Universidad Autónoma de Ciudad Juárez

Instituto de Ingeniería y Tecnología
Departamento de Ingeniería Eléctrica y Computación
Maestría en Inteligencia Artificial y Analítica de Datos

Reporte de práctica

El problema de despacho económico de carga

Alumno: Aminadab Córdova Acosta
Asignatura: Programación para Analítica Prescriptiva y de Apoyo a la Decisión
Instructor: Dr. Josué Domínguez Guerrero

25 de marzo de 2025

Resumen

Este reporte presenta la resolución de un caso práctico de Despacho Económico de Carga utilizando Pyomo en Python. Se formula un modelo de optimización no lineal para minimizar el costo de generación, asegurando el cumplimiento de restricciones técnicas y operativas. La resolución se lleva a cabo con el solver IPOPT, especializado en optimización no lineal. Finalmente, se interpretan los resultados del modelo, destacando su impacto en la toma de decisiones dentro del sistema eléctrico.

Índice

Resumen I

1. Introducción 1

2. Descripción general del caso de estudio 2

3. Modelo matemático 3
- 3.1. Función Objetivo . . . 3
- 3.2. Restricciones de Balance de Potencia . . . 3
- 3.3. Restricciones de Generación . . . 4
- 3.4. Límite de velocidad de rampa . . . 4
- 3.5. Límites de las zonas de operación prohibida . . . 4

4. Código en Python 5

5. Capturas de pantalla 8

6. Interpretación de resultados 9

1. Introducción

El problema de despacho económico de carga consiste en determinar la cantidad de potencia que deben suministrar las unidades de generación a la red eléctrica, de manera que se cubra la demanda total de energía al menor costo posible, cumpliendo al mismo tiempo con las restricciones técnicas y operativas del sistema (Marzbani and Abdelfatah, 2024).

La optimización juega un papel fundamental en este proceso, ya que el despacho económico se plantea como un problema matemático en el que se busca minimizar el costo total de generación, sujeto a diversas restricciones económicas y técnicas del sistema eléctrico (Wood and Wollenberg, 1983).

En este trabajo se replica el caso de estudio del sistema 1 presentado en (Fu et al., 2020), donde se formula un modelo de despacho económico de carga, ilustrado en la Figura 1 y se resuelve utilizando Pyomo, una biblioteca de optimización en Python. Los datos del sistema se toman de esta referencia. Se presenta una descripción detallada del problema, la formulación matemática del modelo, la implementación en Pyomo, y la interpretación de los resultados obtenidos.

2. Descripción general del caso de estudio

El caso de estudio se centra en un sistema eléctrico compuesto por seis unidades generadoras ilustrado en la Fig. 1, cada una con características técnicas y económicas específicas. El objetivo principal es determinar la asignación óptima de potencia generada por cada unidad, minimizando el costo total de generación mientras se cumplen las restricciones operativas y técnicas del sistema.

El sistema considera los siguientes aspectos clave que se encuentran en la Tabla 4 y el apéndice A de (Fu et al., 2020):

- **Restricciones de generación:** Cada unidad generadora tiene límites mínimos y máximos de potencia que deben respetarse.
- **Restricciones de rampa:** Las unidades generadoras tienen límites en la velocidad de cambio de potencia entre periodos consecutivos.
- **Pérdidas de transmisión:** Se incluyen pérdidas de transmisión en las líneas eléctricas, modeladas mediante una matriz de coeficientes B.
- **Demanda total:** La generación total debe satisfacer la demanda del sistema (1263 MW), considerando las pérdidas de transmisión.
- **Costos de generación:** Los costos de generación de cada unidad se modelan como funciones cuadráticas, con términos adicionales para representar efectos no lineales como el punto de válvula.

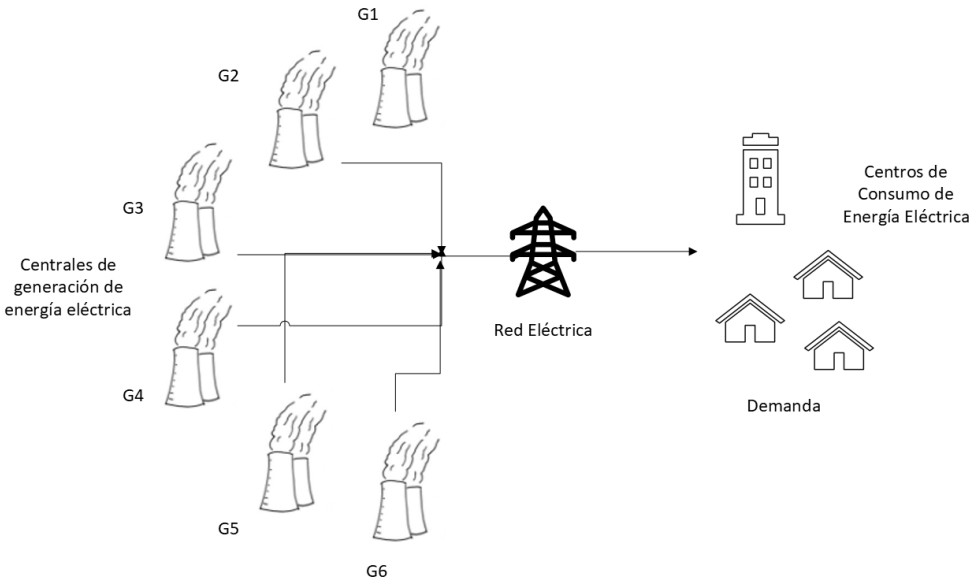

Figura 1: Representación del problema de despacho económico de carga.

3. Modelo matemático

3.1. Función Objetivo

La función objetivo consiste en minimizar el costo total de generación del sistema, que se calcula como la suma de los costos de generación de cada unidad. Cada costo de generación se modela como un polinomio cuadrático dependiente de la potencia generada, con términos adicionales para representar efectos no lineales como el punto de válvula. La función objetivo se expresa de la siguiente manera:

$$\text{mín}(F_T) = \text{mín} \sum_{i=1}^{N} F_i(P_{Gi}) \tag{1}$$

Donde F_T indica el costo total del combustible, N representa el numero total de unidades de generación, P_{Gi} es la salida de potencia activa del i-th generador, y $F_i(P_{Gi})$ significa el costo del consumo de combustible correspondiente del i-th generador, y se puede expresar como un polinomio cuadrático dependiente de la potencian:

$$F_i(P_{Gi}) = a_i(P_{Gi})^2 + b_i(P_{Gi}) + c_i \tag{2}$$

donde a_i, b_i, and c_i representan los coeficientes de costo del combustible. Cuando la válvula de admisión de la turbomáquina se abre repentinamente, el resultado, conocido como 'efecto de punto de válvula', puede representarse generalmente como una función sinusoidal. Esta función sinusoidal se añadirá a la tradicional función de costo de combustible cuadrática, es decir:

$$F_i(P_{Gi}) = a_i(P_{Gi})^2 + b_i(P_{Gi}) + c_i + |d_i \sin(e_i(P_{Gi}^{\text{mín}} - P_{Gi}))| \tag{3}$$

Donde $P_{Gi}^{\text{mín}}$ representa el límite inferior de la potencia de salida del i-th generador, y $|d_i|$ y e_i son los coeficientes de costo.

3.2. Restricciones de Balance de Potencia

Las restricciones de equilibrio de potencia son las restricciones más críticas en la operación de la unidad generadora. Si no se cumple esta restricción, se producirá la parálisis del sistema eléctrico y se amenazará seriamente la fiabilidad de la operación del sistema. Esta restricción se puede resumir en que la producción total de todas las unidades generadoras debe ser igual a la suma de la carga y la pérdida de transmisión

$$\sum_{i=1}^{N} P_{Gi} = P_D + P_L \tag{4}$$

$$P_L = \sum_{i=1}^{N} \sum_{j=1}^{N} P_{Gi} B_{ij} P_{Gj} + \sum_{i=1}^{N} B_{i0} P_{Gi} + B_{00} \tag{5}$$

3.3. Restricciones de Generación

La potencia de salida de cada generador debe estar dentro de los límites de generación, es decir:

$$P_{Gi}^{\text{mín}} \leq P_{Gi} \leq P_{Gi}^{\text{máx}} \tag{6}$$

donde $P_{Gi}^{\text{mín}}$ y $P_{Gi}^{\text{máx}}$ son los límites inferior y superior de la potencia de salida del i-th generador, respectivamente.

3.4. Límite de velocidad de rampa

Dentro de un período de tiempo, la variación de potencia está limitada por el funcionamiento del generador. Esta restricción se describe a continuación:

$$-DR_i \leq P_{Gi} - P_{Gi(t-1)} \leq UR_i \tag{7}$$

Donde DR_i y UR_i son las tasas de disminución y aumento de potencia del i-th generador, respectivamente.

3.5. Límites de las zonas de operación prohibida

La potencia de salida de cada generador debe estar fuera de las zonas de operación prohibida, es decir:

$$P_{Gi}^{\text{mín}} \leq P_{Gi} \leq P_{Gi}^{\text{máx}} \tag{8}$$

Los cojinetes del generador producirán vibraciones intensas cuando el generador funcione en algunas zonas. Por lo tanto, la potencia de salida debe ajustarse para evitar zonas de funcionamiento prohibidas durante la operación real. Los límites de las zonas de funcionamiento prohibidas se enumeran a continuación:

$$\begin{cases} P_{Gi}^{\text{mín}} \leq P_{Gi} \leq P_{Gi}^{1} \\ P_{Gi}^{k-1} \leq P_{Gi} \leq P_{Gi}^{k}, \quad k = 2, 3, \ldots, n_Z \\ P_{Gi}^{n_Z} \leq P_{Gi} \leq P_{Gi}^{\text{máx}} \end{cases} \tag{9}$$

4. Código en Python

El modelo matemático presentado anteriormente se implementa en Pyomo, una biblioteca de optimización en Python. Pyomo permite definir modelos de optimización de manera sencilla y eficiente, y proporciona una interfaz para resolverlos con diversos solvers. A continuación se presenta la implementación del modelo de despacho económico de carga en Pyomo.

```python
from pyomo.environ import *

# Crear el modelo
model = ConcreteModel()

# Conjunto de generadores
model.Generadores = Set(initialize=['G1', 'G2', 'G3', 'G4', 'G5', 'G6'])

# Parámetros de los generadores
datos_generadores = {
    'G1': {'Pmax': 500, 'Pmin': 100, 'ai': 0.007, 'bi': 7, 'ci': 240, 'UR': 80, 'DR': 120, 'Pi': 440},
    'G2': {'Pmax': 200, 'Pmin': 50, 'ai': 0.0095, 'bi': 10, 'ci': 200, 'UR': 50, 'DR': 90, 'Pi': 170},
    'G3': {'Pmax': 300, 'Pmin': 80, 'ai': 0.009, 'bi': 8.5, 'ci': 220, 'UR': 65, 'DR': 100, 'Pi': 200},
    'G4': {'Pmax': 150, 'Pmin': 50, 'ai': 0.009, 'bi': 11, 'ci': 200, 'UR': 50, 'DR': 90, 'Pi': 150},
    'G5': {'Pmax': 200, 'Pmin': 50, 'ai': 0.008, 'bi': 10.5, 'ci': 220, 'UR': 50, 'DR': 90, 'Pi': 190},
    'G6': {'Pmax': 120, 'Pmin': 50, 'ai': 0.0075, 'bi': 12, 'ci': 190, 'UR': 50, 'DR': 90, 'Pi': 110}
}

# Definir parámetros en el modelo
model.Pmax = Param(model.Generadores, initialize={g: datos_generadores[g]['Pmax'] for g in model.Generadores})
model.Pmin = Param(model.Generadores, initialize={g: datos_generadores[g]['Pmin'] for g in model.Generadores})
model.ai = Param(model.Generadores, initialize={g: datos_generadores[g]['ai'] for g in model.Generadores})
model.bi = Param(model.Generadores, initialize={g: datos_generadores[g]['bi'] for g in model.Generadores})
model.ci = Param(model.Generadores, initialize={g: datos_generadores[g]['ci'] for g in model.Generadores})
model.UR = Param(model.Generadores, initialize={g: datos_generadores[g]['UR'] for g in model.Generadores})
model.DR = Param(model.Generadores, initialize={g: datos_generadores[g]['DR'] for g in model.Generadores})
model.Pi = Param(model.Generadores, initialize={g: datos_generadores[g]['Pi'] for g in model.Generadores})

# Parámetro de demanda total
demanda_total = 1263
model.Demanda = Param(initialize=demanda_total)
```

```
32
33 # Matriz B y coeficientes (Para el cálculo aproximado de las pérdidas de transmisi
       ón)
34 B1 = [[0.01 * val for val in row] for row in [
35     [0.0017, 0.0012, 0.0007, -0.0001, -0.0005, -0.0002],
36     [0.0012, 0.0014, 0.0009,  0.0001, -0.0006, -0.0001],
37     [0.0007, 0.0009, 0.0031,  0.0000, -0.0010, -0.0006],
38     [-0.0001, 0.0001, 0.0000,  0.0024, -0.0006, -0.0008],
39     [-0.0005, -0.0006, -0.0010, -0.0006, 0.0129, -0.0002],
40     [-0.0002, -0.0001, -0.0006, -0.0008, -0.0002, 0.0150]
41 ]]
42
43 B2 = [0.001 * val for val in [-0.3908, -0.1297, 0.7047, 0.0591, 0.2161, -0.6635]]
44 B3 = 0.56
45
46 # Definir parámetros de pérdidas
47 model.B1 = Param(model.Generadores, model.Generadores, initialize={(g1, g2): B1[i
       ][j] for i, g1 in enumerate(model.Generadores) for j, g2 in enumerate(model.
       Generadores)})
48 model.B2 = Param(model.Generadores, initialize={g: B2[i] for i, g in enumerate(
       model.Generadores)})
49 model.B3 = Param(initialize=B3)
50
51 # Variable de generación de cada generador
52 model.Pg = Var(model.Generadores, within=NonNegativeReals)
53
54 # Pérdidas
55 model.PL = Expression(expr=sum(model.B1[g1, g2] * model.Pg[g1] * model.Pg[g2] for
       g1 in model.Generadores for g2 in model.Generadores) +
56                             sum(model.B2[g] * model.Pg[g] for g in model.
                                 Generadores) + model.B3)
57
58 # Restricciones de límites de generación
59 model.LimiteInferior = Constraint(model.Generadores, rule=lambda m, g: m.Pmin[g]
       <= m.Pg[g])
60 model.LimiteSuperior = Constraint(model.Generadores, rule=lambda m, g: m.Pg[g] <=
       m.Pmax[g])
61
62 # Restricciones de rampas
63 model.RampaSubida = Constraint(model.Generadores, rule=lambda m, g: m.Pg[g] - m.Pi
       [g] <= m.UR[g])
64 model.RampaBajada = Constraint(model.Generadores, rule=lambda m, g: m.Pi[g] - m.Pg
       [g] <= m.DR[g])
65
66 # Restricción de demanda considerando pérdidas
67 model.DemandaTotal = Constraint(rule=lambda m: sum(m.Pg[g] for g in m.Generadores)
        == m.Demanda + m.PL)
68
69 # Función objetivo: minimizar el costo de generación
70 model.CostoTotal = Objective(
71     expr=sum(model.ai[g] * model.Pg[g] ** 2 + model.bi[g] * model.Pg[g] + model.ci
          [g] for g in model.Generadores),
72     sense=minimize
```

```
)

# Resolver el modelo con IPOPT
solver = SolverFactory('ipopt')
result = solver.solve(model)

# Mostrar resultados
print("Resultados de la generación de cada unidad:")
for g in model.Generadores:
    print(f"Generación de {g}: {model.Pg[g].value:.2f} MW")

print(f"\nPérdidas totales: {model.PL():.2f} MW")
print(f"Costo total de generación: {model.CostoTotal.expr():.2f} $")
```

Listing 1: Modelo de Despacho Económico con Pyomo

5. Capturas de pantalla

A continuación, se presentan las capturas de pantalla que muestran los resultados obtenidos al ejecutar el código en el entorno de desarrollo Visual Studio Code

6. Interpretación de resultados

El problema de despacho económico de carga se puede formular como un problema de programación lineal, donde se busca minimizar el costo total de generación sujeto a diversas restricciones. A continuación se presenta una formulación simplificada del problema, donde se considera un sistema eléctrico con n plantas generadoras y m demandas de energía.

Referencias

Fu, C., Zhang, S., and Chao, K.-H. (2020). Energy management of a power system for economic load dispatch using the artificial intelligent algorithm. *Electronics*, 9(1).

Marzbani, F. and Abdelfatah, A. (2024). Economic dispatch optimization strategies and problem formulation: A comprehensive review. *Energies*, 17(3).

Wood, A. J. and Wollenberg, B. F. (1983). *Power generation, operation, and control*. John Wiley and Sons,New York, NY.