Министерство образования и науки Российской Федерации

Севастопольский государственный университет

Кафедра ИС

Отчет

По дисциплине: “Теория принятия решений”

Лабораторная работа №2

“Исследование применения аппарата теории одномерной

полезности для решения задач выбора альтернатив”

Выполнил:

ст.гр. ИС/б-32

Долженко И.А.

Проверил:

Дрозин А.Ю.

Севастополь

2020

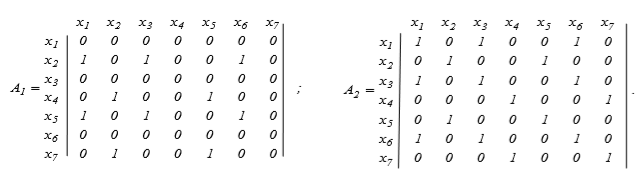
1 ЦЕЛЬ РАБОТЫ

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

2 ПОСТАНОВКА ЗАДАЧИ

Вариант 2

Используя метод, реализующий формирование классов эквивалентности  (), формирование множества /~ неповторяющихся классов эквивалентности , выполнить разработку программы, определяющей значения функции полезности  для этих классов и значения функции  для решений , с последующим определением эффективных решений, для которых . Вид матриц отношений предпочтения и эквивалентности следующий:



3 ХОД РАБОТЫ

Эквивалентность решений на множестве  определяет его разбиение на непересекающиеся непустые классы элементов. Через  обозначим класс эквивалентности, порожденный элементом . Тогда определение  будет выполнено следующим образом:

 и ,

где  - отношение эквивалентности ~.

Сформируем классы эквивалентности.

Т.к. реализация рассматриваемого алгоритма предполагает упорядочивание классов эквивалентности (множеств ), то совпадающие классы могут не рассматриваться.

В результате после выполненных преобразований получим множество /~ в виде:

Определим упорядоченное множество /~ :

k3 -> k2 -> k1

Теперь найдем значения функции полезности для классов эквивалентности. Для этого воспользуемся следующий алгоритмом:

1) рассматривается некоторый «текущий» класс эквивалентности  в предположении, что всем «предшествующим» *(m-1)*-ому классу эквивалентности присвоены значения .

2) для рассматриваемого -го класса эквивалентности возможна одна из трех рассматриваемых ниже ситуаций:

а)  для всех  (понятно, что отношение  вытекает из отношения , где ,), в этом случае ;

б)  для всех  (аналогично отношение  вытекает из отношения , где , ); в этом случае ;

в)  для некоторых  и , и ни для какого  и отличного от  и  не выполняется , тогда значение  принимается равным первому в перечислении  числу  такому, что .

Примем значение функции полезности для класса k1 U(k1) = 0.

Перейдем к классу k2. Классу k2 предшествует класс k1. k2  k1. Это соответствует ситуации "а" алгоритма, поэтому U(k2) = 2.

Перейдем к классу k3. Ему предшествуют классы k1 и k2. k3  k2. k3  k1, так как k3  k2, k2  k1, из чего следует, что k3  k1, так как отношение предпочтения транзитивно по полезности. k3 более предпочитаемый, чем k1 и k2 , поэтому U(k3 ) = 3.

Теперь значения функции полезности для решений могут быть инициализированы значениями функции полезности для классов, в которые они входят.

U(x1 = x3  = x6 ) = 0

U(x2 = x5 ) = 2

U(x4 = x7 ) = 3

Отсюда следует, что эффективными решениями с точки зрения полезности являются x4 и x7.

Результат выполнения программы, реализующей данный алгоритм, представлен на рисунке 1 и 2.

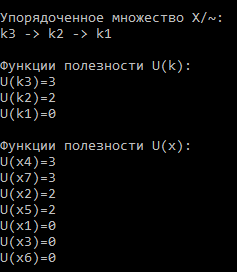
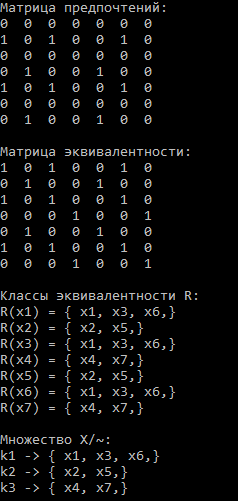


Рисунок 2 – Результат выполнения. Часть 2

Рисунок 1 – Результат выполнения. Часть 1

4. КОД ПРОГРАММЫ

#include <iostream>

#include <iomanip>

using namespace std;

void main()

{

const int n = 7;

int A1[n][n] = {

{ 0, 0, 0, 0, 0, 0, 0 },

{ 1, 0, 1, 0, 0, 1, 0 },

{ 0, 0, 0, 0, 0, 0, 0 },

{ 0, 1, 0, 0, 1, 0, 0 },

{ 1, 0, 1, 0, 0, 1, 0 },

{ 0, 0, 0, 0, 0, 0, 0 },

{ 0, 1, 0, 0, 1, 0, 0 }

};

int A2[n][n] = {

{ 1, 0, 1, 0, 0, 1, 0 },

{ 0, 1, 0, 0, 1, 0, 0 },

{ 1, 0, 1, 0, 0, 1, 0 },

{ 0, 0, 0, 1, 0, 0, 1 },

{ 0, 1, 0, 0, 1, 0, 0 },

{ 1, 0, 1, 0, 0, 1, 0 },

{ 0, 0, 0, 1, 0, 0, 1 }

};

int R[n][n];

int K[n][n];

int unique[n] = { 1, 1, 1, 1, 1, 1, 1 };

int count = 0;

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

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

R[i][j] = A2[i][j];

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

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

K[i][j] = 0;

cout << "Матрица предпочтений:" << endl;

for (int i = 0; i < n; i++) {

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

cout << A1[i][j] << " ";

cout << endl;

}

cout << endl;

cout << "Матрица эквивалентности:" << endl;

for (int i = 0; i < n; i++) {

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

cout << A2[i][j] << " ";

cout << endl;

}

cout << endl;

cout << "Классы эквивалентности R:" << endl;

for (int i = 0; i < n; i++) {

cout << "R(x" << (i + 1) << ") = {";

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

if (R[i][j] == 1)

cout << " x" << (j + 1) << ",";

cout << "}" << endl;

}

cout << endl;

for (int i = 0; i < n; i++) {

if (unique[i] == 1) {

for (int j = 0; j < n; j++) {

K[count][j] = R[i][j];

if ((j != i) && (R[i][j] == 1))

unique[j] = 0;

}

count++;

}

}

cout << "Множество X/~:" << endl;

for (int i = 0, l = 1; i < n; i++) {

if (unique[i] == 1) {

cout << "k" << l << " -> {";

l++;

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

if (R[i][j] == 1)

cout << " x" << (j + 1) << ",";

cout << "}" << endl;

}

}

cout << endl;

int rate\_unique[n] = { 0, 0, 0, 0, 0, 0, 0 };

int sum = 0;

int cnt = count - 1;

while (cnt >= 0) {

for (int i = 0; i < count; i++) {

sum = 0;

for (int j = 0; j < n; j++) {

if (K[i][j] == 1)

for (int t = 0; t < n; t++)

sum = sum + A1[j][t];

}

if (sum == 0) {

rate\_unique[cnt] = i + 1;

cnt--;

for (int j = 0; j < n; j++) {

if (K[i][j] == 1) {

for (int t = 0; t < n; t++) A1[j][t] = 1;

for (int t = 0; t < n; t++) A1[t][j] = 0;

}

}

}

}

}

cout << "Упорядоченное множество X/~:" << endl;

for (int k = 0; k < count; k++) {

if (k) cout << " -> ";

cout << "k" << rate\_unique[k];

}

cout << endl << endl;

int\*\* preference = new int\* [count];

for (int co = 0; co < count; co++)

preference[co] = new int[count];

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

for (int l = 0; l < count; l++)

preference[i][l] = 0;

cnt = count;

for (int i = count - 1; i >= 0; i--) {

for (int j = cnt; j < count; j++)

preference[rate\_unique[i] - 1][rate\_unique[j] - 1] = 1;

cnt--;

}

float u\_ot\_k[n] = { 0, 0, 0, 0, 0, 0, 0 };

for (int k = count - 2; k >= 0; k--) {

if (rate\_unique[k] < rate\_unique[k - 1])

u\_ot\_k[rate\_unique[k] - 1] = rate\_unique[k];

else

u\_ot\_k[rate\_unique[k] - 1] = -rate\_unique[k];

if (k == 0) {

u\_ot\_k[rate\_unique[k] - 1] = rate\_unique[k];

break;

}

else if ((rate\_unique[k] > rate\_unique[k - 1] && rate\_unique[k] > rate\_unique[k + 1]))

u\_ot\_k[rate\_unique[k] - 1] = float(rate\_unique[k - 1]) + (float(rate\_unique[k + 1]) - float(rate\_unique[k - 1])) / 2;

}

cout << "Функции полезности U(k):" << endl;

for (int k = 0; k < count; k++)

cout << "U(k" << (rate\_unique[k]) << ")=" << u\_ot\_k[rate\_unique[k] - 1] << endl;

cout << endl;

cout << "Функции полезности U(x):" << endl;

for (int k = 0; k < count; k++)

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

if (K[rate\_unique[k] - 1][j] == 1)

cout << "U(x" << j + 1 << ")=" << u\_ot\_k[rate\_unique[k] - 1] << endl;

cout << endl;

for (int co = 0; co < count; co++)

delete[] preference[co];

}

ВЫВОД

В ходе данной лабораторной работы было исследовано применение аппарата теории полезности при принятии решений по выбору альтернатив. Программно с использованием технологии .NET был реализован метод, формирующий множество неповторяющихся классов эквивалентности. Были определены значения функции полезности для классов эквивалентности и для решений, принадлежащих данным классам. Были найдены эффективные решения x4 и x7.

ОТВЕТЫ НА КОНТРОЛЬНЫЕ ВОПРОСЫ

**1. Чем вызвана необходимость использования аппарата функции полезности?**

В случаях, когда невозможно определить предпочитаемые решения с использованием функции выбора и максимальные решения.

**2. В чем состоят условия существования функции полезности, определяемой на множестве решений?**

Количество решений счетное и конечное.

**3. Какова связь между бинарным отношением для пары решений и значениями их функции полезности?**

xi>xj <=> U(xi)>U(xj)

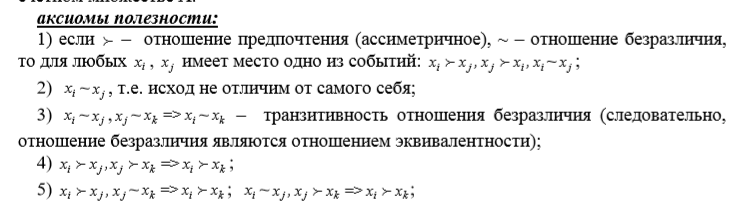
**4. В чем заключается слабый порядок на множестве решений Х?**

Слабый порядок — это когда присутствует эквивалентность либо неопределенность в отношении решений.

**5. В чем заключается строгий порядок на множестве решений Х?**

Строгий порядок — это когда отсутствует эквивалентность либо неопределенность в отношении решений.

**6. В чем состоят аксиомы теории полезности и каков их смысл?**



**7. Каков алгоритм процедуры определения значений функции полезности с использованием множеств доминируемых и доминирующих решений?**

Алгоритм из 1 варианта (x+, x-, итерации).

**8. Какие особенности задания отношений на множестве решений Х позволяет учесть введение классов эквивалентности (какой вид отношения на множестве Х позволяет исключить введение классов эквивалентности)?**

Каждая вершина должна иметь хотя бы одно отношение эквивалентности.

**9. Каким образом реализуется связывание классов эквивалентности с использованием отношения предпочтения для классов?**

Определяется доминированием любой вершины первого класса над любой вершиной второго класса. (этап из 2 и 3 лабы, после построения X/~)

**10. Какое условие должно быть выполнено для существования функции полезности на множестве классов эквивалентности?**

Должна существовать матрица отношений строго предпочтения.

**11. В чем заключается алгоритм определения значений функции полезности для классов эквивалентности?**

Определяется доминированием любой вершины первого класса над любой вершиной второго класса. (этап из 2 и 3 лабы, после построения X/~)

**12. Каким образом определяются значения функции полезности для решений, если значения функции полезности для классов эквивалентности известны?**

Значение функции полезности решения класса равно значению функции полезности этого класса.