НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
ІМЕНІ ІГОРЯ СІКОРСЬКОГО»
Факультет прикладної математики
Кафедра прикладної математики

Звіт
із лабораторної роботи
з дисципліни «АЛГОРИТМИ І СИСТЕМИ КОМП'ЮТЕРНОЇ МАТЕМАТИКИ 1.МАТЕМАТИЧНІ АЛГОРИТМИ»
на тему
"Транспортна задача"

Виконав:	Перевірила:
студент групи КМ-01	Асистент кафедри ПМА
Іваник Ю. П.	Ковальчук-Химюк Л. О.

Київ — 2023

Зміст

Вступ 3
Основна частина 4
    Варіант 17 4
    Вимоги до ПЗ 4
    Теоретичні відомості 4
    Алгоритми розв’язання 6
Висновки 7
Відповіді на контрольні питання 8
Перелік посилань 9
Додаток А – Код програми 10

Вступ

Метою даної роботи є вивчити методи розв'язання транспортної задачі, практичне розв'язання транспортної задачі лінійного програмування на ЕОМ за допомогою СКМ.

Основна частина

Варіант 17

Транспортні витрати (матриця C)	Об'єм виробництва (вектор a)	Об'єм потреб (вектор b)
8 1 19 1 15 8 27 30 7 7 9 20 19 26 20 23 28 25 7 22	18 23 17 22	21 21 9 9 20

Вимоги до ПЗ

1. Реалізувати перевірки на некоректний ввід (порожнє введення, символи замість числа, тощо).
2. У програмі повинно бути передбачено можливість гнучкого налагодження розмірності розв'язуваної задачі.
3. Має зберігатись логічно правильне розв'язання (нижня межа інтегрування не має перевищувати верхню, тощо)

Теоретичні відомості

Транспортна задача є однією з класичних задач оптимізації, яка виникає в різних галузях, де потрібно оптимізувати розподіл ресурсів або товарів з джерела до призначення з мінімальними витратами чи максимізацією прибутку. Основні відомості про транспортну задачу включають:

1. Постановка задачі: Транспортна задача полягає у вирішенні проблеми розподілу обмежених ресурсів (наприклад, товарів чи сировини) від джерел (наприклад, фабрик або складів) до призначень (наприклад, розділових пунктів або споживачів) з мінімальними витратами або максимізацією прибутку.
2. Варіанти задачі: Транспортні задачі можуть бути поділені на прямі (мінімізація витрат) та обернені (максимізація прибутку). Також існують варіації, такі як вирішення задачі найменшої вартості, задачі максимізації прибутку тощо.

3. Основні складові: Транспортна задача включає в себе матрицю вартостей (вартість перевезення одиниці товару від джерела до призначення), вектор обсягів ресурсів (доступність товарів на джерелах) і вектор обсягів запитів (потреба в товарах на призначеннях).
4. Основна мета: Основною метою транспортної задачі є знайти такий план розподілу товарів, який задовольняє обмеженням на ресурси та попит і забезпечує оптимальний результат, такий як мінімізація загальних витрат або максимізація прибутку.
5. Методи розв'язку: Для вирішення транспортних задач використовують різні методи, включаючи симплекс-метод, методи планування, метод потенціалів, метод мінімального елемента тощо. Вибір методу може залежати від конкретної задачі та обсягів даних.
6. Застосування: Транспортні задачі мають широкі застосування в логістиці, постачанні, транспорті, виробництві та багатьох інших галузях. Вони допомагають оптимізувати розподіл ресурсів, планування перевезень та ефективність ділових процесів.
7. Результат: Результатом розв'язку транспортної задачі є оптимальний план розподілу товарів, де вказані обсяги товарів, які перевозяться від кожного джерела до призначення, і вартість цього плану.

Математична модель такої задачі має вигляд:

$$F = \sum_{i=1}^{m} \sum_{j=1}^{n} c_{ij} x_{ij} \to min$$

За умови

$$\sum_{i=1}^{m} x_{ij} = b_j \ (j = \overline{1,n}),$$

$$\sum_{j=1}^{n} x_{ij} = a_j \ (i = \overline{1,m}),$$

$$x_{ij} \geq 0 \ (i = \overline{1,m}; j = \overline{1,n})$$

Для розв’язку транспортної задачі повинна виконуватись умова:

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

Транспортні задачі є важливим інструментом для оптимізації ресурсів та вирішення проблем розподілу в різних галузях бізнесу та науки.

Алгоритми розв’язання

Імпортуємо бібліотеки Oct2py [1], numpy [2], collections [3]

Ініціалізуємо власну функцію umova, яка буде відкривати текстовий файл із умовою задачі та записувати в пам’ять, щоб оперувати далі цими даними.

Для розв’язку мовою Python створимо функцію transport яка буде приймати початкові умови та за допомогою методу потенціалів [4] знаходити оптимальний план перевезень та мінімальну вартість перевезень. Більш детально про цю функцію:

1. **assert sum(supply) == sum(demand)**: Ця перевірка забезпечує, що сума доступних ресурсів (постачання) дорівнює сумі запитів (попиту), що є обов'язковою умовою для розв'язання транспортної задачі.
2. **s, d, C = np.copy(supply), np.copy(demand), np.copy(costs)**: Створюються копії вхідних даних **supply**, **demand**, і **costs** для подальшої обробки. Вони будуть використовуватися для розрахунків без зміни початкових даних.
3. **n, m = C.shape**: Визначається кількість джерел (**n**) і кількість призначень (**m**) на основі розміру матриці вартостей **C**.
4. **X = np.zeros((n, m))**: Створюється пуста матриця **X**, яка представлятиме оптимальний план розподілу товарів.
5. **indices = [(i, j) for i in range(n) for j in range(m)]**: Створюється список індексів у вигляді пар (i, j), які представляють всі можливі комбінації джерел і призначень.
6. **xs = sorted(zip(indices, C.flatten()), key=lambda a_b: a_b[1])**: Створюється відсортований список пар (індекс, вартість) з усіх можливих варіантів розподілу товарів за зростанням вартості.
7. Наступні кроки реалізують алгоритм розподілу товарів. Деталі алгоритму включають в себе обчислення потенціалів (змінних u і v), побудову матриці чутливості (S), вибір мінімального від'ємного елементу і подальшу оптимізацію опорного плану.
8. Основна частина цього алгоритму має на меті визначити оптимальний план розподілу товарів **X** з мінімальними витратами. Кроки циклу повторюються, доки не буде знайдено оптимальний план.
9. Функція повертає оптимальний план розподілу товарів **X** і сумарну вартість розподілу (мінімальну вартість перевезень).

Для розв’язку мовою Octave перетворимо матрицю і вектори у одновимірні списки. Створимо матрицю рівностей у вектор правих частин рівностей. У циклі заповнюємо матрицю так, щоб кожен рядок відповідав обмеженню постачання або попиту. Нижньою межею розв’язку буде 0, а верхньою нескінченність. Викликаємо вбудовану функцію glpk для вирішення задачі. Вона поверне оптимальний план перевезень. Порахуємо мінімальну вартість перевезень та виведемо її на екран разом із оптимальним планом.

Висновки

У результаті розв'язку транспортної задачі досягнуто таких результатів:

```
Розв'язок на мові Python

Оптимальний план перевезень:
[[ 0. 18.  0.  0.  0.]
 [ 3.  0.  0.  0. 20.]
 [17.  0.  0.  0.  0.]
 [ 1.  3.  9.  9.  0.]]
Мінімальна вартість перевезень: 730.0
```

```
Розв'язок на мові Octave

Оптимальний план перевезень:
[[ 0.  0.  0.  0.  3.]
 [18.  3. 20.  0.  9.]
 [ 0.  0. 17.  0.  9.]
 [ 0.  0.  0.  1.  0.]]
Мінімальна вартість перевезень: 730.0
```

Як бачимо, ми отримали різні оптимальні плани, але однакову мінімальну вартість перевезень. Це сталось через те що ми використали різні методи розв'язання задачі. Проте оскільки вартість однакова можемо зробити висновок, що в обох розв'язках ми дійсно знайшли мінімальну вартість.

Відповіді на контрольні питання

1. Формулювання транспортної задачі полягає в знаходженні оптимального способу перевезення товарів з одного місця в інше при мінімізації витрат. Ранг системи рівнянь визначається як (m + n - 1), де m - кількість рядків у таблиці запитів, а n - кількість стовпців.
2. Постановка транспортної задачі з обмеженнями на пропускну здатність включає обмеження на кількість товарів, які можуть бути перевезені через кожний маршрут. Умови можливості розв'язання полягають у тому, що сума запасів (пропонованих товарів) повинна дорівнювати сумі вимог (заявок) і має існувати хоча б один спосіб забезпечити цю рівність.
3. Розв'язання задач з надлишком запасів чи заявок використовується, коли сума запасів або заявок більша, ніж сума протилежних значень у таблиці запитів. У такому випадку вводять фіктивні запити або заявки для досягнення рівності.
4. Методи знаходження опорного плану включають методи "південно-західного кута", "мінімальної вартості" та "найменшого розходження". Вони використовуються для визначення початкового опорного плану.
5. Сутність угорського методу полягає в знаходженні оптимального розв'язку транспортної задачі шляхом визначення мінімального кількості клітин, які мають бути обрані для перевезення, при цьому забезпечуючи рівновагу між запасами і вимогами.
6. Метод потенціалів використовується для знаходження опорного плану та обчислення мінімального вартісного покриття. Він ґрунтується на визначенні потенціалів для рядків і стовпців таблиці запитів і використовується для оптимізації вартостей перевезення товарів в транспортній задачі.

Перелік посилань

1. Oct2py - https://pypi.org/project/oct2py/
2. Numpy - https://numpy.org
3. Collections - https://www.digitalocean.com/community/tutorials/python-counter-python-collections-counter
4. Метод потенціалів - https://elib.lntu.edu.ua/sites/default/files/elib_upload/ЕНП_ММОЕС_19/page20.html

Додаток А – Код програми

Вміст файлу main.py :

```
from oct2py import octave
import numpy as np
from collections import Counter


def umova():
    with open('data.txt', 'r', encoding='utf-8') as file:
        lines = [line.strip() for line in file.readlines()]

    # Знайти рядки з мітками
    indices = {label: lines.index(label) for label in ['вектор a', 'вектор b',
'матриця C']}

    # Читання векторів та матриці
    a = np.fromstring(lines[indices['вектор a'] + 1], sep=',')
    b = np.fromstring(lines[indices['вектор b'] + 1], sep=',')
    C_start, C_end = indices['матриця C'] + 1, len(lines)
    C = np.genfromtxt(lines[C_start:C_end], delimiter=',')

    return a, b, C


def transport(supply, demand, costs):
    assert sum(supply) == sum(demand)

    s, d, C = np.copy(supply), np.copy(demand), np.copy(costs)
    n, m = C.shape
    X = np.zeros((n, m))
    indices = [(i, j) for i in range(n) for j in range(m)]
    xs = sorted(zip(indices, C.flatten()), key=lambda a_b: a_b[1])

    for (i, j), _ in xs:
        if d[j] != 0:
            remains = s[i] - d[j] if s[i] >= d[j] else 0
            grabbed = s[i] - remains
            X[i, j], s[i], d[j] = grabbed, remains, d[j] - grabbed

    while True:
        u = np.array([np.nan] * n)
        v = np.array([np.nan] * m)
        S = np.zeros((n, m))

        nonzero = list(zip(*np.where(X > 0)))
        u[nonzero[0][0]] = 0

        while any(np.isnan(u)) or any(np.isnan(v)):
```

```
            for i, j in nonzero:
                if np.isnan(u[i]) and not np.isnan(v[j]):
                    u[i] = C[i, j] - v[j]
                elif not np.isnan(u[i]) and np.isnan(v[j]):
                    v[j] = C[i, j] - u[i]

        S = C - u[:, None] - v
        s = np.min(S)

        if s >= 0:
            break

        start = tuple(np.argwhere(S == s)[0])
        T = np.copy(X)
        T[start] = 1

        while True:
            _xs, _ys = np.nonzero(T)
            xcount, ycount = Counter(_xs), Counter(_ys)

            for x in xcount.keys():
                if xcount[x] <= 1: T[x, :] = 0
            for y in ycount.keys():
                if ycount[y] <= 1: T[:, y] = 0

            if all(x > 1 for x in xcount.values()) and all(y > 1 for y in ycount.values()):
                break

        path = [start]
        fringe = set(tuple(p) for p in np.argwhere(T > 0))
        size = len(fringe)

        while len(path) < size:
            last = path[-1]
            fringe.remove(last)
            next_ = min(fringe, key=lambda x_y: abs(last[0]-x_y[0]) + abs(last[1]-x_y[1]))
            path.append(next_)

        neg, pos = path[1::2], path[::2]
        q = min(X[tuple(zip(*neg))])
        X[tuple(zip(*neg))] -= q
        X[tuple(zip(*pos))] += q

    return X, np.sum(X * C)

supply, demand, costs = umova()
# print(supply)
# print(demand)
# print(costs)
routes, total_cost = transport(supply, demand, costs)
```

```
print('\n\nРозв\'язок на мові Python\n')
print('Оптимальний план перевезень:')
print(routes)
print(f'Мінімальна вартість перевезень: {total_cost}')



C = costs.astype(int).flatten().tolist()
b = np.concatenate((supply, demand)).astype(int).tolist()

m = len(supply)
n = len(demand)

A_eq = octave.zeros(m + n, m * n)
b_eq = octave.zeros(m + n, 1)

for i in range(m):
    A_eq[i, i*n:(i+1)*n] = 1
    b_eq[i] = b[i]

for j in range(n):
    A_eq[m+j, j::n] = 1
    b_eq[m+j] = b[m+j]

lb = octave.zeros(m * n, 1)
ub = octave.inf(m * n, 1)

param = {'msglev': 1}  #  двофазний простий симплекс
ctype = 'S' * (m + n)
vartype = 'C' * (m * n)
sense = 1
results = octave.glpk(C, A_eq, b_eq, lb, ub, ctype, vartype, sense, param)
plan = np.reshape(results, (n, m)).T
oct_total_cost = octave.dot(C, results)

print('\n\nРозв\'язок на мові Octave\n')
print('Оптимальний план перевезень:')
print(plan)
print(f'Мінімальна вартість перевезень: {oct_total_cost}')
```