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 №5 на тему "Рациональный дележ в кооперативных играх"

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

Задание

Для заданной характеристической функцией игры выполнить следующее:

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

Вариант со значениями характерестической функции представлены в таблице 1.

Таблица 1 – Задание характеристической функции

Множество	**Значение** $v(I)$
$\emptyset$	0
$\{1\}$	2
$\{2\}$	3
$\{3\}$	4
$\{4\}$	1
$\{1,2\}$	6
$\{1,3\}$	7
$\{1,4\}$	4
$\{2,3\}$	7
$\{2,4\}$	5
$\{3,4\}$	6
$\{1,2,3\}$	12
$\{1,2,4\}$	9
$\{1,3,4\}$	9
$\{2,3,4\}$	10
$\{1,2,3,4\}$	14

ХОД РАБОТЫ

В проекте был написан класс для решения рационального дележа в кооперативных играх.

Пример запуска программы:

```
go run cmd/lw5/main.go
```

1 Проверка на супераддитивность

Игра является супераддитивной, если

$$\forall\, S, T \subseteq I \quad (S \cap T = \emptyset)\colon\ v(S \cup T) \geq v(S) + v(T). \tag{1}$$

Данная игра является супераддитивной, так как для всех пар множеств выполняется условие (1).

2 Проверка на выпуклость

Игра является выпуклой, если

$$\forall S, T \subseteq I \colon v(S \cup T) + v(S \cap T) \geq v(S) + v(T). \tag{2}$$

Данная игра не является выпуклой. Например, подставим $S = \{1, 2, 3\}$ и $T = \{1, 2, 4\}$ в формулу (2):

$$\begin{gathered} v(\{1,2,3,4\}) + v(\{1,2\}) \overset{?}{\geq} v(\{1,2,3\}) + v(\{1,2,4\}) \\ 14 + 6 \overset{?}{\geq} 12 + 9 \\ 20 < 21 \end{gathered}$$

3 Вектор Шепли

Найдем вектор Шепли, элементы которого выражаются следующим выражением:

$$x_i(v) = \frac{1}{N!} \sum_{S:\, i \in S} (|S| - 1)!\,(N - |S|)!\,\big(v(S) - v(S \setminus \{i\})\big), \tag{3}$$

С помощью программы посчитаем вектор Шепли:

$$X(v) = (3.25, 4.083, 4.75, 1.917)$$

Проверим выполнение условия групповой рационализации, которое задается условием (4).

$$\sum_{i \in I} x_i(v) = v(I) \tag{4}$$

Для данной игры условие выполняется.

Проверим выполнение условия индивидуальной рационализации, которое задается условием (4).

$$x_i(v) \geq v\big(\{i\}\big), \quad i \in I \tag{5}$$

Для данной игры условие выполняется. Доказательство представлено на рисунке 1.

```
x_1=3.250 >= v(1)=2.000
x_2=4.083 >= v(2)=3.000
x_3=4.750 >= v(3)=4.000
x_4=1.917 >= v(4)=1.000
```

Рисунок 1 – Проверка индивидуальной рационализации

ЗАКЛЮЧЕНИЕ

В ходе исследования были успешно решены все поставленные задачи: доказана супераддитивность исходной кооперативной игры, разработан и реализован алгоритм расчёта вектора Шепли, получены его конкретные числовые значения. Проведённая верификация подтвердила выполнение ключевых условий - как индивидуальной, так и групповой рационализации. Это свидетельствует о том, что найденное распределение выигрыша является не только математически корректным, но и удовлетворяет критериям справедливости и стабильности, что делает его приемлемым для всех участников коалиции. Полученные результаты имеют практическую ценность для решения задач оптимального распределения ресурсов между участниками совместных проектов.

ПРИЛОЖЕНИЕ А

Класс реализации рационального дележа в кооперативных играх

Листинг А.1 – game.go

```
package cooperative

import (
  "errors"
  "fmt"
  "math"
)

type CooperativeGame struct {
  players         int
  totalCoalition  []int
  charMapping     map[string]float64
  shapleyVec      []float64
  allCoalitions   [][]int
  coalitionPairs  [][2][]int
}

func New(n int, charValues []float64) (*CooperativeGame, error) {
  if len(charValues) != 1<<n {
    return nil, fmt.Errorf("length of charValues should be %d, got
       %d", 1<<n, len(charValues))
  }

  g := &CooperativeGame{
    players:         n,
    totalCoalition: make([]int, n),
    charMapping:     make(map[string]float64),
  }

  for i := range g.totalCoalition {
    g.totalCoalition[i] = i + 1
  }

  g.generateCoalitions()

  for i, c := range g.allCoalitions {
    key := coalitionKey(c)
```

```
    g.charMapping[key] = charValues[i]
  }

  g.generateCoalitionPairs()

  return g, nil
}

func (g *CooperativeGame) GetShapleyVector() ([]float64, error) {
  if !g.IsSuperadditiveGame() {
    return nil, errors.New("game is not superadditive")
  }
  if g.shapleyVec != nil {
    return g.shapleyVec, nil
  }

  n := g.players
  g.shapleyVec = make([]float64, n)
  totalPerm := factorial(n)

  for i := 1; i <= n; i++ {
    var sum float64

    for _, S := range g.allCoalitions {
      if contains(S, i) {
        Swithout := remove(S, i)
        weight := float64(factorial(len(S)-1)) * float64(factorial(
           n-len(S)))
        vDiff := float64(g.charFunction(S)) - float64(g.
           charFunction(Swithout))
        sum += float64(weight * vDiff)
      }
    }
    g.shapleyVec[i-1] = float64(float64(1)/float64(totalPerm)) *
       sum
  }
  return g.shapleyVec, nil
}

func (g *CooperativeGame) IsGroupRational() bool {
  sum := 0.0
  for _, v := range g.shapleyVec {
```

```
    sum += float64(v)
  }
  totalValue := float64(g.charFunction(g.totalCoalition))

  return math.Abs(sum-totalValue) < 1e-9
}

func (g *CooperativeGame) IsIndividualRational() (bool, [][3]
   float64) {
  res := make([][3]float64, 0)

  for i := 1; i <= g.players; i++ {
    if g.shapleyVec[i-1] < g.charFunction([]int{i}) {
      return false, [][3]float64{{float64(i), g.shapleyVec[i-1], g.
         charFunction([]int{i})}}
    }

    res = append(res, [3]float64{float64(i), g.shapleyVec[i-1], g.
       charFunction([]int{i})})
  }
  return true, res
}

func (g *CooperativeGame) charFunction(c []int) float64 {
  return g.charMapping[coalitionKey(c)]
}

func (g *CooperativeGame) IsSuperadditiveGame() bool {
  for _, pair := range g.coalitionPairs {
    A := toSet(pair[0])
    B := toSet(pair[1])
    if disjoint(A, B) {
      union := toSlice(unionSet(A, B))
      vUnion := g.charFunction(union)
      vA := g.charFunction(pair[0])
      vB := g.charFunction(pair[1])
      if vUnion < vA+vB {
        return false
      }
    }
  }
  return true
```

```
}

func (g *CooperativeGame) IsConvex() (bool, [2][]int) {
  for _, pair := range g.coalitionPairs {
    A := toSet(pair[0])
    B := toSet(pair[1])
    union := toSlice(unionSet(A, B))
    inter := toSlice(intersectionSet(A, B))
    vUnion := g.charFunction(union)
    vInter := g.charFunction(inter)
    vA := g.charFunction(pair[0])
    vB := g.charFunction(pair[1])

    if vUnion+vInter < vA+vB {
      return false, pair
    }
  }
  return true, [2][]int{}
}

func (g *CooperativeGame) generateCoalitions() {
  g.allCoalitions = [][]int{}
  players := g.totalCoalition
  n := len(players)

  for k := 0; k <= n; k++ {
    g.combine(players, k, []int{}, 0)
  }
}

func (g *CooperativeGame) combine(players []int, k int, curr []int,
    start int) {
  if len(curr) == k {
    coal := append([]int{}, curr...)
    g.allCoalitions = append(g.allCoalitions, coal)
    return
  }
  for i := start; i < len(players); i++ {
    g.combine(players, k, append(curr, players[i]), i+1)
  }
}
```

```
func (g *CooperativeGame) generateCoalitionPairs() {
  for i := range len(g.allCoalitions) {
    for j := i + 1; j < len(g.allCoalitions); j++ {
      g.coalitionPairs = append(g.coalitionPairs, [2][]int{g.
         allCoalitions[i], g.allCoalitions[j]})
    }
  }
}
```

Листинг А.2 – game.go

```
package cooperative

import (
  "fmt"
  "sort"
)

func factorial(n int) int {
  if n <= 1 {
    return 1
  }
  return n * factorial(n-1)
}

func coalitionKey(c []int) string {
  cCopy := append([]int(nil), c...)
  sort.Ints(cCopy)
  key := ""
  for _, i := range cCopy {
    key += fmt.Sprintf("%d,", i)
  }
  return key
}

func toSet(c []int) map[int]bool {
  m := make(map[int]bool)
  for _, v := range c {
    m[v] = true
  }
  return m
}
```

```
func toSlice(m map[int]bool) []int {
  s := []int{}
  for k := range m {
    s = append(s, k)
  }
  sort.Ints(s)
  return s
}

func disjoint(a, b map[int]bool) bool {
  for k := range a {
    if b[k] {
      return false
    }
  }
  return true
}

func unionSet(a, b map[int]bool) map[int]bool {
  res := make(map[int]bool)
  for k := range a {
    res[k] = true
  }
  for k := range b {
    res[k] = true
  }
  return res
}

func intersectionSet(a, b map[int]bool) map[int]bool {
  res := make(map[int]bool)
  for k := range a {
    if b[k] {
      res[k] = true
    }
  }
  return res
}

func contains(c []int, x int) bool {
  for _, v := range c {
```

```
    if v == x {
      return true
    }
  }
  return false
}

func remove(c []int, x int) []int {
  newC := []int{}
  for _, v := range c {
    if v != x {
      newC = append(newC, v)
    }
  }
  return newC
}
```

ПРИЛОЖЕНИЕ Б

Точка входа в программу

Листинг Б.3 – main.go

```
package main

import (
  "fmt"
  "log"

  "github.com/themilchenko/game_theory/internal/cooperative"
)

const (
  N = 4
)

var charValues = []float64{0, 2, 3, 4, 1, 6, 7, 4, 7, 5, 6, 12, 9,
   9, 10, 14}

func main() {
  g, err := cooperative.New(N, charValues)
  if err != nil {
    log.Fatal(err)
  }

  isSuper := g.IsSuperadditiveGame()
  fmt.Println("Игра супераддитивна?", isSuper)

  isConvex, values := g.IsConvex()
  fmt.Println("Игра выпуклая?", isConvex)
  if !isConvex {
    fmt.Printf("Нарушение выпуклостимеждукоалициями   %v и %v\n", values
       [0], values[1])
  }

  shapley, err := g.GetShapleyVector()
  if err != nil {
    log.Fatalf("Ошибка прирасчётевектораШепли   : %w", err)
  }
  fmt.Printf("Вектор Шепли: ( ")
```

```
  for _, val := range shapley {
    fmt.Printf("%.3f ", val)
  }
  fmt.Println(")")

  groupRational := g.IsGroupRational()
  fmt.Println("Групповая рационализациявыполнена ?", groupRational)

  individualRational, vals := g.IsIndividualRational()
  fmt.Println("Индивидуальная рационализациявыполнена ?",
     individualRational)
  char := ">="
  if !groupRational {
    char = "<"
  }

  for _, v := range vals {
    fmt.Printf("x_%d=%.3f %s v(%d)=%.3f\n", int(v[0]), v[1], char,
       int(v[0]), v[2])
  }
}
```