**Отчёт по лабораторной работе №3**

**по дисциплине**

**Методы вычислений**

Выполнил: студент ИУ7-17

Пахомов А.А.

Вариант 1

МГТУ, 2015 г.

**Теоретическая часть**

**Содержательная постановка транспортной задачи:**

Имеется производителей некоторой однородной продукции, мощность -го производителя единиц, . Имеется так же потребителей этой же продукции. Мощность -го потребителя обозначим через единиц, . Стоимость перевозки единицы продукции от -го производителя к -му потребителю составляет единиц. Необходимо составить такой план перевозок от производителей к потребителям при котором:

1. Вся продукция вывезена от производителей;
2. Вся продукция доставлена потребителю с учетом ограничений на мощность;
3. Общая стоимость перевозок минимальная.

**Математическая постановка транспортной задачи:**

**Входные данные:**

Входные данные задаются в виде значений мощностей производителей и потребителей , а также матрицы стоимостей , для которых требуется найти решение методом северо-западного угла и методом потенциалов.

Кроме того, из лабораторной работы №1 были взяты входные данные в виде матрицы стоимостей для решения задачи о назначениях с использованием метода потенциалов.

**Метод потенциалов для решения транспортной задачи:**

**C:\Users\Peter\AppData\Local\Microsoft\Windows\INetCache\Content.Word\Zamechanie.png**

**Текст программы**

*lab3.m*

C = dlmread('costs.txt');

S = dlmread('have.txt');

D = dlmread('need.txt');

clc;

if(sum(S) ~= sum(D))

fprintf('Задача несбалансирована!');

else

fprintf('Матрица стоимостей:\n');

disp(C);

fprintf('Источники:');

disp(S);

fprintf('Стоки:');

disp(D);

debug\_mode = false;

[transportation, cost] = transportation\_task(C, S, D, debug\_mode);

fprintf('Оптимальный план перевозки:\n');

disp(transportation);

fprintf('Оптимальная стоимость:');

disp(cost);

end;

*NW\_angle.m*

function [x, indexes] = NW\_angle(S, D)

m = length(S);

n = length(D);

i = 1;

j = 1;

x = zeros(m,n);

k = 1;

while (i <= m && j <= n)

indexes(k,:) = [i, j];

k = k + 1;

if (D(j) > S(i))

x(i, j) = S(i);

D(j) = D(j) - S(i);

i = i + 1;

else

x(i, j) = D(j);

S(i) = S(i) - D(j);

j = j + 1;

end;

end;

end

*transportation\_task.m*

function [ xx, ff ] = transportation\_task(C, S, D, debug\_mode)

[xx, indexes] = NW\_angle(S, D);

ff = sum(sum(C.\*xx));

if(debug\_mode)

fprintf('Начальный план перевозки:\n');

disp(xx);

fprintf('Стоимость:');

disp(ff);

end;

k = 0;

while(true)

k = k + 1;

if(debug\_mode)

fprintf('Итерация %d:\n', k);

end;

[ii, ji] = toBasis(C, indexes, debug\_mode);

if(ii == -1)

break;

end;

[xx, io, jo] = getOutOfBasis(xx, indexes, ii, ji);

[~,ind] = ismember([io,jo], indexes, 'rows');

indexes(ind,:) = [ii, ji];

ff = sum(sum(C.\*xx));

if(debug\_mode)

fprintf('Элемент [%d, %d] в базис\n', ii, ji);

fprintf('Элемент [%d, %d] из базиса\n', io, jo);

fprintf('План перевозки:\n');

disp(xx);

fprintf('Стоимость:');

disp(ff);

end;

end;

end

*toBasis.m*

function [ii, jj] = toBasis(C, indexes, debug\_mode)

[u, v] = getUV(C, indexes(:,1), indexes(:,2));

[m, n] = size(C);

mi = 1 - indexesToMatrix(indexes);

d = 0;

ii = -1;

jj = -1;

for i = 1: m

for j = 1: n

if(mi(i, j) == 1)

nd = C(i, j)-u(i)-v(j);

if(debug\_mode)

fprintf('d[%d,%d] = %d\n', i, j, nd);

end;

if(nd < d)

d = nd;

ii = i;

jj = j;

end;

end;

end;

end;

if(debug\_mode)

fprintf('\n');

if(d ~= 0)

fprintf('d[%d,%d] = %d минимально\n', ii, jj, d);

else

fprintf('Отрицательных d не найдено, последний найденный план оптимален\n');

end;

end;

end

*getUV.m*

function [u, v] = getUV(C, ui, vi)

m = length(unique(ui));

n = length(unique(vi));

% решу через СЛАУ

coef = zeros(m+n);

y = zeros(1, m+n);

coef(1,1) = 1;

y(1) = 0;

k = 2;

for i = 1 : length(ui)

coef(k, ui(i)) = 1;

coef(k, m + vi(i)) = 1;

y(k) = C(ui(i), vi(i));

k = k + 1;

end;

x = coef^(-1)\*y';

u = x(1:m);

v = x(m+1 : m+n);

end

*getOutOfBasis.m*

function [xx, io, jo] = getOutOfBasis(xx, indexes, ii, ji)

matr = indexesToMatrix(indexes);

index = [ii, ji];

path = findCycle(index, ii, matr, [], false);

path = flipud(path);

io = path(2,1);

jo = path(2,2);

m = xx(io, jo);

k = 4;

while(k <= size(path, 1))

tm = xx(path(k,1), path(k,2));

if(tm < m)

io = path(k,1);

jo = path(k,2);

m = tm;

end;

k = k + 2;

end;

for k = 1: size(path, 1)

i = path(k,1);

j = path(k,2);

if(mod(k, 2) ~= 0)

xx(i,j) = xx(i,j) + m;

else

xx(i,j) = xx(i,j) - m;

end;

end;

end

*findCycle.m*

function path = findCycle(index, baseRow, matr, path, isFindInRow)

if(~isFindInRow)

col = matr(:, index(2));

rows = find(col==1);

for i=1:length(rows)

if(rows(i) ~= index(1))

path = findCycle([rows(i), index(2)], baseRow, matr, path, ~isFindInRow);

if(~isempty(path))

ind = size(path, 1) + 1;

path(ind,:) = index;

return;

end;

end;

end;

else

if(index(1) == baseRow)

ind = size(path, 1) + 1;

path(ind,:) = index;

return;

else

row = matr(index(1), :);

cols = find(row==1);

for i=1:length(cols)

if(cols(i) ~= index(2))

path = findCycle([index(1), cols(i)], baseRow, matr, path, ~isFindInRow);

if(~isempty(path))

ind = size(path, 1) + 1;

path(ind,:) = index;

return;

end;

end;

end;

end;

end;

end

**Результаты вычислений**

**Транспортная задача:**

Матрица стоимостей:

4 2 1 3 7

1 5 4 6 3

5 4 8 7 2

9 9 3 2 5

3 4 7 8 2

Источники: 1 1 1 1 1

Стоки: 1 1 1 1 1

Оптимальный план перевозки:

0 0 1 0 0

1 0 0 0 0

0 1 0 0 0

0 0 0 1 0

0 0 0 0 1

Оптимальная стоимость: 10

**Задача о назначениях симплексным методом:**

Матрица стоимостей:

5 4 3 4

3 2 5 5

1 6 3 2

Источники: 140 100 60

Стоки: 80 80 60 80

Оптимальный план перевозки:

0 0 60 80

20 80 0 0

60 0 0 0

Оптимальная стоимость: 780