Учреждение образования
«БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ»
Кафедра информатики

Отчет по лабораторной работе №5
Матричная транспортная задача.

Выполнил:
студент гр. 953505
Басенко К. А.

Руководитель:
доцент
Алёхина А. Э.

Минск 2022

Постановка задачи

Пусть имеется *m* пунктов производства и *n* пунктов его потребления. Для каждого пункта производства $i = 1, 2, \ldots, m$ и для каждого пункта потребления $j = 1, 2, \ldots, n$ заданы следующие величины:

- Объем производства ai в пункте производства i
- Объем потребления bj в пункте потребления j
- Затраты на перевозку единицы продукта cij от пункта производства i до пункта потребления j

Предполагается, что суммарное производство равно суммарному потреблению $\sum_{i=1}^{m} a_i = \sum_{j=1}^{n} b_j$.

Цель задачи: составить план перевозок, позволяющий полностью вывезти продукты всех производителей, полностью обеспечивающий потребности всех потребителей и дающий минимум суммарных затрат на перевозку. Обозначим как xij объемы перевозок от поставщика i до потребителя j, тогда:

$$\sum_{j=1}^{n} x_{ij} = a_i,\ i = 1, 2, \ldots, m$$

$$\sum_{i=1}^{m} x_{ij} = b_j,\ j = 1, 2, \ldots, n$$

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

Краткие теоретические сведения

Имеется m предприятий, на которых производятся $a1, a2, \ldots, am$ единиц продукции, и n пунктов ее потребления с потребностями $b1, b2, \ldots, bn$. Известна стоимость (затраты) cij перевозки единицы продукции из i-го пункта производства в j-й пункт потребления.

Требуется определить такой план перевозок продукции по пунктам ее потребления, при котором весь продукт из пунктов производства будет вывезен, спрос всех потребителей удовлетворен, а транспортные расходы минимальны.

Составим математическую модель задачи. Рассмотрим план задачи в виде матрицы планирования (матрицы перевозок)$X = (x_{ij}),\ i = \overline{1,\ m},\ j = \overline{1,\ n}$, где x_{ij} - количество продукции, перевозимой из i-го пункта ее производства в j-й пункт ее потребления. Тогда $c_{ij}x_{ij}$ - стоимость перевозки продукта от i-го производителя к j-му потребителю.

$\sum_{i=1}^{m}\sum_{j=1}^{m} c_{ij}\, x_{ij} \rightarrow \min$, т.к. Требуется минимизировать целевую функцию z (общую стоимость всех перевозок) при условии, что вся продукция из пунктов вывезена: $\sum_{j=1}^{n} x_{ij} = a_i,\ i = \overline{1, m}$, и все запросы удовлетворены $\sum_{i=1}^{m} x_{ij} = b_j,\ i = \overline{j,\ n}$, при этом $x_{ij} \geq 0,\ i = \overline{1, m},\ j = \overline{1, n}$. Следовательно, математическая модель транспортной задачи также представляет собой ЗЛП.

Программная реализация

Транспортная задача реализована в функции *matrix_transport_problem*, на вход в которую идут: список производителей *a*, список потребителей *b*, матрица перевозок *c*:

```
def matrix_transport_problem(a, b, c):
```

Сначала мы балансируем задачу:

```
sum_a = sum(a)
sum_b = sum(b)

if sum_a < sum_b:
    a += [sum_b - sum_a]
    c += [[0] * len(b)]
elif sum_a > sum_b:
    b += [sum_a - sum_b]
    c = [ci + [0] for ci in c]
```

Далее идет метод потенциалов. В первой фазе мы начальный базисный план перевозок с соответствующим множеством базисных позиций с помощью метода “северо-западного угла”:

```python
x = np.zeros((m, n))
B = []

a_copy = np.copy(a)
b_copy = np.copy(b)

i, j = 0, 0
while i < m:
    while j < n:
        minimal = min(a_copy[i], b_copy[j])

        x[i][j] = minimal
        a_copy[i] -= minimal
        b_copy[j] -= minimal

        B.append(Position(i, j))

        if a_copy[i] == 0:
            i += 1
            break

        j += 1
```

Вторая фаза метода потенциалов. Для каждой позиции $(i, j) \in B$ записываем уравнение $u_i + v_j = c_{ij}$, получаем систему, решаем ее:

```
while True:
    syst = [[1] + [0] * (m + n - 1)]
    b = [0]

    u_offset = 0
    v_offset = m

    for pos in B:
        syst.append([0.] * (m + n))

        syst[-1][u_offset + pos.i] = 1
        syst[-1][v_offset + pos.j] = 1

        b.append(c[pos.i][pos.j])

    res = np.linalg.solve(syst, b)

    u = res[:v_offset]
    v = res[v_offset:]
```

Далее проверяем условие оптимальности текущего базисного плана: если $\forall(i, j) \in N\colon u_i + v_j \le c_{ij}$, то текущий базисный план перевозок является планом с минимальной стоимостью. В противном случае, позицию для которой не выполняется условие оптимальности добавим в множество B и запомним ее:

```
    for i in range(len(u)):
        for j in range(len(v)):
            if ((i, j) not in B) and (u[i] + v[j] > c[i][j]):
                B.append(Position(i, j))
                break
        else:
            continue

        break
    else:
        print("Базисный план перевозок с минимальной стоимостью:")
        print(x)
        return

    new_pos = B[-1]
```

Составляем граф G_B, вершинами которого являются базисные позиции плана перевозок:

```
for node in B:
    nodes.append((node.i, node.j))

    for i in range(node.i - 1, -1, -1):
        if (i, node.j) in B:
            edges.append(((i, node.j), (node.i, node.j)))
            break
    for i in range(node.i + 1, n):
        if (i, node.j) in B:
            edges.append(((i, node.j), (node.i, node.j)))
            break

    for j in range(node.j - 1, -1, -1):
        if (node.i, j) in B:
            edges.append(((node.i, j), (node.i, node.j)))
            break
    for j in range(node.j + 1, m):
        if (node.i, j) in B:
            edges.append(((node.i, j), (node.i, node.j)))
            break
```

Находим в нем цикл, который проходит через последнюю добавленную в базис позицию:

```
G = nx.MultiDiGraph()
G.add_nodes_from(nodes)
G.add_edges_from(edges)

cycle: list
for found_cycle in list(nx.simple_cycles(G)):
    if new_pos in found_cycle and len(found_cycle) > 2:
        cycle = [Position(i,j) for (i,j) in found_cycle]
        break
```

Отмечаем угловые позиции:

```
ninety_degrees = []
for i in range(len(cycle)):
    i_pre = i - 1
    i_next = (i + 1) % len(cycle)

    pos_pre = cycle[i_pre]
    pos_next = cycle[i_next]
    pos = cycle[i]

    if (pos_pre.i == pos.i and pos_next.j == pos.j or \
        pos_pre.j == pos.j and pos_next.i == pos.i):
        ninety_degrees.append(pos)
```

Отмечаем угловые вершины ‘+’ или ‘-’:

```
pluses = ninety_degrees[::2]
```

Изменяем кол-во продукции на угловых позициях, запоминаем позицию, на которой достигается минимум:

```
min_x = ninety_degrees[-1]

for pos in ninety_degrees:
    if (pos not in pluses and \
        (x[pos.i][pos.j] < x[min_x.i][min_x.j])):
            min_x.i, min_x.j = pos.i, pos.j

theta = x[min_x.i][min_x.j]

for pos in ninety_degrees:
    x[pos.i][pos.j] += theta * (1 if pos in pluses else -1)
```

Получаем новый базисный план перевозок:

```
B.remove((min_x.i, min_x.j))
```

Далее выполняем вторую фазу метода потенциалов, пока не найдем план с оптимальной стоимостью.

Тестовые примеры

Данные для тестирования:

```
test_cases = [
    [
        a := [50, 50, 100],
        b := [40, 90, 70],
        c := [
                [2, 5, 3],
                [4, 3, 2],
                [5, 1, 2],
        ],
        res := [
            [40, 0, 10],
            [0, 0, 50],
            [0, 90, 10],
        ],
    ],
```

```
    [
        a := [90, 90, 100],
        b := [50, 100, 130],
        c := [
                [2, 5, 4],
                [7, 6, 5],
                [9, 8, 10],
        ],
        res := [
            [50, 0, 40],
            [0, 0, 90],
            [0, 100, 0],
        ],
    ],
```

```
    [
        a := [140, 80, 80],
        b := [100, 100, 100],
        c := [
                [1, 1, 4],
                [5, 2, 1],
                [6, 1, 3],
        ],
        res := [
            [100, 40, 0],
            [0, 0, 80],
            [0, 60, 20],
        ],
    ],
]
```

Выполняется метод с предоставленными данными, выводится полученный результат, выводится ожидаемый результат:

```
for *data, res in test_cases:
        matrix_transport_problem(*data)
        print(f'\nПравильный ответ:\n{np.array(res, dtype=float)}\n\n')
```

Вывод, соответствующий предоставленным входным данным:

```
Базисный план перевозок с минимальной стоимостью:
[[40.  0. 10.]
 [ 0.  0. 50.]
 [ 0. 90. 10.]]

Правильный ответ:
[[40.  0. 10.]
 [ 0.  0. 50.]
 [ 0. 90. 10.]]
```

```
Базисный план перевозок с минимальной стоимостью:
[[ 50.   0.  40.]
 [  0.   0.  90.]
 [  0. 100.   0.]]

Правильный ответ:
[[ 50.   0.  40.]
 [  0.   0.  90.]
 [  0. 100.   0.]]
```

```
Базисный план перевозок с минимальной стоимостью:
[[100.  40.   0.]
 [  0.   0.  80.]
 [  0.  60.  20.]]

Правильный ответ:
[[100.  40.   0.]
 [  0.   0.  80.]
 [  0.  60.  20.]]
```