Catatan Kuliah Minggu ke-13:
Economic Dispatch Sistem Tenaga Listrik Berbasis Metaheuristik Particle Swarm Optimization (PSO)

Ir. Novalio Daratha S.T., M.Sc., Ph.D.
Program Studi S1 Teknik Elektro — Universitas Bengkulu

Semester Ganjil 2026

Ringkasan Eksekutif dan CPMK

Dokumen ini menyajikan panduan perkuliahan komprehensif mengenai aplikasi *Particle Swarm Optimization* (PSO) pada masalah *Economic Dispatch* (ED) dalam sistem tenaga listrik. Mahasiswa mempelajari pembentukan fungsi biaya bahan bakar pembangkit termal (termasuk efek *valve-point loading*), perumusan kendala keseimbangan daya dan batas generator, serta mekanisme pengkodean partikel PSO dan *constraint handling*.

Daftar Isi

1 Pendahuluan Economic Dispatch **2**
1.1 Karakteristik Biaya Bahan Bakar Pembangkit Termal . . . 2
1.2 Efek Valve-Point Loading (Non-Konveksitas) . . . 2

2 Formulasi Matematis Masalah ED **2**
2.1 Fungsi Tujuan . . . 2
2.2 Kendala-Kendala (Constraints) . . . 2
2.2.1 1. Keseimbangan Daya (Power Balance Constraint) . . . 2
2.2.2 2. Batas Kapasitas Pembangkit (Generator Limits) . . . 3

3 Penyelesaian dengan Particle Swarm Optimization (PSO) **3**
3.1 Representasi Partikel . . . 3
3.2 Fungsi Fitness dan Penalti . . . 3
3.3 Mekanisme Repair Generator (Slack Generator) . . . 3

4 Implementasi Program Python **3**

5 Kesimpulan dan Diskusi **5**

1 Pendahuluan Economic Dispatch

Dalam pengoperasian sistem tenaga listrik modern, penyediaan energi listrik kepada konsumen harus memenuhi kriteria utama: **andal, berkualitas, dan ekonomis**. Masalah *Economic Dispatch* (ED) berada pada tingkatan perencanaan operasi jangka pendek (*short-term operational planning*) yang bertujuan menentukan alokasi daya aktif ($P_{G,i}$) untuk setiap unit pembangkit termal yang sedang beroperasi sedemikian rupa sehingga total biaya bahan bakar ($/jam) mencapai nilai minimum.

1.1 Karakteristik Biaya Bahan Bakar Pembangkit Termal

Fungsi biaya bahan bakar pembangkit termal pada kondisi ideal umumnya didekati dengan persamaan kuadratik:

$$F_i(P_{G,i}) = a_i P_{G,i}^2 + b_i P_{G,i} + c_i \quad [\$/\text{jam}] \tag{1}$$

di mana:

- $P_{G,i}$: Daya aktif keluaran pembangkit ke-i dalam MegaWatt (MW).
- a_i: Koefisien biaya bahan bakar kuadratik ($/MW^{2}h).
- b_i: Koefisien biaya bahan bakar linier ($/MWh).
- c_i: Biaya tanpa beban / *no-load cost* ($/jam).

1.2 Efek Valve-Point Loading (Non-Konveksitas)

Pada kenyataannya, turbin uap menggunakan sekumpulan katup (*valves*) untuk mengatur masukan uap. Saat katup pembuka baru diaktifkan, terjadi efek pembukaan tercekik (*wire-drawing effect*) yang menimbulkan kerugian efisiensi sementara. Fenomena ini dimodelkan dengan menambahkan komponen sinusoidal pada fungsi biaya kuadratik:

$$F_i(P_{G,i}) = a_i P_{G,i}^2 + b_i P_{G,i} + c_i + |e_i \sin(f_i(P_{i,\min} - P_{G,i}))| \tag{2}$$

di mana e_i dan f_i adalah koefisien efek *valve-point*. Penambahan suku rippel ini membuat fungsi biaya bersifat **non-konveks, non-diferensiabel, dan memiliki banyak optimum lokal**, sehingga metode optimisasi berbasis turunan klasik tidak lagi efektif.

2 Formulasi Matematis Masalah ED

2.1 Fungsi Tujuan

Meminimalkan total biaya bahan bakar dari seluruh N_g pembangkit:

$$\min_{\mathbf{P}_G} F_T = \sum_{i=1}^{N_g} F_i(P_{G,i}) \tag{3}$$

2.2 Kendala-Kendala (Constraints)

2.2.1 1. Keseimbangan Daya (Power Balance Constraint)

Total daya pembangkitan harus sama dengan jumlah total beban sistem (P_D) dan rugi-rugi transmisi (P_L):

$$\sum_{i=1}^{N_g} P_{G,i} - P_D - P_L = 0 \tag{4}$$

Rugi-rugi transmisi P_L dimodelkan menggunakan matriks rugi-rugi Kron (B-coefficients):

$$P_L = \sum_{i=1}^{N_g} \sum_{j=1}^{N_g} P_{G,i} B_{ij} P_{G,j} + \sum_{i=1}^{N_g} B_{0i} P_{G,i} + B_{00} \tag{5}$$

2.2.2 2. Batas Kapasitas Pembangkit (Generator Limits)

Daya keluaran setiap unit pembangkatan dibatasi oleh kemampuan mekanis dan termal unit:

$$P_{i,\min} \leq P_{G,i} \leq P_{i,\max}, \quad i = 1, 2, \ldots, N_g \tag{6}$$

3 Penyelesaian dengan Particle Swarm Optimization (PSO)

3.1 Representasi Partikel

Pada masalah ED dengan N_g unit pembangkit, setiap partikel k mewakili satu calon solusi alokasi daya:

$$\mathbf{x}_k = [P_{G,1}^{(k)}, P_{G,2}^{(k)}, \ldots, P_{G,N_g}^{(k)}] \tag{7}$$

Kecepatan partikel mewakili laju perubahan alokasi daya:

$$\mathbf{v}_k = [v_1^{(k)}, v_2^{(k)}, \ldots, v_{N_g}^{(k)}] \tag{8}$$

3.2 Fungsi Fitness dan Penalti

Untuk menangani kendala kesetaraan daya, digunakan fungsi penalti kuadratik:

$$\text{Fitness}(\mathbf{x}_k) = \sum_{i=1}^{N_g} F_i(P_{G,i}^{(k)}) + \lambda_{\text{pen}} \left(\sum_{i=1}^{N_g} P_{G,i}^{(k)} - P_D - P_L \right)^2 \tag{9}$$

di mana λ_{pen} adalah faktor penalti berbobot besar (misal 10^5).

3.3 Mekanisme Repair Generator (Slack Generator)

Pendekatan alternatif yang sangat efisien untuk menangani kesetaraan daya adalah penetapan generator terakhir N_g sebagai *slack generator*:

$$P_{G,N_g} = P_D + P_L - \sum_{i=1}^{N_g - 1} P_{G,i} \tag{10}$$

Setelah nilai P_{G,N_g} dihitung, dilakukan *clamping* batas daya:

$$P_{G,N_g} = \min\left(P_{N_g,\max}, \max\left(P_{N_g,\min}, P_{G,N_g}\right)\right) \tag{11}$$

4 Implementasi Program Python

Berikut adalah skrip Python lengkap untuk menyelesaikan Economic Dispatch 3 Pembangkit menggunakan PSO:

Listing 1: Skrip Python Economic Dispatch dengan PSO

```python
import numpy as np

# Data Pembangkit: [a ($/MW^2h), b ($/MWh), c ($/h), Pmin (MW), Pmax (MW)]
gen_data = np.array([
    [0.00156, 7.92, 561, 100, 600],
    [0.00194, 7.85, 310, 100, 400],
    [0.00482, 7.97, 78, 50, 200]
])

P_D = 850.0  # Total beban (MW)
N_g = len(gen_data)

# Parameter PSO
N_particles = 40
Max_iter = 100
w = 0.7
c1 = 1.5
c2 = 1.5

# Inisialisasi partikel
Pmin = gen_data[:, 3]
Pmax = gen_data[:, 4]
X = np.random.uniform(Pmin, Pmax, size=(N_particles, N_g))
V = np.zeros((N_particles, N_g))

def evaluate_cost(x_vec):
    P = np.copy(x_vec)
    # Terapkan clamping pada N_g - 1 generator
    P[:-1] = np.clip(P[:-1], Pmin[:-1], Pmax[:-1])
    # Hitung generator slack
    P[-1] = P_D - np.sum(P[:-1])

    # Hitung penalti jika slack melanggar batas
    penalty = 0
    if P[-1] < Pmin[-1] or P[-1] > Pmax[-1]:
        penalty = 1e5 * (abs(P[-1] - np.clip(P[-1], Pmin[-1], Pmax[-1])))**2

    cost = np.sum(gen_data[:, 0] * P**2 + gen_data[:, 1] * P + gen_data[:, 2])
    return cost + penalty, P

# Evaluasi awal
pbest_X = np.copy(X)
pbest_fit = np.array([evaluate_cost(X[i])[0] for i in range(N_particles)])
gbest_idx = np.argmin(pbest_fit)
gbest_X = np.copy(pbest_X[gbest_idx])
gbest_fit = pbest_fit[gbest_idx]

# Loop Utama PSO
for t in range(Max_iter):
    for i in range(N_particles):
        r1, r2 = np.random.rand(N_g), np.random.rand(N_g)
        V[i] = w * V[i] + c1 * r1 * (pbest_X[i] - X[i]) + c2 * r2 * (gbest_X -
            X[i])
        X[i] = X[i] + V[i]

        fit_val, _ = evaluate_cost(X[i])
        if fit_val < pbest_fit[i]:
            pbest_fit[i] = fit_val
            pbest_X[i] = np.copy(X[i])
            if fit_val < gbest_fit:
                gbest_fit = fit_val
                gbest_X = np.copy(X[i])
```

```
62
63 _, P_opt = evaluate_cost(gbest_X)
64 print(f"Alokasi Daya Optimal: P1 = {P_opt[0]:.2f} MW, P2 = {P_opt[1]:.2f} MW,
       P3 = {P_opt[2]:.2f} MW")
65 print(f"Total Biaya Operasi Minimum: ${gbest_fit:.2f} / jam")
```

5 Kesimpulan dan Diskusi

1. Masalah *Economic Dispatch* dapat diformulasikan secara matematis dan diselesaikan secara cepat menggunakan PSO. 2. Penggunaan strategi *slack generator repair* mempercepat konvergensi PSO dibandingkan penalti biasa. 3. Pendekatan berbasis PSO sangat adaptif untuk memperhitungkan non-konveksitas *valve-point loading* dan rugi-rugi transmisi B_{ij}.