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)

Лабораторная работа №6 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.
- Назначить всем агентам случайное начальное мнение из заданного отрезка числовой оси. Найти итоговое мнение агентов.
- Случайным образом выбрать количество и номера (непересекающиеся) агентов влияния из общего числа агентов для первого и второго игроков. Назначить им начальные мнения первого и второго игроков. Остальным агентам (нейтральным) назначить случайные начальные мнения. Смоделировать информационное управление в рамках игры и определить итоговое мнение агентов.

ХОД РАБОТЫ

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

Пример запуска программы:

```
go run cmd/lw6/main.go
```

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

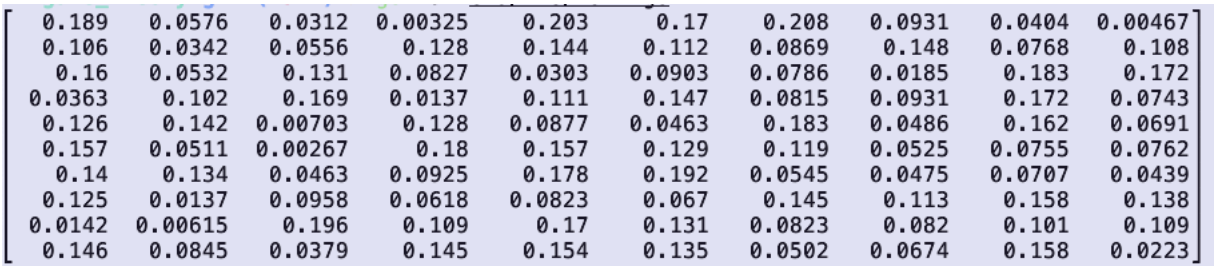

Рисунок 1 – Случайная матрица доверя 10x10

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)=[8.089   7.42  7.484  7.726  6.812  7.615   7.56  7.474  8.346  6.654]
x(1)=[7.527  7.477   7.61  7.539  7.625  7.541  7.516  7.559  7.423  7.636]
x(2)=[7.546  7.555   7.54  7.541  7.522  7.545  7.544  7.545  7.566  7.532]
x(3)=[7.541  7.541  7.546  7.545  7.546  7.542  7.543  7.544   7.54  7.545]
x(4)=[7.543  7.544  7.543  7.543  7.543  7.543  7.543  7.543  7.544  7.543]
x(5)=[7.543  7.543  7.543  7.543  7.543  7.543  7.543  7.543  7.543  7.543]
x(6)=[7.543  7.543  7.543  7.543  7.543  7.543  7.543  7.543  7.543  7.543]
x(7)=[7.543  7.543  7.543  7.543  7.543  7.543  7.543  7.543  7.543  7.543]
x(8)=[7.543  7.543  7.543  7.543  7.543  7.543  7.543  7.543  7.543  7.543]
x(9)=[7.543  7.543  7.543  7.543  7.543  7.543  7.543  7.543  7.543  7.543]
x(10)=[7.543  7.543  7.543  7.543  7.543  7.543  7.543  7.543  7.543  7.543]
```

Рисунок 3 – Итерации примененного алоритма

После взаимодействия агентов, вектор мнений сходится к значению 7.543. Результирующая матрица доверия представлена на рисунке 4.

```
Res trust matrix:
⎡ 0.12  0.0723  0.0724  0.0953  0.137  0.124  0.115  0.0736  0.115  0.0762⎤
⎢ 0.12  0.0723  0.0724  0.0953  0.137  0.124  0.115  0.0736  0.115  0.0762⎥
⎢ 0.12  0.0723  0.0724  0.0953  0.137  0.124  0.115  0.0736  0.115  0.0762⎥
⎢ 0.12  0.0723  0.0724  0.0953  0.137  0.124  0.115  0.0736  0.115  0.0762⎥
⎢ 0.12  0.0723  0.0724  0.0953  0.137  0.124  0.115  0.0736  0.115  0.0762⎥
⎢ 0.12  0.0723  0.0724  0.0953  0.137  0.124  0.115  0.0736  0.115  0.0762⎥
⎢ 0.12  0.0723  0.0724  0.0953  0.137  0.124  0.115  0.0736  0.115  0.0762⎥
⎢ 0.12  0.0723  0.0724  0.0953  0.137  0.124  0.115  0.0736  0.115  0.0762⎥
⎢ 0.12  0.0723  0.0724  0.0953  0.137  0.124  0.115  0.0736  0.115  0.0762⎥
⎣ 0.12  0.0723  0.0724  0.0953  0.137  0.124  0.115  0.0736  0.115  0.0762⎦
```

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: [0 2 3 4 5 6 7 9]
Indices of agents of influence for the second player: [1]
```

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:  56
Initial opinion of 2 player's agents:  -10
```

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)=[ 50.1  49.75  42.97  40.31  38.15   48.7  43.45  46.86  50.35  42.21]
x(1)=[45.45  44.67  45.95  45.81  45.76   44.7  45.58  45.35  43.78  45.66]
x(2)=[45.31  45.35  45.18  45.11  45.14  45.33  45.21  45.26  45.39  45.14]
x(3)=[45.25  45.23  45.25  45.26  45.26  45.23  45.25  45.24  45.22  45.26]
x(4)=[45.25  45.25  45.24  45.24  45.24  45.25  45.24  45.25  45.25  45.24]
x(5)=[45.25  45.24  45.24  45.25  45.24  45.24  45.25  45.24  45.24  45.25]
x(6)=[45.24  45.24  45.24  45.24  45.24  45.24  45.24  45.24  45.24  45.24]
x(7)=[45.24  45.24  45.24  45.24  45.24  45.24  45.24  45.24  45.24  45.24]
x(8)=[45.24  45.24  45.24  45.24  45.24  45.24  45.24  45.24  45.24  45.24]
x(9)=[45.24  45.24  45.24  45.24  45.24  45.24  45.24  45.24  45.24  45.24]
x(10)=[45.24  45.24  45.24  45.24  45.24  45.24  45.24  45.24  45.24  45.24]
x(11)=[45.24  45.24  45.24  45.24  45.24  45.24  45.24  45.24  45.24  45.24]
```

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 =
[45.2  45.2  45.2  45.2  45.2  45.2  45.2  45.2  45.2  45.2]
```

Рисунок 8 – Вектор мнений

Найдем расстояния от итогового мнения до изначальных мнений игроков, чтобы определить победителя.

$$\Delta_1 = |45.24 - u| = |45.24 - 56| = 10.76$$
$$\Delta_2 = |45.24 - v| = |45.24 + 10| = 55.24$$

Так как $\Delta_1 < \Delta_2$, то победил первый игрок.

ЗАКЛЮЧЕНИЕ

В ходе выполнения работы была исследована теоретико-игровая модель информационного противоборства в социальных сетях. Проведено моделирование процесса информационного управления в рамках игры, а также определено итоговое мнение агентов.

Была разработана программа, реализующая генерацию случайной стохастической матрицы доверия, формирование начального вектора мнений агентов, а также алгоритм вычисления результирующих мнений и матрицы доверия после взаимодействия.

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()))
  fmt.Printf("Result trust matrix:\n%.3v\n", mat.Formatted(
     resTrustMatrix))

  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 = 1, 2, 1, 5
  gF, gS      = 1, 2
)

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

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