МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федерально автономное бюджетное образовательное учреждение высшего образования

«Севастопольский государственный университет»

кафедра Информационных систем

Куркчи Ариф Эрнестович

Институт информационных технологий и управления в технических системах

курс 3 группа ИС/б-31-о

09.03.02 Информационные системы и технологии (уровень бакалавриата)

ОТЧЕТ

по лабораторной работе №5

по дисциплине «Теория принятия решений»

на тему «Исследование методов решения многокритериальных задач принятия решений на основе построения множества Парето»

Отметка о зачете \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_

(дата)

Руководитель практикума

ст. преподаватель   А. Ю. Дрозин

(должность) (подпись) (инициалы, фамилия)

Севастополь 2016

1. Цель работы

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

2. Постановка задачи

Вариант №3

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



3. Текст программы

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  | | --- | --- | | 1 | #include <iostream> | | 2 | #include <fstream> | | 3 | #include <cmath> | | 4 | #include <iomanip> | | 5 |  | | 6 | using namespace std; | | 7 |  | | 8 | *int* \*\*read(*int* &*n*, *int* &*m*) { | | 9 | ifstream is("input.txt"); | | 10 | is >> n >> m; | | 11 | *int* \*\*matrix = new *int* \*[n]; | | 12 | for (*int* i = 0; i < n; i++) { | | 13 | matrix[i] = new *int*[m + 1]; | | 14 | for (*int* j = 0; j < m; j++) { | | 15 | is >> matrix[i][j]; | | 16 | } | | 17 | } | | 18 | return matrix; | | 19 | } | | 20 |  | | 21 | *void* pareto(*int* *n*, *int* *m*, *int* \*\**matrix*) { | | 22 | *int* gt, lt, et; | | 23 | for (*int* i = 0; i < n; i++) { | | 24 | if (matrix[i][m] == 0) { | | 25 | continue; | | 26 | } | | 27 | for (*int* j = i + 1; j < n; j++) { | | 28 | if (matrix[j][m] == 0) { | | 29 | continue; | | 30 | } | | 31 | et = gt = lt = 0; | | 32 | for (*int* k = 0; k < m; k++) { | | 33 | if (matrix[i][k] == matrix[j][k]) { | | 34 | et++; | | 35 | } else if (matrix[i][k] > matrix[j][k]) { | | 36 | gt++; | | 37 | } else { | | 38 | lt++; | | 39 | } | | 40 | } | | 41 | if (gt > 0 && lt == 0) { | | 42 | matrix[j][m] = 0; | | 43 | } | | 44 | if (gt == 0 && lt > 0) { | | 45 | matrix[i][m] = 0; | | 46 | } | | 47 | } | | 48 | } | | 49 | } | | 50 |  | | 51 | *int* \*utopia(*int* *n*, *int* *m*, *int* \*\**matrix*) { | | 52 | *int* \*result = new *int*[m]; | | 53 | *int* max; | | 54 | for (*int* i = 0; i < m; i++) { | | 55 | max = matrix[0][i]; | | 56 | for (*int* j = 1; j < n; j++) { | | 57 | if (matrix[j][i] > max) { | | 58 | max = matrix[j][i]; | | 59 | } | | 60 | } | | 61 | result[i] = max; | | 62 | } | | 63 | return result; | | 64 |  | | 65 | } | | 66 |  | | 67 | *double* \*distance(*int* *n*, *int* *m*, *int* \*\**matrix*, *int* \**utopia\_point*) { | | 68 | *double* \*rx = new *double*[n]; | | 69 | for (*int* i = 0; i < n; i++) { | | 70 | rx[i] = 0; | | 71 | if (matrix[i][m] == 0) { | | 72 | continue; | | 73 | } | | 74 | for (*int* j = 0; j < m; j++) { | | 75 | rx[i] += pow((utopia\_point[j] - matrix[i][j]), 2); | | 76 | } | | 77 | rx[i] = sqrt(rx[i]); | | 78 | } | | 79 | return rx; | | 80 | } | | 81 |  | | 82 | *int* effective(*int* *n*, *int* *m*, *double* \**rx*) { | | 83 | *int* result = 0; | | 84 | *double* min; | | 85 | min = 0; | | 86 | *bool* was\_set = false; | | 87 | for (*int* i = 0; i < n; i++) { | | 88 | if (rx[i] == 0) { | | 89 | continue; | | 90 | } | | 91 | if (!was\_set || min > rx[i]) { | | 92 | was\_set = true; | | 93 | min = rx[i]; | | 94 | result = i + 1; | | 95 | } | | 96 | } | | 97 | return result; | | 98 | } | | 99 |  | | 100 | *char* \*concat(*char* *c*, *int* *d*) { | | 101 | *char* \*result = new *char*[4]; | | 102 | sprintf(result, "%c%d", c, d); | | 103 | return result; | | 104 | } | | 105 |  | | 106 | *void* print(*int* *n*, *int* *m*, *int* \*\**matrix*) { | | 107 | cout << setw(4) << ""; | | 108 | for (*int* i = 0; i < m; i++) { | | 109 | cout << setw(3) << concat('f', i + 1) << "|"; | | 110 | } | | 111 | cout << endl; | | 112 | for (*int* i = 0; i < n; i++) { | | 113 | cout << setw(3) << concat('x', i + 1) << ":"; | | 114 | for (*int* j = 0; j < m; j++) { | | 115 | cout << setw(3) << matrix[i][j] << "|"; | | 116 | } | | 117 | cout << endl; | | 118 | } | | 119 | } | | 120 |  | | 121 | *int* main() { | | 122 | *int* n, m; | | 123 | *int* \*\*matrix = read(n, m); | | 124 | cout << " Входные данные:" << endl; | | 125 | print(n, m, matrix); | | 126 | cout << endl; | | 127 | *int* cout\_width = (*int*) ceil(log10(n)) + 1; | | 128 |  | | 129 | pareto(n, m, matrix); | | 130 | cout << " Решения, находящиеся на Парето-границе" << endl; | | 131 | *bool* was\_printed = false; | | 132 | for (*int* i = 0; i < n; i++) { | | 133 | if (matrix[i][m]) { | | 134 | if (was\_printed) { | | 135 | cout << ", "; | | 136 | } | | 137 | was\_printed = true; | | 138 | cout << setw(cout\_width + 1) << concat('x', i + 1); | | 139 | } | | 140 | } | | 141 | if (!was\_printed) { | | 142 | cout << "Отсутствуют" << endl; | | 143 | return 0; | | 144 | } | | 145 | cout << endl << endl; | | 146 |  | | 147 | *int* \*utopia\_point = utopia(n, m, matrix); | | 148 | cout << " Точка утопии" << endl; | | 149 | cout << "{"; | | 150 | for (*int* i = 0; i < m; i++) { | | 151 | cout << setw(cout\_width) << utopia\_point[i]; | | 152 | if (i != m - 1) { | | 153 | cout << ", "; | | 154 | } | | 155 | } | | 156 | cout << "}" << endl; | | 157 |  | | 158 | *double* \*rx = distance(n, m, matrix, utopia\_point); | | 159 | cout << " Расстояния до точки утопии" << endl; | | 160 | for (*int* i = 0; i < n; i++) { | | 161 | if (matrix[i][m]) { | | 162 | cout << "r[" << setw(cout\_width + 1) << concat('x', i + 1) << "] = " | | 163 | << setw(16) << setprecision(14) << rx[i] << endl; | | 164 | } | | 165 | } | | 166 | cout << endl; | | 167 |  | | 168 | *int* effective\_solution = effective(n, m, rx); | | 169 | cout << " Эффективное решение" << endl; | | 170 | cout << concat('x', effective\_solution) << endl; | | 171 |  | | 172 | return 0; | | 173 | } | |  |

4. Результат

На рисунке 4.1 представлен скриншот демонстрирующий работу написанной программы.

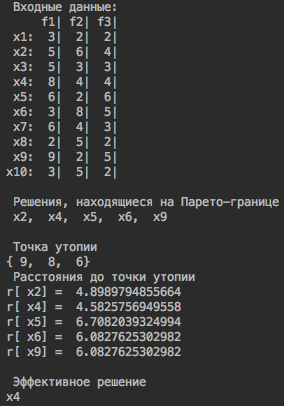


Рисунок 4.1 – Тестовый пример

Вывод

В ходе выполнения лабораторной работы были исследованы способы формирования множества Парето-оптимальных решений и определения эффективных решений в этом множестве.