Отчёт к лабораторной работе

по дисциплине  
«Теория принятия решений»

выполнил   
студент гр. ИС/б-18-1-з Демиденко А. А.  
принял Кротов К. В.

Лабораторной работа № 2  
«Исследование применения аппарата одномерной полезности для решения задачи выбора альтернатив»

## Цель работы

Исследовать применение аппарата теории полезности при принятии решений по выбору альтернатив.

## Программа выполнения работы

Для варианта задания, связанного с использованием множеств и , предусматривается следующий порядок действий по выполнению лабораторной работы:

1. реализовать объявление и инициализацию матрицы отношений между решениями в соответствии с вариантом задания;
2. реализовать процедуру определения для каждого рассматриваемого решения xn+1 соответствующих ему множеств и , которые определяют для решения xn+1 не худшие по отношению к нему решения (множество ) и не лучшие по отношению к нему решения (множество ); при определении множества необходимо выполнять просмотр (n+1)-го столбца матрицы отношений, при определении множества необходимо выполнять просмотр (n+1)-ой строки матрицы отношений, для рассматриваемого элемента xn+1 выполнить вывод множеств и ;
3. реализовать процедуру выполнения условий (), (), (); тем самым определяется способ вычисления значений функции полезности для решения xn+1; реализовать вывод информации о выполняющемся условии;
4. реализовать процедуру вычисления значения функции полезности для текущего рассматриваемого решения xn+1;
5. реализовать процедуру управления процессом вычисления значений функции полезности для каждого элемента множества Х (решения множества Х); реализовать в рассматриваемой процедуре определение максимального значения функции полезности и соответствующего ему решения; выполнить вывод всех решений и соответствующих им значений функции полезности.

**Вариант 1.** Используя метод, реализующий формирование множеств и , а также их последующий анализ (с точки зрения (), (), ()), выполнить для заданного вида матрицы отношения предпочтения определение значений функции полезности U(xi) решений и определение по формируемым значениям функции полезности эффективных решений . Матрица отношения предпочтения имеет следующий вид:

## Ход работы

Реализуем объявление и инициализацию матрицы отношений между решениями:

type Row a = [a]

type Matrix a = [Row a]

type RowI a = [(Int, a)]

type MatrixI a = [(Int, RowI a)]

input :: Matrix Bool

input =

[ [ False, True, False, False, False, False, False ]

, [ True, False, True, False, False, False, False ]

, [ False, False, False, True, False, False, True ]

, [ False, True, False, False, True, True, False ]

, [ False, True, False, False, False, False, False ]

, [ False, False, True, False, False, True, False ]

]

Реализуем процедуры определения для каждого рассматриваемого решения xn+1 соответствующих ему множеств и :

getColumn :: Matrix a -> Int -> Row a

getColumn matrix n = map (!! (n - 1)) matrix

getRow :: Matrix a -> Int -> Row a

getRow matrix n = matrix !! (n - 1)

calcX :: (Matrix Bool -> Int -> Row Bool) -> Matrix Bool -> Int -> [Int]

calcX getter matrix n

| n < 1 || n > length matrix = []

| otherwise = foldr (\(i, e) res -> if e then i : res else res) [] elements

where

elements = zip [1..] . take n . getter matrix $ n

calcXplus :: Matrix Bool -> Int -> [Int]

calcXplus = calcX getColumn

calcXminus :: Matrix Bool -> Int -> [Int]

calcXminus = calcX getRow

Реализуем проверку условий на пустое множество:

import Data.List (intersect)

calcXplusAndMinus :: Matrix Bool -> Int -> [Int]

calcXplusAndMinus matrix n = calcXplus matrix n `intersect` calcXminus matrix n

checkXplusEmpty :: Matrix Bool -> Int -> Bool

checkXplusEmpty matrix = null . calcXplus matrix

checkXminusEmpty :: Matrix Bool -> Int -> Bool

checkXminusEmpty matrix = null . calcXminus matrix

checkXplusAndXminusEmpty :: Matrix Bool -> Int -> Bool

checkXplusAndXminusEmpty matrix = null . calcXplusAndMinus matrix

Реализуем вычисление функции полезности:

calcU :: Matrix Bool -> Int -> Double

calcU matrix n

| plusEmpty && not minusEmpty = realToFrac 1

| minusEmpty && not plusEmpty = realToFrac $ x' - 1

| not plusEmpty && not minusEmpty && bothEmpty = realToFrac (x' + x'') / 2

| not plusEmpty && not minusEmpty && not bothEmpty = anyX

| otherwise = 0

where

plusEmpty = checkXplusEmpty matrix n

minusEmpty = checkXminusEmpty matrix n

bothEmpty = checkXplusAndXminusEmpty matrix n

x' = calcU matrix . minimum $ calcXplus matrix n

x'' = calcU matrix . maximum $ calcXminus matrix n

anyX = calcU matrix . head $ calcXplusAndMinus matrix n

Реализуем функции для вывода всех значений функции полезности и наилучшего.

findSolution :: Matrix Bool -> [(Double, Int)]

findSolution matrix = zip (map (calcU matrix) [1 .. length matrix]) [1..]

findBestSolution :: Matrix Bool -> (Double, Int)

findBestSolution = maximum . findSolution

## Выводы

В ходе лабораторной работы была рассмотрена задача выбора оптимальных решений из альтернатив при помощи аппарата теории полезности. В частности, был рассмотрен подход, в котором используются множества и . Для решения задачи был написан ряд функций на языке Haskell.