Министерство образования Российской Федерации

МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ ИМ. Н.Э. БАУМАНА

Факультет: Информатика и системы управления
Кафедра: Информационная безопасность (ИУ8)

МЕТОДЫ ОПТИМИЗАЦИИ И ИССЛЕДОВАНИЕ ОПЕРАЦИЙ В ИНФОРМАЦИОННОЙ БЕЗОПАСНОСТИ

Лабораторная работа №1 на тему:

«Линейное программирование. Симплекс-метод»

Вариант 6

Преподаватель:
Коннова Н.С.

Студент:
Асатрян А.Г

Группа:
ИУ8-73

Москва 2019

Цель работы

Изучение симплекс-метода решения задачи линейного программирования.

Постановка задачи

Требуется найти решение следующей задачи:

$$F = \boldsymbol{cx} \rightarrow max,$$
$$\boldsymbol{Ax} \leq \boldsymbol{b},$$
$$\boldsymbol{x} \geq 0.$$

Здесь $\boldsymbol{x} = [x_1, x_2, \ldots, x_n]^T$ – искомый вектор решения;

$\boldsymbol{c} = [c_1, c_2, \ldots, c_n]$ – вектор коэффициентов целевой функции (ЦФ) F;

$\boldsymbol{A} = \begin{bmatrix} a_{11} & \cdots & a_{1n} \\ \cdots & \cdots & \cdots \\ a_{m1} & \cdots & a_{mn} \end{bmatrix}$ – матрица системы ограничений;

$\boldsymbol{b} = [b_1, b_2, \ldots, b_m]^T$ – вектор правой части системы ограничений.

Ход работы.

Имеем следующие данные:

c =[2, 6, 7];

A = $\begin{bmatrix} 3 & 1 & 1 \\ 1 & 2 & 0 \\ 0 & 0.5 & 2 \end{bmatrix}$;

b =[3 8 1];

Приведем задачу ЛП к канонической форме, введя фиктивные переменные x4, x5, x6:

$$F = 2x_1 + 6x_2 + 7x_3 \rightarrow min$$
$$\begin{cases} 3x_1 + x_2 + x_3 + x_4 = 3; \\ x_1 + 2x_2 + x_5 = 8; \\ 0.5x_2 + 2x_3 + x_6 = 1; \end{cases}$$
$$x_1, x_2, x_3, x_4, x_5, x_6 \geq 0.$$

x_4 , x_5, x_6 – базисные переменные;

x_1 , x_2, x_3 - свободные переменные.

Тогда имеем:

$$F = -(-2x_1 - 6x_2 - 7x_3) \to min$$

$$\begin{cases} x_4 = 3 - (3x_1 + x_2 + x_3) \\ x_5 = 8 - (x_1 + 2x_2) \\ x_6 = 1 - (0.5x_2 + 2x_3) \end{cases}$$

$$x_1, x_2, x_3, x_4, x_5, x_6 \geq 0.$$

Исходная симплекс-таблица записывается в виде:

	S_0	x_1	x_2	x_3
x_4	3	3	1	1
x_5	8	1	2	0
x_6	1	0	0.5	2
F	0	2	6	7

Так как все элементы столбца S_0, кроме коэффициента целевой функции, неотрицательны, имеем опорное решение:

$x_1 = x_2 = x_3 = 0, \quad x_4 = 3, \quad x_5 = 8, \quad x_6 = 1.$

Целевая функция $F = 0$.

Возьмем столбец с максимальным по модулю числом 7 и заменим переменные x_4 на x_1.

Получим преобразованную симплекс-таблицу:

	S_0	x_4	x_2	x_3
x_1	1	0.33	0.33	0.33

x_5	7	-0.33	1.67	-0.33
x_6	1	0	0.5	2
F	-2	-0.67	5.33	6.33

Анализируем последнюю строку симплекс-таблицы и ищем первый минимальный по модулю элемент: в столбце x_2 (разрешающий столбец)

Найдем минимальное положительное отношение элемента свободных членов S_0 к соответствующему элементу в разрешающем столбце:

$$\frac{1}{0.33} < \frac{1}{0.5}$$

Следовательно, x_2 –разрешающая строка.

Заменим базисную переменную x_6 на свободную x_2.

Получим преобразованную симплекс-таблицу:

	S_0	x_4	x_5	x_3
x_1	0.33	0.33	-0.67	-1
x_2	3.67	-0.33	-3.33	-7
x_6	2	0	2	4
F	-12.67	-0.67	-10.67	-15

Новое решение имеет вид:

$x_1 = 0.33$
$x_2 = 2$
$x_3 = 0$
$x_4 = 0$
$x_5 = 3.67$
$x_6 = 0$
Целевая функция $F = 12.67$

Подставим получившиеся значения в исходную задачу :

$F = 2 \cdot 0.33 + 6 \cdot 2 + 7 \cdot 0 = 12.67$

$$\begin{cases} 0.99 + 2 + 0 \leq 3; \\ 0.33 + 4 \leq 8; \\ 2 \cdot 0.5 + 0 \leq 1; \end{cases}$$

$$x_1, x_2, x_3 \geq 0$$

Верно!

Вывод:

В ходе работы был изучен симплекс-метод решения задачи ЛП. Была решена конкретно поставленная задача и получено оптимальное значение целевой функции. Ответ, полученный при ручных вычислениях, совпал с ответом в программе и прошел проверку, поэтому решение верное.

Приложение

URL: https://github.com/lightman1998/lab_01

```
#include <iostream>
#include <iomanip>
#include <vector>

using namespace std;

class SimplexMethod {
	// строка таблицы
	struct Row {
		vector<double> a; // коэффициенты при x
		double b; // правая часть
	};

	int n; // число переменных
	int m; // число ограничений
	vector<Row> table; // симплекс таблица
	vector<double> c; // коэффициенты оптимизируемой функции
	vector<int> variables; // все переменные
	vector<int> basis; // базисные переменные
	vector<double> deltas; // дельты

	void CalculateDeltas(); // вычисление дельт
	int GetArgMaxDelta(); // вычисление номера минимальной дельты

public:
	SimplexMethod(int n, int m);

	void Read(); // ввод значений
	void Print(); // вывод таблицы

	void Solve(); // решение ОЗЛП
};

SimplexMethod::SimplexMethod(int n, int m) {
	this->n = n; // запоминаем количество переменных
	this->m = m; // запоминаем количество условий

	table = vector<Row>(m, { vector<double>(n), 0 }); // создаём таблицу
```

```
        c = vector<double>(n); // создаём вектор коэффициентов
}

// ввод значений
void SimplexMethod::Read() {
        cout << "Enter function coefficients (c): ";
        c = vector<double>(n); // создаём вектор коэффициентов

        // считываем коэффициенты оптимизируемой функции
        for (int i = 0; i < n; i++)
                cin >> c[i];

        cout << "Enter restrictions coefficients:" << endl;

        // считываем коэффициенты ограничений
        for (int i = 0; i < m; i++) {
                cout << "Enter restriction " << (i + 1) << ": ";

                for (int j = 0; j < n; j++)
                        cin >> table[i].a[j];

                cin >> table[i].b;
        }

        variables.clear(); // очищаем переменные

        // добавляем переменные
        for (int i = 0; i < n; i++)
                variables.push_back(i);

        for (int i = 0; i < m; i++) {
                c.push_back(0); // добавляем нули в функцию
                variables.push_back(n + i); // добавляем доп переменные
                basis.push_back(n + i); // делаем их базисными

                // добавляем коэффициенты для переменных с коэффициентом 1,
если они стоят на главной диагонали, иначе с нулём
                for (int j = 0; j < m; j++)
                        table[i].a.push_back(i == j);
        }
}

// вывод таблицы
void SimplexMethod::Print() {
        int vars = variables.size();

        cout << endl;
        cout << "+-----+";

        for (int i = 0; i < vars; i++)
                cout << "-----------+";

        cout << endl;

        cout << "|  C  |";

        for (int i = 0; i < vars; i++)
                cout << " " << setw(9) << c[i] << " |";

        cout << endl;

        cout << "+-----+";

        for (int i = 0; i <= vars; i++)
```

```
                cout << "-----------+";

        cout << endl;

        cout << "|basis|";
        for (int i = 0; i < vars; i++)
                cout << "    x" << setw(2) << left << (i + 1) << "    |";

        cout << "     b     |" << endl;
        cout << "+-----+";

        for (int i = 0; i <= vars; i++)
                cout << "-----------+";

        cout << endl;

        for (int i = 0; i < m; i++) {
                cout << "| x" << setw(2) << left;

                if (i < basis.size())
                        cout << (basis[i] + 1);
                else
                        cout << "?";

                cout  << " |";

                for (int j = 0; j < table[i].a.size(); j++)
                        cout << " " << setw(9) << table[i].a[j] << " |";

                cout << " " << setw(9) << table[i].b << " |" << endl;
        }

        cout << "+-----+";

        for (int i = 0; i <= vars; i++)
                cout << "-----------+";

        cout << endl;

        if (!deltas.size())
                return;

        cout << "|  D  |";

        for (int i = 0; i < deltas.size(); i++)
                cout << " " << setw(9) << deltas[i] << " |";

        cout << endl;

        cout << "+-----+";

        for (int i = 0; i <= vars; i++)
                cout << "-----------+";

        cout << endl;
}

// вычисление дельт
void SimplexMethod::CalculateDeltas() {
        deltas.clear(); // очищаем массив дельт

        // проходимся по всем переменным
        for (int i = 0; i <= variables.size(); i++) {
                double delta = 0;
```

```
                // вычилсяем дельту
                for (int j = 0; j < basis.size(); j++)
                        delta += c[basis[j]] * (i < variables.size() ?
table[j].a[i] : table[j].b);

                // вычитаем коэффициент функции
                if (i < variables.size())
                        delta -= c[i];

                deltas.push_back(delta); // добавляем дельту в массив
        }
}

// вычисление номера минимальной дельты
int SimplexMethod::GetArgMaxDelta() {
        int imax = 0; // считаем, что первая дельта максимальна

        // проходимся по всем дельтам
        for (int i = 1; i < deltas.size(); i++)
                if (deltas[i] > deltas[imax]) // если дельта стала больше
максимальной
                        imax = i; // обновляем индекс максимума

        return imax; // возвращаем индекс максимума
}

// решение ОЗЛП
void SimplexMethod::Solve() {
        int iteration = 1; // начинаем с первой итерации

        while (true) {
                CalculateDeltas(); // рассчитываем дельты

                int jmax = GetArgMaxDelta(); // ищем индекс максимальной
                double maxDelta = deltas[jmax]; // получаем максимальную
дельту

                cout << "Max delta: " << maxDelta << endl; // выводим
максимальную дельту

                // если она не положительна
                if (maxDelta <= 0) {
                        cout << "Plan is OK" << endl; // выводим, что план
оптимален
                        Print(); // выводим таблицу
                        break; // и выходим
                }

                cout << "Iteration " << iteration++ << ":" << endl; // выводим
номер итерации
                cout << "Calculating deltas:" << endl;
                Print(); // выводим таблицу

                vector<double> Q(m); // создаём симплекс отношения
                int imin = -1;

                // идём по ограничениям
                for (int i = 0; i < m; i++) {
                        if (table[i].a[jmax] == 0) { // если коэффициент равен
0
                                Q[i] = 0; // то отношение равно нулю
                        }
                        else {
```

```
                                Q[i] = table[i].b / table[i].a[jmax]; //
вычисляем результат отношения

                                // если оно отрицательно, то идём дальше
                                if (Q[i] < 0)
                                        continue;

                                // иначе обновляем минимальное симплекс
отношение
                                if (imin == -1 || Q[i] < Q[imin])
                                        imin = i;
                        }
                }

                basis[imin] = jmax; // делаем переменную базисноц
                double pivot = table[imin].a[jmax]; // получаем опорный
элемент

                cout << "Min Q: " << Q[imin] << endl; // выводим минимальное
симплекс отношение
                cout << "x" << (jmax + 1) << " is new basis variable" << endl;
// выводим новую базисную переменную
                cout << "Divide row " << (imin + 1) << " by " << pivot <<
endl; // делим строку на элемент

                // делим строку на элемент
                for (int i = 0; i < table[imin].a.size(); i++)
                        table[imin].a[i] /= pivot;

                table[imin].b /= pivot;

                // вычитаем из всех остальных строк эту строку, умноженную на
элемент в столбце jmax
                for (int i = 0; i < m; i++) {
                        if (i == imin)
                                continue;

                        double value = table[i].a[jmax];

                        for (int j = 0; j < table[i].a.size(); j++)
                                table[i].a[j] -= table[imin].a[j] * value;

                        table[i].b -= table[imin].b * value;
                }
        }

        cout << "Fmin: " << deltas[n + m] << endl; // выводим минимальное
значение фукнции
}

int main() {
        int m;
        int n;

        cout << "Enter number of variables: ";
        cin >> n; // считываем число переменных
        cout << "Enter number of restrictions: ";
        cin >> m; // считываем число ограничений

        SimplexMethod method(n, m); // создаём метод

        method.Read(); // считыааем данные

        cout << "Initial simplex table: ";
```

```
        method.Print();  // выводим начальную таблицу

        method.Solve(); // решаем
}
```