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 № 4  
«Исследование методов решения многокритериальных задач принятия решений на основе построения множества Парето»

## Цель работы

Исследовать способы формирования множества Парето-оптимальных решений и определения эффективных решений в этом множестве.

## Программа выполнения работы

Для первого и третьего вариантов в соответствии с заданием необходимо реализовать следующий порядок действий для выполнения лабораторной работы:

1. Разработать процедуру определения на основе задаваемого множества решений Х и соответствующих им значений критериев f1 и f2 множества Р(Х), представляющего собой Парето-границу Х;
2. разработать процедуру определения координат идеальной точки (, ) (точки утопии);
3. разработать процедуру расчета расстояния до точки утопии для координат текущего рассматриваемого решения xi;
4. разработать процедуру определения эффективного решения x\*, расстояние до которого от идеальной точки (, ) является минимальным.

**Вариант 1.** Требуется для задаваемого множества Х в виде: выполнить определение эффективных решений двухкритериальной задачи выбора с использованием метода идеальной точки. Значения критериев f1 и f2 для соответствующих решений сведены в матрицу, представленную ниже:

## Ход работы

Реализуем объявление и инициализацию матрицы со значениями критериев:

type Row a = [a]

type Matrix a = [Row a]

input :: Matrix Int

input =

[ [ 3, 2 ]

, [ 4, 5 ]

, [ 5, 3 ]

, [ 8, 3 ]

, [ 6, 2 ]

, [ 3, 8 ]

, [ 6, 4 ]

, [ 2, 5 ]

, [ 6, 4 ]

, [ 2, 5 ]

]

Реализуем процедуру определения множества P(X):

import Data.List (delete)

calcP :: Matrix Int -> Row Int

calcP matrix = calculate (zip [1..] matrix) [1 .. length matrix]

where

check a b = all (== True) $ zipWith (>=) a b

calculate [] p = p

calculate ((i, m):mx) p = calculate mx $ foldr (\(j, row) result ->

if check m row then delete j result

else if check row m then delete i result

else result

) p mx

Реализуем расчёт идеальной точки:

import Control.Applicative (ZipList (..))

type Point = (Int, Int)

transpose :: Matrix Int -> Matrix Int

transpose = getZipList . traverse ZipList

calcMax :: Matrix Int -> Point

calcMax matrix = calculate $ transpose matrix

where

calculate (f1:f2:\_) = (maximum f1, maximum f2)

calculate \_ = (0, 0)

Реализуем процедуру расчёта расстояния до идеальной точки:

calcR :: Point -> Point -> Double

calcR (p1\_max, p2\_max) (p1, p2) = sqrt . fromIntegral $

(p1\_max - p1) ^ 2 + (p2\_max - p2) ^ 2

Реализуем поиск решения:

row2point :: Row Int -> Point

row2point (f1:f2:\_) = (f1, f2)

row2point \_ = (0, 0)

findSolution :: Matrix Int -> (Double, Int)

findSolution matrix = minimum $ zip rs [1 ..]

where

rs = map (calcR ideal . row2point) matrix

ideal = calcMax matrix

λ> findSolution input

(4.47213595499958,7)

## Выводы

В ходе лабораторной работы была рассмотрена задача принятия решения при помощи множества Парето, с использованием метода идеальной точки. Для решения задачи был написан ряд функций на языке Haskell.