DAFTAR PUSTAKA

Ahmed, M. M., Khan, A. R., Ahmed, F., & Uddin, M. S. (2016). Incessant Allocation Method for Solving Transportation Problems. *American Journal of Operations Research*, *6*(3), 236-244.

Amaliah, B., Fatichah, C., & Suryani, E. (2022). A New Heuristic Method of Finding the Initial Basic Feasible Solution to Solve the Transportation Problem. *Journal of King Saud University-Computer and Information Sciences*, *34*(5), 2298-2307.

Astuti, N. D., Utomo, R. H. S., & Suryoto. (2016). Solusi Masalah Transportasi Menggunakan *Tocm-sum Approach* dengan Indikator Distribusi. *Jurnal Matematika Undip*, *19*(3), 121-126.

Chandra, T. (2016). Penerapan Algoritma *North West Corner* dalam Penyelesaian Masalah Transportasi. *Jurnal Times*, *5*(1), 12-16.

Damian, R. T., Shofa, S. N., Tandean, K. Y., Hernanda, D. A., Agusetiawan, A., & Yuliawati, E. (2023). Penerapan Model Programa Linier untuk Memaksimalkan Profit Toko Oleh-Oleh Khas Surabaya Honest. In *Prosiding SENASTITAN: Seminar Nasional Teknologi Industri Berkelanjutan,* (3): 1-8.

Fikri, A. J., Aini, S., Sukandar, R. S., Safiyanah, I., & Listiasari, D. (2021). Optimalisasi Keuntungan Produksi Makanan Menggunakan Pemrograman Linier Melalui Metode Simpleks. *Jurnal Bayesian: Jurnal Ilmiah Statistika dan Ekonometrika*, *1*(1), 1-16.

Hani, N., & Harahap, E. (2021). Optimasi Produksi T-Shirt Menggunakan Metode Simpleks. *Matematika: Jurnal Teori Dan Terapan Matematika*, *20*(2), 27-32.

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.

Jiantari, N. K., Gandhiadi, G. K., dan Widiastuti, R. S. (2022). Analisis Perbandingan Metode *Exponential Approach* dan Metode *Improved Zero Point* untuk Meminimumkan Biaya Pendistribusian. *Jurnal Matematika*, *11*(3): 174-183.

Kamalia, A., & Soelistyo, R. H. (2022). Penyelesaian Masalah Transportasi Menggunakan Metode RCWMCAM dan Metode MODI. *Techno. Com*, *21*(3), 689-699.

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..

Mathirajan, M., Reddy, S., & Rani, M. V. (2021). An Experimental Study of Newly Proposed Initial Basic Feasible Solution Methods for A Transportation Problem. *Opsearch*, *59*(1), 102-145.

Padillah, C., Adelia, E., & Sapta, A. (2018). Optimasi Distribusi Roti Menggunakan Model *Stepping Stone* (Studi Kasus: Pabrik Roti Gedangan, Asahan). *Journal of Science and Social Research*, *1*(2), 148-152.

Pandian, P., & Natarajan, G. (2010). A New Algorithm for Finding A Fuzzy Optimal Solution for Fuzzy Transportation Problems. *Applied mathematical sciences*, *4*(2), 79-90.

Raharjo, W. S., dan Wulan, E. R. (2017) Penggunaan Metode *Maximum Supply with Minimum Cost* untuk Mendapatkan Solusi Layak Awal Masalah Transportasi. *Jurnal Kubik*, *2*(2): 11-16.

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

Samuel, A. E. (2012). *Improved zero point method* (IZPM) for the Transportation Problems. *Applied mathematical sciences*, *6*(109), 5421-5426.

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

Siregar, B. H., & Mansyur, A. (2021). Program Linear dan Aplikasinya pada Berbagai *Software*. Jakarta Timur: Bumi Aksara.

Taha, H. A. (2017). Operation Reseacrh an Introduction Tenth Edition Global Edition. Fayetteville: Pearson Education.

Utami, K. A., & Dewi, M. P. (2019). Optimasi Pendistribusisan Air Menggunakan *Improved Zero Point Method* (Studi Kasus di PDAM Tirta Kepri). *Journal of Mathematics UNP*, *4*(1).

LAMPIRAN

Lampiran 1. *Source Code* Program *Improved Zero Point Method* (IZPM)

```
% PROGRAM IMPROVED ZERO POINT METHOD (IZPM)
clear all; clc;
tic

% Langkah 1:
% Membuat tabel transportasi
cost = [4, 6, 8, 6; 3, 5, 2, 5; 3, 9, 6, 5]
supply = [700, 400, 600]
demand = [400, 450, 350, 500]

% Mengecek seimbang atau tidak seimbang
if sum(supply) == sum(demand)
    disp('Masalah Transportasi Telah Seimbang')
else
    disp('Masalah Transportasi Tidak Seimbang')
    if sum(supply) < sum(demand)
        cost(end+1, :) = zeros(1, size(demand, 2))
        supply(end+1) = sum(demand) - sum(supply)
    elseif sum(demand) < sum(supply)
        cost(:, end+1) = zeros(1, size(supply, 2))
        demand(end+1) = sum(supply)- sum(demand)
    end
end

[m,n] = size(cost);
disp('Tabel Transportasi')
disp(cost)
matrix_allocation=zeros(size(cost));
c = cost;

% Langkah 2:
% Mengurangi setiap elemen baris dengan elemen terkecil pada baris
tersebut
for i = 1:m
    minimum_cost = min(cost(i,:));
    cost(i,:)= cost(i,:) - minimum_cost;
end
disp('Hasil Pengurangan Setiap Elemen Baris dengan Nilai
Terkecil')
disp(cost)

% Mengurangi setiap elemen kolom dengan elemen terkecil pada kolom
for j = 1:n
    minimum_cost = min(cost(:,j));
    cost(:,j)= cost(:,j) - minimum_cost;
end
disp('Hasil Pengurangan Setiap Elemen Kolom dengan Elemen
Terkecil')
disp(cost)

iterasi1 = 0;
for i = 1:m*n
```

```
    iterasi1 = iterasi1 + 1
    both_conditions_met = false;

    % Langkah 3
    % Memeriksa jumlah permintaan pada setiap kolom
    disp('Hasil Periksa Kolom')
    for j = 1:n ;
        if sum(cost(:, j) == 0) > 0;
            col_demand = sum(demand(j));
            num_row_supply = sum(supply(cost(:, j) == 0));
            if col_demand <= num_row_supply;
                disp([num2str(col_demand), ' <= ',
num2str(num_row_supply)]);
            else
                disp([num2str(col_demand), ' > ',
num2str(num_row_supply)]);
            end
            if col_demand <= num_row_supply;
                both_conditions_met = true;
            else
                both_conditions_met = false;
                break
            end
        end
    end

    if both_conditions_met;
        %langkah 3:
        % Memeriksa jumlah persediaan pada setiap baris
        disp('Hasil Periksa Baris');
        for i = 1:m;
            if sum(cost(i, :) == 0) > 0;
                row_supply = sum(supply(i));
                num_col_demand = sum(demand(cost(i, :) == 0));
                if row_supply <= num_col_demand;
                    disp([num2str(row_supply), ' <= ',
num2str(num_col_demand)]);
                else
                    disp([num2str(row_supply), ' > ',
num2str(num_col_demand)]);
                end
                if row_supply <= num_col_demand
                    both_conditions_met = true;
                else
                    both_conditions_met = false;
                    break
                end
            end
        end
    end

    if both_conditions_met
        break
    else
        disp('Kedua syarat tidak terpenuhi, lanjut ke langkah 4.')
        temp = cost;

        % Langkah 4:
```

```
        % Menutup semua elemen nol dengan garis mendatar & tegak
        [row, col] = size(temp);
        sum_zeros_row = sum(temp == 0, 2);
        max_zeros_row = max(sum_zeros_row);
        sum_zeros_col = sum(temp == 0, 1);
        max_zeros_col = max(sum_zeros_col);
        indices_closed_rows = [];
        indices_closed_cols = [];

        while max_zeros_col > 0 || max_zeros_row > 0
            if max_zeros_col > max_zeros_row;
                col_to_close = find(sum_zeros_col ==
max_zeros_col, 1);
                temp(:, col_to_close) = Inf;
                sum_zeros_col(col_to_close) = Inf;
                indices_closed_cols = [indices_closed_cols,
col_to_close]
            elseif max_zeros_row > max_zeros_col;
                row_to_close = find(sum_zeros_row ==
max_zeros_row, 1);
                temp(row_to_close, :) = Inf;
                sum_zeros_row(row_to_close) = Inf;
                indices_closed_rows = [indices_closed_rows,
row_to_close];
            else
                num_row_closed =
numel(unique(indices_closed_rows));
                num_col_closed =
numel(unique(indices_closed_cols));
                if num_row_closed == 0 && num_col_closed == 0
                    row_to_close = find(sum_zeros_row ==
max_zeros_row, 1);
                    temp(row_to_close, :) = Inf;
                    sum_zeros_row(row_to_close) = Inf;
                    indices_closed_rows = [indices_closed_rows,
row_to_close];
                elseif num_row_closed > num_col_closed
                    col_to_close = find(sum_zeros_col ==
max_zeros_col, 1);
                    temp(:, col_to_close) = Inf;
                    sum_zeros_col(col_to_close) = Inf;
                    indices_closed_cols = [indices_closed_cols,
col_to_close];
                elseif num_col_closed > num_row_closed
                    row_to_close = find(sum_zeros_row ==
max_zeros_row, 1);
                    temp(row_to_close, :) = Inf;
                    sum_zeros_row(row_to_close) = Inf;
                    indices_closed_rows = [indices_closed_rows,
row_to_close];
                else
                    col_to_close = find(sum_zeros_col ==
max_zeros_col, 1);
                    temp(:, col_to_close) = Inf;
                    sum_zeros_col(col_to_close) = Inf;
                    max_zeros_col = max(sum_zeros_col);
                end
            end
            sum_zeros_row = sum(temp == 0, 2);
```

```
            max_zeros_row = max(sum_zeros_row);
            sum_zeros_col = sum(temp == 0, 1);
            max_zeros_col = max(sum_zeros_col);
        end
        disp('Menutup Semua Elemen Nol dengan Garis Mendatar dan
Tegak')
        disp(temp)

        % Langkah 5:
        % a) Menemukan nilai biaya tereduksi terkecil
        min_element = min(min(temp));

        [row_Inf, col_Inf] = find(temp == Inf);
        for i = 1:length(row_Inf)
            row_index = row_Inf(i);
            col_index = col_Inf(i);
            if temp(row_index, col_index) == Inf &&
all(temp(row_index, :) == Inf) && all(temp(:, col_index) == Inf);
                element = cost(row_index, col_index);
            end
        end

        % b) mengurangkan elemen terkecil kesemua elemen yang
tidak tertutup oleh garis
        for i = 1:size(cost,1);
            for j = 1:size(cost,2);
                if(temp(i,j)~= inf);
                    cost(i,j) = cost(i,j)-min_element;
                end
            end
        end

        disp('Tabel Hasil Pengurangan dengan Elemen Terkecil');
        disp(cost);

        % b) menambahkan elemen terkecil kesemua elemen yang
ditutup oleh dua garis
        [row_Inf, col_Inf] = find(temp == Inf);
        for i = 1:length(row_Inf)
            row_index = row_Inf(i);
            col_index = col_Inf(i);
            if temp(row_index, col_index) == Inf &&
all(temp(row_index, :) == Inf) && all(temp(:, col_index) == Inf);
                cost(row_index, col_index) = cost(row_index,
col_index) + min_element;
            end
        end
        disp('Tabel Setelah Pengurangan dan Penambahan Elemen
Terkecil')
        disp(cost);
    end
end
disp('Jumlah Iterasi Pada Syarat 3 adalah: ')
disp(iterasi1)

% Langkah 6 sampai 8
iterasi2 = 0;
for i = 1:m*n
```

```
    iterasi2 = iterasi2 + 1

    %Langkah 6:
    %Memilih Sel dengan Elemen Terbesar
    max_value = max(cost(:));
    if max_value > 0
        [max_rows, max_cols] = find(cost == max_value);
        max_row = min(max_rows);
        max_value_col = max_cols(max_rows == max_row);
        max_col = max_value_col(1);

        %Langkah 7:
        %Memilih Sel pada Baris/Kolom yang Memiliki Elemen
Terkecil
        %Perhitungan pada Baris
        min_value_row = min(cost(max_row, :));
        num_row_indices = sum(cost(max_row, :) == 0);
        [~, row_indices_col_row] = find(cost(max_row, :) == 0);
        row_indices_row = ones(size(row_indices_col_row)) *
max_row;

        %Perhitungan pada Kolom
        min_value_col = min(cost(:,max_col));
        num_col_indices = sum(cost(:, max_col) == 0);
        [row_indices_col, ~] = find(cost(:, max_col) == 0);
        col_indices_col = ones(size(row_indices_col)) * max_col;

        if num_row_indices == 1
            row = row_indices_row;
            col = row_indices_col_row;
        elseif num_col_indices == 1
            col = col_indices_col;
            row = row_indices_col;
        elseif num_row_indices == 1 && num_col_indices == 1
            row = row_indices_col;
            col = col_indices_col;
        elseif num_row_indices == 0 && num_col_indices ==0
            row = max_row;
            col = max_col;
        else
            selected_supply_row = supply(row_indices_row);
            selected_demand_row = demand(row_indices_col_row);
            total_supply_demand_row = selected_supply_row +
selected_demand_row;

            selected_supply_col = supply(row_indices_col);
            selected_demand_col = demand(col_indices_col);
            total_supply_demand_col = selected_supply_col +
selected_demand_col;

            sum_greatest_values = max([total_supply_demand_row,
total_supply_demand_col]);
            index_row = find(total_supply_demand_row ==
sum_greatest_values, 1);
            index_col = find(total_supply_demand_col ==
sum_greatest_values, 1);
```

```
            if isempty(index_row) && ~isempty(index_col);
                disp('sum_greatest_values berada pada
total_supply_demand_col');
                row = row_indices_col(index_col);
                col = col_indices_col(index_col);
            elseif ~isempty(index_row) && isempty(index_col);
                disp('sum_greatest_values berada pada
total_supply_demand_row');
                row = row_indices_row(index_row);
                col = row_indices_col_row(index_row);
            else
                disp('sum_greatest_values tidak ditemukan dalam
kedua array');
            end
        end
    else
        [max_rows, max_cols] = find(cost == max_value);
        selected_supply= supply(max_rows);
        current_demand = demand(max_cols);
        total_supply_demand = selected_supply + current_demand;
        max_values= max(total_supply_demand);
        index_values = find(total_supply_demand == max_values);
        row = max_rows(index_values);
        col = max_cols(index_values);
    end

    allocation = min(supply(row), demand(col));
    supply(row) = supply(row) - allocation;
    demand(col) = demand(col) - allocation;

    %Langkah 8:
    %Membentuk Tabel Transportasi yang Telah di Perbaiki
    matrix_allocation(row,col) = allocation

    if sum(supply) == 0 && sum(demand) == 0
        break
    end

    if supply(row)==0
        cost(row,:)= NaN;
    end
    if demand(col)==0
        cost(:,col)= NaN;
    end


end
totalCost=sum(sum(c.*matrix_allocation));
disp('Hasil Alokasi')
disp(matrix_allocation)
disp('Total Cost:')
disp(totalCost)
disp('Jumlah Iterasi Pada Proses Alokasi adalah :')
disp(iterasi2)

toc
```

Lampiran 2. *Source Code* Program *Row Column Weighted Minimum Cost Allocation Method* (RCWMCAM)

```
% PROGRAM Row Column Weighted Minimum Cost Allocation Method
(RCWMCAM)
tic
clear all; clc;

% Langkah 1:
% Membuat tabel transportasi
cost = [4, 6, 8, 6; 3, 5, 2, 5; 3, 9, 6, 5]
supply = [700, 400, 600]
demand = [400, 450, 350, 500]

% Mengecek seimbang atau tidak seimbang
if sum(supply) == sum(demand)
    disp('Masalah Transportasi Telah Seimbang')
else
    disp('Masalah Transportasi Tidak Seimbang')
    if sum(supply) < sum(demand)
        cost(end+1, :) = zeros(1, size(demand, 2))
        supply(end+1) = sum(demand) - sum(supply)
    elseif sum(demand) < sum(supply)
        cost(:, end+1) = zeros(1, size(supply, 2))
        demand(end+1) = sum(supply)-sum(demand)
    end
end


[m,n] = size(cost);
matrix_allocation =zeros(size(cost));
c = cost;

while any(cost(:) ~= Inf)
    for i=1:m
        if nnz(cost(i, :)) == 0
            continue
        end
        non_zero_element = cost(i, cost(i, :) >= 0);
        min_cost_row(i, :) = min( non_zero_element);
        [row, col] = find(cost(i,:) == min_cost_row(i, :));
        col = col(1); %
        FQ_row(i,:) = min(supply(i), demand(col));
        MCA_row(i,:) = min_cost_row(i, :).*FQ_row(i,:);
        penalty_rows = zeros(m, 1);
        non_inf_elements = cost(i, isfinite(cost(i, :)) & cost(i,
:) > 0);
        num_non_inf_elements = numel(non_inf_elements);
        if num_non_inf_elements > 1
            sorted_row = sort(non_inf_elements);
            min_cost_row = sorted_row(1);
            second_min_cost_row = sorted_row(find(sorted_row ~=
min_cost_row, 1));
            if isempty(second_min_cost_row)
                penalty_row = min_cost_row;
```

```
            elseif num_non_inf_elements == 2 &&
numel(unique(sorted_row)) == 1;
                penalty_row = unique(sorted_row);
            else
                penalty_row = second_min_cost_row - min_cost_row;
            end
        elseif num_non_inf_elements == 1
            penalty_row = non_inf_elements;
        else
            penalty_row = 0;
        end
        penalty_rows(i) = penalty_row;
        WMCA_row(i,:) = MCA_row(i,:).*penalty_rows(i);
    end

    for j=1:n
        if nnz(cost(:, j)) == 0;
            continue
        end

        non_zero_element = cost(cost(:, j)>=0,j);
        min_cost_col(:, j) = min(non_zero_element);
        [row, col] = find(cost(:, j) == min_cost_col(:, j));
        row=row(1);
        FQ_col(:,j) = min(supply(row), demand(j));
        MCA_col(:,j) = min_cost_col(:,j).*FQ_col(:,j);

        penalty_cols = zeros(n, 1);
        non_inf_elements = cost(isfinite(cost(:, j)) & cost(:, j)
> 0, j);
        num_non_inf_elements = numel(non_inf_elements);

        if num_non_inf_elements > 1
            sorted_col = sort(non_inf_elements);
            min_cost_col = sorted_col(1);
            second_min_cost_col = sorted_col(find(sorted_col ~=
min_cost_col, 1));
            if isempty(second_min_cost_col);
                penalty_col = min_cost_col;
            elseif num_non_inf_elements == 2 &&
numel(unique(sorted_col)) == 1;
                penalty_col = unique(sorted_col);
            else
                penalty_col = second_min_cost_col - min_cost_col;
            end
        elseif num_non_inf_elements == 1
            penalty_col = non_inf_elements;
        else
            penalty_col = 0;
        end
        penalty_cols(j) = penalty_col;
        WMCA_col(:,j) = MCA_col(:,j).*penalty_col;
    end

    [~, baris_terbesar] = max(WMCA_row);
    [~, kolom_terbesar] = max(WMCA_col);

    if max(WMCA_row) > max(WMCA_col);
```

```
        nilai_WMCA_terbesar = max(WMCA_row);
        index_terbesar = baris_terbesar;
        [~, supply_index] = min(cost(index_terbesar, :));
        allocated_supply = min(supply(index_terbesar),
demand(supply_index));

        if allocated_supply == demand(supply_index);
            cost(:, supply_index) = Inf;
        end

        baris = baris_terbesar;
        kolom = supply_index;

        supply(baris) = supply(baris) - allocated_supply;
        demand(kolom) = demand(kolom) - allocated_supply;
        matrix_allocation(baris, kolom) = allocated_supply;
    else
        nilai_WMCA_terbesar = max(WMCA_col);
        index_terbesar = kolom_terbesar;
        [~, demand_index] = min(cost(:, index_terbesar));
        allocated_demand =
min(supply(demand_index),demand(kolom_terbesar));

        baris = demand_index;
        kolom = kolom_terbesar;

        if allocated_demand == supply(baris);
            cost(baris, :) = Inf;
        end

        supply(baris) = supply(baris) - allocated_demand;
        demand(kolom) = demand(kolom)-allocated_demand;
        matrix_allocation(baris, kolom) = allocated_demand;
    end

    if supply(baris)==0
        cost(baris,:)= Inf;
    end
    if demand(kolom)==0
        cost(:,kolom)= Inf;
    end

    if all(cost(:) == Inf)
        disp ('semua biaya telah habis, maka iterasi dihentikan')
        break
    end
end

totalCost=sum(sum(c.*matrix_allocation));
disp('Hasil Alokasi')
disp(matrix_allocation)
disp('Total Cost:')
disp(totalCost)

toc
```

Lampiran 3. *Source Code* Program Membangkitkan Bilangan Acak

```
% PROGRAM MEMBANGKITKAN BILANGAN ACAK
clear all; clc;

% 1) Membangkitkan Matriks Ukuran 8 x 8
cost = randi([1, 50], 8, 8)
supply = randi([5,50], 1, 8)
demand = randi([5,50], 1, 8)

% 2) Membangkitkan Matriks Ukuran 9 x 10
cost = randi([1, 50], 9, 10)
supply = randi([5,50], 1, 9)
demand = randi([5,50], 1, 10)

% 3) Membangkitkan Matriks Ukuran 20 x 20
cost = randi([1, 50], 20, 20)
supply = randi([5,50], 1, 20)
demand = randi([5,50], 1, 20)

% 4) Membangkitkan Matriks Ukuran 23 x 25
cost = randi([1, 50], 23, 25)
supply = randi([5,50], 1, 23)
demand = randi([5,50], 1, 25)
```

Lampiran 4. Matriks Tabel Transportasi

Tabel L. 3. 1 Kasus Tidak Seimbang 3 x 5

Sumber	Tujuan					Persediaan
	D1	D2	D3	D4	D5	
S1	7	11	6	7	5	150
S2	1	3	1	9	1	200
S3	8	9	10	14	4	125
Permintaan	80	100	75	45	125	

Tabel L. 3. 2 Kasus Seimbang 5 x 5

Sumber	Tujuan					Persediaan
	D1	D2	D3	D4	D5	
S1	73	40	9	79	20	8
S2	62	93	96	8	13	7
S3	96	65	80	50	65	9
S4	57	58	29	12	87	3
S5	56	23	87	18	12	5
Permintaan	6	8	10	4	4	

Tabel L. 3. 3 Kasus Seimbang 8 x 8

Sumber	Tujuan								Persediaan
	D1	D2	D3	D4	D5	D6	D7	D8	
S1	20	3	21	4	8	49	28	33	28
S2	34	25	46	10	28	12	30	50	48
S3	38	23	5	21	40	20	4	34	32
S4	9	25	10	15	23	12	29	22	34
S5	26	9	48	36	50	14	45	33	37
S6	50	19	6	12	22	21	24	30	44
S7	36	45	20	48	16	50	20	38	22
S8	5	38	15	16	13	46	6	33	37
Permintaan	34	26	29	41	20	55	54	23	

Tabel L. 3. 4 Kasus Tidak Seimbang 9 x 10

Sumber	Tujuan										Persediaan
	D1	D2	D3	D4	D5	D6	D7	D8	D9	D10	
S1	42	10	18	35	45	25	28	45	16	29	26
S2	24	50	43	38	2	13	10	48	35	33	33
S3	44	25	49	24	48	4	47	43	26	41	22
S4	33	26	15	35	45	9	14	20	37	39	44
S5	28	39	49	36	11	34	12	5	33	15	27
S6	26	34	31	11	50	36	26	37	4	10	26
S7	22	37	10	8	13	43	13	35	50	43	18
S8	41	12	24	25	32	14	1	22	10	33	30
S9	33	20	7	50	26	2	49	8	24	31	31
Permintaan	11	38	14	17	37	13	24	14	11	42	

Tabel L. 3. 5 Kasus Seimbang 20 x 20

Sumber	Tujuan																				Persediaan
	D1	D2	D3	D4	D5	D6	D7	D8	D9	D10	D11	D12	D13	D14	D15	D16	D17	D18	D19	D20	
S1	18	42	48	21	17	41	20	38	25	39	44	5	10	18	22	45	42	46	34	11	5
S2	23	31	30	18	6	14	19	27	19	45	15	7	16	15	37	15	17	50	33	28	31
S3	43	22	33	8	5	9	34	49	8	24	5	50	6	5	24	13	38	27	30	24	33
S4	46	47	9	13	29	16	4	13	18	30	23	49	11	7	6	28	46	14	17	15	32
S5	39	2	33	41	24	6	32	29	3	24	37	44	17	24	29	42	26	23	12	46	20
S6	2	10	49	14	45	18	31	30	16	37	20	39	32	27	23	38	22	1	30	18	42
S7	47	22	43	9	8	28	35	35	12	15	40	29	31	18	34	10	12	20	39	28	22
S8	29	12	27	50	4	5	49	31	20	7	33	3	16	30	15	43	19	28	11	4	45
S9	45	35	27	34	17	32	28	16	14	45	2	41	4	18	37	35	50	45	38	23	5
S10	45	16	24	4	11	26	17	17	24	12	5	24	26	42	44	4	15	14	24	35	42
S11	39	23	38	24	11	1	33	8	48	42	29	23	28	44	2	47	7	36	50	4	20
S12	9	8	47	4	3	36	36	30	17	22	47	47	37	14	38	28	26	3	20	29	19
S13	23	41	45	20	50	35	20	29	21	21	8	13	47	2	35	27	37	39	50	23	18
S14	32	48	18	32	11	29	46	41	20	45	31	8	1	19	44	17	16	37	26	1	16
S15	24	1	43	45	20	41	5	11	17	16	19	2	9	24	13	37	4	20	37	39	42
S16	25	23	38	40	4	2	2	9	30	38	41	12	46	17	42	16	44	29	13	28	16
S17	35	28	12	30	23	17	13	34	25	42	29	41	5	35	40	11	33	19	49	1	23
S18	2	17	43	35	38	21	17	16	27	26	50	3	18	16	41	8	8	19	47	18	35
S19	5	44	12	50	8	44	37	36	10	16	33	45	14	13	5	5	33	37	9	31	46
S20	1	8	38	8	26	33	45	7	47	41	10	19	10	38	32	10	10	43	4	8	50
Permintaan	30	7	48	20	8	17	9	33	28	46	15	37	50	44	33	33	28	36	23	17	

Tabel L. 3. 6 Kasus Tidak Seimbang 23 x 25

Sumber	Tujuan																									Persediaan
	D1	D2	D3	D4	D5	D6	D7	D8	D9	D10	D11	D12	D13	D14	D15	D16	D17	D18	D19	D20	D21	D22	D23	D24	D25	
S1	44	40	1	5	5	22	14	43	29	11	46	2	33	4	37	16	16	5	25	27	18	8	24	24	6	45
S2	11	43	20	9	23	25	37	5	31	21	19	17	22	3	14	4	24	18	1	32	12	40	36	8	6	35
S3	29	30	36	49	3	8	49	22	31	2	1	49	41	12	23	11	43	21	22	27	49	29	37	45	12	7
S4	41	42	28	30	16	29	12	25	45	35	23	48	44	32	26	35	39	23	26	17	10	5	50	27	21	13
S5	4	43	31	44	34	29	38	20	7	24	12	37	46	20	21	32	33	24	16	5	44	12	32	6	46	7
S6	12	44	17	20	37	9	40	14	32	45	5	17	50	40	49	6	15	38	31	3	36	13	47	26	15	5
S7	4	47	7	40	15	49	2	18	43	10	4	47	12	32	49	18	22	2	42	49	2	11	49	13	50	13
S8	25	11	32	40	20	16	28	1	28	32	15	24	37	24	12	22	37	12	42	36	10	40	8	31	18	35
S9	1	45	1	1	28	30	23	40	23	43	40	11	13	50	8	1	32	11	41	23	13	35	14	45	36	16
S10	48	9	5	22	48	37	30	4	28	35	18	3	42	43	44	47	27	7	30	20	50	44	13	33	37	5
S11	39	7	47	42	17	25	16	44	38	15	23	17	26	33	11	26	3	7	5	12	4	13	8	7	23	48
S12	13	32	14	39	14	28	39	13	1	21	35	24	8	23	26	41	49	33	2	25	42	8	47	49	28	19
S13	10	23	46	14	14	45	34	4	13	18	41	39	45	5	27	12	29	14	10	26	22	21	39	49	2	14
S14	19	33	50	44	39	42	42	16	25	18	8	38	49	11	19	47	48	40	12	20	25	20	5	24	1	37
S15	16	28	32	3	21	19	6	33	21	32	31	26	27	1	37	47	21	13	48	49	4	13	9	26	45	16
S16	45	1	19	7	33	7	6	12	25	42	11	4	9	43	11	13	1	31	7	4	37	26	31	9	36	6
S17	30	9	20	35	26	28	27	48	23	23	19	20	27	39	50	8	34	7	6	1	30	10	21	11	23	9
S18	5	36	18	43	5	10	1	16	7	21	36	38	31	41	49	16	29	4	16	1	4	46	11	38	8	46
S19	45	44	10	17	12	10	34	19	45	14	32	12	5	10	33	15	11	10	45	44	3	29	46	17	37	38
S20	5	44	7	41	11	38	37	26	34	29	28	6	50	38	14	20	11	40	30	38	42	36	9	35	37	24
S21	5	19	39	9	1	14	4	8	11	19	2	28	8	35	47	6	3	28	14	49	15	13	26	13	23	41
S22	22	2	46	10	28	9	8	47	18	14	47	45	38	30	46	20	48	44	44	18	10	1	47	25	31	27
S23	5	46	30	37	9	31	4	30	26	33	45	17	48	34	35	24	8	24	37	46	9	3	46	34	49	29
Permintaan	15	38	37	21	20	23	9	21	9	41	20	5	45	43	15	27	16	17	14	28	10	19	35	48	15	