**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.

# Цель и задачи выполнения лабораторной работы

**Цель работы –** изучить критерии выбора стратегий в неантагонистической бескоалиционной 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 , пользуясь теоремами о свойствах оптимальных решений, найти ситуации, равновесные по Нэшу, для исходной игры и её смешанного расширения.

# Теоретическая часть

Пусть дана игра где – множество игроков, а – множество стратегий игрока , определённая на декартовом произведении множеств стратегий игроков – множество ситуаций игры, – называется *бескоалиционной игрой*.

Ситуация получается случайным независимым выбором для каждого игрока . Каждый игрок получает свой выигрыш , после чего игра заканчивается. Если все множества чистых стратегий игроков конечны, игра называется конечной *бескоалиционной игрой лиц.*

Ситуацией *равновесия по Нэшу* в чистых стратегиях называется такая ситуация , при которой :

где .

*Множеством Нэша* называют совокупность всех равновесных по Нэшу ситуаций.

Рассмотрим множество , что является множеством значений вектор-выигрышей игроков во всех возможных ситуациях .

Ситуация в бескоалиционной игре называется *оптимальной по Парето*, если не существует ситуации , для которой верно:

Пусть – биматричная (2×2)-игра, где – невырожденные матрицы.

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:

где ;

* Вполне смешанная стратегия вычисляется по формулам:

Возможны 3 случая:

1. Если в исходной игре по крайней мере один игрой имеет строго доминирующую стратегию, тогда игра и её смешанное расширение имеют единственную ситуацию равновесия по Нэшу;
2. Если игра не имеет ситуации равновесия по Нэшу в чистых стратегиях, то в игре существует вполне смешанная ситуация равновесия ;
3. Если игра имеет две равновесные по Нэшу ситуации, в смешанном дополнении игры существует ещё одна вполне смешанная ситуация равновесия .

# 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>.

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 (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:

Множество Парето:

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

С выводом результатов выполнения программы можно ознакомиться в приложении Б.

## Часть 2

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

Для игрока 1 матрица стоимости игры:

Для игрока 2 матрица стоимости игры:

Множество Нэша:

Множество Парето:

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

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

Выигрыши при этом первого и второго игрока соответственно:

# 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 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/lab](https://github.com/hms2010/GameTheory/tree/master/lab3)3.

# 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'].strat + 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)