**EVALUASI MANDIRI OPTIMISASI**

**MATAKULIAH OPTIMISASI   
MENULIS MASALAH OPTIMISASI DENGAN JuMP**



Disusun oleh: **RENDI YUDHA FRENDIKA  
G1D021002**

Dosen Pengampu: **Ir. Novalio Daratha S.T., M.Sc., Ph.D.**

**PROGRAM STUDI TEKNIK ELEKTRO**

**FAKULTAS TEKNIK**

**UNIVERSITAS BENGKULU  
2024**

Nama : Rendi Yudha Frendika   
NPM : G1D021002  
Matakuliah : Optimisasi

**Menulis Masalah Optimisasi Dengan JuMP**

1. *Example of network flow*

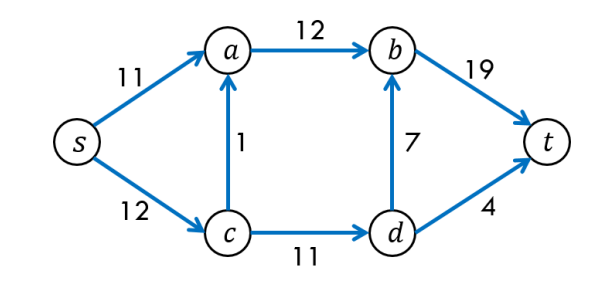


Figure 1. Network Flow

Gambar ini menggambarkan sebuah network flow yang umum digunakan dalam algoritma terkait teori graf, seperti algoritma Ford-Fulkerson untuk menemukan aliran maksimum dalam sebuah jaringan.

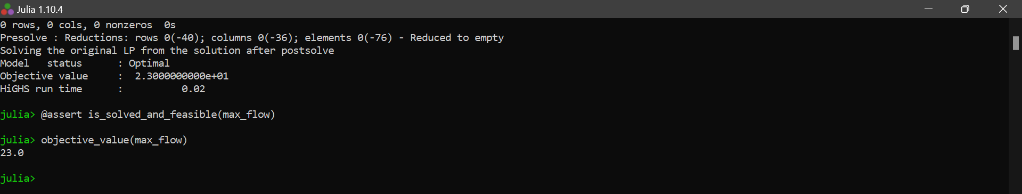
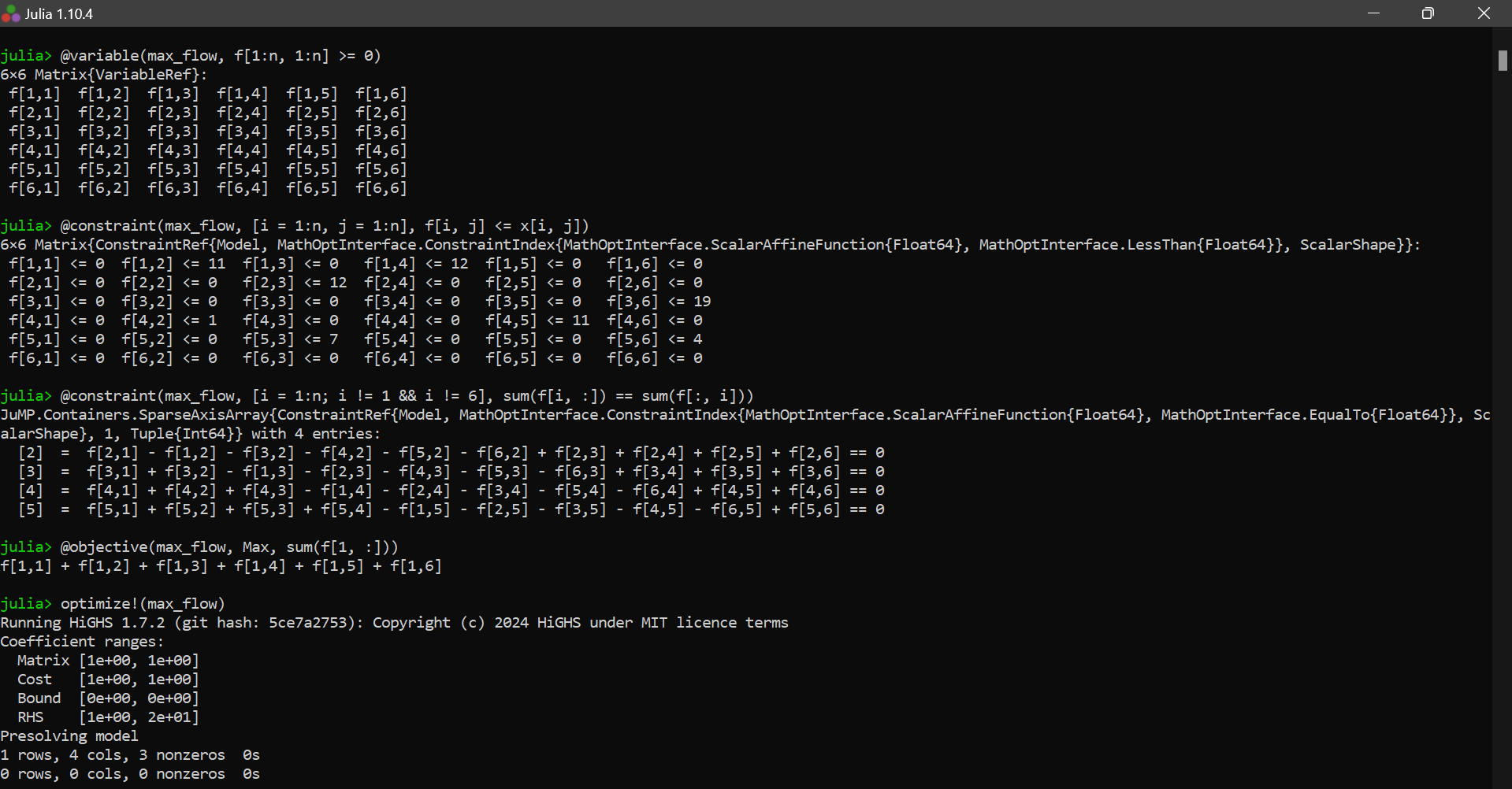
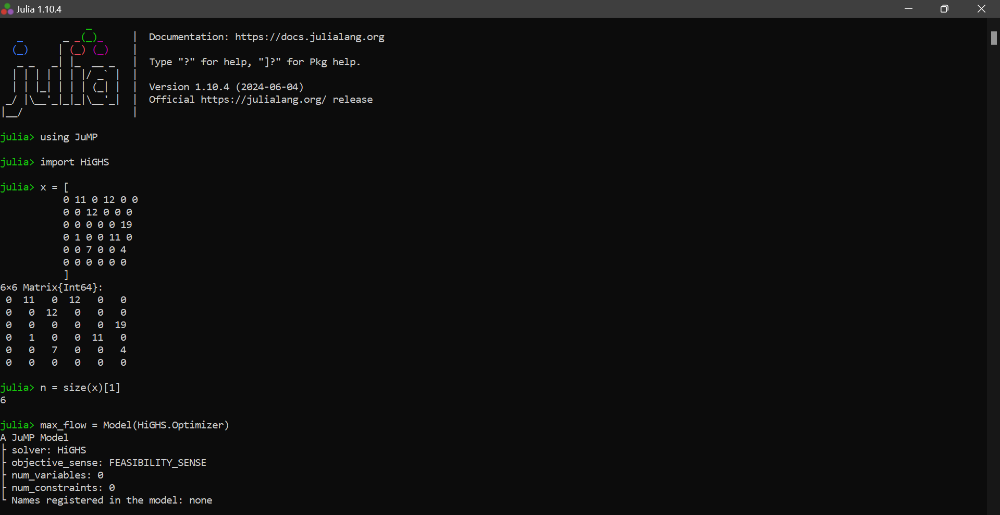
Pada network flow diatas dapat dideskripsikan sebagai berikut dari jaringan tersebut:

* Simpul (Node/Vertex): s, a, b, c, d, t

Dengan s sebagai simpul sumber yang merupakan sebagai simpul awal dalam network flow dan t sebagai simpul tujuancyang merupakan titik akhir dari network flow berhenti.

* **Tepi (Edges):** Menunjukkan arah dan kapasitas aliran antara simpul-simpul.
  + s → a (Kapasitas: 11)
  + s → c (Kapasitas: 12)
  + a → b (Kapasitas: 12)
  + a → c (Kapasitas: 1)
  + c → d (Kapasitas: 11)
  + b → d (Kapasitas: 7)
  + b → t (Kapasitas: 19)
  + d → t (Kapasitas: 4)

Dari network flow diatas langkah yang digunakan adalah untuk menentukan max\_flow. Max\_flow adalah optimisasi dari seberapa besar aliran maksimum yang dapat ditransfer dari s ke t melalui jaringan, dengan memperhatikan kapasitas maksimum dari setiap tepi (*edge*) yang menghubungkan simpul-simpul di dalam jaringan. Untuk menyelesaikan optimisasi dari network flow diatas maka digunakanlah pemrograman dengan bahasa Julia.



Untuk menyelesaikan masalah max\_flow dengan menggunakan bahasa pemrograman Julia.

1. Baris kode **using JuMP** dengan **Import HiGHS** Untuk dapat mendefinisikan variabel keputusan, fungsi tujuan yang akan digunakan, di kasus ini adalah network flow
2. Langkah kedua adalah dengan memodelkan network flow tersebut dengan menggunakan matriks, yaitu untuk merepresentasikan jaringan tersebut dalam bentuk matriks kapasitas. Matriks tersebut menunjukkan kapasitas flow antara setiap pasangan simpul dalam jaringan. Berdasarkan jumlah node pada network flow maka matriksnya dapat membentuk matriks ordo 6x6 yang merepresentasikan kapasitas antar node sebagai berikut

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | s | a | b | c | d | e |
| s | 0 | 11 | 0 | 12 | 0 | 0 |
| a | 0 | 0 | 12 | 0 | 0 | 0 |
| b | 0 | 0 | 0 | 0 | 0 | 19 |
| c | 0 | 1 | 0 | 0 | 11 | 0 |
| d | 0 | 0 | 7 | 0 | 0 | 4 |
| t | 0 | 0 | 0 | 0 | 0 | 0 |

Dari matriks tersebut kemudian dimasukkan ke Julia untuk dieksekusi sehingga Julia mendeklarasikan bahwa user memasukkan nilai matriks.

1. Baris kode **n = size(x)[1]** adalah mendeklarasikan variabel n sebagai bentuk ukuran dari matriks x dengan ordo 6x6
2. Baris kode mendeklarasikan sebuah model variabel max\_flow solver optimasi program
3. Baris kode **@variable(max\_flow, f[1:n, 1:n] >= 0)** mendefinisikan variabel keputusan f[i, j] dalam model max\_flow, yang merupakan matriks berukuran n x n. Setiap elemen dari matriks ini merepresentasikan aliran antara dua node dalam sebuah jaringan. Kondisi >= 0 memastikan bahwa aliran ini adalah non-negatif, sesuai dengan aturan dalam network flow problem
4. baris kode **@constraint(max\_flow, [i = 1:n, j = 1:n], f[i, j] <= G[i, j])** mendefinisikan setiap edge pada model max\_flow di network flow dari satu untuk setiap pasangan (i, j) dari node dalam jaringan.
5. Baris kode **@constraint(max\_flow, [i = 1:n; i != 1 && i != 6], sum(f[i, :]) == sum(f[:, i]))** ini mendeklarasikan bahwa untuk setiap node i yang tidak sama dengan 1 hingga 6, jumlah aliran yang keluar dari node i harus sama dengan jumlah aliran yang masuk ke node i.
6. Baris kode @**objective(max\_flow, Max, sum(f[1, :]))** ini mendeklarasikan fungsi tujuan dalam model max\_flow untuk memaksimalkan total flow dari node sumber (node s) ke semua node lainnya.
7. Baris kode **optimize!(max\_flow)** ini mendeklarasikan perintah untuk menjalankan solver dan menyelesaikan model optimasi
8. Baris kode **objective\_value(max\_flow)** memberikan nilai terbaik dari fungsi tujuan yang dicapai setelah optimasi model max\_flow. Dari pemrograman diatas didapatlah nilai maksimal terbaik setelah optimasi max\_flow adalah 23.0