DAFTAR PUSTAKA

Carlier, J. (1978). Ordonnancements a contraintes disjonctives. *RAIRO-Operations Research*, *12*(4), 333-350.

Desale, S., Rasool, A., Andhale, S., & Rane, P. (2015). Heuristic and meta-heuristic algorithms and their relevance to the real world: a survey. *Int. J. Comput. Eng. Res. Trends*, *351*(5), 2349-7084.

Engelbrecht, A. P. (2007). *Computational intelligence: an introduction*. John Wiley & Sons.

Hayat, I., Tariq, A., Shahzad, W., Masud, M., Ahmed, S., Ali, M. U., & Zafar, A. (2023). Hybridization of Particle Swarm Optimization with Variable Neighborhood Search and Simulated Annealing for Improved Handling of the Permutation Flow-Shop Scheduling Problem. *Systems*, *11*(5), 221.

Heller, J. (1960). Some numerical experiments for an M× J flow shop and its decision-theoretical aspects. *Operations Research*, *8*(2), 178-184.

Irman, A., Febianti, E., & Khasanah, U. (2019, December). Minimizing makespan on flow shop scheduling using Campbel Dudek and Smith, particle swarm optimization, and proposed heuristic algorithm. In *IOP Conference Series: Materials Science and Engineering* (Vol. 673, No. 1, p. 012099). IOP Publishing.

Baker, K. R., & Trietsch, D. (2019). *Principles of sequencing and scheduling second edition*. John Wiley & Sons.

Mashuri, C., Mujianto, A. H., Sucipto, H., Arsam, R. Y., & Permadi, G. S. (2019). Production Time Optimization using Campbell Dudek Smith (CDS) Algorithm for Production Scheduling. In *E3S Web of Conferences* (Vol. 125, p. 23009). EDP Sciences.

Pinedo, M. L. (2012). *Scheduling* (Vol. 29). New York: Springer.

Mishra, A., & Shrivastava, D. (2020). A discrete Jaya algorithm for permutation flow-shop scheduling problem. *International Journal of Industrial Engineering Computations*, *11*(3), 415-428.

Mladenović, N., & Hansen, P. (1997). Variable neighborhood search. *Computers & operations research*, *24*(11), 1097-1100.

Poli, R., Kennedy, J., & Blackwell, T. (2007). Particle swarm optimization: An overview. *Swarm intelligence*, *1*, 33-57.

Shi, Y., & Eberhart, R. (1998, May). A modified particle swarm optimizer. In *1998 IEEE international conference on evolutionary computation proceedings. IEEE world congress on computational intelligence (Cat. No. 98TH8360)* (pp. 69-73). IEEE.

Reeves, C. R. (1995). A genetic algorithm for flowshop sequencing. *Computers & operations research*, *22*(1), 5-13.

Sidabutar, S. N., Amin, M., & Putri, A. (2020). Penjadwalan Operasi Mesin Produksi Dengan Metode CDS (Campbell Dudek Smith) di PT Tjokro Bersaudara Balikpapanindo. *PROTON*, *11*(2), 53-61.

Tasgetiren, M. F., Sevkli, M., Liang, Y. C., & Gencyilmaz, G. (2004, September). Particle swarm optimization algorithm for permutation flowshop sequencing problem. In *International Workshop on Ant Colony Optimization and Swarm Intelligence* (pp. 382-389). Berlin, Heidelberg: Springer Berlin Heidelberg.

Tasgetiren, M. F., Liang, Y. C., Sevkli, M., & Gencyilmaz, G. (2007). A particle swarm optimization algorithm for makespan and total flowtime minimization in the permutation flowshop sequencing problem. *European journal of operational research*, *177*(3), 1930-1947.

Zaied, A. N. H., Ismail, M. M., & Mohamed, S. S. (2021). Permutation flow shop scheduling problem with makespan criterion: literature review. *J. Theor. Appl. Inf. Technol*, *99*(4), 830-848.

L A M P I R A N

Lampiran 1. Kode Matlab Algoritma Campbell Dudek Smith

```
clear all;
clc;

% Matriks flowshop (contoh)
flowshop_matrix = [
   375    12   142   245   412;
   632   452   758   278   398;
    12   876   124   534   765;
   460   542   523   120   499;
   528   101   789   124   999;
   796   245   632   375   123;
   532   230   543   896   452;
    14   124   214   543   785;
   257   527   753   210   463;
   896   896   214   258   259;
   532   302   501   765   988
    ];

% Menghitung jumlah pekerjaan (jobs) dan jumlah mesin (machines)
[num_jobs, num_machines] = size(flowshop_matrix);

% Inisialisasi urutan optimal
optimal_sequence = zeros(1, num_jobs);

% Jumlah tahap penyelesaian atau iterasi
num_stages = num_machines - 1;

% Inisialisasi cell array untuk menyimpan job sequence tiap
iterasi
job_sequences = cell(num_stages, 1);

% Inisialisasi variabel untuk menyimpan minimum total completion
time
min_total_completion_time = Inf;
min_total_completion_time_sequence = [];

for i = 1:num_stages
    a_j = zeros(num_jobs, 1);
    b_j = zeros(num_jobs, 1);

    for j = 1:num_jobs
        a_j(j) = sum(flowshop_matrix(j, 1:i));
        b_j(j) = sum(flowshop_matrix(j, num_machines - i +
1:num_machines));
    end
    combined_matrix = [a_j, b_j];
    minValues = zeros(size(combined_matrix, 1), 1);
    minRowIndices = zeros(size(combined_matrix, 1), 1);
    minColIndices = zeros(size(combined_matrix, 1), 1);

    for k = 1:size(combined_matrix, 1)
        [minValues(k), minColIndices(k)] = min(combined_matrix(k,
:));
        minRowIndices(k) = k;
    end
```

```
    combinedData = [minValues, minRowIndices, minColIndices]

    % Mengurutkan baris-baris dalam combinedData berdasarkan nilai
yang terdapat pada kolom ke-3
    sortedData = [];
    for k = 1:max(combinedData(:, 3))
        idx = combinedData(:, 3) == k;
        if k == 1
            sortedData = [sortedData; sortrows(combinedData(idx,
:), 1)];
        elseif k == 2
            sortedData = [sortedData; sortrows(combinedData(idx,
:), -1)];
        end
    end
    job_sequence = sortedData(:, 2)';
    job_sequences{i} = job_sequence;

    % Menampilkan sortedData terbaru
    disp(['Sorted Data (Iteration ', num2str(i), '):']);
    disp(sortedData);

    % Memulai algoritma Permutation Flowshop
    disp('Permutation Flow Shop Algorithm:');
    permuted_matrix = flowshop_matrix(job_sequence, :);

    % Menampilkan permutasi job untuk iterasi i
    disp(['Permuted Matrix (Iteration ', num2str(i), '):']);
    disp(permuted_matrix)

    % Menghitung waktu penyelesaian dari Permutation Flowshop
    total_completion_time = zeros(1, num_machines);
    for j = 1:num_jobs
        for k = 1:num_machines
            if j == 1 && k == 1
                total_completion_time(k) = permuted_matrix(j, k);
            elseif j == 1 && k > 1
                total_completion_time(k) = total_completion_time(k
- 1) + permuted_matrix(j, k);
            elseif k == 1 && j > 1
                total_completion_time(k) =
total_completion_time(k) + permuted_matrix(j, k);
            else
                total_completion_time(k) =
max(total_completion_time(k), total_completion_time(k - 1)) +
permuted_matrix(j, k);
            end
        end
    end
    disp(['Total Completion Time (Iteration ', num2str(i), '): ',
num2str(max(total_completion_time))]);

    % Menentukan jadwal dan makespan terbaik yang diperoleh
    if max(total_completion_time) < min_total_completion_time
        min_total_completion_time = max(total_completion_time);
        min_total_completion_time_sequence = job_sequence;
    end
end
```

```
% Menamplkan permutasi job dengan waktu penyelesaian total terbaik
disp(['Minimum Total Completion Time: ',
num2str(min_total_completion_time)]);
disp('Corresponding Job Sequence:');
disp(min_total_completion_time_sequence);
```

Lampiran 2. Kode Matlab Algoritma Particle Swarm Optimization

```
clear all;
clc;

% Masukkan matriks dari processing time
processing_time_matrix = [
   375    12   142   245   412;
   632   452   758   278   398;
    12   876   124   534   765;
   460   542   523   120   499;
   528   101   789   124   999;
   796   245   632   375   123;
   532   230   543   896   452;
    14   124   214   543   785;
   257   527   753   210   463;
   896   896   214   258   259;
   532   302   501   765   988
]; % Isi dengan matriks processing time yang diberikan

% Inisialisasi parameter PSO
C1 = 2; % Konstanta percepatan personal best
C2 = 2; % Konstanta percepatan global best
inertia_weight = 1.2;

% Mendapatkan ukuran matriks processing time
num_jobs = size(processing_time_matrix, 1); % Jumlah pekerjaan
num_machines = size(processing_time_matrix, 2); % Jumlah
mesinrandi(num_jobs);randi(num_jobs);
num_particles = 2 * num_jobs;
max_iterations = 25;

% Inisialisasi posisi Partikel
min_value = 0;
max_value = 4;
particles = (max_value - min_value) .* rand(num_particles,
num_jobs) + min_value; % Inisialisasi kecepatan dengan nol

% Inisialisasi kecepatan partikel
min_value = -4;
max_value = 4;
velocities = (max_value - min_value) .* rand(num_particles,
num_jobs) + min_value; % Inisialisasi kecepatan dengan nol

% Evaluasi fungsi tujuan awal (fitness)
makespan = zeros(num_particles, 1);
for i = 1:num_particles
    % Lakukan perankingan untuk setiap partikel
    [~, idx] = sort(particles(i, :), 'ascend');
    ranking = idx;

    % Lakukan permutasi terhadap matriks processing time
    permuted_matrix = processing_time_matrix(idx, :);

    % Hitung makespan atau evaluasi fitness menggunakan algoritma
yang sesuai
```

```
    % di sini, nilai makespan dihitung sebagai total waktu selesai
pada mesin terakhir
    machine_times = zeros(1, num_machines);
    for j = 1:num_jobs
        job = ranking(j);
        for k = 1:num_machines
            if j == 1 && k == 1
                machine_times(k) = permuted_matrix(j, k);
            elseif j == 1 && k > 1
                machine_times(k) = machine_times(k - 1) +
permuted_matrix(j, k);
            elseif k == 1 && j > 1
                machine_times(k) = machine_times(k) +
permuted_matrix(j, k);
            else
                machine_times(k) = max(machine_times(k),
machine_times(k - 1)) + permuted_matrix(j, k);
            end
        end
    end
    makespan(i) = max(machine_times);
    fitness(i) = makespan(i); % Untuk kasus ini, makespan
digunakan sebagai fitness
end

% Inisialisasi personal_best
personal_best = particles; % Menggunakan nilai awal yang sama
dengan nilai partikel

% Temukan global best
[global_best_fitness, global_best_idx] = min(fitness);
global_best = personal_best(global_best_idx, :);

% Iterasi PSO
for iter = 1:max_iterations

    % Pembaruan posisi dan kecepatan partikel
    for i = 1:num_particles
        r1 = rand();
        r2 = rand();

        % Update kecepatan
        velocities(i, :) = inertia_weight * velocities(i, :) + C1
* r1 * (personal_best(i, :) - particles(i, :)) + C2 * r2 *
(global_best - particles(i, :));

        % Batasan pada kecepatan
        velocities(i, :) = max(-4, min(4, velocities(i, :)));

        % Update posisi
        particles(i, :) = particles(i, :) + velocities(i, :);

        % Batasan pada posisi
        particles(i, :) = max(0, min(4, particles(i, :)));

        % Update beban inersia
        inertia_weight = inertia_weight * 0.975;
```

```
        % Batasan beban inersia
        inertia_weight = max(0.4, inertia_weight);
    end

    % Evaluasi fitness untuk setiap partikel
    for i = 1:num_particles
        % Lakukan perankingan untuk setiap partikel
        [~, idx] = sort(particles(i, :), 'ascend');
        ranking =idx;
        % Lakukan permutasi terhadap matriks processing time
        permuted_matrix = processing_time_matrix(idx, :);

        % Hitung makespan
        machine_times = zeros(1, num_machines);
        for j = 1:num_jobs
            job = ranking(j);
            for k = 1:num_machines
                if j == 1 && k == 1
                    machine_times(k) = permuted_matrix(j, k);
                elseif j == 1 && k > 1
                    machine_times(k) = machine_times(k - 1) +
permuted_matrix(j, k);
                elseif k == 1 && j > 1
                    machine_times(k) = machine_times(k) +
permuted_matrix(j, k);
                else
                    machine_times(k) = max(machine_times(k),
machine_times(k - 1)) + permuted_matrix(j, k);
                end
            end
        end
        makespan(i) = max(machine_times);

        % Pembaruan personal best
        if makespan(i) < fitness(i)
            personal_best(i, :) = particles(i, :);
            fitness(i) = makespan(i); % Update nilai fitness
        end
    end

    % Pembaruan global best
    [min_fitness, min_index] = min(fitness);
    if min_fitness < global_best_fitness
        global_best_fitness = min_fitness;
        global_best = personal_best(min_index, :);
    end

    currentMakespan = global_best;
    disp(['Iteration: ', num2str(iter), ', Global Best Makespan:
', num2str(min_fitness)]);

    % Hitung ranking untuk global best
    [~, idx] = sort(global_best, 'ascend');
    ranking = idx
```

```
    disp(['Iteration: ', num2str(iter), ', Optimal Solution
Ranking: ']);
    disp(ranking);

    % Implementasi Variable Neighborhood Search (VNS)
    S0 = ranking; % Solusi awal VNS

    % 1. a, b dipilih secara random untuk integer [1, n(banyaknya
job)] dengan a tidak boleh sama sengan b.
    a = randi(num_jobs)
    b = randi(num_jobs)
    while a == b
        b = randi(num_jobs)
    end

    % 2. Mendapatkan permutasi job S yang merupakan permutasi job
baru melalui operasi insert.
    S = S0;
    if a < b
        % Pindahkan job yang berada pada dimensi a dan sisipkan
pada dimensi b-1
        temp = S(a);
        for i=a+1:b-1
            S(i-1)=S(i);
        end
        S(b-1)=temp
    else
        % Pindahkan job yang berada pada dimensi a dan sisipkan
pada dimensi b
        temp = S(a);
        S(a) = [];
        S = [S(1:b) temp S(b+1:end)]
    end
     permuted_matrix = processing_time_matrix(S, :);
        % Hitung makespan hanya untuk urutan pekerjaan yang
diberikan oleh 'S'
        machine_times = zeros(1, num_machines);
        for j = 1:num_jobs
            job = S(j);
            for k = 1:num_machines
                if j == 1 && k == 1
                    machine_times(k) = permuted_matrix(job, k);
                elseif j == 1 && k > 1
                    machine_times(k) = machine_times(k - 1) +
permuted_matrix(job, k);
                elseif k == 1 && j > 1
                    machine_times(k) = machine_times(k) +
permuted_matrix(job, k);
                else
                    machine_times(k) = max(machine_times(k),
machine_times(k - 1)) + permuted_matrix(job, k);
                end
            end
        end
        makespan = max(machine_times);

        % Atur loop = 0, kcount = 0
        loop = 0;
```

```
        kcount = 0;
        maxmethod = 2; % Tetapkan dua jenis operasi

        % Kriteria berhenti
        max_iterations = num_jobs*(num_jobs - 1);

        % Iterasi VNS dengan for loop
        while loop < max_iterations
            loop = loop + 1;

            disp(['Loop ', num2str(loop), ': S = ', num2str(S)]);

            % Dapatkan dua jenis operasi dalam satu iterasi
            for kcount = 0:1
                % (a) Jika kcount = 0, dapatkan permutasi job S1
dengan menggunakan operasi insert.
                if kcount == 0
                    a = randi(num_jobs);
                    b = randi(num_jobs);
                    while a == b
                        b = randi(num_jobs);
                    end

                    fprintf('Loop %d (kcount = 0): a = %d, b =
%d\n', loop, a, b);

                    S1 = S;
                    if a < b
                        temp = S1(a);
                        for i=a+1:b-1;
                            S1(i-1)=S1(i);
                        end
                        S1(b-1)=temp
                    else
                        temp = S1(a);
                        S1(a) = [];
                        S1 = [S1(1:b) temp S1(b+1:end)]
                    end
                else
                    % (b) Jika kcount = 1, dapatkan permutasi job
S1 dengan menggunakan operasi interchange.
                    a = randi(num_jobs);
                    b = randi(num_jobs);
                    while a == b
                        b = randi(num_jobs);
                    end

                    fprintf('Loop %d (kcount = 1): a = %d, b =
%d\n', loop, a, b);

                    S1 = S;
                    temp = S1(a);
                    S1(a) = S1(b)
                    S1(b) = temp
                end
                permuted_matrix = processing_time_matrix(S1, :);
```

```
                % Hitung makespan hanya untuk urutan pekerjaan
yang diberikan oleh 'S'
                machine_times = zeros(1, num_machines);
                for j = 1:num_jobs
                    job = S1(j);
                    for k = 1:num_machines
                        if j == 1 && k == 1
                            machine_times(k) = permuted_matrix(j,
k);
                        elseif j == 1 && k > 1
                            machine_times(k) = machine_times(k -
1) + permuted_matrix(j, k);
                        elseif k == 1 && j > 1
                            machine_times(k) = machine_times(k) +
permuted_matrix(j, k);
                        else
                            machine_times(k) =
max(machine_times(k), machine_times(k - 1)) + permuted_matrix(j,
k);
                        end
                    end
                end
                makespan_S1 = max(machine_times);

                if makespan_S1 < makespan
                    S = S1;
                    makespan = makespan_S1;

                    % Reset kcount to 0 to perform insert
operation in the next iteration
                    kcount = 0;

                    % Perbarui global best
                    if makespan < global_best_fitness
                        global_best = S;
                        global_best_fitness = makespan;

                        disp(['Updated Global Best = ',
num2str(global_best), ', Makespan = ',
num2str(global_best_fitness)]);
                    end

                        % (a) Dapatkan permutasi job S1 dengan
menggunakan operasi insert.
                        a = randi(num_jobs);
                        b = randi(num_jobs);
                        while a == b
                            b = randi(num_jobs);
                        end

                        fprintf('Loop %d (kcount = 0): a = %d, b =
%d\n', loop, a, b);

                        S1 = S;
                        if a < b
                            temp = S1(a);
                            for i=a+1:b-1
```

```
                                S1(i-1)=S1(i);
                            end
                            S1(b-1)=temp
                        else
                            temp = S1(a);
                            S1(a) = [];
                            S1 = [S1(1:b) temp S1(b+1:end)]
                        end
                        permuted_matrix =
processing_time_matrix(S1, :);

                        % Hitung makespan hanya untuk urutan
pekerjaan yang diberikan oleh 'S1'
                        machine_times = zeros(1, num_machines);
                        for j = 1:num_jobs
                            job = S1(j);
                            for k = 1:num_machines
                                if j == 1 && k == 1
                                    machine_times(k) =
permuted_matrix(j, k);
                                elseif j == 1 && k > 1
                                    machine_times(k) =
machine_times(k - 1) + permuted_matrix(j, k);
                                elseif k == 1 && j > 1
                                    machine_times(k) =
machine_times(k) + permuted_matrix(j, k);
                                else
                                    machine_times(k) =
max(machine_times(k), machine_times(k - 1)) + permuted_matrix(j,
k);
                                end
                            end
                        end
                        makespan_S1 = max(machine_times);

                        if makespan_S1 < makespan
                            S = S1;
                            makespan = makespan_S1;

                            % Perbarui global best
                            if makespan < global_best_fitness
                                global_best = S;
                                global_best_fitness = makespan;

                                disp(['Updated Global Best = ',
num2str(global_best), ', Makespan = ',
num2str(global_best_fitness)]);
                            end
                        end
                    end

                end
            end
end

% Setelah iterasi PSO dan VNS selesai
disp('Optimal Solution:');
disp('Global Best Makespan:');
```

```
disp(global_best_fitness);

% Hitung ranking untuk global best
[~, idx] = sort(global_best, 'ascend');
ranking = idx;

disp('Optimal Solution Ranking:');
disp(ranking);
```

Lampiran 3. Dataset Benchmark Problem

Tabel L. 1 Matriks Processing Time Car01

$\boldsymbol{Jobs}$ $\boldsymbol{/Machines}$	$\boldsymbol{M_1}$	$\boldsymbol{M_2}$	$\boldsymbol{M_3}$	$\boldsymbol{M_4}$	$\boldsymbol{M_5}$
$\boldsymbol{J_1}$	375	12	142	245	412
$\boldsymbol{J_2}$	632	452	758	278	398
$\boldsymbol{J_3}$	12	876	124	534	765
$\boldsymbol{J_4}$	460	542	523	120	499
$\boldsymbol{J_5}$	528	101	789	124	999
$\boldsymbol{J_6}$	796	245	632	375	123
$\boldsymbol{J_7}$	532	230	543	896	452
$\boldsymbol{J_8}$	14	124	214	543	785
$\boldsymbol{J_9}$	257	527	753	210	463
$\boldsymbol{J_{10}}$	896	896	214	258	259
$\boldsymbol{J_{11}}$	532	302	501	765	988

Tabel L. 2 Matriks Processing Time Car08

$\boldsymbol{Jobs}$ $\boldsymbol{/Machines}$	$\boldsymbol{M_1}$	$\boldsymbol{M_2}$	$\boldsymbol{M_3}$	$\boldsymbol{M_4}$	$\boldsymbol{M_5}$	$\boldsymbol{M_6}$	$\boldsymbol{M_7}$	$\boldsymbol{M_8}$
$\boldsymbol{J_1}$	456	654	852	145	632	425	214	654
$\boldsymbol{J_2}$	789	123	369	678	581	396	123	789
$\boldsymbol{J_3}$	654	123	632	965	475	325	456	654
$\boldsymbol{J_4}$	321	456	581	421	32	147	789	123
$\boldsymbol{J_5}$	456	789	472	365	536	852	654	123
$\boldsymbol{J_6}$	789	654	586	824	325	12	321	456
$\boldsymbol{J_7}$	654	321	320	758	863	452	456	789
$\boldsymbol{J_8}$	789	147	120	639	21	863	789	654

Tabel L. 3 Matriks Processing Time Rec01

$\boldsymbol{Jobs}$ $\boldsymbol{/Machines}$	$\boldsymbol{M_1}$	$\boldsymbol{M_2}$	$\boldsymbol{M_3}$	$\boldsymbol{M_4}$	$\boldsymbol{M_5}$
$\boldsymbol{J_1}$	5	76	74	99	26
$\boldsymbol{J_2}$	74	21	83	52	90
$\boldsymbol{J_3}$	67	48	6	66	38
$\boldsymbol{J_4}$	97	36	71	68	81
$\boldsymbol{J_5}$	87	86	64	11	31
$\boldsymbol{J_6}$	1	42	20	90	23
$\boldsymbol{J_7}$	69	32	99	26	57
$\boldsymbol{J_8}$	69	12	54	80	16
$\boldsymbol{J_9}$	11	63	24	16	89
$\boldsymbol{J_{10}}$	87	52	43	10	26
$\boldsymbol{J_{11}}$	25	59	88	87	40
$\boldsymbol{J_{12}}$	50	42	72	77	29
$\boldsymbol{J_{13}}$	58	76	71	82	94
$\boldsymbol{J_{14}}$	79	48	20	63	97
$\boldsymbol{J_{15}}$	35	57	78	99	80
$\boldsymbol{J_{16}}$	70	76	53	2	19
$\boldsymbol{J_{17}}$	79	22	77	74	95
$\boldsymbol{J_{18}}$	34	99	49	3	61
$\boldsymbol{J_{19}}$	37	24	32	35	4
$\boldsymbol{J_{20}}$	50	88	46	63	76

Tabel L. 4 Matriks Processing Time Hel2

$\boldsymbol{Jobs}$ $\boldsymbol{/Machines}$	$\boldsymbol{M_1}$	$\boldsymbol{M_2}$	$\boldsymbol{M_3}$	$\boldsymbol{M_4}$	$\boldsymbol{M_5}$	$\boldsymbol{M_6}$	$\boldsymbol{M_7}$	$\boldsymbol{M_8}$	$\boldsymbol{M_9}$	$\boldsymbol{M_{10}}$
$\boldsymbol{J_1}$	1	1	1	4	3	5	5	7	6	4
$\boldsymbol{J_2}$	2	5	4	3	1	9	5	4	7	0
$\boldsymbol{J_3}$	5	6	8	4	4	2	5	6	7	5
$\boldsymbol{J_4}$	4	1	5	6	5	7	9	2	6	2

$Jobs$ /$Machines$	M_1	M_2	M_3	M_4	M_5	M_6	M_7	M_8	M_9	M_{10}
J_5	4	4	2	7	3	6	5	2	4	1
J_6	7	6	2	5	4	1	4	7	5	5
J_7	8	5	8	7	9	5	3	5	1	5
J_8	4	2	5	8	9	9	4	7	5	8
J_9	2	7	4	2	5	4	5	8	4	3
J_{10}	6	5	1	9	4	4	7	6	5	1
J_{11}	5	4	7	3	9	1	4	7	3	2
J_{12}	2	4	9	2	4	5	2	1	4	2
J_{13}	4	0	1	2	2	3	1	4	2	8
J_{14}	1	2	5	7	8	6	2	1	4	8
J_{15}	6	4	5	1	2	4	5	6	2	9
J_{16}	4	5	3	1	8	7	0	1	4	6
J_{17}	7	3	1	4	7	0	4	1	5	6
J_{18}	5	2	4	1	2	7	5	3	2	3
J_{19}	8	6	8	5	7	4	2	5	9	5
J_{20}	4	5	3	5	7	9	2	4	5	8

Tabel L. 5 Matriks Processing Time Rec19

$Jobs$ /$Machines$	M_1	M_2	M_3	M_4	M_5	M_6	M_7	M_8	M_9	M_{10}
J_1	40	16	50	59	100	78	38	76	9	68
J_2	39	3	35	70	65	80	40	49	52	50
J_3	31	56	88	71	83	69	48	98	88	96
J_4	65	77	58	66	86	93	69	49	85	51
J_5	11	5	75	12	56	64	20	6	83	49
J_6	48	56	41	55	3	94	11	87	78	48
J_7	47	60	16	67	61	36	36	62	13	74
J_8	67	100	3	85	70	19	58	87	61	51
J_9	32	47	40	47	66	85	99	50	19	45

$\boldsymbol{Jobs}$ $\boldsymbol{/Machines}$	$\boldsymbol{M_1}$	$\boldsymbol{M_2}$	$\boldsymbol{M_3}$	$\boldsymbol{M_4}$	$\boldsymbol{M_5}$	$\boldsymbol{M_6}$	$\boldsymbol{M_7}$	$\boldsymbol{M_8}$	$\boldsymbol{M_9}$	$\boldsymbol{M_{10}}$
$\boldsymbol{J_{10}}$	27	97	84	30	68	28	26	98	88	96
$\boldsymbol{J_{11}}$	81	12	1	88	63	32	38	82	68	61
$\boldsymbol{J_{12}}$	58	56	53	88	100	8	57	92	39	45
$\boldsymbol{J_{13}}$	38	30	81	51	70	28	10	93	53	45
$\boldsymbol{J_{14}}$	14	13	1	84	97	69	20	68	19	83
$\boldsymbol{J_{15}}$	70	98	83	22	27	44	93	46	91	45
$\boldsymbol{J_{16}}$	75	30	45	64	13	47	6	49	57	21
$\boldsymbol{J_{17}}$	69	41	37	12	3	81	92	25	24	36
$\boldsymbol{J_{18}}$	47	92	28	4	28	3	32	85	8	94
$\boldsymbol{J_{19}}$	52	7	97	56	90	60	37	42	19	15
$\boldsymbol{J_{20}}$	58	11	18	100	47	24	41	48	51	65
$\boldsymbol{J_{21}}$	61	69	45	17	4	31	83	32	68	5
$\boldsymbol{J_{22}}$	63	22	5	77	99	19	99	37	92	19
$\boldsymbol{J_{23}}$	63	25	83	78	89	66	8	57	89	56
$\boldsymbol{J_{24}}$	42	86	8	83	39	26	99	75	60	67
$\boldsymbol{J_{25}}$	51	100	42	53	10	66	19	2	24	41
$\boldsymbol{J_{26}}$	100	90	68	91	46	5	59	11	10	44
$\boldsymbol{J_{27}}$	42	41	76	76	61	52	44	78	40	57
$\boldsymbol{J_{28}}$	83	24	14	100	26	41	19	18	21	12
$\boldsymbol{J_{29}}$	5	84	57	6	60	91	18	83	44	87
$\boldsymbol{J_{30}}$	69	35	72	62	90	8	44	67	4	77

Tabel L. 6 Matriks Processing Time Rec25

$\boldsymbol{Jobs/Machines}$	$\boldsymbol{M_1}$	$\boldsymbol{M_2}$	$\boldsymbol{M_3}$	$\boldsymbol{M_4}$	$\boldsymbol{M_5}$	$\boldsymbol{M_6}$	$\boldsymbol{M_7}$	$\boldsymbol{M_8}$	$\boldsymbol{M_9}$	$\boldsymbol{M_{10}}$	$\boldsymbol{M_{11}}$	$\boldsymbol{M_{12}}$	$\boldsymbol{M_{13}}$	$\boldsymbol{M_{14}}$	$\boldsymbol{M_{15}}$
$\boldsymbol{J_1}$	68	91	17	68	46	86	24	43	58	86	40	27	38	82	86
$\boldsymbol{J_2}$	34	21	4	48	67	24	63	68	96	45	91	97	96	3	46
$\boldsymbol{J_3}$	57	69	12	2	51	68	34	8	17	55	80	61	51	32	36
$\boldsymbol{J_4}$	22	24	24	3	76	65	94	69	73	33	86	36	48	42	85
$\boldsymbol{J_5}$	28	6	47	71	81	93	94	21	32	23	73	48	35	67	59
$\boldsymbol{J_6}$	79	80	27	21	56	36	24	94	53	50	55	7	78	14	53
$\boldsymbol{J_7}$	98	55	71	80	23	45	44	22	40	93	38	4	96	42	53
$\boldsymbol{J_8}$	69	84	63	15	27	66	73	98	64	38	3	69	46	27	34
$\boldsymbol{J_9}$	65	21	79	50	67	68	56	53	71	29	63	36	62	77	35
$\boldsymbol{J_{10}}$	9	92	84	88	48	71	71	90	24	54	77	96	66	49	29
$\boldsymbol{J_{11}}$	92	22	73	4	38	31	55	49	66	83	75	82	87	82	89
$\boldsymbol{J_{12}}$	27	67	89	10	35	67	88	43	51	22	23	60	54	22	76
$\boldsymbol{J_{13}}$	10	65	77	85	5	25	14	78	32	23	21	11	65	60	23
$\boldsymbol{J_{14}}$	56	74	66	61	27	41	100	26	92	79	100	39	11	59	97
$\boldsymbol{J_{15}}$	93	96	79	50	35	28	100	84	78	81	65	69	17	96	19

$\boldsymbol{Jobs}$ $\boldsymbol{/Machines}$	$\boldsymbol{M_1}$	$\boldsymbol{M_2}$	$\boldsymbol{M_3}$	$\boldsymbol{M_4}$	$\boldsymbol{M_5}$	$\boldsymbol{M_6}$	$\boldsymbol{M_7}$	$\boldsymbol{M_8}$	$\boldsymbol{M_9}$	$\boldsymbol{M_{10}}$	$\boldsymbol{M_{11}}$	$\boldsymbol{M_{12}}$	$\boldsymbol{M_{13}}$	$\boldsymbol{M_{14}}$	$\boldsymbol{M_{15}}$
$\boldsymbol{J_{16}}$	2	99	100	13	15	35	3	58	39	56	57	48	82	86	53
$\boldsymbol{J_{17}}$	2	72	49	44	84	48	90	48	27	45	49	26	36	20	33
$\boldsymbol{J_{18}}$	86	54	79	52	2	67	69	78	38	92	13	25	40	37	80
$\boldsymbol{J_{19}}$	51	99	68	2	66	44	98	83	50	53	13	57	39	50	92
$\boldsymbol{J_{20}}$	38	25	18	97	35	7	45	98	81	18	60	73	86	34	3
$\boldsymbol{J_{21}}$	4	29	84	11	61	47	17	2	68	85	93	64	98	34	62
$\boldsymbol{J_{22}}$	23	95	66	57	91	15	90	84	25	88	65	24	80	98	76
$\boldsymbol{J_{23}}$	81	43	11	95	78	1	87	11	26	80	29	100	28	40	37
$\boldsymbol{J_{24}}$	54	33	39	23	44	32	16	96	29	87	34	25	80	14	83
$\boldsymbol{J_{25}}$	56	58	10	92	95	95	73	83	57	83	24	54	48	81	20
$\boldsymbol{J_{26}}$	81	24	37	21	97	60	25	21	53	34	57	12	34	28	87
$\boldsymbol{J_{27}}$	55	33	74	43	66	65	32	96	29	7	33	78	30	36	45
$\boldsymbol{J_{28}}$	2	67	56	82	49	74	97	11	75	76	65	41	76	9	80
$\boldsymbol{J_{29}}$	6	88	52	32	54	20	10	7	64	14	35	81	92	22	81
$\boldsymbol{J_{30}}$	88	65	62	51	52	89	23	55	63	79	63	94	79	39	50

Tabel L. 7 Matriks Processing Time Rec31

$Jobs$ /$Machines$	M_1	M_2	M_3	M_4	M_5	M_6	M_7	M_8	M_9	M_{10}
J_1	59	47	20	43	49	74	38	46	18	12
J_2	30	1	90	97	5	70	59	63	15	93
J_3	22	58	68	3	33	48	27	12	65	21
J_4	70	81	2	32	72	57	32	25	13	87
J_5	38	17	48	53	57	17	25	50	72	72
J_6	51	15	72	8	34	90	40	44	47	77
J_7	63	84	75	75	71	13	10	97	81	31
J_8	48	62	71	70	6	94	10	71	29	99
J_9	92	29	91	99	54	64	89	89	38	87
J_{10}	91	21	56	49	43	20	27	68	99	73
J_{11}	62	6	3	89	48	97	79	21	96	77
J_{12}	67	83	70	49	50	50	60	28	15	50
J_{13}	73	18	55	49	66	56	90	29	87	4
J_{14}	27	94	71	33	31	68	45	52	95	40
J_{15}	48	28	46	73	89	35	98	97	67	9
J_{16}	7	51	48	4	29	62	37	15	10	66
J_{17}	55	46	65	48	61	36	69	14	78	100
J_{18}	4	4	31	49	28	78	73	26	29	26
J_{19}	19	97	37	30	37	16	15	89	11	16
J_{20}	30	95	86	22	17	16	61	79	24	9
J_{21}	71	39	93	87	38	7	24	1	91	34
J_{22}	83	40	37	25	68	47	81	62	96	19
J_{23}	45	33	12	63	32	40	60	54	66	92
J_{24}	40	67	83	11	62	69	46	93	80	50
J_{25}	100	6	82	78	5	43	18	73	86	62
J_{26}	97	75	81	22	38	2	53	44	73	74

$Jobs$ $/Machines$	M_1	M_2	M_3	M_4	M_5	M_6	M_7	M_8	M_9	M_{10}
J_{27}	74	89	14	33	11	43	70	58	47	8
J_{28}	73	46	62	27	63	34	58	91	11	80
J_{29}	14	24	27	62	72	85	98	99	25	7
J_{30}	100	29	8	55	88	96	23	98	19	79
J_{31}	53	11	74	66	94	66	98	87	5	85
J_{32}	68	77	88	47	51	73	16	17	87	96
J_{33}	69	40	46	62	23	31	45	21	15	40
J_{34}	34	41	10	17	25	33	17	28	45	68
J_{35}	7	26	79	76	35	92	77	15	27	69
J_{36}	47	31	83	28	92	83	96	18	84	45
J_{37}	54	96	8	28	94	50	20	28	99	65
J_{38}	9	13	81	1	94	82	29	82	27	45
J_{39}	64	22	51	33	9	25	22	64	78	88
J_{40}	38	25	16	24	62	4	39	77	36	60
J_{41}	72	6	40	56	23	39	38	5	75	44
J_{42}	26	33	37	84	61	86	22	94	93	17
J_{43}	88	39	63	43	98	27	32	20	25	25
J_{44}	73	70	57	5	100	31	34	11	98	76
J_{45}	77	4	85	50	9	45	35	3	41	80
J_{46}	20	36	9	89	4	32	76	20	84	6
J_{47}	99	64	7	68	67	85	60	23	55	52
J_{48}	13	7	80	57	22	78	75	17	70	55
J_{49}	40	87	34	96	27	78	53	40	72	91
J_{50}	77	8	14	76	19	82	86	21	10	51

Tabel L. 8 Matriks Processing Time Hel2

$\boldsymbol{Jobs}$ $\boldsymbol{/Machines}$	$\boldsymbol{M_1}$	$\boldsymbol{M_2}$	$\boldsymbol{M_3}$	$\boldsymbol{M_4}$	$\boldsymbol{M_5}$	$\boldsymbol{M_6}$	$\boldsymbol{M_7}$	$\boldsymbol{M_8}$	$\boldsymbol{M_9}$	$\boldsymbol{M_{10}}$
$\boldsymbol{J_1}$	1	1	1	4	3	5	5	7	6	4
$\boldsymbol{J_2}$	2	5	4	3	1	9	5	4	7	0
$\boldsymbol{J_3}$	5	6	8	4	4	2	5	6	7	5
$\boldsymbol{J_4}$	4	1	5	6	5	7	9	2	6	2
$\boldsymbol{J_5}$	4	4	2	7	3	6	5	2	4	1
$\boldsymbol{J_6}$	7	6	2	5	4	1	4	7	5	5
$\boldsymbol{J_7}$	8	5	8	7	9	5	3	5	1	5
$\boldsymbol{J_8}$	4	2	5	8	9	9	4	7	5	8
$\boldsymbol{J_9}$	2	7	4	2	5	4	5	8	4	3
$\boldsymbol{J_{10}}$	6	5	1	9	4	4	7	6	5	1
$\boldsymbol{J_{11}}$	5	4	7	3	9	1	4	7	3	2
$\boldsymbol{J_{12}}$	2	4	9	2	4	5	2	1	4	2
$\boldsymbol{J_{13}}$	4	0	1	2	2	3	1	4	2	8
$\boldsymbol{J_{14}}$	1	2	5	7	8	6	2	1	4	8
$\boldsymbol{J_{15}}$	6	4	5	1	2	4	5	6	2	9
$\boldsymbol{J_{16}}$	4	5	3	1	8	7	0	1	4	6
$\boldsymbol{J_{17}}$	7	3	1	4	7	0	4	1	5	6
$\boldsymbol{J_{18}}$	5	2	4	1	2	7	5	3	2	3
$\boldsymbol{J_{19}}$	8	6	8	5	7	4	2	5	9	5
$\boldsymbol{J_{20}}$	4	5	3	5	7	9	2	4	5	8
$\boldsymbol{J_{21}}$	3	5	7	9	6	2	4	4	7	3
$\boldsymbol{J_{22}}$	0	2	4	5	4	7	4	5	4	8
$\boldsymbol{J_{23}}$	4	2	5	7	4	5	3	2	8	5
$\boldsymbol{J_{24}}$	7	8	2	1	9	6	7	8	4	1
$\boldsymbol{J_{25}}$	4	8	5	2	6	8	9	5	8	5
$\boldsymbol{J_{26}}$	4	5	7	2	3	7	3	6	5	4
$\boldsymbol{J_{27}}$	4	2	1	5	1	3	5	6	5	5
$\boldsymbol{J_{28}}$	5	8	5	7	8	2	5	8	3	5

$Jobs$ /$Machines$	M_1	M_2	M_3	M_4	M_5	M_6	M_7	M_8	M_9	M_{10}
J_{29}	5	4	5	4	5	7	6	2	5	9
J_{30}	8	2	1	5	5	6	5	8	7	5
J_{31}	8	3	5	9	5	4	5	2	4	2
J_{32}	8	5	2	5	7	6	2	8	9	5
J_{33}	3	7	4	6	8	2	4	5	2	3
J_{34}	5	5	4	7	9	8	2	5	2	5
J_{35}	5	2	5	2	5	4	8	2	1	3
J_{36}	5	5	9	5	4	9	8	5	3	5
J_{37}	2	1	2	1	4	3	3	5	2	6
J_{38}	8	8	4	7	2	6	8	6	3	5
J_{39}	9	7	5	8	5	6	5	8	9	4
J_{40}	5	6	9	6	5	3	1	8	7	4
J_{41}	6	4	7	4	3	6	1	4	5	8
J_{42}	4	3	7	5	1	9	2	4	2	5
J_{43}	4	2	8	7	3	4	9	8	7	4
J_{44}	2	5	9	4	2	5	3	0	4	7
J_{45}	9	5	4	2	3	7	0	2	1	6
J_{46}	2	3	2	5	1	0	8	9	5	3
J_{47}	5	2	7	9	4	3	6	2	5	0
J_{48}	7	8	2	1	4	7	5	8	9	4
J_{49}	1	4	2	3	6	8	2	4	7	5
J_{50}	2	5	4	5	6	8	4	1	7	5
J_{51}	8	3	0	2	5	6	8	2	9	4
J_{52}	7	2	4	3	6	2	9	4	1	8
J_{53}	3	5	7	5	3	8	6	4	8	1
J_{54}	5	0	5	6	0	0	2	4	7	8
J_{55}	1	9	5	2	4	7	5	0	2	5
J_{56}	0	2	9	6	1	4	0	0	5	2
J_{57}	0	2	5	8	3	6	9	1	2	4

$Jobs$ /$Machines$	M_1	M_2	M_3	M_4	M_5	M_6	M_7	M_8	M_9	M_{10}
J_{58}	7	9	6	3	5	1	7	5	4	5
J_{59}	4	3	5	2	1	4	9	7	4	1
J_{60}	0	3	5	2	4	9	4	7	5	4
J_{61}	7	8	5	6	3	9	8	7	4	6
J_{62}	1	9	6	7	0	2	4	8	3	6
J_{63}	6	1	2	0	3	5	4	1	7	3
J_{64}	6	5	1	4	9	7	3	5	6	4
J_{65}	1	8	2	6	9	4	7	5	8	4
J_{66}	0	1	6	2	9	4	8	5	7	6
J_{67}	4	2	5	6	8	5	6	4	1	4
J_{68}	3	4	5	8	4	1	2	3	6	8
J_{69}	9	8	2	3	1	4	0	2	4	5
J_{70}	4	3	2	5	6	4	1	8	9	2
J_{71}	5	7	1	2	6	8	2	3	4	7
J_{72}	2	1	4	3	8	4	6	2	4	5
J_{73}	6	7	9	2	4	3	2	5	6	7
J_{74}	2	4	0	2	5	3	4	7	8	6
J_{75}	2	7	5	4	3	1	5	6	0	2
J_{76}	3	5	7	0	2	4	5	2	5	7
J_{77}	2	4	5	3	4	7	8	3	2	4
J_{78}	9	9	1	4	5	7	6	5	3	2
J_{79}	8	2	5	2	2	5	1	5	7	8
J_{80}	4	5	2	4	7	9	5	4	2	4
J_{81}	5	5	1	2	4	2	3	8	5	1
J_{82}	0	2	9	5	4	2	5	9	6	5
J_{83}	1	2	3	5	6	2	4	0	2	5
J_{84}	4	2	3	5	4	2	3	5	4	2
J_{85}	4	4	5	8	9	8	5	2	8	3
J_{86}	4	2	3	3	2	5	8	8	1	2

$Jobs$ $/Machines$	M_1	M_2	M_3	M_4	M_5	M_6	M_7	M_8	M_9	M_{10}
J_{87}	5	3	6	2	5	6	4	7	9	3
J_{88}	4	2	3	6	8	5	3	4	7	2
J_{89}	5	8	8	3	5	6	5	6	5	2
J_{90}	5	6	4	2	5	4	6	5	8	4
J_{91}	2	1	4	7	4	5	9	8	5	6
J_{92}	2	1	4	6	5	8	6	1	3	5
J_{93}	9	5	1	3	5	7	9	1	2	5
J_{94}	3	5	4	9	7	2	6	5	2	1
J_{95}	2	5	0	3	2	4	7	8	9	5
J_{96}	5	3	5	7	9	2	4	5	5	8
J_{97}	6	3	1	5	0	1	4	8	9	8
J_{98}	3	0	4	3	7	2	6	9	4	1
J_{99}	1	7	4	2	2	4	5	0	6	9
J_{100}	1	7	8	4	6	5	4	8	5	2