



Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Московский государственный технический университет
имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ
КАФЕДРА

Информатика и системы управления (ИУ)
Информационная безопасность (ИУ8)

Теория игр и исследование операций

Расчётно-графическое домашнее задание

Вариант: 1

Студент:

Александров Алексей Николаевич, группа ИУ8-104
(5 курс)

(подпись, дата)

Преподаватель:

к.т.н., доцент кафедры ИУ8
Коннова Наталья Сергеевна

(подпись, дата)

Оглавление

Цель работы	3
Задание.....	3
Ход работы	4
1. Нормализация матрицы	5
2. Уменьшение размерности матричной игры.....	5
3. Графоаналитический метод	7
4. Аналитический (матричный) метод.....	9
5. Графический метод (задача ЛП)	11
6. Симплекс-метод (задача ЛП).....	14
7. Расчёт цены игры исходной матрицы	16
Вывод.....	17
ПРИЛОЖЕНИЕ А Листинг интерактивного блокнота rgz.ipynb	18
ПРИЛОЖЕНИЕ Б Ссылка на репозиторий с исходным кодом задачи.....	33

Цель работы

Изучить постановку антагонистической игры двух лиц с нулевой суммой. Получить практические навыки нахождения решения игры за обоих игроков в смешанных стратегиях следующими методами:

- графоаналитическим;
- аналитическим (матичным);
- графическим (задача линейного программирования);
- симплекс-методом (задача линейного программирования).

Задание

Задана платежная матрица прямоугольной игры с нулевой суммой (на рисунке 1 приведена матрица по варианту).

4	-3	5	6	4
6	5	-3	4	7
6	5	-3	-3	5
-3	-3	4	4	4
7	6	4	5	6

Рисунок 1 – Матрица игры с нулевой суммой для варианта 1

1. Нормализовать матрицу (привести к матрице с неотрицательными элементами) и свести исходную игру к матричной игре размерности 2×2 следующими способами:

1. путем поглощения доминируемых стратегий;
2. путем удаления NBR-стратегий.

2. Найти смешанные стратегии игроков следующими методами:

1. графоаналитическим;
2. аналитическим (матричным);
3. графически метод (задача линейного программирования);
4. симплекс-методом (задача линейного программирования).

3. Рассчитать цену игры для исходной матрицы.

Ход работы

Для реализации решения расчётно-графического домашнего задания был использован язык программирования Python. К защите представляется интерактивный блокнот Jupyter Notebook в файле rgz.ipynb (см. приложение А).

В исполняемом «ноутбуке» импортируются реализованные модули *matrix_games* и *simplex*, инкапсулирующие логику и алгоритмы для матричных игр и симплекс-метода соответственно. Ознакомиться со всем исходным кодом данной работы можно, посетив репозиторий, ссылка на который представлена в приложении Б. Инициализация матрицы представлена на рисунке 2. Нижняя и верхняя цены игры: 4 и 5. Седловой точки нет. Для идентификации назовём игроков А и В.

```
[1]: import json
import logging
from pathlib import Path

import matplotlib.pyplot as plt
import numpy as np

from game_theory.utils.matrix_games.analytical import AnalyticalSolver
from game_theory.utils.matrix_games.game_matrix import GameMatrix
from game_theory.utils.matrix_games.mixed_strategies import (
    check_resulted_game_price,
    get_resulted_mixed_strategies,
)
from game_theory.utils.simplex.dual_problem import DualProblem
from game_theory.utils.simplex.simplex_problem import SimplexProblem

logging.basicConfig(level=logging.INFO, format='%(message)s')
```

```
[2]: # Входная матрица прямоугольной игры с нулевой суммой.
matrix = np.array(
    [
        [4, -3, 5, 6, 4],
        [6, 5, -3, 4, 7],
        [6, 5, -3, -3, 5],
        [-3, -3, 4, 4, 4],
        [7, 6, 4, 5, 6],
    ],
    dtype=int,
)

game_matrix = GameMatrix(matrix)
game_matrix
```

```
[2]: +-----+
|               Таблица стратегий (игрока А)               |
+-----+-----+-----+-----+-----+-----+-----+
| Стратегии | b1 | b2 | b3 | b4 | b5 | MIN выигрыш А |
+-----+-----+-----+-----+-----+-----+-----+
| a1         | 4  | -3 | 5  | 6  | 4  | -3             |
| a2         | 6  | 5  | -3 | 4  | 7  | -3             |
| a3         | 6  | 5  | -3 | -3 | 5  | -3             |
| a4         | -3 | -3 | 4  | 4  | 4  | -3             |
| a5         | 7  | 6  | 4  | 5  | 6  | 4              |
| MAX проигрыш В | 7  | 6  | 5  | 6  | 7  |                |
+-----+-----+-----+-----+-----+-----+-----+
```

Рисунок 2 – Задание исходной матрицы игры с нулевой суммой

1. Нормализация матрицы

Для нормализации матрицы достаточно прибавить ко всем элементам матрицы максимальное по модулю отрицательное число (если оно существует). В нашем случае это слагаемое 3 (см. рисунок 3). Нижняя и верхняя цены игры: 7 и 8.

1. Нормализация матрицы. Уменьшение размерности исходной матричной игры

```
[4]: normalizer: int = game_matrix.normalize_matrix()  
game_matrix
```

Прибавили ко всем элементам исходной матрицы 3

```
[4]:
```

Таблица стратегий (игрока A)						
Стратегии	b1	b2	b3	b4	b5	MIN выигрыш A
a1	7	0	8	9	7	0
a2	9	8	0	7	10	0
a3	9	8	0	0	8	0
a4	0	0	7	7	7	0
a5	10	9	7	8	9	7
MAX проигрыш B	10	9	8	9	10	

```
[5]: print(f"Нижняя цена игры: {game_matrix.lowest_game_price[1]}\n"  
        f"Верхняя цена игры: {game_matrix.highest_game_price[1]}")
```

Нижняя цена игры: 7

Верхняя цена игры: 8

Рисунок 3 – Нормализация прямоугольной матрицы игры

2. Уменьшение размерности матричной игры

Для упрощения, сведём матричную игру 5x5 к матричной игре 2x2 двумя способами: поглощением доминируемых стратегий и удалением NBR-стратегий (Never Best Response). Но для начала следует проверить наличие дублирующихся стратегий (их тоже следует удалить). В нашем случае таких не нашлось, так что сразу переходим к основным методом уменьшения размерности.

Доминирующая строка (столбец) содержит элементы, поразрядно не меньшие (не бóльшие) элементов доминируемой строки (столбца). Итеративным обходом таблицы стратегий поглощаются стратегии a_2 , a_3 , a_4 , b_1 , b_4 , b_5 . (см. рисунок 4).

```
[6]: dominant_reduced_game: GameMatrix = game_matrix.reduce_dimension(method='dominant_absorption')
      dominant_reduced_game
```

```
a2 > a3: поглощение стратегии a3 доминирующей стратегией a2
a1 > a4: поглощение стратегии a4 доминирующей стратегией a1
b2 > b1: поглощение стратегии b1 доминирующей стратегией b2
b3 > b4: поглощение стратегии b4 доминирующей стратегией b3
b2 > b5: поглощение стратегии b5 доминирующей стратегией b2
a5 > a2: поглощение стратегии a2 доминирующей стратегией a5
```

```
[6]: +-----+
      |           Таблица стратегий (игрока A)           |
      +-----+-----+-----+-----+
      | Стратегии | b2 | b3 | MIN выигрыш A |
      +-----+-----+-----+-----+
      |      a1     |  0 |  8 |           0     |
      |      a5     |  9 |  7 |           7     |
      | MAX проигрыш B |  9 |  8 |
      +-----+-----+-----+-----+
```

```
[7]: print(f"Нижняя цена игры: {dominant_reduced_game.lowest_game_price[1]}\n"
      f"Верхняя цена игры: {dominant_reduced_game.highest_game_price[1]}")
```

```
Нижняя цена игры: 7
Верхняя цена игры: 8
```

Рисунок 4 – Снижение размерности методом доминируемых стратегий

Аналогичное проделаем методом удаление NBR-стратегий. NBR-строкой (столбцом) назовём такую строку (столбец), которая объективно не будет разыгрываться игроком А (В) для всех наперёд фиксированных стратегий В (А). На рисунке 5 легко убедиться, что алгоритм, реализующий данный метод, даёт на выходе ту же матрицу игры размера 2x2.

```
[8]: # Вычеркиваем столбцы и строки, которые мы точно не выберем при фиксированной стратегии.
      nbr_reduced_game: GameMatrix = game_matrix.reduce_dimension(method='nbr_drop')
      nbr_reduced_game
```

```
Удаление NBR-стратегий ['a3', 'a4']
Удаление NBR-стратегий ['b1', 'b4', 'b5']
Удаление NBR-стратегий ['a2']
```

```
[8]: +-----+
      |           Таблица стратегий (игрока A)           |
      +-----+-----+-----+-----+
      | Стратегии | b2 | b3 | MIN выигрыш A |
      +-----+-----+-----+-----+
      |      a1     |  0 |  8 |           0     |
      |      a5     |  9 |  7 |           7     |
      | MAX проигрыш B |  9 |  8 |
      +-----+-----+-----+-----+
```

```
[9]: print(f"Нижняя цена игры: {nbr_reduced_game.lowest_game_price[1]}\n"
      f"Верхняя цена игры: {nbr_reduced_game.highest_game_price[1]}")
```

```
Нижняя цена игры: 7
Верхняя цена игры: 8
```

```
[10]: assert dominant_reduced_game == nbr_reduced_game
       reduced_game: GameMatrix = nbr_reduced_game
```

Рисунок 5 – Снижение размерности удалением NBR-стратегий

3. Графоаналитический метод

Пусть x_1 – вероятность выбора игроком А стратегии a_1 . $x_5 = 1 - x_1$ – вероятность выбора игроком А стратегии a_5 .

Ожидаемый выигрыш А при реализации стратегии b_2 :

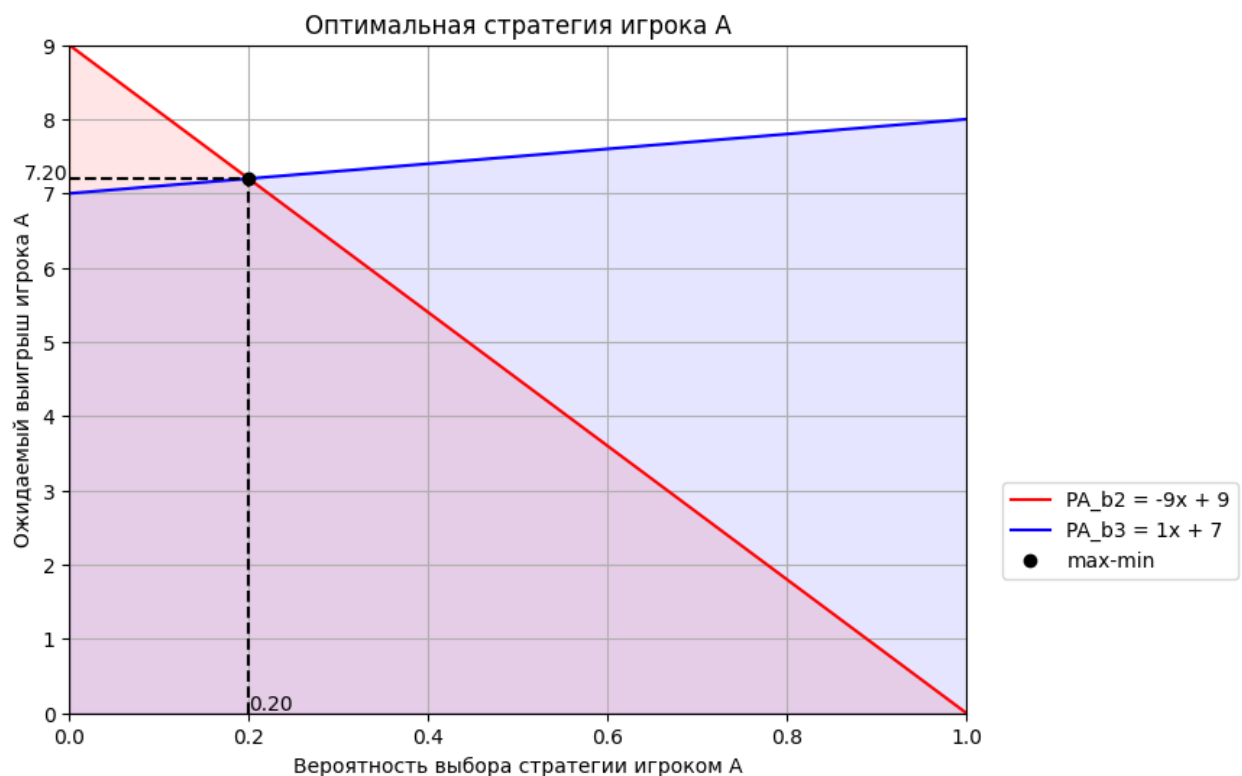
$$PA_{b_2} = c_{12}x_1 + c_{52}x_5 = (c_{12} - c_{52})x_5 + c_{52}$$

Ожидаемый выигрыш А при реализации стратегии b_3 :

$$PA_{b_3} = c_{13}x_1 + c_{53}x_5 = (c_{13} - c_{53})x_5 + c_{53}$$

Оптимальная стратегия А: $PA_{b_2} = PA_{b_3}$.

На рисунке 6 найдено решение для оптимальной стратегии игрока А. Таким образом смешанные стратегии игрока А: $(0,2; 0; 0; 0; 0,8)$. Цена игры: 7,2.



Смешанные стратегии игрока А и цена игры. ●●●

Цена игры: 7 ≤ 7.200 ≤ 8

Смешанные стратегии игрока А				
a1	a2	a3	a4	a5
0.20	0.00	0.00	0.00	0.80

Рисунок 6 – Графоаналитический метод поиска смешанных стратегий игрока А

Пусть y_2 – вероятность выбора игроком В стратегии b_2 . $y_3 = 1 - y_2$ – вероятность выбора игроком В стратегии b_3 .

Ожидаемый проигрыш В при реализации стратегии a_1 :

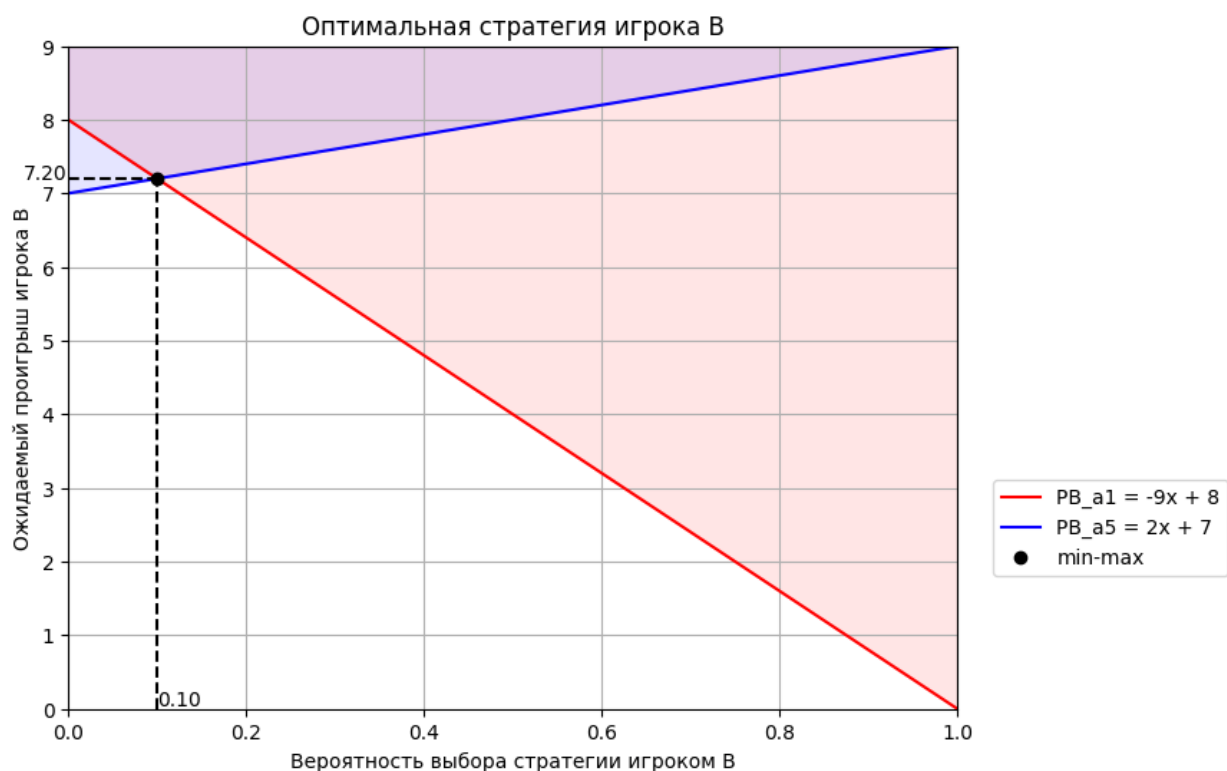
$$PB_{a1} = c_{12}y_2 + c_{13}y_3 = (c_{12} - c_{13})y_2 + c_{13}$$

Ожидаемый проигрыш В при реализации стратегии a_5 :

$$PB_{a5} = c_{52}y_2 + c_{53}y_3 = (c_{52} - c_{53})y_2 + c_{53}$$

Оптимальная стратегия В: $PB_{a1} = PB_{a5}$.

На рисунке 7 найдено решение для оптимальной стратегии игрока В. Таким образом смешанные стратегии игрока В: $(0; 0,1; 0,9; 0; 0)$. Цена игры: 7,2.



Смешанные стратегии игрока В и цена игры. ***

Цена игры: $7 \leq 7.200 \leq 8$

Смешанные стратегии игрока В				
b1	b2	b3	b4	b5
0.00	0.10	0.90	0.00	0.00

Рисунок 7 – Графоаналитический метод поиска смешанных стратегий игрока В

4. Аналитический (матричный) метод

Для игрока А (h – цена игры; y_1, \dots, y_m – смешанные стратегии игрока А) задача сводится к решению следующей СЛАУ матричным методом:

$$\begin{cases} c_{11}y_1 + \dots + c_{m1}y_m = h \\ c_{12}y_1 + \dots + c_{m2}y_m = h \\ \dots \dots \dots \\ c_{1n}y_1 + \dots + c_{mn}y_m = h \\ y_1 + \dots + y_m = 1 \end{cases}$$

$$\begin{cases} c_{11}y_1 + \dots + c_{m1}y_m - h = 0 \\ c_{12}y_1 + \dots + c_{m2}y_m - h = 0 \\ \dots \dots \dots \\ c_{1n}y_1 + \dots + c_{mn}y_m - h = 0 \\ y_1 + \dots + y_m = 1 \end{cases}$$

$$\begin{pmatrix} c_{11} & \dots & c_{m1} & -1 \\ c_{12} & \dots & c_{m2} & -1 \\ \dots & \dots & \dots & \dots \\ c_{1n} & \dots & c_{mn} & -1 \\ 1 & \dots & 1 & 0 \end{pmatrix} \begin{pmatrix} y_1 \\ y_2 \\ \dots \\ y_m \\ h \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ \dots \\ 0 \\ 1 \end{pmatrix}$$

$$\begin{pmatrix} y_1 \\ y_2 \\ \dots \\ y_m \\ h \end{pmatrix} = \begin{pmatrix} c_{11} & \dots & c_{m1} & -1 \\ c_{12} & \dots & c_{m2} & -1 \\ \dots & \dots & \dots & \dots \\ c_{1n} & \dots & c_{mn} & -1 \\ 1 & \dots & 1 & 0 \end{pmatrix}^{-1} \begin{pmatrix} 0 \\ 0 \\ \dots \\ 0 \\ 1 \end{pmatrix}.$$

Для игрока В (g – цена игры ($h=g$); x_1, \dots, x_n – смешанные стратегии игрока А) задача сводится к решению следующей аналогичной СЛАУ (с точностью до транспонирования матрицы игры):

$$\begin{cases} c_{11}x_1 + \dots + c_{1n}x_n = g \\ c_{21}x_1 + \dots + c_{2n}x_n = g \\ \dots \dots \dots \\ c_{m1}x_1 + \dots + c_{mn}x_n = g \\ x_1 + \dots + x_n = 1 \end{cases}$$

...

$$\begin{pmatrix} x_1 \\ x_2 \\ \dots \\ x_n \\ g \end{pmatrix} = \begin{pmatrix} c_{11} & \dots & c_{1n} & -1 \\ c_{21} & \dots & c_{2n} & -1 \\ \dots & \dots & \dots & \dots \\ c_{m1} & \dots & c_{mn} & -1 \\ 1 & \dots & 1 & 0 \end{pmatrix}^{-1} \begin{pmatrix} 0 \\ 0 \\ \dots \\ 0 \\ 1 \end{pmatrix}.$$

На рисунках 8 и 9 представлены решения матричной игры в смешанных стратегиях для игроков А и В соответственно.

```
[17]: analytical_solver = AnalyticalSolver(reduced_game)
first_mixed_strategy, second_mixed_strategy, game_price_value = analytical_solver.player_a_solve()
```

Смешанные стратегии игрока А и цена игры. ●●●

Цена игры: 7 <= 7.200 <= 8

Смешанные стратегии игрока А					
a1	a2	a3	a4	a5	
0.20	0.00	0.00	0.00	0.80	

Рисунок 8 – Аналитический метод поиска смешанных стратегий игрока А

```
[19]: analytical_solver = AnalyticalSolver(reduced_game)
first_mixed_strategy, second_mixed_strategy, game_price_value = analytical_solver.player_b_solve()
```

Смешанные стратегии игрока В и цена игры. ●●●

Цена игры: 7 <= 7.200 <= 8

Смешанные стратегии игрока В					
b1	b2	b3	b4	b5	
0.00	0.10	0.90	0.00	0.00	

Рисунок 9 – Аналитический метод поиска смешанных стратегий игрока В

Полученные цена игры и смешанные стратегии совпали со значениями, полученными предыдущим методом: $(0,2; 0; 0; 0; 0,8)$ для игрока А и $(0; 0,1; 0,9; 0; 0)$ – для игрока В; цена игры – 7,2.

5. Графический метод (задача ЛП)

Задача линейного программирования (далее – ЗЛП) формулируется для матричной игры так, как показано на рисунке 10. Таким образом для игрока А будем решать двойственную задачу (ДЗ ЛП), а для В – прямую (ПЗ ЛП).

Задача линейного программирования с целевой функцией

$$F(\xi) = \sum_{i=1}^n \xi_i \rightarrow \text{MIN}$$

так как необходимо

$$\omega \rightarrow \text{MAX}$$

Пусть $\xi_1 = \frac{q_1}{\omega}, \quad \xi_2 = \frac{q_2}{\omega}, \quad \dots \quad \xi_n = \frac{q_n}{\omega}$

тогда
$$\begin{cases} c_{11}\xi_1 + c_{12}\xi_2 + \dots + c_{1n}\xi_n \geq 1 \\ c_{21}\xi_1 + c_{22}\xi_2 + \dots + c_{2n}\xi_n \geq 1 \\ \dots \\ c_{m1}\xi_1 + c_{m2}\xi_2 + \dots + c_{mn}\xi_n \geq 1 \end{cases}$$

$$\begin{cases} c_{11}q_1 + c_{12}q_2 + \dots + c_{1n}q_n \geq \omega \\ c_{21}q_1 + c_{22}q_2 + \dots + c_{2n}q_n \geq \omega \\ \dots \\ c_{m1}q_1 + c_{m2}q_2 + \dots + c_{mn}q_n \geq \omega \end{cases}$$

где ω – гарантированный MIN выигрыш

где $\xi_j \geq 0, \quad \sum_{i=1}^n \xi_i = \frac{1}{\omega}$

Рисунок 10 – Сведение матричной игры с нулевой суммой к ЗЛП

Для игрока А определим полуплоскости на основе следующих ограничений:

$$\begin{cases} \Phi = y_1 + y_2 \rightarrow \min \\ 9y_2 \geq 1 \\ 8y_1 + 7y_2 \geq 1 \\ y_1, y_2 \geq 0 \end{cases}$$

В системе координат Oy_1y_2 были построены полуплоскости ограничений и найдена точка $(0,028; 0,111)$, через которую проходит линия уровня:

$$y_2 = \Phi - y_1 \text{ при } \Phi = 0,028 + 0,111 \approx 0,14.$$

Чтобы из полученных значений восстановить цену игры, произведём действия, обратные тем, что были выполнены при сведении матричной игры к ЗЛП: $h = 1/\Phi$ – цена игры, $(h \cdot y_1; 0; 0; 0; h \cdot y_2)$ – смешанные стратегии игрока А (см. рисунок 11).

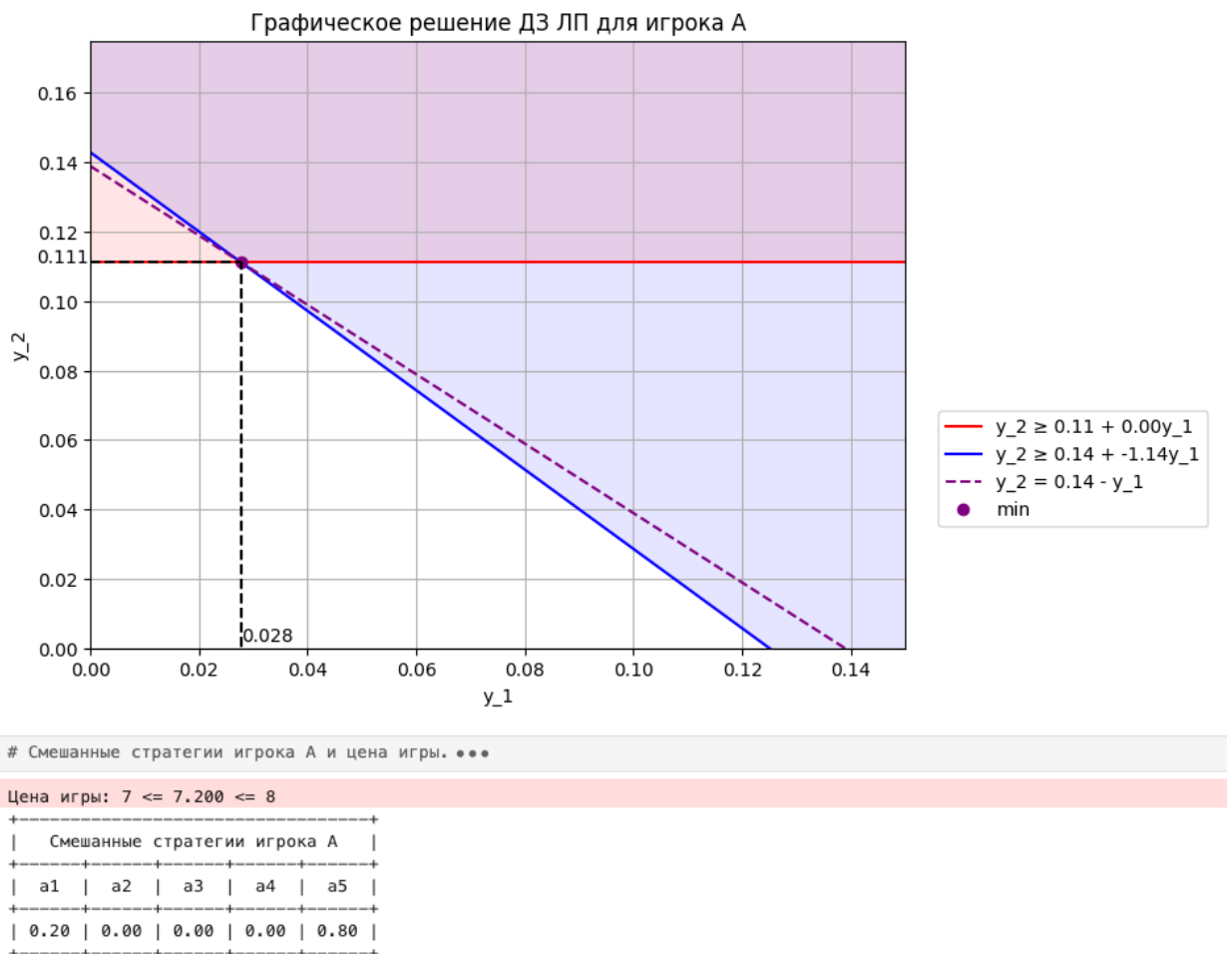


Рисунок 11 – Графическое решение ДЗ ЛП для игрока А

Для игрока В аналогично определим полуплоскости на основе следующих ограничений:

$$F = x_1 + x_2 \rightarrow \max$$

$$\begin{cases} 8x_2 \leq 1 \\ 9x_1 + 7x_2 \leq 1 \\ x_1, x_2 \geq 0 \end{cases}$$

В системе координат Ox_1x_2 были построены полуплоскости ограничений и найдена точка $(0,014; 0,125)$, через которую проходит линия уровня:

$$x_2 = F - x_1 \text{ при } F = 0,014 + 0,125 \approx 0,14.$$

Чтобы из полученных значений восстановить цену игры, произведём действия, обратные тем, что были выполнены при сведении матричной игры к ЗЛП: $g = 1/F$ – цена игры, $(0; g \cdot x_1; g \cdot x_2; 0; 0)$ – смешанные стратегии игрока В (см. рисунок 12).

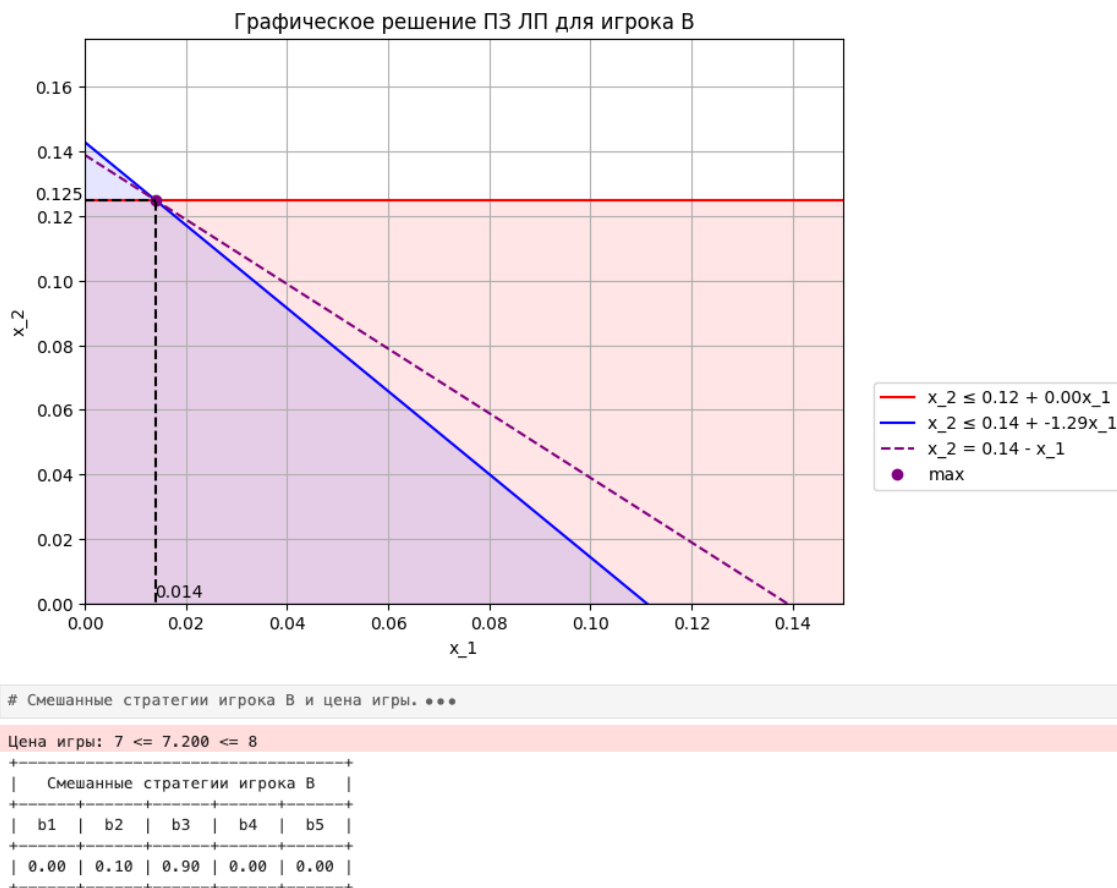


Рисунок 12 – Графическое решение ПЗ ЛП для игрока В

Полученные цена игры и смешанные стратегии совпали со значениями, полученными предыдущими методами: (0,2; 0; 0; 0; 0,8) для игрока А и (0; 0,1; 0,9; 0; 0) – для игрока В; цена игры – 7,2.

6. Симплекс-метод (задача ЛП)

Прямая и двойственная задачи ЛП уже были сформулированы в предыдущем пункте. Осталось применить уже известный симплекс-метод для решения задач ДЗ и ПЗ ЛП для игроков А и В соответственно.

Решение для игрока А представлено на рисунке 13. Из полученных значений по аналогии с предыдущим методом получаем цену игры $h = 1/\Phi$ и смешанные стратегии игрока А ($h \cdot y_1; 0; 0; 0; h \cdot y_2$).

2.4.1. Двойственная задача ЛП для игрока А

```
[28]: player_a_problem = DualProblem(input_path)

F = b^T·x -> min,
A^T·x >= c^T,
x1, x2, ..., xn >= 0
b^T = [1 1]
A^T =
[[0 9]
 [8 7]],
c^T = [1 1].

[29]: player_a_solution = player_a_problem.solve()
player_a_var_values, player_a_target_function_value = player_a_solution
```

Процесс решения:
Поиск опорного решения:
Исходная симплекс-таблица:

	Si0	x1	x2
x3	-1.0000	0.0000	-9.0000
x4	-1.0000	-8.0000	-7.0000
F	0.0000	-1.0000	-1.0000

Разрешающая строка: x3
Разрешающий столбец: x2

	Si0	x1	x3
x2	0.1111	-0.0000	-0.1111
x4	-0.2222	-8.0000	-0.7778
F	0.1111	-1.0000	-0.1111

Разрешающая строка: x4
Разрешающий столбец: x1

	Si0	x4	x3
x2	0.1111	-0.0000	-0.1111
x1	0.0278	-0.1250	0.0972
F	0.1389	-0.1250	-0.0139

Опорное решение найдено!
x4 = x3 = 0,
x2 = 0.111, x1 = 0.028
Целевая функция: F = 0.139
Поиск оптимального решения:
Оптимальное решение найдено!
x4 = x3 = 0,
x2 = 0.111, x1 = 0.028
Целевая функция: F = 0.139

Рисунок 13 – Решение ДЗ ЛП симплекс-методом для игрока А

Решение для игрока В представлено на рисунке 13. Из полученных значений по аналогии с предыдущим методом получаем цену игры $g = 1/F$ и смешанные стратегии игрока В $(0; g \cdot x_1; g \cdot x_2; 0; 0)$.

2.4.2. Прямая задача ЛП для игрока В

```
[31]: player_b_problem = SimplexProblem(input_path)
      player_b_problem

[31]: F = c·x -> max,
      Ax <= b,
      x1,x2, ..., xn >= 0
      c^T = [-1 -1],
      A =
      [[0 8]
       [9 7]],
      b^T = [1 1].

[32]: player_b_solution = player_b_problem.solve()
      player_b_var_values, player_b_target_function_value = player_b_solution
```

Процесс решения:
Поиск опорного решения:
Исходная симплекс-таблица:

	Si0	x1	x2
x3	1.0000	0.0000	8.0000
x4	1.0000	9.0000	7.0000
F	0.0000	1.0000	1.0000

Опорное решение найдено!
 $x1 = x2 = 0$,
 $x3 = 1.000$, $x4 = 1.000$
 Целевая функция: $F = 0.000$
 Поиск оптимального решения:
 Разрешающая строка: x4
 Разрешающий столбец: x1

	Si0	x4	x2
x3	1.0000	-0.0000	8.0000
x1	0.1111	0.1111	0.7778
F	-0.1111	-0.1111	0.2222

Разрешающая строка: x3
 Разрешающий столбец: x2

	Si0	x4	x3
x2	0.1250	-0.0000	0.1250
x1	0.0139	0.1111	-0.0972
F	-0.1389	-0.1111	-0.0278

Оптимальное решение найдено!
 $x4 = x3 = 0$,
 $x2 = 0.125$, $x1 = 0.014$
 Целевая функция: $F = 0.139$

Рисунок 14 – Решение ПЗ ЛП симплекс-методом для игрока В

Полученные цена игры и смешанные стратегии совпали со значениями, полученными предыдущими методами: $(0,2; 0; 0; 0; 0,8)$ для игрока А и $(0; 0,1; 0,9; 0; 0)$ – для игрока В; цена игры – 7,2.

7. Расчёт цены игры исходной матрицы

Вернёмся к исходной матричной игре, вычитанием скаляра 3, который мы добавляли ко всем элементам матрицы для нормализации (см. рисунок 15). Смешанные стратегии игроков останутся неизменными (как вероятности), а итоговая цена игры уменьшится:

$$g - 3 = h - 3 = 7,2 - 3 = 4,2.$$

Задача решена.

3. Расчёт цены игры исходной матрицы

Вспомним изначальную заданную матрицу игры, до нормализации и найдём её цену игры и смешанные стратегии игроков.

```
[4]: # Вычитаем ранее добавленное слагаемое нормализации.  
game_matrix.matrix -= normalizer  
game_matrix
```

```
[4]: +-----+  
|               Таблица стратегий (игрока A)               |  
+-----+-----+-----+-----+-----+-----+-----+  
| Стратегии | b1 | b2 | b3 | b4 | b5 | MIN выигрыш A |  
+-----+-----+-----+-----+-----+-----+-----+  
| a1        | 4  | -3 | 5  | 6  | 4  | -3             |  
| a2        | 6  | 5  | -3 | 4  | 7  | -3             |  
| a3        | 6  | 5  | -3 | -3 | 5  | -3             |  
| a4        | -3 | -3 | 4  | 4  | 4  | -3             |  
| a5        | 7  | 6  | 4  | 5  | 6  | 4              |  
| MAX проигрыш B | 7 | 6 | 5 | 6 | 7 |                |  
+-----+-----+-----+-----+-----+-----+-----+
```

```
[5]: print(f"Нижняя цена игры: {game_matrix.lowest_game_price[1]}\n"  
      f"Верхняя цена игры: {game_matrix.highest_game_price[1]}")
```

Нижняя цена игры: 4
Верхняя цена игры: 5

```
[6]: original_game_price: float = game_price_value - normalizer  
print(f"Цена игры исходной матрицы: {original_game_price:.2f}")
```

Цена игры исходной матрицы: 4.20

Рисунок 15 – Нахождение цены игры исходной матричной игры

Вывод

В данном расчётно-графическом задании была исследована антагонистическая игра двух лиц с нулевой суммой. Для этого матрица 5×5 игроков А и В была нормализована и сведена к игре меньшей размерности (2×2) методами поглощения доминируемых стратегий и удалением NBR-стратегий (полученные матрицы совпали).

Далее для полученной матричной игры 2×2 были реализованы методы для нахождения смешанных стратегий обоих игроков. Примечательно, что графоаналитический и графический (ЛП) методы имели хорошую наглядность, что в частном случае будет менее предпочтительно в пространствах высших размерностей. Универсальными показали себя аналитический (матричный) и симплекс-методы.

В конечном итоге была найдена цена игры исходной матричной игры. Частоты выбора стратегий игроками А и В не претерпели изменений, что согласуется с теорией. В итоге была получена цена игры 4,2 и смешанные стратегии игроков:

$(0,2; 0; 0; 0; 0,8)$ для игрока А;

$(0; 0,1; 0,9; 0; 0)$ – для игрока В.

ПРИЛОЖЕНИЕ А

Листинг интерактивного блокнота rgz.ipynb

```
1. #!/usr/bin/env python
2. # coding: utf-8
3.
4. # # Расчетно-графическое домашнее задание
5. #
6. # **Выполнил: Александров А. Н., ИУ8-104**
7. #
8. # **Вариант: 1**
9. #
10. # ## Задание
11. # Задана платежная матрица прямоугольной игры с нулевой суммой.
12. #
13. #
14. # | 4 | -3 | 5 | 6 | 4 |
15. # | :--: | :--: | :--: | :--: | :--: |
16. # | 6 | 5 | -3 | 4 | 7 |
17. # | 6 | 5 | -3 | -3 | 5 |
18. # | -3 | -3 | 4 | 4 | 4 |
19. # | 7 | 6 | 4 | 5 | 6 |
20. #
21. #
22. # 1. Нормализовать матрицу (привести к матрице с неотрицательными элементами) и свести
    исходную игру к матричной игре 2×2 следующими способами:
23. #     - [x] поглощением доминируемых стратегий;
24. #     - [x] удалением NBR-стратегий (Never Best Response).
25. # 2. Найти смешанные стратегии игроков следующими методами:
26. #     - [x] графоаналитическим;
27. #     - [x] аналитическим (матричным);
28. #     - [x] графически (задача ЛП);
29. #     - [x] симплекс-методом (задача ЛП).
30. # 3. Рассчитать цену игры для исходной матрицы.
31.
32. # In[1]:
33.
34.
35. import json
36. import logging
37. from pathlib import Path
38.
39. import matplotlib.pyplot as plt
40. import numpy as np
41.
42. from game_theory.utils.matrix_games.analytical import AnalyticalSolver
43. from game_theory.utils.matrix_games.game_matrix import GameMatrix
44. from game_theory.utils.matrix_games.mixed_strategies import (
45.     check_resulted_game_price,
46.     get_resulted_mixed_strategies,
47. )
```

```

48. from game_theory.utils.simplex.dual_problem import DualProblem
49. from game_theory.utils.simplex.simplex_problem import SimplexProblem
50.
51. logging.basicConfig(level=logging.INFO, format='%(message)s')
52.
53.
54. # In[2]:
55.
56.
57. # Входная матрица прямоугольной игры с нулевой суммой.
58. matrix = np.array(
59.     [
60.         [4, -3, 5, 6, 4],
61.         [6, 5, -3, 4, 7],
62.         [6, 5, -3, -3, 5],
63.         [-3, -3, 4, 4, 4],
64.         [7, 6, 4, 5, 6],
65.     ],
66.     dtype=int,
67.)
68.
69. game_matrix = GameMatrix(matrix)
70. game_matrix
71.
72.
73. # In[3]:
74.
75.
76. print(f"Нижняя цена игры: {game_matrix.lowest_game_price[1]}\n"
77.       f"Верхняя цена игры: {game_matrix.highest_game_price[1]}")
78.
79.
80. # ## 1. Нормализация матрицы. Уменьшение размерности исходной матричной игры
81.
82. # In[4]:
83.
84.
85. normalizer: int = game_matrix.normalize_matrix()
86. game_matrix
87.
88.
89. # In[5]:
90.
91.
92. print(f"Нижняя цена игры: {game_matrix.lowest_game_price[1]}\n"
93.       f"Верхняя цена игры: {game_matrix.highest_game_price[1]}")
94.
95.
96. # ### 1.1. Поглощение доминируемых стратегий
97. # <u>Опр. **Доминирующая (поглощающая) строка**</u> содержит элементы  $\geq$  элементам
    другой строки (поглощаемой).
98. #

```

```

99. # <u>Опр. **Доминирующий (поглощающая) столбец**</u> содержит элементы  $\leq$  элементам
    другого столбца (поглощаемого).
100.
101. # In[6]:
102.
103.
104. dominant_reduced_game: GameMatrix =
    game_matrix.reduce_dimension(method='dominant_absorption')
105. dominant_reduced_game
106.
107.
108. # In[7]:
109.
110.
111. print(f"Нижняя цена игры: {dominant_reduced_game.lowest_game_price[1]}\n"
112.       f"Верхняя цена игры: {dominant_reduced_game.highest_game_price[1]}")
113.
114.
115. # ### 1.2. Удаление NBR-стратегий
116. #
117. # <u>Опр. **NBR-строка**</u> - строка, которая объективно не будет разыгрываться
    игроком А для всех наперёд фиксированных стратегий В.
118. #
119. # <u>Опр. **NBR-столбец**</u> - столбец, который объективно не будет разыгрываться
    игроком В для всех наперёд фиксированных стратегий А.
120.
121. # In[8]:
122.
123.
124. # Вычеркиваем столбцы и строки, которые мы точно не выберем при фиксированной
    стратегии.
125. nbr_reduced_game: GameMatrix = game_matrix.reduce_dimension(method='nbr_drop')
126. nbr_reduced_game
127.
128.
129. # In[9]:
130.
131.
132. print(f"Нижняя цена игры: {nbr_reduced_game.lowest_game_price[1]}\n"
133.       f"Верхняя цена игры: {nbr_reduced_game.highest_game_price[1]}")
134.
135.
136. # In[10]:
137.
138.
139. assert dominant_reduced_game == nbr_reduced_game
140. reduced_game: GameMatrix = nbr_reduced_game
141.
142.
143. # ## 2. Нахождение смешанных стратегий и цены игры
144.
145. # ### 2.1. Графоаналитический метод

```

```

146.
147. # #### 2.1.1. Для игрока A
148. # Пусть
149. # -  $x_1$  - вероятность выбора игроком A стратегии  $a_1$ .
150. # -  $x_5 = 1 - x_1$  - вероятность выбора игроком A стратегии  $a_5$ .
151. #
152. # Ожидаемый выигрыш A при реализации стратегии  $b_2$ :
153. #
154. #  $PA_{b2} = c_{12}x_1 + c_{52}x_5 = (c_{12} - c_{52})x_5 + c_{52}$ 
155. #
156. # Ожидаемый выигрыш A при реализации стратегии  $b_3$ :
157. #
158. #  $PA_{b3} = c_{13}x_1 + c_{53}x_5 = (c_{13} - c_{53})x_5 + c_{53}$ 
159. #
160. # Оптимальная стратегия A:  $PA_{b2} = PA_{b3}$ 
161.
162. # In[11]:
163.
164.
165. assert reduced_game.matrix.shape == (2, 2)
166.
167. (a, b), (c, d) = reduced_game.matrix.tolist()
168. PA_b_first = lambda x: (a - c) * x + c
169. PA_b_second = lambda x: (b - d) * x + d
170.
171. # Находим точку пересечения решая систему  $Ax = b$ .
172. PA_A = np.array([
173.     [-(a - c), 1],
174.     [-(b - d), 1],
175. ])
176. PA_b = np.array([c, d])
177. (x_intersect, y_intersect) = np.linalg.solve(PA_A, PA_b)
178.
179.
180. # In[12]:
181.
182.
183. # Отрисовка графиков пересечения.
184. X = np.linspace(0, 1, 50)
185. PA_b_first_y = PA_b_first(X)
186. PA_b_second_y = PA_b_second(X)
187. max_PA_b_y = np.max(np.concatenate((PA_b_first_y, PA_b_second_y)))
188.
189. plt.figure(figsize=(8, 6))
190. plt.title("Оптимальная стратегия игрока A")
191.
192. PA_b_first_label, PA_b_second_label = reduced_game.player_b_strategy_labels
193. plt.plot(X, PA_b_first_y, label=f"PA_{PA_b_first_label} = {a - c}x + {c}",
194.          color="red")
194. plt.plot(X, PA_b_second_y, label=f"PA_{PA_b_second_label} = {b - d}x + {d}",
195.          color="blue")
195.

```

```

196. # Точка пересечения.
197. plt.plot(x_intersect, y_intersect, "o", color="black", label="max-min")
198. # Проекция точки пересечения на оси.
199. plt.vlines(x_intersect, min(PA_b_first_y), y_intersect, color="black",
    linestyle='dashed')
200. plt.hlines(y_intersect, min(X), x_intersect, color="black", linestyle='dashed')
201. # Ограничение [0, 1] - вероятность.
202. plt.xlim(0, 1)
203. # Ограничение на нормализованные элементы матрицы.
204. plt.ylim(0, max_PA_b_y)
205. # Подписи осей.
206. plt.xlabel("Вероятность выбора стратегии игроком А")
207. plt.ylabel("Ожидаемый выигрыш игрока А")
208. # Сегменты под графиками.
209. plt.fill_between(X, PA_b_first_y, color='red', alpha=0.1)
210. plt.fill_between(X, PA_b_second_y, color='blue', alpha=0.1)
211. # Отображение значений координат точки пересечения на осях
212. plt.text(x_intersect, 0.05, f'{x_intersect:.2f}')
213. plt.text(-0.05, y_intersect, f'{y_intersect:.2f}')
214.
215. plt.legend(loc=(1.04, 0.2))
216. plt.grid(True)
217.
218.
219. # In[13]:
220.
221.
222. # Смешанные стратегии игрока А и цена игры.
223. assert check_resulted_game_price(reduced_game, y_intersect)
224. mixed_strategies = get_resulted_mixed_strategies(
225.     player_labels=game_matrix.player_a_strategy_labels,
226.     labels_to_probability=dict(zip(
227.         reduced_game.player_a_strategy_labels,
228.         [x_intersect, 1 - x_intersect],
229.     )),
230.     player_name="А",
231. )
232. print(mixed_strategies)
233.
234.
235. # #### 2.1.2. Для игрока В
236. # Пусть
237. # -  $y_2$  - вероятность выбора игроком В стратегии  $b_2$ .
238. # -  $y_3 = 1 - y_2$  - вероятность выбора игроком В стратегии  $b_3$ .
239. #
240. # Ожидаемый проигрыш В при реализации стратегии  $a_1$ :
241. #
242. #  $PB_{a1} = c_{12}y_2 + c_{13}y_3 = (c_{12} - c_{13})y_2 + c_{13}$ 
243. #
244. # Ожидаемый проигрыш В при реализации стратегии  $a_5$ :
245. #
246. #  $PB_{a5} = c_{52}y_2 + c_{53}y_3 = (c_{52} - c_{53})y_2 + c_{53}$ 

```

```

247. #
248. # Оптимальная стратегия В:  $PB_{a1} = PB_{a5}$ 
249.
250. # In[14]:
251.
252.
253. assert reduced_game.matrix.shape == (2, 2)
254.
255. (a, b), (c, d) = reduced_game.matrix.tolist()
256. PB_a_first = lambda x: (a - b) * x + b
257. PB_a_second = lambda x: (c - d) * x + d
258.
259. # Находим точку пересечения решая систему  $Ax = b$ .
260. PB_A = np.array([
261.     [-(a - b), 1],
262.     [-(c - d), 1],
263. ])
264. PB_b = np.array([b, d])
265. (x_intersect, y_intersect) = np.linalg.solve(PB_A, PB_b)
266.
267.
268. # In[15]:
269.
270.
271. # Отрисовка графиков пересечения.
272. X = np.linspace(0, 1, 50)
273. PB_a_first_y = PB_a_first(X)
274. PB_a_second_y = PB_a_second(X)
275. max_PB_a_y = np.max(np.concatenate((PB_a_first_y, PB_a_second_y)))
276.
277. plt.figure(figsize=(8, 6))
278. plt.title("Оптимальная стратегия игрока В")
279.
280. PB_a_first_label, PB_a_second_label = reduced_game.player_a_strategy_labels
281. plt.plot(X, PB_a_first_y, label=f"PB_{PB_a_first_label} = {a - c}x + {b}",
282.     color="red")
283.
284. plt.plot(X, PB_a_second_y, label=f"PB_{PB_a_second_label} = {c - d}x + {d}",
285.     color="blue")
286.
287. # Точка пересечения.
288. plt.plot(x_intersect, y_intersect, "o", color="black", label="min-max")
289.
290. # Проекция точки пересечения на оси.
291. plt.vlines(x_intersect, min(PB_a_first_y, y_intersect), y_intersect, color="black",
292.     linestyle='dashed')
293.
294. plt.hlines(y_intersect, min(X), x_intersect, color="black", linestyle='dashed')
295.
296. # Ограничение [0, 1] - вероятность.
297. plt.xlim(0, 1)
298.
299. # Ограничение на нормализованные элементы матрицы.
300. plt.ylim(0, max_PB_a_y)
301.
302. # Подписи осей.
303. plt.xlabel("Вероятность выбора стратегии игроком В")
304.
305. plt.ylabel("Ожидаемый проигрыш игрока В")

```

```

296. # Сегменты над графиками.
297. plt.fill_between(X, PB_a_first_y, max_PB_a_y, color='red', alpha=0.1)
298. plt.fill_between(X, PB_a_second_y, max_PB_a_y, color='blue', alpha=0.1)
299. # Отображение значений координат точки пересечения на осях.
300. plt.text(x_intersect, 0.05, f'{x_intersect:.2f}')
301. plt.text(-0.05, y_intersect, f'{y_intersect:.2f}')
302.
303. plt.legend(loc=(1.04, 0.2))
304. plt.grid(True)
305.
306.
307. # In[16]:
308.
309.
310. # Смешанные стратегии игрока B и цена игры.
311. assert check_resulted_game_price(reduced_game, y_intersect)
312. mixed_strategies = get_resulted_mixed_strategies(
313.     player_labels=game_matrix.player_b_strategy_labels,
314.     labels_to_probability=dict(zip(
315.         reduced_game.player_b_strategy_labels,
316.         [x_intersect, 1 - x_intersect],
317.     )),
318.     player_name="B",
319. )
320. print(mixed_strategies)
321.
322.
323. # ### 2.2. Аналитический (матричный) метод
324.
325. # #### 2.2.1. Обратная матрица для игрока A
326. #
327. # Для игрока $A$ ($h$ - цена игры; $y_1, \dots, y_m$ - смешанные стратегии игрока $A$):
328.
329. # ![analytical_A](./img/analytical_A.png)
330.
331. # In[17]:
332.
333.
334. analytical_solver = AnalyticalSolver(reduced_game)
335. first_mixed_strategy, second_mixed_strategy, game_price_value =
    analytical_solver.player_a_solve()
336.
337.
338. # In[18]:
339.
340.
341. # Смешанные стратегии игрока A и цена игры.
342. assert check_resulted_game_price(
343.     game_matrix=reduced_game,
344.     game_price_value=game_price_value,
345. )
346.

```



```

347. mixed_strategies = get_resulted_mixed_strategies(
348.     player_labels=game_matrix.player_a_strategy_labels,
349.     labels_to_probability=dict(zip(
350.         reduced_game.player_a_strategy_labels,
351.         (first_mixed_strategy, second_mixed_strategy),
352.     )),
353.     player_name="A",
354. )
355. print(mixed_strategies)
356.
357.
358. # #### 2.2.1. Прямая матрица для игрока B
359. # Для игрока $B$ ($g$ - цена игры; $x_1, \dots, y_n$ - смешанные стратегии игрока $B$):
360.
361. # ![analytical_B](./img/analytical_B.png)
362.
363. # In[19]:
364.
365.
366. analytical_solver = AnalyticalSolver(reduced_game)
367. first_mixed_strategy, second_mixed_strategy, game_price_value =
    analytical_solver.player_b_solve()
368.
369.
370. # In[20]:
371.
372.
373. # Смешанные стратегии игрока B и цена игры.
374. assert check_resulted_game_price(
375.     game_matrix=reduced_game,
376.     game_price_value=game_price_value,
377. )
378.
379. mixed_strategies = get_resulted_mixed_strategies(
380.     player_labels=game_matrix.player_b_strategy_labels,
381.     labels_to_probability=dict(zip(
382.         reduced_game.player_b_strategy_labels,
383.         (first_mixed_strategy, second_mixed_strategy),
384.     )),
385.     player_name="B",
386. )
387. print(mixed_strategies)
388.
389.
390. # #### 2.3. Графический метод (задача ЛП)
391. #
392. # <div style="text-align:left;">
393. #     
394. # </div>
395. # <div style="text-align:left;">
396. #     

```

```

397. # 
398. # </div>
399.
400. # In[21]:
401.
402.
403. # Подготовка входных данных ЗЛП.
404. n_rows, n_cols = reduced_game.matrix.shape
405. input_data = {
406.     #  $F = x_1 + x_2$ 
407.     "obj_func_coeffs": [1] * n_cols,
408.     # A - матрица сериализуется в массив массивов JSON.
409.     "constraint_system_lhs": reduced_game.matrix.tolist(),
410.     # b - вектор-столбец ограничений.
411.     "constraint_system_rhs": [1] * n_rows,
412.     # Экстремум, направление оптимизации функции.
413.     "func_direction": "max"
414. }
415.
416. input_path = Path('input_LPP.json')
417. _ = input_path.write_text(json.dumps(input_data, indent=2))
418.
419.
420. # #### 2.3.1. Двойственная ЗЛП для игрока А
421. # ![LP_problem_A](./img/LP_problem_A.png)
422.
423. # In[22]:
424.
425.
426. assert reduced_game.matrix.shape == (2, 2)
427.
428. (a, b), (c, d) = reduced_game.matrix.tolist()
429. # Прямая для 1-го ограничения.
430. first_constraint = lambda y_1: (1 / c) - (a / c) * y_1
431. # Прямая для 2-го ограничения.
432. second_constraint = lambda y_1: (1 / d) - (b / d) * y_1
433. # Точка пересечения.
434. (y_1_intersect, y_2_intersect) = np.linalg.solve(
435.     np.array([
436.         [a, c],
437.         [b, d],
438.     ]),
439.     np.array([1, 1]),
440. )
441. # Выражаем целевую функцию через  $y_2$ :  $y_2 = F - y_1$ , где  $F \rightarrow \min$ .
442. player_a_target_function_value = y_1_intersect + y_2_intersect
443. F_func = lambda y_1: (y_1_intersect + y_2_intersect) - y_1
444.
445.
446. # In[23]:
447.
448.

```

```

449. # Отрисовка графиков пересечения.
450. x = np.linspace(0, 1, 100)
451. first_constraint_y2 = first_constraint(X)
452. second_constraint_y2 = second_constraint(X)
453. F_y2 = F_func(X)
454. max_y2 = np.max(np.concatenate((first_constraint_y2, second_constraint_y2))) + 0.15
455.
456. plt.figure(figsize=(8, 6))
457. plt.title("Графическое решение ДЗ ЛП для игрока А")
458.
459. plt.plot(X, first_constraint_y2, label=f" $y_2 \geq \{1 / c:.2f\} + \{-a / c:.2f\}y_1$ ",
           color="red")
460. plt.plot(X, second_constraint_y2, label=f" $y_2 \geq \{1 / d:.2f\} + \{-b / d:.2f\}y_1$ ",
           color="blue")
461. plt.plot(X, F_y2, "--", label=f" $y_2 = \{y_1\_intersect + y_2\_intersect:.2f\} - y_1$ ",
           color="purple")
462.
463. # Точка пересечения.
464. plt.plot(y_1_intersect, y_2_intersect, "o", color="purple", label="min")
465. # Проекция точки пересечения на оси.
466. plt.vlines(y_1_intersect, min(second_constraint_y2), y_2_intersect, color="black",
           linestyle='dashed')
467. plt.hlines(y_2_intersect, min(X), y_1_intersect, color="black", linestyle='dashed')
468. # Ограничения на неотрицательные решения.
469. plt.xlim(0, 0.15)
470. plt.ylim(0, 0.175)
471. # Подписи осей.
472. plt.xlabel("y_1")
473. plt.ylabel("y_2")
474. # Сегменты над графиками.
475. plt.fill_between(X, first_constraint_y2, max_y2, color='red', alpha=0.1)
476. plt.fill_between(X, second_constraint_y2, max_y2, color='blue', alpha=0.1)
477.
478. # Отображение значений координат точки пересечения на осях.
479. plt.text(y_1_intersect, 0.002, f'{y_1_intersect:.3f}')
480. plt.text(-0.01, y_2_intersect, f'{y_2_intersect:.3f}')
481.
482. plt.legend(loc=(1.04, 0.2))
483. plt.grid(True)
484.
485.
486. # In[24]:
487.
488.
489. # Смешанные стратегии игрока А и цена игры.
490. game_price_value = 1 / player_a_target_function_value
491. assert check_resulted_game_price(
492.     game_matrix=reduced_game,
493.     game_price_value=game_price_value,
494. )
495.
496. player_a_mixed_strategies = [

```

```

497.     var_value * game_price_value
498.     for var_value in [y_1_intersect, y_2_intersect]
499. ]
500. player_a_mixed_strategies =
    player_a_mixed_strategies[:len(reduced_game.player_a_strategy_labels)]
501. mixed_strategies = get_resulted_mixed_strategies(
502.     player_labels=game_matrix.player_a_strategy_labels,
503.     labels_to_probability=dict(zip(
504.         reduced_game.player_a_strategy_labels,
505.         player_a_mixed_strategies,
506.     )),
507.     player_name="A",
508. )
509. print(mixed_strategies)
510.
511.
512. # #### 2.3.2. Прямая ЗЛП для игрока B
513. # ![LP_problem_B](./img/LP_problem_B.png)
514.
515. # In[25]:
516.
517.
518. assert reduced_game.matrix.shape == (2, 2)
519.
520. (a, b), (c, d) = reduced_game.matrix.tolist()
521. # Прямая для 1-го ограничения.
522. first_constraint = lambda x_1: (1 / b) - (a / b) * x_1
523. # Прямая для 2-го ограничения.
524. second_constraint = lambda x_1: (1 / d) - (c / d) * x_1
525. # Точка пересечения.
526. (x_1_intersect, x_2_intersect) = np.linalg.solve(
527.     np.array([
528.         [a, b],
529.         [c, d],
530.     ]),
531.     np.array([1, 1]),
532. )
533. # Выражаем целевую функцию через x_2: x_2 = F - x_1, где F -> max.
534. player_b_target_function_value = x_1_intersect + x_2_intersect
535. F_func = lambda x_1: (x_1_intersect + x_2_intersect) - x_1
536.
537.
538. # In[26]:
539.
540.
541. # Отрисовка графиков пересечения.
542. x = np.linspace(0, 1, 100)
543. first_constraint_x2 = first_constraint(x)
544. second_constraint_x2 = second_constraint(x)
545. F_x2 = F_func(x)
546.
547. plt.figure(figsize=(8, 6))

```

```

548. plt.title("Графическое решение ПЗ ЛП для игрока В")
549.
550. plt.plot(X, first_constraint_x2, label=f"x_2 ≤ {1 / b:.2f} + {- a / b:.2f}x_1",
            color="red")
551. plt.plot(X, second_constraint_x2, label=f"x_2 ≤ {1 / d:.2f} + {- c / d:.2f}x_1",
            color="blue")
552. plt.plot(X, F_x2, "--", label=f"x_2 = {x_1_intersect + x_2_intersect:.2f} - x_1",
            color="purple")
553.
554. # Точка пересечения.
555. plt.plot(x_1_intersect, x_2_intersect, "o", color="purple", label="max")
556. # Проекция точки пересечения на оси.
557. plt.vlines(x_1_intersect, min(second_constraint_x2), x_2_intersect, color="black",
            linestyle='dashed')
558. plt.hlines(x_2_intersect, min(X), x_1_intersect, color="black", linestyle='dashed')
559. # Ограничения на неотрицательные решения.
560. plt.xlim(0, 0.15)
561. plt.ylim(0, 0.175)
562. # Подписи осей.
563. plt.xlabel("x_1")
564. plt.ylabel("x_2")
565. # Сегменты над графиками.
566. plt.fill_between(X, first_constraint_x2, color='red', alpha=0.1)
567. plt.fill_between(X, second_constraint_x2, color='blue', alpha=0.1)
568.
569. # Отображение значений координат точки пересечения на осях
570. plt.text(x_1_intersect, 0.002, f'{x_1_intersect:.3f}')
571. plt.text(-0.01, x_2_intersect, f'{x_2_intersect:.3f}')
572.
573. plt.legend(loc=(1.04, 0.2))
574. plt.grid(True)
575.
576.
577. # In[27]:
578.
579.
580. # Смешанные стратегии игрока В и цена игры.
581. game_price_value = 1 / player_b_target_function_value
582. assert check_resulted_game_price(
583.     game_matrix=reduced_game,
584.     game_price_value=game_price_value,
585. )
586.
587. player_b_mixed_strategies = [var_value * game_price_value for var_value in
    [x_1_intersect, x_2_intersect]]
588. player_b_mixed_strategies =
    player_b_mixed_strategies[:len(reduced_game.player_b_strategy_labels)]
589. mixed_strategies = get_resulted_mixed_strategies(
590.     player_labels=game_matrix.player_b_strategy_labels,
591.     labels_to_probability=dict(zip(
592.         reduced_game.player_b_strategy_labels,
593.         player_b_mixed_strategies,

```

```

594.    )),
595.     player_name="B",
596. )
597. print(mixed_strategies)
598.
599.
600. # ### 2.4. Симплекс-метод (задача ЛП)
601. #
602. # <div style="text-align:left;">
603. #     
604. # </div>
605. #
606.
607. # #### 2.4.1. Двойственная задача ЛП для игрока А
608.
609. # In[28]:
610.
611.
612. player_a_problem = DualProblem(input_path)
613.
614.
615. # In[29]:
616.
617.
618. player_a_solution = player_a_problem.solve()
619. player_a_var_values, player_a_target_function_value = player_a_solution
620.
621.
622. # In[30]:
623.
624.
625. # Смешанные стратегии игрока А и цена игры.
626. game_price_value = 1 / player_a_target_function_value
627. assert check_resulted_game_price(
628.     game_matrix=reduced_game,
629.     game_price_value=game_price_value,
630. )
631.
632. player_a_mixed_strategies = [var_value * game_price_value for var_value in
633.     player_a_var_values]
634. player_a_mixed_strategies =
635.     player_a_mixed_strategies[:len(reduced_game.player_a_strategy_labels)]
636. mixed_strategies = get_resulted_mixed_strategies(
637.     player_labels=game_matrix.player_a_strategy_labels,
638.     labels_to_probability=dict(zip(
639.         reduced_game.player_a_strategy_labels,
640.         player_a_mixed_strategies,
641.     )),
642.     player_name="A",
643. )
644. print(mixed_strategies)
645.

```

```

644.
645. # ##### 2.4.2. Прямая задача ЛП для игрока В
646.
647. # In[31]:
648.
649.
650. player_b_problem = SimplexProblem(input_path)
651. player_b_problem
652.
653.
654. # In[32]:
655.
656.
657. player_b_solution = player_b_problem.solve()
658. player_b_var_values, player_b_target_function_value = player_b_solution
659.
660.
661. # In[33]:
662.
663.
664. # Смешанные стратегии игрока В и цена игры.
665. game_price_value = 1 / player_b_target_function_value
666. assert check_resulted_game_price(
667.     game_matrix=reduced_game,
668.     game_price_value=game_price_value,
669. )
670.
671. player_b_mixed_strategies = [var_value * game_price_value for var_value in
    player_b_var_values]
672. player_b_mixed_strategies =
    player_b_mixed_strategies[:len(reduced_game.player_b_strategy_labels)]
673. mixed_strategies = get_resulted_mixed_strategies(
674.     player_labels=game_matrix.player_b_strategy_labels,
675.     labels_to_probability=dict(zip(
676.         reduced_game.player_b_strategy_labels,
677.         player_b_mixed_strategies,
678.     )),
679.     player_name="B",
680. )
681. print(mixed_strategies)
682.
683.
684. # ##### 3. Расчёт цены игры исходной матрицы
685. # Вспомним изначальную заданную матрицу игры, до нормализации и найдём её цену игры и
    смешанные стратегии игроков.
686.
687. # In[34]:
688.
689.
690. # Вычитаем ранее добавленное слагаемое нормализации.
691. game_matrix.matrix -= normalizer
692. game_matrix

```

```
693.  
694.  
695. # In[35]:  
696.  
697.  
698. print(f"Нижняя цена игры: {game_matrix.lowest_game_price[1]}\n"  
699.       f"Верхняя цена игры: {game_matrix.highest_game_price[1]}")  
700.  
701.  
702. # In[36]:  
703.  
704.  
705. original_game_price: float = game_price_value - normalizer  
706. print(f"Цена игры исходной матрицы: {original_game_price:.2f}")
```


ПРИЛОЖЕНИЕ Б

Ссылка на репозиторий с исходным кодом задачи

https://github.com/aaaaaaaalesha/10-game_theory