PENERAPAN ALGORITMA *CLONING-BASED* DAN ALGORITMA *EDMONDS-KARP* DALAM PENYELESAIAN *MAXIMUM FLOW PROBLEM*

SKRIPSI

Oleh

Erni Rahayu
NIM 121810101019

JURUSAN MATEMATIKA
FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM
UNIVERSITAS JEMBER
2016

PENERAPAN ALGORITMA *CLONING-BASED* DAN ALGORITMA *EDMONDS-KARP* DALAM PENYELESAIAN *MAXIMUM FLOW PROBLEM*

SKRIPSI

diajukan guna melengkapi tugas akhir dan memenuhi salah satu syarat
untuk menyelesaikan Program Studi Matematika (S1)
dan mencapai gelar Sarana Sains

Oleh

Erni Rahayu
NIM 121810101019

JURUSAN MATEMATIKA
FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM
UNIVERSITAS JEMBER
2016

PERSEMBAHAN

Dengan menyebut nama Allah S.W.T dan sholawat serta salam kepada Nabi Muhammad S.A.W, skripsi ini saya persembahkan untuk:

1. Kedua orang tuaku yang tersayang dan terkasih, Ayahanda Sukarjiono dan Ibunda Iis Widiawati, serta kedua Adikku yang tersayang Chintya Puji Lestari dan Diabela Anastasya Putri, yang senantiasa memberikan doa, perhatian, kasih sayang, semangat dan inspirasi serta dorongan untuk meraih cita-cita;
2. Ahmad Kamsyakawuni, S.Si, M.Kom. dan M. Ziaul Arif, S.Si., M.Sc. yang dengan sabar dan ikhlas telah membimbing serta memberikan bantuan sehingga penulisan skripsi ini dapat terselesaikan dengan baik;
3. Almamater tercinta Jurusan Matematika Fakultas Matematika dan Ilmu Pengetahuan Alam Universitas Jember, SMK Al-Azhar, SMP Al-Azhar, MI Salafiyah Tugung, dan TK Aisiyah;
4. Widi Heru Prasetyo yang selalu setia mendampingi, selalu mendoakan dan senantiasa memberi dukungan, perhatian, kasih sayang, dan semangat dalam penyelesaian skripsi ini;

MOTTO

“Sesungguhnya bersama kesulitan ada kemudahan. Maka apabila engkau telah selesai (dari sesuatu urusan), tetaplah bekerja keras (untuk urusan yang lain). Dan hanya kepada Tuhanmulah engkau berharap.”
(terjemahan Surat *Al-Insyirah* Ayat 6-8)[*)]

"Pendidikan merupakan senjata paling ampuh yang bisa kamu gunakan untuk merubah dunia."
(Nelson Mandela)[**)]

“Sesungguhnya Allah tidak merubah keadaan suatu kaum sehingga mereka merubah keadaan yang ada pada diri mereka sendiri.”
(terjemahan Surat *Ar-Rad* Ayat 11)[***)]

[*)] Departemen Agama Republik Indonesia. 2004. *Al-Qur’an dan Terjemahannya*. Bandung: CV Penerbit Diponegoro.

[***)] Departemen Agama Republik Indonesia. 2004. *Al-Qur’an dan Terjemahannya*. Bandung: CV Penerbit Diponegoro.

PERNYATAAN

Saya yang bertanda tangan dibawah ini :

Nama : Erni Rahayu

NIM : 121810101019

Menyatakan dengan sesungguhnya bahwa skripsi yang berjudul "Penerapan Algoritma *Cloning-Based* dan Algoritma *Edmonds-Karp* dalam Penyelesaian *Maximum Flow Problem*" adalah benar-benar hasil karya sendiri, kecuali kutipan yang sudah saya sebutkan sumbernya, belum pernah diajukan pada institusi manapun, dan bukan karya jiplakan. Saya bertanggung jawab atas keabsahan dan kebenaran isinya sesuai dengan sikap ilmiah yang harus dijunjung tinggi.

Demikian pernyataan ini saya buat dengan sebenarnya, tanpa ada tekanan dan paksaan dari pihak manapun serta bersedia mendapat sanksi akademik jika ternyata dikemudian hari pernyataan ini tidak benar.

Jember, 24 Juli 2016

Yang menyatakan,

Erni Rahayu

NIM 121810101019

SKRIPSI

PENERAPAN ALGORITMA *CLONING-BASED* DAN ALGORITMA *EDMONDS-KARP* DALAM PENYELESAIAN *MAXIMUM FLOW PROBLEM*

Oleh

Erni Rahayu
NIM 121810101019

Pembimbing

Dosen Pembimbing Utama : Ahmad Kamsyakawuni, S.Si, M.Kom
Dosen Pembimbing Anggota : M. Ziaul Arif, S.Si., M.Sc

PENGESAHAN

Skripsi berjudul "Penerapan Algoritma *Cloning-Based* Dan Algoritma *Edmonds-Karp* Dalam Penyelesaian *Maximum Flow Problem*" telah diuji dan disahkan pada:

hari, tanggal :

tempat : Fakultas Matematika dan Ilmu Pengetahuan Alam Universitas Jember

Tim Penguji:

Ketua,	Sekretaris,
Ahmad Kamsyakawuni, S.Si., M.Kom NIP. 19721129 199802 1 001	M. Ziaul Arif, S.Si., M.Sc NIP. 19850111 200812 1 002
Anggota I,	Anggota II,
Kusbudiono, S.Si., M.Si NIP. 19770430 200501 1 001	Dian Anggraeni, S.Si., M.Si NIP. 19820216 200604 2 002

Mengesahkan
Dekan,

Drs. Sujito, Ph.D.
NIP.19610204 198711 1 001

RINGKASAN

Penerapan Algoritma *Cloning-Based* dan Algoritma *Edmonds-Karp* dalam Penyelesaian *Maximum Flow Problem*; Erni Rahayu, 121810101019; 2016; 68 halaman; Jurusan Matematika Fakultas Matematika dan Ilmu Pengetahuan Alam Universitas Jember.

Network Flow adalah sebuah graf berarah yang tiap sisinya memiliki kapasitas dan terdapat arus yang mengalir antara dua simpul yang mengapit sisi tersebut. Salah satu aplikasi *network flow* adalah *maximum flow*. *Maximum flow* merupakan sebuah model yang dapat digunakan untuk mengetahui nilai *maximum* seluruh arus didalam sebuah sistem jaringan. *Maximum flow* memiliki tujuan untuk memaksimalkan jumlah aliran yang mengalir pada sebuah *network* yang hanya memiliki satu *source* (sumber) dan satu *sink* (tujuan).

Penelitian ini menggunakan data sekunder berupa data jaringan distribusi listrik wilayah distribusi kebasen 11 kota Tegal. Jaringan tersebut terdiri dari 46 simpul dan 64 sisi. Simpul 1 merupakan gardu induk dan merupakan *source* (simpul awal), sedangkan simpul 2 sampai simpul 46 diasumsikan sebagai tiang listrik yang menghubungkan kabel satu dengan yang lain, dan sisi diasumsikan sebagai kabel yang menghubungkan antar tiang listrik. Simpul 46 merupakan *sink* (simpul tujuan). Serta terdapat kapasitas masing-masing kabel yang digunakan untuk membatasi jumlah aliran yang melewati kabel tersebut. Jumlah aliran tidak boleh melebihi kapasitasnya. Hasil penelitian ini berupa *maximum flow*, *running time*, dan lintasan *maximum flow*.

Penerapan algoritma *cloning based* dan algoritma *edmonds karp* pada data jaringan distribusi listrik wilayah distribusi kebasen 11 kota Tegal dengan melakukan sepuluh kali percobaan, *maximum flow* yang dihasilkan algoritma *edmonds karp* lebih besar dari pada algoritma *cloning based*. Karena pada algoritma *cloning based* tidak ada aturan untuk memulai lintasan awal. Sedangkan pada algoritma *edmonds karp* lintasan awal merupakan lintasan terpendek.

Sehingga dapat dikatakan bahwa algoritma *edmonds karp* lebih baik dari pada algoritma *cloning based*. Sedangkan *running time* yang diperoleh algoritma *cloning based* lebih kecil daripada algoritma *edmonds karp*. Jika dilihat dari *running time* yang diperoleh, algoritma *cloning based* lebih baik daripada algoritma *edmonds karp*

PRAKATA

Puji syukur penulis panjatkan kehadirat Allah SWT, karena berkat Rahmat dan karunia-Nya penulis dapat menyelesaikan penyusunan skripsi yang berjudul "Penerapan Algoritma *Cloning-Based* dan Algoritma *Edmonds-Karp* dalam Penyelesaian *Maximum Flow Problem*". Shalawat beserta salam semoga tetap tercurahkan kepada Nabi Muhammad SAW.

Skripsi ini diajukan untuk memenuhi salah satu syarat menyelesaikan pendidikan strata satu (S1) pada Jurusan Matematika Fakultas Matematika dan Ilmu Pengetahuan Alam Universitas Jember. Penulisan skripsi ini tidak lepas dari hambatan dan kesulitan, namun berkat bimbingan, bantuan, nasihat, saran, dan kerjasama dari berbagai pihak, akhirnya segala hambatan tersebut dapat diatasi dengan baik. Pada kesempatan ini penulis dengan tulus mengucapkan terimakasih kepada:

1. Ahmad Kamsyakawuni, S. Si., M. Kom., selaku Dosen Pembimbing Utama, M. Ziaul Arif, S.Si., M.Sc., Selaku Pembimbing Anggota yang telah meluangkan waktu, pikiran, dan perhatian dalam penulisan skripsi ini;
2. Dian Anggraeni, S.Si., M.Si., selaku Dosen Pembimbing Akademik yang telah membimbing selama penulis menjadi mahasiswa;
3. Kusbudiono, S.Si., M.Si., selaku Dosen Penguji I dan Dian Anggraeni, S.Si., M.Si., selaku Dosen Penguji II yang telah meluangkan waktu dan memberikan kritik serta saran demi kesempurnaan skipsi ini;
4. semua guru dan dosen sejak taman kanak-kanak hingga perguruan tinggi yang dengan sabar telah mendidik dan memberi banyak ilmu;
5. seluruh sahabatku dalam keluarga besar Matematika Angkatan 2012 BATHICS'12 dan Keluarga besar Kos Bangka 3 No. 20 yang selalu membantu dan memberikan semangat.
6. sahabat-sahabatku dalam UKM Kependudukan Universitas Jember dan UKM GANNAZ Universitas Jember;

7. seluruh karyawan dilingkungan Fakultas Matematika dan Ilmu Pengetahuan Alam Universitas Jember.
8. pihak-pihak lain yang tidak dapat penulis sebutkan satu persatu yang telah membantu hingga terselesaikannya skripsi ini.

Semoga segala bantuan yang telah diberikan senantiasa mendapat balasan dan Ridho Allah SWT. Penulisan skripsi ini tidak lepas dari kekurangan, baik aspek kualitas maupun aspek kuantitas dari materi yang disajikan dan diteliti, serta masih jauh dari kata sempurna. Sehingga penulis membutuhkan banyak kritik dan saran yang bersifat membangun untuk kemajuan pendidikan di masa yang akan datang.

Jember, 24 Juli 2016 Erni Rahayu

DAFTAR ISI

Halaman

DAFTAR TABEL

Halaman

DAFTAR GAMBAR

Halaman

DAFTAR LAMPIRAN

Halaman

BAB 1. PENDAHULUAN

1.1 Latar Belakang

Kehidupan manusia sekarang tidak dapat dipisahkan dengan permasalahan jaringan. Jaringan yang sering dijumpai dalam kehidupan sehari-hari adalah jaringan pendistribusian barang, jaringan aliran air, jaringan aliran listrik, jaringan telepon, dan jaringan komputer. Manusia terus mengembangkan metode untuk memudahkan dalam menyelesaikan permasalahan jaringan, contohnya mencari rute terpendek, menyusun jadwal agar waktu yang digunakan lebih optimal, dan mencari lintasan agar pendistribusian barang lebih maksimal tetapi tidak membutuhkan banyak biaya. Permasalahan-permasalahan tersebut biasa disebut dengan *network flow*. Aplikasi *Network flow* diantaranya adalah *maximum flow*, *shortest path*, *minimum spanning tree,* dan *minimum cost flow*.

Salah satu masalah *network flow* yang sering terjadi pada kehidupan sehari-hari adalah mencari *maximum flow*. *Maximum flow* merupakan permasalahan optimasi yang memiliki tujuan untuk memaksimalkan jumlah aliran yang mengalir pada sebuah *network* yang hanya memiliki satu *source* (sumber) dan satu *sink* (tujuan). *Maximum flow* sering diaplikasikan pada jaringan aliran air dan jaringan aliran listrik. Pada jaringan aliran air, jika air yang mengalir pada pipa kurang maksimal, maka terdapat pipa yang kurang bermanfaat. Jika air yang mengalir melebihi batas maksimum, maka pipa bisa pecah sehingga menyebabkan kebocoran. Sedangkan pada jaringan aliran listrik, jika listik yang mengalir pada kawat kurang maksimal, maka terdapat kawat yang kurang bermanfaat. Jika listrik yang mengalir melebihi batas maksimum, dapat menimbulkan arus pendek dan dapat menyebabkan kebakaran. Oleh sebab itu aliran maksimal sebuah jaringan perlu ditentukan terlebih dahulu sebelum jaringan tersebut dialiri.

Banyak algoritma yang telah dikembangkan untuk penyelesaian *maximum flow*, misalnya algoritma *djikstra,* algoritma *ford-fulkerson*, dan algoritma

prefllow. Merdekawaty (2011) telah membahas masalah *maximum flow* dengan menggunakan algoritma *ford-fulkerson*. Sedangkan pada artikel Widodo dkk (2012) telah dibahas perbandingan algoritma *cloning-based* dengan algoritma *djikstra*, dari penelitian tersebut dapat disimpulkan algoritma *cloning-based* memiliki solusi yang lebih baik dan lebih efisien daripada algoritma djikstra.

Algoritma *cloning-based* merupakan algoritma yang terinspirasi dari *cloning-selection* yang terjadi pada sistem kekebalan tubuh manusia yang terdiri dari antigen dan patogen. Algoritma ini merupakan bagian dari *bio-inspired algorithm* yaitu metode yang terinspirasi dari alam. Sedangkan algoritma *edmonds-karp* merupakan implementasi dari metode *ford-fulkerson*. Dalam algoritma *edmonds-karp*, lintasan penambah yang dipilih merupakan lintasan penambah terpendek yaitu lintasan dengan jumlah busur minimum. Hal inilah yang membuat peneliti mempertimbangkan untuk menerapkan algoritma *cloning-based* dengan algoritma *edmonds-karp* dalam penyelesaian *maximum flow problem* untuk mencari aliran yang maksimal.

1.2 Rumusan Masalah

Rumusan masalah yang akan dibahas pada penelitian ini jika ditinjau dari latar belakang diatas adalah sebagai berikut:

a. Bagaimana menyelesaikan *maximum flow problem* dengan menggunakan algoritma *cloning-based* dan algoritma *edmonds-karp*.
b. Bagaimana perbandingan hasil dari algoritma *cloning-based* dan algoritma *edmonds-karp*.

1.3 Tujuan Penelitian

Berdasarkan latar belakang dan rumusan masalah diatas tujuan yang ingin dicapai dalam penelitian ini adalah :

a. Mendapatkan penyelesaian *maximum flow problem* dengan menggunakan algoritma *cloning-based* dan algoritma *edmonds-karp.*

b. Mengetahui algoritma terbaik dari hasil perbandingan penyelesaian *maximum flow problem* dengan menggunakan algoritma *cloning-based* dan algoritma *edmonds-karp.*

1.4 Manfaat Penelitian

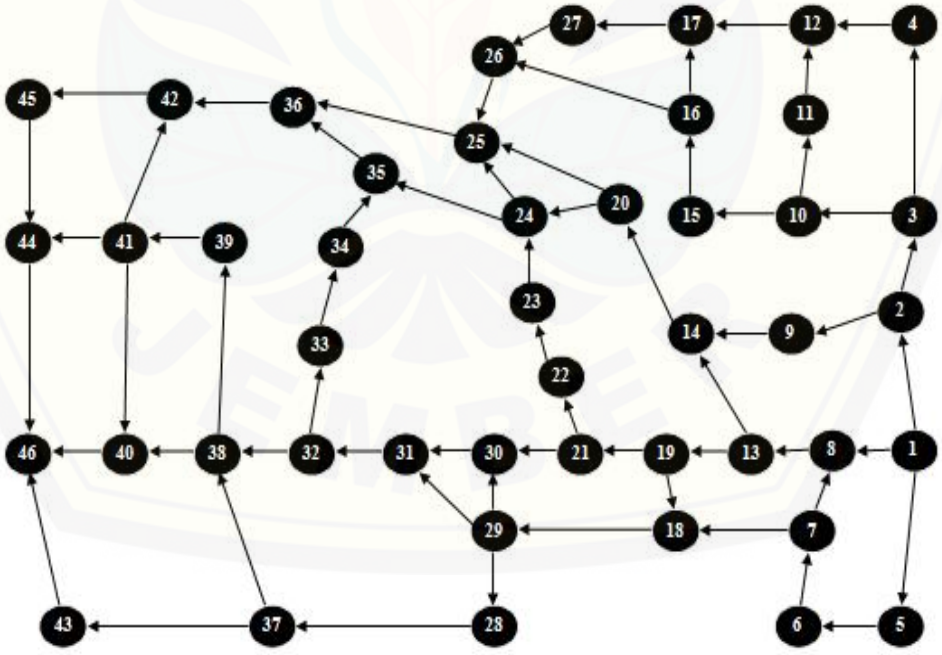
Manfaat yang diperoleh dari penelitian ini adalah sebagai berikut:

a. memperoleh solusi optimal yang berupa jumlah *maximum flow*;
b. menambah wawasan tentang algoritma *cloning-based* dan algoritma *edmonds-karp* dalam penyelesaian *maximum flow problem.*

BAB 2. TINJAUAN PUSTAKA

2.1 Teori Graf

Teori graf merupakan pokok bahasan yang memiliki banyak terapan. Graf digunakan untuk merepresentasikan objek – objek diskrit dan hubungan antara objek – objek tersebut. Representasi visual dari graf adalah dengan menyatakan objek sebagai bulatan atau titik. Sedangkan hubungan antara objek dinyatakan dengan garis. Sebagai contoh, Gambar 2.1 adalah sebuah peta jaringan distribusi listrik wilayah distribusi kebasen 11 kota Tegal. Sesungguhnya peta tersebut adalah sebuah graf, yang dalam hal ini tiang listrik dinyatakan sebagai bulatan sedangkan kabel yang menghubungkan antar tiang listrik dinyatakan sebagai garis (Munir, 2010).



Gambar 2.1 jaringan distribusi listrik wilayah distribusi kebasen 11 kota Tegal (Farizal, 2013)

2.1.1 Definisi Graf

Sebuah Graf *G* didefinisikan sebagai pasangan himpunan (*V, E)*, ditulis dengan notasi *G* =(*V, E*), yang dalam hal ini *V* adalah himpunan tidak kosong dari simpul – simpul (*vertecs* atau *node*) dan *E* adalah himpunan sisi (*edges* atau *arcs*) yang menghubungkan sepasang simpul. Sebuah graf dimungkinkan tidak mempunyai sisi sama sekali, tetapi simpulnya harus ada. Simpul pada graf dapat dinomori dengan huruf, seperti *a, b, c, …, V, Z* atau dengan bilangan asli 1, 2, 3, atau gabungan keduanya. Sedangkan sisi yang menghubungkan simpul *u* dengan simpul *v* dinyatakan dengan pasangan (*u, v*) atau dinyatakan dengan lambang e_1, e_2, e_3, …, e_n. Dengan kata lain, jika *e* adalah sisi yang menghubungkan simpul *u* dengan simpul *v*, maka *e* dapat ditulis sebagai *e* = (*u, v*)

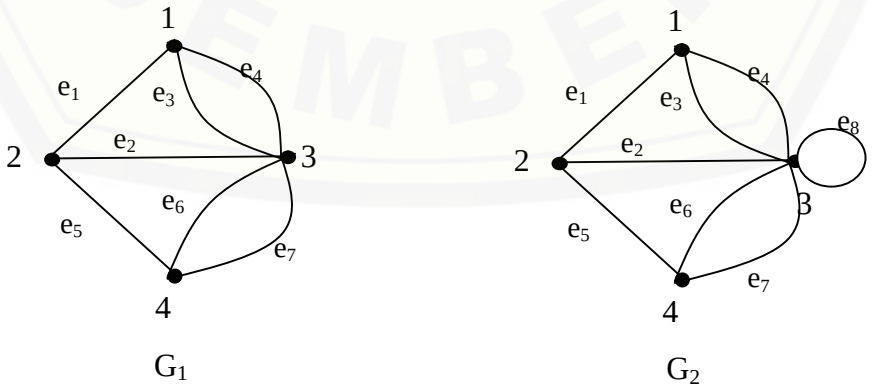
Secara geometri graf dapat digambarkan sebagai sekumpulan noktah (simpul) didalam bidang dwimatra yang dihubungkan dengan sekumpulan garis (sisi). Pada Gambar 2.2 memperlihatkan 2 buah graf, G_1 dan G_2. G_1 adalah graf dengan himpunan simpul *V* dan himpunan sisi *E* sebagai berikut

V = {1, 2, 3, 4}

E = {(1, 2), (2, 3), (1, 3), (1, 3), (2, 4), (3, 4), (3,4)} = himpunan ganda

= { e_1, e_2, e_3, e_4, e_5, e_6, e_7 }

Pada G_1 sisi e_3 = (1, 3) dan sisi e_4 = (1, 3) dinamakan sisi ganda (*multiple edges* atau *pararel edges*) karena kedua sisi ini menghubungkan dua buah simpul yang sama, yaitu simpul 1 dan simpul 3. Pada G_2, sisi e_8 =(3, 3) dinamakan gelang atau kalung karena berawal dan berakhir pada simpul yang sama (Munir, 2010).



Gambar 2.2 Dua buah graf (G_1) graf ganda, (G_2) graf semu

2.1.2 Jenis – Jenis Graf

Graf dapat dikelompokkan menjadi beberapa kategori (jenis) bergantung pada sudut pandang pengelompokannya. Pengelompokan graf dapat dipandang berdasarkan ada tidaknya sisi ganda atau sisi kalung, berdasarkan jumlah simpul, atau berdasarkan orientasi arah pada sisi. Berdasarkan ada tidaknya gelang atau sisi ganda suatu graf, maka graf digolongkan menjadi dua jenis :

a. Graf sederhana (*simple graph*)

   Graf yang tidak mengandung gelang maupun sisi ganda dinamakan graf sederhana.

b. Graf tak sederhana (*unsimple graph*)

   Graf yang mengandung sisi ganda atau gelang dinamakan graf tak sederhana. G_1 dan G_2 adalah contoh graf tak sederhana.

Berdasarkan jumlah simpul pada suatu graf, maka secara umum graf dapat digolongkan menjadi dua jenis :

a. Graf berhingga (*limited graph*)

   Graf berhingga adalah graf yang jumlah simpulnya *n* berhingga.

b. Graf tak berhingga (*unlimited graph*)

   Graf tak berhingga adalah graf yang jumlah simpulnya *n* tidak berhingga banyaknya (Purwanto, 2006).

Berdasarkan orientasi arah pada sisi, maka secara umum graf dibedakan atas dua jenis :

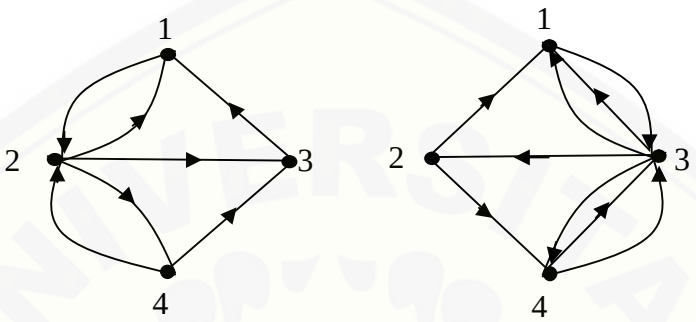
a. Graf tak berarah (*undirected graph*)

   Graf yang sisinya tidak mempunyai orientasi arah disebut sebagai graf tak berarah. Pada graf tak berarah, urutan pasangan simpul yang dihubungkan oleh sisi tidak diperhatikan. Jadi, $(u, v) = (v, u)$ adalah sisi yang sama. Pada jaringan telepon, sisi pada graf berarah menyatakan bahwa saluran telepon dapat beroperasi pada dua arah.

b. Graf berarah (*directed graph* atau *digraph*)

   Graf yang setiap sisinya diberikan orientasi arah disebut graf berarah. Sisi berarah biasa disebut dengan busur (*arc*). Pada graf berarah (u, v) dan (v, u) menyatakan dua buah busur yang berbeda, dengan kata lain $(u, v) \neq (v, u)$

untuk busur (*u*, *v*), simpul *u* dinamakan simpul asal (*initial vertex*) dan simpul *v* dinamakan simpul terminal (*terminal vertex*). Graf berarah sering dipakai untuk menggambarkan aliran proses, peta lalu lintas suatu kota (jalan searah atau dua arah). Pada graf berarah, gelang diperbolehkan tetapi sisi ganda tidak. G_3 dan G_4 merupakan contoh graf berarah (Munir, 2010).



(a) G_3 (b) G_4

Gambar 2.3 (a) graf berarah, (b) graf ganda berarah

2.2 *Network Flow*

Network Flow adalah sebuah graf berarah yang tiap sisinya memiliki kapasitas atau bobot dan pada tiap sisi tersebut terdapat arus (*flow*) yang mengalir antara 2 simpul yang mengapit sisi tersebut. Jumlah arus yang mengalir pada tiap sisi harus lebih kecil atau sama dengan kapasitas sisi tersebut. Pada aplikasinya, sebuah graf berarah sering disebut dengan *network*. Setiap arus (*flow*) yang ada dalam *network*, harus memenuhi sebuah batasanya yaitu arus yang masuk pada suatu simpul harus sama dengan arus yang keluar pada simpul tersebut, kecuali pada *source*, yang keluarnya lebih besar dari arus masuk, dan *sink*, yang arus masuknya lebih besar dari arus keluar sebuah *network* biasanya digunakan untuk memodelkan sistem lalu lintas, saluran pipa, dan saluran listrik (Septiana, 2010).

Network banyak dipakai dalam banyak hal untuk kegunaan yang berbeda-beda. Jaringan transportasi, jaringan listrik dan jaringan telekomunikasi adalah contoh-contoh dimana *network* ditemukan dalam kehidupan sehari-hari. *Representasi network* juga dipakai dalam produksi, distribusi, *project planning*,

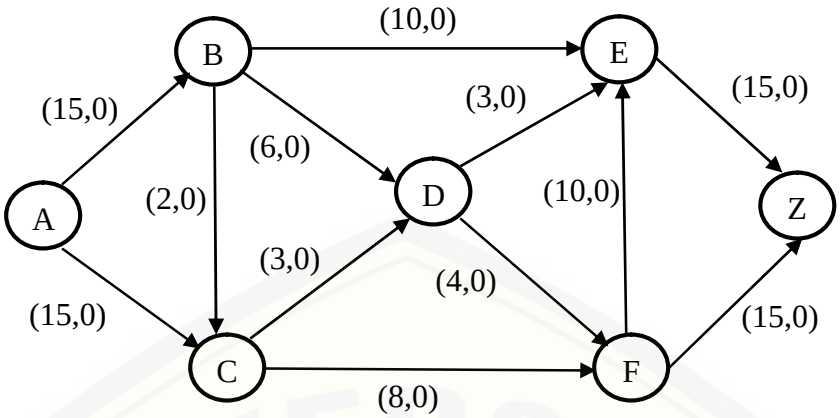
penempatan fasilitas, dan *financial planning.* Suatu *network* diperlukan karena memberi gambaran visual dan bantuan konseptual yang lebih jelas untuk mengetahui hubungan antar komponen dalam sistem yang sering dijumpai dalam banyak kasus. Banyak permasalahan *Network flow* yang sebenarnya berbentuk *linear programming.* Macam - macam aplikasi *network flow* antara lain :

a. *Shortest-Path Problem.*
b. *Minimum Spanning Tree Problem.*
c. *Maximum Flow Problem.*
d. *Minimum Cost Flow Problem* (Habibi, 2008).

2.3 *Maximum Flow Problem*

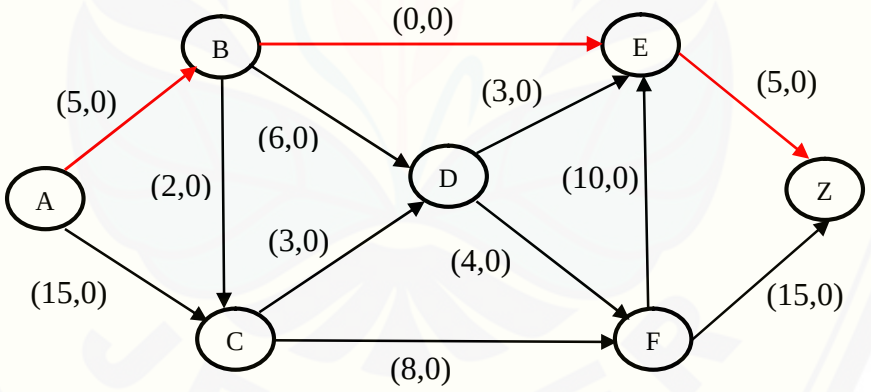
Maximum flow problem dideskripsikan sebagai masalah pencarian untuk mencari arus maksimum yang dapat mengalir pada sebuah *network* yang hanya memiliki satu sumber dan satu tujuan. *Maximum flow* adalah sebuah model yang dapat digunakan untuk mengetahui nilai *maximum* seluruh arus di dalam sebuah sistem jaringan. Contoh *maximum flow* pada sebuah jaringan adalah jaringan listrik, saluran pipa, dan lintasan lalu lintas. Kapasitas pada setiap jaringan akan membatasi jumlah arus atau aliran yang melewatinya. *Maximum flow* mempunyai tujuan untuk memaksimalkan jumlah arus yang melewati jaringan dalam sebuah sistem jaringan. Hal ini tentunya sangat umum terjadi pada bidang transportasi, produksi, komunikasi, dan distribusi (Fakhri, 2008).

Merujuk pada teori graf, pada *Maximum flow problem* diberikan sebuah jaringan graf berbobot dan berarah. Setiap sisinya terdapat kapasitas *c* yang di asosiasikan dengannya, simpul awal disebut sebagai *source,* dan simpul akhir disebut *sink.* Kita disuruh mencari nilai *f* yang persyaratan $f \leq c$ untuk setiap sisi selain *source* dan *sink,* dan jumlah nilai *f* yang masuk kedalam suatu sisi pasti sama dengan jumlah nilai yang meningalkannya. Kemudian mencari nilai *maximum flow* yang memenuhi persyaratan di atas. Sebagai contoh tentukan aliran maksimal pada Gambar 2.4 dibawah ini:



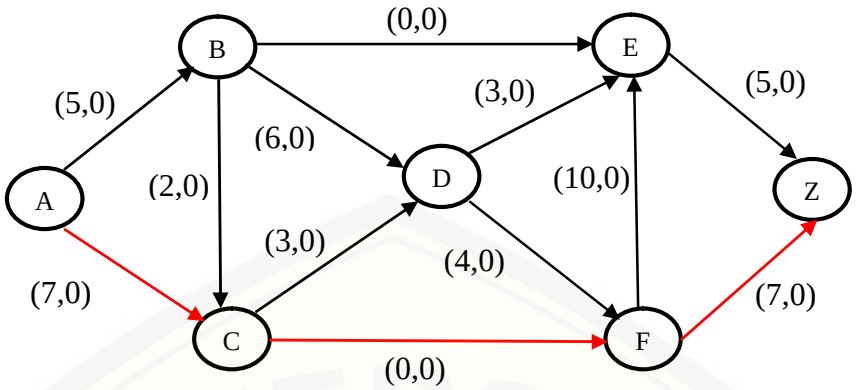
Gambar 2.4 Graf awal (Farizal, 2013)

Pilih lintasan dari sumber ke tujuan untuk memulai iterasi pertama, misalkan kita pilih lintasan A – B – E – Z. Tentukan nilai terkecil dari lintasan ini yaitu min {15, 10, 15} = 10, ditemukan pada sisi B – E. Sehingga nilai yang digunakan f = 10. Nilai kapasitas B – E telah 0, sehingga untuk pencarian berikutnya sisi B – E sudah tidak dapat dilewati lagi.



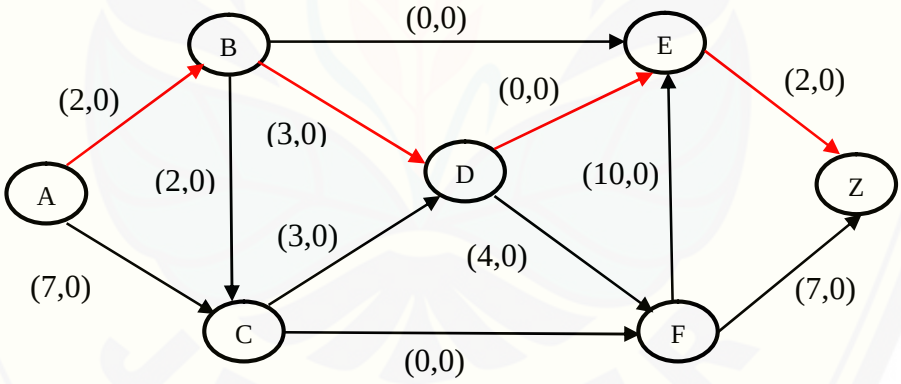
Gambar 2.5 Lintasan A – B – E – Z

Lintasan kedua pada Gambar 2.4 yaitu A – C – F – Z. Nilai terkecil dari lintasan ini adalah min {15, 8, 15} = 8. Jadi nilai yang digunakan f = 8. Nilai kapasitas C – F menjadi 0, sehingga untuk pencarian berikutnya sisi C – F sudah tidak ditelurusi lagi.



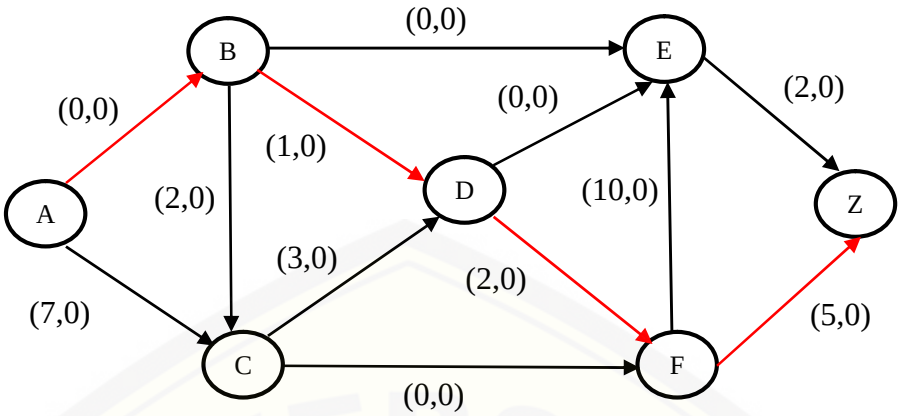
Gambar 2.6 Lintasan A – C – F – Z

Lintasan ketiga yaitu A – B – D – E – Z dengan nilai kapasitas terkecil yaitu min {5, 6, 3, 5} = 3. Jadi nilai *f* yang digunakan adalah *f* = 3. Sehingga kapasitas sisi D – E telah 0, sehingga untuk pencarian berikutnya sisi D – E sudah tidak dapat dilewati lagi. Gambar 2.7 merupakan iterasi ketiga dengan nilai kapasitas masing-masing sisi.



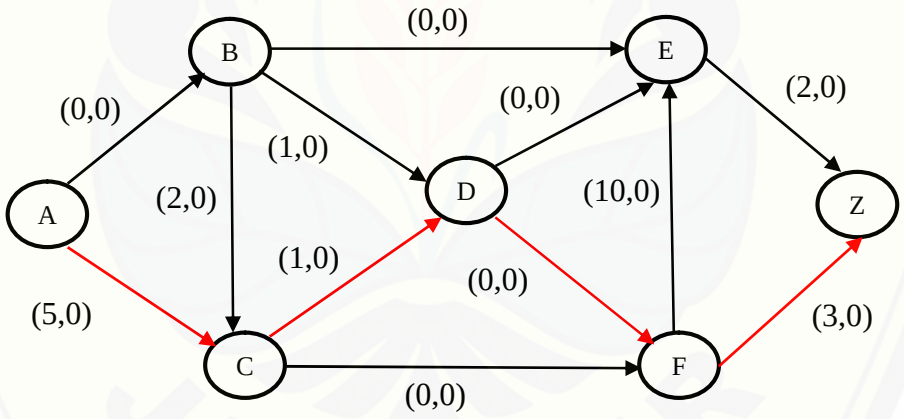
Gambar 2.7 Lintasan A – B – D – E – Z

Lintasan keempat yaitu A – B – D – E – Z. Nilai terkecil dari lintasan ini adalah min {2, 3, 4, 7} = 2. Jadi nilai yang digunakan *f* = 2. Karena nilai kapasitas A – B adalah 2, maka lintasan ini menjadi 0, sehingga untuk pencarian berikutnya sisi A – B sudah tidak ditelurusi lagi.



Gambar 2.8 Lintasan A – B – D – F – Z

Lintasan A – C – D – F – Z merupakan lintasan kelima dari Graf 2.4, nilai terkecil dari lintasan ini adalah min {7, 3, 2, 5} = 2. Jadi nilai *f* yang digunakan adalah *f* = 2. Sehingga kapasitas sisi D – F telah 0, sehingga untuk pencarian berikutnya sisi D – F sudah tidak dapat dilewati lagi.



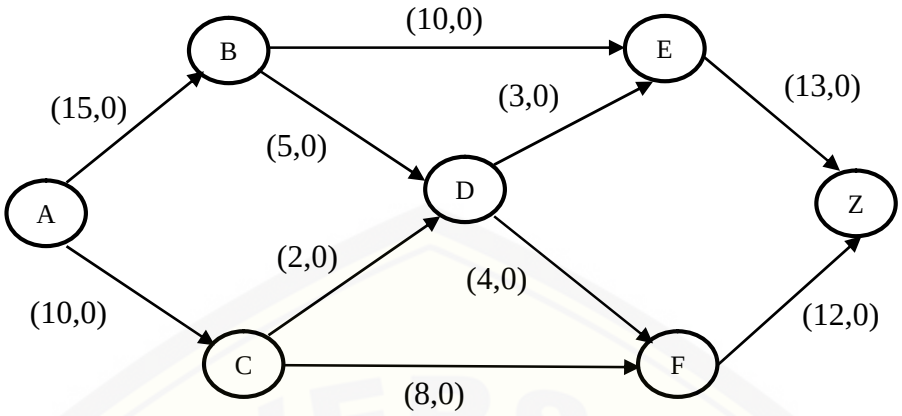
Gambar 2.9 Lintasan A – C – D – F – Z

Karena nilai kapasitas sisi A – B, D – E, D – F sudah 0, maka tidak ada lintasan yang dapat dilewati lagi untuk menuju ke Z. Sehingga pencarian berhenti dan Graf 2.4 memiliki *maximum flow* sebesar

lintasan 1 + lintasan 2 + lintasan 3 + lintasan 4 + lintasan 5 = 10 + 8 + 3 + 2 + 2

= 25

Contoh Graf pada Gambar 2.4 diperoleh aliran maksimal sebesar 25 dengan jaringan aliran maksimal yang ditunjukkan pada Gambar 2.10.



Gambar 2.10 Aliran maksimal

Pada Gambar 2.4, kita di minta untuk mencari arus maksimal yang dapat mengalir dari simpul *A* (*source*) ke simpul *Z* (*sink)* melalui beberapa sisi yang masing-masing memilik kapasitas tertentu. Dengan melihat gambar 2.10 maka dapat disimpulkan bahwa arus maksimal yang dapat mengalir pada *network* di atas adalah 25.

Secara umum *Maximum flow* dapat dijelaskan sebagai berikut:

a. Semua aliran yang melalui sebuah jaringan yang berarah dan terhubung dari *node* awal ke *node* akhir.
b. *Node* awal disebut sumber (*source*) dan *node* akhir disebut tujuan (*sink*). *Node* sisa yang lain dinamakan *node* antara.
c. Aliran hanya diperbolehkan ke arah yang ditunjukkan oleh anak panah dalam suatu cabang. Pada *node* awal, semua cabang meninggalkan *node*, sedangkan pada *node* akhir, semua cabang mengarah masuk ke *node*.
d. Tujuan *maximum flow* adalah memaksimalkan aliran dari sumber ke tujuan.
e. Kapasitas setiap jaringan akan membatasi jumlah arus atau aliran yang melewatinya. Sebagai contoh, sebuah pipa air dengan kapasitas 9 kubik akan pecah apabila kita memaksa pipa itu dilewati oleh air sebesar 30 kubik pada tingkat kapasitas yang sama.

Maximum flow sering diterapkan dalam kehidupan sehari – hari, berikut contoh aplikasi *maximum flow* dikehidupan sehari hari :

a. Maksimasi aliran minyak.

b. Maksimasi aliran air.
c. Maksimasi aliran kendaraan.
d. Maksimasi aliran listrik
e. Maksimasi jaringan distribusi barang.

2.4 Algoritma *Cloning-Based*

Sekarang sudah mulai ditemukan algoritma-algoritma baru yang disebut sebagai *bio-inspired algorithm* yaitu algoritma komputasi yang diinspirasi oleh mekanisme alamiah makhluk hidup. Sebagai contoh adalah *neural network* yang didasari oleh mekanisme jaringan syaraf, algoritma genetika yang didasari sistem evolusi, algoritma semut yang didasari sistem koloni semut, dan komputasi antar protokol yang didasari oleh proses penyebaran penyakit dan masih banyak lagi. Salah satu metode dalam *bio-inspired algorithm* yang baru berkembang adalah *immune system-based algorithm* yaitu algoritma yang diinspirasi dari cara kerja sistem kekebalan tubuh pada manusia. Pada sistem kekebalan tubuh manusia terjadi proses yang dinamakan *clonal selection* (Sasongko, 2001).

Menurut Sasongko (2001), *clonal selection* adalah mekanisme yang digunakan oleh sistem kekebalan tubuh untuk menyeleksi sel yang akan diperbanyak (di-*cloning*) berdasarkan kemampuan untuk mengenali antigen pada patogen (benda asing dari luar tubuh). Sistem kekebalan tubuh tidak dapat mendeteksi secara langsung adanya patogen yang masuk kedalam tubuh, tetapi dapat dideteksi melalui bagian dari patogen yang disebut antigen.

Analogi antara sistem kekebalan tubuh dan masalah optimasi adalah sebagai berikut, response dari sistem *imune* mempresentasikan solusi dan antigen mempresentasikan masalah yang harus diselesaikan. Lebih tepatnya, Sel B adalah sebagai agen-agen buatan yang menjelajahi dan mengeksplorasi lingkungan buatan. Patogen sebagai masalah optimasi, dalam kasus ini masalah optimasi digambarkan oleh antigen dan patogen. Mekanisme seleksi positif dan seleksi negatif digunakan untuk mengontrol perbanyakan agen dengan mengeliminasi solusi yang buruk atau tidak berguna. Jadi, aturan seleksi positif dan seleksi negatif dapat dipertimbangkan sebagai mekanisme yang tidak hanya memilih

solusi yang tepat, tetapi juga mengatur jumlah populasi agen yang tumbuh pada proses *cloning.* Algoritma ini diajukan oleh M. Bakhouya and J. Gaber dari *Universite de Technologie de Belfort-Montbeliard* pada tahun 2006 (Widodo, 2012).

Tabel 2.1 Analogi Sistem Kekebalan Tubuh pada Masalah Optimasi

Sistem Kekebalan Tubuh	**Masalah optimasi**
Patogen	Permasalahan (dalam hal ini graf kota dimana tiap titik (kota) digambarkan sebagai antigen)
Respon Tubuh	Solusi
Sel B	Agen Pencari
Clonal Selection	Menciptakan agen pencari baru untuk menjelajahi kota
Seleksi Positif dan Seleksi Negatif	Penyeleksian agen yang buruk atau tidak berguna untuk membunuh dirinya sendiri (apoptosis)

(Bakhouya, 2007)

Penerapan algoritma *cloning-based* pada penyelesaian *maximum flow problem* adalah sebagai berikut:

a. Titik mempresentasikan tiap kota, yang dianalogikan sebagai antigen pada patogen.
b. Sel B adalah agen pencari yang bergerak dari satu kota ke kota tetangganya dan dapat mengkloning dirinya atau menghancurkan dirinya sendiri berdasarkan kriteria seleksi positif dan seleksi negatif.
c. Kapasitas jalan dianalogikan sebagai daya afinitas yang dipunyai Sel B untuk merangsang respon dari tubuh terhadap patogen.
d. Solusi dari permasalahan dianalogikan sebagai afinitas terbesar yang dibutuhkan sampai tubuh merespon adanya patogen.

Secara formal, misalkan $C=\{a, \ldots, z\}$ adalah himpunan kota-kota, $A=\{(x, y) : x, y \in C\}$ adalah ujung – ujung dari sisi tersebut dan $\delta(x, y)$ adalah kapasitas jalan antara $x, y \in C$. Seleksi positif terjadi jika kota tidak membangun rute dan tidak mencapai kota tujuan. Sedangkan seleksi negatif terjadi apabila semua agen telah menyelesaikan perjalanannya, maka agen yang membentuk rute dengan afinitas paling besar akan menjadi solusi. Sedangkan agen yang tidak dapat membentuk rute dengan jumlah afinitas lebih besar atau sama dengan afinitas minimal setelah melalui suatu titik untuk menuju ke titik tujuan akan dimatikan dan rute yang telah terbentuk dihapus.

Langkah-langkah dari algoritma *cloning-based* sebagai berikut:

a. Suatu agen bergerak melalui salah satu sisi dari titik asal dan kemudian mengkloning ke titik lain yang belum terlewati. Jika suatu agen dari salah satu sisi sumber dan tidak mampu membentuk rute untuk mencapai kota tujuan maka akan memicu seleksi positif, maka kota dan rute yang dihasilkan sebelumnya tidak berguna (dimatikan).
b. Pada *maximum flow problem* agen yang telah mencapai kota tujuanlah yang tidak akan mengkloning dirinya dan proses kloning selesai.
c. Pada *maximum flow problem* ketika semua agen telah mencapai kota tujuan yang melalui satu sisi yang sama yang terhubung dengan titik awal maka akan memicu seleksi negatif dan akan dipilih rute yang memiliki bobot terbesar dan rute lain dihapus (Widodo, 2012).

2.5 Algoritma *Edmonds-Karp*

Algoritma *Edmonds-karp* pertama kali diperkenalkan oleh seorang ilmuan Rusia, Dinic pada tahun 1970, dan dipopulerkan oleh Jack Edmonds dan Richard Karp pada tahun 1972 (Dey, 1993). Dalam algoritma *Edmonds-karp*, lintasan penambah yang dipilih merupakan lintasan penambah terpendek. Untuk memilih lintasan penambah terpendek yang dimaksud, digunakan algoritma *Breadth First Search* (BFS). Sehingga hanya diperlukan sedikit iterasi dalam percarian *maximum flow* dalam suatu permasalahan yang ada dibandingkan dengan algoritma *Ford Fulkerson*.

Adapun untuk langkah-langkah dari algoritma *Edmons-karp* sebagai berikut:

a. Tentukan *residual network* dari *N*.
b. Inisialisasi *flow* untuk setiap busur *ij* pada *N* sebesar nol ($F_{ij} = 0$)
c. Identifikasikan suatu lintasan penambah pada *residual network* dengan menggunakan algoritma BFS (*Breadth First Search*).
d. Jika telah diperoleh suatu lintasan penambah, maka tentukan kapasitas residu lintasan penambah tersebut yang dinotasikan dengan Δ.
e. Tambahkan *flow* sebesar Δ ke setiap busur pada lintasan penambah tersebut.

Jika masih ada lintasan penambah yang lain, ulangi langkah 3 sampai dengan langkah 5. Jika tidak ada lintasan penambah yang lain, hitung aliran pada setiap busur (Fathimatuzzahro, 2013).

Algoritma *Breadth-First Search* (BFS) atau dikenal juga dengan nama algoritma pencarian melebar adalah algoritma yang melakukan pencarian secara melebar yang dimulai dari simpul awal, lalu dilanjutkan dengan mengunjungi simpul-simpul yang bertetangga dengan simpul awal tersebut. Setelah itu kunjungi simpul-simpul yang bertetangga dengannya dan belum dikunjungi, demikian seterusnya sampai seluruh simpul berhasil dikunjungi (Madanella, 2007).

Sedangkan untuk langkah-langkah dari algoritma BFS (*Breadth First Search*) sebagai berikut :

a. Masukkan simpul awal kedalam antrian.
b. Ambil simpul dari awal antrian, lalu cek apakah simpul merupakan solusi.
c. Jika simpul merupakan solusi, pencarian selesai.
d. Jika simpul bukan solusi, masukkan seluruh simpul yang bertetangga dengan simpul tersebut (simpul anak) kedalam antrian.
e. Jika antrian kosong dan setiap simpul sudah dicek, pencarian selesai dan solusi tidak ditemukan.
f. Ulangi pencarian dari langkah kedua (Madanella, 2007).

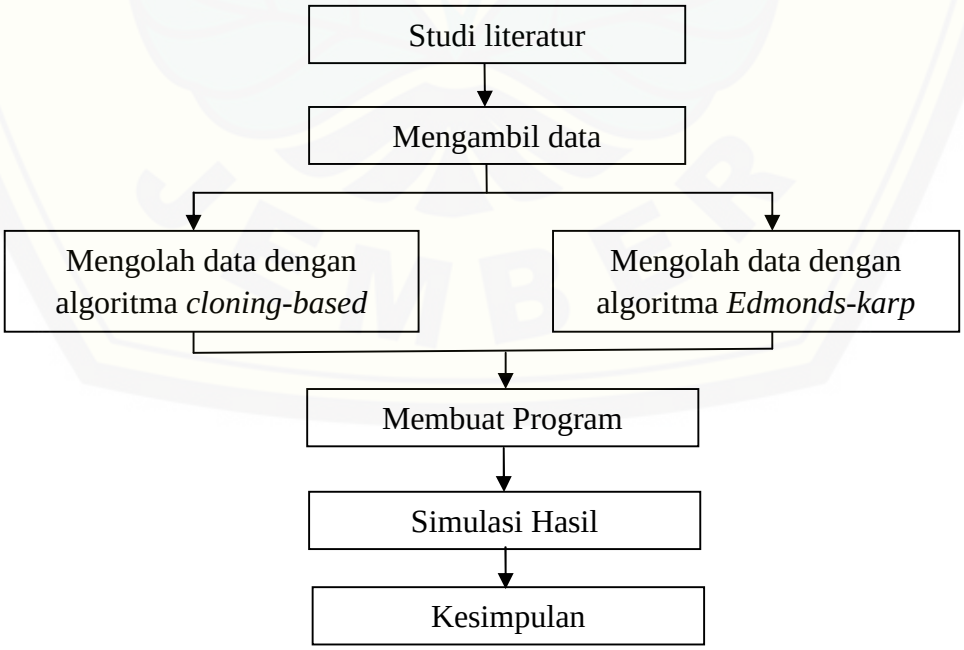
BAB 3. METODE PENELITIAN

3.1 Data Penelitian

Data yang digunakan dalam penelitian ini adalah data sekunder yang diambil dari tugas akhir Farizal (2013) yang berupa data jaringan listrik Kota Tegal wilayah Distribusi Kebasen 11. Pada penelitian ini, simpul 1 merupakan gardu induk, sedangkan simpul 2 sampai simpul 46 diasumsikan sebagai tiang listrik, dan sisi diasumsikan sebagai kabel yang menghubungkan antar tiang listrik. Kapasitas kabel yang digunakan dalam satuan Ampere.

3.2 Langkah-langkah Penelitian

Pada penelitian ini dibahas tentang penggunaan algoritma *cloning-based* dan algoritma *edmonds-karp* dalam menyelesaikan *maximum flow problem*. Langkah-langkah yang dilakukan dalam penelitian ini dapat dilihat secara skematik pada Gambar 3.1.



Gambar 3.1 Skema Metode Penelitian

Berikut adalah penjelasan dari skema pada Gambar 3.1 :

a. Studi literatur

   Penelitian ini diawali dengan mempelajari buku–buku, jurnal, dan sumber lain yang membahas tentang *maximum flow problem*, algoritma *cloning-based*, dan algoritma *edmonds-karp*.

b. Pengambilan data

   Data yang digunakan adalah jaringan listrik Kota Tegal wilayah Distribusi Kebasen 11 yang terdapat pada tugas akhir Farizal (2013).

c. Mengolah data

   Data yang didapat pada poin b, diselesaikan menggunakan algoritma *cloning-based* dan algoritma *edmons-karp*.

d. Membuat program

   Peneliti akan membuat *script* pemrograman algoritma *cloning-based* dan algoritma *edmonds-karp* menggunakan program Matlab R2009a untuk menyelesaikan *maximum flow problem.*

e. Simulasi hasil

   Peneliti akan membandingkan hasil *ouput* program antara algoritma *cloning-based* dan algoritma *edmonds-karp* berdasarkan aliran maksimal yang diperoleh.

f. Kesimpulan

   Kesimpulan yang akan peneliti ambil berdasarkan simulasi hasil yang diperoleh.

BAB 5. PENUTUP

5.1 Kesimpulan

Berdasarkan hasil yang diperoleh, didapat kesimpulan sebagai berikut:

a. Penerapan algoritma *cloning based* dan algoritma *edmonds karp* pada data jaringan listrik Kota Tegal wilayah Distribusi Kebasen 11 dengan sepuluh kali percobaan menunjukkan bahwa *maximum flow* yang dihasilkan algoritma *edmonds karp* lebih besar dari pada algoritma *cloning based*. Algoritma *cloning based* menghasilkan *maximum flow* yang berubah-ubah. Hal ini dipengaruhi oleh lintasan awal yang diinputkan. Sedangkan *algoritma edmonds karp* menghasilkan *maximum flow* yang sama yaitu sebesar 1300.
b. Algoritma *edmonds karp* lebih baik dari pada algoritma *cloning based* jika dilihat dari *maximum flow* yang dihasilkan pada sepuluh kali percobaan.
c. *Running time* algoritma *cloning based* lebih baik dari pada algoritma *edmonds karp* karena *running time* yang dihasilkan algoritma *cloning based* lebih sedikit dari pada algoritma *edmonds karp*.

5.2 Saran

Peneliti selanjutnya dapat mengembangkan dan menerapkan algoritma *cloning based* dan algoritma *edmonds karp* dalam permasalahan lainnya, misalkan untuk mencari rute terpendek, *travelling salesman problem,* maupun masalah *knapsack*. Penelitian selanjutnya juga dapat mengembangkan algoritma *cloning based* dalam penentuan lintasan awal agar hasil yang diperoleh maksimal.

DAFTAR PUSTAKA

Apriliani, I. 2011. "Penyelesaian *Travelling Salesman Problem* (TSP) Algoritma Menggunakan Semut dan Algoritma *Cheapest Insertion Heuristic* (CIH)". Tidak dipublikasikan. Skripsi. Jember: Jurusan Matematika Fakultas MIPA Universitas Jember.

Bakhouya, M., J. Gaber. 2007. "An Immune Inspired-based Optimization Algorithm: Application to the Travelling Salesman Problem". AMO – Advanced Modeling and Optimization, **9** (1): 105-116.

Dey, T.K. 2009. *Edmons-Karp Algorithm* (*Advanced Algorithms (CSE 794)*). http://www.cse.ohio-state.edu/~tamaldey/course/794/ek-algo.pdf. [1 Desember 2015].

Fakhri. 2008. Penerapan Algoritma Djikstra dalam Pencarian Solusi *Maximum Flow Problem*. *Strategi Algoritmik*.

Farizal, T. 2013. "Pencarian Aliran Maksimal Dengan Algoritma Ford-Fulkerson." Tidak Diterbitkan. Skripsi. Semarang: Fakultas MIPA Universitas Negeri Semarang.

Fathimatuzzahro. 2013. "Penyelesaian Aliran Maksimum Menggunakan *Edmons-Karp Algorithm*". Malang: Universitas Negeri Malang.

Habibi, L. 2008. "Pemodelan *Network Flow Analysis* Sistem Distribusi Air Menggunakan Algoritma Genetika – Metode Newton". Tidak Diterbitkan. Tesis. Bandung: ITB.

Madanella, E.D.M. 2007. "Analisis Penggunaan Algoritma Pencarian Melebar (BFS) dan Algoritma Pencarian Mendalam (DFS) dalam Teori Graf". Bandung: ITB.

Merdekawaty, E. 2011. "Penyelesaian Masalah Aliran Maksimal Di Dalam Jaringan Dengan Menggunakan Algoritma *Ford-Fulkerson*". Skripsi. Tidak Diterbitkan. Jakarta: Universitas Gunadarma.

Munir, R. 2010. *MATEMATIKA DISKRIT*. Bandung: Informatika Bandung.

Purwanto, Gina, dan Erlina. 2006. *Matematika Diskrit*. Cirebon: PT Ercontara Rajawali.

Sasongko, B.P. 2001. "*Cloning-Based Algorithm* dan Aplikasinya Dalam *Travelling Salesperson Problem*". Bandung: ITB.

Sedgewick, R. 2002. *Algorithm in Java: parts 1-4 Third Edition*. Boston: Pearson Education, lnc.

Septiana, E.P., dan Ketut, E. 2010. "Pengembangan Simulasi Aliran Air Pada Saluran Drainase Kota Menggunakan Pemodelan *Network Flow*". Surabaya: ITS.

Widodo, Purwanto, dan Subanji. 2012. "Penyelesaian *Maximum Flow Problem* dengan Algoritma *Cloning-Based*". Malang: Universitas Negeri Malang.

LAMPIRAN

LAMPIRAN A. Data Penelitian

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
1	0	585	0	0	585	0	0	585	0	0	0	0	0	0	0	0	0
2	0	0	585	0	0	0	0	0	585	0	0	0	0	0	0	0	0
3	0	0	0	585	0	0	0	0	0	320	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0	0	0	225	0	0	0	0	0
5	0	0	0	0	0	320	0	0	0	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	320	0	0	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	585	0	0	0	0	0	0	0	0	0
8	0	0	0	0	0	0	0	0	0	0	0	0	585	0	0	0	0
9	0	0	0	0	0	0	0	0	0	0	0	0	0	585	0	0	0
10	0	0	0	0	0	0	0	0	0	0	225	0	0	0	320	0	0
11	0	0	0	0	0	0	0	0	0	0	0	170	0	0	0	0	0
12	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	170
13	0	0	0	0	0	0	0	0	0	0	0	0	0	585	0	0	0
14	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
15	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	365	0
16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	225
17	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

18	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
19	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
20	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
21	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
22	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
23	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
24	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
25	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
26	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
27	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
28	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
29	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
30	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
31	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
32	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
33	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
34	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
35	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
36	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
37	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
38	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
39	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
40	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
41	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
42	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

43	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
44	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
45	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
46	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

18	**19**	**20**	**21**	**22**	**23**	**24**	**25**	**26**	**27**	**28**	**29**	**30**	**31**	**32**	**33**	**34**
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
585	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	585	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	325	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	225	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	170	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	585	0	0	0	0	0

320	0	0	585	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	170	170	0	0	0	0	0	0	0	0	0
0	0	0	0	320	0	0	0	0	0	0	0	585	0	0	0	0
0	0	0	0	0	320	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	320	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	585	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	320	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	320	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	585	0	585	585	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	585	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	585	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	585	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	585
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

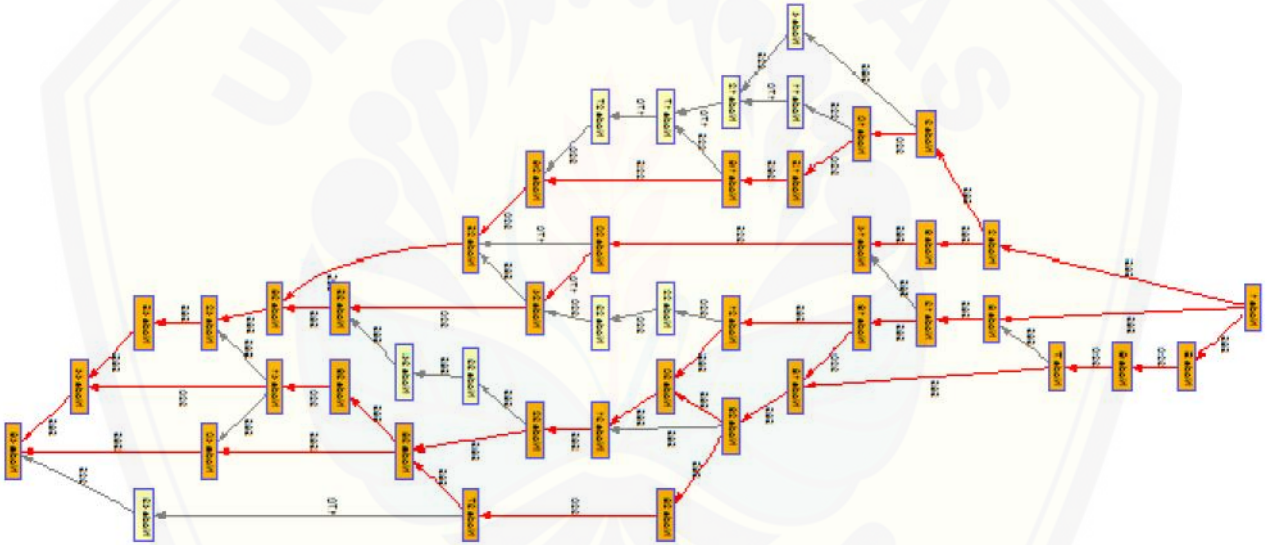
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

35	**36**	**37**	**38**	**39**	**40**	**41**	**42**	**43**	**44**	**45**	**46**
0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0

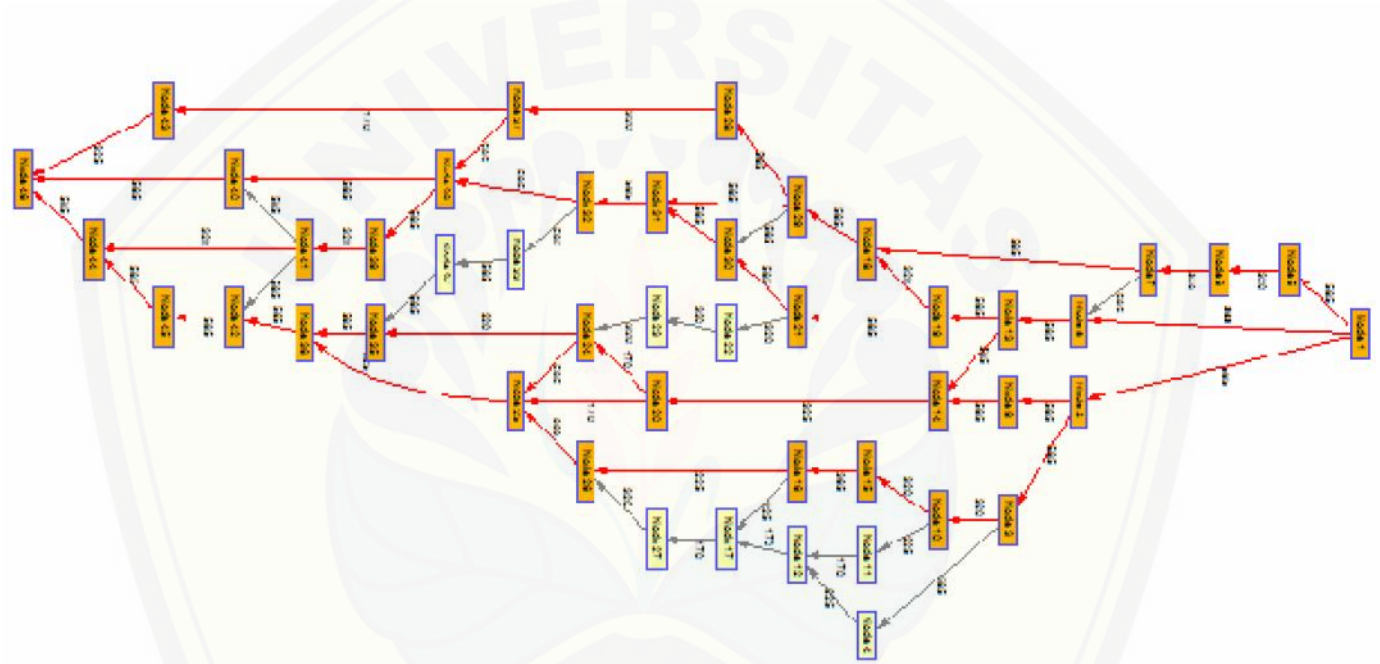
0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0
320	0	0	0	0	0	0	0	0	0	0	0
0	225	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0
0	0	320	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	585	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0
585	0	0	0	0	0	0	0	0	0	0	0
0	585	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	585	0	0	0	0
0	0	0	585	0	0	0	0	170	0	0	0
0	0	0	0	585	585	0	0	0	0	0	0
0	0	0	0	0	0	320	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	585
0	0	0	0	0	585	0	585	0	320	0	0
0	0	0	0	0	0	0	0	0	0	585	0
0	0	0	0	0	0	0	0	0	0	0	325
0	0	0	0	0	0	0	0	0	0	0	585

0	0	0	0	0	0	0	0	0	585	0	0
0	0	0	0	0	0	0	0	0	0	0	0

LAMPIRAN B. Data Awal Lintasan yang Dilewati dengan Menggunakan Algoritma *Cloning Based*



LAMPIRAN C. Data Awal Lintasan yang Dilewati dengan Menggunakan Algoritma *Edmonds Karp*



LAMPIRAN D. *Script* Program Algoritma *Cloning Based*

```
function [simp_urut bobot_maks titik]=cloning_based(data,gbr,sol_awal)
load('simp_data1.mat');
[urutan bobot]=rute(data);
% bobot_maks=zeros(1,length(bobot));
k=0;
% gbr=0;
data0=data;
n=length(data);
data2=zeros(n,n);
bobot_maks=0;

while ~isempty(bobot) && k<100
k=k+1;
[urutan bobot]=rute(data0);
% [bbt_min indx]=min(bobot);
if k>1
    if isequal(data,data00)
    urut1=cell(1,1);
    urut2=cell(1,1);
    urut3=cell(1,1);
    k1=0; k2=0; k3=0;
    bobota=[]; bobotb=[]; bobotc=[];
    for ii=1:length(urutan)
        a11=cell2mat(urutan(ii,1));
        if ~isempty(a11)
        if isequal([1 2],a11(1:2))
            k1=k1+1;
            urut1(k1,1)=urutan(ii,1);
            bobota(k1)=bobot(ii);
        elseif isequal([1 8],a11(1:2))
            k2=k2+1;
            urut2(k2,1)=urutan(ii,1);
            bobotb(k2)=bobot(ii);
```

```
34 -            elseif isequal([1 5],a11(1:2))
35 -                k3=k3+1;
36 -                urut3(k3,1)=urutan(ii,1);
37 -                bobotc(k3)=bobot(ii);
38 -            end
39 -            end
40
41 -        end
42 -        if k1>0
43 -            urutan=cell(1,1); bobot=[];
44 -            urutan=urut1;
45 -            bobot=bobota;
46 -        elseif k2>0
47 -            urutan=cell(1,1); bobot=[];
48 -            urutan=urut2;
49 -            bobot=bobotb;
50
51 -        elseif k3>0
52 -            urutan=cell(1,1); bobot=[];
53 -            urutan=urut3;
54 -            bobot=bobotc;
55
56 -        end
57
58 -        end
59
60 -        [bbt_min indx]=max(bobot);
61 -    else
62 -        bnr=0;
63 -        if ~isempty(sol_awal)
64 -            bb=[];
65 -            for i1=1:length(sol_awal)-1
66 -                bb(i1)=data0(sol_awal(i1),sol_awal(i1+1));
```

```
67 -            end
68 -            if sol_awal(end)~=n || min(bb)==0
69 -                warndlg('Urutan Awal Salah!!!');
70 -                break
71 -            else
72 -                urutan=cell(1,1);
73 -                urutan={[sol_awal]};
74 -                bobot=min(bb);
75 -                ii=1;
76 -            end
77 -        end
78
79 -        if bnr==1
80 -        indx=ii;
81 -        bbt_min=bobot(indx);
82 -        else
83 -        indx=1;
84 -        bbt_min=bobot(indx);
85
86 -        if isequal(data,data00)
87 -        urut1=cell(1,1);
88 -        urut2=cell(1,1);
89 -        urut3=cell(1,1);
90 -        k1=0; k2=0; k3=0;
91 -        bobota=[]; bobotb=[]; bobotc=[];
92 -        for ii=1:length(urutan)
93 -            a11=cell2mat(urutan(ii,1));
94 -            if ~isempty(a11)
95 -            if isequal([1 2],a11(1:2))
```

```
            k1=k1+1;
            urut1(k1,1)=urutan(ii,1);
            bobota(k1)=bobot(ii);
        elseif isequal([1 8],a11(1:2))
            k2=k2+1;
            urut2(k2,1)=urutan(ii,1);
            bobotb(k2)=bobot(ii);
        elseif isequal([1 5],a11(1:2))
            k3=k3+1;
            urut3(k3,1)=urutan(ii,1);
            bobotc(k3)=bobot(ii);
        end
        end

    end
    if k1>0
        urutan=cell(1,1); bobot=[];
        urutan=urut1;
        bobot=bobota;
    elseif k2>0
        urutan=cell(1,1); bobot=[];
        urutan=urut2;
        bobot=bobotb;

    elseif k3>0
        urutan=cell(1,1); bobot=[];
        urutan=urut3;
        bobot=bobotc;

    end

    [bbt_min indx]=max(bobot);
    end
```

```
129
130 -       end
131 -   end
132 -   if ~isempty(indx)
133 -   i=indx(1);
134     %----------
135 -   if isequal(data,data00)
136     % %     bbt_min=bobot1(k);
137     % %     urutan(i,1)=urutan1(k,1);
138 -   end
139     %----------
140 -       a=cell2mat(urutan(i,1));
141 -       for j=1:length(a)-1
142 -           data0(a(j),a(j+1))=data0(a(j),a(j+1))-bbt_min;
143 -           data2(a(j),a(j+1))=data2(a(j),a(j+1))+bbt_min;
144 -       end
145 -       bobot_maks(k)=bbt_min;
146 -       simp_urut(k)=urutan(i,1);
147
148 -   end
149 -   end
150
151 -   titik=zeros(1,length(data));
152 -   for i=1:length(bobot_maks)
153 -       a=cell2mat(simp_urut(i));
154 -       for i1=1:length(a)
155 -           titik(a(i1))=1;
156 -       end
157
158 -   end
159     % titik
```

```
162 -     if gbr==1
163 -     h=view(biograph(data2,[],'ShowWeights','on'));
164 -     h.ShowTextInNodes = 'label';
165 -     dolayout(h);
166
167 -     h=view(biograph(data,[],'ShowWeights','on'));
168 -     h.ShowTextInNodes = 'label';
169 -     dolayout(h);
170 -     for i=1:length(bobot_maks)
171 -         a=cell2mat(simp_urut(i));
172 -         data=[];
173 -         for j=1:length(a)
174 -             if a(j)~=0
175 -                 data(j)=a(j);
176 -             end
177 -         end
178 -         if ~isempty(data)
179 -         if i>=1
180 -             set(h.nodes(data),'Color',[1 0.7 0]);
181 -     edges = getedgesbynodeid(h,get(h.Nodes([1 data 1]),'ID'));
182 -     set(edges,'LineColor',[1 0 0])
183 -     set(edges,'LineWidth',1)
```

```
225 -         end
226 -         end
227 -     end
228 -     end
```

LAMPIRAN E. *Script* Program Algoritma *Edmonds Karp*

```
function [simp_urut bobot_maks titik]=EKD(data,gbr)
load('simp_data2.mat');
[urutan bobot]=rute(data);
% bobot_maks=zeros(1,length(bobot));
k=0;
% gbr=0;
data0=data;
data1=data;
n=length(data);
data2=zeros(n,n);
bobot_maks=0;

while ~isempty(bobot) && k<100
k=k+1;
[urutan bobot]=rute(data0);
% [bbt_min indx]=min(bobot);
%-------------
n_rute=[];
for i=1:length(bobot)
    n_rute(i)=length(cell2mat(urutan(i,1)));
end
[pp indx]=min(n_rute);
if ~isempty(indx)
    bb=[];
    for i=1:length(indx)
        bb(i)=bobot(indx(i));
    end
    [bb0 cc]=min(bb);
bbt_min=bobot(indx(cc(1)));
```

```
33 -    if ~isempty(indx)
34 -    i=indx(cc(1));
35      %----------
36 -    if isequal(data,data00)
37 -        bbt_min=bobot1(k);
38 -        urutan(i,1)=urutan1(k,1);
39 -    end
40      %----------
41 -        a=cell2mat(urutan(i,1));
42 -        for j=1:length(a)-1
43 -            data0(a(j),a(j+1))=data0(a(j),a(j+1))-bbt_min;
44 -            data2(a(j),a(j+1))=data2(a(j),a(j+1))+bbt_min;
45 -            data1(a(j),a(j+1))=data1(a(j),a(j+1))-bbt_min;
46 -            data1(a(j+1),a(j))=data1(a(j+1),a(j))+bbt_min;
47 -        end
48 -        bobot_maks(k)=bbt_min;
49 -        simp_urut(k)=urutan(i,1);
50
51 -    end
52 -    end
53 -    k=k-1;
54 -    [urutan bobot]=rute(data1);
55
56      % h=view(biograph(data1,[],'ShowWeights','on'));
57      % h.ShowTextInNodes = 'label';
58      % dolayout(h);
59 -    while ~isempty(bobot) && k<100
60 -    k=k+1;
61 -    [urutan bobot]=rute(data1);
```

```
%-------------
n_rute=[];
for i=1:length(bobot)
    n_rute(i)=length(cell2mat(urutan(i,1)));
end
[pp indx]=min(n_rute);
if ~isempty(indx)
    bb=[];
    for i=1:length(indx)
        bb(i)=bobot(indx(i));
    end
    [bb0 cc]=min(bb);
bbt_min=bobot(indx(cc(1)));
end
%------------------

if ~isempty(indx)
i=indx(cc(1));
%----------
if isequal(data,data00)
    bbt_min=bobot1(k);
    urutan(i,1)=urutan1(k,1);
end
%----------
    a=cell2mat(urutan(i,1));
    for j=1:length(a)-1
        data1(a(j),a(j+1))=data1(a(j),a(j+1))-bbt_min;
        if data(a(j),a(j+1))~=0
        data2(a(j),a(j+1))=data2(a(j),a(j+1))+bbt_min;
```

```
92 -            end
93 -            data1(a(j+1),a(j))=data1(a(j+1),a(j))+bbt_min;
94 -        end
95 -        bobot_maks(k)=bbt_min;
96 -        simp_urut(k)=urutan(i,1);
97
98 -    end
99 -    end
100
101 -   titik=zeros(1,length(data));
102 -   for i=1:length(bobot_maks)
103 -       a=cell2mat(simp_urut(i));
104 -       for i1=1:length(a)
105 -           titik(a(i1))=1;
106 -       end
107
108 -   end
109
110     %================================
111 -   if gbr==1
112 -   h=view(biograph(data2,[],'ShowWeights','on'));
113 -   h.ShowTextInNodes = 'label';
114 -   dolayout(h);
115
116 -   h=view(biograph(data,[],'ShowWeights','on'));
117 -   h.ShowTextInNodes = 'label';
118 -   dolayout(h);
119
120 -   for i=1:length(bobot_maks)
121 -       a=cell2mat(simp_urut(i));
122 -       data=[];
123 -       for j=1:length(a)
```

```
123 -       for j=1:length(a)
124 -           if a(j)~=0
125 -               data(j)=a(j);
126 -           end
127 -       end
128 -       if ~isempty(data)
129 -       if i>=1
130 -           set(h.nodes(data),'Color',[1 0.7 0]);
131 -   edges = getedgesbynodeid(h,get(h.Nodes([1 data 1]),'ID'));
132 -   set(edges,'LineColor',[1 0 0])
133 -   set(edges,'LineWidth',1)
```

```
175 -       end
176 -       end
177 -   end
178 -   end
```

LAMPIRAN F. *Script* Program Pencarian Rute

```
function [urut bobot posisi]=rute(data)

data0=data;

n=length(data);
posisi=cell(1,1);
pos=1; k=0;
while ~isempty(find(data(1,:))>0) && k<1000
k=k+1;
        posisi(k,1)={[1]};
    for j=1:n-1
    i=pos;
    indx=find(data(i,:)>0);
    if ~isempty(indx)
        for ii=1:length(indx)
            b1=cell2mat(posisi(k,1));
            if isempty(find(indx(ii)==cell2mat(posisi(k,1)))) && b1(end)~=n
        posisi(k,1)={[cell2mat(posisi(k,1)) indx(ii)]};
        pos=indx(ii);
        break
            end
        end
    end
    end
        jj=cell2mat(posisi(k,1));
    for j=length(jj):-1:1
    indx=find(data(jj(j),:)>0);
        if length(indx)>1

            data(jj(j),indx(1))=0;
            break
        end
```

```
34 -        end
35 -        pos=1;
36 -        if k>1
37 -        if isequal(cell2mat(posisi(k,1)),cell2mat(posisi(k-1,1)))
38      %           k=k-1;
39 -        end
40 -        end
41 -    end
42
43 -    urut=cell(1,1);
44 -    m=0;
45 -            aa=cell2mat(posisi(1,1));
46 -            if ~isempty(aa)
47 -            if aa(end)==n
48 -            m=m+1;
49 -            urut(m,1)=posisi(1,1);
50 -            end
51 -            end
52 -    for i=2:k
53 -       if ~isequal(cell2mat(posisi(i,1)),cell2mat(posisi(i-1,1)))
54 -            aa=cell2mat(posisi(i,1));
55 -            if aa(end)==n
56 -            m=m+1;
57 -            urut(m,1)=posisi(i,1);
58      %          cell2mat(posisi(i,1))
59 -            end
60 -        end
61 -    end
62
63 -    bobot=[];
64 -    if m>0
65 -    for i=1:m
```

```
66 -        a=cell2mat(urut(i,1));
67 -        b=[];
68 -        for j=1:length(a)-1
69 -            b(j)=data0(a(j),a(j+1));
70 -        end
71 -        bobot(i)=min(b);
72
73 -    end
74 -    end
```

LAMPIRAN G. *Script* Program Proses

```
 1 -    clc;
 2
 3 -    data=get(tabel1,'data');
 4 -    gbr1=1;
 5 -    gbr2=1;
 6      %================================
 7 -    wkt=tic;
 8 -    sol0=str2num(get(edit1,'string'));
 9 -    [simp_urut1 bobot_maks1 titik1]=cloning_based(data,gbr1,sol0);
10 -    wkt1=toc(wkt);
11 -    k=0;
12 -    ket1=cell(1,1);
13 -    k=k+1;
14 -    ket1(k)={'Metode Cloning Based'};
15 -    k=k+1;
```

```
16 -    ket1(k)={'Lintasan: '};
17 -    for i=1:length(bobot_maks1)
18 -        a=[];
19 -    a=cell2mat(simp_urut1(i));
20 -    k=k+1;
21 -    ket1(k)={['     Lintasan ' num2str(i) ' :    ' num2str(a)]};
22 -    k=k+1;
23 -    ket1(k)={['     Bobot Lintasan ' num2str(i) ' :    ' num2str(bobot_maks1(i))]};
24
25 -    end
26 -    k=k+1;
27 -    ket1(k)={['Hasil Maximum Flow: ' num2str(sum(bobot_maks1))]};
28 -    k=k+1;
29 -    ket1(k)={['Running Time: ' num2str(wkt1) ' detik']};
30 -    ket1=char(ket1);
31
32 -    set(output1,'string',ket1);
33
34
35      %================================
36 -    wkt=tic;
37 -    [simp_urut2 bobot_maks2 titik2]=EKD(data,gbr2);
38 -    wkt2=toc(wkt);
39 -    k=0;
40 -    ket2=cell(1,1);
41 -    k=k+1;
42 -    ket2(k)={'Metode Edmonds Karp'};
43 -    k=k+1;
44 -    ket2(k)={'Lintasan: '};
45 -    for i=1:length(bobot_maks2)
```

```
46 -        a=[];
47 -    a=cell2mat(simp_urut2(i));
48 -    k=k+1;
49 -    ket2(k)={['      Lintasan ' num2str(i) ' :    ' num2str(a)]};
50 -    k=k+1;
51 -    ket2(k)={['      Bobot Lintasan ' num2str(i) ' :    ' num2str(bobot_maks2(i))]};
52
53 -    end
54 -    k=k+1;
55 -    ket2(k)={['Hasil Maximum Flow: ' num2str(sum(bobot_maks2))]};
56 -    k=k+1;
57 -    ket2(k)={['Running Time: ' num2str(wkt2) ' detik']};
58 -    ket2=char(ket2);
59
60 -    set(output2,'string',ket2);
```