КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ТАРАСА ШЕВЧЕНКА
Факультет комп'ютерних наук та кібернетики
Кафедра теоретичної кібернетики

Кваліфікаційна робота
на здобуття ступеня бакалавра
за спеціальністю 122 Комп'ютерні науки
на тему:

ЗАДАЧА ТРАНСПОРТУВАННЯ РЕСУРСІВ СПОЖИВАЧАМ

Виконав студент 4-го курсу
Костянтин МИРОНЮК

(підпис)

Науковий керівник:
доцент кафедри теоретичної кібернетики
кандидат фіз.-мат. наук
Андрій СТАВРОВСЬКИЙ

(підпис)

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

Студент

(підпис)

Роботу розглянуто й допущено до захисту на засіданні кафедри теоретичної кібернетики
«1» червня 2022р., протокол № 11
Завідувач кафедри
доктор фіз.-мат. наук, професор
Юрій КРАК

(підпис)

КИЇВ - 2022

РЕФЕРАТ

Обсяг роботи складає 45 сторінок, в ній містяться 5 ілюстрацій, 13 формул, використано 9 джерел посилань.

ВІЗУАЛІЗАЦІЯ, ЛІНІЙНЕ ПРОГРАМУВАННЯ, МЕТОД ПОТЕНЦІАЛІВ, МІНІМІЗАЦІЯ ВИТРАТ, ОПТИМІЗАЦІЯ, РОЗПОДІЛЬНИЙ МЕТОД, ТРАНСПОРТНА ЗАДАЧА, ТРАНСПОРТУВАННЯ РЕСУРСІВ.

Об’єктом роботи є процес розв’язування транспортної задачі, суть якого полягає у пошуку оптимального плану транспортування ресурсів від постачальників до споживачів. Розв’язок виконується за допомогою розробленого програмного засобу. Предметом роботи є програмний застосунок, що отримує задану користувачем інформацію про постачальників, споживачів, ціни на перевезення ресурсів і на основі даної інформації розв’язує відповідну транспортну задачу разом із проміжними кроками.

Метою роботи є аналіз підходів в теорії транспортування та створення програмного засобу для розв’язання і візуалізації задач даної галузі.

Інструменти розробки: мова програмування C#, інструмент розробки Unity, а також середовище Visual Studio.

Результати роботи: проведений теоретичний аналіз підходів вирішення задачі транспортування ресурсів споживачам. На основі теоретичних даних реалізовано програмний засіб, що виконує розв’язання даної задачі разом з візуалізацією кроків та результату. Окрім, цього було проведено порівняння швидкості виконання різних методів розв’язку.

Створений застосунок можна застосувати у таких сферах: підприємництво, виробництво, навчання.

ЗМІСТ

ВСТУП

Оцінка сучасного стану об’єкту. Напевно, кожне виробництво зіткається з проблемою постачання ресурсів споживачам. Правильна організація логістики та постачання є запорукою успішних процесів в будь-якій промисловості. Наразі, багато людей вирішують таке завдання стихійно, вручну, хоча очевидно, що набагато ефективніше в даному випадку використовувати програмне забезпечення. Автоматизація і інтеграція інформаційних технологій в цій сфері спонукають нас до нового погляду на підходи до рішення теорії транспортування та дозволяють нам створювати нові програмні продукті, що виконують такі задачі.

Формулюванням і вирішенням оригінальної задачі займались та внесли великий вклад такі відомі люди як Леонід Канторович, Гаспар Монж, Джорж Данціг, Френк Гічкок та інші. Активізація досліджень була обумовлена різними причинами, часто через те, що їх можна було використати в воєнній сфері. Наприклад, Монж сформулював проблему оптимального транспортування в 1781, будучи вмотивованим її воєнним застосуванням, а Канторович зіграв важливу роль в забезпеченні правильного розподілу ресурсів під час Другої світової войни.

Актуальність роботи та підстави для її виконання. Часто виникає ситуація, коли в певній промисловості треба оптимально розподілити ресурси, розрахувати обсяг постачань, задовольнити потреби. Отже, необхідно мати програмний засіб, який буде просто і ефекти виконувати це завдання.

Основна мета й завдання роботи. Метою роботи є розробка застосунку для розрахунку оптимального плану перевезень ресурсів постачальниками споживачам. Для того, щоб досягнути дану мету було складено наступний план завдань:

- Дослідити наявні теоретичні матеріали.

- Проаналізувати різноманітні способи вирішення задачі і можливість їхнього застосування серед алгоритму програми.
- Реалізувати алгоритм розрахунку результатів та проміжних кроків різними способами.
- Розробити користувацький інтерфейс та дизайн застосунку, який включає в себе візуалізацію зв'язків й величин поставлених ресурсів між постачальниками й споживачами, а також відображення ітерацій алгоритму.

Об'єкти, засоби і методи розробки. Об'єктом розробки є процес складення оптимального плану для задачі транспортування ресурсів споживачам. Перед тим, як розпочати створення програмного продукту було сформовано математичну модель задачі.

В якості інструментарію було обрано мову програмування C#, засіб розробки Unity, а також середовище Visual Studio. C# вдало підходить для досліджуваної задачі, адже ця мова сфокусована об'єктно орієнтованому підході. Разом з цим, в ній наявна підтримка технік функціонального програмування [1]. Це принагідно для розв'язання задачі різними методами. Окрім цього, на платформі .NET наявні компоненти, що спрощують багато завдань, які виникають під час процесу написання програмного коду. Наприклад, компонент LINQ, що в реалізованій програмі використовується для пошуку кліток таблиці за певними параметрами, сортування, пошуку мінімальних та максимальних значень. Також для платформи .NET існує багато бібліотек для Unit-тестування, що теж спрощує процес розробки.

Unity – це інструмент розробки, призначений для створення відеоігор, симуляції та візуалізації математичних, фізичних, хімічних та інших завдань. У ньому наявні інструменти для рендерингу і розробки користувацького інтерфейсу. Unity – крос-платформний, отже реалізований застосунок можна легко перенести на інші операційні системи, в тому числі мобільні.

В якості інтегрованого середовища розробки (IDE) було взято Microsoft Visual Studio. У Visual Studio наявний функціонал для роботи з Unity, а також інтегровані засоби для Unit-тестування.

Сфери застосування. Найбільш очевидною сферою для застосування є оптимізація процесів постачання в промисловості. Ця галузь розглядається в роботі як основна. Проте середовище застосування є доволі універсальним. Для того, щоб використовувати транспортну задачу для інших сфер, потрібно замінити постачальників, споживачів і ресурси на інші сутності. Наприклад, для воєнної сфери можна обрати як постачальників засоби ураження, споживачів замінити на цілі ураження, а ресурси замінити на снаряди.

Враховуючи, що реалізований застосунок знаходить проміжні кроки розв’язку, можливим застосуванням також є навчання. Студенти, що вивчають лінійне програмування та дослідження операцій можуть отримати зрозумілий та детальний розв’язок транспортної задачі.

За останні роки транспортну задачу все частіше використовують для навчання нейронних мереж [6]. Також алгоритми розв’язання цієї задачі знайшли застосування у зіставленні зображень, сейсмології, економічному моделюванні.

РОЗДІЛ 1. ПОСТАНОВКА І МАТЕМАТИЧНА МОДЕЛЬ ЗАДАЧІ

Неформальну аналогію до цієї задачі презентував математик Гаспар Монж: працівник має перемістити велику кількість матеріалу, яка лежить на будівельному майданчику. Ціль працівника – за допомогою усього цього матеріалу звести певну конструкцію заданої форми. Очевидно, що при цьому робітник хоче якомога сильніше зменшити свої загальні зусилля, які вимірюються у затраченому часі або пройденій відстані. В цьому й полягає ціль задачі [6].

1.1 Постановка задачі

Наявний деякий ресурс однакового виду. Також наявні m постачальників $A_1, A_2, \dots, A_m$ і n споживачів $B_1, B_2, \dots, B_n$ цього ресурсу [7]. Роль ресурсу може виконувати певний матеріал, вид товару, робітник тощо. Постачальники $A_1, A_2, \dots, A_m$ займаються випуском (поставками) ресурсу, при цьому обсяг випуску для постачальника A_i сягає a_i одиниць ресурсу, $i = \overline{1, m}$. Постачальниками можуть виступати підприємства, склади, тощо. У споживачів $B_1, B_2, \dots, B_n$ є потреба в ресурсі, обсяг якої для споживача B_j рівний b_j одиницям ресурсу, $j = \overline{1, n}$. Наприклад, споживачами можуть бути магазини, підприємства, склади, або ж завдання/робота, якщо ресурсом виступають працівники. Дозволяється транспортування ресурсів від будь-якого постачальника до будь-якого споживача і вартість транспортування одиниці ресурсу від постачальника A_i до споживача B_j рівна $c_{i\,j}$ одиницям витрат, $i = \overline{1, m},\ j = \overline{1, n}$. Витрати можуть вимірюватись в одиницях грошей, часу, відстані чи інших, які обумовлені конкретною задачею, що вирішується. Для вирішення задачі потрібно знайти значення невідомих $x_{i\,j}$, що позначають скільки одиниць ресурсу постачальник A_i транспортує споживачу B_j, $i = \overline{1, m},\ j = \overline{1, n}$.

1.2 Математична модель

Транспортна задача належить до класу задач лінійного програмування.

Формально математична модель [5] описується наступним чином: потрібно визначити план транспортування $X = \{x_{i\,j}\}, i = \overline{1,m}, j = \overline{1,n}$, який задовольняє наступні умови:

$$\begin{cases} \sum_{j=1}^{n} x_{i\,j} = a_i, & i = \overline{1,m} \\ \sum_{i=1}^{m} x_{i\,j} = b_j, & j = \overline{1,n} \\ x_{i\,j} \geq 0, & \forall\, i,j \end{cases} \tag{1.1}$$

А також мінімізує наступну функцію:

$$L(X) = \sum_{i=1}^{m} \sum_{j=1}^{n} c_{i\,j} x_{i\,j} \to min \tag{1.2}$$

Для транспортної задачі характерні дві особливості, які дозволяють використовувати для розв'язку алгоритми, що є суттєво простішими за симплекс-метод:

1. Кожне невідоме входить в систему (1.1) рівно два рази.
2. Всі коефіцієнти при невідомих в системі (1.1) можуть бути рівними тільки нулю або одиниці.

1.3 Існування розв'язку, закрита модель

В описаній математичній моделі необхідною і достатньою умовою існування розв'язку транспортної задачі є наступна умова [8]:

$$\sum_{i=1}^{m} a_i = \sum_{j=1}^{n} b_j$$

Тобто, загальний обсяг випуску ресурсів усіх постачальників повинен бути рівним загальним потребам усіх споживачів. Транспортна задача, для якої виконується дана умова називається збалансованою, або ж транспортною задачею із правильним балансом. Описана модель називаються закритою.

1.4 Відкрита модель

Не кожна задача є збалансованою. Модель називається відкритою, а задача називається незбалансованою, якщо виконується одна з наступних умов:

$$\sum_{i=1}^{m} a_i < \sum_{j=1}^{n} b_j \tag{1.3}$$

$$\sum_{i=1}^{m} a_i > \sum_{j=1}^{n} b_j \tag{1.4}$$

Транспортна задача з неправильним балансом приводиться до збалансованого вигляду. Перетворення виконується шляхом введення фіктивного постачальника/споживача.

Якщо виконується умова (1.3), то вводиться постачальник A_{m+1}, який позначає нестачі в потребах споживачів. Обсяг випуску для нього рівний різниці сумарних потреб і поставок: $\sum_{j=1}^{n} b_j - \sum_{i=1}^{m} a_i$. Вартості транспортування, як правило, рівні нулю: $c_{m+1\,j} = 0,\ j = \overline{1,n}$. Якщо відома вартість неможливості поставити одиницю ресурсу (наприклад, втрачений дохід споживача), то варто замість нуля використовувати таке значення. Змінні $x_{m+1\,j},\ j = \overline{1,n}$ показують скільки одиниць ресурсу не вистачає споживачу B_j.

Аналогічно, якщо виконується умова (1.4), то вводиться споживач B_{n+1}, який позначає кількість ресурсу, що залишається у постачальників. Потреба для нього рівна різниці сумарного випуску і потреб: $\sum_{i=1}^{m} a_i - \sum_{j=1}^{n} b_j$. Вартості транспортування, як правило, рівні нулю: $c_{i\,n+1} = 0,\ i = \overline{1,m}$. Якщо відома вартість зберігання одиниці ресурсу, то варто замість нуля використовувати таке значення. Змінні $x_{i\,n+1},\ i = \overline{1,m}$ показують скільки одиниць ресурсу залишається у постачальника A_i.

1.5 Допустимий і оптимальний план транспортувань

План транспортувань називається допустимим, якщо виконуються обмеження моделі (1.1).

План транспортувань називається оптимальним, якщо виконуються обмеження моделі (1.1) і при цьому досягається мінімум функції (1.2).

1.6 Опорний план транспортувань, виродженість

Якщо значення $x_{i\,j}$ не дорівнює нулю, то відповідна клітка в таблиці вирішення транспортної задачі називається зайнятою, в іншому випадку – незайнятою (вільною).

Допустимий транспортний план, що містить $m + n - 1$ зайнятих клітин називається опорним.

Якщо в розв’язку менше ніж $m + n - 1$ зайнятих клітин, то він називається виродженим [5]. Щоб побороти виродженість потрібно додати нескінченно мале значення e в незайняту клітку, так щоб вона не створювала із зайнятими клітками циклу. Значення e при обчисленні загальної вартості не враховується. Незважаючи на це, така клітка вважається зайнятою.

1.7 Цикл

Цикл це впорядкована послідовність хоча б з чотирьох кліток для якої виконуються наступні три умови:

1. Будь-які дві послідовні клітки знаходяться в одному рядку або стовпчику.
2. Три і більше послідовних кліток не лежать в одному рядку або стовпчику.
3. Остання клітка знаходиться в одному рядку або стовпчику що й перша.

РОЗДІЛ 2. МЕТОДИ ПОШУКУ ОПОРНОГО РОЗВ'ЯЗКУ

Щоб зменшити значну кількість кроків, необхідних для отримання оптимального рішення, доцільно перейти до опорного плану, що є близьким до оптимального. Одною з проблем на даному етапі є те, що неможливо просто передбачити чи транспортна задача матиме вироджене рішення. Для визначення опорного плану транспортування існує декілька методів. Вони, як правило, розділяють частину однакових кроків послідовності дій, проте принцип вибору клітки у кожному методі сильно різниться.

Перед початком пошуку опорного плану умову задачі потрібно переписати у табличну форму. Для згаданої задачі з m постачальниками та n споживачами таблиця складається відповідно з m рядків і n стовпців. Для зручності посилання, клітка, що розташована в перетині рядка i і стовпця j, позначається як «клітка (i, j))». Параметри задачі передаються у наведеному нижче форматі:

	B_1	…	B_n	
A_1	$x_{1\,n}/c_{1\,n}$	…	$x_{1\,n}/c_{1\,n}$	a_1
…	…	…	…	…
A_m	$x_{m\,1}/c_{m\,1}$	…	$x_{m\,n}/c_{m\,n}$	a_m
	b_1	…	b_n	

Після вибору клітки (i, j) на черговій ітерації методу пошуку опорного розв’язку, як правило, в клітці призначається максимально можливе значення $x_{i\,j}$, після чого вносяться відповідні зміни до випуску (потреб) і викреслюється рядок (стовпець), якщо відповідна величина вичерпана. Виконується це наступним чином: обчислюється значення $\min\{a_i, b_j\}$ і призначається клітці (i, j). Якщо $\min\{a_i, b_j\} = a_i$, то випуск постачальника A_i вичерпаний і потреба споживача B_j

залишається рівна $b_j - a_i$, а рядок i викреслюється із таблиці. В протилежному випадку, якщо $\min\{a_i, b_j\} = b_j$, то потреба споживача B_j вичерпана і випуск постачальника A_i залишається рівним $a_i - b_j$, а стовпець j викреслюється із таблиці. Після того, як рядок (стовпець) викреслюється, всі значення $x_{i\,j}$ у рядку (стовпці) крім визначеного встановлюються рівні нулю і надалі не розглядаються.

2.1 Метод північно-західного кута

Основна перевага методу правила північно-західного кута полягає в тому, що він є легким у застосуванні і не потребує допоміжних обчислень. Його основним недоліком, однак, є те, що він не чутливий до витрат і, отже, призводить до не сильно якісних початкових рішень. Для визначення початкового розв'язку за допомогою правила північно-західного кута виконуються наступні кроки [5]:

1. Задача записується у табличній формі.
2. Процес переноситься до північно-західної клітки таблиці, тобто клітки серед невикреслених рядків і стовпців з найменшими індексами. Ця клітка позначається як (i, j).
3. В клітці (i, j) встановлюється максимально допустиме значення $x_{i\,j}$, після чого вносяться відповідні зміни до значень поставок, потреб і таблиці.
4. Повторюються кроки 2, 3 доки не буде вичерпано всі випуски постачальників і не будуть виконані всі потреби споживачів.

Якщо під час виконання третього кроку пропозиція і попит до завершення алгоритму задовольняються одночасно, то обирається рядок або стовпець, щоб його викреслити. Наступна змінна, яка буде додана до опорного рішення, обов'язково буде рівна нулю, а клітка - незайнята. В цьому випадку рішення є виродженим.

2.1.1 Приклад пошуку опорного розв'язку методом північно-західного кута

Задано наступну транспортну задачу в табличній формі:

	B_1	B_2	B_3	B_4	Постачання
A_1	$^{-}/_{5}$	$^{-}/_{10}$	$^{-}/_{6}$	$^{-}/_{9}$	45
A_2	$^{-}/_{3}$	$^{-}/_{15}$	$^{-}/_{7}$	$^{-}/_{3}$	65
A_3	$^{-}/_{4}$	$^{-}/_{20}$	$^{-}/_{12}$	$^{-}/_{13}$	20
Потреби	35	25	40	30	130

Обирається північно-західна клітка $(1,1)$. Відповідний обсяг потреб a_1 менший за обсяг постачання b_1, тому в клітці виділяється кількість ресурсу $x_{1\,1} = 35$. Вводяться зміни до значень поставки і потреби: $a_1 = 45 - 35 = 10$, $b_1 = 35 - 35 = 0$. Потреба вичерпана, отже відповідний стовпець викреслюється.

	~~B_1~~	B_2	B_3	B_4	Постачання
A_1	$^{35}/_{5}$	$^{-}/_{10}$	$^{-}/_{6}$	$^{-}/_{9}$	10
A_2	$^{0}/_{3}$	$^{-}/_{15}$	$^{-}/_{7}$	$^{-}/_{3}$	65
A_3	$^{0}/_{4}$	$^{-}/_{20}$	$^{-}/_{12}$	$^{-}/_{13}$	20
Потреби	0	25	40	30	130

Обирається північно-західна клітка $(1,2)$. Відповідний обсяг постачання b_2 менший за обсяг потреб a_1, тому в клітці виділяється кількість ресурсу $x_{1\,2} = 10$. Вводяться зміни до значень поставки і потреби: $a_1 = 10 - 10 = 0$, $b_2 = 25 - 10 = 15$. Постачання вичерпано, отже відповідний рядок викреслюється.

	~~B_1~~	B_2	B_3	B_4	Постачання
~~A_1~~	$^{35}/_{5}$	$^{10}/_{10}$	$^{0}/_{6}$	$^{0}/_{9}$	~~0~~
A_2	$^{0}/_{3}$	$^{-}/_{15}$	$^{-}/_{7}$	$^{-}/_{3}$	65

A_3	$^{0}/_{4}$	$^{-}/_{20}$	$^{-}/_{12}$	$^{-}/_{13}$	20
Потреби	0	15	40	30	130

Аналогічні дії повторюються поки залишаються невикреслені клітки.

	B_1	B_2	B_3	B_4	Постачання
A_1	$^{35}/_{5}$	$^{10}/_{10}$	$^{0}/_{6}$	$^{0}/_{9}$	0
A_2	$^{0}/_{3}$	$^{15}/_{15}$	$^{-}/_{7}$	$^{-}/_{3}$	50
A_3	$^{0}/_{4}$	$^{0}/_{20}$	$^{-}/_{12}$	$^{-}/_{13}$	20
Потреби	0	0	40	30	130

	B_1	B_2	B_3	B_4	Постачання
A_1	$^{35}/_{5}$	$^{10}/_{10}$	$^{0}/_{6}$	$^{0}/_{9}$	0
A_2	$^{0}/_{3}$	$^{15}/_{15}$	$^{40}/_{7}$	$^{-}/_{3}$	10
A_3	$^{0}/_{4}$	$^{0}/_{20}$	$^{-}/_{12}$	$^{-}/_{13}$	20
Потреби	0	0	0	30	130

	B_1	B_2	B_3	B_4	Постачання
A_1	$^{35}/_{5}$	$^{10}/_{10}$	$^{0}/_{6}$	$^{0}/_{9}$	0
A_2	$^{0}/_{3}$	$^{15}/_{15}$	$^{40}/_{7}$	$^{10}/_{3}$	0
A_3	$^{0}/_{4}$	$^{0}/_{20}$	$^{0}/_{12}$	$^{20}/_{13}$	0
Потреби	0	0	0	0	130

Усі потреби та постачання вичерпані. Кількість зайнятих кліток рівна $m + n - 1 = 6$. Отже, наведений нижче план транспортувань є опорним:

	B_1	B_2	B_3	B_4	Постачання
A_1	$^{35}/_{5}$	$^{10}/_{10}$	$^{-}/_{6}$	$^{-}/_{9}$	45
A_2	$^{0}/_{3}$	$^{15}/_{15}$	$^{40}/_{7}$	$^{10}/_{3}$	65
A_3	$^{0}/_{4}$	$^{-}/_{20}$	$^{-}/_{12}$	$^{20}/_{13}$	20
Потреби	35	25	40	30	130

Загальна вартість транспортувань: $5 \cdot 35 + 10 \cdot 10 + 15 \cdot 15 + 7 \cdot 40 + 3 \cdot 10 + 13 \cdot 20 = 1070$.

2.2 Метод найменшої вартості

У методі найменшої вартості розподіли здійснюються на основі економічної доцільності, через що знаходиться кращий розв'язок. Етапи визначення початкового рішення за допомогою цього методу наступні [8]:

1. Задача записується у табличній формі.
2. Обирається клітка, вартість транспортування в якій мінімальна. Якщо така клітка не є унікальною, то дозволяється вибір будь-якої з них. Ця клітка позначається як (i, j).
3. В клітці (i, j) встановлюється максимально допустиме значення $x_{i\,j}$, після чого вносяться відповідні зміни до значень поставок, потреб і таблиці.
4. Повторюються кроки 2, 3 доки не буде вичерпано всі випуски постачальників і не будуть виконані всі потреби споживачів.

2.3 Методи мінімуму в рядку і мінімуму стовпці

Близькими до методу найменшої вартості є методи мінімуму в рядку (стовпці). Єдиною розбіжністю є те, що клітка з транспортуванням найменшої вартості обирається не серед цілої матриці, а лише в рядку (стовпці), що розглядається на поточному кроці. Це дає невелику перевагу в швидкості виконання пошуку опорного розв’язку.

2.4 Метод подвійної переваги

Суть методу подвійної переваги полягає в маркуванні в кожному рядку клітки з найменшою вартістю деякою позначкою. Аналогічна дія виконується в кожному стовпці таблиці. Після виконання даного процесу у деяких кліток з’являються дві позначки, що означає, що дана клітка має мінімальну вартість і в своєму рядку, і в своєму стовпці. В довільному порядку таким кліткам виділяється максимально можливий об’єм транспортувань, що задовольняє відповідну потребу споживача і не перевищує обсяг випуску постачальника. Далі вносяться відповідні зміни в таблицю так, як це відбувається в попередньо розглянутих методах. Після того як кліток з двома маркуваннями не залишається, алгоритм переходить до кліток з однією позначкою. Для решти таблиці значення об’єму перевезень $x_{i\,j}$ визначаються як у методі найменшої вартості.

2.5 Метод апроксимації Фогеля

Метод апроксимації Фогеля [9] (часто використовують абревіатуру англійською мовою VAM) є покращеною версією методу найменшої вартості, який здебільшого (але не завжди) забезпечує кращі опорні розв’язки. Метод базується на концепції мінімізації альтернативних (або штрафних) витрат. Альтернативна

вартість для даного рядка пропозиції або стовпця попиту визначається як різниця між найнижчою вартістю та наступною найнижчою альтернативою. Цьому методу надається перевага перед іншими алгоритмами, оскільки він, як правило, призводить до оптимального або близького до нього розв'язку. Це означає, що кількість ітерацій та часу, необхідних для обчислення оптимального рішення, значно зменшується. Метод виконується у такій послідовності:

1. Задача записується у табличній формі.
2. Обчислюється різниця між мінімальною і наступною по величині вартістю в рядку. Ця різниця називається штрафом. Аналогічні дії виконуються для стовпців.
3. Обирається рядок (стовпчик) з найбільшим штрафом. Нехай це рядок i (стовпчик j). У ньому обирається клітка з найменшою вартістю транспортування, нехай це клітка (i, j).
4. В клітці (i, j) встановлюється максимально допустиме значення $x_{i\ j}$, після чого вносяться відповідні зміни до значень поставок, потреб і таблиці.
5. Повторюються кроки 2, 3, 4 доки не буде вичерпано всі випуски постачальників і не будуть виконані всі потреби споживачів.

РОЗДІЛ 3. МЕТОДИ ПОШУКУ ОПТИМАЛЬНОГО РОЗВ'ЯЗКУ

Транспортну задачу можна вирішувати методами, які є універсальними для всіх задач класу лінійного програмування. Проте, більш ефективно застосовувати методи, що використовують особливості транспортної задачі. Для перевірки оптимальності розв'язку використовують означений цикл. Якщо в циклі позначити кутові клітки з непарними номерами знаком «плюс», а клітки з парними номерами знаком «мінус», то цей цикл називається означеним. Для нього характерне поняття перерозподілу – збільшення величини змінних у клітках, позначених знаком «плюс», і зменшення величини у клітках, позначених знаком «мінус», на задану величину θ.

3.1 Розподільний метод

В опорному розв'язку для будь-якої незайнятої клітки існує лише один цикл, в якому вона міститься разом із деякими зайнятими клітками. При виконанні перерозподілу даного циклу на число $\theta = \min\{x_{i\,j}\}$ ($x_{i\,j}$ – клітки позначені знаком «мінус») одержується новий опорний розв'язок.

Вартості кліток циклу позначаються наступним чином:

– $c_{i\,j}^{(+)}$ – вартість клітки з непарним номером.

– $c_{i\,j}^{(-)}$ – вартість клітки з парним номером.

Приріст функції при перерозподілі в клітці (l,k) рівний $\Delta_{l\,k} = \sum c_{i\,j}^{(+)} - c_{i\,j}^{(-)}$. Якщо це значення є від'ємним, цільова функція зменшується на $\theta\Delta_{l\,k}$, тобто існує кращий розв'язок з меншим значенням цільової функції. На противагу, якщо всі значення оцінок для незайнятих кліток є невід'ємними, поточний план є оптимальним.

Отже, послідовність дій при оптимізації розподільним методом наступна [4]:

1. Довільним методом знаходиться опорний розв’язок.
2. Розв’язок перевіряється на виродженість. При виродженості шукається вільна клітка, яка не утворює цикл із зайнятими клітками. У знайденій клітці встановлюється нескінченно мале значення e.
3. Розглядаються вільні клітки опорного розв’язку. Нехай на поточному кроці це клітка (l, k).
4. Для клітки (l, k) виконується пошук циклу. Клітки знайденого циклу позначаються знаками «плюс» та «мінус».
5. Обраховується оцінка $\Delta_{l\,k}$:
   - При невід’ємній оцінці виконується пошук наступної вільної клітки, для якої не обраховано оцінку. Якщо вільних кліток не залишилось, то оптимальний розв’язок знайдено. В протилежному випадку, для знайденої клітки виконуються кроки 3, 4, 5.
   - При від’ємній оцінці виконується перерозподіл циклу на величину $\theta = \min\{x_{i\,j}\}$, внаслідок чого одержується новий опорний розв’язок. Кроки 2, 3, 4, 5 повторюються заново для нового опорного розв’язку.

3.2 Метод потенціалів

У розподільному методі потрібно шукати цикл для кожної вільної клітки, яких в таблиці є $m \cdot n - (m + n - 1)$. Враховуючи, що пошук циклу потребує трудомістких обчислень, алгоритм займає багато часу. В методі потенціалів оцінки вільних кліток обраховуються одночасно і під час кожної ітерації потрібно знайти лише один цикл. Таким чином, метод потенціалів сильно заощаджує час для обчислень.

Для оптимального розв'язку існують потенціали постачальників $u_i,\ i = \overline{1,m}$ і потенціали споживачів $v_j,\ j = \overline{1,n}$. Згадані потенціали задовольняють наступні умови [5]:

$$u_i + v_j \le c_{i\,j}, \qquad x_{i\,j} = 0$$

$$u_i + v_j = c_{i\,j}, \qquad x_{i\,j} > 0$$

З математичної моделі двоїстої задачі випливає, що кількість невідомих дорівнює $m + n$, при цьому система складається із $m + n - 1$ рівнянь. Це означає, що одному з невідомих потрібно присвоїти довільне значення (як правило, нульове).

Потенціали використовуються, щоб перевірити план на оптимальність. Кроки розв'язку для методу потенціалів наступні:

1. Довільним методом знаходиться опорний розв'язок.
2. Розв'язок перевіряється на виродженість. При виродженості шукається вільна клітка, яка не утворює цикл із зайнятими клітками. У знайденій клітці встановлюється нескінченно мале значення e.
3. Знаходяться потенціали $u_i,\ i = \overline{1,m},\ v_j,\ j = \overline{1,n}$. Для цього значення u_1 встановлюється рівним нулю. Значення невідомих потенціалів послідовно обчислюються на основі знайдених значень: $u_i = \mathrm{c}_{ij} - v_j,\ v_j = \mathrm{c}_{ij} - v_j$.
4. Розглядаються вільні клітки опорного розв'язку. Для вільних кліток (i,j) обраховуються оцінки $\Delta_{i\,j} = u_i + v_j - c_{i\,j}$. Серед оцінок обирається клітка (l,k) з максимальною оцінкою.
   - Якщо оцінка $\Delta_{l\,k} > 0$, для клітки (l,k) виконується пошук циклу. Клітки знайденого циклу позначаються знаками «плюс» та «мінус». Виконується перерозподіл циклу на величину $\theta = \min\{x_{i\,j}\}$,

внаслідок чого одержується новий опорний розв’язок. Кроки 2, 3, 4 повторюються заново для нового опорного розв’язку.

– Якщо оцінка $\Delta_{l\,k} \leq 0$, то оптимальний розв’язок знайдений.

3.3 Симплекс-метод

Транспортна задача – задача класу лінійного програмування. Це означає, що її можна вирішити звичайним симлпекс-методом. Це універсальний метод для задач лінійного програмування, який був розроблений Джорджом Данцігом [8].

Вхідні дані транспортної таблиці виражають через лінійні рівняння. Для вирішення задачі умови математичної моделі переписуються в наступній формі:

$$\begin{cases} x_{1\,1} + x_{1\,2} + \cdots + x_{1\,n} = a_1 \\ \cdots \\ x_{m\,1} + x_{m\,2} + \cdots + x_{m\,n} = a_m \\ x_{1\,1} + x_{2\,1} + \cdots + x_{m\,1} = b_1 \\ \cdots \\ x_{1\,n} + x_{2\,n} + \cdots + x_{m\,n} = b_n \\ c_{1\,1}x_{1\,1} + \cdots + c_{1\,n}x_{1\,n} + c_{2\,1}x_{2\,1} + \cdots + c_{2\,n}x_{2\,n} + c_{m\,1}x_{m\,1} + \cdots + c_{m\,n}x_{m\,n} = z \end{cases}$$

У використанні сиплекс-метода для транспортної задачі є свої переваги та недоліки. Позитивним є те, що цей метод є популярним та універсальним. Отже, існує багато готових алгоритмів та бібліотек для розробки програмного забезпечення, які ефективно використовують його для вирішення задач лінійного програмування. Проте, розглянуті раніше методи є специфічними для транспортної задачі, а також мають більш простий та зрозумілий алгоритм пошуку розв’язку. При цьому рішення знаходиться швидше і за меншу кількість ітерацій.

3.4 Генетичний алгоритм

Генетичні алгоритми, розроблені Джоном Холландом, є алгоритмами, які використовують механізми подібні до біологічної еволюції, щоб стимулювати «виживання» найкращих проміжних розв’язків [3]. У генетичних алгоритмах структурованим способом створюються стрічки чисел, щоб сформувати алгоритм пошуку рішення для конкретного завдання. Генетичні алгоритми використовують концепцію вибору найпридатніших стрічок для використання у наступному поколінні, відкидаючи недостатньо вдалі. У кожному поколінні створюється новий набір стрічок використовуючи частини кращих стрічок з попередніх поколінь. Оскільки забезпечується зберігання найкращих, стрічки стають кращими кожного покоління. Таким чином, після певної кількості пройдених поколінь найкраще значення обирається як остаточне рішення.

Перше покоління (початкова популяція), як правило, випадкове. Деякі стрічки в першому поколінні обчислюються методами пошуку опорного розв’язку. Це гарантує, що перше покоління матиме допустимі рішення. Числа в рядку відповідають змінним у цільовій функції.

Основним критерієм відтворення є функція допасованості. Ця цільова функція відображає наскільки близьким до досягнення поставлених цілей є рішення, що розглядається. Кожній стрічці надається значення допасованості залежно від допустимості рішення. Щоб створити наступне покоління для розмноження обираються найкращі значення.

Покоління утворюються методом кросинговеру. В генетичному алгоритмі наявні два основних види цього оператору:

– Одноточковий кросинговер: обирається точка серед батьківських рядків (точка розриву), яка розділяє рядки на ліву та праву частини. Після цього рядки обмінюються правими частинами.

– Двоточковий перехід: обираються дві точки серед батьківських рядків. Усі числа в межах двох точок перетину залишаються незмінними. Ліва та права від них частини обмінюються.

Розміщуючи точки перетину в різних положеннях, можна створити багато комбінацій. Тому навіть з невеликої кількості рядків, що належать до першого покоління, можна утворити велику кількість нащадків.

Операція мутації теж використовується для створення нових поколінь, але, як правило, значно рідше. Вважається, що в популяції є деяка доля «мутантів», для яких при утворенні нового покоління виконуються зміни згідно з визначеними правилами мутації. Це, як правило, генерація випадкових значень.

Щоб вирішити транспортну задачу за допомогою цього евристичного методу, виконуються такі кроки:

1. Створюється функція допасованості шляхом перемноження вартості транспортування для змінних у цільовій функції на найбільш можливе значення – мінімум серед обсягу випуску постачальника і потреби споживача.
2. Перше покоління створюється випадковим чином (або використовуються значення опорного розв’язку). Згодом рядки з найнижчим значенням допасованості відбираються для розмноження.
3. Наступне покоління створюється за допомогою операцій кросинговеру та мутації. Згодом найкращі рядки відбираються для створення наступного покоління.
4. Перевіряється допустимість плану транспортування, згенерованого значеннями стрічок. План повинен відповідати всім обмеженням математичної моделі. Непридатні рядки відхиляються.
5. Обчислюється значення функції придатності, яка відповідає функції із моделі задачі. Знайдене значення зберігається разом із рядком.

Далі кроки 3, 4, 5 повторюються для кожного покоління. Після генерування достатньої кількості поколінь, значення порівнюються і найнижче значення вважається за оптимальне рішення.

3.5 Пошук максимального потоку мінімальної вартості

Транспортну задача можна представити і вирішити як задачу теорії графів. Розглядається двочастковий граф, у якому пункти постачальників перебувають у верхній частці, а пункти споживання у нижній. Пункти постачальників та споживачів попарно з'єднуються ребрами нескінченної пропускної спроможності та вартості.

До постачальників приєднується джерело. Пропускна здатність ребер з джерела до вершин рівна об’єму випуску ресурсів цього пункту, а вартість одиниці потоку дорівнює нулю. Для нижньої частки виконуються ідентичні дії. До пунктів споживання приєднується стік. Пропускна здатність ребер від споживачів у стік рівна потребі ресурсів цього пункту, а вартість одиниці потоку дорівнює нулю.

Далі вирішується задача пошуку максимального потоку мінімальної вартості [5]. Це можливо виконати за допомогою алгоритму Форда-Фалкерсона. Проте, замість найкоротшого доповнюючого потоку обчислюється найдешевший. Для цієї задачі замість пошуку у ширину використовується алгоритм Беллмана-Форда. При поверненні потоку вартість рахується негативною.

Для вирішення задачі пошук опорного розв’язку є необов’язковим. Проте, процес пошуку розв’язку в такому разі буде виконуватись довше. Складність алгоритму - не більше $O(e^2v^2)$ операцій (e – кількість ребер, v – кількість вершин). Для середньостатистичного випадку потрібно менше операцій – складність рівна $O(ev)$ операцій.

РОЗДІЛ 4. МОДИФІКАЦІЇ ЗАДАЧІ

4.1 Багатоіндексна транспортна задача

Ціль багатоіндексних транспортних задач залишається незмінною – пошук оптимального плану транспортувань. Відмінністю є те, що такі задачі враховують неоднорідність типів ресурсів, які треба розподілити. Також може братись до уваги неоднорідність засобів транспорту чи якісь інші змінні.

Найбільш популярним з таких узагальнень є триіндексна транспортна задача в якій присутні p різних видів ресурсу E_k, $k = \overline{1,p}$. Модель такої задачі можна записати наступним чином:

$$\begin{cases} \sum_{j=1}^{n} x_{i\,j\,k} = a_{i\,k}, & i = \overline{1,m}, k = \overline{1,p} \\ \sum_{i=1}^{m} x_{i\,j\,k} = b_{j\,k}, & j = \overline{1,n}, k = \overline{1,p} \\ \sum_{k=1}^{p} x_{i\,j\,k} = e_{i\,j}, & i = \overline{1,m}, j = \overline{1,n} \\ x_{i\,j\,k} \geq 0, & \forall\, i,j,k \end{cases}$$

$$L(X) = \sum_{i=1}^{m}\sum_{j=1}^{n}\sum_{k=1}^{p} c_{i\,j\,k} x_{i\,j\,k} \to min$$

Для існування рішення задачі необхідно і достатньо, щоб виконувались наступні умови балансу:

- Обсяг поставок кожного типу ресурсу рівний обсягу потреби в ньому.
- Обсяг випуску кожного постачальника рівний обсягу транспортувань від його пункту.
- Обсяг потреби кожного споживача рівний обсягу транспортувань до його пункту.

Дана модифікація транспортної задачі вирішується за допомогою узагальнення методу потенціалів. Опорний розв'язок знаходиться методом найменшої вартості.

Також задачу можна розділити на підзадачі і вирішити симплекс-методом.

4.2 Транспортна задача з проміжними пунктами

Частою має місце наступна ситуація: постачальник транспортує ресурси не напряму споживачу, а через деякого посередника. Такий посередник називається проміжним пунктом. В його ролі може виступати, наприклад, склад із ресурсами. Постановка транспортної задачі з проміжними пунктами відрізняється від оригінального формулювання тим, що у цій модифікації наявні p проміжних пунктів $E_k, k = \overline{1,p}$ і додаткові потреби в ресурсі e_k. Якщо обсяг додаткової потреби задається від'ємним значенням, то ця потреба є надлишком. Також задаються затрати на транспортування одиниці ресурсу $c^{(1)}_{k\,i}$ від постачальника A_i у проміжний пункт E_k і затрати $c^{(2)}_{k\,j}$ з проміжного пункту E_k до споживача B_j.

Математична модель:

$$\begin{cases} \sum\limits_{k=1}^{p} x_{k\,i} = a_i, & i = \overline{1,m} \\ \sum\limits_{k=1}^{p} y_{k\,i} = b_j, & j = \overline{1,n} \\ \sum\limits_{i=1}^{m} x_{k\,i} - \sum\limits_{j=1}^{n} y_{k\,j} = e_k, & k = \overline{1,p} \\ x_{k\,i} \geq 0, & \forall\, i,k \\ y_{k\,j} \geq 0, & \forall\, k,j \end{cases}$$

$$L(X) = \sum_{i=1}^{m}\sum_{k=1}^{p} c_{k\,i}^{(1)} x_{k\,i} + \sum_{k=1}^{p}\sum_{j=1}^{n} c_{k\,j}^{(2)} y_{k\,j} \to min$$

Критерій розв’язності:

$$\sum_{i=1}^{m} a_i - \sum_{j=1}^{n} b_j = \sum_{k=1}^{p} e_k$$

Щоб вирішити транспортну задачу з проміжними пунктами потрібно знайти опорний розв’язок і оптимізувати його модифікованим методом потенціалів, який буде враховувати випадок з від’ємними значеннями.

4.3 Транспортна задача для максимізації прибутку

Існують варіації транспортних задач, у яких цільова функція має бути максимізована, а не мінімізована. Така модифікація доцільна коли потрібно отримати максимальний прибуток від поставок ресурсу. Ці задачі можна вирішити, перетворивши задачу максимізації в задачу мінімізації. Конвертування функції максимуму в мінімум здійснюється шляхом призначення нових значень вартостям перевезення. Нові значення позначають недостачу до максимальної вартості і стають рівними різниці найбільшого та свого значення: $c'_{i\,j} = \max\{C\} - c_{i\,j},\ i = \overline{1,m}, j = \overline{1,n}.$

Тепер задача вирішується ідентично класичній. Після розв’язання слід використовувати початкові значення вартостей.

РОЗДІЛ 5. ПРОГРАМНА РЕАЛІЗАЦІЯ

Реалізований проект складається з бібліотеки для розв’язку транспортної задачі, що написана мовою програмування C# (платформа .NET), а також з програмного застосунку, реалізованого за допомогою інструменту розробки Unity. Окрім цього, було проведено порівняння продуктивності використаних методів розв’язку. Програмний застосунок використовує бібліотеку для пошуку розв’язку і проміжних кроків, виконує функції візуалізації, отримання вхідних даних, вибору методу оптимізації, відображення результату і ітерацій алгоритму.

5.1 Бібліотека для розв’язку транспортної задачі

5.1.1 Класи Problem і Plan

Класи Problem і Plan використовуються для репрезентації параметрів математичної моделі.

Problem – транспортна задача. Містить наступні поля:

- Supplies – обсяги поставок.
- Demands – обсяги потреб.
- Costs – вартості транспортування одиниці ресурсу.

Plan – план перевезень транспортної задачі. Містить наступні поля:

- Problem – задача, для якої застосовується план транспортувань.
- Shipments – обсяги перевезень.

Окрім цього, в класі наявний метод GetTotalCost() – обчислення загальних витрат на транспортування.

5.2.2 Пошук опорного розв’язку

Будувати опорний план можна різними методами. Тому використовується шаблон проектування (патерн) стратегія. Це виконано за допомогою інтерфейсу IReferencePlanStrategy, в якому наявний метод GetReferencePlan(), параметром якого є об’єкт класу Problem, а повертається об’єкт класу Plan. Для всіх методів пошуку опорного плану наявні відповідні класи, які реалізують згаданий інтерфейс:

- NortWestCornerStrategy – метод північно-західного кута.
- LeastCostStrategy – метод найменшої вартості.
- VogelApproximationStrategy – метод апроксимації Фогеля.

5.1.3 Класи Cell і Table

Для представлення транспортної таблиці у програмному засобі використовуються класи Cell і Table.

Cell – клітка транспортної таблиці. Містить наступні поля:

- Row – номер рядка у таблиці.
- Col – номер стовпця у таблиці.
- Cost – вартість транспортування одиниці ресурсу.
- Allocation – кількість одиниць ресурсу, виділених для транспортування.
- AllocationE – нескінченно мала кількість виділеного ресурсу. Використовується для позбавлення плану перевезень від виродженості.

Також реалізований клас CellAllocationComparer для порівняння кліток за величиною виділених одиниць ресурсу.

Table – транспортна таблиця. Містить поле Cells (клітки таблиці) і метод GetPlan() для конвертування таблиці в план транспортувань.

5.1.4 Пошук оптимального розв’язку

Методи побудови оптимального плану реалізовані аналогічно методам побудови опорного плану. Інтерфейс IOptimalPlanStrategy містить методи

GetOptimalPlan() і GetOptimalPlanWithSteps(). Алгоритми пошуку оптимального розв’язку реалізують інтерфейс у відповідних класах:

- SteppingStoneStrategy – розподільний метод.
- ModiStrategy – метод потенціалів.

Класи SteppingStoneStrategy і ModiStrategy наслідують клас RedistributiveMethod – клас алгоритму пошуку оптимального плану, використовуючи перерозподіл циклу. Методи цього класу:

- GetAllocatedCells() – повертає список зайнятих кліток.
- GetFreeCells() – повертає список вільних кліток.
- GetPath() – повертає цикл для заданої клітки. Цикл задається списком кліток. Використовується алгоритм пошуку у ширину.
- GetReachableCells() – повертає список кліток, які можна доєднати до циклу.
- GetLeastAllocationCellInPath() – повертає клітку з найменшою кількістю виділених для транспортування одиниць ресурсу. Враховуються нескінченно малі значення.
- RedistributePath() – виконує перерозподіл у циклі.
- FixDegeneracy() – позбувається виродженості у розв’язку.

Клас SteppingStoneStrategy окрім методів описаних в інтерфейсі також містить наступні методи:

- MakeIteration() – виконує ітерацію алгоритму.
- GetPathCost() – повертає загальну вартість в означеному циклі.

Клас ModiStrategy містить наступні методи:

- MakeIteration() – виконує ітерацію алгоритму.

- CalculateUV() – обчислює потенціали постачальників та споживачів. Спочатку задається нульове значення для потенціалу постачальника з найнижчим індексом. Обчислення відбуваються за допомогою структури даних черги. У чергу додаються клітки, для яких обчислено лише один з відповідних їм потенціалів.
- CalculateEstimates() – обчислює оцінки для вільних кліток.
- GetMaxEstimate() – повертає максимальну оцінку.

5.1.5 Клас SolutionSteps

Для зберігання проміжних кроків використовується клас StepSaver. Він складається з наступних полей:

- OriginalProblem – оригінальна задача.
- BalancedProblem – задача після зведення до збалансованого виду і зведення до задачі мінімізації витрат.
- ReferencePlan – опорний розв’язок.
- OptimalPlanIterations – список розв’язків, отриманих після кожної ітерації методу оптимізації опорного плану.
- Paths – список циклів, отриманих після кожної ітерації методу оптимізації опорного плану.
- OptimalPlan – оптимальний розв’язок.

5.1.6 Клас Solver

Клас Solver виконує функцію повного розв’язку транспортної задачі. Він включає такі поля:

- problem – вхідна задача.
- referencePlanStrategy – об’єкт класу, що реалізує інтерфейс IReferencePlanStrategy (метод пошуку опорного плану).

- optimalPlanStrategy – об’єкт класу, що реалізує інтерфейс IOptimalPlanStrategy (метод пошуку оптимального плану).
- targetFunction – цільова функція. Приймає два значення: TargetFunction.Min (для задачі мінімізації витрат) і TargetFunction.Max (для задачі максимізації прибутку).
- totalIterations – загальна кількість ітерацій, виконаних у методі пошуку оптимального розв’язку.
- maxCost – максимальне значення серед початкових вартостей перевезення. Використовується для конвертування задачі мінімізації витрат в задачу максимізації прибутку і навпаки.

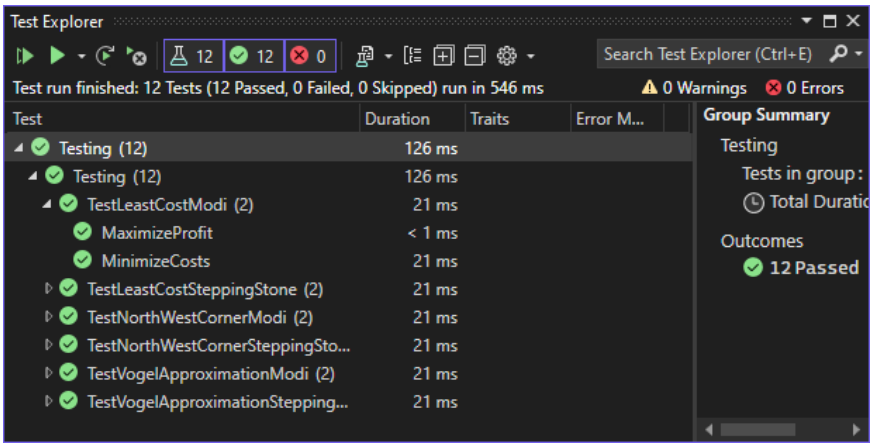
Методи класу Solver:

- Solve() – виконує повний розв’язок транспортної задачі. Послідовно викликаються методи ConvertToMaxProblem() (якщо цільова функція – Max), BalanceProblem(), GetReferencePlan(), GetOptimalPlan(), ConvertToMinProblem (якщо задачу було зведено до задачі максимізації прибутку).
- SolveWithSteps() – виконує повний розв’язок транспортної задачі. Копії проміжних станів задачі і планів транспортувань зберігаються в об’єкті класу SolutionSteps, що повертається даним методом.
- BalanceProblem() – виконує зведення задачі до збалансованого вигляду.
- ConvertToMinProblem() – виконує зведення задачі до задачі мінімізації витрат.
- ConvertToMaxProblem() – виконує зведення задачі до задачі максимізації прибутку.

5.2 Unit-тестування

Для проведення Unit-тестування використовується безкоштовний інструмент з відкритим кодом xUnit [2]. Для кожної комбінації методу пошуку опорного розв’язку і методу пошуку оптимального розв’язку реалізований клас із двома функціями: тестування оптимальності розв’язку транспортної задачі для мінімізації витрат і тестування оптимальності розв’язку транспортної задачі для максимізації прибутку.

Результати тестування:



5.3 Порівняння методів

Для порівняння методів було створено проект з двома класами: DataGenerator і MethodComparer. В класі DataGenerator наявний метод GenerateRandomProblem(), який повертає випадково згенеровану транспортну задачу з вказаною розмірністю і обмеженнями випадкових чисел. В класі MethodComparer наявний метод CompareMethods, який генерує задану кількість випадкових транспортних задач за допомогою класу DataGenerator і для всіх пар методів пошуку опорного розв’язку і методів пошуку оптимального розв’язку вирішує згенеровані задачі. Після чого, для кожної пари обчислюється середня швидкість виконання і середня кількість ітерацій. Для п’яти згенерованих випадкових задач результати наступні:

N, M	Метод пошуку опорного рішення	Метод пошуку оптимального рішення	Середній час виконання, с	Середня кількість ітерацій
30	Метод північно-західного кута	Розподільний метод	20,389	690
		Метод потенціалів	0,037	128
	Метод найменшої вартості	Розподільний метод	4,276	136
		Метод потенціалів	0,01	46
	Метод Фогеля	Розподільний метод	2,196	75
		Метод потенціалів	0,008	31
40	Метод північно-західного кута	Розподільний метод	127,978	1228
		Метод потенціалів	0,149	190
	Метод найменшої вартості	Розподільний метод	22,173	222
		Метод потенціалів	0,035	65
	Метод Фогеля	Розподільний метод	9,252	108
		Метод потенціалів	0,026	42
50	Метод північно-західного кута	Розподільний метод	319,439	1512
		Метод потенціалів	0,29	249
	Метод найменшої вартості	Розподільний метод	53,166	210
		Метод потенціалів	0,077	82
	Метод Фогеля	Розподільний метод	29,367	134
		Метод потенціалів	0,051	49
60	Метод північно-західного кута	Розподільний метод	973,348	2329
		Метод потенціалів	0,563	314
	Метод найменшої вартості	Розподільний метод	142,706	326
		Метод потенціалів	0,128	85
	Метод Фогеля	Розподільний метод	66,752	157
		Метод потенціалів	0,111	63

5.4 Програмний застосунок

Панель з користувацьким інтерфейсом застосунку містить вкладки «Вхідні дані», «Балансування задачі», «Опорний розв’язок», «Оптимальний розв’язок», «Результат». Для переходу між вкладками використовується клас TabSelector. В ньому присутнє перерахування Tab, значення якого відповідають наведеним вкладкам. Функція OpenTab() з параметром типу Tab активує об’єкти розміщенні на заданій вкладці і деактивує усі інші.

Для позначення постачальників і споживачів наявні класи Supplier і Customer відповідно. Клас Supplier містить поля Name (ім’я) і Supply (обсяг поставок ресурсу). Клас Customer містить поля Name (ім’я) і Demand (обсяг потреби в ресурсі).

5.4.1 Відображення таблиці

Для відображення заголовку таблиці використовуються клас HeaderCellUI. У ньому присутнє поле Caption – текст заголовку. Для відображення клітки таблиці використовується клас TableCellUI. В цьому класі присутні поля Cost (ціна) і Allocation (обсяг перевезень).

Для відображення таблиці використовується клас TableUI. Клас містить наступні поля:

- grid – елемент користувацького інтерфейсу, в якому елементи розташовані в вигляді сітки.
- tableCells – список значень кліток.

Методи:

- CreateTable() – заповнює сітку заголовками і клітками таблиці.

– DisplayTable() – відображає в користувацькому інтерфейсі задачу, план перевезень або транспортну таблицю (наявні три перевантаження методу).

5.4.2 Введення даних

Клас SolutionParametersUI використовується для вибору параметрів розв’язку транспортної задачі. В ньому наявні наступні поля:

– ReferencePlanMethod – метод пошуку опорного плану.
– OptimalPlanMethod – метод пошуку оптимального плану.
– TargetFunction – цільова функція.

Клас FileInputUI використовується для завантаження даних з файлів. Поля класу:

– InputFileSelected – логічне значення, яке позначає чи в випадному меню вибрано деякий файл.
– inputFiles – список файлів.

Методи класу:

– LoadFileList() – знаходить список назв файлів типу JSON із папки вхідних файлів і заповнює цими значеннями випадний список.
– LoadSelectedFileData() – завантажує дані із файлу, який обраний в випадному списку.

Для вводу і відображення даних про постачальників, споживачів, вартості перевезень використовуються класи SuppliersInputUI, CustomersInputUI, CostsInputUI. Реалізація класів майже ідентична.

Поля класу SuppliersInputUI:

– amountInputField – поле для вводу кількості постачальників.
– inputFields – поля для вводу імен і об’ємів пропозиції постачальника.

– inputFieldTemplate – шаблон поля для вводу даних постачальника.

Методи класу SuppliersInputUI:

– DisplaySuppliersData() – відображає інформацію про постачальників в полях для вводу.
– GetSuppliersData() – зчитує інформацію про споживачів із полей для воду.
– GenerateInputFields() – створює задану кількість полей для воду даних про постачальників.

Клас InputTab (вкладка користувацького інтерфейсу «Вхідні дані») виконує функції введення та відображення в користувацькому інтерфейсі вхідних даних.

Поля класу InputTab:

– suppliers – список постачальників.
– customers – список споживачів.
– costs – список вартостей транспортувань.
– suppliersInputUI – елемент користувацького інтерфейсу для вводу і відображення даних про постачальників
– customersInputUI – елемент користувацького інтерфейсу для вводу і відображення даних про споживачів.
– costsInputUI – елемент користувацького інтерфейсу для вводу і відображення даних про вартості транспортування одиниці ресурсу.
– inputFileUI – елемент користувацького інтерфейсу, що виконує вибір вхідного файлу і завантаження із нього даних.
– solutionParametersUI – елемент користувацького інтерфейсу для вибору методу пошуку опорного і оптимального плану, а також цільової функції.
– balanceTab, referenceTab, iterationsTab, resultTab – посилання на об’єкти вкладок із рішенням задачі.

Методи класу:

- LoadInputFileData() – завантаження вхідних даних задачі з файлу, обраного користувачем в елементі inputFileUI.
- UpdateSuppliersAmount() – викликається при оновленні значення поля для вводу кількості постачальників. Якщо кількість більша за поточну, то зайві постачальники видаляються. В протилежному випадку, додаються нові постачальники з стандартними значеннями. Після цього зміни постачальників і вартостей відображаються в користувацькому інтерфейсі.
- UpdateCustomersAmount() – метод аналогічний UpdateSuppliersAmount(), але замість постачальників використовуються споживачі.
- UpdateSuppliers(), UpdateCustomers() – методи, які викликаються при натисканні кнопок оновлення даних про постачальників (споживачів). Для оновлення даних, інформація з відповідних полей для вводу зчитується і записується в поля suppliers, customers, costs. Також в користувацькому інтерфейсі відображаються зміни для полей вводу вартостей (оновлюються імена постачальників і споживачів).
- Solve() – виконує розв’язок задачі, викликається при натисканні кнопки «Знайти розв’язок». На основі значень suppliers, customers, costs створюється об’єкт класу Problem. Цільова функція, метод пошуку опорного плану, метод розв’язку отримуються із solutionParametersUI. Створюється об’єкт класу Solver і виконується метод SolveWithSteps(). Якщо в розв’язку присутній фіктивний постачальник або споживач, то вносяться відповідні зміни в відображення постачальників та споживачів. На основі отриманого розв’язку та кроків заповнюються наступні вкладки.

5.4.3 Відображення результатів і проміжних кроків

Відображення результатів та проміжних кроків відбувається у наступних класах, кожен з яких відповідає вкладці користувацького інтерфейсу:

- BalanceTab – вкладка «Балансування задачі».
- ReferencePlanTab – вкладка «Опорний розв’язок».
- IterationsTab – вкладка «Оптимальний розв’язок».
- ResultTab – вкладка «Результати».

У класах BalanceTab і ReferencePlanTab знаходиться по одному методу CreateTable(), що створює об’єкт класу TableUI і заповнює його. Для BalanceTab() використовується перевантаження методу DisplayTable(Problem problem), а для ReferencePlanTab перевантаження DisplayTable(Plan plan).

Методи класу IterationsTab:

- CreateTable() – створення таблиці.
- PrevStep() – перехід на попередній крок методу пошуку оптимального плану.
- PrevStep() – перехід на наступний крок методу пошуку оптимального плану.
- DisplayCurrentStep() – відображає поточний крок: номер кроку і відповідний опорний план.

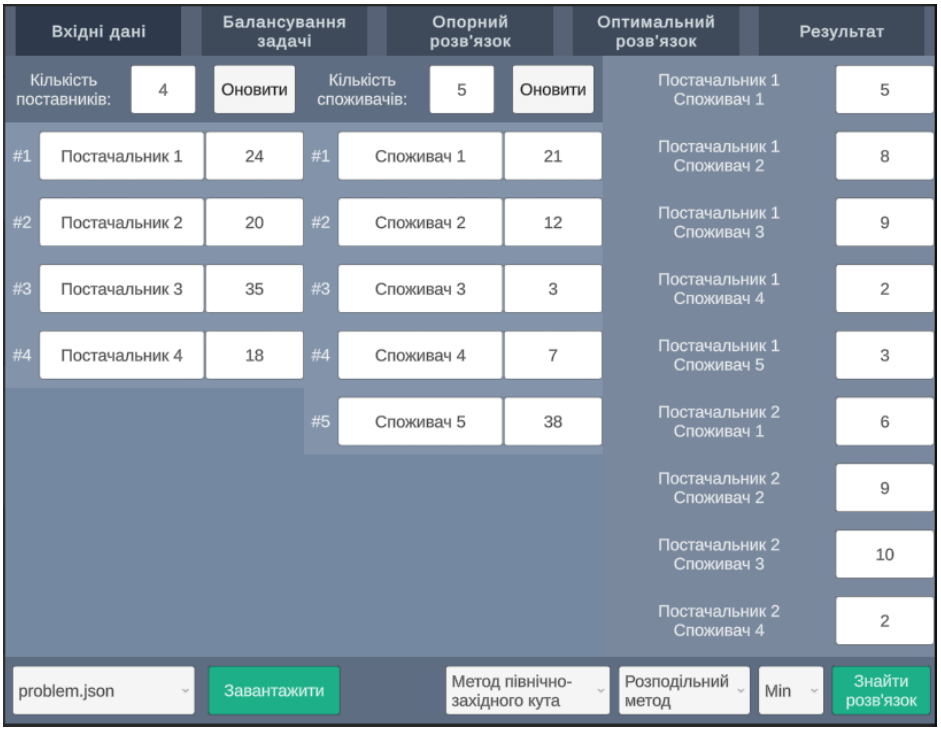
Поля класу ResultTab:

- totalCostText – текстовий елемент інтерфейсу, позначає загальну вартість оптимального плану.
- shipmentTextTemplate – шаблон для відображення перевезення.

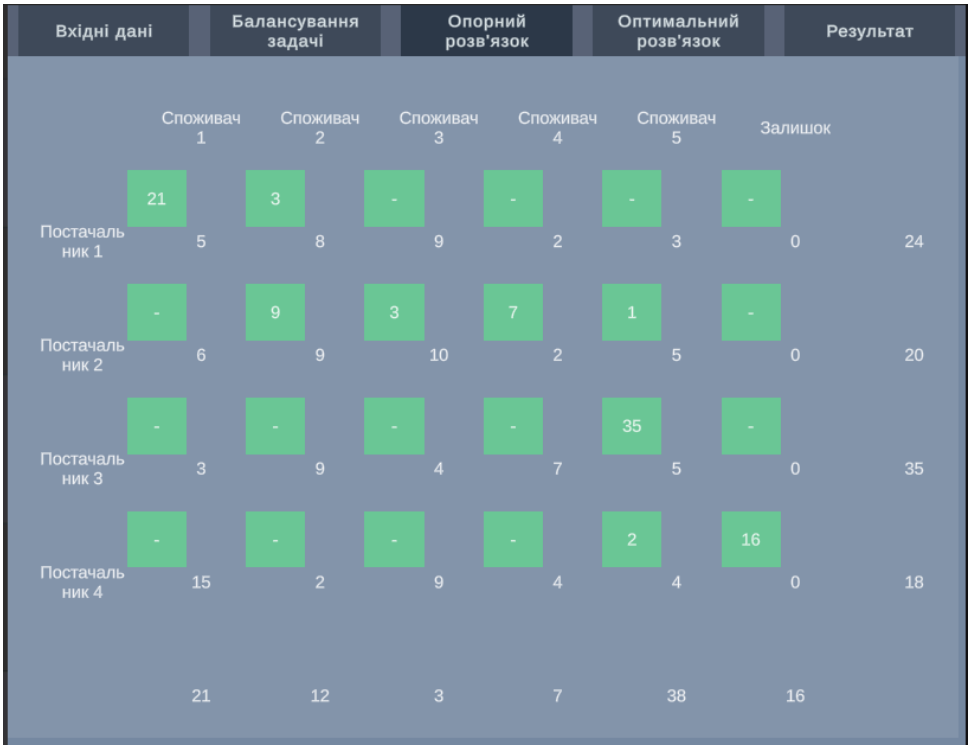
У класі ResultTab за відображення результату відповідає метод DisplayResults().

5.6 Робота програмного застосунку

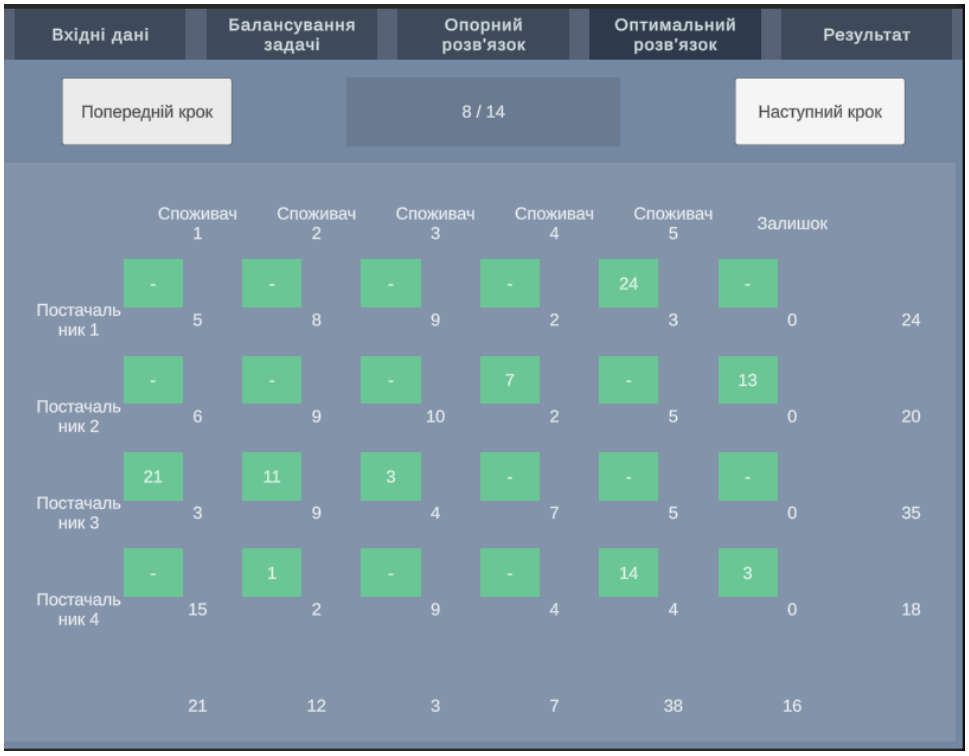
Вкладка «Вхідні дані»:



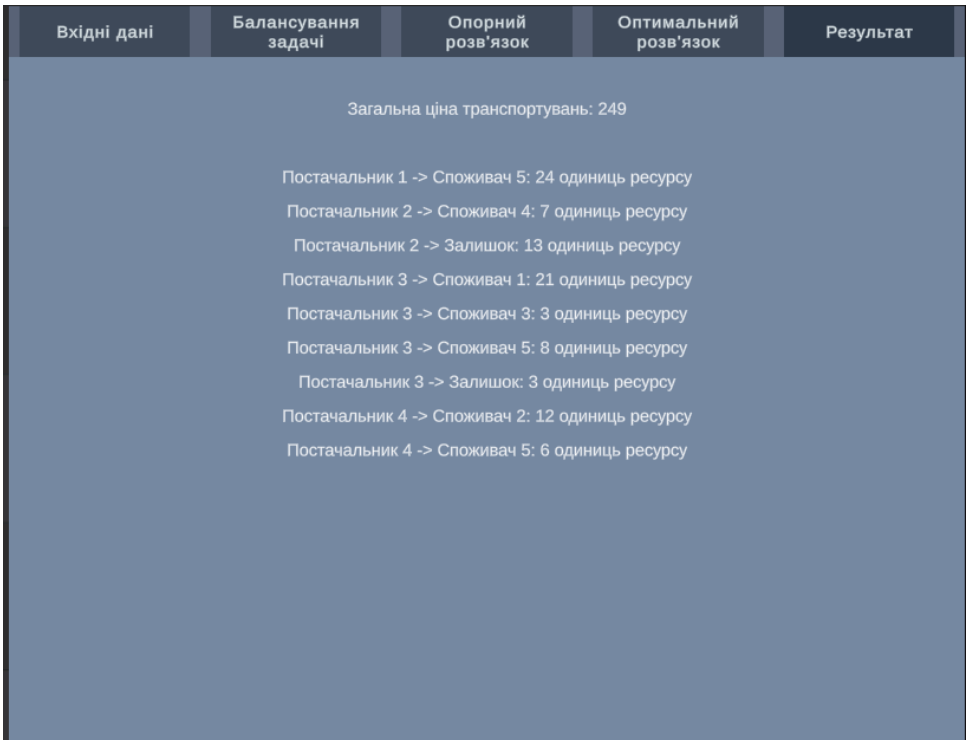
Вкладка «Опорний розв’язок»:



Вкладка «Оптимальний розв’язок», восьмий крок алгоритму:



Вкладка «Результат»:



ВИСНОВКИ

Метою кваліфікаційної роботи була розробка застосунку для розрахунку оптимального плану транспортувань ресурсів від постачальників до споживачів, а також теоретичний аналіз та порівняння часу виконання існуючих методів розв’язання даної задачі.

В ході виконання роботи були досягнуті наступні результати:

- Було проаналізовано теоретичний матеріал, що стосується математичної моделі, методів розв’язку та модифікацій транспортної задачі.
- Було створено бібліотеку для вирішення транспортної задачі, в якій були реалізовані різні методи пошуку опорного розв’язку (метод північно-західного кута, метод найменшої вартості, метод апроксимації Фогеля), різні методи пошуку оптимального розв’язку (розподільний метод, метод потенціалів), різні варіації математичної моделі (незбалансована задача, задача максимізації прибутку).
- Було створено користувацький інтерфейс та програмний застосунок, що використовує реалізовану бібліотеку і вирішує задану користувачем задачу. При цьому, у виконаній програмі можна детально переглянути усі кроки пошуку рішення.

Реалізовану програму доцільно використовувати для прикладних задач, які схожі на транспортну задачу. Наприклад, логістика підприємства, виробництво, промисловість тощо. Також програмний застосунок підходить для навчання, адже користувач отримує повний розв’язок задачі різними методами із проміжними кроками. Програма має простий та зрозумілий користувацький інтерфейс, що сприяє можливості впровадження в розглянуті галузі.

Роботу доцільно продовжувати, адже існують інші варіації досліджуваної задачі, які не були розглянуті, але могли б значно розширити функціонал та сферу застосування програми.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. A tour of the C# language [Електронний ресурс] – Режим доступу до ресурсу: https://docs.microsoft.com/en-us/dotnet/csharp/tour-of-csharp/
2. xUnit [Електронний ресурс] – Режим доступу до ресурсу: https://en.wikipedia.org/wiki/XUnit.net
3. Genetic Algorithm [Електронний ресурс] – Режим доступу до ресурсу: https://en.wikipedia.org/wiki/Genetic_algorithm
4. Stepping Stone | Transportation Problem [Електронний ресурс] – Режим доступу до ресурсу: https://www.educationlessons.co.in/notes/stepping-stone
5. Гольштейн Е. Г., Юдин Д. Б. Задачи линейного программирования транспортного типа. 1969.
6. Computational Optimal Transport. Gabriel Peyré, Marco Cuturi. 2018.
7. L. R. Ford Jr., D. R. Fulkerson. Flows in Networks. 1962.
8. Кузнецов А. В, Холод Н. И., Костевич Л.С. Руководство к решению задач по математическому программированию. 1978.
9. Рейнфельд, Фогель. Математическое программирование. Методы решения производственных и транспортных задач. 1960.