Министерство Образования Республики Беларусь

Учреждение образования

«Гомельский государственный технический университет

имени П. О. Сухого»

Кафедра «Информатика»

**Лабораторная работа № 1**

по дисциплине: **«Системный анализ и исследование операций»**

Выполнил студент

группы ИП-41

Коваленко А.И.

Проверил преподаватель

Бородин Н.Н.

**ЛАБОРАТОРНАЯ РАБОТА №1**

**ПРОГРАММНАЯ РЕАЛИЗАЦИЯ РЕШЕНИЯ ЗАДАЧ ЦЕЛОЧИСЛЕННОГО ПРОГРАММИРОВАНИЯ**

**Цель работы:** получить практические навыки решения задач целочисленного программирования.

**Задание**

1. Решить задачи целочисленного программирования (варианты см. в таблице 1.1); проверить правильность полученных решений средствами пакетов *Mathcad* или *MS Excel*.

2) Решить (см. общую постановку ЗЦП и таблицу вариантов ниже):

а) частично целочисленную задачу ( – любое, – целочисленное);

б) полностью целочисленную задачу ( и  – целочисленные)

средствами пакетов Mathcad (графическим методом и с помощью блока решения) и MS Excel (с помощью надстройки «Поиск решения»). Сравнить полученные результаты.









, 

**Вариант задания**

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Вариант** | **a** | **b** | **c** | **d** | **e** | **f** | **g** | **h** | **p** | **r** | **q** |
| **14** | 17 | 6 | 5 | -5 | 4 | -6 | 7 | 5 | 2 | 2 | 7 |

Система ограничений:

Целевая функция:

**Программная реализация:**

import numpy as np  
import matplotlib.pyplot as plt  
  
def solve\_linear\_programming(*a*, *b*, *c*, *d*, *e*, *f*, *g*, *h*, *p*, *r*, *q*):  
 x = np.linspace(0, 10, 200)  
 y1 = (*e* - *c* \* x) / *d* y2 = (*h* - *f* \* x) / *g* y3 = (*q* - *p* \* x) / *r  
  
 # First graph: Maximize (x любое)* plt.subplot(1, 2, 1)  
 plt.plot(x, y1, *label*='c\*x + d\*y <= e')  
 plt.plot(x, y2, *label*='f\*x + g\*y <= h')  
 plt.plot(x, y3, *label*='p\*x + r\*y <= q')  
  
 x\_vals = np.linspace(0, 10, 500)  
 y\_vals = np.linspace(0, 10, 500)  
 X, Y = np.meshgrid(x\_vals, y\_vals)  
  
 feasible\_region = (*c* \* X + *d* \* Y <= *e*) & (*f* \* X + *g* \* Y <= *h*) & (*p* \* X + *r* \* Y <= *q*)  
 plt.imshow(feasible\_region, *extent*=(0, 10, 0, 10), *origin*='lower', *cmap*='Blues', *alpha*=0.3)  
  
 *# Constraints: x >= 0 and y >= 0* plt.axvline(*x*=0, *color*='black', *linewidth*=0.5)  
 plt.axhline(*y*=0, *color*='black', *linewidth*=0.5)  
  
 *# Find maximum point (x любое)* max\_point = None  
 max\_value = *float*('-inf')  
  
 for x\_val in np.linspace(0, 10, 1000):  
 for y\_val in *range*(11):  
 if (*c* \* x\_val + *d* \* y\_val <= *e*) and (*f* \* x\_val + *g* \* y\_val <= *h*) and (*p* \* x\_val + *r* \* y\_val <= *q*):  
 current\_value = *a* \* x\_val + *b* \* y\_val  
 if current\_value > max\_value:  
 max\_value = current\_value  
 max\_point = [x\_val, y\_val]  
 *print*("Значение целевой функции для одной целочисленной переменной")  
 *print*(*a*\*max\_point[0]+*b*\*max\_point[1])  
 if max\_point is not None:  
 plt.scatter(\*max\_point, *color*='red', *label*='Максимум (x любое)')  
 plt.annotate(f'({max\_point[0]:.2f}, {max\_point[1]:.0f})', *xy*=max\_point, *xytext*=(5, -10),  
 *textcoords*='offset points')  
 X, Y = np.meshgrid(x\_vals, y\_vals)  
 Z = *a* \* X + *b* \* Y  
 plt.contour(X, Y, Z, *levels*=[max\_value], *colors*=['red'], *extend*='both')  
 *# Find other feasible points* feasible\_points = []  
 for x\_val in np.linspace(0, 10, 1000):  
 for y\_val in *range*(11):  
 if (*c* \* x\_val + *d* \* y\_val <= *e*) and (*f* \* x\_val + *g* \* y\_val <= *h*) and (*p* \* x\_val + *r* \* y\_val <= *q*):  
 if [x\_val, y\_val] != max\_point:  
 feasible\_points.append([x\_val, y\_val])  
  
 feasible\_points = np.array(feasible\_points)  
 if *len*(feasible\_points) > 0:  
 plt.scatter(feasible\_points[:, 0], feasible\_points[:, 1], *color*='orange', *label*='Допустимые точки', *s*=1)  
  
 *# Add vector of increase* start\_point = max\_point  
 direction = np.array([1, 1]) *# Example direction, you can modify it as needed* plt.scatter([start\_point[0] + direction[0]], [start\_point[1] + direction[1]], *color*='red',  
 *label*='Вектор возрастания')  
 plt.xlabel('x')  
 plt.ylabel('y')  
 plt.legend()  
 plt.grid(True)  
  
 *# Second graph: Maximize (x и y целочисленные)* plt.subplot(1, 2, 2)  
 plt.plot(x, y1, *label*='c\*x + d\*y <= e')  
 plt.plot(x, y2, *label*='f\*x + g\*y <= h')  
 plt.plot(x, y3, *label*='p\*x + r\*y <= q')  
  
 x\_vals = np.linspace(0, 10, 500)  
 y\_vals = np.linspace(0, 10, 500)  
 X, Y = np.meshgrid(x\_vals, y\_vals)  
  
 feasible\_region = (*c* \* X + *d* \* Y <= *e*) & (*f* \* X + *g* \* Y <= *h*) & (*p* \* X + *r* \* Y <= *q*)  
 plt.imshow(feasible\_region, *extent*=(0, 10, 0, 10), *origin*='lower', *cmap*='Blues', *alpha*=0.3)  
  
 *# Constraints: x >= 0 and y >= 0* plt.axvline(*x*=0, *color*='black', *linewidth*=0.5)  
 plt.axhline(*y*=0, *color*='black', *linewidth*=0.5)  
  
 *# Find maximum point (x и y целочисленные)* max\_point\_both\_int = None  
 max\_value\_both\_int = *float*('-inf')  
  
 for x\_val in *range*(11):  
 for y\_val in *range*(11):  
 if (*c* \* x\_val + *d* \* y\_val <= *e*) and (*f* \* x\_val + *g* \* y\_val <= *h*) and (*p* \* x\_val + *r* \* y\_val <= *q*):  
 current\_value\_both\_int = *a* \* x\_val + *b* \* y\_val  
 if current\_value\_both\_int > max\_value\_both\_int:  
 max\_value\_both\_int = current\_value\_both\_int  
 max\_point\_both\_int = [x\_val, y\_val]  
 if max\_point\_both\_int is not None:  
 plt.scatter(\*max\_point\_both\_int, *color*='blue', *label*='Максимум (x и y целочисленные)')  
 plt.annotate(f'({max\_point\_both\_int[0]:.0f}, {max\_point\_both\_int[1]:.0f})',  
 *xy*=max\_point\_both\_int,  
 *xytext*=(5, -10),  
 *textcoords*='offset points')  
 X, Y = np.meshgrid(x\_vals, y\_vals)  
 Z = *a* \* X + *b* \* Y  
  
 plt.contour(X, Y, Z, *levels*=[max\_value\_both\_int], *colors*=['red'], *extend*='both')  
 *# Find other feasible points* feasible\_points\_both\_int = []  
 for x\_val in *range*(11):  
 for y\_val in *range*(11):  
 if (*c* \* x\_val + *d* \* y\_val <= *e*) and (*f* \* x\_val + *g* \* y\_val <= *h*) and (  
 *p* \* x\_val + *r* \* y\_val <= *q*):  
 if [x\_val, y\_val] != max\_point\_both\_int:  
 feasible\_points\_both\_int.append([x\_val, y\_val])  
  
 feasible\_points\_both\_int = np.array(feasible\_points\_both\_int)  
 if *len*(feasible\_points\_both\_int) > 0:  
 plt.scatter(feasible\_points\_both\_int[:, 0], feasible\_points\_both\_int[:, 1], *color*='purple',  
 *label*='Допустимые точки (x и y целочисленные)')  
  
 *# Add the line coinciding with the contour line of the objective function through the second maximum point  
  
 # Add vector of increase* start\_point = max\_point  
 direction = np.array([1, 1]) *# Example direction, you can modify it as needed* plt.scatter([start\_point[0] + direction[0]], [start\_point[1] + direction[1]], *color*='red',  
 *label*='Вектор возрастания')  
  
 plt.xlabel('x')  
 plt.ylabel('y')  
 plt.legend()  
 plt.grid(True)  
  
 *# Add contour line for the objective function in the second graph* X, Y = np.meshgrid(x\_vals, y\_vals)  
 Z = *a* \* X + *b* \* Y  
 plt.contour(X, Y, Z, *levels*=[max\_value\_both\_int], *colors*=['red'], *extend*='both')  
  
 *# Set the x-axis limits to include the left part of the graph* plt.xlim(0, 10)  
 *print*("Значение целевой функции для двух целочисленных переменных")  
 *print*(*a* \* max\_point\_both\_int[0] + *b* \* max\_point\_both\_int[1])  
 plt.tight\_layout()  
 plt.show()  
  
a = 17  
b = 6  
c = 5  
d = -5  
e = 4  
f = -6  
g = 7  
h = 5  
p = 2  
r = 2  
q = 7  
solve\_linear\_programming(a, b, c, d, e, f, g, h, p, r, q)

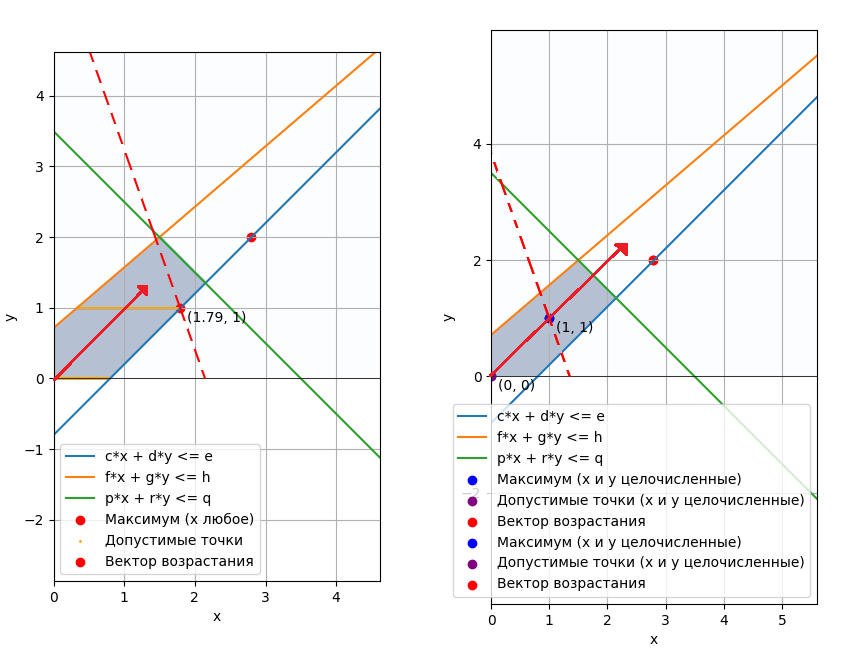


Рисунок 1 — график системы в python

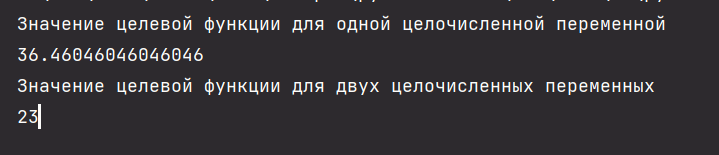


Рисунок 2 — значение целевой функции

**Вывод:** были получены практические навыки решения задач целочисленного программирования.