EVALUASI MANDIRI OPTIMISASI
MATAKULIAH OPTIMISASI
MENGENALI OPTIMISASI INTEGER CAMPURAN

Disusun oleh:

SHARA ALYA GIFANI MUHYISUNAH

G1D021038

Dosen Pengampu:

Ir. Novalio Daratha S.T., M.Sc., Ph.D.

PROGRAM STUDI TEKNIK ELEKTRO

FAKULTAS TEKNIK

UNIVERSITAS BENGKULU

2024

Optimasi Integer Campuran (Mixed Integer Optimization / MIP) adalah jenis masalah optimasi di mana sebagian dari variabel keputusan harus berupa bilangan bulat (integer), sementara sebagian lainnya dapat berupa variabel kontinu (real numbers).

Ciri-ciri dari Optimasi Integer Campuran:

1. Variabel Integer: Beberapa variabel dalam model optimasi harus berupa bilangan bulat. Contohnya adalah ketika kita memutuskan jumlah produk yang harus diproduksi (misalnya, kita tidak bisa memproduksi setengah unit produk).
2. Variabel Kontinu: Selain variabel integer, ada juga variabel yang dapat mengambil nilai kontinu atau real, yang artinya bisa berupa bilangan pecahan. Contohnya adalah waktu atau jumlah bahan baku yang digunakan.
3. Kombinasi: Dalam model ini, terdapat campuran antara variabel integer dan variabel kontinu, yang membuatnya lebih kompleks untuk diselesaikan dibandingkan masalah optimasi linear biasa (yang hanya melibatkan variabel kontinu).

CODE PROGRAM

```
using JuMP

using GLPK


# Membuat model optimisasi

model = Model(GLPK.Optimizer)


# Mendefinisikan variabel keputusan

@variable(model, y1 >= 0, Int)  # Produk C (integer)

@variable(model, y2 >= 0)       # Produk D (continuous)


# Menambahkan fungsi tujuan Z = 4y1 + 3y2

@objective(model, Max, 4 * y1 + 3 * y2)


# Menambahkan kendala

@constraint(model, 3 * y1 + 2 * y2 <= 10)  # 3y1 + 2y2 <= 10

@constraint(model, y1 >= 2)  # y1 >= 2


# Menyelesaikan model
```

```
optimize!(model)


# Menampilkan hasil
println("Status: ", termination_status(model))
println("Jumlah Produk C yang diproduksi: ", value(y1))
println("Jumlah Produk D yang diproduksi: ", value(y2))
println("Keuntungan maksimum: ", objective_value(model))
```

1. Membuat Model Optimasi

```
model = Model(GLPK.Optimizer)
```

- Kode ini membuat sebuah model optimasi menggunakan paket JuMP dan solver GLPK (GNU Linear Programming Kit). Model ini akan digunakan untuk menyelesaikan masalah optimasi yang Anda definisikan.

2. Mendefinisikan Variabel Keputusan

```
@variable(model, y1 >= 0, Int)  # Produk C (integer)
@variable(model, y2 >= 0)       # Produk D (continuous)
```

- y1: Merupakan variabel keputusan yang mewakili jumlah Produk C yang diproduksi. Variabel ini harus berupa bilangan bulat (integer) karena jumlah produk tidak bisa berupa angka pecahan.
- y2: Merupakan variabel keputusan yang mewakili jumlah Produk D yang diproduksi. Variabel ini dapat berupa bilangan kontinu karena jumlah produk dapat dinyatakan dalam angka pecahan.

3. Menambahkan Fungsi Tujuan

```
@objective(model, Max, 4 * y1 + 3 * y2)
```

- Fungsi tujuan mendefinisikan apa yang ingin dicapai dalam masalah optimasi, dalam hal ini keuntungan maksimum. Fungsi tujuan ini adalah untuk memaksimalkan keuntungan yang dihasilkan dari jumlah produk yang diproduksi.
    - Keuntungan dari Produk C (y1) adalah 4 per unit.
    - Keuntungan dari Produk D (y2) adalah 3 per unit.
- Dengan demikian, fungsi tujuan adalah 4 * y1 + 3 * y2.

4. Menambahkan Kendala

```
@constraint(model, 3 * y1 + 2 * y2 <= 10)  # 3y1 + 2y2 <= 10
@constraint(model, y1 >= 2)  # y1 >= 2
```

- Kendala pertama: 3 * y1 + 2 * y2 <= 10 membatasi kombinasi produk C (y1) dan produk D (y2) yang diproduksi. Secara fisik, misalnya ini bisa merujuk pada batasan sumber daya (seperti waktu atau bahan baku) yang tersedia, di mana 3 unit dari Produk C dan 2 unit dari Produk D tidak boleh melebihi 10 unit total.
- Kendala kedua: y1 >= 2 menyatakan bahwa jumlah Produk C (y1) yang diproduksi harus lebih besar atau sama dengan 2 unit, yaitu pengusaha ingin memproduksi setidaknya 2 unit Produk C.

5. Menyelesaikan Model

```
optimize!(model)
```

- Fungsi optimize! digunakan untuk menyelesaikan model optimasi. Di sini, solver GLPK akan mencari solusi yang optimal, yaitu nilai y1 dan y2 yang memenuhi semua kendala dan memaksimalkan fungsi tujuan (keuntungan).

6. Menampilkan Hasil

```
println("Status: ", termination_status(model))

println("Jumlah Produk C yang diproduksi: ", value(y1))

println("Jumlah Produk D yang diproduksi: ", value(y2))

println("Keuntungan maksimum: ", objective_value(model))
```

- termination_status(model): Menampilkan status hasil optimasi (misalnya apakah solusi ditemukan atau tidak).
- value(y1): Menampilkan jumlah unit Produk C yang diproduksi dalam solusi optimal.
- value(y2): Menampilkan jumlah unit Produk D yang diproduksi dalam solusi optimal.
- objective_value(model): Menampilkan keuntungan maksimum yang dicapai dengan memproduksi Produk C dan Produk D dalam jumlah yang optimal.