НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
ІМЕНІ ІГОРЯ СІКОРСЬКОГО»

Факультет прикладної математики

Кафедра прикладної математики

Звіт

із лабораторної роботи

із дисципліни «АЛГОРИТМИ І СИСТЕМИ КОМП'ЮТЕРНОЇ МАТЕМАТИКИ 1.МАТЕМАТИЧНІ АЛГОРИТМИ»

на тему:

Транспортна задача

Виконав:
студент групи КМ-13
Онищенко В.С.

Перевірила:
асистент кафедри ПМА
Ковальчук-Химюк Л.О.

Київ — 2024

ЗМІСТ

1. ВСТУП

Мета роботи: дослідження методів розв’язання транспортної задачі та практичне застосування цих методів для розв’язку задачі лінійного програмування на електронно-обчислювальній машині (ЕОМ) з використанням систем комп’ютерної математики (СКМ). Для розробки використати мови Python [1] та Octave[2]

2. Порядок виконання роботи

1. Ознайомитися з теоретичним матеріалом і методичними вказівками до роботи.

2. Одержати варіант індивідуального завдання /вихідні дані для задачі, метод розв’язання задачі/. Формалізувати задачу лінійного програмування: скласти математичну модель у загальному вигляді; використовуючи конкретні числові дані перетворити вихідну математичну модель до вигляду, що допускає застосування підпрограм СКМ.

3. Розробити блок-схему алгоритму розв’язання задачі.

4. Написати, налагодити і виконати програму. У програмі передбачити опис усіх використовуваних масивів і введення значень їх елементів, вивід вихідних даних, результатів розрахунків і значення ознаки результату.

5. Розв’язати задачу лінійного програмування.

6. Оформити звіт по роботі.

3. Основна частина

3.1. Вихідні дані по роботі.

Умови для варіанту 9 зазначені в таблицях 3.1.1 та 3.1.2

Таблиця 3.1.1 – умови варіанту 9, транспортна здача

Варіант	Транспортні витрати (Матриця C)	Об’єм виробництва (вектор a)	Об’єм потреб (вектор b)
9	30 24 11 12 25 26 4 29 20 24 27 14 14 10 18 6 14 28 8 2	21 19 15 25	15 15 15 15 20

Таблиця 3.1.2 – Умови варіанту 9, транспортна задача з обмеженнями на пропускну здатність

Варіант	Транспортні витрати	Обмеження на перевезення (матриця D)	Об’єм виробництва (вектор a)	Об’єм потреб (вектор b)
9	5 4 20 5 14 19 20 2 4 5 9 6 8 19 1 7 15 20 6 11	5 20 14 25 13 19 22 15 4 12 6 8 15 5 10 14 7 2 23 20	60 50 40 20	10 50 40 50 20

3.2. Типовий перелік вимог до ПЗ

Програмне забезпечення, розроблюване в рамках виконання кожної лабораторної роботи, повинно задовольняти низку вимог, які можна розділити на *обов'язкові* (які ПЗ повинно задовольняти незалежно від лабораторної роботи) та *варіативні* (які для кожної лабораторної роботи унікальні). До обов'язкових вимог належать:

- У програмі повинно бути передбачено перевірки на некоректне введення для всіх полів введення, зокрема:

    1. порожнє введення;
    2. синтаксично некоректне введення (наприклад, літери в полі для числових коефіцієнтів);
    3. уведення спеціальних символів;
    4. уведення чисел, які перевищують максимальний розмір для чисел відповідного типу даних (для перевірки на переповнення розрядної сітки).

    У випадку некоректного введення повинно з'являтися діалогове вікно з відповідним повідомленням.

- Для всіх полів введення повинно бути визначено гранично допустиму кількість символів (для числових полів — гранично допустимі значення).

- У програмі повинно відслідковуватися переповнення розрядної сітки під час виконання обчислень. У випадку переповнення повинно з'являтися діалогове вікно з відповідним повідомленням.

- У графічному інтерфейсі користувача повинно бути передбачено можливість гнучкого налаштування розмірності розв’язуваної задачі (наприклад, можливість зміни розмірності матриці чи кількості складів у транспортній задачі).

До варіативних вимог належать вимоги щодо перевірки на коректне опрацювання виключних ситуацій, які можуть виникати під час застосування заданого методу до розв’язання поставленої задачі (наприклад, коли сума заявок не збігається з сумою ресурсів у транспортній задачі, нижня межа інтегрування перевищує верхню тощо).

3.3. Опис методів

3.3.1. Оптимізація розподілу ресурсів

Постановка задачі

Задача оптимізації розподілу ресурсів є ключовою складовою в управлінні ланцюгами поставок. Метою є забезпечення ефективного використання доступних ресурсів для доставки продукції від кількох постачальників до споживачів при мінімізації витрат [3].

Математична постановка задачі

Розглянемо систему з p джерел постачання S_k ($k = 1, \ldots, p$) і q точок споживання T_l ($l = 1, \ldots, q$), де кожне джерело має обсяг ресурсів r_k, а кожна точка споживання має потребу в обсязі d_l. Вартість переміщення одиниці ресурсу між S_k і T_l позначається через c_{kl}.

Задача формулюється наступним чином:

$$\text{Мінімізувати:} \quad C = \sum_{k=1}^{p} \sum_{l=1}^{q} c_{kl} z_{kl}$$

при виконанні умов:

$$\sum_{l=1}^{q} z_{kl} \leq r_k, \quad \forall k,$$

$$\sum_{k=1}^{p} z_{kl} \geq d_l, \quad \forall l,$$

$$z_{kl} \geq 0, \quad \forall k, l.$$

де z_{kl} — кількість ресурсу, переміщеного між S_k і T_l.

Методи вирішення задачі | Метод Північно-Західного кута

Метод Північно-Західного кута — це простий алгоритм побудови початкового розподілу для задач оптимізації. Він полягає у поступовому заповненні матриці перевезень, починаючи з верхнього лівого кута, з урахуванням обмежень попиту та пропозиції.

Етапи методу:

1. Почати заповнення таблиці з верхнього лівого кута, призначаючи максимально можливий обсяг перевезення, обмежений r_k і d_l.
2. Відповідно до використаних обсягів зменшити доступний ресурс джерела r_k або попит точки d_l.
3. Перейти до наступного невикористаного елемента таблиці праворуч або вниз.
4. Повторювати, поки всі ресурси і попити не будуть розподілені.

Метод Північно-Західного кута підходить для створення початкового плану, який далі можна покращувати за допомогою методів оптимізації.

3.3.2. Обмеження логістичної мережі

Додаткові обмеження, такі як пропускна здатність маршрутів b_{kl}, можуть бути включені в модель [3]:

$$\text{Мінімізувати} \quad C = \sum_{k=1}^{p} \sum_{l=1}^{q} c_{kl} z_{kl}$$

$$\text{з обмеженнями:} \quad \sum_{l=1}^{q} z_{kl} \leq r_k, \quad k = 1, \ldots, p,$$

$$\sum_{k=1}^{p} z_{kl} \geq d_l, \quad l = 1, \ldots, q,$$

$$0 \leq z_{kl} \leq b_{kl}, \quad \forall k, l.$$

Ця модель дозволяє враховувати реальні обмеження, забезпечуючи більш точне планування. Алгоритм включає побудову початкового плану, перевірку його оптимальності та ітеративне вдосконалення до досягнення найкращого результату.

3.4. Побудова математичної моделі

3.4.1. Транспортна задача без обмежень

Формулювання задачі

Цільова функція, яке треба мінімізувати

$$\begin{aligned}\min Z = {} & 30x_{1,1} + 24x_{1,2} + 11x_{1,3} + 12x_{1,4} + 25x_{1,5} + \\ & 26x_{2,1} + 4x_{2,2} + 29x_{2,3} + 20x_{2,4} + 24x_{2,5} + \\ & 27x_{3,1} + 14x_{3,2} + 14x_{3,3} + 10x_{3,4} + 18x_{3,5} + \\ & 6x_{4,1} + 14x_{4,2} + 28x_{4,3} + 8x_{4,4} + 2x_{4,5}.\end{aligned}$$

Обмеження за умовою

1. Обмеження постачання:

$$\begin{aligned}\sum_{j=1}^{5} x_{1,j} &= 21, \\ \sum_{j=1}^{5} x_{2,j} &= 19, \\ \sum_{j=1}^{5} x_{3,j} &= 15, \\ \sum_{j=1}^{5} x_{4,j} &= 25.\end{aligned}$$

2. Обмеження попиту:

$$\sum_{i=1}^{4} x_{i,1} = 15,$$
$$\sum_{i=1}^{4} x_{i,2} = 15,$$
$$\sum_{i=1}^{4} x_{i,3} = 15,$$
$$\sum_{i=1}^{4} x_{i,4} = 15,$$
$$\sum_{i=1}^{4} x_{i,5} = 20.$$

3. Обмеження невід'ємності:

$$x_{i,j} \geq 0, \quad \forall i, j.$$

3.4.2. Транспортна задача із обмеженнями

Цільова функція, яку треба мінімізувати

$$\begin{aligned} \min Z = {} & 5x_{1,1} + 4x_{1,2} + 20x_{1,3} + 5x_{1,4} + 14x_{1,5} + \\ & 19x_{2,1} + 20x_{2,2} + 2x_{2,3} + 4x_{2,4} + 5x_{2,5} + \\ & 9x_{3,1} + 6x_{3,2} + 8x_{3,3} + 19x_{3,4} + x_{3,5} + \\ & 7x_{4,1} + 15x_{4,2} + 20x_{4,3} + 6x_{4,4} + 11x_{4,5}. \end{aligned}$$

Обмеження за умовою

1. Обмеження постачання:

$$\sum_{j=1}^{5} x_{1,j} = 60,$$
$$\sum_{j=1}^{5} x_{2,j} = 50,$$
$$\sum_{j=1}^{5} x_{3,j} = 40,$$
$$\sum_{j=1}^{5} x_{4,j} = 20.$$

2. Обмеження попиту:

$$\sum_{i=1}^{4} x_{i,1} = 10,$$
$$\sum_{i=1}^{4} x_{i,2} = 50,$$
$$\sum_{i=1}^{4} x_{i,3} = 40,$$
$$\sum_{i=1}^{4} x_{i,4} = 50,$$
$$\sum_{i=1}^{4} x_{i,5} = 20.$$

3. Обмеження пропускної здатності:

$$0 \leq x_{ij} \leq d_{ij}, \quad \forall i, j,$$

4. Обмеження невід'ємності:

$$x_{i,j} \geq 0, \quad \forall i, j.$$

3.5. Блок-схеми алгоритмів методів

3.5.1. Транспортна задача

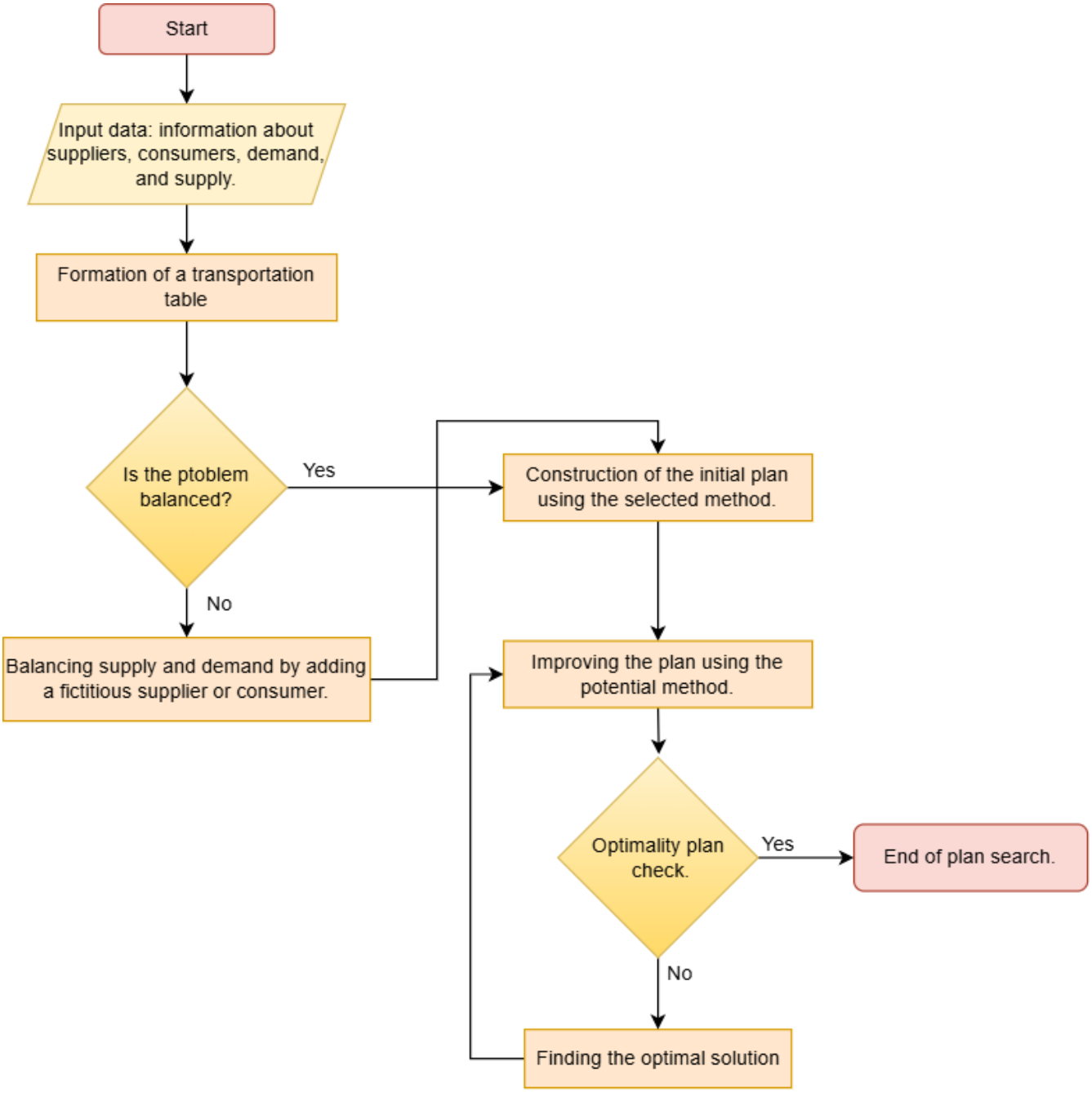

Рисунок 3.5.1.1 – Діаграма алгоримту для вирішення транспортної задачі

3.5.2. Транспортна задача з обмеженнями

Діаграма виконання алгоритму для розв'язання транспортної задачі з обмеженнями представлена на рисунку 3.5.2.1

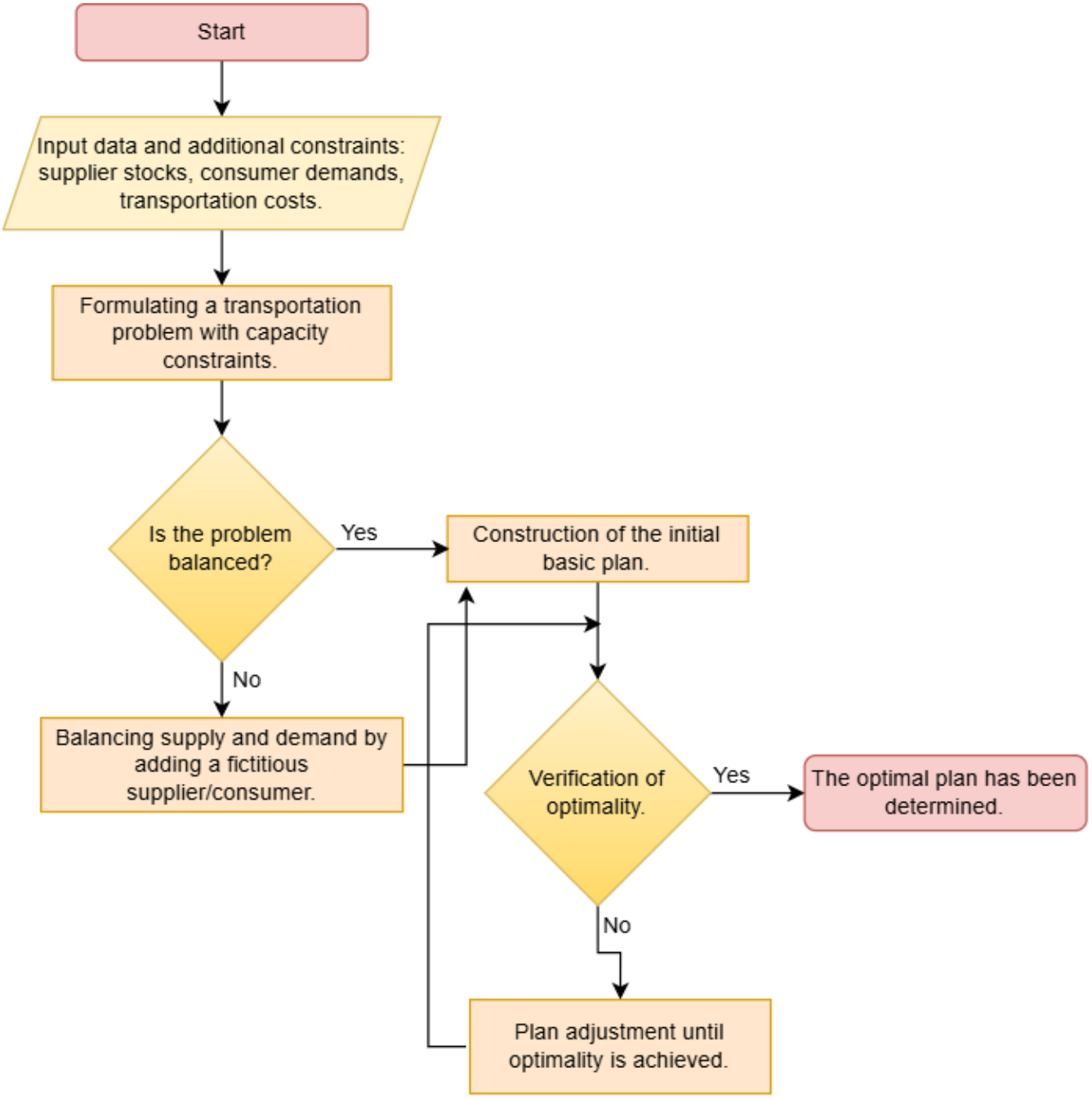

Рисунок 3.5.2.1 – Діаграма алгоритму для вирішення транспортної задачі з урахуванням обмежень.

3.6. Аналіз правильності функціонування програми

3.6.1. Python

- Кожна таблиця повинна включати числові значення та мати фіксований розмір.
- Потрібно виконати перевірку розмірів векторів a і b, а також матриці C.
- Необхідно перевірити, чи співпадає сума елементів вектора a з сумою елементів вектора b.

3.6.2. Octave

Програма, створена для GNU Octave, не здійснює перевірку коректності введених даних.

3.7. Приклад роботи програми

3.7.1. Python

Транспортна задача

За програмної реалізації розв’язку заданої задачі отримуємо наступні результати (рис. 3.7.1.1):

```
============================================================
              Результати розв'язання транспортної задачі
============================================================

Матриця розв'язку:
------------------------------------------------------------
    0.00 |     0.00 |    15.00 |     6.00 |     0.00
    4.00 |    15.00 |     0.00 |     0.00 |     0.00
    0.00 |     0.00 |     0.00 |     9.00 |     6.00
   11.00 |     0.00 |     0.00 |     0.00 |    14.00
------------------------------------------------------------

Загальна вартість перевезення:
💰 Вартість: 693.00 одиниць
------------------------------------------------------------

Деталі постачань (постачальник -> споживач):
🚚 Постачальник P1 -> Споживач P3: 15.00 одиниць
🚚 Постачальник P1 -> Споживач P4: 6.00 одиниць
🚚 Постачальник P2 -> Споживач P1: 4.00 одиниць
🚚 Постачальник P2 -> Споживач P2: 15.00 одиниць
🚚 Постачальник P3 -> Споживач P4: 9.00 одиниць
🚚 Постачальник P3 -> Споживач P5: 6.00 одиниць
🚚 Постачальник P4 -> Споживач P1: 11.00 одиниць
🚚 Постачальник P4 -> Споживач P5: 14.00 одиниць
============================================================
```

Рисунок 3.7.1.1 – Вихідні дані розв’язання транспортної задачі (Python)

Транспортна задача із обмеженнями

За програмної реалізації розв’язку заданої задачі отримуємо наступні результати (рис. 3.7.1.2):

```
============================================================
              Результати розв'язання транспортної задачі
============================================================

Матриця обмеження:
------------------------------------------------------------
    5.00 |    20.00 |    14.00 |    25.00 |    13.00
   19.00 |    22.00 |    15.00 |     4.00 |    12.00
    6.00 |     8.00 |    15.00 |     5.00 |    10.00
   14.00 |     7.00 |     2.00 |    23.00 |    20.00
------------------------------------------------------------

Матриця розв'язку:
------------------------------------------------------------
    5.00 |    20.00 |    10.00 |    25.00 |     0.00
    0.00 |    21.00 |    15.00 |     4.00 |    10.00
    5.00 |     8.00 |    15.00 |     2.00 |    10.00
    0.00 |     1.00 |     0.00 |    19.00 |     0.00
------------------------------------------------------------

Загальна вартість перевезення:
💰 Вартість: 1,336.00 одиниць
------------------------------------------------------------

Деталі постачань (постачальник -> споживач):
🚚 Постачальник P1 -> Споживач P1: 5.00 одиниць
🚚 Постачальник P1 -> Споживач P2: 20.00 одиниць
🚚 Постачальник P1 -> Споживач P3: 10.00 одиниць
🚚 Постачальник P1 -> Споживач P4: 25.00 одиниць
🚚 Постачальник P2 -> Споживач P2: 21.00 одиниць
🚚 Постачальник P2 -> Споживач P3: 15.00 одиниць
🚚 Постачальник P2 -> Споживач P4: 4.00 одиниць
🚚 Постачальник P2 -> Споживач P5: 10.00 одиниць
🚚 Постачальник P3 -> Споживач P1: 5.00 одиниць
🚚 Постачальник P3 -> Споживач P2: 8.00 одиниць
🚚 Постачальник P3 -> Споживач P3: 15.00 одиниць
🚚 Постачальник P3 -> Споживач P4: 2.00 одиниць
🚚 Постачальник P3 -> Споживач P5: 10.00 одиниць
🚚 Постачальник P4 -> Споживач P2: 1.00 одиниць
🚚 Постачальник P4 -> Споживач P4: 19.00 одиниць
============================================================
```

Рисунок 3.7.1.2 – Вихідні дані розв’язання транспортної задачі з обмеженнями (Python)

3.7.2. Octave

Транспортна задача

За програмної реалізації розв’язку заданої задачі отримуємо наступні результати (рис. 3.7.2.1):

```
Матриця розв'язку:
------------------------------------------------------------
     0.00 |     0.00 |    15.00 |     6.00 |     0.00 |
     4.00 |    15.00 |     0.00 |     0.00 |     0.00 |
     0.00 |     0.00 |     0.00 |     9.00 |     6.00 |
    11.00 |     0.00 |     0.00 |     0.00 |    14.00 |
------------------------------------------------------------

Загальна вартість перевезення:
Вартість: 693.00 одиниць
------------------------------------------------------------

Деталі постачань (постачальник -> споживач):
Постачальник P1 -> Споживач S3: 15.00 одиниць
Постачальник P1 -> Споживач S4: 6.00 одиниць
Постачальник P2 -> Споживач S1: 4.00 одиниць
Постачальник P2 -> Споживач S2: 15.00 одиниць
Постачальник P3 -> Споживач S4: 9.00 одиниць
Постачальник P3 -> Споживач S5: 6.00 одиниць
Постачальник P4 -> Споживач S1: 11.00 одиниць
Постачальник P4 -> Споживач S5: 14.00 одиниць
```

Рисунок 3.7.2.1 – Вихідні дані розв’язання транспортної задачі (Octave)

Транспортна задача із обмеженнями

За програмної реалізації розв’язку заданої задачі отримуємо наступні результати (рис. 3.7.2.2):

```
Матриця обмеження:
-------------------------------------------------------------
     5.00 |    20.00 |    14.00 |    25.00 |    13.00 |
    19.00 |    22.00 |    15.00 |     4.00 |    12.00 |
     6.00 |     8.00 |    15.00 |     5.00 |    10.00 |
    14.00 |     7.00 |     2.00 |    23.00 |    20.00 |
-------------------------------------------------------------

Матриця розв'язку:
-------------------------------------------------------------
     5.00 |    20.00 |    10.00 |    25.00 |     0.00 |
     0.00 |    21.00 |    15.00 |     4.00 |    10.00 |
     5.00 |     8.00 |    15.00 |     2.00 |    10.00 |
     0.00 |     1.00 |     0.00 |    19.00 |     0.00 |
-------------------------------------------------------------

Загальна вартість перевезення:
Вартість: 1336.00 одиниць
-------------------------------------------------------------

Деталі постачань (постачальник -> споживач):
Постачальник P1 -> Споживач S1: 5.00 одиниць
Постачальник P1 -> Споживач S2: 20.00 одиниць
Постачальник P1 -> Споживач S3: 10.00 одиниць
Постачальник P1 -> Споживач S4: 25.00 одиниць
Постачальник P2 -> Споживач S2: 21.00 одиниць
Постачальник P2 -> Споживач S3: 15.00 одиниць
Постачальник P2 -> Споживач S4: 4.00 одиниць
Постачальник P2 -> Споживач S5: 10.00 одиниць
Постачальник P3 -> Споживач S1: 5.00 одиниць
Постачальник P3 -> Споживач S2: 8.00 одиниць
Постачальник P3 -> Споживач S3: 15.00 одиниць
Постачальник P3 -> Споживач S4: 2.00 одиниць
Постачальник P3 -> Споживач S5: 10.00 одиниць
Постачальник P4 -> Споживач S2: 1.00 одиниць
```

Рисунок 3.7.2.2 – Вихідні дані розв’язання транспортної задачі з обмеженнями (Octave)

4. ВИСНОВКИ

У ході цієї лабораторної роботи здобуто навички застосування методів розв’язання транспортних задач та їх реалізації для лінійного програмування з використанням програмного забезпечення.

Розроблено інструменти для розв’язання двох типів оптимізаційних задач: стандартної ”Транспортної задачі” та ”Транспортної задачі з обмеженнями”. Також було впроваджено механізм перевірки введених даних для підвищення точності результатів.

За результатами роботи можемо зробити висновки щодо застосування методів розв’язання транспортних задач. Для першої задачі без обмежень було сформовано оптимальну матрицю розв’язку, в якій загальна вартість перевезення склала 693 одиниці. Система поставок показує, як ефективно можна здійснити транспортування між постачальниками і споживачами, при цьому мінімізуючи витрати. Для другого випадку, коли задачею додаються певні обмеження, отримано нову матрицю розв’язку, де загальна вартість транспортування складає 1336 одиниць, що значно більше через впровадження додаткових обмежень. Ці зміни підвищують загальні витрати на транспортування, але водночас вони можуть відображати умови реальних логістичних задач, де часто є обмеження на наявність ресурсів, транспорту, чи інших чинників. Нові обмеження значно змінили розподіл ресурсів, зокрема призвели до істотного зростання вартості, що вказує на важливість ретельного врахування таких обмежень під час моделювання подібних задач.

5. СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Python Software Foundation. *Python Documentation* (2023). `https://docs.python.org/3.12/`.
2. Eaton, J. W., Bateman, D., Hauberg, S. & Wehbring, R. *GNU Octave 4.0 Reference Manual: Free Your Numbers* (Free Software Foundation, 2015). ISBN: 978-9888381050. `https://docs.octave.org/octave-4.0.0.pdf`.
3. Johnson, R. *Supply Chain Management: Strategies and Practices* 2nd (Pearson, New York, 2024).

6. ДОДАТОК А (код роботи програм)

Python

```
import numpy as np
from scipy.optimize import linprog

costs_1 = np.array([
    [30, 24, 11, 12, 25],
    [26, 4, 29, 20, 24],
    [27, 14, 14, 10, 18],
    [6, 14, 28, 8, 2]
])

supply_1 = np.array([21, 19, 15, 25])
demand_1 = np.array([15, 15, 15, 15, 20])

costs_2 = np.array([
    [5, 4, 20, 5, 14],
    [19, 20, 2, 4, 5],
    [9, 6, 8, 19, 1],
    [7, 15, 20, 6, 11]
])

capacities = np.array([
    [5, 20, 14, 25, 13],
    [19, 22, 15, 4, 12],
    [6, 8, 15, 5, 10],
    [14, 7, 2, 23, 20]
])

supply_2 = np.array([60, 50, 40, 20])
```

```
demand_2 = np.array([10, 50, 40, 50, 20])

import numpy as np
from scipy.optimize import linprog

def check_numeric_data(C, supply, demand, capacity=None):
    if not (np.issubdtype(C.dtype, np.number) and np.all(np.isfinite(C))):
        raise ValueError("Cost matrix 'C' must contain only numeric and fi
    if not (np.issubdtype(supply.dtype, np.number) and np.all(np.isfinite(
        raise ValueError("Supply vector must contain only numeric and fini
    if not (np.issubdtype(demand.dtype, np.number) and np.all(np.isfinite(
        raise ValueError("Demand vector must contain only numeric and fini
    if capacity is not None:
        if not (np.issubdtype(capacity.dtype, np.number) and np.all(np.isf
            raise ValueError("Capacity matrix must contain only numeric an

def validate_dimensions(C, supply, demand, capacity=None):
    check_numeric_data(C, supply, demand, capacity)
    if len(supply) != C.shape[0]:
        raise ValueError("Supply vector length must match the number of ro
    if len(demand) != C.shape[1]:
        raise ValueError("Demand vector length must match the number of co
    if capacity is not None and capacity.shape != C.shape:
        raise ValueError("Capacity matrix dimensions must match those of c

def ensure_balanced_supply_demand(supply, demand):
    if not np.isclose(np.sum(supply), np.sum(demand)):
        raise ValueError("Total supply and demand must be equal.")

def adjust_imbalance(supply, demand, cost_matrix, capacity=None, high_valu
    supply_total = np.sum(supply)
    demand_total = np.sum(demand)
```

```
61
62      if supply_total > demand_total:
63          demand = np.append(demand, supply_total - demand_total)
64          cost_matrix = np.hstack((cost_matrix, np.zeros((cost_matrix.shape
65          if capacity is not None:
66              capacity = np.hstack((capacity, np.full((capacity.shape[0], 1)
67      elif demand_total > supply_total:
68          supply = np.append(supply, demand_total - supply_total)
69          cost_matrix = np.vstack((cost_matrix, np.zeros((1, cost_matrix.sha
70          if capacity is not None:
71              capacity = np.vstack((capacity, np.full((1, capacity.shape[1])
72
73      return supply, demand, cost_matrix, capacity
74
75  def transport_simple(C, supply, demand):
76      rows, cols = C.shape
77      cost_vector = C.flatten()
78
79      equality_constraints = np.zeros((rows + cols, rows * cols))
80      supply_demand_vector = np.concatenate((supply, demand))
81
82      for i in range(rows):
83          equality_constraints[i, i * cols:(i + 1) * cols] = 1
84
85      for j in range(cols):
86          equality_constraints[rows + j, j::cols] = 1
87
88      bounds = [(0, None)] * (rows * cols)
89
90      solution = linprog(cost_vector, A_eq=equality_constraints, b_eq=supply
91
92      if solution.success:
```

```
        return solution.x.reshape(rows, cols), solution.fun
    raise ValueError("No feasible solution found for the transportation pr

def transport_with_capacity(C, capacity, supply, demand):
    rows, cols = C.shape
    cost_vector = C.flatten()

    equality_constraints = np.zeros((rows + cols, rows * cols))
    supply_demand_vector = np.concatenate((supply, demand))

    upper_bounds = np.eye(rows * cols)
    capacity_vector = capacity.flatten()

    for i in range(rows):
        equality_constraints[i, i * cols:(i + 1) * cols] = 1

    for j in range(cols):
        equality_constraints[rows + j, j::cols] = 1

    bounds = [(0, None)] * (rows * cols)

    solution = linprog(cost_vector, A_eq=equality_constraints, b_eq=supply

    if solution.success:
        return solution.x.reshape(rows, cols), solution.fun
    raise ValueError("No feasible solution found for the transportation pr

def solve_transport_problem(C, supply, demand, use_capacity=False, capacit
    try:
        validate_dimensions(C, supply, demand, capacity)
        ensure_balanced_supply_demand(supply, demand)

```

```
        if use_capacity:
            if capacity is None:
                raise ValueError("Capacity matrix must be provided when us
            supply, demand, C, capacity = adjust_imbalance(supply, demand,
            return transport_with_capacity(C, capacity, supply, demand)

        supply, demand, C, _ = adjust_imbalance(supply, demand, C)
        return transport_simple(C, supply, demand)
    except ValueError as error:
        return str(error)

solution, cost = solve_transport_problem(costs_1, supply_1, demand_1, use_

print("=" * 60)
print(" " * 15 + "Результати розвязання' транспортної задачі")
print("=" * 60)
print("\Матрицяn розвязку':")
print("-" * 60)
for row in solution:
    print(" | ".join(f"{cell:8.2f}" for cell in row))
print("-" * 60)

print("\Загальнаn вартість перевезення:")
print(f"" Вартість: {cost:,.2f} одиниць")
print("-" * 60)

print("\Деталіn постачань постачальник( -> споживач):")
for i, provider in enumerate([f'P{i}' for i in range(1, len(supply_1)+1)]):
    for j, customer in enumerate([f'P{i}' for i in range(1, len(demand_1)+
        if solution[i][j] > 0:  # Show only active connections
            print(f"" Постачальник {provider} -> Споживач {customer}: {so

```

```
print("=" * 60)
solution, cost = solve_transport_problem(costs_2, supply_2, demand_2, use_

print("=" * 60)
print(" " * 15 + "Результати розв'язання транспортної задачі")
print("=" * 60)
print("\Матрицяn обмеження:")
print("-" * 60)
for row in capacities:
    print(" | ".join(f"{cell:8.2f}" for cell in row))
print("-" * 60)
print("\Матрицяn розв'язку:")
print("-" * 60)
for row in solution:
    print(" | ".join(f"{cell:8.2f}" for cell in row))
print("-" * 60)

print("\Загальнаn вартість перевезення:")
print(f"" Вартість: {cost:,.2f} одиниць")
print("-" * 60)

print("\Деталіn постачань постачальник( -> споживач):")
for i, provider in enumerate([f'P{i}' for i in range(1, len(supply_2)+1)]):
    for j, customer in enumerate([f'P{i}' for i in range(1, len(demand_2)+
        if solution[i][j] > 0:
            print(f"" Постачальник {provider} -> Споживач {customer}: {so

print("=" * 60)
```

Octave

```
costs_1 = [
    30, 24, 11, 12, 25;
    26,  4, 29, 20, 24;
    27, 14, 14, 10, 18;
     6, 14, 28,  8,  2
];

supply_1 = [21; 19; 15; 25];
demand_1 = [15; 15; 15; 15; 20];

costs_2 = [
     5,  4, 20,  5, 14;
    19, 20,  2,  4,  5;
     9,  6,  8, 19,  1;
     7, 15, 20,  6, 11
];

capacities = [
     5, 20, 14, 25, 13;
    19, 22, 15,  4, 12;
     6,  8, 15,  5, 10;
    14,  7,  2, 23, 20
];

supply_2 = [60; 50; 40; 20];
demand_2 = [10; 50; 40; 50; 20];


disp('Solving transportation problem without capacity constraints:');
try
    [x_no_capacity, cost_no_capacity] = transportation_problem_no_capacity
    fprintf("%s\n", repmat("=", 1, 60));
```

```
    fprintf("%s\n", repmat(" ", 1, 15), "Результати розвязання' транспорт
    fprintf("%s\n", repmat("=", 1, 60));
    disp(x_no_capacity);
    fprintf("\Матрицяn розвязку':\n");
    fprintf("%s\n", repmat("-", 1, 60));
    for i = 1:size(x_no_capacity, 1)
        fprintf(" %8.2f |", x_no_capacity(i, :));
        fprintf("\n");
    end
    fprintf("%s\n", repmat("-", 1, 60));

    fprintf("\Загальнаn вартість перевезення:\n");
    fprintf("Вартість: %.2f одиниць\n", cost_no_capacity);
    fprintf("%s\n", repmat("-", 1, 60));

    fprintf("\Деталіn постачань постачальник( -> споживач):\n");
    for i = 1:size(x_no_capacity, 1)
        for j = 1:size(x_no_capacity, 2)
            if x_no_capacity(i, j) > 0
                fprintf("Постачальник P%d -> Споживач S%d: %.2f одиниць\n"
            end
        end
    end

    fprintf("%s\n", repmat("=", 1, 60));

catch err
    disp('Error in solving transportation problem without capacity constra
    disp(err.message);
end
```

```
disp('Solving transportation problem with capacity constraints:');
try
    [x_with_capacity, cost_with_capacity] = transportation_problem_with_ca
    fprintf("%s\n", repmat("=", 1, 60));
    fprintf("%s\n", repmat(" ", 1, 15), "Результати розвязання' транспорт
    fprintf("%s\n", repmat("=", 1, 60));

    fprintf("\Матрицяn обмеження:\n");
    fprintf("%s\n", repmat("-", 1, 60));
    for i = 1:size(capacities, 1)
        fprintf(" %8.2f |", capacities(i, :));
        fprintf("\n");
    end
    fprintf("%s\n", repmat("-", 1, 60));

    fprintf("\Матрицяn розвязку':\n");
    fprintf("%s\n", repmat("-", 1, 60));
    for i = 1:size(x_with_capacity, 1)
        fprintf(" %8.2f |", x_with_capacity(i, :));
        fprintf("\n");
    end
    fprintf("%s\n", repmat("-", 1, 60));

    fprintf("\Загальнаn вартість перевезення:\n");
    fprintf("Вартість: %.2f одиниць\n", cost_with_capacity);
    fprintf("%s\n", repmat("-", 1, 60));

    fprintf("\Деталіn постачань постачальник( -> споживач):\n");
    for i = 1:size(x_with_capacity, 1)
        for j = 1:size(x_with_capacity, 2)
            if x_with_capacity(i, j) > 0
                fprintf("Постачальник P%d -> Споживач S%d: %.2f одиниць\n"
```

```
            end
        end
    end

    fprintf("%s\n", repmat("=", 1, 60));

catch err
    disp('Error in solving transportation problem with capacity constraint
    disp(err.message);
end
```