, що вказує на те, коли ребро змінюється. Початковий та кінцевий часи відображають моменти, коли переміщення починається від u до v , при цьому інтервали упорядковані за зростанням часу початку і визначають діапазон часу відправлення. Всі інтервали, пов'язані з ребрами (u, v) , мають однакові значення часу завершення, що вказує на час завершення подорожі в точці (u, v) замість останнього можливого часу відправлення. У цій моделі час, що визначений для u , відповідає інтервалу з номером i [5]-[7].

Шлях у часовому графі представляє собою послідовність вершин та час початку з дозволим часом початку від початкової до кінцевої точки. Цей шлях визначається через з'єднання вершин та часу, утворюючи точку та напрямок [3]-[6].

Проаналізуємо маршрути у часовому графі, які розпочинаються у вершині u і закінчуються в іншій вершині v у конкретний момент часу. Позначимо $S(u, v)$ як усі прямі маршрути від u до v , які стартують з вершини u в певний момент. Критичний маршрут – це той, який відбувається до досягнення v з точки $S(u, v)$, мінімальний стрибок – маршрут з найменшою кількістю переходів у $S(u, v)$, швидкий маршрут – це оптимальний маршрут у $S(u, v)$ (час прибуття v – час відправлення з u), а найкоротший маршрут – це маршрут $S(u, v)$, який мінімізує суму переходів [3]-[6].

Було розроблено поліноміальні алгоритми для найдовших, мінімальних стрибків і найшвидших маршрутів у часових графіках. Відповідні алгоритми також були розроблені для зворотних траєкторій (маршрутів, що розпочинаються пізніше і закінчуються у фіксованій вершині) [1]-[9].

Для роботи з графами при пошуку оптимальних шляхів у програмного забезпеченні використовують пов'язані списки суміжності для представлення інтервальних часових графіків. Також можна використовувати списки суміжності у вигляді таблиць. Наприклад, інтервальний часовий граф може бути відображений у вигляді списку матриць. Структура даних складається з вектора, де зберігається інформація про кожну вершину графа. Цей вертикальний вектор представляє собою опис відповідних даних, де для кожної вершини u вектор містить інші вершини, які є сусідніми до u . Наприклад, у вертикальному векторі поряд з вершиною можуть бути розташовані вершини-сусіди. Кожна сусідня вершина v до u має відсортовані за часом кортежі, які є частиною ребра (u, v) [1]-[9].

Проаналізуємо два методи представлення даних для діаграми послідовності контактів. Перший з них – граф у формі впорядкованої послідовності ребер (кортежів), які упорядковані за зростанням часу t . Друге

представлення являє собою діаграму. Дослідження показали, що другий метод має гірші результати в усіх випробуваних випадках траєкторій, за винятком найшвидшої траєкторії, де обидва методи конкурують. Графічне представлення в другому методі має більше ребер, ніж впорядкована послідовність. Оскільки час змінюється, кожен часовий інтервал графа можна перетворити на граф послідовності зв'язків, що включає шляхи прямого, мінімального стрибка, найкоротшого, найшвидшого та зворотного найшвидшого шляху. При цьому перетворенні кожне ребро інтервального часового графа замінюється ребром послідовності контактів, який відповідає дозволеному часу початку інтервалу ребра [1]-[9]. Зрозуміло, що ця трансформація зберігає правильні траєкторії, але може значно збільшити кількість ребер та обсяг пам'яті, необхідний для зберігання даних. Це збільшення розміру може призвести до проблеми траєкторій в інтервальній моделі, тоді як у моделі послідовності контактів це може бути поліноміальною задачею [1]-[9].

Також, кожен граф часової послідовності контактів може бути перетворений у відповідний інтервальний часовий граф шляхом об'єднання одного ребра (u, v) та декількох ребер, які з'єднують впорядковану послідовність інтервалів за певний час. Час початку кожного інтервалу стає часом закінчення [1]-[9]. Основний статичний граф будь-якої послідовності контактів у часі – це граф, утворений заміною кожного ребра (u, v) ребром (u, v) , а потім заміною кількох входжень того самого краю (u, v) відповідним ребром. Така заміна може відбуватися кілька разів. Для інтервально-часового графа наступний статичний граф формується шляхом заміни кожного ребра іншим часовим ребром [1]-[9].

Проблема ациклічного шляху без очікування полягає у пошуку шляху від вершини u до вершини v у перехідному графі, який має ациклічну базову структуру. Як було відзначено раніше, визначення таких шляхів у графах інтервальних часових моделей є досить складною задачею, якщо графи не мають обмеження на ациклічність. У цьому випадку проблема для

інтервальних моделей може бути розв'язана поліноміальним методом при використанні контактної послідовності (ациклічних графів) [1]-[9].

У наш час відомі алгоритм для пошуку найбільш значущих шляхів від початкової точки s до всіх інших точок у інтервальному часовому графіку. Однак ці алгоритми передбачають, що значення всіх інтервалів ребер є однаковими (хоча можуть відрізнятися для різних ребер). Невелике розширення цього алгоритму дозволяє працювати з інтервальними графами, де значення ребра змінюються від інтервалу до інтервалу для того ж самого ребра [1]-[9].

Для пошуку найбільш важливих шляхів широко використовуються методи на основі алгоритму пошуку найкоротших шляхів Дейкстри. У таких методах спочатку визначаються найбільш важливі шляхи від стартової точки до самої себе, після чого відображаються найбільш вагомні шляхи до інших вершин у відповідності до їх часу прибуття. Після знаходження нового найважливішого шляху, його розширення розглядаються як одноразові розширення згідно з алгоритмом Дейкстри, а також оцінюється приблизна доступність найважливішого шляху в найближчій точці, оновлюючи час. Мінімальний пріоритет формує чергу з найбільш важливих запланованих шляхів, оцінюючи мінімальний час для досягнення найважливішого шляху в кожному раунді [1]-[9].

Алгоритм Дейкстри є важливим інструмент для знаходження найкоротшого шляху від однієї точки до іншої, який широко використовується у відповідному програмному забезпечення. Цей метод, створений голландським математиком Е. В. Дейкстрою, призначений для роботи з ваговими графами. Навіть зараз цей алгоритм широко використовується в різних програмних системах маршрутизації та навігації для знаходження оптимальних шляхів [1]-[9].

У методі Дейкстри спочатку визначаються найкоротші маршрути до певної частини усього графа. Потім поступово розширюється підграф і оцінюється, як це розширення впливає на найкоротший шлях [1]-[9].

Алгоритм Дейкстри базується на ітеративному розширенні набору найдешевших доступних вузлів, що робить його розвитком вагового пошуку в ширину з використанням жадібного підходу [1]-[9].

Теорема Дейкстри стверджує: якщо найкоротший маршрут від А до С пролягає через В, то ділянка маршруту між А і В є найкоротшим з'єднанням між цими двома вузлами [1]-[9].

Алгоритм Дейкстри дозволяє обчислити найкоротший маршрут або найменшу вартість лише у випадку, якщо всі витрати є додатніми. У випадку негативних витрат алгоритм припиняє свою роботу [1]-[9].

Використання алгоритму Дейкстри полягає у розумному виборі ребер, щоб знайти найдешевший шлях [1]-[9]. Цей алгоритм обчислює витрати найменшого маршруту від стартового вузла до всіх інших на графі [1]-[9].

Алгоритм Дейкстри вибирає поточний найдешевший маршрут від стартового вузла до інших можливих вузлів. Це продовжується, поки всі вузли не будуть відвідані та знайдено найкращий шлях [1]-[9].

Для роботи алгоритму Дейкстри використовується черга, в якій зберігаються відвідані вузли. На початку черги знаходиться вузол, який є найдешевшим шляхом від початкового вузла [1]-[9].

У кінці роботи алгоритму Дейкстри можна скласти найдешевший маршрут до кожного вузла через перші ребра від стартового вузла. Якщо шляху до конкретного вузла немає, вартість вважається нескінченною, що означає, що до цього вузла неможливо дістатися [1]-[9].

Час виконання алгоритму Дейкстри визначається операціями видалення вузлів та корекції значень відстані сусідніх вузлів. У випадку, якщо всі операції черги мають сталий час виконання, а граф складається з m ребер та n вузлів, то час виконання складе $O(m + n)$ [1]-[9].

Існує ще один удосконалений варіант алгоритму Дейкстри, який використовує евристику для швидкого видалення невідповідних перевірок шляху та завжди забезпечує виявлення найкоротшого маршруту [1]-[9].

Для графів з від'ємними вагами краще скористатися алгоритмом Беллмана-Форда, який є більш жадібним [1]-[9]. Алгоритм Беллмана-Форда, названий на честь його розробників Річарда Беллмана та Лестера Форда, використовується для пошуку найкоротших шляхів у графах. Цей алгоритм може враховувати негативні ваги ребер, проте слід бути обережним і уникати циклів з від'ємними вагами [1]-[9].

Пошук оптимального шляху або маршрутизація є процесом, у якому програмне забезпечення визначає найбільш оптимальний маршрут між двома пунктами. Це ефективний спосіб вирішення проблеми шляхів. У теорії графів ця область досліджень сильно залежить від алгоритму Дейкстри для пошуку найкоротшого шляху у графах з вагами [1]-[9].

Як правило, методи пошуку оптимального шляху, що використовуються у відповідному програмному забезпеченні, починають свою роботу з одного кінця графа і просуваються до сусідніх вузлів, поки не досягають цільового вузла, зазвичай за найбільш економним шляхом. Пошукові методи графів, такі як пошук у ширину, можуть займати багато часу, щоб знайти шлях, але інші методи, які швидше просуваються по графу, досягають цілі швидше. Замість того, щоб розглядати всі можливі шляхи, методи пошуку оптимального шляху та відповідне програмне забезпечення зазвичай йдуть у напрямку цілі, обираючи шлях з меншими перешкодами, щоб мінімізувати витрачений час [1]-[9].

Існують два основні типи задачі пошуку шляху. Перший тип задачі полягає у визначенні шляху між двома вузлами на графі. Другий тип задачі полягає у знаходженні найкоротшого шляху між цими точками. Основні алгоритми, такі як пошук у ширину та пошук у глибину, розв'язують перше завдання, проходячи через всі доступні маршрути від певної точки до мети. Ці алгоритми працюють за час $O(|V| + |E|)$, що означає лінійний час, де $|V|$ є кількістю вершин, а $|E|$ позначає кількість ребер між вершинами [1]-[9].

Задача пошуку найкращого шляху є складною. Для вирішення цієї задачі застосовується розширений підхід, відомий як алгоритм Беллмана-

Форда, який вимагає часу $O(|V||E|)$ або квадратичний час. Однак для знаходження найкращого шляху не обов'язково розглядати всі можливі маршрути. Програмні засоби та алгоритми, такі як A^* та Дейкстра, стратегічно виключають шляхи, використовуючи евристику чи динамічне програмування. Виключаючи неможливі варіанти, ці алгоритми можуть досягти часової складності $O(|E|\log(|V|))$ [1]-[9].

Ці алгоритми є найбільш універсальними та працездатними у відношенні до різних графів і не вимагають попередньої підготовки. Однак у реальних програмних засобах пошуку оптимального шляху використання алгоритмів, які можуть попередньо опрацьовувати графи, може призвести до більшої ефективності за рахунок вищої часової складності. Одним із таких алгоритмів є алгоритм ланцюжкового стиснення [1]-[9]. Цей метод базується на алгоритмі A^* . Алгоритм A^* є варіацією алгоритму Дейкстри, часто використовується в ігровій індустрії. Кожному відкритому вузлу A^* присвоює вагу, що дорівнює вазі краю цього вузла та приблизній відстані від нього до цільової точки. Ця оцінка отримується за допомогою евристичного методу і представляє мінімальну відстань між ним і ціллю. Для довгих маршрутів пошук може бути припинений, як тільки знайдений перший маршрут. Якщо від початкової точки до мети існує маршрут довжиною x , то цей вузол не перевіряється [1]-[9].

Алгоритм A^* використовує наближення для поліпшення процесу пошуку оптимального шляху, порівняно з алгоритмом Дейкстри. При нульовому значенні евристики алгоритм A^* виконується як алгоритм Дейкстри. Зі збільшенням точності наближення до фактичної відстані алгоритм A^* шукає оптимальніші шляхи, при цьому працюючи швидше (проходить менше вузлів). У разі, коли евристичне значення дорівнює фактичній відстані, алгоритм A^* розглядає мінімальну кількість вузлів. Підвищення значення евристики призводить до меншого обсягу досліджень вузлів, але алгоритм A^* не гарантує оптимальний шлях. Це є прийнятним або

навіть бажаним результатам у багатьох програмних засобах, особливо в відеоіграх, де важлива швидкість алгоритму [1]-[9].

Для пошуку оптимального шляху також може використовуватися метод Quadtree, який заснований на ієрархічному підході. Ця ідея спершу з'явилася у галузі відеоігор, коли виникла потреба у виготовленні карт великих розмірів за допомогою меншої обчислювальної потужності. Використання абстракцій та евристик у запропонованому методі є відомою концепцією для ефективного вивчення простору в логічних іграх. Подібним методом є навігаційна сітка, що використовується для геометричного планування в іграх і для багатомодального планування транспортних маршрутів в задачі комівояжера з використанням кількох транспортних засобів [1]-[9]. У такому підході карта поділена на групи. У шарі верхнього рівня планується шлях між кластерами. Після отримання плану пропонується інший маршрут у кластері нижчого рівня. Це означає, що проектування виконується у два етапи: керований локальний пошук у реальному просторі. Перевагою є те, що вузли малі, а алгоритм працює добре. Недоліком є те, що планувальник ієрархічного шляху важко виконати [1]-[9].

Ефективним засобом пошуку найкоротшого шляху є застосування мультиагентного підходу, що передбачає використання групи агентів або сутностей, кожен з яких приймає рішення для пошуку найкращого маршруту у великій області або графі. Цей підхід дозволяє розділити завдання пошуку шляху між кількома агентами, кожен з яких може мати власні правила та стратегії для пошуку найбільш оптимального маршруту від початкової точки до кінцевої. Агенти можуть обмінюватися інформацією про поточний стан і рішення для вдосконалення процесу пошуку. Цей підхід особливо корисний у складних системах або задачах, де потрібно враховувати багато факторів та обмежень при пошуку оптимального маршруту [5]-[9].

Мультиагентний пошук шляху включає в себе завдання пошуку маршруту для декількох агентів, які переміщуються від свого місця розташування до пункту призначення, уникаючи зіткнень та оптимізуючи

функцію вартості, яка враховує загальну довжину шляху для всіх агентів. Це розширення завдання пошуку шляху. Багато алгоритмів для багатоагентного пошуку шляху базуються на алгоритмі A^* або використовують підходи, що ґрунтуються на інших вивчених проблемах, таких як цілочисельне лінійне програмування. Проте ці алгоритми, як правило, не є повністю ефективними, тобто їхню здатність знаходити рішення за поліноміальний час не було доведено. Інший тип алгоритмів зосереджується на швидкості та ефективності, використовуючи відомі шаблони навігації (наприклад, потік трафіку) або топологію простору проблеми [5]-[9].

Визначено, що програмні засоби для пошуку найкоротших маршрутів відіграють важливу роль у різних галузях, таких як транспорт, логістика, мережеве планування, інформаційні технології і багато інших. Вони набувають особливої важливості в сучасному світі, де управління маршрутами, оптимізація доставки та аналіз мережі стають критичними. Показано, що мета пошуку найкоротшого шляху полягає в визначенні найбільш оптимального шляху між двома точками в графі, де сума ваг ребер, утворюючи цей шлях, є найменшою. Відзначено, що розв'язання задач пошуку оптимального шляху зводиться до знаходження найкоротшого шляху між вершинами графа. Показано, що існують різні алгоритми вирішення задачі пошуку оптимального шляху. Проаналізовано методи розв'язання цієї задачі. Визначено, що для пошуку найбільш важливих шляхів широко використовуються методи на основі алгоритму пошуку найкоротших шляхів Дейкстри, у таких методах спочатку визначаються найбільш важливі шляхи від стартової точки до самої себе, після чого відображаються найбільш вагомі шляхи до інших вершин у відповідності до їх часу прибуття. Відзначено, що пошукові методи на основі теорії графів можуть займати багато часу, особливо в задачах з великою кількістю пунктів призначення. Показано, що ефективним засобом пошуку найкоротшого шляху є застосування мультиагентного підходу, що передбачає використання групи агентів або сутностей, кожен з яких приймає рішення для пошуку найкращого маршруту у

великій області або графі. Відзначено, що мультиагентний підхід є особливо корисним у складних системах або задачах, де потрібно враховувати багато факторів та обмежень при пошуку оптимального маршруту.

1.2 Аналіз програмного забезпечення для пошуку найкоротшого шляху

Проаналізуємо програмне забезпечення підтримки для пошуку найкоротшого шляху [10]-[16].

Програмне забезпечення Circuit for Teams надає можливості, спрямовані на полегшення процесу доставки і задоволення потреб команд, які працюють у сфері доставки та водіїв [10], [14]. Ці можливості включають у себе ефективне планування маршрутів і підтвердження виконання доставок. Програмне забезпечення Circuit for Teams дозволяє планувати шляхи команд, які займаються доставкою останньою милі. Програмне забезпечення Circuit for Teams надає можливість отримання доступу до розширеної аналітики доставок [10], [14]. Ця можливість дозволяє відстежувати та аналізувати показники, такі як загальна кількість зупинок, відсоток запізнень та середня вартість доставки. Ці показники є ключовими для оцінки продуктивності та визначення напрямків для покращень [10], [14]. Використовуючи програмне забезпечення Circuit for Teams для детального аналізу груп, компанії, що займаються доставкою, можуть визначити оптимальні методи доставки, виявити сильні сторони та визначити сфери, які потребують уваги або вдосконалення [10], [14].

Використання програмного забезпечення Circuit for Teams для маршрутизації надає значні переваги у плануванні та виконанні маршрутів доставки. Програмне забезпечення Circuit for Teams дозволяє враховувати те, що маршрути можуть раптово змінюватись і включати багато факторів, які ускладнюють процес планування. Крім того, програмне забезпечення Circuit for Teams (рис. 1.1) надає засоби для оцінки трафіку, визначення відстаней між

зупинками, врахування обмежень на швидкість та розрахунок часу, необхідного для кожної зупинки водіями [10], [14].

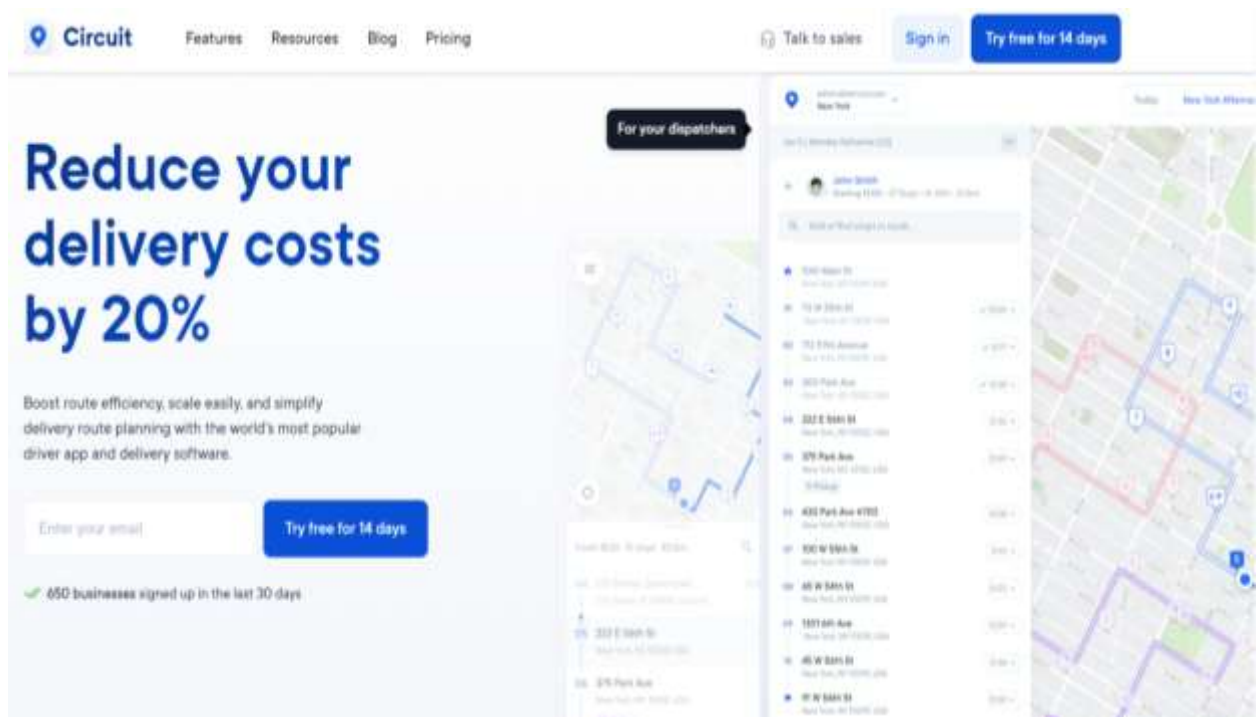


Рисунок 1.1 – Програмне забезпечення Circuit for Teams [10], [14]

Програмне забезпечення Circuit for Teams для оптимізації маршрутів дозволяє полегшити процес доставки та мінімізувати витрат часу на планування. Програмне забезпечення Circuit for Teams пропонує розширені можливості, такі як реальний час відстеження, сповіщення для клієнтів, підтвердження доставки та аналіз [10], [14].

Програмне забезпечення Circuit for Teams планування маршрутів забезпечує високу налаштовуваність та гнучкість, що дозволяє підприємствам адаптуватися в реальному часі, інтегруватися з іншими платформами та програмним забезпеченням, а також налаштовувати функціонал відповідно до конкретних операційних потреб [10], [14].

Програмне забезпечення Circuit for Teams надає можливість планування та перегляду. Ця програмна платформа дозволяє диспетчерам швидко створювати оптимальні маршрути лише за один клік, автоматично

розраховуючи найкращий шлях для доставки незалежно від часу, відстані, швидкості та параметрів зупинок [10], [14].

Зручний вебінтерфейс програмного забезпечення Circuit for Teams підтримує імпорт електронних таблиць на будь-який комп'ютер і дозволяє налаштовувати маршрути відповідно до доступних водіїв. Крім цього, програмна платформа Circuit for Teams забезпечує можливість відстеження водіїв у режимі реального часу, інформацію про завершені та майбутні зупинки, а також оновлені часи прибуття, щоб забезпечити безперебійність доставки [10], [14].

Програмне забезпечення Circuit for Teams легко інтегрується з популярними навігаційними додатками, такими як Google Maps, Waze та Apple Maps. Це дозволяє водіям налаштовувати та змінювати маршрути відповідно до непередбачених змін [10], [14].

Програма Circuit Delivery for Teams забезпечує моніторинг у реальному часі та спрощує процес доставки, використовуючи можливість відстеження у реальному часі. Ця функція забезпечує постійний контакт, швидке підтвердження доставки, визначення місця розташування і актуалізацію статусу [10], [14]. Завдяки цьому програмному забезпеченню агенти можуть адаптувати маршрути в реальному часі за необхідності, що полегшує легку інтеграцію будь-яких змін або доповнень до маршрутів та миттєве сповіщення водіїв [10], [14]. Ця гнучкість програмного забезпечення надзвичайно важлива для щоденного оновлення та непередбачених змін [10], [14].

Програмне забезпечення Circuit for Teams надає розширені сповіщення клієнтам [10], [14]. При виїзді водія програмне забезпечення Circuit for Teams відправляє сповіщення через текстові повідомлення, електронну пошту або обидва варіанти, надаючи клієнтам наближений час доставки та забезпечуючи можливість відстеження статусу посилки у режимі реального часу протягом дня, надаючи посилання на інформаційну панель [10], [14]. Користувач може встановлювати різні проміжки для доставки, створюючи таким чином кращі умови для спілкування зі своїми клієнтами [10], [14]. Користувач може

встановити фіксований час доставки, який являє собою визначений проміжок часу для всіх поставок, наприклад з дев'ятої до дванадцятої години. Динамічний час змінюється під час подорожі [10], [14]. Програма надає можливість клієнтам безпосередньо спілкуватися з водієм через службу обміну повідомленнями. Це дозволяє клієнтові повідомити водія про своє місцеположення або іншу важливу інформацію [10], [14].

Схема перевірки доставки у програмі Circuit for Teams забезпечує, що всі поставки перевірені та захищені. Ця функція дозволяє водіям збирати важливі докази, такі як підписи або фотографії, для підтвердження того, що доставка була успішно здійснена. Програмний модуль для водіїв у програмного забезпеченні Circuit for Teams представляє фотографії, зроблені водіями, для допомоги клієнту знайти своє замовлення [10], [14].

Програмний продукт Circuit for Teams має зручний мобільний додаток, який працює на усіх пристроях під управлінням операційних систем Android та iOS [10], [14]. Водії можуть вручну вводити маршрути або імпортувати електронні таблиці. Проте можливість імпорту електронних таблиць на мобільних пристроях наразі доступна лише для користувачів операційної системи Android [10], [14].

Програмне забезпечення Circuit for Teams пропонує прості та масштабовані плани тарифів, які адаптовані до розміру проєктної команди [10], [14]. Стартовий план (100 доларів на місяць) включає вартість для двох водіїв, а кожний наступний водій коштує додаткові гроші. Цей план дозволяє створювати та оптимізувати маршрути, а також відстежувати рух водіїв у реальному часі [10], [14].

Серед особливостей програмного забезпечення Circuit for Teams для пошуку оптимального шлях можна виділити такі [10], [14].

– можливість пошуку оптимального маршруту та здійснення планування маршрутів. Програма Circuit for Teams дозволяє швидко та ефективно планувати маршрути для доставок. Програмне забезпечення Circuit

for Teams надає можливість оптимізації маршрутів, зменшуючи час і вартість доставки [10], [14];

- засоби відстеження в реальному часі. Програмне забезпечення Circuit for Teams надає можливість відстежувати рух водіїв у реальному часі, забезпечує відстеження статусу та локації доставок в реальному часі для керівництва та клієнтів [10], [14];

- засоби взаємодії з клієнтами: програма забезпечує сповіщення клієнтам про статус їхніх замовлень та приблизний час доставки. Є можливість надсилати сповіщення через текстові повідомлення або електронну пошту [10], [14];

- сертифікати доставки: програмне забезпечення Circuit for Teams дозволяє збирати докази виконаної доставки, такі як підписи або фото, для підтвердження доставки товару [10], [14];

- гнучкі тарифні плани: програмне забезпечення Circuit for Teams має різні тарифні плани, адаптовані до розміру та потреб команди доставки. Це дозволяє обирати оптимальний план в залежності від розміру та потреб бізнесу [10], [14];

- мобільний застосунок: програмний продукт має зручний мобільний застосунок для роботи на пристроях під управлінням Android і iOS [10], [14];

- можливість вводити маршрути вручну або імпортувати електронні таблиці (функція доступна лише на пристроях Android [10], [14];

- засоби аналітики доставки: програмне забезпечення Circuit for Teams надає можливість аналізувати ключові показники ефективності доставки, такі як кількість зупинок, відсоток запізнень та середня вартість доставки. Дозволяє оцінювати продуктивність та визначати області для покращення. Гнучкі вікна доставки: можливість встановлювати статичні та динамічні часові вікна для оптимізації доставок. Статичні вікна дозволяють вказати фіксований інтервал для всіх доставок, тоді як динамічні вікна змінюються в реальному часі відповідно до маршруту [10], [14];

– засоби підтримки взаємодії клієнта та водія. Клієнти можуть безпосередньо спілкуватися з водіями через сервіс обміну повідомленнями. Це забезпечує зручну комунікацію клієнтів із водіями для уточнення деталей або повідомлення про спеціальні інструкції щодо доставки [10], [14].

Серед недоліків програмного забезпечення Circuit for Teams можна відзначити складність використання, зокрема, складність налаштування програми та роботу з деякими її функціями. Крім того, ця програма вимагає оплати, що ускладнює її використання для організацій з невеликим бюджетом. Іншим недоліком є недостатня інтеграція з іншими платформами [10], [14].

Програмне забезпечення Routific надає можливості ефективного планування маршрутів і підтвердження виконання доставок. Програмне забезпечення Routific (рис. 1.2) дозволяє планувати маршрути, а також надає можливість отримання доступу до розширеної аналітики доставок [10], [15]. Ця можливість дозволяє відстежувати та аналізувати показники, такі як загальна кількість зупинок, відсоток запізнень та середня вартість доставки.

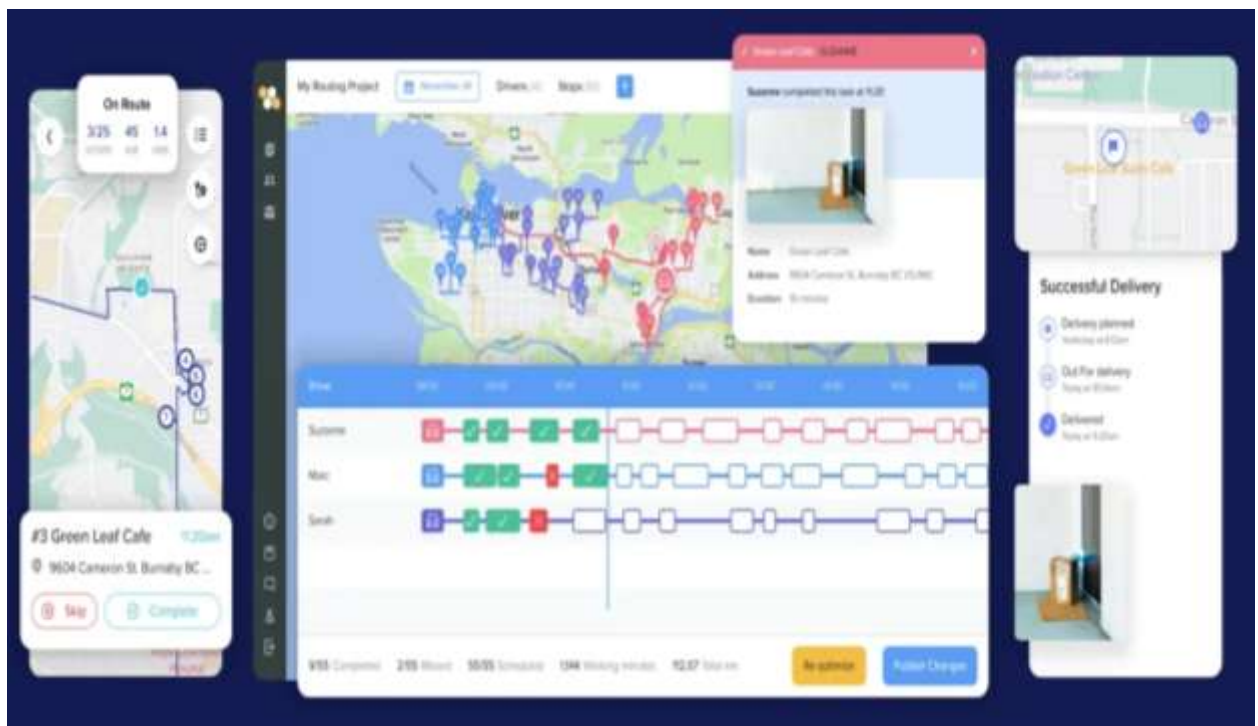


Рисунок 1.2 – Програмне забезпечення Routific [10], [15]

Програмне забезпечення Routific пропонує розширені можливості, такі як реальний час відстеження, сповіщення для клієнтів, підтвердження доставки та аналіз [10], [15]. Програмне забезпечення Routific для оптимізації маршрутів дозволяє полегшити процес доставки та мінімізувати витрат часу на планування. Використання програмного забезпечення Routific для маршрутизації надає значні переваги у плануванні та виконанні маршрутів доставки. Програмне забезпечення Routific дозволяє враховувати те, що маршрути можуть раптово змінюватись і включати багато факторів, які ускладнюють процес планування. Крім того, програмне забезпечення Routific надає засоби для оцінки трафіку, визначення відстаней між зупинками, врахування обмежень на швидкість та розрахунок часу, необхідного для кожної зупинки водіями [10], [15].

Програмне забезпечення Routific надає можливість планування та перегляду. Ця програмна платформа дозволяє диспетчерам швидко створювати оптимальні маршрути лише за один клік, автоматично розраховуючи найкращий шлях для доставки незалежно від часу, відстані, швидкості та параметрів зупинок [10], [15]. Програмне забезпечення Routific планування маршрутів забезпечує високу налаштовуваність та гнучкість, що дозволяє підприємствам адаптуватися в реальному часі, інтегруватися з іншими платформами та програмним забезпеченням, а також налаштовувати функціонал відповідно до конкретних операційних потреб [10], [15].

Програмна платформа Routific забезпечує можливість відстеження водіїв у режимі реального часу, інформацію про завершені та майбутні зупинки, а також оновлені часи прибуття, щоб забезпечити безперебійність доставки [10], [15].

Програмний продукт Routific має зручний мобільний додаток, який працює на усіх пристроях під управлінням різних операційних систем [10], [15]. Водії можуть вручну вводити маршрути або імпортувати електронні таблиці. Програмне забезпечення Routific пропонує прості та масштабовані плани тарифів [10], [15].

Серед особливостей програмного забезпечення Routific для пошуку оптимального шляху можна виділити такі [10], [15].

- засоби аналітики та формування звітів: програмне забезпечення Routific надає засоби для аналізу ефективності маршрутів та надає звіти для оцінки продуктивності та виявлення областей для покращення [10], [15];

- засоби оптимізації маршрутів: алгоритми програми Routific дозволяють оптимізувати маршрути для водіїв чи команд доставки, зменшуючи витрати на паливе та час на пересування [10], [15];

- можливість побудови оптимальних маршрутів: програмне забезпечення Routific допомагає побудувати оптимальні маршрути, враховуючи різні фактори, такі як час доставки, обмеження швидкості, обсяги роботи та інші фактори, що впливають на ефективність маршруту [10], [15];

- засоби відстеження в реальному часі. Програмне забезпечення Routific надає можливість відстежувати рух водіїв у реальному часі, забезпечує відстеження статусу та локації доставок в реальному часі для керівництва та клієнтів [10], [15];

- засоби взаємодії з клієнтами: програма забезпечує сповіщення клієнтам про статус їхніх замовлень та приблизний час доставки. Є можливість надсилати сповіщення через текстові повідомлення або електронну пошту [10], [15];

- гнучкі тарифні плани: програмне забезпечення Routific має різні тарифні плани, адаптовані до розміру та потреб команди доставки. Це дозволяє обирати оптимальний план в залежності від розміру та потреб бізнесу [10], [15];

- мобільний застосунок: програмний продукт має зручний мобільний застосунок для роботи на пристроях під управлінням різних операційних систем [10], [15];

- засоби підтримки користувачів: Routific надає підтримку користувачам, що може включати онлайн-поради, документацію, чат-підтримку або інші ресурси для вирішення питань користувачів [10], [15].

Серед недоліків програмного забезпечення Routific можна відзначити складність використання, зокрема, складність налаштування програми та роботу з деякими її функціями. Крім того, ця програма вимагає оплати, що ускладнює її використання для організацій з невеликим бюджетом. Іншим недоліком є недостатня інтеграція з іншими платформами [10], [15].

Програмне забезпечення MyRouteOnline (рис. 1.3) є інструментом для пошуку оптимальних шляхів та планування маршрутів, що дозволяє оптимізувати до трьохста адрес для команд доставки. Адреси можна імпортувати у програму MyRouteOnline з файлів Excel або CSV. У більш розширених планах інформацію можна зберегти для подальшого використання [10], [16].

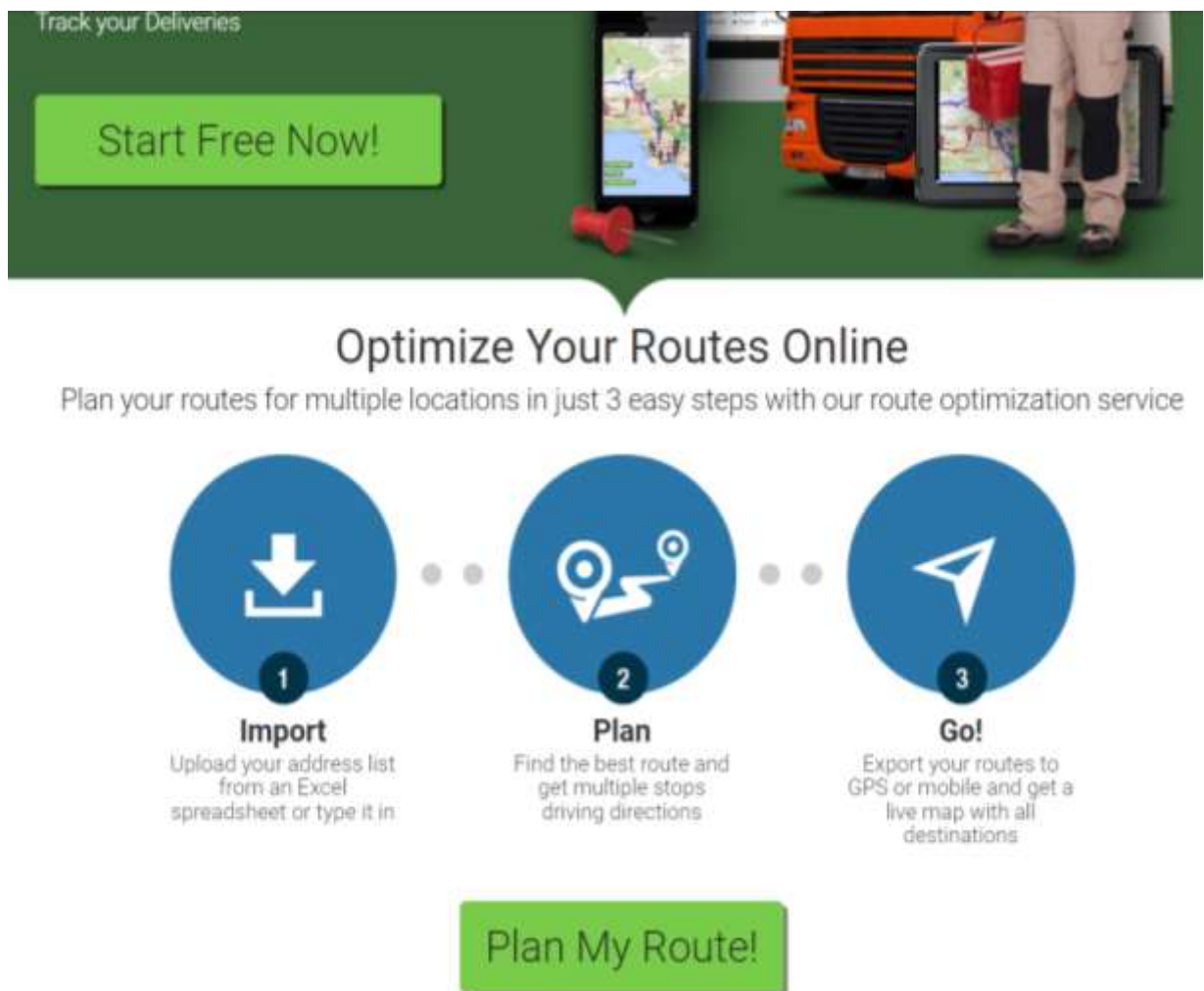


Рисунок 1.3 – Програмне забезпечення MyRouteOnline [10], [16]

Планування маршрутів у MyRouteOnline відбувається трьома способами [10], [16]:

- мінімізація відстані: оптимізація маршрутів для проїзду найкоротшою відстанню;
- мінімізація часу: створення найшвидших маршрутів, які забирають мінімум часу в дорозі;
- балансування відстані і часу: створення маршрутів, що поєднують переваги перших двох [10], [16].

Після оптимізації маршрутів, їх можна експортувати у форматі Excel для друку, надсилати водіям по електронній пошті або експортувати на пристрої TomTom чи Garmin. Застосунок MyRouteOnline дає водіям змогу увійти до системи та отримати свої маршрути [10], [16].

Програмне забезпечення MyRouteOnline є інструментом, спрямованим виключно на оптимізацію маршрутів, тому він не має додаткових функцій, які можуть бути потрібні командам доставки. Щодо ціноутворення MyRouteOnline, воно базується на кредитах, де кожна адреса коштує один кредит (адресу можна використовувати без додаткових кредитів протягом 30 днів). Однак, різні плани мають різні функції, наприклад, базовий план не включає можливість «Зберегти та перезавантажити», яка спрощує планування регулярних маршрутів [10], [16].

Програмне забезпечення MyRouteOnline дозволяє планувати шляхи команд, які займаються доставкою останньою милі. Програмне забезпечення MyRouteOnline надає можливість отримання доступу до розширеної аналітики доставок [10], [16]. Ця можливість дозволяє відстежувати та аналізувати показники, такі як загальна кількість зупинок, відсоток запізнень та середня вартість доставки. Ці показники є ключовими для оцінки продуктивності та визначення напрямків для покращень [10], [16]. Використовуючи програмне забезпечення MyRouteOnline для детального аналізу груп, компанії, що займаються доставкою, можуть визначити оптимальні методи доставки,

виявити сильні сторони та визначити сфери, які потребують уваги або вдосконалення [10], [16].

Використання програмного забезпечення MyRouteOnline для маршрутизації надає значні переваги у плануванні та виконанні маршрутів доставки. Програмне забезпечення MyRouteOnline дозволяє враховувати те, що маршрути можуть змінюватись і включати багато факторів, які ускладнюють процес планування [10], [16].

Програмне забезпечення MyRouteOnline для оптимізації маршрутів дозволяє полегшити процес доставки та мінімізувати витрат часу на планування. Програмне забезпечення MyRouteOnline пропонує розширені можливості, такі як реальний час відстеження, сповіщення для клієнтів, підтвердження доставки та аналіз [10], [16].

Програмне забезпечення MyRouteOnline планування маршрутів забезпечує високу налаштовуваність та гнучкість, що дозволяє підприємствам адаптуватися в реальному часі, інтегруватися з іншими платформами та програмним забезпеченням, а також налаштовувати функціонал відповідно до конкретних операційних потреб [10], [16].

Серед особливостей програмного забезпечення MyRouteOnline для пошуку оптимального шляху можна виділити такі [10], [16].

- засоби пошуку оптимального шляху. Програмне забезпечення MyRouteOnline дозволяє оптимізувати маршрути за різними критеріями, такими як мінімізація відстані, мінімізація часу або збалансоване поєднання обох [10], [16];

- імпорт адрес: програмне забезпечення MyRouteOnline дозволяє імпортувати адреси для планування маршрутів за допомогою файлів Excel або CSV [10], [16];

- експорт маршрутів: після планування маршруту дозволяє експортувати його у форматі Excel, надсилати водіям електронною поштою чи експортувати на пристрої TomTom або Garmin [10], [16];

– мобільний доступ: програмне забезпечення MyRouteOnline має мобільний застосунок, який працює як на пристроях Android, так і на iOS, що дає змогу водіям отримувати маршрути з будь-якого пристрою [10], [16];

– ціноутворення на основі кредитів: робота програмного забезпечення базується на кредитній системі, де одна унікальна адреса дорівнює одному кредиту [10], [16];

– різні рівні доступу: програмне забезпечення MyRouteOnline пропонує різні плани з різними функціями, що відповідають різним потребам користувачів. Наприклад, деякі функції можуть бути доступні тільки у розширених планах [10], [16];

– засоби взаємодії з клієнтами: програма забезпечує сповіщення клієнтам про статус їхніх замовлень та приблизний час доставки. Є можливість надсилати сповіщення через текстові повідомлення або електронну пошту [10], [16];

– засоби підтримки взаємодії клієнта та водія. Клієнти можуть безпосередньо спілкуватися з водіями через сервіс обміну повідомленнями. Це забезпечує зручну комунікацію клієнтів із водіями для уточнення деталей або повідомлення про спеціальні інструкції щодо доставки [10], [16];

– засоби відстеження в реальному часі. Програмне забезпечення MyRouteOnline надає можливість відстежувати рух водіїв у реальному часі, забезпечує відстеження статусу та локації доставок в реальному часі для керівництва та клієнтів [10], [16];

– спрощене планування регулярних маршрутів: Вищі плани можуть включати додаткові функції, які спрощують планування регулярних маршрутів, наприклад, можливість зберігати та перезавантажувати маршрути для подальшого використання [10], [16].

Серед недоліків програмного забезпечення MyRouteOnline можна відзначити відсутність розширених засобів аналітики. Іншим недоліком є недостатня інтеграція з іншими платформами [10], [16].

Порівняльну характеристику програмного забезпечення для пошуку найкоротшого шляху наведено у таблиці 1.1.

Таблиця 1.1 – Порівняння програмного забезпечення для пошуку найкоротшого шляху

Критерій порівняння	Circuit for Teams [14]	Routific [15]	MyRouteOnline [16]
Зручність планування маршрутів	+	+	+-
Інтеграція з іншими програмними засобами	+	+-	-
Сертифікати доставки	+	-	-
Мобільна версія	+	+	-
Розширені засоби аналітики даних	+	+-	-
Безкоштовність	-	-	+-

За результатами проведеного аналізу можна зробити висновок, що у наш час існує досить багато програмних засобів для пошуку найкоротшого шляху. Проте деякі програми засоби для пошуку оптимального шляху можуть використовувати неефективні алгоритми оптимального пошуку, які базуються на пошуку по всіх можливих шляхах та, відповідно, є досить складними для реалізації та використання (вимагають великої кількості часу та апаратних ресурсів). Деякі алгоритми, що використовуються у відповідному програмному забезпеченні, можуть не мати можливості урахувати деякі особливості, такі як зміна умов транспортного потоку або різних обмежень, що можуть виникати у реальних умовах. Інші програмні засоби можуть вимагати значної кількості пам'яті для зберігання великих транспортних мереж або матриць шляхів, особливо у складних системах. Тому актуальною є

розробка програмного забезпечення для пошуку найкоротшого шляху на основі мультиагентного підходу.

При розробці програмного забезпечення для пошуку найкоротшого шляху необхідно забезпечити такі функціональні вимоги:

- можливість пошуку найкоротшого шляху;
- можливість завантаження вхідних даних з файлу;
- можливість генерування графу з файлу даних;
- можливість додавання та видалення ребер графу, що відображає умову задачі пошуку найкоротшого шляху, у випадку ручного введення даних;
- можливість задавання параметрів задачі та методу її розв'язання;
- візуалізація знайденого результату.

1.3 Висновки за розділом 1

Визначено, що програмні засоби для пошуку найкоротших маршрутів відіграють важливу роль у різних галузях, таких як транспорт, логістика, мережеве планування, інформаційні технології і багато інших. Вони набувають особливої важливості в сучасному світі, де управління маршрутами, оптимізація доставки та аналіз мережі стають критичними. Показано, що мета пошуку найкоротшого шляху полягає в визначенні найбільш оптимального шляху між двома точками в графі, де сума ваг ребер, утворюючи цей шлях, є найменшою. Відзначено, що розв'язання задач пошуку оптимального шляху зводиться до знаходження найкоротшого шляху між вершинами графа. Показано, що існують різні алгоритми вирішення задачі пошуку оптимального шляху. Проаналізовано методи розв'язання цієї задачі. Визначено, що для пошуку найбільш важливих шляхів широко використовуються методи на основі алгоритму пошуку найкоротших шляхів Дейкстри, у таких методах спочатку визначаються найбільш важливі шляхи від стартової точки до самої себе, після чого відображаються найбільш вагомні шляхи до інших вершин у відповідності до їх часу прибуття. Відзначено, що

пошукові методи на основі теорії графів можуть займати багато часу, особливо в задачах з великою кількістю пунктів призначення. Показано, що ефективним засобом пошуку найкоротшого шляху є застосування мультиагентного підходу, що передбачає використання групи агентів або сутностей, кожен з яких приймає рішення для пошуку найкращого маршруту у великій області або графі. Відзначено, що мультиагентний підхід є особливо корисним у складних системах або задачах, де потрібно враховувати багато факторів та обмежень при пошуку оптимального маршруту.

За результатами проведеного аналізу зроблено висновок, що у наш час існує досить багато програмних засобів для генерації текстової інформації. Проте деякі програмні засоби для пошуку оптимального шляху можуть використовувати неефективні алгоритми оптимального пошуку, які базуються на пошуку по всіх можливих шляхах та, відповідно, є досить складними для реалізації та використання (вимагають великої кількості часу та апаратних ресурсів). Деякі алгоритми, що використовуються у відповідному програмному забезпеченні, можуть не мати можливості урахувати деякі особливості, такі як зміна умов транспортного потоку або різних обмежень, що можуть виникати у реальних умовах. Інші програмні засоби можуть вимагати значної кількості пам'яті для зберігання великих транспортних мереж або матриць шляхів, особливо у складних системах [1]-[7].

Тому актуальною є розробка програмного забезпечення для пошуку найкоротшого шляху на основі мультиагентного підходу.

Сформульовано функціональні вимоги до програмного забезпечення для пошуку найкоротшого шляху.

2 МАТЕРІАЛИ І МЕТОДИ

2.1 Вибір мови програмування

При розробці програмного забезпечення для пошуку найкоротшого шляху було використано мову програмування Java [17], [18].

Java є програмною мовою, яка базується на об'єктно-орієнтованому підході. Програми, написані на Java, зазвичай перетворюються у байт-код для виконання на будь-якій віртуальній машині Java, незалежно від архітектури пристрою [17], [18].

Переваги використання мови програмування Java включають [17], [18]:

- платформена незалежність: Один код може функціонувати на різних операційних системах [17], [18];
- схожість з C++: Синтаксис Java схожий на C++, що полегшує розуміння для програмістів, які володіють C і C++ [17], [18];
- об'єктно-орієнтованість: У Java все є об'єктом, що полегшує розробку складних проєктів [17], [18];
- надійність: Мова Java має заходи безпеки, що дозволяють уникнути багатьох помилок [17], [18];
- велика спільнота користувачів: Є велика кількість форумів і ресурсів, де користувачі обмінюються досвідом [17], [18].

Деякі обмеження мови Java включають обмеження використання масивів та арифметики вказівників, а також автоматичне вивільнення пам'яті. Також у Java заборонено неправильне використання операторів присвоєння та порівняння рівності, що допомагає уникнути плутанини в коді [17], [18].

Основні характеристики мови Java включають в себе динамічні типи даних, автоматичне управління пам'яттю, розширену обробку винятків, можливості роботи з винятками, підтримку багатопотоковості і просту високорівневу структуру даних [17], [18].

У Java код організований у вигляді функцій та класів, які можуть бути об'єднані в програмні модулі. Ці модулі, у свою чергу, можуть бути об'єднані в програмні пакети [17], [18].

Вбудовані типи даних в Java включають в себе логічні значення, рядки, цілі числа будь-якої точності, числа з плаваючою точкою, комплексні числа та інші. Мова також має вбудовані колекції, такі як списки, кортежі, словники та інші структури даних. Усе в Java є об'єктом, включаючи функції, методи, модулі та класи. Додатково, розробник може створювати нові типи даних, створюючи класи або визначаючи їх у програмних модулях розширень, таких як на мові C [17], [18].

Обґрунтування вибору мови програмування для пошуку найкоротшого шляху наведено у таблиці 2.1.

Таблиця 2.1 – Обґрунтування вибору мови програмування для пошуку найкоротшого шляху

Критерій порівняння мов програмування	Мова програмування		
	Java	Python	C++
Зручність розробки програмного забезпечення для пошуку найкоротшого шляху	+	–	–
Робота з текстовими даними	+	+–	+
Підтримка багатопоточності	+	+–	+
Наявність великої кількості програмних модулів для розв'язання оптимізаційних завдань	+	+–	+–
Засоби безпеки даних	+	+	+–

Отже, для реалізації програмного забезпечення для пошуку найкоротшого шляху обрано мову програмування Java, яка є платформо незалежною, об'єктно-орієнтованою та надійною мовою, а також має велику спільноту розробників та готових програмних модулів для розв'язання оптимізаційних завдань.

2.2 Вибір середовища розробки для пошуку найкоротшого шляху

В якості середовища розробки для створення програмного забезпечення для пошуку найкоротшого шляху було обрано IntelliJ IDEA [19], [20].

IntelliJ IDEA, як комерційне інтегроване середовище розробки від JetBrains для Java, володіє рядом переваг. Кожен компонент IntelliJ IDEA максимально оптимізований для забезпечення високої ефективності розробки. Розумний редактор коду поєднується з ергономічним дизайном для комфортної та приємної роботи над проєктами [19], [20].

Інтегроване середовище розробки IntelliJ IDEA забезпечує глибоке розуміння коду та швидку розробку завдяки інтелектуальному автозавершенню, аналізу коду у реальному часі та надійному рефакторингу. Не потрібно витрачати час на пошук та встановлення плагінів для інтеграції з системою контролю версій чи підтримки різних мов та фреймворків [19], [20].

IntelliJ IDEA підтримує так званий розумний код, що надає елементи коду відповідно до поточного контексту, не тільки пропонує назви класів, методів і полів [19], [20].

Інтегроване середовище розробки IntelliJ IDEA підтримує роботи з різними мовами, включаючи розуміння роботи з SQL, JPQL, HTML, JavaScript та іншими мовами, забезпечуючи інтелектуальну підтримку у роботі з ними. Також забезпечується автоматизація рутинних процесів розробки, дозволяючи розробникам зосередитися на основній меті [19], [20].

Ергономічна атмосфера, де функціонал та дизайн інтегрованого середовища розробки IntelliJ IDEA спрямовані на уникнення відволікань та максимізацію продуктивності розробника, мінімізуючи втрату контексту [19], [20]. IntelliJ IDEA є ефективним інструментом для програмістів, які працюють з мовою програмування Java. Він не тільки надає корисні функції, але також надає інструменти аналізу коду та оптимізації, які допомагають підвищити продуктивність процесу розробки програмного забезпечення [19], [20].

Обґрунтування вибору середовища розробки для створення програмного забезпечення для пошуку найкоротшого шляху наведено у таблиці 2.2.

Таблиця 2.2 – Обґрунтування вибору середовища розробки для створення програмного забезпечення для пошуку найкоротшого шляху

Критерій порівняння середовищ розробки	Середовище розробки		
	IntelliJ IDEA	Visual Studio	Eclipse
Зручність розробки на мові Java	+	+—	+
Інструменти для роботи з кодом	+	+	+—
Функціонал для рефакторингу	+	+	+—
Зручність для створення ПЗ для пошуку найкоротшого шляху	+	—	—
Рівень підтримки розробників	+	+	+—

Таким чином, для створення програмного забезпечення для пошуку найкоротшого шляху обрано середовище розробки IntelliJ IDEA, що являє собою ефективний інструмент для створення програм, написаних на мові програмування Java та забезпечує інтелектуальну підтримку процесу розробки програмного забезпечення.

2.3 Інтелектуальні методи для пошуку оптимального шляху

Для пошуку оптимального шляху ефективно можуть використовуватися методи дискретної оптимізації на основі еволюційного та мультиагентного підходів [21]-[23].

Еволюційне програмування — це тип еволюційного алгоритму, який характеризується незмінними структурами даних агентів, які демонструють «розумну поведінку», прогножуючи стани. Еволюційне програмування відноситься до методів штучного інтелекту, сутність яких полягає у моделюванні еволюційного процесу як практики розумної поведінки та прогнозування явищ навколишнього середовища [21]-[23]. Методи еволюційного програмування працюють з кінцевими автоматами, графічними моделями, які використовуються для опису поведінки або програмного та апаратного забезпечення, моделюють поведінку індивіду, який у популяції вирішує проблеми. Ці скінченні автомати спрямовані на використання символів цифрового порядку, які все підходять для вирішення проблеми в міру їх розвитку [21]-[23].

В еволюційному програмуванні популяція скінченних автоматів оцінюється в експериментальному середовищі, де задано фіксований рядок. Для кожного батьківського автомата у процесі роботи еволюційного алгоритму виконується процес оцінки значення цієї функції вихідних втрат шляхом порівняння кожного наступного символу з відповідним передбачуваним вихідним символом автомата. Після останнього прогнозу розраховується здійсненність такого автомата або програми [21]-[23].

Автомати-нащадки створюють і оцінюють батьківські автомати за допомогою випадкових мутацій. Для наступного покоління відбираються найкращі машини, і процес повторюється. Якщо потрібно передбачити нові символи, до старих спостережень додаються нові [21]-[23]. В еволюційному програмуванні, на відміну від генетичних алгоритмів, еволюція моделюється більше як адаптивна поведінка особин у популяції або виду, а не як процес генетичної адаптації [21]-[23].

Метод мультиагентної оптимізації на основі мурашиного алгоритму є багаторівневим алгоритмом для пошуку наближених розв'язків задачі пошуку оптимального шляху та для виконання таких завдань, як пошук шляхів у графі. Ідея алгоритму походить від поведінки агенту-мурахи у мурашнику в довільних напрямках. Знайшовши їжу, він повертається в мурашник, залишаючи за собою феромонний слід [21]-[23].

Ці феромони приваблюють інших сусідніх агентів-комах, які віддають перевагу цьому шляху. У мурашнику вони також виділяють свої феромони і зміцнюють слід [21]-[23]. Якщо є два шляхи, більше мурах витрачають час на коротший шлях, ніж на довший шлях, і коротший шлях є більш вигідним. Випаровування феромонів викликає поступове зникнення довгих неефективних шляхів [21]-[23].

Основною ідеєю мурашиного алгоритму є оптимізація, заснована на непрямому спілкуванні між автономними агентами. Мурашиний алгоритм моделює багатоагентну систему. Його представників у майбутньому називатимуть мурахами. Кожна мураха зберігає в пам'яті список вузлів, які вона перетнула. Цей список називається списком табу або просто мурашиною пам'яттю. Коли ви вибираєте вузли для наступного кроку, мураха «запам'ятовує» пройдені вузли і не вважає їх можливими для переміщення. На кожному кроці новий вузол заповнює список обмежень і очищається перед наступною ітерацією алгоритму, тобто до того, як агент-мураха знову вибере шлях [21]-[23].

На додаток до забороненого списку, мураха контролює вибір вузлів для транспортування за допомогою «привабливості» країв, які він може пройти. Привабливість залежить, з одного боку, від відстані між вузлами (тобто товщини пляжу), а з іншого боку, від слідів феромонів, залишених на пляжі комахами, які пройшли повз раніше. Насправді, на відміну від вагових коефіцієнтів, які є статичними, сліди феромонів оновлюються з кожною ітерацією алгоритму: як і природа, сліди з часом зникають, а повз мурахи роблять навпаки. , це підтверджується [21]-[23].

Припустимо, що розташування агенту-мурашки є повнозв'язним неорієнтованим графом. Кожне ребро має вагу, задану як відстань між двома залученими вершинами. Оскільки графік двосторонній, мураха може рухатися то в одну, то в іншу сторону [21]-[23].

Імовірність розміщення краю на шляху окремої мурашки пропорційна кількості феромонів на цьому краю, а кількість осідлого феромону пропорційна довжині шляху. Чим коротший маршрут, тим більше феромонів з боків, тому більше мурах включає їх у свій маршрут синтезу. Моделювання таких підходів, засноване виключно на позитивному зворотному зв'язку, призводить до передчасної конвергенції: більшість мурах рухаються локально оптимальними маршрутами [21]-[23].

Цього можна уникнути, моделюючи негативний зворотний зв'язок як випаровування феромонів. Крім того, швидке висихання феромону призводить до втрати пам'яті колонії та забування хороших розчинів, і навпаки, збільшення часу сушіння може призвести до стабільного локального оптимального рішення [21]-[23].

Мураха являє собою програмний агент, член великої колонії, який використовується для вирішення конкретних завдань. Антті має просте правило для вибору шляхів у графі. Він підтримує список вузлів, які він відвідав, і може пройти через кожен вузол лише один раз [21]-[23].

Вузли в списку «Поточний візит» упорядковані за відвідуваннями мурахи. Пізніше список використовується для визначення довжини шляху між

вузлами. Справжня мураха залишає феромони, рухаючись по доріжці. В мурашиному алгоритмі агент посиляє феромони на краю графа після завершення своєї подорожі [21]-[23].

Початкова точка для розміщення мурах залежить від обмежень, встановлених умовами завдання, оскільки розміщення мурах є вирішальним у кожному завданні. Усі вони розміщені в одному місці або в різних місцях, повторюються чи ні [21]-[23].

При цьому встановлюється початковий рівень феромонів. Він ініціалізується невеликим позитивним числом, так що ймовірність переходу до наступного піку на початковій фазі дорівнює нулю [21]-[23].

Рух мурашки базується на простому ймовірнісному рівнянні. Оскільки агент-мураха переміщується тільки до вузлів, які ще не були відвідані, ймовірність обчислюється лише для ребер, які призводять до таких невідведаних вузлів. Шлях мурахи розглядається, коли вона пройшла всі вузли на графіку. Будь-які цикли відсутні, оскільки алгоритм містить список табу. Після цього можна обчислити довжину шляху, яка є сумою довжини всіх ребер, які пройшов мураха. Результатом є визначена концентрація феромону. Короткі шляхи характеризуються великими концентраціями феромону, у той час як довгі шляхи – низькими [21]-[23]. Кожне ребро на початку шляху має однаковий шанс бути вибраним. Процес випаровування феромону застосовується до всіх ребер для поступового вилучення тих, що належать до найгірших шляхів на графіку [21]-[23].

Коефіцієнт оновлення зворотньої траєкторії використовується для розпаровування феромону. Під час проходження мурахою свого шляху краї оновлюються в залежності від довжини цього шляху, феромон випаровується з усіх країв, і алгоритм повторюється з початку. Список заборонених шляхів очищується, а довжина шляху скидається. Мурахам дозволяється пересуватися по діаграмі. Цю процедуру можна повторювати для кількох послідовних підходів або кількох циклів, поки не буде виявлено стабільні зміни. Після цього обирається найкращий шлях, який є оптимальним рішенням [21]-[23].

Алгоритми мурашиної колонії успішно використовуються для вирішення складних завдань оптимізації. Основна мета розв'язання таких складних задач оптимізації полягає у пошуку та визначенні найбільш прийняттого рішення для оптимізації (мінімізації або максимізації) цільової функції (вартість, точність, час, відстань тощо) серед кількох можливих варіантів. Ці задачі виникають у багатьох галузях, включаючи бізнес, дизайн, виробництво тощо. Дослідження показали, що метод мурашиного колонію може дати кращі результати, ніж використання генетичних алгоритмів і нейронних мереж [21]-[23].

Задача пошуку оптимального шляху або комівояжера є однією з ключових задач комбінаторної оптимізації в графових моделях, де потрібні зусилля для знаходження ефективного рішення. Цільова функція цієї задачі полягає у створенні циклу, який охоплює всі вершини графа і має найкоротшу довжину. Ефективність комівояжера виявляється через агентів (мурах), що виділяють феромони по шляху, і вища концентрація феромонів на коротших маршрутах. Висока концентрація феромонів вказує на найкращий шлях, який мурашка пройшов [21]-[23].

Цей алгоритм вважається ефективним методом вирішення проблеми комівояжера. Робота зосереджена на пошуку найкоротшого маршруту для транспортування. Подальші кроки спрямовані на вирішення ключових завдань у цьому напрямі [21]-[23].

Сценарій передбачає наявність одного транспортного засобу та кількох замовлень від споживачів, кожен з яких має свій встановлений час обслуговування. До відправлення від складу відомо кількість споживачів, їх координати, а також відстані між ними та вартість доставки до кожної точки мережі. Крім того, транспортний засіб має обмеження на вантажопідйомність, яку не можна перевищувати [21]-[23].

2.4 Розробка моделі взаємодії користувача з програмним забезпеченням пошуку найкоротшого шляху

З метою можливості розробки програмного забезпечення для пошуку найкоротшого шляху створимо відповідну модель у вигляді діаграми прецедентів.

При розробленні моделі взаємодії користувача з програмним забезпеченням для пошуку найкоротшого шляху враховуємо те, що користувачі програми мають можливості пошуку найкоротшого шляху, завантаження вхідних даних з файлу, генерування графу з файлу даних, додавання та видалення ребер графу, що відображає умову задачі пошуку найкоротшого шляху, у випадку ручного введення даних, задавання параметрів задачі та методу її розв'язання, візуалізації знайденого результату.

Основний успішний сценарій користувача при роботі з програмним забезпеченням для пошуку найкоротшого шляху можна описати таким чином:

- користувач програмного забезпечення для пошуку найкоротшого шляху запускає програму;
- після запуску програми для пошуку найкоротшого шляху користувачу відображається головна форма;
- користувач програми для пошуку найкоротшого шляху бачить головну форму та завантажує або вводить вхідні дані;
- користувач вводить параметри методу пошуку оптимального шляху;
- користувач здійснює пошук найкоротшого шляху;
- користувач бачить результат роботи програми, зокрема оптимальний шлях;
- користувач завершує роботу з програмним забезпеченням для пошуку найкоротшого шляху.

Враховуючи сценарії роботи користувачів при роботі з програмним забезпеченням для пошуку найкоротшого шляху, модель взаємодії

користувача з програмним забезпеченням для розв'язання задачі пошуку найкоротшого шляху, зображено на рис. 2.1.

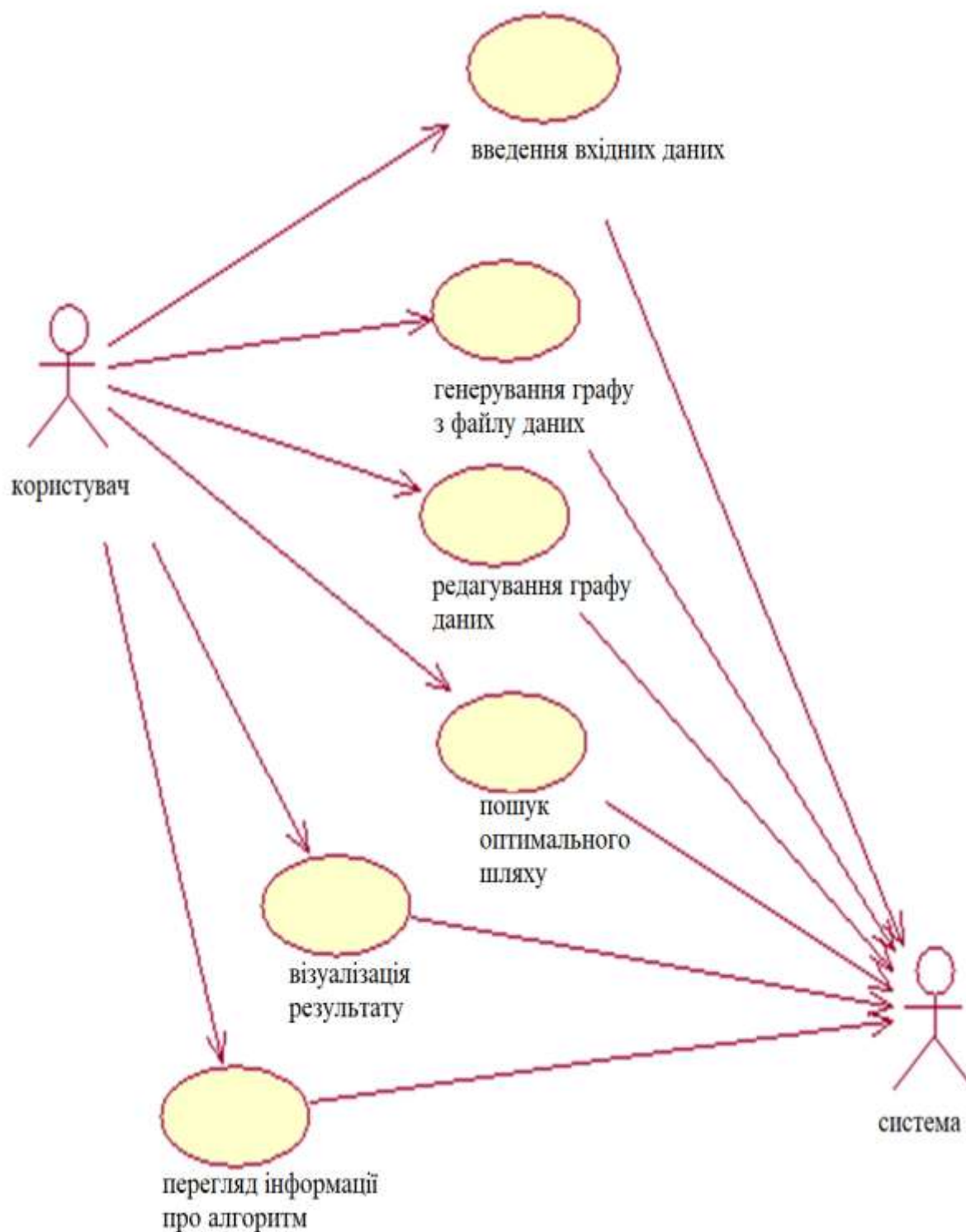


Рисунок 2.1 – Модель взаємодії користувача з програмним забезпеченням для пошуку найкоротшого шляху

Таким чином, створено модель взаємодії користувача з програмним забезпеченням для пошуку найкоротшого шляху, яка подана у вигляді діаграми прецедентів, та дозволяє розроблювати програмне забезпечення для пошуку найкоротшого шляху.

2.5 Висновки за розділом 2

Для реалізації програмного забезпечення для розв’язання завдання пошуку найкоротшого шляху обрано мову програмування Java, яка є платформо незалежною, об’єктно-орієнтованою та надійною мовою, а також має велику спільноту розробників та готових програмних модулів для розв’язання оптимізаційних завдань.

Для створення програмного забезпечення для розв’язання задачі пошуку найкоротшого шляху обрано середовище розробки IntelliJ IDEA, що являє собою ефективний інструмент для створення програм, написаних на мові програмування Java та забезпечує інтелектуальну підтримку процесу розробки програмного забезпечення.

Створено модель взаємодії користувача з програмним забезпеченням для пошуку найкоротшого шляху, яка подана у вигляді діаграми прецедентів, та дозволяє розроблювати програмне забезпечення для пошуку найкоротшого шляху.

3 ОПИС ПРОГРАМИ

3.1 Структура программного обеспечения для поиска наикоротшого шляху

Структуру программного обеспечения для поиска наикоротшого шляху подано на рис. 3.1.

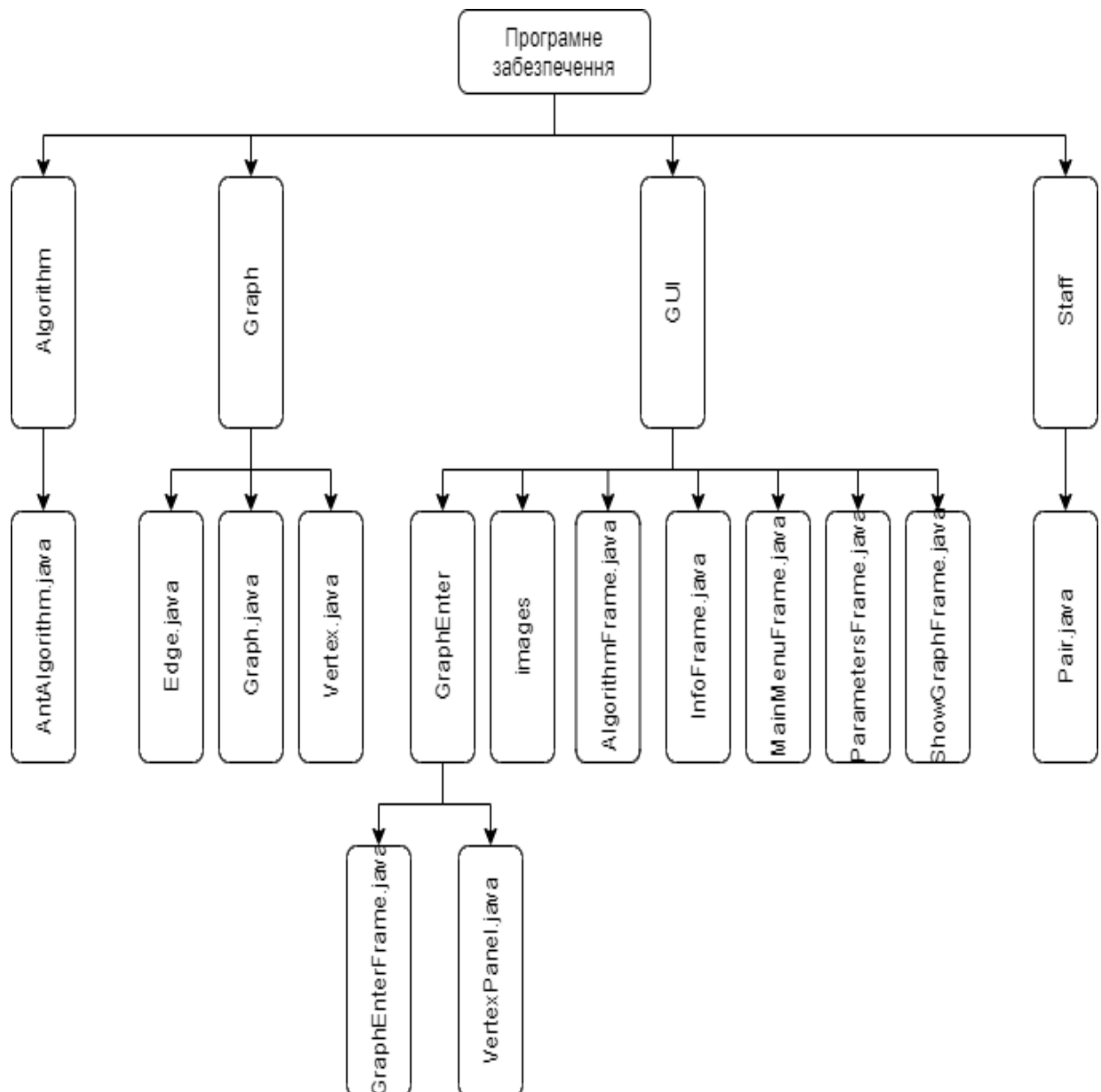


Рисунок 3.1 – Структура программного обеспечения для поиска наикоротшого шляху

Як видно з рисунку, програмне забезпечення для пошуку найкоротшого шляху складається з таких компонентів:

- каталог Algorithm – містить клас, який описує алгоритм мурашиних колоній;
- каталог Graph – містить класи для зберігання даних, необхідних для побудови графу;
- каталог GUI – містить підкаталоги і файли необхідні для розробки графічного інтерфейсу програмного забезпечення;
- підкаталог GarphEnter – містить класи, які дозволяють користувачу створювати граф шляхом інтерактивної взаємодії;
- підкаталог image – містить файли зображень, необхідні для забезпечення гарного вигляду програмного забезпечення;
- каталог Staff – містить клас для створення дерева графу.

3.2 Функціонування програмного забезпечення для пошуку найкоротшого шляху

Схема функціонування програмного забезпечення для пошуку найкоротшого шляху наведена на рис. 3.2.

Як видно з рис. 3.2 після запуску програми користувачу буде доступне головне меню, де можна обрати опцію «Параметри» або «Інформація про алгоритм». В розділі «Параметри» можна ввести граф та додаткові параметри для подальшої роботи. Тут же є можливість відображення сформованого графу. У вкладці «Інформація про алгоритм» доступно ознайомитися з основною інформацією про алгоритм мурашиних колоній. Після створення графу для подальшої роботи важливо повернутися до головного меню та запустити алгоритм. Тут користувачеві буде необхідно вказати початкову та кінцеву вершини. Потім можна вручну керувати процесом алгоритму чи запустити автоматичне виконання.

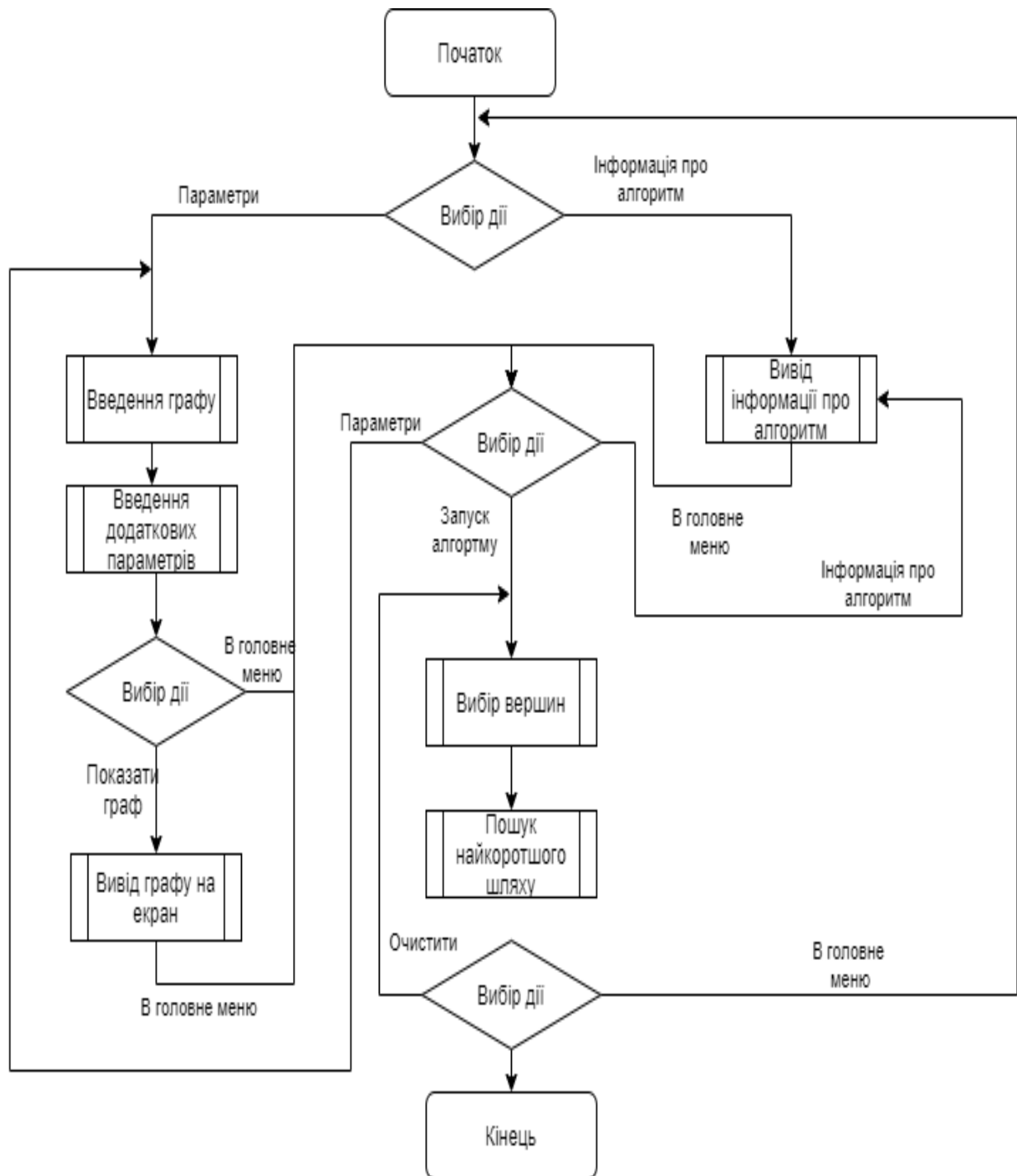


Рисунок 3.2 – Схема функціонування програмного забезпечення для пошуку найкоротшого шляху

Після завершення роботи алгоритму буде відображено знайдений найкоротший шлях від початкової до кінцевої вершини.

Усі описані кроки можна повторювати неодноразово, надалі формуючи нові графи та знаходячи нові шляхи.

3.3 Основні рішення щодо розробки програмного забезпечення для генерації текстової інформації

Розглянемо основні класи і методи, які було розроблено для функціонування програмного забезпечення.

Каталог Algorithm містить клас MurahAlgrt.java. Даний клас реалізує виконання алгоритму мурашиних колоній. Він містить наступні методи:

- MurahAlgrt(Grph grph_, double grd_, double evprtSp_, int nmberndMurahs_, int nmbrMurahs_, Pair<Integer, Integer> pth);
- boolean fnshd();
- int getCnt();
- List<Integer> stp();
- List<Integer> avtalg();
- List<Integer> fndPth().

Каталог Grph містить класи Ed, Grph, Vertex.

Клас Ed описує сутність «Ребро» графу, містить метод-конструктор `public Ed(int wgt_, int frstNd_, int scndNd_)`.

Клас Grph зберігає дані про граф, містить наступні методи:

- Grph();
- Grph(Grph g);
- boolean isNull();
- isCreated();
- void deleteGrph();
- void createMatrix(int n);
- void createVerticesAndEds();
- void makeWarshallAlgorithm();
- void generateGrph(int nmbrVertices, int linksPercent, int leftBound, int rightBound);
- boolean createGrphFromFile(Scanner input);

- boolean isNullRow(int[][] mm, int index);
- void removePhrmn();
- void createMatrixFromEdsAndVertices().

Клас `Vertex` описує сутність «Вершина» графу, містить метод-конструктор `public Vertex(int x, int y, Character letter_)`.

Каталог `GUI` містить підпапки `GrphEnter` і `images`, а також класи `AlgorithmFrame`, `InfoFrame`, `MainMenuFrame`, `ParametersFrame`, `ShowGrphFrame`.

Клас `AlgorithmFrame` призначено для відмальовки вікна «Запуск алгоритму». Він містить наступні методи:

- `AlgorithmFrame(MainMenuFrame mainMenuFrame_, Grph grph_, Pair<Double, Double> params_, Pair<Integer, Integer> murahParams_);`
- `GrphPanel();`
- `void paintComponent(Graphics g);`
- `void disableComboBoxes();`
- `void enableComboBoxes();`
- `void setAlgorithm();`
- `void showPthOfMurah(List<Integer> pthOfMurah);`
- `void showMurah(List<Integer> pthOfMurah);`
- `void actionPerformed(ActionEvent e).`

Клас `InfoFrame` призначено для відмальовки вікна «Інформація про алгоритм», він містить наступні функції:

- `InfoFrame(MainMenuFrame mainMenuFrame_);`
- `void actionPerformed(ActionEvent e).`

Клас `MainMenuFrame` призначено для відмальовки вікна головного меню програми, він містить такі функції:

- `public MainMenuFrame();`
- `void actionPerformed(ActionEvent e);`
- `static void main(String[] args).`

Клас ParametersFrame призначено для відмальовки вікна «Параметри», він містить наступні методи:

- ParametersFrame(MainMenuFrame mainMenuFrame_, Grph grph_, Pair<Double, Double> params_, Pair<Integer, Integer> murahParams_);
- void disableEnterFromFileGrph();
- void disableGenerationGrph();
- void disableHandEnterGrph();
- void enableEnterFromFileGrph();
- void enableGenerationGrph();
- void enableHandEnterGrph();
- void stateChanged(ChangeEvent e);
- void propertyChange(PropertyChangeEvent e);
- void actionPerformed(ActionEvent e).

Клас ShowGrphFrame призначено для відмальовки вікна «Показати граф», він містить наступні методи:

- ShowGrphFrame(ParametersFrame parametersFrame_, Grph grph_);
- void paintComponent(Graphics g);
- void actionPerformed(ActionEvent e).

Підкаталог GrphEnter містить класи GrphEnterFrame і VertexPanel.

Клас GrphEnterFrame призначено для відмальовки вікна «Введення графу», він містить наступні методи:

- GrphEnterFrame(ParametersFrame parametersFrame_, Grph grph_);
- void saveGrph();
- void mouseEntered(MouseEvent e);
- void mouseMoved(MouseEvent e);
- void mouseExited(MouseEvent e);
- void mousePressed(MouseEvent e);
- void mouseReleased(MouseEvent e);
- void mouseClicked(MouseEvent e);

- void mouseDragged(MouseEvent e);
- void actionPerformed(ActionEvent e).

Клас VertexPanel призначено для відмальовки вершин графу при інтерактивному введенні графу, він містить наступні функції:

- VertexPanel(int x, int y);
- int getX();
- int getY();
- Character getLetter();
- void setX(int x0);
- void setY(int y0);
- void setLetter(char n).

Підкаталог images містить файли зображень необхідні для візуального оформлення програмного забезпечення.

Каталог Staff містить клас Pair. Даний клас призначено для формування пар вершин для побудови ребер графу, він містить метод-конструктор Pair(A first_, B scnd_).

3.4 Проєктування інтерфейсу програмного забезпечення для пошуку найкоротшого шляху

Проєктування інтерфейсу взаємодії користувача з програмним забезпеченням для пошуку найкоротшого шляху з урахуванням вимог технічного завдання.

Згідно технічного завдання програмне забезпечення для пошуку найкоротшого шляху повинно забезпечувати візуалізацію вхідних даних та результату пошуку найкоротшого шляху.

Головну форму програми пошуку найкоротшого шляху наведено на рис. 3.3.

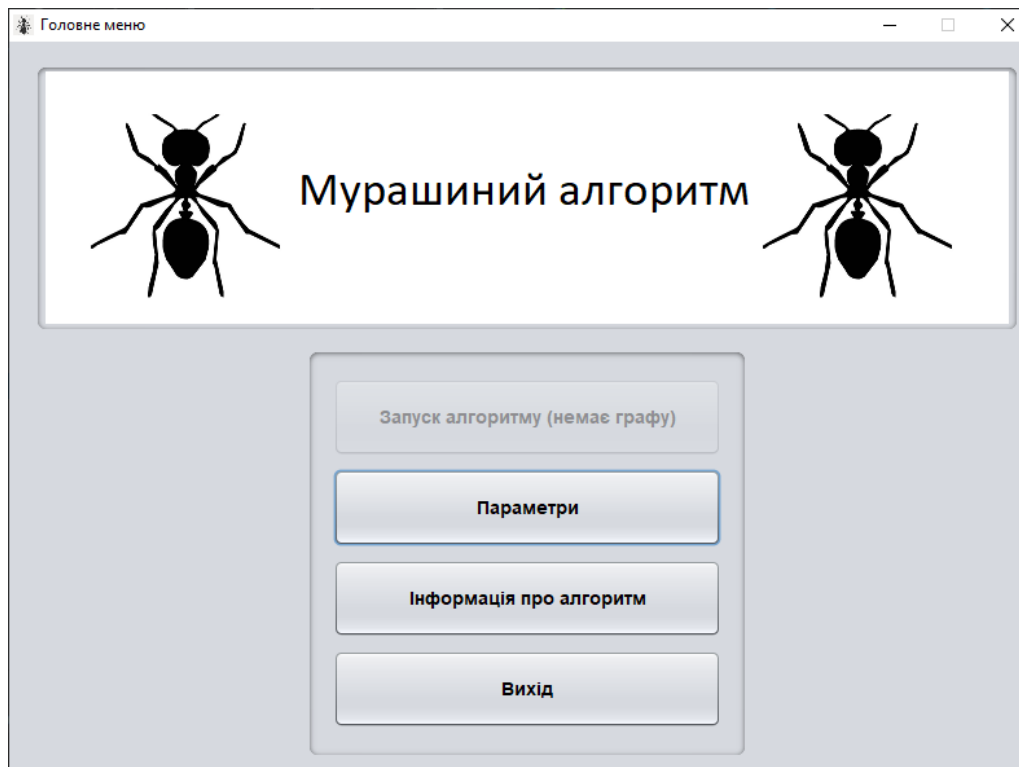


Рисунок 3.3 – Головна форму програми пошуку найкоротшого шляху

Форма з вибором параметрів мультиагентного методу для пошуку найкоротшого шляху наведена на рис. 3.4.

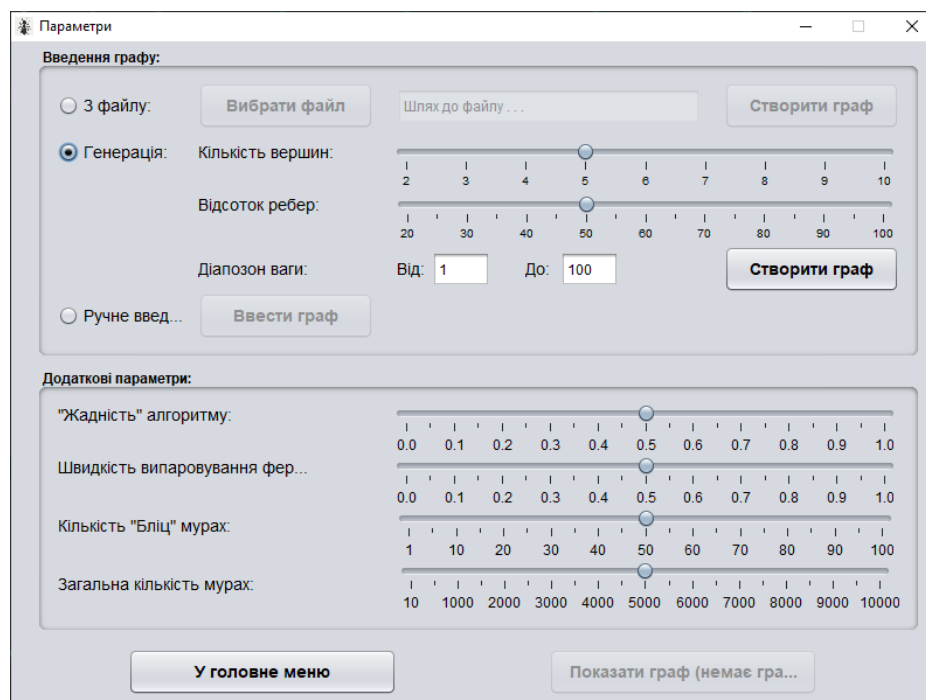


Рисунок 3.4 – Форма з вибором параметрів мультиагентного методу для пошуку найкоротшого шляху

Форма з вікном, де наведено приклад нарисованого вхідного графу, наведена на рис. 3.5.

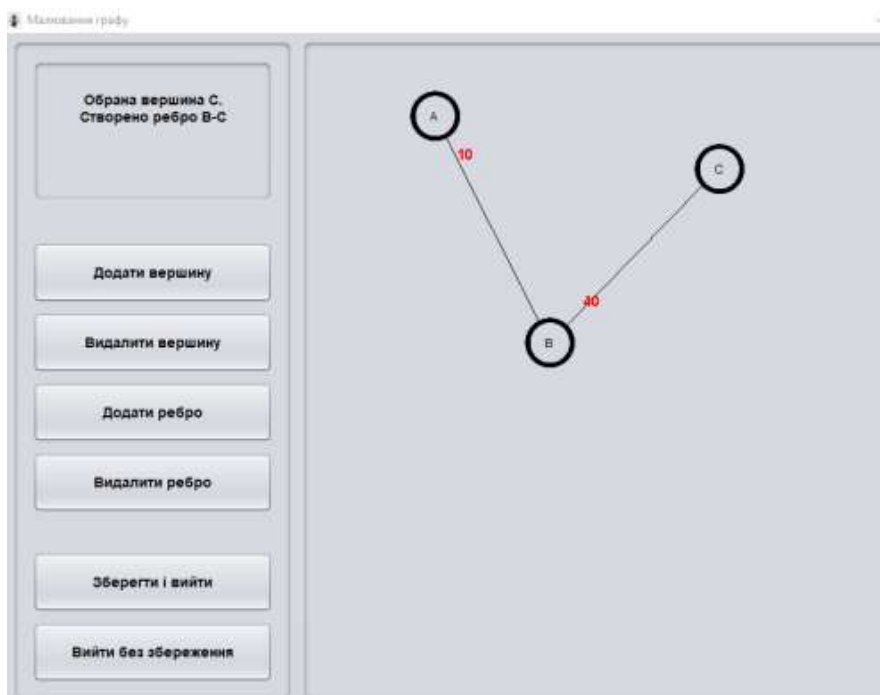


Рисунок 3.5 – Вікно рисування графу

Після створення графу його можна переглянути його у вікні «Показати граф» (рис. 3.6).

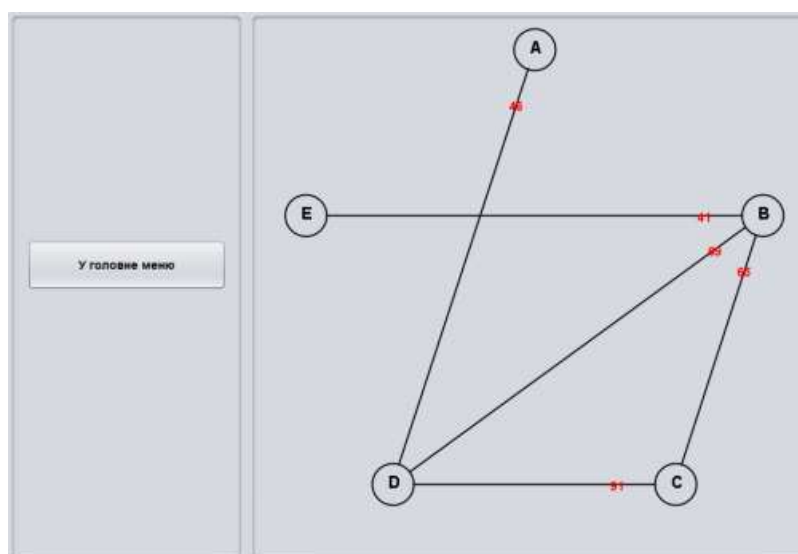


Рисунок 3.6 – Вікно «Показати граф»

Вікно «Запуск алгоритму» наведено на рис. 3.7.

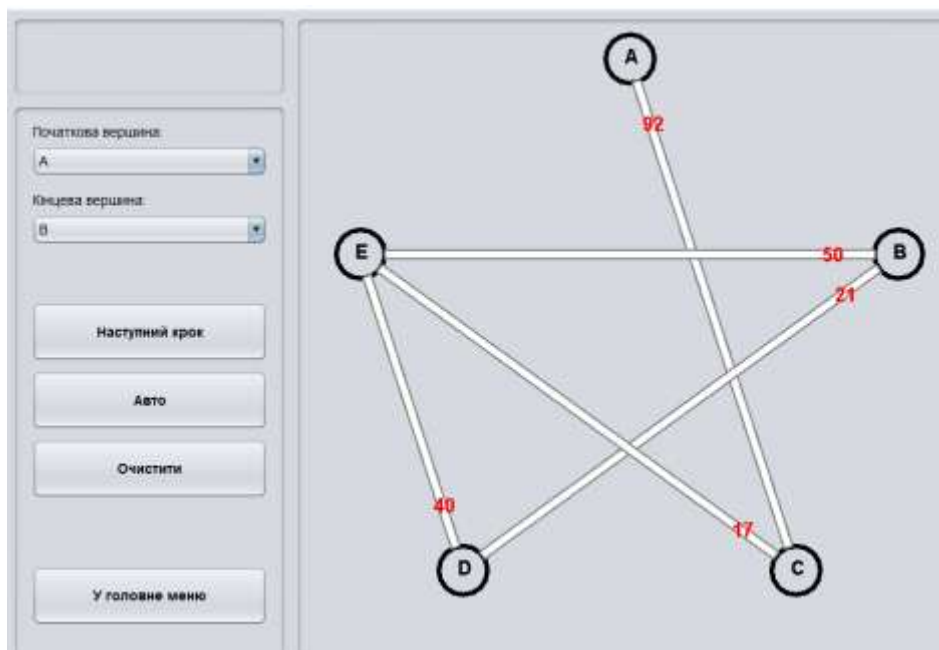


Рисунок 3.7 – Форма «Запуск алгоритму»

Форма з відображенням найкоротшим шляхом від початкової до кінцевої точки наведена на рис. 3.8.

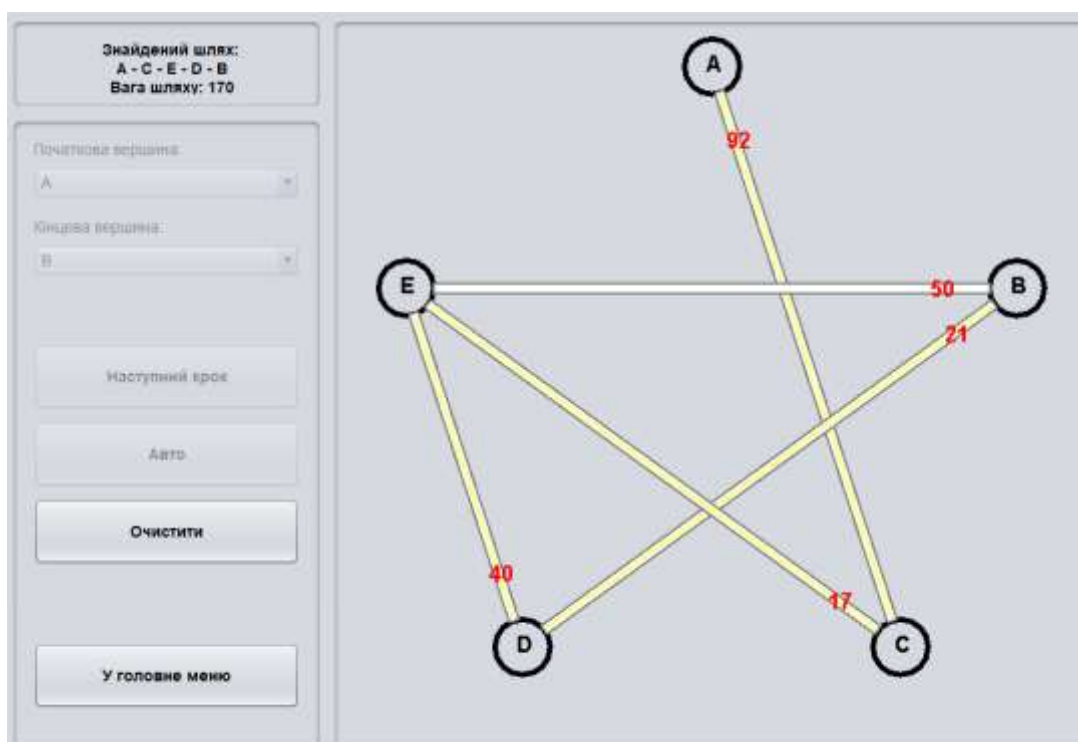


Рисунок 3.8 – Форма з відображенням найкоротшим шляхом

3.5 Висновки за розділом 3

Розроблено програмне забезпечення для розв'язання задачі пошуку найкоротшого шляху. Запропоновано структуру та описано основні модулі програмного забезпечення для пошуку найкоротшого шляху.

Описано функціонування програмного забезпечення для пошуку найкоротшого шляху. Описано особливості реалізації програмного забезпечення для пошуку найкоротшого шляху. Виконано проєктування інтерфейсу взаємодії користувача з програмним забезпеченням для пошуку найкоротшого шляху.

4 ЕКСПЛУАТАЦІЯ, ТЕСТУВАННЯ ТА ЕКСПЕРИМЕНТАЛЬНЕ ДОСЛІДЖЕННЯ ПРОГРАМИ

4.1 Призначення й умови застосування програми

Програма призначена для розв'язання задачі пошуку найкоротшого шляху. Програма написана на мові Java, яка є платформо незалежною, об'єктно-орієнтованою та надійною мовою, а також має велику спільноту розробників та готових програмних модулів для розв'язання оптимізаційних завдань. В якості середовища розробки для створення програми обрано IntelliJ IDEA, що являє собою ефективний інструмент для створення програм, написаних на мові програмування Java та забезпечує інтелектуальну підтримку процесу розробки програмного забезпечення.

Програмне забезпечення для розв'язання задачі пошуку найкоротшого шляху забезпечує виконання таких функцій:

- можливість пошуку найкоротшого шляху;
- можливість завантаження вхідних даних з файлу;
- можливість генерування графу з файлу даних;
- можливість додавання та видалення ребер графу, що відображає умову задачі пошуку найкоротшого шляху, у випадку ручного введення даних;
- можливість задавання параметрів задачі та методу її розв'язання;
- візуалізація знайденого результату.

Для експлуатації програмного забезпечення для розв'язання задачі пошуку найкоротшого шляху необхідно таке апаратне та програмне забезпечення:

- оперативна пам'ять об'ємом 4Гб і більше;
- процесор із тактовою частотою 2,5 GHz і більше;
- вільне місце на диску 250 Мб або більше;
- встановлена операційна система Windows.

Функціональні характеристики розробленого програмного забезпечення для розв'язання задачі пошуку найкоротшого шляху наведено у

технічному завданні (додаток А). Фрагмент тексту програмного забезпечення для розв'язання задачі пошуку найкоротшого шляху наведено у додатку Б.

4.2 Характеристики програми для розв'язання задачі пошуку найкоротшого шляху

Програмне забезпечення для розв'язання задачі пошуку найкоротшого шляху може працювати візуальному режимі. Програмне забезпечення для розв'язання задачі пошуку найкоротшого шляху може використовуватись в операційній системі Windows.

Програмне забезпечення для розв'язання задачі пошуку найкоротшого шляху а забезпечує користувачів усією необхідною інформацією для комфортної та ефективної роботи. Усі кнопки та елементи інтерфейсу програмного забезпечення для розв'язання задачі пошуку найкоротшого шляху мають підписи, що чітко пояснюють їх функції. Крім того, користувач може ознайомитися з описом виконання мультиагентного алгоритму у відповідному вікні програми.

Програмне забезпечення для розв'язання задачі пошуку найкоротшого шляху дозволяє легко переміщатися між різними вікнами та повертатися до попередніх, якщо це потрібно. Якщо користувач програмного забезпечення для розв'язання задачі пошуку найкоротшого шляху закриває поточне вікно, йому автоматично відображається попереднє.

Програмне забезпечення для розв'язання задачі пошуку найкоротшого шляху контролює коректність дій користувача, наприклад, перевіряє розширення завантажених файлів для побудови графу, вимагає введення лише числових даних у відповідні поля, перевіряє на наявність пустих значень та надає можливість виконання лише певних операцій.

Дизайн інтерфейсу програмного забезпечення для розв'язання задачі пошуку найкоротшого шляху є зрозумілим та ергономічним. Простий та легкий для сприйняття дизайн дозволяє користувачам швидко зорієнтуватися,

зменшує час, витрачений на освоєння інтерфейсу, та спрощує роботу з програмним забезпеченням для розв'язання задачі пошуку найкоротшого шляху.

4.3 Інструкція по експлуатації програми

4.3.1 Звернення до програми

Для запуску програмного забезпечення для розв'язання задачі пошуку найкоротшого шляху необхідно запустити головний файл, що знаходиться в кореневій директорії програми. Запуск системи здійснюється методом `public static void main(String[] args)`, який знаходиться у класі `MainMenuFrame`.

4.3.2 Вхідні й вихідні дані

Вхідними даними до програмного забезпечення для розв'язання задачі пошуку найкоротшого шляху є інформація про пункти призначення, між якими необхідно побудувати оптимальний шлях, або файл з інформацією про пункти призначення, між якими необхідно побудувати оптимальний шлях. Додатковими параметрами є параметри алгоритму мультиагентного пошуку, зокрема, швидкість випаровування феромону, жадібність алгоритму, загальна кількість агентів та ін.

Вихідними даними програмного забезпечення для розв'язання задачі пошуку найкоротшого шляху є знайдений найкоротший шлях, а також його графічне відображення.

4.3.3 Повідомлення

Користувач програмного забезпечення для розв'язання задачі пошуку найкоротшого шляху може отримати такі повідомлення: повідомлення про

некоректність введених вхідних даних, а також про успішне створення графу з найкоротшим шляхом.

4.4 Виконання програмного забезпечення для розв’язання задачі пошуку найкоротшого шляху

Після запуску програми користувачу відображається головна форма (рис. 4.1), де можна ввести параметри задачі пошуку найкоротшого шляху, а також параметри алгоритму пошуку. Користувач також має можливість отримати інформацію про мультиагентний алгоритм, за допомогою якого розв’язується відповідна задача.

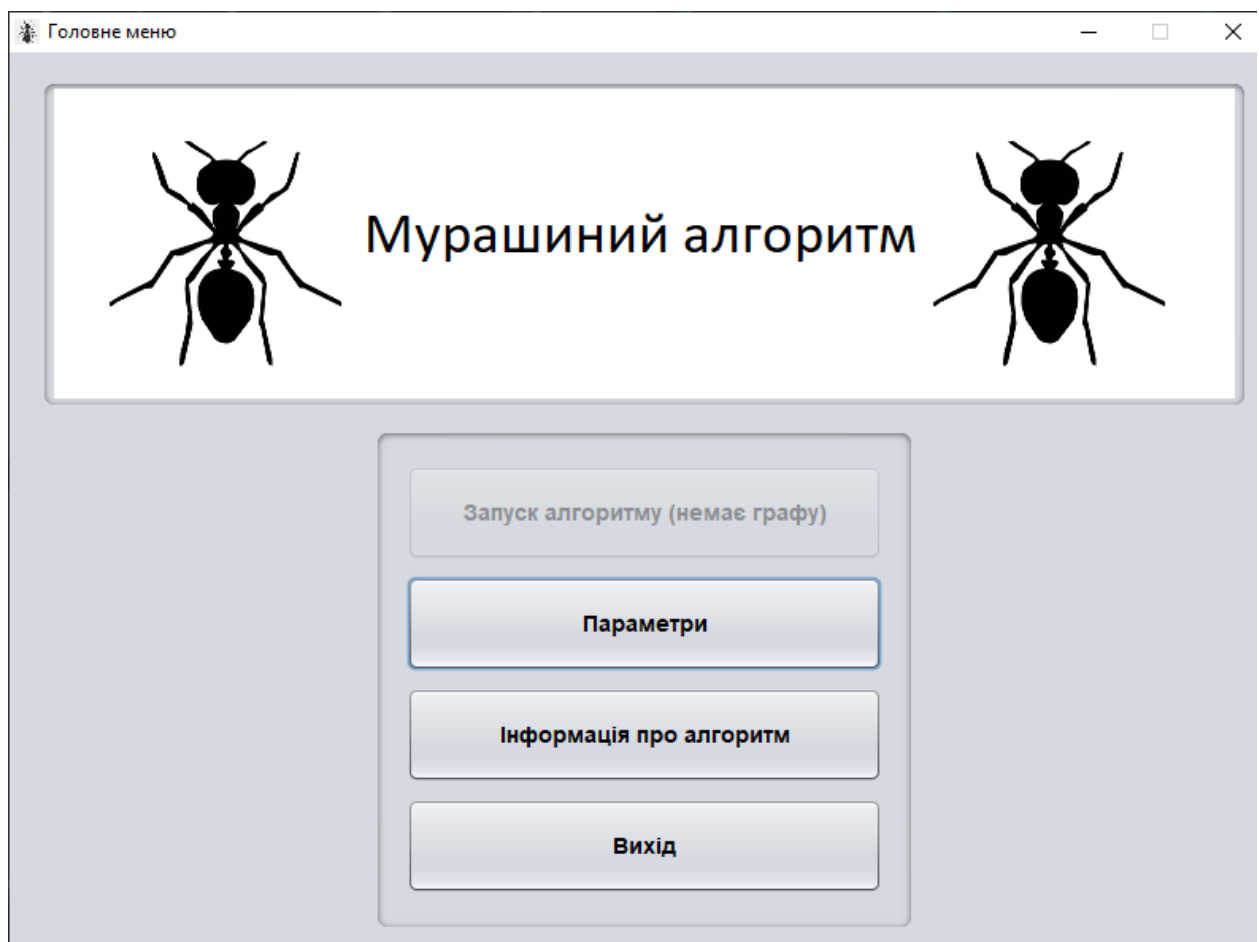


Рисунок 4.1 – Головна форма програми пошуку найкоротшого шляху

Форма з вибором параметрів задачі та мультиагентного методу для пошуку найкоротшого шляху наведена на рис. 4.2. Тут користувач може ввести граф задачі і додаткові параметри методу.

Рисунок 4.2 – Форма з вибором параметрів задачі та мультиагентного методу для пошуку найкоротшого шляху

При виборі пункту «Ручне введення» у вікні «Параметри» відображається форма з вікном, де наведено користувач може нарисувати вхідний граф задачі (рис. 4.3).

Після створення графу його можна переглянути його у вікні «Показати граф» (рис. 4.4).

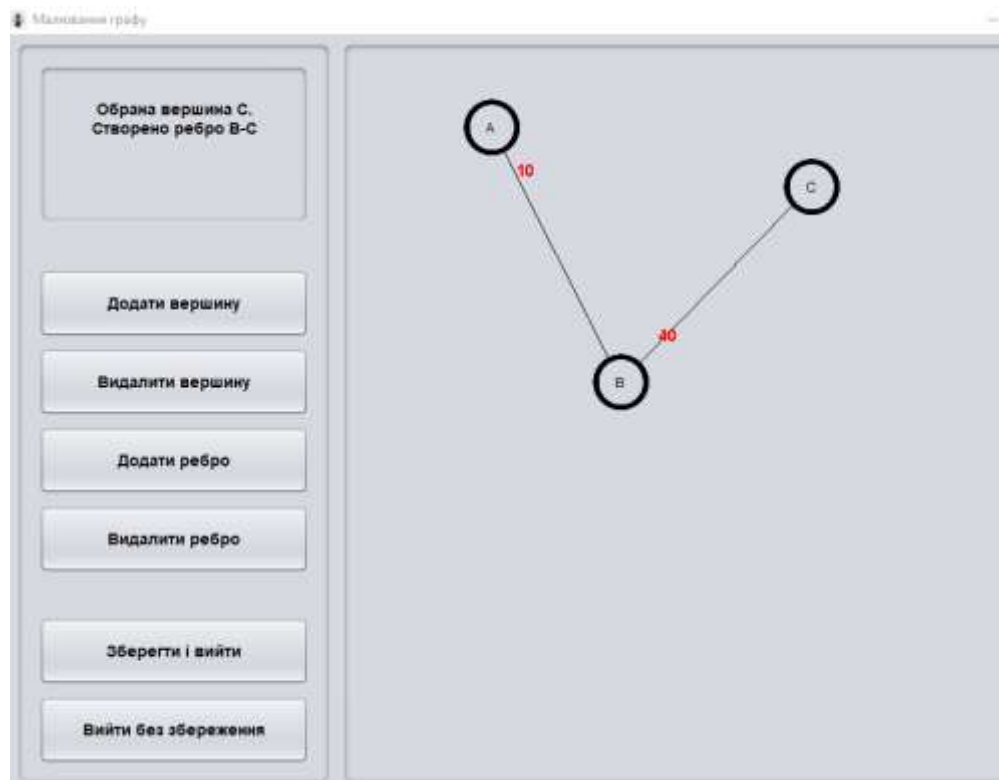


Рисунок 4.3 – Вікно рисування графу

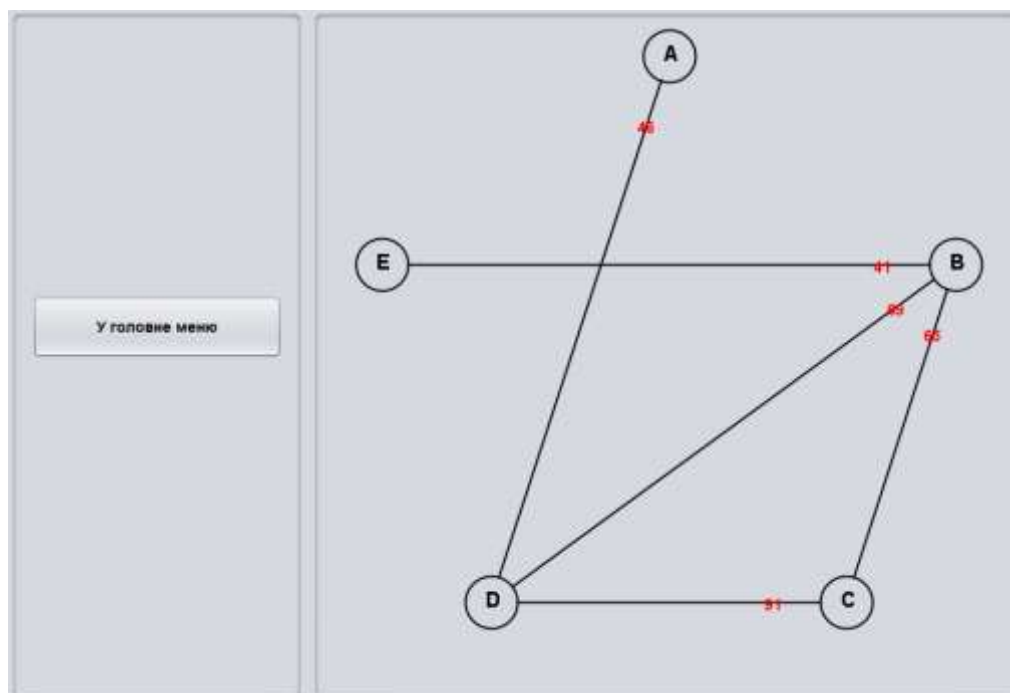


Рисунок 4.4 – Вікно «Показати граф»

Після створення графу потрібно повернутися у головне меню та натиснути кнопку «Запуск алгоритму» (рис. 4.5).

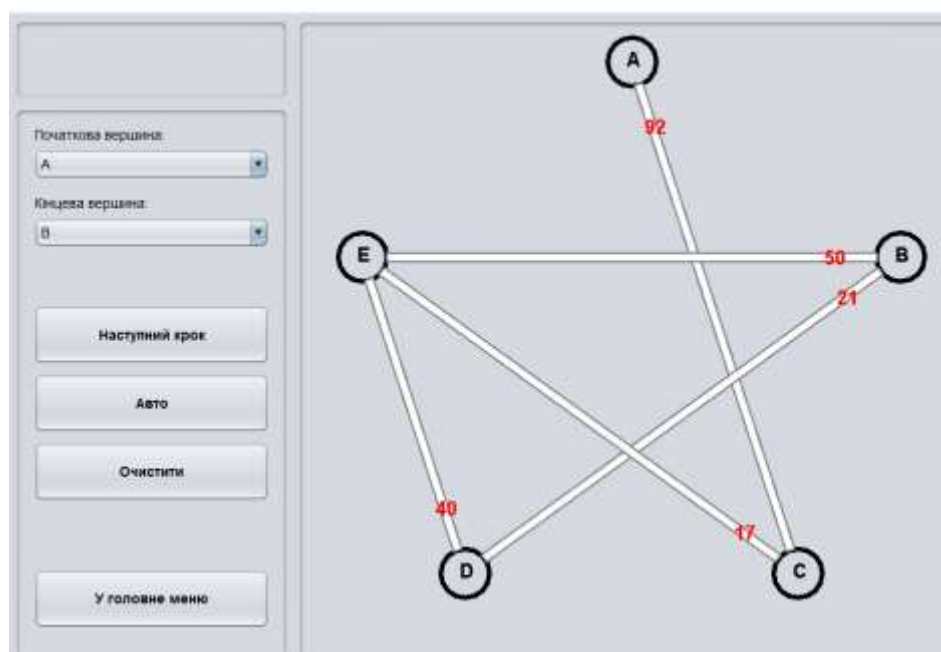


Рисунок 4.5 – Форма «Запуск алгоритму»

У цій формі користувач може вказати початкову і кінцеву вершини, а також запускати алгоритм покроково або автоматично.

Форма з відображенням найкоротшим шляхом від початкової до кінцевої точки наведена на рис. 4.6.

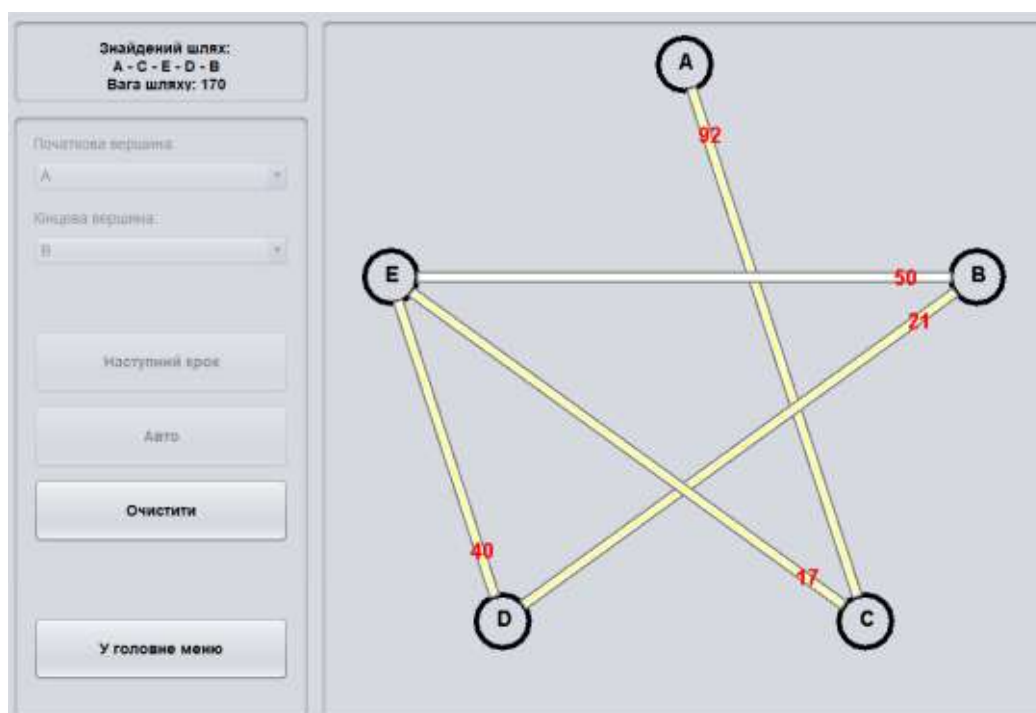


Рисунок 4.6 – Форма з відображенням найкоротшим шляхом

При натисканні на кнопку «Інформація про алгоритм» користувачу відображається форма, де він має можливість ознайомитися з інформацією про роботу мультиагентного методу мурашиних колоній (рис. 4.7).

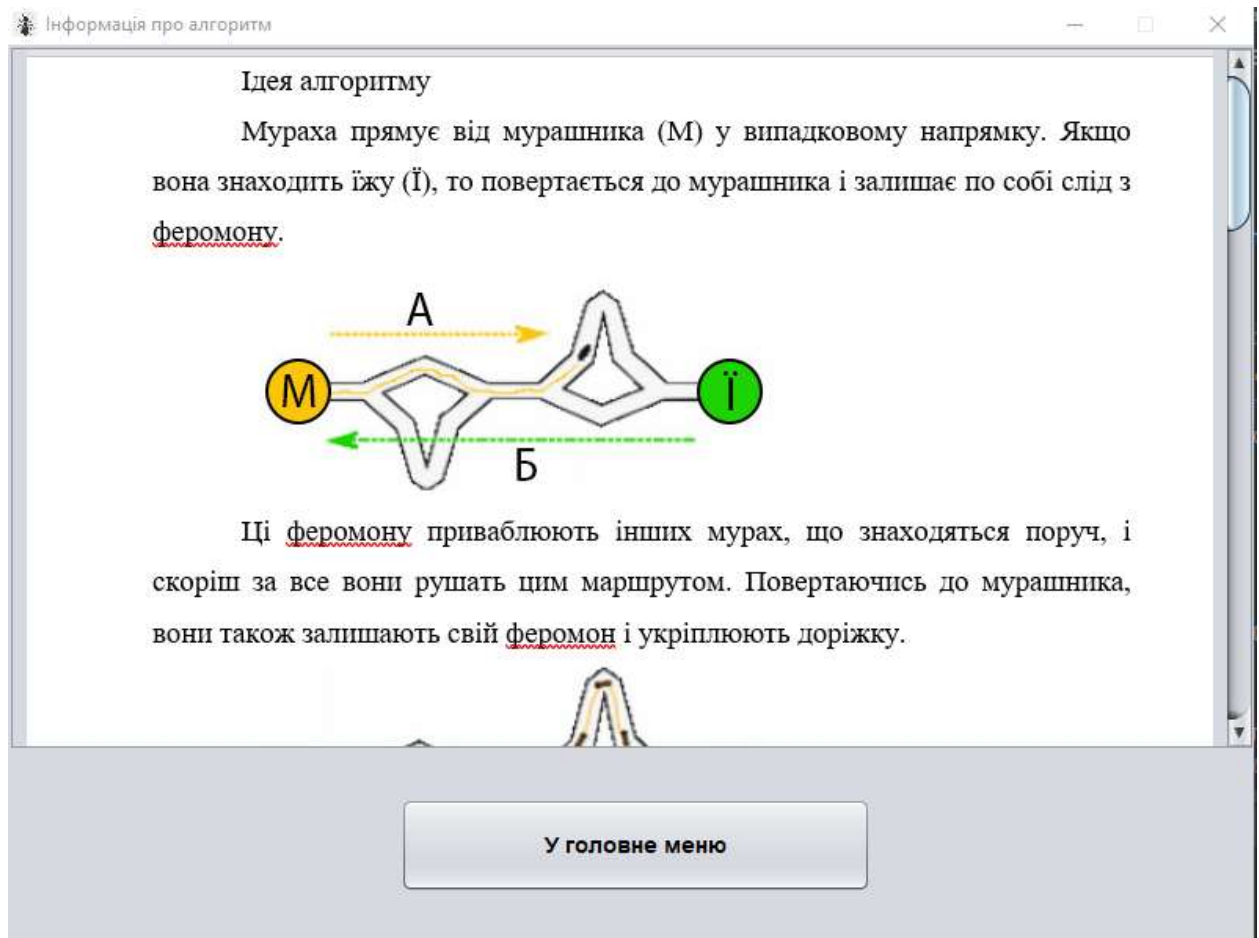


Рисунок 4.7 – Форма з інформацією про алгоритм

4.5 Тестування програмного забезпечення для пошуку оптимального шляху

Виконано тестування програмного забезпечення для пошуку найкоротшого шляху.

Виявлено, що програма для пошуку найкоротшого шляху функціонує правильно. Вона реалізує всі функціональні вимоги та успішно виконує свою основну задачу – розв’язання задачі генерації пошуку найкоротшого шляху.

Розроблене програмне забезпечення дозволяє розв'язувати завдання пошуку найкоротшого шляху.

4.6 Висновки за розділом 4

Описано програмне забезпечення для розв'язання задачі пошуку найкоротшого шляху. Виконано тестування розробленого програмного забезпечення для розв'язання задачі пошуку найкоротшого шляху. Результати тестування програмного забезпечення показали, що розроблена програма дозволяє розв'язувати задачу пошуку найкоротшого шляху.

ВИСНОВКИ

В ході виконання дипломної кваліфікаційної роботи магістра було проаналізовано та досліджено процес розробки програмного забезпечення для багатовимірної оптимізації.

Визначено, що програмні засоби для пошуку найкоротших маршрутів відіграють важливу роль у різних галузях, таких як транспорт, логістика, мережеве планування, інформаційні технології і багато інших. Вони набувають особливої важливості в сучасному світі, де управління маршрутами, оптимізація доставки та аналіз мережі стають критичними. Показано, що мета пошуку найкоротшого шляху полягає в визначенні найбільш оптимального шляху між двома точками в графі, де сума ваг ребер, утворюючи цей шлях, є найменшою. Відзначено, що розв'язання задач пошуку оптимального шляху зводиться до знаходження найкоротшого шляху між вершинами графа. Показано, що існують різні алгоритми вирішення задачі пошуку оптимального шляху. Проаналізовано методи розв'язання цієї задачі. Визначено, що для пошуку найбільш важливих шляхів широко використовуються методи на основі алгоритму пошуку найкоротших шляхів Дейкстри, у таких методах спочатку визначаються найбільш важливі шляхи від стартової точки до самої себе, після чого відображаються найбільш вагомні шляхи до інших вершин у відповідності до їх часу прибуття. Відзначено, що пошукові методи на основі теорії графів можуть займати багато часу, особливо в задачах з великою кількістю пунктів призначення. Показано, що ефективним засобом пошуку найкоротшого шляху є застосування мультиагентного підходу, що передбачає використання групи агентів або сутностей, кожен з яких приймає рішення для пошуку найкращого маршруту у великій області або графі. Відзначено, що мультиагентний підхід є особливо корисним у складних системах або задачах, де потрібно враховувати багато факторів та обмежень при пошуку оптимального маршруту.

За результатами проведеного аналізу зроблено висновок, що у наш час існує досить багато програмних засобів для генерації текстової інформації. Проте деякі програмні засоби для пошуку оптимального шляху можуть використовувати неефективні алгоритми оптимального пошуку, які базуються на пошуку по всіх можливих шляхах та, відповідно, є досить складними для реалізації та використання (вимагають великої кількості часу та апаратних ресурсів). Деякі алгоритми, що використовуються у відповідному програмному забезпеченні, можуть не мати можливості урахувати деякі особливості, такі як зміна умов транспортного потоку або різних обмежень, що можуть виникати у реальних умовах. Інші програмні засоби можуть вимагати значної кількості пам'яті для зберігання великих транспортних мереж або матриць шляхів, особливо у складних системах.

Тому актуальною є розробка програмного забезпечення для пошуку найкоротшого шляху на основі мультиагентного підходу.

Сформульовано функціональні вимоги до програмного забезпечення для пошуку найкоротшого шляху.

Для реалізації програмного забезпечення для розв'язання завдання пошуку найкоротшого шляху обрано мову програмування Java, яка є платформо незалежною, об'єктно-орієнтованою та надійною мовою, а також має велику спільноту розробників та готових програмних модулів для розв'язання оптимізаційних завдань.

Для створення програмного забезпечення для розв'язання задачі пошуку найкоротшого шляху обрано середовище розробки IntelliJ IDEA, що являє собою ефективний інструмент для створення програм, написаних на мові програмування Java та забезпечує інтелектуальну підтримку процесу розробки програмного забезпечення.

Новизна роботи полягає в тому, що створено модель взаємодії користувача з програмним забезпеченням для пошуку найкоротшого шляху, яка подана у вигляді діаграми прецедентів, та дозволяє розроблювати програмне забезпечення для пошуку найкоротшого шляху.

Практичне значення роботи полягає в тому, що розроблено програмне забезпечення для пошуку найкоротшого шляху.

Запропоновано структуру та описано основні модулі програмного забезпечення для пошуку найкоротшого шляху.

Описано функціонування програмного забезпечення для пошуку найкоротшого шляху. Описано особливості реалізації програмного забезпечення для пошуку найкоротшого шляху. Виконано проєктування інтерфейсу взаємодії користувача з програмним забезпеченням для пошуку найкоротшого шляху.

Виконано тестування розробленого програмного забезпечення для розв'язання задачі пошуку найкоротшого шляху. Результати тестування програмного забезпечення показали, що розроблена програма дозволяє розв'язувати задачу пошуку найкоротшого шляху.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАНЬ

1. Algorithm to find the "optimal" pth in a given grph [Electronic resource]. – Access mode: <https://math.stackexchange.com/questions/936417/algorithm-to-find-the-optimal-pth-in-a-given-grph>.
2. Dijkstra-Algorithmus: Wie findet man den kürzesten Weg [Electronic resource]. – Access mode: <https://datascientest.com/de/was-ist-der-dijkstra-algorithmus>.
3. Kürzeste Wege und günstigste Wege [Electronic resource]. – Access mode: https://algorithms.discrete.ma.tum.de/grph-algorithms/spp-bellman-ford/index_de.html.
4. Optimal Pth [Electronic resource]. – Access mode: <https://www.sciencedirect.com/topics/mathematics/optimal-pth>.
5. Pth finding [Electronic resource]. – Access mode: <https://en.wikipedia.org/wiki/Pthfinding>.
6. Finding an Optimal Pth [Electronic resource]. – Access mode: <https://www.mathworks.com/matlabcentral/fileexchange/36321-finding-an-optimal-pth>.
7. Algorithms for optimal min hop and foremost pths in interval tp moral grphs [Electronic resource]. – Access mode: <https://appliednetsci.springeropen.com/articles/10.1007/s41109-022-00499-3>.
8. An efficient index method for the optimal pth query over multi-cost networks [Electronic resource]. – Access mode: <https://link.springer.com/article/10.1007/s11280-021-00871-w>.
9. Grph Optimization [Electronic resource]. – Access mode: [https://math.libretexts.org/Bookshelves/Combinatorics_and_Discrete_Mathematics/Applied_Discrete_Structures_\(Doerr_and_Levasseur\)/09%3A_Grph_Theory/9.05%3A_Grph_Optimization](https://math.libretexts.org/Bookshelves/Combinatorics_and_Discrete_Mathematics/Applied_Discrete_Structures_(Doerr_and_Levasseur)/09%3A_Grph_Theory/9.05%3A_Grph_Optimization).

10. The Best Route Planning Software: 8 Different Route Planners Compared [Electronic resource]. – Access mode: <https://getcircuit.com/teams/blog/route-planning-software>.
11. Best Route Planning Software [Electronic resource]. – Access mode: <https://www.g2.com/categories/route-planning>.
12. The 10 Best Route Planning Software Solutions in 2023 [Electronic resource]. – Access mode: <https://tech.co/fleet-management/best-route-planning-software>.
13. Route Planning Software [Electronic resource]. – Access mode: <https://www.capterra.com/route-planning-software/>.
14. Circuit for Teams [Electronic resource]. – Access mode: <https://getcircuit.com/teams>.
15. Routific [Electronic resource]. – Access mode: <https://www.routific.com/>.
16. MyRouteOnline [Electronic resource]. – Access mode: <https://www.myrouteonline.com/>.
17. JDK Development Kit [Electronic resource]. – Access mode: <https://www.oracle.com/java/technologies/downloads/>.
18. Java Tutorial [Electronic resource]. – Access mode: <https://www.w3schools.com/java/>.
19. IntelliJ IDEA [Electronic resource]. – Access mode: <https://www.jetbrains.com/idea/download/?section=windows>.
20. IntelliJ IDEA – the Leading Java and Kotlin IDE [Electronic resource]. – Access mode : <https://www.jetbrains.com/idea/>.
21. Encyclopedia of artificial intelligence [Text] / Eds.: J. R. Dopico, J. D. de la Calle, A. P. Sierra. – New York : Information Science Reference, 2009. – Vol. 1-3. – 1677 p.
22. Gen M. Genetic algorithms and engineering design [Text] / M. Gen, R. Cheng. – New Jersey : John Wiley & Sons, 1997. – 352 p.

23. Субботін С. О. Ітеративні, еволюційні та мультиагентні методи синтезу нечіткологічних і нейромережних моделей: монографія [Текст] / С. О. Субботін, А. О. Олійник, О. О. Олійник ; під заг. ред. С. О. Субботіна. – Запоріжжя : ЗНТУ, 2009. – 375 с.

ДОДАТОК А
Технічне завдання

Вступ

Програмне забезпечення може використовуватися для розв’язання задачі пошуку найкоротшого шляху.

A.1 Підстава для розробки

Підставою для розробки є завдання на дипломну кваліфікаційну роботу на тему «Дослідження та програмна реалізація засобів пошуку найкоротшого шляху на основі мультиагентного підходу, затверджене наказом Національного університету «Запорізька політехніка» № 437 від 14 листопада 2023 р.

A.2 Призначення розробки

Програмний продукт призначений для розв’язання задачі найкоротшого шляху.

A.3 Основні вимоги до програми, що розробляється

A.3.1 Вимоги до функціональних характеристик

Програмне забезпечення для розв’язання задачі пошуку найкоротшого шляху забезпечує виконання таких функцій:

- можливість пошуку найкоротшого шляху;
- можливість завантаження вхідних даних з файлу;
- можливість генерування графу з файлу даних;
- можливість додавання та видалення ребер графу, що відображає умову задачі пошуку найкоротшого шляху, у випадку ручного введення даних;
- можливість задавання параметрів задачі та методу її розв’язання;
- візуалізація знайденого результату.

А.3.2 Вимоги до інтерфейсу програми

Інтерфейс програмного забезпечення для пошуку найкоротшого шляху повинен бути зручним для користувачів. Повинна бути забезпечена можливість роботи за допомогою візуального інтерфейсу.

А.3.3 Вимоги до надійності

Програмне забезпечення для пошуку найкоротшого шляху повинно забезпечити надійне функціонування.

А.3.4 Умови експлуатації

Для експлуатації програмного забезпечення для розв'язання задачі пошуку найкоротшого шляху необхідна наявність персонального комп'ютера у користувача.

А.3.5 Вимоги до складу та параметрів технічних засобів

Для експлуатації програмного забезпечення для розв'язання задачі пошуку найкоротшого шляху необхідно таке апаратне та програмне забезпечення:

- оперативна пам'ять об'ємом 4Гб і більше;
- процесор із тактовою частотою 2,5 GHz і більше;
- вільне місце на диску 250 Мб або більше;
- встановлена операційна система Windows.

A.3.6 Вимоги до маркування і пакування

Програма для розв'язання задачі пошуку найкоротшого шляху може бути записана на будь-якому носії інформації.

На пакуванні повинна бути назва програми – «Програмне забезпечення для пошуку найкоротшого шляху».

A.4 Вхідні дані до роботи

Вхідними даними до програмного забезпечення для розв'язання задачі пошуку найкоротшого шляху є інформація про пункти призначення, між якими необхідно побудувати оптимальний шлях, або файл з інформацією про пункти призначення, між якими необхідно побудувати оптимальний шлях. Додатковими параметрами є параметри алгоритму мультиагентного пошуку, зокрема, швидкість випаровування феромону, жадібність алгоритму, загальна кількість агентів та ін.

Вихідними даними програмного забезпечення для розв'язання задачі пошуку найкоротшого шляху є знайдений найкоротший шлях, а також його графічне відображення.

ДОДАТОК Б
Фрагмент тексту програми

```

public class MurahAlgrt {

    private Grph grph;
    private double grd;
    private double grgns;
    public double evprtSp;
    private int cnt;
    private int strddx;
    private int fnshdx;
    private int crntdx;
    private int nmberndMurahs;
    private int nmbrMurahs;

    public MurahAlgrt(Grph grph_,
                      double grd_,
                      double evprtSp_,
                      int nmberndMurahs_,
                      int nmbrMurahs_,
                      Pair<Integer, Integer> pth) {

        grph = grph_;

        if (grd_ <= 1) {
            grd = grd_;
            grgns = 1 - grd;
        } else {
            grd = 0.5;
            grgns = 0.5;
        }

        if (evprtSp < 1)
            evprtSp = evprtSp_;
        else
            evprtSp = 0.5;
    }
}

```

```

        nmberndMurahs = nmberndMurahs_;
        nmbrMurahs = nmbrMurahs_;

        strddx = pth.frst;
        fnshdx = pth.scnd;

        cnt = 0;
    }

    public boolean fnshd() {
        return cnt == nmbrMurahs;
    }

    public int getCnt() {
        return cnt;
    }

    public List<Integer> stp() {
        List<Integer> wyn = new ArrayList<>();
        List<Ed> wye= new ArrayList<>();
        List<Integer> ndsInPth = new ArrayList();

        wyn.add(strddx);
        crntdx = strddx;
        List<Integer> bndn = new ArrayList<>();

        if (cnt < nmberndMurahs) {
            while (crntdx != fnshdx) {
                List<Integer> bsplnn= new ArrayList<>();
                for (int i = 0; i < grph.nmbrVertices; ++i)
                    if (grph.adjacencyMatrix[crntdx][i] != 0) {
                        boolean okk = true;
                        for (Integer j : wyn)
                            if (i == j) {
                                okk = false;
                                break;

```

```

        }
        if (okk) {
            for (Integer j : bndn)
                if (j == i)
                    okk = false;
            if (okk) bsplnn.add(i);
        }
    }

    Rndom rnd = new Rndom();

    if (bsplnn.size() != 0) {
        crntdx = bsplnn.get(Math.abs(rnd.nextInt()) %
bsplnn.size());

        wyn.add(crntdx);
    } else {
        bndn.add(crntdx);
        crntdx = wyn.get(wyn.size() - 2);
        wyn.remove(wyn.size() - 1);
        wye.remove(wye.size() - 1);
        bsplnn.clear();
        continue;
    }

    for (Ed i : grph.eds)
        if ((i.frstNd == Math.min(crntdx,
wyn.get(wyn.size() - 2)))
            && (i.scndNd == Math.max(crntdx,
wyn.get(wyn.size() - 2)))) {
            wye.add(i);
            break;
        }
    }

    for (Ed i : wye)
        i.incrntPth = true;

```

```

for(Integer i : wyn)
    ndsInPth.add(i);

wyn.clear();

double wwgt = 0.0;
for (Ed i : wye) {
    wwgt += i.wgt;
}

ndsInPth.add(new Integer((int)wwgt));

for (Ed i : grph.eds) {
    i.phrmn = (1.0 - evprtSp) * i.phrmn;
    if (i.incrntPth) {
        i.phrmn += 1.0 / wwgt;
        i.incrntPth = false;
    }
}

bndn.clear();

} else {
    while (crntdx != fnshdx) {

        List<Integer> bsplnn= new ArrayList<>();
        List<Ed> pslbnn = new ArrayList<>();
        for (int i = 0; i < grph.nmbrVertices; ++i) {
            if (grph.adjacencyMatrix[crntdx][i] != 0) {
                boolean okk = true;
                for (Integer j : wyn)
                    if (i == j) {
                        okk = false;
                        break;
                    }
            }
        }
    }
}

```

```

        if (okk) {
            for (Integer j : bndn)
                if (j == i)
                    okk = false;
            if (okk) bsplnn.add(i);
        }
    }

    for (int j : bsplnn) {
        for (Ed i : grph.eds)
            if ((i.frstNd == Math.min(j, crntdx))
                && (i.scndNd == Math.max(j,
crntdx))) {

                pslbnn.add(i);
                break;
            }
    }

    List<Double> ver = new ArrayList<>();
    double msm = 0.0;

    for (Ed i : pslbnn) {
        double k = Math.pow(i.phrmn,grgns) * Math.pow
(1.0 / i.wgt, grd);
        msm += k;
        ver.add(k);
    }

    if (msm == 0.0) {
        bndn.add(crntdx);
        crntdx = wyn.get(wyn.size() - 2);
        wyn.remove(wyn.size() - 1);
        wye.remove(wye.size() - 1);
        bsplnn.clear();
        continue;
    }

```

```

    }

    for (int i = 0; i < ver.size(); ++i) {
        double tpm = ver.get(i);
        ver.set(i, tpm/msm);
    }

    for (int i = 1; i < ver.size(); ++i) {
        double tpm = ver.get(i);
        ver.set(i, tpm + ver.get(i - 1));
    }

    msm = Math.rndom();

    for (int i = 0; i < ver.size(); ++i)
        if ( ver.get(i) >= msm) {
            if ( pslbnn.get(i).firstNd ==
wyn.get(wyn.size() - 1) )
                crntdx = pslbnn.get(i).scndNd;
            else
                crntdx = pslbnn.get(i).firstNd;
            break;
        }

    if (bsplnn.size() == 0) {
        bndn.add(crntdx);
        crntdx = wyn.get(wyn.size() - 2);
        wyn.remove(wyn.size() - 1);
        wye.remove(wye.size() - 1);
        bsplnn.clear();
        continue;
    }

    for ( Ed i : grph.eds)
        if ( (i.firstNd == Math.min(crntdx,
wyn.get(wyn.size() - 1)) )

```

```

                                && (i.scndNd == Math.max(crntdx,
wyn.get(wyn.size() - 1)) ) ){
                                wye.add(i);
                                break;
                                }

                                wyn.add(crntdx);
                                }

                                for(Ed i : wye)
                                    i.incrntPth = true;

                                for(Integer i : wyn)
                                    ndsInPth.add(i);

                                wyn.clear();

                                double wwgt = 0.0;
                                for (Ed i : wye) {
                                    wwgt += i.wgt;
                                }

                                ndsInPth.add(new Integer((int)wwgt));

                                for (Ed i : grph.eds) {
                                    i.phrmn = (1.0 - evprtSp) * i.phrmn;
                                    if (i.incrntPth){
                                        i.phrmn += 1.0 / wwgt;
                                        i.incrntPth = false;}
                                }

                                bndn.clear();
                                }

                                ++cnt;

```

```

        return ndsInPth;
    }

    public List<Integer> avtalg() {
        while (cnt < nmbrMurahs)
            stp();

        return fndPth();
    }

    public List<Integer> fndPth() {
        List<Integer> ndsInPth = new ArrayList<>();
        List<Ed> probEds = new ArrayList<>();
        ndsInPth.add(strddx);
        List<Ed> bnd = new ArrayList<>();
        crntdx = strddx;
        int wwgt = 0;

        while (crntdx != fnshdx) {
            for (Ed i : grph.eds)
                if ( i.frstNd == crntdx || i.scndNd == crntdx ){
                    boolean okk = true;
                    for (Ed k : bnd)
                        if (k == i)
                            okk = false;
                    if (okk)
                        probEds.add(i);
                }

            Ed ed = probEds.get(0);
            for (Ed i : probEds)
                if (i.phrmn > ed.phrmn)
                    ed = i;

            bnd.add(ed);
            probEds.clear();

```

```
        wwgt += ed.wgt;

        if (ed.frstNd == crntdx)
            ndsInPth.add(ed.scndNd);
        else
            ndsInPth.add(ed.frstNd);

        crntdx = ndsInPth.get(ndsInPth.size() - 1);
    }

    ndsInPth.add(wwgt);

    return ndsInPth;
}
}
```

ДОДАТОК В
Слайди презентації

Міністерство освіти і науки України
Національний університет «Запорізька політехніка»
Кафедра програмних засобів

ДОСЛІДЖЕННЯ ТА ПРОГРАМНА РЕАЛІЗАЦІЯ ЗАСОБІВ ПОШУКУ НАЙКОРОТШОГО ШЛЯХУ НА ОСНОВІ МУЛЬТИАГЕНТНОГО ПІДХОДУ

**СТУДЕНТ: ЖЕЛНІН Михайло,
КНТ-112М**

**КЕРІВНИК: КОЦУР Михайло,
к.т.н., доцент**

Рисунок В.1 – Слайд 1

ОБ'ЄКТ, ПРЕДМЕТ ТА МЕТА РОБОТИ

Об'єкт дослідження – процес розробки програмного забезпечення для багатовимірної оптимізації.

Предмет дослідження – методи та програмні засоби для розв'язання завдань пошуку найкоротшого шляху.

Мета роботи – дослідження та розробка програмного забезпечення для пошуку найкоротшого шляху на основі мультиагентного підходу.

2

Рисунок В.2 – Слайд 2

ЗАВДАННЯ РОБОТИ

Для досягнення поставленої мети у кваліфікаційній роботі магістра необхідно розв'язати такі задачі:

- виконати аналіз предметної області та програмних засобів для пошуку найкоротшого шляху;
- розробити модель взаємодії користувача з програмним забезпеченням для пошуку найкоротшого шляху;
- здійснити проектування програмного забезпечення для пошуку найкоротшого шляху на основі мультиагентного підходу;
- створити програмне забезпечення для пошуку найкоротшого шляху на основі мультиагентного підходу;
- виконати тестування розробленого програмного забезпечення для пошуку найкоротшого шляху.

3

Рисунок В.3 – Слайд 3

ПОРІВНЯННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ ПОШУКУ НАЙКОРОТШОГО ШЛЯХУ

Критерій порівняння	Circuit for Teams	Routific	MyRoute Online
Зручність планування маршрутів	+	+	+—
Інтеграція з іншими програмними засобами	+	+—	—
Сертифікати доставки	+	—	—
Мобільна версія	+	+	—
Розширені засоби аналітики даних	+	+—	—
Безкоштовність	—	—	+—

4

Рисунок В.4 – Слайд 4

ОБҐРУНТУВАННЯ ВИБОРУ МОВИ ПРОГРАМУВАННЯ

Критерій порівняння мов програмування	Мова програмування		
	Java	Python	C++
Зручність розробки програмного забезпечення для пошуку найкоротшого шляху	+	-	-
Робота з текстовими даними	+	+—	+
Підтримка багатопоточності	+	+—	+
Наявність великої кількості програмних модулів для розв'язання оптимізаційних завдань	+	+—	+—
Засоби безпеки даних	+	+	+—

5

Рисунок В.5 – Слайд 5

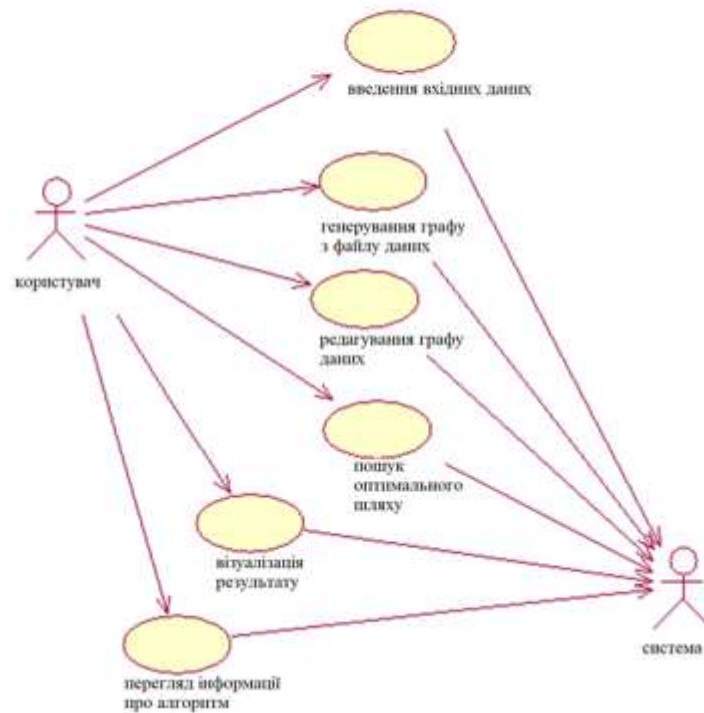
ОБҐРУНТУВАННЯ ВИБОРУ СЕРЕДОВИЩА РОЗРОБКИ

Критерій порівняння середовищ розробки	Середовище розробки		
	IntelliJ IDEA	Visual Studio	Eclipse
Зручність розробки на мові Java	+	+—	+
Інструменти для роботи з кодом	+	+	+—
Функціонал для рефакторингу	+	+	+—
Зручність для створення ПЗ для пошуку найкоротшого шляху	+	-	-
Рівень підтримки розробників	+	+	+—

6

Рисунок В.6 – Слайд 6

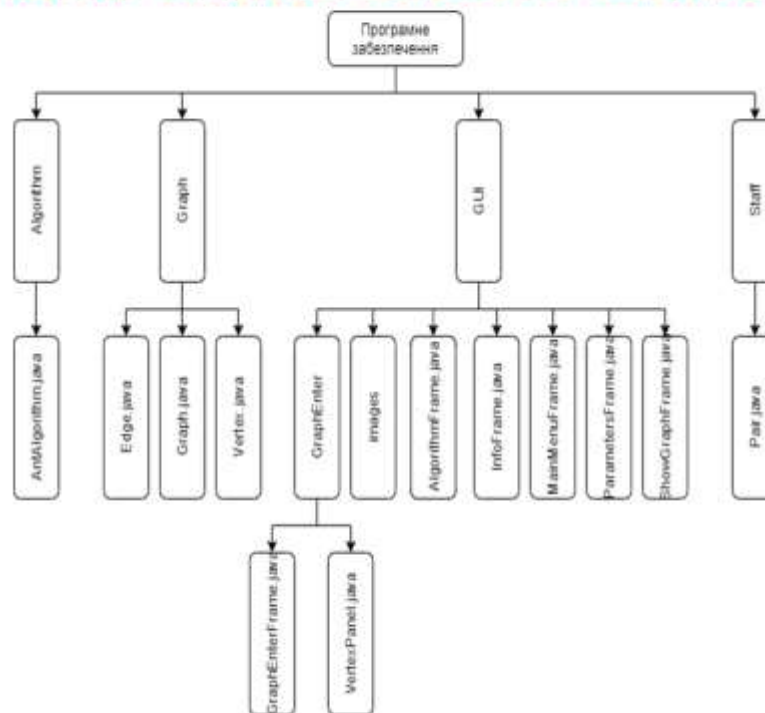
МОДЕЛЬ ВЗАЄМОДІЇ КОРИСТУВАЧА З ПЗ ДЛЯ ПОШУКУ НАЙКОРОТШОГО ШЛЯХУ



7

Рисунок В.7 – Слайд 7

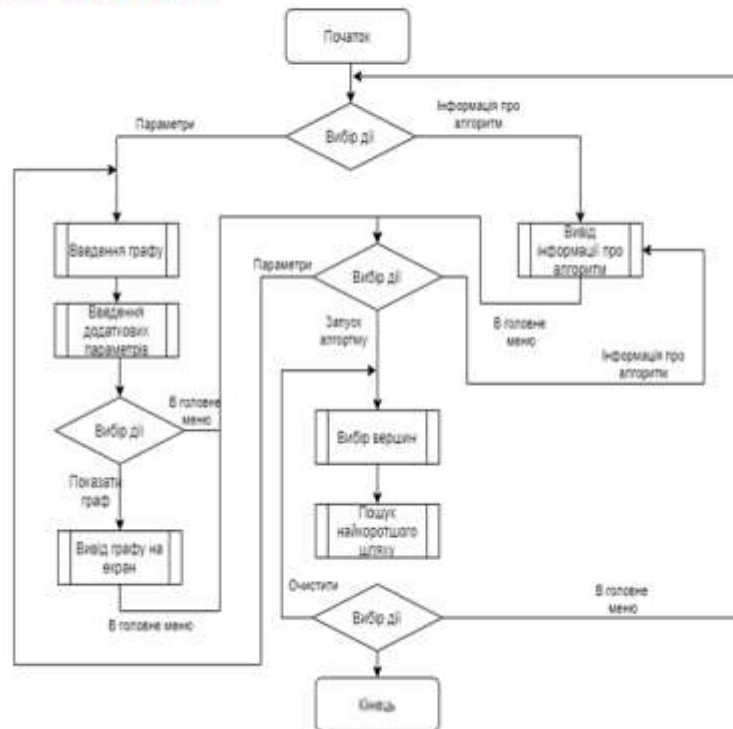
СТРУКТУРА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ ПОШУКУ НАЙКОРОТШОГО ШЛЯХУ



8

Рисунок В.8 – Слайд 8

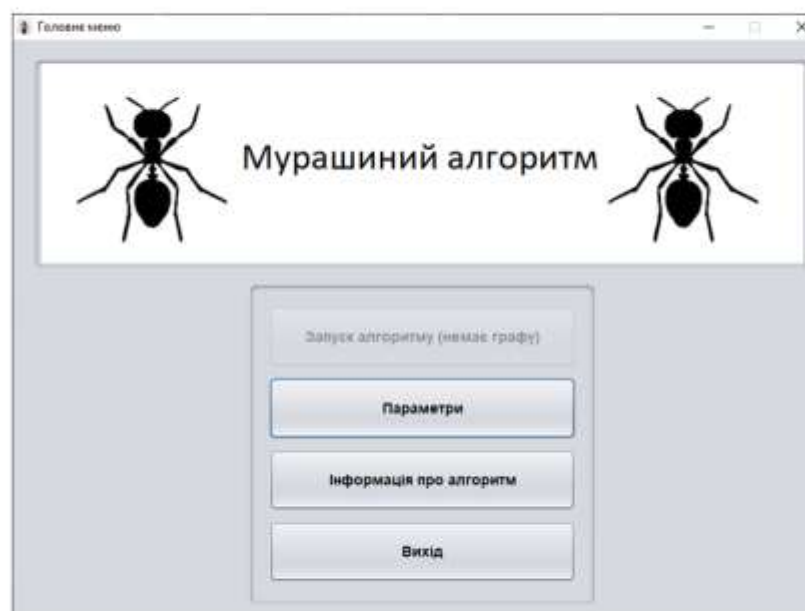
ФУНКЦІОНУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ



9

Рисунок В.9 – Слайд 9

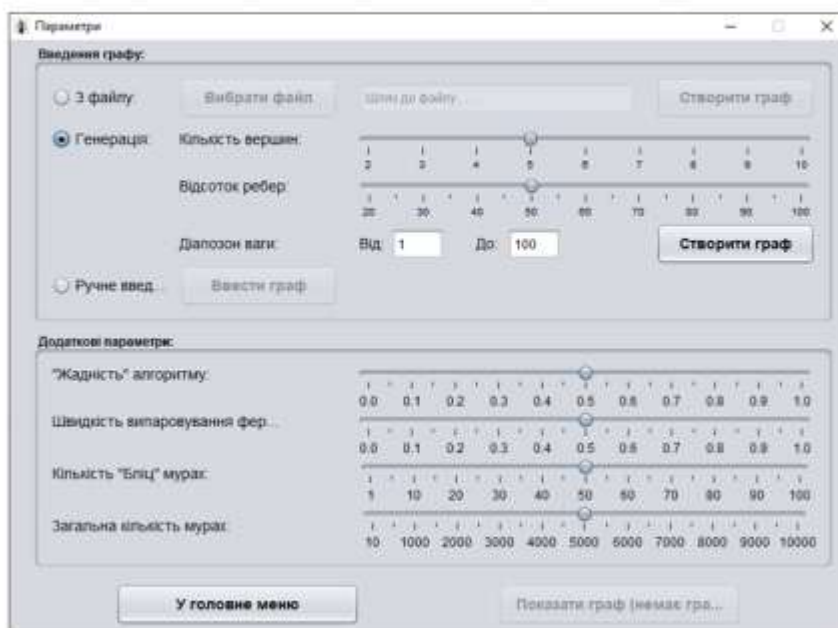
РОБОТА КОРИСТУВАЧА З ПРОГРАМОЮ. ГОЛОВНА ФОРМА



10

Рисунок В.10 – Слайд 10

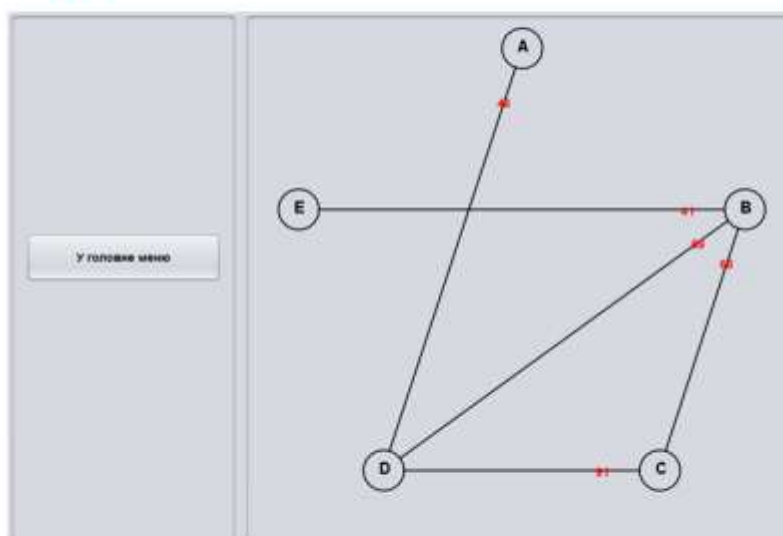
РОБОТА КОРИСТУВАЧА З ПРОГРАМОЮ. ФОРМА З ВИБОРОМ ПАРАМЕТРІВ



11

Рисунок В.11 – Слайд 11

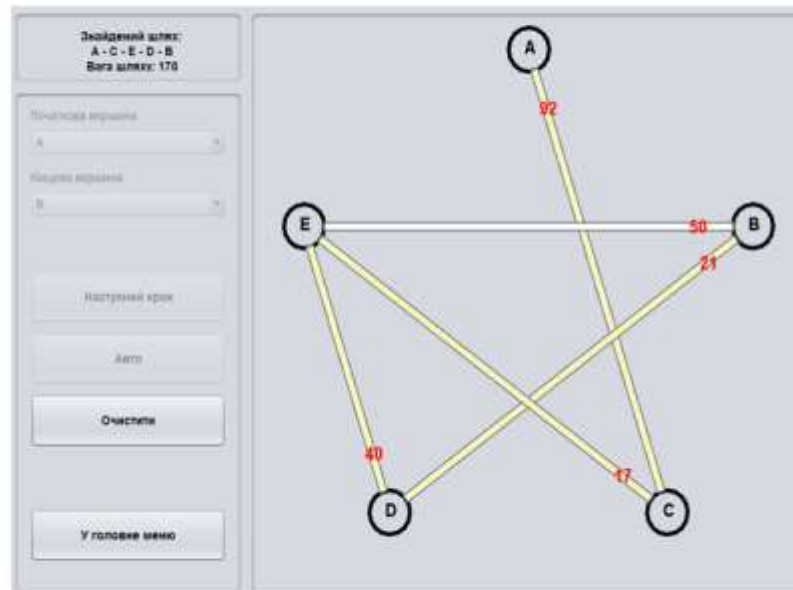
РОБОТА КОРИСТУВАЧА З ПРОГРАМОЮ. ФОРМА ДЛЯ РИСУВАННЯ ГРАФУ



12

Рисунок В.12 – Слайд 12

РОБОТА КОРИСТУВАЧА З ПРОГРАМОЮ. ВІДОБРАЖЕННЯ НАЙКОРОТШОГО ШЛЯХУ



13

Рисунок В.13 – Слайд 13

ВИСНОВКИ

Під час виконання дипломної кваліфікаційної роботи бакалавра було проаналізовано та досліджено процес розробки програмного забезпечення для багатовимірної оптимізації.

Новизна роботи полягає в тому, що створено модель взаємодії користувача з програмним забезпеченням для пошуку найкоротшого шляху, яка подана у вигляді діаграми прецедентів, та дозволяє розроблювати програмне забезпечення для пошуку найкоротшого шляху.

Практичне значення роботи полягає в тому, що розроблено програмне забезпечення для пошуку найкоротшого шляху.

Результати тестування програмного забезпечення пошуку найкоротшого шляху показали, що програма функціонує коректно та відповідає технічному завданню.

14

Рисунок В.14 – Слайд 14