Министерство образования и науки Российской Федерации

Севастопольский государственный университет

Кафедра ИС

Отчет

По дисциплине: “Теория принятия решений”

Лабораторная работа №4

“Исследование методов решения многокритериальных задач принятия

решений на основе построения множества Парето”

Выполнил:

ст.гр. ИС/б-17-2-о

Черняев Н.Г.

Проверил:

Кротов К.В.

Севастополь

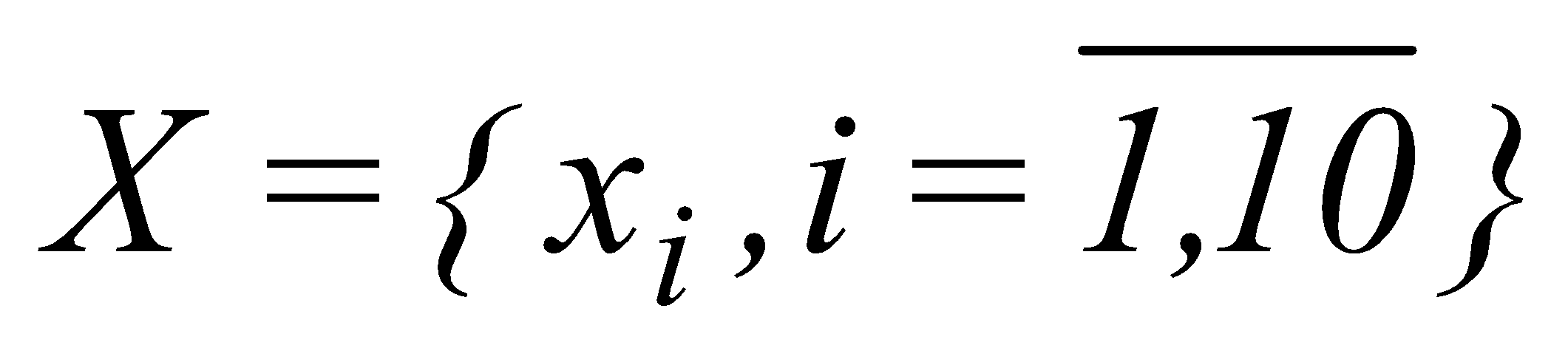
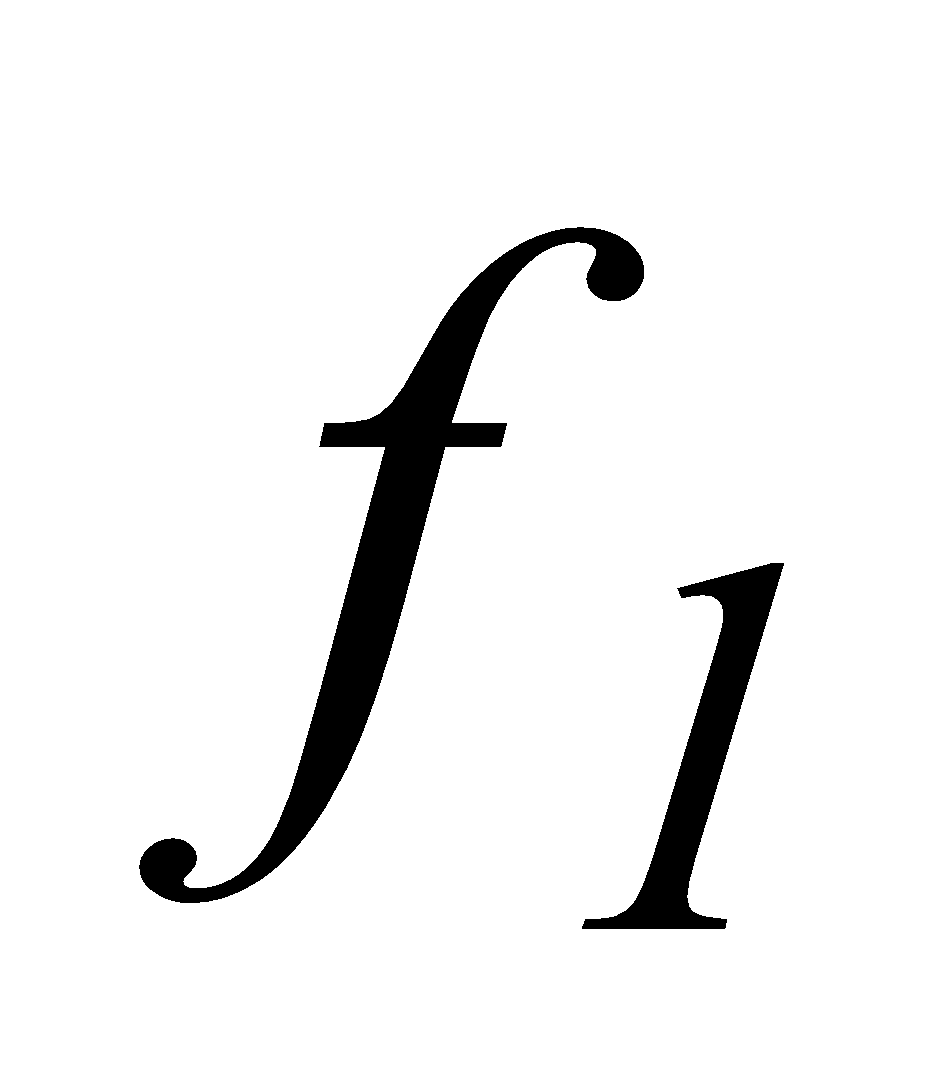
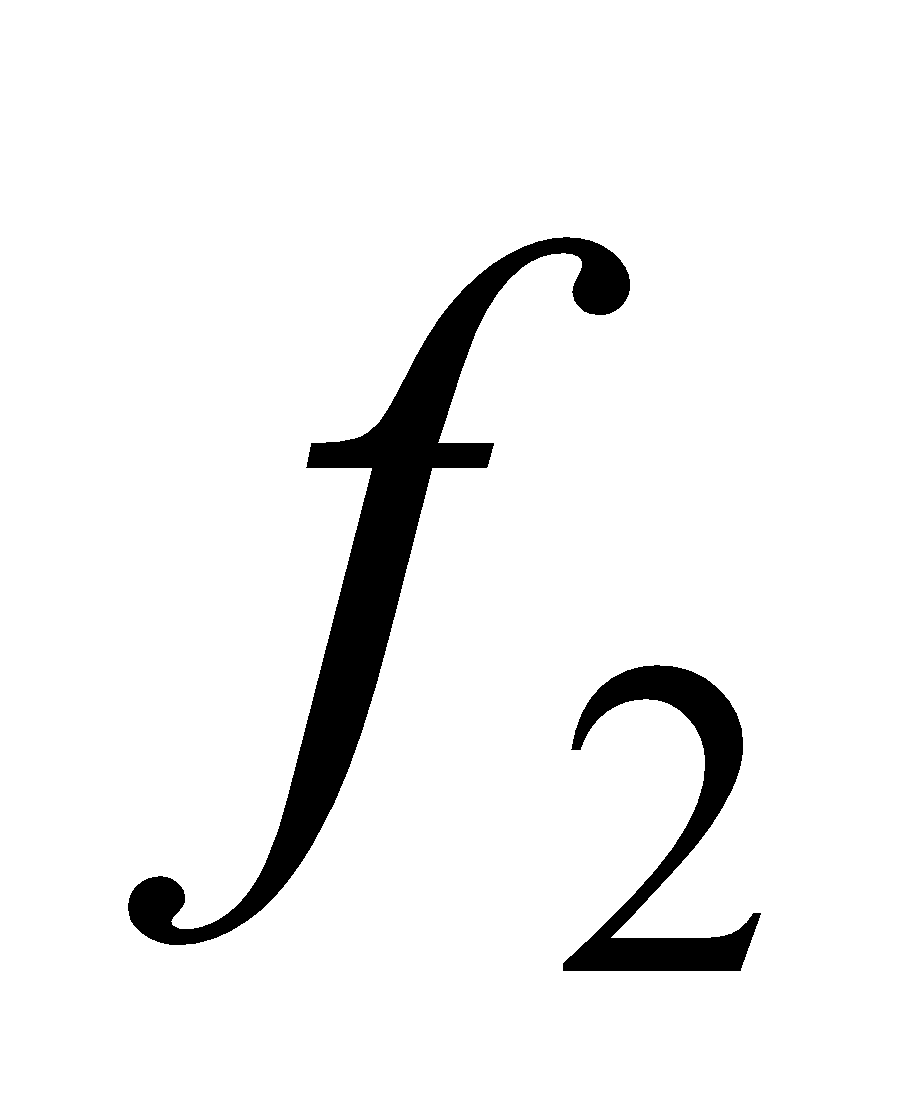
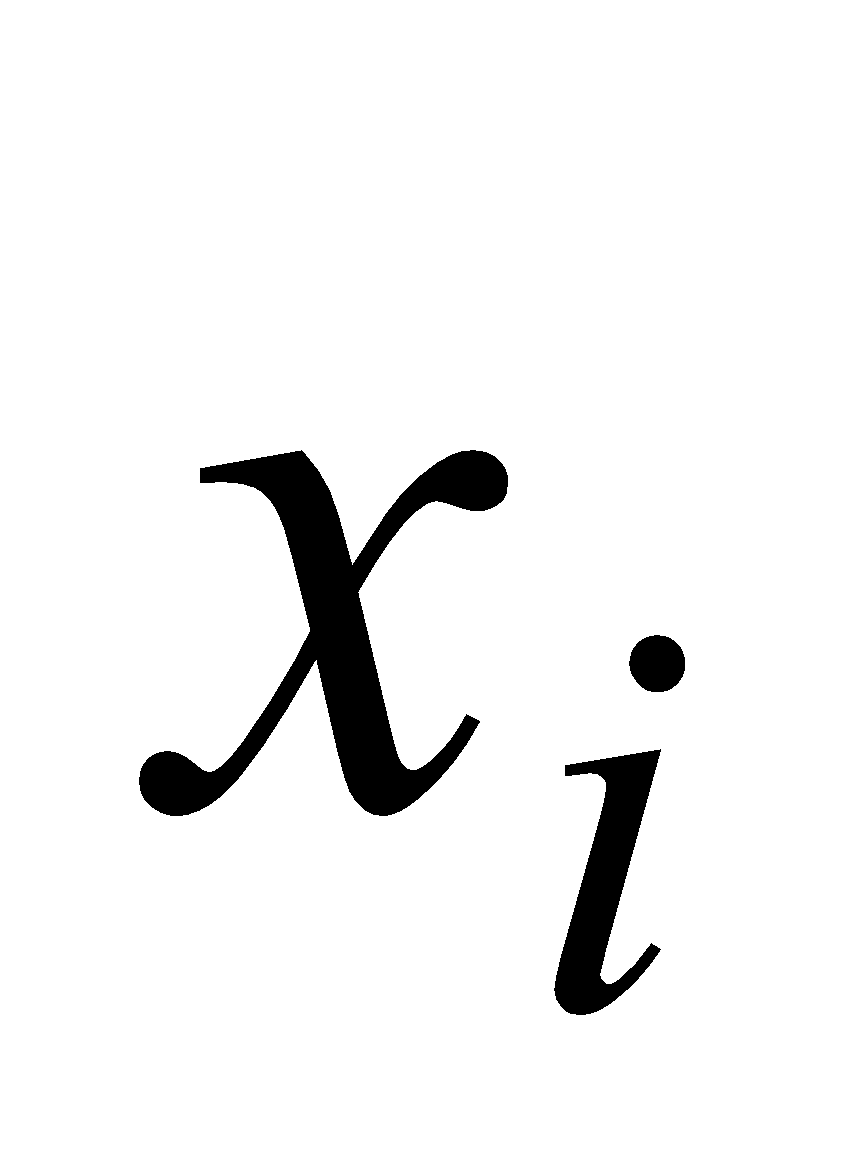
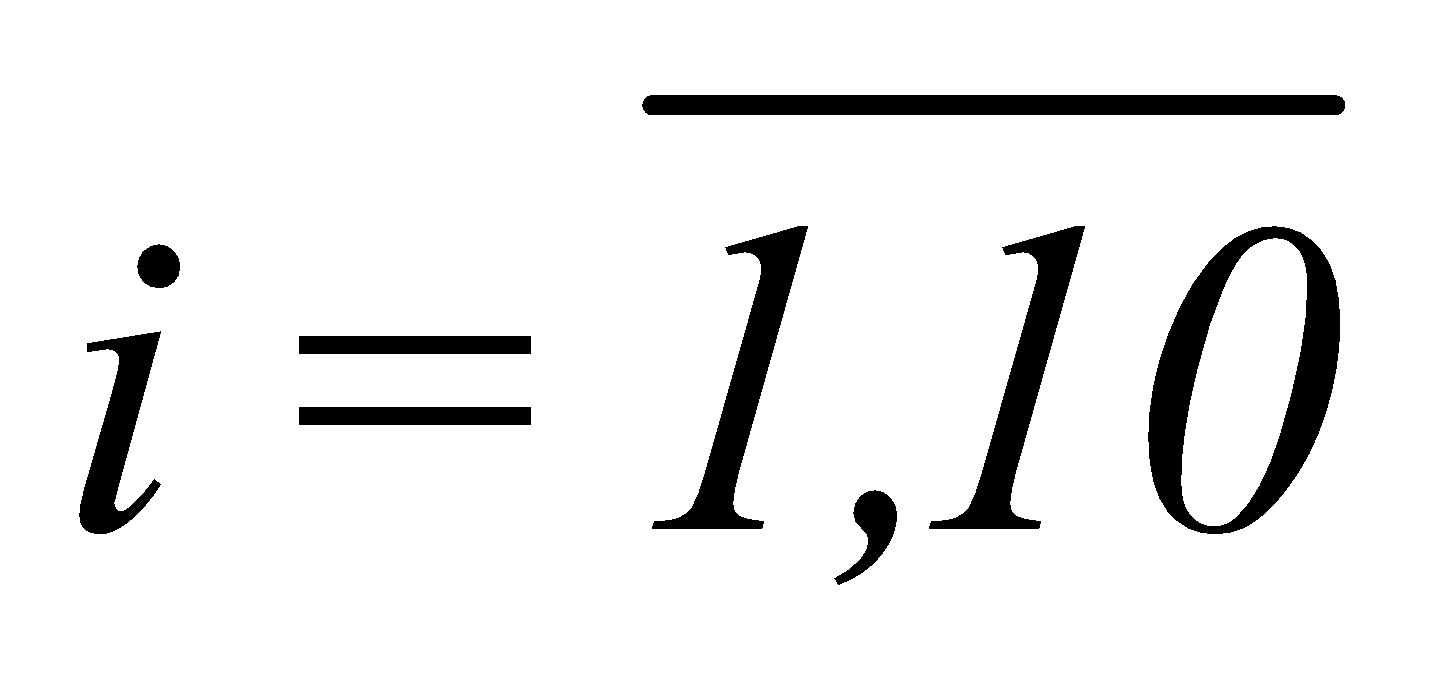
2020

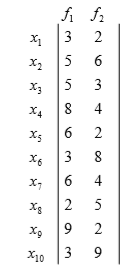
1 ЦЕЛЬ РАБОТЫ

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

2 ПОСТАНОВКА ЗАДАЧИ

Вариант 2

Требуется для задаваемого множества Х в виде:  выполнить определение эффективных решений двухкритериальной задачи выбора с использованием метода последовательных уступок. Значения критериев  и  для соответствующих решений  () сведены в матрицу, представленную ниже.



3 КОД ПРОГРАММЫ

#include <iostream>

using namespace std;

const int n\_f = 2;

const int n\_x = 10;

int in\_array[n\_x][n\_f] = {

{3, 2},

{5, 6},

{5, 4},

{8, 4},

{6, 2},

{3, 8},

{6, 4},

{2, 5},

{9, 2},

{3, 9}

};

int\*\* out\_array, \*\* array\_f1, \*\* array\_f2;

int delta[n\_f] = { 0, 0 };

int index[n\_f] = { 1, 1 };

int result[n\_f];

void find\_dominated(int n\_x, int n\_f, int\* matrix);

int\*\* fill\_out\_array(int n\_x, int n\_f, int x, int\* matrix);

unsigned int count\_not\_dominated(int n\_x, int\* matrix);

void sort(int ind, int n\_x, int n\_f, int\*\* matrix);

int main()

{

setlocale(LC\_ALL, "Russian");

find\_dominated(n\_x, n\_f, &in\_array[0][0]);

unsigned int x = count\_not\_dominated(n\_x, &in\_array[0][0]);

out\_array = fill\_out\_array(n\_x, n\_f, x, &in\_array[0][0]);

// вспомогательные массивы для метода уступок

array\_f1 = new int\* [x];

for (unsigned int i = 0; i < x; i++)

array\_f1[i] = new int[n\_f];

array\_f2 = new int\* [x];

for (unsigned int i = 0; i < x; i++)

array\_f2[i] = new int[n\_f];

// заполним их

for (unsigned int i = 0; i < x; i++)

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

array\_f1[i][j] = array\_f2[i][j] = out\_array[i][j];

// отсортируем их

sort(0, x, n\_f, array\_f1);

sort(1, x, n\_f, array\_f2);

// выведем все наши матрицы

cout << "Парето - граница:" << endl;

for (unsigned int i = 0; i < x; i++) {

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

cout << out\_array[i][j] << " ";

cout << endl;

}

cout << "f1max:" << endl;

for (unsigned int i = 0; i < x; i++) {

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

cout << array\_f1[i][j] << " ";

cout << endl;

}

cout << "f2max:" << endl;

for (unsigned int i = 0; i < x; i++) {

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

cout << array\_f2[i][j] << " ";

cout << endl;

}

// метод уступок

while (true) {

if (array\_f1[index[0]][0] == array\_f2[index[1]][0] and array\_f1[index[0]][1] == array\_f2[index[1]][1]) {

result[0] = array\_f1[index[0]][0];

result[1] = array\_f1[index[0]][1];

break;

}

if (array\_f1[index[0] - 1][0] == array\_f2[index[1]][0] and array\_f1[index[0]][1] == array\_f2[index[1] - 1][1]) {

delta[0] += (array\_f1[index[0] - 1][0] - array\_f1[index[0]][0]);

delta[1] += (array\_f2[index[1] - 1][1] - array\_f2[index[1]][1]);

if (delta[0] > delta[1]) {

result[0] = array\_f1[index[0] - 1][0];

result[1] = array\_f1[index[0] - 1][1];

break;

}

if (delta[0] < delta[1]) {

result[0] = array\_f2[index[0] - 1][0];

result[1] = array\_f2[index[0] - 1][1];

break;

}

result[0] = array\_f1[index[0] - 1][0];

result[1] = array\_f1[index[0] - 1][1];

break;

}

delta[0] += (array\_f1[index[0] - 1][0] - array\_f1[index[0]][0]);

delta[1] += (array\_f2[index[1] - 1][1] - array\_f2[index[1]][1]);

if (delta[0] == delta[1]) {

index[0]++, index[1]++;

continue;

}

if (delta[0] > delta[1]) {

index[1]++;

continue;

}

if (delta[0] < delta[1]) {

index[0]++;

continue;

}

}

cout << "Наиболее эффективное решение:" << endl;

cout << "f1 = " << result[0] << ", f2 = " << result[1] << endl;

// освобождение памяти

for (unsigned int i = 0; i < x; i++) {

delete[]out\_array[i];

delete[]array\_f1[i];

delete[]array\_f2[i];

}

return 0;

}

void find\_dominated(int n\_x, int n\_f, int\* matrix)

{

unsigned int i\_want\_to\_eat\_smt\_delicious;

unsigned int second\_one\_wins;

for (int now = 0; now < n\_x; now++)

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

i\_want\_to\_eat\_smt\_delicious = second\_one\_wins = 0;

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

if ((\*(matrix + now \* n\_f + j)) > (\*(matrix + i \* n\_f + j)))

i\_want\_to\_eat\_smt\_delicious++;

else {

if ((\*(matrix + now \* n\_f + j)) == (\*(matrix + (i)\*n\_f + j))) {

i\_want\_to\_eat\_smt\_delicious++;

second\_one\_wins++;

}

else

second\_one\_wins++;

}

}

if (i\_want\_to\_eat\_smt\_delicious == second\_one\_wins)

continue;

else {

if (i\_want\_to\_eat\_smt\_delicious > second\_one\_wins) {

for (int k = 0; k < n\_f; k++)

(\*(matrix + (i)\*n\_f + k)) = -100;

}

else

for (int k = 0; k < n\_f; k++) {

(\*(matrix + now \* n\_f + k)) = -100;

break;

}

}

}

}

int\*\* fill\_out\_array(int n\_x, int n\_f, int x, int\* matrix)

{

int\*\* array = new int\* [x];

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

array[i] = new int[n\_f];

int count = 0;

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

if ((\*(matrix + i \* n\_f)) != -100) {

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

array[count][j] = \*(matrix + i \* n\_f + j);

count++;

}

}

return array;

}

unsigned int count\_not\_dominated(int n\_x, int\* matrix)

{

unsigned int count = 0;

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

if ((\*(matrix + i \* n\_f)) != -100)

count++;

return count;

}

void sort(int ind, int n\_x, int n\_f, int\*\* matrix)

{

for (int i = n\_x - 1; i >= 0; i--)

{

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

{

if (matrix[j][ind] < matrix[j + 1][ind])

{

int tmp[2];

tmp[0] = matrix[j][0];

tmp[1] = matrix[j][1];

matrix[j][0] = matrix[j + 1][0];

matrix[j][1] = matrix[j + 1][1];

matrix[j + 1][0] = tmp[0];

matrix[j + 1][1] = tmp[1];

}

}

}

}

4 РЕЗУЛЬТАТ ВЫПОЛНЕНИЯ ПРОГРАММЫ

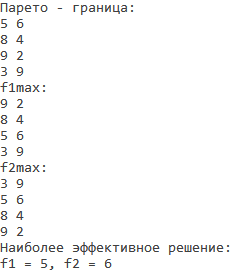


Рисунок 1 – Результат выполнения

ВЫВОДЫ

В ходе выполнения лабораторной работы было найдено эффективное решение двухкритериальной задачи при помощи метода последовательных уступок. Была найдена Парето–граница и при помощи программы, реализующей метод последовательных уступок, было найдено эффективное решение с f1 = 5 и f2 = 6.