Изображение выглядит как эмблема, символ, герб, нашивка

Автоматически созданное описание

|  |
| --- |
| МИНОБРНАУКИ РОССИИ |
| Федеральное государственное бюджетное образовательное учреждение  высшего образования  **"МИРЭА - Российский технологический университет"**  **РТУ МИРЭА** |

**Институт** Информационных Технологий

**Кафедра** Вычислительной Техники

**ПРАКТИЧЕСКАЯ РАБОТА №6**

**по дисциплине**

**«Теория принятия решений»**

**Двойственная задача**

Студент группы:ИКБО-04-22 \_\_Кликушин В.И.\_\_\_\_\_\_\_\_\_\_ *(Ф. И.О. студента)*

Преподаватель \_\_Железняк Л.М.\_\_

*(Ф.И.О. преподавателя)*

|  |  |  |
| --- | --- | --- |
|  |  |  |

Москва 2024

**СОДЕРЖАНИЕ**

[ВВЕДЕНИЕ 3](#_Toc133218949)

[1ДВОЙСТВЕННАЯ ЗАДАЧА 4](#_Toc133218950)

[1.2 Постановка задачи 4](#_Toc133218951)

[1.2 Математическая модель исходной задачи 4](#_Toc133218952)

[1.3 Соответствующая исходной двойственная задача 4](#_Toc133218953)

[1.4 Первая теорема двойственности 5](#_Toc133218954)

[1.5 Вторая теорема двойственности 7](#_Toc133218955)

[1.6 Третья теорема двойственности 9](#_Toc133218956)

[ЗАКЛЮЧЕНИЕ 13](#_Toc133218957)

[СПИСОК ИНФОРМАЦИОННЫХ ИСТОЧНИКОВ 14](#_Toc133218958)

[ПРИЛОЖЕНИЯ 15](#_Toc133218959)

ВВЕДЕНИЕ

Обычно с задачей линейного программирования (ЗЛП) связана другая линейная задача, называемая двойственной. Тогда первоначальная задача называется исходной или прямой. Математические модели двойственных задач могут быть симметричными или несимметричными. В симметричных задачах система ограничений как исходной, так и двойственной задачи задается неравенствами, причем на двойственные переменные налагается условие не отрицательности. В несимметричных двойственных задачах система ограничений исходной задачи задается в виде равенств, а в двойственной – в виде неравенств, причем в последней переменные могут быть и отрицательными.

**1 ДВОЙСТВЕННАЯ ЗАДАЧА**

**1.1 Постановка задачи**

**Вариант №13**

Задание. Решить прямую ЗЛП с помощью симплексного метода и обратную с помощью теорем двойственности. Определить интервалы устойчивости.

Задача. В кондитерском цехе выпускают печенье двух сортов. В таблице 1 указан расход продуктов для каждого сорта и количество имеющихся продуктов.

*Таблица 1.1. Исходные данные задачи.*

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Сорт | Масло | Яйца | Сахар | Молоко | Цена за 1 кг, ден. ед. |
| 1-й сорт | 0,2 | 0,75 | 0,15 | 0,15 | 1,4 |
| 2-й сорт | 0,1 | 0,20 | 0,20 | 0,25 | 0,9 |
| Запасы продуктов | 100 | 150 | 100 | 150 |

Определить, какое общее количество печенья каждого сорта надо выпекать, чтобы общая стоимость была наибольшей.

**1.2 Математическая модель исходной задачи**

Пусть х1 – количество печенья первого сорта, х2 – количество печенья второго сорта. Прибыль от продажи печенья составит 1.4x1 + 0.9x2, прибыль требуется максимизировать.

Ограничения задачи:

Таким образом, переходим к задаче линейного программирования:

**1.3 Соответствующая исходной двойственная задача**

Найдем соответствующую двойственную задачу. Введем вектор двойственных переменных размерности два . Соответствующие векторы и матрица ограничений имеет вид:

.

Запишем двойственную задачу. Найти минимум функции.

При ограничениях:

**1.4 Первая теорема двойственности**

Если одна из пары двойственных задач имеет оптимальный план, то и другая имеет оптимальный план, причем экстремальные значения целевых функций равны. В ходе решения прямой задачи было определено, что максимальный доход от продажи составляет тыс. ден. ед., оптимальный план

Оптимальное решение двойственной задачи может быть получено из оптимального решения прямой задачи. Так как прямая задача имеет решение, то на основании первой теоремы о двойственности задача также разрешима. Ее решение может быть найдено из формулы:

Где D – матрица, составленная из компонентов векторов входящих в последний базис, при котором получен оптимальный план исходной задачи.

В последней симплекс-таблице базисными переменными являются 𝑥3, 𝑥1, 𝑥2, 𝑥6. Соответствующие этим переменным векторы , в разложении используются для формирования столбцов матрицы D.

Тогда,

Для вычисления обратной матрицы 𝐷-1 запишем матрицу 𝐷 дописав к ней справа единичную матрицу.

Для нахождения обратной матрицы 𝐷-1 используем элементарные преобразования над строками матрицы. Таким образом, преобразуются левая часть полученной матрицы в единичную.

Разделим вторую строку на 0.75;

от первой строки отнимем вторую строку, умноженную на 0.2; от третьей строки отнимем вторую строку, умноженную на 0.15, от четвертой строки отнимем вторую строку, умноженную на 0.15;

разделим третью строку на 0.16;

от первой строки отнимем третью строку, умноженную на ; от второй строки отнимем третью, умноженную на ; от четвертой строки отнимем третью строку, умноженную на 0.21;

Запишем обратную матрицу.

Базисными переменными в симплекс-таблице являются , тогда

При этом минимальное значение целевой функции двойственной задачи

совпадает с максимальным значением 𝑓𝑚𝑎𝑥 = [тыс. ден.ед.] прямой задачи, что является результатом взаимодвойственности. Таким образом,

**1.5 Вторая теорема двойственности**

Для того, чтобы планы и ЗЛП двойственной пары были оптимальными, необходимо и достаточно, чтобы эти планы удовлетворяли условиям дополняющей нежесткости.

Итак, имеем оптимальное решение прямой задачи: объем производства печенья первого сорта – 𝑥1 = ; объем производства печенья второго сорта – 𝑥2 = 437.5; максимальный доход от продажи 𝑓𝑚𝑎𝑥 = [тыс. ден.ед.]. Рассмотрим выполнение неравенств прямой задачи при подстановке 𝑥1, 𝑥2 в систему ограничений (Таблица 1.2).

Согласно Таблице 1.2 имеем следующую систему уравнений:

Решим данную систему уравнений

Решение, найденное из первой теоремы двойственности равнозначно решению из второй теоремы.

Таким образом, вторая теорема дает нахождение оптимального решения двойственной задачи, пользуясь условием обращения в равенство сопряженных неравенств в системах ограничения.

*Таблица 1.2 – Выполнение неравенств прямой задачи*

|  |  |  |
| --- | --- | --- |
| Ограничение | Расчет | Вывод |
| 0.2х1 + 0.1х2 ≤ 100 | 0.2\* + 0.1\*437.5 <100  60 < 100 | Первое ограничение прямой задачи выполняется как строгое неравенство, остается спрос на печенье первого сорта. Значит, этот ресурс не является дефицитным и его оценка в оптимальном плане равна нулю (𝑦1 = 0). |
| 0.75х1 + 0.2х2 ≤ 150 | 0.75\* + 0.2\*437.5 = 150  150=150 | Второе ограничение прямой задачи выполняется как равенство. Это означает, что печенье первого сорта полностью используется в оптимальном плане, является дефицитным и его оценка согласно второй теоремы двойственности отлична от нуля (𝑦2 ≠ 0). |
| 0.15х1 + 0.2х2 ≤ 100 | 0.15\* + 0.2\*437.5 = 100  100=100 | Третье ограничение прямой задачи выполняется как равенство. Это означает что печенье второго сорта полностью используется в оптимальном плане, является дефицитным и его оценка согласно второй теоремы двойственности отлична от нуля (𝑦3 ≠ 0). |
| 0.15х1 + 0.25х2 ≤ 150 | 0.15\* + 0.25\*437.5 < 150  121 < 150 | Четвёртое ограничение прямой задачи выполняется как строгое неравенство, остается спрос на печенье второго сорта. Значит, этот ресурс не является дефицитным и его оценка в оптимальном плане равна нулю (𝑦4 = 0). |
| х1 ≥ 0 | > 0 | Первое ограничение в двойственной задаче будет равенством 0.2y1 + 0.75y2 + 0.15y3 + 0.15y4 = 1.4 |
| х2 ≥ 0 | 437.5 > 0 | Второе ограничение в двойственной задаче будет равенством 0.1y1 + 0.2y2 + 0.2y3 + 0.25y4 = 0.9 |

**1.6 Третья теорема двойственности**

Третью теорему двойственности иногда называют теоремой об оценках. Рассматривая ограничения ЗЛП, можно констатировать: изменение правых частей ограничений исходной задачи приводит к изменению максимального значения целевой функции 𝑍𝑚𝑎𝑥.

Выпишем необходимые элементы из прямой задачи о максимальном доходе. Обратная матрица базиса оптимального плана:

Индексы базисных переменных оптимального плана:

Свободные члены неравенств (ограничений) прямой задачи:

Теперь воспользуемся формулами для нахождения нижней и верхней границ интервалов устойчивости оценок по видам ресурсов.

*Ресурс 1 (Масло)*. Найдем нижнюю границу. В четвёртом столбце обратной матрицы один положительный элемент (1), ему соответствует индекс базисной переменной оптимального плана ().

Найдем верхнюю границу. Среди элементов четвёртого столбца отсутствуют отрицательные элементы.

Таким образом, получаем

Тогда первый ресурс может изменяться в интервале:

При таком значении оптимальный план двойственной задачи остается неизменным. Аналогичные рассуждения позволяют найти интервалы устойчивости оценок для остальных ресурсов.

*Ресурс 2 (Яйца).* Рассматриваем третий столбец обратной матрицы, в котором один положительный элемент () и три отрицательных (, , ). Данным элементам соответствуют следующие индексы базисных переменных оптимального плана: положительного элемента – ; для отрицательных – ,150.

Тогда находим нижнюю границу.

Найдем верхнюю границу.

Выбираем наибольшее значение, равное .

Получаем

Тогда второй ресурс может изменяться в интервале:

*Ресурс 3 (Сахар).* Рассматриваем второй столбец обратной матрицы, в котором два положительных элемента (, ). Данным элементам соответствуют индексы соответствующего базисного переменного оптимального плана – 150, 150.

Находим нижнюю границу.

Выбираем наименьшее значение, равное 90.

Найдем верхнюю границу.

Выбираем наибольшее значение, равное .

Тогда, получаем что

Получаем интервал устойчивости оценок по отношению к третьему ограничению:

*Ресурс 4 (Молоко)*. Найдем нижнюю границу. В первом столбце обратной матрицы один положительный элемент (1), ему соответствует индекс базисной переменной оптимального плана ().

Найдем верхнюю границу. Среди элементов четвёртого столбца отсутствуют отрицательные элементы.

Таким образом, получаем

Тогда первый ресурс может изменяться в интервале:

Далее оценим влияние изменения объема ресурсов на величину максимальной стоимости продукции. Как известно, это дефицитные ресурсы  *).* Введем верхние границы в формулу:

Совместное влияние изменений этих ресурсов приводит к изменению максимальной стоимости продукции 𝐺𝑚𝑎𝑥 на величину:

Следовательно, оптимальное значение целевой функции при максимальном изменении ресурсов:

Таким образом, двойственные оценки позволяют судить о чувствительности решения к изменениям.

**ЗАКЛЮЧЕНИЕ**

Каждая из задач двойственной пары фактически является самостоятельной задачей линейного программирования и может быть решена не зависимо от другой. Связь задач заключается в том, что решение одной из них может быть получено непосредственно из решения другой. Взаимная симметрия прямой и двойственной задач определяет существование определенного соответствия между их оптимальными решениями. Эти соответствия устанавливают теоремы двойственности.

**СПИСОК ИНФОРМАЦИОННЫХ ИСТОЧНИКОВ**

1. Болотова Л. С. Многокритериальная оптимизация. Болотова Л. С., Сорокин А. Б. [Электронный ресурс] / Метод. указания по вып. курсовой работы — М.: МИРЭА, 2015.
2. Сорокин А. Б. Методы оптимизации: гибридные генетические алгоритмы. Сорокин А. Б. [Электронный ресурс] / Учебно-метод. пособие — М.: МИРЭА, 2016.
3. Сорокин А. Б. Линейное программирование: практикум. Сорокин А. Б., Бражникова Е. В., Платонова О. В. [Электронный ресурс] / Учебно-метод. пособие — М.: МИРЭА, 2017.

**ПРИЛОЖЕНИЯ**

Приложение А – Код реализации двойственной задачи на языке Python.

**Приложение А**

Код реализации двойственной задачи на языке Python

*Листинг А.1. Реализация двойственной задачи.*

import re

import math

import sympy

import numpy as np

NUM\_CRITERIA = 4 # Количество ограничений в математической моделе

PRECISION = 8 # Количество знаков после запятой при округлении

SEP = 25 # Разделитель для вывода таблицы

def print\_table(system, coef\_basis, coef\_not\_basis, basis\_values, not\_basis\_values):

print(' '.ljust(SEP), end='')

print('Cj'.ljust(SEP), end='')

for i in range(len(coef\_not\_basis) + 1):

if i == len(coef\_not\_basis):

print(' '.ljust(SEP))

else:

print(str(coef\_not\_basis[i]).ljust(SEP), end='')

print('Cv'.ljust(SEP), end='')

for i in range(len(not\_basis\_values) + 1):

if i == 0:

print(''.ljust(SEP), end='')

else:

print(str(not\_basis\_values[i-1]).ljust(SEP), end='')

print('A0'.ljust(SEP))

system.insert(0, coef\_basis + [' '])

system.insert(1, basis\_values + ['f'])

for column in range(len(system[0])):

for row in range(len(system)):

print(str(system[row][column]).ljust(SEP), end='')

print()

del system[0]

del system[0]

def get\_coefficients(data):

'''Функция для получения списка коэффициентов из системы ограничений'''

criteria\_coefficients, boundaries = list(), list()

for exp in data:

if '<=' in exp:

parse = exp.split('<=')

elif '>=' in exp:

parse = exp.split('>=')

elif '<' in exp:

parse = exp.split('<')

elif '>' in exp:

parse = exp.split('>')

elif '=' in exp:

parse = exp.split('=')

parse = list(map(str.strip, parse))

boundaries.append(float(parse[1]))

criteria\_coefficients.append(list(map(float, [i.group(1) for i in re.finditer(

r'(\d+(\.\d+)?) {0,}[\*]? {0,}\w', parse[0])])))

*Продолжение Листинга А.1.*

return criteria\_coefficients, boundaries

def count\_scalar\_product(vec1, vec2):

'''Функция для рассчёта скалярного произведения двух векторов'''

res = 0

for i in range(len(vec1)):

res += (vec1[i] \* vec2[i])

return res

def create\_simplex\_table(system, coef\_basis, coef\_not\_basis, basis\_values, not\_basis\_values):

F\_str = [0] \* len(not\_basis\_values)

for i in range(len(not\_basis\_values)):

F\_str[i] = count\_scalar\_product(

coef\_basis, system[i]) - coef\_not\_basis[i]

Q = count\_scalar\_product(coef\_basis, system[-1])

for i in range(len(F\_str)):

system[i].append(F\_str[i])

system[-1].append(Q)

return F\_str, Q

def simplex\_iteration(system, coef\_basis, coef\_not\_basis, basis\_values, not\_basis\_values, F\_str, Q):

index\_column = F\_str.index(min(F\_str))

mini = 1e10

for i in range(len(system[-1]) - 1):

tmp = system[-1][i] / system[index\_column][i]

if tmp < mini:

mini = tmp

index\_row = i

key\_element = system[index\_column][index\_row]

basis\_values.insert(index\_row, not\_basis\_values[index\_column])

not\_basis\_values.insert(index\_column, basis\_values.pop(index\_row + 1))

del not\_basis\_values[index\_row]

coef\_basis[index\_row], coef\_not\_basis[index\_column] = coef\_not\_basis[index\_column], coef\_basis[index\_row]

new\_key\_element = round(1 / key\_element, PRECISION)

data = [[0] \* (len(coef\_basis) + 1)

for \_ in range(len(coef\_not\_basis) + 1)]

for i in range(len(system[index\_column])):

data[index\_column][i] = - \

round(system[index\_column][i] / key\_element, PRECISION)

for i in range(len(system)):

data[i][index\_row] = round(

system[i][index\_row] / key\_element, PRECISION)

data[index\_column][index\_row] = new\_key\_element

for row in range(len(data[0])):

for column in range(len(data)):

if data[column][row] == 0:

data[column][row] = round(((system[column][row] \* key\_element) - (

system[index\_column][row] \* system[column][index\_row])) / key\_element, PRECISION)

F\_str = [data[i][-1] for i in range(len(data) - 1)]

Q = data[-1][-1]

return data, coef\_basis, coef\_not\_basis, basis\_values, not\_basis\_values, F\_str, Q

*Продолжение Листинга А.1.*

def check\_inequality(inequality, variables, индексы\_базисных\_переменных\_оптимального\_плана):

inequality = inequality.replace('\*', '')

for i in range(len(variables)):

if variables[i] not in inequality:

continue

inequality = inequality.replace(

variables[i], '\*' + str(индексы\_базисных\_переменных\_оптимального\_плана[i]))

# Символьное вычисление неравенства

inequality += '- 0.1'

result = str(sympy.sympify(inequality))

# Возвращение True, если неравенство выполняется, иначе False

return eval(result)

def dual\_task():

коэффициенты\_целевой\_функции = np.array(target\_coefficients)

свободные\_члены\_неравенств = np.array(boundaries)

матрица\_ограничений, \_ = get\_coefficients(criteria\_function)

транспонированная\_матрица\_ограничений = np.transpose(матрица\_ограничений)

индексы\_базисных\_переменных\_оптимального\_плана = np.array(system[-1][:-1])

y = np.array([])

D = list()

for i in range(len(basis\_values)):

index = int(basis\_values[i][1:]) - 1

if index < len(транспонированная\_матрица\_ограничений):

D.append(транспонированная\_матрица\_ограничений[index])

else:

D.append(

np.array([1 if i == j else 0 for j in range(NUM\_CRITERIA)]))

D\_inversed = np.linalg.inv(np.transpose(D))

def first\_duality\_theorem():

y = np.dot(np.array(coef\_basis), D\_inversed)

G\_min = np.dot(свободные\_члены\_неравенств, y)

print(f"Gmin is {G\_min} by first\_duality\_theorem")

assert abs(G\_min - Q) < 0.00001

def second\_duality\_theorem():

nonlocal y

zeros = list()

for i in range(NUM\_CRITERIA):

if check\_inequality(

criteria\_function[i], basis\_values, индексы\_базисных\_переменных\_оптимального\_плана):

zeros.append(i)

система\_уравнений = транспонированная\_матрица\_ограничений.copy()

for i in range(len(zeros)):

система\_уравнений = np.delete(

система\_уравнений, zeros[i], 1)

for j in range(i + 1, len(zeros)):

zeros[j] -= 1

y = np.linalg.solve(система\_уравнений, коэффициенты\_целевой\_функции)

for i in range(len(zeros)):

y = np.insert(y, zeros[i], 0)

for j in range(i + 1, len(zeros)):

zeros[j] += 1

G\_min = np.dot(свободные\_члены\_неравенств, y)

*Продолжение Листинга А.1.*

print(f"Gmin is {G\_min} by second\_duality\_theorem")

assert abs(G\_min - Q) < 0.00001

def third\_duality\_theorem():

нижняя\_граница = list()

верхняя\_граница = list()

b = list()

for i in range(len(D\_inversed) - 1, -1, -1):

pozitive = list()

negative = list()

bH = - math.inf

bB = math.inf

for j in range(len(D\_inversed)):

if D\_inversed[j][i] > 0:

pozitive.append(

(свободные\_члены\_неравенств[j], D\_inversed[j][i]))

elif D\_inversed[j][i] < 0:

negative.append(

(свободные\_члены\_неравенств[j], D\_inversed[j][i]))

if len(pozitive) > 1:

elem = min(pozitive, key=lambda x: abs(

pozitive[0][0] / pozitive[0][1]))

нижняя\_граница.append(elem[0] / elem[1])

elif len(pozitive) == 1:

нижняя\_граница.append(pozitive[0][0] / pozitive[0][1])

else:

нижняя\_граница.append(bH)

if len(negative) > 1:

elem = max(negative, key=lambda x: abs(

negative[0][0] / negative[0][1]))

верхняя\_граница.append(abs(elem[0] / elem[1]))

elif len(negative) == 1:

верхняя\_граница.append(negative[0][0] / negative[0][1])

else:

верхняя\_граница.append(bB)

b.append(свободные\_члены\_неравенств[i])

print(f'Ресурс №{len(D\_inversed)-i}')

print(

f'b{len(D\_inversed)-i} ∈ ({нижняя\_граница[-1]}; {верхняя\_граница[-1]})')

print(f'{len(D\_inversed)-i}-й ресурс изменяется в интервале: ', end='')

if нижняя\_граница[-1] == - math.inf:

print(f'({нижняя\_граница[-1]}; ', end='')

else:

print(f'({b[-1] - нижняя\_граница[-1]}; ', end='')

if верхняя\_граница[-1] == math.inf:

print(f'{верхняя\_граница[-1]})')

else:

print(f'{b[-1] + верхняя\_граница[-1]})')

total = 0

for i in range(len(y)):

if y[i] != 0:

total += y[i] \* верхняя\_граница[i]

print(f'∆Gmax{i + 1} = y{i+1} \* bB{i +

1} = {y[i] \* верхняя\_граница[i]}')

print(f'Совместное влияние изменений этих ресурсов приводит к изменению максимальной стоимости продукции 𝐺𝑚𝑎𝑥 на величину: {total}')

print(f'Следовательно, оптимальное значение целевой функции при

*Продолжение Листинга А.1.*

максимальном изменении ресурсов: {Q+total}')

first\_duality\_theorem()

second\_duality\_theorem()

third\_duality\_theorem()

with open('TPR\_PRACT5.csv', encoding='utf-8') as file:

target\_function = file.readline().rstrip() # Целевая функция

target\_coefficients = list(map(float, [i.group(1) for i in re.finditer(

# Список коэффициентов целевой функции

r'(\d+(\.\d+)?) {0,}[\*]? {0,}\w', target\_function)]))

criteria\_function = [file.readline().rstrip() for \_ in range(NUM\_CRITERIA)]

criteria\_coefficients, boundaries = get\_coefficients(criteria\_function)

print('Переходим к задаче линейного программирования:',

target\_function, sep='\n')

for i in criteria\_function:

print("{ " + i)

system = list(map(list, list(zip(\*criteria\_coefficients))))

system.append(boundaries.copy())

# Вектор коэффициентов целевой функции при базисных переменных

coef\_basis = [0] \* NUM\_CRITERIA

# Коэффициенты целевой функции, соответствующие небазисным переменным

coef\_not\_basis = target\_coefficients.copy()

not\_basis\_values = re.findall(r'[A-Za-z]\d{1,}', target\_function)

basis\_values = [f'{not\_basis\_values[-1][0]}{i}' for i in range(

int(not\_basis\_values[-1][1]) + 1, NUM\_CRITERIA + int(not\_basis\_values[-1][1]) + 1)]

F\_str, Q = create\_simplex\_table(

system, coef\_basis, coef\_not\_basis, basis\_values, not\_basis\_values)

num\_iteration = 0

while num\_iteration < 50 and min(F\_str) < 0:

print(

('\x1b[6;30;42m' + f"Итерация №{num\_iteration}" + '\x1b[0m').center(201))

system, coef\_basis, coef\_not\_basis, basis\_values, not\_basis\_values, F\_str, Q = simplex\_iteration(

system, coef\_basis, coef\_not\_basis, basis\_values, not\_basis\_values, F\_str, Q)

print\_table(system, coef\_basis, coef\_not\_basis,

basis\_values, not\_basis\_values)

num\_iteration += 1

if num\_iteration != 50:

print(f'Решение найдено! Общая прибыль составила {

round(Q, 3)} денежных единиц')

else:

print('Поставленная задача решения не имеет')

exit(0)

dual\_task()