

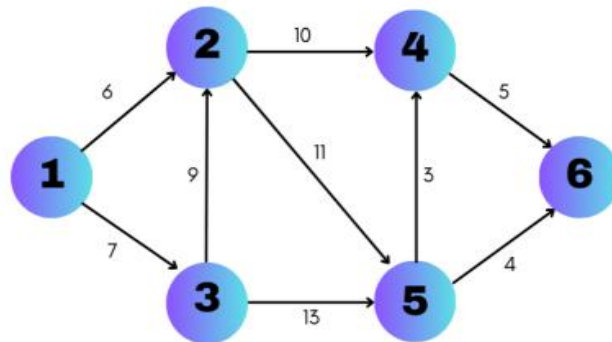
TUGAS 1
OPTIMISASI
NETWORK FLOW PROBLEM



Oleh:
Muhammad Trisaputra

PROGRAM STUDI TEKNIK ELEKTRO
FAKULTAS TEKNIK
UNIVERSITAS BENGKULU
2024

1. Network Flow Problem



Gambar 1 Network Flow

Aliran $A = [$

```
0 6 7 0 0 0
0 0 0 10 11 0
0 9 0 0 13 0
0 0 0 0 0 5
0 0 0 3 0 4
0 0 0 0 0 0
]
```

2. Menyelesaikan Masalah Menggunakan Julia

```
Select Julia 1.10.5
Documentation: https://docs.julialang.org
Type "?" for help, "]" for pkg help.
Version 1.10.5 (2024-08-27)
Official https://julialang.org/ release

julia> using JuMP
julia> import HiGHS

julia> x = [
    0 6 7 0 0 0
    0 0 0 10 11 0
    0 9 0 0 13 0
    0 0 0 0 0 5
    0 0 0 3 0 4
    0 0 0 0 0 0
]

6x6 Matrix{Int64}:
0 6 7 0 0 0
0 0 0 10 11 0
0 9 0 0 13 0
0 0 0 0 0 5
0 0 0 3 0 4
0 0 0 0 0 0

julia> n = size(x)[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]
```

```

Select Julia 1.10.5
julia> @constraint(max_flow, [i = 1:n, j = 1:n], f[i, j] <= x[i, j])
6x6 Matrix{ConstraintRef{Model, MathOptInterface.ConstraintIndex{MathOptInterface.ScalarAffineFunction{Float64}, MathOptInterface.LessThan{Float64}}, ScalarShape}}:
 f[1,1] <= 0 f[1,2] <= 0 f[1,3] <= 7 f[1,4] <= 0 f[1,5] <= 0 f[1,6] <= 0
 f[2,1] <= 0 f[2,2] <= 0 f[2,3] <= 0 f[2,4] <= 10 f[2,5] <= 11 f[2,6] <= 0
 f[3,1] <= 0 f[3,2] <= 9 f[3,3] <= 0 f[3,4] <= 0 f[3,5] <= 13 f[3,6] <= 0
 f[4,1] <= 0 f[4,2] <= 0 f[4,3] <= 0 f[4,4] <= 0 f[4,5] <= 0 f[4,6] <= 5
 f[5,1] <= 0 f[5,2] <= 0 f[5,3] <= 0 f[5,4] <= 3 f[5,5] <= 0 f[5,6] <= 4
 f[6,1] <= 0 f[6,2] <= 0 f[6,3] <= 0 f[6,4] <= 0 f[6,5] <= 0 f[6,6] <= 0

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> @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> 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     [3e+00, 1e+01]
Presolving model
4 rows, 9 cols, 14 nonzeros 0s
4 rows, 9 cols, 14 nonzeros 0s
Presolve : Reductions: rows 4(-36); columns 9(-27); elements 14(-62)
Solving the presolved LP
Using EKK dual simplex solver - serial
Iteration      Objective      Infeasibilities num(sum)
0               0 0.000000000e+00 Phi: 0(0) 0s
5            -9.000000000e+00 Pr: 0(0) 0s
Solving the original LP from the solution after postsolve
Model status   : Optimal
Simplex iterations: 5
Objective value : 9.000000000e+00
HiGHS run time  : 0.07

julia> @assert is_solved_and_feasible(max_flow)

julia> objective_value(max_flow)
9.0

```

1. using JuMP dan import HiGHS

Berfungsi untuk menggunakan library JuMP dan HiGHS

2. Pada Command Line

$$\mathbf{x} = \begin{bmatrix} 0 & 6 & 7 & 0 & 0 & 0 \\ 0 & 0 & 0 & 10 & 11 & 0 \\ 0 & 9 & 0 & 0 & 13 & 0 \\ 0 & 0 & 0 & 0 & 0 & 5 \\ 0 & 0 & 0 & 3 & 0 & 4 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Digunakan untuk mendeskripsikan Aliran jaringan dalam bentuk matriks A dengan ukuran 6x6

3. Pada baris **n = size(x)[1]**

Untuk mendeklarasikan variabel n sebagai bentuk ukuran dari matriks x

4. Pada baris **max_flow = Model(HiGHS.Optimizer)**

Baris yang digunakan mendeklarasikan sebuah model variabel max_flow menggunakan paket HiGHS

5. Pada baris kode **@variable(max_flow, f[1:n, 1:n] >=0)**

mendefinisikan variabel keputusan f, dalam model max_flow. Setiap elemen dari matriks ini merepresentasikan aliran antara dua node dalam sebuah jaringan.

6. Pada baris kode **@constraint(max_flow, [i = 1:n, j = 1:n], f[i, j] <= x[i, j])**

mendefinisikan setiap edge pada model max_flow di network flow dari satu untuk setiap pasangan (i, j) dari node dalam jaringan

7. **@constraint(max_flow, [i = 1:n; i != 1 && i != 6], sum(f[i, :]) == sum(f[:, i]))** mendeklarasikan bahwa jumlah aliran yang keluar dari node i harus sama dengan jumlah aliran yang masuk ke node i.
8. **@objective(max_flow, Max, sum(f[1, :]))** memiliki fungsi untuk memaksimalkan total flow dari node sumber ke node tujuan
9. **optimize!(max_flow)** model yang telah dibuat dilakukan pengoptimalan dengan perintah 'optimize'.
10. **@assert is_solved_and_feasible(max_flow)** berfungsi untuk memastikan model telah terpecahkan.
11. Pada baris **objective_value(max_flow)** menampilkan nilai terbaik dari fungsi tujuan yang dicapai, yaitu 14

Link Github : <https://github.com/muhammadtrisaputra/Optimization>

Link Youtube : <https://youtu.be/E84phTiZM84>