**Министерство образования Российской Федерации**

# МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ им. Н.Э. БАУМАНА

Факультет: Информатика и системы управления Кафедра: Информационная безопасность (ИУ8)

# ТЕОРИЯ ИГР И ИССЛЕДОВАНИЕ ОПЕРАЦИЙ

**Лабораторная работа №3 на тему:**

«Неантагонистические игры. Критерий выбора оптимальных стратегий в бескоалиционных играх нескольких игроков»

Вариант 9

|  |  |
| --- | --- |
| Преподаватель: | Коннова Н.С. |
| Студент: | Киселев В.А. |
| Группа: | ИУ8-104 |

Москва, 2025

# Цель работы

Изучить критерий выбора стратегий в неантагонистической бескоалиционной игре двух игроков на основе равновесия Нэша и оптимальности по Парето. Проверить данные критерии на примере игр

«Семейный спор», «Перекресток», «Дилемма заключенного». Исследовать свойства оптимальных решений неантагонистических бескоалиционных игр на примере биматричных (2 × 2) игр.

# Задание

1. Сгенерировать случайную биматричную игру (10 × 10). Найти ситуации, равновесные по Нэшу и оптимальные по Парето, а также пересечение множеств этих ситуаций. Выполнить проверку реализованных алгоритмов на примере игр «Семейный спор», «Перекресток», «Дилемма заключенного».
2. Для заданной биматричной игры

Γ(𝐴, 𝐵) = ((5, 1) (10, 4)),

(8, 6) (6, 9)

пользуясь теоремой о свойствах оптимальных решений, найти ситуации, равновесные по Нэшу, для искомой игры и для ее смешанного расширения.

# Ход работы

## Задание 1

Биматрица случайной игры (10 × 10) приведена далее на рисунке 1: первое значение в ячейке соответствуют стратегиям первого игрока, второе значение – второго игрока.ЗАДАНИЕ 1: Случайная биматричная игра 10×10

============================================================

Парето-оптимальные ситуации (индексы строк и столбцов): [(0, 5), (1, 3), (3, 6), (5, 4), (5, 5), (8, 6), (9, 2)]

Равновесия Нэша в чистых стратегиях: []

Матрица выигрышей с выделением оптимальных стратегий:

(63, -72) (-94, 89) (-30, -38) (-43, -65) (88, -74) < (73, 89) > (39, -78) (51, 8) (-92, -93) (-77, -45)

(-41, 29) (54, -94) (43, -50) < (83, 66) > (79, 39) (7, -44) (14, 50) (-29, -99) (94, -60) (78, 8)

(-13, -29) (-61, -45) (95, -14) (-74, -77) (-3, -76) (-9, -12) (54, -33) (-89, 86) (17, 37) (-69, -4)

(-80, 41) (-25, 60) (58, -8) (47, -51) (80, -83) (-89, 69) <(-42, 97) > (-26, -80) (-41, -75) (-3, -29)

(16, 62) (-7, -59) (-6, -10) (-47, 71) (-32, 79) (74, 65) (-82, 55) (62, -57) (36, 86) (-38, -59)

(18, -3) (-31, 63) (76, 42) (-44, 75) <(-17, 96) > <(98, -86) > (-42, -92) (-20, 2) (-32, -84) (-46, 45)

(83, -20) (-46, 67) (27, 1) (64, 17) (-64, -33) (-65, -37) (90, 43) (37, -33) (91, 49) (9, 49)

(2, -8) (-44, -65) (30, 26) (-77, 93) (-88, -72) (-61, 60) (-60, 74) (8, 52) (-84, -2) (-3, 52)

(19, 35) (-36, 41) (-98, 74) (84, -71) (74, 37) (92, -32) < (96, 64) > (-13, -72) (-25, 11) (-60, 16)

(-100, 84) (84, -33) < (28, 95) > (-55, 29) (-73, 60) (-24, 63) (29, 55) (-50, -61) (-5, 95) (-59, 38)

Легенда:

< > - Парето-оптимальная ситуация

{ } - Равновесие Нэша

[ ] - Парето-оптимальная ситуация и равновесие Нэша

Анализ игры 'Случайная игра 10×10':

Парето-оптимальные ситуации:

Позиция (0, 5): (73, 89)

Позиция (1, 3): (83, 66)

Позиция (3, 6): (-42, 97)

Позиция (5, 4): (-17, 96)

Позиция (5, 5): (98, -86)

Позиция (8, 6): (96, 64)

Позиция (9, 2): (28, 95)

Равновесия Нэша:

Равновесий Нэша не найдено

Пересечение множеств (Парето-оптимальные и равновесия Нэша):

Пересечений нет

============================================================

ЗАДАНИЕ 2: Анализ игры для варианта 9

============================================================

Матрица выигрышей для варианта 9:

Матрица выигрышей с выделением оптимальных стратегий:

(5, 1) [(10, 4)]

<(8, 6) > <(6, 9) >

Легенда:

< > - Парето-оптимальная ситуация

{ } - Равновесие Нэша

[ ] - Парето-оптимальная ситуация и равновесие Нэша

Результаты анализа:

Равновесия Нэша в чистых стратегиях: [(0, 1)]

Смешанного равновесия не найдено

Дополнительный анализ:

В ситуации (0, 1):

Выигрыш первого игрока: 10

Выигрыш второго игрока: 4

Проверка устойчивости:

Первому игроку невыгодно менять стратегию (10 >= 6)

Второму игроку невыгодно менять стратегию (4 >= 1)

============================================================

АНАЛИЗ ИЗВЕСТНЫХ ИГР

============================================================

------------------------------------------------------------

Анализ игры: Семейный спор

------------------------------------------------------------

Матрица выигрышей с выделением оптимальных стратегий:

[(4, 1)] (0, 0)

(0, 0) [(1, 4)]

Легенда:

< > - Парето-оптимальная ситуация

{ } - Равновесие Нэша

[ ] - Парето-оптимальная ситуация и равновесие Нэша

Анализ игры 'Семейный спор':

Парето-оптимальные ситуации:

Позиция (0, 0): (4, 1)

Позиция (1, 1): (1, 4)

Равновесия Нэша:

Позиция (0, 0): (4, 1)

Позиция (1, 1): (1, 4)

Пересечение множеств (Парето-оптимальные и равновесия Нэша):

Позиция (1, 1): (1, 4)

Позиция (0, 0): (4, 1)

------------------------------------------------------------

Анализ игры: Перекресток

------------------------------------------------------------

Матрица выигрышей с выделением оптимальных стратегий:

<(1.0, 1.0)> [(0.5, 2.0)]

[(2.0, 0.5)] (0.0, 0.0)

Легенда:

< > - Парето-оптимальная ситуация

{ } - Равновесие Нэша

[ ] - Парето-оптимальная ситуация и равновесие Нэша

Анализ игры 'Перекресток':

Парето-оптимальные ситуации:

Позиция (0, 0): (1.0, 1.0)

Позиция (0, 1): (0.5, 2.0)

Позиция (1, 0): (2.0, 0.5)

Равновесия Нэша:

Позиция (0, 1): (0.5, 2.0)

Позиция (1, 0): (2.0, 0.5)

Пересечение множеств (Парето-оптимальные и равновесия Нэша):

Позиция (0, 1): (0.5, 2.0)

Позиция (1, 0): (2.0, 0.5)

------------------------------------------------------------

Анализ игры: Дилемма заключенного (заданная)

------------------------------------------------------------

Матрица выигрышей с выделением оптимальных стратегий:

{(-5, -5)} <(0, -10)>

<(-10, 0)> <(-1, -1)>

Легенда:

< > - Парето-оптимальная ситуация

{ } - Равновесие Нэша

[ ] - Парето-оптимальная ситуация и равновесие Нэша

Анализ игры 'Дилемма заключенного (заданная)':

Парето-оптимальные ситуации:

Позиция (0, 1): (0, -10)

Позиция (1, 0): (-10, 0)

Позиция (1, 1): (-1, -1)

Равновесия Нэша:

Позиция (0, 0): (-5, -5)

Пересечение множеств (Парето-оптимальные и равновесия Нэша):

Пересечений нет

------------------------------------------------------------

Анализ игры: Классическая дилемма заключенного

------------------------------------------------------------

Матрица выигрышей с выделением оптимальных стратегий:

<(-1, -1)> <(-3, 0) >

<(0, -3) > {(-2, -2)}

Легенда:

< > - Парето-оптимальная ситуация

{ } - Равновесие Нэша

[ ] - Парето-оптимальная ситуация и равновесие Нэша

Анализ игры 'Классическая дилемма заключенного':

Парето-оптимальные ситуации:

Позиция (0, 0): (-1, -1)

Позиция (0, 1): (-3, 0)

Позиция (1, 0): (0, -3)

Равновесия Нэша:

Позиция (1, 1): (-2, -2)

Пересечение множеств (Парето-оптимальные и равновесия Нэша):

Пересечений нет

# Выводы

В ходе лабораторной работы были изучены критерии выбора стратегий в неантагонистической некооперативной игре двух участников: равновесие Нэша и оптимальность по Парето. Проведено исследование свойств оптимальных решений и проверена корректность работы созданной программы на примере биматричных игр размерности (2×2) и (10×10).

Равновесие Нэша демонстрирует устойчивость стратегий, но может не обеспечивать коллективно выгодного результата.

Парето-оптимальные решения могут увеличить выигрыши обоих игроков, но могут быть неустойчивыми с точки зрения индивидуальной рациональности.

В случае отсутствия равновесия в чистых стратегиях смешанное расширение гарантирует существование решения

import random

import numpy as np

# Параметры генерации случайной игры

GAME\_SIZE = (10, 10)

MIN\_VAL = -100

MAX\_VAL = 100

def generate\_bimatrix(rows, cols, min\_val, max\_val):

    """Генерация случайной биматричной игры"""

    matrix = []

    for i in range(rows):

        row = []

        for j in range(cols):

            p1 = random.randint(min\_val, max\_val)

            p2 = random.randint(min\_val, max\_val)

            row.append((p1, p2))

        matrix.append(row)

    return matrix

def find\_pareto\_optimal(matrix):

    """Поиск Парето-оптимальных ситуаций"""

    pareto\_optimal = []

    rows = len(matrix)

    cols = len(matrix[0])

    for i in range(rows):

        for j in range(cols):

            current = matrix[i][j]

            is\_dominated = False

            for x in range(rows):

                for y in range(cols):

                    other = matrix[x][y]

                    if (other[0] >= current[0] and other[1] >= current[1]) and \

                       (other[0] > current[0] or other[1] > current[1]):

                        is\_dominated = True

                        break

                if is\_dominated:

                    break

            if not is\_dominated:

                pareto\_optimal.append((i, j))

    return pareto\_optimal

def find\_nash\_equilibria(matrix):

    """Поиск равновесий Нэша в чистых стратегиях"""

    nash\_equilibria = []

    rows = len(matrix)

    cols = len(matrix[0])

    for i in range(rows):

        for j in range(cols):

            current = matrix[i][j]

            # Проверка оптимальности для первого игрока

            is\_p1\_optimal = True

            for x in range(rows):

                if matrix[x][j][0] > current[0]:

                    is\_p1\_optimal = False

                    break

            # Проверка оптимальности для второго игрока

            is\_p2\_optimal = True

            for y in range(cols):

                if matrix[i][y][1] > current[1]:

                    is\_p2\_optimal = False

                    break

            if is\_p1\_optimal and is\_p2\_optimal:

                nash\_equilibria.append((i, j))

    return nash\_equilibria

def analyze\_2x2\_game(matrix):

    """Анализ биматричной игры 2×2"""

    # Поиск равновесий в чистых стратегиях

    pure\_nash = find\_nash\_equilibria(matrix)

    # Проверка наличия смешанного равновесия

    A = np.array([[matrix[0][0][0], matrix[0][1][0]],

                 [matrix[1][0][0], matrix[1][1][0]]])

    B = np.array([[matrix[0][0][1], matrix[0][1][1]],

                 [matrix[1][0][1], matrix[1][1][1]]])

    mixed\_nash = None

    payoffs = None

    try:

        u = np.array([1, 1])

        A\_inv = np.linalg.inv(A)

        B\_inv = np.linalg.inv(B)

        v1 = 1 / (u @ A\_inv @ u)

        v2 = 1 / (u @ B\_inv @ u)

        x = v2 \* (u @ B\_inv)

        y = v1 \* (A\_inv @ u)

        # Проверка, что вероятности в допустимом диапазоне

        if all(0 <= p <= 1 for p in x) and all(0 <= p <= 1 for p in y):

            mixed\_nash = (x.tolist(), y.tolist())

            payoffs = (v1, v2)

    except np.linalg.LinAlgError:

        # Матрицы вырождены, смешанного равновесия нет

        pass

    return pure\_nash, mixed\_nash, payoffs

def print\_matrix\_with\_highlight(matrix, pareto\_indices=None, nash\_indices=None):

    """Вывод матрицы с выделением оптимальных стратегий"""

    rows = len(matrix)

    cols = len(matrix[0])

    # Определяем максимальную длину элементов для форматирования

    max\_len = 0

    for i in range(rows):

        for j in range(cols):

            cell\_len = len(f"({matrix[i][j][0]}, {matrix[i][j][1]})")

            if cell\_len > max\_len:

                max\_len = cell\_len

    print("Матрица выигрышей с выделением оптимальных стратегий:")

    for i in range(rows):

        for j in range(cols):

            cell = f"({matrix[i][j][0]}, {matrix[i][j][1]})"

            # Проверка, нужно ли выделять ячейку

            is\_pareto = pareto\_indices and (i, j) in pareto\_indices

            is\_nash = nash\_indices and (i, j) in nash\_indices

            if is\_pareto and is\_nash:

                cell = f"[{cell:^{max\_len}}]"  # Оба критерия

            elif is\_pareto:

                cell = f"<{cell:^{max\_len}}>"  # Только Парето

            elif is\_nash:

                cell = f"{{{cell:^{max\_len}}}}"  # Только Нэш

            else:

                cell = f" {cell:^{max\_len}} "  # Без выделения

            print(cell, end=" ")

        print()

    # Легенда

    print("\nЛегенда:")

    print("< > - Парето-оптимальная ситуация")

    print("{ } - Равновесие Нэша")

    print("[ ] - Парето-оптимальная ситуация и равновесие Нэша")

def print\_detailed\_analysis(matrix, pareto\_indices, nash\_indices, game\_name=""):

    """Подробный анализ оптимальных ситуаций"""

    if game\_name:

        print(f"\nАнализ игры '{game\_name}':")

    print("Парето-оптимальные ситуации:")

    for i, j in pareto\_indices:

        payoff = matrix[i][j]

        print(f"  Позиция ({i}, {j}): {payoff}")

    print("\nРавновесия Нэша:")

    if nash\_indices:

        for i, j in nash\_indices:

            payoff = matrix[i][j]

            print(f"  Позиция ({i}, {j}): {payoff}")

    else:

        print("  Равновесий Нэша не найдено")

    # Пересечение множеств (ситуации, оптимальные по обоим критериям)

    intersection = set(pareto\_indices) & set(nash\_indices)

    print("\nПересечение множеств (Парето-оптимальные и равновесия Нэша):")

    if intersection:

        for i, j in intersection:

            payoff = matrix[i][j]

            print(f"  Позиция ({i}, {j}): {payoff}")

    else:

        print("  Пересечений нет")

def analyze\_known\_game(matrix, game\_name):

    """Анализ известной игры"""

    print(f"\n{'-'\*60}")

    print(f"Анализ игры: {game\_name}")

    print(f"{'-'\*60}")

    # Поиск оптимальных ситуаций

    pareto\_optimal = find\_pareto\_optimal(matrix)

    nash\_equilibria = find\_nash\_equilibria(matrix)

    # Вывод матрицы с выделением оптимальных стратегий

    print\_matrix\_with\_highlight(matrix, pareto\_optimal, nash\_equilibria)

    # Подробный анализ

    print\_detailed\_analysis(matrix, pareto\_optimal, nash\_equilibria, game\_name)

# Классическая дилемма заключенного

def classical\_prisoners\_dilemma():

    """Классическая дилемма заключенного"""

    return [

        [(-1, -1), (-3, 0)],

        [(0, -3), (-2, -2)]

    ]

# Основная часть программы

if \_\_name\_\_ == "\_\_main\_\_":

    # Задание 1: Случайная игра 10×10

    random.seed(42)

    game\_matrix = generate\_bimatrix(10, 10, MIN\_VAL, MAX\_VAL)

    print("=" \* 60)

    print("ЗАДАНИЕ 1: Случайная биматричная игра 10×10")

    print("=" \* 60)

    # Поиск оптимальных ситуаций

    pareto\_optimal = find\_pareto\_optimal(game\_matrix)

    nash\_equilibria = find\_nash\_equilibria(game\_matrix)

    # Вывод результатов

    print("Парето-оптимальные ситуации (индексы строк и столбцов):", pareto\_optimal)

    print("Равновесия Нэша в чистых стратегиях:", nash\_equilibria)

    # Вывод матрицы с выделением оптимальных стратегий

    print\_matrix\_with\_highlight(game\_matrix, pareto\_optimal, nash\_equilibria)

    # Подробный анализ оптимальных ситуаций

    print\_detailed\_analysis(game\_matrix, pareto\_optimal, nash\_equilibria, "Случайная игра 10×10")

    # Задание 2: Анализ игры для варианта 9

    print("\n" + "=" \* 60)

    print("ЗАДАНИЕ 2: Анализ игры для варианта 9")

    print("=" \* 60)

    variant\_9\_matrix = [

        [(5, 1), (10, 4)],

        [(8, 6), (6, 9)]

    ]

    # Анализ игры

    pure\_nash, mixed\_nash, payoffs = analyze\_2x2\_game(variant\_9\_matrix)

    # Вывод матрицы игры

    print("Матрица выигрышей для варианта 9:")

    pareto\_optimal\_v9 = find\_pareto\_optimal(variant\_9\_matrix)

    nash\_equilibria\_v9 = find\_nash\_equilibria(variant\_9\_matrix)

    print\_matrix\_with\_highlight(variant\_9\_matrix, pareto\_optimal\_v9, nash\_equilibria\_v9)

    # Вывод результатов анализа

    print("\nРезультаты анализа:")

    print("Равновесия Нэша в чистых стратегиях:", pure\_nash)

    if mixed\_nash:

        print("Смешанное равновесие:")

        print(f"  Стратегия первого игрока: {mixed\_nash[0]}")

        print(f"  Стратегия второго игрока: {mixed\_nash[1]}")

        print(f"  Ожидаемые выигрыши: {payoffs}")

    else:

        print("Смешанного равновесия не найдено")

    # Дополнительный анализ для варианта 9

    print("\nДополнительный анализ:")

    if pure\_nash:

        for i, j in pure\_nash:

            print(f"В ситуации ({i}, {j}):")

            print(f"  Выигрыш первого игрока: {variant\_9\_matrix[i][j][0]}")

            print(f"  Выигрыш второго игрока: {variant\_9\_matrix[i][j][1]}")

            # Проверка устойчивости

            print("  Проверка устойчивости:")

            # Для первого игрока

            other\_payoff = variant\_9\_matrix[1-i][j][0] if i == 0 else variant\_9\_matrix[0][j][0]

            if variant\_9\_matrix[i][j][0] >= other\_payoff:

                print(f"    Первому игроку невыгодно менять стратегию ({variant\_9\_matrix[i][j][0]} >= {other\_payoff})")

            else:

                print(f"    Первому игроку выгодно менять стратегию ({variant\_9\_matrix[i][j][0]} < {other\_payoff})")

            # Для второго игрока

            other\_payoff = variant\_9\_matrix[i][1-j][1] if j == 0 else variant\_9\_matrix[i][0][1]

            if variant\_9\_matrix[i][j][1] >= other\_payoff:

                print(f"    Второму игроку невыгодно менять стратегию ({variant\_9\_matrix[i][j][1]} >= {other\_payoff})")

            else:

                print(f"    Второму игроку выгодно менять стратегию ({variant\_9\_matrix[i][j][1]} < {other\_payoff})")

    # Анализ известных игр

    print("\n" + "=" \* 60)

    print("АНАЛИЗ ИЗВЕСТНЫХ ИГР")

    print("=" \* 60)

    # 1. Семейный спор (Battle of the Sexes)

    battle\_of\_sexes = [

        [(4, 1), (0, 0)],

        [(0, 0), (1, 4)]

    ]

    analyze\_known\_game(battle\_of\_sexes, "Семейный спор")

    # 2. Перекресток (Crossroad)

    crossroad = [

        [(1.0, 1.0), (0.5, 2.0)],

        [(2.0, 0.5), (0.0, 0.0)]

    ]

    analyze\_known\_game(crossroad, "Перекресток")

    # 3. Дилемма заключенного (Prisoner's Dilemma)

    prisoners\_dilemma = [

        [(-5, -5), (0, -10)],

        [(-10, 0), (-1, -1)]

    ]

    analyze\_known\_game(prisoners\_dilemma, "Дилемма заключенного (заданная)")

    # 4. Классическая дилемма заключенного

    classical\_pd = classical\_prisoners\_dilemma()

    analyze\_known\_game(classical\_pd, "Классическая дилемма заключенного")