Министерство науки и высшего образования Российской Федерации

Московский государственный технический университет имени

Н. Э. Баумана

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

Лабораторная работа №3

ПО ДИСЦИПЛИНЕ «ТЕОРИЯ ИГР И ИССЛЕДОВАНИЕ ОПЕРАЦИЙ»

«Неантагонистические бескоалиционные игры»

Вариант 1

Студент: Анаян М. С., ИУ8-104
Преподаватель: Коннова Н. С.

Москва, 2020

Цель и задачи выполнения лабораторной работы

Цель работы – изучить критерии выбора стратегий в неантагонистической бесколиционной игре двух игроков на основе равновесия Нэша и оптимальности по Парето. Проверить данные критерии на примере игр «Семейный спор», «Дилемма заключённого» и «Перекрёсток». Исследовать свойства оптимальных решений неантагонистических бескоалиционных игр на примере биматричных (2×2)-игр.

Постановка задачи

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 трёх известных игр: «Семейный спор», «Дилемма заключённого» и «Перекрёсток».
2. Для заданной биматричной (2×2)-игры $\Gamma(A, B)$, пользуясь теоремами о свойствах оптимальных решений, найти ситуации, равновесные по Нэшу, для исходной игры и её смешанного расширения.

Теоретическая часть

Пусть дана игра $\Gamma = (N, \{x_i\}_{i\in N}, \{H_i\}_{i\in N})$, где $N = \{1, 2, \dots, n\}$ – множество игроков, а X_i – множество стратегий игрока i, определённая на декартовом произведении множеств стратегий игроков $X = (X_1, X_2, \dots, X_n)$ – множество ситуаций игры, – называется *бескоалиционной игрой*.

Ситуация $x = (x_1, \dots, x_n), x_i \in X_i, i = \overline{1, n}$ получается случайным независимым выбором x_i для каждого игрока i. Каждый игрок i получает свой выигрыш $H_i(x) = H_i(x_1, \dots, x_n)$, после чего игра заканчивается. Если все множества чистых стратегий игроков X_i конечны, игра называется конечной *бескоалиционной игрой n лиц.*

Ситуацией *равновесия по Нэшу* в чистых стратегиях называется такая ситуация $x^* = (x_1^*, x_2^*, \dots, x_n^*)$, при которой $\forall i \in N, \forall x_i \in X_i$:

$$H_i(x^*) \geq H_i(x'),$$

где $x' = (x_1', x_2', \dots, x_n')$, где $x_k' = x_k^*$, если $k \neq i$, иначе $x_k' = x_k$.

Множеством Нэша называют совокупность всех равновесных по Нэшу ситуаций.

Рассмотрим множество $\{H(x)\} = \{H_1(x), \dots, H_n(x)\}, x \in X, x = (x_1, \dots, x_n)$, что является множеством значений вектор-выигрышей игроков во всех возможных ситуациях $x \in X$.

Ситуация $\overline{x} \in X$ в бескоалиционной игре Г называется *оптимальной по Парето*, если не существует ситуации $x \in X$, для которой верно:

$$H_i(x) \geq H_i(\overline{x}) \text{ для } \forall i \in N,$$

$$\exists j: H_j(x) > H_j(\overline{x}).$$

Пусть $\Gamma(A, B)$ – биматричная (2×2)-игра, где (A, B) – невырожденные матрицы.

$$(A, B) = \begin{pmatrix} (\alpha_1, \beta_1) & (\alpha_1, \beta_2) \\ (\alpha_2, \beta_1) & (\alpha_2, \beta_2) \end{pmatrix}.$$

В случае существования вполне смешанной стратегии в смешанном дополнении игры, ситуация равновесия может быть вычислена следующим образом:

– Стоимости игры 1 игрока и 2 игрока вычисляются по следующим формулам соответственно:

$$v_1 = \frac{1}{uA^{-1}u^T}, v_2 = \frac{1}{uB^{-1}u^T},$$

где $u = (1, 1)$;

– Вполне смешанная стратегия вычисляется по формулам:

$$x = v_2 uB^{-1}, y = v_1 A^{-1}u^T.$$

Возможны 3 случая:

1. Если в исходной игре Г по крайней мере один игрой имеет строго доминирующую стратегию, тогда игра Г и её смешанное расширение $\overline{\Gamma}$ имеют единственную ситуацию равновесия по Нэшу;

2. Если игра Г не имеет ситуации равновесия по Нэшу в чистых стратегиях, то в игре существует вполне смешанная ситуация равновесия (x^*, y^*);

3. Если игра Г имеет две равновесные по Нэшу ситуации, в смешанном дополнении игры существует ещё одна вполне смешанная ситуация равновесия (x^*, y^*).

Практическая часть

Часть 1

При выполнении лабораторной работы 1 были реализованы функции поиска ситуаций, равновесных по Нэшу и оптимальных по Парето. Исходный код представлен в приложении А и репозитории по ссылке: [https://github.com/hms2010/GameTheory/tree/master/lab3.](https://github.com/hms2010/GameTheory/tree/master/lab3)

Результаты выполнения функций поиска ситуаций, равновесных по Нэшу и оптимальных по Парето, для игр «Семейный спор», «Дилемма заключённого» и «Перекрёсток» соответствуют ожидаемым.

Случайно сгенерированная матрица G биматричной игры $\Gamma(A, B)$ (10×10):

```
-15/-28,  44/ 36, -39/ 25,  46/ 46,   5/ -5,  26/-42, -22/-48,  16/-12,  42/ 31,   8/ 27
-48/  1,  -2/-46, -49/-24,  34/-43, -13/-48, -28/-35,  42/ 35, -50/ 45,  22/ 50, -22/-39
 -3/-27,  43/  0, -11/-19,  -3/-27, -15/  9,  -2/-16,  21/ 31,  22/ 48, -19/ 47, -44/-25
-19/ 38,  46/ 41,  28/-20, -11/ 29, -41/-49,  13/  5,  40/-35,  39/ 15,  -5/ -3,  41/-14
-14/-26,  -1/ 15,   1/ 34,  33/ 50,  11/ 31,  -3/ 12, -26/-27,  40/ -9,  36/ 25,  31/  7
 47/ -9,   1/ 45, -11/ 38,  27/-48,  -4/-45,  47/-24,  11/ -4,  35/ 24, -26/ 28,  50/ 35
-27/-31,  16/-24,  33/ -5, -30/-41, -21/ 43, -45/ 50, -23/ -1,  -6/-12,  -1/-46, -38/-25
-13/ 38, -33/ -3,  23/ 22, -18/-34,  20/  9, -27/-47, -14/ -1,  22/-46,  15/ 39,  21/ 13
-50/ 17, -18/-33,  26/-24,  28/ 37,  27/-39,  27/-50,   9/ 42,  23/ 46, -12/ -4, -22/ 49
 37/ 27,  34/-46, -24/-29, -29/ -7,  16/ 17,  40/ 42,  32/ 44, -36/ 41, -13/ -9,   0/ 26
```

Стратегии первого игрока записаны по строкам, второго – по столбцам.

Ситуации, равновесные по Нэшу, выделены *красным курсивом*.

Ситуации, оптимальные по Парето, выделены **жирным синим**.

Ситуации, и равновесные по Нэшу, и оптимальные по Парето, выделены ***жирным фиолетовым курсивом***.

Множество Нэша:

$$\{(1,4): H(1,4) = (46,46); (4,2): H(4,2) = (46,41)\}.$$

Множество Парето:

$$\{(1,4): H = (46,46); (5,4): H = (33,50);\ (6,10): H = (50,35)\}.$$

Ситуации, и равновесные по Нэшу, и оптимальные по Парето:

$$\{(1,4): H = (46,46)\}.$$

С выводом результатов выполнения программы можно ознакомиться в приложении Б.

Часть 2

Рассмотрим биматричную (2×2)-игру $\Gamma(A, B)$ с платёжной матрицей G (стратегии 1 игрока – строки, 2 игрока – столбцы):

$$G = \begin{pmatrix} 5, 0 & 8, 4 \\ 7, 6 & 6, 3 \end{pmatrix}.$$

Для игрока 1 матрица стоимости игры:

$$A = \begin{pmatrix} 5 & 8 \\ 7 & 6 \end{pmatrix},$$

Для игрока 2 матрица стоимости игры:

$$B = \begin{pmatrix} 0 & 4 \\ 6 & 3 \end{pmatrix}.$$

Множество Нэша:

$$\{(1, 2) \colon H(1, 2) = (8, 4); (2, 1) \colon H(2, 1) = (7, 6)\}.$$

Множество Парето:

$$\{(1, 2) \colon H(1, 2) = (8, 4); (2, 1) \colon H(2, 1) = (7, 6)\}.$$

Ситуации, и равновесные по Нэшу, и оптимальные по Парето:

$$\{(1, 2) \colon H = (8, 4); (2, 1) \colon H = (7, 6)\}.$$

Т. к. у игры две равновесные по Нэшу ситуации в чистых стратегиях, в смешанном дополнении игры существует ещё одна вполне смешанная стратегия:

$$x = \left(\frac{3}{7}, \frac{4}{7}\right), y = \left(\frac{1}{2}, \frac{1}{2}\right).$$

Выигрыши при этом первого и второго игрока соответственно:

$$v_1 = \frac{13}{2} = 6.5, v_2 = \frac{24}{7} \approx 3.4.$$

Выводы

В результате выполнения лабораторной работы получены следующие результаты:

– Изучены критерии выбора стратегий в неантагонистической бескоалиционной игре двух игроков на основе равновесия Нэша и оптимальности Парето;

– Реализованы и протестированы на играх «Семейный спор», «Перекрёсток» и «Дилемма заключённого» функции для определения ситуаций равновесия Нэша и оптимальности Парето для биматричной неантагонистической бескоалиционной игры;

– Найдены ситуации равновесия Нэша и оптимальности Парето, а также их пересечения для случайно сгенерированной биматричной игры (10×10);

– Для заданной биматричной (2×2)- игры $\Gamma(A, B)$ при помощи теорем о свойствах оптимальных решений найдены ситуации, равновесные по Нэшу, для исходной игры и для её смешанного расширения.

Исходные коды программ представлены по ссылке: https://github.com/hms2010/GameTheory/tree/master/lab3.

Приложение А

```
import random

MAX_COST = 50
MIN_COST = -MAX_COST

def generate_game(nrows, ncols):
    random.seed()
    G = []
    for i in range(nrows):
        G.append([0] * ncols)
        for j in range(ncols):
            G[i][j] = {'a': random.randint(MIN_COST, MAX_COST), 'b': random.randint(MIN_COST, MAX
_COST)}
    return G

def print_game_matrix(G):
    for i in G:
        line = ""
        for j in i:
            cur_elem = "{:6.2f}/{:6.2f}".format(j['a'], j['b'])
            if not line:
                line = cur_elem
            else:
                line = ", ".join([line, cur_elem])
        print(line)

class player_strategy:
    strat = None
    cost = None
    def __init__(self, _strat, _cost):
        self.strat = _strat
        self.cost = _cost

    def __eq__(self, other):
        return isinstance(other, player_strategy) and self.strat == other.strat and self.cost ==
other.cost

def print_point(full_point):
    # full_point = {'a': player_strategy(i, G[i][j]['a']), 'b': player_strategy(j, G[i][j]['b'])}
    print("G[{:d}][{:d}] = ({:6.2f}/{:6.2f})".format(full_point['a'].strat + 1, full_point['b'].s
trat + 1, full_point['a'].cost, full_point['b'].cost))

def check_nash_equilibrium(point, G):
    na = len(G)
    nb = len(G[0])
    is_nash_equilibrium = True
    # point = {'a': player_strategy(i, G[i][j]['a']), 'b': player_strategy(j, G[i][j]['b'])}
    fixed_strat = point['a'].strat
    for b_strat in range(nb):
        if point['b'].strat == b_strat:
            continue
        if point['b'].cost < G[fixed_strat][b_strat]['b']:
            is_nash_equilibrium = False

    fixed_strat = point['b'].strat
    for a_strat in range(na):
        if point['a'].strat == a_strat:
            continue
        if point['a'].cost < G[a_strat][fixed_strat]['a']:
            is_nash_equilibrium = False
    return is_nash_equilibrium

def get_nash_equilibrium_strats(G):
```

```
    points = []
    # point = {'a': player_strategy(i, G[i][j]['a']), 'b': player_strategy(j, G[i][j]['b'])}
    na = len(G)
    nb = len(G[0])
    for i in range(na):
        for j in range(nb):
            point = {'a': player_strategy(i, G[i][j]['a']), 'b': player_strategy(j, G[i][j]['b'])
}
            if check_nash_equilibrium(point, G):
                if point not in points:
                    points.append(point)
    return points

def check_pareto_efficiency(point, G):
    na = len(G)
    nb = len(G[0])
    # point = {'a': player_strategy(i, G[i][j]['a']), 'b': player_strategy(j, G[i][j]['b'])}
    x = point['a'].strat
    y = point['b'].strat

    for a_strat in range(na):
        for b_strat in range(nb):
            if(G[a_strat][b_strat]['a'] >= G[x][y]['a'] and G[a_strat][b_strat]['b'] >= G[x][y]['
b']):
                if G[a_strat][b_strat]['a'] > G[x][y]['a'] or G[a_strat][b_strat]['b'] >  G[x][y]
['b']:
                    return False
    return True

def get_pareto_efficiency_strats(G):
    points = []
    # point = {'a': player_strategy(i, G[i][j]['a']), 'b': player_strategy(j, G[i][j]['b'])}
    na = len(G)
    nb = len(G[0])
    for i in range(na):
        for j in range(nb):
            point = {'a': player_strategy(i, G[i][j]['a']), 'b': player_strategy(j, G[i][j]['b'])
}
            if check_pareto_efficiency(point, G):
                if point not in points:
                    points.append(point)
    return points

def get_common_elems(a, b):
    common_elems = []
    if len(a) > len(b):
        for i in range(len(b)):
            if b[i] in a:
                common_elems.append(b[i])
    else:
        for i in range(len(a)):
            if a[i] in b:
                common_elems.append(a[i])
    return common_elems

def fam_bet_game():
    G = [
            [{'a': 4, 'b': 1}, {'a': 0, 'b': 0}],
            [{'a': 0, 'b': 0}, {'a': 1, 'b': 4}]
        ]
    print("Family bet game:")
    print_game_matrix(G)
    nash_eq_points = get_nash_equilibrium_strats(G)
    pareto_opt_points = get_pareto_efficiency_strats(G)
    both = get_common_elems(nash_eq_points, pareto_opt_points)

    print("Nash equilibrium points:")
```

```
    for i in nash_eq_points:
       print_point(i)
    print("Pareto optimal points:")
    for i in pareto_opt_points:
        print_point(i)
    print("Both criterias:")
    for i in both:
        print_point(i)

def prisoners_dilemma_game():
    G = [
            [{'a': -5, 'b': -5}, {'a': 0, 'b': -10}],
            [{'a': -10, 'b': 0}, {'a': -1, 'b': -1}]
        ]
    print("Prisoner's dilemma game:")
    print_game_matrix(G)
    nash_eq_points = get_nash_equilibrium_strats(G)
    pareto_opt_points = get_pareto_efficiency_strats(G)
    both = get_common_elems(nash_eq_points, pareto_opt_points)

    print("Nash equilibrium points:")
    for i in nash_eq_points:
       print_point(i)
    print("Pareto optimal points:")
    for i in pareto_opt_points:
        print_point(i)
    print("Both criterias:")
    for i in both:
        print_point(i)

def crossroads_game():
    random.seed()
    eps = random.randint(1, 100) / 100
    G = [
            [{'a': 1, 'b': 1}, {'a': 1 - eps, 'b': 2}],
            [{'a': 2, 'b': 1 - eps}, {'a': 0, 'b': 0}]
        ]
    print("Crossroads game:")
    print_game_matrix(G)
    nash_eq_points = get_nash_equilibrium_strats(G)
    pareto_opt_points = get_pareto_efficiency_strats(G)
    both = get_common_elems(nash_eq_points, pareto_opt_points)

    print("Nash equilibrium points:")
    for i in nash_eq_points:
       print_point(i)
    print("Pareto optimal points:")
    for i in pareto_opt_points:
        print_point(i)
    print("Both criterias:")
    for i in both:
        print_point(i)

def main():
    fam_bet_game()
    print()
    prisoners_dilemma_game()
    print()
    crossroads_game()
    print()
    try:
        from game_data import G
        print("Game was imported")
    except ModuleNotFoundError:
        n = 10 # int(input("n = "))
        m = 10 # int(input("m = "))
        G = generate_game(n, m)
```

```
        with open("game_data.py", 'w') as fout:
            fout.write('G = {:}'.format(G))
            fout.close()
        print("Game was created")

    print("Game n x m:")
    print_game_matrix(G)
    nash_eq_points = get_nash_equilibrium_strats(G)
    print("Nash equilibrium points:")
    for i in nash_eq_points:
       print_point(i)
    pareto_opt_points = get_pareto_efficiency_strats(G)
    print("Pareto optimal points:")
    for i in pareto_opt_points:
        print_point(i)
    print("Both criterias:")
    both = get_common_elems(nash_eq_points, pareto_opt_points)
    for i in both:
        print_point(i)

if __name__ == '__main__':
    main()
```

Приложение Б

```
Family bet game:
  4.00/  1.00,   0.00/  0.00
  0.00/  0.00,   1.00/  4.00
Nash equilibrium points:
G[1][1] = (  4.00/  1.00)
G[2][2] = (  1.00/  4.00)
Pareto optimal points:
G[1][1] = (  4.00/  1.00)
G[2][2] = (  1.00/  4.00)
Both criterias:
G[1][1] = (  4.00/  1.00)
G[2][2] = (  1.00/  4.00)

Prisoner's dilemma game:
 -5.00/ -5.00,   0.00/-10.00
-10.00/  0.00,  -1.00/ -1.00
Nash equilibrium points:
G[1][1] = ( -5.00/ -5.00)
Pareto optimal points:
G[1][2] = (  0.00/-10.00)
G[2][1] = (-10.00/  0.00)
G[2][2] = ( -1.00/ -1.00)
Both criterias:

Crossroads game:
  1.00/  1.00,   0.90/  2.00
  2.00/  0.90,   0.00/  0.00
Nash equilibrium points:
G[1][2] = (  0.90/  2.00)
G[2][1] = (  2.00/  0.90)
Pareto optimal points:
G[1][1] = (  1.00/  1.00)
G[1][2] = (  0.90/  2.00)
G[2][1] = (  2.00/  0.90)
Both criterias:
G[1][2] = (  0.90/  2.00)
G[2][1] = (  2.00/  0.90)

Game was created
Game n x m:
-15.00/-28.00,  44.00/ 36.00, -39.00/ 25.00,  46.00/ 46.00,   5.00/ -5.00,  26.00/-42.00, -22.00/-48.00,  16.00/-12.00,  42.00/ 31.00,   8.00/ 27.00
-48.00/  1.00,  -2.00/-46.00, -49.00/-24.00,  34.00/-43.00, -13.00/-48.00, -28.00/-35.00,  42.00/ 35.00, -50.00/ 45.00,  22.00/ 50.00, -22.00/-39.00
 -3.00/-27.00,  43.00/  0.00, -11.00/-19.00,  -3.00/-27.00, -15.00/  9.00,  -2.00/-16.00,  21.00/ 31.00,  22.00/ 48.00, -19.00/ 47.00, -44.00/-25.00
-19.00/ 38.00,  46.00/ 41.00,  28.00/-20.00, -11.00/ 29.00, -41.00/-49.00,  13.00/  5.00,  40.00/-35.00,  39.00/ 15.00,  -5.00/ -3.00,  41.00/-14.00
-14.00/-26.00,  -1.00/ 15.00,   1.00/ 34.00,  33.00/ 50.00,  11.00/ 31.00,  -3.00/ 12.00, -26.00/-27.00,  40.00/ -9.00,  36.00/ 25.00,  31.00/  7.00
 47.00/ -9.00,   1.00/ 45.00, -11.00/ 38.00,  27.00/-48.00,  -4.00/-45.00,  47.00/-24.00,  11.00/ -4.00,  35.00/ 24.00, -26.00/ 28.00,  50.00/ 35.00
-27.00/-31.00,  16.00/-24.00,  33.00/ -5.00, -30.00/-41.00, -21.00/ 43.00, -45.00/ 50.00, -23.00/ -1.00,  -6.00/-12.00,  -1.00/-46.00, -38.00/-25.00
-13.00/ 38.00, -33.00/ -3.00,  23.00/ 22.00, -18.00/-34.00,  20.00/  9.00, -27.00/-47.00, -14.00/ -1.00,  22.00/-46.00,  15.00/ 39.00,  21.00/ 13.00
-50.00/ 17.00, -18.00/-33.00,  26.00/-24.00,  28.00/ 37.00,  27.00/-39.00,  27.00/-50.00,   9.00/ 42.00,  23.00/ 46.00, -12.00/ -4.00, -22.00/ 49.00
 37.00/ 27.00,  34.00/-46.00, -24.00/-29.00, -29.00/ -7.00,  16.00/ 17.00,  40.00/ 42.00,  32.00/ 44.00, -36.00/ 41.00, -13.00/ -9.00,   0.00/ 26.00
Nash equilibrium points:
G[1][4] = ( 46.00/ 46.00)
G[4][2] = ( 46.00/ 41.00)
Pareto optimal points:
G[1][4] = ( 46.00/ 46.00)
G[5][4] = ( 33.00/ 50.00)
G[6][10] = ( 50.00/ 35.00)
Both criterias:
G[1][4] = ( 46.00/ 46.00)
```