Министерство науки и Высшего образования Российской Федерации
Севастопольский государственный университет
Кафедра ИС

Отчет
по лабораторной работе №2
«Исследование методов принятия решений в условиях стохастической неопределенности»
по дисциплине
«ПОДДЕРЖКА ПРИНЯТИЯ РЕШЕНИЙ В УСЛОВИЯХ НЕОПРЕДЕЛЕННОСТИ»

Выполнил студент группы ИС/б-17-2-о
Горбенко К. Н.
Проверил
Кротов К.В.

Севастополь
2021

1 ЦЕЛЬ РАБОТЫ

Изучить и исследовать методы принятия решений при наличии информации о стохастической связи между экспериментами и их исходами, между принимаемыми решениями и их результатами.

2 ЗАДАНИЕ НА РАБОТУ

Для дерева принятия решений, представленного на 1, и соответствующих этому дереву распределений вероятностей, выполнить определение эффективных стратегий проведения эксперимента и принятия решений с использованием метода анализа дерева решений в экстенсивной форме. Таблицы распределений вероятностей, состав множеств экспериментов, исходов, решений и состояний системы сформировать самостоятельно в соответствии с видом дерева.

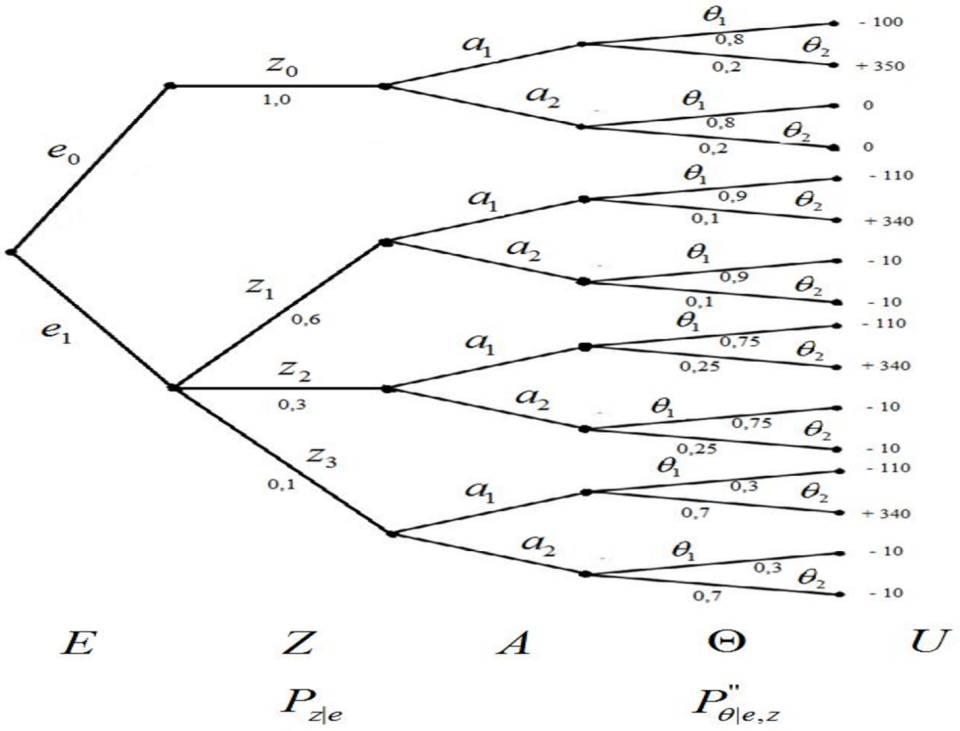

Рисунок 1 – Дерево принятия решений

3 ХОД РАБОТЫ

Текст программы:

```
const int P_TETA_COUNT = 1;
const int A_COUNT = 2;
const int TETA_COUNT = 2;
const int Z_COUNT = 4;
const int E_COUNT = 2;
void createMatrix(float** &matrix, int row, int col);
void printMatrix(float** matrix, int row, int col);
void readFile_Matrix(string name, int n, int numCollmus,float** matrix);
int main()
{float** P_TetaMatrix;float** P_Z_EMatrix;float** P_Teta_ZMatrix;float****
    Umatrix;
char choice;string P_TetaFile="D:\\1.txt";string P_Z_EFile="D:\\2.txt";
string P_Teta_ZFile="D:\\3.txt";ifstream UFile("D:\\4.txt");
createMatrix(P_TetaMatrix, TETA_COUNT, P_TETA_COUNT);
createMatrix(P_Z_EMatrix, Z_COUNT, E_COUNT);
createMatrix(P_Teta_ZMatrix, TETA_COUNT, Z_COUNT);
    UMatrix = new float***[E_COUNT];
    for (int i = 0; i < E_COUNT-1; i++) {
        UMatrix[i] = new float**[Z_COUNT];
        for (int j = 0; j < Z_COUNT; j++){
            UMatrix[i][j] = new float*[A_COUNT];
            for (int l = 0; l < A_COUNT; l++)   {
                UMatrix[i][j][l] = new float[TETA_COUNT];   }   }   }
    readFile_Matrix(P_TetaFile, TETA_COUNT, P_TETA_COUNT,P_TetaMatrix);
    readFile_Matrix(P_Z_EFile, Z_COUNT, E_COUNT,P_Z_EMatrix);
    readFile_Matrix(P_Teta_ZFile, TETA_COUNT, Z_COUNT,P_Teta_ZMatrix);
    for (int i = 0; i < E_COUNT-1; i++)
    {
        for (int j = 0; j < Z_COUNT; j++)
        {
            for (int l = 0; l < A_COUNT; l++)
            {
                for (int k = 0; k < TETA_COUNT; k++)
                {
                    UFile >> UMatrix[i][j][l][k];
                }   }}}
    cout << "P_Teta matrix" << endl;
    printMatrix(P_TetaMatrix, TETA_COUNT, P_TETA_COUNT);
    cout << "P_Z_E matrix" << endl;
    printMatrix(P_Z_EMatrix, Z_COUNT, E_COUNT);
    cout << "P_Teta_Z matrix" << endl;
        printMatrix(P_Teta_ZMatrix, TETA_COUNT, Z_COUNT);
    cout << "U matrix" << endl;
        for (int i = 0; i < E_COUNT - 1; i++)
        {
```

```
45              for (int j = 0; j < Z_COUNT; j++)
46              {
47                  for (int k = 0; k < A_COUNT; k++)
48                  {
49                      for (int l = 0; l < TETA_COUNT; l++)    {
50                          cout << UMatrix[i][j][k][l] << " "      }}}}
51          cout << endl;
52      //1
53      float* U_EZA_massive = new float[Z_COUNT*A_COUNT];
54      for (int i = 0; i < Z_COUNT*A_COUNT; i++)
55      {   U_EZA_massive[i] = 0;}
56      for (int h = 0; h < E_COUNT-1; h++)
57      {   for (int i = 0, l = 0; i < Z_COUNT; i++)
58          {   for (int j = 0; j < A_COUNT; j++)   {
59                  for (int k = 0; k < TETA_COUNT; k++)        {
60                      U_EZA_massive[l] += UMatrix[h][i][j][k] * P_Teta_ZMatrix[k
                            ][i];
61                  }l++;   }}} cout << "U(e,z,a): " << endl;
62      for (int i = 0; i < Z_COUNT*A_COUNT; i++)   {
63          cout << U_EZA_massive[i] << " ";}cout << endl;
64      //2
65      float* U_EZ_massive = new float[Z_COUNT];
66      for (int i = 0; i < Z_COUNT; i++){
67          U_EZ_massive[i] = 0;}
68      float max;
69      for (int i = 0, j=0; i < Z_COUNT; i++)  {
70          max = U_EZA_massive[j];
71          for (int l = 0; l < TETA_COUNT; l++)    {
72              if (U_EZA_massive[l+j] > max)
73              {           max = U_EZA_massive[l+j];
74              }       }       U_EZ_massive[i] = max;
75          j += A_COUNT;   }
76      cout << "U(e,z): " << endl;
77      for (int i = 0; i < Z_COUNT; i++)   {
78          cout << U_EZ_massive[i] << " ";}
79      cout << endl;
80      //3
81      float* U_E_massive = new float[E_COUNT];
82      for (int i = 0; i < E_COUNT; i++)
83      {   U_E_massive[i] = 0;}
84      U_E_massive[0] = U_EZ_massive[0];
85      for (int i = 1, l = 1 ; i < E_COUNT; i++)
86      {   for (int k = 1; k < Z_COUNT; k++)
87          {
88              U_E_massive[i] += U_EZ_massive[k] * P_Z_EMatrix[k][i];
89          }   }   cout << "U(e): " << endl;
90      for (int i = 0; i < E_COUNT; i++)   {
91          cout << U_E_massive[i] << " ";  }
```

```
92      cout << endl;
93      //4
94      float best_solution = U_E_massive[0];
95      for (int i = 1; i < E_COUNT; i++)   {
96          if (U_E_massive[i] > best_solution) {
97              best_solution = U_E_massive[i]; }   }
98      cout << "\n U* = " << best_solution<<endl;
99      int a_number=0;
100     int e_number=0;
101     for (int i = 0; i < Z_COUNT*A_COUNT; i++){
102         if (U_EZA_massive[i] == best_solution)
103         {   a_number = i + 1;}
104         if (U_E_massive[i] == best_solution)        {
105             e_number = i;
106         }   }   cout << "answer: e"<<e_number<<", a"<<a_number<< endl;
107     cin.get();getchar();return 0;}
108 void createMatrix(float** &matrix, int row, int col)
109 {matrix = new float*[row];
110     for (int i = 0; i < row; i++){
111         matrix[i] = new float[col];}}
112 void printMatrix(float** matrix, int row, int col){
113     for (int i = 0; i < row; i++){
114         for (int j = 0; j < col; j++){
115             cout << matrix[i][j] << " ";}
116         cout << endl;}cin.get();}
117 void readFile_Matrix(string name, int n, int numCollums, float** matrix) {
118     FILE *inp; int i,j;
119         if ((inp = fopen(name.c_str(), "r")) == NULL){
120         cout<< "Error by open" <<endl;
121         return;}
122     else
123     for (i = 0; i<n; i++)       for (j = 0; j<numCollums; j++)
124         fscanf(inp, "\%f", &matrix[i][j]);      fclose(inp);
125 }
```

```
P_Teta matrix
0.55
0.45

P_Z_E matrix
1 0
0 0.6
0 0.3
0 0.1

P_Teta_Z matrix
0.8 0.9 0.75 0.3
0.2 0.1 0.25 0.7

U matrix
-100 350 0 0 -110 340 -10 -10 -110 340 -10 -10 -110 340 -10 -10
U(e,z,a):
-10 0 -65 -10 2.5 -10 205 -10
U(e,z):
0 -10 2.5 205
U(e):
0 15.25

 U* = 15.25
answer: e1, a0
```

Рисунок 2 – Результат работы программы

ВЫВОДЫ

В ходе выполнения лабораторной работы исследовали методы принятия решений при наличии информации о стохастической связи между экспериментами и их исходами, между принимаемыми решениями и их результатами. По дереву принятия решений заполнили таблицы вероятностей, написали программу, обрабатывающую таблицы и определяющую эффективное решение.