ДВНЗ «Донецький національний технічний університет»
Факультет комп'ютерно-інформаційних технологій та автоматизації
(повна назва факультету)

Кафедра електронної техніки
(повна назва кафедри)

«До захисту допущено»
В.о. зав. кафедри електронної техніки
_____ Сергій КОВАЛЬОВ
(підпис) (Ім'я та ПРІЗВИЩЕ)
«___» _____ 2023 р.

Випускна кваліфікаційна робота

магістра
(освітній ступінь)

на тему «Система підтримки прийняття рішення логістичних задач»

Виконав студент 2 курсу, групи КІм–22
(шифр групи)

спеціальності 123 Комп'ютерна інженерія
(код і назва спеціальності)

освітньої програми Комп'ютерна інженерія
(назва освітньої програми)

Квілінський Олексій Станіславович
(Прізвище, Ім'я, По-батькові)

(підпис)

Керівник в.о. каф.ЕТ, к.т.н., доц. _____ Сергій КОВАЛЬОВ
(посада, науковий ступінь, вчене звання,) (підпис) (Ім'я та ПРІЗВИЩЕ)

Рецензент зав. каф. ПМІ, к.т.н., доц. _____ Наталія МАСЛОВА
(посада, науковий ступінь, вчене звання,) (підпис) (Ім'я та ПРІЗВИЩЕ)

Рецензент доц. каф. АТ, к.т.н., доц. _____ Анна ВОРОПАЄВА
(посада, науковий ступінь, вчене звання,) (підпис) (Ім'я та ПРІЗВИЩЕ)

Нормоконтроль
асистент кафедри ЕТ

Засвідчую, що в цій випускній кваліфікаційній роботі немає запозичень з праць інших авторів без відповідних посилань.

_____ Артем ЛЮБИМОВ
(підпис)

Студент _____
(підпис)

Луцьк – 2023 р.

ДВНЗ «Донецький національний технічний університет»
Факультет комп'ютерно-інформаційних технологій та автоматизації
(повна назва факультету)

Кафедра електронної техніки
(повна назва кафедри)

Захист відбувся _____
(дата)

з оцінкою_____
Секретар ЕК _____
(підпис)

Випускна кваліфікаційна робота

магістра
(освітній ступінь)

Тема: «Система підтримки прийняття рішення логістичних завдань»

Виконавець, студент
гр. КІм – 22 _____ Олексій КВІЛІНСЬКИЙ
(підпис, дата, Ім'я та ПРІЗВИЩЕ)

Керівник _____ Сергій КОВАЛЬОВ
(підпис, дата, Ім'я та ПРІЗВИЩЕ)

Консультанти: _____
(підпис, дата, Ім'я та ПРІЗВИЩЕ)

_____
(підпис, дата, Ім'я та ПРІЗВИЩЕ)

_____
(підпис, дата, Ім'я та ПРІЗВИЩЕ)

Луцьк – 2023

ДВНЗ «Донецький національний технічний університет»

Факультет комп'ютерно-інформаційних технологій і автоматизації

Кафедра Електронної техніки

Освітній ступінь Магістр

Спеціальність 123 Комп'ютерна інженерія
(шифр і назва)

ЗАТВЕРДЖУЮ:
Завідувач кафедри
_____ Сергій КОВАЛЬОВ
"____" _____2023 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ

Квілінському Олексію Станіславовичу
(прізвище, ім'я, по батькові)

1. Тема роботи «Система підтримки прийняття рішення логістичних завдань»

Керівник роботи Ковальов Сергій Олександрович, к.т.н., доцент
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом від 22.09.2023 р. № 478.

2. Строк подання студентом роботи 10.12.2023 р.

3. Вихідні дані до роботи результати науково-дослідної роботи
результати переддипломної практики

4. **Зміст пояснювальної записки (перелік питань, які потрібно розробити):**

1. Опис об'єкта дослідження та постановка задачі
2. Алгоритм визначення маршрутів перевезення
3. Програмне забезпечення підсистеми визначення маршрутів
4. Моделювання й аналіз експериментальних результатів

Консультанти розділів кваліфікаційної роботи:

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Охорона праці	доц. каф. УГВ, д.т.н., доц. Сергієнко О.І.		

6. Дата видачі завдання 20.09.2023 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	*Назва етапів випускної кваліфікаційної роботи*	*Строк виконання етапів роботи*	*Примітка*
1	Аналіз предметної області	02.09.23 – 24.09.23	
2	Постановка завдання	25.09.23 – 26.09.23	
3	Огляд тематики роботи, актуальність роботи, формулювання мети на задач	27.09.23 – 03.10.23	
4	Опис та обґрунтування основних ідей роботи	04.10.23 – 07.10.23	
5	Розробка алгоритмів	08.10.23 – 14.10.23	
6	Програмна реалізація, аналіз результатів дослідження, формулювання висновків та тенденцій	15.10.23 – 31.10.23	
7	Оформлення пояснювальної записки	01.11.23 – 07.12.23	
8	Нормоконтроль пояснювальної записки та додаткових матеріалів	08.12.23 – 12.12.23	
9	Рецензування роботи	13.12.23 – 15.12.23	
10	Захист роботи	22.12.23	

Студент _____ Квілінський О.С.
(підпис) (прізвище та ініціали)

Керівник роботи _____ Ковальов С.О.
(підпис) (прізвище та ініціали)

ЛИСТ ЗАУВАЖЕНЬ

Посада, Ім’я та ПРІЗВИЩЕ	Суть зауваження, оцінка та підпис

АНОТАЦІЯ

Квілінський, О.С. Система підтримки прийняття рішення логістичних завдань / Випускна кваліфікаційна робота на здобуття освітнього ступеня «магістр» зі спеціальності 123 Комп'ютерна інженерія. – ДВНЗ «ДонНТУ», Луцьк, 2023.

Пояснювальна записка: 105 стор., 24 рис., 14 табл., 22 посилання.

Кваліфікаційна робота присвячена питанню розвитку комп'ютерних систем для логістики, демонструючи, як інноваційні алгоритми та розподілені комп'ютерні системи можуть підвищити ефективність управління ланцюгами постачання. Під час виконання роботи було виконано такі завдання:

- Проведено всебічний аналіз сучасного стану логістичних систем в Україні, включаючи оцінку ефективності існуючих комп'ютерних рішень.

- Досліджено принципи та методики розробки розподілених комп'ютерних систем, визначено ключові критерії для їх ефективного проектування.

- Здійснено аналіз методів кластеризації та алгоритмів побудови оптимальних маршрутів для логістичних цілей, включаючи оцінку впливу вхідних даних на розрахунки.

- Розроблено вдосконалений генетичний алгоритм для побудови ефективних логістичних маршрутів, демонструючи його переваги порівняно з традиційними методами.

- Спроектовано архітектуру розподіленої системи, враховуючи технічні аспекти розгортання, визначено вибір програмного та апаратного забезпечення, а також розроблено інтерфейс користувача.

Ключові слова: кластеризація, маршрут, генетичний алгоритм, розподілена система, розподілена база даних, програмне забезпечення.

ABSTRACT

Kvilinsky, O. Decision Support System for Logistic Tasks / Graduation qualification work for obtaining an educational degree 'Master' in specialty 123 “Computer Engineering”. – SHEI 'DonNTU', Lutsk, 2023.

Explanatory note: 105 p., 24 fig., 14 tables., 22 references.

This qualification work is dedicated to the development of computer systems for logistics, demonstrating how innovative algorithms and distributed computer systems can enhance the efficiency of supply chain management. The following tasks were accomplished during the research:

- Conducted a comprehensive analysis of the current state of logistics systems in Ukraine, including an assessment of the effectiveness of existing computer solutions.

- Explored the principles and methodologies for developing distributed computer systems, identifying key criteria for their effective design.

- Analyzed methods of clustering and algorithms for constructing optimal routes for logistic purposes, including the evaluation of the impact of input data on calculations.

- Developed an improved genetic algorithm for creating effective logistic routes, demonstrating its advantages over traditional methods.

- Designed the architecture of a distributed system, considering technical aspects of deployment, determined the choice of software and hardware, and developed a user interface.

Key words: Clustering, Route, Genetic Algorithm, Distributed System, Distributed Database, Software.

ЗМІСТ

ВСТУП 9
1 ОПИС ОБ'ЄКТА АВТОМАТИЗАЦІЇ 12
1.1 Характеристика об'єкта дослідження 12
1.2 Застосування маршрутизації на підприємствах 14
1.3 Огляд існуючих систем доставки товарів 16
1.4 Постановка задачі маршрутизації транспорту 19
1.5 Мета й задачі створення системи 21
1.6 Висновки 21
2 АЛГОРИТМ ВИЗНАЧЕННЯ МАРШРУТІВ ПЕРЕВЕЗЕННЯ 23
2.1 Формалізація задачі побудови маршрутів 23
2.2 Аналіз існуючих методів вирішення задачі кластеризації 25
2.3 Аналіз існуючих методів вирішення задачі маршрутизації 34
2.4 Обґрунтування вибору методу для рішення задачі маршрутизації 38
2.5 Проектування та розробка алгоритму вирішення задачі маршрутизації 42
2.6 Висновки 48
3 ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ ПІДСИСТЕМИ ВИЗНАЧЕННЯ МАРШРУТІВ 49
3.1 Вимоги до програмного забезпечення 49
3.2 Об’єктна модель системи 54
3.3 Інформаційне забезпечення 55
3.4 Опис модулів і об'єктів системи 59
3.5 Інтерфейс програмного забезпечення 62
3.6 Висновки 67
4 МОДЕЛЮВАННЯ Й АНАЛІЗ ЕКСПЕРИМЕНТАЛЬНИХ РЕЗУЛЬТАТІВ
4.1 Вхідні та вихідні дані 69
4.2 Аналіз експериментальних результатів 74
4.3 Аналіз параметрів алгоритму 75
4.4 Висновки 78
ВИСНОВКИ 80
ПЕРЕЛІК ПОСИЛАНЬ 82
Додаток А. Лістинг програмного забезпечення 84
Додаток Б. Охорона праці 99

ВСТУП

Актуальність теми. Транспортно-логістична галузь стала однією з ключових сфер сучасної економіки України. Вона відіграє критичну роль у глобалізації ринків, сприяючи ефективному руху товарів між різними країнами та континентами. Сучасна ділова динаміка вимагає від компаній максимальної оперативності, спроможності швидко реагувати на ринкові зміни та забезпечувати надійність своїх послуг.

Однак, зі зростанням вимог до оперативності та точності, традиційні методи планування транспортних маршрутів часто не витримують конкуренції через нездатність адаптуватися до змін умов та врахування непередбачуваних факторів. Це веде до підвищення вартості та зниження якості логістичних послуг.

У цьому контексті, розвиток та впровадження розподілених комп'ютерних систем для управління логістикою відкриває нові горизонти. Такі системи, завдяки своїй масштабованості та гнучкості, можуть ефективно обробляти великі обсяги даних, забезпечуючи оптимізацію маршрутів у реальному часі. Впровадження інтегрованих комп’ютерних систем вносять значний вклад у підвищення ефективності та надійності логістичних процесів, сприяючи зменшенню витрат та покращенню якості обслуговування. для управління логістичними процесами відкриває нові можливості для підвищення ефективності та конкурентоспроможності у цій галузі.

Таким чином, розробка та впровадження розподіленої системи для оптимізації логістичних маршрутів є не тільки актуальним, але й дуже перспективним напрямком дослідження. Це не тільки підвищує конкурентоспроможність компаній, але й сприяє загальному розвитку економіки, демонструючи важливість та перспективність таких досліджень у галузі комп'ютерних систем та компонентів.

Мета і задачі роботи. Метою роботи є концептуалізація, проектування та створення розподіленої комп’ютеризованої системи, здатної визначати оптимальні та квазі-оптимальні маршрути для вантажних перевезень, мінімізуючи витрати часу та ресурсів. При вирішенні задачі необхідно враховувати реалії і обмеження, з якими перевізники стикаються на практиці.

Задачі дослідження:

1)Аналіз сучасних методів, моделей, алгоритмів та інструментальних засобів оптимізації маршрутів вантажних перевезень.

2) Вивчення специфіки різних видів вантажів та особливостей їх транспортування.

3) Вибір та обґрунтування методу для вирішення задач оптимізації маршрутів

4) Розробка та програмна реалізація алгоритму складання оптимальних маршрутів доставки

5) Проектування архітектури розподіленої системи, зосереджуючись на гнучкості та масштабованості.

6) Втілення ключових компонентів цієї системи та їх інтеграція для забезпечення єдиної функціональної структури.

7) Експериментальна оцінка ефективності та точності розроблених рішень на основі практичних прикладів використання..

Об'єктом дослідження є процес визначення маршрутів перевезення по містах та областях країни.

Предметом дослідження є архітектура розподіленої системи, а також методи і алгоритми оптимізації побудови маршрутів руху транспортних засобів.

Методи дослідження. У процесі досліджень застосовано методи кластеризації, а саме метод «найближчого сусіда», метод Варда та метод k-середніх, генетичні алгоритми, а також методи об'єктно-орієнтованого проектування та програмування.

Наукова новизна одержаних результатів полягає у створені розподіленої системи, ядром програмного забезпечення якої є модифікований алгоритм визначення маршрутів перевезення, що дозволяє формувати оптимальні маршрути руху транспортних засобів під час доставки по кожній групі, отримуючи при цьому мінімальний сумарний пробіг усіх транспортних засобів і таким чином підвищуючи економічну ефективність роботи підприємства.

Крім того, розробка методології інтеграції розподілених систем у логістичні процеси може стати новим напрямком у вивченні логістичних систем.

Розробка системи, що може масштабуватися та адаптуватися до змінюваних обставин, може внести новизну у підхід до планування маршрутів.

Практична цінність отриманих результатів

1) Ефективність ресурсів: Автоматизована система може допомогти компаніям знизити витрати на планування та виконання перевезень, оптимізувавши маршрути та ресурси.

2) Зменшення помилок: Автоматизація процесу планування може зменшити кількість помилок, пов'язаних з людським фактором, та покращити якість послуг.

3) Швидкість реакції: Система, що працює в реальному часі, може дозволити компаніям швидко реагувати на непередбачені обставини, такі як аварії на дорогах чи зміни в попиті з боку клієнтів.

4) Стратегічне планування: Надана інформація та аналітика можуть бути використані для стратегічного планування та розвитку бізнесу.

5) Програмний додаток є цілком самостійним продуктом і може бути застосований в умовах будь-якої торгівельної компанії.

1 ОПИС ОБ'ЄКТА АВТОМАТИЗАЦІЇ

1.1 Характеристика об'єкта дослідження

Об'єктом автоматизації є процес визначення маршрутів розвезення товарів по території міста та суміжних регіонах. Головна мета – оптимізація дистрибуції з урахуванням географічних особливостей та потреб споживачів.

Використовуючи географічну інформаційну систему (ГІС), можна отримати карту автомобільних доріг із зазначеними пунктами доставки. Кожен пункт доставки має свої специфікації, такі як обсяг та асортимент товарів, а також їх фізичні параметри. Головна задача полягає в тому, щоб для заданого парку автотранспорту визначити оптимальні маршрути доставки, які мінімізують загальні витрати.

Вхідні дані для системи включають заявки від споживачів та матрицю відстаней між пунктами доставки. Процес розвезення товарів включає два ключових етапи: кластеризація пунктів доставки для формування оптимальних зон обслуговування та маршрутизація, або визначення порядку об'їзду пунктів в межах кожної зони.

Вирішення поставленої задачі розвозу товарів здійснюється у два етапи. На першому етапі вирішується задача розбивки регіону на компактні зони обслуговування – напрямки (групування об'єктів-одержувачів для кожного маршруту). Цю задачу будемо називати задачею кластеризації. На другому етапі вирішується задача знаходження оптимального за заданим критерієм (сумарній відстані, часу, вартості доставки), порядку об'їзду одержувачів для кожного маршруту. Цю задачу будемо називати задачею маршрутизації.

Наступним етапом є процедура формування маршрутних листів, де для кожного автомобіля прописується розклад руху, а також відмічається порядок завантаження автомобіля замовленими товарами в залежності від максимального об’єму, що може вмістити автомобіль, та від географічного положення всіх клієнтів, яких треба об’їхати. Для всіх цих етапів необхідно спроектувати та розробити алгоритми, що дозволяють одержувати результат в залежності від описаних параметрів та умов.

Результатом розрахунків є маршрутний аркуш у якому будуть зазначені:

– кількість клієнтів, яким треба доставити замовлення;

– кілометраж;

– час відправлення й час повернення на центральний або локальний склад підприємства;

– час знаходження на маршруті;

– час приймання товару в клієнта;

– час розрахункового прибуття до клієнта;

– усі ділянки доріг за якими проходить маршрут руху із вказівкою довжини кожного й напрямку повороту (така форма може використовуватися у випадку залучення найманих водіїв або при доставці за межі міста);

– примітки для експедитора за кожним клієнтом.

Надрукована форма маршрутного аркуша є необхідним і достатнім документом для водія й експедитора на маршруті, а також інструментом оцінки їх роботи. Повна форма маршрутного аркуша із вказівкою всіх ділянок руху дозволяє виключити помилки водія під час руху за маршрутом, скоротити час знаходження на маршруті й забезпечити прибуття товару покупцеві вчасно, незалежно від кваліфікації водія.

Таким чином, усі поточні заявки розбиваються за маршрутами та сортуються оптимальним чином. Після цього для кожного маршруту виводяться друковані

форми: для водія й експедитора – маршрутний аркуш, а для складу – аркуш складання, який містить список накладних для конкретного маршруту й дозволяє здійснювати на складі попереднє складання й навантаження товару відразу в тій послідовності, у якій він буде вивантажуватися на маршруті, згідно з накладними. Крім того, в аркуші складання зазначені примітки, які необхідно враховувати під час комплектування заявки на складі.

Таке складання дозволяє заощаджувати час під час завантаження машини на складі й оперативно вивантажувати товар у покупця. При наявності нормованих показників завантаження/вивантаження можна скласти щохвилинний розклад подачі автотранспорту, що повністю виключить простій автомобілів і дозволить вивести заявки на маршрути в найкоротший термін.

Заявки, які за якими-небудь причинами не були доставлені покупцеві й повернулися з розвозу, автоматично потрапляють до черги доставки наступного дня й виділяються в логіста як прострочені. Накладні, які з будь-яких причин не були доставлені клієнтам, потрапляють під особливий контроль логіста й можуть бути вивезені в першу чергу, що не тільки запобігає плутанині, але й допомагає згладити можливе невдоволення покупця порушенням термінів поставки.

1.2 Застосування маршрутизації на підприємствах

Маршрутизація на автомобільному транспорті – напрямок транспортного засобу через мережу автодоріг із забезпеченням переміщення за найкоротшим шляхом, з мінімальними витратами часу, (ідеально коли комбінація того й іншого).

Задачі, що стоять перед диспетчером:

1) Розробка схеми дислокації споживачів – нанесення на карту району, що обслуговується, точок доставки (необхідні точні адреси всіх клієнтів).

2) Визначення й виділення районів доставки (напрямків) – за створеною схемою.

3) Розробка графіка доставки за напрямками – з урахуванням побажань клієнтів щодо частоти доставки.

4) Складання матриць відстаней – для спрощення роботи.

Складання оперативних маршрутів (на підставі отриманих заявок клієнтів) відбувається відразу, як тільки всі зроблять заявки.

Для організації доставки товарів найчастіше використовується вірант кільцевого маршруту. Основними проблемами під час розробки подібних схем є велика кількість точок зупинки (розвантаження/завантаження) і визначення послідовності об'їзду. З економіко-математичних методів вибору послідовності об'їзду можна застосувати вирішення «задачі комівояжера», але через трудомісткість цього методу й великої кількості необхідних вихідних даних, його застосування на практиці в рамках підприємства поки не представляється можливим. У зв'язку із цим існує досить вдалий графічний спосіб визначення раціональних маршрутів – маршрути у вигляді квітки маргаритки (де немає пересічних шляхів і перетинань сусідніх маршрутів). Така схема може бути отримана в такий спосіб:

1) Вибрати найбільш вилучену точку маршруту (напрямку).

2) Знайти наступну доступну точку, найближчу до центру точок даної групи (квітки маргаритки). Додати цю точку до маршруту транспортного засобу.

3) Повторить крок 2, доки транспортний засіб не буде завантажений повністю.

4) Встановити послідовність зупинок (з урахуванням побажань вантажоодержувачів щодо часу доставки).

5) Знайти наступну найбільш вилучену точку, не включену до пелюстки й повторити попередні кроки.

Якщо конкретного часу клієнт не вказав, то розвіз слід починати із самої далекої точки маршруту. Для цього виїзду автомобіля на лінію повинен бути спланований так, щоб до цього пункту він прибув до початку часу його роботи. Як показує практика, останній вантажоодержувач повинен одержати товар не пізніше 14 годин, що так само необхідно враховувати під час складання маршруту.

1.3 Огляд існуючих систем доставки товарів

Система Arclogistics [1]. Цей програмний продукт розроблений провідною американською компанією ESRI, є важливим інструментом у сфері геоінформаційних систем (ГІС). Це програмне забезпечення є незамінним для диспетчерів у щоденному управлінні транспортними парками, пропонуючи функції розрахунку оптимальних маршрутів, створення маршрутних листів, ведення звітів та аналізу пробігу та ефективності роботи. Arclogistics, розроблений для користувачів, які не є експертами у сфері ГІС, має простий та зрозумілий інтерфейс.

Серед ключових переваг Arclogistics можна виділити наступні:

- Ефективне використання автопарку.
- Наявність докладних карт доріг по всій Європі.
- Сумісність з іншими продуктами та форматами даних ESRI.
- Гнучкі налаштування для транспортних засобів.
- Можливість інтеграції з зовнішніми інформаційними системами.
- Інструменти для створення звітів та маршрутних листів.
- Врахування зон відповідальності водіїв і можливість оперативного внесення змін.

Розглядаючи його основні можливості, слід відмітити, що використання Arclogistics для розробки ідеальних маршрутів сприяє скороченню загального

пробігу автомобілів, зниженню витрат на утримання автопарку та підвищенню рівня задоволеності клієнтів. Програма адаптована до особливостей кожного автомобіля, враховуючи його вантажопідйомність, максимальну кількість замовлень, вагу та вартість експлуатації за годину або кілометр. В залежності від вимог, користувачі можуть вибирати між оптимізацією витрат та пріоритетом оперативності доставки. Ключовою функцією є геокодування, яке включає локалізацію адрес на карті. Автоматизація цього процесу в Arclogistics зменшує навантаження на операторів та скорочує час обробки замовлень. Програма також дозволяє призначати транспортні засоби до певних зон і враховувати дорожні перепони. Створення маршрутних листів та звітів виконується через Crystal Reports, що дозволяє генерувати документацію на основі даних з різних баз, у тому числі Arcgis. Crystal Reports надає можливість створювати індивідуальні шаблони звітів для щоденного використання. Arclogistics, як самостійний продукт, не вимагає додаткового програмного забезпечення з лінійки Arcgis [3].

Система Truckstops. Truckstops – це ГІС, створена компанією Microanalytics, є передовим програмним рішенням для маршрутизації та планування транспорту, призначеним для підприємств, які експлуатують великі автопарки (від п'яти та більше транспортних засобів). Цей продукт допомагає оптимізувати різні аспекти логістики, включаючи загальний час подорожі, відстань, витрати на оплату праці водіїв, обслуговування транспорту та витрати на паливо. Використовуючи Truckstops, компанії мають змогу знижувати витрати на доставку, покращувати обслуговування клієнтів, розробляти економічно ефективні маршрути та підвищувати рівень управління та контролю [4].

Наступним з топ-систем є ГІС «Ділова карта» [5]., Створена компанією ІНГІТ, ця ГІС має за основу потужний і адаптивний до змін параметрів механізм для розрахунку логістики доставки вантажів. Його гнучкість дозволяє адаптувати систему до різноманітних логістичних завдань: від розвезення товарів з

центрального складу до збору вантажів із різних пунктів на центральний склад, а також координації доставок між декількома складами або філіями в складі одного підприємництва. Проте, така універсальність вимагає великої кількості вхідних даних для налаштування, що може ускладнити використання програми для дуже специфічних задач.

До функціональних можливостей ГІС «Ділова карта» відносяться такі:

1) Управління базою даних клієнтів, з автоматичним відображенням адрес на карті, сортуванням за адміністративно-територіальними одиницями та власно створеними зонами, як-от дилерські квартали або зони обслуговування.

2) Технологія прив'язки адрес, яка забезпечує автоматизований підбір адрес, ідентифікацію помилок і надання рекомендацій.

Інтегруваня Google Map та інших сучасних карт в програмне забезпечення дозволяє ефективно управляти базою даних клієнтів, виконуючи вибірки за різними просторовими параметрами, такими як зони обслуговування та відстані до певних точок. Керуючись географічним розташуванням клієнтів, ГІС дозволяє планувати маршрути з урахуванням таких факторів, як довжина маршруту, кількість обслуговуваних клієнтів та загальне навантаження, наприклад, вагу або об'єм, порівнюючи ці дані з можливостями транспортних засобів.

Автоматизована система планування маршрутів враховує організацію дорожнього руху (дорожнє покриття, можливі затори на дорогах, погодні умови. тощо) і забезпечує створення оптимальних маршрутів для доставки, виходячи з вибраних з бази даних або відібраних на карті клієнтів з урахуванням обмежень за вагою та кількістю. Програма дозволяє налаштовувати маршрути згідно з різними користувацькими алгоритмами.

Унікальний алгоритм розрахунків у цій системі дозволяє швидко планувати маршрути для всього автопарку, виконуючи всі заплановані денні замовлення. Під час розрахунків враховуються характеристики транспорту, терміновість

замовлень, час доставки, оптимізація маршрутів з урахуванням дорожніх обмежень та інші фактори, наприклад, для доставки свіжих продуктів. Система також надає динамічний огляд роботи транспорту, дозволяючи в будь-який момент бачити на карті розташування транспортних засобів та їхню активність.

Єдиним недоліком є вартість програми, яка стартує від 1200 доларів, з додатковою платою 50 доларів за підключення кожного нового робочого місця.

1.4 Постановка задачі маршрутизації транспорту

Задача маршрутизації транспорту (Vehicle Routing Problems, VRP) [6] – задача комбінаторної оптимізації, у яких для парку транспортних засобів, розташованих в одному або декількох депо, повинен бути визначений набір маршрутів до декількох віддалених точок-споживачів. Зацікавленість у задачі маршрутизації транспортних засобів (VRP) зумовлена її важливістю для практичних застосувань, не дивлячись на її складність. На рис. 1.1 наведено узагальнену схему доставки товарів при вирішенні задачі маршрутизації транспорту на підприємстві.

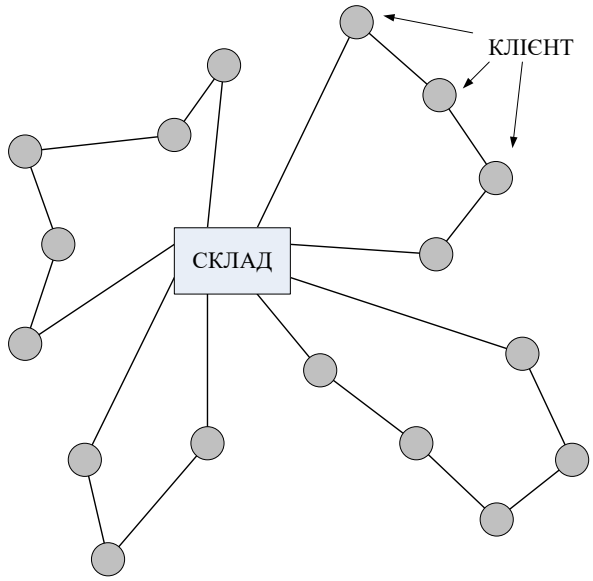


Рисунок 1.1 – Схема доставки товарів

VRP – добре відома задача цілочисельного програмування, яке відноситься до класу NP-важких задач, що означає, що обчислювальна складність задачі залежить від розміру вхідних даних експоненціально.

Зазвичай, в таких задачах шукаються квазі-оптимальні рішення, знаходяться досить швидко й досить точні для необхідних цілей. Звичайно це досягається різними евристичними методами.

Задача маршрутизації транспортних засобів (VRP) поєднує елементи двох інших відомих проблем:

Задача комівояжера (TSP): Якщо вважати, що кожен транспортний засіб має необмежену вантажопідйомність (або достатню для всіх вантажів), то VRP перетворюється на множинну задачу комівояжера (MTSP). Це досягається шляхом додавання до початкового графа к-1 копій початкової точки і її з'єднань (де к - кількість маршрутів), але без з'єднань між цими копіями.

Задача про впакування рюкзака (BPP): Вирішення цієї задачі стає еквівалентним вирішенню VRP, якщо всі відстані вважаються рівними нулю. У такому випадку, оптимізація зводиться до розподілу вантажів по транспортних засобах таким чином, щоб врахувати обмеження на вантажопідйомність, при цьому ефективність будь-якого прийнятного рішення буде однаковою.

Не дивлячись на наявність величезної кількості ГІС, задача маршрутизації є ключовим в галузі транспортних перевезень, переміщення й логістики. У багатьох областях ринку доставка товару додає до його вартості суму, порівнянну з вартістю самого товару. Проте, використання комп'ютерних методів оптимізації доставки товару часто виражається в економії порядку 5-20% від загальної його вартості.

1.5 Мета й задачі створення системи

Метою створення даної підсистеми є підвищення економічної вигідності транспортних вантажоперевезень: збільшення середнього завантаження рейсу, збільшення обсягів перевезень, збільшення кількості точок доставки. Також під час використання даної системи буде прискорений процес складання маршрутів диспетчером.

Система повинна, враховуючи пріоритети терміновості виконання замовлень, вирішувати наступні задачі:

– розрахунки оптимальних з точки зору транспортних витрат і часових витрат маршрутів доставки замовлень до клієнтів зі складу з урахуванням їх потреб, можливостей автотранспорту й вартості транспортування різними типами автомобілів;

– ведення обліку заявок клієнтів;

– ведення обліку складених маршрутів.

1.6 Висновки

У контексті аналізу інформаційних систем, що відповідають критеріям вирішення поставлених задач в області транспортної логістики, було ідентифіковано ряд недоліків у наявних програмних продуктах. Поширеним мінусом багатьох комерційних рішень є висока вартість та наявність зайвих функціональних можливостей, які не завжди корелюють з потребами конкретних підприємств, що негативно позначається на економічній доцільності їх застосування. Придбання таких систем може призводити до надмірних фінансових витрат на невикористані можливості.

Імпорт програмних продуктів іноземного виробництва може зумовити додаткові логістичні витрати, що істотно підвищує їх кінцеву вартість, а також ускладнює процес локалізації. Через збереження авторських прав на програмні продукти, вони постачаються без вихідних кодів, що обмежує можливості їх кастомізації без додаткових фінансових витрат та очікування оновлень. Закритий характер коду обмежує доступ до повної інформації про використовувані алгоритми, не дозволяючи користувачам адекватно оцінювати ефективність знайдених рішень.

В якості розв'язку транспортної задачі було обрано концепцію маршрутизації з урахуванням вантажопідйомності транспортних засобів (CVRP), яка відповідає поставленим вимогам до проектування та розробки комп’ютеризованої розподіленої системи підтримки прийняття рішень логістичних завдань.

2 АЛГОРИТМ ВИЗНАЧЕННЯ МАРШРУТІВ ПЕРЕВЕЗЕННЯ

2.1 Формалізація задачі побудови маршрутів

Для формалізації задачі побудови маршрутів розвезення необхідно зробити аналіз дорожньої мережі, що містить значну кількість точок обслуговування (вузли), через які мають проходити транспортні маршрути. Для цього дорожній мережі присвоюється відповідний граф, у якому вершинами виступають вузли мережі, а ребра графа представляють собою дорожні відрізки, що з'єднують ці вузли. Для кожного ребра графа визначається його довжина, яка еквівалентна відстані між двома вузлами, які воно з'єднує.

До системи надходять заявки від клієнтів зі списком обраних товарів та їх кількістю. Кожна позиція характеризується найменуванням і обсягом упакування, який буде враховуватися під час завантаження транспортного засобу. На вхід подається матриця відстаней між клієнтами, де кожна ланка зберігає відстань від клієнта з номером стовпця, до клієнта з номером рядка.

Перед системою ставиться задача пошуку оптимальних маршрутів, що починаються на складі підприємства, далі прокладається маршрут між клієнтами з урахуванням завантаженості кузова автомобіля й закінчується у вихідній точці на складі. Так само дану задачу можна модифікувати, використовуючи замість відстаней середній час у дорозі або вартість грошових витрат.

Тому відстані між об'єктами визначаються за допомогою квадратної матриці відстаней C=[i,j] з розмірністю n×n, де C[i,j] представляє собою відстань від пункту i до пункту j. Важливо зазначити, що ця матриця відстаней, як правило, не є симетричною.

Існує N клієнтів, до яких необхідно організувати доставку з мінімізацією загальних витрат. Для кожного маршруту існують певні обмеження, які впливають на його планування:

– будь-який маршрут автомобіля має бути таким, щоб автомобіль виїжджав зі складу один раз та повертався на склад теж один раз, тобто замикався;

– у кожного клієнта автомобіль повинен побувати лише один раз, тобто треба обов'язково обійти всіх клієнтів, при цьому не побувавши в жодного двічі.

Для розрахунків витрат існує матриця переходів, що містить витрати на перехід з одного клієнта до іншого, при цьому вважається, що можна перейти з будь-якого клієнта до кожного, крім того ж самого (у матриці викреслюється діагональ). Метою вирішення є пошук такого маршруту, що буде мати найменшу довжину, задовольняючи, при цьому, всім умовам.

У першу чергу необхідно розбити множину клієнтів на групи (кластери), які будуть обслуговуватися одним автомобілем, але щоб обсяг замовленої продукції даними клієнтами не перевищував обсягу кузова автомобіля. Далі для кожного кластера вирішується задача пошуку оптимального маршруту об'їзду клієнтів (задача маршрутизації).

Математична модель задачі маршрутизації:

N – число клієнтів у кластері;

$C[i,j]$, i, j = 1..N – матриця відстаней між клієнтами, де C_{ij} – відстань від i-го клієнта до j-го;

V_i – сумарний обсяг товарів у заявці для кожного клієнта;

V_{al} – обсяг кузова автомобіля;

V_{ml} – обсяг товарів у кузові, під час розвозу;

l = 1..K, де K – кількість оперативних маршрутів;

X_{ij} – матриця переходів зі значеннями:

X_{ij} = 1, якщо автомобіль робить перехід з i-го клієнта до j-го,

X_{ij} = 0, якщо не робить переходу,

де i, j = 1..N та i≠j.

Критерій:

$$F(X) = \sum_{i=1}^{N} \sum_{j=1}^{N} C_{ij} \cdot X_{ij} \to \min , \quad (2.1)$$

де X_{ij} – матриця переходів, де X_{ij} = 1, якщо перехід зроблений і X_{ij} = 0 якщо ні.

Обмеження:

$$\sum_{i=1}^{N} X_{ij} = 1, \text{ i = 1..N} \quad (2.2)$$

$$\sum_{j=1}^{N} X_{ij} = 1, \text{ j = 1..N} \quad (2.3)$$

$$V_{ml} < V_{al} \quad (2.4)$$

Умова (2.2) означає, що автомобіль від кожного клієнта виїжджає тільки один раз; умова (2.3) – заїжджає до кожного клієнта лише один раз; умова (2.4) визначає неможливість перевантаження кузова транспортного засобу товаром [1].

2.2 Аналіз існуючих методів вирішення задачі кластеризації

Задача кластеризації реалізується набором методів (алгоритмів), кожний з яких здійснює розбивки регіону на компактні зони обслуговування.

Апарат нечітких множин - метод заснований на виборі транзитивно найближчих повідомлень. При даному підході кожному об'єкту повідомлення ставиться у відповідність пункт на карті. Пункти характеризуються відстанями між собою [9].

Нехай e = max a(i, j) по всім i, j – максимальний елемент матриці відстаней A. Тоді матриця B = [b(i, j)] розмірності n x n, де b(i, j) = 1 - a(i,j)/e задає нечітке відношення подібності. $0 \le b(i, j) \le 1$.

Транзитивне замикання нечіткого відношення подібності задається матрицею D = [d(i, j)] розмірності n x n. $0 \le d(i, j) \le 1$.

$D = B \cup B2 \cup B3 \cup \ldots \cup Bn$,

де U – операція об'єднання нечітких відносин (MAX).

B2 = B ° B – являє собою (Max - min) – композицію нечіткого відношення самого на себе.

D – матриця, що задає нечітке відношення еквівалентності (подоби). Це відношення рефлексивне, симетричне й транзитивне. Згідно з теоремою про декомпозицію для відношення подоби, для кожного значення матриці D (у порядку зростання їх значення) одержуємо транзитивно найближчі повідомлення (пункти).

Таким чином, у підсумку одержуємо розбивку множини об'єктів на заздалегідь задане число компактних (транзитивно-найближчих) груп.

Розфарбування графів [10]. Задача кластеризації може бути зведена до задачі розфарбування вершин графа. Для цього будується граф несумісності. Вершинам графа відповідають об'єкти-споживачі (пункти на карті), і дві вершини – суміжні, якщо відповідні до них об'єкти не можуть перебувати в одній групі. Причини несумісності визначаються різними факторами: велика відстань між об'єктами, відмінність заявок – об'єктами замовлені різні товари, і т.п.

Ставиться задача розфарбування вершин такого графа несумісності при різних умовах (обмеженнях, критеріях), серед яких:

– Мінімальне розфарбування графа (отримання мінімального числа компактних груп).

– Розфарбування графа в задане число кольорів (розбивка на групи, що відповідають певному парку автотранспорту та можливостям перевезення).

– Розфарбування графа в задане число кольорів із обмеженням на кількість вершин, які мають майже однакові відтінки (розбивка на групи, що відповідають певному парку автотранспорту та можливостям перевезення).

Для реалізації методу використовуються оцінки хроматичного числа графа й спектр алгоритмів розфарбування графів, що враховують розмірність задачі, що вирішується й відведений на це час [10].

Частотно-матричний метод [11]. Підхід до постановки задачі аналогічний попередньому, але в якості вихідної моделі розглядається матриця інціденцій $Q = [q(i, j)]$. Стовпцям матриці відповідають букви – об'єкти (пункти на карті), рядкам – підмножина, обумовлена словесним відношенням S_i, i = 1, 2, …, n, та q(i, j) = 1, якщо до і-го слова входить j-та буква, й q(i, j) = 0 – якщо ні.

Матриця інцидентності будується таким чином, що кожне словесне відношення визначається за умови спільності (або неспільності) входження відповідних об'єктів до однієї групи.

На основі побудованої матриці визначається частотна матриця відносин $F = QT$ x Q, елементи якої f(i, j) використовуються для цілеспрямованого знаходження найближчих об'єктів і включення їх до однієї групи. В основі даного підходу лежить апарат диференціювання дискретних моделей [10].

Алгоритм Свіра [12]. Розробка кільцевих маршрутів на початковому етапі можлива за допомогою методу, відомого як алгоритм Свіра або метод двірника-склоочисника. Цей метод передбачає розміщення споживача матеріального потоку у полярній системі координат. Центр цієї системи, точку 0, розташовують на локації розподільного складу. Встановлюється початкове положення полярної осі, f=0. Місцеположення споживача визначається за допомогою двох параметрів: відстані від центру (полюса системи) та кута f, який формується між полярною віссю і напрямком на споживача, що виходить з точки 0.

Варто відмітити, що метод, описаний вище, є ефективним в умовах евклідової транспортної мережі. Це означає, що відстань між вузлами мережі вздовж існуючих доріг є приблизно рівною прямолінійній відстані між ними.

На кільцеві маршрути можливе накладення додаткових умов, таких як обмеження за часом. У випадку, коли час проходження певного кільцевого маршруту перевищує дозволений ліміт, потрібно відкоригувати розмір цього сектора, зменшуючи його і відповідно збільшуючи сусідній сектор. Алгоритм Свіра дозволяє раціонально розділити обслуговувану зону на кілька секторів для оптимізації маршрутизації.

Метод k-середніх [11] – Один з найвідоміших і елементарних підходів до кластеризації елементів полягає у використанні методу, що орієнтується на мінімізацію внутрішньокластерної варіативності та максимізацію міжкластерної відстані. Алгоритм k-середніх працює, обраховуючи квадрати евклідових відстаней між точками даних у кластері та вектором, який відображає середнє значення кластера. Метод досягає збіжності та формує кінцеву кількість кластерів k, коли зазначена сумарна відстань мінімізується.

Цей алгоритм є варіацією EM-алгоритму (Expectation-Maximization) і розділяє набір елементів у векторному просторі на попередньо задану кількість кластерів k. Основна концепція полягає в тому, що на кожній ітерації відбувається перерахунок центру маси кожного кластера на основі даних попереднього етапу, після чого точки даних знову розподіляються по кластерах в залежності від того, до якого з нових центрів вони ближчі.

Алгоритм завершується, коли на певній ітерації не відбувається змін у положенні центрів мас кластерів. Це гарантовано станеться через кінцеве число ітерацій, оскільки загальна кількість можливих розподілів кінцевої множини обмежена, а сумарне квадратичне відхилення V не зростає з кожним кроком, що виключає можливість зациклення. Алгоритм призначає кожній точці даних лише

один кластер і не допускає невизначеності у питанні приналежності точки до певного кластера.

Метод найближчого сусіда [11] полягає в тому, що перед ініціацією алгоритму виконується розрахунок матриці евклідових відстаней між аналізованими об'єктами. Алгоритм "найближчого сусіда", інакше відомий як метод одиночного зв'язку, базується на послідовному аналізі цієї матриці, визначаючи на кожній ітерації мінімальне значення відстані, що корелює з двома найближчими об'єктами. Ці об'єкти конституюють первинний кластер.

У подальшому процесі, вибір наступного об'єкта для включення у кластер здійснюється на основі критерію найменшої відстані до будь-якого з уже включених об'єктів у кластер. Процедура інкорпорації нових об'єктів у кластер продовжується за аналогічним принципом на наступних ітераціях.

Цей процес ітеративно повторюється до досягнення кінцевої стадії, коли усі окремі кластери злиті в єдину агломеративну структуру, формуючи тим самим один загальний кластер.

Метод Варда (Ward’s method) [13] Цей метод кластерного аналізу унікальний у своєму роді, оскільки він апелює до принципів дисперсійного аналізу для визначення відстаней між кластерами. Ключовою особливістю методу є мінімізація загальної суми квадратів відхилень (SS) для будь-якої пари потенційних кластерів, які можуть бути утворені на кожному етапі алгоритму.

В ході процедури спочатку проводиться обчислення середніх значень для всіх змінних у кожному з кластерів, що базуються на усіх доступних спостереженнях. Надалі визначаються квадрати евклідових відстаней від кожного окремого спостереження до середнього значення його кластера. Ці відстані сумуються для отримання інтегральної дистанції.

Далі, на основі цієї інтегральної суми, відбувається об'єднання тих кластерів, які при злитті спричиняють найменше збільшення загальної суми відстаней. За

загальною оцінкою, цей метод є високоефективним у кластеризації, однак має тенденцію формувати кластери меншого розміру.

З розглянутих методів для дослідження слід виділити наступні методи: метод найближчого сусіда, метод Варда та метод k-середніх. Дані методи було досліджено на основі вхідних даних про клієнтів та замовлення, серед яких було відібрано наступні показники: віддаленість клієнта від центрального складу, обсяг замовлення. Конкретні значення показників було про нормовано за наступними формулами:

$$K_i^j = \frac{X_i^j - X_{\min.i}}{X_{\max.i} - X_{\min.i}} \tag{2.5}$$

$$K_i^j = 1 - \frac{X_i^j - X_{\min.i}}{X_{\max.i} - X_{\min.i}}, \tag{2.6}$$

де X_i^j – значення і-го показника j-ї клієнта;

$X_{\min.i}$ – мінімальне значення і-го показника клієнта серед усіх клієнтів;

$X_{\min.i}$ – максимальне значення і-го показника клієнта серед усіх клієнтів.

Формула (2.5) використовувалась для випадку, коли більші значення відповідають кращому результату, а формула (2.6) – коли менші значення відповідають кращому результату. Таким чином, отримані нормовані показники K_i^j знаходяться в інтервалі [0;1].

Дослідження проводились за допомогою вбудованих функцій пакета Matlab. Для реалізації використовувалась матриця NxM, де M – кількість характеристик, а N – кількість клієнтів, яка при дослідженні змінювалась від 3 до 100. При цьому отримана кількість кластерів варіювалася від 3 до 10. На рис. 2.1 представлено результати кластеризації за методом «найближчого сусіда» для N=30 у вигляді дендрограми (ієрархічного дерева кластерів). Вихідні дані для наведеного варіанта було згруповано в 3 кластери.

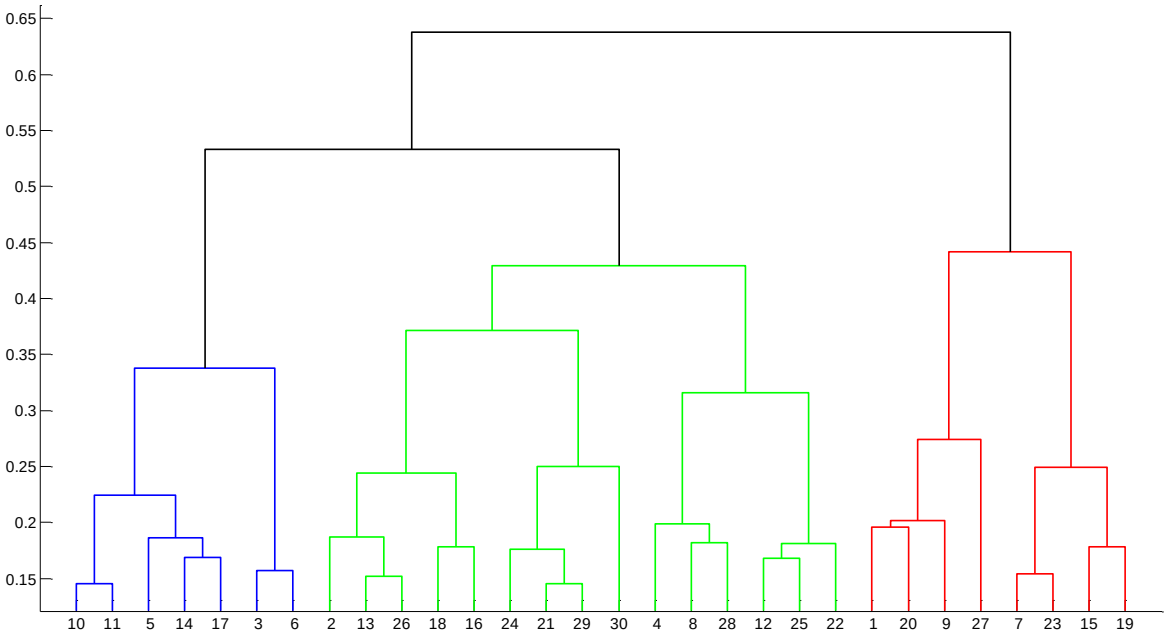


Рисунок 2.1 – Результати кластеризації для N=30 за методом «найближчого сусіда»

На рис. 2.2 представлено результати кластеризації за методом k-середніх для N=30. Вихідні дані для наведеного варіанта було згруповано в 4 кластери.

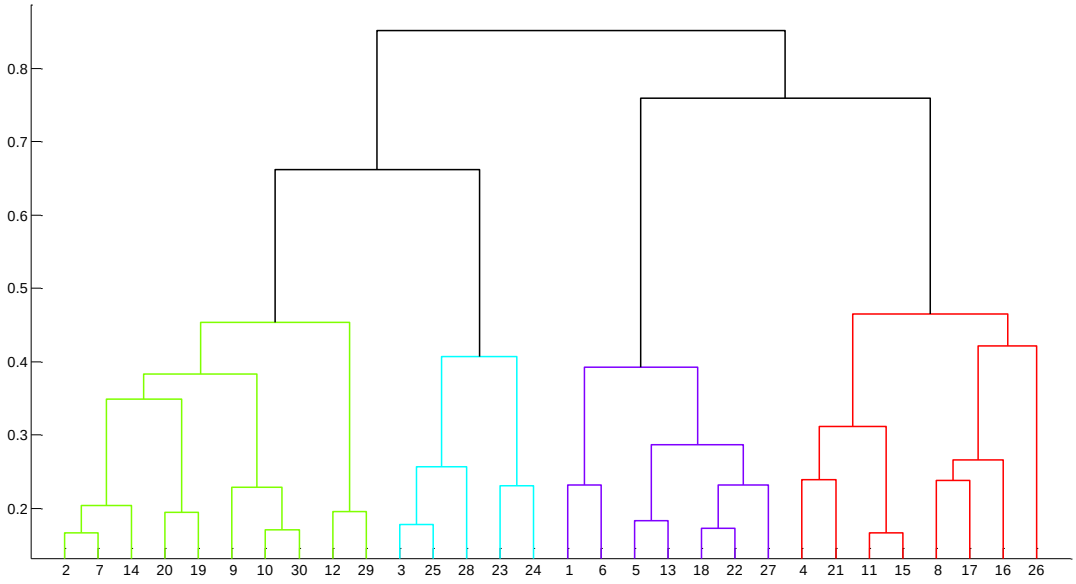


Рисунок 2.2 – Результати кластеризації для N=30 за методом k-середніх

На рис. 2.3 представлено результати кластеризації за методом Варда для N=30. Вихідні дані для наведеного варіанта було згруповано в 5 кластерів.

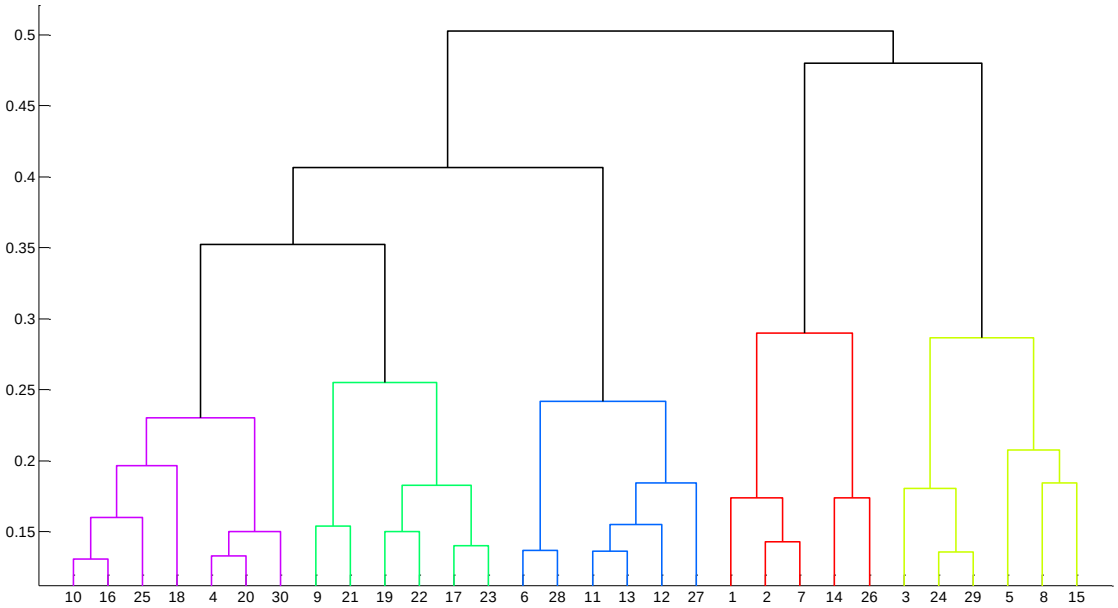


Рисунок 2.3 – Результати кластеризації для N=30 за методом Варда

З точки зору кластеризації найкращі результати було досягнуто при використанні методу «найближчого сусіда».

На рис.2.4-2.6 представлено результати кластеризації у вигляді точкових графіків. Кожна точка відповідає одному клієнту. Для відображення точок на площині було введено нормований показник $K_{норм}$, що узагальнює вихідні характеристики клієнта. Цей показник є абстрактним та безрозмірним. Результати представлено для N=30 (рис. 2.4, 2.5) та N=100 (рис. 2.6) із кількістю кластерів 5, 7 та 10 відповідно.

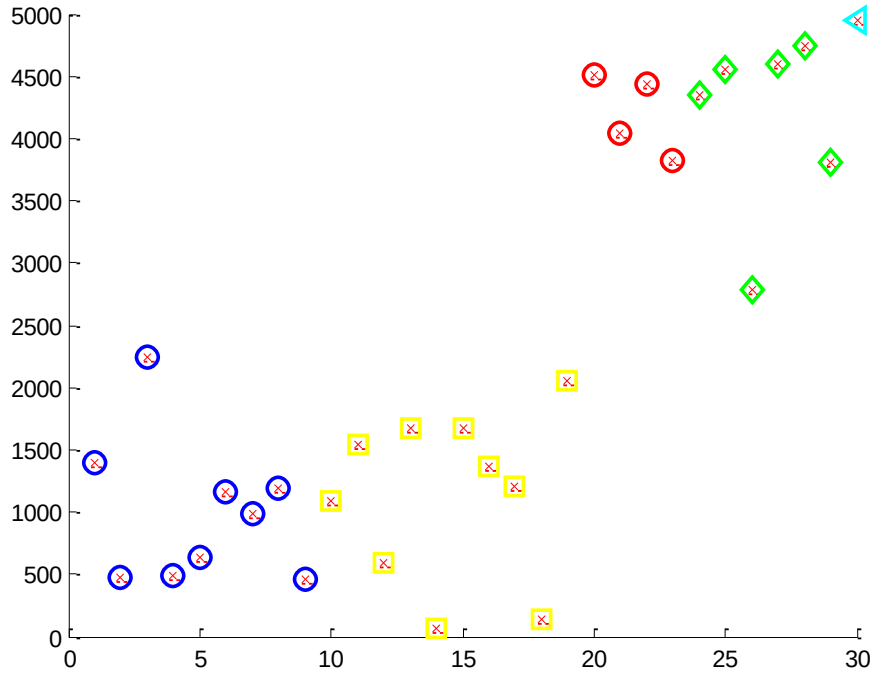


Рисунок 2.4 – Результати кластеризації у вигляді точкового графіка для N=30 за методом «найближчого сусіда» із групуванням у 5 кластерів

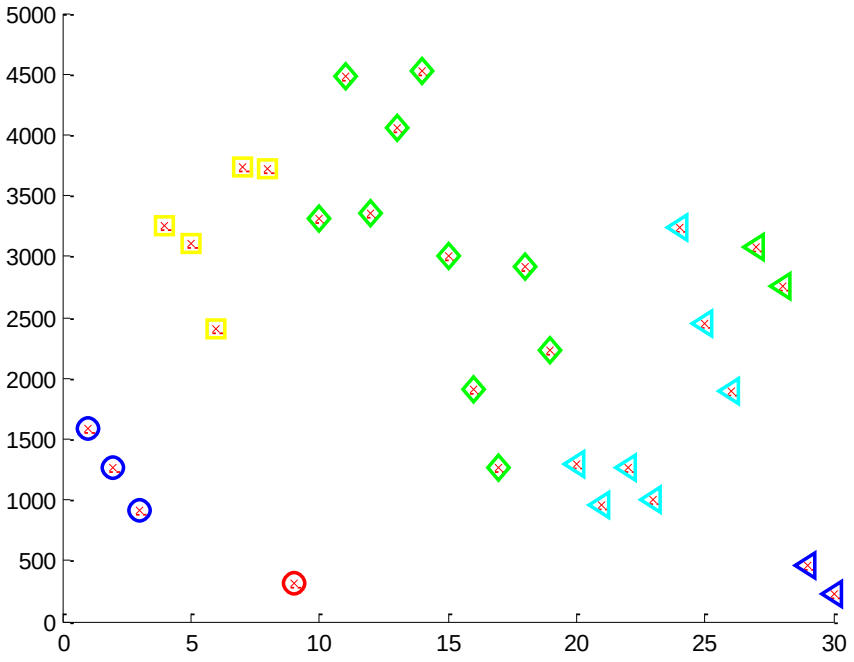


Рисунок 2.5 – Результати кластеризації у вигляді точкового графіка для N=30 за методом «найближчого сусіда» із групуванням у 7 кластерів

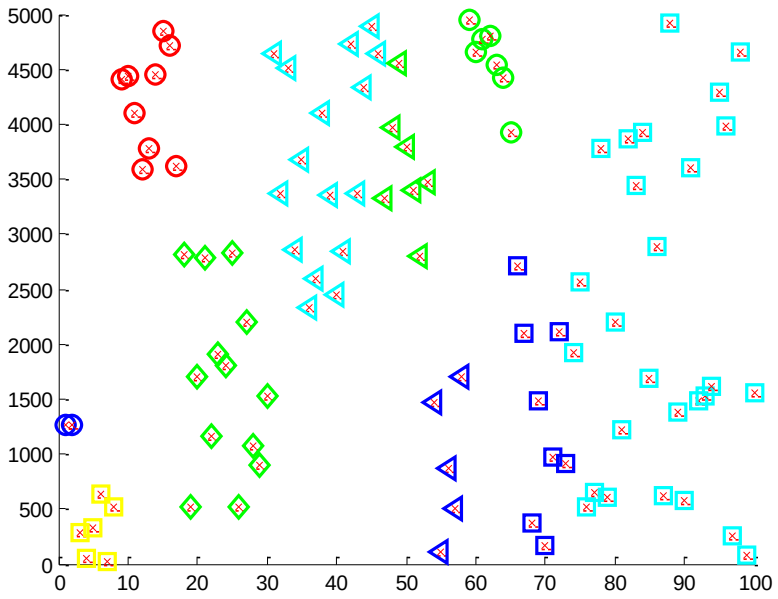


Рисунок 2.6 – Результати кластеризації у вигляді точкового графіка для N=100 за методом «найближчого сусіда» із групуванням у 10 кластерів

Отримані результати є задовільними з точки зору кластеризації, однак використання зазначених методів занадто ускладнює або взагалі робить неможливим знаходження ефективного рішення задачі маршрутизації усередині кластерів, оскільки дані методи не враховують вантажопідйомність автомобілів, що будуть розвозити замовлення по отриманих кластерах. Тому було прийнято рішення у якості методу кластеризації взяти за основу метод «найближчого сусіда», однак виконати деяку його модифікацію для урахування вантажопідйомності автомобілів. Опис розробленого методу буде наведено далі разом з описом методу маршрутизації.

2.3 Аналіз існуючих методів вирішення задачі маршрутизації

У рамках оптимізації логістичних систем, важливим аспектом є використання комп’ютеризованих методів для вирішення задач маршрутизації. Це

включає алгоритми для розрахунку найефективніших маршрутів доставки, як-от задача комівояжера [14], яка передбачає визначення оптимального маршруту між набором місць доставки. Сучасні комп’ютерні системи використовують складні алгоритми, такі як мурашині алгоритми для моделювання поведінки мурах та імітації відпалу для оптимізації маршрутів.

Задача маршрутизації реалізується набором алгоритмів, кожний з яких здійснює вирішення задачі комівояжера. Комівояжер повинен об'їхати всіх одержувачів товарів усередині однієї зони обслуговування. Він виїжджає з деякого пункту й повинен повернутися до нього ж наприкінці подорожі. Передбачається, що комівояжер ніколи не буває двічі в одному пункті. Відстані між одержувачами товарів задаються за допомогою квадратної матриці відстаней C = [c(i, j)] розмірності k x k, де k – кількість одержувачів товарів, c(i, j) – відстань від одержувача i до одержувача j. Відзначається, що у загальному випадку матриця відстаней не є симетричною. Діагональні елементи матриці відстаней дорівнюють нулю.

Задача комівояжера полягає в такому об'їзді всіх одержувачів, щоб сумарна пройдена відстань була мінімальною. Слід обрати один оптимальний маршрут з $(k-1)$ можливих [14]. Розглянемо декілька варіантів вирішення задачі.

Мурашині алгоритми [15] представляють собою стохастичну евристику, базуючись на принципах імовірності, які калібруються з урахуванням інформації про ефективність рішень, виведених з попередніх ітерацій. Ці алгоритми застосовні як до статичних, так і до динамічних задач комбінаторної оптимізації. Забезпечується гарантована збіжність до оптимального рішення, хоча темпи збіжності можуть бути невизначеними.

Процес роботи алгоритму імітує поведінку мурах, орієнтуючись на розподіл феромону по тропах, які відповідають ребрам графа у задачі комівояжера. Імовірність включення конкретного ребра до маршруту мурахи корелює з

кількістю феромону на цьому ребрі. Чим коротший маршрут, тим більше феромону відкладається на його ребрах, збільшуючи шанси, що інші мурахи будуть слідувати цим шляхом.

Однак такий підхід, що базується лише на позитивному зворотному зв'язку, може призвести до передчасної збіжності, коли більшість мурах слідує локально оптимальним маршрутам. Це можна уникнути, імплементуючи негативний зворотний зв'язок у вигляді випару феромону. Швидке випарування феромону може спричинити втрату пам'яті колонії та забуття ефективних рішень, у той час як повільне випарування може закріпити колонію на локально оптимальних рішеннях.

На старті алгоритму рівень феромону на кожному ребрі ініціалізується низьким, але позитивним значенням. Загальна кількість мурах в алгоритмі фіксована і відповідає кількості міст у задачі, причому кожна мураха розпочинає свій маршрут із власного міста.

Однією з можливих модифікацій алгоритму є впровадження так званих «елітних» мурах. Ці мурахи спеціально призначені для посилення ребер, що входять у склад найкращого маршруту, виявленого з моменту запуску алгоритму. Такий підхід сприяє підвищенню ваги оптимальних шляхів у процесі пошуку рішення, тим самим прискорюючи конвергенцію до більш ефективних маршрутів.

Метод гілок і меж [16] є варіантом поліпшеного перебору, що вдосконалює класичний метод повного перебору. У задачі комівояжера, де міста вважаються вершинами графа, а зв'язки між ними - дугами (i,j), вимога знаходження мінімального шляху, який проходить через кожне місто один раз і повертається назад, трансформується у пошук мінімального гамільтонового контуру [21] на графі.

Практичне застосування методу гілок і меж у контексті цієї задачі вимагає розробки способу визначення нижніх границь для підмножин та розбиття множини можливих гамільтонових контурів на ці підмножини (розгалуження).

Процес визначення нижніх границь ґрунтується на принципі, що зміна будь-якого рядка i або стовпця j матриці відстаней C на певну величину не впливає на оптимальність загального маршруту комівояжера, змінюючи лише довжину кожного потенційного гамільтонового контуру на цю ж величину.

Алгоритм Дейкстри [17] розв'язує завдання знаходження мінімального шляху між двома заданими вершинами в орієнтованій, неорієнтованій, або змішаній мережі (де деякі дороги мають односторонній рух). Для реалізації алгоритму використовуються три масиви з n елементів кожен, де n - це кількість вершин мережі. Перший масив *a* використовується для маркування вершин статусами 0 (не розглянута) та 1 (вже розглянута). Другий масив *b* зберігає поточні мінімальні відстані від v[i] до відповідної вершини. Третій масив *c* містить номери вершин, де c[k] позначає номер передостанньої вершини на найкоротшому шляху від v[i] до v[k]. Матриця відстаней D[i,k] визначає довжини дуг *d*[i,k]; для випадків, коли дуга відсутня, d[i,k] призначається значення Б, що еквівалентно "машинній нескінченності".

Одним з потенційних недоліків алгоритму є необхідність знання не лише матриці відстаней, але й точних координат кожного міста у площині. Якщо відомі тільки відстані, але невідомі координати, це вимагатиме розв'язання n систем квадратних рівнянь із n невідомими для кожної координати, що стає особливо складним при збільшенні кількості міст. Натомість, якщо є доступні координати міст, але відсутня матриця відстаней, створення такої матриці є значно простішим завданням. [17].

Метод градієнтного спуску [16]. Суть методу полягає в наступному. Будується послідовність $\{x0,x1,x2,x3,\ldots\} \subset X$, у якій початковий елемент *x0*

обирається довільно (можливо, випадковим образом), а кожний наступний є одним із сусідів попереднього, причому саме тим із сусідів, для якого значення функції *U* буде найменшим. Побудова послідовності завершується тоді, коли послідовність значень цільової функції $\{U(x0), U(x1), U(x2), U(x3), \ldots\}$ припинить монотонно спадати.

Останній елемент побудованої послідовності називають точкою локального мінімуму. Це така точка, у яких значення *U* суворо менше, ніж у всіх сусідніх з нею. У слово «локальний» укладений головний недолік описаного методу. Локальних мінімумів у функції *U* може бути багато, і кожному з них відповідає, як правило, своє локальне мінімальне значення цільової функції. Нас же цікавить абсолютний мінімум функції й той елемент множини *X*, у якому він досягається. Якби було легко знайти всі точки локального мінімуму, перебором серед них ми знайшли б точку абсолютного мінімуму. Але метод градієнтного спуску не дає рецепта пошуку всіх точок локального мінімуму, він дозволяє знайти лише яку-небудь [17].

Метод імітації відпалу [18] є модифікацією імовірнісного методу градієнтного спуску. Відмінність полягає в поведінці алгоритму, коли $U(x) - U(\tilde{x})$, де x – черговий елемент послідовності, а $\tilde{x}$ – його сусід, обраний навмання. Імовірнісний метод градієнтного спуску відкидав такого сусіда безумовно, а метод імітації відпалу допускає додавання такого «поганого» сусіда до послідовності, щоправда, з деякою ймовірністю *p*, що залежить від того, наскільки поганий сусід погіршив цільову функцію.

2.4 Обґрунтування вибору методу для рішення задачі маршрутизації

У сфері комп’ютерних систем і компонентів, особливої уваги заслуговують генетичні алгоритми. Ці алгоритми ефективно вирішують задачі оптимізації за

допомогою механізмів селекції, кросинговеру, мутації та схеми роботи, яка імітує природний відбір. Генетичні алгоритми демонструють високу гнучкість і можуть адаптуватися до змінних умов, що робить їх ідеальними для застосування в комплексних системах маршрутизації. Його застосування має ряд переваг:

- не вимагає ніякої інформації про поверхню відповіді;
- стійкий до потрапляння до локальних оптимумів;
- добре працює при вирішенні великомасштабних проблем оптимізації;
- бути використаний для широкого класу задач;
- простий та прозорий в реалізації;
- може бути використаний в завданнях з мінливим середовищем [19, 20].

У рамках застосування генетичних алгоритмів (ГА), задача оптимізації трансформується в завдання визначення максимального або мінімального значення цільової функції $f(x_1, x_2, \ldots, x_n)$. Ця функція відома як функція пристосованості (fitness function), і ключовим її атрибутом є здатність приймати лише ненегативні значення в своїй області визначення, що забезпечує позитивну пристосованість для кожної особини в популяції. У загальному випадку, кожен параметр функції пристосованості представлений через бітовий рядок. Особина у контексті ГА визначається як бітовий рядок, що є результатом конкатенації бітових рядків, які представляють упорядкований набір параметрів:

1011 10010 101 … 1001

| x1 | x2 | x3 | … | xn |

Унікальність генетичних алгоритмів (ГА) полягає в їх універсальності: специфіка конкретної оптимізаційної задачі впливає лише на певні аспекти, а саме на функцію пристосованості та спосіб кодування рішень. Всі інші етапи виконання генетичного алгоритму залишаються стандартними та однаковими для різних задач. Це означає, що процедура відбору, кросинговеру, мутації та інші процеси

ГА не змінюються від задачі до задачі, забезпечуючи широкий спектр застосувань цих алгоритмів у різних сферах оптимізації.

Схема роботи генетичного алгоритму показана на рисунку 2.7.

Кожна ітерація генетичного алгоритму структурована на трьох ключових стадіях:

1) Генерація проміжної популяції: На цій стадії відбувається відбір особин з поточного покоління, які увійдуть до проміжної популяції. Відбір базується на принципі пристосованості, де особини з вищою пристосованістю мають більшу ймовірність бути вибраними.
2) Схрещування (кросинговер) особин проміжної популяції: Ця стадія передбачає обмін генетичним матеріалом між особинами проміжної популяції, що призводить до створення нового покоління. Кросинговер є ключовим механізмом для забезпечення генетичного різноманіття в популяції.
3) Мутація нового покоління: На цьому етапі відбувається випадкова модифікація генів у особин нового покоління, що сприяє додатковому генетичному різноманіттю і може допомогти уникнути локальних оптимумів у процесі пошуку глобального оптимального рішення.

Проміжна популяція являє собою групу особин, відібраних для репродукції. В цій популяції особини з вищою пристосованістю мають збільшений шанс бути представленими, у той час як менш пристосовані особини мають нижчу ймовірність потрапляння до неї.

У класичних генетичних алгоритмах (ГА) процес селекції визначає ймовірність вибору кожної особини для включення у проміжну популяцію на основі її пристосованості. Цей процес часто реалізується за допомогою механізму, відомого як "Колесо рулетки"[22].

Метод "Колесо рулетки" передбачає розбивку колеса на сектори, пропорційні пристосованості кожної особини в популяції. Кожен член популяції отримує свій сектор на колесі. Чим вища пристосованість особини, тим більший сектор вона займає на колесі, тим вища ймовірність її вибору.

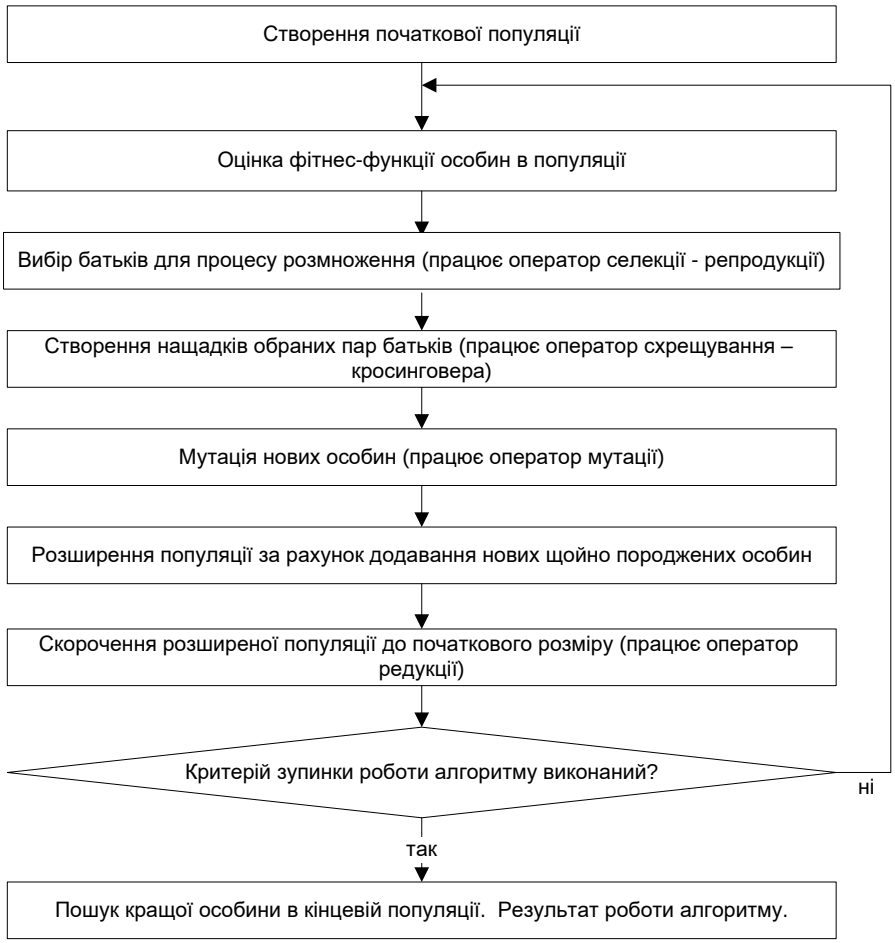


Рисунок 2.7 – Схема роботи генетичного алгоритму

Процес відбору відбувається шляхом "запуску" рулетки n разів, де кожен запуск визначає, яка особина буде вибрана для участі в наступних етапах алгоритму. Цей механізм забезпечує, що особини з вищою пристосованістю мають більше шансів на репродукцію, в той час як менш пристосовані особини мають менші шанси бути вибраними.

Розмір і-го сектору пропорційний величині $P_{sel}(i)$, що розраховується за формулою:

$$P_{sel}(i) = \frac{f(i)}{\sum_{i=1}^{n} f\{i\}} \tag{2.7}$$

Під час процесу відбору в генетичних алгоритмах, застосовуючи механізм "Колеса рулетки", особини з вищою пристосованістю мають більшу ймовірність бути вибраними у порівнянні з тими, чия пристосованість нижча. Це відбувається завдяки тому, що розмір сектора колеса рулетки для кожної особини є пропорційним її рівню пристосованості: чим вище пристосованість, тим більший сектор і, відповідно, вищі шанси на вибір цієї особини. Таким чином, цей метод селекції сприяє вибору найбільш адаптованих особин для утворення наступного покоління, що сприяє посиленню загальної адаптивності популяції [19, 20].

Критерієм зупинки може виступати задана кількість поколінь або сходження популяції, що означає, що досягнуте рішення близьке до оптимального. Підсумковим рішенням задачі є пристосована особина останнього покоління.

Таким чином, для розробки генетичного алгоритму необхідно визначити поняття особини (вид кодування, довжина), популяції (розмірність), розробити оператори кросинговеру й мутації, задати цільову функцію.

2.5 Проектування та розробка алгоритму вирішення задачі маршрутизації

Розробка алгоритму маршрутизації передбачає створення комплексної комп’ютерної системи, що включає збір та обробку великих обсягів даних, аналіз маршрутів та оптимізацію роботи транспорту. Це вимагає від системи здатності інтегруватися з іншими корпоративними системами, гарантування безпеки даних

та підтримки масштабованості для адаптації до розширення або зменшення обсягів роботи.

Нижче наведений перелік вхідної інформації, яка необхідна для функціонування алгоритму:

– файл, у якому зберігається матриця відстаней ($C[n,n]$) між клієнтами, де n – кількість клієнтів у системі;

– введені за допомогою програми обліку списки автомобілів, клієнтів, товарів;

– створені заявки, які необхідно обслужити;

– дата, на яку необхідно розвести товари.

Важливою умовою є відповідність номера рядка й стовпця клієнта його ідентифікатору в базі даних програми обліку.

На першому етапі вирішення задачі необхідно розбити всю множину клієнтів, що обслуговуються в цей день, на регіони, які будуть обслуговуватися одним автомобілем для кожного регіону. Здійснюється вибірка заявок на певну дату. Ця задача вирішується в такий спосіб:

1) Обирається автомобіль, визначається обсяг його кузова.

2) Відшукується клієнт з найбільшою відстанню від складу й до нього послідовно додаються найближчі з урахуванням росту групи. У матриці C шукається рядок, сума елементів якого максимальна. Відповідний їй пункт вноситься до групи. i_1 – координата рядка.

3) Відшукується найближчий до попереднього клієнт, що внесений до рішення – мінімальний елемент матриці A у стовпці i_1. Відповідний їй пункт вноситься до групи. i_2 – координата рядка.

4) Відшукується найближчий клієнт до того, що внесений до рішення – мінімальна сума елементів матриці C у стовпцях i_1 та i_2. Відповідний йому пункт вноситься до групи. i_3 – координата рядка.

При кожному додаванні заявки до маршруту відбувається накопичення значення завантаженості кузова. Коли на черговому кроці загальний обсяг товарів для даного регіону перевищив припустимий обсяг кузова автомобіля, обирається наступна заявка й перевіряється, чи підходить її обсяг для даного автомобіля. Якщо додавання заявки веде до перевантаження кузова, то вона додається до списку заявок, не актуальних для даного маршруту, й обирається наступна заявка. Таким чином, відбувається перебір заявок, що залишилися. Коли всі заявки розглянуті, маршрут вважається сформованим, і кроки повторюються для нового регіону, поки не буде «розвезений» увесь товар.

На другому етапі є набір маршрутів, у кожному наборі втримуються заявки, які потрібно впорядкувати так, щоб об'їзд клієнтів був оптимальним.

Для розрахунків витрат створюється матриця переходів, що містить факт переходу від одного клієнта до іншого. Метою вирішення є знаходження маршруту, що має мінімальну довжину.

При використанні ГА для вирішення даного завдання необхідно:

1) Обрати або розробити форму кодування хромосом.
2) Задати цільову функцію, за допомогою якої можна буде оцінити оптимальність поточного рішення.
3) Розробити оператори кросинговеру та мутації.
4) Визначити основні параметри ГА (потужність популяції, імовірності кросинговеру й мутації).

Після цього необхідно розробити відповідні методи для виконання етапів генетичного алгоритму:

1) Генерація вихідної популяції.
2) Відбір хромосом у проміжну популяцію.
3) Спосіб редукції для нової популяції.
4) Критерій зупинки.

На початковому етапі випадково формуються хромосоми початкової популяції. Але при цьому повинен бути дотриманий ряд умов:

– Значення генів хромосом повинні належати множині клієнтів кластера.

– Значення генів не повинні повторюватися.

– На першу й останню позицію записується номер, який означає склад підприємства (було ухвалене рішення вважати це точкою *0*).

– Довжина хромосоми дорівнює кількості клієнтів у кластері + 2 (із врахуванням початкової й кінцевої точок складу).

У якості фітнес функції обрана довжина маршруту. До неї передається хромосома. Сума переходів від одного гена (клієнт) до іншого відповідно матриці відстаней накопичується і є значенням фітнес функції (2.8).

$$F(P_i) = \sum C[P_{i[j]}, P_{i[j+1]}], \qquad (2.8)$$

де C – матриця відстаней,

$P_{i[j]}$ – j-й ген і-ї хромосоми популяції,

ps – розмір популяції. j = 1 .. N, i = 1 .. ps.

Для реалізації кросинговеру був обраний циклічний варіант реалізації кросинговеру, що забезпечить зручне використання шляхового методу кодування й виключає колізії.

Нехай є наступні батьківські хромосоми $ch_{1,1}$ = {1,2,3,4,5,6,7,8,9,10,1} та $ch_{1,2}$ = {1,4,6,2,9,8,7,3,10,5,1}. Формування нащадка $ch_{2,1}$={1,x,x,x,x,x,x,x,x,x,1} йде кроками. Спочатку в саму ліву незайняту позицію ставиться елемент із самої лівої позиції батька $ch_{1,1}$, який не викликав би конфлікт із вже проставленими в $ch_{2,1}$ елементами. Потім у саму праву незайняту позицію ставиться елемент із самої правої позиції батька $ch_{1,2}$, який не викликав би конфлікт із вже проставленими в $ch_{2,1}$ елементами. Так триває доти, доки не будуть зайняті всі позиції в $ch_{2,1}$ (обидва цикли зустрінуться посередині). Отримано: $ch_{2,1}$ = {1,2,3,4,6,9,8,7,10,5,1}. Формування нащадка $ch_{2,2}$ = {1,x,x,x,x,x,x,x,x,x,1} йде кроками. Спочатку в саму

ліву незайняту позицію ставиться елемент із самої лівої позиції батька $ch_{1,2}$, який не викликав би конфлікт із вже проставленими в $ch_{2,2}$ елементами. Потім у саму праву незайняту позицію ставиться елемент із самої правої позиції батька $ch_{1,1}$, який не викликав би конфлікт із вже проставленими в $ch_{2,2}$ елементами. Так триває доти, доки не будуть зайняті всі позиції в $ch_{2,2}$ (обидва цикли зустрінуться посередині). Отримано: $ch_{2,2}$ = {1,4,6,2,7,3,5,8,9,10,1} [19].

Якість функціонування ГА також сильно залежить від ефективного застосування оператора мутації. В роботі буде використовуватися модифікований оператор мутації, при якому відбувається обмін значеннями між двома випадково обраними елементами хромосоми.

У результаті розробки алгоритму запропоновано використовувати середній розмір популяції (80-100 шт.), щоб при великому обсязі роботи не було витрачено неприпустимо багато часу. Як було зазначено вище, початкова популяція формується випадково, дотримуючись описаних обмежень. Довжина початкової популяції задається перед запуском алгоритму й прийнята рівною значенню 100.

Далі відбувається селекція хромосом батьків у проміжну популяцію. Для цього за формулою (2.9) розраховується кількість разів попадання хромосоми до тимчасової популяції.

$$P(i) = \frac{f(i)}{Avg} \tag{2.9}$$

Вона являє собою відношення функції приналежності конкретної особини до середнього значення функції приналежності всіх особин даної популяції.

Після проведення кросинговеру та мутації кількість хромосом у тимчасовій популяції збільшується. Необхідно зменшити цю кількість до заданого розміру популяції. Для цього всі хромосоми сортуються за значенням фітнес функції й особини, що потрапили за межі встановленого розміру популяції, відкидаються.

Інший важливий момент – визначення критеріїв зупинки. Звичайно для цього застосовуються обмеження на максимальне число епох функціонування алгоритму або визначення його збіжності, звичайно шляхом порівняння пристосованості популяції на декількох епохах і зупинки під час стабілізації цього параметра. Саме такий спосіб був застосований у даному алгоритмі. За замовчуванням було задано максимальне число поколінь, яке дорівнює 1000, а значення числа епох, при яких результат не поліпшується – 250.

На рисунку 2.12 представлена блок-схема функціонування алгоритму маршрутизації.

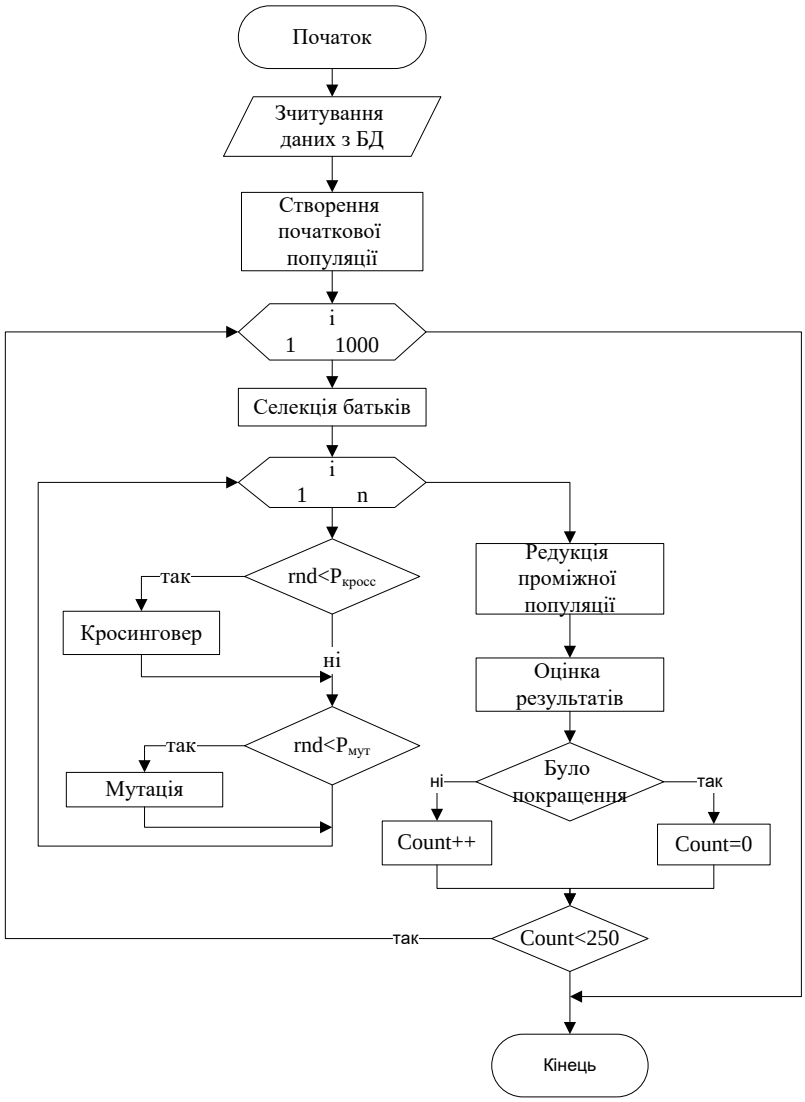


Рисунок 2.12 – Блок-схема алгоритму маршрутизації

2.6 Висновки

У результаті виконання даного розділу було описано математичну постановку задачі, а так само розглянуто методи й алгоритми, за допомогою яких вже було вирішено задачу маршрутизації транспорту. Було ухвалено рішення використовувати генетичні алгоритми для реалізації основної частини системи, тому що вони мають ряд переваг, як наприклад простота реалізації, відносно висока швидкість роботи, паралельний пошук рішення відразу декількома особинами, що дозволяє уникнути потрапляння до «пастки» локальних оптимумів.

В якості критерію оптимальності прийнято довжину маршруту. Було сформульовано обмеження до створення хромосом, розроблено відповідні оператори кросинговеру й мутації, які не викликають колізій. Визначено правила відбору хромосом і редукції. Розроблено схему алгоритму.

3 ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ СИСТЕМИ ВИЗНАЧЕННЯ МАРШРУТІВ

3.1 Вимоги до програмного забезпечення

Ефективне програмне забезпечення для систем маршрутизації повинно бути гнучким і масштабованим, забезпечувати високий рівень безпеки даних та мати можливість інтеграції з іншими корпоративними системами. Це включає забезпечення надійного зберігання, обробки та передачі інформації, а також захист від несанкціонованого доступу.

При проектуванні та розробці розподіленої системи логістики виникає ряд ключових вимог до програмного забезпечення:

- Масштабованість та гнучкість: Система має підтримувати зміну кількості (збільшення або зменшення) користувачів, маршрутів, транспортних засобів, а також вміти адаптуватися до змінних умов роботи без значних втрат продуктивності. В рамках проекту масштабованість також передбачає просте підключення фізичного робочого місця, легку інсталяцію необхідних для повноцінного функціонування компонентів

- Безпека даних: Необхідно забезпечити надійне зберігання, обробку і передачу інформації, а також захист від несанкціонованого доступу. Це означає, що потрібно визначити всіх користувачів системи, їх роль та права доступу. З точки зору фізичної безпеки, треба визначитися з апаратним забезпеченням серверної частини та користувацької.

- Інтеграція з іншими системами: Система має мати можливість інтеграції з іншими системами, наприклад, з ERP, CRM, системами складського обліку тощо.

- Реалізація алгоритмів оптимізації маршрутів: Система повинна мати вбудовані алгоритми для автоматичного визначення найбільш ефективних маршрутів в залежності від вхідних параметрів, що надає користувач системи.

- Мультиплатформеність: Система має працювати на різних платформах (персональний комп'ютер, мобільні пристрої, планшети) і мати інтуїтивний інтерфейс для різних типів користувачів.

- Ведення log-файлів та контроль активності користувачів в системі: Необхідність зберігання історії дій користувачів, змін маршрутів, транзакцій, для можливості аналізу та контролю.

- Розширені можливості звітності: Система повинна мати засоби для генерації звітів з різних аспектів логістичної діяльності, як-от ефективність маршрутів, витрати на пальне, продуктивність працівників тощо.

Звичайно, існує ще велика кількість вимог до розробки програмного забезпечення, але такий набір вимог є одночасно необхідним та достатнім для реалізації поставленої задачі, оскільки основними функціональними вимогами до програмного забезпечення системи є наступні:

1) Ведення обліку довідникової інформації (додавання, редагування, видалення).
2) Додавання заявок від клієнтів.
3) Задавання параметрів роботи алгоритму.
4) Обчислення маршрутів доставки лікарських засобів.
5) Візуальне відображення ходу пошуку рішення і результатів.

Ці основні функціональні можливості створюють фундамент системи, на якому може бути побудовано майбутнє її розширення та модифікація. Маючи на увазі динамічний характер таких вимог, можна розглядати ряд мов програмування та СУБД.

Мови програмування:

- Java: Завдяки масштабованості, портативності (завдяки JVM) та багатьом фреймворкам (Spring Boot, Jakarta EE) Java є популярним вибором для розподілених систем.

- Python: Ця мова зручна для швидкої розробки та інтеграції з різноманітними системами. Фреймворки, як-от Django або Flask, можуть бути корисними для веб-розробки.

- C# (.NET Core): Платформа .NET Core підтримує кросплатформеність і має багатий набір бібліотек.

- Go: Завдяки високій продуктивності та простоті, Go стає все популярнішим для розробки великих розподілених систем.

- Node.js: Для асинхронних задач та веб-сервісів на базі JavaScript.

Системи управління базами даних (СУБД):

Relational (Реляційні):

- PostgreSQL: Відкрита система з високою надійністю, розширюваністю та підтримкою JSON.

- MySQL: Широко використовується, підтримує велику кількість операцій та має добру репутацію з надійності.

- Microsoft SQL Server: Добре інтегрується з іншими продуктами Microsoft.

NoSQL:

- MongoDB: Документоорієнтована СУБД, хорошо підходить для неструктурованих даних.

- Cassandra: Розподілена, високодоступна СУБД, підходить для великих масштабів.

- Redis: Високошвидкісна система для роботи з даними у пам'яті, часто використовується як кеш.

NewSQL (сучасні реляційні системи для великих масштабів):

- CockroachDB: Розподілена СУБД з підтримкою ACID-транзакцій.

- Google Spanner: Глобально розподілена СУБД від Google.

Враховуючи, що розробка спрямована, перш за все, на розробку десктопного додатку, під операційну систему Windows, логічним є використання продуктів Microsoft, а саме – в якості середовища програмної розробки – Visual Studio, мова програмування C#. В якості СУБД – MS SQL Server.

Інтеграція: C# та MS SQL Server є продуктами компанії Microsoft, тому між ними є відмінна інтеграція. Завдяки цьому розробники можуть ефективно використовувати інструменти і можливості, надані платформою .NET, для роботи з базою даних.

Висока продуктивність: C# є сучасною, об'єктно-орієнтованою мовою програмування, яка дозволяє швидко розробляти надійні і безпечні додатки. MS SQL Server — це потужна система управління базами даних, яка може легко масштабуватися і забезпечує високий рівень продуктивності і безпеки даних.

Безпека: Microsoft постійно оновлює свої продукти, щоб вони відповідали найсучаснішим стандартам безпеки. Інтегровані засоби безпеки в MS SQL Server дозволяють ефективно управляти доступом до даних, шифруванням, аудитом та іншими аспектами безпеки.

Масштабованість: MS SQL Server дозволяє легко масштабувати додатки, від невеликих локальних баз даних до великих корпоративних рішень в хмарі.

Спільнота та підтримка: І C#, і MS SQL Server мають велику активну спільноту розробників, що означає доступ до великої кількості ресурсів, таких як бібліотеки, інструменти, форуми та документація. Крім того, комерційна підтримка від Microsoft також є дуже сильним аргументом в користь цих технологій.

Гнучкість: Завдяки платформі .NET ви отримуєте доступ до широкого спектру бібліотек і засобів для різних задач — від веб-розробки до роботи з машинним навчанням.

Вибір C# та інших продуктів Microsoft для розробки програмного забезпечення систем маршрутизації виявився обґрунтованим з декількох причин. По-перше, C# як мова програмування, що розвивається під егідою Microsoft, забезпечує високий рівень інтеграції з Windows, найбільш поширеною операційною системою в бізнес-середовищі. Це забезпечує легкість у розробці, тестуванні та розгортанні програмних продуктів.

По-друге, C# відомий своєю потужною та гнучкою екосистемою, яка включає в себе різноманітні бібліотеки та фреймворки, такі як .NET Core, що підвищують продуктивність розробки та забезпечують широкі можливості для реалізації складних функціональних вимог.

Також важливою перевагою є стабільність та безпека продуктів Microsoft, що є ключовими факторами для бізнес-додатків. Microsoft надає постійну підтримку та оновлення для своїх продуктів, забезпечуючи захист від потенційних загроз безпеці та відповідність сучасним стандартам.

Використання C# та MS SQL Server, спеціально розроблених для ефективної роботи в екосистемі Microsoft, дозволяє досягти високої продуктивності, масштабованості та надійності. Це особливо актуально для бізнес-додатків, де важлива швидкість обробки даних, стабільність роботи та можливість легко інтегруватися з іншими бізнес-процесами та системами.

У підсумку, вибір C# та продуктів Microsoft для розробки програмного забезпечення систем маршрутизації є доцільним із точки зору інтеграції, безпеки, підтримки, продуктивності та гнучкості, що є вирішальними факторами для успішної реалізації та експлуатації такого роду систем.

3.2 Об’єктна модель системи

Для подальшої розробки програмного забезпечення було розроблено об’єктну модель системи.

Розглянемо детальніше основні складові системи визначення маршрутів перевезення замовлень:

- підсистема приймання вхідних даних;
- підсистема обліку даних;
- підсистема обчислювання інформації;
- підсистема виводу.

Об'єктна модель призначена для опису структури об'єктів, що становлять систему, а так само їх взаємозв'язок один з одним, уточняються їх атрибути і методи.

Об'єкт "Матриця доріг" наповнюється з підсистеми ГІС, далі з використанням цієї інформації "АСУ" робить розподіл "Заявок" по "Автомобілях", заповнюючи таким чином "Маршрутний лист" для кожного автомобіля. Існує база товарів, яка описується об'єктом "Товар", - він є частиною заявки, поданої "Клієнтом". Кінцевий вид об'єктної моделі представлений на рисунку 3.1.

В загальному вигляді робота системи описується наступним чином: кожний клієнт формує заявку на постачання певних товарів. Ці дані надсилаються постачальникові, цифруються (або одразу вносяться в систему клієнтом у вигляді електронної заявки). Система перевіряє наявність товарів, наявність автомобілів та формує маршрутний лист.

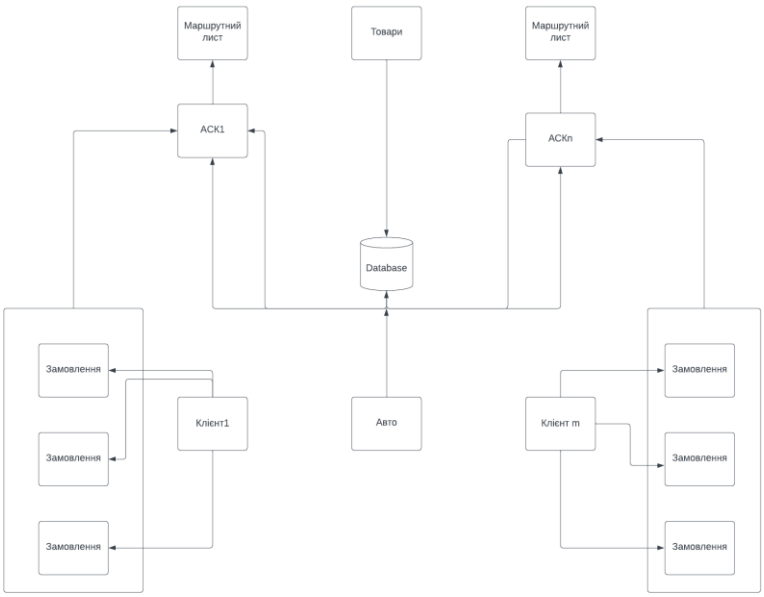


Рисунок 3.1 – Об’єктна модель системи

3.3 Інформаційне забезпечення

Інформаційне забезпечення – це сукупність інформації про об'єкт управління, яке переробляється в комп'ютерній системі управління. На відміну від об'єкту управління, який вже існує, інформаційне забезпечення треба представляти і враховувати розробникові при створенні комп'ютерної системи.

Для зберігання інформації зручно використати систему управління базами даних (СУБД). Обираючи MS SQL Server як СУБД для зберігання інформації, необхідної для виконання алгоритму оптимізації в програмному забезпеченні систем маршрутизації, було зроблено зважений вибір, враховуючи ключові переваги цієї СУБД.

MS SQL Server, будучи продуктом Microsoft, забезпечує високу продуктивність, масштабованість і безпеку, що є критично важливим для бізнес-додатків, особливо тих, які залежать від ефективного управління великими обсягами даних та складних запитів.

Крім того, MS SQL Server пропонує вдосконалені можливості управління даними та аналітики, що дозволяє використовувати більш складні алгоритми для оптимізації маршрутів. Ця СУБД також підтримує інтеграцію з іншими продуктами Microsoft та забезпечує легшість у розробці та управлінні додатками, що використовують C# та інші технології Microsoft.

Таким чином, враховуючи необхідність забезпечення високої продуктивності, надійності та масштабованості системи, MS SQL Server виявився оптимальним вибором для розробки програмного забезпечення систем маршрутизації. Його використання гарантує ефективну роботу з даними, що є ключовим для успішного виконання алгоритмів оптимізації маршрутів та забезпечення надійної логістики.

Зберігання інформації, необхідної для виконання алгоритму оптимізації, реалізоване за допомогою Для забезпечення функціонування програмного забезпечення було створено 8 таблиць.

- Перелік товарів (код, найменування, строк придатності, форма випуску, номер замовлення)
- Перелік населених пунктів (назва, тип населеного пункту, область)
- Замовлення (Клієнт, номер та дата замовлення, місто, ПІБ замовника)
- Перелік товарів в замовленні (номер замовлення, перелік товарів, кількість товарів, ціна)
- Перелік автомобілів (тип автомобіля, номер, марка, вмістимість)
- Клієнт (назва, форма власності, ПІБ керівника, адреса, контактний телефон)
- Маршрут (номер маршруту, водій, авто)
- Маршрутний лист (номер маршруту, номер заявки, перелік міст)

Розглянемо детально структуру кожної таблиці.

Таблиця 3.1 Структура таблиці «avto»

№	Поле	Тип	Розмір	Підпис
1	*id	лічильник	4	Ключ таблиці
2	number	Текстовий	8	Номер автомобіля
3	capacity	числовий	4	Об’єм кузову
4	owner	текстовий	255	Назва власника

Таблиця 3.2 Структура таблиці «tovar»

№	Поле	Тип	Розмір	Підпис
1	*id	лічильник	4	Ключ таблиці
2	title	Текстовий	255	Назва товару
3	code	Текстовий	10	Шифр
4	capacity	Числовий	4	Об’єм

Таблиця 3.3 Структура таблиці «klient»

№	Поле	Тип	Розмір	Підпис
1	*id	лічильник	4	Ключ таблиці
2	title	Текстовий	255	Назва
3	address	Текстовий	255	Адреса клієнта

Таблиця 3.4 Структура таблиці «marshrut»

№	Поле	Тип	Розмір	Підпис
1	*id	лічильник	4	Ключ таблиці
2	id_avto	Числовий	4	Автомобіль, що виконує розвезення
3	mdate	Дата	8	Дата розвозу

Таблиця 3.5 Структура таблиці «zayavka»

№	Поле	Тип	Розмір	Підпис
1	*id	лічильник	4	Ключ таблиці
2	id_klient	Числовий	4	Клієнт
3	zdate	Дата	8	Дата заявки
4	capacity	Числовий	4	Об’єм товарів

Таблиця 3.6 Структура таблиці «marshrut_list»

№	Поле	Тип	Розмір	Підпис
1	*id	лічильник	4	Ключ таблиці
2	id_marshrut	Числовий	4	Маршрут
3	id_zayavka	Числовий	4	Заявка

Таблиця 3.7 Структура таблиці «zayavka_list»

№	Поле	Тип	Розмір	Підпис
1	*id	лічильник	4	Ключ таблиці
2	id_zayavka	Числовий	4	Заявка
3	id_tovar	Числовий	4	Товар
4	count	Числовий	2	Кількість

Таблиця 3.8 Структура таблиці «nas_punkt»

№	Поле	Тип	Розмір	Підпис
1	*id	лічильник	4	Ключ таблиці
2	title	Числовий	255	Назва міста
3	Type_nas	текстовий	255	Тип міста
4	region	Текстовий	255	Область

3.4 Опис модулів і об'єктів системи

Під час створення програмного забезпечення було розроблено модуль введення даних, модуль обліку інформації та модуль обчислювання, який складається з 4х основних класів. Перелік цих класів наведено у таблиці 3.9.

Таблиця 3.9 Опис класів модулю обчислювання

№	Назва класу	Опис
1	ReadData.cs	Забезпечує зчитування вхідних даних із файлу та БД «Склад»
2	Route.cs	Вирішує задачу кластеризації поданих заявок
3	Chromosome.cs	Являє собою опис хромосоми генетичного алгоритму
4	Population.cs	Являє собою опис популяції генетичного алгоритму

Розглянемо більш детально структуру класів модулю обчислювання.

1) ReadData.cs

Атрибути класу:

- double[,] Coordinate – декартові координати клієнтів для відображення на формі;
- double[,] Distance – матриця відстаней між клієнтами
- int[] numbers – номера кліжнтів, що задіяні на обрану дату у розвозі;
- double[] capacities – сумарний об’єм товарів у заявці кожного клієнта;
- List<double> truck_capacities – об’єм кузову транспортних засобів;
- List<int> trucks – номера транспортних засобів;
- int size – кількість клієнтів, що задіяні у розвозі;
- String CommandText, ConnectionString – для підключення до БД.

Методи класу:

- ReadData – конструктор класу, читає з файлу координати клієнтів, читає дані про заявки, автомобілі з БД;
- InitDistance – ствоює матрицю відстаней на основі координат клієнтів;
- double[] GetCapacities – вертає масив об’ємів заявок;
- double[] GetCoordinate – вертає матрицю координат;
- double[] GetDistance – вертає матрицю відстаней;
- int[] GetNumbers – вертає номера клієнтів;
- int GetSize – вертає кількість клієнтів;
- List<double> GetTruckCapacities – вертає об’єми кузовів автомобілів;
- List<double> GetTrucks – вертає номера автомобілів.

2) Route.cs

Атрибути класу:

- int n – кількість клієнтів;
- int k – кількість маршрутів;
- double[,] Distance – матриця відстаней;
- double[] capacities – об’єм кожної заявки;
- List<double> truck_cap - об’єм кузовів автомобілів.

Методи класу:

- Route – ініціалізація параметрів;
- InitDistance – ініціалізація матриці відстаней;
- CopyMatr – метод копіювання матриць;
- GetClusters – виконує задачу кластерізації: поділяє множину клієнтів на регіони з урахуванням загрузки автомобілів.

3) Chromosome.cs

Атрибути класу:

- List<int> route – маршрут, що складається з номерів клієнтів

Методи класу:

- Chromosome – ініціалізація параметрів;
- AddBit – додає одну аптеку до хромосоми перевіряючи, що вона не була додана раніше;
- GetBit – повертає номер аптеки по заданому індексу;
- SetBit – встановлює номер клієнта по заданому індексу, перевіряючи, що його не було додано раніше;
- Equals – перевіряє рівність між двома хромосомами;
- GetLength – повертає розмір хромосоми.

4) Population.cs

Атрібути класу:

- List<int> Cluster – номера клієнтів, що задіяні у даному кластері;
- List<Chromosome> population – початковий набір хромосом;
- Chromosome Best – краща хромосома;
- double[,] Distance – матриця відстаней;
- int dist_size – розмір матриці відстаней;
- int n – розмір хромосоми;
- int pop_size – розмір популяції;
- int MaxIters – максимальна кількість ітерацій;
- int MaxEps – максимальна кількість епох, коли хромосома не покращується;
- List<double> Statistics – дані для створення графіків.

Методи класу:

- Population – ініціалізація параметрів;
- GeneratePopulation – створення первинної популяції;
- Roulette – селекція батьківських особин;

- Crossingover – реалізує кросинговер;
- Mutation – реалізує мутацію;
- Reduction – реалізує редукцію;
- Run – запускає алгоритм пошуку оптимального маршруту;
- GetDistance – повертає довжину маршруту прийнятої хромосоми;
- SortPopulation – сортує популяцію;
- GetBestChromosome – повертає найкращу хромосому;
- SetCross – встановлює вірогідність кросинговера;
- SetMut – встановлює вірогідність мутації;
- GetStatistics – вертає дані для створення графіків.

3.5 Інтерфейс програмного забезпечення

Головна форма розділена на 5 частин. Вид головної форми програмного рішення представлено на рисунку 3.4. У верхній частині знаходяться поля для вводу параметрів алгоритму та дати, на яку треба створити маршрути. У лівій частині знаходиться перелік довжин кожного з маршрутів. У центральній частині знаходиться графіки по кожному маршруту, що показують покращення відповіді з кожною ітерацією. Якщо у маршруті тільки одна заявка, то графік не виводиться. У правій частині розташована мапа зі вказаними координатами аптек, так лініями маршрутів. Вони мають різний колір. Знизу відображується статус виконання програми, а у правому верхньому куту знаходиться кнопка «БД», що веде до підсистеми обліку. Кнопка «Запуск» запускає виконання алгоритму.

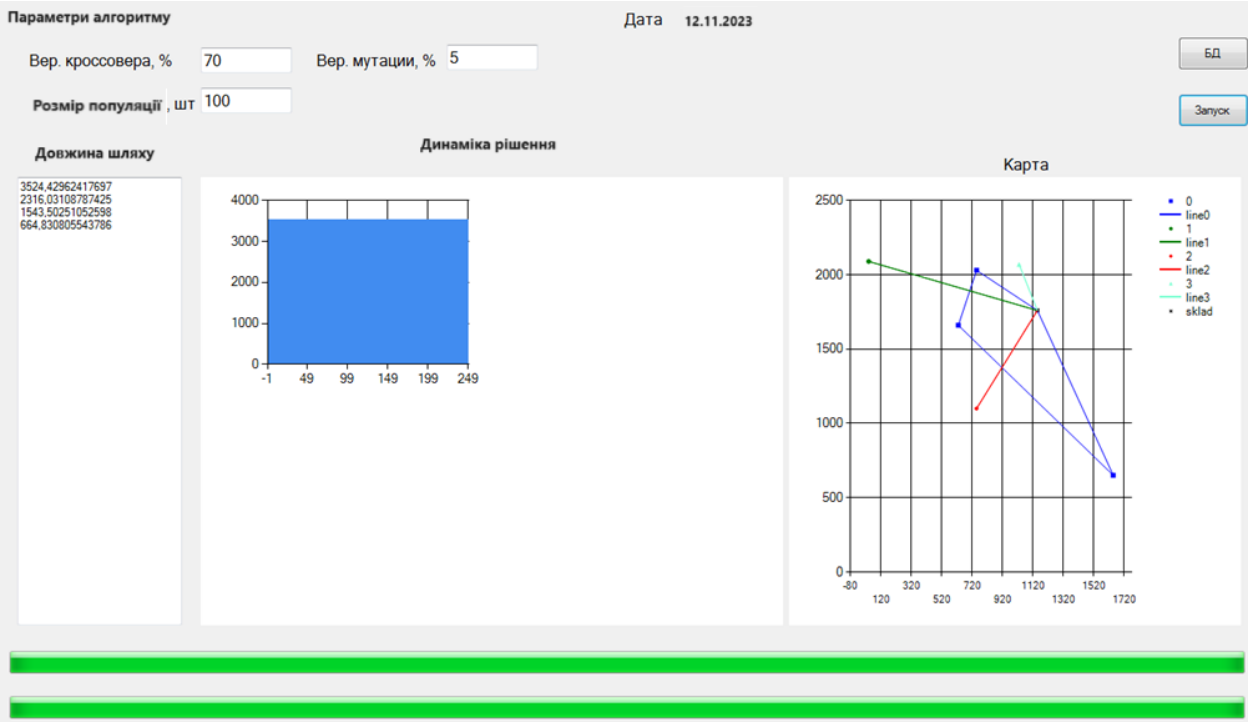


Рисунок 3.4 – Головна форма програмного рішення

На рисунку 3.5 зображено головну форму підсистеми обліку. На ній знаходяться кнопки, що відкривають форми редагування елементів підсистеми.

Рисунок 3.5 – Головна форма підсистеми обліку

Для реалізації підсистеми обліку було розроблено 4 екранних форми:

Форму роботи з довідниками «Товари», «Автомобілі» та «Клієнти» зображено на рисунку 3.6. На ній ми бачимо таблицю з даними, які можна редагувати безпосередньо у таблиці користуючись допомогою навігатору, що знаходиться у нижній частині форми.

Наименование	Шифр	Объем, м^3
Аспирин	НКА2134	0,0001
Анальгин	ОАА8414	0,00012
Цитрамон	ЕВР9234	5E-5
Абитаксел	ГТТ2394	0,00015
Абомин®	ТАЦ9232	10,001
Авамис	ТАУ9233	0,00015
Авандия™	ТАЩ8405	0,00012
Валеодикрамен	ГКВ3255	0,00013
Валериана	ННЕ8354	0,001
Валериана Форте	ОАА2323	0,00015
Валз	АРА9235	0,001
Валокормид	ОАА2384	0,00012
Вальсакор® НД160	ШГК2345	0,00015
Вазотенз®	ТТТ8833	0,001
Вакцина гриппозная инактивированная элюатн	ВВА3445	0,00012

Рисунок 3.6 – Форма роботи з довідниками

Форму для створення заявок представлено на рисунку 3.7. Додаючи заявку, необхідно обрати аптеку, що її створює, та дату виконання. Після цього треба натиснути кнопку «Прийняти» та у таблицю нижче ввести список товарів, та їх кількість.

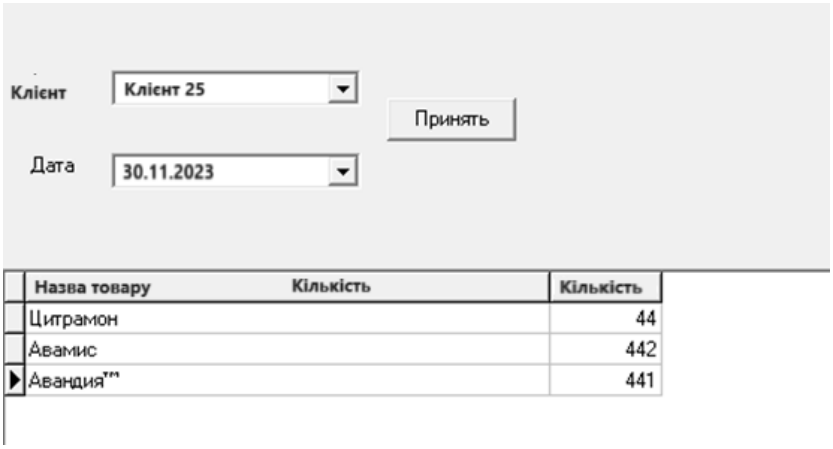
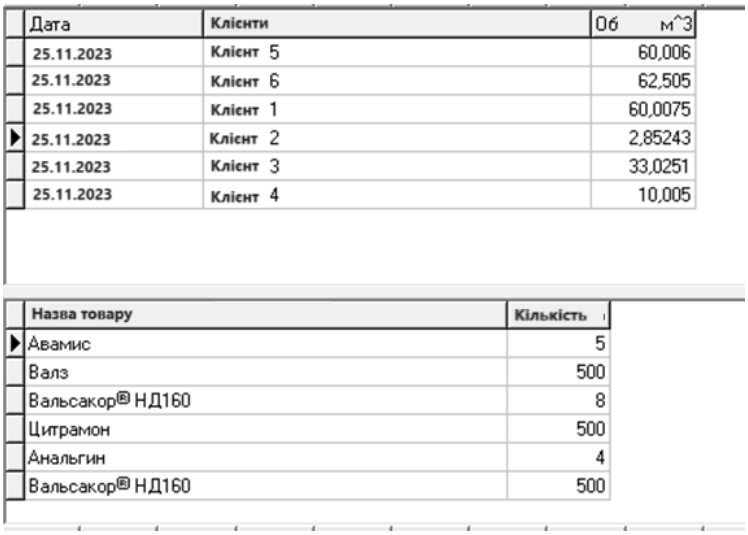


Рисунок 3.7 – Форма створення заявок

Для перегляду та редагування заявок було створено форму перегляду заявок, вона відображена на рисунку 3.8. Обравши заявку у верхній частині форми, у нижній буде відображено список товарів, що входять до її складу. Також усю інформацію можна редагувати безпосередньо у цих таблицях.

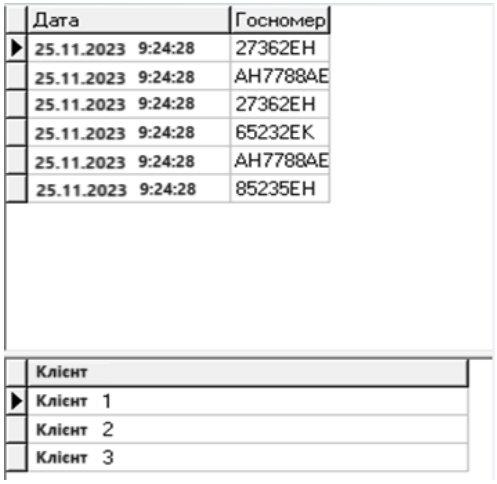


Дата	Клієнти	Об м^3
25.11.2023	Клієнт 5	60,006
25.11.2023	Клієнт 6	62,505
25.11.2023	Клієнт 1	60,0075
25.11.2023	Клієнт 2	2,85243
25.11.2023	Клієнт 3	33,0251
25.11.2023	Клієнт 4	10,005

Назва товару	Кількість
Авамис	5
Валз	500
Вальсакор® НД160	8
Цитрамон	500
Анальгин	4
Вальсакор® НД160	500

Рисунок 3.8 – Форма перегляду заявок

Форма перегляду сформованих маршрутів відображена на рисунку 3.9. Обравши маршрут у верхній частині форми, знизу буде відображено порядок об’їзду клієнтів.



Дата	Госномер
25.11.2023 9:24:28	27362EH
25.11.2023 9:24:28	AH7788AE
25.11.2023 9:24:28	27362EH
25.11.2023 9:24:28	65232EK
25.11.2023 9:24:28	AH7788AE
25.11.2023 9:24:28	85235EH

Клієнт
Клієнт 1
Клієнт 2
Клієнт 3

Рисунок 3.9 – Форма відображення маршрутів

Лістинг розробленого програмного забезпечення наведено у додатку Б.

Особливо важливим моментом є етап впровадження системи, що починається з встановлення (інсталяції) системи. На даний момент запропонована комп’ютеризована система являє собою десктопний клієнт-серверний додаток. На серверній частині встановлюється СУБД налаштовуються параметри сервера, включаючи порти, налаштування безпеки, резервне копіювання та відновлення. На клієнтській частині обирається каталог, в який буде встановлено додаток. Після цього в каталог з серверу копіюються необхідні файли, такі як виконавчий файл програми, файл конфігурації, всі файли шаблонів звітів. Далі необхідно ввести параметри для підключення до сервера, включаючи IP-адресу або ім'я хоста, порт, ім'я користувача та пароль. Після встановлення треба перевірити для всіх користувачів працездатність виданих логінів та паролей для входу в систему. Ролі користувачів в системи наразі визначені наступним чином:

- Адміністратор – має повний доступ до всієї функціональності розробленої системи
- Менеджер – користувач, що має доступ до функціональності, яка відповідає обробці замовлень та складання дорожних карт
- Працівник складу – користувач, що має доступ до функціональності, яка відповідає за облік товарів, що зберігаються на складі (як головний склад, так і допоміжні склади)
- Менеджер автопарку – користувач, що відповідає за ведення інформації про парк автомобілів, що займаються розвезенням замовлень.
- Менеджер по підбору персоналу – користувач, який співпрацює з системою обліку кадрів.

- Звичайний користувач – такий користувач необхідний в системи лише для перегляду стану підприємства: кількість замовлень, актуальні карти, стан замовлень, рівень завантаженості складів товарами, тощо.

Враховуючи той факт, що система є розподіленою, то клієнти системи можуть знаходитися як в одному приміщенні, так і в окремих містах або регіонах країни і мати віддалений доступ до головного серверу, що фізично знаходиться в приміщенні головного складу. Крім того, в системі передбачена безпека передачі та обробки даних за допомогою VPN (Virtual Private Network): VPN забезпечує безпечне підключення до сервера через інтернет, шифруючи дані та захищаючи їх від стороннього доступу. Також, додатковим елементом безпеки є мережевий Firewall.

Для доступу до системи використовуються механізми авторизації та аутентифікації для забезпечення того, що користувачі та системи мають доступ лише до необхідних ресурсів і функцій. Використання ролевого доступу для обмеження привілегій користувачів дозволяє надавати доступ лише до певної функціональності системи.

Для передбачення різноманітних ситуацій, коли система може вийти з ладу або з причин перепаду/зникнення напруги, з причини несанкціонованого доступу в системі передбачено резервне копіювання та відновлення: В автоматичному режимі регулярно створюються резервні копії даних та передбачено налаштування процедури відновлення системи в разі виходу її з ладу.

3.6 Висновки

В цьому розділі було виконано наступні задачі:

1) Розглянуто загальні вимоги до програмного забезпечення.

2) Розглянуто існуючі операційні рішення і інструментальні засоби реалізації. Обрано наступні рішення:
   - операційна система сімейства Windows як найпоширеніша операційна система в Україні;
   - обрано СУБД SQL Server 2019, з оновленнями версій 15.0.2104.1 та 15.0.4335.1 в жовтні 2023 року.
   - інструментальним засобом - Visual Studio 2022;
   - мова програмування – C# на платформі .NET 4.0.
3) Розглянуто схему взаємодії основних модулів системи. Представлено схему бази даних. Описано склад класів, що входять до програми.
4) Наведено та описано інтерфейс програмного рішення.

Тестування програмного забезпечення показало працездатність та виконання мінімальних заявлених вимог.

4 МОДЕЛЮВАННЯ Й АНАЛІЗ ЕКСПЕРИМЕНТАЛЬНИХ РЕЗУЛЬТАТІВ

4.1 Вхідні та вихідні дані

Для оцінки працездатності та ефективності запропонованих алгоритмів необхідно провести ряд експериментів за допомогою методів моделювання, проаналізувати отримані результати, надати пропозиції щодо значень вхідних даних для алгоритмів.

Експериментальне дослідження розроблених рішень проводилося за двома напрямками: для умов міста та для умов регіону. Для кожного напрямку було застосовано свій набір вхідних даних, що було отримано виходячи з заявок клієнтів на конкретну добу та узагальнено перераховано до величини об’єму, що буде зайнято відповідно до заявок. Набори вихідних даних наведено відповідно у табл. 4.1–4.2.

Таблиця 4.1 Набір вихідних даних для умов міста

Клієнт	Об'єм, $м^3$
Клієнт 1	6,00754
Клієнт 2	2,85243
Клієнт 3	2,00396
Клієнт 4	1,50497
Клієнт 5	6,006
Клієнт 6	5,505
Клієнт 7	0,0005
Клієнт 8	0,00755
Клієнт 9	0, 1152
Клієнт 10	4,03997
Клієнт 11	2,00416
Клієнт 12	0,0208

Таблиця 4.2 – Набір вихідних даних для умов регіону

Клієнт	Об'єм, $м^3$
Клієнт 1	8,1054
Клієнт 2	10,1543
Клієнт 3	3,7813
Клієнт 4	1,9047
Клієнт 5	7,006
Клієнт 6	9,505
Клієнт 7	3,0005
Клієнт 8	1,00755
Клієнт 9	3, 1152
Клієнт 10	0,3997
Клієнт 11	3,00416
Клієнт 12	4,0208
Клієнт 13	3,4825
Клієнт 14	3,3716
Клієнт 15	2,0504
Клієнт 16	8,9802
Клієнт 17	0,4112
Клієнт 18	6,3351
Клієнт 19	4,3789
Клієнт 20	1,4215

Також було обрано наступні параметри генетичного алгоритму: вірогідність мутації – 5%, вірогідність кросинговеру – 70%, об’єм популяції – 55 шт. Для прикладу обрано дату: 21.11.2023.

У результаті експериментальних досліджень з вихідними даними було отримано результати, що наведено у табл. 4.3 – 4.4. Для кожного автомобіля визначено відсортований список клієнтів, що являє собою маршрут руху транспортного засобу.

Таблиця 4.3 Набір результатів для умов міста

Автомобіль	Список клієнтів
00124523	Клієнт 9
	Клієнт 8
	Клієнт 4
	Клієнт 6
	Клієнт 12
	Клієнт 7
	Клієнт 2
00124510	Клієнт 10
	Клієнт 3
	Клієнт 11
00124511	Клієнт 1
00124513	Клієнт 5

Було створено чотири маршрути розвозу товарів протяжністю 4582, 3013, 1728, 1059 метрів відповідно. Візуальне відображення отриманих маршрутів без урахування вигинів доріг наведено на рисунку 4.1.

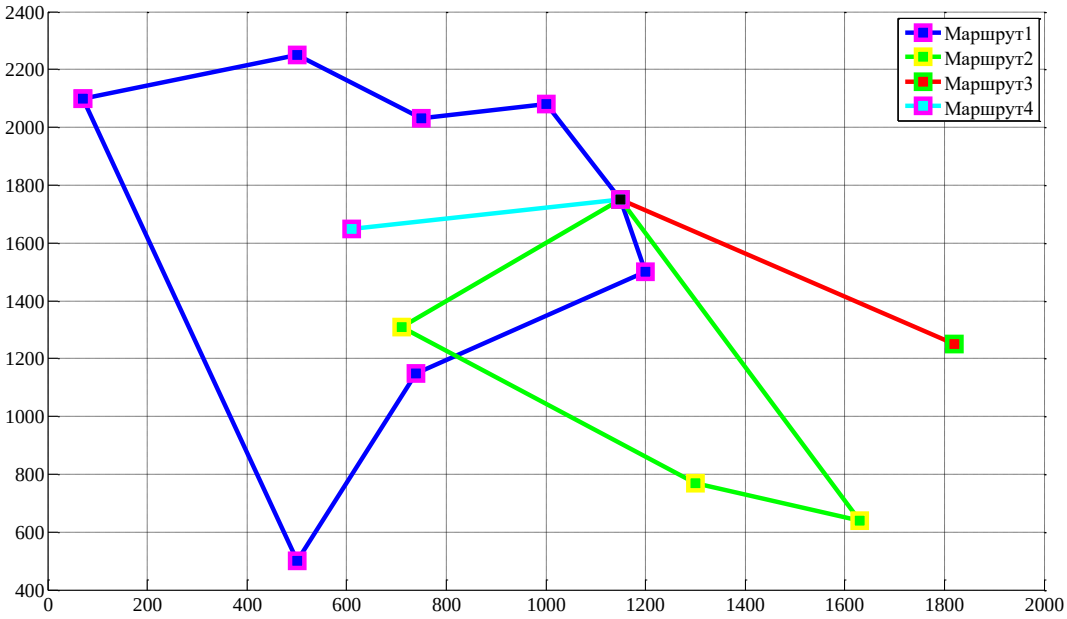


Рисунок 4.1 – Візуалізація маршрутів розвезення товарів

Таблиця 4.4 Набір результатів для умов регіону

Автомобіль	Список клієнтів
1	2
00112312	Клієнт 18
	Клієнт 12
	Клієнт 10
00114897	Клієнт 16
	Клієнт 4
	Клієнт 7
	Клієнт 13
	Клієнт 19
00114571	Клієнт 2
00112544	Клієнт 1
	Клієнт 15

Продовження таблиці 4.4

00112100	Клієнт 6
00113002	Клієнт 14
	Клієнт 11
	Клієнт 9
	Клієнт 17
00113410	Клієнт 3
	Клієнт 8
	Клієнт 5
	Клієнт 20

Було створено сім маршрутів розвозу товарів протяжністю 85, 128, 73, 92, 79, 105, 138 км відповідно. Візуальне відображення отриманих маршрутів без урахування вигинів доріг наведено на рисунку 4.2.

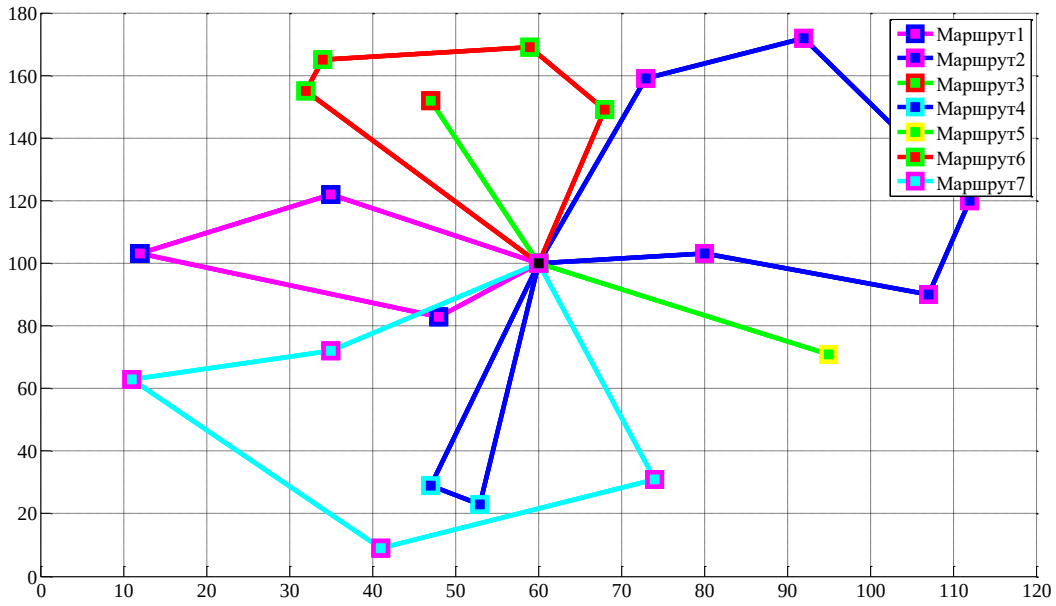


Рисунок 4.2 – Візуалізація маршрутів розвезення товарів для умов регіону

Візуальне відображення результатів відповідає отриманим чисельним даним.

4.2 Аналіз експериментальних результатів

Проаналізуємо результати, отримані у пункті 4.1 на прикладі умов міста. У зв’язку з тим, що у двох останніх маршрутах обслуговується лише по одному клієнту, очевидно, що генетичний алгоритм не було застосовано. На рисунку 4.3 зображено графік пошуку найкоротшої відстані між клієнтами для першого маршруту.

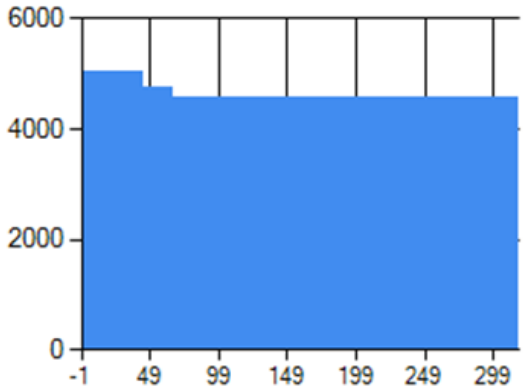


Рисунок 4.3 – Графік пошуку найкоротшої відстані між клієнтами першого маршруту

З рисунку 4.3 видно, що оптимальний результат було знайдено на 60-й ітерації, але згідно з пунктом 2.4 було пройдено ще 250 ітерацій, щоб упевнитися у оптимальності знайденого рішення.

На рисунку 4.4 зображено графік пошуку найкоротшої відстані між клієнтами для другого маршруту.

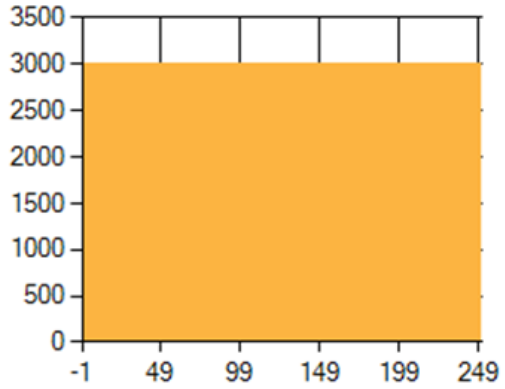


Рисунок 4.4 – Графік пошуку найкоротшої відстані між клієнтами другого маршруту

З наведеного графіку видно, що оптимальний результат було отримано у першій популяції та за 250 ітерацій його не було покращено. Такий результат було отримано тому що у маршрут входять три клієнти, а при розмірі популяції у 55 особин, оптимальний маршрут було сформовано випадковим чином.

Останні два маршрути мають по одному клієнту кожен, у зв’язку з чим результат є один і не може бути покращеним.

4.3 Аналіз параметрів алгоритму

У своїй роботі генетичний алгоритм використовує наступні основні параметри:

- $P_к$ – вірогідність кросинговеру;
- $P_м$ – вірогідність мутації;
- N – потужність популяції.

Також необхідними є кількість епох і критерій припинення роботи алгоритму, але безпосереднього впливу на процес пошуку рішення вони не мають. Для досягнення оптимальних за якістю результатів з прийнятними часовими і

обчислювальними витратами необхідно підібрати значення основних параметрів ГА.

Від вірогідності кросинговеру залежить кількість нових особин, які можуть з'явитися після селекції, отже – здатність популяції еволюціонувати, оскільки без нових хромосом процес пошуку зупиняється.

Для виявлення оптимальних значень параметрів алгоритму його було запущено для визначення довжини маршруту, що складається з 28 клієнтів. Для пошуку залежності результату від параметрів було проведено серію експериментів. Вірогідність кросинговеру починаючи з 0% поступово збільшувалось на 5% до досягнення 100% при потужності популяції 100 особин, та вірогідності мутації 5%. Результат зображено на рисунку 4.5.

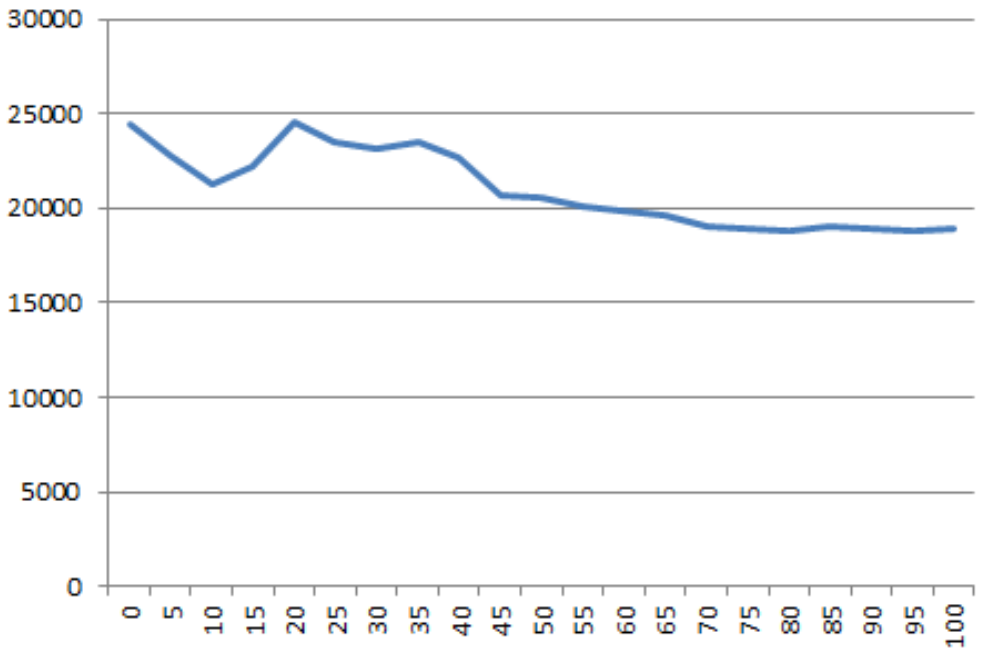


Рисунок 4.5 – Залежність довжини маршруту від вірогідності кросинговеру

З наведеного графіку можна побачити, що довжина маршруту зменшується з ростом вірогідності кросинговеру аж до значення 70%, а далі стабілізується. З чого можна зробити висновок, що оптимальним значенням вірогідності кросинговеру є

70% оскільки при підвищенні цього показника результати не покращуються, а затрати на обчислювання зростають.

Другим важливим параметром генетичного алгоритму є вірогідність мутації. Частіше за все її значення встановлюється рівним 1%, але в умовах не великого розміру популяції це значення можна підвищити до 5%. Мутація служить для того, щоб ГА не сходився у областях локальних екстремумів. Для пошуку оптимальної вірогідності було проведено аналогічний експеримент із зміною вірогідності мутації від 1% до 15% з кроком 1%. Результат наведено на рисунку 4.6.

Як можна побачити, довжина маршруту не залежить від вірогідності мутації так явно, як вірогідність кросинговеру, тому було обрано ту вірогідність, де довжина маршруту була найменшою – 5%.

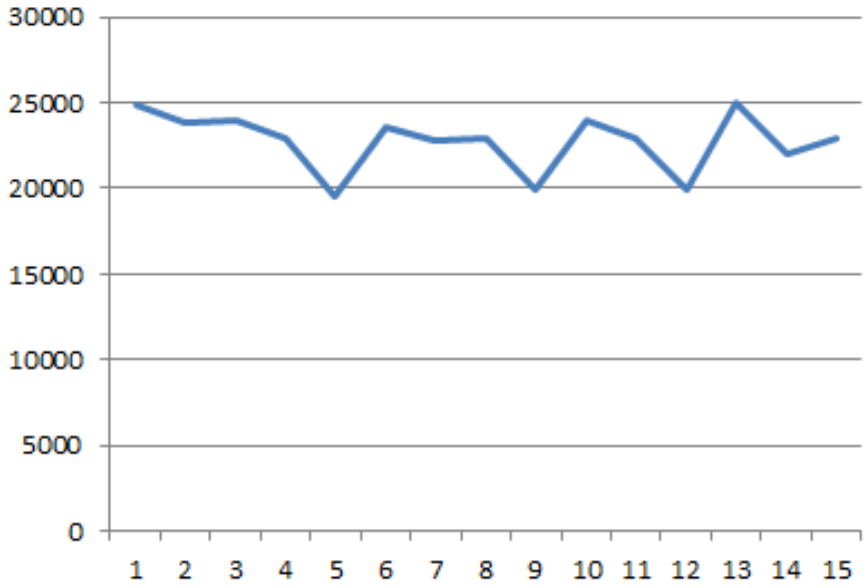


Рисунок 4.6 – Залежність довжини маршруту від вірогідності мутації

Далі було проаналізовано, як впливає потужність початкової популяції на результат. Для цього було розглянуто значення від 5 до 100 особин з шагом 5 особин. Результати наведено на рисунку 4.7.

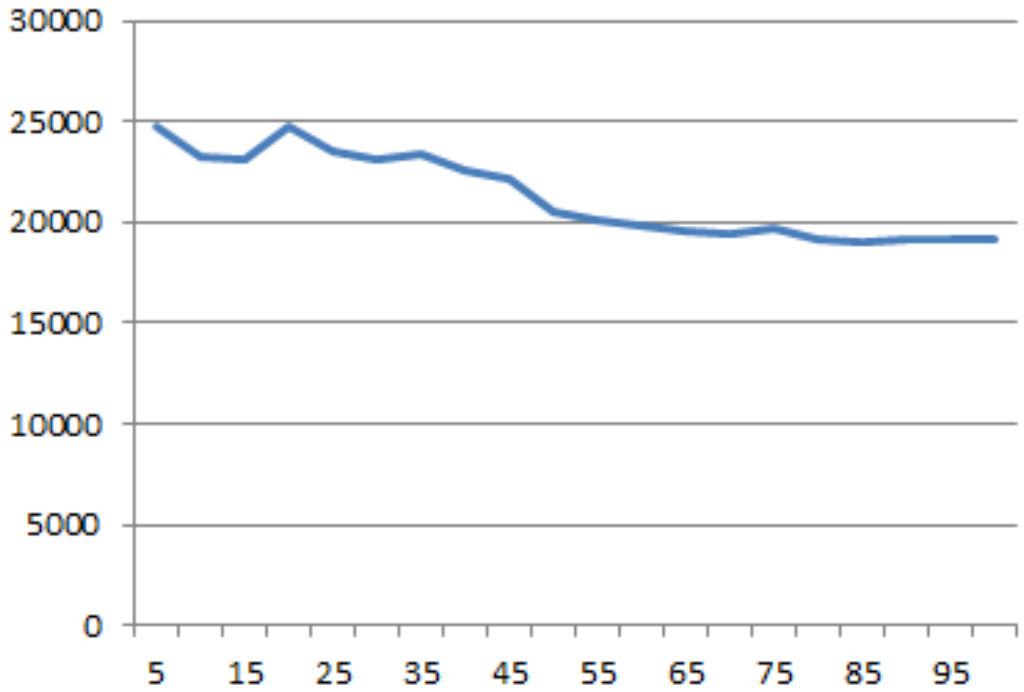


Рисунок 4.7 – Залежність довжини маршруту від потужності популяції

З графіку можна побачити, що починаючи з відмітки 55 результат перестає суттєво покращуватися, а це значить, що 55 особин є достатньо для цієї задачі. Потужність популяції суттєво впливає на якість рішення, але завищення цього показника дуже сильно підвищує затрати на обчислювання алгоритму.

Слід зазначити, що при підвищені кількості клієнтів у маршруті слід коригувати зазначені вище параметри алгоритму, особливо це стосується потужності популяції.

4.4 Висновки

Експериментальним шляхом було знайдено оптимальні параметри генетичного алгоритму для тестових даних:

- Вірогідність кросинговеру – 70%.
- Вірогідність мутації – 5%.

– Потужність популяції – 55 особин.

Результати експериментів зі знаходження оптимальних параметрів генетичного алгоритму було наведено у графічному вигляді, а отримані значення задіяні у програмному рішенні, що було описано у попередньому розділі.

Доведено працездатність розроблених моделей для різних вихідних тестових даних.

ВИСНОВКИ

Рішення задачі побудови маршрутів доставки товарів є однією з найважливіших задач у функціонуванні кожного складу нарівні з системою підтримки запасів на ньому.

Метою даної роботи була розробка комп'ютеризованої підсистеми визначення маршрутів перевезення замовлень в умовах центрального складу товарів для підвищення економічної ефективності роботи за рахунок складання оптимальних маршрутів руху транспортних засобів під час доставки.

В результаті виконання кваліфікованої роботи магістра було вирішено наступні задачі:

- Проаналізовано сучасний стан логістичних систем, який показав високий рівень розробок. Проте було відмічено, що більшість систем є комерційними, мають високу вартість, не надають детальну інформацію про використані алгоритми побудови маршрутів, що ставить під сумнів використання дійсно оптимальних або квазі оптимальних маршрутів. Крім того, надлишкові функціональні можливості роблять ці системи складним у використанні.
- Проаналізовано класичні методи кластеризації та алгоритми побудови маршрутів, було обрано та промоделювано в Matlab зазначені методи та виявлено, що для рішення задачі кластеризації найбільш придатним є алгоритм найближчого сусіда, але необхідно відмітити, зо він потребує певної модифікації в рамках вирішуваної задачі побудови маршруту.
- Проаналізовані існуючі методи рішення задачі маршрутизації та обрано за основу генетичний алгоритм.

- На підставі отриманого алгоритму було розроблено програмне забезпечення, що дозволяє вести облік заявок аптек і здійснювати побудову маршрутів доставки.
- В результаті проведених досліджень роботи програмного додатку були обрані параметри генетичного алгоритму і перевірена функціональність програмного рішення.

В цілому, всі поставлені задачі можна вважати цілком виконаними.

Перспективи подальшої розробки:

- Вдосконалення системи захисту за рахунок наступних дій:

1) Перехід на UNIX подібні системи, застосування найсучасніших протоколів захистів,
2) Проектування та впровадження модулю онлайн-оплати замовлення.
3) Захист даних, які передаються між компонентами системи, за допомогою шифрування, такого як SSL/TLS для захищених з'єднань.
4) Проектування та розробка web-версії розглянутої системи
5) Застосування хмарних технологій для зберігання даних центрального серверу.
6) Застосування сучасних нейромережевих технологій для розрахунку оптимального маршруту.

Також, в якості вдосконалення програмного забезпечення системи, можна розглянути варіант застосування технології ETL для більш гнучкої роботи з різними підсистемами компанії.

ПЕРЕЛІК ПОСИЛАНЬ

1. https://desktop.arcgis.com/en/arcmap/latest/manage-data/geocoding/what-is-geocoding.htm - ArcGIS Desktop - What is geocoding?
2. https://desktop.arcgis.com/en/arcmap/latest/manage-data/raster-and-images/esri-grid-format.htm Esri Grid format
3. https://www.esri.com/en-us/arcgis/products/arcgis-online/overview - ArcGIS Online. Cloud-based software to create and share interactive web maps
4. https://truckstop.com/ - Turn loads into profit.
5. http://www.ingit.ru/businessmap/ - Ділова карта – універсальна АРМ логістика
6. https://en.wikipedia.org/wiki/Vehicle_routing_problem - Vehicle routing problem
7. https://www.geeksforgeeks.org/traveling-salesman-problem-tsp-implementation/ - Traveling Salesman Problem (TSP) Implementation
8. https://pub.towardsai.net/optimizing-e-commerce-solving-the-bin-packing-problem-66d2437a0b1a - Optimizing E-commerce: Solving the Bin Packing Problem
9. В. В. Кирик Математичний апарат штучного інтелекту в електроенергетичних системах. Підручник Київ, КПІ ім. Ігоря Сикорського, 2019, Видачництво «Політехніка», 224 с. (https://ela.kpi.ua/bitstream/123456789/30080/1/Pidruchnyk_MatAp.pdf)
10. А.М. Козел Деякі застосування теорії графів для аналізу міжнародних відносин (https://ena.lpnu.ua:8443/server/api/core/bitstreams/c2b2044c-db9d-4f8a-8224-7ead8d2587e9/content)
11. Machine learning: стартовий курс : електронний навчальний посібник / Штовба С.Д., Козачко О.М. – Вінниця : ВНТУ, 2020. – 81 с.
12. https://studme.org/12120124/logistika/vybor_perevozchika - Вибір перевізника
13. Є. А. Настенко, В. С. Уманець Метод нечітких k-середніх з обмеженою масою робочої області формування кластерів довільної форми Electronic Archive of Kyiv Polytechnic Institute, січень 2018, https://core.ac.uk/download/323535393.pdf

14. www.mathros.net.ua – Сайт для студентів спеціальності «Інформатика». Задача комівояжера. Математична постановка задачі

15. M. Dorigo, L. M. Gambardella, Ant Colony System: A Cooperative Learning Approach to the Traveling Salesman Problem // IEEE Transactions on Evolutionary Computation Vol. 1, 1, стор. 53-66, 1997 р.

16. Наконечний С. І., Савіна С. С. Математичне програмування Навч. посіб. — К.: КНЕУ, 2003. — 452 с.

17. https://disted.edu.vn.ua/media/doc/%D0%90%D0%BB%D0%B3%D0%BE%D1%80%D0%B8%D1%82%D0%BC%20%D0%94%D1%96%D0%B5%D0%BA%D1%81%D1%82%D1%80%D0%B8.pdf Алгоритм Дейкстри

18. Gavrilenko V. Про застосування методу імітації відпалу для розв'язання нечіткої задачі комівояжера / V. Gavrilenko, K. Ivokhina, N. Rudoman // Системи управління, навігації та зв'язку. Збірник наукових праць. – Полтава: ПНТУ, 2022. – Т. 3 (69). – С. 60-63. – doi:https://doi.org/10.26906/SUNZ.2022.3.060.

19. Eiben, A. E., Raué, P., Ruttkay, Z. (1994). Genetic algorithms with multi-parent recombination. PPSN III: Proceedings of the International Conference on Evolutionary Computation. The Third Conference on Parallel Problem Solving from Nature Jerusalem, October 9–14, 2010, Israel, 78-87.

20. Steeb, W.-H. (2001). The Nonlinear Workbook: Chaos, Fractals, Cellular Automata, Neural Networks, Genetic Algorithms, Fuzzy Logic: with C++, Java, SymbolicC++ and Reduce Programs. Singapore: World Scientific.

21. Гамільтонів граф – електронний ресурс. Режим доступу: Гамільтонів граф - Wikiwand.

22. Селекція хромосом «Методом рулетки» - електронний ресурс. Режим доступу: https://studbooks.net/2091138/informatika/selektsiya_hromosom#google_vignette

Додаток А

Лістинг програмного забезпечення системи

Main.cs

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
using System.Windows.Forms.DataVisualization.Charting;

namespace ga
{
    public partial class Main : Form
    {
        List<Chromosome> Answer;

        public Main()
        {
            InitializeComponent();
        }

        private void Run_Click(object sender, EventArgs e)
        {

          Answer = new List<Chromosome>();
           int PopulationSize = Int32.Parse(tbPopSize.Text);
           ReadData data = new
ReadData("D:\\input.txt",dtp1.Value.Month.ToString(), dtp1.Value.Day.ToString(),
dtp1.Value.Year.ToString());
           List<List<int>> Clusters = new List<List<int>>();
           List<List<double>> Stat = new List<List<double>>();
           Route rt=new
Route(data.GetSize(),data.GetDistance(),data.GetCapacities(),
data.GetTruckCapacities()); //Сюда передавать в GetDistance номера задействованных
                                                                    // на дату
аптек и их кол-во. Маршрутов кол. не передавать
           Clusters = rt.GetClusters();
           tbRes.Clear();
           foreach (List<int> Cluster in Clusters)
            {
                Population pop = new Population(data.GetDistance(),
data.GetSize(), Cluster, PopulationSize);
                pop.SetCross(Double.Parse(tbCross.Text) / 100.0);
                pop.SetMut(Double.Parse(tbMut.Text) / 100.0);
                tbRes.AppendText(pop.Run(this).ToString()+"\n");
                Answer.Add(pop.GetBestChromosome());
                Stat.Add(pop.GetStatistics());

            }
           pbAll.Value = pbAll.Maximum;
```

```
            pbCur.Value = pbCur.Maximum;
            ShowChart(data.GetCoordinate(), data.GetSize(), Clusters, Stat);
            int val=-1;
            for (int i = 0; i < Answer.Count; i++)
            {
                marshrutTableAdapter.Insert(data.GetTrucks()[i], dtp1.Value);
                skladDataSet.AcceptChanges();
                marshrutTableAdapter.Update(skladDataSet.marshrut);
                this.marshrutTableAdapter.Fill(this.skladDataSet.marshrut);
                foreach (DataRow row in skladDataSet.Tables["marshrut"].Rows)
                {
                    val =
Int32.Parse(row[skladDataSet.Tables["marshrut"].Columns["id"]].ToString());
                }
                //MessageBox.Show(val.ToString());
                for (int j = 1; j < Answer[i].route.Count - 1; j++)//Обрезаем по
краям склад
                {
                    marshrut_listTableAdapter.Insert(val,
data.GetNumbers()[Answer[i].route[j]-1],42);
                    skladDataSet.AcceptChanges();
                    marshrut_listTableAdapter.Update(skladDataSet.marshrut_list);

this.marshrut_listTableAdapter.Fill(this.skladDataSet.marshrut_list);
                }

            }
        }

        void ShowChart(double[,] coord, int size, List<List<int>> Clusters,
List<List<double>>Stat)
        {
            chart1.Series.Clear(); int j;
            try
            {
                chart1.Legends.Add("Legend");
            }catch{}
            for (int i = 0; i < Clusters.Count; i++)
            {
                chart1.Series.Add(i.ToString());
                chart1.Series[i.ToString()].ChartType = SeriesChartType.Point;
                chart1.Series.Add("line"+i.ToString());
                chart1.Series["line" + i.ToString()].ChartType =
SeriesChartType.Line;
            }
            chart1.Series.Add("sklad");
            chart1.Series["sklad"].ChartType = SeriesChartType.Point;
            chart1.Series["sklad"].Color = Color.Black;
            List<Color> Colors = new List<Color>();
            Colors.Add(Color.Blue);
            Colors.Add(Color.Green);
            Colors.Add(Color.Red);
            Colors.Add(Color.Aquamarine);
            Colors.Add(Color.Aqua);
            Colors.Add(Color.Azure);
            Colors.Add(Color.Beige);
            Colors.Add(Color.Bisque);
```

```
            Colors.Add(Color.Chocolate);
            Colors.Add(Color.DarkSalmon);
            Colors.Add(Color.DimGray);
            Colors.Add(Color.DeepSkyBlue);
            for (int i=0; i<Clusters.Count; i++)
            {
                chart1.Series[i.ToString()].Color = Colors[i];
                chart1.Series["line"+i.ToString()].Color = Colors[i];
            }

            for (int i = 1; i < size; i++)
            {
                for (j = 0; j < Clusters.Count; j++)
                    if (Clusters[j].Contains(i))
                        break;
                if (j >= Clusters.Count) j = 0;
                chart1.Series[j.ToString()].Points.AddXY(coord[i, 0], coord[i,
1]);

            }
            for (int i = 0; i < Clusters.Count; i++)
                for (j = 0; j < Answer[i].GetLength(); j++ )
                    chart1.Series["line" +
i.ToString()].Points.AddXY(coord[Answer[i].GetBit(j), 0],
coord[Answer[i].GetBit(j), 1]);
            chart1.Series["sklad"].Points.AddXY(coord[0, 0], coord[0, 1]);
            chart2.Series.Clear();
            chart2.ChartAreas.Clear();
            for (int i = 0; i < Stat.Count; i++)
            {
                chart2.ChartAreas.Add(i.ToString());
                chart2.Series.Add(i.ToString());
                chart2.Series[i.ToString()].ChartArea = i.ToString();
                chart2.Series[i.ToString()].ChartType = SeriesChartType.Column;

                for (j = 0; j < Stat[i].Count; j++)
                    chart2.Series[i.ToString()].Points.AddXY(j, Stat[i][j]);
            }
        }

        private void Main_Load(object sender, EventArgs e)
        {
            // TODO: данная строка кода позволяет загрузить данные в таблицу
"skladDataSet.marshrut_list". При необходимости она может быть перемещена или
удалена.
            this.marshrut_listTableAdapter.Fill(this.skladDataSet.marshrut_list);
            // TODO: данная строка кода позволяет загрузить данные в таблицу
"skladDataSet.marshrut". При необходимости она может быть перемещена или удалена.
            this.marshrutTableAdapter.Fill(this.skladDataSet.marshrut);

        }

        private void label5_Click(object sender, EventArgs e)
        {

        }
```

```
        private void marshrutBindingNavigatorSaveItem_Click(object sender,
EventArgs e)
        {
            this.Validate();
            this.marshrutBindingSource.EndEdit();
            this.tableAdapterManager.UpdateAll(this.skladDataSet);

        }

        private void button1_Click(object sender, EventArgs e)
        {
            System.Diagnostics.ProcessStartInfo psi = new
System.Diagnostics.ProcessStartInfo("DataInput.exe");
            System.Diagnostics.Process.Start(psi);
        }
    }
}
```

ReadData.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.IO;
using System.Data.OleDb;
using System.Data;
namespace ga
{
    class ReadData
    {
        double[,] Coordinate;
        double[,] Distance;
        int[] numbers;
        double[] capacities;
        List<double> truck_capacities;
        List<int> trucks;
        int size;
        String CommandText, ConnectionString;
        public ReadData(string path, string month, string day, string year)
        {

            FileStream input;
            int i;
            input = new FileStream(path, FileMode.Open, FileAccess.Read);
            StreamReader SR = new StreamReader(input);
            truck_capacities = new List<double>();
            trucks = new List<int>();
            String line;
            //--
            CommandText = "SELECT id, capacity, zdate FROM zayavka WHERE zdate=#"
+ month + "/" + day + "/" + year + "#";
            ConnectionString = "Provider=Microsoft.Jet.OLEDB.4.0;Data
Source='Sklad.mdb'";
            OleDbDataAdapter dataAdapter = new OleDbDataAdapter(CommandText,
ConnectionString);
            DataSet ds = new DataSet();
            dataAdapter.Fill(ds, "zayavka");
```

```
            //=========================================
            size = ds.Tables["zayavka"].Rows.Count;
            if (size == 0) System.Windows.Forms.MessageBox.Show("На выбранную дату
нет заявок");
            size++;
            Distance = new double[size, size];
            Coordinate = new double[size, 2];
            numbers = new int[size-1];
            capacities = new double[size-1];
            i=0;
            foreach (DataRow row in ds.Tables["zayavka"].Rows)
            {
                numbers[i] =
Int32.Parse(row[ds.Tables["zayavka"].Columns["id"]].ToString());
                capacities[i] =
Double.Parse(row[ds.Tables["zayavka"].Columns["capacity"]].ToString());
                i++;
            }
            DataSet ds2 = new DataSet();
            CommandText = "SELECT id,capacity FROM avto";
            OleDbDataAdapter dataAdapter2 = new OleDbDataAdapter(CommandText,
ConnectionString);
            dataAdapter2.Fill(ds2, "avto");
            foreach (DataRow row2 in ds2.Tables["avto"].Rows)
            {

truck_capacities.Add(Double.Parse(row2[ds2.Tables["avto"].Columns["capacity"]].ToS
tring()));

trucks.Add(Int32.Parse(row2[ds2.Tables["avto"].Columns["id"]].ToString()));
            }
            string[] Strings = new string[size];
            line = SR.ReadLine(); //Пропуск кол-ва
            line = SR.ReadLine(); //Склад
            Strings = line.Split(' ');
            Coordinate[0, 0] = Double.Parse(Strings[0]);
            Coordinate[0, 1] = Double.Parse(Strings[1]);
            int line_num = 0;
            i = 1;
            while (!SR.EndOfStream)
            {
                line = SR.ReadLine();
                line_num++;
                if (numbers.Contains(line_num))
                {
                    Strings = line.Split(' ');
                    for (int j = 0; j < 2; j++)
                    {
                        Coordinate[i, j] = Double.Parse(Strings[j]);
                    }
                    i++;
                }
            }
            SR.Close();
            input.Close();
            InitDistance();
```

```
        }

        public void InitDistance()
        {
            for (int i=0; i<size; i++)
                for (int j=0; j<size; j++)
                    Distance[i,j] = Math.Sqrt(Math.Pow(Coordinate[i,0] -
Coordinate[j,0],2) + Math.Pow(Coordinate[i,1] - Coordinate[j,1],2));
        }
        public double[,] GetCoordinate()
        {
            return Coordinate;
        }
        public double[,] GetDistance()
        {
            return Distance;
        }
        public int GetSize()
        {
            return size;
        }

        internal double[] GetCapacities()
        {
            return capacities;
        }

        internal List<double> GetTruckCapacities()
        {
            return truck_capacities;
        }
        internal List<int> GetTrucks()
        {
            return trucks;
        }
        internal int[] GetNumbers()
        {
            return numbers;
        }
    }
}
```

Route.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;


namespace ga
{
    class Route
    {
        private int n; // Count of points
        private int k; //Count of routes/cars
        private double[,] Distance; //Matrix of distances
        private double[] capacities;// Объем каждой заявки
```

```
        private List<double> truck_cap;// Объем автомобилей
        public Route(int new_n, double[,] new_distance, double[] new_capacities,
List<double> new_truck_cap)
        {
            n = new_n;
            capacities = new double[n-1];//Отбрасывание склада
            for (int i = 0; i < n-1; i++)
                capacities[i] = new_capacities[i];
            Distance = new double[n, n];
            truck_cap = new List<double>(new_truck_cap);
            InitDistance(new_distance);
        }

        public void InitDistance(double[,] new_distance)
        {
            for (int i = 0; i < n; i++)
                for (int j = 0; j < n; j++)
                    Distance[i,j] = new_distance[i,j];
        }

        private void CopyMatr(double[,] a, double[,] b)
        {
            for (int i = 0; i < n; i++)
                for (int j = 0; j < n; j++)
                    a[i, j] = b[i, j];
        }
        public List<List<int>> GetClusters()
        {
            List<List<int>> res = new List<List<int>>();
            double cur_line_sum, max_line_sum=0, min_line_sum=0;
            int max_line_sum_index,min_line_sum_index; int y;
            double[,] a = new double[n, n];
            List<int> cur_constr = new List<int>();
            CopyMatr(a, Distance);
            bool go;
            double cur_capacity;
            int cnt = 0; //Кол-во обработанных аптек
            y = 0;
            cnt = 0;
            while (cnt<n-1)
            {
                cur_capacity = 0;
                cur_constr.Clear();
                           res.Add(new List<int>());
                            max_line_sum = 0;
                            max_line_sum_index = 1;//=0

                            for (int i = 1; i < n; i++) //i=0
                            {
                                go = true;
                                for (int cn=0; cn<res.Count; cn++)
                                    if (res[cn].Contains(i))
                                    {
                                        go = false;
                                        break;
                                    }
                                if (!go) continue;
```

```
                                cur_line_sum = 0;
                                for (int j = 1; j < n; j++)//j=0
                                    cur_line_sum += a[i, j];
                                if (cur_line_sum > max_line_sum)
                                {
                                    max_line_sum = cur_line_sum;
                                    max_line_sum_index = i;
                                }
                            }
                            res[y].Add(max_line_sum_index);
                            cnt++;
                            if (cnt == n - 1) break;
                            cur_capacity += capacities[max_line_sum_index-1];
                            bool b = true;

                    while (b)
                    {
                        min_line_sum = int.MaxValue;
                        min_line_sum_index = 1; //=0
                        for (int i = 1; i < n; i++) //i=0
                        {
                            go = true;
                            for (int cn = 0; cn < res.Count; cn++)
                                if (res[cn].Contains(i) || cur_constr.Contains(i))
                                {
                                    go = false;
                                    break;
                                }
                            if (!go) continue;
                                cur_line_sum = 0;
                                for (int j = 1; j < res[y].Count; j++) //j=0
                                    cur_line_sum += Distance[i, res[y][j]];
                                if (min_line_sum > cur_line_sum)
                                {
                                    min_line_sum = cur_line_sum;
                                    min_line_sum_index = i;
                                }
                        }
                        if (cur_capacity + capacities[min_line_sum_index - 1] <=
truck_cap[y])
                        {
                            cur_capacity += capacities[min_line_sum_index - 1];
                            res[y].Add(min_line_sum_index);
                            cnt++;
                        }
                        else cur_constr.Add(min_line_sum_index);

                        if (cnt + cur_constr.Count >= n-1) b = false;
                    }
                    y++;
                }
                return res;
            }

        }
}
```

Population.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Windows.Forms;
namespace ga
{
    class Population
    {
        List<int> Cluster; // Множество точек, кластер, на котором происходит
поиск маршрута
        List<Chromosome> population; // Исходный набор хромосом данной популяции
        Chromosome Best; // Лучшая хромосома
        double[,] Distance; // Матрица расстояний
        int dist_size; // Размер матрицы расстояний
        int n; //Chromosome length, количество точек
        int pop_size; // Population size
        double p_cross = 0.6, p_mut = 0.05; // Вероятности кроссинговера и мутации
        int MaxIter = 250; int MaxEps = 1000;
        List<double> Statistics;
        public Population(double[,] new_distance, int new_dist_size, List<int>
new_cluster, int new_pop_size)
        {
            Cluster = new List<int>();
            population = new List<Chromosome>();
            Best = new Chromosome(new List<int>(n));
            pop_size = new_pop_size;
            dist_size = new_dist_size;
            Cluster.AddRange(new_cluster);
            n = new_cluster.Count +2; // В начало и в конце добавляется склад
            Distance = new double[dist_size, dist_size];
            for (int i = 0; i < dist_size; i++)
            {
                for (int j = 0; j < dist_size; j++)
                    Distance[i, j] = new_distance[i, j];
            }
            for (int i = 0; i < pop_size; i++)
                population.Add(new Chromosome());
            Statistics = new List<double>();
        }

        public void GeneratePopulation()
        {
            Random rnd = new Random();
            for (int i = 0; i < pop_size; i++)
            {
                population[i].AddBit(0); // Первым всегда будет склад
                for (int j = 1; j < n-1; j++)
                    while (!population[i].AddBit(Cluster[rnd.Next(n-2)])) { }
                population[i].AddBit(0,false); // Последний - склад
            }
        }
```

```
        public List<Chromosome> Roulette()
        {
            List<Chromosome> TempPopulation = new List<Chromosome>(); // Временная
популяция
            double[] dst=new double[pop_size]; // Длина маршрута для хромосомы
            int[] P = new int[pop_size]; // Вероятность попадания хромомсомы во
временную
            double Avg = 0; // Средняя длина маршрута популяции
            for (int i = 0; i < pop_size; i++)
            {
                dst[i] = GetDistance(population[i]);
                Avg += dst[i];
            }
            Avg /= pop_size;
            for (int i = 0; i < pop_size; i++)
            {
                P[i] = (int)Math.Round(dst[i] / Avg); // Определение вероятности
попоадания хромосомы во временную популяцию
                if (P[i] == 50) MessageBox.Show("ololo");
                for (int j = 0; j < P[i]; j++)
                    TempPopulation.Add(population[i]);
            }

          return TempPopulation;
        }

        public List<Chromosome> Crossingover(List<Chromosome> TempPopulation)
        {
            List<Chromosome> ChildPopulation = new List<Chromosome>();
            Random rnd = new Random(); int rn1, rn2; bool go;
            for (int i = 0; i < TempPopulation.Count - 1; i++)
            {
                if (new Random().NextDouble() < p_cross) {
                Chromosome Child1 = new Chromosome(n);
                Chromosome Child2 = new Chromosome(n);
                go = true;
                int recuc = 0;
                do
                {
                    rn1 = rnd.Next(TempPopulation.Count);
                    rn2 = rnd.Next(TempPopulation.Count);
                    if (!TempPopulation[rn1].Equals(TempPopulation[rn2]))
                    {
                        for (int j = 1; j < n/2; j++) // Крайние точки не трогаем,
это склад
                        {
                            int num1 = j;

                            while (!Child1.SetBit(j,
TempPopulation[rn1].GetBit(num1))) // Запись в левый Child1
                            {
                                num1++;

                            }

                            num1 = n-j-1;
```

```
                            while (!Child1.SetBit(n - j - 1,
TempPopulation[rn2].GetBit(num1))) // Запись в правый Child1
                            {
                                num1--;

                            }
                            num1 = j;

                            while (!Child2.SetBit(j,
TempPopulation[rn2].GetBit(num1))) // Запись в левый Child2
                            {
                                num1++;

                            }
                            num1 = n - j - 1;

                            while (!Child2.SetBit(n - j - 1,
TempPopulation[rn1].GetBit(num1))) // Запись в правый Child2
                            {
                                num1--;

                            }
                        }
                        if (n % 2 != 0) //поледний шаг, если нечетная длина
хромосомы
                        {
                            int num1 = 1;
                            while (!Child1.SetBit(n / 2,
TempPopulation[rn1].GetBit(num1))) // Запись в левый Child1
                            {
                                num1++;

                            }
                            num1 = 1;
                            while (!Child2.SetBit(n / 2,
TempPopulation[rn2].GetBit(num1))) // Запись в левый Child2
                            {
                                num1++;

                            }
                        }
                        go = false;
                    }
                    recuc++;
                    if (recuc > 10000) break;
                }
                while (go);
                ChildPopulation.Add(Child1);
                ChildPopulation.Add(Child2);
            } //if p_cross
            }

            return ChildPopulation;
        }

        public List<Chromosome> Mutation(List<Chromosome> input_ChildPopulation)
```

```
        {
            List<Chromosome> ChildPopulation = new List<Chromosome>();
            ChildPopulation.AddRange(input_ChildPopulation);
            Random rnd = new Random(); int rn1, rn2,tmp;
            for (int i = 0; i < ChildPopulation.Count; i++ )
                if (new Random().NextDouble() < p_mut)
                {
                    do
                    {
                        rn1 = rnd.Next(n);
                        rn2 = rnd.Next(n);

                    }
                    while (rn1 == rn2 || rn1==0 || rn2==0 || rn1==n-1 || rn2==n-
1);
                    tmp = ChildPopulation[i].GetBit(rn1);
                    ChildPopulation[i].SetBit(rn1,
ChildPopulation[i].GetBit(rn2),false);
                    ChildPopulation[i].SetBit(rn2, tmp,false);

                }
                return ChildPopulation;
        }

        public void Reduction(List<Chromosome> ChildPopulation) // Добавление
детей в основную. Сортировка. Обрезание.
        {
            population.AddRange(ChildPopulation);
            SortPopulation();
            population.RemoveRange(pop_size, (population.Count - pop_size));
        }
        public double Run(Main fm)
        {
            //form
            fm.pbCur.Maximum = MaxIter;
            fm.pbAll.Maximum = MaxEps;
            List<Chromosome> tmp = new List<Chromosome>();
            GeneratePopulation();
            Best = population[0];
            if (n > 3) //т.е. более одной аптеки в развозе
            {

                int counter = 0;
                for (int i = 0; i < MaxEps; i++)
                {
                    tmp = Roulette();
                    tmp = Crossingover(tmp);
                    tmp = Mutation(tmp);
                    Reduction(tmp);
                    if (GetDistance(population[0]) < GetDistance(Best))
                    {
                        Best = population[0];
                        counter = 0;
                    }
                    else counter++;
                    if (counter > MaxIter) break;
```

```
                //form output
                fm.pbAll.Value = i;
                fm.pbCur.Value = counter;
                Statistics.Add(GetDistance(population[0]));
            }

        }
        return GetDistance(Best);
    }
    public double GetDistance(Chromosome chrom)
    {
        double dst=0;
        for (int i = 0; i < n - 1; i++)
            dst += Distance[chrom.GetBit(i), chrom.GetBit(i + 1)];
        return dst;
    }

    public void SortPopulation()
    {
        Chromosome tmp = new Chromosome();
        bool flag=true;
        for (int i = 0; i<population.Count; i++)
        {
            if (flag==false) break;
            flag = false;
            for (int j=0; j<population.Count-1; j++)
                if (GetDistance(population[j]) > GetDistance(population[j+1]))
                {
                    tmp = population[j];
                    population[j] = population[j+1];
                    population[j+1] = tmp;
                    flag = true;
                }
        }
    }

    public Chromosome GetBestChromosome()
    {
        return Best;
    }
    public void SetCross(double new_p_cross)
    {
        p_cross = new_p_cross;
    }
    public void SetMut(double new_p_mut)
    {
        p_mut = new_p_mut;
    }
    public List<double>GetStatistics()
    { return Statistics; }
}
}
```

Chromosome.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
```

```
using System.Text;

namespace ga
{
   public class Chromosome
    {
       public List<int> route; // Путь, состоящий из номеров городов

        public Chromosome()
        {
            route = new List<int>();
        }
        public Chromosome(int n)
        {
            route = new List<int>(n);
            for (int i = 0; i < n; i++)
                route.Add(0);
        }
        public Chromosome(List<int> new_route)
        {
            route = new List<int>();
            route.AddRange(new_route);
        }
        public bool AddBit(int val)
        {
            if (!route.Contains(val))
            {
                route.Add(val);
                return true;
            }
            else return false;
        }
        public void AddBit(int val,bool t)
        {
                route.Add(val);
        }
        public int GetBit(int k)
        {
            return route[k];
        }

        public bool SetBit(int k, int val)
        {
            if (!route.Contains(val))
            {
                route[k]=val;
                return true;
            }
            else return false;
        }
        public void SetBit(int k, int val, bool t)
        {
           route[k] = val;
        }
        public bool Equals(Chromosome input_ch)
        {
            bool res=true;
```

```
            for (int i = 0; i < input_ch.route.Count; i++)
                if (route[i] != input_ch.route[i])
                { res = false; break; }
            return res;
         }
        public int GetLength()
        {return route.Count;}
    }
}
```

ДОДАТОК Б

Охорона праці. Аналіз умов праці відділу розробки програмного забезпечення для планування на промисловому підприємстві

Відділ розробки програмного забезпечення для планування на промисловому підриємстві розташовується на п’ятому поверсі п’ятиповерхового будинку. Розмір приміщення офісу наступний: довжина 5м; ширина 3м; висота 2,5м.

Виходячи з цих даних маємо:

- Площа приміщення

$$Sп = 5 * 4 = 20 м^2$$

- Об’єм приміщення

$$Vп = 5 * 4 * 2,5 = 50 м^3$$

Таким чином на одного програміста припадає площа в 20 $м^2$ і об’єм в 50 $м^3$ на людину.

Стіни мають шпалери бежевого кольору, коефіцієнт відбиття для стін 75%, для стелі – 50%. Висота столів над рівнем підлоги – 0,75м.

Для освітлення використовуються світильники типу ЛБ-80, які розміщуються на стелі. Вони розміщуються на відстані 0,15 м від стелі. Сумарна площа віконних прорізів - 2м [2.]

Відділ займається розробкою, удосконаленням і технічною підтримкою програмного забезпечення для планування на промисловому підприємстві. У відділі працює один програміст.

Виконуються стандарти щодо розміщення меблі в приміщенні:

– мінімум 1м між комп’ютером та стіною з вікном;

– мінімум 1,2 м між боковими сторонами моніторів;

– мінімум 2,5 м між тильною стороною монітора та іншого комп’ютера або робочого місця;

– мінімум 1 м прохід.

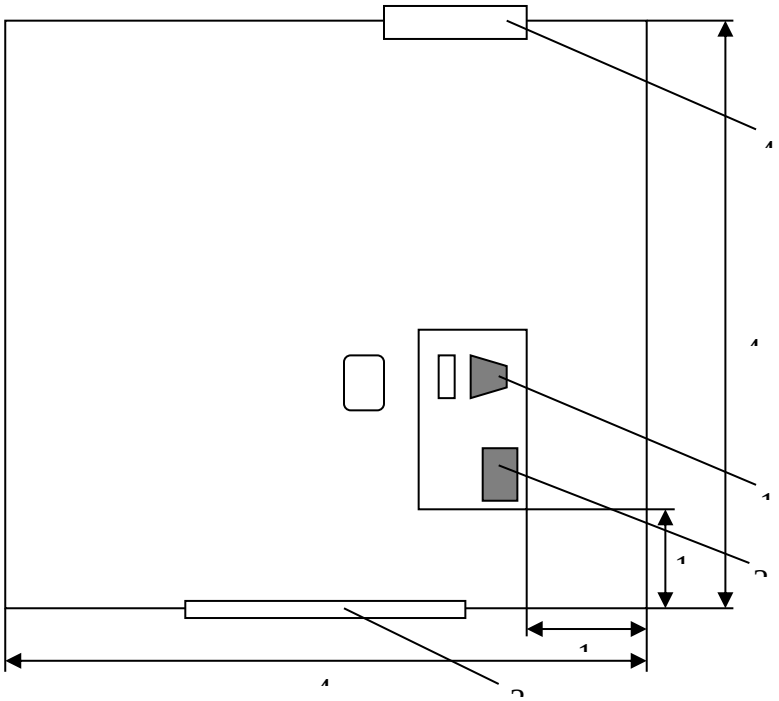


Рисунок Б.1 – План промислового приміщення(1 - робоче місце, 2 - принтер, 3 - віконний проріз, 4 - дверний проріз)

Всі заземлені металеві предмети, такі як радіатори опалення, водопровід і газопровід, повинні бути закриті діелектричним щитом, щоб уникнути контакт з ними. У нас в приміщенні є радіатори опалення і вони закриті діелектричним щитом.

Робоче місце програміста знаходиться в офісі, а основна діяльність програміста пов'язана з роботою за комп'ютером. Таким чином, основні шкідливі фактори, що впливають на працівника офісу, пов'язані з комп'ютерами.

Стандарт 12.0.003-74 ССБТ «Небезпечні та шкідливі виробничі фактори» дає класифікацію небезпечних і шкідливих факторів на виробництві.

Можна виділити виробничі фактори, які виникають внаслідок роботи комп'ютера: підвищений рівень шуму; іонізація повітря; електричні і магнітні

поля; підвищена яскравість світла; фізичні перевантаження (кисті рук, спина); психологічні перевантаження.

Всі ці шкідливі виробничі фактори викликають професійні захворювання у співробітника офісу, до яких відносяться наступні хвороби: ослаблення пам'яті; головні болі; серцево-судинні захворювання; ослаблення зору. Крім того, розумове напруження може спровокувати різні психологічні захворювання.

Розрахунок системи загального рівномірного освітлення з лампами розжарювання для приміщення, в якому використовуються зорові роботи високої точності

Офісне приміщення має розміри 5х4х2,5 м; коефіцієнт відбиття стін 70%, стелі - 50%; висота столів над рівнем підлоги - 0,75 м; відстань від світильника до стелі - 0,15 м.

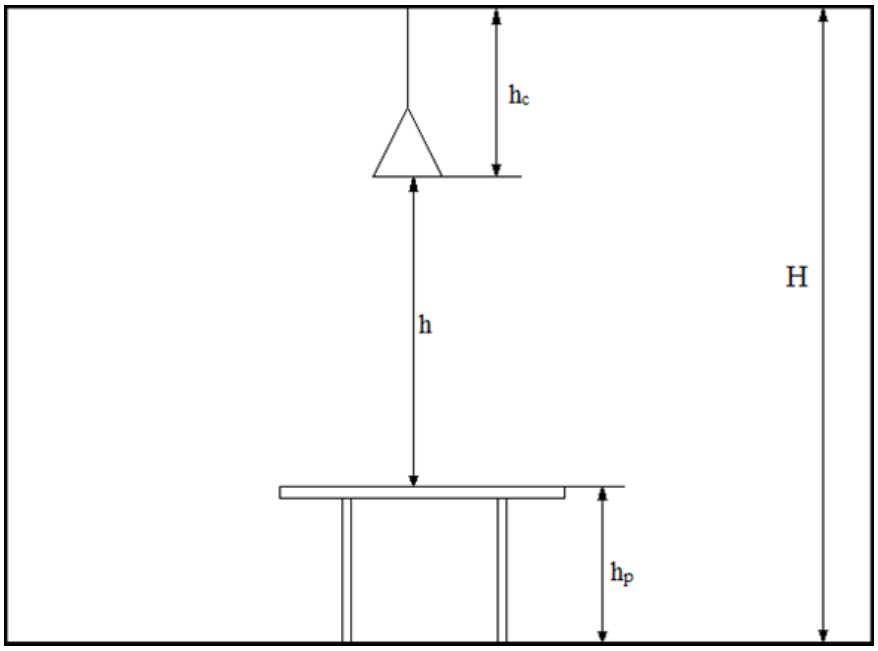


Рисунок Б.2 – Розрахунок індексу приміщення

Згідно СНіП II-4-79 «Будівельні норми і правила. Природне і штучне освітлення. Норми проектування », мінімальна освітленість складає Е = 400-500лк.

Насамперед визначимо висоту підвісу світильника над робочим місцем:

$$H_c = H - h_C - h_p = 2,5 - 0,15 - 0,75 = 1,6 \text{ м}$$

Далі визначимо рекомендовану відстань між світильниками за такою формулою:

$$L = 1,4\ H_c = 1,4 * 1,6 = 2,24 \text{ м}$$

Світловий потік світильника визначається такою формулою:

$$Ф_л = \frac{E \cdot K_З \cdot S \cdot Z}{N \cdot n \cdot \eta} ЛМ,$$

де E - нормативна освітленість, лк;

$K_з$ - коефіцієнт запасу, що враховує зниження освітленості в результаті забруднення та старіння ламп $K_з = 1,3$

S - площа приміщення, що освітлюється, $м^2$;

Z - коефіцієнт нерівномірності освітлення для ламп розжарювання (=1,15);

N - кількість світильників;

n - кількість ламп у світильнику;

η - коефіцієнт використання світового потоку, який визначається за світлотехнічними таблицями залежно від показника приміщення (i) та коефіцієнтів відбиття стін та стелі.

Для світильників, що мають світловий потік $Ф_1 = 4320$ лм. (При i = 1,09, $Р_{стелі} = 70\%$, $Р_{стін} = 50\%$), коефіцієнт використання становить η = 0,5.

Обчислимо світловий потік одного світильника:

$Ф_л = (E * S * K_З * Z)/ \eta = (400 * 20 * 1,3 * 1,15) / 0,5 = 23920$ лм.

Кількість необхідних світильників можна порахувати, розділивши нормативний світловий потік на світловий потік, створюваний одним світильником:

$$N = Ф_л / Ф$$

$$N = 23920 / 4320 = 6 \text{ шт}$$

Таким чином, для освітлення приміщення необхідно наявність 6 світильників типу ЛБ-80.

Комп'ютер, розташований в офісі, є джерелом тепла, таким чином, необхідно визначити умови його вентиляції.

Наступна формула дозволяє знайти повітрообмін по теплу:

$$L = \frac{Q_{над}}{c \times p(t_в - t_н)},$$

де $Q_{над}$ - надлишкове виділення тепла в робочому приміщенні, ккал/год.;

с - теплоємкість повітря (0,237 ккал/кг°С);

р - обсягова вага повітря (1,226 кг/ м^3);

$t_в$ - температура витяжного повітря (30°С);

$t_н$ - температура приточного повітря (20°С).

Надмірне надходження тепла можна обчислити за формулою:

$$Q_{над} = Q_{уст} + Q_{пер} + Q_{осв} + Q_{ср}$$

де $Q_{уст}$ - виділення тепла від устаткування;

$Q_{пер}$ - виділення тепла робітниками;

$Q_{осв}$ - надходження тепла від електричного освітлення;

$Q_{ср}$ - надходження тепла від сонячної радіації через вікна.

Виділення тепла від установок обчислюється за такою формулою:

$$Q_{уст} = Р \times К_а \times К_б \times 860,$$

де Р - сумарна потужність устаткування, кВт/год;

$К_а$ - коефіцієнт установленої потужності (0,95);

$К_б$ - коефіцієнт одночасної роботи (1,0).

В офісі маємо одне робоче місце, з 1 системним блоком і 1 монітором. Крім того, в офісі є один принтер. Потужність системного блоку 0,35 КВт, потужність монітора 0,04 КВт, потужність принтера 0,3. Таким чином, обчислюємо Р:

$$Р = 0,35 + 0,04 + 0,3 = 0,69 \text{ кВт/год}$$

Таким чином, отримуємо:

$$Q_{уст} = 0,69 \times 0,95 \times 1 \times 860 = 563,73 \text{ ккал/год}$$

Визначимо виділення тепла працівниками за такою формулою:

$$Q_{пер} = n \times g,$$

де n - кількість працюючих;

g - кількість тепла, що виділяє один працівник за годину (100 ккал/год.).

Таким чином, отримуємо:

$$Q_{пер} = 1 \times 100 = 100 \text{ ккал/год}$$

Тепер, визначимо виділення тепла від електричного освітлення

$$Q_{осв} = E_м \times g_1 \times S \quad \text{ккал/год}$$

де $E_м$ - нормована освітленість для цієї зорової роботи приймаємо рівним 400 лк;

g_1 - питоме тепловиділення на 1 м^2 підлоги при 1 лк освітленості (для люмінесцентних ламп - 0,05 ккал/год.).

S - площа приміщення, м^2.

Підставивши числові значення, отримаємо:

$Q_{осв} = 400 \times 0,05 \cdot \times 20 = 400$ ккал/год

Далі визначимо тепло, створюване сонячною радіацією, яка потрапляє через вікна:

$$Q_{ср} = F \cdot g_2 \cdot K_{осл},$$

де F - площа віконних прорізів (2 м^2);

g_2 - кількість тепла, що надходить через 1 м^2 віконного прорізу

(65 ккал/год.);

$К_{осл}$ - коефіцієнт ослаблення, приймаємо - 0,4.

Підставивши числові значення, маємо:

$$Q_{ср} = 2 \cdot 65 \cdot 0,4 = 52 \text{ ккал/год}$$

Порахуємо сумарне значення надлишкового тепла:

$$Q_{над} = 563,73 + 100 + 400 + 52 = 1115,73 \text{ ккал/год}$$

Визначимо витрати повітря в приміщенні:

$$L = 1115,73 / (0,237 * 1,226 * (30-20)) = 384,73 \text{ м}^3\text{/год}$$

Система вентиляції, встановлена в приміщенні, має потужність 1000м^3/год, таким чином, вона задовольняє нормативам.

Параметри мікроклімату на робочих місцях регламентуються ДСН 3.3.6.042-99 «Санітарні норми мікроклімату виробничих приміщень». Відповідно до даних санітарних норм:

У холодні періоди року: температура повітря 21°-23°С; швидкість руху повітря 0.1 метра в секунду; відносна вологість 40-60 %.

У теплі періоди року: температура повітря 22°-24°С; швидкість руху повітря 0.2 метра в секунду; відносна вологість 40-60 %.

Вище зазначені норми цілком відповідають фактичним даним приміщення.