Лабораторная работа №1

Исследование методов принятия решений в условиях полной неопределенности

Цель работы

Изучить и исследовать методы принятия решений при отсутствии какой-либо информации о связях принимаемых решений и исходов.

Постановка задачи

1. Выполнить определение эффективных решений вручную, для этого сформировать таблицы соответствующего вида, включив в них столбцы для промежуточных (вспомогательных) результатов и столбцы для значений критериев на заключительной стадии принятия решений.
2. Выполнить формирование программного кода соответствующей процедуры определения эффективных решений.
3. Выполнить вывод результатов работы процедуры и сравнить полученные в процедуре результаты с результатами, сформированными аналитически.

Вариант 2.

Одно из предприятий должно определить уровень предложения услуг так, чтобы удовлетворить потребности клиентов в течение предстоящих праздников (уровень предложения услуг соответственно а1, а2, а3, а4. Точное число клиентов не известно, но ожидается, что оно может принять одно из четырех значений: 200, 250, 300 или 350 клиентов (воздействия внешней среды, соответственно, qj). Для каждого из этих возможных значений существует наилучший уровень предложения (с точки зрения возможных затрат). Отклонения от этих уровней приводят к дополнительным затратам либо из-за превышения предложения над спросом, либо из-за неполного удовлетворения спроса. На рисунке 1 приведены потери в тысячах долларов.

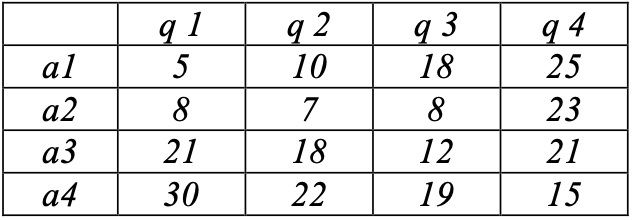


Рисунок 1 - Таблица потерь в тысячах долларов

Задав коэффициент оптимизма равным 0.3, определить эффективное решение. Определить эффективные решения с использованием критерия Гурвица.

Ход работы

Аналитическая часть

Критерий Гурвица является критерием пессимизма - оптимизма. За оптимальную принимается та стратегия, для которой выполняется соотношение:  
max(si)  
где si = y min(aij) + (1-y)max(aij)

При y = 1 получим критерий Вальде, при y = 0 получим – оптимистический критерий (максимакс).  
Критерий Гурвица учитывает возможность как наихудшего, так и наилучшего для человека поведения природы. Как выбирается y? Чем хуже последствия ошибочных решений, тем больше желание застраховаться от ошибок, тем y ближе к 1.

Рассчитываем si.  
s1 = 0.3\*5+(1-0.3)\*25 = 19  
s2 = 0.3\*7+(1-0.3)\*23 = 18.2  
s3 = 0.3\*12+(1-0.3)\*21 = 18.3  
s4 = 0.3\*15+(1-0.3)\*30 = 25.5

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Ai | П1 | П2 | П3 | П4 | min(aij) | max(aij) | y min(aij) + (1-y)max(aij) |
| A1 | 5 | 10 | 18 | 25 | 5 | 25 | 19 |
| A2 | 8 | 7 | 8 | 23 | 7 | 23 | 18.2 |
| A3 | 21 | 18 | 12 | 21 | 12 | 21 | 18.3 |
| A4 | 30 | 22 | 19 | 15 | 15 | 30 | 25.5 |

Выбираем из (19; 18.2; 18.3; 25.5) максимальный элемент max=25.5  
Вывод: выбираем стратегию N=4.  
Таким образом, в результате решения статистической игры по различным критериям чаще других рекомендовалась стратегия A4.

Программаная часть

Для реализации задачи была написана программа на языке C++. Листинг программы представлен ниже.

Листинг 1 – task.cpp

#include <iostream>

#include <stdio.h>

#include <fstream>

using namespace std;

int \*\*readMatrixFromFile(string filename, int rows, int cols);

int \*\*allocate2DArray(int rows, int cols);

void freeArray(int \*\*matrix, int size);

void displayMatrix(int rows, int cols, int \*\*matrix);

double countMax(int currentRow, int cols, int \*\*matrix);

double countMin(int currentRow, int cols, int \*\*matrix);

int main(int argc, char \*argv[])

{

int rowsByDefault = 4;

int colsByDefaut = 4;

double q = 0.3;

double \*z = new double[rowsByDefault];

for (int i = 0; i < rowsByDefault; i++)

{

z[i] = 0;

}

int \*\*matrix = readMatrixFromFile("data.txt", rowsByDefault, colsByDefaut);

cout << endl

<< "Матрица платежей: " << endl << endl;

displayMatrix(rowsByDefault, colsByDefaut, matrix);

for (int i = 0; i < rowsByDefault; i++)

{

double currentMax = countMax(i, colsByDefaut, matrix);

double currentMin = countMin(i, colsByDefaut, matrix);

double Zi = q \* currentMin + (1 - q) \* currentMax;

cout << "Z" << (i + 1) << " = " << Zi << endl;

z[i] = Zi;

}

int maxIndex = 0;

double maxZ = z[maxIndex];

for (int i = 1; i < rowsByDefault; i++)

{

if (z[i] > maxZ)

{

maxZ = z[i];

maxIndex = i;

}

}

cout << endl

<< "Значение Zmax = " << maxZ << endl;

cout << endl

<< "Наилучшей является стратегия № " << (maxIndex + 1) << endl;

freeArray(matrix, rowsByDefault);

delete[] z;

return 0;

}

int \*\*readMatrixFromFile(string filename, int rows, int cols)

{

int \*\*a = allocate2DArray(rows, cols);

ifstream fp(filename);

if (!fp)

{

cout << "Error, file couldn't be opened" << endl;

exit(1);

}

for (int row = 0; row < rows; row++)

{

for (int col = 0; col < cols; col++)

{

fp >> a[row][col];

if (!fp)

{

cout << "Error reading file for element " << row << "," << col << endl;

exit(1);

}

}

}

fp.close();

return a;

}

int \*\*allocate2DArray(int rows, int cols)

{

int \*\*a = new int \*[rows];

for (int i = 0; i < rows; i++)

{

a[i] = new int[cols];

}

for (int i = 0; i < rows; i++)

{

for (int j = 0; j < cols; j++)

{

a[i][j] = 0;

}

}

return a;

}

void displayMatrix(int rows, int cols, int \*\*matrix)

{

for (int i = 0; i < rows; i++)

{

for (int j = 0; j < cols; j++)

{

cout << matrix[i][j] << " ";

}

cout << endl;

}

cout << endl;

}

void freeArray(int \*\*matrix, int size)

{

for (int i = 0; i < size; i++)

{

delete[] matrix[i];

}

delete[] matrix;

}

double countMax(int currentRow, int cols, int \*\*matrix)

{

double max = matrix[currentRow][0];

for (int i = 1; i < cols; i++)

{

if (matrix[currentRow][i] > max)

{

max = matrix[currentRow][i];

}

}

return max;

}

double countMin(int currentRow, int cols, int \*\*matrix)

{

double min = matrix[currentRow][0];

for (int i = 1; i < cols; i++)

{

if (matrix[currentRow][i] < min)

{

min = matrix[currentRow][i];

}

}

return min;

}

Результаты выполнения программы

На рисунке 2 представлен результат работы программы с заданной матрицей, соответствующей условию. Видно, что результаты совпадают с аналитическими расчётами, что говорит о правильности работы алгоритма.

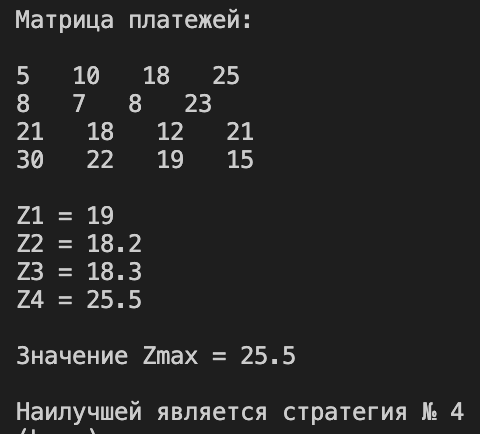


Рисунок 2 - Результат выполнения программы

Выводы

В ходе лабораторной работы были изучены и исследованы методы принятия решений при отсутствии какой-либо информации о связях принимаемых решений и исходов. Была составлена программа, решающая поставленную задачу.