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:
студент групи КМ-03
Шаповалов Г. Г.

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

Київ — 2023

Зміст

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 24

Транспортна задача без обмежень на пропускну здатність

Транспортні витрати (матриця C)	Об'єм виробництва (вектор a)	Об'єм потреб (вектор b)
29 6 29 19 21 14 3 30 10 10 15 27 28 11 24 1 23 25 15 13	13 27 16 14	14 14 14 14 14

Транспортна задача з обмеженням на пропускну здатність

Транспортні витрати	Обмеження на перевезення	Об'єм виробництва	Об'єм потреб
18 14 6 8 7 8 19 9 16 17 18 10 14 7 19 1 6 10 5 13	12 20 30 20 10 20 4 3 2 8 13 10 30 10 15 23 5 6 4 6	72 29 68 26	65 24 50 30 26

Математична модель

Транспортна задача може бути зведена до задачі лінійного програмування (ЛП)

Позначимо c_{ij} як вартість перевезення одиниці товару від джерела i до призначення j, x_{ij} як кількість товарів, яку потрібно перевезти від джерела i до призначення j, a_j як обсяг товарів, які можуть бути відправлені з джерела i, b_j як обсяг товарів, які повинні бути доставлені до призначення j.

Математична модель даної задачі:

1. **Цільова функція (мінімізація вартості перевезення)**:
   Мінімізувати $Z = \sum_{i=1}^{4} \sum_{j=1}^{5} c_{ij} * x_{ij}$, де c_{ij} - вартість перевезення одиниці товару від i-го джерела до j-го призначення.
2. **Обмеження на обсяг товарів, які приходять до кожного призначення**:
   $\sum_{i=1}^{5} x_{ij} \leq a_i$ для j = 1, 2, 3, 4, 5

де a_i – обсяг товарів, які можуть бути доставлені до j-го призначення

3. **Обмеження на обсяг товарів, які виходять з кожного джерела**:
   $\sum_{i=1}^{4} x_{ij} = b_j$ для i = 1, 2, 3, 4
   де b_j – обсяг товарів, які можуть бути відправлені до і-го джерела
4. **Необхідно враховувати, що кількість товарів не може бути від'ємною**:
   $x_{ij} \geq 0$ для i = 1, 2, 3, 4 та j = 1, 2, 3, 4, 5
5. **Додаткове обмеження на кількість товарів, яку можна перевезти**:
   $x_{ij} \leq a_i * b_j$ для i = 1, 2, 3, 4 та j = 1, 2, 3, 4, 5
   Це обмеження враховує той факт, що кількість перевезених товарів не може перевищувати обсяг товарів, які доступні в джерелі *i*, а також обсяг товарів, які потрібно доставити до призначення *j*.

Задача з обмеженням на пропускну здатність

Цільова функція: Мінімізувати вартість перевезень, яка визначається як сума вартості кожного перевезення, помножена на кількість одиниць, що перевозяться.

$$\min(Z) = \sum_{i=1}^{m}\sum_{j=1}^{n} c_{ij}x_{ij}$$

де c_{ij} - вартість перевезення одиниці товару від постачальника *i* до споживача *j*, x_{ij}- кількість одиниць товару, що перевозяться від постачальника *i* до споживача *j*, m - кількість постачальників, n - кількість споживачів.

Обмеження:

1. Обсяги виробництва та споживання:

$$\sum_{j=1}^{n} x_{ij} = S_i \; \forall \in \{1, 2, \dots, m\}$$

$$\sum_{i=1}^{m} x_{ij} = d_i \; \forall \in \{1, 2, \dots, n\}$$

де S_i - обсяг виробництва постачальника i, d_i - обсяг споживання споживача j.

2. Обмеження на пропускну здатність:

$$0 \leq x_{ij} \leq b_{ij} \; \forall i \in \{1, 2, \dots, m\}, \forall j \; \in \{1, 2, \dots \, , n\}$$

де b_{ij} - максимальна пропускна здатність між постачальником *i* та споживачем *j*.

Ця модель використовує лінійне програмування для вирішення транспортної задачі з обмеженнями на пропускну здатність. Застосування цієї моделі дозволяє знайти оптимальний план перевезень, який мінімізує загальні витрати на перевезення.

Задача без обмеження на пропускну здатність

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.

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.

Алгоритми розв’язання

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

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

Код на Python вирішує транспортну задачу без обмежень за допомогою Python. Він використовує метод симплексу для знаходження оптимального плану перевезень, який мінімізує загальні витрати на перевезення.

Спочатку код визначає вхідні дані: матрицю витрат (costs), обсяги виробництва (supply) та обсяги споживання (demand). Потім він створює матрицю рівностей (A_eq) та вектор правих частин рівностей (b_eq) для обмежень, що відповідають обсягам виробництва та споживання.

Після цього код визначає межі для змінних (bounds), які вказують, що кількість одиниць товару, що перевозяться, повинна бути не менше нуля.

Нарешті, код викликає функцію linprog з бібліотеки scipy.optimize, щоб знайти оптимальний план перевезень, який мінімізує загальні витрати на перевезення. Результати виводяться у вигляді оптимального плану перевезень та мінімальної вартості перевезень.

Код на Octave використовує двофазний симплексний метод для знаходження оптимального плану перевезень, який мінімізує загальні витрати на перевезення.

Спочатку код визначає вхідні дані: матрицю витрат (costs), обсяги виробництва (supply) та обсяги споживання (demand). Потім він створює матрицю рівностей (A_eq) та вектор правих частин рівностей (b_eq) для обмежень, що відповідають обсягам виробництва та споживання.

Після цього код визначає нижню та верхню межі для змінних (lb та ub), які вказують, що кількість одиниць товару, що перевозяться, повинна бути не менше нуля, а верхня межа не визначена (inf).

Нарешті, код викликає функцію glpk з бібліотеки Octave, щоб знайти оптимальний план перевезень, який мінімізує загальні витрати на перевезення. Результати виводяться у вигляді оптимального плану перевезень та мінімальної вартості перевезень.

Висновки

У результаті розв’язку транспортної задачі досягнуто таких результатів:

Розв'язок транспорної задачі (без обмежень) на мові Python

Оптимальний план перевезень:

[[0. 0. 0. 2. 0.]
[1. 0. 14. 14. 0.]
[12. 13. 0. 0. 0.]
[0. 0. 0. 14. 0.]]

Мінімальна вартість перевезень: 757.0

Розв'язок транспортної задачі (без обмежень) на мові Octave

Оптимальний план перевезень:

[[0. 0. 0. 2. 0.]
[1. 0. 14. 14. 0.]
[12. 13. 0. 0. 0.]
[0. 0. 0. 14. 0.]]

Мінімальна вартість перевезень: 757.0

Розв'язок транспорної задачі (з обмеженнями) на мові Python

Оптимальний план перевезень:

[[0. 0. 46. 0. 26.]
[29. 0. 0. 0. 0.]
[10. 24. 4. 30. 0.]
[26. 0. 0. 0. 0.]]

Мінімальні витрати перевезень: 1402.0

Розв'язок транспорної задачі (з обмеженнями) на мові Octave

Оптимальний план перевезень:

[[0. 0. 46. 0. 26.]
[29. 0. 0. 0. 0.]
[10. 24. 4. 30. 0.]
[26. 0. 0. 0. 0.]]

Мінімальні витрати перевезень: 1402.0

Розв’язавши задачі різними мовами знайдено абсолютно однакові результати, це каже про те що було досягнуто дійсно оптимальних значень.

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://scipy.org
4. Метод симлекса- https://studfile.net/preview/5470183/page:8/

Додаток А – Код програми

Вміст файлу main.py :

```
from oct2py import octave
import warnings
import os
import numpy as np
from scipy.optimize import linprog
warnings.filterwarnings("ignore", category=DeprecationWarning)


def umova():
    with open('umova.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


supply, demand, costs = umova()
os.system('cls')
# print(supply)
# print(demand)
# print(costs)




"""
Транспортна задача без обмежень на Python
"""

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

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

A_eq = np.zeros((m + n, m * n))
```

```
b_eq = np.zeros(m + n)

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]

bounds = [(0, None)] * (m * n)  # Вказуємо верхню межу для всіх змінних

param = {'disp': False}  # Вивід інформації про процес оптимізації
results = linprog(C, A_eq=A_eq, b_eq=b_eq, bounds=bounds, method='simplex',
options=param)
plan = np.reshape(results.x, (n, m)).T
total_cost = np.dot(C, results.x)

print('\nРозв\'язок транспорної задачі на мові Python')
print('Оптимальний план перевезень:')
print(plan)
print(f'Мінімальна вартість перевезень: {total_cost}')




"""
Транспортна задача без обмежень на Octave
"""

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




"""
Транспортна задача обмеженням на Python
"""

# Матриця витрат
costs = np.array([
    [18, 14, 6, 8, 7],
    [8, 19, 9, 16, 17],
    [18, 10, 14, 7, 19],
    [1, 6, 10, 5, 13]
])

# Об'єми виробництва та об'єми потреб
supplies = [72, 29, 68, 26]
demands = [65, 24, 50, 30, 26]

B = np.array([
    [12, 20, 30, 20, 10],
    [20, 4, 3, 2, 8],
    [13, 10, 30, 10, 15],
    [23, 5, 6, 4, 6]
])

# Перетворення задачі у вигляд лінійного програмування
num_supplies = len(supplies)
num_demands = len(demands)
flatten_costs = costs.flatten()

# Коефіцієнти цільової функції (вартість перевезень)
c = flatten_costs

# Коефіцієнти лівих частин обмежень (обсяги виробництва та споживання)
A_eq = np.zeros((num_supplies + num_demands, num_supplies * num_demands))
for i in range(num_supplies):
    for j in range(num_demands):
        A_eq[i, i * num_demands + j] = 1
        A_eq[num_supplies + j, i * num_demands + j] = 1

# Праві частини обмежень (суми обсягів)
b_eq = np.concatenate([supplies, demands])
```

```python
# # Визначення меж для кількості продуктів (позитивні значення)
# bounds = [(0, None) for _ in range(num_supplies * num_demands)]


# Згенеруйте список кортежів меж для кожного x_ij на основі матриці обмежень B
bounds = [(0, None) for _ in range(num_supplies * num_demands)]

# Виклик функції linprog для знаходження оптимального розв'язку
result = linprog(c, A_eq=A_eq, b_eq=b_eq, bounds=bounds, method='highs')

# Виведення результату
print('\n\n\nРозв\'язок транспорної задачі (з обмеженнями) на мові Python')
print('Оптимальний план перевезень:')
print(result.x.reshape((num_supplies, num_demands)))
print('Мінімальні витрати перевезень:', result.fun)




"""
Транспортна задача обмеженням на Octave
"""

from oct2py import octave
import numpy as np

# Матриця витрат
costs = np.array([
    [18, 14, 6, 8, 7],
    [8, 19, 9, 16, 17],
    [18, 10, 14, 7, 19],
    [1, 6, 10, 5, 13]
])

# Об'єми виробництва та об'єми потреб
supplies = [72, 29, 68, 26]
demands = [65, 24, 50, 30, 26]

B = np.array([
    [12, 20, 30, 20, 10],
    [20, 4, 3, 2, 8],
    [13, 10, 30, 10, 15],
    [23, 5, 6, 4, 6]
])

# Перетворення задачі у вигляд лінійного програмування
num_supplies = len(supplies)
num_demands = len(demands)
flatten_costs = costs.flatten()
```

```
# Коефіцієнти цільової функції (вартість перевезень)
c = flatten_costs

# Коефіцієнти лівих частин обмежень (обсяги виробництва та споживання)
A_eq = np.zeros((num_supplies + num_demands, num_supplies * num_demands))
for i in range(num_supplies):
    for j in range(num_demands):
        A_eq[i, i * num_demands + j] = 1
        A_eq[num_supplies + j, i * num_demands + j] = 1

# Праві частини обмежень (суми обсягів)
b_eq = np.concatenate([supplies, demands])

# Згенеруйте список кортежів меж для кожного x_ij на основі матриці обмежень B
bounds = np.array([(0, np.inf) for _ in range(num_supplies * num_demands)])

# Виклик функції glpk для знаходження оптимального розв'язку
result = octave.feval('glpk', c, A_eq, b_eq, bounds[:, 0], bounds[:, 1])

# Перетворення результату назад у форму costs
result = result.reshape(costs.shape)

# Виведення результату
print('\nРозв\'язок транспорної задачі (з обмеженнями) на мові Octave')
print('Оптимальний план перевезень:')
print(result.reshape((num_supplies, num_demands)))
print(f'Мінімальні витрати перевезень: {np.sum(result * costs)}')
```