

**TUGAS 1 OPTIMISASI  
MATAKULIAH OPTIMISASI  
NETWORK FLOW PROBLEM**



**NAMA: SHARA ALYA GIFANI MUHYISUNAH  
NPM : G1D021038**

**PROGRAM STUDI TEKNIK ELEKTRO  
FAKULTAS TEKNIK  
UNIVERSITAS BENGKULU  
2024**

Nama : Shara Alya Gifani Muhyisunah  
NPM : G1D021038  
Matakuliah : Optimisasi

Tugas 1 Optimisasi  
Network Flow Problem

---



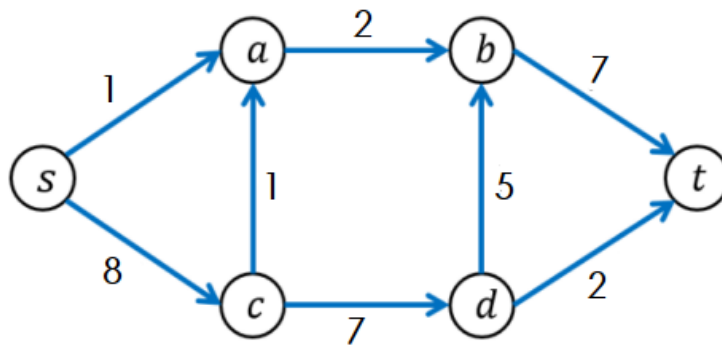
# Tugas 1 Optimisasi Program Studi Teknik Elektro Universitas Bengkulu

Dosen Pengampu: Novalio Daratha

1. Install [Julia](#)
  - a. [Install JuMP pada julia](#)
  - b. Install solver [HiGHS](#) dan [Ipopt](#) pada julia
2. Pelajari tentang network flow problem. [Misalnya dengan memahami link ini.](#)
3. Buat atau cari contoh network flow problem.
4. Pecahkan soal tersebut dengan menggunakan JuMP Dan HiGHS.
5. Buat video YouTube yang membahas kegiatan 3 dan 4.
6. Upload source code yang Anda gunakan ke akun GitHub masing-masing.
7. Buat laporan singkat tentang kegiatan 3 Dan 4. Lampirkan hal berikut:
  - a. Source code yang digunakan.
  - b. Link video YouTube yang menjelaskan tahapan 3 Dan 4.
  - c. Link GitHub terkait.
8. Upload laporan ke GitHub Anda.
9. Kegiatan 1 dimulai pukul 8:00 Rabu, 28 Agustus 2024. Kegiatan 8 selesai sebelum pukul 8:00 Rabu, 4 September 2024.

**Selamat berjuang!**

1. Carilah aliran arus maksimal pada aliran jaringan (Network Flow) berikut!



2. Modelkan bentuk aliran jaringan di atas ke dalam bentuk matriks 6x6, sebagai berikut.

A = [

0	1	8	0	0	0
0	2	0	0	0	0
0	0	0	0	7	0
0	1	0	0	7	0
0	0	5	0	0	2
0	0	0	0	0	0

]

3. Buatlah program Julia untuk menyelesaikannya berdasarkan matriks yang telah dibuat.

```

julia> using JuMP
julia> using HiGHS
julia>
julia> A = [
0 1 8 0 0 0
0 2 0 0 0 0
0 0 0 0 7 0
0 1 0 0 7 0
0 0 5 0 0 2
0 0 0 0 0 0
]
6×6 Matrix{Int64}:
0 1 8 0 0 0
0 2 0 0 0 0
0 0 0 0 7 0
0 1 0 0 7 0
0 0 5 0 0 2
0 0 0 0 0 0
julia> n=size(A)[1]
6
julia> max_flow = Model(HiGHS.Optimizer)
A JuMP Model
└ solver: HiGHS
  objective_sense: FEASIBILITY_SENSE
  num_variables: 0
  num_constraints: 0
  Names registered in the model: none
  
```

4. Tahap pertama dalam membuat program ini adalah mengimpor paket

```
julia> using JuMP
julia> using HiGHS
julia>
```

5. Kemudian mendefinisikan graf dengan matriks

```
julia> A = [
    0 1 8 0 0 0
    0 2 0 0 0 0
    0 0 0 0 0 7
    0 1 0 0 7 0
    0 0 5 0 0 2
    0 0 0 0 0 0
]
6x6 Matrix{Int64}:
 0  1  8  0  0  0
 0  2  0  0  0  0
 0  0  0  0  0  7
 0  1  0  0  7  0
 0  0  5  0  0  2
 0  0  0  0  0  0
```

Matriks A mendefinisikan graf dengan biaya aliran antara node. Elemen  $A[i,j]$  menunjukkan biaya aliran dari aliran dari node  $i$  ke node  $j$ . Jika biayanya 0, berarti tidak ada aliran langsung antara node tersebut.

6. Memebuat jumlah node, bagian ini menghitung jumlah node dalam graf A dengan mengambil ukuran dimensi pertama dari matriks A, dalam contoh ini, n akan bernilai 6 karena matriks A berukuran 6x6.

```
julia> n=size(A)[1]
6
```

7. Kemudian kita dapat membuat model

```
Julia 1.9.3
julia> using JuMP
julia> using HiGHS
julia>
julia> A = [
    0 1 8 0 0 0
    0 2 0 0 0 0
    0 0 0 0 0 7
    0 1 0 0 7 0
    0 0 5 0 0 2
    0 0 0 0 0 0
]
6x6 Matrix{Int64}:
 0  1  8  0  0  0
 0  2  0  0  0  0
 0  0  0  0  0  7
 0  1  0  0  7  0
 0  0  5  0  0  2
 0  0  0  0  0  0

julia> n=size(A)[1]
6

julia> max_flow = Model(HiGHS.Optimizer)
A JuMP Model
+ solver: HiGHS
+ objective_sense: FEASIBILITY_SENSE
+ num_variables: 0
+ num_constraints: 0
+ Names registered in the model: none
```

```

julia> @variable(max_flow, f[1:n, 1:n] >= 0)
6x6 Matrix{VariableRef}:
f[1,1] f[1,2] f[1,3] f[1,4] f[1,5] f[1,6]
f[2,1] f[2,2] f[2,3] f[2,4] f[2,5] f[2,6]
f[3,1] f[3,2] f[3,3] f[3,4] f[3,5] f[3,6]
f[4,1] f[4,2] f[4,3] f[4,4] f[4,5] f[4,6]
f[5,1] f[5,2] f[5,3] f[5,4] f[5,5] f[5,6]
f[6,1] f[6,2] f[6,3] f[6,4] f[6,5] f[6,6]

julia>

julia> @constraint(max_flow, [i = 1:n, j = 1:n], f[i, j] <= A[i, j])
6x6 Matrix{ConstraintRef{Model, MathOptInterface.ConstraintIndex{MathOptInterface.ScalarAffineFunction{Float64}}, MathOptInterface.LessThan{Float64}}, ScalarShape}:
f[1,1] <= 0 f[1,2] <= 1 - f[1,5] <= 0 f[1,6] <= 0
f[2,1] <= 0 f[2,2] <= 2 - f[2,5] <= 0 f[2,6] <= 0
f[3,1] <= 0 f[3,2] <= 0 f[3,5] <= 0 f[3,6] <= 7
f[4,1] <= 0 f[4,2] <= 1 f[4,5] <= 7 f[4,6] <= 0
f[5,1] <= 0 f[5,2] <= 0 f[5,5] <= 0 f[5,6] <= 2
f[6,1] <= 0 f[6,2] <= 0 - f[6,5] <= 0 f[6,6] <= 0

julia>

julia> @constraint(max_flow, [i = 1:n; i != 1 && i != 6], sum(f[i, :]) == sum(f[:, i]))
JuMP.Containers.SparseAxisArray{ConstraintRef{Model, MathOptInterface.ConstraintIndex{MathOptInterface.ScalarAffineFunction{Float64}}, MathOptInterface.EqualTo{Float64}}, ScalarShape, 1, Tuple{Int64}} with 4 entries:
[2] = f[2,1] - f[1,2] - f[3,2] - f[4,2] - f[5,2] - f[6,2] + f[2,3] + f[2,4] + f[2,5] + f[2,6] == 0
[3] = f[3,1] + f[3,2] - f[1,3] - f[2,3] - f[4,3] - f[5,3] - f[6,3] + f[3,4] + f[3,5] + f[3,6] == 0
[4] = f[4,1] + f[4,2] + f[4,3] - f[1,4] - f[2,4] - f[3,4] - f[5,4] - f[6,4] + f[4,5] + f[4,6] == 0
[5] = f[5,1] + f[5,2] + f[5,3] + f[5,4] - f[1,5] - f[2,5] - f[3,5] - f[4,5] - f[6,5] + f[5,6] == 0

julia>

```

8. Mendefinisikan variable biner X untuk setiap arc dalam graf. Variable ini akan bernilai 1 jika arc tersebut termasuk dalam jalur dan variable akan bernilai 0 jika arc tersebut tidak termasuk dalam jalur.

```

julia> @variable(max_flow, f[1:n, 1:n] >= 0)
6x6 Matrix{VariableRef}:
f[1,1] f[1,2] f[1,3] f[1,4] f[1,5] f[1,6]
f[2,1] f[2,2] f[2,3] f[2,4] f[2,5] f[2,6]
f[3,1] f[3,2] f[3,3] f[3,4] f[3,5] f[3,6]
f[4,1] f[4,2] f[4,3] f[4,4] f[4,5] f[4,6]
f[5,1] f[5,2] f[5,3] f[5,4] f[5,5] f[5,6]
f[6,1] f[6,2] f[6,3] f[6,4] f[6,5] f[6,6]

julia>

julia> @constraint(max_flow, [i = 1:n, j = 1:n], f[i, j] <= A[i, j])
6x6 Matrix{ConstraintRef{Model, MathOptInterface.ConstraintIndex{MathOptInterface.ScalarAffineFunction{Float64}}, MathOptInterface.LessThan{Float64}}, ScalarShape}:
f[1,1] <= 0 f[1,2] <= 1 - f[1,5] <= 0 f[1,6] <= 0
f[2,1] <= 0 f[2,2] <= 2 - f[2,5] <= 0 f[2,6] <= 0
f[3,1] <= 0 f[3,2] <= 0 f[3,5] <= 0 f[3,6] <= 7
f[4,1] <= 0 f[4,2] <= 1 f[4,5] <= 7 f[4,6] <= 0
f[5,1] <= 0 f[5,2] <= 0 f[5,5] <= 0 f[5,6] <= 2
f[6,1] <= 0 f[6,2] <= 0 - f[6,5] <= 0 f[6,6] <= 0

julia>

julia> @constraint(max_flow, [i = 1:n; i != 1 && i != 6], sum(f[i, :]) == sum(f[:, i]))
JuMP.Containers.SparseAxisArray{ConstraintRef{Model, MathOptInterface.ConstraintIndex{MathOptInterface.ScalarAffineFunction{Float64}}, MathOptInterface.EqualTo{Float64}}, ScalarShape, 1, Tuple{Int64}} with 4 entries:
[2] = f[2,1] - f[1,2] - f[3,2] - f[4,2] - f[5,2] - f[6,2] + f[2,3] + f[2,4] + f[2,5] + f[2,6] == 0
[3] = f[3,1] + f[3,2] - f[1,3] - f[2,3] - f[4,3] - f[5,3] - f[6,3] + f[3,4] + f[3,5] + f[3,6] == 0
[4] = f[4,1] + f[4,2] + f[4,3] - f[1,4] - f[2,4] - f[3,4] - f[5,4] - f[6,4] + f[4,5] + f[4,6] == 0
[5] = f[5,1] + f[5,2] + f[5,3] + f[5,4] - f[1,5] - f[2,5] - f[3,5] - f[4,5] - f[6,5] + f[5,6] == 0

julia>

```

9. Constraint : Arc dengan biaya 0. Constraint ini memastikan bahwa arc dengan biaya nol tidak termasuk dalam jalur.

```

julia> @variable(max_flow, f[1:n, 1:n] >= 0)
6x6 Matrix{VariableRef}:
f[1,1] f[1,2] f[1,3] f[1,4] f[1,5] f[1,6]
f[2,1] f[2,2] f[2,3] f[2,4] f[2,5] f[2,6]
f[3,1] f[3,2] f[3,3] f[3,4] f[3,5] f[3,6]
f[4,1] f[4,2] f[4,3] f[4,4] f[4,5] f[4,6]
f[5,1] f[5,2] f[5,3] f[5,4] f[5,5] f[5,6]
f[6,1] f[6,2] f[6,3] f[6,4] f[6,5] f[6,6]

julia>

julia> @constraint(max_flow, [i = 1:n, j = 1:n], f[i, j] <= A[i, j])
6x6 Matrix{ConstraintRef{Model, MathOptInterface.ConstraintIndex{MathOptInterface.ScalarAffineFunction{Float64}}, MathOptInterface.LessThan{Float64}}, ScalarShape}:
f[1,1] <= 0 f[1,2] <= 1 - f[1,5] <= 0 f[1,6] <= 0
f[2,1] <= 0 f[2,2] <= 2 - f[2,5] <= 0 f[2,6] <= 0
f[3,1] <= 0 f[3,2] <= 0 f[3,5] <= 0 f[3,6] <= 7
f[4,1] <= 0 f[4,2] <= 1 f[4,5] <= 7 f[4,6] <= 0
f[5,1] <= 0 f[5,2] <= 0 f[5,5] <= 0 f[5,6] <= 2
f[6,1] <= 0 f[6,2] <= 0 - f[6,5] <= 0 f[6,6] <= 0

julia>

julia> @constraint(max_flow, [i = 1:n; i != 1 && i != 6], sum(f[i, :]) == sum(f[:, i]))
JuMP.Containers.SparseAxisArray{ConstraintRef{Model, MathOptInterface.ConstraintIndex{MathOptInterface.ScalarAffineFunction{Float64}}, MathOptInterface.EqualTo{Float64}}, ScalarShape, 1, Tuple{Int64}} with 4 entries:
[2] = f[2,1] - f[1,2] - f[3,2] - f[4,2] - f[5,2] - f[6,2] + f[2,3] + f[2,4] + f[2,5] + f[2,6] == 0
[3] = f[3,1] + f[3,2] - f[1,3] - f[2,3] - f[4,3] - f[5,3] - f[6,3] + f[3,4] + f[3,5] + f[3,6] == 0
[4] = f[4,1] + f[4,2] + f[4,3] - f[1,4] - f[2,4] - f[3,4] - f[5,4] - f[6,4] + f[4,5] + f[4,6] == 0
[5] = f[5,1] + f[5,2] + f[5,3] + f[5,4] - f[1,5] - f[2,5] - f[3,5] - f[4,5] - f[6,5] + f[5,6] == 0

julia>

```

10. Tujuan dari model ini adalah untuk meminimalkan total biaya aliran. Ini dilakukan dengan menjumlahkan hasil perkalian elemen-elemen matriks biaya A dengan variabel x.

```
Julia 1.9.3
julia>
julia> @objective(max_flow, Max, sum(f[1, :]))
f[1,1] + f[1,2] + f[1,3] + f[1,4] + f[1,5] + f[1,6]
julia>
julia> optimize!(max_flow)
```

11. Optimasi dan Hasil. Bagian ini menjalankan optimasi model dan menampilkan hasilnya. `objective_value(model)` memberikan nilai objektif dari solusi optimal, dan `value.(x)` memberikan nilai dari variabel x dalam solusi optimal.

```
Julia 1.9.3
julia>
julia> @objective(max_flow, Max, sum(f[1, :]))
f[1,1] + f[1,2] + f[1,3] + f[1,4] + f[1,5] + f[1,6]
julia>
julia> optimize!(max_flow)
Running HiGHS 1.7.2 (git hash: 5ce7a2753): Copyright (c) 2024 HiGHS under MIT licence terms
Coefficient ranges:
  Matrix [1e+00, 1e+00]
  Cost    [1e+00, 1e+00]
  Bound   [0e+00, 0e+00]
  RHS     [1e+00, 0e+00]
Presolving model
4 rows, 6 cols, 9 nonzeros 0s
0 rows, 0 cols, 0 nonzeros 0s
Presolve : Reductions: rows 0(-40); columns 0(-36); elements 0(-76) - Reduced to empty
Solving the original LP from the solution after postsolve
Model status   : Optimal
Objective value : 7.000000000000e+00
HiGHS run time  : 0.01
julia>
julia> @assert is_solved_and_feasible(max_flow)
julia>
julia> objective_value(max_flow)
7.0
julia> ^C
```