БГУИР

Кафедра информатики

ОТЧЕТ

по лабораторной работе №1

**Решение задачи линейного программирования симплекс-методом**

***Вариант*** *14*

Выполнил: Проверил:

студент гр.453502 Ероминек Катажина

Чудук А. В. Ришардовна

Минск 2017

**Цель выполнения задания:** Изучить решение задачи линейного программирования с помощью симплекс-метода.

**Краткие теоретические сведения.**

**Утверждение 1**.(Признак оптимальности базисного плана.) *Базисный план х является решением задачи (1.6), если*

*  0, j  JH .*

**Утверждение 2.** (Достаточное условие неограниченности сверху целевой функции*.) Если существует индекс j0  JH , для которого  и все компоненты , i = , вектора*

*z = (, i = ) :=* (1.10)

*не положительны, то целевая функция задачи (1.6) не ограничена сверху на множестве планов.*

**Утверждение 3*.***(Возможность строгого улучшения плана.) *Пусть х* — *невырожденный базисный план задачи (1.6) и для некоторого j0  JH справедливы соотношения*

*, *

*где вектор z = (zi, i = ) определен по формуле (1.10). Тогда существует базисный план , для которого .*

Компоненты нового базисного плана *= (j, j  J)* и соответствующий ему базис  строятся по правилу

 ; (1.11)

,

где  *s* – любой индекс из множества



**Алгоритм симплекс-метода**

Пусть *х* – некоторый базисный план задачи (1.6) с базисом *JБ*. В результате исследования, основанного на утверждениях 1-3, выясняется либо 1) оптимальность плана *х*, либо 2) неразрешимость задачи (1.6) в силу неограниченности сверху целевой функции на множестве планов, либо 3) возможность перехода к новому базисному плану *,* для которого *c*’  *c*’.

Последовательный переход от одного базисного плана к “лучшему” базисному плану вплоть до получения оптимального плана составляет основную идею симплекс-метода.

Для реализации симплекс-метода кроме исходных данных задачи (1.6) (векторов *c, b* и матрицы *А*) на каждой итерации необходимо знать следующие параметры:

текущий базисный план  (достаточно знать только его базисные компоненты  *j  JБ* );

соответствующее плану *х*  множество базисных индексов *JБ = {j1, j2,…,jm};*

*mm* – матрицу *B = AБ-1*, обратную к базисной матрице *АБ = (Aj,  JБ) .*

Опишем общую итерацию симплекс-метода по шагам.

*Шаг 1.*  Вычислим вектор потенциалов *u’ = c’БВ*, где *cБ = (cj, j  JБ),* и оценки  , .

*Шаг 2.* Если , то STOP: вектор  является оптимальным планом задачи (1.6). В противном случае переходим к шагу 3.

*Шаг 3*. Выберем индекс , для которого . Построим вектор . Если , то STOP: задача (1.6) не имеет решения в силу неограниченности сверху целевой функции на множестве планов. В противном случае перейдем к шагу 4.

*Шаг 4*. Найдем минимум



и выберем индекс , для которого .

*Шаг 5*. Построим новый базисный план  и соответствующий ему базис  по правилам

;

.

*Шаг 6.* Вычислим матрицу , обратную к новой базисной матрице , по формуле , где матрица  отличается от единичной -матрицы только *s*-м столбцом, который имеет вид  ,  – единичный *m-*вектор с единицей на *i*-м месте.

Переходим к следующей итерации, исходя из новых плана  , базиса  и матрицы , обратной к новой базисной матрице .

**Замечания. 1.** На шаге 3 выбирается индекс , для которого . В общем случае существует несколько индексов, удовлетворяющих этому условию. При численной реализации симплекс-метода для выбора  можно использовать дополнительные «уточняющие» правила, например, следующие:

a)  либо

б) .

**2.** На шаге 4 выбирается индекс  или , для которого . В общем случае этот выбор может оказаться неоднозначным. Можно использовать дополнительные уточняющие правила, например, следующие:

а) либо

б) .

**3.** В современных версиях симплекс-метода для нахождения вектора потенциалов *u* (см. шаг 1) и вектора *z* (см. шаг 3) решаются системы и  соответственно. Для эффективного решения последних используется LU-разложение базисной матрицы .

**4**. Легко подсчитать, что приращение целевой функции при переходе от начального базисного плана *x* к новому базисному плану  равно: . По построению, . Следовательно, при  происходит «строгое улучшение» плана: . Из описания шага 4 видно, что в случае невырожденности начального базисного плана  всегда верно неравенство .

В случае, когда базисный план *x* (с базисом) является вырожденным, может реализоваться ситуация:  При этом мы получаем . В этом случае не происходит улучшения целевой функции, но итерация может оказаться полезной, так как изменяется базис и новый базис может быть ближе к оптимальному базису, чем старый.

**Распечатка кода программы**

#!/usr/bin/python

# -\*- coding: utf-8 -\*-

import numpy as np

def setDate(c, b, a, x\_first, j\_base):

c[0] = -6

c[1] = -9

c[2] = -5

c[3] = 2

c[4] = -6

c[5] = 0

c[6] = 1

c[7] = 3

b[0] = 6

b[1] = 1.5

b[2] = 10

a[0, 0] = 0

a[0, 1] = -1

a[0, 2] = 1

a[0, 3] = 7.5

a[0, 4] = 0

a[0, 5] = 0

a[0, 6] = 0

a[0, 7] = 2

a[1, 0] = 0

a[1, 1] = 2

a[1, 2] = 1

a[1, 3] = 0

a[1, 4] = -1

a[1, 5] = 3

a[1, 6] = -1.5

a[1, 7] = 0

a[2, 0] = 1

a[2, 1] = -1

a[2, 2] = 1

a[2, 3] = -1

a[2, 4] = 0

a[2, 5] = 3

a[2, 6] = 1

a[2, 7] = 1

x\_first[0] = 4

x\_first[1] = 0

x\_first[2] = 6

x\_first[3] = 0

x\_first[4] = 4.5

x\_first[5] = 0

x\_first[6] = 0

x\_first[7] = 0

j\_base[0] = 1 - 1

j\_base[1] = 3 - 1

j\_base[2] = 5 - 1

def createMatrix(a, j\_base, c, m, n):

a\_base = np.zeros((m, m), dtype=float)

c\_base = np.zeros(m, dtype=float)

for item\_i in range(m):

count = 0

for item\_j in range(n):

if item\_j == j\_base[count]:

a\_base[item\_i, count] = a[item\_i, item\_j]

count += 1

if count == m:

break

if np.linalg.det(a\_base) != 0:

a\_base\_obrat = np.linalg.inv(a\_base)

else:

print("Матрица A является вырожденной, det(A)=0")

return

count = 0

for item\_i in range(n):

if item\_i == j\_base[count]:

c\_base[count] = c[item\_i]

count += 1

if count == m:

break

return a\_base, a\_base\_obrat, c\_base

def solve():

while (True):

a\_base, a\_base\_obrat, c\_base = createMatrix(a, j\_base, c, m, n)

u = np.dot(c\_base, a\_base\_obrat)

delta = np.dot(u, a) - c

flag = False

count = 0

for item\_i in range(n):

if (delta[item\_i] >= -epsilon) == False:

if item\_i != j\_base[count]:

flag = True

else:

count += 1

if count == m:

count = m -1

if flag == False:

return "Оптимальный план задали x0 = " + str(x\_first) + "\ncx0 = " + str(np.dot(c.transpose(), x\_first))

j0 = -1

count = 0

for item\_i in range(n):

if (delta[item\_i] >= -epsilon) == False:

if item\_i != j\_base[count]:

j0 = item\_i

break

else:

count += 1

if count == m:

count = m -1

a\_j0 = np.zeros((m, 1), dtype=float)

for item\_i in range(m):

a\_j0[item\_i, 0] = a[item\_i, j0]

z = np.dot(a\_base\_obrat, a\_j0)

flag = False

for item\_i in range(m):

if (z[item\_i] <= epsilon) == False:

flag = True

if flag == False:

return "данная задача не имеет решения, т.к. ее целевая функция не ограничена на множестве планов."

q0 = 10000000000000

s = 10000000000000

for item\_i in range(m):

if z[item\_i] > 0:

last\_q0 = x\_first[j\_base[item\_i]] / z[item\_i]

if q0 > last\_q0:

q0 = last\_q0

s = item\_i

count = 0

for item\_i in range(n):

if item\_i != j\_base[count]:

x\_first[item\_i] = 0

else:

count += 1

if count == m:

count = m -1

x\_first[j0] = q0

count = 0

for item\_i in range(n):

if item\_i == j\_base[count]:

x\_first[item\_i] = x\_first[item\_i] - q0 \* z[count]

count += 1

if count == m:

break

j\_base[s] = j0

j\_base.sort()

return "Error"

c = np.zeros(8, dtype=float)

b = np.zeros(3, dtype=float)

a = np.zeros((3, 8), dtype=float)

m = 3

n = 8

epsilon = 0.001

x\_first = np.zeros(8, dtype=float)

j\_base = np.zeros(3, dtype=int)

setDate(c, b, a, x\_first, j\_base)

print(solve())

**Результат работы программы:**

Оптимальный план задали x0 = [ 0. 0. 0. 0. 0. 1.6 2.2 3. ]

cx0 = 11.2

**Вывод:** В данной лабораторной работе был изучен и програмно реализован симплекс-метод для решении задачи линейного программирования. Недостатком данного алгоритма является наличие двух неопределенностей, которые можно решить по определенным правилам, описанным в теории, выбор этих правил влияет на результат метода, в некоторых случаях, при неправильном выборе, происходит зацикливание симплекс-метода, которое происходит когда новая целевая функция, полученная при построение нового базиса, равна одной из целевых функций высчитанных в предшествующих циклах.