Міністерство освіти і науки України
Національний авіаційний університет
Факультет кібербезпеки, комп’ютерної та програмної інженерії
Кафедра комп’ютеризованих систем управління

Лабораторна робота № 1.2
з дисципліни «Дослідження операцій»
на тему «Побудова оптимізаційних економіко-математичних моделей.
Графічний метод розв’язку задачі лінійного програмування»

Виконав:
студент ФККПІ
групи СП-425
Клокун В. Д.
Перевірила:
Яковенко Л. В.

Київ 2019

1. Завдання роботи

Розв'язати графічним методом:

$$L = -4x_1 - 5x_2 \to \min,$$
$$\begin{cases} 4x_1 + 5x_2 \leqslant 20, \\ -x_1 + 10x_2 \geqslant 10, \\ 7x_1 + 5x_2 \leqslant 35, \\ x_1 \leqslant 7, \\ x_2 \leqslant 4, \\ x_1 \geqslant 0, \\ x_2 \geqslant 0. \end{cases}$$

2. Хід роботи

Щоб розв'язати поставлену задачу графічним методом, необхідно визначити область можливих розв'язків за допомогою многокутника обмежень. Многокутник обмежень будується на основі півплощин, які відповідають нерівностям із системи обмежень. Розробимо програму, яка побудує многокутник обмежень поставленої задачі та покаже область можливих розв'язків (лістинг А.1). Програма виводить рисунок для графічного розв'язку задачі на екран (рис. 1).

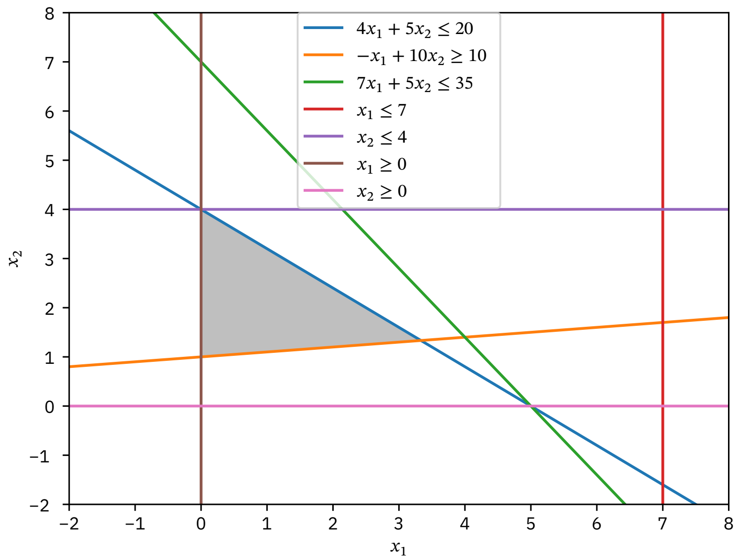

Рис. 1

За отриманим рисунком видно, що вершини многокутника обмежень мають такі координати: $A\,(0;1)$, $B\,(0;4)$, $C\,(10/3;4/3)$ (рис. 2).

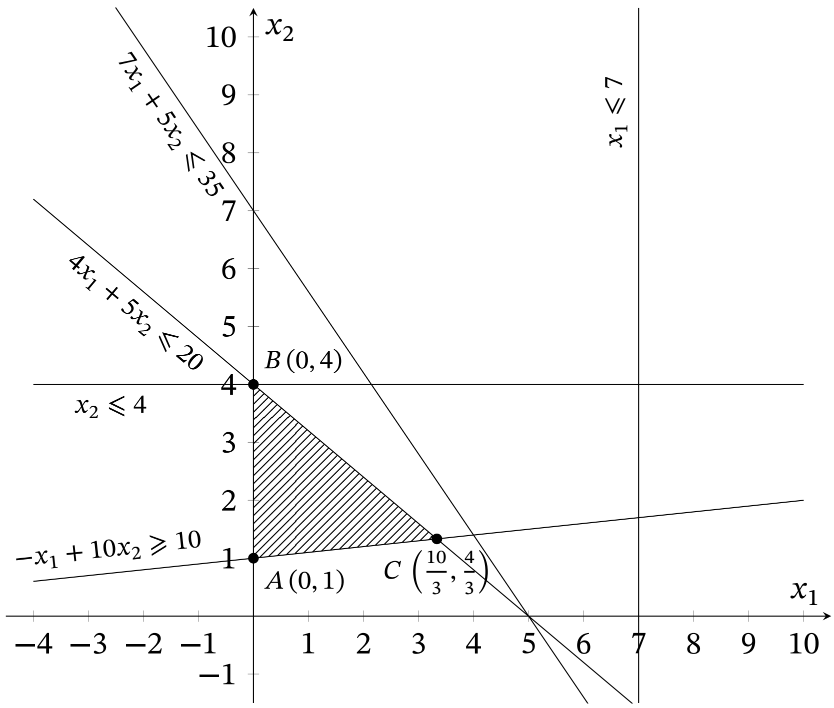

Рис. 2: Графік області можливих розв'язків

Відомо, що якщо задача лінійного програмування має оптимальне рішення, то воно співпадає з однією (двома) вершинами многокутника обмежень. Отже, щоб знайти оптимальне рішення, необхідно визначити вершину, при якій значення функції найменше. Для цього підставляємо значення координат у цільову функцію і знаходимо її значення:

$$
\begin{aligned}
L(A) &= -4 \cdot 0 - 5 \cdot 1 = -5,\\
L(B) &= -4 \cdot 0 - 5 \cdot 4 = -20,\\
L(C) &= -4 \cdot \frac{10}{3} - 5 \cdot \frac{4}{3} = \frac{-40}{3} - \frac{20}{3} = \frac{-60}{3} = -20.
\end{aligned}
$$

Як бачимо, цільова функція L набуває найменших значень у точках $B\,(0;4)$ і $C\,(10/3;4/3)$, а отже розв'язками задачі будуть такі пари значень керованих змінних: $x_1 = 0,\ x_2 = 4$ та $x_1 = 10/3,\ x_2 = 4/3$.

3. Висновок

Виконуючи дану лабораторну роботу, ми навчились використовувати графічний метод розв'язання задач лінійного програмування, а також розробляти програмне забезпечення для допомоги при розв’язанні задачі лінійного програмування графічним методом.

A. Програма для розв’язку поставленої задачі

Лістинг A.1: Початковий код програми для побудови півплощин обмежень

```
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.ticker as pltticker

BOUNDS_X = (-2, 8)
BOUNDS_Y = (-2, 8)

LABEL_X = r'$x_1$'
LABEL_Y = r'$x_2$'


def format_axes(
        ax,
        xlim=BOUNDS_X,
        ylim=BOUNDS_Y,
        xlabel=LABEL_X,
        ylabel=LABEL_Y,
        tick_every=1.0,):
    """Formats the plot axes.

    Args:
        ax (:obj:`matplotlib.Axes`): an Axes object to be formatted.
        xlim (tuple): tuple of x limits in the form of (x_min, x_max)
        ylim (tuple): tuple of y limits in the form of (y_min, y_max)
        xlabel (str): OX axis label.
        ylabel (str): OY axis label.
        tick_every (float): create ticks using this period.
    """
    ax.set_xlim(BOUNDS_X)
    ax.set_ylim(BOUNDS_Y)
    ax.set_xlabel(xlabel)
    ax.set_ylabel(ylabel)

    # Tick every 1.0
    ticker = pltticker.MultipleLocator(base=tick_every)
    ax.xaxis.set_major_locator(ticker)
    ax.yaxis.set_major_locator(ticker)


def apply_restriction(x, y, restriction):
```

```
    """Applies the given restriction to the feasible region.

    Args:
        x (np.array): array of X values.
        y (np.array): array of Y values.
        restriction (callable): a restriction function.

    Returns:
        An `np.array` of True / False values corresponding to whether the
        solution applies here or not.
    """
    restricted = [restriction(x, y) for x, y in zip(x, y)]

    return restricted


# Create 2000 evenly spaced sample points in the BOUNDS_X interval
x1 = np.arange(
    *(BOUNDS_X),
    step=0.01,
)

# Create points for linear restrictions
x2_1 = 4 - 0.8 * x1
x2_2 = 1 + 0.1 * x1
x2_3 = 7 - 1.4 * x1

x1_2 = (0 * x1) + 7
x2_4 = (0 * x1) + 4
x1_3 = (0 * x1) + 0
x2_5 = (0 * x1) + 0

# Plot restrictions
fig, ax = plt.subplots()
ax.plot(x1, x2_1, label=r'$4 x_1 + 5 x_2 \leq 20$')
ax.plot(x1, x2_2, label=r'$-x_1 + 10 x_2 \geq 10$')
ax.plot(x1, x2_3, label=r'$7 x_1 + 5 x_2 \leq 35$')

ax.plot(x1_2, x1, label=r'$x_1 \leq 7$')
ax.plot(x1, x2_4, label=r'$x_2 \leq 4$')
ax.plot(x1_3, x1, label=r'$x_1 \geq 0$')
ax.plot(x1, x2_5, label=r'$x_2 \geq 0$')

# Create points for solution space polygon
solution_space_lim_hi = np.minimum(x2_1, x2_3)
solution_space_lim_lo = np.maximum(x2_2, x2_5)

```

```
feasible_region = solution_space_lim_lo < solution_space_lim_hi

# Apply x_1 >= 0 restriction
feasible_region = apply_restriction(
    x1,
    feasible_region,
    restriction=lambda x, y: False if x < 0 else y
)


# Plot solution space polygon
ax.fill_between(
    x1,
    solution_space_lim_lo,
    solution_space_lim_hi,
    where=feasible_region,
    color='grey',
    alpha=0.5,
)

format_axes(ax)

plt.legend(
    # loc=2,
    borderaxespad=0.0,
)
plt.show()
```