Министерство образования науки российской федерации федеральное
государственное автономное образовательное учреждение высшего образования
«Санкт-Петербургский национальный исследовательский университет
информационных технологий, механики и оптики»

Факультет информационных технологий и программирования

Кафедра информационных систем

Практическая работа № 3

Линейное программирование и методы решения транспортной задачи.

Выполнили студенты группы № M3303:
Черкасов Илья Андреевич
Яцко Полина Олеговна

САНКТ-ПЕТЕРБУРГ

2018

В данной лабораторной работе нам предстоит решить две задачи — транспортную и линейную задачу симплекс-методом.

Линейная задача (решение симплекс-методом)

Вар 2 $A = \begin{pmatrix} -1 & 3 & 0 & 2 & 1 \\ 2 & -1 & 1 & 2 & 3 \\ 1 & -1 & 2 & 1 & 0 \end{pmatrix}$ $b = \begin{pmatrix} 1 \\ 2 \\ 4 \end{pmatrix}$ $c = \begin{pmatrix} -1 & -3 & 2 & 1 & 4 \end{pmatrix}$

Задача в матричной форме вида: $\left.\begin{aligned} \overline{c}^{\mathrm{T}} \cdot \overline{x} &\to \min(\max) \\ A \cdot \overline{x} &\le \overline{b}, \\ \overline{x} &\ge \overline{0}, \end{aligned}\right\}$

называется *симметричной* формой записи задачи ЛП.

Задача линейного программирования вида $\left.\begin{aligned} \overline{c}^{\mathrm{T}} \cdot \overline{x} &\to \min(\max) \\ A \cdot \overline{x} &= \overline{b}, \\ \overline{x} &\ge \overline{0}, \end{aligned}\right\}$

называется *канонической формой* записи задачи линейного программирования.

Изначально задача дана в симметричной форме записи, в задаче требуется найти такой вектор x, больший нуля, чтобы его произведение на матрицу A не превышало b по всем координатам, при этом его произведение с транспонированным вектором c окажется минимальным.

Обозначим вектор x (план) как вектор чисел x1, x2, x3, x4, x5 (столбец, так как его будет нужно умножать на прямоугольную матрицу). Мы получаем систему из трех уравнений и пяти неизвестных:

$$
\begin{cases}
\begin{pmatrix} -1 \\ -3 \\ 2 \\ 1 \\ 4 \end{pmatrix} \cdot (x_1\ x_2\ x_3\ x_4\ x_5)^T \to \min \\
\begin{pmatrix} -1 & 3 & 0 & 2 & 1 \\ 2 & -1 & 1 & 2 & 3 \\ 1 & -1 & 2 & 1 & 0 \end{pmatrix} \cdot (x_1\ x_2\ x_3\ x_4\ x_5)^T \le \begin{pmatrix} 1 \\ 2 \\ 4 \end{pmatrix} \\
\bar{x} \ge 0
\end{cases}
$$

$$
\begin{cases}
-x_1 + 3x_2 + 2x_4 + x_5 \le 1 \\
2x_1 - x_2 + x_3 + 2x_4 + 3x_5 \le 2 \\
x_1 - x_2 + 2x_3 + x_4 \le 4
\end{cases}
$$

$$\downarrow$$

$+ \ x_6, x_7, x_8$

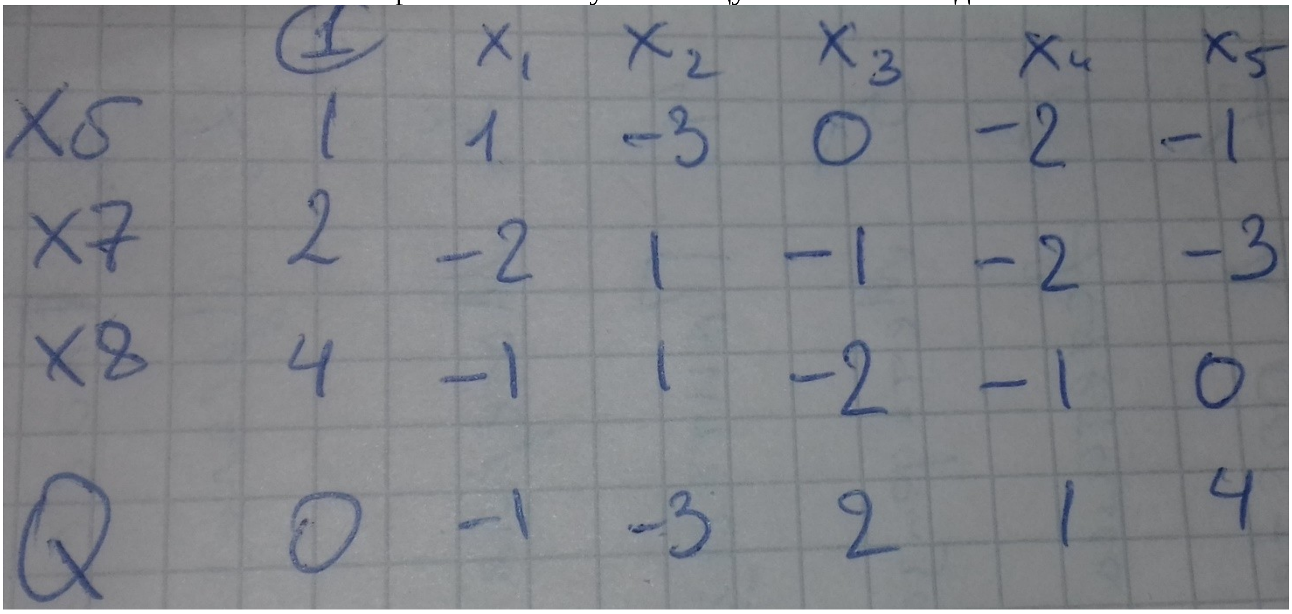
тогда допустимое решение

$(0, 0, 0, 0, 0, 1, 2, 4)$

$$
\begin{cases}
x_6 = 1 + x_1 - 3x_2 - 2x_4 - x_5 \\
x_7 = 2 - 2x_1 + x_2 - x_3 - 2x_4 - 3x_5 \\
x_8 = 4 - x_1 + x_2 - 2x_3 - x_4
\end{cases}
$$

Чтобы перейти к канонической форме, добавляем переменные x6, x7, x8.

По этой системе можно построить начальную таблицу симплекс метода:



	1	x1	x2	x3	x4	x5
x6	1	1	-3	0	-2	-1
x7	2	-2	1	-1	-2	-3
x8	4	-1	1	-2	-1	0
Q	0	-1	-3	2	1	4

Целевая функция Q здесь получила свои коэффициенты из коэффициентов перед иксами в произведении вектора c на вектор x. У нее (функции) два отрицательных коэффициента (перед x1 и x2). Если x1 и x2 будут расти (помним, они неотрицательны), функция будет уменьшаться до бесконечности, а вот если бы все коэффициенты в ряду Q были бы положительными, можно было бы с уверенностью сказать, что наивыгоднейшее для нас значение целевой функции достигается при всех свободных x, равных 0. Тогда это значение будет значением свободного члена – первым в ряду Q. Поэтому при изменении системы уравнений (и, следовательно, таблицы) надо стремиться к положительным значениям в последней строке (кроме первого значения свободного члена, оно может быть любым).

На каждом шаге алгоритма, если целевая функция пока не достигла минимума, выберем наименьший отрицательный коэффициент среди коэффициентов x строки Q. Он определит разрешающий столбец. Из всех отрицательных чисел разрешающего столбца (кроме коэффициента Q) находим частное от деления свободного члена на это число и выбираем минимальное по абсолютной величине. Назовем разрешающей строкой ту, в которой это число находилось (если отрицательных коэффициентов в столбце не было, целевая функция неограниченно убывает и решений в принципе нет). Соответствующий элемент на пересечении разрешающей строки и столбца – опорный элемент.

Теперь начинаем пересчитывать таблицу. Опорные строка и столбец сначала остаются без изменений, меняются все остальные числа в таблице. По правилу прямоугольника каждое число заменяется на разность произведения его и опорного и двух противоположных углов прямоугольника. Далее все числа разрешающей строки меняют знак, опорный элемент становится единицей. После вся таблица делится на исходное значение опорного элемента. В полученной таблице нужно подписать, что переменные поменялись местами.

Ниже приведен код программы, считающей по описанному алгоритму минимальное значение целевой функции:

```java
public class Main {
    private static double[][] A = { // important!!! columns amount is c.length, rows amount is b.length
            {-1, 3, 0, 2, 1},
            {2, -1, 1, 2, 3},
            {1, -1, 2, 1, 0}
    };

    // column!
    private static double[] b = {
            1,
            2,
            4
    };

    // column!
    private static double[] c = {
            -1,
            -3,
            2,
            1,
            4
    };

    private static int[] freeX = new int[c.length]; // numbers of the Xes, which are free
    private static int[] basisX = new int[b.length];
    private static double[][] arr = new double [b.length + 1][c.length + 1];

    private static int permittingColumn; // they are the indexes in arr!
    private static int permittingRow;

    /*
     * Table structure:
     *                         n   freeX[0]   freeX[1]   ........   freeX[c.length - 1]
     *        basisX[0]     = .. + ........ + ........ + ........ + ...................
     *        basisX[1]     = .. + ........ + ........ + ........ + ...................
     *        .........     = .. + ........ + ........ + ........ + ...................
     * basisX[b.length - 1] = .. + ........ + ........ + ........ + ...................
     *           Q          = .. + ........ + ........ + ........ + ...................
     *
     * x1 - x5 are free now, x6 - x8 are basis (is is just an example, it depends on b.length and c.length)
     * They are in the freeX and basisX arrays (the order is important!)
     *
     * There will be b.length of basis vars and c.length of free vars and also 1 not-x number.
     * The last line contains coefficients for the goal function.
     */
    private static void initTable() {
        // at the beginning we have additional vars at the basis, others are free
        // first (№0) column contains free numbers (in the beginning they are b)
        for (int i = 0; i < b.length; i++) {
            arr[i][0] = b[i];
        }

        for (int i = 0; i < b.length; i++) {
            for (int j = 1; j < c.length + 1; j++) {
                arr[i][j] = -A[i][j - 1];
            }
        }

        arr[b.length][0] = 0;
        for (int j = 0; j < c.length; j++) {
            arr[b.length][j + 1] = c[j];
        }
```

```java
        for (int j = 0; j < c.length; j++) {
            freeX[j] = j + 1;
        }

        for (int i = 0; i < b.length; i++) {
            basisX[i] = c.length + i + 1;
        }
    }

    private static boolean isTheGoalMinimum() {
        boolean isMinimum = true;
        for (int j = 1; j < c.length + 1; j++) {
            isMinimum = isMinimum && arr[b.length][j] >= 0;
        }
        return isMinimum;
    }

    private static void findTheBearingElement() {
        // find the permittingColumn
        double minAtQ = arr[b.length][1];
        permittingColumn = 1;
        for (int i = 2; i < c.length + 1; i++) {
            if (arr[b.length][i] < minAtQ) {
                minAtQ = arr[b.length][i];
                permittingColumn = i;
            }
        }

        // find the permitting row
        double minAtPermittingColumn = Double.MAX_VALUE;
        for (int i = 0; i < b.length; i++) {
            double tmp = arr[i][0] / arr[i][permittingColumn];
            if (arr[i][permittingColumn] < 0 && tmp < minAtPermittingColumn) {
                minAtPermittingColumn = tmp;
                permittingRow = i;
            }
        }

        // if there were no elements < 0 in the column
        if (minAtPermittingColumn == Double.MAX_VALUE) {
            System.out.println("Oh, sorry, it seems like there is no solution at all");
            System.exit(0);
        }
    }

    private static void reportTheSolution() {
        System.out.println("I have found the solution!");
        System.out.println("The goal function is " + arr[b.length][0]);
        System.out.println("At:");
        for(int i = 0; i < freeX.length; i++) {
            System.out.println("x" + freeX[i] + " = 0");
        }
        for(int i = 0; i < basisX.length; i++) {
            System.out.println("x" + basisX[i] + " = " + arr[i][0]);
        }
    }

    private static void switchTheXes() {
        double bearingValue = arr[permittingRow][permittingColumn];

        for (int i = 0; i < b.length + 1; i++) {
            for (int j = 0; j < c.length + 1; j++) {
                if (i != permittingRow && j != permittingColumn) {
                    arr[i][j] =
                        arr[i][j] * arr[permittingRow][permittingColumn] -
```

```java
                                arr[permittingRow][j] * arr[i][permittingColumn];
                }
            }
        }

        for (int i = 0; i < c.length + 1; i++) {
            arr[permittingRow][i] *= -1;
        }
        arr[permittingRow][permittingColumn] = 1;

        for (int i = 0; i < b.length + 1; i++) {
            for (int j = 0; j < c.length + 1; j++) {
                arr[i][j] = arr[i][j] / bearingValue;
            }
        }

        int tmp = freeX[permittingColumn - 1];
        freeX[permittingColumn - 1] = basisX[permittingRow];
        basisX[permittingRow] = tmp;
    }

    public static void main(String[] args) {
        initTable();
        while(!isTheGoalMinimum()) {
            findTheBearingElement();
            switchTheXes();
        }
        reportTheSolution();
    }
}
```

При запуске этого кода (в нем введены значения нашего варианта) мы получаем минимальное значение функции в -14, при этом x1 = 6.5, x2 = 2.5, x3 = 0, x4 = 0, x5 = 0. Также дополнительные переменные приняли значения в x6 = 0, x7 = -8.5, x8 = 0.

Транспортная задача

Вар 2

пункты	B1	B2	B3	B4	запасы
A1	1	7	9	5	120
A2	4	2	6	8	280
A3	3	8	1	2	160
потребности	130	220	60	70	480/560

Транспортная задача линейного программирования формулируется следующим образом. Необходимо минимизировать транспортные расходы

$$Q(X) = \sum_{i=1}^{m} \sum_{j=1}^{n} c_{ij} \cdot x_{ij} \to \min$$

при ограничениях

$$\left.\begin{aligned} &\sum_{i=1}^{m} x_{ij} = b_j, && j = \overline{1,n}, \\ &\sum_{j=1}^{n} x_{ij} = a_i, && i = \overline{1,m}, \\ &x_{ij} \geq 0, \quad i = \overline{1,m}, && j = \overline{1,n}, \end{aligned}\right\},$$

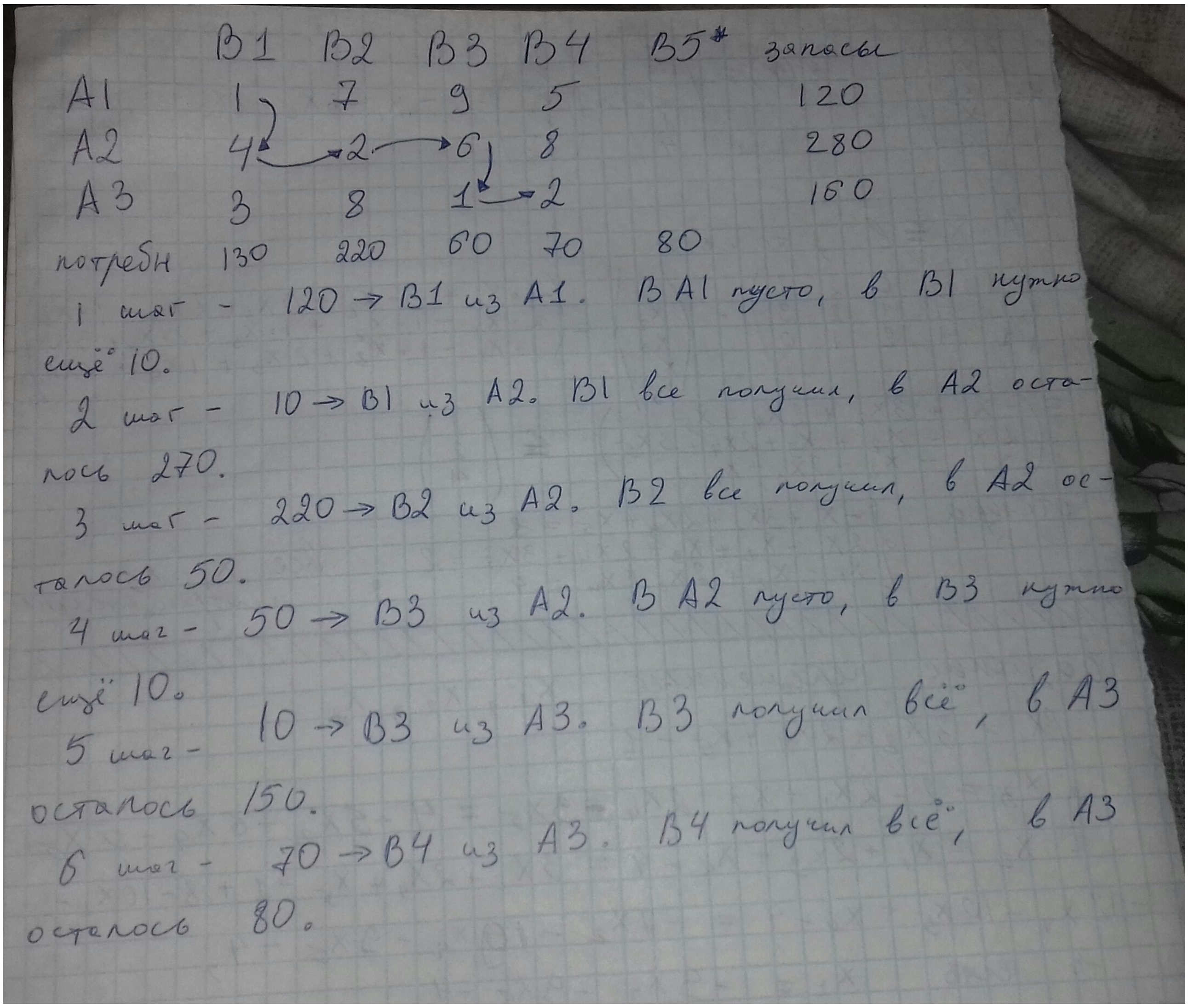
где c_{ij} - стоимость перевозки единицы продукции из пункта i в пункт j; x_{ij} – планируемая величина перевозок из пункта i в пункт j (план перевозок X – матрица размерности $m \times n$); b_j - потребности в продукте в пункте j; a_i - запасы в пункте i.

В нашем случае перевозки производятся из пунктов А в пункты В, a — запасы в пунктах А — находятся в последнем столбце, b — потребности пунктов В — находятся в последней строке, цифры на пересечениях А и B — стоимости перевозки единицы товара.

Видно, что модель так называемого открытого типа (сумма запасов не совпадает с суммой потребностей). Для приведения ее к закрытому типу введем фиктивный пункт потребления на 560-480 = 80 единиц.

Задачу можно решить несколькими методами:

<u>Метод северо-западного угла</u>



	B1	B2	B3	B4	B5*	запасы
A1	1	7	9	5		120
A2	4	2	6	8		280
A3	3	8	1	2		160
потребн	130	220	60	70	80	

1 шаг - 120 → B1 из A1. В A1 пусто, в B1 нужно ещё 10.

2 шаг - 10 → B1 из A2. B1 все получил, в A2 осталось 270.

3 шаг - 220 → B2 из A2. B2 все получил, в A2 осталось 50.

4 шаг - 50 → B3 из A2. В A2 пусто, в B3 нужно ещё 10.

5 шаг - 10 → B3 из A3. B3 получил всё, в A3 осталось 150.

6 шаг - 70 → B4 из A3. B4 получил всё, в A3 осталось 80.

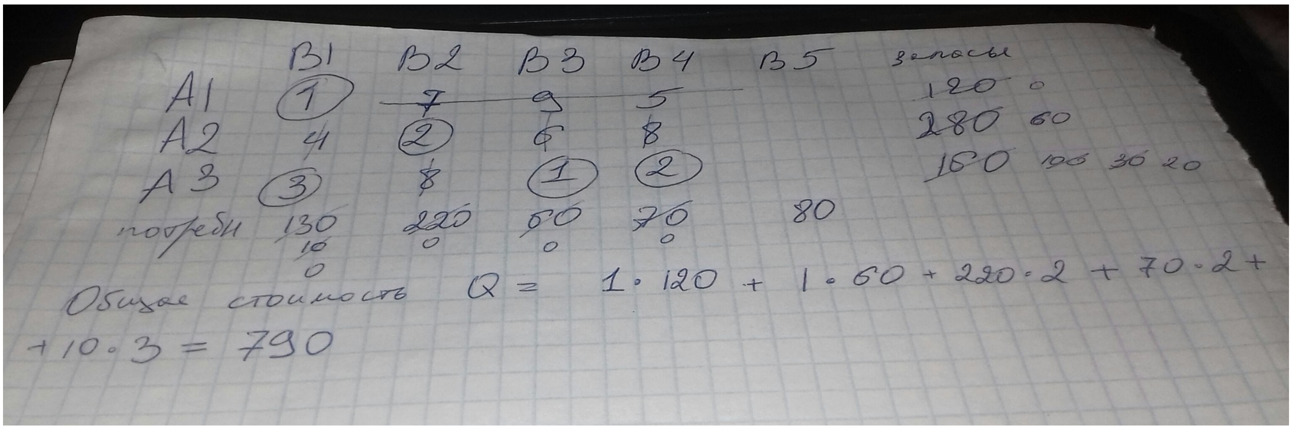
Общая стоимость Q составит 1*120 + 4*10 + 2*220 + 6*50 + 1*10 + 2*70 = 1050

Матрица X тогда примет вид

120	0	0	0
10	220	50	0
0	0	10	70

Как видно, данный метод не оптимизирует распределение ресурсов, поскольку вовсе не учитывает стоимость доставки ресурсов.

Метод минимального элемента дает результат в 790:



	B1	B2	B3	B4	B5	запасы
A1	1	7	9	5		~~120~~ 0
A2	4	2	6	8		~~280~~ 60
A3	3	8	1	2		~~160~~ 100 30 20
потребн	~~130~~ 10 0	~~220~~ 0	~~60~~ 0	~~70~~ 0	80	

Общая стоимость $Q = 1 \cdot 120 + 1 \cdot 60 + 220 \cdot 2 + 70 \cdot 2 + 10 \cdot 3 = 790$

Матрица X:

120	0	0	0
0	220	0	0
10	0	60	70

Метод потенциалов

	B1	B2	B3	B4	B5	
A1	1	7	9	5	0	120
A2	4	2	6	8	0	280
A3	3	8	1	2	0	160
	130	220	60	70	80	

В таблице значения c_{ij}, в фиктивном столбце B5 стоимости доставки нулевые.
Построим первоначальный план (метод северо-западного угла):

$$X = \begin{pmatrix} 120 & 0 & 0 & 0 & 0 \\ 10 & 220 & 50 & 0 & 0 \\ 0 & 0 & 10 & 70 & 80 \end{pmatrix}$$

План не имеет циклов, опорных клеток $3+8-1=7$.

Расставим потенциалы так, чтобы

$c_{ij} = U_i + V_j$ для опорных клеток

	B1	B2	B3	B4	B5	U
A1	0	8	6	1	-2	0
A2	0	0	0	1	-5	3
A3	4	11	0	0	0	-2
V	1	-1	3	4	2	

в таблице значения $c_{ij} - U_i - V_j$

Видно, что в двух клетках A1-B5 и A2-B5 есть отрицательные числа, значит, план не оптимален.

Перераспределим поставки.

Клетка с минимальным оценочным числом A2-B5 (−5). Строим цикл по ней и опорным

$$\begin{matrix} 0 & 8 & 6 & 1 & -2 \\ 0 & 0 & 0^{\ominus} & 1 & -5^{\oplus} \\ 4 & 11 & 0_{\oplus} & 0 & 0_{\ominus} \end{matrix}$$

Перераспределим поставки, минимальное значение перевозок в текущем плане, которое можно перераспределить −50 (клетка опорного плана A2-B3, меньше A3-B5 (80)).

Получаем новый план:

$$X = \begin{pmatrix} 120 & 0 & 0 & 0 & 0 \\ 10 & 220 & 0 & 0 & 50 \\ 0 & 0 & 60 & 70 & 30 \end{pmatrix}$$

Вновь вычислим потенциалы:

$c_{ij} - U_i - V_j$

	B1	B2	B3	B4	B5	U
A1	0	8	11	6	3	0
A2	⊖0	0	5	6	⊕0	3
A3	⊕−1	6	0	0	⊖0	3
V	1	−1	−2	−1	−3	

Видим, что в A3-B1 отрицательное значение, план не оптимален.

Строим цикл. Минимальное значение груза из клеток A2-B1 и A3-B5 − 10 из клетки A2-B1.

Перераспределим:

$$X = \begin{pmatrix} 120 & 0 & 0 & 0 & 0 \\ 0 & 220 & 0 & 0 & 60 \\ 10 & 0 & 60 & 70 & 20 \end{pmatrix}$$

	B					U
A	0	7	10	5	2	0
	1	0	5	6	0	2
	0	6	0	0	0	2
V	1	0	−1	0	−2	

$c_{ij} - U_i - V_j$

Все значения оценочных чисел $c_{ij} - U_i - V_j$ стали ≥ 0, значит, план оптимален.

Затраты оптимального плана:

$Q = 1 \cdot 120 + 3 \cdot 10 + 2 \cdot 220 + 1 \cdot 60 + 2 \cdot 70 + 0 \cdot 20 + 0 \cdot 60 =$
$= 790$ (результат как и в методе мин. элемента)

Таким образом, можно утверждать, метод северо-западного угла может быть применим только если нужно быстро получить хоть какое-нибудь решение, очень редко оно совпадет с оптимальным, а вот метод минимального элемента дает оптимальный вариант в нашем случае, но не всегда это может быть так. В то время, как у метода потенциалов есть четкое правило для остановки и разделение всей задачи на подзадачи, которая позволяет получить именно оптимальное решение.