**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 №5 на тему:**

«Рациональный дележ в кооперативных играх» Вариант 9

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

Москва, 2025

# Цель работы

Изучить постановку кооперативной игры и найти оптимальное распределение выигрыша (дележ) между игроками путем вычисления компонент вектора Шепли.

# Задание

Для заданной характеристической функции игры (см. таблица 1) выполнить следующие задания:

1. Проверить кооперативную игру на супераддитивность и выпуклость. Если игра не супераддитивна, изменить характеристическую функцию таким образом, чтобы игра стала супераддитивной.
2. Составить программу вычисления компонент вектора Шепли и рассчитать его.
3. Проверить условия индивидуальной и групповой рационализации.

|  |  |
| --- | --- |
| 𝑣(∅) | *0* |
| 𝑣({1}) | *3* |
| 𝑣({2}) | *3* |
| 𝑣({3}) | *1* |
| 𝑣({4}) | *2* |
| 𝑣({1, 2}) | *7* |
| 𝑣({1, 3}) | *6* |
| 𝑣({1, 4}) | *7* |
| 𝑣({2, 3}) | *7* |
| 𝑣({2, 4}) | *5* |
| 𝑣({3, 4}) | *7* |
| 𝑣({1, 2, 3}) | *10* |
| 𝑣({1, 2, 4}) | *10* |
| 𝑣({1, 3, 4}) | *10* |
| 𝑣({2, 3, 4}) | *10* |
| 𝑣(𝐼) | *14* |

# Ход работы

Игра не является супераддитивной, то есть существует S и T при которых не выполняется следующие неравенства:

=== ПРОВЕРКА СУПЕРАДДИТИВНОСТИ ===

НАРУШЕНИЕ: v(2, 4) = 4 < v(2,) + v(4,) = 5

КОРРЕКТИРОВКА: v(2, 4) = 5

НАРУШЕНИЕ: v(1, 3, 4) = 9 < v(1,) + v(3, 4) = 10

КОРРЕКТИРОВКА: v(1, 3, 4) = 10

НАРУШЕНИЕ: v(2, 3, 4) = 7 < v(2,) + v(3, 4) = 10

КОРРЕКТИРОВКА: v(2, 3, 4) = 10

НАРУШЕНИЕ: v(1, 2, 3, 4) = 12 < v(1,) + v(2, 3, 4) = 13

КОРРЕКТИРОВКА: v(1, 2, 3, 4) = 13

НАРУШЕНИЕ: v(1, 2, 3, 4) = 13 < v(1, 2) + v(3, 4) = 14

КОРРЕКТИРОВКА: v(1, 2, 3, 4) = 14

=== СКОРРЕКТИРОВАННАЯ ХАРАКТЕРИСТИЧЕСКАЯ ФУНКЦИЯ ===

v() = 0

v(1,) = 3

v(2,) = 3

v(3,) = 1

v(4,) = 2

v(1, 2) = 7

v(1, 3) = 6

v(1, 4) = 7

v(2, 3) = 7

v(2, 4) = 5

v(3, 4) = 7

v(1, 2, 3) = 10

v(1, 2, 4) = 10

v(1, 3, 4) = 10

v(2, 3, 4) = 10

v(1, 2, 3, 4) = 14

=== РЕЗУЛЬТАТЫ РАСЧЕТА ===

Вектор Шепли: [3.8333, 3.6667, 3.1667, 3.3333]

Сумма компонент вектора Шепли: 14.0

v(I) = 14

=== ПРОВЕРКА РАЦИОНАЛИЗАЦИИ ===

Групповая рационализация: 14.0000 = 14 ✓

Индивидуальная рационализация:

x\_1 = 3.8333 >= v({'1'}) = 3 ✓

x\_2 = 3.6667 >= v({'2'}) = 3 ✓

x\_3 = 3.1667 >= v({'3'}) = 1 ✓

x\_4 = 3.3333 >= v({'4'}) = 2 ✓

# Выводы

В ходе выполнения лабораторной работы были изучены основы кооперативной теории игр, а также был определен тип, проведена проверка игра на супераддитивность и выпуклость, найдено оптимальное распределение выигрыша между игроками с помощью вычисления компонент вектора Шепли. Для полученного вектора Шепли была проведена проверка выполнимости условий групповой и индивидуальной рационализации.

Результаты работы показали, что вектор Шепли является эффективным механизмом распределения выигрыша, учитывающим вклад каждого участника. Условия супераддитивности и выпуклости обеспечивают устойчивость коалиций. Проверка рационализации подтвердила, что полученное распределение является устойчивым и приемлемым для всех игроков.

import math

from itertools import combinations

def check\_superadditivity(v, n\_players):

    """

    Проверка супераддитивности и автоматическая корректировка характеристической функции

    """

    print("=== ПРОВЕРКА СУПЕРАДДИТИВНОСТИ ===")

    # Исходные данные для варианта 9

    original\_v = {

        (): 0, (1,): 3, (2,): 3, (3,): 1, (4,): 2,

        (1,2): 7, (1,3): 6, (1,4): 7, (2,3): 7, (2,4): 4, (3,4): 7,

        (1,2,3): 10, (1,2,4): 10, (1,3,4): 9, (2,3,4): 7, (1,2,3,4): 12

    }

    # Начинаем с исходных значений

    corrected\_v = original\_v.copy()

    violations = []

    # Проверка всех возможных разбиений коалиций

    all\_players = tuple(range(1, n\_players+1))

    all\_coalitions = []

    # Генерируем все возможные коалиции

    for size in range(1, n\_players+1):

        for coalition in combinations(all\_players, size):

            all\_coalitions.append(tuple(sorted(coalition)))

    # Проверяем супераддитивность для всех пар непересекающихся коалиций

    for coalition in all\_coalitions:

        if len(coalition) == 0:

            continue

        # Генерируем все возможные разбиения коалиции на две непустые части

        for split\_point in range(1, len(coalition)):

            for part1 in combinations(coalition, split\_point):

                part1 = tuple(sorted(part1))

                part2 = tuple(sorted(set(coalition) - set(part1)))

                if len(part2) == 0:

                    continue

                current\_value = corrected\_v.get(coalition, 0)

                sum\_parts = corrected\_v.get(part1, 0) + corrected\_v.get(part2, 0)

                if current\_value < sum\_parts:

                    violations.append((coalition, part1, part2, current\_value, sum\_parts))

                    # Корректируем значение

                    corrected\_v[coalition] = sum\_parts

                    print(f"НАРУШЕНИЕ: v{coalition} = {current\_value} < v{part1} + v{part2} = {sum\_parts}")

                    print(f"КОРРЕКТИРОВКА: v{coalition} = {sum\_parts}")

    print(f"\nОбнаружено нарушений: {len(violations)}")

    return corrected\_v, violations

def characteristic\_function(coalition, corrected\_values):

    """

    Характеристическая функция с использованием скорректированных значений

    """

    coalition = tuple(sorted(coalition))

    return corrected\_values.get(coalition, 0)

def calculate\_shapley\_value(n, v):

    """

    Вычисление вектора Шепли

    """

    shapley\_values = [0] \* n

    all\_players = list(range(1, n+1))

    for player in range(1, n+1):

        for size in range(0, n):

            for S in combinations([p for p in all\_players if p != player], size):

                S = tuple(S)

                weight = (math.factorial(len(S)) \* math.factorial(n - len(S) - 1)) / math.factorial(n)

                marginal\_contribution = v(S + (player,)) - v(S)

                shapley\_values[player-1] += weight \* marginal\_contribution

    return shapley\_values

# Основная программа

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

    n\_players = 4

    # Шаг 1: Проверка и корректировка характеристической функции

    corrected\_v, violations = check\_superadditivity(None, n\_players)

    print("\n=== СКОРРЕКТИРОВАННАЯ ХАРАКТЕРИСТИЧЕСКАЯ ФУНКЦИЯ ===")

    for coalition in sorted(corrected\_v.keys(), key=lambda x: (len(x), x)):

        print(f"v{coalition} = {corrected\_v[coalition]}")

    # Шаг 2: Вычисление вектора Шепли

    shapley\_values = calculate\_shapley\_value(n\_players,

                                           lambda coalition: characteristic\_function(coalition, corrected\_v))

    print("\n=== РЕЗУЛЬТАТЫ РАСЧЕТА ===")

    print("Вектор Шепли:", [round(val, 4) for val in shapley\_values])

    print("Сумма компонент вектора Шепли:", round(sum(shapley\_values), 4))

    print("v(I) =", corrected\_v[tuple(range(1, n\_players+1))])

    # Шаг 3: Проверка рационализации

    print("\n=== ПРОВЕРКА РАЦИОНАЛИЗАЦИИ ===")

    # Групповая рационализация

    group\_sum = sum(shapley\_values)

    v\_I = corrected\_v[tuple(range(1, n\_players+1))]

    print(f"Групповая рационализация: {group\_sum:.4f} = {v\_I} ✓"

          if abs(group\_sum - v\_I) < 0.001

          else f"Групповая рационализация: {group\_sum:.4f} ≠ {v\_I} ✗")

    # Индивидуальная рационализация

    for i in range(n\_players):

        individual\_value = corrected\_v.get((i+1,), 0)

        shapley\_val = shapley\_values[i]

        status = "✓" if shapley\_val >= individual\_value else "✗"

        print(f"x\_{i+1} = {shapley\_val:.4f} >= v({{'{i+1}'}}) = {individual\_value} {status}")