BAB III
METODE PENELITIAN

3.1 Data Penelitian

Dalam penelitian ini, penghitungan alokasi ekonomis dilakukan berdasarkan data sekunder yang diperoleh dari literatur ilmiah dan studi yang serupa. Informasi yang digunakan berasal dari sistem di wilayah Mahakam, Kalimantan Timur, pada tanggal 9 Desember 2007. Data tersebut kemudian diproses menggunakan metode *Constriction Factor Particle Swarm Optimization* (CFPSO) dan dijalankan melalui perangkat lunak MATLAB2019b.[5]

3.1.1 Sistem Kelistrikan Mahakam

Di Mahakam, terdapat 23 unit pembangkit listrik. Penelitian ini memfokuskan pada pembangkit thermal yang dioperasikan oleh PT. PLN Wilayah Mahakam, yang tercantum dalam tabel 3.1.

Tabel 3.1 Data Output Pembangkit Yang Beroperasi, Range Output Beserta Biaya Operasinya

No	Nama Pembangkit	Persamaan Biaya Bahan Bakar
1	PLTD Gunung Malang Unit 1	$F_1 = 90{,}1715\ P_1^2 + 8467{,}45P_1 + 4895325{,}75$
2	PLTD Gunung Malang Unit 2	$F_2 = 92{,}414\ P_2^2 + 11615{,}575P_2 + 5218198{,}6$
3	PLTD Gunung Malang Unit 3	$F_3 = 16{,}51745\ P_3^2 + 161{,}667P_3 + 4796112{,}375$
4	PLTD Gunung Malang Unit 4	$F_4 = 149{,}29875\ P_4^2 + 16974{,}575P_4 + 5369538{,}025$
5	PLTD Gunung Malang Unit 5	$F_5 = 179{,}078\ P_5^2 + 3473P_5 + 4079930{,}325$
6	PLTD Gunung Malang Unit 6	$F_6 = 165{,}54825\ P_6^2 + 4382{,}0175P_6 + 4797215{,}225$
7	PLTD Batakan Unit 1	$F_7 = 69{,}69\ P_7^2 + 2054{,}0725P_7 + 5919384{,}075$

Tabel 3.1 Data Output Pembangkit Yang Beroperasi, Range Output Beserta Biaya Operasinya (Lanjutan)

NO	Nama Pembangkit	Persamaan Biaya Bahan Bakar
8	PLTD Batakan Unit 2	$F_8 = 241{,}454\ P_8^2 + 28350{,}375\ P_8 + 6960981{,}05$
9	PLTD Keledang Unit 1	$F_9 = 23{,}857325\ P_9^2 + 596{,}16\ P_9 + 6272844{,}05$
10	PLTD Keledang Unit 2	$F_{10} = 220{,}525\ P_{10}^2 + 19107{,}825\ P_{10} + 5386037{,}075$
11	PLTD Keledang Unit 3	$F_{11} = 236{,}6125\ P_{11}^2 + 32649{,}075 P_{11} + 6129707{,}575$
12	PLTD Keledang Unit 4	$F_{12} = 759{,}1725\ P_{12}^2 + 3174{,}5175 P_{12} + 4172226{,}45$
13	PLTD Keledang Unit 5	$F_{13} = 157{,}29125\ P_{13}^2 + 44867{,}25 P_{13} + 118808111{,}7$
14	PLTGU Tanjung Batu Unit 1	$F_{14} = 12{,}20685\ P_{14}^2 + 12839{,}675 P_{14} + 4134429{,}325$
15	PLTGU Tanjung Batu Unit 2	$F_{15} = 13{,}784025\ P_{15}^2 + 12795{,}05 P_{15} + 3710490{,}55$
16	PLTGU Tanjung Batu Unit 3	$F_{16} = 18{,}11265\ P_{16}^2 + 13192{,}425 P_{16} + 2975259{,}25$
17	PLTD Karang Asam Unit 1	$F_{17} = 6{,}4354\ P_{17}^2 + 83{,}67975 P_{17} + 5032347{,}1$
18	PLTD Karang Asam Unit 2	$F_{18} = 67{,}45325\ P_{18}^2 + 2155{,}905 P_{18} + 5055899{,}1$
19	PLTD Karang Asam Unit 3	$F_{19} = 20{,}053125\ P_{19}^2 + 1777{,}67 P_{19} + 5076421{,}425$
20	PLTD Karang Asam Unit 4	$F_{20} = 89{,}8265\ P_{20}^2 + 3512{,}56 P_{20} + 5089456{,}675$
21	PLTD Karang Asam Unit 5	$F_{21} = 138{,}4485\ P_{21}^2 + 6511{,}3 P_{21} + 4973906{,}4$
22	PLTD Karang Asam Unit 6	$F_{22} = 139{,}955\ P_{22}^2 + 44210{,}6 P_{22} + 11884626{,}13$
23	PLTD Karang Asam Unit 7	$F_{23} = 54{,}744025\ P_{23}^2 + 7118{,}5 P_{23} + 8093914{,}475$

3.1.2 Karakteristik Biaya Pembangkit

Karakteristik persamaan biaya pembangkit diperoleh dengan melakukan perkalian antara karakteristiik *input output* dengan bahan bakar pembangkit. Karakteristik biaya pembangkit ditunjukkan oleh table 3.2.

Tabel 3.2 Data Persamaan Biaya Bahan Bakar Pembangkit Sistem Mahakam

No	Nama Pembangkit	Daya Min (MW)	Daya Maks (MW)	Daya Pembangkit (MW)	Biaya Operasi (Rupiah/Jam)
1	PLTD Gunung Malang Unit 1	2	3,2	3,0	4.921.539,6435
2	PLTD Gunung Malang Unit 2	2	3,2	3,0	5.253.877,051
3	PLTD Gunung Malang Unit 3	2	3,3	3,0	4.796.746,033
4	PLTD Gunung Malang Unit 4	2	3,3	3,0	5.421.805,4388
5	PLTD Gunung Malang Unit 5	2	3,2	3,0	4.091.961,027
6	PLTD Gunung Malang Unit 6	2	3,2	3,0	4.811.851,2117
7	PLTD Batakan Unit 1	2	3,8	3,5	5.927.427,0313
8	PLTD Batakan Unit 2	2	3,6	3,5	7.063.165,174
9	PLTD Keledang Unit 1	2	4	3,5	6.275.222,8622
10	PLTD Keledang Unit 2	2	3,2	3,1	5.447.390,5778
11	PLTD Keledang Unit 3	2	3,6	3,4	6.243.449,6705
12	PLTD Keledang Unit 4	2	3,2	3,0	4.188.582,555
13	PLTD Keledang Unit 5	2	6	5,7	119.068.965,4177
14	PLTGU Tanjung Batu Unit 1	10	21	20,4	4.401.438,6977
15	PLTGU Tanjung Batu Unit 2	10	20	18,4	3.950.586,1895
16	PLTGU Tanjung Batu Unit 3	8	16	15,4	3.182.718,1911
17	PLTD Karang Asam Unit 1	2	3,3	3,2	5.032.680,7737
18	PLTD Karang Asam Unit 2	2	3,4	3,2	5.063.488,7173
19	PLTD Karang Asam Unit 3	2	3,3	3,2	5.082.315,313

Tabel 3.2 Data Persamaan Biaya Bahan Bakar Pembangkit Sistem Mahakam (lanjutan)

No	Nama Pembangkit	Daya Min (MW)	Daya Maks (MW)	Daya Pembangkit (MW)	Biaya Operasi (Rupiah/Jam)
20	PLTD Karang Asam Unit 4	2	3,3	3,2	5.101.616,6904
21	PLTD Karang Asam Unit 5	2	3,3	3,2	4.996.160,2726
22	PLTD Karang Asam Unit 6	3	6	5,8	12.145.755,6962
23	PLTD Karang Asam Unit 7	3	6	5,4	8.133.950,7108
Total		**70**	**132,4**	**125,1**	**240.602.694,9458**

3.2 Perhitungan Biaya Operasional Pembangkit

Perhitungan biaya operasional pembangkit listrik dapat bervariasi tergantung pada jenis pembangkitan, skala pembangkitan, teknologi yang digunakan, lokasi geografis, biaya bahan bakar, biaya tenaga kerja, dan faktor-faktor lainnya. Biaya pembangkitan tenaga listrik, biasanya dinyatakan dalam satuan mata uang kilowatt-hour listrik yang dihasilkan (Rp/kWh) yang menyatakan total biaya pembangkitan tenaga listrik. Biaya pembangkita tenaga listrik terdiri dari tiga komponen biaya utama, yaitu biaya modal, biaya bahan bakar dan biaya operasi dan perawatan.

3.2.1 Penjadwalan Unit Pembangkit

Penjadwalan unit pembangkit listrik melibatkan pengaturan jumlah daya yang dihasilkan oleh generator-generator pembangkit listrik untuk memenuhi kebutuhan beban listrik secara efisien dari segi ekonomi. Ini dilaksanakan dengan membagi beban listrik di antara unit-unit pembangkit secara efisien sehingga biaya pengoperasian unit pembangkit listrik dapat diminimalkan.

3.2.2 Perhitungan Menggunakan Metode CFPSO

Penelitian ini menggunakan metode *Contriction Factor Particle Swarm Optimization* (CFPSO). Gambar 2.2 menunjukkan diagram alir dari tahap perencanaan pencarian solusi Economic Dispatch dengan menggunakan kombinasi faktor konstriksi pada algoritma *Particle Swarm Optimization*. Berikut adalah langkah-langkah dari metode CFPSO:

a. Menginputkan data beban, data pembangkit, dan data transmisi ke dalam perangkat lunak MATLAB pada pembangkit listrik termal. Adapun datanya merupakan data pada tabel 3.1 dan 3.2.
b. Menginput parameter-parameter CFPSO ke dalam program MATLAB seperti:
    - Jumlah partikel (*swarm size*) ialah jumlah partikel atau agen yang digunakan dalam algoritma PSO. Semakin besar jumlah partikel, semakin banyak agen yang berpartisipasi dalam pencarian solusi, yang dapat meningkat kemungkinan menemukan solusi yang lebih baik, tetapi juga meningkat komplesitas komputasional. Jumlah partikel yang digunakan dalam penelitian ini 200.
    - Dimensi ruang pencarian ialah jumlah dimensi dalam ruang pencarian solusi. Dalam konteks optimasi, ini bisa menjadi jumlah variabel yang dioptimalkan. misalnya, jika kita memiliki fungsi tujuan dengan 3 variabel, maka dimensi ruang pencarian adalah 3. Adapun data fungsi tujuan pada tabe 3.1.
    - Batas iterasi ialah jumlah iterasi atau langkah pencarian yang akan dilakukan oleh algoritma PSO sebelum sebelum mencapai kondisi

berhenti. Biasanya merupakan kriteria berhenti untuk membatasi jumlah iterasi agar algoritma tidak berjalan terlalu lama. Jumlah iterasi yang digunakan dalam penelitian ini 500

- konstanta akselerasi kognitif (c1) dan konstanta akselerasi sosial (c2) ialah mengontrol seberapa besar partikel akan dipengaruhi oleh pengalaman pribadi (kognitif) dan pengalaman kolegtif dari seluruh populasi (sosial). Nilai-nilai ini menetukan seberapa cepat partikel akan bergerak menuju solusi yang diharapkan. Nilai konstanta yang digunakan dalam penelitian ini 4,1.
- Faktor kontriksi ialah faktor yang mengendalikan seberapa cepat nilai koefisien inersia berkurang selama iterasi. Hal ini membantu dalam mengurangi kecepata partikel seiring berjalannya waktu untuk mendekati solusi optimal.

c. Inisialisasi posisi awal partikel dilakukan secara acak dengan memperhatikan batasan dari setiap unit pembangkit. Dalam konteks PSO, partikel-partikel tersebut mewakili solusi potensial dalam ruang pencarian. Setiap partikel memiliki posisi dan kecepatan yang berkontribusi pada eksplorasi dan eksploitasi ruang pencarian untuk mencari solusi yang optimal. Proses inisialisasi dimulai dengan menentukan posisi awal setiap partikel, ini dilakukan secara acak, yang berarti posisi awal setiap partikel di atur secara acak dalam rentang yang diperbolehkan oleh batasan-batasan masalah. Dalam kasus pembangkit listrik, batasan-batasan tersebut mencakup kapasitas maksimum, ketersediaan bahan bakar, atau parameter teknis lainnya yang terkait dengan setiap unit pembangkit. Jadi, sementara posisi awal partikel dipilih secara acak, batasan-batasan dari setiap unit pembangkit harus dipertimbangkan. Ini berarti bahwa nilai-nilai posisi awal haruslah sesuai dengan batasan-batasan yang telah ditetapkan untuk memastikan bahwa solusi-solusi yang dihasilkan oleh PSO mematuhi kebutuhan teknis dan fungsional dari setiap unit pembangkit listrik. Dengan memperhatikan batasan-batasan ini dalam inisialisasi, PSO dapat memulai pencarian dari berbagai titik dalam

ruang pencarian yang mungkin, meningkatkan kemungkinan menemukan solusi yang optimal atau mendekati optimal untuk masalah optimasi yang diberikan.

Berikut adalah contoh sederhana dari implementasi inisialisasi posisi awal partikel dalam algoritma PSO dengan memperhatikan batasan dari setiap unit pembangkit menggunakan bahasa pemrograman MATLAB:

```
function positions = initialize_positions(swarm_size, dimension, lower_bound, upper_bound
    % swarm_size: Jumlah partikel
    % dimension: Dimensi ruang pencarian
    % lower_bound: Batas bawah dari setiap variabel pencarian
    % upper_bound: Batas atas dari setiap variabel pencarian

    % Inisialisasi posisi awal secara acak dalam batasan yang ditentukan
    positions = rand(swarm_size, dimension) .* (upper_bound - lower_bound) + lower_bound;
end

% Contoh pemanggilan fungsi untuk inisialisasi posisi
swarm_size = 50; % Jumlah partikel
dimension = 3; % Dimensi ruang pencarian
lower_bound = [0, 0, 0]; % Batas bawah dari setiap variabel pencarian
upper_bound = [100, 100, 100]; % Batas atas dari setiap variabel pencarian

positions = initialize_positions(swarm_size, dimension, lower_bound, upper_bound);
disp('Posisi awal partikel:');
disp(positions);
```

Gambar 3.1. implementasi inisialisasi posisi awal partikel dalam algoritma PSO

d. Menentukan kecepatan awal individu secara acak.

Dalam PSO, setiap partikel memiliki dua komponen utama: posisi dan kecepatan. Kecepatan partikel menentukan seberapa cepat dan ke arah mana partikel akan bergerak dalam ruang pencarian saat mencari solusi optimal. Proses menentukan kecepatan awal individu secara acak dilakukan dengan memperhatikan batasan-batasan yang mungkin ada dalam masalah yang dihadapi. Ini dilakukan dengan tujuan untuk menjaga partikel dalam rentang pencarian yang sesuai dengan kebutuhan masalah dan untuk mencegah partikel keluar dari ruang solusi yang valid. Dalam konteks inisialisasi kecepatan awal, biasanya batasan kecepatan maksimum dan minimum juga diperhitungkan, tergantung pada sifat masalah optimasi. Setelah kecepatan awal individu diatur secara acak, partikel siap untuk dimasukkan ke dalam proses iteratif PSO, di mana mereka akan menyesuaikan kecepatan dan posisi mereka berdasarkan

pengalaman pribadi dan sosial. Dengan demikian, proses menentukan kecepatan awal individu secara acak adalah langkah awal yang penting dalam algoritma PSO yang memungkinkan partikel untuk mulai menjelajahi ruang pencarian secara efektif dalam upaya mencari solusi optimal.

Berikut adalah contoh sederhana dari implementasi kecepatan awal individu secara acak dalam algoritma PSO dengan memperhatikan batasan dari setiap unit pembangkit menggunakan bahasa pemrograman MATLAB:

```
function velocities = initialize_velocities(swarm_size, dimension, max_velocity, min_velocity)
    % swarm_size: Jumlah partikel
    % dimension: Dimensi ruang pencarian
    % max_velocity: Batas atas kecepatan
    % min_velocity: Batas bawah kecepatan

    % Inisialisasi kecepatan awal secara acak dalam batasan yang ditentukan
    velocities = (max_velocity - min_velocity) .* rand(swarm_size, dimension) + min_velocity;
end

% Contoh pemanggilan fungsi untuk inisialisasi kecepatan
swarm_size = 50; % Jumlah partikel
dimension = 3; % Dimensi ruang pencarian
max_velocity = 1.5; % Batas atas kecepatan
min_velocity = -1.5; % Batas bawah kecepatan

velocities = initialize_velocities(swarm_size, dimension, max_velocity, min_velocity);
disp('Kecepatan awal partikel:');
disp(velocities);
```

Gambar 3.2. implementasi kecepatan awal individu secara acak

e. Melakukan evaluasi objektif terhadap individu ke-i berarti mengukur kinerja atau nilai objektif dari solusi yang diwakili oleh partikel ke-i dalam populasi. Evaluasi ini dilakukan dengan memasukkan posisi partikel ke dalam fungsi tujuan atau fungsi objektif yang ingin dioptimalkan. Dalam banyak kasus, fungsi objektif merupakan fungsi matematis yang harus diminimalkan atau dimaksimalkan, tergantung pada jenis masalah yang dihadapi. Contohnya, jika PSO digunakan untuk menyelesaikan masalah optimasi di mana kita mencari nilai minimum dari suatu fungsi, maka kita akan mengevaluasi partikel ke-i dengan memasukkan posisinya ke dalam fungsi tersebut dan mengukur nilai yang dihasilkan. Setelah evaluasi objektif dilakukan, nilai tersebut akan digunakan untuk memperbarui informasi terbaik yang dimiliki oleh partikel (misalnya, nilai terbaik yang pernah dicapai oleh partikel itu sendiri dan nilai terbaik yang pernah dicapai oleh seluruh populasi).

Berikut adalah contoh sederhana dari evaluasi objektif terhadap individu ke-i dalam algoritma PSO dengan memperhatikan batasan dari setiap unit pembangkit menggunakan bahasa pemrograman MATLAB :

```
function fitness = evaluate_objective(position)
    % Fungsi evaluasi objektif: Fungsi Rosenbrock
    % position: Vektor posisi partikel

    % Dimensi ruang pencarian
    dimension = length(position);

    % Inisialisasi nilai fitness
    fitness = 0;

    % Evaluasi fungsi Rosenbrock
    for i = 1:(dimension - 1)
        fitness = fitness + 100 * (position(i + 1) - position(i)^2)^2 + (position(i) - 1)^2;
    end
end

% Contoh pemanggilan fungsi evaluasi objektif
position = [0, 0]; % Contoh posisi partikel
fitness_value = evaluate_objective(position);
disp('Nilai fitness dari posisi partikel:');
disp(fitness_value);
```

Gambar 3.3. implementasi evaluasi objektif terhadap individu ke-i

f. Memperbarui kecepatan individu ke-*i* dalam algoritma *Particle Swarm Optimization* (PSO) adalah langkah yang dilakukan setelah melakukan evaluasi objektif terhadap individu tersebut. Tujuannya adalah untuk mengatur arah dan kecepatan pergerakan partikel ke arah solusi yang lebih baik berdasarkan pengalaman pribadi dan sosialnya. Persamaannya menggunakan persamaan 2.9.

g. Memperbarui pencapaian terbaik pribadi dan *global*. Memperbarui pencapaian terbaik pribadi *(personal best)* dan *global (global best)* dalam algoritma *Particle Swarm Optimization* (PSO) adalah langkah yang dilakukan setelah melakukan evaluasi objektif terhadap setiap individu dalam populasi.

- **Pencapaian Terbaik Pribadi (Personal Best)**: Pencapaian terbaik pribadi (disingkat sebagai P_i^{best}) adalah posisi terbaik yang pernah dicapai oleh individu ke-i dalam perjalanannya. Setiap partikel dalam PSO menyimpan informasi tentang pencapaian terbaik pribadinya sendiri. Pada setiap iterasi, setelah evaluasi objektif individu, pencapaian terbaik pribadi akan diperbarui jika nilai fitness dari posisi

baru lebih baik daripada nilai fitness dari pencapaian terbaik sebelumnya.

- **Pencapaian Terbaik Global (Global Best)**: Pencapaian terbaik global (disingkat sebagai P_g^{best}) adalah posisi terbaik yang pernah dicapai oleh seluruh populasi partikel. Ini adalah posisi yang memiliki nilai fitness terendah atau tertinggi di antara semua partikel dalam populasi. Pada setiap iterasi, pencapaian terbaik global akan diperbarui jika ada partikel yang menemukan solusi yang lebih baik daripada solusi terbaik yang pernah ditemukan sebelumnya dalam populasi.

Proses pembaruan pencapaian terbaik pribadi dan global biasanya dilakukan dengan langkah-langkah berikut:

- Setelah evaluasi objektif terhadap individu ke-i, bandingkan nilai fitnessnya dengan nilai fitness pencapaian terbaik pribadi sebelumnya (P_i^{best}). Jika nilai fitness yang baru lebih baik, maka perbarui P_i^{best} dengan posisi baru tersebut.
- Setelah semua partikel dalam populasi telah dievaluasi, bandingkan nilai fitness dari pencapaian terbaik pribadi masing-masing partikel dengan nilai fitness dari pencapaian terbaik global (P_g^{best}). Jika ada partikel yang memiliki nilai fitness yang lebih baik daripada P_g^{best}, maka perbarui P_g^{best} dengan posisi baru dari partikel tersebut.

Dengan memperbarui pencapaian terbaik pribadi dan global setiap iterasi, algoritma PSO dapat terus mengarahkan pencarian ke arah solusi yang lebih baik, berdasarkan pengalaman pribadi dan sosial partikel-partikel dalam populasi. Hal ini memungkinkan PSO untuk mendekati atau mencapai solusi optimum untuk masalah optimasi yang diberikan.

Berikut adalah contoh sederhana dari implementasi Memperbarui pencapaian terbaik pribadi dan *global* dalam algoritma PSO dengan memperhatikan batasan dari setiap unit pembangkit menggunakan bahasa pemrograman MATLAB :

```
function [personal_best, global_best] = update_best_positions(positions, personal_best, global_best, fitness_values)
    % positions: Matriks posisi partikel (swarm_size x dimension)
    % personal_best: Matriks posisi pencapaian terbaik pribadi (swarm_size x dimension)
    % global_best: Vektor posisi pencapaian terbaik global (1 x dimension)
    % fitness_values: Vektor nilai fitness untuk setiap partikel (1 x swarm_size)

    % Mendapatkan jumlah partikel dan dimensi ruang pencarian
    [swarm_size, dimension] = size(positions);

    % Memperbarui pencapaian terbaik pribadi
    for i = 1:swarm_size
        if fitness_values(i) < fitness_values(personal_best(i))
            personal_best(i, :) = positions(i, :);
        end
    end

    % Memperbarui pencapaian terbaik global
    [min_fitness, min_index] = min(fitness_values);
    if min_fitness < fitness_values(global_best)
        global_best = positions(min_index, :);
    end
end
```

```
% Contoh pemanggilan fungsi untuk memperbarui pencapaian terbaik
swarm_size = 50; % Jumlah partikel dalam populasi
dimension = 3; % Dimensi ruang pencarian

% Inisialisasi posisi, pencapaian terbaik pribadi, dan pencapaian terbaik global
positions = rand(swarm_size, dimension); % Inisialisasi posisi secara acak
personal_best = positions; % Pencapaian terbaik pribadi awal sama dengan posisi awal
global_best = positions(1, :); % Pencapaian terbaik global awal sama dengan posisi awal pertama
fitness_values = rand(1, swarm_size); % Contoh nilai fitness secara acak

% Memperbarui pencapaian terbaik
[personal_best, global_best] = update_best_positions(positions, personal_best, global_best, fitness_values);
```

Gambar 3.4. implementasi memperbarui pencapaian terbaik pribadi dan *global*

h. *Stopping criteria* ialah kondisi atau aturan yang digunakan untuk menentukan kapan proses pencarian dalam algoritma optimasi harus dihentikan. Dalam konteks *Particle Swarm Optimization* (PSO), kriteria berhenti menentukan kapan algoritma PSO harus berhenti mencari solusi optimal atau mendekati solusi optimal. Kriteria berhenti harus dipilih dengan bijaksana untuk memastikan bahwa algoritma PSO tidak berjalan terlalu lama atau terlalu singkat, dan mencapai hasil yang memadai dalam batas waktu yang ditentukan.

Berikut adalah contoh sederhana dari *Stopping criteria* dalam algoritma PSO dengan memperhatikan batasan dari setiap unit pembangkit menggunakan bahasa pemrograman MATLAB :

```
function stop_criteria = check_stop_criteria(iteration, max_iterations, fitness_values, threshold)
    % iterasi maksimum
    if iteration >= max_iterations
        stop_criteria = true;
        disp('Stop karena jumlah iterasi maksimum tercapai.');
        return;
    end

    % konvergensi (perubahan nilai fitness tidak signifikan)
    if iteration > 1 && abs(fitness_values(end) - fitness_values(end - 1)) < threshold
        stop_criteria = true;
        disp('Stop karena konvergensi: perubahan nilai fitness tidak signifikan.');
        return;
    end

    % tambahkan kriteria berhenti lainnya di sini

    stop_criteria = false;
end
```

```
% Contoh penggunaan fungsi untuk memeriksa kriteria berhenti
max_iterations = 1000; % jumlah iterasi maksimum
threshold = 1e-6; % ambang batas perubahan nilai fitness
fitness_values = rand(1, max_iterations); % contoh nilai fitness
iteration = 50; % iterasi saat ini

stop_criteria = check_stop_criteria(iteration, max_iterations, fitness_values, threshold);
if stop_criteria
    disp('Kriteria berhenti terpenuhi. Proses pencarian dihentikan.');
else
    disp('Pencarian masih berlanjut.');
end
```

Gambar 3.5. implementasi *Stopping criteria*

i. Melakukan penghitungan total biaya bahan bakar pembangkitan ialah proses untuk mengestimasi total biaya yang diperlukan untuk memproduksi energi listrik oleh pembangkit listrik. Biaya bahan bakar adalah salah satu komponen utama dalam biaya operasional pembangkit listrik, dan penghitungannya tergantung pada beberapa faktor, seperti jenis bahan bakar yang digunakan, efisiensi pembangkit, harga bahan bakar, dan kapasitas pembangkit.

Untuk menghitung total biaya bahan bakar pembangkitan, langkah-langkah umumnya adalah sebagai berikut:

- **Estimasi Konsumsi Bahan Bakar**: Hitung total konsumsi bahan bakar yang diperlukan untuk menghasilkan energi listrik dalam periode waktu tertentu. Ini dapat dihitung berdasarkan kapasitas pembangkit,

efisiensi pembangkit, dan jumlah energi listrik yang diproduksi.

- **Harga Bahan Bakar**: Tentukan harga bahan bakar yang digunakan oleh pembangkit listrik. Harga bahan bakar dapat bervariasi tergantung pada jenis bahan bakar dan pasar lokal.
- **Penghitungan Biaya Bahan Bakar**: Kalikan total konsumsi bahan bakar dengan harga bahan bakar untuk mendapatkan total biaya bahan bakar pembangkitan dalam periode waktu yang sama.

Misalnya, jika sebuah pembangkit listrik menggunakan batu bara sebagai bahan bakar, dan pada suatu periode waktu pembangkit tersebut mengonsumsi 100 ton batu bara dengan harga 50.000 per ton, maka total biaya bahan bakar pembangkitan dalam periode waktu tersebut akan menjadi:

Total Biaya Bahan Bakar=Konsumsi Bahan Bakar×Harga Bahan BakarTotal Biaya Bahan Bakar=Konsumsi Bahan Bakar×Harga Bahan Bakar

Total Biaya Bahan Bakar=100ton×$50.000/ton

Total Biaya Bahan Bakar=$5.000.000

Penghitungan ini dapat disesuaikan dengan variabel-variabel yang relevan dan dengan menggunakan data aktual yang tersedia untuk pembangkit listrik yang bersangkutan. Total biaya bahan bakar pembangkitan adalah salah satu faktor yang penting dalam analisis ekonomi dan keberlanjutan pembangkit listrik, karena dapat mempengaruhi harga jual energi listrik, profitabilitas, dan dampak lingkungan.