**АНОТАЦІЯ**

**Структура та обсяг роботи.** Пояснювальна записка загальної частини комплексного дипломного проекту складається з 4 розділів, містить 55 сторінок, 13 рисунків, 6 таблиць, 3 додатки, 12 джерел.

Комплексний дипломний проект присвячений комплексу задач складання плану перевезень продукції із метою зменшення витрат на перевезення. У загальній частині проекту розглянуто алгоритми та методи розв’язання задачі складання плану перевезень малогабаритної продукції. Індивідуальна частина № 1 присвячена складанню плану перевезень продукції із урахуванням вантажомісткості транспортних засобів та визначенню їх необхідної кількості для здійснення перевезень, а індивідуальна частина № 2 – перевезенням із урахуванням вантажомісткості транспортних засобів, їх обмеженої кількості та пріоритетів замовлень.

У розділі з інформаційного забезпечення визначені вхідні та вихідні дані до комплексу задач та розроблена відповідна структура бази даних.

У розділі з математичного забезпечення наведена математична постановка задачі. Розроблені та описані алгоритми для розв’язання задачі складання плану перевезень малогабаритної продукції і проведений їх порівняльний аналіз на основі отриманих експериментальних даних.

Розділ програмного забезпечення присвячений обґрунтуванню та опису обраної архітектури програмного забезпечення.

МАРШРУТ, ЦИКЛ, ЗАДАЧА МАРШРУТИЗАЦІЇ ТРАНСПОРТНИХ ЗАСОБІВ, ЗАДАЧА КЛАСТЕРИЗАЦІЇ, МЕТОД K-СЕРЕДНІХ, МЕТАЕВРИСТИКА, БДЖОЛИНИЙ АЛГОРИТМ, ЗАДАЧА КОМІВОЯЖЕРА.

**ABSTRACT**

**The structure and scope of paper.** Explanatory note of the general part of the complex degree project consists of 4 sections, contains 112 pages, 13 figures, 6 tables, 3 appendices, 12 references.

The complex thesis project dedicated to a complex of problems of defining a plan of goods transporting in order to reduce transportation cost. The general part of the project deals with algorithms and methods for solving problem of planning transporting of the small-sized products. The individual part № 1 devoted to scheduling transportation of products in consideration of cargo capacity of vehicles and to determining the required number of vehicles, while individual part № 2 devoted to transportation in consideration of cargo capacity and limited number of vehicles and in consideration of order priority.

In the part of information support input and output data of the complex of problems defined and appropriate database structure designed.

The section of mathematical support contains mathematical formulation of the problem and approaches to problem solving reasoning. Algorithms for solving the problem of the small-sized products transporting were developed and a comparative analysis based on experimental data was performed.

The software part dedicated to the chosen software architecture and the reason of choosing it.

ROUTE, LOOP, VEHICLE ROUTING PROBLEM, DATA CLUSTERING, K-MEANS METHOD, METAHEURISTIC, BEES ALGORITHM, TRAVELING SALESMAN PROBLEM.

**ЗМІСТ**

**ЗАГАЛЬНА ЧАСТИНА**

[ВСТУП 11](#_Toc325323524)

[1 ЗАГАЛЬНІ ПОЛОЖЕННЯ 13](#_Toc325323525)

[1.1 Опис предметного середовища 13](#_Toc325323526)

[1.1.1 Опис процесу діяльності 15](#_Toc325323527)

[1.1.2 Опис функціональної моделі 16](#_Toc325323528)

[1.2 Огляд наявних аналогів 17](#_Toc325323529)

[1.3 Постановка задачі 18](#_Toc325323530)

[1.3.1 Призначення розробки 18](#_Toc325323531)

[1.3.2 Цілі та задачі розробки 18](#_Toc325323532)

[Висновок до розділу 20](#_Toc325323533)

[2 ІНФОРМАЦІЙНЕ ЗАБЕЗПЕЧЕННЯ 21](#_Toc325323534)

[2.1 Вхідні дані 21](#_Toc325323535)

[2.2 Вихідні дані 21](#_Toc325323536)

[2.3 Опис структури бази даних 22](#_Toc325323537)

[2.4 Структура масивів інформації 25](#_Toc325323538)

[Висновок до розділу 26](#_Toc325323539)

[3 МАТЕМАТИЧНЕ ЗАБЕЗПЕЧЕННЯ 27](#_Toc325323540)

[3.1 Змістовна постановка задачі 27](#_Toc325323541)

[3.2 Математична постановка задачі 29](#_Toc325323542)

[3.2.1 Класична задача маршрутизації транспортних засобів 29](#_Toc325323543)

[3.3 Обґрунтування методу розв’язання 32](#_Toc325323544)

[3.4 Опис методів розв’язання 33](#_Toc325323545)

[3.4.1 Застосування бджолиного алгоритму для розв’язання задачі маршрутизації транспортних засобів 33](#_Toc325323546)

[3.4.1.1 Опис бджолиного алгоритму у загальному випадку 34](#_Toc325323547)

[3.4.1.2 Застосування бджолиного алгоритму для розв’язання задачі комівояжера 36](#_Toc325323548)

[3.4.2 Застосування 2-етапного алгоритму для розв’язання задачі маршрутизації транспортних засобів 40](#_Toc325323549)

[3.4.2.1 Перший етап – кластеризація вершин графу 40](#_Toc325323550)

[3.4.2.1.1 Постановка задачі кластеризації 40](#_Toc325323551)

[3.4.2.1.2 Метод k-середніх 42](#_Toc325323552)

[3.4.2.1.3 Бджолиний алгоритм для кластеризації 43](#_Toc325323553)

[3.4.2.2 Другий етап – побудова маршрутів відповідно до проведеної кластеризації 45](#_Toc325323554)

[3.5 Порівняльний аналіз алгоритмів 45](#_Toc325323555)

[Висновок до розділу 50](#_Toc325323556)

[4 ПРОГРАМНЕ ТА ТЕХНІЧНЕ ЗАБЕЗПЕЧЕННЯ 51](#_Toc325323557)

[4.1 Архітектура програмного забезпечення 51](#_Toc325323558)

[4.1.1 Діаграма класів 51](#_Toc325323559)

[4.1.2 Діаграма послідовності 53](#_Toc325323560)

[4.1.3 Діаграма компонентів 54](#_Toc325323561)

[4.2 Опис звітів 55](#_Toc325323562)

[Висновок до розділу 56](#_Toc325323563)

[ЗАГАЛЬНІ ВИСНОВКИ 57](#_Toc325323564)

[ПЕРЕЛІК ПОСИЛАНЬ 59](#_Toc325323565)

[ДОДАТОК А Технічне завдання 60](#_Toc325323566)

[ДОДАТОК Б Тексти програмного коду 62](#_Toc325323567)

[ДОДАТОК В Графічний матеріал 80](#_Toc325323568)

**ІНДИВІДУАЛЬНА ЧАСТИНА №1**

ВСТУП 8

1 ЗАГАЛЬНІ ПОЛОЖЕННЯ 9

1.1 Опис предметного середовища 9

1.2 Постановка задачі 9

1.2.1 Призначення розробки 9

1.2.2 Цілі та задачі розробки 9

Висновок до розділу 10

2 МАТЕМАТИЧНЕ ЗАБЕЗПЕЧЕНЯ 11

2.1 Змістовна постановка задачі 11

2.2 Математична постановка задачі 11

2.3 Обґрунтування методу розв’язання 15

2.4 Опис методів розв’язання 15

2.4.1 Опис 2-етапного алгоритму розв’язання CVRP 15

2.4.1.1 Перший етап – кластеризація вершин графу 15

2.4.1.1.1 Алгоритм ланцюга найближчого сусіда 16

2.4.1.1.2 Бджолиний алгоритм кластеризації 18

2.4.1.2 Другий етап – побудова маршрутів відповідно до проведеної кластеризації 20

2.4.2 Опис методу розв’язання CVRP+ 20

2.5 Порівняльний аналіз алгоритмів 21

Висновок до розділу 25

3 ТЕХНОЛОГІЧНИЙ РОЗДІЛ 26

3.1 Керівництво користувача 26

3.2 Випробування програмного продукту 34

3.2.1 Мета випробувань 34

3.2.2 Загальні положення 34

3.2.3 Результат випробувань 34

Висновок до розділу 40

4 РОЗДІЛ З ОХОРОНИ ПРАЦІ 41

4.1 Загальні вимоги до виробничих приміщень 41

4.2 Характеристика робочого місця 43

4.3 Мікроклімат 44

4.4 Характеристика випромінювання 45

4.5 Освітлення 45

4.6 Виробничий шум 45

4.7 Електробезпека 46

4.8 Пожежна безпека 46

4.9 Правила безпечної роботи із ПК 47

Висновок до розділу 48

ЗАГАЛЬНІ ВИСНОВКИ 49

ПЕРЕЛІК ПОСИЛАНЬ 51

ДОДАТОК А Графічний матеріал 52

**ІНДИВІДУАЛЬНА ЧАСТИНА №2**

ВСТУП 8

1 ЗАГАЛЬНІ ПОЛОЖЕННЯ 9

1.1 Опис предметного середовища 9

1.2 Постановка задачі 9

1.2.1 Призначення розробки 9

1.2.2 Цілі та задачі розробки 9

Висновок до розділу 10

2 МАТЕМАТИЧНЕ ЗАБЕЗПЕЧЕНЯ 11

2.1 Змістовна постановка задачі 11

2.2 Математична постановка задачі 11

2.3 Обґрунтування методу розв’язання 14

2.4 Опис 2-етапного алгоритму для розв’язання CVRPP 15

2.4.1 Перший етап – кластеризація вершин графу 15

2.4.1.1 Багатокритеріальна оптимізація 16

2.4.1.2 Скаляризація векторної цільової функції 18

2.4.1.3 Гібридна модифікація алгоритму ланцюга найближчого сусіда та бджолиного алгоритму 20

2.4.1.4 Бджолиний алгоритм кластеризації 21

2.4.2 Другий етап – побудова маршрутів відповідно до проведеної кластеризації 22

2.5 Порівняльний аналіз отриманих результатів 22

Висновок до розділу 26

3 ПРОГРАМНЕ ТА ТЕХНІЧНЕ ЗАБЕЗПЕЧЕННЯ 27

3.1 Засоби розробки 27

3.2 Вимоги до технічного та програмного забезпечення 30

3.3 Архітектура програмного забезпечення 31

3.3.1 Специфікація функцій 31

Висновок до розділу 42

4 РОЗДІЛ З ОХОРОНИ ПРАЦІ 43

4.1 Загальні вимоги до виробничих приміщень 43

4.2 Характеристика робочого місця 45

4.3 Мікроклімат 46

4.4 Характеристика випромінювання 47

4.5 Освітлення 47

4.6 Виробничий шум 47

4.7 Електробезпека 48

4.8 Пожежна безпека 49

4.9 Інструкція із техніки безпеки при роботі із ПК 50

Висновок до розділу 50

ЗАГАЛЬНІ ВИСНОВКИ 51

ПЕРЕЛІК ПОСИЛАНЬ 53

ДОДАТОК А Графічний матеріал 54

# ВСТУП

Логістика – частина економічної науки, предмет якої полягає в організації раціонального процесу просування товарів і послуг від постачальників сировини до споживачів, функціонування сфери обігу продукції, товарів, послуг, управління товарними запасами, створення інфраструктури руху товару. Логістика спрямована на оптимізацію витрат і раціоналізацію процесу виробництва, збуту і супутнього сервісу як в рамках одного підприємства, так і для групи підприємств.

На сьогодні основним завданням логістики є організація перевезень продукції таким чином, щоб витрати на ці перевезення були якомога меншими. Тому даний дипломний проект присвяченийрозробці комплексу задач складання плану перевезень продукції.

У загальній частині дипломного проекту описується задача маршрутизації транспортних засобів при перевезенні малогабаритної продукції. Індивідуальна частина № 1 присвячена задачі складання плану перевезень однорідної продукції із урахуванням вантажомісткості транспортних засобів та визначення необхідної кількості транспортних засобів для здійснення цих перевезень. У індивідуальній частині № 2 розглядається задача складання плану перевезень продукції із урахуванням вантажомісткості та обмеженої кількості транспортних засобів, а також пріоритетів замовлень.

Призначенням комплексу задач є забезпечення логіста засобами автоматизованого створення плану перевезень продукції з мінімальними сумарними витратами.

Метою комплексу задач є зменшення сумарних витрат на перевезення продукції та збитків, пов’язаних із неврахуванням пріоритетів замовлень. Для досягнення цілей у роботі вирішуються задачі, які згадувалися вище та яким присвячені загальна та індивідуальні частини дипломного проекту. У свою чергу для розв’язання цих задач необхідно вирішити такі задачі, як розподілення клієнтів між транспортними засобами та складання маршрутів для кожного транспортного засобу відповідно до виконаного розподілення.

**Практичне значення одержаних результатів.** Розроблено евристичні, метаевристичні алгоритми, а також їх гібридні модифікації для розв’язання задач, що у сукупності складають собою задачу маршрутизації транспортних засобів із урахуванням різних обмежень (таких як вантажомісткість транспортних засобів, їх обмежена кількість, пріоритети замовлень). Проведено порівняльний аналіз ефективності роботи розроблених методів на основі отриманих експериментальних даних.

# ЗАГАЛЬНІ ПОЛОЖЕННЯ

## Опис предметного середовища

**Логістика** – частина економічної науки, предмет якої полягає в організації раціонального процесу просування товарів і послуг від постачальників сировини до споживачів, функціонування сфери обігу продукції, товарів, послуг, управління товарними запасами, створення інфраструктури руху товару.

Більш широке визначення логістики наведене у [1] і трактує її як науку про планування, управління і контроль руху матеріальних, інформаційних та фінансових ресурсів у різних системах. З позиції менеджменту організації логістику можна розглядати як стратегічне управління матеріальними потоками в процесі закупівлі, постачання, перевезення, збуту, і зберігання матеріалів, деталей і готового інвентарю. Поняття включає в себе також управління відповідними інформаційними та фінансовими потоками.

Логістика спрямована на оптимізацію витрат і раціоналізацію процесу виробництва, збуту і супутнього сервісу як в рамках одного підприємства, так і для групи підприємств. Змістом логістики як науки є встановлення причинно-наслідкових зв’язків і закономірностей, властивих процесу товароруху, з метою визначення та реалізації на практиці ефективних організаційних форм і методів управління матеріальними та інформаційними потоками.

**Логістична система** – сукупність дій учасників логістичного ланцюга (підприємств-виробників, транспортних, торговельних організацій, магазинів, тощо), побудованих таким чином, щоб виконувалися основні завдання логістики.

Відповідно до [2] **транспортна логістика** – це система з організації доставки, а саме з переміщення будь-яких матеріальних предметів, речовин тощо з однієї точки в іншу по оптимальному маршруту. Один з основних напрямків науки про управління інформаційними і матеріальними потоками в процесі руху товарів.

**Оптимальним** **маршрутом** вважається такий, по якому можливо доставити логістичний об’єкт в найкоротші терміни (або встановлені терміни) з мінімальними витратами, а також з мінімальною шкодою для об’єкта доставки.

Шкодою для об’єкта доставки вважається негативний вплив на логістичний об’єкт як з боку зовнішніх чинників (умови перевезення), так і з боку часового фактора при доставці об’єктів.

Основною функцією транспортної логістики є транспортування.

**Транспортування** полягає у переміщенні продукції транспортним засобом за певною технологією в ланцюзі поставок і складається з логістичних операцій і функцій. Детальніша інформація про транспортування наведена у [3].

Організація перевезень є важливою функцією. Існує декілька підходів до організації перевезень. Серед них можемо назвати два основних: децентралізоване та централізоване транспортування.

За **децентралізованого** перевезення кожен підрозділ компанії самостійно, без узгодження й координації з іншими займається переміщенням необхідних вантажів.

За **централізованого** перевезення тільки вповноважений транспортний підрозділ має право здійснювати операції із перевезень. Інші підрозділи компанії виставляють замовлення до перевезення.

Транспортування може здійснюватись різними видами транспорту – автомобільним, залізничним, морським тощо.

У процесі прийняття рішення про перевозку найманим чи власним транспортом оцінюється наступне:

* + загальні витрати;
  + рівень контролю;
  + рівень сервісу;
  + гнучкість.

Після вибору виду транспорту й перевізника встановлюється процедура оцінки якості транспортних послуг, яка включає в себе:

* + аналіз витрат;
  + облік пошкоджень продукції;
  + оцінку своєчасності доставки;
  + інші фактори.

Транспортна логістика вирішує наступні задачі:

* + спільне планування транспортних процесів зі складськими і виробничими операціями;
  + спільне планування транспортних процесів на різних видах транспорту;
  + забезпечення технологічної єдності транспортно-складського процесу;
  + визначення раціональних маршрутів поставки;

Усі ці завдання взаємопов’язані і вирішуються у комплексі.

**Логістичний провайдер** – компанія, яка виконує частину або весь комплекс логістики для свого клієнта. Послуги логістичних провайдерів спрямовані на загальне управління транспортуванням і зберіганням вантажів.

### Опис процесу діяльності

Логістичний провайдер працює таким чином.

Замовник надає інформацію про об’єми виробництва та транспортні засоби своєї компанії менеджеру по роботі з клієнтами. Менеджер обробляє цю інформацію і представляє її у відповідному вигляді логісту. Той, в свою чергу, на основі отриманої інформації складає план перевезень і формує звіт, що містить цей план. Менеджер по роботі з клієнтами передає звіт замовникові.

Послідовність виконуваних дій для процесу діяльності логістичного провайдера наведена у додатку В, лист 1 (Схема структурна діяльності).

### Опис функціональної моделі

Для проектування діаграми використання спочатку необхідно визначити дійових осіб (акторів), а потім визначити, які дії у системі може виконувати кожен з акторів, та їх взаємодію.

Акторами у системі є:

* менеджер по роботі з клієнтами, який приймає замовлення, веде переговори із замовником, передає необхідну інформацію логісту;
* спеціаліст із логістики (логіст), який виконує обробку інформації, складає план перевезень і формує звіт для клієнта.

Структурна схема взаємодії акторів наведена у додатку В, лист 2 (Схема структурна варіантів використання).

Розпишемо детальніше варіанти використань:

1. отримання інформації від замовника – менеджер по роботі із клієнтами отримує первинну інформацію про замовлення від самого замовника, уточнює всі нюанси замовлення;
2. реєстрація замовлення – менеджер по роботі із клієнтами реєструє замовлення після детального обговорення із замовником;
3. ведення бази даних – менеджер по роботі із клієнтами працює із БД, він вносить нову інформацію про замовлення, коригує та видаляє вже існуючу;
4. приведення даних до придатного для логіста вигляду – менеджер по роботі із клієнтами формулює інформацію про замовлення у термінах логістики, таким чином, щоб логіст зміг провести математичну обробку даних;
5. отримання інформації для обробки – логіст отримує дані про замовлення, за якими має бути створений оптимальний план перевезення;
6. складання плану перевезень – логіст аналізує отримані дані і виконує наступні дії:
   1. розподіл споживачів між транспортними засобами – логіст приписує кожному транспортному засобу клієнтів, яким він має доставити продукцію;
   2. складання маршрутів для транспортних засобів – відповідно до виконаного розподілення клієнтів між транспортними засобами, для кожного із них визначається послідовність обслуговування клієнтів, які до нього приписані;
   3. визначення необхідної кількості транспортних засобів – при складанні плану логісту іноді необхідно визначити кількість транспортних засобів необхідних для перевезень;
7. формування звіту – логіст складає звіт, в якому надається уся необхідна клієнтові інформація щодо плану перевезень;
8. надання звіту замовникові – менеджер по роботі з клієнтами передає звіт замовникові.

## Огляд наявних аналогів

У ході дослідження предметної області були виявлені програмні продукті, що вирішують аналогічні задачі. Розглянемо деякі з них:

* TransTrade – програма для автоматизації транспортних підприємств, а також будь-яких відділів транспортної логістики, чия діяльність, так чи інакше, пов’язана з вирішенням транспортних задач та здійсненням перевезень (детальніша інформація про програмний продукт наведена на сайті розробника [4]);
* proLOG – повноцінний програмний комплекс для автоматизації підприємств, що працюють в сегменті логістики, управлінні перевезеннями. Програма дозволяє автоматизувати всі бізнес-процеси транспортної компанії, діяльність якої пов’язана з перевезенням вантажів (детальніша інформація про програмний продукт наведена на сайті розробника [5]);
* 1С-Рарус:Транспортная логистика и экспедирование – програмний комплекс, призначений для автоматизації управління процесом перевезення в компаніях, що займаються наданням послуг з експедирування вантажів різними видами транспорту: автомобільним, залізничним, авіаційним, морським. Даний продукт може використовуватися в компаніях, які мають власні транспортні засоби або ні (детальніша інформація про програмний продукт наведена на сайті розробника [6]).

## Постановка задачі

### Призначення розробки

Призначенням комплексу задач є забезпечення логіста засобами автоматизованого створення плану перевезень продукції з мінімальними сумарними витратами (витрати на перевезення, збитки пов’язані з неврахуванням пріоритетності замовлень тощо).

### Цілі та задачі розробки

Метою комплексу задач є зменшення сумарних витрат на перевезення продукції та збитків, пов’язаних із неврахуванням пріоритетів замовлень.

Для досягнення поставленої мети мають бути вирішені наступні задачі:

* облік складів;
* облік споживачів;
* облік транспортних засобів;
* облік відстаней прямих (безпосередніх) переїздів від складів до споживачів;
* облік відстаней прямих (безпосередніх) переїздів між споживачами;
* складання плану перевезень малогабаритної продукції за умов використання усіх наявних транспортних засобів;
* складання плану перевезень однорідної продукції з урахуванням вантажомісткості транспортних засобів та визначення необхідної кількості транспортних засобів для здійснення цих перевезень;
* складання плану перевезень однорідної продукції з урахуванням вантажомісткості транспортних засобів, обмеженості автопарку та пріоритетів замовлень;
* формування звіту, що містить план перевезень продукції з мінімальними сумарними витратами.

У свою чергу для розв’язання задач складання плану перевезень мають бути у свою чергу розв’язані наступні задачі:

* розподілення клієнтів між транспортними засобами;
* складання маршрутів для транспортних засобів відповідно до виконаного розподілення.

Розподіл задач комплексного дипломного проекту на індивідуальні частини виглядає наступним чином:

* індивідуальна частина № 1 – складання плану перевезень однорідної продукції з урахуванням вантажомісткості транспортних засобів та визначення необхідної кількості транспортних засобів для здійснення цих перевезень;
* індивідуальна частина № 2 – складання плану перевезень продукції з урахуванням вантажомісткості та обмеженої кількості транспортних засобів і пріоритетів замовлень.

## Висновок до розділу

У даному розділі описані загальні положення дипломного проекту. Надано детальний опис предметного середовища поставленої задачі. За допомогою діаграми активності зображено процес діяльності логістичного провайдера. Опис функціональної моделі здійснено за допомогою діаграми варіантів використання, на якій зображені основні актори та їхні дії.

Також надано огляд наявних програмних продуктів, що вирішують аналогічні даному комплексу задач проблеми.

У кінці розділу визначене призначення та мета комплексу задач, а також перелічені задачі, які необхідно розв’язати для досягнення встановленої мети.

# ІНФОРМАЦІЙНЕ ЗАБЕЗПЕЧЕННЯ

## Вхідні дані

Первинні дані у базу даних заносить користувач за допомогою графічного інтерфейсу. Вводиться інформація по складам, споживачам, наявним транспортним засобам, відстань між складами і споживачами.

Дані по складам:

* географічне положення складу;
* об’єм продукції на складі;
* місткість складу.

Дані по споживачам:

* географічне положення споживача;
* об’єм замовленої продукції.

Дані по транспортним засобам:

* місткість транспортного засобу;
* кількість пального, що він витрачає на перевезення продукції на одну одиницю відстані;
* кількість транспортних засобів такого типу в наявності.

Дані про взаємне розташування міст:

* безпосередні відстані між споживачами;
* безпосередні відстані від складів до споживачів.

## Вихідні дані

Вихідними даними комплексу є звіт із планом перевезень продукції, який містить маршрути для кожного транспортного засобу, їх довжину, витрати на перевезення, об’єми перевезення по транспортному засобу і сумарні штрафи.

## Опис структури бази даних

У таблицях 2.1 – 2.5 наведено детальний опис кожної із таблиць бази даних.

Таблиця 2.1 – Таблиця складів

|  |  |  |  |
| --- | --- | --- | --- |
| **Назва таблиці** | **Назва стовпця** | **Тип даних** | **Детальна інформація** |
| DEPOT – таблиця, що містить інформацію про склади | ID | numeric | Ідентифікатор складу |
| NAME | nvarchar | Назва складу |
| AMOUNT | numeric | Об’єм продукції на складі |
| VOLUME | numeric | Місткість складу |
| COORDINATES\_N | float | Координати складу (у градусах північної широти) |
| COORDINATES\_E | float | Координати складу (у градусах східної довготи) |

Таблиця 2.2 – Таблиця споживачів

| **Назва таблиці** | **Назва стовпця** | **Тип даних** | **Детальна інформація** |
| --- | --- | --- | --- |
| CONSUMER – таблиця, що містить інформацію про споживачів | ID | numeric | Ідентифікатор споживача |
| NAME | nvarchar | Назва споживача |
| AMOUNT | numeric | Об’єм замовлення |
| COORDINATES\_N | float | Координати споживача (у градусах північної широти) |

Продовження табл. 2.2

|  |  |  |  |
| --- | --- | --- | --- |
| **Назва таблиці** | **Назва стовпця** | **Тип даних** | **Детальна інформація** |
|  | COORDINATES\_E | float | Координати споживача (у градусах східної довготи) |
| FINE | numeric | Збиток при несвоєчасному обслуговуванні споживача |

Таблиця 2.3 – Таблиця транспортних засобів

|  |  |  |  |
| --- | --- | --- | --- |
| **Назва таблиці** | **Назва стовпця** | **Тип даних** | **Детальна інформація** |
| VEHICLE – таблиця, що містить інформацію про транспортні засоби | ID | numeric | Ідентифікатор транспортного засобу |
| NAME | nvarchar | Назва транспортного засобу |
| CONSUMPTION | float | Витрати палива |
| COUNT | numeric | Кількість транспортних засобів такого типу |
| VOLUME | numeric | Місткість транспортного засобу |

Таблиця 2.4 – Таблиця відстаней від складів до споживачів

|  |  |  |  |
| --- | --- | --- | --- |
| **Назва таблиці** | **Назва стовпця** | **Тип даних** | **Детальна інформація** |
| DISTANCE\_DEPOT\_CONSUMER – таблиця, що містить дані про відстані між складами і споживачами | ID | numeric | Ідентифікатор |
| ID\_DEPOT\_FROM | numeric | Ідентифікатор пункту відправлення (складу) |
| ID\_CONSUMER\_TO | numeric | Ідентифікатор пункту призначення (споживача) |
| DISTANCE | float | Відстань між пунктом відправлення і пунктом призначення |

Таблиця 2.5 – Таблиця відстаней між споживачами

|  |  |  |  |
| --- | --- | --- | --- |
| **Назва таблиці** | **Назва стовпця** | **Тип даних** | **Детальна інформація** |
| DISTANCE\_CONSUMERS – таблиця, що містить дані про відстані між споживачами | ID | numeric | Ідентифікатор |
| ID\_CONSUMER\_FROM | numeric | Ідентифікатор пункту відправлення (споживача) |
| ID\_CONSUMER\_TO | numeric | Ідентифікатор пункту призначення (споживача) |
| DISTANCE | float | Відстань між пунктом відправлення і пунктом призначення |

Взаємозв’язки між таблицями бази даних зображені на фізичній моделі даних представленій у додатку В, лист 3 (Схема бази даних).

## Структура масивів інформації

Експорт даних із БД виконується у XML-файл, інформація із якого далі використовується програмою для математичної обробки. У додатку В, лист 4 (Схема структурна XML) представлена модель XML.

Розглянемо елементи моделі більш детально.

SerializationData – представляє собою всю інформацію, яку необхідно зберігати. Цей об’єкт зберігає елемент Nodes і Distances.

Nodes у свою чергу є колекцією об’єктів Node (вузол).

Node має наступні властивості:

* Id – ідентифікатор вузла;
* Name – назва вузла;
* Type – тип вузла (Depot – склад, Consumer – споживач, Auxiliary – допоміжний);
* Volume – об’єм замовлення;
* Fine – збитки (штраф) пов’язані із відмовою від термінового обслуговування вузла;
* ScreenPosition – координати відображення вузла на екрані;
* RealPosition – реальні координати вузла, які використовуються при обчисленнях.

Distances є колекцією об’єктів Distance (відстань).

Distance має наступні властивості:

* Id1 – ідентифікатор вузла 1;
* Id2 – ідентифікатор вузла 2;
* Value – довжина (вартість) безпосереднього переїзду між вузлами 1 та 2;

## Висновок до розділу

У даному розділі був приведений структурний опис первісних даних системи, запропонованого звіту у якості вихідних даних.

Відповідно до структури вхідних та вихідних даних розроблена структура бази даних, що складається з 5 таблиць. Наведений детальний опис структури таблиць бази та призначення полів таблиць.

Також розроблена структура XML-файлу, який використовується при експорті даних з БД у застосунок для математичної обробки інформації.

# МАТЕМАТИЧНЕ ЗАБЕЗПЕЧЕННЯ

Даний комплексний дипломний проект присвячений складанню плану перевезень продукції за різних умов: перевезення малогабаритної продукції, перевезення продукції із урахуванням вантажомісткості транспортних засобів та визначення необхідної кількості транспортних засобів для здійснення цих перевезень, перевезення із урахуванням вантажомісткості та обмеженої кількості транспортних засобів і пріоритетів замовлень. 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’язання є основою для розв’язання інших задач перевезення продукції, у яких враховуються різноманітні обмеження.

## Змістовна постановка задачі

Опишемо стандартну постановку задачі. Логістичний провайдер 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 [7]). 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. У [8] виділені наступні найбільш важливі та суттєві 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.

Згідно [9] 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 . Парк 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 запропонована у [8].

Точні методи розв’язання забезпечують знаходження оптимального розв’язку. Серед 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 (2005 рік) 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 [10], 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 знайдені 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 зупину (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**

Відповідно до [11] задача кластеризації (Data clustering) – це 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 об’єктами , і маємо скінченну 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 квадратичного 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 [12] – найпопулярніший метод кластеризації. Дія алгоритму спрямована на мінімізацію сумарного квадратичного 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 відповідно 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).

## Порівняльний аналіз алгоритмів

Метою даного дипломного проекту була не просто розробка методу розв’язання задачі складання плану перевезень продукції із ціллю зменшення витрат на перевезення, а розробка ефективного методу розв’язку, тобто такого, що буде знаходити прийнятні розв’язки за прийнятний час. Тому було розроблено декілька алгоритмів, і у цьому випадку важливим пунктом є порівняльний аналіз створених алгоритмів.

Аналіз алгоритмів був проведений на основі експериментальних даних, отриманих для задач із різними початковими вхідними даними. Аналіз алгоритмів проводився за двома напрямами: точність розв’язку, який він забезпечує, та час за який цей розв’язок було отримано.

У таблиці 3.1 наведені результати проведення експериментів.

Таблиця 3.1 – Результати застосування розроблених алгоритмів на різних моделях задачі

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Кількість замовників** | **Зведення до ЗК і розв’язання бджолиним алгоритмом** | | **2-х етапний метод із застосуванням бджолиного алгоритму для кластеризації** | | **2-х етапний метод із застосуванням методу k-середніх** | |
| **Розв’язок** | **Час** | **Розв’язок** | **Час** | **Розв’язок** | **Час** |
| Кількість транспортних засобів – 3 шт. | | | | | | |
| 10 | 2512,09 | 0,03 | 2610 | 0,03 | 3021,76 | 0,01 |
| 15 | 2236,86 | 0,08 | 2547,71 | 0,05 | 2987,75 | 0,01 |
| 20 | 2558,09 | 0,18 | 2880,43 | 0,11 | 3269,53 | 0,02 |
| 25 | 2907,12 | 0,5 | 3343,77 | 0,14 | 3777,25 | 0,04 |
| 30 | 2922,25 | 1,01 | 3228,01 | 0,31 | 3743,44 | 0,09 |
| 35 | 3158,23 | 2,15 | 3554,88 | 0,4 | 3756,29 | 0,14 |
| 40 | 3468,75 | 3,42 | 3798 | 0,5 | 4006,35 | 0,19 |
| 45 | 3474,15 | 7,67 | 3819,79 | 0,73 | 3898,49 | 0,45 |
| 50 | 3618,47 | 10,79 | 4337,9 | 1,31 | 4503,15 | 0,48 |
| 55 | 3890,59 | 17,07 | 4403,73 | 1,24 | 4740,77 | 0,47 |
| 60 | 3962,97 | 20,5 | 4643,13 | 1,18 | 4832,77 | 0,91 |
| 65 | 3957,44 | 44,3 | 4382,57 | 1,87 | 4542,73 | 1,16 |
| 70 | 4096,65 | 49,65 | 4594,37 | 2,77 | 4697,78 | 1,82 |
| 75 | 4299,83 | 80,51 | 4836,1 | 3,67 | 4887,63 | 2,35 |
| 80 | 4372,40 | 94,87 | 4838,96 | 3,44 | 4916,62 | 2,72 |

Продовження табл. 3.1

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Кількість замовників** | **Зведення до ЗК і розв’язання бджолиним алгоритмом** | | **2-х етапний метод із застосуванням бджолиного алгоритму для кластеризації** | | **2-х етапний метод із застосуванням методу k-середніх** | |
| **Розв’язок** | **Час** | **Розв’язок** | **Час** | **Розв’язок** | **Час** |
| Кількість транспортних засобів – 5 шт. | | | | | | |
| 65 | 4546,56 | 25,39 | 5266,03 | 1,23 | 5609,95 | 0,35 |
| 70 | 4307,77 | 56,27 | 5415,77 | 1,73 | 5974,05 | 0,34 |
| 75 | 4481,12 | 46,84 | 5433,66 | 2,03 | 5779,68 | 0,67 |
| 80 | 4575,53 | 75,88 | 5381,05 | 1,93 | 5581,54 | 0,48 |
| 85 | 4579,40 | 113,97 | 5514,45 | 2,26 | 5707,67 | 0,74 |
| 90 | 4835,31 | 138,66 | 5976,56 | 2,34 | 6239,57 | 0,85 |
| 95 | 4764,09 | 132,51 | 5978,07 | 3,34 | 6786,85 | 1,23 |
| 100 | 5220,65 | 215,74 | 6136,65 | 3,06 | 6451,75 | 1,2 |
| 105 | 5166,59 | 332,26 | 6047,36 | 5,58 | 6338,73 | 1,29 |
| 110 | 4995,08 | 327,24 | 6153,87 | 4,32 | 6345,4 | 1,79 |
| 115 | 5528,86 | 338,28 | 6779,86 | 7,78 | 7224,73 | 2,15 |
| 120 | 5447,54 | 551,56 | 6339,9 | 6,7 | 6591,23 | 2,22 |
| Кількість транспортних засобів – 6 шт. | | | | | | |
| 100 | 5481,03 | 248,39 | 6566,53 | 4,52 | 6960,85 | 1,2 |
| 105 | 5473,81 | 394,97 | 6414,17 | 6,02 | 6635,63 | 1,75 |
| 110 | 5243,56 | 316,28 | 6575,17 | 7,19 | 7050,91 | 1,02 |
| 115 | 5566,37 | 374,02 | 7120,49 | 6,06 | 7842,02 | 1,1 |
| 120 | 5550,94 | 539,63 | 6890,56 | 6,54 | 7451,55 | 1,28 |
| 125 | 5629,73 | 627,98 | 6602,47 | 7,44 | 6692,69 | 1,3 |
| 130 | 5824,95 | 619,65 | 6852,53 | 10,57 | 7521,91 | 2,14 |
| 135 | 6030,11 | 762,78 | 7391,67 | 12,1 | 7742,65 | 2,86 |
| 140 | 5896,70 | 842,47 | 7243,95 | 13,61 | 7724,8 | 3,41 |

Продовження табл. 3.1

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Кількість замовників** | **Зведення до ЗК і розв’язання бджолиним алгоритмом** | | **2-х етапний метод із застосуванням бджолиного алгоритму для кластеризації** | | **2-х етапний метод із застосуванням методу k-середніх** | |
| **Розв’язок** | **Час** | **Розв’язок** | **Час** | **Розв’язок** | **Час** |
| Кількість транспортних засобів – 10 шт. | | | | | | |
| 30 | 4935,22 | 2,19 | 5167,11 | 0,56 | 6370,75 | 0,04 |
| 35 | 4508,91 | 3,91 | 4713,16 | 0,87 | 7172,7 | 0,05 |
| 40 | 4932,35 | 6,25 | 5245,6 | 0,88 | 7090,22 | 0,06 |
| 45 | 5217,83 | 10,26 | 5533,11 | 1,13 | 7774,84 | 0,06 |
| 50 | 5454,29 | 13,12 | 5993,52 | 1,73 | 8813,51 | 0,06 |
| 55 | 4956,22 | 24,18 | 5557,05 | 2,15 | 8564,89 | 0,08 |
| 60 | 4959,80 | 36,86 | 5609,48 | 2,82 | 7126,91 | 0,08 |
| 65 | 4995,76 | 47,37 | 5626,19 | 2,52 | 7323,53 | 0,09 |
| 70 | 5100,11 | 75,43 | 5789,63 | 3,4 | 7621,55 | 0,13 |

У додатку В, лист 1 (Графіки порівняння роботи алгоритмів для розв’язання VRP) наведені порівняльні графіки застосування розроблених алгоритмів для розв’язання задачі маршрутизації транспортних засобів із різноманітними вхідними даними.

Із графіків стає зрозуміло, що чим кращий розв’язок знаходить алгоритм, тим більше часу він потребує на знаходження цього розв’язку.

Найкращим з точки зору точності знайденого розв’язку виявився метод зведення задачі маршрутизації транспортних засобів до задачі комівояжера, яка в свою чергу розв’язується бджолиним алгоритмом. Але основним недоліком даного методу був навіть не затрачений на пошук розв’язку час, а те, що у знайденому розв’язку неоднорідно розподілялось навантаження на окремі транспортні засоби. Наприклад, одному транспортному засобу призначався маршрут довжиною 100 км, а іншому – 10 км.

На рисунку 3.11 наведений приклад неоднорідного розподілу замовлень між транспортними засобами.

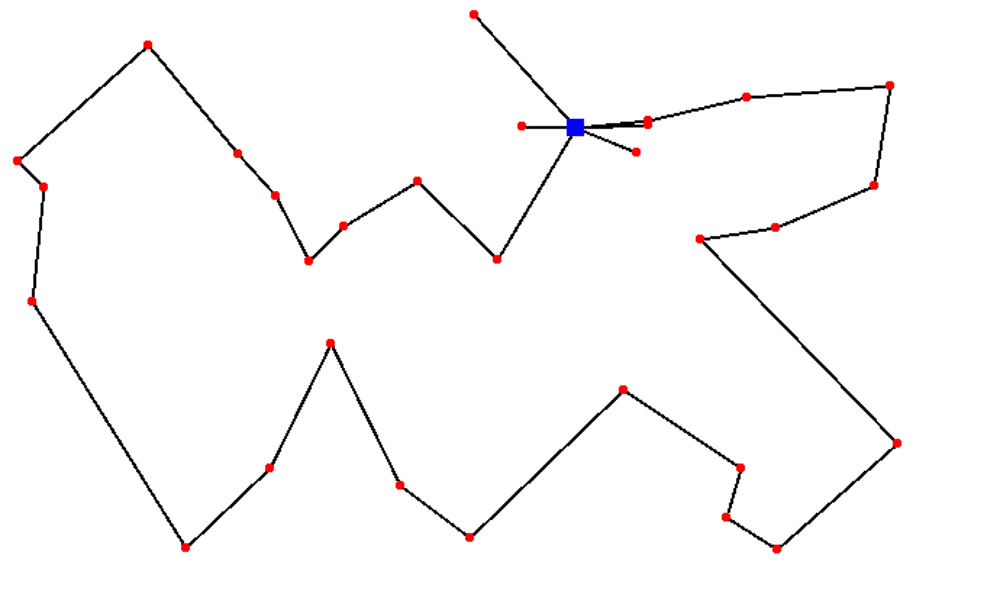


Рисунок – 3.11 Приклад неоднорідного розподілу замовлень між транспортними засоби при розв’язанні VRP зведенням до ЗК

З точки зору прикладного застосування метод зведення задачі маршрутизації транспортних засобів до задачі комівояжера із подальшим розв’язанням бджолиним алгоритмом виявися неефективним.

Тому більшої уваги заслуговують 2-етапні методи розв’язання задачі. У даній роботі розглянуто два 2-етапних методи: один із застосування бджолиного алгоритму для кластеризації, і один із застосування методу k‑середніх для кластеризації

З точки зору точності отриманого результату кращим виявися 2-етапний метод із застосуванням бджолиного алгоритму для кластеризації, а з точки зору затраченого часу – 2-етапний метод із застосуванням методу k‑середніх.

## Висновок до розділу

Розділ з математичного забезпечення присвячений формулюванню змістовної та математичної постановки задачі, розробленню математичного апарату для розв’язання цієї задачі.

У розділі детально описано розроблені методи та алгоритми розв’язання задачі та обґрунтовано їх вибір.

Також важливим пунктом розділу з математичного забезпечення є проведення порівняльного аналізу розроблених алгоритмів із метою визначення найбільш ефективного з них.

# ПРОГРАМНЕ ТА ТЕХНІЧНЕ ЗАБЕЗПЕЧЕННЯ

Опис засобів розробки та вимоги до технічного та програмного забезпечення викладені у 3-ому розділі індивідуальної частини № 2 дипломного проекту.

## Архітектура програмного забезпечення

Для описання архітектури програмного забезпечення використаємо діаграми класів, послідовності та компонентів. Специфікація функцій наведена у індивідуальній частині № 2 дипломного проекту у п. 3.3.1.

### Діаграма класів

У додатку В, лист 5 (Схема структурна класів) наведена структурна схема класів, які відповідають за виконання основних функцій програми: обробка даних, складання плану і маршрутів перевезень.

Діаграма містить наступні класи:

* TaskController – клас, функціями якого є зв’язок користувацького інтерфейсу з класами алгоритмів (запуск, зупинка алгоритму), графічне відображення вузлів, збереження усіх вузлів, робота з файлами;
* Algorithm – клас алгоритму, в якому реалізована спільна для ітеративних алгоритмів логіка: запуск одної ітерації, запуск певної кількості ітерацій, проведення ітерацій, поки не буде досягнуто розв’язку з заданою точністю. Також даний клас має функцію ведення лог-файлу для оцінки ефективності алгоритму;
* BeesColony – клас, в якому реалізовано бджолиний алгоритм в загальному вигляді: початкова ініціалізація, перехід до кращого сусіднього розв’язку, генерація нових випадкових розв’язків;
* ClusteringAlgorithm – реалізує спільну логіку для алгоритмів кластеризації: збереження кластерів, додавання кластера, підрахунок цільової функції;
* KMeans – клас реалізує логіку алгоритму кластеризації k‑середніх;
* NearestNeighbourChain – клас реалізує логіку алгоритму кластеризації nearest-neighbor chain;
* ClusteringToTsp – клас, що складає маршрути обходу кожного з кластерів, що їх було отримано попередньо алгоритмами кластеризації, підраховує кінцеву оцінку витрат на перевезення і штрафи;
* Site – клас, який представляє розв’язок бджолиного алгоритму, а також реалізовує спільну логіку пошуку сусідніх розв’язків і переходу до кращого з них;
* SiteVrpTsp – клас розв’язку задачі VRP, що зводиться до ЗК. Конкретизує логіку пошуку сусідніх розв’язків для цієї задачі, обчислює відстані між вузлами;
* SiteClusteringVrp – клас розв’язку задачі кластеризації для VRP. Конкретизує логіку пошуку сусідніх розв’язків для цієї задачі, обчислює суму квадратів відстаней вузлів від центру кластера для кожного з кластерів;
* SiteClusteringCvrp – клас розв’язку задачі кластеризації для CVRP. Конретизує логіку пошуку сусідніх розв’язків для цієї задачі, зберігає некластеризовані вузли;
* SiteClusteringCvrpp – клас розв’язку задачі кластеризації для CVRPP. Конретизує логіку пошуку сусідніх розв’язків для цієї задачі, проводить оцінку довжини шляху обходу кластерів;
* SiteClusteringCvrppNnc – клас розв’язку задачі кластеризації для CVRPP модифікованим алгоритмом. Проводить кластеризацію за допомогою алгоритму ланцюга найближчого сусіда, і на основі цього генерує нові розв’язки для бджолиного алгоритму;
* Node – клас вузла. Зберігає інформацію, що стосується вузла: ідентифікатор, ім’я, тип, реальну позицію, позицію відображення на екрані, з’єднання з іншими вузлами;
* Connection – клас з’єднання з вузлом. Зберігає приєднаний вузол і колір з’єднання.
* Cluster – клас кластера. Зберігає інформацію, що стосується кластера: вузли, що відносяться до цього кластеру, центр кластеру, місткість кластеру. Реалізує логіку додавання, видалення вузлів, об’єднання з іншим кластером;
* SerializationData – клас, що тимчасово зберігає інформацію про вузли і відстані між ними, а також здійснює зчитування та запис інформації до файлу.

### Діаграма послідовності

Діаграма послідовності процесу ітерації алгоритму бджолиної колонії наведена у додатку В, лист 6 (Схема структурна послідовності процесу ітерації бджолиного алгоритму). Вона відображає послідовність дій, виконувану під час кожної ітерації бджолиного алгоритму, впорядковану у часі.

Діаграма послідовності процесу ітерації алгоритму кластеризації k‑середніх наведена у додатку В, лист 7 (Схема структурна послідовності процесу ітерації алгоритму k-середніх). Вона відображає послідовність дій, виконувану під час кожної ітерації алгоритму k-середніх (визначення вузлів, що належать кластерам, визначення нового центру кластера), впорядковану у часі.

Діаграма послідовності процесу ітерації алгоритму ланцюга найближчого сусіда для кластеризації наведена у додатку В, лист 8 (Схема структурна послідовності процесу ітерації алгоритму ланцюга найближчого сусіда). Вона відображає послідовність дій, виконувану під час кожної ітерації алгоритму ланцюга найближчого сусіда, впорядковану у часі.

### Діаграма компонентів

На рисунку 4.1 наведена діаграма компонентів створеного програмного забезпечення. На ній відображено основні компоненти програмного продукту та зв’язки між ними.

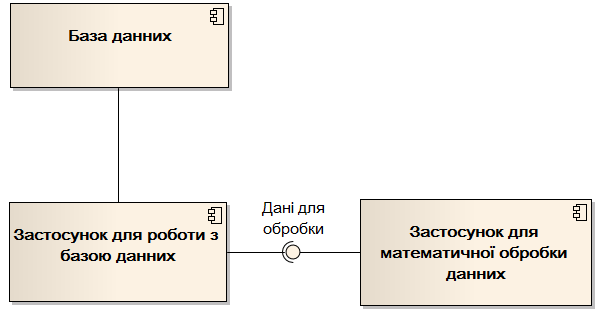


Рисунок 4.1 – Схема структурна компонентів програмного забезпечення

Основними компонентами в системі є база даних, застосунок для роботи з базою даних, застосунок для математичної обробки даних. Застосунок для роботи з БД забезпечує введення інформації до БД та її зчитування і корегування, а також перетворює отриману із БД інформацію до вигляду із яким може працювати застосунок математичної обробки даних. Застосунок математичної обробки даних не має прямого зв’язку з базою даних.

## Опис звітів

За допомогою створеного програмного продукту можливо генерувати звіти із планом перевезень продукції, який містить маршрути для кожного транспортного засобу, їх довжину, витрати на перевезення, об’єми перевезення по транспортному засобу і сумарні штрафи.

Звіт формується автоматично. Для цього необхідно обрати пункт меню “Report – Generate report …”. Після цього на екран буде виведено вікно, яке буде містити текстовий звіт щодо складених маршрутів перевезень. Звіт можна зберегти у файл обравши пункт меню “File – Save report to file …”. На рисунку 4.2 Зображено приклад сформованого звіту.

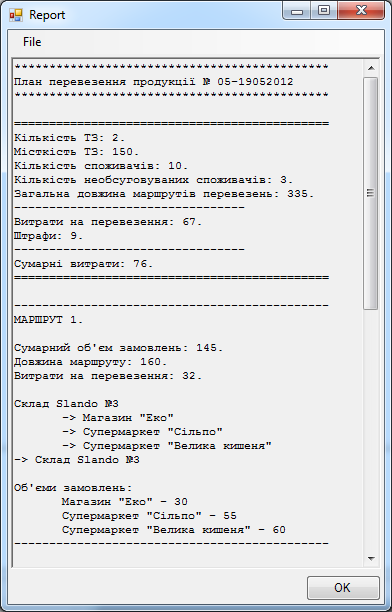


Рисунок 4.2 – Вигляд звіту плану перевезень продукції

## Висновок до розділу

Даний розділ містить інформацію з програмного та технічного забезпечення.

Детально описана архітектура програмного забезпечення, а саме наведений опис діаграми класів, діаграм послідовностей розроблених алгоритмів, і діаграми компонентів, яка відображає структуру розробленого програмного забезпечення.

Мінімальні вимоги до технічного та програмного забезпечення для можливості роботи із розробленим програмним комплексом та опис засобів розробки наведений у ПЗ індивідуальної частини № 2 дипломного проекту.

# 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, що дозволить зменшити витрати на перевезення, а створення якомога більш ефективного методу розв’язання 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 необхідної кількості транспортних засобів для здійснення цих перевезень. У індивідуальній частині № 2 описані дослідження задачі складання плану перевезень із урахуванням вантажомісткості транспортних засобів, їх обмеженої кількості та пріоритетів замовлень.

Для розробки програмного продукту була обрана мова програмування C# як мова розробки застосуванків 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 ПЗ загальної частини дипломного проекту також описана архітектура програмного забезпечення.

# ПЕРЕЛІК ПОСИЛАНЬ

1. Статья “Логистика” [Електронний ресурс] // Режим доступу: http://ru.wikipedia.org/wiki/Логистика
2. Статья “Транспортная логистика» [Електронний ресурс] // Режим доступу: http://ru.wikipedia.org/wiki/Транспортная\_логистика
3. Стаття “Транспортування” [Електронний ресурс] // Режим доступу: http://uk.wikipedia.org/wiki/Транспортування
4. Корпоративний сайт TRANSOFT [Електронний ресурс] // Режим доступу: http://transoft.ru/software/
5. Корпоративний сайт проЛОГ [Електронний ресурс] // Режим доступу: http://www.pro-log.org/
6. Корпоративний сайт ABIS SOFT [Електронний ресурс] // Режим доступу: http://www.abissoft.com/soft/logistics/
7. Article “Vehicle routing problem” [Електронний ресурс] // Режим доступу: [http://en.wikipedia.org/wiki/Vehicle\_routing\_problem](http://en.wikipedia.org/wiki/K-means_clustering)
8. Сайт присвячений задачі маршрутизації транспортних засобів та її варіаціям [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)
9. Bruce Golden. The Vehicle Routing Problem: Last Advances and New Challenges [Текст] // Bruce Golden, S. Raghavan, Edward Wasil // Springer, 2008. – 589 с.
10. Article “Travelling salesman problem” [Електронний ресурс] // Режим доступу: [http://en.wikipedia.org/wiki/Travelling\_salesman\_problem](http://en.wikipedia.org/wiki/K-means_clustering)
11. Article “Cluster analysis” [Електронний ресурс] // Режим доступу: <http://en.wikipedia.org/wiki/Cluster_analysis>
12. Article “k-means clustering” [Електронний ресурс] // Режим доступу: <http://en.wikipedia.org/wiki/K-means_clustering>

# ДОДАТОК А Технічне завдання

**Найменування та сфера застосування**

«Комплекс задач складання плану перевезень продукції». Сферою застосування є логістичні процеси пов’язані із складанням плану перевезень продукції з у рахуванням різних обмежень.

**Призначення розробки**

Призначенням комплексу задач є забезпечення логіста засобами автоматизованого створення плану перевезень продукції з мінімальними сумарними витратами (витрати на перевезення, збитки пов’язані з неврахуванням пріоритетності замовлень тощо).

Метою комплексу задач є зменшення сумарних витрат на перевезення продукції та збитків, пов’язаних із неврахуванням пріоритетів замовлень.

**Вихідні дані**

Вихідними даними комплексу є звіт із планом перевезень продукції, який містить маршрути для кожного транспортного засобу, їх довжину, витрати на перевезення, об’єми перевезення по транспортному засобу і сумарні штрафи.

**Технічні вимоги**

Для можливості роботи з даним програмним продуктом до складу технічних засобів повинні входити: процесор з тактовою частотою не нижче 1.4 ГГц, об’єм оперативної пам’яті не менше 512 МБ, наявність 20 Мб вільного простору на жорсткому диску, графічна карта з об’ємом відео-пам’яті не менше 32 Мб. Додатково має бути встановлене наступне програмне забезпечення: операційна система Windows XP/Vista/7, СУБД Microsoft SQL Server 2008 R2, платформа Microsoft .NET Framework 4.0;

**Етапи розробки**

Вивчення рекомендованої літератури; аналіз існуючих розробок; постановка задачі; розробка інформаційного, програмного та математичного забезпечення; оформлення пояснювальної записки.

# ДОДАТОК Б Тексти програмного коду

|  |
| --- |
| ***Тексти програмного коду програмного комплексу задач складання плану перевезень продукції*** |
| (Найменування програми (документа)) |
| *CD-RW* |
| (Вид носія даних) |
| *??? Мб* |
| (Обсяг програми (документа), арк., Мб) |
| *студента групи ІС-81 IV курсу*  *Воротіліна В.В.* |
| *студента групи ІС-81 IV курсу*  *Головні І.А.* |
| 2012 |

**// Algorithm.cs**

using System;

using System.Collections.Generic;

using System.IO;

using System.Linq;

namespace Diploma

{

public abstract class Algorithm

{

protected List<Node> Nodes;

protected List<Node> Depots;

protected List<Node> Consumers;

protected int DepotsCount;

protected int ConsumersCount;

public int IterationNumber { get; private set; }

public abstract double Value { get; }

protected bool Stopped { get; private set; }

private const double Epsilon = 0.0001;

protected Algorithm()

{

IterationNumber = 0;

}

public void SetNodes(List<Node> nodesForSet)

{

Nodes = new List<Node>();

Depots = (from node in nodesForSet

where node.Type == Node.NodeType.Depot

select node).ToList();

Nodes.AddRange(Depots);

DepotsCount = Depots.Count;

Consumers = (from node in nodesForSet

where node.Type == Node.NodeType.Consumer

select node).ToList();

Nodes.AddRange(Consumers);

ConsumersCount = Consumers.Count;

}

public int LastChangedIteration { get; private set; }

public bool IsCalcLastChange { get; set; }

public string LogFileName { get; set; }

private StreamWriter writer = null;

private void OpenLogFile()

{

if (File.Exists(LogFileName))

{

writer = new StreamWriter(LogFileName, true);

writer.WriteLine(string.Format("Starting new algorithm: {0}. Time: {1}.", GetType().Name, DateTime.Now));

}

}

private double lastValue = double.PositiveInfinity;

private void Iteration()

{

if (Stopped)

return;

if (IterationNumber == 0)

{

OpenLogFile();

}

InnerIteration();

IterationNumber++;

double newValue = double.PositiveInfinity;

if (writer != null || IsCalcLastChange)

{

newValue = Value;

}

if (writer != null && !Stopped && (IterationNumber - 1) % 10 == 0)

{

writer.WriteLine(string.Format("{0}\t{1:0.00}", IterationNumber, newValue));

}

if (IsCalcLastChange && Math.Abs(lastValue - newValue) >= lastValue \* Epsilon)

{

LastChangedIteration = IterationNumber;

}

lastValue = newValue;

}

public void Iterations (int count = 1)

{

for (int i = 0; i != count; i++)

{

Iteration();

}

}

public void IterateToStop()

{

while (!Stopped)

{

Iteration();

}

}

public void Stop()

{

if (Stopped)

{

return;

}

if (writer != null)

{

writer.Close();

}

Stopped = true;

}

protected abstract void InnerIteration();

public abstract void DrawNodes();

public virtual string Info()

{

return "";

}

}

}

**// BeesColony.cs**

using System;

using System.Collections.Generic;

using System.Drawing;

using System.Linq;

using System.Text;

namespace Diploma

{

public class BeesColony : Algorithm

{

public enum ProblemType

{

VRP\_TSP,

CLUSTERING\_VRP,

CLUSTERING\_CVRP,

CLUSTERING\_CVRPP,

CLUSTERING\_CVRPP\_NNC

}

public ProblemType Problem;

public int ScoutsCount;

public int GoodSitesCount;

public int BestSitesCount;

public int NeighboursForGoodSites;

public int NeighboursForBestSites;

public int ClustersCount;

public int ClusterCapacityLimit;

public double KilometerCost;

private List<Site> sites;

private Site CreateNewSite ()

{

switch (Problem)

{

case ProblemType.VRP\_TSP:

return new SiteVrpTsp(Nodes, DepotsCount, ConsumersCount, ClustersCount);

case ProblemType.CLUSTERING\_VRP:

return new SiteClusteringVrp(Nodes, ClustersCount);

case ProblemType.CLUSTERING\_CVRP:

return new SiteClusteringCvrp(Nodes, ClusterCapacityLimit);

case ProblemType.CLUSTERING\_CVRPP:

return new SiteClusteringCvrpp(Nodes, ClusterCapacityLimit, ClustersCount, KilometerCost);

case ProblemType.CLUSTERING\_CVRPP\_NNC:

return new SiteClusteringCvrppNnc(Nodes, ClusterCapacityLimit, ClustersCount, KilometerCost);

default:

return null;

}

}

public void CreateSites ()

{

sites = new List<Site>();

Site.IsStartingInitialized = false;

CreateNewSite().StartingInitialize();

for (int i = 0; i != ScoutsCount; i++)

{

sites.Add(CreateNewSite());

}

}

public Site BestSite

{

get

{

sites.Sort();

return sites[0];

}

}

public override double Value

{

get { return BestSite.Price; }

}

protected override void InnerIteration ()

{

if (IterationNumber > 100 && (IterationNumber < 500 && IterationNumber > LastChangedIteration \* 3.0 || IterationNumber > 500 && IterationNumber > LastChangedIteration \* 1.5))

{

Stop();

return;

}

sites.Sort();

for (int i = 0; i != BestSitesCount; i++)

{

while (sites[i].GoToBestNeighbour(NeighboursForBestSites))

{}

}

for (int i = BestSitesCount; i != GoodSitesCount; i++)

{

while (sites[i].GoToBestNeighbour(NeighboursForGoodSites))

{}

}

for (int i = GoodSitesCount; i != ScoutsCount; i++)

{

sites[i] = CreateNewSite();

}

}

public override void DrawNodes ()

{

BestSite.DrawNodes();

}

public override string Info()

{

string str = "";

switch (Problem)

{

case ProblemType.CLUSTERING\_CVRP:

str = string.Format("Clusters count: {0}", (BestSite as SiteClusteringCvrp).Clusters.Count);

break;

case ProblemType.CLUSTERING\_CVRPP:

str = string.Format("Estimate length: {0:0.000}. Fines: {1}.", (BestSite as SiteClusteringCvrpp).EstimateLength, (BestSite as SiteClusteringCvrpp).Fines);

break;

}

return str;

}

}

}

**// Cluster.cs**

using System.Collections.Generic;

using System.Drawing;

using System.Linq;

namespace Diploma

{

public class Cluster

{

public List<Node> Nodes = new List<Node>();

public Node Depot;

public int CapacityLimit = int.MaxValue;

private bool nodesUpdated = true;

private Node.Point center;

public Node.Point Center

{

get

{

if (!nodesUpdated)

{

return center;

}

if (Nodes.Count == 0)

{

return null;

}

double sx = 0;

double sy = 0;

foreach (Node node in Nodes)

{

sx += node.RealPosition.x;

sy += node.RealPosition.y;

}

center = new Node.Point(sx / Nodes.Count, sy / Nodes.Count);

nodesUpdated = false;

return center;

}

}

public double GetPrice ()

{

Node.Point currentCenter = Center;

if (Nodes.Count == 0)

{

return 0;

}

double distance = 0;

foreach (Node node in Nodes)

{

distance += Node.Point.SqrDistance(node.RealPosition, currentCenter);

}

if (Depot != null)

{

distance += 2 \* Node.Point.SqrDistance(Depot.RealPosition, currentCenter);

}

return distance;

}

public int Volume

{

get

{

int volume = 0;

foreach (Node node in Nodes)

{

volume += node.Volume;

}

return volume;

}

}

public Cluster() {}

public Cluster(Cluster cluster)

{

Nodes = new List<Node>();

foreach (Node node in cluster.Nodes)

{

Nodes.Add(node);

}

Depot = cluster.Depot;

CapacityLimit = cluster.CapacityLimit;

}

public List<Node> GetDrawingNodes(Color connectionsColor)

{

List<Node> drawingNodes = new List<Node>();

if (Nodes.Count == 0)

{

return drawingNodes;

}

Node center = new Node(-1, Node.NodeType.Auxiliary, (int)Center.x, (int)Center.y, Center.x, Center.y);

foreach (Node node in Nodes)

{

center.ConnectTo(node, connectionsColor);

drawingNodes.Add(node);

}

drawingNodes.Add(center);

return drawingNodes;

}

public void Merge(Cluster cluster)

{

Nodes.AddRange(cluster.Nodes);

Nodes = Nodes.Distinct().ToList();

nodesUpdated = true;

}

public void AddNode(Node node)

{

Nodes.Add(node);

nodesUpdated = true;

}

public void AddNodes(List<Node> nodes)

{

if (nodes != null)

{

Nodes.AddRange(nodes);

}

nodesUpdated = true;

}

public void RemoveNodes()

{

Nodes.Clear();

nodesUpdated = true;

}

public void RemoveNode(Node node)

{

Nodes.Remove(node);

nodesUpdated = true;

}

public override string ToString()

{

string str = "{";

foreach (Node node in Nodes)

{

str += node.Id + ",";

}

str += "}";

return str;

}

}

}

**// ClusteringAlgorithm.cs**

using System;

using System.Collections.Generic;

using System.Drawing;

using System.Linq;

using System.Text;

namespace Diploma

{

public abstract class ClusteringAlgorithm : Algorithm

{

public List<Cluster> Clusters { get; protected set; }

protected Node Depot;

protected int ClustersCount;

protected void AddCluster()

{

Cluster newCluster = new Cluster();

newCluster.Depot = Depot;

Clusters.Add(newCluster);

}

protected ClusteringAlgorithm(List<Node> nodes)

{

SetNodes(nodes);

if (Depots.Count != 0)

{

Depot = Depots[0];

}

}

protected ClusteringAlgorithm(List<Node> nodes, int clustersCount)

: this(nodes)

{

ClustersCount = clustersCount;

Clusters = new List<Cluster>();

for (int i = 0; i != ClustersCount; i++)

{

AddCluster();

}

}

public override void DrawNodes()

{

List<Node> drawingNodes = new List<Node>();

for (int i = 0; i != Clusters.Count; i++)

{

Cluster cluster = Clusters[i];

if (cluster.Nodes.Count == 0)

{

continue;

}

Node center = new Node(-1, Node.NodeType.Auxiliary, (int)Clusters[i].Center.x, (int)Clusters[i].Center.y, Clusters[i].Center.x, Clusters[i].Center.y);

if (cluster.Nodes.Count == 0)

{

drawingNodes.Add(center);

continue;

}

foreach (Node node in cluster.Nodes)

{

center.ConnectTo(node, Color.LightGray);

drawingNodes.Add(node);

}

drawingNodes.Add(center);

}

Node depot = Clusters[0].Depot;

if (depot != null)

{

drawingNodes.Add(depot);

}

TaskController.DrawNodes(drawingNodes);

}

public override double Value

{

get

{

double value = 0;

foreach (Cluster cluster in Clusters)

{

value += cluster.GetPrice();

}

return value;

}

}

}

}

**// KMeans.cs**

using System;

using System.Collections.Generic;

using System.Drawing;

using System.Linq;

using System.Text;

namespace Diploma

{

public class KMeans : ClusteringAlgorithm

{

public KMeans(List<Node> nodes, int clustersCount)

: base(nodes, clustersCount)

{

}

private Node.Point[] centers;

private void SetCenters()

{

if (centers == null)

{

centers = new Node.Point[Clusters.Count];

var xs = from node in Nodes

select node.RealPosition.x;

double xMin = xs.Min();

double xMax = xs.Max();

var ys = from node in Nodes

select node.RealPosition.y;

double yMin = ys.Min();

double yMax = ys.Max();

for (int i = 0; i != centers.Length; i++)

{

double cx = TaskController.Rnd.NextDouble() \* (xMax - xMin) + xMin;

double cy = TaskController.Rnd.NextDouble() \* (yMax - yMin) + yMin;

centers[i] = new Node.Point(cx, cy);

}

}

else

{

for (int i = 0; i != centers.Length; i++)

{

if (Clusters[i].Nodes.Count == 0)

continue;

centers[i] = Clusters[i].Center;

}

}

}

private void GenerateClusters()

{

foreach (Cluster cluster in Clusters)

{

cluster.RemoveNodes();

}

foreach (Node node in Nodes)

{

if (node.Type != Node.NodeType.Consumer)

continue;

if (IterationNumber == 0)

{

bool b = false;

foreach (Cluster cluster in Clusters)

{

if (cluster.Nodes.Count == 0)

{

cluster.AddNode(node);

b = true;

break;

}

}

if (b)

continue;

}

double minDist = double.PositiveInfinity;

Cluster closest = Clusters[0];

for (int i = 0; i < centers.Length; i++)

{

double dist = Node.Point.SqrDistance(centers[i], node.RealPosition);

if (dist < minDist)

{

minDist = dist;

closest = Clusters[i];

}

}

closest.AddNode(node);

}

}

protected override void InnerIteration()

{

if (IterationNumber - LastChangedIteration > 2)

{

Stop();

return;

}

SetCenters();

GenerateClusters();

}

}

}

**// NearestNeighbourChain.cs**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

namespace Diploma

{

public class NearestNeighbourChain : ClusteringAlgorithm

{

private readonly int capacityLimit;

public NearestNeighbourChain(List<Node> nodes, int capacityLimit)

: base(nodes)

{

this.capacityLimit = capacityLimit;

GenerateClusters();

lastReviewedCluster = -1;

}

private void GenerateClusters()

{

Clusters = new List<Cluster>();

for (int i = 0; i != Consumers.Count; i++)

{

AddCluster();

Clusters[i].AddNode(Consumers[i]);

Clusters[i].CapacityLimit = capacityLimit;

}

}

private int lastReviewedCluster;

private int NearestCluster(int cluster)

{

double minDistance = double.PositiveInfinity;

int result = -1;

for (int i = 0; i != Clusters.Count; i++)

{

if (i == cluster)

continue;

double dist = Node.Point.SqrDistance(Clusters[i].Center, Clusters[cluster].Center);

if (dist < minDistance)

{

minDistance = dist;

result = i;

}

}

return result;

}

private bool merged = false;

protected override void InnerIteration()

{

lastReviewedCluster++;

if (lastReviewedCluster >= Clusters.Count)

{

if (!merged)

{

Stop();

return;

}

lastReviewedCluster = 0;

merged = false;

}

if (Clusters.Count <= 1)

{

return;

}

int startingCluster = lastReviewedCluster;

int currentCluster = startingCluster;

int lastCluster = startingCluster;

do

{

int nearest = NearestCluster(currentCluster);

if (nearest == lastCluster)

{

break;

}

lastCluster = currentCluster;

currentCluster = nearest;

} while (true);

if (Clusters[lastCluster].Volume + Clusters[currentCluster].Volume <= Clusters[lastCluster].CapacityLimit)

{

Clusters[lastCluster].Merge(Clusters[currentCluster]);

Clusters.RemoveAt(currentCluster);

merged = true;

}

else if (Clusters[lastCluster].Volume + Clusters[currentCluster].Volume <= Clusters[currentCluster].CapacityLimit)

{

Clusters[currentCluster].Merge(Clusters[lastCluster]);

Clusters.RemoveAt(lastCluster);

merged = true;

}

}

public override string Info()

{

return string.Format("Clusters count: {0}. Stopped: {1}", Clusters.Count, Stopped);

}

}

}

**// Site.cs**

using System;

using System.Collections.Generic;

using System.Drawing;

using System.Linq;

using System.Text;

namespace Diploma

{

public abstract class Site : IComparable

{

protected List<Node> Nodes;

public List<Node> DrawingNodes { get; protected set; }

protected Site (List<Node> nodes)

{

Nodes = new List<Node>();

for (int i = 0; i != nodes.Count; i++)

{

Nodes.Add(nodes[i]);

}

}

public abstract double Price { get; }

protected abstract void GoToNeighbour(Site site);

protected abstract Site GetNeighbour();

public abstract object Result { get; }

private List<Site> GenerateNeighbours(int count)

{

List<Site> result = new List<Site>();

for (int i = 0; i != count; i++)

{

result.Add(GetNeighbour());

}

return result;

}

public bool GoToBestNeighbour(int countOfNeightbours)

{

List<Site> neighbours = GenerateNeighbours(countOfNeightbours);

neighbours.Sort();

if (neighbours[0].Price < this.Price)

{

GoToNeighbour(neighbours[0]);

return true;

}

return false;

}

public static bool IsStartingInitialized { protected get; set; }

public virtual void StartingInitialize()

{

IsStartingInitialized = true;

}

public abstract List<Node> PrepareToDraw(Color connectionsColor);

public void DrawNodes()

{

DrawingNodes = PrepareToDraw(TaskController.ConnectionsPen.Color);

TaskController.DrawNodes(DrawingNodes);

}

public virtual int CompareTo(object obj)

{

Site site = obj as Site;

return Price.CompareTo(site.Price);

}

}

}

**// ClusteringToTsp.cs**

using System;

using System.Collections.Generic;

using System.Drawing;

using System.Linq;

using System.Text;

namespace Diploma

{

class ClusteringToTsp : Algorithm

{

private List<Cluster> clusters;

private List<Node> notClusteredNodes;

private List<double> values;

private List<BeesColony> colonies;

private double kilometerCost;

public ClusteringToTsp (List<Cluster> clusters, List<Node> notClusteredNodes, double kilometerCost)

{

this.clusters = clusters;

this.notClusteredNodes = notClusteredNodes;

this.kilometerCost = kilometerCost;

}

public void Calculate(int scoutsCount, int goodSitesCount, int bestSitesCount, int neighboursForGoodSites, int neighboursForBestSites)

{

values = new List<double>();

colonies = new List<BeesColony>();

foreach (Cluster cluster in clusters)

{

if (cluster.Nodes.Count == 0)

{

continue;

}

BeesColony colony = new BeesColony();

colony.Problem = BeesColony.ProblemType.VRP\_TSP;

colony.ClustersCount = 1;

colony.IsCalcLastChange = true;

colony.ScoutsCount = scoutsCount;

colony.GoodSitesCount = goodSitesCount;

colony.BestSitesCount = bestSitesCount;

colony.NeighboursForGoodSites = neighboursForGoodSites;

colony.NeighboursForBestSites = neighboursForBestSites;

List<Node> colonyNodes = new List<Node>();

colonyNodes.AddRange(cluster.Nodes);

if (cluster.Depot != null)

{

colonyNodes.Add(cluster.Depot);

}

colony.SetNodes(colonyNodes);

colony.CreateSites();

colony.IterateToStop();

values.Add(colony.Value);

colonies.Add(colony);

}

Stop();

}

private double SumValue

{

get

{

if (values == null)

{

return -1;

}

return values.Sum();

}

}

public override double Value

{

get

{

if (notClusteredNodes == null)

{

return SiteClusteringCvrpp.GetPrice(SumValue, kilometerCost, 0);

}

double fines = (from node in notClusteredNodes

select node.Fine).Sum();

return SiteClusteringCvrpp.GetPrice(SumValue, kilometerCost, fines);

}

}

protected override void InnerIteration()

{

//throw new NotImplementedException();

}

public override void DrawNodes()

{

List<Node> drawingNodes = new List<Node>();

for (int i = 0; i != colonies.Count; i++)

{

drawingNodes.AddRange(colonies[i].BestSite.PrepareToDraw(TaskController.GetDrawingColor(i)));

}

if (notClusteredNodes != null)

{

drawingNodes.AddRange(notClusteredNodes);

}

TaskController.DrawNodes(drawingNodes);

}

}

}

# ДОДАТОК В Графічний матеріал