Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Московский государственный технический
университет имени Н.Э. Баумана»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ «Информатика и системы управления»
КАФЕДРА «Программное обеспечение ЭВМ и информационные технологии»

Лабораторная работа № 1
По курсу «Методы вычислений».

Венгерский метод решения задачи о назначениях

Студент Кондрашова О.П.
Группа ИУ7-11М
Вариант 13
Преподаватель Власов П.А.

Москва, 2021 г.

Цель работы: изучение венгерского метода решения задачи о назначениях.
Содержание работы

1. реализовать венгерский метод решения задачи о назначениях в виде программы на ЭВМ;
2. провести решение задачи с матрицей стоимостей, заданной в индивидуальном варианте, рас- смотрев два случая:
   (a) задача о назначениях является задачей минимизации,
   (b) задача о назначениях является задачей максимизации.

1 Теоретическая часть

1.1 Содержательная постановка задачи

В распоряжении работодателя имеется n работ и n исполнителей. Стоимость выполнения i-ой работы j-ым исполнителем составляет $c_{ij} \geq 0$ единиц. Требуется распределить работу между исполнителями так, чтобы:

1. каждый исполнитель выполнял ровно одну работу;
2. общая стоимость выполнения всех работ была минимальной.

1.2 Математическая постановка задачи

Обозначим $C = (c_{ij})_{i,j=\overline{1;n}}$, где C – матрица стоимостей. Введем управляемые переменные:

$$x_{ij}(x) = \begin{cases} 1, & \text{если } i\text{-ую работу выполняет } j\text{-ый работник} \\ 0, & \text{иначе} \end{cases}, \tag{1}$$

где $i, j = \overline{1, n}$

Обозначим $X = (x_{ij})_{i,j=\overline{1;n}}$, где X – матрица назначений.
Тогда общая стоимость выполнения работ:

$$f = \sum_{i=1}^{n} \sum_{j=1}^{n} c_{ij} \cdot x_{ij} \tag{2}$$

Условие того, что j-й работник выполняет ровно одну работу:

$$\sum_{i=1}^{n} x_{ij} = 1, \tag{3}$$

где $j = \overline{1; n}$
Условие того, что i-ю работу выполняет один работник:

$$\sum_{j=1}^{n} x_{ij} = 1, \tag{4}$$

где $i = \overline{1; n}$
Тогда математическая постановка задачи о назначениях имеет следующий вид:

$$x_{ij}(x) = \begin{cases} f = \sum\limits_{i=1}^{n} \sum\limits_{j=1}^{n} c_{ij} \cdot x_{ij} \to \min \\ \sum\limits_{i=1}^{n} x_{ij} = 1, \quad j = \overline{1; n} \\ \sum\limits_{j=1}^{n} x_{ij} = 1, \quad i = \overline{1; n} \\ x_{i,j} \subseteq 0, 1, \quad i, j = \overline{1; n} \end{cases}, \tag{5}$$

где f – целевая функция, c_{ij} – элементы матрицы стоимостей C, x_{ij} – элементы матрицы назначений X.

1.3 Исходные данные индивидуального варианта

$$\begin{bmatrix} 10 & 4 & 9 & 8 & 5 \\ 9 & 3 & 5 & 7 & 8 \\ 2 & 5 & 8 & 10 & 5 \\ 4 & 5 & 7 & 9 & 3 \\ 8 & 7 & 10 & 9 & 6 \end{bmatrix} \tag{6}$$

1.4 Венгерский метод

Венгерский метод решения задачи о назначениях применяется для решения задачи минимизации, то есть для случая, когда матрица стоимостей C содержит в себе стоимости работ. Также матрица стоимостей C может использоваться для решения задачи максимизации, в этом случае ее элементы отражают прибыль работодателя от назначения j-го работника на i-ю работу.

Решение задачи максимизации отличается от решения задачи минимизации тем, что в данном случае в начале алгоритма добавляется шаг, на котором необходимо найти максимальный элемент столбца, вычесть его из каждого элемента матрицы и домножить матрицу на -1.

Если решается задача минимизации, алгоритм начинается со следующих шагов:

1. в каждом столбце матрицы стоимостей выбирается наименьший элемент и вычитается из всех элементов этого столбца;

2. в каждой строке полученной матрицы выбирается наименьший элемент и вычитается из всех элементов этой строки.

В результате получается матрица, в каждой строке и столбце которой присутствует хотя бы один нуль.

Системой независимых нулей (СНН) будем называть такой набор нулей матрицы стоимостей, в котором никакие два элемента не стоят ни в общей строке, ни в общем столбце.

Для построения начальной СНН необходимо просмотреть столбцы текущей матрицы с размерностью $n \times n$ и первый найденный в столбце 0, в одной строке с которым нет элемента 0^*, отметить как 0^*.

Если полученная СНН содержит n элементов, значит, найдено оптимальное решение. Иначе СНН необходимо улучшить.

Для улучшения СНН необходимо (первый способ):

1. отметить + столбцы, в которых стоят 0^* (данные столбцы и все их элементы будем называть выделенными);

2. найти 0 среди невыделенных элементов и отметить его следующим образом: $0'$;

3. если в одной строке с $0'$ стоит 0^*, снять выеделение со столбца с найденным 0^* и отметить строку с текущим $0'$, затем повторить поиск;

4. если в строке с выделенным $0'$ нет эелемента 0^*, то строится L-цепочка:

   $0' \rightarrow$ по столбцу $\rightarrow 0^* \rightarrow$ по строке $\rightarrow 0' \rightarrow ... \rightarrow 0'$

   В пределах цепочки заменить элементы: $0^* \rightarrow 0, 0' \rightarrow 0^*$

Если среди невыделенных элементов не найдено нулей, то необходимо (второй способ):

1. среди всех невыделенных элементов выбрать наименьший элемент $h > 0$;

2. вычесть h из элементов в невыделенных столбцах;

3. прибавить h к элементам в выделенных строках.

В результате преобразований получим эквивалентную матрицу, среди невыделенных элементов которой есть 0, после этого можно повторить преобразования первого способа.

На рис. 1 представлена схема алгоритма венгерского метода решения задачи о назначениях для задачи минимизации.

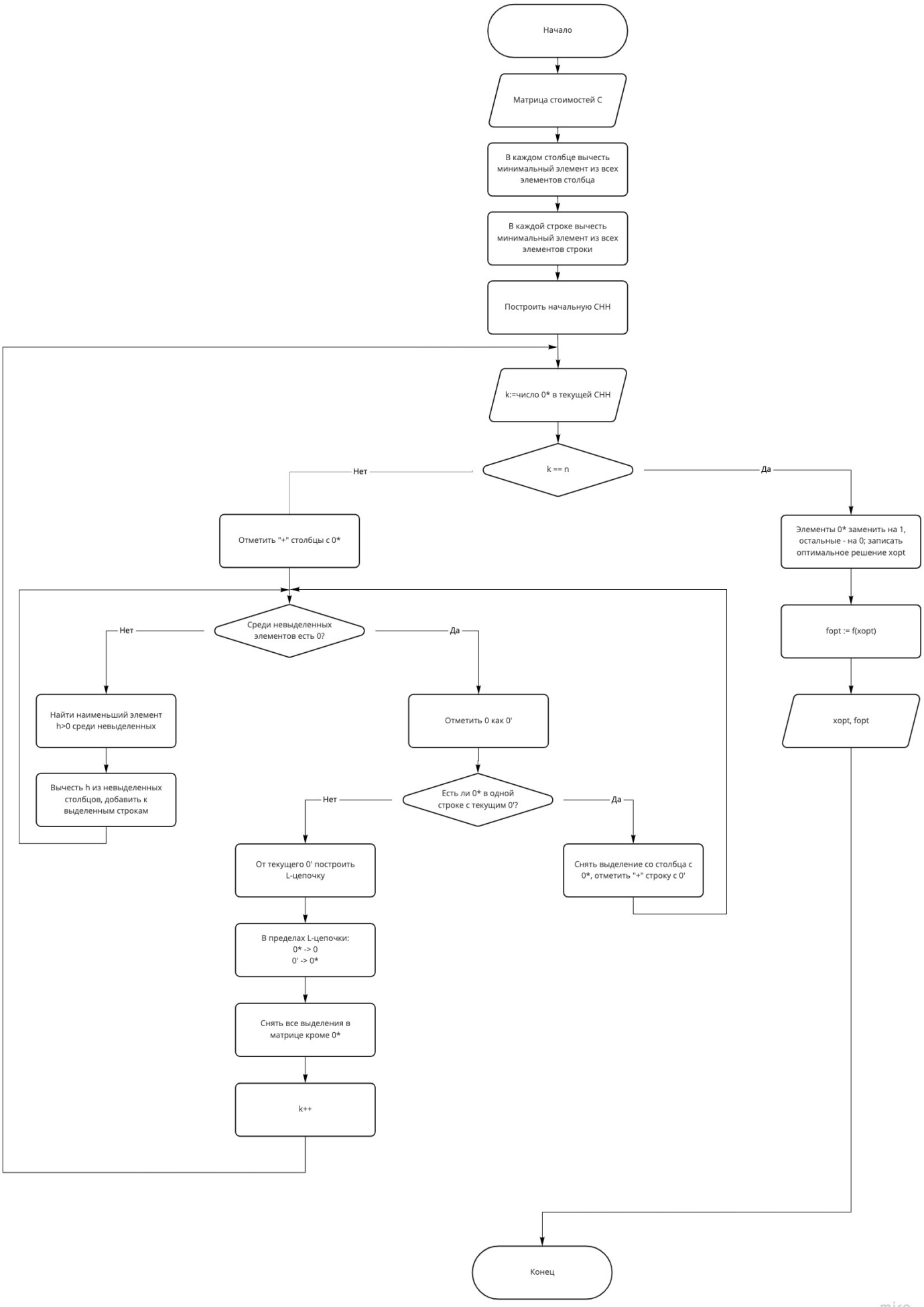

Рис. 1: Результат работы программы

2 Практическая часть

2.1 Листинг программы

В листинге 1 приведен текст программы.

Листинг 1: Листинг программы

```
function lab1()
C = [10, 4, 9, 8, 5;
     9, 3, 5, 7, 8;
     2, 5, 8, 10, 5;
     4, 5, 7, 9, 3;
     8, 7, 10, 9, 6];

debug = true;
%debug = false;
task_max = true;
%task_max = false;

[rows, cols] = size(C);

C1 = C;

if task_max
    % Поиск максимального элемента, вычитание его из элементов
    % матрицы и домножение на -1
    max_cost = 0;
    for i = 1:rows
        for j = 1:cols
            if C(i,j) > max_cost
                max_cost = C(i,j);
            end
        end
    end
    temp_c = C;
    for i = 1:rows
        for j = 1:cols
            C1(i,j) = temp_c(i,j)*(-1) + max_cost;
        end
    end

    if debug
        disp("Матрица после первого шага максимизации");
        disp(C1);
    end
end

% Поиск наименьшего элемента столбца и его вычитание из элементов столбца
min_c = min(C1, [], 1);
C1 = C1 - min_c;
if debug
    disp("Матрица после вычитания наименьшего элемента по столбцам");
    disp(C1);
end

% Поиск наименьшего элемента строки и его вычитание из элементов строки
min_r = min(C1, [], 2);
C1 = C1 - min_r;
if debug
    disp("Матрица после вычитания наименьшего элемента по строкам");
    disp(C1);
```

```
55 end
56
57 % Первичный поиск 0*
58 stars = find_stars(C1);
59
60 N = size(C1, 1);
61 quotes = zeros(N);
62
63 if debug
64     disp("Выделенные 0* для определения СНН");
65     debug_matrix(C1, stars, quotes, [], zeros(N));
66 end
67 % |СНН|
68 CNN = nnz(stars);
69 if debug
70     disp("СНН = ");
71     disp(CNN);
72 end
73 if CNN < cols
74     if debug
75         disp("СНН нужно улучшать");
76     end
77
78     cur_row = 0;
79     cur_col = 0;
80
81     iteration = 1;
82     while CNN < rows
83         if debug
84             disp("Номер итерации: ");
85             disp(iteration);
86         end
87
88         markedCol = find_marked_cols(stars);
89         markedRows = zeros(size(C1,2));
90         while true
91             % Поиск неотмеченных нулей
92             [succ, cur_col, cur_row] = find_unm_zeros(C1, markedCol, markedRows);
93             % Если 0 найден
94             if succ
95                 quotes(cur_row,cur_col) = 1;
96                 if debug
97                     disp("Матрица после нахождения неотмеченного нуля");
98                     debug_matrix(C1, stars, quotes, markedCol, markedRows);
99                 end
100                % Отметить и продолжать поиск
101                [res, col] = check_marked_zero_in_row(cur_row, stars);
102                if res == true
103                    markedRows(cur_row) = 1;
104                    markedCol(col) = 0;
105                    if debug
106                        disp("Матрица после переопределения выделений строк и столбцо
                               в");
107                        debug_matrix(C1, stars, quotes, markedCol, markedRows);
108                    end
109                % Если ноль не найден, построить L-цепочку
110                else
111                    stars = create_L_chain(stars, quotes, cur_row, cur_col);
112                    CNN=CNN+1;
113                    markedCol = find_marked_cols(stars);
114                    markedRows = zeros(size(C1,2));
```

```
115                         if debug
116                             disp("Матрица 0* после построения L–цепочки: ");
117                             disp(stars);
118                         end
119                         break;
120                     end
121                 % Если ноль не найден, вычесть h
122                 else
123                     C1 = calc_h(C1, markedRows, markedCol);
124                     if debug
125                         disp("Матрица после вычитания h из столбцов и прибавления к строкам");
126                         debug_matrix(C1, stars, quotes, markedCol, markedRows);
127                     end
128                 end
129             end
130         iteration = iteration + 1;
131         end
132 end
133
134 % X_opt и f_opt
135 X_opt = zeros(cols);
136 for i = 1:rows
137     for j = 1:cols
138         if stars(i,j)
139             X_opt(i,j) = 1;
140         end
141     end
142 end
143 disp("X_opt = ");
144 disp(X_opt);
145 f_opt = 0;
146 for i = 1:rows
147     for j = 1:cols
148         if stars(i,j)
149             f_opt = f_opt + C(i,j)*stars(i,j);
150         end
151     end
152 end
153 disp("f_opt = ");
154 disp(f_opt);
155
156 % Функция первичного поиска 0*
157 function res = find_stars(C1)
158     N = size(C1,1);
159     stars = zeros(N);
160     for i = 1:N
161         for j = 1:N
162             if C1(i,j) == 0
163                 if max(stars(:,j))==0 && max(stars(i,:))==0
164                     stars(i,j) = 1;
165                     break;
166                 end
167             end
168         end
169     end
170     res = stars;
171 end
172
173 % Функция поиска отмеченных столбцов
174 function marked_cols = find_marked_cols(stars)
```

```
175     marked_cols = sum(stars);
176 end
177
178 % Функция поиска неотмеченного нуля в матрице
179 function [res,col,row] = find_unm_zeros(C1, markedCol, markedRows)
180     N = size(C1,1);
181     col = 0;
182     row = 0;
183     res = false;
184     for j = 1:N
185         for i = 1:N
186             if C1(i,j) == 0 && markedCol(j) == 0 && markedRows(i) == 0
187                 res = true;
188                 col = j;
189                 row = i;
190                 break;
191             end
192         end
193     end
194 end
195
196 % Функция, которая определяет, есть ли в строке матрицы 0*
197 function [res, col] = check_marked_zero_in_row(row, stars)
198     res = false;
199     col = 0;
200     for j = 1:size(stars, 2)
201         if stars(row, j) == 1
202             res = true;
203             col = j;
204             break
205         end
206     end
207 end
208
209 % Функция поиска 0* в столбце
210 function [res, row] = check_marked_zero_in_column(col,stars)
211     [res,row] = check_marked_zero_in_row(col,stars');
212 end
213
214 % Функция пересчета матрицы на основе поиска h и его вычитания из невыделенных столбц
    ов
215 function matrix = calc_h(C1, markedRows, markedCol)
216     matrix = C1;
217     N = size(C1,1);
218     min_el = -1;
219
220     % Поиск минимального элемента из невыделенных
221     for j = 1:N
222         for i = 1:N
223             if markedCol(j) == 0 && markedRows(i) == 0
224                 if matrix(i, j) < min_el || min_el == -1
225                     min_el = matrix(i, j);
226                 end
227             end
228         end
229     end
230
231     % Вычитание минимального элемента из невыделенных
232     for j = 1:N
233         for i = 1:N
234             if markedCol(j) == 0 && markedRows(i) == 0
```

```
235                     matrix(i,j) = matrix(i,j) - min_el;
236                 end
237                 if markedCol(j) == 1 && markedRows(i) == 1
238                     matrix(i,j) = matrix(i,j) + min_el;
239                 end
240             end
241         end
242     end
243
244     % Функция построения L-цепочки
245     function created_L = create_L_chain(stars, quotes, row, col)
246         res = true;
247         cur_col = col;
248         cur_row = row;
249         created_L = stars;
250         while res
251             created_L(cur_row,cur_col) = 1;
252             [res,cur_row] = check_marked_zero_in_column(cur_col,stars);
253             if(res == true)
254                 stars(cur_row, cur_col) = 0;
255                 created_L(cur_row, cur_col) = 0;
256                 [res, cur_col] = check_marked_zero_in_row(cur_row,quotes);
257             end
258         end
259     end
260
261     % Функция вывода матрицы с 0*, 0'
262     function debug_matrix(C1, stars, quotes, markedCol, markedRows)
263         for i = 1:size(C1,1)
264             for j = 1:size(C1,2)
265                 if stars(i,j) == 1
266                     fprintf("%d* \t", C1(i,j));
267                 else
268                     if quotes(i,j) == 1
269                         fprintf("%d' \t", C1(i,j));
270                     else
271                         fprintf("%d \t", C1(i,j));
272                     end
273                 end
274             end
275             if markedRows(i)
276                 fprintf(' +\n');
277             else
278                 fprintf(' \n');
279             end
280         end
281         for i = 1:length(markedCol)
282             if markedCol(i)
283                 fprintf("+ \t");
284             else
285                 fprintf(" \t");
286             end
287         end
288         fprintf("\n");
289     end
290
291     end
```

2.2 Задача максимизации

Исходная матрица:

$$C = \begin{bmatrix} 10 & 4 & 9 & 8 & 5 \\ 9 & 3 & 5 & 7 & 8 \\ 2 & 5 & 8 & 10 & 5 \\ 4 & 5 & 7 & 9 & 3 \\ 8 & 7 & 10 & 9 & 6 \end{bmatrix} \tag{7}$$

На первом шаге максимизации определяем максимальный элемент матрицы стоимостей – 10. Вычитаем найденный максимум из всех элементов матрицы и домножаем их на -1:

$$\begin{bmatrix} 0 & 6 & 1 & 2 & 5 \\ 1 & 7 & 5 & 3 & 2 \\ 8 & 5 & 2 & 0 & 5 \\ 6 & 5 & 3 & 1 & 7 \\ 2 & 3 & 0 & 1 & 4 \end{bmatrix} \tag{8}$$

Для каждого из столбцов вычтем минимальные элементы:

$$\begin{bmatrix} 0 & 3 & 1 & 2 & 3 \\ 1 & 4 & 5 & 3 & 0 \\ 8 & 2 & 2 & 0 & 3 \\ 6 & 2 & 3 & 1 & 5 \\ 2 & 0 & 0 & 1 & 2 \end{bmatrix} \tag{9}$$

Для каждой из строк вычтем минимальные элементы:

$$\begin{bmatrix} 0 & 3 & 1 & 2 & 3 \\ 1 & 4 & 5 & 3 & 0 \\ 8 & 2 & 2 & 0 & 3 \\ 5 & 1 & 2 & 0 & 4 \\ 2 & 0 & 0 & 1 & 2 \end{bmatrix} \tag{10}$$

Выделим 0* и составим СНН:

$$\begin{bmatrix} 0^* & 3 & 1 & 2 & 3 \\ 1 & 4 & 5 & 3 & 0^* \\ 8 & 2 & 2 & 0^* & 3 \\ 5 & 1 & 2 & 0 & 4 \\ 2 & 0^* & 0' & 1 & 2 \end{bmatrix} \tag{11}$$

$$\begin{matrix} + & + & - & + & + \end{matrix} \tag{12}$$

$|CHH| = 4 < 5 = n$, следовательно, СНН нужно улучшать.
1 итерация.
Матрица после переопределения выделений строк и столбцов:

$$\begin{bmatrix} 0^* & 3 & 1 & 2 & 3 \\ 1 & 4 & 5 & 3 & 0^* \\ 8 & 2 & 2 & 0^* & 3 \\ 5 & 1 & 2 & 0 & 4 \\ 2 & 0^* & 0' & 1 & 2 \end{bmatrix} \begin{matrix} - \\ - \\ - \\ - \\ + \end{matrix} \tag{13}$$

$$\begin{matrix} + & - & - & + & + \end{matrix} \tag{14}$$

Находим $h = 1 > 0$ – наименьший элемент в невыделенных столбцах. Вычитаем h из невыделенных столбцов:

$$\begin{bmatrix} 0^* & 2 & 0 & 2 & 3 \\ 1 & 3 & 4 & 3 & 0^* \\ 8 & 1 & 1 & 0^* & 3 \\ 5 & 0 & 1 & 0 & 4 \\ 2 & -1^* & -1' & 1 & 2 \end{bmatrix} \tag{15}$$

Прибавляем h к выделенным строкам:

$$\begin{bmatrix} 0^* & 2 & 0 & 2 & 3 \\ 1 & 3 & 4 & 3 & 0^* \\ 8 & 1 & 1 & 0^* & 3 \\ 5 & 0 & 1 & 0 & 4 \\ 3 & 0^* & 0' & 2 & 3 \end{bmatrix} \tag{16}$$

Находим неотмеченный 0 и переопределяем выделение столбцов и строк:

$$\begin{bmatrix} 0^* & 2 & 0' & 2 & 3 \\ 1 & 3 & 4 & 3 & 0^* \\ 8 & 1 & 1 & 0^* & 3 \\ 5 & 0 & 1 & 0 & 4 \\ 3 & 0^* & 0' & 2 & 3 \end{bmatrix} \begin{matrix} + \\ - \\ - \\ - \\ + \end{matrix} \tag{17}$$

$$\begin{matrix} - & - & - & + & + \end{matrix} \tag{18}$$

Построим L-цепочку, начиная с элемента с индексами (4; 2), где 4 – номер строки, 2 – номер столбца. Тогда в цепочку так же войдут элементы с индексами (5; 2) и (5; 3).

Матрица после построения L-цепочки и переопределения отмеченных элементов:

$$\begin{bmatrix} 0^* & 2 & 0' & 2 & 3 \\ 1 & 3 & 4 & 3 & 0^* \\ 8 & 1 & 1 & 0^* & 3 \\ 5 & 0^* & 1 & 0 & 4 \\ 3 & 0 & 0^* & 2 & 3 \end{bmatrix} \tag{19}$$

$|CHH| = 5 = n$, следовательно, СНН больше улучшать не нужно.

$$X_{opt} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \end{bmatrix} \tag{20}$$

$$f_{opt} = 10 + 5 + 10 + 10 + 8 = 43 \tag{21}$$

2.3 Задача минимизациии

Исходная матрица:

$$C = \begin{bmatrix} 10 & 4 & 9 & 8 & 5 \\ 9 & 3 & 5 & 7 & 8 \\ 2 & 5 & 8 & 10 & 5 \\ 4 & 5 & 7 & 9 & 3 \\ 8 & 7 & 10 & 9 & 6 \end{bmatrix} \tag{22}$$

Для каждого из столбцов вычтем минимальные элементы:

$$\begin{bmatrix} 8 & 1 & 4 & 1 & 2 \\ 7 & 0 & 0 & 0 & 5 \\ 0 & 2 & 3 & 3 & 2 \\ 2 & 2 & 2 & 2 & 0 \\ 6 & 4 & 5 & 2 & 3 \end{bmatrix} \tag{23}$$

Для каждой из строк вычтем минимальные элементы:

$$\begin{bmatrix} 7 & 0 & 3 & 0 & 1 \\ 7 & 0 & 0 & 0 & 5 \\ 0 & 2 & 3 & 3 & 2 \\ 2 & 2 & 2 & 2 & 0 \\ 4 & 2 & 3 & 0 & 1 \end{bmatrix} \tag{24}$$

Выделим 0* и составим СНН:

$$\begin{bmatrix} 7 & 0^* & 3 & 0 & 1 \\ 7 & 0 & 0^* & 0 & 5 \\ 0^* & 2 & 3 & 3 & 2 \\ 2 & 2 & 2 & 2 & 0^* \\ 4 & 2 & 3 & 0^* & 1 \end{bmatrix} \tag{25}$$

$$+ \quad + \quad - \quad + \quad + \tag{26}$$

$|CHH| = 5 = n$, следовательно, СНН больше улучшать не нужно.

$$X_{opt} = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 \end{bmatrix} \tag{27}$$

$$f_{opt} = 2 + 4 + 5 + 9 + 3 = 23 \tag{28}$$

3 Заключение

В результате выполнения лабораторной работы был реализован венгерский метод решения задачи о назначениях для задач минимизации и максимизации. В ходе выполнения лабораторной работы была написана программа на языке MatLab, позволяющая применять венгерский метод решения задачи о назначениях.