**АНОТАЦІЯ**

**ABSTRACT**

**ЗМІСТ**

**ЗАГАЛЬНА ЧАСТИНА**

**зміст**

**ІНДИВІДУАЛЬНА ЧАСТИНА №1**

**зміст**

**ІНДИВІДУАЛЬНА ЧАСТИНА №2**

**зміст**

# ВСТУП

# ЗАГАЛЬНІ ПОЛОЖЕННЯ

## Опис предметного середовища

### Опис процесу діяльності

### Опис функціональної моделі

## Огляд наявних аналогів

## Постановка задачі

### Призначення розробки

### Цілі та задачі розробки

## Висновок до розділу

# ІНФОРМАЦІЙНЕ ЗАБЕЗПЕЧЕННЯ

## Вхідні дані

## Вихідні дані

## Опис структури бази даних

## Структура масивів інформації

## Висновок до розділу

# МАТЕМАТИЧНЕ ЗАБЕЗПЕЧЕНЯ

Даний комплексний дипломний проект присвячений складанню плану перевезень продукції за різних умов: перевезення малогабаритної продукції, перевезення продукції із урахуванням вантажомісткості транспортних засобів та визначення оптимальної кількості транспортних засобів для здійснення цих перевезень, перевезення із урахуванням вантажомісткості транспортних засобів, обмеженості автопарку та наявної системи штрафів. Індивідуальні частини дипломного проекту присвячені останнім двом видам перевезень. У загальній частині наведемо математичний апарат для розв’язання задачі перевезення малогабаритної продукції, оскільки математична постановка цієї задачі та підходи до її розв’язку є основою для розв’язання інших задач перевезення продукції, у яких враховується різноманітні обмеження.

## Змістовна постановка задачі

Опишемо стандартну постановку задачі. Логістична компанія займається перевезенням продукції із складів до клієнтів. Компанія має у своєму розпорядженні склад у певному місті та парк транспортних засобів, розташований у цьому ж місті. Клієнти, яким необхідно доставити продукцію, розташовані у різних містах.

При стандартній постановці вантажомісткість транспортних засобів не враховується. Такий випадок можливий, якщо компанія займається перевезеннями малогабаритної продукції (наприклад, розвезенням пошти).

Задача полягає у приписуванні маршрутів окремим транспортними засобам, за якого сумарний маршрут буде мінімальним за вартістю (довжиною). При цьому має бути здійснене обслуговування усіх клієнтів, до того ж кожний клієнт має бути відвіданий лише один раз. Кожний транспортний засіб має починати свій маршрут із міста, у якому розташований склад, та закінчувати його у тому ж місті.

Розв’язання цієї задачі дозволить логістичній компанії значно зменшити витрати на перевезення продукції. Практика показує, що складання вдалого плану перевезення продукції дозволяє зменшити пов’язані із перевезенням витрати до 40%.

Описана вище задача відома під назвою «задача маршрутизації транспортних засобів» (Vehicle Routing Problem – VRP). Цей термін є універсальним і застосовується до цілого класу задач, у яких для заданого парку транспортних засобів, що зосереджені на одному або декількох складах, будується маршрут для певної кількості географічно розрізнених клієнтів.

Зазвичай у реальному світі, виникає багато сторонніх обмежень, що накладаються на описану вище спрощену постановку задачі. Найбільш важливими та суттєвими є такі обмеження:

* кожний транспортний засіб має обмежену вантажомісткість (Capacitated VRP – CVRP);
* кожний клієнт має бути обслужений протягом певного часового проміжку (VRP with Time Windows – VRPTW);
* компанія володіє не одним, а декількома складами розташованими у різних містах (Multiple Depot VRP – MDVRP);
* клієнти можуть бути обслужені різними транспортними засобами, а не лише одним (Split Delivery VRP – SDVRP);
* на складі розміщується лише автопарк, а продукція знаходиться у певних пунктах реалізації продукції, тобто необхідно попередньо забрати продукцію, а потім їхати до клієнта (VRP with Pick-Up and Delivering – VRPPD);
* доставки продукції мають здійснюватися періодично у визначені дні (Periodic VRP – PVRP);
* деякі з величин (кількість клієнтів, їх замовлення, час доставки тощо) є випадковими (Stochastic VRP – SVRP).

Також можливе поєднання цих обмежень у будь-яких комбінаціях. Наприклад, задача маршрутизації транспортних засобів із часовими вікнами та обмеженою вантажомісткістю транспорту (Capacitated VRP with Time Windows – CVRPTW), або задача маршрутизації транспортних засобів із часовими вікнами та декількома складами (Multiple Depot VRP with Time Windows – MDVRPTW).

## Математична постановка задачі

Як було сказано вище у загальній частині ми розглянемо найпростіший випадок, коли компанія має лише один склад, вантажомісткість транспортних засобів вважається необмеженою, а також немає ні часових, ні будь-яких інших обмежень.

### Класична задача маршрутизації транспортних засобів

Класична задача маршрутизації транспортних засобів полягає у приписуванні маршрутів усім наявним транспортним засобам, таким чином щоб їх сумарна вартість (довжина) була мінімальною. При цьому усі клієнти мають бути відвідані лише один раз одним транспортним засобом.

Задача маршрутизації транспортних засобів може бути визначена на орієнтованому графі , де – множина вершин, – множина ребер графа. Вершина відповідає складу, а множина інших вершин відповідає споживачам (клієнтам). Також кожній дузі приписана вага , що відповідає вартості (довжині) переїзду із міста у місто . При цьому якщо , то . Автопарк нараховує транспортних засобів.

Сформулюємо задачу у термінах цілочисельного лінійного програмування. Нехай – бінарна змінна, що приймає значення 1, якщо маршрут -ого транспортного засобу включає безпосередній переїзд із міста у місто , і значення 0 у іншому випадку.

Використовуючи введену вище нотацію задача може бути сформульована наступним чином.

Цільова функція – мінімізація сумарної вартості (довжини) складеного маршруту:

|  |  |
| --- | --- |
|  | (3.1) |

Обмеження:

|  |  |
| --- | --- |
|  | (3.2) |
|  | (3.3) |
|  | (3.4) |
|  | (3.5) |
|  | (3.6) |

Обмеження (3.2) гарантують, що кожний споживач буде відвіданий лише один раз одним транспортним засобом. Обмеження (3.3) забезпечують виїзд транспортного засобу від кожного відвіданого ним споживача. Обмеження (3.4) гарантують виїзд кожного транспортного засобу із міста-складу, забезпечуючи таким чином використання усіх наявних транспортних засобів. Обмеження (3.5) унеможливлюють утворення у розв’язку маршрутів, які не містять склад.

На рисунку 3.1 наведена графічна ілюстрація класичної постановки задачі, а на рисунку 3.2 представлене її розв’язання.



Рисунок 3.1 – Типові початкові дані для класичної задачі маршрутизації транспортних засобів



Рисунок 3.2 – Розв’язок задачі наведеної вище

## Обґрунтування методу розв’язання

Існує багато підходів до розв’язання задачі маршрутизації транспортних засобів. Нижче представлена класифікація можливих методів розв’язку поставленої задачі.

Точні методи розв’язання забезпечують знаходження оптимального розв’язку. Середи них:

* повний перебір;
* метод гілок і меж;
* метод гілок та відсічень.

Евристичні методи виконують обмежене дослідження пошукового простору і , як правило, знаходять достатньо непогані розв’язки за невеликий час. Можна виділити два підвиди евристичних методів:

1. конструктивні методи, які поступово будують допустимий розв’язок, слідкуючи при цьому за вартістю розв’язку, але не містять етап покращення розв’язку;
2. двохетапні алгоритми передбачають розбиття задачі на дві природні складові:
   1. розподілення споживачів між транспортними засобами (кластеризація вершин графу);
   2. побудова маршрутів відповідно до здійсненого на минулому етапі розподілення.

Метаевристичні методи направлені на глибоке дослідження найбільш багатообіцяючих областей множини розв’язків. Якість цих методів вища ніж, та яку надає класична евристика. Серед методів даного напрямку варто згадати наступні:

* алгоритм мурашиних колоній;
* алгоритм імітації відпалу;
* генетичний алгоритм;
* табу пошук;
* програмування в обмеженнях.

Оскільки задача маршрутизації транспортних засобів є NP-повною, то використання точних методів розв’язання є недоцільним при . Тому, щоб досягти компромісу між точністю знайденого розв’язку (відносно оптимального) та часом затраченим на знаходження цього розв’язку, використовують евристичні та метаевристичні алгоритми.

Оскільки заздалегідь передбачити для алгоритму співвідношення «час-точність» важко, для розв’язання цієї задачі було обрано два різні підходи, щоб обрати ефективніший з них. Один із них – двохетапний алгоритм (евристичний метод). Доцільність його використання обумовлена тим що, він розбиває задачу на дві природні складові. Також було обрано один із метаевристичних алгоритмів – бджолиний алгоритм, який є відносно новим і який показав себе як більш ефективний алгоритм у порівнянні із алгоритмами мурашиних колоній та генетичним при розв’язанні інших NP-повних задач.

## Опис методів розв’язання

### Застосування бджолиного алгоритму для розв’язання задачі маршрутизації транспортних засобів

Класичним підходом для розв’язання стандартної задачі маршрутизації транспортних засобів є зведення її до задачі комівояжера, яку далі розв’язують метаевристичними алгоритмами.

Для того, щоб звести VRP до задачі комівояжера (ЗК) необхідно продублювати вершину 0, що відповідає складу, відповідно до кількості транспортних засобів. Тобто граф має містити вершин , що відповідають складу. При цьому вартість переїзду між цими вершинами . Це забезпечить формування окремих маршрутів.

На рисунку 3.3 наведена графічна ілюстрація зведення VRP наведеної на рисунку 3.1 до ЗК та її розв’язок.



Рисунок 3.3 – Зведення VRP до задачі комівояжера

#### Опис бджолиного алгоритму у загальному випадку

Бджолиний алгоритм моделює поведінку бджіл при пошуку їжі у природному середовищі. Бджолина колонія здатна розлітатися на декілька кілометрів у різних напрямках від вулика. Бджоли задіяні у пошуку їжі поділяються на розвідників та фуражирів (робочих). Бджоли-розвідники шукають нектар, літаючи від одної квіткової ділянки до іншої. Якість квіткової ділянки оцінюється за декількома параметрами: відстанню ділянки до вулика, а також якістю та кількістю нектару на даній ділянці. Знайшовши квіткову ділянку бджола розвідник повертається до вулика, де вона зустрічається з робочими бджолами на «танцполі». Там вона за допомогою спеціального танцю повідомляє інформацію про знайдену ділянку (напрям, відстань та якість) робочим бджолам. Залежно від якості кожна квіткова ділянка відвідується відповідною кількістю бджіл-фуражирів або ж зовсім ігнорується ними.

Імітування поведінки бджіл при розв’язанні задач оптимізації виявилось ефективним у порівнянні із багатьма іншими алгоритмами. Наведемо загальну покрокову схему бджолиного алгоритму.

**Крок 0.** Ініціалізація початкових параметрів алгоритму.

На цьому кроці задаються такі параметри:

* кількість бджіл-розвідників;
* кількість бджіл-фуражирів;
* кількість кращих квіткових ділянок, знайдених розвідниками;
* розмір квіткової ділянки.

**Крок 1.** Пошук випадкових розв’язків.

Бджоли-розвідники, які знайшли на попередній ітерації кращі розв’язки залишаються на своїх квіткових ділянках. Інших бджіл відправляємо на пошук нових квіткових ділянок з нектаром. Тобто на першій ітерації на пошук відправляються усі розвідники, а далі лише ті бджоли, що знайшли гірші квіткові ділянки.

**Крок 2.** Локальна оптимізація

2.1 Вибір ділянок для оптимізації

Оцінюємо наявні квіткові ділянки (найкращі ділянки з попередньої ітерації та ділянки знайденій на першому кроці поточної ітерації) за критерієм якості (значенням цільової функції). Далі обираємо кращі ділянки (їх кількість визначена на кроці 0) для яких і буде проводитися локальна оптимізація.

2.2 Локальна оптимізація

Розсилаємо бджіл-фуражирів по кращих обраних ділянках у межах їх розміру, визначеному на нульовому кроці алгоритму, за принципом «чим краща ділянка – тим більше фуражирів». Кожний фуражир оцінює якість квіткової ділянки у точці, у якій він знаходиться (значення цільової функції у даній точці). Локальна оптимізація полягає у наступному: серед усіх фуражирів ділянка обирається той, що знайшов найкращий розв’язок, і якщо цей розв’язок кращий за знайдений бджолою-розвідником даної ділянки, то розвідник переміщується на місце цього фуражира.

**Крок 3.** Завершення алгоритму

Якщо виконується критерій зупинки (наприклад, виконано певну кількість ітерацій, або розв’язки за останні ітерації змінюються несуттєво тощо), завершити алгоритм. Інакше перейти на перший крок.

#### Застосування бджолиного алгоритму для розв’язання задачі комівояжера

Класична постановка ЗК визначена наступним чином: комівояжеру необхідно побувати у кожному із міст, закінчуючи свій маршрут у початковому місті, при цьому він не має двічі заїжджати до жодного із міст. Вартість (відстань) переїздів між містами задані ( – вартість переїзду між містами та ). Необхідно знайти найдешевший (найкоротший) цикл обходу усіх міст.

Наведемо математичну постановку ЗК. Нехай змінна , якщо цикл містить переїзд , у іншому випадку .

Цільова функція:

|  |  |
| --- | --- |
|  | (3.7) |

Обмеження:

|  |  |
| --- | --- |
|  | (3.8) |
|  | (3.9) |
|  | (3.10) |
|  | (3.11) |

Обмеження (3.8) та (3.9) забезпечують відповідно один в’їзд до кожного міста і один виїзд. Обмеження (3.10) виключають можливість утворення підциклів у розв’язку.

Для того щоб застосувати бджолиний алгоритм для розв’язку конкретної задачі необхідно визначити, як генерувати випадкові розв’язки, так як знаходити сусідні розв’язки (тобто розв’язки у межах заданої «квіткової ділянки»).

Уведемо поняття циклу. Цикл – це набір із пар міст, які забезпечують маршрут, що проходить через кожне місто лише один раз:

|  |  |
| --- | --- |
|  | (3.12) |

Для зручності будемо використовувати іншу форму запису циклу:

|  |  |
| --- | --- |
|  | (3.13) |

Формування випадкового розв’язку не є складною задачею і полягає у формуванні перестановок порядку . Більш цікавим є обрання стратегії пошуку сусідніх розв’язків, оскільки саме від цього залежить ефективність етапу локальної оптимізації.

Спочатку сусідні розв’язки визначались наступним чином: для поточного розв’язку обиралось від 2 до міст, для яких випадковим чином змінювався порядок проходження у маршруті.

Наприклад, цикл до перестановки мав вигляд:

|  |  |
| --- | --- |
|  | (3.14) |

а після:

|  |  |
| --- | --- |
|  | (3.15) |

На рисунку 3.4 наведений приклад того, як змінивши таким методом порядок проходження вершин, отримали кращий розв’язок. Початковий розв’язок був 1–2–**6**–4–5–**3**–1, а після перестановки – 1–2–**3**–4–5–**6**–1.

|  |  |
| --- | --- |
|  |  |
| Рисунок 3.4 – Початковий розв’язок (зліва) та отриманий при перестановці на його основі оптимум (справа) | |

Але на практиці виявилось, що при застосуванні такої стратегії пошуку сусідніх розв’язків, високою є ймовірність «застрягти» у локальному оптимумі. На рисунку 3.5 показаний вигляд локального оптимуму, із якого алгоритм не міг вибратися. У зображеному випадку отримати кращий розв’язок можна лише помінявши місцями дві пари вершин одночасно, так як будь-яка інша перестановка приводить до гіршого розв’язку (приклад такої перестановки також наведений на рисунку 3.5).

|  |  |
| --- | --- |
|  |  |
| Рисунок 3.5 – Локальний оптимум (зліва) та отримана на його основі перестановка (справа) | |

Оскільки для задач великої розмірності ймовірність, того що буде здійснена послідовність перестановок, що дозволить вийти із локального оптимуму, є малою, було вирішено випробувати й інші стратегії пошуку сусідніх розв’язків.

Розглянувши останній приклад (рис. 3.5) стає зрозуміло, що змінювати необхідно порядок проходження не для будь-яких міст, а для групи міст, що йдуть підряд у заданому розв’язку.

Був застосований метод, що полягає у інвертуванні частини розв’язку. Для поточного розв’язку обиралось від 2 до міст, для яких порядок проходження у маршруті змінювався на зворотній.

Тобто, якщо цикл до інверсії мав наступний вигляд

|  |  |
| --- | --- |
|  | (3.16) |

і для інверсії обрали вершини , то після інверсії цикл матиме такий вигляд:

|  |  |
| --- | --- |
|  | (3.17) |

На рисунку 3.6 показано приклад того, як дана стратегія справляється із «застряванням» у локальному оптимумі. З рисунку видно, що маючи початковий розв’язок 1–2–3–**7–6–5–4**–8–1 та інвертуючи секцію 7–6–5–4, отримали кращий розв’язок 1–2–3–**4–5–6–7**–8–1.

|  |  |
| --- | --- |
|  |  |
| Рисунок 3.6 – Локальний оптимум (зліва) та отриманий на його основі кращий розв’язок (справа) | |

Використовуючи новий метод знаходження сусідніх розв’язків, алгоритм доволі швидко знаходив розв’язки близькі до оптимального.

Виявилось, що застосовуючи лише цю стратегію пошуку сусідніх розв’язків (без будь-яких інших методів), алгоритм знаходить розв’язок кращий і за меншу кількість ітерацій, аніж, якщо застосовувати додаткові методи.

### Застосування 2-етапного алгоритму для розв’язання задачі маршрутизації транспортних засобів

#### Перший етап – кластеризація вершин графу

##### **Постановка задачі кластеризації**

Задача кластеризації (Data clustering) – задача розбиття вибірки об’єктів на підмножини, що називаються кластерами, так, щоб кожний кластер складався із схожих об’єктів, а об’єкти різних кластерів суттєво відрізнялись.

Наведемо формальну постановку задачі кластеризації. Нехай – множина об’єктів, а – множина порядкових номерів (імен, міток) кластерів. Задана функція відстані між xоб’єктами , і маємо скінчену вибірку об’єктів . Необхідно розбити вибірку на множини (кластери), що не перетинаються, так щоб кожний кластер складався із об’єктів близьких за метрикою , а об’єкти різних кластерів суттєво відрізнялись. При цьому кожному об’єкту ставиться у відповідність кластер .

Алгоритм кластеризації – це функція , яка будь-якому об’єкту ставить у відповідність номер кластеру . Множина у деяких випадках може бути задана, але частіше необхідно визначити оптимальну кількість кластерів з точки зору якогось критерію якості.

Для розв’язання VRP 2-етапним методом необхідно розбити вершини графу, що відповідають клієнтам, за територіальною ознакою (метрика ) на кластери, кількість яких дорівнює кількості наявних транспортних засобів. Нехай кожному транспортному засобу відповідає свій кластер . Для кластерів має виконуватися наступне:

|  |  |
| --- | --- |
|  | (3.18) |
|  | (3.19) |

Задача спрямована на територіальне розбиття вершин, для цього як критерій однорідності об’єктів було вирішено використовувати функцію, яка направлена на мінімізацію сумарного середньоквадратичного відхилення точок кластеру від їхніх центрів:

|  |  |
| --- | --- |
|  | (3.20) |

де – центр мас кластеру .

Центр мас розраховується за формулою:

|  |  |
| --- | --- |
|  | (3.21) |

де – маса вектора .

У випадку VRP маси усіх точок вважаються рівними 1. Тому центр мас обчислюється за формулою:

|  |  |
| --- | --- |
|  | (3.22) |

##### **Метод k-середніх**

Метод k-середніх – найпопулярніший метод кластеризації. Дія алгоритму спрямована на мінімізацію сумарного середньоквадратичного відхилення точок кластерів від їхніх центрів (формула (3.20)).

Основна ідея алгоритму полягає у тому, що на кожній ітерації заново обчислюється центр мас кожного кластеру, отриманих на попередній ітерації. Потім вектори знов розбиваються на кластери у відповідності із тим, який із нових кластерних центрів виявився ближчим відповідно обраної метрики . Алгоритм завершується, коли на деякій ітерації не відбувається зміни кластерів. Алгоритм завершується за скінченну кількість кроків, бо кількість можливих розбиттів скінченної множини є скінченою.

Серед проблем методу k-середніх виділяють наступні:

* не гарантується досягнення глобального мінімуму, а лише одного із локальних;
* результат залежить від початкового вибору центрів кластерів, їх оптимальний вибір не відомий;
* кількість кластерів необхідно знати заздалегідь.

Для того щоб алгоритм не формував порожніх кластерів при першому розподілі точок по кластерах необхідно кожному із кластерів приписати по одній будь-якій точці, усі інші точки розподілити по кластерах відподно до того який кластерний центр ближчий.

На рисунку 3.7 наведений приклад застосування методу k-середніх.

|  |  |
| --- | --- |
| 1. Початковий вибір центрів кластерів | 2. Розподіл точок по кластерах |
| 3. Визначення нових центрів кластерів | 4. Перерозподіл точок по кластерах |
| Рисунок 3.7 – Приклад застосування методу k-середніх | |

##### **Бджолиний алгоритм для кластеризації**

Основною проблемою методу k-середніх виявилось те, що цей метод ніяк не враховує розміщення складу.

Тому було вирішено застосувати бджолиний алгоритм для кластеризації, який буде враховувати цей нюанс. Щоб алгоритм враховував відстань до складу необхідно модифікувати цільову функцію (формула (3.20)) наступним чином:

|  |  |
| --- | --- |
|  | (3.23) |

де – відстань від центру кластеру до складу (вершина 0).

На рисунку 3.8 показаний приклад того, як впливає врахування розташування складу при кластеризації на кінцевий розв’язок задачі. Бджолиний алгоритм враховує розташування складу і формує кластери таким чином щоб якомога більша їх кількість була ближча до складу, у той час як у методі k-середніх можливе формування великої кількості кластерів далеко від складу.



Рисунок 3.8 – Порівняння методу k-середніх (зліва) та бджолиного алгоритму кластеризації (справа)

Випадкові розв’язки формуються випадковим розкидуванням усіх вершин графу у кластерів. Якщо якийсь кластер виявляється пустим, розв’язок вважається недопустимим. На рисунку 3.9 наведений приклад випадкової кластеризації вершин на 4 кластери.



Рисунок 3.9 – Випадковий розв’язок задачі кластеризації

Сусідні розв’язки отримувались із заданого розв’язку шляхом переміщення випадкової вершини із одного кластеру у інший. При цьому переміщення вершини із кластеру, що складається тільки із цієї вершини заборонено. На рисунку 3.10 наведений приклад формування сусіднього розв’язку. Заданого методу отримання сусідніх розв’язків цілком вистачає, оскільки усі інші методи будуть просто комбінацією випадкових переміщень.



Рисунок 3.10 – Формування сусіднього розв’язку

#### Другий етап – побудова маршрутів відповідно до проведеної кластеризації

Для побудованих кластерів будуються маршрути проходження вершин кластеру із початком маршруту у вершині-складі. Для цього разів розв’язується задача комівояжера для вершин кожного кластеру об’єднаних із вершиною-складом, тобто для .

Задача комівояжера для кожного кластеру розв’язується бджолиним алгоритмом (див. п. 3.4.1.2).

## Висновок до розділу

# ПРОГРАМНЕ ТА ТЕХНІЧНЕ ЗАБЕЗПЕЧЕННЯ

## Засоби розробки

## Вимоги до технічного забезпечення

### Загальні вимоги

## Архітектура програмного забезпечення

### Діаграма класів

### Діаграма послідовності

### Діаграма компонентів

### Специфікація функцій

## Опис звітів

## Висновок до розділу

# ТЕХНОЛОГІЧНИЙ РОЗДІЛ

## Керівництво користувача

## Випробування програмного продукту

### Мета випробувань

### Загальні положення

### Результат випробувань

## Висновок до розділу

# ЗАГАЛЬНІ ВИСНОВКИ

При виконанні комплексного дипломного проекту були детально розглянуті питання, які виникають у процесі створення плану перевезення продукції із складу до споживачів, та виділені основні ключові етапи та взаємозв’язки між ними притаманні цьому процесу.

Для розв’язання поставленої задачі був проведений ґрунтовний аналіз предметного середовища та ретельно описаний бізнес-процес складання плану перевезень. На основі даних отриманих в процесі проведеного аналізу була сформульована математична постановка задачі та розроблений математичний апарат для її розв’язку.

Метою даного дипломного проекту була не просто розробка методу розв’язання даної задачі, а створення якомога більш ефективного методу розв’язання даної задачі, тобто такого методу який буде зберігати баланс між точністю знайденого розв’язку та часом, затраченим на знаходження цього розв’язку. Тому для розв’язання задачі було застосовано декілька підходів та проведений глибокий порівняльний аналіз на основі отриманих експериментальних даних.

Даний дипломний проект є комплексним, у загальній його частині описана задача перевезення малогабаритної продукції, наведені реалізовані методи розв’язання та проведений аналіз цих методів. Індивідуальна частина № 1 дипломного проекту присвячена складанню плану перевезень із урахуванням вантажомісткості транспортних засобів та визначенню оптимальної кількості транспортних засобів для здійснення цих перевезень. У індивідуальні частині № 2 описані дослідження складання плану перевезень із урахуванням вантажомісткості транспортних засобів, обмеженого автопарку та системи штрафів.

Для розробки програмного продукту була обрана мова програмування C# як мова розробки застосувань для платформи [Microsoft .NET Framework](http://ru.wikipedia.org/wiki/.NET_Framework). Для зберігання вхідних даних та зв’язків між ними було вирішено використовувати реляційну базу даних. Microsoft SQL Server було обрано у якості системи управління бази даних, оскільки платформа [Microsoft .NET Framework](http://ru.wikipedia.org/wiki/.NET_Framework) надає зручні засоби роботи із даною СУБД.

У загальній частині дипломного проекту також описана архітектура програмного забезпечення та вимоги до технічного забепечення, наведена інструкція користувача по експлуатації комплексу задач, описана методика випробувань.

# ПЕРЕЛІК ПОСИЛАНЬ

1. Article “” [Електронний ресурс] // Режим доступу: [http://neo.lcc.uma.es/radi‑aeb/WebVRP/index.html?/Problem\_Descriptions/VRPPDDesc.html](http://neo.lcc.uma.es/radiaeb/WebVRP/index.html?/Problem_Descriptions/VRPPDDesc.html);
2. Article “Vehicle routing problem” [Електронний ресурс] // Режим доступу: [http://en.wikipedia.org/wiki/Vehicle\_routing\_problem](http://en.wikipedia.org/wiki/K-means_clustering);
3. ФИО первого автора. The Vehicle Routing Problem: Last Advances and New Challenges [Текст] // ИОФ всех авторов // Город печатания: Springer, 2008. – 589 с.
4. Article “Travelling salesman problem” [Електронний ресурс] // Режим доступу: [http://en.wikipedia.org/wiki/Travelling\_salesman\_problem](http://en.wikipedia.org/wiki/K-means_clustering);
5. Article “Cluster analysis” [Електронний ресурс] // Режим доступу: <http://en.wikipedia.org/wiki/Cluster_analysis>;
6. Article “k-means clustering” [Електронний ресурс] // Режим доступу: <http://en.wikipedia.org/wiki/K-means_clustering>;