DAFTAR PUSTAKA

Ahmed, M. M., Khan, A. R., Ahmed, F., & Uddin, Md. S. (2016). Incessant Allocation Method for Solving Transportation Problems. *American Journal of Operations Research*, *06*(03), 236–244.

Amaliah, B., Fatichah, C., & Suryani, E. (2019). Total opportunity cost matrix–Minimal total: A new approach to determine initial basic feasible solution of a transportation problem. *Egyptian Informatics Journal*, *20*(2), 131-141.

Ariyanto, M., & Caesarendra, W. (2011). *Panduan Belajar Mandiri MATLAB*. https://www.researchgate.net/publication/311717122

Basriati, S., & Cahyani, D. (2017). Penyelesaian Model Transportasi Menggunakan Metode ASM, RDI dan MODI (Studi Kasus: PT. Melayu Bumi Lestari). *Jurnal Sains Matematika dan Statistika*, *3*(2), 67-73.

Jamali, S., Soomro, A. S., & Shaikh, M. M. (2020). The Minimum Demand Method–A New and Efficient Initial Basic Feasible Solution Method for Transportation Problems. *Journal of Mechanics of Continua and Mathematical Sciences*, *15*(19), 94-109.

Juman, Z. A. M. S., & Hoque, M. A. (2015). An efficient heuristic to obtain a better initial feasible solution to the transportation problem. *Applied Soft Computing Journal*, *34*, 813–826.

Juman, Z. A. M. S., & Nawarathne, N. G. S. A. (2019). An efficient alternative approach to solve a transportation problem. *Ceylon Journal of Science*, *48*(1), 19.

Karagul, K., & Sahin, Y. (2020). A novel approximation method to obtain initial basic feasible solution of transportation problem. *Journal of King Saud University - Engineering Sciences*, *32*(3), 211–218.

Meflinda, A., & Mahyarni. (2011). Operations Research (Riset Operasi). Pekanbaru: Unri Press.

Mohanram, J.S. 2021. TRANSPORTATION-PROBLEM-SOLVER. GitHub. https://github.com/m0hanram/TRANSPORTATION-PROBLEM-SOLVER/tree/main

Rangkuti, A. (2022). 7 Model Riset Operasi & Aplikasinya Edisi Revisi. Surabaya: Brilian Internasional.

Saputri, Z. E., Nasution, Y. N., & Wasono, W. (2019). Perbandingan Hasil Revised Distribution Method dan Metode Stepping Stone dengan Penentuan Nilai Awal Menggunakan Metode North West Corner dalam Meminimumkan Biaya Pendistruibusian Barang. *EKSPONENSIAL*, *10*(1), 59-66.

Septiana, A. R., Solikhin, S., & Ratnasari, L. (2017). Metode ASM pada Masalah Transportasi Seimbang. *Jurnal Matematika*, *20*(2), 71-78.

Taha, Hamdy., (2007). *Operation Research:An Introduction*. Eighth edition, Mc Graw-Hill, New York

Ullah, M. W., Uddin, M. A., & Kawser, R. (2016). A modified Vogel's approximation method for obtaining a good primal solution of transportation problems. *Annals of Pure and Applied Mathematics*, *11*(1), 63-71.

Wijaya A., (2021), Pengantar Riset Operasi. Jakarta: Mitra Wacana Media

L A M P I R A N

Lampiran 1. Syntax Program Metode MDM

```
clear all;clc;
cost_matrix = [2010 4575 5100 2097 2008;
               9789 673 648 870 2931];
supply = [1849 4586];
demand = [101 220 189 1180 337];

allocation_matrix = zeros(size(cost_matrix));

%check balanced/unbalanced problem
if sum(supply)==sum(demand)
    fprintf('Given Transportation Problem is Balanced\n')
else
    fprintf('Given Transportation Problem is Unbalanced\n')
    if sum(supply)<sum(demand)
        fprintf('Dummy in supply\n')
        cost_matrix(end+1, :)=zeros(1, size(demand, 2))
        supply(end+1)=sum(demand)-sum(supply)
    elseif sum(demand)<sum(supply)
        fprintf('Dummy in demand\n')
        cost_matrix(:, end+1)=zeros(1, size(supply, 2))
        demand(end+1)=sum(supply)-sum(demand)
    end
end


c = cost_matrix;
while sum(demand) > 0 && sum(supply)>0
    %total_updates = 0
    for k = 1:length(demand)
        if sum(demand)==0 && sum(supply)==0
            break
        end
        %menemukan permintaan minimum dalam matriks permintaan
        permintaan_minimum = min(demand(demand>0));
        %menemukan indeks permintaan minimum
        indeks_permintaan_minimum = find(demand == permintaan_
                                         minimum);
        for j = 1:length(indeks_permintaan_minimum)
            % Ambil indeks_permintaan_minimum saat ini
            current_demand_index = indeks_permintaan_minimum(j);
            % Temukan biaya terkecil di kolom permintaan_minimum
            saat ini
            kolom_permintaan_minimum = current_demand_index;
            kolom_biaya = cost_matrix(:, kolom_permintaan_
                          minimum);
            [biaya_terkecil,baris_biaya_terkecil]=min(kolom_biaya)

            % Simpan indeks sel dengan biaya terkecil
            indeks_sel_biaya_terkecil(j, 1)= baris_biaya_terkecil;
            indeks_sel_biaya_terkecil(j, 2)= kolom_permintaan_
                                             minimum;
            % Alokasi unit sesuai dengan supply dan demand
            baris = indeks_sel_biaya_terkecil(j, 1);
```

```
            kolom = indeks_sel_biaya_terkecil(j, 2);
            supply_avail = supply(baris);
            demand_avail = demand(kolom);

            allocation = min(supply_avail, demand_avail);
            allocation_matrix(baris, kolom) = allocation;

            supply(baris) = supply(baris) - allocation;
            demand(kolom) = demand(kolom) - allocation;

            % Menghapus baris atau kolom yang sudah terpenuhi
            if supply(baris) == 0
                cost_matrix(baris, :) = Inf;
            end
            if demand(kolom) == 0
                cost_matrix(:, kolom) = Inf;
            end
        end
    end
end
disp('Allocation Matrix:');
disp(allocation_matrix)
totalCost = sum(sum(c.*allocation_matrix))
```

Lampiran 2. Sintax Kode Program Metode MVAM

```
clear all;clc;
tic;

costMatrix =randi([10, 100], 10, 10)
supply = randi([100, 700], 1, 10)
demand =randi([100, 700], 1, 10)

allocation_matrix = zeros(size(costMatrix));

%check balanced/unbalanced problem
if sum(supply)==sum(demand)
    fprintf('Given Transportation Problem is Balanced\n')
else
    fprintf('Given Transportation Problem is Unbalanced\n')
    if sum(supply)<sum(demand)
        fprintf('Dummy in supply\n')
        costMatrix(end+1, :)=zeros(1, size(demand, 2))
        supply(end+1)=sum(demand)-sum(supply)
    elseif sum(demand)<sum(supply)
        fprintf('Dummy in demand\n')
        costMatrix(:, end+1)=zeros(1, size(supply, 2))
        demand(end+1)=sum(supply)-sum(demand)
    end
end

[m, n] = size(costMatrix);
for i = 1:m
    % Temukan elemen terbesar dalam baris saat ini
    maxCost = max(costMatrix(i, :));
    % Kurangkan elemen terbesar dengan setiap elemen dalam baris
    modifiedCostMatrix_row(i, :) = maxCost - costMatrix(i, :);
end
for j = 1:n
    % Temukan elemen terbesar dalam kolom saat ini
    maxCost = max(costMatrix(:, j));
    % Kurangkan elemen terbesar dengan setiap elemen dalam kolom
    modifiedCostMatrix_col(:, j) = maxCost-costMatrix(:, j);
end
% Membentuk matriks hasil yang merupakan penjumlahan dari matriks
% langkah 1 dan langkah 2
reducedMatrix = modifiedCostMatrix_row +  modifiedCostMatrix_col;
X = zeros(size(reducedMatrix));
[m,n] = size(reducedMatrix);

iterasi=0;
for i = 1:m*n
    iterasi=iterasi+1
    if sum(supply)==0 && sum(demand)==0
        break
    end
    %mengurutkan elemen di kolom dari terkecil ke terbesar
    Col = sort(reducedMatrix, 1);
    %mengurutkan elemen di baris dari terkecil ke terbesar
    Row = sort(reducedMatrix, 2);
    %penalty antara 2 nilai terbesar dalam setiap baris
reducedMatrix
```

```
    pRow = Row(:, end)-Row(:, end-1);
    %penalty antara 2 nilai terbesar dalam setiap kolom
reducedMatrix
    pCol = Col(end,:)-Col(end-1, :);
    %mengidentifikasi baris yang memiliki penalti tertinggi
    R = max(pRow);
    %mengidentifikasi kolom yang memiliki penalti tertinggi.
    C = max(pCol);
    %mencari index baris yang memiliki penalti tertinggi yang
setara dengan nilai maksimum antara R dan C
    Rmax = find(pRow==max(R));
    %mencari index kolom yang memiliki penalti tertinggi yang
setara dengan nilai maksimum antara R dan C
    Cmax = find(pCol==max(C));
    %mengambil semua nilai pada baris cost_matrix yang sesuai
dengan indeks di Rmax
    Cr = reducedMatrix(Rmax, :);
    %mengambil semua nilai pada kolom cost_matrix yang sesuai
dengan indeks di Cmax
    Cc = reducedMatrix(:, Cmax);

    if max(pRow)~= max(pCol)
        if max(pRow)>max(pCol)%pilih Row
            %indeks elemen dalam matriks reducedMatrix yang
memiliki nilai maksimum dalam Cr
            [rowind,
colind]=find(max(max(Cr))==reducedMatrix(Rmax, :));
            %mengisi elemen dalam baris Cr yang memiliki penalti
tertinggi. Ini dilakukan dengan mencari elemen dalam baris yang
memiliki
            %nilai tertinggi dalam Cr dan mengambil indeks baris
dan kolomnya.
            row1 = Rmax(rowind);
            col1 = colind;
        else
            [rowind, colind] = find(max(max(Cc))==reducedMatrix(:,
Cmax));
            %mengisi elemen dalam kolom Cc yang memiliki penalti
tertinggi.
            %Ini dilakukan dengan mencari elemen dalam kolom yang
memiliki nilai tertinggi dalam Cc dan mengambil indeks baris dan
kolomnya.
            row1=rowind;
            col1=Cmax(colind);
        end
        %menentukan jumlah maksimum yang dapat diangkut dari
supply di baris row1 ke demand di kolom col1
        x11=min(supply(row1), demand(col1));
        [val, ind] = max(x11);
        ii = row1(ind);
        jj = col1(ind);
    else
        [rowind1, colind1]=find(max(max(Cr))==reducedMatrix(Rmax,
:));
        row1 = Rmax(rowind1);
        col1 = colind1;
        C1 = reducedMatrix(row1, col1);
```

```
        [rowind2, colind2] = find(max(max(Cc))==reducedMatrix(:,
Cmax));
        row2=rowind2;
        col2=Cmax(colind2);
        C2 = reducedMatrix(row2, col2);

        if C1<C2
            x11 = min(supply(row2), demand(col2));
            [val, ind]=max(x11);
            ii=row2(ind);
            jj=col2(ind);
        else
            x11=min(supply(row1), demand(col1));
            [val, ind]=max(x11);
            ii=row1(ind);
            jj=col1(ind);
        end
    end
    y11 = min(supply(ii), demand(jj));
    X(ii, jj) = y11;
    supply(ii)=supply(ii)-y11;
    demand(jj) = demand(jj)-y11;

    if supply(ii)==0
        reducedMatrix(ii,:)=0;
    end
    if demand(jj)==0
        reducedMatrix(:,jj)=0;
    end
end

disp('Allocation')
disp(X)
total_cost = sum(sum(costMatrix.*X))
toc;
```

Lampiran 3. Syntax Kode Program Metode MODI

```
x=allocation_matrix;
[m,n]=size(c);
r=0.0001;
%MODI METHOD
fprintf('\nApplying MODI Method on Allocation:\n')


%% Start
iteration=0;
for q=1:m*n
    iteration=iteration+1
    %Degeneracy test
    num_allocated=0;
    for i=1:m
        for j=1:n
            if x(i,j)>0
                num_allocated=num_allocated+1;
            end
        end
    end
    disp(num_allocated)

    if num_allocated~=m+n-1
        fprintf('Note: MODI Method received degenerate solution.
Handling degeneracy ...\n')
        % Count the number of the basic cell on each row and
column
        remaining=m+n-1-num_allocated;
        for A=1:remaining
            for j=1:n
                countcol=0;
                for i=1:m
                    if x(i,j)>0
                        countcol=countcol+1;
                    end
                end
                x(m+1,j)=countcol;
            end
            for i=1:m
                countrow=0;
                for j=1:n
                    if x(i,j)>0
                        countrow=countrow+1;
                    end
                end
                x(i,n+1)=countrow;
            end


            % Assign adding one on the entering cell
            for j=1:n-1
                if x(m+1,j)==1
                    jenter=j;
                    for i=1:m-1
                        if x(i,n+1)==1
                            ienter=i;
```

```
                            break
                        end
                    end
                end
            end
            if remaining>0 && x(ienter,jenter)== 0
                x(ienter,jenter)=r;
                remaining=remaining-1
            else
                disp('Degeneracy Handled')
                break
            end
        end
    else
        fprintf('Non-Degenerate Solution ...\n')


    end

    %% Calculating values for U dan V
    U = ones(1, m, 'int32') * intmax('int32');
    V = ones(1, n, 'int32') * intmax('int32');
    % Mengatur elemen pertama dari array U menjadi 0
    U(1) = 0;
    % Menginisialisasi variabel num_calculated dengan nilai 1
    num_calculated = 1;

    while num_calculated<m+n
        for i=1:m
            for j=1:n
                if x(i,j)>0
                    if V(j) == intmax('int32') && U(i) ~=
intmax('int32')
                        V(j) = c(i, j) - U(i);
                        num_calculated = num_calculated + 1;
                    elseif U(i) == intmax('int32') && V(j) ~=
intmax('int32')
                        U(i) = c(i, j) - V(j);
                        num_calculated = num_calculated + 1;
                    end
                end
            end
        end
    end

    %% Finding the unallocated cell with maximum Indeks Perbaikan
    Ip=zeros(m,n);
    for i = 1:m
        for j = 1:n
            if x(i, j) <= 0
                Ip(i,j) = U(i) + V(j) - c(i, j);

            end
        end
    end
```

```
    max_opportunity =0;
    imax = -1;
    jmax = -1;

    for i = 1:m
        for j = 1:n
            if x(i, j) <= 0
                if Ip(i,j) > max_opportunity
                    max_opportunity = Ip(i,j);
                    imax = i;
                    jmax = j;
                end
            end
        end
    end

    if max_opportunity <= 0
        % Every cell has opportunity_cost <= 0: Optimal solution
found
        fprintf('Optimal Solution Reached in iteration:\n')
        disp(iteration)
        break;
    else
        % There exists at least one cell with opportunity_cost >
0: Optimal solution not reached
        fprintf('Finding loop starting from cell (%d, %d) :\n',
imax, jmax);
    end

    %% Entering a new basic variable add into the basic variable
matrix
    x1=zeros(m+1,n+1);
    x2=zeros(m+1,n+1);
    % Construct the equivalent basic variable matrix
    for j=1:n
        for i=1:m
            if x(i,j)>0
                x1(i,j)=x(i,j);
                x2(i,j)=x(i,j);
            end
        end
    end
    % Entering the new variable
    x1(imax,jmax)=inf;
    x2(imax,jmax)=inf;

    for j=1:n
        countcol=0;
        for i=1:m
            if x1(i,j)>0
                countcol=countcol+1;
            end
        end
        x1(m+1,j)=countcol;
        x2(m+1,j)=countcol;
    end

    for i=1:m
```

```
        countrow=0;
        for j=1:n
            if x1(i,j)>0
                countrow=countrow+1;
            end
        end
        x1(i,n+1)=countrow;
        x2(i,n+1)=countrow;
    end

    for p=1:m
        for i=1:m
            if x2(i,n+1)==1
                ieliminate=i;
                for j=1:n
                    if x2(ieliminate,j)>0
                        jeliminate=j;
                        x2(ieliminate,jeliminate)=0;% Eliminate
the basic variable on row
                        x2(ieliminate,n+1)=x2(ieliminate,n+1)-1; %
decrease the number of the basic variable on row one unit
                        x2(m+1,jeliminate)=x2(m+1,jeliminate)-1;%
decrease the number of the basic variable on column one unit
                    end
                end
            end
        end
        % Eliminate the basic variables that has only one on each
column
        for j=1:n
            if x2(m+1,j)==1
                jeliminate1=j;
                for i=1:m
                    if x2(i,jeliminate1)>0;
                        ieliminate1=i;
                        x2(ieliminate1,jeliminate1)=0
                        x2(m+1,jeliminate1)=x2(m+1,jeliminate1)-1;
                        x2(ieliminate1,n+1)=x2(ieliminate1,n+1)-1;
                    end
                end
            end
        end
        % Control the constructing loop path
        for j=1:n
            for i=1:m
                if (x2(i,n+1)==0 || x2(i,n+1)==2) && (x2(m+1,j)==0
|| x2(m+1,j)==2)
                    break
                end
            end
        end
    end
    %% Make +/-sign on basic variables in the loop path (x2)
    %1. Add - sign on basic variable on row(imax) and on basic
     variable on column (jmax)

    for j=1:n
        if  (x2(imax,j)~=0 && x2(imax,n+1)==2)
```

```
            jneg=j;
            jneg(1)=jmax;
            x2(imax,jneg)=(-1)*x2(imax,jneg);
            x2(m+1,jneg)=1;
            x2(imax,n+1)=1;
            for i=1:m
                if (x2(i,jneg)>0 && x2(m+1,jneg)==1)
                    ineg=i;
                end
            end
        end
    end

    for p=1:n
        for j=1:n
            if (j~=jneg && x2(ineg,j)>0 )&& (x2(ineg,n+1)==2)
                jneg1=j;
                x2(ineg,jneg1)=(-1)*x2(ineg,jneg1);
                x2(ineg,n+1)=1;
                x2(m+1,jneg1)=1;
                for i=1:m
                    if (x2(i,jneg1)>0 && x2(m+1,jneg1)==1)
                        ineg1=i;
                        ineg=ineg1;
                        jneg=jneg1;
                    end
                end
            end
        end
        % Control loop
        if  jneg1==jmax
            break
        end
    end
    %% Search the net smallest negative basic variable in the loop
path
    small=inf;
    for i = 1:m
        for j = 1:n
            % Check if the element is positive
            if x2(i, j) > 0
                % Update the smallest positive value if the
current element is smaller
                small = min(small, x2(i, j));
            end
        end
    end
    % Construct the loop path
    x3=zeros(m,n);
    for j=1:n
        for i=1:m
            x3(i,j)=x2(i,j);
        end
    end
    %% Add the smallest value to the positive basis variable and
subtract to the negative basic variable
    %inisialisasi matriks visited
    visited=false(m,n);
    for i=1:m
```

```
        for j=1:n
            x3(imax,jmax)=small;
            visited(imax,jmax)=true;
            if ~visited(i,j)
                if x3(i,j)~=0
                    if x3(i,j)<0
                        x3(i,j)=abs(x3(i,j))+small;
                    elseif x3(i,j)>0
                        x3(i,j)=x3(i,j)-small;
                        if x3(i,j)==0
                            x3(i,j)=inf;
                        end
                    end
                    %set visited flag to true for this element
                    visited(i,j)=true;
                end
            end
        end
    end
    %% Combine the new absolute loop path to the x matrix
    xpath=zeros(m,n);
    for j=1:n
        for i=1:m
            xpath(i,j)=x(i,j);
        end
    end
    for j=1:n
        for i=1:m
            if x3(i,j)~=0
                if x3(i,j)==inf
                    xpath(i,j)=0;
                else
                    xpath(i,j)=round(abs(x3(i,j)));
                end
            end
        end
    end
    %% Transfer x to xpath
    for j=1:n
        for i=1:m
            x(i,j)=round(xpath(i,j));
        end
    end
    %% The objective function
    Z=sum(sum(c.*xpath))
end
```

Lampiran 4. Data Matriks Masalah Transportasi

Tabel L. 1 Matriks Transportasi Pr01

Sumber	Tujuan				Persediaan
	D1	D2	D3	D4	
S1	7	5	27	22	60
S2	4	3	24	12	50
S3	6	16	60	20	40
S4	2	6	21	6	30
Permintaan	60	60	40	20	

Tabel L. 2 Matriks Transportasi Pr03

Sumber	Tujuan						Persediaan
	D1	D2	D3	D4	D5	D6	
S1	152	934	934	1738	913	956	1123
S2	1756	319	878	1317	1038	319	1256
S3	1330	1296	102	2251	443	205	1203
S4	525	1002	715	1955	692	739	1124
S5	1438	863	246	2506	329	246	1294
Permintaan	920	985	1255	950	890	1000	

Tabel L. 3 Matriks Transportasi Pr06

$[C_{ij}]_{2x94}$ = [18 36 57 43 33 85 31 72 43 13 7 7 83 68 40 42 39 14 52 94 36 99 82 20 93 6 73 13 60 10 39 90 59 69 90 36 71 17 5 31 40 10 22 9 54 15 16 9 59 68 21 38 67 71 52 28 48 47 44 42 12 63 20 100 39 43 56 71 67 40 62 21 3 80 98 28 33 28 4 57 52 79 52 77 35 14 89 33 80 11 28 22 48 74; 61 60 69 15 60 73 73 25 47 38 4 99 75 97 38 3 93 23 96 21 61 66 62 89 49 45 20 90 76 16 43 1 22 56 68 27 38 2 51 74 39 21 25 48 2 40 25 28 2 42 44 46 68 31 97 5 84 47 99 64 55 32 24 3 55 86 82 37 40 51 16 60 83 26 95 82 13 8 42 83 54 54 37 58 100 59 69 87 57 39 60 64 36 99]

$[S_i]_{2x1}$ = [17027, 28173]

$[D_j]_{1x94}$ = [662 759 867 789 625 264 703 765 20 599 380 836 364 659 843 429 86 21 554 576 374 574 289 245 90 659 337 257 168 477 333 631 674 525 512 344 690 720 811 12 675 222 882 154 788 274 607 136 938 889 917 383 543 59 188 182 700 555 808 862 285 174 322 52 984 205 88 805 819 109 429 957 597 218 249 589 295 346 410 226 460 560 327 374 946 201 60 228 461 503 472 956 534 674]

Tabel L. 4 Matriks Transportasi Pr02

Sumber	Tujuan												Persediaan
	D1	D2	D3	D4	D5	D6	D7	D8	D9	D10	D11	D12	
S1	209	220	200	184	207	208	204	204	197	202	192	205	3920
S2	213	184	201	223	195	203	200	203	204	198	225	202	7840
Permintaan	1300	1250	1200	1150	1000	970	950	900	880	850	680	550	

Tabel L. 5 Matriks Transportasi Pr05

Sumber	Tujuan										Persediaan
	D1	D2	D3	D4	D5	D6	D7	D8	D9	D10	
S1	41	51	45	55	49	60	63	69	47	33	571
S2	57	45	63	29	85	10	79	51	20	17	144
S3	60	80	51	62	66	79	17	86	50	49	336
S4	24	76	14	21	57	87	70	58	37	33	102
S5	61	49	30	71	88	93	57	60	46	37	232
S6	73	73	85	64	18	99	25	71	85	48	100
S7	48	96	11	15	92	55	95	43	46	20	213
S8	86	81	88	15	19	34	63	31	45	55	185
S9	76	74	17	23	57	19	50	62	42	74	261
S10	42	19	70	11	23	56	95	88	22	32	205
Permintaan	183	459	641	664	232	390	325	414	259	141	

Tabel L. 6 Matriks Transportasi Pr07

Sumber	Tujuan															Persediaan
	D1	D2	D3	D4	D5	D6	D7	D8	D9	D10	D11	D12	D13	D14	D15	D16	
S1	42	82	58	14	73	46	56	66	72	12	22	40	22	38	30	14	362
S2	77	66	21	46	87	19	53	85	34	87	37	81	29	37	47	64	127
S3	60	20	55	82	28	10	40	25	38	48	13	60	73	73	13	19	198
S4	27	90	38	83	81	82	17	82	22	81	69	30	80	74	21	45	159
S5	40	86	64	23	86	23	44	84	46	79	45	75	54	38	16	77	238
S6	67	73	37	57	25	78	61	68	78	79	42	54	82	37	73	18	162
S7	12	60	36	46	60	48	20	40	82	40	17	26	86	42	24	73	144
S8	33	86	27	88	79	35	65	66	44	40	76	79	34	55	36	42	341
S9	60	19	40	72	35	20	77	25	32	59	84	28	50	66	34	62	240
S10	18	80	45	86	24	64	78	85	66	63	76	58	41	76	18	59	387
S11	40	71	47	29	68	83	58	28	64	34	19	30	86	47	85	11	372
S12	50	65	31	15	57	37	74	58	72	18	24	82	85	24	27	57	200
S13	36	68	81	28	47	26	68	12	42	12	74	61	69	88	44	44	240
Permintaan	373	160	308	195	316	212	321	119	298	235	257	348	395	248	285	217	

Tabel L. 7 Matriks Biaya Transportasi Pr08

$[C_{ij}]_{10x94}$	=	[13 93 61 63 30 96 77 26 93 99 65 52 47 46 28 82 80 39 20 94 14 71 51 72 51 92 59 13 23 74 97 15 18 85 80 79 58 17 47 77 64 89 61 80 78 44 23 26 88 10 15 84 78 68 93 58 90 35 78 40 98 63 90 30 22 59 97 97 71 64 85 27 40 20 36 7 59 60 78 96 77 97 26 73 68 57 65 3 49 88 21 80 29 30; 96 35 79 30 43 64 9 26 16 4 20 36 71 77 95 44 30 32 37 27 92 52 49 80 97 55 16 50 98 98 69 68 43 99 85 94 72 10 18 27 66 50 58 96 84 95 84 57 37 47 98 88 12 27 70 96 27 85 66 20 43 38 4 21 58 46 67 27 33 60 39 38 95 21 76 65 78 48 12 63 20 1 98 24 38 43 14 1 1 24 74 13 66 24; 66 44 86 24 62 99 98 83 10 65 75 72 69 5 50 87 34 92 11 33 37 20 46 5 31 78 97 66 8 25 96 84 24 52 90 15 77 92 51 32 27 67 31 92 1 31 55 5 86 58 11 16 46 19 68 80 61 54 58 13 21 69 75 28 42 1 72 98 18 29 83 3 74 69 86 75 44 63 40 48 40 97 26 11 10 9 16 40 97 96 69 27 62 76; 60 79 49 48 57 45 58 71 29 47 83 4 3 75 51 32 35 50 74 11 25 58 100 30 46 19 39 14 4 88 29 74 29 39 80 87 90 72 67 57 35 1 86 53 40 86 47 9 46 10 4 97 33 13 30 90 97 87 90 28 62 81 54 33 71 6 35 85 20 16 85 43 20 2 66 19 85 68 30 81 74 51 92 66 84 52 31 33 82 33 56 100 92 42; 24 65 70 28 3 79 72 65 13 75 62 7 6 51 40 7 8 42 93 63 87 29 90 68 27 92 45 4 53 96 19 18 74 94 43 30 35 95 6 64 17 95 66 4 71 89 39 29 24 98 71 32 32 52 52 76 54 22 58 46 41 43 46 72 7 6 45 8 43 59 64 81 61 1 91 34 98 84 62 69 9 36 93 73 46 14 45 83 65 40 80 78 20 25; 44 80 81 96 93 86 91 15 17 32 26 97 3 60 91 97 84 11 7 67 28 52 43 76 42 73 9 43 79 31 74 24 4 46 35 35 16 12 71 3 52 32 72 26 48 67 87 61 86 34 38 36 76 61 80 12 80 18 35 37 10 38 38 90 55 30 11 40 69 94 52 52 98 13 4 27 100 100 73 92 42 33 48 32 9 4 20 66 78 4 35 81 6 84; 13 97 10 2 33 4 11 35 77 89 100 50 82 78 65 47 97 89 3 20 99 19 95 14 68 99 44 99 26 89 22 10 98 61 28 12 94 14 23 27 22 54 59 84 14 46 51 83 8 53 53 77 56 43 68 3 44 6 32 6 80 12 60 1 97 87 37 37 97 99 70 78 50 96 81 39 75 31 87 7 71 32 10 52 8 12 51 16 23 54 74 44 36 98; 47 45 77 63 43 34 54 59 81 59 66 52 7 74 23 65 29 97 98 70 56 42 86 82 83 14 16 19 78 95 31 31 17 67 22 95 34 82 24 11 48 60 55 25 90 43 31 65 45 55 31 43 34 96 51 19 1 59 84 68 49 28 86 57 93 45 10 24 6 81 82 7 20 96 83 29 17 10 8 12 40 2 46 16 61 5 56 84 47 11 83 1 16 36; 46 4 18 85 86 30 97 1 84 15 39 13 73 86 100 80 76 1 69 32 92 92 54 63 85 90 55 58 36 65 63 7 37 53 71 66 86 70 95 78 10 94 100 82 43 67 49 92 52 30 55 49 93 39 44 94 36 33 44 88 90 84 69 74 81 36 3 65 79 40 41 17 75 16 24 41 93 77 32 51 50 83 72 51 50 24 24 38 47 81 76 56 51 29; 75 16 79 6 81 36 100 54 90 80 45 41 55 32 68 47 4 10 89 31 43 94 23 55 12 48 43 23 20 56 60 15 84 29 78 75 90 34 96 2 39 41 22 99 86 72 33 47 13 13 82 80 13 18 11 94 55 76 82 35 61 46 75 7 38 29 42 95 13 5 4 86 92 83 67 83 75 53 19 100 79 92 19 11 73 40 13 41 72 6 20 32 94 74]
$[S_i]_{10x1}$	=	[4486, 4876, 4297, 4086, 4413, 4557, 4089, 4602, 4839, 6562]
$[D_j]_{1x94}$	=	[523 818 818 314 770 517 503 285 740 143 332 905 961 494 800 31 784 963 636 59 5 460 245 451 299 297 157 937 704 903 148 287 723 338 711 603 970 265 429 763 522 387 667 369 430 43 186 584 437 721 189 958 735 282 681 423 291 76 49 494 621 606 271 974 830 937 240 607 170 385 886 483 777 607 699 177 981 141 22 11 733 854 848 7 565 709 238 71 749 512 609 179 473 220]

Tabel L. 8 Matriks Biaya Transportasi Pr04

Sumber	Tujuan								Persediaan
	D1	D2	D3	D4	D5	D6	D7	D8	
S1	84	97	48	71	35	49	74	97	635
S2	92	97	93	78	14	44	78	40	676
S3	21	24	82	77	18	79	35	63	428
S4	93	98	97	45	84	82	71	30	183
S5	67	97	69	69	73	27	69	78	189
S6	18	54	13	25	38	54	24	33	254
S7	35	82	87	74	96	50	20	56	605
S8	59	22	94	12	13	68	55	73	252
S9	84	97	48	71	35	49	74	97	635
S10	92	97	93	78	14	44	78	40	676
Permintaan	589	246	658	310	218	250	470	384	