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 «Информатика и системы управления» (ИУ)
КАФЕДРА «Информационная безопасность» (ИУ8)

Рубежный контроль №3 на тему "Информационное противоборство"

по дисциплине «Теория игр и исследование операций»

Вариант 12

Студент ИУ8-104 (Группа) Мильченко И. Д. (И. О. Фамилия) _____ (Подпись, дата)

Преподаватель Коннова Н. С. (И. О. Фамилия) _____ (Подпись, дата)

Москва, 2025 г.

ЦЕЛЬ РАБОТЫ

Изучить теоретико-игровую модель информационного противоборства в социальных сетях. Найти аналитическое решение игры с не противоположными интересами двух игроков и определить итоговые расстояния до точки утопии.

Задание

Необходимо проделать следующие шаги для выполнения работы:

- Для 10 агентов случайным образом сгенерировать стохастическую матрицу доверия.
- Получить результирующую матрицу доверия.
- Случайным образом выбрать номера агентов из общего числа агентов для первого и второго игроков.
- Определить функции выигрыша, целевые функции, точку утопии, найти аналитическое решение игры с не противоположными интересами двух игроков.

Номер варианта	a	b	c	d	g_f	g_s
12	2	1	4	4	1	3

ХОД РАБОТЫ

В проекте был написан класс для решения игр с моделью информационного противоборства в социальных сетях.

Пример запуска программы:

```
go run cmd/rk3/main.go
```

Сгенерируем стохастическую матрицу доверия для 10 агентов случайным образом при помощи написанной программы. Вывод программы представлен на рисунке 1

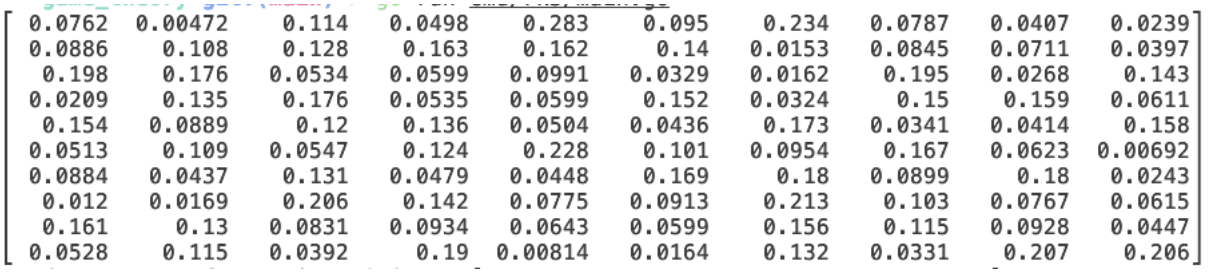

Рисунок 1 – Случайная матрица доверя 10x10

Сгенерированный вектор изначальных мнений агентов представлен на рисунке 2.

Рисунок 2 – Вектор изначальных мнений агентов

1 Нахождение итогового мнения агентов без влияния

Применим алогоритм согласно формуле (1) с точностью $\mathcal{E} = 10^{-6}$.

$$\mathbf{x}(t) = A\mathbf{x}(t-1). \tag{1}$$

где A – случайная матрица доверия, а $x(0)$ – вектор изначальных мнений агентов.

Спустя 11 итераций алгоритм остановится, так как разница между двумя последними векторами в каждой координате оказались меньше 10^{-6}. Вывод итераций представлен на рисунке 3.

```
x(0)=[16.17  14.08  12.82  12.48   14.1  15.05  14.89  15.69  14.75  12.28]
x(1)=[14.41  14.11  14.45  14.23  14.01   14.4  14.56  13.99  14.51  13.78]
x(2)=[14.28  14.24  14.14  14.26  14.26  14.21  14.37  14.31  14.29  14.24]
x(3)=[14.27  14.24  14.26  14.24  14.26  14.27  14.26  14.26  14.27  14.27]
x(4)=[14.26  14.26  14.26  14.26  14.26  14.26  14.26  14.26  14.26  14.26]
x(5)=[14.26  14.26  14.26  14.26  14.26  14.26  14.26  14.26  14.26  14.26]
x(6)=[14.26  14.26  14.26  14.26  14.26  14.26  14.26  14.26  14.26  14.26]
x(7)=[14.26  14.26  14.26  14.26  14.26  14.26  14.26  14.26  14.26  14.26]
x(8)=[14.26  14.26  14.26  14.26  14.26  14.26  14.26  14.26  14.26  14.26]
x(9)=[14.26  14.26  14.26  14.26  14.26  14.26  14.26  14.26  14.26  14.26]
x(10)=[14.26  14.26  14.26  14.26  14.26  14.26  14.26  14.26  14.26  14.26]
```

Рисунок 3 – Итерации примененного алоритма

После взаимодействия агентов, вектор мнений сходится к значению 14.26.

Результирующая матрица доверия представлена на рисунке 4.

Res trust matrix:

0.0924	0.0918	0.113	0.102	0.105	0.0927	0.125	0.107	0.0948	0.075
0.0924	0.0918	0.113	0.102	0.105	0.0927	0.125	0.107	0.0948	0.075
0.0924	0.0918	0.113	0.102	0.105	0.0927	0.125	0.107	0.0948	0.075
0.0924	0.0918	0.113	0.102	0.105	0.0927	0.125	0.107	0.0948	0.075
0.0924	0.0918	0.113	0.102	0.105	0.0927	0.125	0.107	0.0948	0.075
0.0924	0.0918	0.113	0.102	0.105	0.0927	0.125	0.107	0.0948	0.075
0.0924	0.0918	0.113	0.102	0.105	0.0927	0.125	0.107	0.0948	0.075
0.0924	0.0918	0.113	0.102	0.105	0.0927	0.125	0.107	0.0948	0.075
0.0924	0.0918	0.113	0.102	0.105	0.0927	0.125	0.107	0.0948	0.075
0.0924	0.0918	0.113	0.102	0.105	0.0927	0.125	0.107	0.0948	0.075

Рисунок 4 – Результирующая матрица доверия

2 Решение игры с информационным влиянием

Индексы агентов влияния для 2-х игроков представлены на рисунке 5.

```
Indices of agents of influence for the first player: [1 2 5 9]
Indices of agents of influence for the second player: [0 4 6 8]
```

Рисунок 5 – Индексы агентов влияния для 2-х игроков

Случайные изначальные мнения для двух игроков представлены на рисунке 6.

```
Initial opinion of 1 player's agents:  92
Initial opinion of 2 player's agents:  -44
```

Рисунок 6 – Случайные изначальные мнения для 2-х игроков

Аналогично первому пункту, применим алгоритм согласно формуле (1) и остановимся, когда точность не будет превышать 10^{-6}. Итерации алгоритма представлены на рисунке 7.

```
x(0)=[-3.458   28.41   27.3   40.22   22.93   11.53   14.88   22.84    12.5   21.68]
x(1)=[18.88  22.04  18.96  21.07  19.72  21.96  16.94  22.42   18.4  21.95]
x(2)=[19.42  20.51  20.84  20.54   19.9   20.4  19.65  19.71  19.94  20.11]
x(3)=[19.99  20.21  20.01  20.23  20.09  20.07  20.07  20.18  20.02  20.12]
x(4)=[20.08  20.11  20.11   20.1   20.1  20.12  20.07  20.09   20.1  20.11]
x(5)=[20.09   20.1   20.1   20.1   20.1   20.1   20.1   20.1  20.09   20.1]
x(6)=[20.1  20.1  20.1  20.1  20.1  20.1  20.1  20.1  20.1  20.1]
x(7)=[20.1  20.1  20.1  20.1  20.1  20.1  20.1  20.1  20.1  20.1]
x(8)=[20.1  20.1  20.1  20.1  20.1  20.1  20.1  20.1  20.1  20.1]
x(9)=[20.1  20.1  20.1  20.1  20.1  20.1  20.1  20.1  20.1  20.1]
x(10)=[20.1  20.1  20.1  20.1  20.1  20.1  20.1  20.1  20.1  20.1]
x(11)=[20.1  20.1  20.1  20.1  20.1  20.1  20.1  20.1  20.1  20.1]
x(12)=[20.1  20.1  20.1  20.1  20.1  20.1  20.1  20.1  20.1  20.1]
```

Рисунок 7 – Итерации алгоритма

После взаимодействия агентов, вектор мнений сходится к значения, показанный на рисунке 8

```
After the agents interact, the opinion vector converges to the value of X =
[20.1  20.1  20.1  20.1  20.1  20.1  20.1  20.1  20.1  20.1]
```

Рисунок 8 – Вектор мнений

Результирующая матрица доверия представлена на рисунке 9.

```
Result trust matrix:
```

0.0924	0.0918	0.113	0.102	0.105	0.0927	0.125	0.107	0.0948	0.075
0.0924	0.0918	0.113	0.102	0.105	0.0927	0.125	0.107	0.0948	0.075
0.0924	0.0918	0.113	0.102	0.105	0.0927	0.125	0.107	0.0948	0.075
0.0924	0.0918	0.113	0.102	0.105	0.0927	0.125	0.107	0.0948	0.075
0.0924	0.0918	0.113	0.102	0.105	0.0927	0.125	0.107	0.0948	0.075
0.0924	0.0918	0.113	0.102	0.105	0.0927	0.125	0.107	0.0948	0.075
0.0924	0.0918	0.113	0.102	0.105	0.0927	0.125	0.107	0.0948	0.075
0.0924	0.0918	0.113	0.102	0.105	0.0927	0.125	0.107	0.0948	0.075
0.0924	0.0918	0.113	0.102	0.105	0.0927	0.125	0.107	0.0948	0.075
0.0924	0.0918	0.113	0.102	0.105	0.0927	0.125	0.107	0.0948	0.075

Рисунок 9 – Результирующая матрица доверия

Получим r_f, r_s (см. рисунок 10).

```
(r_f, r_s) = (0.373, 0.418)
```

Рисунок 10 – Полученные значения r_f, r_s

Для целевых функций игроков:

$$\Phi_f(u,v) = a(r_f u + r_s v) - b(r_f u + r_s v)^2 - g_f \frac{u^2}{2}$$

$$\frac{\partial \Phi_f(u,v)}{\partial u} = a r_f - 2b(r_f u + r_s v) r_f - g_f u$$

$$\Phi_s(u,v) = c(r_f u + r_s v) - d(r_f u + r_s v)^2 - g_s \frac{v^2}{2}$$

$$\frac{\partial \Phi_s(u,v)}{\partial v} = c r_s - 2d(r_f u + r_s v) r_s - g_s v$$

Решаем систему:

$$\begin{cases} a r_f - 2b(r_f u + r_s v) r_f - g_f u = 0 \\ c r_s - 2d(r_f u + r_s v) r_s - g_s v = 0 \end{cases}$$

Из первого уравнения:

$$v = \frac{u(-2b r_f^2 - g_f) + a r_f}{2b r_f r_s} \qquad (2)$$

Из второго уравнения:

$$u = \frac{2adr_f r_s^2 + ar_f g_s - 2bcr_f r_c^2}{g_f g_s + 2br_f^2 g_s + 2dr_s^2 g_f} \tag{3}$$

Подставим числа из варианта и получим уравнения, представленные на рисунке 11.

```
Φf(u, v) = -0.639 * u^2 + 0.746 * u + 0.836 * v + -0.312 * u * v + -0.175 v^2
Φs(u, v) = -0.556 * u^2 + 1.492 * u + 1.672 * v + -1.247 * u * v + -2.199 v^2
Φf(u, v)|'_u = -1.278 * u + -0.312 * v + 0.746
Φs(u, v)|'_v = -4.397 * v + -1.247 * u + 1.672
```

Рисунок 11 – Полученные уравнения

Согласно (2) и (3) получим ответы.

$$\begin{cases} u = 0.527 \\ v = 0.231 \end{cases}$$

Точка утопии:

$$X_{max} = 0.48$$

Оптимальные мнения:

$$X_{maxf} = 1$$
$$X_{maxs} = 0.5$$

Найдем расстояния:

$$\Delta_{x_f} = 0.52$$
$$\Delta_{x_s} = 0.02$$

Как мы видим, $\Delta_{x_f} > \Delta x_s$, поэтому выиграл второй игрок.

ЗАКЛЮЧЕНИЕ

В ходе выполнения работы для 10 агентов была случайным образом сгенерирована стохастическая матрица доверия, на основе которой получена результирующая матрица. Затем, используя выборку из случайных агентов, были заданы функции выигрыша, определены целевые функции, найдена точка утопии и получено аналитическое решение игры с неполной противоположностью интересов двух игроков. Для нахождения параметров uu и vv было выполнено дифференцирование целевых функций и решена соответствующая система уравнений.

По сгенерированной матрице доверия был получен результат $X = 0.48$, $\Delta_{x_f} = 0.52$, $\Delta_{x_s} = 0.02$. Так как $\Delta_{x_f} > \Delta x_s$, победу одержал второй игрок.

ПРИЛОЖЕНИЕ А

Класс реализация игр модели информационного противоборства

Листинг А.1 – game.go

```
package informationalwarfare

import (
  "fmt"
  "math"
  "math/rand"
  "sort"

  "gonum.org/v1/gonum/mat"
)

type SolutionType int

const (
  General SolutionType = iota
  TargetFunction

  absMaxInfuence = 100
  diff           = 20
)

type InformationalWarfare struct {
  n, a, b, c, d, gF, gS int
}

func New(n, a, b, c, d, gF, gS int) *InformationalWarfare {
  return &InformationalWarfare{
    n:  n,
    a:  a,
    b:  b,
    c:  c,
    d:  d,
    gF: gF,
    gS: gS,
  }
}
```

```
func (g *InformationalWarfare) Solve(solType SolutionType) {
  m := g.generateTrustMatrix()

  fmt.Printf("%.3v\n", mat.Formatted(m))

  if !g.isStochastic(m) {
    fmt.Println("matrix is not stochastic")
  }

  a, b := 1, 20
  initialOpinions := g.generateInitialOpinions(a, b+1, g.n)

  fmt.Printf("Random vector of agent's opinions: %v\n", mat.
     Formatted(initialOpinions.T()))

  finalOpinions, resTrustMatrix := g.computeFinalOpinions(m,
     initialOpinions, 0.000001)

  fmt.Printf("Final opinions:\n %.3v\nRes trust matrix:\n %.3v\n",
    mat.Formatted(finalOpinions.T()), mat.Formatted(resTrustMatrix)
       )

  player1InfluenceIndices, player2InfluenceIndices := g.
     getPlayerAgentIndices(g.n)

  fmt.Printf("Indices of agents of influence for the first player:
     %v\n", player1InfluenceIndices)
  fmt.Printf("Indices of agents of influence for the second player:
      %v\n", player2InfluenceIndices)

  player1Influence := rand.Intn(absMaxInfuence)
  player2Influence := -rand.Intn(absMaxInfuence)

  fmt.Println("Initial opinion of 1 player's agents: ",
     player1Influence)
  fmt.Println("Initial opinion of 2 player's agents: ",
     player2Influence)

  for _, idx := range player1InfluenceIndices {
    initialOpinions.SetVec(idx, float64(player1Influence))
  }
```

```
  for _, idx := range player2InfluenceIndices {
    initialOpinions.SetVec(idx, float64(player2Influence))
  }

  finalOpinions, resTrustMatrix = g.computeFinalOpinions(m,
     initialOpinions, 0.000001)
  fmt.Printf("After the agents interact, the opinion vector
     converges to the value of X =\n%.3v\n", mat.Formatted(
     finalOpinions.T()))

  switch solType {
  case General:
    g.WithGeneral(finalOpinions, player1Influence, player2Influence
       )
  case TargetFunction:
    g.WithTargetFunction(player1InfluenceIndices,
       player2InfluenceIndices, resTrustMatrix)
  default:
    fmt.Println("Undefined solution type")
  }
}

func (g *InformationalWarfare) WithGeneral(finalOpinions *mat.
   VecDense, player1Influence, player2Influence int) {
  diff1 := math.Abs(finalOpinions.AtVec(0) - float64(
     player1Influence))
  diff2 := math.Abs(finalOpinions.AtVec(0) - float64(
     player2Influence))

  if diff1 < diff2 {
    fmt.Println("First player won")
  } else {
    fmt.Println("Second player won")
  }
}

func (g *InformationalWarfare) WithTargetFunction(
   player1InfluenceIndices, player2InfluenceIndices []int,
  resTrustMatrix *mat.Dense,
) {
  fmt.Printf("Result trust matrix:\n%.3v\n", mat.Formatted(
     resTrustMatrix))
```

```
rF := 0.0
for _, idx := range player1InfluenceIndices {
  rF += resTrustMatrix.At(0, idx)
}

rS := 0.0
for _, idx := range player2InfluenceIndices {
  rS += resTrustMatrix.At(1, idx)
}

fmt.Printf("(r_f, r_s) = (%.3v, %.3v)\n", rF, rS)

targetF := fmt.Sprintf("%.3f * u^2 + %.3f * u + %.3f * v + %.3f *
    u * v + %.3f v^2",
  -float64(g.b)*rF*rF-float64(g.gF)/2,
  float64(g.a)*rF,
  float64(g.a)*rS,
  -float64(2)*float64(g.b)*rF*rS,
  -float64(g.b)*rS*rS,
)

targetS := fmt.Sprintf("%.3f * u^2 + %.3f * u + %.3f * v + %.3f *
    u * v + %.3f v^2",
  -float64(g.d)*rF*rF,
  float64(g.c)*rF,
  float64(g.c)*rS,
  -float64(2)*float64(g.d)*rF*rS,
  -float64(g.d)*rS*rS-(float64(g.gS)/2),
)

fmt.Printf("Φf(u, v) = %s\n", targetF)
fmt.Printf("Φs(u, v) = %s\n", targetS)

targetFDiffU := fmt.Sprintf("%.3f * u + %.3f * v + %.3f",
  -2*float64(g.b)*rF*rF-float64(g.gF),
  -2*float64(g.b)*rS*rF,
  float64(g.a)*rF,
)
targetSDiffV := fmt.Sprintf("%.3f * v + %.3f * u + %.3f",
  -2*float64(g.d)*rS*rS-float64(g.gS),
  -float64(2)*float64(g.d)*rF*rS,
```

```
    float64(g.c)*rS,
  )

  fmt.Printf("Φf(u, v)|'_u = %s\n", targetFDiffU)
  fmt.Printf("Φs(u, v)|'_v = %s\n", targetSDiffV)

  u := (2*rS*rS*float64(g.d)*float64(g.a)*rF + float64(g.a)*rF*
     float64(g.gS) - 2*rF*rS*rS*float64(g.c)*float64(g.b)) /
    (float64(g.gF)*float64(g.gS) + 2*float64(g.b)*rF*rF*float64(g.
       gS) + 2*float64(g.d)*rS*rS*float64(g.gF))
  v := (u*(-2*float64(g.b)*rF*rF-float64(g.gF)) + float64(g.a)*rF)
     / (2 * float64(g.b) * rF * rS)

  fmt.Printf("v = %.3f, u = %.3f\n", v, u)

  X := u*rF + v*rS
  fmt.Printf("X = %.3f\n", X)

  xMaxF, xMaxS := float64(g.a)/float64((2*g.b)), float64(g.c)/
     float64((2*g.d))
  deltaXF := math.Abs(X - float64(xMaxF))
  deltaXS := math.Abs(X - float64(xMaxS))

  fmt.Printf("X_max_f = %.3f, X_max_s = %.3f\n Let's find distance
     :\ndelta_x_f = %.3f, delta_x_s = %.3f\n",
    xMaxF, xMaxS, deltaXF, deltaXS)

  if deltaXF < deltaXS {
    fmt.Println("First player won")
  } else {
    fmt.Println("Second player won")
  }
}

func (g *InformationalWarfare) generateTrustMatrix() *mat.Dense {
  data := make([]float64, g.n*g.n)

  for i := range g.n {
    var rowSum float64
    for j := range g.n {
      val := rand.Float64()
      data[i*g.n+j] = val
```

```
      rowSum += val
    }

    for j := range g.n {
      data[i*g.n+j] /= rowSum
    }
  }

  return mat.NewDense(g.n, g.n, data)
}

func (g *InformationalWarfare) isStochastic(matrix *mat.Dense) bool
    {
  for i := range g.n {
    row := matrix.RawRowView(i)
    var rowSum float64
    for _, val := range row {
      rowSum += val
    }

    if !(rowSum > 0.999999 && rowSum < 1.000001) {
      return false
    }
  }
  return true
}

func (g *InformationalWarfare) generateInitialOpinions(minVal,
   maxVal, N int) *mat.VecDense {
  data := make([]float64, N)
  for i := range N {
    data[i] = float64(rand.Intn(maxVal-minVal+1) + minVal)
  }
  return mat.NewVecDense(N, data)
}

func (g *InformationalWarfare) computeFinalOpinions(trustMatrix *
   mat.Dense,
  initialOpinions *mat.VecDense, epsilon float64,
) (*mat.VecDense, *mat.Dense) {
  n, _ := trustMatrix.Dims()
```

```
  currentOpinions := mat.NewVecDense(n, nil)
  currentOpinions.CopyVec(initialOpinions)
  previousOpinions := mat.NewVecDense(n, nil)

  resTrustMatrix := mat.DenseCopyOf(trustMatrix)

  i := 0

  for {
    difference := mat.NewVecDense(n, nil)
    difference.SubVec(currentOpinions, previousOpinions)
    norm := mat.Norm(difference, 2)
    if norm <= epsilon {
      break
    }

    previousOpinions.CopyVec(currentOpinions)

    currentOpinions.MulVec(trustMatrix, currentOpinions)

    temp := mat.NewDense(n, n, nil)
    temp.Mul(resTrustMatrix, trustMatrix)
    resTrustMatrix = temp

    fmt.Printf("x(%d)=%.4v\n", i, mat.Formatted(currentOpinions.T()
       ))

    i++
  }

  return currentOpinions, resTrustMatrix
}

func (g *InformationalWarfare) getPlayerAgentIndices(nAgents int)
   ([]int, []int) {
  if nAgents < 2 {
    return []int{}, []int{}
  }
  player1Count := rand.Intn(nAgents-1) + 1
  player2Count := rand.Intn(nAgents-player1Count) + 1

  perm := rand.Perm(nAgents)
```

```
  player1Indices := make([]int, player1Count)
  player2Indices := make([]int, player2Count)
  copy(player1Indices, perm[:player1Count])
  copy(player2Indices, perm[player1Count:player1Count+player2Count
     ])

  sort.Ints(player1Indices)
  sort.Ints(player2Indices)
  return player1Indices, player2Indices
}
```

ПРИЛОЖЕНИЕ Б

Точка входа в программу

Листинг Б.2 – main.go

```
package main

import informationalwarfare "github.com/themilchenko/game_theory/
   internal/informational_warfare"

const (
  N           = 10
  a, b, c, d = 2, 1, 4, 4
  gF, gS      = 1, 3
)

func main() {
  g := informationalwarfare.New(N, a, b, c, d, gF, gS)

  g.Solve(informationalwarfare.TargetFunction)
}
```