

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)

бакалавр

(ступінь вищої освіти)

на тему ПРОГРАМНА РЕАЛІЗАЦІЯ ВИРІШЕННЯ ЗАДАЧ
МАРШРУТИЗАЦІЇ З ПІДБОРОМ ТА ВАГОВИМИ
ОБМЕЖЕННЯМИ
SOFTWARE IMPLEMENTATION OF SOLVING
ROUTING PROBLEMS WITH MULTIPLE OPTIONS
AND WEIGHT LIMITS

Виконав(ла): студент(ка) 4 курсу, групи КНТ-111сп
Спеціальності 121 Інженерія програмного
забезпечення

(код і найменування спеціальності)

Освітня програма (спеціалізація)
Інженерія програмного забезпечення

НЕСТЕРОВ Г.Д.

(ПРИЗВИЩЕ та ініціали)

Керівник ОЛІЙНИК А.О.

(ПРИЗВИЩЕ та ініціали)

Рецензент КОЦУР М.І.

(ПРИЗВИЩЕ та ініціали)

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Факультет КНТ
Кафедра програмних засобів
Ступінь вищої освіти бакалавр
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
“ ” 2024 року

З А В Д А Н Н Я
НА ДИПЛОМНИЙ ПРОЄКТ (РОБОТУ) СТУДЕНТА(КИ)

НЕСТЕРОВА Геннадія Денисовича

(ПРИЗВИЩЕ, ім'я, по батькові)

1. Тема проєкту (роботи) Програмна реалізація вирішення задач маршрутизації з підбором та ваговими обмеженнями. Software Implementation of Solving Routing Problems with Multiple Options and Weight Limits

керівник проєкту (роботи) д.т.н., професор ОЛІЙНИК Андрій Олександрович,
(науковий ступінь, вчене звання, ПРИЗВИЩЕ, ім'я, по батькові)

затверджені наказом закладу вищої освіти від “ 24 ” квітня 2024 року № 168

2. Строк подання студентом проєкту (роботи) 10 червня 2024 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. Вибір та обґрунтування структури об'єкта, який проєктується. 3. Основні рішення щодо реалізації компонентів системи. 4. Керівництво програміста. 5. Керівництво оператора. 6. Тестування і результати.

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
1-6 Основна частина	ОЛІЙНИК А.О., професор		
Нормоконтроль	ЛИПОВЕЦЬ М.В., асистент		

7. Дата видачі завдання “ 15 ” квітня 2024 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	Розробка схеми функціонування системи та модулів Програми.	2 тиждень	Розділ 2
4	Розробка програми	2-3 тижні	Текст програми
5	Опис реалізованих рішень та створення керівництв	3-4 тижні	Розділи 3, 4, 5
6	Тестування та експериментальне дослідження програмного забезпечення.	5 тиждень	Розділ 6
7	Оформлення пояснювальної записки та документів до неї.	6 тиждень	Додатки
8	Нормоконтроль та рецензування.	7 тиждень	
9	Захист роботи.	8 тиждень	

Студент(ка) _____ Геннадій НЕСТЕРОВ
(підпис) (Імя ПРИЗВИЩЕ)

Керівник проєкту (роботи) _____ Андрій ОЛІЙНИК
(підпис) (Імя ПРИЗВИЩЕ)

РЕФЕРАТ

Пояснювальна записка до дипломної кваліфікаційної роботи бакалавра: 80 с., 6 табл., 29 рис., 3 дод., 24 джерела.

VRP, VRPPD, МЕТАЕВРИСТИЧНІ АЛГОРИТМИ, ГЕНЕТИЧНИЙ АЛГОРИТМ, ТАБУ ПОШУК, ЖАДІБНИЙ АЛГОРИТМ.

Об'єкт дослідження – процес організації транспортних перевезень.

Предмет дослідження – задача маршрутизації транспортних засобів з підбором та ваговими обмеженнями.

Методи дослідження – модифіковані методи дослідження операцій, зокрема жадібні алгоритми.

Технічні засоби: мова програмування JavaScript, мова програмування Python, Google Maps Api.

Результатом роботи стала сукупність алгоритмів розв'язання задач маршрутизації транспортних засобів з підбором та ваговими обмеженнями.

Актуальність та проблематика роботи полягає в тому, що останнім часом збільшився попит на доставку товарів, внаслідок чого проєктувальники маршрутів мають діло з великою кількістю вхідної інформації. Обробка та аналіз великих масивів інформації і висока динаміка їх параметрів призвели до необхідності розробки нових методів планування маршрутів доставок.

Висновки. Доведено, що пропонуваний підхід забезпечує стійку збіжність процесу пошуку за прийнятну кількість кроків без штучного обмеження простору пошуку та використання додаткової експертної інформації про доцільність можливих рішень.

Галузь використання – логістичні програмні засоби.

ABSTRACT

Explanatory note to the diploma qualifying work of the bachelor: 80 pages, 6 tables, 29 figures, 3 appendixes, 24 sources.

VRP, VRPPD, METAHEURISTIC ALGORITHMS, GENETIC ALGORITHM, TABOO SEARCH, GREEDY ALGORITHM.

Object of study is a software for the organizing transportation.

Subject of study is software implementation of solving routing problems with multiple options and weight limits

Research methods – modified operations research methods, in particular greedy algorithms.

Technical means: JavaScript programming language, Python programming language, Google Maps Api.

The result of the work was a collection of algorithms for solving vehicle routing problems with selection and weight restrictions.

The relevance and problematic of the work is that the demand for the delivery of goods has recently increased, as a result of which route planners have to deal with a large amount of incoming information. The processing and analysis of large amounts of information and the high dynamics of their parameters led to the need to develop new methods of planning delivery routes.

Conclusions. It is proven that the proposed approach ensures stable convergence of the search process in an acceptable number of steps without artificially limiting the search space and using additional expert information about the feasibility of possible solutions.

The field of use is a logistics software.

ЗМІСТ

	С.
Перелік скорочень та умовних познач.....	8
Вступ.....	9
1 Аналіз предметної області.....	11
1.1 Огляд існуючих методів вирішення завдання	11
1.2 Огляд існуючих програмних засобів, що вирішують аналогічні завдання	13
1.2.1 Програмний продукт LocalSolver.....	13
1.2.2 Програмний продукт ArcGis Desktop	14
1.2.3 Програмний продукт OptaPlanner	15
1.2.4 Порівняльна таблиця існуючих та розроблюваного програмних продуктів.....	16
1.3 Постановка завдання роботи.....	17
1.4 Висновки за розділом 1	18
2 Вибір та обґрунтування структури об'єкта, який проєктується	19
2.1 Вибір інструментарію розробки програмного забезпечення	19
2.1.1 Огляд особливостей мови програмування	22
2.1.2 Огляд особливостей обраного середовища розробки	27
2.2 Структурна схема програми	28
2.3 Опис модулів розроблюваної системи.....	29
2.4 Висновки за розділом 2	30
3 Основні рішення щодо реалізації компонентів системи.....	31
3.1 Алгоритм функціонування розробленої програми.....	31
3.1.1 Генетичний метод вирішення задачі маршрутизації	32
3.1.2 Модифікація генетичного методу «подвоєні хромосоми».....	34
3.1.3 Метод пошуку з табу для вирішення задачі маршрутизації	36
3.1.4 Жадібний метод вирішення задачі маршрутизації.....	39
3.1.5 Метод мурашиної колонії	40
3.1.6 Метод вирішення задачі тривимірної упаковки методом перебору..	43
3.2 Вибір шаблону проєктування	44
3.3 Опис модулів розроблюваної системи.....	46
3.4 Висновки за розділом 3	47

4 Керівництво програміста.....	48
4.1 Призначення й умови застосування програми.....	48
4.2 Характеристики програми.....	49
4.3 Звертання до програми	49
4.4 Початкові та вихідні дані	50
4.5 Повідомлення програмісту.....	50
4.6 Висновок	51
5 Керівництво оператора	52
5.1 Призначення програми	52
5.2 Умови виконання програми	52
5.3 Виконання програми.....	53
5.4 Повідомлення оператору.....	56
5.5 Висновок	56
6 Тестування і результати.....	57
6.1 Опис методики проведення тестування.....	57
6.2 Результати використання розробленого програмного забезпечення	58
Висновки	60
Перелік джерел посилань	61
Додаток А Технічне завдання	64
Додаток Б Фрагмент тексту програми	68
Додаток В Слайди презентації.....	73

ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАК

API	– Application Programming Interface;
CSS	– Cascade Style Sheet;
HTML	– Hypertext Markup Language;
IDE	– Integrated Development Environment;
JS	– Javascript;
TS	– Tabu Search;
VRP	– Vechicle Routing Problem;
CVRP	– Capacitated Vehicle Routing Problem;
PDVRP	– Pickup-and-delivery VRP;
DVRP	– Dynamic VRP;
VRPSF	– Vehicle Routing Problem with Satellite Facilities;
VRPTW	– VRP with Time Windows;
КГ	– кілограм;
МК	– мікроконтролер;
ПЗ	– програмне забезпечення;
ПП	– програмний продукт;
ТЗ	– транспортний засіб.

ВСТУП

Згідно з 30м щорічним звітом з логістики від The Council of Supply Chain Management Professionals (CSCMP) лише в США за рік було витрачено понад 1.64 трильйонів доларів на логістичні та транспортні операції, а кінцева вартість продукту, що потрапляє до споживача, може складатись до 70% з транспортних витрат [1].

Тож перед галуззю транспортної логістики однією з важливіших задач постає економія ресурсів та мінімізація екологічного сліду при транспортуванні вантажів. Для вирішення цієї проблеми стоїть питання розробки алгоритмів та програмних продуктів, що будуть скорочувати маршрути транспортних засобів. В останні роки значно збільшилась частка персональних доставок, які прив'язанні до зайнятості клієнтів, актуальним є питання врахування часових вподобань одержувачів вантажу.

Математичне формулювання цієї задачі відоме як задача маршрутизації транспортних засобів (далі VRP) з урахуванням часових вікон (далі VRPTW), яка накладає певні часові обмеження на обслуговування клієнтів транспортної мережі [2]. Робота присвячена дослідженню та удосконаленню розв'язання задачі маршрутизації транспортних засобів з урахуванням часових вікон.

Швидкість обробки та ємність пам'яті сучасних комп'ютерів зростає в геометричній прогресії, що дозволяє вирішувати більші екземпляри VRP, кожного року пропонується нові різновиди задач VRP та розробляються нові методи їх розв'язання, все це стимулює прогрес у дослідницькій галузі та розробку комерційного програмного забезпечення для VRP [3]. На сьогодні програмне забезпечення VRP використовується тисячами компаній, зокрема Coca-Cola Enterprises та Anheuser-Busch Inbev, у багатьох галузях промисловості.

Не дивлячись на розвиток галузі, специфіка задачі не дозволяє розробити єдиний програмний засіб для її вирішення при будь-яких видах вхідних даних. Це дозволяє говорити про актуальність розробки нових

алгоритмів. Актуальність нашої роботи полягає у необхідності розробки модифікованого алгоритму, що вирішував би задачі маршрутизації транспортних засобів із обмеженнями габаритів вантажу, різноманітним набором транспортних засобів із обмеженням габаритів доступного для перевезення вантажу та необмеженою кількістю точок підбору та точок доставки.

Саме у такій постановці, задача має прикладний характер і доповнює доцільність використання реальної карти у програмі. Після вивчення предметної області та тестування різних методів вирішення задачі, було створено модифікований генетичний алгоритм, який дозволяє визначити наближений до оптимального розподіл вантажу за транспортним парком, а транспорт за точками доставки. Це дозволить забезпечити організацію доставки замовлень із мінімальними втра-тами при доставці. Програма буде виконувати наступні функції:

- приймати координати точок підбору і доставки;
- приймати кількості та параметри транспортних засобів;
- приймати кількості та параметри замовлень;
- розраховувати оптимальні маршрути та розподіли товарів;
- виводити результати розрахунків.

Всі необхідні дані будуть зберігатися у інтерфейсі прикладного програмування Google, що зробить адміністрування більш зручним та забезпечить захист персональних даних та позбавить від необхідності зберігати дані самотійно.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Огляд існуючих методів вирішення завдання

Завдання маршрутизації транспорту є із класу завдань логістики. Очевидно, своєю метою має мінімізацію транспортних засобів, пройденого засобу видалення найважливіших кранів. Вперше проблему маршрутизації було запропоновано Г. Данцигом та Дж. Рамсером у 1959 році [4].

Завдання полягало у пошуку оптимального маршруту доставки конкретно заданої кількості одних товарів, використовуючи транспортні засоби однакової вантажопідйомності. Автори поставили математичне формулювання та вирішили завдання про постачання бензину на заправні станції. Це завдання є однією з найвідоміших у галузі комбінаторної оптимізації навіть на сьогоднішній день [4]. Вони описали реальне застосування щодо доставки бензину до СТО та запропонували перше формулювання математичного програмування та алгоритмічний підхід.

Кілька років потому, в 1964 році, Кларк Дж. і Райт Дж.Р. запропонували ефективний жадібний алгоритм названий алгоритмом заощаджень, що вдосконалив підхід Данцига-Рамзера [5]. Після цих двох основних робіт було запропоновано сотні моделей та алгоритмів для оптимального та наближеного розв'язання різних версій VRP [6]. Значний вклад у дослідження задач VRP внесли також Фред Гловер, який вперше описав Табу пошук та ввів термін «метаевристика»; Маріус Соломон, який досліджував підходи чутливих часових аспектів ланцюга постачань [7], часові обмеження при плануванні [8], маршрутизацію та управління складськими системами; Марко Доріго, що запропонував метаевристичний «Мурашиний алгоритм» [9].

Сучасні моделі VRP значно відрізняються від тих, що були запропоновані Данцигом і Рамсером та Кларком і Райтом [10], оскільки з кожним роком підвищується необхідність враховувати складнощі реального життя (час поїздки, що залежить від дорожнього трафіку; часові вікна для

завантаження та розвантаження; попит, який динамічно змінюється з плином часу; та інші). Ці особливості істотно ускладнюють задачу: найбільші екземпляри VRP, які можуть бути послідовно вирішені за допомогою найбільш ефективних точних алгоритмів, містять близько 20 клієнтів, тоді як більші екземпляри можуть бути вирішені оптимально лише в окремих випадках. Реальні задачі значно більші за масштабами: тисячі клієнтів та десятки складів, чисельні транспортні засоби, різноманітні обмеження, тож евристики та метоевристики значно більше застосовувані для даних об'ємів [10].

Існує кілька варіантів і спеціалізацій проблеми маршрутизації транспортного засобу:

Проблема маршрутизації транспортного засобу з прибутком (VRPP) - проблема максимізації, де відвідування всіх клієнтів не є обов'язковим. Мета задачі - відвідати клієнтів один раз, максимізуючи суму зібраного прибутку, дотримуючись обмеження часу транспортного засобу. Автомобілі повинні починати і закінчувати маршрути в депо. Найбільш відомі і вивчені VRPP:

- задача командного орієнтування (TOP), яка є найбільш вивченим варіантом VRPP;
- проблема командного орієнтування на місцевості (STOP);
- ТОП з вікнами часу (TOPTW).

Проблема маршрутизації транспортного засобу з отриманням і доставкою (VRPPD) - ряд товарів потрібно перемістити з певних місць отримання в інші місця доставки. Мета задачі - знайти оптимальні маршрути для парку транспортних засобів для відвідування місць посадки та висадки.

Проблема маршрутизації транспортного засобу з LIFO - Подібна до VRPPD, за винятком того, що накладено додаткове обмеження на завантаження транспортних засобів: у будь-якому місці доставки товар, який доставляється, має бути останнім отриманим. Ця схема скорочує час завантаження та розвантаження в місцях доставки, оскільки немає

необхідності тимчасово вивантажувати товари, окрім тих, які потрібно висадити [11].

Проблема маршрутизації транспортного засобу з часовими вікнами (VRPTW) - Місця доставки мають часові вікна, протягом яких повинні бути здійснені доставки (або відвідування).

Проблема маршрутизації ємного транспортного засобу: CVRP або CVRPTW. Транспортні засоби мають обмежену вантажопідйомність щодо вантажів, які необхідно доставити.

Проблема маршрутизації транспортного засобу з кількома поїздками (VRPMT) - транспортні засоби можуть виконувати декілька маршрутів.

Відкрита проблема маршрутизації транспортних засобів (OVRP) - транспортним засобам не потрібно повертатися до депо.

Проблема маршрутизації запасів (IRP) - транспортні засоби відповідають за задоволення потреб у кожній точці доставки.

Проблема маршрутизації транспортних засобів із кількома депо (MDVRP) - існує кілька депо, з яких транспортні засоби можуть відправлятися та зупинятися [12].

Проблема маршрутизації транспортних засобів із пересадками (VRPWT) - товари можна переміщувати між транспортними засобами в спеціально призначених пересадочних вузлах.

Незважаючи на те, що VRP пов'язана з проблемою планування робіт, ці дві проблеми зазвичай вирішуються за допомогою різних методів [10].

1.2 Огляд існуючих програмних засобів, що вирішують аналогічні завдання

1.2.1 Програмний продукт LocalSolver

Програмний продукт LocalSolver (рис. 1.1 [13]) — відносно новий розв'язувач математичної оптимізації. Він пропонує високорівневі прості API для Python, Java, C# і C++ для моделювання та вирішення математичних

проблем різних типів, у тому числі й задачі маршрутизації. LocalSolver Optimizer поєднує найсучасніші алгоритми, що робить його швидшим і більш масштабованим, ніж традиційні розв'язувачі.

Основні переваги сервісу:

- легкість моделювання;
- висока швидкість роботи;
- потужні можливості для масштабування;
- сервіс підтримки клієнтів;
- конкурентна ціна.

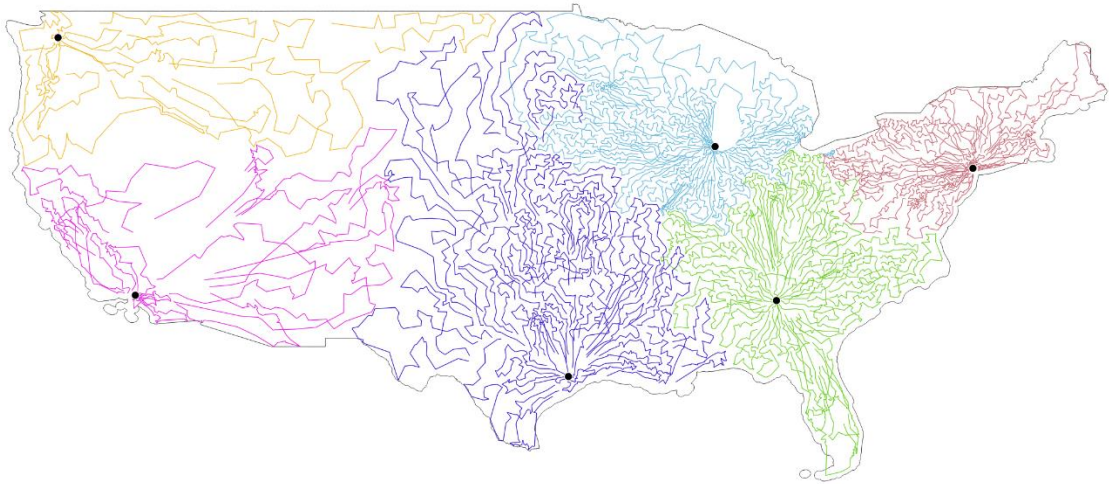


Рисунок 1.1 – Результат роботи LocalSolver [13]

1.2.2 Програмний продукт ArcGis Desktop

Програмний продукт ArcGis Desktop (рис. 1.2 [14]) – програмне забезпечення для математичних робіт з мапами, у тому числі для вирішення задач маршрутизації та виведення результатів на екран. Користувачі ArcMap можуть створювати та маніпулювати наборами даних, щоб включати різноманітну інформацію. Наприклад, карти, створені в ArcMap, зазвичай включають такі функції, як стрілки на північ, масштабні шкали, заголовки, легенди, акуратні лінії тощо. Пакет програмного забезпечення включає набір

стилів цих функцій. А також можливість завантажувати численні інші стилі посилання для застосування до будь-якої функції відображення.

Основні переваги сервісу:

- зручний інтерфейс;
- велика кількість функцій для відображення на мапі;
- можливість 3д-відображення;
- величезна бібліотека документації;
- сервіс підтримки.



Рисунок 1.2 – Програмний продукт ArcGis Desktop [14]

1.2.3 Програмний продукт OptaPlanner

Програмний продукт OptaPlanner (рис. 1.3 [15]) –це засіб визначення обмежень Java™ з відкритим вихідним кодом призначений для оптимізації проблеми маршрутизації транспортного засобу, проблеми комівояжера та подібних випадків використання. Він охоплює будь-який тип планування транспортних груп - літаків, вантажівок, автобусів, таксі, велосипедів і

кораблів, незалежно від того, чи транспортні засоби перевозять продукти чи пасажирів, чи водії надають послуги.

Основні переваги сервісу:

- відкритий код;
- безшкотовність;
- легкість у освоєнні;
- незалежність від типу транспорту;
- велика спільнота розробників.



Рисунок 1.3 – Програмний продукт OptaPlanner [15]

1.2.4 Порівняльна таблиця існуючих та розроблюваного програмних продуктів

Вище було наведено найбільш популярні на сьогоднішній день сервіси та програми, які дозволяють вирішувати задачі маршрутизації. Але крім основних переваг, усі вони мають і недоліки. Наприклад, складний інтерфейс, застарілі методи, відсутність української мови, необхідність завантаження.

Також дійсно існують переваги характерні для усіх програм: великий асортимент методів, певна фінансова доступність.

У ході виконання роботи буде реалізовано програмний продукт, що буде мати наступні переваги, на відміну від існуючих аналогів: українська мова, новіші алгоритми, відсутність потреби у завантаженні, більш широка область застосування і масштабованість, сучасний та інтуїтивно-зрозумілий інтерфейс.

В таблиці 1.1 проведено порівняння за деякими показниками вже існуючих програмних продуктів та програми, яка буде розроблена в ході виконання роботи.

Таблиця 1.1 — Порівняння програмних продуктів

Критерії порівняння	LocalSolver	ArcGis Desktop	OptaPlanner	Створений проєкт
Можливість роботи через Internet	-	-	-	++
Багатоплатформність	-	-	-	+++
Додаткове ПЗ	-	++	-	-
Документація	+	+++	+	+
Ціна	+	+	+	++
Результативність	+	++	+	++

1.3 Постановка завдання роботи

ПП повинен забезпечувати виконання нижче перерахованих функцій:

- введення даних для задачі маршрутизації;
- обробка та валідація введених даних;
- вирішення задачі маршрутизації;
- виведення результатів у текстовому форматі;
- виведення результатів на мапу;
- виведення попереджень при помилках.

1.4 Висновки за розділом 1

В першому розділі було проаналізовано предметну область, її історію та вже існуючі програмні аналоги проєкту.

У ході аналізу було виявлено переваги та недоліки аналогів, функції, що повинен задовольняти проєкт.

Актуальність була доведена зростом кур'єрського бізнесу і необхідністю його оптимізації.

2 ВИБІР ТА ОБГРУНТУВАННЯ СТРУКТУРИ ОБ'ЄКТА, ЯКИЙ ПРОЄКТУЄТЬСЯ

2.1 Вибір інструментарію розробки програмного забезпечення

Розробка проєктів завжди починається з вибору середовища розробки. Цей вибір впливає на зручність роботи з кодом, доступ до фреймворків, можливості рефакторингу, швидкість компіляції. Серед варіантів були наступні програми: Visual Studio, Visual Studio Code, Eclipse.

Microsoft Visual Studio — лінійка продуктів компанії Microsoft, що включають інтегроване середовище розробки програмного забезпечення та інших інструментів. Дані продукти дозволяють розробляти як консольні програми, так і ігри та програми з графічним інтерфейсом, у тому числі з підтримкою технології Windows Forms, UWP а також веб-сайти, веб-додатки, веб-служби як в рідному, так і в керованому коді всіх платформ, підтримуваних Windows, Windows Mobile, Windows CE, .NET Framework, .NET Core, .NET, MAUI, Xbox, Windows Phone .NET Compact Framework та Silverlight. Після покупки компанії Xamarin корпорацією Microsoft з'явилася можливість розробки IOS та Android програм.

Visual Studio включає редактор вихідного коду з підтримкою технології IntelliSense і можливістю найпростішого рефакторингу коду. Вбудований налагоджувач може працювати як відладчик рівня вихідного коду, так і відладчик машинного рівня. Інші вбудовані інструменти включають редактор форм для спрощення створення графічного інтерфейсу програми, веб-редактор, дизайнер класів і дизайнер схеми бази даних [16].

Він дозволяє створювати та підключати сторонні доповнення (плагіни) для розширення функціональності практично на кожному рівні, включаючи додавання підтримки систем контролю версій вихідного коду (як, наприклад, Subversion та Visual SourceSafe), додавання нових наборів інструментів (наприклад, для редагування та візуального проєктування) коду предметно-орієнтованими мовами програмування) або інструментами для

інших аспектів процесу розробки програмного забезпечення (наприклад, клієнт Team Explorer для роботи з Team Foundation Server).

Visual Studio Code, який також зазвичай називають VS Code,— це редактор вихідного коду, створений Microsoft із Electron Framework для Windows, Linux і macOS. Функції включають підтримку налагодження, підсвічування синтаксису, інтелектуальне завершення коду, фрагменти, рефакторинг коду та вбудований Git. Користувачі можуть змінювати тему, комбінації клавіш, параметри та встановлювати розширення, які додають функціональність.

Visual Studio Code — це редактор вихідного коду, який можна використовувати з різними мовами програмування, включаючи C, C#, C++, Fortran, Go, Java, JavaScript, Node.js, Python, Rust. Він базується на структурі Electron, яка використовується для розробки веб-додатків Node.js, які працюють на механізмі компонування Blink. Visual Studio Code використовує той самий компонент редактора (під кодовою назвою «Monaco»), який використовується в Azure DevOps (раніше називався Visual Studio Online і Visual Studio Team Services).

Visual Studio Code містить базову підтримку для більшості поширених мов програмування. Ця базова підтримка включає підсвічування синтаксису, зіставлення дужок, згортання коду та настроювані фрагменти. Visual Studio Code також постачається з IntelliSense для JavaScript, TypeScript, JSON, CSS і HTML, а також підтримує налагодження Node.js. Підтримка додаткових мов може бути забезпечена безкоштовними розширеннями на VS Code Marketplace.

Замість системи проєктів вона дозволяє користувачам відкривати один або кілька каталогів, які потім можна зберегти в робочих областях для подальшого використання. Це дозволяє йому працювати як мовно-агностичний редактор коду для будь-якої мови. Він підтримує багато мов програмування та набір функцій, який відрізняється для кожної мови. Небажані файли та папки можна виключити з дерева проєкту за допомогою

налаштувань. Багато функцій Visual Studio Code не доступні через меню чи інтерфейс користувача, але доступ до них можна отримати через палітру команд [16].

Код Visual Studio можна розширити за допомогою розширень , доступних через центральне сховище. Це включає доповнення до редактора та підтримку мови. Примітною функцією є можливість створювати розширення, які додають підтримку нових мов, тем, налагоджувачів, налагоджувачів подорожей у часі, виконують статичний аналіз коду та додають лінтери коду за допомогою протоколу Language Server .

Eclipse - це інтегроване середовище розробки (IDE), яке використовується в комп'ютерному програмуванні. Він містить базову робочу область і розширювану систему плагінів для налаштування середовища. Це друга за популярністю IDE для розробки Java, і до 2016 року вона була найпопулярнішою. Eclipse здебільшого написаний на Java, і його основне використання — для розробки додатків Java , але його також можна використовувати для розробки додатків на інших мовах програмування за допомогою плагінів, включаючи Ada, ABAP, C, C++, C#, Clojure, COBOL, D, Erlang, Fortran, Groovy, Haskell, JavaScript, Julia, Lasso, Lua, NATURAL, Perl, PHP, Prolog, Python, R, Ruby (включаючи структуру Ruby on Rails), Rust, Scala та Scheme . Його також можна використовувати для розробки документів за допомогою LaTeX (через плагін TeXlipse) і пакетів для програмного забезпечення Mathematica. Середовища розробки включають інструменти розробки Eclipse Java (JDT) для Java і Scala, Eclipse CDT для C/C++ і Eclipse PDT для PHP, серед інших [16].

Для цього проєкту середовищем розробки було обрано Visual Studio Code, по причині простоти його інтерфейсу, швидкості компіляції, взаємодії з фреймворками та додатками. Цей вибір порівнюється із іншими популярними середовищами у таблиці 2.1 за параметрами, які розглядаються без контексту.

Таблиця 2.1 – Порівняння середовищ розробки

Критерії порівняння	Visual studio code	Visual Studio	Eclipse
Перевірка орфографії	++	+	-
Багатоплатформність	++	++	+
Фреймворки	++	++	+
Робота у групі	+++	-	-
Швидкість	-	++	+++
Форматування коду	++	++	++

2.1.1 Огляд особливостей мови програмування

Виконана програма створена на мовах html, javascript та python.

HTML (англ. HyperText Markup Language — мова розмітки гіпертексту) — стандартизована мова розмітки документів для перегляду веб-сторінок у браузері. Веб-браузери отримують HTML документ від сервера за протоколами HTTP/HTTPS або відкривають з локального диска, далі інтерпретують код в інтерфейс, який відображатиметься на екрані монітора.

Елементи HTML є будівельними блоками сторінок HTML. За допомогою конструкцій HTML, зображення та інші об'єкти, такі як інтерактивні форми, можуть бути вбудовані у візуалізовану сторінку. HTML надає засоби для створення структурованих документів, позначаючи структурну семантику тексту, наприклад заголовки, абзаци, списки, посилання, цитати та інші елементи. Елементи HTML окреслені тегами, написаними з використанням кутових дужок. Теги на кшталт `<img />` чи `<input />` безпосередньо виводять вміст на сторінку. Інші теги, такі як `<p>`, оточують текст і надають інформацію про нього, а також можуть включати інші теги як піделементи. Браузери не показують теги HTML, але використовують їх для інтерпретації вмісту сторінки.

В HTML можна вбудовувати програми, написані на скриптових мовах, наприклад JavaScript, які впливають на поведінку та вміст вебсторінок. Включення CSS визначає вигляд і компоновання вмісту. World Wide Web Consortium (W3C), який супроводжує стандарти HTML та CSS, заохочує використання CSS над явним презентаційним HTML з 1997 року.

JavaScript (JS) — динамічна, об'єктно-орієнтована прототипна мова програмування. Реалізація стандарту ECMAScript. Найчастіше використовується для створення сценаріїв вебсторінок, що надає можливість на боці клієнта (пристрої кінцевого користувача) взаємодіяти з користувачем, керувати браузером, асинхронно обмінюватися даними з сервером, змінювати структуру та зовнішній вигляд вебсторінки.

JavaScript класифікують як прототипну (підмножина об'єктно-орієнтованої), скриптову мову програмування з динамічною типізацією. Окрім прототипної, JavaScript також частково підтримує інші парадигми програмування (імперативну та частково функціональну) і деякі відповідні архітектурні властивості, зокрема: динамічна та слабка типізація, автоматичне керування пам'яттю, прототипне наслідування, функції як об'єкти першого класу.

Мова JavaScript використовується для:

- написання сценаріїв вебсторінок для надання їм інтерактивності;
- створення односторінкових та прогресивних вебзастосунків (React, AngularJS, Vue.js);
- програмування на боці сервера (Node.js(Express.js));
- стаціонарних застосунків (Electron, NW.js);
- мобільних застосунків (React Native, Cordova);
- сценаріїв в прикладних програмах (наприклад, в програмах зі складу Adobe Creative Suite чи Apache JMeter);
- всередині PDF-документів тощо.

— Vue надає різноманітні шляхи для застосування ефектів переходу, коли елемент додають, оновлюють або видаляють з DOM. Наприклад:

- автоматичне застосування класів для CSS переходів та анімацій;
- інтегрування сторонніх бібліотек для CSS анімацій, таких як `Animate.css`;
- використовувати JavaScript для прямих маніпуляцій з DOM під час переходів;
- інтегрування сторонніх JavaScript бібліотек анімацій, таких як `Velocity.js`;

Коли елемент, який загорнутий перехідний компонент видаляють чи вставляють, ось що відбувається:

- vue автоматично перевірить чи має застосовують до цього елементу CSS анімації та переходи. Якщо так, CSS класи для переходів будуть додані/видалені у відповідний час;
- якщо перехідний компонент має JavaScript зачіпки, ці зачіпки будуть викликані у відповідний час;
- якщо ніяких CSS переходів/анімацій не було знайдено та ніяких JavaScript не було надано, DOM операції для додавання і/та видалення відбудеться одразу ж в наступному фреймі.

jQuery — популярна JavaScript-бібліотека з відкритим кодом. Вона була представлена у січні 2006 року у BarCamp NYC Джоном Ресігом (John Resig). Згідно з дослідженнями організації W3Techs, JQuery використовується понад половиною від мільйона найвідвідуваніших сайтів. jQuery є найпопулярнішою бібліотекою JavaScript, яка посилено використовується на сьогоднішній день.

jQuery є вільним програмним забезпеченням під ліцензією MIT (до вересня 2012 було подвійне ліцензування під MIT та GNU General Public License другої версії).

Синтаксис jQuery розроблений, щоб зробити орієнтування у навігації зручнішим завдяки вибору елементів DOM, створенню анімації, обробки подій, і розробки AJAX-застосунків. jQuery також надає можливості для розробників, для створення плагінів у верхній частині бібліотеки JavaScript. Використовуючи ці об'єкти, розробники можуть створювати абстракції для низькорівневої взаємодії та створювати анімацію для ефектів високого рівня. Це сприяє створенню потужних і динамічних вебсторінок.

Python (найчастіше вживане прочитання — «Пайтон», запозичено назву з британського шоу Монті Пайтон) — інтерпретована об'єктно-орієнтована мова програмування високого рівня зі строгою динамічною типізацією. Розроблена в 1990 році Гвідо ван Россумом. Структури даних високого рівня разом із динамічною семантикою та динамічним зв'язуванням роблять її привабливою для швидкої розробки програм, а також як засіб поєднування наявних компонентів. Python підтримує модулі та пакети модулів, що сприяє модульності та повторному використанню коду. Інтерпретатор Python та стандартні бібліотеки доступні як у скомпільованій, так і у вихідній формі на всіх основних платформах. В мові програмування Python підтримується кілька парадигм програмування, зокрема: об'єктно-орієнтована, процедурна, функціональна та аспектно-орієнтована.

Серед основних її переваг можна назвати такі:

- переносність програм (що властиве більшості інтерпретованих мов);
- чистий синтаксис (для виділення блоків слід використовувати відступи);
- стандартний дистрибутив має велику кількість корисних модулів (включно з модулем для розробки графічного інтерфейсу);
- можливість використання Python в діалоговому режимі (дуже корисне для експериментування та розв'язання простих задач);

- стандартний дистрибутив має просте, але разом із тим досить потужне середовище розробки, яке зветься IDLE і яке написане мовою Python;

- зручний для розв'язання математичних проблем (має засоби роботи з комплексними числами, може оперувати з цілими числами довільної величини, у діалоговому режимі може використовуватися як потужний калькулятор);

- відкритий код (можливість редагувати його іншими користувачами).

Python має ефективні структури даних високого рівня та простий, але ефективний підхід до об'єктно-орієнтованого програмування. Елегантний синтаксис Python, динамічна обробка типів, а також те, що це інтерпретована мова, роблять її ідеальною для написання скриптів та швидкої розробки прикладних програм у багатьох галузях на більшості платформ.

Інтерпретатор мови Python і багата Стандартна бібліотека (як вихідні тексти, так і бінарні дистрибутиви для всіх основних операційних систем) можуть бути отримані з сайту Python www.python.org [Архівовано 17 квітня 2018 у Wayback Machine.], і можуть вільно розповсюджуватися. Цей самий сайт має дистрибутиви та посилання на численні модулі, програми, утиліти та додаткову документацію.

Інтерпретатор мови Python може бути розширений функціями та типами даних, розробленими на C чи C++ (або на іншій мові, яку можна викликати із C). Python також зручна як мова розширення для прикладних програм, що потребують подальшого налагодження.

Розробники мови Python є прихильниками певної філософії програмування, яку називають «The Zen of Python» («Дзен Пайтона»). Її текст можна отримати в інтерпретаторі Python за допомогою команди `import this` (лише один раз за сесію). Автором цієї філософії вважається Тім Пейтерс [en].

Тези філософії:

- гарне краще за потворне;
- явне краще за неявне;
- просте краще за складне;
- складне краще за заплутане;
- плоске краще за вкладене;
- розріджене краще за щільне;
- легкість читання має значення;
- особливі випадки не є настільки особливими, щоб порушувати норми;
- хоча практичність є важливішою за бездоганність;
- помилки ніколи не повинні проходити непомітно;
- якщо їх приховування не прописано явно;
- зустрівши неоднозначність, опирайтесь спокусі вгадати;
- має бути 1 і, бажано, тільки один - очевидний спосіб зробити це, хоча спочатку він може бути й не очевидним, якщо ви не голландець;
- зараз - краще, ніж ніколи;
- хоча ніколи, найчастіше, - краще, ніж просто зараз;
- якщо реалізацію важко пояснити - задум поганий;
- якщо реалізацію легко пояснити - можливо, задум добрий;
- простори імен - чудова річ, тож робімо їх більше!

2.1.2 Огляд особливостей обраного середовища розробки

Вибір середовища розробки є питанням зручності програміста. Тут грає роль рівень автоматизованості середовища, швидкість запуску, можливість компіляції, гнучкість налаштування, та головне – специфіка проєкту, який буде розроблятися. Розглядаючи конкретно випадок розроблюваного додатку, від середовища розробки не потрібні ні

автоматизація процесу розробки, ні компіляція коду – лише швидкість роботи і гнучність кастомізації, тож було обрано Visual Studio Code (табл. 2.2).

Таблиця 2.2 – Порівняння середовищ розробки

Критерії порівняння	Visual Studio	Visual Studio Code	Eclipse
Багатоплатформність	+	+	+
Швидкість модернізації	+	++	+
Наявність візуального конструктора	++	-	-
Підтримка репозиторіїв	+	+	+
Швидкість компіляції	-	++	++
Додатки	+	++	++
Зручність	+-	++	+

2.2 Структурна схема програми

Програма має клієнт-серверну архітектуру, тож і на структурній схемі будемо розглядати окремо роботу клієнта та сервера.

На стороні клієнта повинні бути модулі для роботи користувача з програмою, але користувач працює на одній інтерактивній сторінці без перезавантажень та переходів на інші сторінки, тож замість модулів були зображені логічні блоки головного модулю для відображення структури роботи клієнтської частини ПО.

На частині серверу, були розроблені модулі: `routing.py`, `Scheduler.py`, `GA.py`, `TS.py`, `Greedy.py`, які будуть більш детально розглянуті у розділах 2.3 та 3.3.

Потік роботи починається на стороні клієнту: користувач вводить у відповідні блоки інформації. Про точки та про транспортні засоби. Після сторона клієнта відправляє дані на сервер і починається їх обробка; `routing.py` викликає методи з `Scheduler.py`, де дані приводяться до зручного для методів формату і після цього, методи починають пошук найефективнішого шляху. Знайдений шлях, разом із додатковими проміжними даними, повертається до `routing.py`, адже лише цей модуль взаємодіє із клієнтом. `Routing.py` повертає клієнту результати, які потрібно відобразити і на стороні клієнта запускається блок отримання даних. В ньому викликаються функції для відображення результату у текстовому і графічному вигляді(на мапі) і користувач наглядно бачить, який шлях програма вважає найкращим і може прийняти власне рішення, опираючись на результати розрахунків.

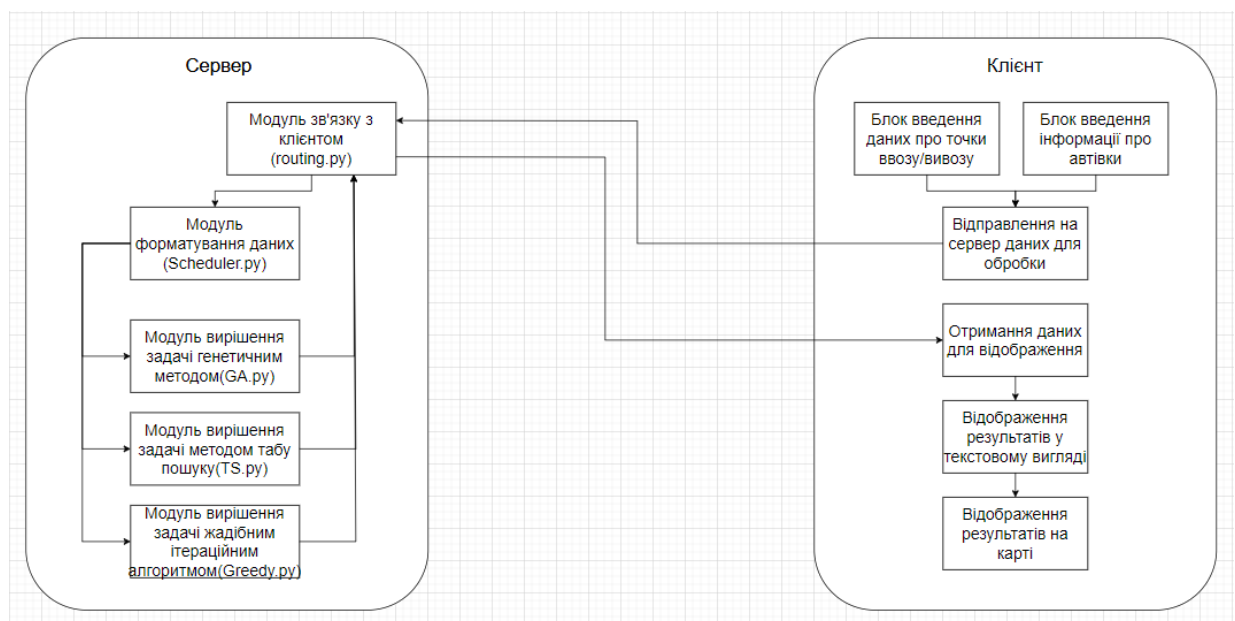


Рисунок 2.1 – Структурна схема ПЗ

2.3 Опис модулів розроблюваної системи

У таблиці 2.3 наведено опис модулів реалізованого ПЗ з визначенням їх основних функцій.

Таблиця 2.3 – Опис модулів реалізованого ПЗ з визначенням основних функцій

Модуль	Опис
routing.py	Модуль зв'язку з клієнтом. Отримує вхідні дані від клієнта, викликає інші модулі для їх обробки та повертає до клієнта результат для відображення.
Scheduler.py	Модуль форматування даних. Збірає методи для обробки вхідних даних, додаткові функції, що використовуються в усіх методах вирішення задачі та форматує дані для повернення клієнту.
GA.py	Модуль вирішення задачі генетичним методом - основний метод програми, з якого береться відображаємий результат.
TS.py	Модуль вирішення задачі методом табу пошуку, розроблений для порівняння ефективності основного методу
Greedy.py	модуль вирішення задачі жадібним методом, розроблений для порівняння ефективності основного методу
Головна сторінка програми	Інтерфейс користувача – складається з файлів main.html, style.css та script.js. Функціонал буде краще описаний у пункті 5 пояснювальної записки.

2.4 Висновки за розділом 2

У другому розділі було розглянуто середовища розробки та мови програмування, обрано певні з них для розробки проекту. Виявлено структуру проекту.

У ході аналізу було виявлено сильні та слабкі сторони розглянутих середовищ, візуалізовано структурну схему та модулі проекту.

3 ОСНОВНІ РІШЕННЯ ЩОДО РЕАЛІЗАЦІЇ КОМПОНЕНТІВ СИСТЕМИ

3.1 Алгоритм функціонування розробленої програми

Основні методи, які можуть виконуватися при роботі із розробленою програмою зображені на рис. 3.1.



Рисунок 3.1 – Функціональна схема розробленої програми

На схемі зображено потік даних по функціональних блоках програми. Перша функція викликається при завантаженні програми клієнтом – відображення гугл мапи завдяки апі. Після цього, в окремому полі користувач починає вводити адреси точок доставки, а з апі гугл мап йому пропонуються адреси, що можуть бути відображені, максимально наближені до вводу користувача. Після вибору адреси користувачем, програма отримує координати точки з того ж апі і відправляє зібрані дані на сервер.

На сервері дані формуються у зручний для методів формат і відправляються всередину трьох методів: генетичний, жадібний та табу пошуку оптимального шляху. Коли всі методи повертають свої результати,

програма бере результат основного(генетичного) методу і розподілює автівки по точках. Так, як у точках ще є параметри вантажу, розподілення буде враховувати і їх, а задача переросте у задачу тривимірної упаковки, яка вирішується у програмі методом перебору.

Отриманий результат форматується для виводу і відправляється на сторону клієнта. На стороні клієнта результат відображається текстовим і графічним способами. Текст, сформований на сервері, просто виводиться на екран, а от для графічного знову звертаємося до апі гугл мап. Через відправлення координат точок до розділу апі «directions service», на мапі з'являється графічне зображення знайденого шляху.

Як видно зі схеми під час розробки ПП і як було розглянуто раніше, були розроблені наступні методи:

- генетичний метод вирішення задачі маршрутизації;
- метод пошуку з табу для вирішення задачі маршрутизації;
- жадібний метод вирішення задачі маршрутизації;
- метод вирішення задачі тривимірної упаковки методом перебору.

3.1.1 Генетичний метод вирішення задачі маршрутизації

Для розуміння роботи методу, наведемо текстовий опис генетичного методу для вирішення звичайної задачі маршрутизації.

Спочатку подається рішення у вигляді хромосоми: Кожен потенційний розподіл об'єктів можна представити як хромосому генетичного методу. У випадку задачі маршрутизації, хромосома може містити інформацію про розподіл об'єктів, наприклад, які об'єкти перевозяться якою машині або вантажівці.

Після цього визначаються функції пристосованості: Функція пристосованості оцінює якість кожного розподілу об'єктів. В задачі маршрутизації, це може бути об'єктивна функція, яка враховує витрати на

перевезення, час доставки або інші критерії ефективності. Прикладом такої функції може послужити формула (3.1):

$$fitness[i] = \sum(c[i, j]), \quad (3.1)$$

де $c[i, j]$ – витрати на доставку, або відстань між точками.

Потім потрібно створити генетичні оператори: Генетичний метод використовує такі генетичні оператори, як схрещування (crossover), мутація (mutation) та відбір (selection), щоб генерувати нові хромосоми та покращувати рішення з кожним поколінням.

Далі йде ітеративний процес: Генетичний метод виконує ітеративний процес, де нові хромосоми створюються, оцінюються за допомогою функції пристосованості, піддаються генетичним операціям і відбору. Цей процес повторюється протягом декількох поколінь досягнення оптимального рішення.

Ймовірність вибору кожної i -ї хромосоми визначається у вигляді формули (3.2):

$$p(x_i) = \frac{1}{K} \left(a - (2a - 2) \frac{i-1}{K-1} \right), i \in 1, K, \quad (3.2)$$

де K – кількість хромосом;

a – коефіцієнт оновлення;

a приймає значення в діапазоні від 1 до 2. Значення a більше 1 робить функцію більш пологою, тобто ймовірність вибору менших індексів стає більшою порівняно з ймовірністю вибору більших індексів.

Наступним кроком є схрещування - відбувається комбінація генетичної інформації з двох батьківських хромосом, щоб створити нові дитячі хромосоми. У випадку задачі маршрутизації, це може означати комбінування розподілу об'єктів з різних батьківських хромосом для отримання нового потенційного розподілу, після чого іноді виконується

мутація, яка змінює деякі властивості хромосоми випадковим чином. У задачі маршрутизації це може означати випадкову зміну асоціацій об'єктів та транспортних засобів.

Після цього відбувається вибір найкращих хромосом для наступного покоління. Чим краще функція пристосованості хромосоми, тим більша ймовірність, що вона буде відібрана для продовження у наступному поколінні. Метод продовжує ітерацію через попередні кроки до досягнення критерію зупинки. Це може бути досягнення заданої кількості поколінь, зближення до оптимального розв'язку або задоволення іншої зупинки, встановленої користувачем.

Робота завершується оцінкою розв'язку - метод оцінює кращу знайдену хромосому та повертає її як розв'язок задачі.

3.1.2 Модифікація генетичного методу «подвоєні хромосоми»

Через специфікацію задачі, звичайний генетичний метод для вирішення VRP було змінено:

Представлення хромосоми: Зазвичай, кожен ген у хромосомі відповідає відвідуванню однієї точки. В VRPPD, кожна точка може мати як поставки, так і забори, тому гени повинні кодувати обидва типи діяльності. У нашому випадку, це досягається шляхом подвоєння координат, що зберігає кожен ген; Із представленням, змінилася і формула вибору хромосоми для мутації та схрещування на формулу (3.3):

$$p(x_i) = \frac{1}{k} \exp\left(-\frac{1}{T(n)}\right) + \frac{F1(x_i)}{\sum_{s=1}^K F1(x_s)} \left(1 - \exp\left(-\frac{1}{T(n)}\right)\right), i \in 1, K, \quad (3.3)$$

де K – кількість хромосом;

$$T(n) = \beta T(n-1), 0 < \beta < 1;$$

n - номер ітерації.

До того ж, була змінена функція пристосованості: Функція пристосованості повинна враховувати обидва аспекти задачі - доставку та забір товарів. Вона повинна оцінювати якість розподілу вантажів на маршрути, дотримання обмежень щодо ємності транспортних засобів та мінімізувати загальну відстань маршрутів і функція пристосованості отримала вигляд формули (3.4):

$$F(x) = d_{x_M x_1} + \sum_{i=1}^{M-1} d_{x_i x_{i+1}} \rightarrow \min, \quad (3.4)$$

де $d_{x_M x_1}$ – довжина шляху між точками;

M – кількість точок.

Також визначимо ймовірність мутації гену за формулою (3.5):

$$p_0 = p_0 \exp\left(-\frac{1}{T(n)}\right), \quad (3.5)$$

де $T(n) = \beta T(n-1)$, $0 < \beta < 1$, $T_0 > 0$;

n – номер покоління

Вихідними даними модифікованого методу, буде найефективніший шлях між точками, але розподіл точок за автівками має бути обмеженим габаритами та/або вагою вантажу. В нашому випадку, вантаж представлено і габаритами, у вигляді трьох параметрів, і вагою, а в автівок є максимально допустимі параметри всіх чотирьох. Якщо відійти від задачі маршрутизації, у цих умовах можна побачити задачу тривимірної упаковки, рішення якої буде розглянуто у пункті 3.1.4.

До додаткових модифікацій, слід віднести пошук відстані між двома точками. Було розглянуто багато способів отримання цієї інформації, серед яких найточнішим було звернення до google maps api, проте у пр-задачі час виконання і відсутність помилок можуть бути важливіші за точність, тож було вирішено вираховувати відстань через формулу гаверсінуса.

Формула гаверсинуса (3.6) — важливе рівняння у навігації, яке дозволяє обчислити відстань між точками на сфері, за їхніми довготою та широтою. І цей підхід до отримання відстані дозволяє істотно збільшити швидкість роботи програми:

$$\text{havsine}\left(\frac{d}{r}\right) = \text{havsine}(\varphi_2 - \varphi_1) + \cos(\varphi_1) \text{havsine}(a_2 - a_1), \quad (3.6)$$

де d - відстань між двома точками;

r - радіус сфери;

φ_1, φ_2 : широта точки 1 та широта точки 2;

a_1, a_2 : довгота точки 1 та довгота точки 2.

Сферою в нас виступає земля, тож радіус беремо середній по землі і він приблизно дорівнює 6371км, а усі інші дані ми вже отримали з google maps api при виборі точок користувачем.

3.1.3 Метод пошуку з табу для вирішення задачі маршрутизації

Метод пошуку з табу (Tabu Search) є евристичним методом оптимізації. Основна ідея методу полягає в ефективному обході простору рішень, униканні уловов локального оптимуму та знаходженні найкращого розв'язку.

У чомі відмінність від інших методів, простежимо із математичного опису методу (3.7),

$$\min \{F(x) | x \in N(x_k) \setminus \bigcup_{i=0}^k x_i\}, \quad (3.7)$$

де множина x_i від 0 до k - список заборон.

Основні кроки методу пошуку з табу для вирішення задачі маршрутизації:

Встановлюються початкові значення, маршрутизація транспортних засобів та обчислюється функція вартості для поточного розподілу замовлень та маршрутів, після чого виконуються операції для генерації сусідніх рішень, такі як перестановка транспорту, зміна послідовності відвідування точок або перерозподіл ресурсів.

Після встановлюються обмеження на деякі рухи або операції, які не можуть бути виконані або повторюватись. Це допомагає уникнути повторення поганої або непродуктивної дії. Наступним кроком є вибір кращого рішення - знаходиться найкраще сусіднє рішення з урахуванням функції пристосованості та критеріїв табу. Вносяться зміни в табу-список, щоб відображати заборонені або обмежені рухи або операції та перевіряється, чи був досягнутий критерій зупинки - максимальна кількість ітерацій. Процес повторюється доти, поки не буде досягнутий критерій зупинки.

На кожному кроці нове рішення вираховується з умовами (3.8):

- $z \in N^*(x)$;
- $F(z) = \min \{F(y) | y \in N^*(x)\}$;
- перехід $x \rightarrow z$ не є забороненим або $F(z) < F_{pz}$. (3.8)

При $F(x) < F_{pz}$, змінюємо F_{pz} – рекорд на ітераційному кроці.

Модифікації методу відповідні постановці задачі:

- функція пристосованості: Функція пристосованості повинна враховувати обидва аспекти задачі - доставку та забір товарів. Вона повинна оцінювати якість розподілу вантажів на маршрути, дотримання обмежень щодо ємності транспортних засобів та мінімізувати загальну відстань маршрутів.

- оцінка якості: оцінка якості обчислюється як сума відстані між двома точками доставки із відстанню між двома маршрутами доставки і показує якість обраного шляху (3.9):

$$f(x) = (\sum_{i=2}^n \sqrt{(x_i - x_{i-1})^2 + (y_i - y_{i-1})^2} + \sqrt{(x_1 - x_n)^2 + (y_1 - y_n)^2}), \quad (3.9)$$

де x і y – координати точок.

Структури пам'яті, які використовуються в пошуку табу, можна приблизно розділити на три категорії:

- короткостроковий: список нещодавно розглянутих рішень. Якщо потенційне рішення з'являється в списку табу, його не можна переглянути, поки воно не досягне терміну дії;
- середньостроковий: правила інтенсифікації, призначені для зміщення пошуку в бік перспективних областей простору пошуку;
- довгостроковий: правила диверсифікації, які спрямовують пошук у нові регіони (тобто щодо скидання, коли пошук застряє на плато або в неоптимальному глухому куті).

На практиці короткочасна, середньострокова та довгострокова пам'ять можуть збігатися. У межах цих категорій пам'ять можна далі диференціювати за такими показниками, як частота та вплив внесених змін. Одним із прикладів середньострокової структури пам'яті є така, яка забороняє або заохочує рішення, які містять певні атрибути (наприклад, рішення, які включають небажані або бажані значення для певних змінних), або структура пам'яті, яка запобігає або спонукає певні рухи (наприклад, на основі частотної пам'яті, застосованої до рішень, які мають спільні риси з непривабливими або привабливими рішеннями, знайденими в минулому). У короткочасній пам'яті вибрані атрибути нещодавно відвіданих рішень позначаються як «tabu-active». Розчини, що містять табу-активні елементи, заборонені. Критерії прагнення використовуються для перевизначення стану табу рішення, таким чином включаючи рішення, виключене інакше, до дозволеного набору (за умови, що рішення є «достатньо хорошим» відповідно до показника якості або різноманітності). Простий і широко використовуваний критерій прагнення полягає в тому, щоб дозволити рішення, які є кращими, ніж відомі на даний момент найкращі рішення.

Лише короткочасної пам'яті може бути достатньо, щоб отримати рішення, які перевершують ті, що знайдені звичайними методами локального пошуку, але проміжні та довготривалі структури часто необхідні для вирішення складніших проблем. Пошук табу часто порівнюють з іншими метаевристичними методами, такими як симуляція відпалу, генетичні методи, методи оптимізації мурашиних колоній, оптимізація реактивного пошуку, керований локальний пошук або жадібний рандомізований адаптивний пошук. Крім того, пошук табу іноді поєднується з іншими метаевристичними методами для створення гібридних методів. Найпоширеніший гібрид пошуку табу виникає шляхом об'єднання TS із пошуком розкиду, класу процедур на основі генеральної сукупності, які мають спільні корені з пошуком табу та часто використовуються для вирішення великих нелінійних задач оптимізації.

3.1.4 Жадібний метод вирішення задачі маршрутизації

Жадібний метод (Greedy algorithm) є простим та евристичним підходом до вирішення задачі маршрутизації. Він працює шляхом поетапного вибору оптимального рішення на кожному кроці, на основі локальних оптимальних виборів, без урахування глобальної оптимальності.

Основна ідея жадібного методу вирішення задачі маршрутизації полягає в наступних кроках:

— спочатку встановлюються початкові значення, такі як початкове розподілення замовлень, маршрутизація транспортних засобів тощо, після чого вибирається локально оптимальне рішення - на кожному кроці вибирається оптимальний вибір найближчої точки доставки. Обраний вибір виконується шляхом призначення замовлення до відповідного транспортного засобу та розподілу замовлень;

- після цього необхідно перевірити, чи був досягнутий критерій зупинки, такий як оптимальність розподілу, максимальна кількість ітерацій чи заданий ліміт часу;

- цей процес повторюється доти, поки не буде досягнутий критерій зупинки.

Жадібний метод — це простий та інтуїтивно зрозумілий підхід до вирішення TSP. Хоча він не завжди може створювати оптимальне рішення, він може забезпечити достатньо хороші рішення за прийнятний проміжок часу.

Важливо пам'ятати, що жадібний метод не гарантує знайдення оптимального розв'язку, оскільки він здійснює локальні оптимальні вибори на кожному кроці і його точність нижча з усіх наведених. Часто жадібний метод використовується як етап побудови початкового розв'язку для більш складних методів оптимізації, проте в нашому випадку він використовувався лише для порівняння результатів.

Було запропоновано кілька модифікацій і варіацій жадібного методу для покращення його продуктивності, наприклад метод 2-Opt, який ітеративно покращує тур шляхом заміни пар ребер для зменшення загальної відстані.

Проведені в методі модифікації вже були зазначені в попередніх методах, тож не будемо їх повторювати.

3.1.5 Метод мурашиної колонії

Мурашиний метод (метод оптимізації наслідуванням мурашиної колонії, ant colony optimization, ACO) - один з ефективних поліноміальних методів для знаходження наближених розв'язків задачі комівояжера, а також вирішення аналогічних завдань пошуку маршрутів на графах. Належить до метаевристичних оптимізацій і має деякі подібності з оцінкою методів

розподілу. Суть підходу полягає в аналізі та використанні моделі поведінки мурах, що шукають шляхи від колонії до джерела живлення.

Оптимізація колонії мурашок — це клас методів оптимізації, змодельованих на основі дій колонії мурашок. Штучні «мурахи» (агенти моделювання) знаходять оптимальні рішення, пересуваючись простором параметрів, що представляє всі можливі рішення. Справжні мурахи виділяють феромони, спрямовуючи один одного на ресурси, досліджуючи навколишнє середовище. Змодельовані «мурахи» так само записують свої позиції та якість своїх рішень, так що в наступних ітераціях моделювання більше мурах знаходять кращі рішення.

В методах оптимізації мурашиної колонії штучна мураха є простим обчислювальним агентом, який шукає хороші рішення для даної проблеми оптимізації. Щоб застосувати метод мурашиної колонії, задачу оптимізації потрібно перетворити на задачу пошуку найкоротшого шляху на зваженому графі. На першому кроці кожної ітерації кожна мураха стохастично будує рішення, тобто порядок, у якому слід дотримуватися ребер на графіку. На другому кроці порівнюються шляхи, знайдені різними мурахами. Останній крок полягає в оновленні рівнів феромонів на кожному краю.

Кожна мурашка повинна побудувати рішення, щоб переміщатися по графіку. Щоб вибрати наступне ребро у своєму турі, мураха розглядатиме довжину кожного ребра, доступного з його поточної позиції, а також відповідний рівень феромону. На кожному кроці методу кожна мураха переходить із стану x у стан y , що відповідає більш повному проміжному розв'язку. Таким чином, кожна мураха k обчислює набір $A_k(x)$ можливих розширень до свого поточного стану на кожній ітерації та переходить до одного з них за ймовірністю. Для мурашки k ймовірність p_{xy}^k переходу зі стану x до стану y залежить від комбінації двох значень, привабливості n_{xy} переміщення, як обчислюється деякою евристикою що вказує на апріорну бажаність цього ходу та рівень сліду t_{xy} цього ходу, вказуючи на те,

наскільки досвідченим був цей конкретний хід у минулому. Рівень сліду є апостеріорним показником бажаності цього руху.

Загалом, k -та мураха переходить із стану x у стан y з імовірністю, що розраховується за формулою (3.10):

$$p_{xy}^k = \frac{(t_{xy}^a)(n_{xy}^b)}{\sum_{z \in \text{allowed}} (t_{xz}^a)(t_{kz}^b)}, \quad (3.10)$$

де (t_{xy}^a) — це кількість феромону, що осідає для переходу від стану x до y ;

$0 \leq \alpha$ — параметр для керування впливом (t_{xy}^a) ;

(n_{xy}^b) — це бажаність переходу стану x у стан y (апріорне знання, як правило, $1/d_{xy}$, де d — відстань);

$\beta \geq 1$ — це параметр для керування впливом (n_{xy}^b) ;

(t_{xz}^{ab}) і (n_{xz}^b) представляють рівень сліду та привабливість для інших можливих переходів стану.

Сліди зазвичай оновлюються, коли всі мурахи завершили своє рішення, збільшуючи або зменшуючи рівень слідів, що відповідають ходам, які були частиною «хороших» або «поганих» рішень відповідно. Прикладом глобального правила оновлення феромонів є формула (3.11):

$$t_{xy} \leftarrow (1 - \rho)t_{xy} + \sum_k^m \Delta t_{xy}^k, \quad (3.11)$$

де t_{xy} — кількість феромону, що осідає для переходу стану x у стан y ;

ρ — коефіцієнт випаровування феромону;

m — кількість мурах;

Δt_{xy}^k — це кількість феромону, що виділяється k -тою мурахою.

Кількість феромону зазвичай задана для проблеми TSP (з ходами, що відповідають дугам графіка) за формулою (3.12):

$$\Delta t_{xy}^k = \begin{cases} \frac{Q}{L_k} & \text{якщо } k \text{ перетинає } xy \text{ на своєму шляху} \\ 0 & \text{інакше} \end{cases}, \quad (3.12)$$

де L_k — вартість k -го туру мурахи (зазвичайна довжина);

Q — розрахована константа.

3.1.6 Метод вирішення задачі тривимірної упаковки методом перебору

В пункті 3.1.1, в результаті роботи методу, було отримано умови задачі тривимірної упаковки. Потрібно розфасувати вантажі, в яких є параметри: ширина/висота/довжина/вага по автівках, в яких є усі відповідні параметри. За певних умов, подібна задача також стає пр-складною і потребує використання метаевристичних або оптимізаційних методів вирішення, проте в нашому випадку було використано метод перебору варіантів, що дозволило збільшити точність результату.

Метод вирішення задачі тривимірної упаковки методом перебору (Brute Force) передбачає перебір всіх можливих комбінацій розташування вантажів у транспортному засобі з метою знаходження найефективнішого розв'язку.

Метод використовує рекурсивний підхід для перебору всіх можливих комбінацій розташування вантажів у транспортному засобі. Основні кроки методу такі:

Процес починається з визначення розміру контейнера і списку об'єктів, які необхідно розмістити. Усі можливі комбінації розташування об'єктів у автівці перебираються рекурсивним методом. Під час перебору перевіряється валідність кожної комбінації, щоб переконатися, що всі об'єкти вміщаються у контейнер і не перекриваються, їх вага не більше максимальної для засобу і т.д;

Кожній валідній комбінації присвоюється оцінка відповідно до критеріїв оптимальності, в нашому випадку це максимальна завантаженість

транспортного засобу та обирається комбінація з найкращою оцінкою, яка відповідає оптимальному розташуванню об'єктів у контейнері;

Процес завершається поверненням оптимальне розташування об'єктів у контейнері.

В цьому методі, з модифікацій лише додавання ваги до габаритів вантажу. Як вже було зазначено, метод перебору не є найкращим рішенням для цієї задачі за умови збільшення об'єму вхідних даних, проте умови задачі не передбачають подібних проблем.

3.2 Вибір шаблону проєктування

На рис. 3.2 ви можете побачити даграму композитної структури, на якій відображено зв'язки між класами програми. Почнемо з класів, які використовуються усіма компонентами – «Dot» і «Car». Клас «Dot» зберігає інформацію про точку на карті: координати, розміри грузу, який потрібно забрати і його вага. Клас «Car» зберігає максимальну вагу, максимальні розміри, скільки засіб витрачає палива на 100 кілометрів і скільки коштує літр палива для засобу. Ці класи використовуються для заповнення даними масивів у «routing» та передаються у «VRPSolver», де проводяться розрахунки шляхів. Проте шляхи у результаті повинні бути прив'язані до автівок і тому ми наслідуюмо класом «CarRes» клас «Car» і у «VRPSolver» копіюємо дані до об'єктів нового класу, не змінюючи оригінал. Новий клас же містить методи для заповнення нових полів – заповненість, взяті вантажі і пройдений шлях. Тепер можна повернути результат масивом об'єктів нового класу і зручно оброблювати ці дані. Говорячи про шаблони проєктування, можемо виділити вже на цій діграмі кілька з них і розглянути далі. Складність виділення шаблонів проєктування полягає у тому, що для їх наглядності потрібно зіштовхнутися із розширенням проєкту – додаванні функціоналу, даних або і того і іншого одночасно. До цієї події, класи виглядають просто взаємопов'язаними.

Адаптер — це структурний патерн проєктування, що дає змогу об'єктам із несумісними інтерфейсами працювати разом. У нашій діаграмі цей патерн використовується двічі: у «routing» створюються дані типів «Dot» і «Car» і у «VRPSolver» створюється масив типу «CarRes». В обох випадках масиви були створені лише для того, щоб передати їх в інший клас, де очікуються дані певного типу і ці класи виступають адаптерами.

Шаблонний метод — це поведінковий патерн проєктування, який визначає кістяк методу, перекладаючи відповідальність за деякі його кроки на підкласи. Патерн дозволяє підкласам перевизначати кроки методу, не змінюючи його загальної структури. На рис. 3.2 можна його побачити у класах «Car» і «CarRes», а також «routing» і «VRPSolver». Почнемо з останнього: у «routing» цілком можна було б одразу знайти шлях, проте замість цього, тут лише викликається клас для пошуку шляху, адже при наявності другого методу вирішення задачі, буде куди легше викликати його із цього місця, ніж вигадувати способи розділити 2 методи в одному файлі. Що до автовок, «CarRes» - це наш формат відповіді, що дозволяє у випадку розширення функціоналу програми, зробити декілька шаблонів відповідей і відрізнити їх по класу результату.

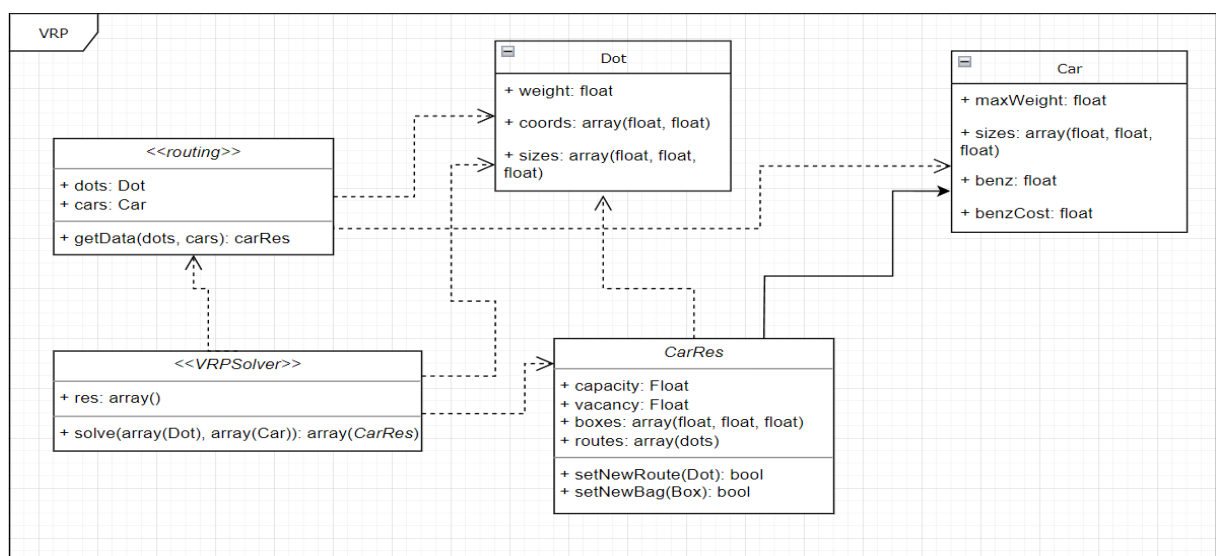


Рисунок 3.2– Uml-діаграма композитної структури класів для вирішення задачі

Ітератор — це поведінковий патерн проєктування, що дає змогу послідовно обходити елементи складових об'єктів, не розкриваючи їхньої внутрішньої організації. На рис. 3.2 не було зображено роботу ітератора, адже клас «VRPSolver» включає повністю вирішення задачі, а не розписаний поетапно, проте в ньому всередині використовуються ітератори, наприклад, для одночасного проходження по колекції, як це зображено на рис. 3.3.

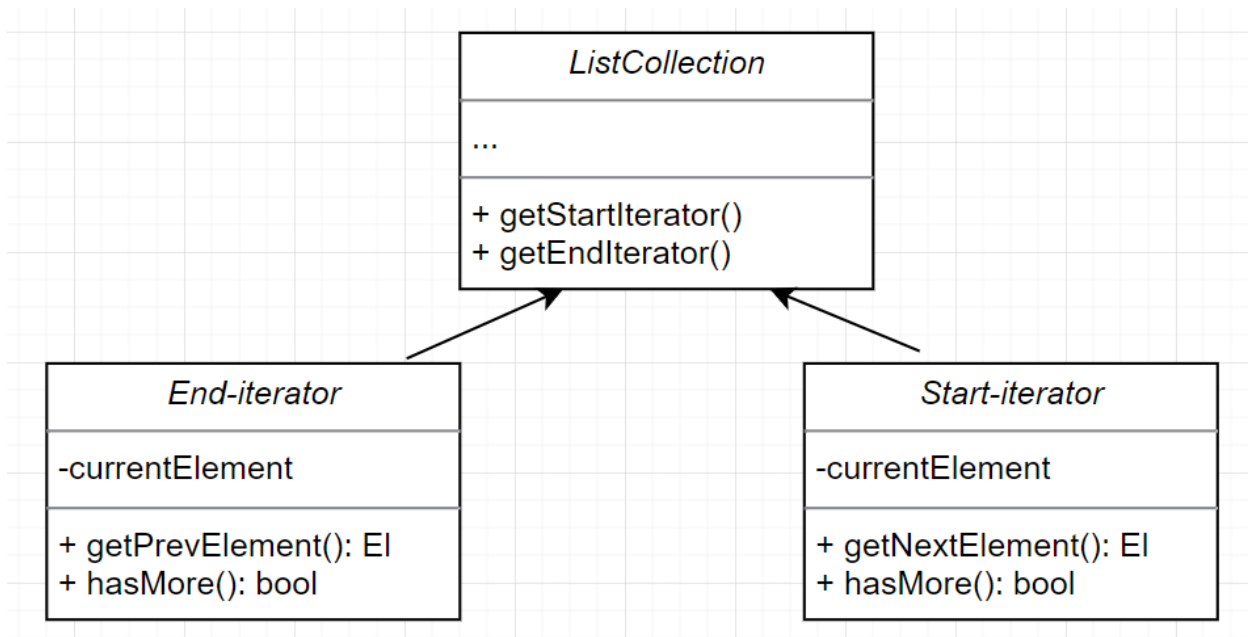


Рисунок 3.3 – Приклад ітератора

Патерни проєктування прискорюють роботу, полегшують супровід ПЗ і дають можливість подальших його розширень, проте для їх правильного використання потрібен чималий досвід у проєктуванні і програмуванні одночасно, адже після проєктування настає час реалізації і після цього розділу настає розділ огляду розроблених модулів.

3.3 Опис модулів розроблюваної системи

У розділі 2.3 вже були коротко розглянуті модулі додатку, проте зараз зобразимо їх на uml-діаграмі, яку можна побачити на рис. 3.4 для наочності.

На цій діаграмі, зображені усі користувальницькі класи, що були використані у програмі. Для 3 методів вирішення задачі по класу і по класу для точки і автівки. Кожен клас методу розв’язання займає окремий модуль, а «Vehicle» і «Dot» знаходяться у модулі «Scheduler». Модуль «routing» не має окремих класів всередині, адже у цьому модулі викликаються усі інші.

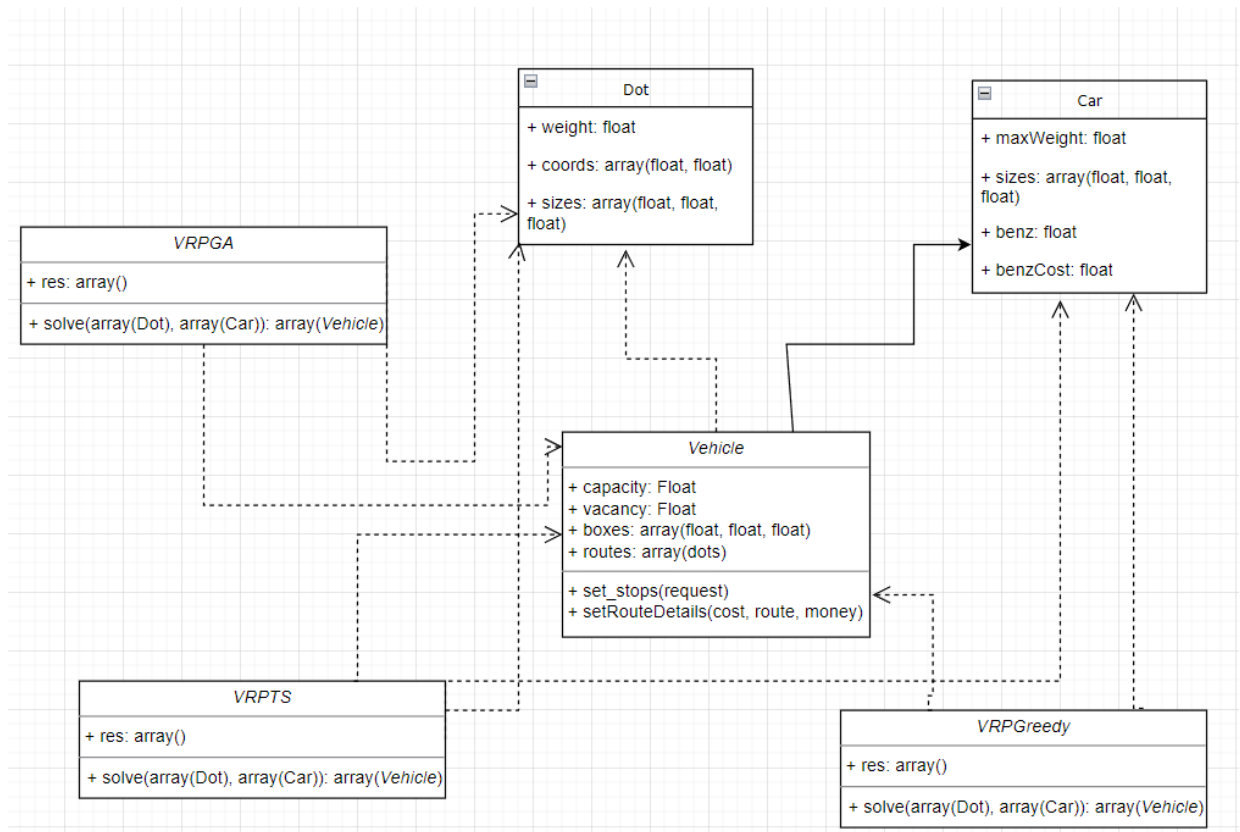


Рисунок 3.4 – Uml діаграма класів

3.4 Висновки за розділом 3

У третьому розділі було розглянуто функціональну схеми програми, були виявлені зв’язки між її модулями, визначені паттерни проєктування, що використовувалися при розробці і розглянуто потік виконання функцій програмою.

4 КЕРІВНИЦТВО ПРОГРАМІСТА

4.1 Призначення й умови застосування програми

Програма призначена для вирішення задачі маршрутизації. Це забезпечить оптимальне використання ресурсів підприємства доставки.

Програма виконує наступні функції:

- введення початкових даних;
- розрахунок оптимальних маршрутів для введених даних;
- виведення розрахованих маршрутів на мапі.

Для нормального функціонування програми, повинні виконуватися наступні умови до апаратної частини сервера:

- Pentium-сумісний процесор x86 архітектури з частотою 1ГГц і вище з підтримкою PAE, та NX SSE2 або система на кристалі;
- 4 ГБ оперативної пам'яті;
- відеокарта з підтримкою DirectX 9 та WDDM версії 1.3 або пізніше;
- 20 гб або вище вільного місця на диску;
- дозвіл екрану 800 x 600, або більше;
- клавіатура і миша або аналогічний пристрій введення;
- Для нормального функціонування програми, повинні виконуватися наступні умови до апаратної частини клієнту:

- Pentium-сумісний процесор з частотою 1ГГц IA-32 або вище;
- 1 ГБ оперативної пам'яті;
- відеокарта з підтримкою DirectX 9 та WDDM версії 1.0 або пізніше
- 16 гб або вище вільного місця на диску;
- клавіатура і миша або аналогічний пристрій введення;

Умови до програмної частини, для нормального функціонування програми:

- операційна система Windows 7;
- браузер з підтримкою HTML версії 4 або пізніше , CSS3, ECMAScript 3.7 або пізніше.

4.2 Характеристики програми

Додаток виконує наступні функції:

- виведення мапи;
- скачування інформації про об'єкти на мапі;
- фільтрація списку місць, відносно введеної строки;
- вибір місць із списку для відповідних точок маршруту;
- введення ваги доставки маршруту;
- введення габаритів доставки маршруту;
- введення ціни доставки товару;
- можливість змінювати інформацію про маршрути;
- додавання нових маршрутів до списку;
- введення вантажопід'ємності ТЗ;
- введення габаритів ТЗ;
- зміна інформації про ТЗ;
- додавання нових ТЗ до списку;
- розрахунок оптимальних маршрутів;
- виведення розрахованих маршрутів;
- виведення ціни кожного маршруту;
- візуалізація розрахованих маршрутів на мапі.

4.3 Звертання до програми

Для доступу до додатку, потрібно емулювати роботу хоста на локальному сервері, перевірити наявність виконуваної директорії та ввести її

адресу у пошуковому рядку браузера, або запустити на виконання файл routing.py із кореневої директорії.

Для доступу до додатку користувача необхідно лише ввести адресу у пошуковому рядку.

4.4 Початкові та вихідні дані

Початковими даними програми є дані про точки маршруту, габарити, вагу товару; габарити, вантажо-під'ємність та бензиновмісткість ТЗ.

Вихідними даними є оптимальні маршрути та їх ціни.

4.5 Повідомлення програмісту

Під час роботи розроблений ПП генерує повідомлення, що описані в табл. 4.1.

Таблиця 4.1 – Приклад виводу даних додатку на стороні клієнта передбаченою функцією

Тип повідомлення	Опис виводимих даних
Dots	Повідомлення означає, які точки були обрані користувачем
Dotsid	Повідомлення означає координати обраних точок
Cars	Повідомлення означає характеристики транспортних засобів
Weights	Повідомлення означає характеристики товарів
Помилка: Недостатньо інформації у маршруті	Повідомлення означає, що у певному маршруті не вистачає інформації
Помилка: Нульові характеристики у транспортному засобі	Повідомлення означає, що у характеристиках певного транспортного засобу нульові характеристики

4.6 Висновок

В даному розділі було проведено детальний опис призначення та функції розробленого ПП. Були визначені умови, необхідні для його встановлення та підтримки на комп'ютері користувача та сервері. Описані основні характеристики та особливості створеного ПП. Розроблений ПП має відкритий код, тому він може редагуватися іншими розробниками ПЗ та його модулі можуть бути використані для створення інших систем. Також в даному розділі наведено опис вхідних і вихідних даних та перелік повідомлень, що генеруються під час роботи.

5 КЕРІВНИЦТВО ОПЕРАТОРА

5.1 Призначення програми

Додаток призначений для часткової автоматизації роботи кур'єрського підприємства шляхом рішення задач маршрутизації транспорту. Одним з основних завдань планування маршрутів доставок є розміщення товарів, а саме розміщення n товарів різних та однакових типорозмірів та одночасного закріплення за ними транспортних засобів.

Додаток створено в текстовому редакторі Visual Studio code.

5.2 Умови виконання програми

Для емулювання роботи серверу на локальному хості достатньо запустити файл на мові пайтон routing.py

Клієнтській частині ж потрібен лише браузер на вибір користувача(хоча Internet explorer може не підтримувати стилі, що використані на сторінках) та стабільний доступ до інтернету.

Після налаштування серверної й клієнтської частини, можна перейти до перегляду додатку. Для цього, слід перейти у будь-якому браузері за посиланням <https://localhost/>.

Прграмний продукт призначений для запуску в середовищі Windows на ЕОМ з наступними параметрами:

- Pentium-сумісний процесор з частотою 1ГГц IA-32 або вище;
- 1 ГБ оперативної пам'яті;
- відеокарта з підтримкою DirectX 9 та WDM версії 1.0 або пізніше;
- 16 гб або вище вільного місця на диску;
- клавіатура і миша або аналогічний пристрій введення;
- операційна система Windows 7;

- операційне забезпечення для хостингу серверів Open Server версії 4.0.0 або пізніше;
- браузер з підтримкою HTML версії 4 або пізніше , CSS3, ECMAScript 3.7 або пізніше.

5.3 Виконання програми

Для початку роботи необхідно перейти за посиланням у браузері. Праворуч буде зображено мапу, зліва – головне меню (рис. 5.1) Після цього необхідно ввести початкові дані – точки та ТЗ.

Головне меню складається з налаштувань маршруту та транспортних засобів. У налаштуванні маршруту можна додати новий маршрут, визначити адреси точок та розрахувати маршрут.

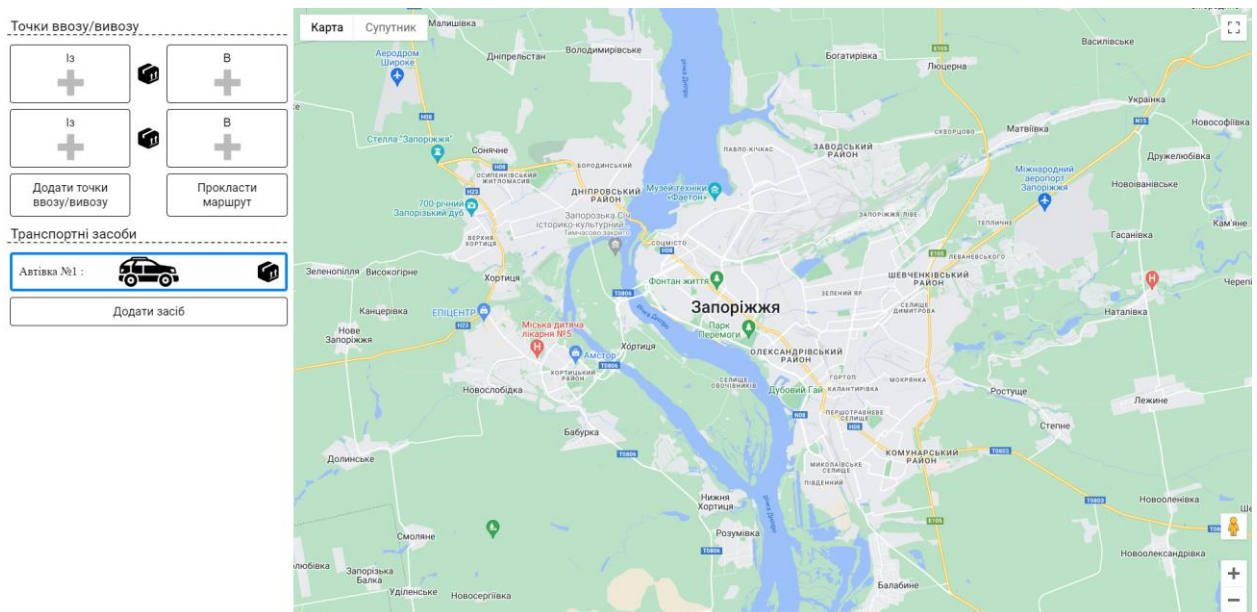


Рисунок 5.1 – Головне вікно

Для додавання товарів необхідно натиснути «Додати точку ввозу/вивозу», після чого у формах кнопок «Із» та «В» ввести відповідні адреси (рис. 5.2). Опісля необхідно натиснути параметри вантажу (рис. 5.3).

Точки ввозу/вивозу

Запорозький Національний Університет Політехніки

В

Із

В

Додати точки ввозу/вивозу

Прокласти маршрут

Транспортні засоби

Автівка №1

Додати засіб

Почніть вводити назву точки, після під полем з'явиться випадаючий список, із якого ви зможете обрати потрібну адресу

ЗНТУ

Запорозький Національний Університет Політехніки вулиця...

ЗНТУ Кафедра Автомобілів вулиця Гоголя, Запоріжжя, Запорізька о...

Ідальня ЗНТУ вулиця Дніпровська, Запоріжжя, Запорізька область, Ук...

Навчальний корпус ЗНТУ №2 вулиця Гоголя, Запоріжжя, Запорізьк...

Бібліотека ЗНТУ вулиця Жуковського, Запоріжжя, Запорізька область,...

powered by Google

Приховати

Рисунок 5.2 – Введення адреси

У налаштуванні транспортних засобів можна створити нові транспортні засоби та задати їх характеристики через спеціальну форму.

Введіть габарити грузу, який потрібно перевезти

Назва грузу

Груз 1

Довжина грузу(м)

1

Ширина грузу(м)

1

Висота грузу(м)

1

Вага грузу(кг)

1

Рисунок 5.3 – Форма введення даних про вантаж

Для додавання ТЗ необхідно натиснути «Додати засіб» і над кнопкою з’явиться нова панель з автомобілем. Для змін параметрів треба натиснути на панель відповідного автомобілю, що покаже форму вводу ТЗ (рис. 5.4).

Введіть габарити грузу, що може перевезти Автівка №2



Максимальна довжина(м)

2

Максимальна ширина(м)

2

Максимальна висота(м)

2

Максимальна вага(кг)

500

Витрата палива на 100км(л)

9

Вартість літру палива(грн/л)

47

Рисунок 5.4 – Форма введення даних про ТЗ

Після введення даних, треба просто натиснути «Розрахувати маршрут» і програма виведе форму з даними (рис. 5.5).

Точки ввозу/вывозу

проспект Соборний, 222, Запоріжжя, Запорізька область, Україна, 69000

проспект Соборний, 111, Запоріжжя, Запорізька область, Україна, 69000

вулиця Ігоря Сікорського, 444, Запоріжжя, Запорізька область, Україна, 69000

вулиця Тимирязєва, 333, Запоріжжя, Запорізька область, Україна, 69000

Додати точки ввозу/вывозу

Прокласти маршрут

Транспортні засоби

Автівка №1 : 

Автівка №2 : 

Додати засіб

Автівка №1 :

А. проспект Соборний, 222, Запоріжжя, Запорізька область, Україна, 69000

В. проспект Соборний, 111, Запоріжжя, Запорізька область, Україна, 69000

Груз: _____

Загруженість авто(вага): _____

589.590кг

Підсумки шляху

Загальний профільний шлях(км): 17.564

Витрати палива(л): 1.229

Вартість палива(грн): 57.79

Автівка №2 :

А. вулиця Ігоря Сікорського, 444, Запоріжжя, Запорізька область, Україна, 69000

В. вулиця Тимирязєва, 333, Запоріжжя, Запорізька область, Україна, 69000

Груз: _____

Загруженість авто(вага): _____

499.500кг

Підсумки шляху

Загальний профільний шлях(км): 2.613

Витрати палива(л): 0.235

Вартість палива(грн): 11.05

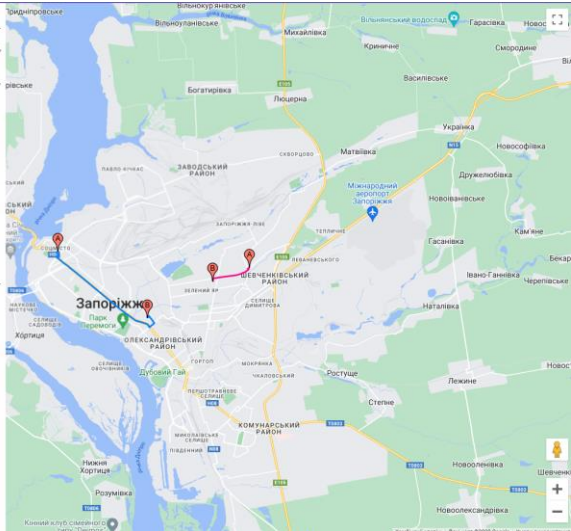


Рисунок 5.5 – Результат роботи програми

5.4 Повідомлення оператору

Під час роботи розробленого ПП генеруються повідомлення, які призначені для інформування користувача про помилку, початок роботи розрахунків, завершення виконуваної дії, і т. п. випадків, які не потребують від користувача будь-якого вибору. Перелік можливих повідомлень, опис їхнього змісту та відповідні дії зазначений в розділі «Керівництво програміста» в таблиці 4.1.

5.5 Висновок

В даному розділі було описано призначення та функції розробленого ПП та мінімально необхідні апаратні умови для його встановлення та використання. Була наведена детальна інструкція по роботі оператора з ПП, а саме приклад введення інформації про товари та транспорт і розрахування ефективних маршрутів їх доставки та одночасного закріплення за ними транспортних засобів. Було приведено перелік повідомлень, що генеруються під час роботи ПП, а також представлено приклад вхідних та вихідних даних. Інтерфейс користувача розробленого ПП є простим та інтуїтивно зрозумілим і не потребує додаткового навчання оператора.

6 ТЕСТУВАННЯ І РЕЗУЛЬТАТИ

6.1 Опис методики проведення тестування

Зазвичай, тестування алгоритмів вирішення задач маршрутизації проводиться із використанням прорахованих умов із Соломонових задач, розроблених ще у 1987 році. Цей метод тестування дозволяє порівнювати результати з вже відомими кращими розв'язками, оцінювати швидкість методу та рівень відхилення від оптимального розв'язку. Приклад вхідних даних для входу, можна побачити на рис. 6.1.

VEHICLE NUMBER 25	CAPACITY 200					
CUSTOMER CUST NO.	XCOORD.	YCOORD.	DEMAND	READY TIME	DUE DATE	SERVICE
0	40	50	0	0	1236	0
1	45	68	10	912	967	90
2	45	70	30	825	870	90
3	42	66	10	65	146	90
4	42	68	10	727	782	90
5	42	65	10	15	67	90
6	40	69	20	621	702	90
7	40	66	20	170	225	90
8	38	68	20	255	324	90
9	38	70	10	534	605	90
10	35	66	10	357	410	90
11	35	69	10	448	505	90
12	25	85	20	652	721	90
13	22	75	30	30	92	90

Рисунок 6.1 – Приклад вхідних даних задачі 102 з бенчмарку Соломона

Проте, як можна побачити, ці дані для розрахунку задачі маршрутизації із часовими вікнами у той час, як в нас задача з підбором та обмеженим об'ємом вантажу, тож тестування буде проходити через автоматично згенеровані дані вірного формату, приклад яких зображено на рис. 6.2.

VEHICLE						
NUMBER	MAX WEIGHT	LENGTH	HEIGHT	WIDTH	FUEL	FUEL_COST
0	243	1	3	7	12	12
1	221	5	5	3	12	18
2	21	8	7	4	13	17
3	663	8	2	6	11	13
4	837	2	4	6	12	12
5	973	1	8	1	14	16
6	113	7	2	5	14	17
7	40	3	1	7	13	15
8	406	2	8	2	16	18
9	363	8	9	8	15	12
10	105	8	2	3	15	12
11	214	2	3	4	12	15

CUSTOMER							
CUST NO.	XCOORD.	YCOORD.	WEIGHT	LENGTH	HEIGHT	WIDTH	WHERE_TO
0	38	39	2	4	7	5	1
1	37	68	0	0	0	0	-1
2	35	70	7	5	4	3	3
3	34	66	0	0	0	0	-1
4	43	68	5	6	8	1	5
5	42	46	0	0	0	0	-1
6	41	64	5	4	3	2	7
7	36	25	0	0	0	0	-1
8	37	73	1	5	7	8	9
9	35	26	0	0	0	0	-1
10	38	16	8	6	9	3	11

Рисунок 6.2 – Приклад вхідних даних для тестування

За умовами задачі, є точки із яких потрібно забрати вантаж і точки, куди його потрібно відвезти. Для других, параметри висоти/ширини/довжини будуть дорівнювати 0, а точка куди везти буде мати значення «-1».

6.2 Результати використання розробленого програмного забезпечення

Для тестування, було розроблено 3 метода розв'язку задачі: модифікований генетичний метод, метод табу пошуку та жадібний. В залежності від результатів, можна додати до програми вивід результатів другого методу, або змінити основний, якщо це буде мати сенс.

Почнемо з тестування швидкості роботи. Усі методи працюють з одними й тими ж самими даними, розмірністю 25, 50 та 100 точок. Кількість автівок не змінюється – 10.

Як бачимо на рис. 6.3, найшвидше відпрацював жадібний метод, не сильно від нього відстав генетичний, а метод табу пошуку сильно відстає від

обох. Так жадібний метод має стати основним через швидкість? Залежить від умов використання програми, проте істотним мінусом жадібного методу є його точність, результати перевірки якої зображено на табл. 6.1.

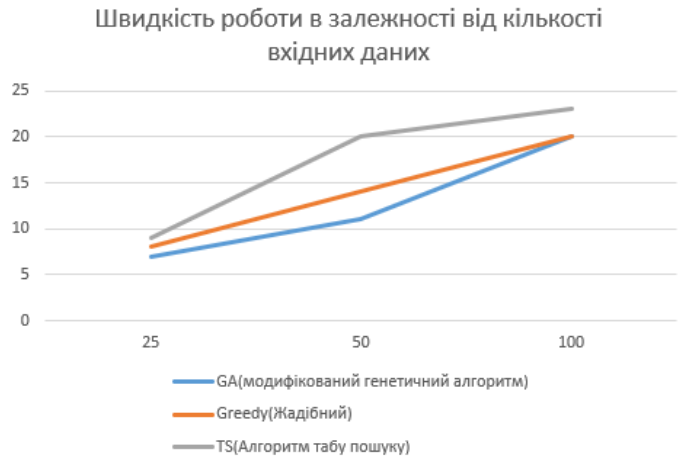


Рисунок 6.3 – Порівняння швидкості роботи методів

Таблиця 6.1 – Порівняння точності методів

Метод	GA(генетичний)		Greedy(жадібний)		TS(табу пошук)	
Кількість точок	Загальна довжина маршруту	Кількість заданих ТЗ	Загальна довжина маршруту	Кількість заданих ТЗ	Загальна довжина маршруту	Кількість заданих ТЗ
25	617,1	7	682	7	617,1	8
50	1231	9	1375	10	1232	10
100	1655	10	1827	10	1680	10

Загальна довжина маршруту жадібного методу значно вища, ніж в останніх двох, а ось різниця між табу пошуком і генетичним методом невелика. Підсумовуючи результати досліджень, модифікований генетичний метод показав найкращі результати і стає основним методом програми.

ВИСНОВКИ

У ході виконання проєкту, проведено детальне ознайомлення із предметною областю – вирішенням задач маршрутизації, ознайомлення із найбільш важливими питаннями цієї проблемної області та найвагомішими рішеннями: методами табу пошуку, генетичними, жадібними, локального пошуку та іншими.

Під час роботи із першим розділом було проаналізовано предметну область, її історію та вже існуючі аналоги проєкту.

Також у першому розділі було доведено актуальність проєкту.

У другому розділі розглянуто мови програмування - Javascript, Python та засоби створення інтерфейсу: HTML, CSS, а також спроектовано структуру додатку і розглянуто, необхідні для її відтворення, модулі.

У третьому розділі розглянуто методи, застосовані у проєкті і алгоритм функціонування програми в цілому.

В решті-решт, було розроблено ПП, який реалізовував усі поставлені задачі та список функціональних вимог:

- введення початкових даних користувача;
- розрахунок оптимальних маршрутів та розвантажень для введених даних;
- виведення розрахованих маршрутів на мапі та списку результатів.

Тестування ПП довело, що обраний метод є найефективнішим з реалізованих і програма виводить очікувані результати при тестових даних.

В подальшому можливо модернізувати ПП в напрямку автоматизації, наприклад, шляхом додавання можливості вводу даних у вигляді завантаження файлів різних типів, що скоротить час вводу даних для розрахунків.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАНЬ

1. Skjoett-Larsen, T. (2000), "European logistics beyond 2000", International Journal of Physical Distribution & Logistics Management, Vol. 30 No. 5, pp. 377-387. <https://doi.org/10.1108/09600030010336144>.
2. А. Д. Баранова, Е. Г. Жданова. Задача маршрутизації транспортних засобів з часовими вікнами // VI Всеукраїнська науково-практична конференція молодих вчених та студентів «Інформаційні системи та технології управління»(ІСТУ-2021). – 2021. – Kyiv, Ukraine.
3. CSCMP's 30 th Annual State of Logistics Report and Presentation. // CSCMP's Annual State of Logistics Report ® and Presentation. – 2019. – №30.
4. Braekers, K., Ramaekers, K., & Nieuwenh, I. V. (2016). The Vehicle Routing Problem: State of the Art Classification and Review. Computers & Industrial Engineering. ^[1]_{SEP}.
5. G. Clarke, J. W. Wright. Scheduling of vehicles from a central depot to a number of delivery points // Operations Research. –1962. – pp. 568–581.
6. Paul A.,Kumar R.S., Rout C. et al. Designing a multi-depot multi-period vehicle routing problem with time window: hybridization of tabu search and variable neighbourhood search algorithm. Sādhanā 46, 183 (2021). <https://doi.org/10.1007/s12046-021-01693-2>.
7. M. M. Solomon. Algorithms for the VR and SP rith TM Constraints // vol. 35 .–1987. – no. 2. –pp. 254–265.
8. M. M. Solomon, J. Desrosiers. Time Window Constrained Routing and Scheduling Problems. –1988. – pp. 1–13.
9. Cantu-Paz, E., 2016. Solving Travelling Salesman Problem with an Improved Hybrid Genetic Algorithm. Journal of Computer and Communications, 4, ISSN Online: 2327-5227 ISSN Print: 2327-5219.
10. Feiyue Li, Bruce Golden, and Edward Wasil. Very large-scale vehicle routing: new test problems, algorithms, and results. Computers & Operations Research, 32(5):1165–1179, 2005.

11. Bruck B P and Iori M 2017 Non-elementary formulations for single vehicle routing problems with pickups and deliveries *Oper. Res.* 65 1597-1614.
12. Tânia Rodrigues Pereira Ramos, Maria Isabel Gomes & Ana Paula Barbosa Póvoa (2020) Multi-depot vehicle routing problem: a comparative study of alternative formulations, *International Journal of Logistics Research and Applications*, 23:2, 103-120, DOI: 10.1080/13675567.2019.1630374 Volume 1, Pages 20-21.
13. LocalSolver [Electronic resource]. – Access mode: <https://www.localsolver.com/benchmark/huge-instances-capacitated-vehicle-routing-problem-cvrp>.
14. Arcgis-Desktop [Electronic resource]. – Access mode: <https://www.esri.com/en-us/arcgis/products/arcgis-desktop/resources>.
15. OptaPlanner [Electronic resource]. – Access mode: <https://www.optaplanner.org/learn/useCases/vehicleRoutingProblem.html>.
16. Soebiyanto R. P., Kiang R. K., 2000. Modeling Influenza Transmission Using Environmental Parameters. *International Archives of the Photogrammetry, Remote Sensing and Spatial Information Science*, XXXVIII, 8, pp. 330–334.
17. Eduardo, U., Diego, P. (2017). New benchmark instances for the Capacitated Vehicle Routing Problem. *European Journal of Operational Research*.
18. Ehsan, T., & Vahid, K. (2016). Enhanced intelligent water drops and cuckoo search algorithms for solving the capacitated vehicle routing problem. *Information Sciences*.
19. Window by Considering the Flexible End Depot in Each Route. *International Journal of Supply and Operations Management*. November 2016, Volume 3, Issue 3, pp. 1373-1390. ISSN-Print: 2383-1359. ISSN-Online: 2383-2525
20. Inman, James (1835). *Navigation and Nautical Astronomy: For the Use of British Seamen* (3 ed.). London, UK: W. Woodward, C. & J. Rivington. Retrieved 2015-11-09 pp. 1-9.

21. Haupt, R., Haupt I., 2004. Practical genetic algorithms. John Wiley & Sons, 2, pp. 261.
22. Oliinyk A. A Evolutionary method for solving the traveling salesman problem / A. Oliinyk, I. Fedorchenko, A. Stepanenko, M. Rud, D. Goncharenko // Problems of Infocommunications. Science and Technology : 5th International Scientific-Practical Conference PICST2018, Kharkiv, 9–12 October 2018 : proceedings of the conference. – Kharkiv: Kharkiv National University of Radioelectronics, 2018 — P. 331 – 339. DOI: 10.1109/INFOCOMMST.2018.8632033.
23. Fedorchenko, I., Oliinyk, A., Stepanenko, A., Zaiko, T., Korniienko S., Burtsev N.: Development of a genetic algorithm for placing power supply sources in a distributed electric network. Eastern European Journal of Enterprise Technologies, issue 5/3 (101), pp. 6–16 (2019). DOI: 10.15587/1729-4061.2019.180897.
24. Oliinyk, A., Fedorchenko, I., Stepanenko, A., Rud, M., Goncharenko, D.: Combinatorial optimization problems solving based on evolutionary approach. In: 2019 15th International Conference on the Experience of Designing and Application of CAD Systems (CADSM), pp. 41-45 (2019). DOI: 10.1109/CADSM.2019.8779290.

ДОДАТОК А
Технічне завдання

Вступ

ПП призначено для автоматизації розрахунку оптимальних маршрутів та розкладу вантажів для задач маршрутизації.

В роботі наведено аналіз та дослідження методів розв'язання задач маршрутизації. Розроблено нові методи розрахунку маршрутів. Розроблені методи також вирішують питання комбінаторної оптимізації у відношенні вибору розміщення товарів по ТЗ. Їх принципова відмінність від існуючих полягає у тому, що запропоновані методи враховують весь матеріал завдання та мають спеціалізовані способи кодування можливих рішень, що забезпечують максимальну адаптацію до специфіки розв'язуваної задачі.

A.1 Підстава для розробки

Підставою для розробки є завдання на дипломну кваліфікаційну роботу на тему «Програмна реалізація вирішення задач маршрутизації з підбором та ваговими обмеженнями», затверджене наказом Національного університету «Запорізька політехніка» № 168 від 24 квітня 2024 р.

A.2 Призначення розробки

Додаток призначено для автоматизації розрахунку оптимальних маршрутів для задач маршрутизації, що може стати в нагоді при будь-якому логістичному завданні, що має схожі параметри.

A.3 Основні вимоги до програми, що розробляється

A.3.1 Вимоги до функціональних характеристик

ПП забезпечує виконання таких функцій:

- введення даних для задачі маршрутизації;

- обробка та валідація введених даних;
- вирішення задачі маршрутизації;
- виведення результатів у текстовому форматі;
- виведення результатів на мапу;
- виведення попереджень при помилках.

A.3.2 Вимоги до інтерфейсу програми

Інтерфейс програми повинен забезпечувати передачу даних між користувачем і алгоритмом програми, бути зрозумілим і інтуїтивним, тож можна навести такі вимоги до нього:

- швидке оновлення інформації на екрані;
- інтуїтивність формату введення даних та їх валідація;
- виведення попереджень при помилках;
- виведення результатів розрахунків у графічному вигляді;
- виведення результатів у текстовому вигляді.

A.3.3 Вимоги до надійності

До основних вимог до надійності програми відносяться наступні:

- перевірка даних, введених у комірки;
- цілодобова робота сайту;
- захист від ін'єкцій;
- доступ до АПІ.

A.3.4 Умови експлуатації

Для експлуатації програмного забезпечення, користувачу потрібен доступ до інтернету і браузер.

A.3.5 Вимоги до складу та параметрів технічний засобів

Для експлуатації програмного забезпечення для комунікації між користувачами необхідно таке апаратне забезпечення:

- процесор Intel I3/ amd ryzen 3 чи новіше;
- оперативна пам'ять – 2 Гб, або більше;
- встановлене на сервері програмне забезпечення: python із усіма необхідними бібліотеками;
- вільне місце на диску – 1 Гб.

A.3.6 Вимоги до маркування і пакування

Програма може бути записана на будь-якому носії інформації.

A.4 Вхідні дані до роботи

Вхідними даними до програмного забезпечення, є адреси точок ввозу і вивозу, кількість ТЗ в автопарку, габарити допустимого у ТЗ вантажу, витрати палива для кожного ТЗ і габарити вантажу для перевезення.

Вихідними даними, є текстове та графічне відображення на мапі шляхів для кожного ТЗ. Також, у текстовому вигляді виводиться кошторис перевезення для кожного ТЗ.

ДОДАТОК Б
Фрагмент тексту програми

```

tournament_size = 3
num_parents = 0
crossover_probability = 0.9
mutation_probability = 0.5

def create_initial_pop(request, pop_size=400):
    global num_parents
    num_parents = (int(pop_size*0.30))
    request_list = []
    for i in (request.keys()):
        request_list.append(i)
    initial_pop = []
    for i in range(pop_size):
        temp = request_list.copy()
        random.shuffle(temp)
        initial_pop.append(temp)
    return initial_pop

def run_num_generations(pop_list, request, num_generations):
    x = datetime.datetime.now()
    a = x.strftime("%X")
    b = a.replace(":", "")
    for i in range(num_generations):
        fitness = calc_fitness(pop_list, request)
        best_ind = calc_best_chromosome(pop_list, fitness)
        temp_pop_list = pop_list.copy()
        temp_fitness = fitness.copy()
        parent = []
        pt = []
        for j in range(num_parents):
            parent.append(parent_selection(temp_pop_list, request, temp_fitness, tournament_size))
            if parent[j] in pop_list:
                index = temp_pop_list.index(parent[j])
                del temp_pop_list[index]
                del temp_fitness[index]

        rand_crossover = random.uniform(0,1)
        if (rand_crossover <= crossover_probability):
            for i in parent:
                temp = i.copy()
                pt.append(temp)
            offsprings = crossover(pt, request)
            rand_mutation = random.uniform(0,1)
            if (rand_mutation <= mutation_probability):
                mutated_offsprings = mutation(offsprings)
            else:
                mutated_offsprings = []
        else:
            offsprings = []
            mutated_offsprings = []
        pop_list = new_population_selection(pop_list, parent, parent, offsprings, mutated_offsprings)
    return best_ind

def calc_fitness(pop_list, request):
    total_distance_individual = []
    for i in pop_list:
        sum_route = euclidean_distance(i, request)
        total_distance_individual.append(sum_route)

    return total_distance_individual

```

```

def euclidean_distance(individual,request):
    sum_dist = 0
    for j in range(len(individual)-1):
        x1 = request[individual[j]]['pickup'][1]
        y1 = request[individual[j]]['pickup'][0]
        x2 = request[individual[j+1]]['pickup'][1]
        y2 = request[individual[j+1]]['pickup'][0]

        sum_dist += math.sqrt(((x2-x1)**2+((y2-y1)**2)

    return sum_dist

def parent_selection(pop_list,request,fitness,tournament_size):
    parent = []
    if(len(pop_list) <= tournament_size):
        return (pop_list[0])
    else:
        select_tour = random.sample(range(len(pop_list)), tournament_size)

        selected_chromosomes = []
        fitness_selected_chromosome = []
        for i in select_tour:
            selected_chromosomes.append(pop_list[i])
            fitness_selected_chromosome.append(fitness[i])
        ranked_selected_chromosomes = [x for _,x in
sorted(zip(fitness_selected_chromosome,selected_chromosomes))]
        ranked_fitness_selected_chromosome = [y for y,_ in
sorted(zip(fitness_selected_chromosome,selected_chromosomes))]
        return ranked_selected_chromosomes[0]

def fitness():
    return fitness

def crossover(par,request):
    parents = []
    offsprings = []
    for r in range(len(par)-1):
        a = par[r].copy()
        b = par[r+1].copy()
        parents.append(a)
        parents.append(b)

    length = len(parents)
    if (length%2 == 1):
        length -=1
    for i in range(0,(length-1),2):

        fitness_append_parent1_ele1 = []
        fitness_append_parent2_ele1 = []
        fitness_append_parent1_ele2 = []
        fitness_append_parent2_ele2 = []

        length_parent = len(parents[i])

        rand_1 = random.randint(0,length_parent-2)
        rand_2 = random.randint(0,length_parent-2)

        del_parent1_1 = parents[i+1][rand_2]
        del_parent1_2 = parents[i+1][rand_2+1]
        del_parent2_1 = parents[i][rand_1]
        del_parent2_2 = parents[i][rand_1+1]

```

```

parents[i].remove(del_parent1_1)
parents[i].remove(del_parent1_2)
parents[i+1].remove(del_parent2_1)
parents[i+1].remove(del_parent2_2)

for j in range(len(parents[i])+1):
    parents[i].insert(j,del_parent1_1)
    parents[i+1].insert(j,del_parent2_1)

    fitp1_1 = euclidean_distance(parents[i],request)

    fitness_append_parent1_ele1.append(fitp1_1)

    fitp2_1 = euclidean_distance(parents[i+1],request)

    fitness_append_parent2_ele1.append(fitp2_1)

    parents[i].remove(del_parent1_1)
    parents[i+1].remove(del_parent2_1)

indp1_1 = np.argmin(fitness_append_parent1_ele1)

indp2_1 = np.argmin(fitness_append_parent2_ele1)

parents[i].insert(indp1_1,del_parent1_1)
parents[i+1].insert(indp2_1,del_parent2_1)

for j in range(len(parents[i])+1):
    parents[i].insert(j,del_parent1_2)
    parents[i+1].insert(j,del_parent2_2)

    fitp1_2 = euclidean_distance(parents[i],request)

    fitness_append_parent1_ele2.append(fitp1_2)

    fitp2_2 = euclidean_distance(parents[i+1],request)

    fitness_append_parent2_ele2.append(fitp2_2)

    parents[i].remove(del_parent1_2)
    parents[i+1].remove(del_parent2_2)

indp1_2 = np.argmin(fitness_append_parent1_ele2)

indp2_2 = np.argmin(fitness_append_parent2_ele2)

parents[i].insert(indp1_2,del_parent1_2)
parents[i+1].insert(indp2_2,del_parent2_2)

offsprings = parents

return offsprings

def mutation(offsprings):
    for i in range(len(offsprings)):

        rand_1 = random.randint(0,len(offsprings[i]))

```

```

    rand_2 = random.randint(0,len(offsprings[i]))
    while(rand_2 == rand_1):
        rand_2 = random.randint(0,len(offsprings))
    if(rand_1>rand_2):
        a = rand_2
        b = rand_1
    else:
        a = rand_1
        b = rand_2
    b+=1
    offsprings[i][a:b] = offsprings[i][a:b][::-1]
return offsprings

def new_population_selection(pop_list,parent,parents,offsprings,mutated_offsprings):
    temp_pop_list = []
    for k in pop_list:
        if(k not in parents):
            temp_pop_list.append(k)
    new_pop = []
    for p1 in parents:
        new_pop.append(p1)
    for b1 in offsprings:
        new_pop.append(b1)
    for c1 in mutated_offsprings:
        new_pop.append(c1)
    for i in range(int(len(temp_pop_list)*0.6)):
        new_pop.append(temp_pop_list[i])
    return new_pop

def calc_best_chromosome(pop_list,fitness):
    best_fitness_index = np.argmin(fitness)
    best_chromosome = {"best":{"tour":pop_list[best_fitness_index],"fitness":fitness[best_fitness_index]}}
    return best_chromosome

```

ДОДАТОК В
Слайди презентації

ПРОГРАМНА РЕАЛІЗАЦІЯ ВИРІШЕННЯ ЗАДАЧ МАРШРУТИЗАЦІЇ З ПІДБОРОМ ТА ВАГОВИМИ ОБМЕЖЕННЯМИ

Виконав:
ст. групи КНТ 111-сп

Геннадій НЕСТЕРОВ

Керівник:
д.т.н., професор

Андрій ОЛІЙНИК

Рисунок В.1 – Слайд 1

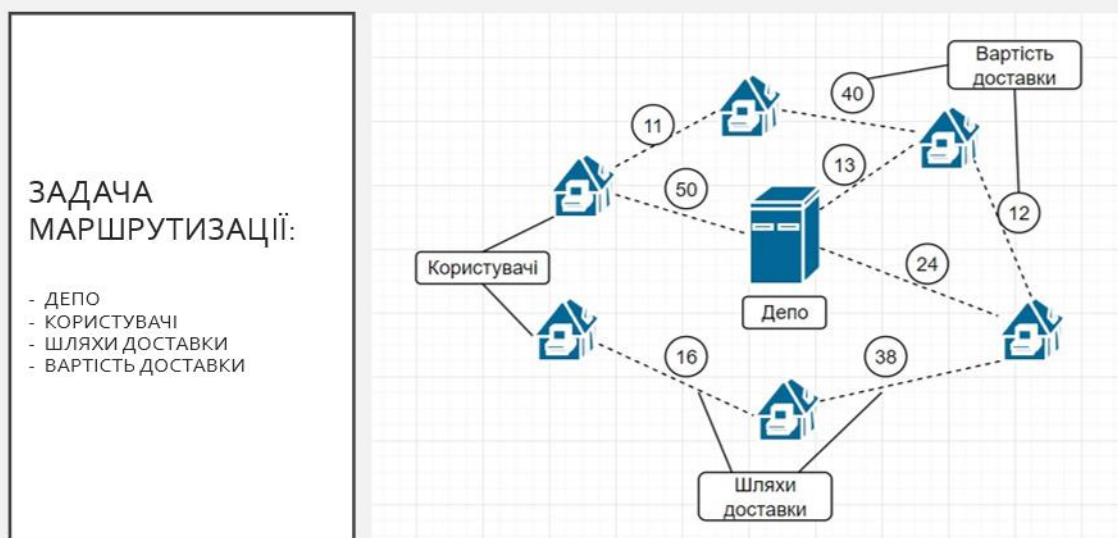
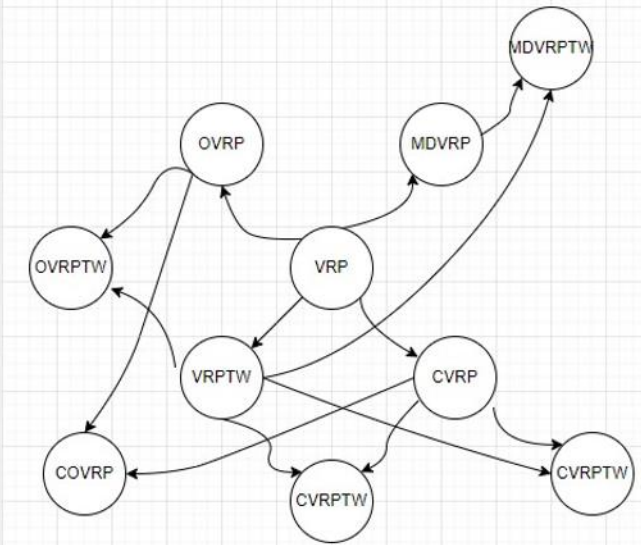


Рисунок В.2 – Слайд 2

ВИДИ ЗАДАЧ МАРШРУТИЗАЦІЇ ЗА ОБМЕЖЕННЯМИ :

- В ЧАСІ
- В ОБ'ЄМІ ВАНТАЖУ
- В КІЛЬКОСТІ ДЕПО
- В ПОРЯДКУ ПРОХОДЖЕННЯ ТОЧОК



3

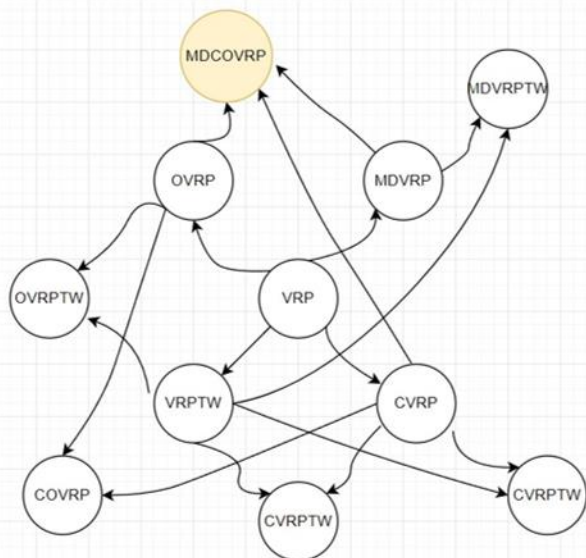
Рисунок В.3 – Слайд 3

ОБМЕЖЕННЯ В ОБРАННІЙ ЗАДАЧІ:

- В ОБ'ЄМІ ВАНТАЖУ
- В КІЛЬКОСТІ ДЕПО
- В ПОРЯДКУ ПРОХОДЖЕННЯ ТОЧОК

=

VRPPD (VEHICLE ROUTING
PROBLEM WITH PICKUP AND
DELIVERY)



4

Рисунок В.4 – Слайд 4

МЕТА РОБОТИ

- Розробити вебзастосунок для вирішення транспортної задачі з підбором (VRPPD), обмеженнями ваги та габаритів вантажу із використанням прикладних даних

ЗАДАЧІ ПРОЕКТУ

- Забезпечити можливість ведення даних для задачі маршрутизації із відображенням на мапі;
- Розробити алгоритми вирішення задачі маршрутизації із введеними даними;
- Вивести результати алгоритмів у текстовому форматі (вартість подорожі, текстовий опис шляху, опис вантажу, завантаженість авто);
- Вивести оптимальний шлях на мапу;

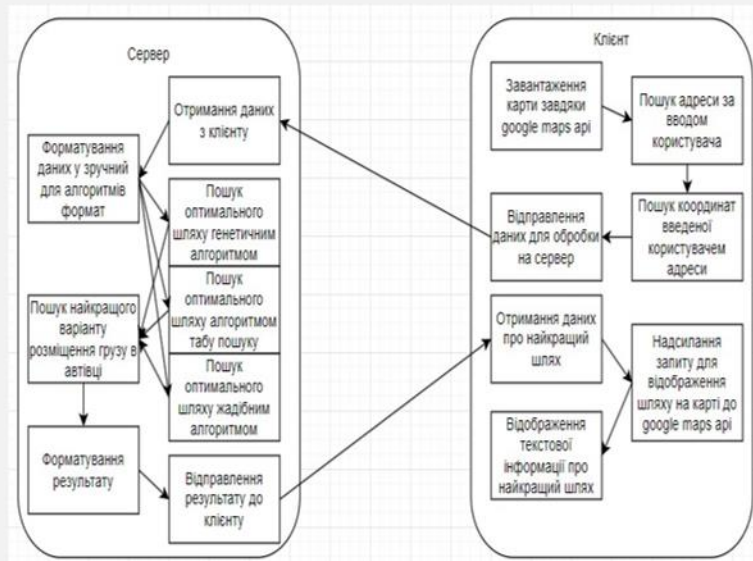
5

Рисунок В.5 – Слайд 5

ФУНКЦІОНАЛЬНА СХЕМА ПРОГРАМИ

Клієнтом виступає браузер і тому всередині записане усе, що виконується завдяки js(+html+css)

Замість серверу, емулявана бібліотекою eel, поведінка скриптів на мові python



6

Рисунок В.6 – Слайд 6

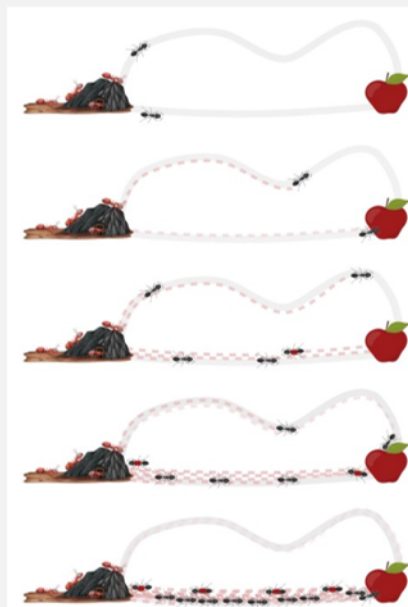
ГЕНЕТИЧНИЙ МЕТОД

Спочатку два різні шляхи мають однаковий пріоритет

Але в процесі проходження шляхів, мурахи приносять інформацію про свої досягнення (довжину пройденого шляху, витрачений час і т.

З кожним поколінням все більше і більше мурах йдуть найкоротшим шляхом

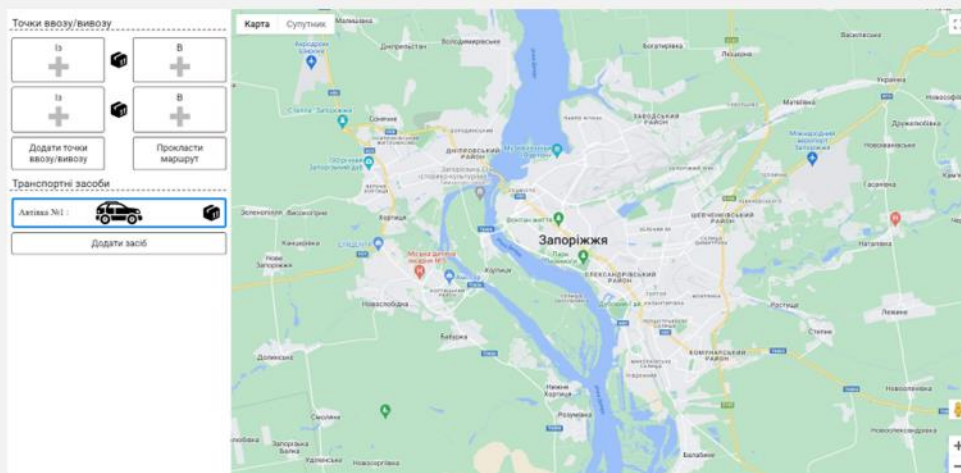
Поки найкращий шлях не стане очевидним



7

Рисунок В.7 – Слайд 7

ГОЛОВНА СТОРІНКА



8

Рисунок В.8 – Слайд 8

ВВЕДЕННЯ ДАНИХ ПРО ТОЧКУ ДОСТАВКИ

Точки зліва являють собою депо, вантаж із якого потрібно відвезти у точку праворуч. Таким чином, маршрути мають враховувати, що праву точку нема сенсу відвідувати до того, як буде відвідана точка злева

Точки ввозу/вывозу

Зaporozhskий Національний Університет Політехніка

із

Додати точки ввозу/вывозу

В

В

Прокласти маршрут

Транспортні засоби

Автомобіль №1 :

Додати засіб

Почніть вводити назву точки, після чого з'явиться випадаючий список, із якого ви зможете обрати потрібну адресу

ЗНТУ

Зaporozhskий Національний Університет Політехніка курс...

ЗНТУ Кафедра Автомобільної техніки Голос, Запоріжжя, Запоріжжя в...

Ідальний ЗНТУ курсу Дипломанти, Запоріжжя, Запоріжжя область, Ук...

Навчальний корпус ЗНТУ №2 курсу Голос, Запоріжжя, Запоріжжя...

Бібліотека ЗНТУ курсу Мультиязык, Запоріжжя, Запоріжжя область, powered by Google

Принтовати

9

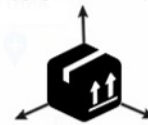
Рисунок В.9 – Слайд 9

ВВЕДЕННЯ ДАНИХ ПРО ВАНТАЖ

В кожному транспортному засобі є такі ж поля введення для максимально допустимих параметрів вантажу.

При натисканні на поле введення із довжиною, шириною чи висотою на малюнку згори, стрілочка відповідного параметру змінить колір на червоний, що полегшує розуміння інтерфейсу користувачем.

Введіть габарити вантажу, який потрібно перевезти



Назва вантажу

Груз 1

Довжина вантажу(м)

1

Ширина вантажу(м)

1

Висота вантажу(м)

1

Вага вантажу(кг)

1

10

Рисунок В.10 – Слайд 10

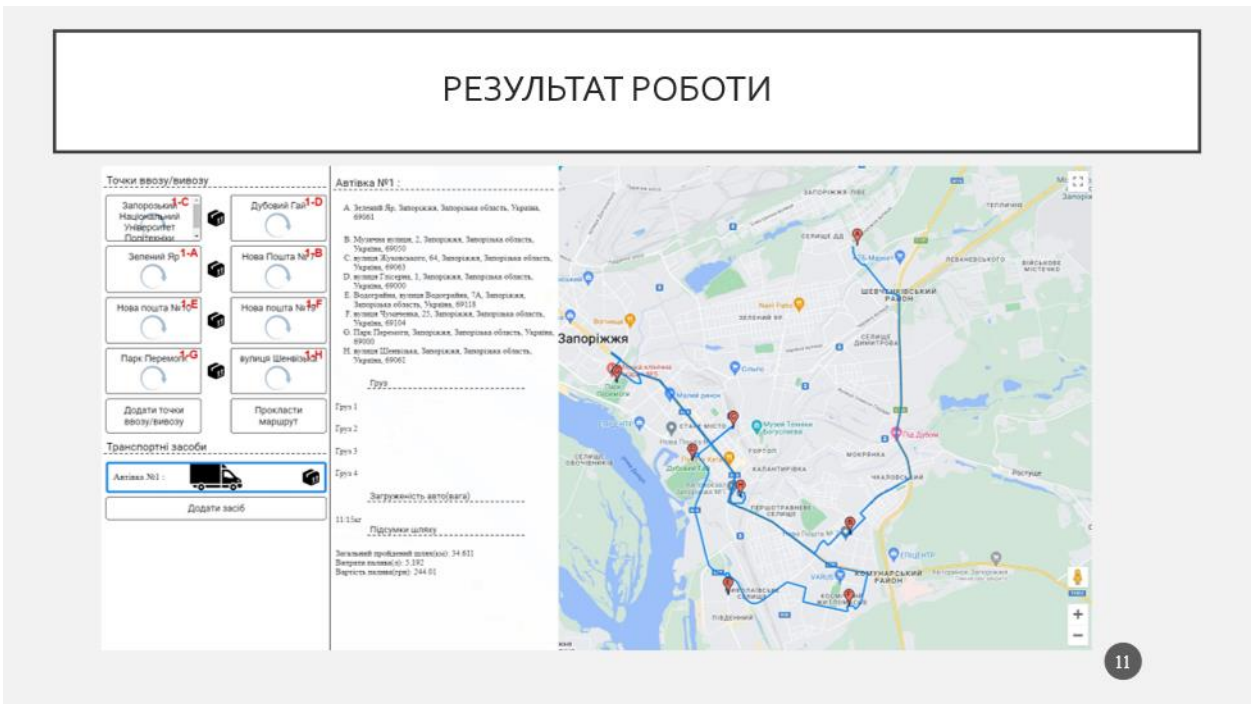


Рисунок В.11 – Слайд 11



Рисунок В.12 – Слайд 12



ДЯКУЮ ЗА УВАГУ

Рисунок В.13 – Слайд 13