МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

**«БЕЛГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ им. В. Г. ШУХОВА»**

**(БГТУ им. В.Г. Шухова)**

Кафедра программного обеспечения вычислительной техники и автоматизированных систем

Лабораторная работа №3

по дисциплине: Исследование операций

тема: Модификации симплекс-метода. Методы искусственного базиса и

больших штрафов.

Выполнил: ст. группы ПВ-223 Игнатьев Артур

Проверил:

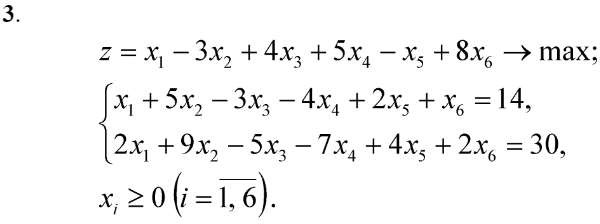
Вирченко Юрий Петрович

Белгород 2024 г.

**Цель работы:** изучение методов искусственного базиса и больших штрафов решения задач ЛП в канонической форме, не подготовленных к работе симплекс-методом в чистом виде.

**Задания для подготовки к работе**

1. Изучить метод и алгоритм искусственного базиса и составить программу решения задачи ЛП этим методом.
2. Изучить метод и алгоритм больших штрафов и составить программу решения задачи ЛП этим методом.
3. Запрограммировать изученные алгоритмы и отладить соответствующие программы. В рамках подготовки тестовых данных решить вручную следующую задачу.



**Ручной расчет**

Введем искусственные переменные x: в 1-ом равенстве вводим переменную x7; во 2-ом равенстве вводим переменную x8;

Для постановки задачи на максимальную целевую функцию запишем так: F(X) = x1-3x2+4x3+5x4-1x5+8x6 - Mx7 - Mx8 → max

За использование искусственных переменных, вводимых в целевую функцию, накладывается так называемый штраф величиной М, очень большое положительное число, которое обычно не задается.

Полученный базис называется искусственным, а метод решения называется методом искусственного базиса.

Из уравнений выражаем искусственные переменные: x7 =14-x1-5x2+3x3+4x4-2x5-x6

x8 = 30-2x1-9x2+5x3+7x4-4x5-2x6

которые поставим в целевую функцию:

F(X) = x1-3x2 + 4x3 + 5x4 - x5 + 8x6 - M(14-x1-5x2+3x3+4x4-2x5-x6) - M(30-2x1-

9x2+5x3+7x4-4x5-2x6) → max

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Базис | В | X1 | X2 | X3 | X4 | X5 | X6 | X7 | X8 |
| X7 | 14 | 1 | 5 | -3 | -4 | 2 | 1 | 1 | 0 |
| X8 | 30 | 2 | 9 | -5 | -7 | 4 | 2 | 0 | 1 |
| F(x0) | -44M | -1-3M | 3-14M | -4+8M | -  5+11M | 1-6M | -8-3М | 0 | 0 |

В качестве ведущего выберем столбец, соответствующий переменной x2, так как это наибольший коэффициент по модулю.

1. ая строка ведущая. Разрешающий элемент равен 5

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Базис | B | X1 | X2 | X3 | X4 | X5 | X6 | X7 | X8 | min |
| X7 | 14 | 1 | 5 | -3 | -4 | 2 | 1 | 1 | 0 | 14/5 |
| X8 | 30 | 2 | 9 | -5 | -7 | 4 | 2 | 0 | 1 | 10/3 |
| F(x1) | -  44M | -1-  3M | 3-  14M | -  4+8M | -  5+11M | 1-  6M | -8-  3M | 0 | 0 |  |

Представим расчет каждого элемента в виде таблицы

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| B | X1 | X2 | X3 | X4 | X5 | X6 | X7 | X8 |
| 14/5 | 1/5 | 5/5 | -3/5 | -4/5 | 2/5 | 1/5 | 1/5 | 0/5 |
| 30- | 2- | 9- | -5-(- | -7-(- | 4- | 2- | 0- | 1- |
| (14\*9)/5 | (1\*9)/5 | (5\*9)/5 | 3\*9):5 | 4\*9):5 | (2\*9):5 | (1\*9):5 | (1\*9):5 | (0\*9):5 |
| 0- | (-1-3M)- | (3- | (- | (- | (1-6M)- | (-8-3M)- | (0)- | (0)- |
| (14\*(3- | (1\*(3- | 14M)- | 4+8M)- | 5+11M)- | (2\*(3- | (1\*(3- | (1\*(3- | (0\*(3- |
| 14M))/5 | 14M))/5 | (5\*(3- | (-3\*(3- | (-4\*(3- | 14M)):5 | 14M)):5 | 14M)):5 | 14M)):5 |
|  |  | 14M)):5 | 14M)):5 | 14M)):5 |  |  |  |  |

Получаем новую симплекс таблицу

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Базис | B | x1 | x2 | x3 | x4 | x5 | x6 | x7 | x8 |
| x2 | 14/5 | 1/5 | 1 | -3/5 | -4/5 | 2/5 | 1/5 | 1/5 | 0 |
| x8 | 24/5 | 1/5 | 0 | 2/5 | 1/5 | 2/5 | 1/5 | -9/5 | 1 |
| F(X1) | -42/5-  24/5M | -8/5-  M | 0 | -11/5-  2/5M | -13/5-  M | -1/5-  2/5M | -43/5-  M | -  3/5+14/5M | 0 |

Текущий опорный план неоптимален, так как в индексной строке находятся отрицательные коэффициенты.

В качестве ведущего выберем столбец, соответсвующий переменной x3, так как наибольший коэффициент по модулю

1. ая строка является ведущей. Разрешающий элемент равен (2/5) и находится на пересечении ведущего столбца и ведущей строки.

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Базис | B | x1 | x2 | x3 | x4 | x5 | x6 | x7 | x8 | min |
| x2 | 14/5 | 1/5 | 1 | -3/5 | -4/5 | 2/5 | 1/5 | 1/5 | 0 | - |
| x8 | 24/5 | 1/5 | 0 | 2/5 | 1/5 | 2/5 | 1/5 | -9/5 | 1 | 12 |
| F(X2) | -42/5- 24/5M | -8/5-  M | 0 | -11/5- 2/5M | -13/5-  M | -1/5- 2/5M | -43/5-  M | -  3/5+14/5M | 0 |  |

Представим расчет каждого элемента таблицы:

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| B | x1 | x2 | x3 | x4 | x5 | x6 | x7 | x8 |
| 14/5-  (24/5\*- | 1/5-  (1/5\*- | 1-(0\*-  3/5):2/5 | -3/5-  (2/5\*- | -4/5-  (1/5\*- | 2/5-  (2/5\*- | 1/5-  (1/5\*- | 1/5-(-  9/5\*- | 0-(1\*-  3/5):2/5 |
| 3/5):2/5 | 3/5):2/5 |  | 3/5):2/5 | 3/5):2/5 | 3/5):2/5 | 3/5):2/5 | 3/5):2/5 |  |
| 24/5 : 2/  5 | 1/5 : 2/5 | 0 : 2/5 | 2/5 : 2/5 | 1/5 : 2/5 | 2/5 : 2/5 | 1/5 : 2/5 | -9/5 : 2/5 | 1 : 2/5 |
| (0)- | (-8/5- | (0)- | (-11/5- | (-13/5- | (-1/5- | (-43/5- | (- | (0)- |
| (24/5\*(- | M)- | (0\*(- | 2/5M)- | M)- | 2/5M)- | M)- | 3/5+14/5 | (1\*(- |
| 11/5- | (1/5\*(- | 11/5- | (2/5\*(- | (1/5\*(- | (2/5\*(- | (1/5\*(- | M)-(- | 11/5- |
|  | 11/5- |  | 11/5- | 11/5- | 11/5- | 11/5- | 9/5\*(-11/5- |  |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 2/5M)):2  /5 | 2/5M)):2  /5 | 2/5M)):2  /5 | 2/5M)):2  /5 | 2/5M)):2  /5 | 2/5M)):2  /5 | 2/5M)):2  /5 | 2/5M)):2/  5 | 2/5M)):2  /5 |

Получаем новую симплекс-таблицу:

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Базис | B | x1 | x2 | x3 | x4 | x5 | x6 | x7 | x8 |
| x2 | 10 | 1/2 | 1 | 0 | -1/2 | 1 | 1/2 | -5/2 | 3/2 |
| x3 | 12 | 1/2 | 0 | 1 | 1/2 | 1 | 1/2 | -9/2 | 5/2 |
| F(X2) | 18 | -1/2 | 0 | 0 | -3/2 | 2 | -15/2 | -21/2+M | 11/2+M |

Текущий опорный план неоптимален, так как в индексной строке находятся отрицательные коэффициенты.

В качестве ведущего выберем столбец, соответствующий переменной x6, так как это наибольший коэффициент по модулю.

Следовательно, 1-ая строка является ведущей.

Разрешающий элемент равен (1/2) и находится на пересечении ведущего столбца и ведущей строки.

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Базис | B | x1 | x2 | x3 | x4 | x5 | x6 | x7 | x8 | min |
| x2 | 10 | 1/2 | 1 | 0 | -1/2 | 1 | 1/2 | -5/2 | 3/2 | 20 |
| x3 | 12 | 1/2 | 0 | 1 | 1/2 | 1 | 1/2 | -9/2 | 5/2 | 24 |
| F(X3) | 18 | -1/2 | 0 | 0 | -3/2 | 2 | -15/2 | -21/2+M | 11/2+M |  |

Представим расчет каждого элемента в таблице

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| B | x1 | x2 | x3 | x4 | x5 | x6 | x7 | x8 |
| 10 : 1/2 | 1/2 : 1/2 | 1 : 1/2 | 0 : 1/2 | -  1/2 : 1/2 | 1 : 1/2 | 1/2 : 1/2 | -  5/2 : 1/2 | 3/2 : 1/2 |
| 12- | 1/2- | 0- | 1- | 1/2-(- | 1- | 1/2- | -9/2-(- | 5/2- |
| (10\*1/2 | (1/2\*1/2 | (1\*1/2 | (0\*1/2 | 1/2\*1/2 | (1\*1/2 | (1/2\*1/2 | 5/2\*1/2 | (3/2\*1/2 |
| ):1/2 | ):1/2 | ):1/2 | ):1/2 | ):1/2 | ):1/2 | ):1/2 | ):1/2 | ):1/2 |
| (11/2+M | (-1/2)- | (0)- | (0)- | (-3/2)-(- | (2)- | (-15/2)- | (- | (11/2+M |
| )-(10\*(- | (1/2\*(- | (1\*(- | (0\*(- | 1/2\*(- | (1\*(- | (1/2\*(- | 21/2+M) | )-(3/2\*(- |
| 15/2)):1/2 | 15/2)):1/2 | 15/2)):1/ | 15/2)):1/ | 15/2)):1/ | 15/2)):1/ | 15/2)):1/2 | -(-5/2\*(- | 15/2)):1/2 |
|  |  | 2 | 2 | 2 | 2 |  | 15/2)):1/ |  |
|  |  |  |  |  |  |  | 2 |  |

Получаем новую симплекс-таблицу:

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Базис | B | x1 | x2 | x3 | x4 | x5 | x6 | x7 | x8 |
| x6 | 20 | 1 | 2 | 0 | -1 | 2 | 1 | -5 | 3 |
| x3 | 2 | 0 | -1 | 1 | 1 | 0 | 0 | -2 | 1 |
| F(X3) | 168 | 7 | 15 | 0 | -9 | 17 | 0 | -48+M | 28+M |

Текущий опорный план неоптимален, так как в индексной строке находятся отрицательные коэффициенты.

В качестве ведущего выберем столбец, соответствующий переменной x4, так как это наибольший коэффициент по модулю.

2-ая строка является ведущей.

Разрешающий элемент равен (1) и находится на пересечении ведущего столбца и ведущей строки.

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Базис | B | x1 | x2 | x3 | x4 | x5 | x6 | x7 | x8 | min |
| x6 | 20 | 1 | 2 | 0 | -1 | 2 | 1 | -5 | 3 | - |
| x3 | 2 | 0 | -1 | 1 | 1 | 0 | 0 | -2 | 1 | 2 |
| F(X4) | 168 | 7 | 15 | 0 | -9 | 17 | 0 | -48+M | 28+M |  |

Представим расчет каждого элемента в виде таблицы:

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| B | x1 | x2 | x3 | x4 | x5 | x6 | x7 | x8 |
| 20-(2\*-  1):1 | 1- (0\*-  1):1 | 2-(-  1\*-  1):1 | 0- (1\*-  1):1 | -1-  (1\*-  1):1 | 2- (0\*-  1):1 | 1- (0\*-  1):1 | -5-(-2\*- 1):1 | 3-(1\*-  1):1 |
| 2 : 1 | 0 : 1 | -1 : 1 | 1 : 1 | 1 : 1 | 0 : 1 | 0 : 1 | -2 : 1 | 1 : 1 |
| (28+M)- | (7)- | (15)- | (0)- | (-9)- | (17)- | (0)- | (- | (28+M)- |
| (2\*(- | (0\*(- | (- | (1\*(- | (1\*(- | (0\*(- | (0\*(- | 48+M)- | (1\*(- |
| 9)):1 | 9)):1 | 1\*(- | 9)):1 | 9)):1 | 9)):1 | 9)):1 | (-2\*(- | 9)):1 |
|  |  | 9)):1 |  |  |  |  | 9)):1 |  |

Получаем новую симплекс-таблицу

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Базис | B | x1 | x2 | x3 | x4 | x5 | x6 | x7 | x8 |
| x6 | 22 | 1 | 1 | 1 | 0 | 2 | 1 | -7 | 4 |
| x4 | 2 | 0 | -1 | 1 | 1 | 0 | 0 | -2 | 1 |
| F(X4) | 186 | 7 | 6 | 9 | 0 | 17 | 0 | -66+M | 37+M |

Среди значений индексной строки нет отрицательных. Поэтому эта таблица определяет оптимальный план задачи.

Так как в оптимальном решении отсутствуют искусственные переменные (они равны нулю), то данное решение является допустимым.

Оптимальный план можно записать так: x1 = 0, x2 = 0, x3 = 0, x4 = 2, x5 = 0, x6 = 22

F(X) = 1\*0 -3\*0 + 4\*0 + 5\*2 -1\*0 + 8\*22 = 186

Блок-схемы:

Функция simplexAlgorithmCalculataion

Изображение выглядит как текст, диаграмма, линия, снимок экрана

Автоматически созданное описание

Функция CheckOptimality:

Изображение выглядит как текст, диаграмма, зарисовка, рисунок

Автоматически созданное описание

Функция doPivotting:

Изображение выглядит как текст, чек, черно-белый, документ

Автоматически созданное описание

Функция findPivotColumn:

Изображение выглядит как текст, диаграмма, снимок экрана, зарисовка

Автоматически созданное описание

Функция findPivotRow

Изображение выглядит как текст, диаграмма, зарисовка, рисунок

Автоматически созданное описание

Функция CalculateSimplex

Изображение выглядит как зарисовка, текст, рисунок, диаграмма

Автоматически созданное описание

Код программы:

#include <iostream> // Подключение заголовочного файла для ввода-вывода в стандартные потоки

#include <cmath> // Подключение заголовочного файла для математических функций

#include <vector> // Подключение заголовочного файла для использования векторов

using namespace std; // Использование стандартного пространства имён

class Simplex // Объявление класса Simplex

{

private: // Приватная секция класса

int rows, cols; // Переменные для хранения количества строк и столбцов vector<vector<float>> A; // Матрица коэффициентов переменных vector<float> B; // Массив констант ограничений

vector<float> C; // Массив коэффициентов целевой функции

float maximum; // Переменная для хранения максимального значения целевой функции

bool isUnbounded; // Флаг, указывающий, является ли задача неограниченной

public: // Публичная секция класса

Simplex(vector<vector<float>> matrix, vector<float> b, vector<float> c)

// Конструктор класса

{

maximum = 0; // Инициализация максимального значения isUnbounded = false; // Инициализация флага неограниченности rows = matrix.size(); // Установка количества строк

cols = matrix[0].size(); // Установка количества столбцов A.resize(rows, vector<float>(cols, 0)); // Инициализация размера

матрицы A

B.resize(b.size()); // Инициализация размера массива B C.resize(c.size()); // Инициализация размера массива C

for (int i = 0; i < rows; i++) // Цикл по строкам матрицы

{

for (int j = 0; j < cols; j++) // Цикл по столбцам матрицы

{

A[i][j] = matrix[i][j]; // Присваивание значений матрицы

}

}

for (int i = 0; i < c.size(); i++) // Цикл по массиву коэффициентов целевой функции

{

C[i] = c[i]; // Присваивание значений массива C

}

for (int i = 0; i < b.size(); i++) // Цикл по массиву констант ограничений

{

B[i] = b[i]; // Присваивание значений массива B

}

}

bool simplexAlgorithmCalculataion() // Функция для выполнения итерации метода симплекс

{

// Проверка на оптимальность таблицы if (checkOptimality() == true)

{

return true; // В случае оптимальности завершаем алгоритм

}

int pivotColumn = findPivotColumn(); // Находим столбец для опорного элемента

if (isUnbounded == true) // Проверяем на условие неограниченности

{

ошибке

}

cout << "Error unbounded" << endl; // Выводим сообщение об return true; // Завершаем алгоритм

int pivotRow = findPivotRow(pivotColumn); // Находим строку для опорного элемента

doPivotting(pivotRow, pivotColumn); // Производим обновление

таблицы

return false; // Возвращаем флаг продолжения выполнения алгоритма

}

bool checkOptimality() // Функция для проверки оптимальности таблицы

{

bool isOptimal = false; // Инициализируем флаг оптимальности int positveValueCount = 0; // Счётчик положительных значений

// Проверяем коэффициенты целевой функции на неотрицательность for (int i = 0; i < C.size(); i++)

{

float value = C[i]; if (value >= 0)

{

positveValueCount++;

}

}

// Если все коэффициенты неотрицательны, то таблица оптимальна if (positveValueCount == C.size())

{

isOptimal = true; // Устанавливаем флаг оптимальности print(); // Выводим текущее состояние таблицы

}

return isOptimal; // Возвращаем флаг оптимальности

}

void doPivotting(int pivotRow, int pivotColumn) // Функция для обновления таблицы после выбора опорного элемента

{

float pivetValue = A[pivotRow][pivotColumn]; // Получаем значение опорного элемента

float pivotRowVals[cols]; // Массив значений строки с опорным элементом

float pivotColVals[rows]; // Массив значений столбца с опорным элементом

float rowNew[cols]; // Массив значений обновлённой строки

maximum = maximum - (C[pivotColumn] \* (B[pivotRow] / pivetValue));

// Обновляем максимальное значение

// Получаем значения строки с опорным элементом for (int i = 0; i < cols; i++)

{

pivotRowVals[i] = A[pivotRow][i];

}

// Получаем значения столбца с опорным элементом for (int j = 0; j < rows; j++)

{

pivotColVals[j] = A[j][pivotColumn];

}

// Обновляем значения обновлённой строки for (int k = 0; k < cols; k++)

{

rowNew[k] = pivotRowVals[k] / pivetValue;

}

B[pivotRow] = B[pivotRow] / pivetValue; // Обновляем значение вектора B

// Обновляем остальные значения в матрице A for (int m = 0; m < rows; m++)

{

if (m != pivotRow) // Пропускаем строку с опорным элементом

{

for (int p = 0; p < cols; p++)

{

float multiplyValue = pivotColVals[m];

A[m][p] = A[m][p] - (multiplyValue \* rowNew[p]);

}

}

}

// Обновляем значения вектора B for (int i = 0; i < B.size(); i++)

{

if (i != pivotRow)

{

float multiplyValue = pivotColVals[i];

B[i] = B[i] - (multiplyValue \* B[pivotRow]);

}

}

float multiplyValue = C[pivotColumn]; // Получаем коэффициент для обновления вектора C

// Обновляем значения вектора C for (int i = 0; i < C.size(); i++)

{

C[i] = C[i] - (multiplyValue \* rowNew[i]);

}

// Заменяем строку с опорным элементом на обновлённую строку for (int i = 0; i < cols; i++)

{

A[pivotRow][i] = rowNew[i];

}

}

void print() // Функция для вывода текущего состояния таблицы

{

for (int i = 0; i < rows; i++) // Цикл по строкам

{

for (int j = 0; j < cols; j++) // Цикл по столбцам

{

cout << A[i][j] << " "; // Вывод элемента матрицы

}

cout << "" << endl; // Переход на новую строку

}

cout << "" << endl; // Переход на новую строку

}

int findPivotColumn() // Функция для поиска столбца с наименьшим коэффициентом целевой функции

{

int location = 0; // Переменная для хранения индекса столбца float minm = C[0]; // Переменная для хранения минимального

коэффициента

функции

for (int i = 1; i < C.size(); i++) // Цикл по коэффициентам целевой

{

if (C[i] < minm) // Поиск минимального коэффициента

{

minm = C[i]; location = i;

}

}

return location; // Возвращаем индекс столбца

}

int findPivotRow(int pivotColumn) // Функция для поиска строки с опорным элементом

{

float positiveValues[rows]; // Массив для хранения положительных значений

vector<float> result(rows, 0); // Вектор для хранения результатов int negativeValueCount = 0; // Счётчик отрицательных значений

for (int i = 0; i < rows; i++) // Цикл по строкам

{

if (A[i][pivotColumn] > 0) // Проверка положительности элемента

{

positiveValues[i] = A[i][pivotColumn];

}

else

{

positiveValues[i] = 0; // Игнорируем отрицательные значения negativeValueCount += 1; // Увеличиваем счётчик

отрицательных значений

}

}

// Проверка условия неограниченности if (negativeValueCount == rows)

{

isUnbounded = true;

}

else

{

for (int i = 0; i < rows; i++) // Цикл по строкам

{

float value = positiveValues[i];

if (value > 0) // Проверка на положительность

{

}

else

{

}

}

}

result[i] = B[i] / value; // Вычисление результатов

result[i] = 0;

// Поиск минимального значения в векторе результатов float minimum = 99999999;

int location = 0;

for (int i = 0; i < sizeof(result) / sizeof(result[0]); i++)

{

if (result[i] > 0)

{

if (result[i] < minimum)

{

minimum = result[i]; location = i;

}

}

}

return location; // Возвращаем индекс строки с опорным элементом

}

void CalculateSimplex() // Функция для выполнения метода симплекс

{

bool end = false; // Флаг завершения алгоритма

cout << "initial array(Not optimal)" << endl; // Вывод сообщения о начальном состоянии таблицы

print(); // Вывод начального состояния таблицы

cout << " " << endl; // Переход на новую строку

cout << "final array(Optimal solution)" << endl; // Вывод сообщения о конечном состоянии таблицы

while (!end) // Цикл выполнения итераций алгоритма

{

bool result = simplexAlgorithmCalculataion(); // Выполняем итерацию алгоритма

if (result == true) // Проверяем флаг окончания алгоритма

{

end = true; // Завершаем цикл

}

}

cout << "Answers for the Constraints of variables" << endl; // Выводим результаты

for (int i = 0; i < A.size(); i++) // Цикл по строкам матрицы

{

int count0 = 0; // Счётчик нулевых элементов int index = 0; // Индекс для переменной

for (int j = 0; j < rows; j++) // Цикл по строкам

{

if (A[j][i] == 0.0) // Проверка на ноль

{

count0 += 1;

}

else if (A[j][i] == 1) // Проверка на единицу

{

index = j; // Получаем индекс переменной

}

}

элементов

if (count0 == rows - 1) // Проверка на количество нулевых

{

cout << "variable" << index + 1 << ": " << B[index] << endl; // Выводим значение переменной

}

else

{

cout << "variable" << index + 1 << ": " << 0 << endl; //

Выводим ноль

}

}

cout << "" << endl; // Переход на новую строку

cout << "maximum value: " << maximum << endl; // Выводим максимальное значение целевой функции

}

};

int main() // Основная функция программы

{

int colSizeA = 9; // Задаём размер столбцов матрицы A int rowSizeA = 3; // Задаём размер строк матрицы A

float C[] = {0, -7, 0, -8, 0, -1, 0, 0, 0}; // Задаём массив коэффициентов целевой функции

float B[] = {10, 26, 12}; // Задаём массив констант ограничений

float a[3][9] = { // Инициализируем матрицу A

{1, -3, 0, -4, 0, -2, 1, 0, 0},

{0, 5, 0, 5, 1, 1, 0, 1, 0},

{0, 4, 1, -6, 0, -3, 0, 0, 1}};

vector<vector<float>> vec2D(rowSizeA, vector<float>(colSizeA, 0)); // Инициализируем двумерный вектор

vector<float> b(rowSizeA, 0); // Инициализируем вектор b vector<float> c(colSizeA, 0); // Инициализируем вектор c

for (int i = 0; i < rowSizeA; i++) // Цикл по строкам

{

for (int j = 0; j < colSizeA; j++) // Цикл по столбцам

{

vec2D[i][j] = a[i][j]; // Присваивание значений двумерному

вектору

}

}

for (int i = 0; i < rowSizeA; i++) // Цикл по строкам

{

b[i] = B[i]; // Присваивание значений вектору b

}

for (int i = 0; i < colSizeA; i++) // Цикл по столбцам

{

c[i] = C[i]; // Присваивание значений вектору c

}

Simplex simplex(vec2D, b, c); // Создание объекта класса Simplex simplex.CalculateSimplex(); // Выполнение метода симплекс

return 0; // Возврат нулевого значения

}import numpy as np # Импорт библиотеки numpy для работы с массивами

class Simplex: # Объявление класса Simplex

def init (self, matrix, b, c): # Метод инициализации класса с параметрами matrix, b, c

self.maximum = 0 # Инициализация переменной maximum для хранения максимального значения

self.isUnbounded = False # Инициализация переменной isUnbounded для проверки на ограниченность

self.rows, self.cols = len(matrix), len(matrix[0]) # Получение количества строк и столбцов матрицы

self.A = np.array(matrix) # Преобразование матрицы A в массив

numpy

self.B = np.array(b) # Преобразование вектора B в массив numpy self.C = np.array(c) # Преобразование вектора C в массив numpy

def simplexAlgorithmCalculation(self): # Метод выполнения шага симплекс-алгоритма

if self.checkOptimality(): # Проверка на оптимальность текущего

решения

return True

pivotColumn = self.findPivotColumn() # Нахождение опорного столбца

if self.isUnbounded: # Проверка на неограниченность print("Error unbounded") # Вывод сообщения об ошибке

неограниченности

return True

pivotRow = self.findPivotRow(pivotColumn) # Нахождение опорной

строки

self.doPivoting(pivotRow, pivotColumn) # Выполнение операции пересчета

return False

def checkOptimality(self): # Проверка на оптимальность текущего решения

if all(value >= 0 for value in self.C): # Если все значения вектора C неотрицательны

print("Optimal solution:") # Вывод сообщения об оптимальном

решении

print(self.A) # Вывод матрицы A return True

return False

def doPivoting(self, pivotRow, pivotColumn): # Метод выполнения операции пересчета

pivotValue = self.A[pivotRow][pivotColumn] # Получение значения опорного элемента

pivotRowVals = self.A[pivotRow] # Получение значений опорной

строки

pivotColVals = self.A[:, pivotColumn] # Получение значений

опорного столбца

rowNew = pivotRowVals / pivotValue # Новая опорная строка после пересчета

self.maximum -= self.C[pivotColumn] \* (self.B[pivotRow] / pivotValue) # Обновление максимального значения

self.A[pivotRow] = rowNew # Обновление опорной строки self.B[pivotRow] /= pivotValue # Обновление значения вектора B

for i in range(self.rows): # Цикл по строкам матрицы A if i != pivotRow:

self.A[i] -= pivotColVals[i] \* rowNew

for i in range(self.rows): if i != pivotRow:

self.B[i] -= pivotColVals[i] \* self.B[pivotRow]

self.C -= self.C[pivotColumn] \* rowNew

def findPivotColumn(self): # Нахождение опорного столбца

return np.argmin(self.C) # Возвращает индекс минимального элемента в векторе C

def findPivotRow(self, pivotColumn): # Нахождение опорной строки positiveValues = np.where(self.A[:, pivotColumn] > 0, self.A[:,

pivotColumn], 0) # Выделение положительных значений столбца

if np.all(positiveValues == 0): # Если все значения нулевые self.isUnbounded = True # Установка флага неограниченности return 0

result = np.where(positiveValues > 0, self.B / positiveValues, 0) # Вычисление результатов для нахождения опорной строки

pivotRow = np.argmin(result[result > 0]) + 1 # Нахождение индекса минимального положительного значения

return pivotRow

def calculateSimplex(self): # Метод выполнения симплекс-метода end = False

print("Исходная матрица: ") # Вывод исходной матрицы print(self.A)

print("\nПолученная матрица с помощью метода искусственого метода:

")

while not end: # Цикл выполнения шагов симплекс-алгоритма result = self.simplexAlgorithmCalculation()

if result:

end = True

print("\n>Базисные переменные:") for i, row in enumerate(self.A.T):

if np.count\_nonzero(row) == 1: index = np.argmax(row)

print(f"x {index + 1}: {self.B[index]}") else:

print(f"x {i + 1}: 0")

print("\nЗначение целевой функции::", self.maximum) # Определение матрицы A, векторов b и c

matrix\_A = [[1, 5, -3, -4, 2, 1, 1, 0],

[2, 9, -5, -7, 4, 2, 0, 1]]

vector\_b = [14, 30]

vector\_c = [-1, 3, -4, -5, 1, -8, 0, 0]

# Создание экземпляра класса Simplex и выполнение симплекс-метода simplex = Simplex(matrix\_A, vector\_b, vector\_c) simplex.calculateSimplex()

Результат работы программы:

Изображение выглядит как текст, снимок экрана, Шрифт, дизайн

Автоматически созданное описание

**Вывод:** в ходе выполнения лабораторной работы был изучен симплекс- метод для решения задачи линейного прораммирования с использованием симплекс-таблицы. Были получены навыки кодирования изученного

алгоритма, отладки и тестирования соответсвующих программ. Ручной метод и программа выдает одинаковые ответы, из чего можно сделать вывод, что алгоритм решения правильный.