|  |  |
| --- | --- |
| Gerb-BMSTU_01 | **Министерство образования и науки Российской Федерации**  **Федеральное государственное бюджетное образовательное учреждение**  **высшего образования**  **«Московский государственный технический университет**  **имени Н.Э. Баумана**  **(национальный исследовательский университет)»**  **(МГТУ им. Н.Э. Баумана)** |

Отчет по лабораторной работе № 1

по курсу

«Методы вычислений»

на тему

«Венгерский метод решения задачи о назначениях»

Вариант № 13

Студент: Пенской И. С., ИУ7-11М

Преподаватель: Власов П. А.

Москва, 2017

**Содержательная постановка задачи:**

В распоряжении работодателя имеется n видов работ и n исполнителей. Стоимость выполнения i-ой работы j-м исполнителем составляет единиц. Требуется распределить работы между исполнителями так чтобы:

1) каждой из них выполняет ровно одну работу;

2) общая стоимость выполнения всех работ была минимальна (эффективность работы была максимальна).

Введем управляемые переменные:

где

**Математическая постановка задачи:**

**Исходная матрица стоимостей:**

**Краткое описание Венгерского метода:**

1) в каждом столбце матрицы С найти наименьший элемент и вычесть его из каждого элемента этой строки;

2) аналогично пункту 1) провести операции со столбцами;

3) из нулей матрицы С попытаться сформировать систему независимых нулей, содержащую как можно больше элементов;

4) если СНН содержит менее n нулей, то необходимо ее улучшать.

Алгоритм улучшения:

1) столбцы, в которых стоят 0\*, отметим «+» и будем называть их выделенными. Элементы этих столбцов также будем называть выделенными;

2) среди новых элементов есть 0. Попытаемся включить его в СНН. Отметим его штрихом;

3) если 0’ включается в СНН, то ни один нуль из его строки не может быть включен в СНН, отметим эту строку «+»;

4) при этом в одной строке с тек 0’ есть 0\*, тогда этот 0\* должен быть исключен из СНН, снимем выделение столбца, где стоит этот 0\*;

5) среди невыделенных элементов элементов есть 0, отметим его как 0’;

6) т. к. в одной строке с этим 0’ не стоит 0\*, то выделения строк и столбцов изменять не надо и нужно построить L-цепочку;

7) в пределах построенной L-цепочки заменяем 0’ на 0\* и наоборот, удаляем все штрихи и выделения строк и столбцов;

8) если среди невыделенных элементов не обнаружилось нулей, то необходимо найти среди них минимум и прибавить к выделенным строкам, а отнять от выделенных столбцов и повторить поиск невыделенных нулей.

Замечание: в случае задачи максимизации необходимо предварительно вычесть из каждого столбца максимум этого столбца.

Текст программы:

function lab\_1()

% Отладочный режим

DEBUG = 0;

% Если 0, решается задача мимнимизации, иначе - максимизации

MAXIMIZE = 0;

% Матрица стоимостей

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];

% Размер матрицы стоимостей

n = size(C, 1);

% Копия матрицы стоимостей для операций

Cm = C;

% Координаты СНН

independent\_zeros = zeros(1, n);

% Первоначальная печать матрицы стоимостей

fprintf("\n");

print\_Cm(Cm, independent\_zeros);

% В случае задачи макисимизации вычитаем элементы из максимума столбца

if MAXIMIZE ~= 0

column\_max = max(Cm);

for i = 1:n

for j = 1:n

Cm(i, j) = column\_max(j) - Cm(i, j);

end

end

end

% В каждом столбце вычитаем минимум этого столбца

column\_min = min(Cm);

for i = 1:n

for j = 1:n

Cm(i, j) = Cm(i, j) - column\_min(j);

end

end

% В каждой строке вычитаем минимум этой строки

for i = 1:n

minimal = min(Cm(i, :));

for j = 1:n

Cm(i, j) = Cm(i, j) - minimal;

end

end

% Строим начальную СНН

for i = 1:n

j = 1;

flag = 0;

while j <= n && flag == 0

if Cm(j, i) == 0 && sum(independent\_zeros == j) == 0

independent\_zeros(i) = j;

flag = 1;

end

j = j + 1;

end

end

if DEBUG ~= 0

print\_Cm(Cm, independent\_zeros);

end

% Выделенные столбцы

marked\_columns = zeros(1, n);

% Выделяем столбцы, содержащие независимые нули

for i = 1:n

if independent\_zeros(i) > 0

marked\_columns(i) = 1;

end

end

% Выделенные строки

marked\_strings = zeros(1, n);

% Координаты штрихованных нулей

marked\_zeros = zeros(1, n);

% Цикл до тех пор, пока решение не будет оптимальным

while sum(independent\_zeros == 0) ~= 0

% Строка обнаруженного невыделенного нуля

new\_zero\_string = 0;

% Отмечаем невыделенный нуль штрихом

i = 1;

flag = 0;

while i <= n && flag == 0

if marked\_columns(i) == 0

j = 1;

while j <= n && flag == 0

if Cm(j, i) == 0 && marked\_strings(j) == 0

new\_zero\_string = j;

marked\_zeros(i) = j;

flag = 1;

end

j = j + 1;

end

end

i = i + 1;

end

% Обнаружили невыделенный нуль

if flag == 1

% Проверяем, что в строке обнаруженного нуля нет 0\*

if sum(independent\_zeros == new\_zero\_string) == 0

% Строим L-цепочку

l\_columns = find(marked\_zeros ~= 0);

l\_chain = zeros(size(l\_columns, 2) \* 2 - 1, 2);

l\_chain(1, :) = [marked\_zeros(l\_columns(1)) l\_columns(1)];

for i = 2:size(l\_columns, 2)

l\_chain(i \* 2 - 2, :) = [independent\_zeros(l\_columns(i - 1)) l\_columns(i - 1)];

l\_chain(i \* 2 - 1, :) = [marked\_zeros(l\_columns(i)) l\_columns(i)];

end

% Удаляем штрихи и отметки строк и столбцов

marked\_zeros = zeros(1, n);

marked\_strings = zeros(1, n);

marked\_columns = zeros(1, n);

% Формируем новую СНН

for i = 1:size(l\_chain, 1)

if mod(i, 2) ~= 0

independent\_zeros(l\_chain(i, 2)) = l\_chain(i, 1);

end

end

% Заново выделяем столбцы

for i = 1:n

if independent\_zeros(i) > 0

marked\_columns(i) = 1;

end

end

if DEBUG ~= 0

print\_Cm(Cm, independent\_zeros);

end

else

% Переносим отметку со столбца на строку

marked\_strings(new\_zero\_string) = 1;

marked\_columns(independent\_zeros == new\_zero\_string) = 0;

end

else

% Невыделенных нулей необнаружено, создаем новые нули

minimal = Inf();

for i = 1:n

for j = 1:n

if marked\_strings(i) == 0 && marked\_columns(j) == 0 && Cm(i, j) < minimal

minimal = Cm(i, j);

end

end

end

for i = 1:n

for j = 1:n

if marked\_strings(i) == 1

Cm(i, j) = Cm(i, j) + minimal;

end

if marked\_columns(j) == 0

Cm(i, j) = Cm(i, j) - minimal;

end

end

end

end

end

% Печать матрицы назначений

for i = 1:n

for j = 1:n

if independent\_zeros(j) == i

fprintf(" 1 ");

else

fprintf(" 0 ");

end

end

fprintf("\n");

end

fprintf("\n");

% Вывод итогового значения

s = 0;

for i = 1:n

s = s + C(independent\_zeros(i), i);

end

if MAXIMIZE ~= 0

fprintf(" Максимальная эффективность: %d\n\n", s);

else

fprintf(" Минимальные затраты: %d\n\n", s);

end

end

% Печать матрицы стоимостей

function print\_Cm(Cm, independent\_zeros)

n = size(Cm, 1);

for i = 1:n

for j = 1:n

if independent\_zeros(j) == i

fprintf(" %3d\* ", Cm(i, j));

else

fprintf(" %3d ", Cm(i, j));

end

end

fprintf("\n");

end

fprintf("\n");

end

**Результаты расчетов:**

а) для задачи минимизации:

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

Минимальные затраты: 23

б) для задачи максимизации:

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

Максимальная эффективность: 43