PERBANDINGAN ALGORITMA *REVERSE-DELETE* DAN *ANT COLONY OPTIMIZATION* PADA JARINGAN *FIBER OPTIC*

SKRIPSI

Oleh
Rifki Ilham Baihaki
151810101052

JURUSAN MATEMATIKA
FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM
UNIVERSITAS JEMBER
2019

PERBANDINGAN ALGORITMA *REVERSE-DELETE* DAN *ANT COLONY OPTIMIZATION* PADA JARINGAN *FIBER OPTIC*

SKRIPSI

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

Oleh
Rifki Ilham Baihaki
151810101052

JURUSAN MATEMATIKA
FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM
UNIVERSITAS JEMBER
2019

PERSEMBAHAN

Puji sykur kehadirat Allah SWT saya dapat menyelesaikan skripsi yang saya persembahkan untuk:

1. Kedua orang tua saya Ayahanda Muhajir dan Ibunda Siti Muyasaroh yang telah memberikan kasih sayang, motivasi dan doa;
2. Segenap keluarga besar saya yang telah memberikan doa dan dukungan;
3. Semua guru sejak TK Al-Furqan, SD Al-Furqan, SMP N 1 Jember, SMA N 3 Jember, serta bapak ibu dosen di perguruan tinggi yang telah memberikan ilmu dan bimbingannya dengan penuh kesabaran;
4. Kakak tingkat, teman seangkatan SIGMA ‘15, serta adik tingkat yang telah memberikan semangat, motivasi dan menemani selama perkuliahan ini;
5. Teman-teman KKN 201 Maskuning Wetan yang telah menemani waktu luang;
6. HIMATIKA “Geokompstat” yang telah memberikan pengalaman dan membantu berproses selama perkuliahan ini;
7. Almamater Jurusan Matematika Fakultas Matematika dan Ilmu Pengetahuan Alam Universitas Jember.

MOTTO

"Maka apabila kamu telah selesai dari satu urusan maka kerjakanlah dengan sungguh-sungguh urusan yang lain."

(QS. Al Insyirah: 7)

"Belajar dari kesalahan dan berusaha menjadi lebih baik."

(Rifki Ilham Baihaki)

"Sabar dan ikhlas adalah kunci utama dalam kehidupan."

(Rifki Ilham Baihaki)

PERNYATAAN

Saya yang bertanda tangan di bawah ini:

Nama : Rifki Ilham Baihaki

NIM. : 151810101052

Menyatakan dengan sesungguhnya bahwa karya ilmiah yang berjudul "Perbandingan Algoritma *Reverse-Delete* dan *Ant Colony Optimization* pada Jaringan *Fiber Optic* " 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 tenyata di kemudian hari pernyataan ini tidak benar.

Jember, Juli 2019

Yang menyatakan,

Rifki Ilham Baihaki

NIM. 151810101052

SKRIPSI

PERBANDINGAN ALGORITMA *REVERSE-DELETE* DAN *ANT COLONY OPTIMIZATION* PADA JARINGAN *FIBER OPTIC*

Oleh

Rifki Ilham Baihaki

NIM. 151810101052

Pembimbing:

Dosen Pembimbing Utama : Abduh Riski, S.Si., M.Si.

Dosen Pembimbing Anggota : Kusbudiono, S.Si., M.Si.

PENGESAHAN

Skripsi berjudul "Perbandingan Algoritma *Reverse-Delete* dan *Ant Colony Optimization* pada Jaringan *Fiber Optic*" telah diuji dan disahkan pada:

Hari, tanggal :

Tempat : Fakultas Matematika dan Ilmu Pengetahuan Alam Universitas Jember.

Tim Penguji

Ketua,	Anggota I,
Abduh Riski, S.Si., M.Si.	Kusbudiono, S.Si., M.Si.
NIP. 199004062015041001	NIP. 197704302005011001
Anggota II	Anggota III
Kosala Dwidja Purnomo, S.Si., M.Si.	Prof. Drs. I Made Tirta, M.Sc., Ph.D.
NIP. 196908281998021001	NIP. 195912201985031002

Mengesahkan

Dekan Fakultas Matematika dan Ilmu Pengetahuan Alam

Universitas Jember

Drs. Sujito, Ph.D.

NIP. 196102041987111001

RINGKASAN

Perbandingan Algoritma *Reverse-Delete* dan *Ant Colony Optimization* pada Jaringan *Fiber Optic*; Rifki Ilham Baihaki, 151810101052; 2019; 76 halaman; Jurusan Matematika Fakultas Matematika dan Ilmu Pengetahuan Alam Universitas Jember.

Minimum Spanning Tree merupakan permasalahan optimasi untuk mencari jumlah bobot terkecil yang menghubungkan semua titik pada suatu jaringan. Permasalahan jaringan yang diteliti pada skripsi ini yaitu jaringan *fiber optic*. Data yang digunakan dalam penelitian ini adalah data primer yang diambil dari jaringan *fiber optic* di Perumahan Kodim Kecamatan Sukorambi Kabupaten Jember. Dalam data ini *Optical Distribution Point* (ODP) yang terpasang pada tiang diasumsikan sebagai titik dan panjang kabel antar tiang diasumsikan sebagai sisi pada graf. Jumlah titik yang diteliti adalah 35 titik.

Data yang diperoleh ini akan dicari bentuk *minimum spanning tree* menggunakan algoritma *Reverse-Delete* dan *Ant Colony Optimization*. Penelitian ini bertujuan untuk mencari panjang kabel minimum yang menghubungkan semua ODP pada jaringan. Selain itu juga untuk membandingkan hasil yang didapat berupa total panjang kabel dan lama waktu komputasi yang diperlukan.

Berdasarkan hasil dan pembahasan yang telah dilakukan dapat disimpulkan bahwa algoritma *Reverse-Delete* dan *Ant Colony Optimization* sama-sama menemukan rute terpendek dengan jarak 1317 m. Sedangkan algoritma yang efisien pada penelitian ini adalah *Ant Colony Optimization* dengan lama waktu komputasi 0,6 detik.

PRAKATA

Puji syukur kehadirat Allah SWT atas segala rahmat dan kuasa-Nya sehingga penulis dapat menyelesaikan tugas akhir yang berjudul "Perbandingan Algoritma *Reverse-Delete* dan *Ant Colony Optimization* pada Jaringan *Fiber Optic*". Penulisan tugas akhir ini disusun guna memenuhi salah satu syarat untuk menyelesaikan pendidikan strata satu (S1) pada Jurusan Matematika Fakultas Matematika dan Ilmu Pengetahuan Alam Universitas jember.

Pada kesempatan ini, dengan segala hormat penulis mengucapkan terima kasih atas bantuan dan bimbingan dalam penyusunan skripsi ini, kepada:

1. Bapak Abduh Riski, S.Si., M.Si. selaku dosen pembimbing utama dan Bapak Kusbudiono, S.Si., M.Si. selaku dosen pembimbing anggota;
2. Bapak Kosala Dwidja Purnomo, S.Si., M.Si. selaku dosen penguji I dan Bapak Prof. Drs. I Made Tirta, M.Sc., Ph.D. selaku dosen penguji II;
3. Seluruh dosen dan karyawan Jurusan Matematika FMIPA Universitas Jember;
4. Keluarga yang telah memberikan motivasi, doa, dan kasih sayang;
5. Serta semua pihak yang tidak dapat penulis sebutkan satu persatu yang telah membantu dalam penyelesaian skripsi ini.

Penulis menyadari bahwa penelitian ini masih jauh dari sempurna. Oleh karena itu penulis mengharap kritik dan saran demi kesempurnaan skripsi ini. Semoga skripsi ini dapat bermanfaat untuk kita semua.

Jember, Juli 2019

Penulis

DAFTAR ISI

DAFTAR GAMBAR

Halaman

DAFTAR TABEL

Halaman

DAFTAR LAMPIRAN

Halaman

BAB 1. PENDAHULUAN

1.1 Latar Belakang

Era modern saat ini memudahkan kita untuk bertukar informasi atau melakukan komunikasi dari jarak yang jauh sekalipun. Hal tersebut dapat dilakukan karena telah dibangun jaringan informasi dan komunikasi. Salah satu jaringan informasi dan komunikasi yang digunakan saat ini adalah jaringan *fiber optic*. Pembangunan jaringan *fiber optic* masih memunculkan permasalahan karena belum tentu jaringan yang dibangun memiliki panjang kabel minimum. Untuk memperkecil biaya yang digunakan panjang kabel *fiber optic* yang dibuat harus minimum. Permasalahan tersebut dapat diselesaikan menggunakan *minimum spanning tree*. Prinsip dari *minimum spanning tree* adalah mencari panjang kabel minimum yang menghubungkan semua titik dalam jaringan tersebut.

Algoritma jalur terpendek dapat menemukan panjang kabel minimum dari satu titik ke titik yang lain dalam jaringan dan membentuk *minimum spanning tree*. Sebuah *minimum spanning tree* adalah rentangan pohon dengan berat kurang dari atau sama dengan berat setiap rentangan pohon lainnya. Salah satu algoritma jalur terpendek dalam teori graf adalah algoritma *Reverse-Delete*. Algoritma *Reverse-Delete* adalah algoritma yang digunakan untuk mencari *minimum spanning tree* dari graf yang terhubung. Prinsip dari algoritma ini adalah dengan mengurutkan bobot dari yang paling besar dalam suatu graf, kemudian menghapus bobot terbesar di setiap sirkuit dan menyisakan sebuah graf *minimum spanning tree* (Mohanram dan Sudhakar, 2011). Selain menggunakan algoritma yang ada dalam teori graf, *minimum spanning tree* juga dapat dibentuk menggunakan metode metaheuristik.

Salah satu algoritma pencarian dalam metode metaheuristik adalah *Ant Colony Optimization* (ACO). Setiap semut dalam kawanan yang berjalan akan meninggalkan jejak *pheromone* (semacam zat kimia) di jalur yang telah dilaluinya. *Pheromone* ini menjadi sinyal bagi semut lainnya yang akan melalui jalur tersebut. Jalur terpendek akan menyisakan sinyal yang lebih kuat. Sehingga, semut berikutnya dalam memutuskan jalur yang akan dipilih biasanya akan cenderung memilih untuk mengikuti jalur dengan sinyal terkuat (Neumann dan Witt, 2010).

Mengacu dari dua metode yang berbeda dalam membentuk *minimum spanning tree*, penulis tertarik menerapkan algoritma *Reverse-Delete* dan *Ant Colony Optimization* (ACO) untuk mencari panjang minimum kabel *fiber optic*. Kedua algoritma tersebut diterapkan pada permasalahan yang sama juga untuk mencari waktu komputasi yang dihasilkan dan akan dibandingkan mana yang lebih efisien.

1.2 Rumusan Masalah

Berdasarkan latar belakang yang telah diuraikan, maka dapat dirumusan masalah bagaimana penerapan algoritma *Reverse-Delete* dan *Ant Colony Optimization* (ACO) pada permasalahan *minimum spanning tree*?

1.3 Batasan Masalah

Adapun pada masalah penelitian ini dibatasi hanya untuk meminimalkan panjang lintasan (kabel *fiber optic*) menggunakan algoritma *Reverse-Delete* dan *Ant Colony Optimization* (ACO).

1.4 Tujuan Penelitian

Berdasarkan rumusan masalah yang telah disampaikan, maka tujuan yang ingin dicapai dari penelitian ini adalah menerapkan algoritma *Reverse-Delete* dan *Ant Colony Optimization* (ACO) pada permasalahan *minimum spanning tree* khususnya jaringan *fiber optic*.

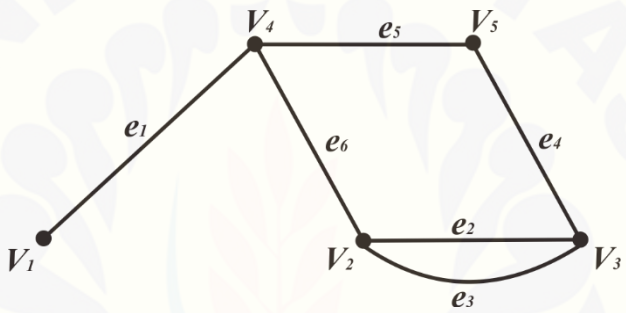
1.5 Manfaat Penelitian

Manfaat penelitian ini yaitu mengetahui penerapan algoritma *Reverse-Delete* dan *Ant Colony Optimization* (ACO) dalam membentuk *minimum spanning tree* di permasalahan jaringan khususnya jaringan *fiber optic*, juga sebagai bahan acuan atau pertimbangan dalam pemasangan jaringan *fiber optic* di Perumahan Kodim Kecamatan Sukorambi Kabupaten Jember agar memiliki panjang yang minimal.

BAB 2. TINJAUAN PUSTAKA

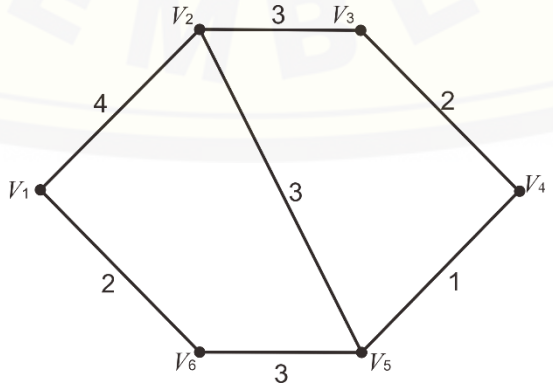
2.1 Graf

Graf G adalah pasangan (v, e) dimana v adalah himpunan tak kosong yang anggotanya disebut dengan titik (*vertex*) dan e adalah himpunan yang anggotanya tak terurut dari *vertex* v yang disebut dengan sisi (*edges*). Definisi graf tersebut bisa kita nyatakan bahwa v tidak boleh kosong, sedangkan e boleh kosong. Sebuah graf dimungkinkan tidak memiliki sisi, tetapi minimal harus memiliki satu titik (Vasudev, 2006). Gambar 2.1 merupakan contoh graf. Pada graf tersebut memiliki 5 titik dan 6 sisi.



Gambar 2.1 Graf G dengan 5 titik 6 sisi

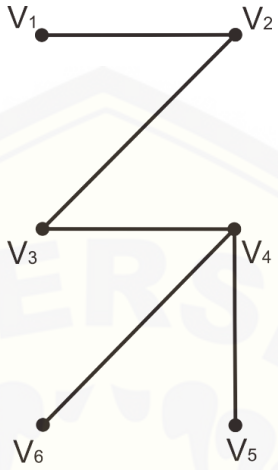
Graf memiliki beberapa kelas, salah satunya adalah graf berbobot. Graf berbobot merupakan graf yang memiliki bilangan disetiap sisinya. Bilangan ini dapat menyatakan jarak, biaya perjalanan, waktu tempuh, dan sebagainya. Graf berbobot juga biasa digunakan untuk memodelkan suatu jaringan (Rosen, 2012). Gambar 2.2 merupakan contoh dari graf berbobot. Pada graf tersebut masing-masing sisi memiliki bobot yang bervariasi.



Gambar 2.2 Graf berbobot (Sumber: Rosen, 2012)

2.2 Pohon (*Tree*)

Graf pohon adalah graf terhubung yang tidak berarah dan tidak memiliki sirkuit (*cycle*). Gambar 2.3 merupakan contoh dari graf pohon.



Gambar 2.3 Graf pohon (Sumber: Purwanto dkk., 2006)

Pada Gambar 2.3 titik-titik v_1, v_5, v_6 dinamakan daun sedangkan titik-titik v_2, v_3, v_4 dinamakan titik cabang (Purwanto dkk., 2006).

Sebuah *minimum spanning tree* dalam graf terboboti memiliki kemungkinan jumlah terkecil dari bobot pada sisinya. Pada permasalahan jaringan, *minimum spanning tree* dapat digunakan sebagai acuan dalam mengoptimalkan jaringan karena jumlah bobot sisinya yang minimum. Untuk membangun *minimum spanning tree* digunakan beberapa algoritma jalur terpendek.

2.3 Algoritma *Reverse-Delete*

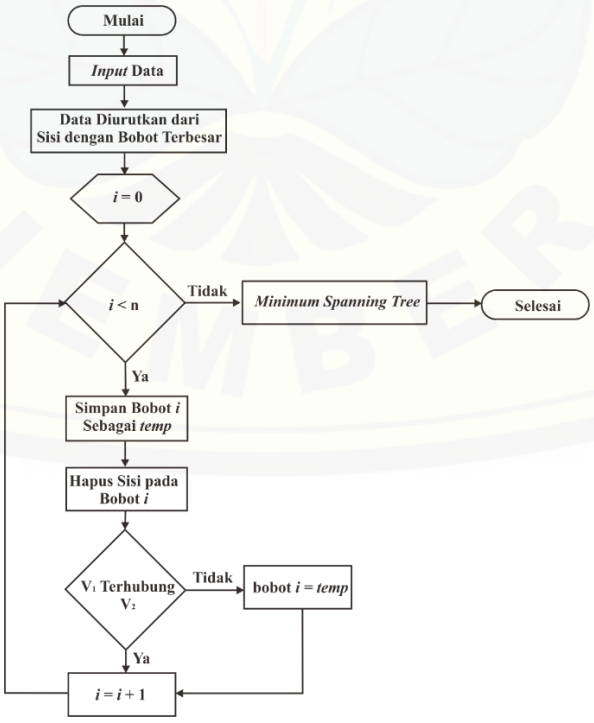
Algoritma *Reverse-Delete* adalah suatu algoritma dalam teori graf yang digunakan untuk mendapatkan *minimum spanning tree* dari suatu graf terhubung. Jika graf tersebut terputus, algoritma ini akan menemukan *minimum spanning tree*. Algoritma ini merupakan algoritma serakah, memilih pilihan terbaik yang diberikan setiap situasinya. Algoritma ini juga kebalikan dari algoritma *Kruskal*. Jika dalam algoritma *Kruskal* dipilih sebuah sisi pada graf dengan bobot terkecil hingga bobot terbesar. Sedangkan pada algoritma *Reverse-Delete* dipilih sisi pada graf dari bobot terbesar hingga bobot terkecil. Kemudian sisi dengan bobot terbesar yang

membentuk sirkuit dihapus hingga membentuk *minimum spanning tree* (Mohanram dan Sudhakar, 2011).

Berikut adalah langkah-langkah yang dilakukan oleh *Reverse-Delete* dalam mendapatkan *minimum spanning tree* adalah sebagai berikut:

a. Data dikonversi dari graf yang terboboti;
b. Data yang telah dikonversi selanjutnya dirutkan bobotnya dimulai dari sisi dengan bobot terbesar;
c. Inisialisasi indeks $i = 0$;
d. Jika indeks $i < n$ maka sisi dengan bobot i akan disimpan sebagai temp;
e. Sisi dengan bobot i yang telah disimpan kemudian dihapus;
f. Jika V_1 terhubung V_2 maka kembali ke langkah d, dengan $i = i + 1$;
g. Jika V_1 tidak terhubung V_2 maka bobot i = *temp*, kemudian kembali ke langkah f;
h. Jika indeks $i = n$ maka akan membentuk *minimum spanning tree* dan program selesai dijalankan.

Gambar 2.4 merupakan skema algoritma *Reverse-Delete* dari langkah-langkah yang sudah dijelaskan sebelumnya.

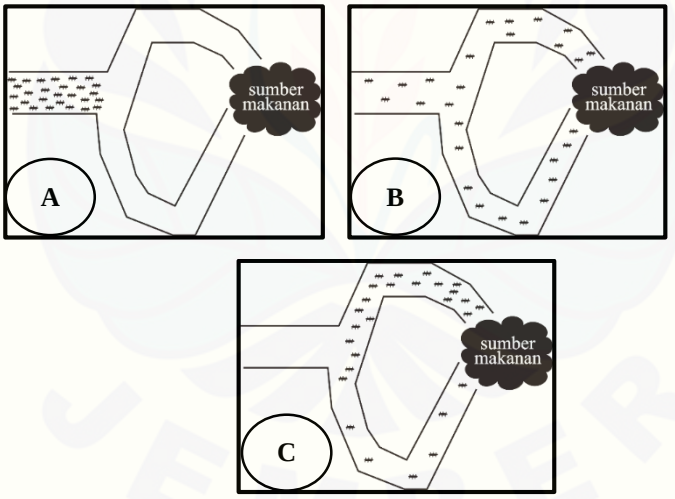


Gambar 2.4 Skema algoritma *Reverse-Delete*

2.4 Algoritma Semut (*Ant Colony Optimization*)

Algoritma semut diperkenalkan oleh Moyson dan Manderick dan secara luas telah dikembangkan oleh Marco Dorigo. Algoritma ini menggunakan teknik probabilistik untuk menyelesaikan masalah komputasi dengan menemukan rute terbaik. Algoritma ini diambil dengan analogi oleh perilaku semut dalam menemukan rute dari koloninya menuju sumber makanan (Elbeltagi E. dkk., 2005). Koloni semut dapat menemukan rute terpendek antara sarang dengan sumber makanan berdasarkan jejak *pheromone* yang ditimbulkan oleh koloni semut pada lintasan yang telah dilalui. Semakin banyak semut yang melalui suatu lintasan, maka akan semakin jelas jejak *pheromone*nya. Hal inilah yang akan membuat semut lainnya mengikuti lintasan tersebut dan menemukan rute terpendek (Karjono dkk., 2016).

Berikut pada gambar 2.5 merupakan ilustrasi dari algoritma *Ant Colony Optimization*:



Gambar 2.5 Ilustrasi *ant colony optimization*

Pada Gambar 2.5 (A) terdapat sebuah koloni semut yang hendak menuju sumber makanan tetapi harus melewati dua jalur, yaitu jalur atas dan jalur bawah.

Pada Gambar 2.5 (B) koloni semut terpisah menjadi dua bagian. Ada yang melewati jalur yang atas dan ada yang melewati jalur yang bawah. Kecepatan intenstitas jejak kaki semut yang sama akan meninggalkan *pheromone* dijalur yang dilaluinya.

Pada Gambar 2.5 (C) *pheromone* yang dilalui semut bagian atas penguapannya lebih lama daripada semut bagian bawah karena jarak tempuhnya lebih pendek. Semut lainnya akhirnya memutuskan untuk melewati jalur atas karena sinyal pheromone yang lebih kuat. Semakin lama jumlah semut dibagian bawah akan berkurang dan bahkan hilang karena sinyal pheromone yang ditinggalkan lemah. Akibatnya terbentuk jalur terpendek yang dilalui semut menuju sumber makanan, yaitu jalur atas. Berdasarkan ilustrasi *Ant Colony Optimization* yang telah dibuat kemudian dibuat langkah-langkah pembentukan minimum spanning tree.

a. Input parameter dalam algoritma ini adalah:

   m : Banyak semut

   α : Tetapan pengendali *pheromone*

   β : Tetapan pengendali jarak

   ρ : Tetapan penguapan *pheromone* $(0-1)$

   t : Bilangan iterasi

   T : Bilangan iterasi maksimal.

b. Inisialisasi *pheromone* awal $(\tau_{(u,v)} = \frac{Q}{Bobot\ sisi\ terbesar})$

c. Inisialisasi indeks semut $(k = 1)$.

d. Semua sisi dimasukkan kedalam *list*.

e. Semut akan memilih sisi yang dilaluinya pertama secara acak.

$$P_{(u,v)} = [\tau_{(u,v)}]^{\alpha} \times [\eta_{(u,v)}]^{\beta} \quad (2.1)$$

dimana:

$P_{(u,v)}$: Probabilitas sisi yang akan dikunjungi

$\tau_{(u,v)}$: Intensitas *pheromone*

$\eta_{(u,v)}$: Visibilitas jarak sisi $\left(\frac{1}{C_{(u,v)}}\right)$

Selanjutnya menghitung probabilitas relatif setiap sisi yang akan dilalui.

$$Pr_{(u,v)} = \frac{P_{(u,v)}}{\sum[\tau_{(u,v)}]^{\alpha} \times [\eta_{(u,v)}]^{\beta}} \quad (2.2)$$

dimana:

$Pr_{(u,v)}$: Probabilitas relatif sisi yang akan dikunjungi

Selanjutnya menghitung probabilitas komulatif setiap sisi yang akan dilalui.

$$Pk_{(u,v)} = \sum Pr_{(u,v)} \quad (2.3)$$

$Pk_{(u,v)}$: Probabilitas komulatif sisi yang akan dikunjungi

Terakhir, membuat interval untuk memilih sisi yang akan dilalui. Pemilihan sisi diacak menggunakan *roulete wheel*.

f. Titik dan sisi yang dimasukkan dalam himpunan menggunakan aturan: jika terdapat titik yang termuat dalam sub himpunan sebelumnya maka sisi ditambahkan. Jika titik yang termuat tidak terdapat dalam sub himpunan sebelumnya maka sisi dimasukkan dalam sub himpunan baru.

g. Sub himpunan akan membentuk spanning tree dan sisi yang terpilih dihapus dari *list*.

h. Ulangi langkah (f) sampai (g) jika semua titik belum masuk sub himpunan atau terdapat lebih dari satu sub himpunan.

i. Ulangi langkah (d) sampai (h) jika indeks semut (k) belum sama dengan jumlah semut (m).

j. Perjalanan semut antar titik akan meninggalkan jejak *pheromone* dilintasan yang dilaluinya. Adanya penguapan menyebabkan kemungkinan terjadinya perubahan pheromone.

$$\Delta\tau_{(u,v)} = \begin{cases} \frac{Q}{C_{(u,v)}}, & \text{jika sisi (u, v) masuk spanning tree} \\ 0 \ , & \text{jika sisi (u, v) tidak masuk} \end{cases} \quad (2.4)$$

$\Delta\tau^{k}{}_{(u,v)}$ adalah perubahan pheromone pada sisi yang telah dilalui semut.

$$\Delta\tau^{k}{}_{(u,v)} = \frac{Q}{C_{(u,v)}} \quad (2.5)$$

dimana:

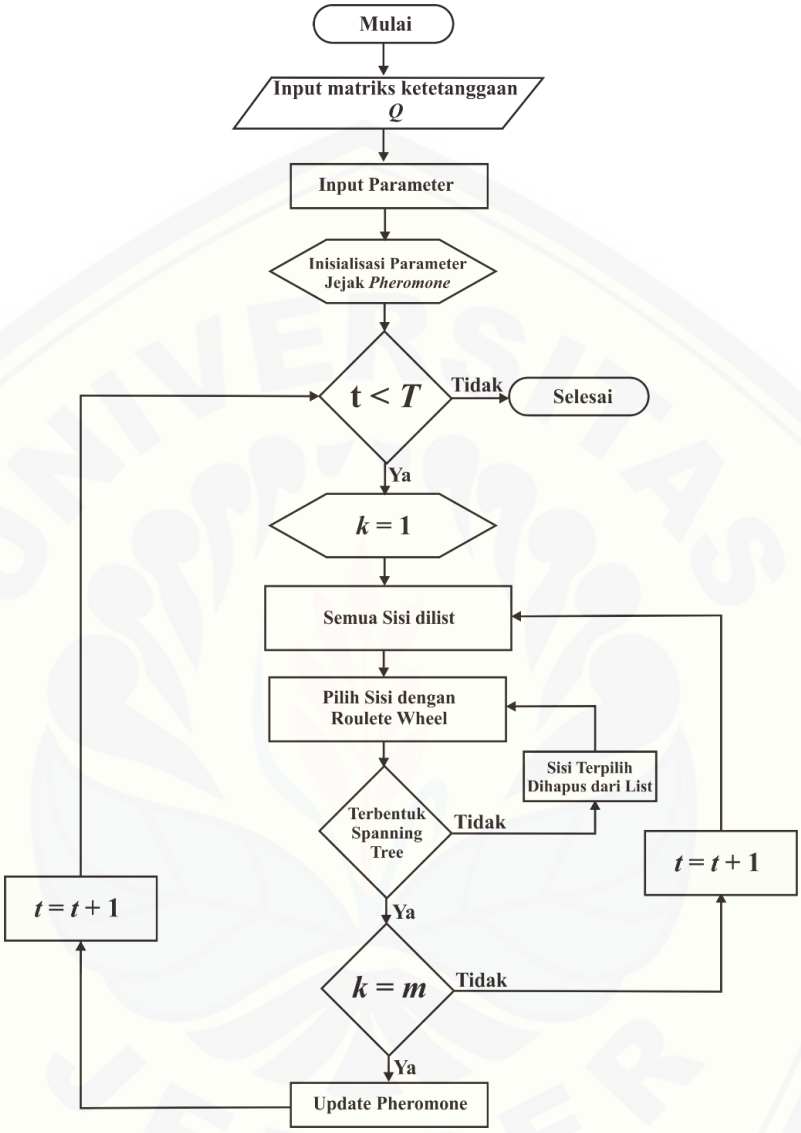
Q : Harga kabel tiap meter = Rp. 2500.

k. Perubahan pheromone dalam satu iterasi dilintasan yang telah dilalui setiap semut kemudian di*update*.

$$\tau_{(u,v)}(t) = ((1-\rho) \times \tau_{(u,v)}(t-1)) + \Delta\tau_{(u,v)};\ t = 1, 2, \dots, T \quad (2.6)$$

l. Ulangi langkah (c) sampai (k) jika t belum sama dengan T.

Gambar 2.6 merupakan skema dari *Ant Colony Optimization* dalam membentuk *minimum spanning tree*.



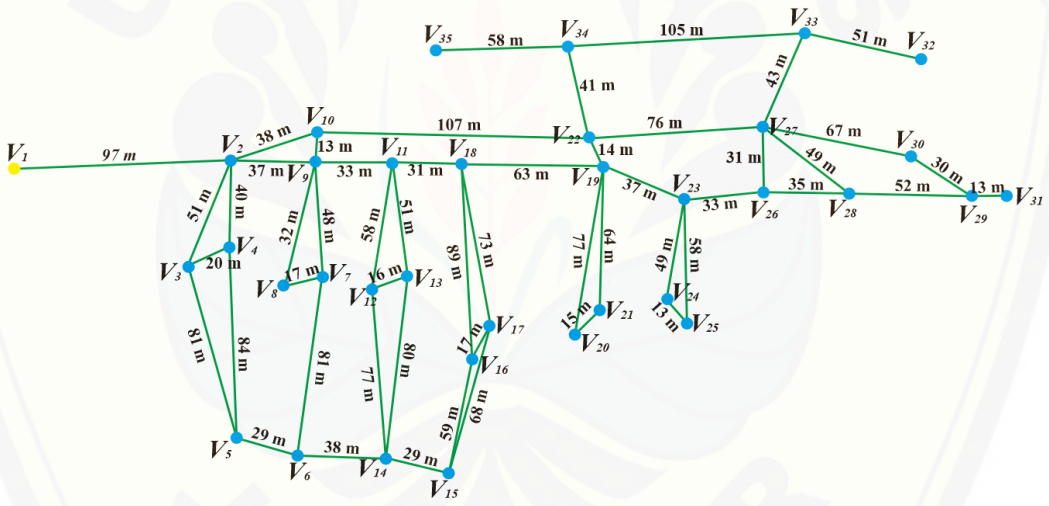
Gambar 2.6 Skema *Ant Colony Optimization*

BAB 3. METODE PENELITIAN

Pada bab ini akan dijelaskan mengenai prosedur untuk mendapatkan data dan langkah penelitian yang akan dilakukan.

3.1 Data Penelitian

Data penelitian ini merupakan data primer yang diambil di Perumahan Kodim Kecamatan Sukorambi Kabupaten Jember. Dalam data ini *Optical Distribution Center* (ODC) dan *Optical Distribution Point* (ODP) yang terpasang pada tiang disebut titik pada graf. Panjang kabel disebut sebagai sisi pada graf. Gambar 3.1 merupakan peta jaringan *fiber optic* di Perumahan Kodim.



Gambar 3.1 Peta jaringan *fiber optic*

3.2 Langkah-langkah Penelitian

Penelitian menerapkan algoritma *Reverse-Delete* dan *Ant Colony Optimization* (ACO) untuk menganalisis jaringan *fiber optic* dilakukan dengan langkah-langkah sebagai berikut:

a. Studi Literatur

Langkah awal dilakukan dengan mencari dan mengumpulkan informasi mengenai algoritma *Reverse-Delete*, *Ant Colony Optimization* (ACO), dan permasalahan *minimum spanning tree* pada jurnal internasional dan buku-buku yang berkaitan dengan tiga hal tersebut;

b. Pengambilan Data

Proses pengambilan data ini yaitu dengan melihat langsung panjang kabel *fiber optic* serta koordinat tiang pada *global positioning system* (GPS) Perumahan Kodim. Kemudian dari data yang didapatkan tadi dilakukan diskusi dengan pihak Telkom untuk membuat peta jaringan *fiber optic*;

c. Pembuatan Peta Jaringan

Proses pembuatan peta jaringan *fiber optic* adalah dengan cara menentukan titik yang diumpamakan sebagai ODP pada tiang dan panjang kabel yang menghubungkan satu tiang dengan tiang yang lain. Bentuk peta jaringan ini seperti graf yang berbobot;

d. Penerapan algoritma *Reverse Delete* dan *Ant Colony Optimization*

Langkah selanjutnya yaitu dengan menerapkan algoritma *Reverse-Delete* dan *Ant Colony Optimization* (ACO) untuk mencari *minimum spanning tree* dari peta jaringan *fiber optic* yang telah dibuat sebelumnya. Langkah pembentukan *minimum spanning tree* algoritma *Reverse-Delete* dan *Ant Colony Optimization* (ACO) seperti yang sudah dijelaskan pada sub bab 2.3 dan sub bab 2.4;

e. Percobaan Program

Program algoritma *Reverse-Delete* dan *Ant Colony Optimization* (ACO) dibuat dengan menggunakan *software* MATLAB R2015b. Pada langkah ini penulis membuat skrip program dan dari program tersebut dilakukan percobaan. Tabel 3.1 merupakan rencana parameter yang akan digunakan untuk percobaan *Ant Colony Optimization* (ACO);

Tabel 3.1 Rencana parameter ACO yang akan digunakan

No.	Parameter	Nilai
1	Tetapan pengendali *pheromone* (α)	2
2	Tetapan pengendali jarak (β)	4
3	Tetapan penguapan jejak semut (ρ)	0,5
4	Iterasi maksimal (T)	5
5	Harga kabel *fiber optic* tiap meter (Q)	Rp. 2500
6	Jumlah semut (m)	5

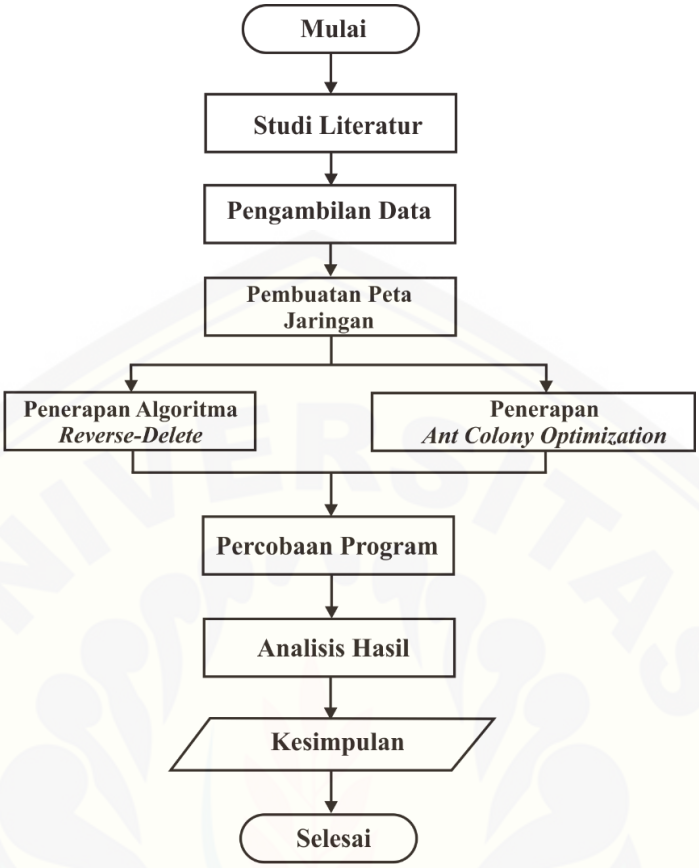
f. Analisis Hasil

Program algoritma *Reverse-Delete* dan *Ant Colony Optimization* (ACO) akan mendapatkan rute *minimum spanning tree* dari peta jaringan *fiber optic*. Kemudian dilakukan analisis dengan melihat total jarak dari masing-masing algoritma dan waktu komputasinya;

g. Kesimpulan

Rute *minimum spanning tree* yang telah diperoleh kemudian dijadikan acuan pemasangan jaringan *fiber optic* di Perumahan Kodim. Serta waktu komputasi kedua algoritma dibandingkan dan disimpulkan mana yang lebih cepat dan stabil dalam menemukan rute *minimum spanning tree*.

Gambar 3.1 adalah skema penelitian yang dilakukan untuk menganalisis jaringan *fiber optic* di Perumahan Kodim.



Gambar 3.2 Skema Penelitian

BAB. 5 PENUTUP

5.1 Kesimpulan

Berdasarkan hasil dan pembahasan pada bab sebelumnya, dapat disimpulkan bahwa algoritma *Reverse-Delete* dan *Ant Colony Optimization* (ACO) dapat diterapkan pada jaringan *fiber optic*. Pada algoritma *Reverse-Delete* lama waktu komputasi yang dibutuhkan dipengaruhi oleh bentuk jaringannya. Semakin rumit jaringan tersebut maka akan semakin lama algoritma *Reverse-Delete* bekerja. Hal ini berbeda dengan ACO. Meskipun jaringannya rumit, tapi waktu komputasi ACO masih lebih cepat. Tapi tentunya untuk mempercepat waktu komputasi ACO perlu diperhatikan parameter yang akan digunakan (m = 5; α = 0,1, β = 17, ρ = 0,5; dan T = 5). Berikut adalah rute terpendek yang dihasilkan kedua algoritma tersebut:

Rute : (V_1,V_2), (V_2,V_9), (V_2,V_3), (V_3,V_4), (V_9,V_{10}), (V_8,V_9), (V_8,V_7), (V_6,V_7), (V_5,V_6), (V_9,V_{11}), (V_{11},V_{13}), (V_{12},V_{13}), (V_{11},V_{18}), (V_{17},V_{18}), (V_{16},V_{17}), (V_{15},V_{16}), (V_{14},V_{15}), (V_{18},V_{19}), (V_{19},V_{21}), (V_{19},V_{22}), (V_{19},V_{23}), (V_{20},V_{21}), (V_{23},V_{24}), (V_{24},V_{25}), (V_{23},V_{26}), (V_{26},V_{27}), (V_{26},V_{28}), (V_{28},V_{29}), (V_{29},V_{30}), (V_{29},V_{31}), (V_{27},V_{33}), (V_{32},V_{33}), (V_{22},V_{34}), (V_{34},V_{35})

Total jarak yang dihasilkan dari rute tersebut adalah 1317 m. Berdasarkan rute yang telah dihasilkan dapat diperoleh biaya kabel *fiber optic* yang diperlukan adalah Rp. 3.292.500.

5.2 Saran

Penelitian selanjutnya diharapkan dapat menyelesaikan permasalahan jaringan *fiber optic* menggunakan metode optimasi yang berbeda yaitu TSP (*Travelling Salesman Problem*). Menggunakan algoritma lain yang lebih optimal. Juga melakukan *power link budget* untuk melihat pengaruh jaringan tersebut yang diminimalkan dengan aliran transmisi data.

DAFTAR PUSTAKA

Afrianto, I., dan Jamilah E. W. 2012. *Penyelesaian Masalah Minimum Spanning Tree Menggunakan Ant Colony System*. I(2) : 35-40.

Chartrand, G., Lesniak L. dan Zhang P. 2011. *Graphs and Disgraphs, 5th Ed*. California : Chapman & Hall.

Elbeltagi E., Hegazy T., dan Grierson D. 2005. Comparison Among Five Evolutionary-based Optimization Algorithms. *Advanced Engineering Informatics*. 45-43.

Harary, F. 1994. *Graph Teory*. Reading, MA : Addison-Wesley.

Karjono, Moedjiono, dan D. Kurniawan. 2016. Ant Colony Optimization. *Jurnal TICOM*. 4(3) : 119-125.

Mohanram, S. dan T. D. Sudhhakar. 2011. Power System Restoration using Reverse Delete Algorithm Implemented in FGPA. *Chennai and Dr. MGR University Second International Conference on Sustainable Energy and Intelligent System*. 373-378.

Neumann, F. dan C. Witt. 2010. Ant Colony Optimazation and the Minimum Spanning Tree Problem. *Theoritical Computer Science*. 411: 2406-2413.

Purwanto, H., G. Indriyani dan E. Dayanti. 2006. *Matematika Diskrit*. Jakarta : PT. Ercontara Rajawali.

Rosen, K. H. 2012. *Discreate Mathemathics and Its Application, 7th Ed*. New York: VAGA.

Vasudev, C. 2006. *Graph Theory With Application*. New Delhi : New Age International (P), Ltd.

LAMPIRAN

LAMPIRAN 1. Data Koordinat Titik

Titik	Latitude	Longitude
1	-8,19356	113,642309
2	-8,192641	113,641974
3	-8,192591	113,64239
4	-8,19252	113,642404
5	-8,192254	113,643286
6	-8,191969	113,643204
7	-8,192217	113,642278
8	-8,192276	113,64233
9	-8,192316	113,641896
10	-8,192401	113,641806
11	-8,19202	113,641784
12	-8,191962	113,642243
13	-8,191926	113,642168
14	-8,191663	113,643043
15	-8,191417	113,642955
16	-8,19165	113,642174
17	-8,191591	113,642116
18	-8,191789	113,641708

Titik	Latitude	Longitude
19	-8,191408	113,641542
20	-8,191317	113,64216
21	-8,191245	113,642128
22	-8,191459	113,641453
23	-8,191084	113,641494
24	-8,191001	113,642019
25	-8,190923	113,642054
26	-8,190639	113,64117
27	-8,190358	113,640973
28	-8,189902	113,641079
29	-8,189166	113,641089
30	-8,189099	113,641
31	-8,188946	113,641065
32	-8,189676	113,640555
33	-8,190344	113,64059
34	-8,191496	113,641024
35	-8,191972	113,641228

LAMPIRAN 2. Data Jarak Antar Titik

Jarak	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
1	0	97	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	97	0	51	40	0	0	0	0	37	38	0	0	0	0	0	0	0	0	0	0	0	0	0
3	0	51	0	20	81	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4	0	40	20	0	84	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
5	0	0	81	84	0	29	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
6	0	0	0	0	29	0	81	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
7	0	0	0	0	0	81	0	17	48	0	0	0	0	0	0	0	0	0	0	0	0	0	0
8	0	0	0	0	0	0	17	0	32	0	0	0	0	0	0	0	0	0	0	0	0	0	0
9	0	37	0	0	0	0	48	32	0	13	33	0	0	0	0	0	0	0	0	0	0	0	0
10	0	38	0	0	0	0	0	0	13	0	0	0	0	0	0	0	0	0	0	0	0	107	0
11	0	0	0	0	0	0	0	0	33	0	0	58	51	0	0	0	0	31	0	0	0	0	0
12	0	0	0	0	0	0	0	0	0	0	58	0	16	77	0	0	0	0	0	0	0	0	0
13	0	0	0	0	0	0	0	0	0	0	51	16	0	80	0	0	0	0	0	0	0	0	0
14	0	0	0	0	0	0	0	0	0	0	0	77	80	0	29	0	0	0	0	0	0	0	0
15	0	0	0	0	0	0	0	0	0	0	0	0	0	29	0	59	68	0	0	0	0	0	0
16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	59	0	17	89	0	0	0	0	0
17	0	0	0	0	0	0	0	0	0	0	0	0	0	0	68	17	0	73	0	0	0	0	0
18	0	0	0	0	0	0	0	0	0	0	31	0	0	0	0	89	73	0	63	0	0	0	0
19	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	63	0	77	64	14	37
20	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	77	0	15	0	0

Jarak	24	25	26	27	28	29	30	31	32	33	34	35
1	0	0	0	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	0	0	0	0	0
8	0	0	0	0	0	0	0	0	0	0	0	0
9	0	0	0	0	0	0	0	0	0	0	0	0
10	0	0	0	0	0	0	0	0	0	0	0	0
11	0	0	0	0	0	0	0	0	0	0	0	0
12	0	0	0	0	0	0	0	0	0	0	0	0
13	0	0	0	0	0	0	0	0	0	0	0	0
14	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
16	0	0	0	0	0	0	0	0	0	0	0	0
17	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
19	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

Jarak	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
21	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	64	15	0	0	0
22	0	0	0	0	0	0	0	0	0	107	0	0	0	0	0	0	0	0	14	0	0	0	0
23	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	37	0	0	0	0
24	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	49
25	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	58
26	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	33
27	0	0	0	0	0	0	0	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	0	0	0	0	0	0	0
29	0	0	0	0	0	0	0	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	0	0	0	0	0	0	0
31	0	0	0	0	0	0	0	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	0	0	0	0	0	0	0
33	0	0	0	0	0	0	0	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	0	0	0	0	0	41	0
35	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Jarak	24	25	26	27	28	29	30	31	32	33	34	35
21	0	0	0	0	0	0	0	0	0	0	0	0
22	0	0	0	0	0	0	0	0	0	0	41	0
23	49	58	33	0	0	0	0	0	0	0	0	0
24	0	13	0	0	0	0	0	0	0	0	0	0
25	13	0	0	0	0	0	0	0	0	0	0	0
26	0	0	0	31	35	0	0	0	0	0	0	0
27	0	0	31	0	49	0	67	0	0	43	0	0
28	0	0	35	49	0	52	0	0	0	0	0	0
29	0	0	0	0	52	0	30	13	0	0	0	0
30	0	0	0	67	0	30	0	0	0	0	0	0
31	0	0	0	0	0	13	0	0	0	0	0	0
32	0	0	0	0	0	0	0	0	0	51	0	0
33	0	0	0	43	0	0	0	0	51	0	105	0
34	0	0	0	0	0	0	0	0	0	105	0	58
35	0	0	0	0	0	0	0	0	0	0	58	0

LAMPIRAN 3. Skrip Utama algoritma Reverse-Delete dan ACO

```
% Update handles structure
guidata(hObject, handles);
clc;
movegui(gcf,'center');
set(gcf,'Name','Rifki Program');
set(handles.uitable1,'data',[],'userdata',[],'rowname',
1:20,'columnname',{'latitude','longitude'});
set(handles.uitable2,'data',[],'userdata',[],'rowname',
1:20,'columnname',1:20);
set(handles.edit1,'string','');
set(handles.edit2,'string','');
set(handles.edit3,'string','');
set(handles.edit4,'string','');
set(handles.edit5,'string','');
set(handles.edit6,'string','');
set(handles.radiobutton1,'value',0);
set(handles.radiobutton2,'value',0,'userdata',[]);
set(handles.radiobutton3,'value',0,'userdata',[]);
cla(handles.axes1,'reset');
set(handles.axes1,'XTick',[],'YTick',[]);
set(handles.text8,'string','Total Jarak Reverse-Delete
= 0 m');
set(handles.text9,'string','Total Jarak ACO = 0 m');
set(handles.text10,'string','Waktu Kom. Reverse-Delete
= 0 detik');
set(handles.text11,'string','Waktu Kom. ACO = 0
detik');
cla(handles.axes2,'reset');
axes(handles.axes2);
ylabel('Total Jarak');
set(handles.axes2,'FontSize',8,'FontWeight','bold');
set(handles.text12,'string','Konvergen saat iterasi ke-
');
% UIWAIT makes RifkiProgram wait for user response (see
UIRESUME)
% uiwait(handles.figure1);

% --- Executes on button press in pushbutton1.
function pushbutton1_Callback(hObject, eventdata,
handles)
% hObject    handle to pushbutton1 (see GCBO)
% eventdata  reserved - to be defined in a future
version of MATLAB
```

```
% handles    structure with handles and user data (see
GUIDATA)
clc;
[File, Path]=uigetfile({'*.xlsx;*.xls','Excel
Spreadsheet (*.xlsx, *xls)'},'Open');
if File~=0
    [Coor, txt]=xlsread(fullfile(Path,File),1);
    node=size(Coor,1);

set(handles.uitable1,'data',Coor(:,2:3),'rowname',1:nod
e);
    [Dist, txt]=xlsread(fullfile(Path,File),2);

set(handles.uitable2,'data',Dist(2:end,2:end),'rowname'
,1:node,'columnname',1:node);
    set(gcf,'Name',['Rifki Program - '
fullfile(Path,File)]);
end

% --- Executes on button press in pushbutton2.
function pushbutton2_Callback(hObject, eventdata,
handles)
% hObject    handle to pushbutton2 (see GCBO)
% eventdata  reserved - to be defined in a future
version of MATLAB
% handles    structure with handles and user data (see
GUIDATA)
clc;
movegui(gcf,'center');
set(gcf,'Name','Rifki Program');
set(handles.uitable1,'data',[],'userdata',[],'rowname',
1:20,'columnname',{'latitude','longitude'});
set(handles.uitable2,'data',[],'userdata',[],'rowname',
1:20,'columnname',1:20);
set(handles.edit1,'string','');
set(handles.edit2,'string','');
set(handles.edit3,'string','');
set(handles.edit4,'string','');
set(handles.edit5,'string','');
set(handles.edit6,'string','');
set(handles.radiobutton1,'value',0);
set(handles.radiobutton2,'value',0,'userdata',[]);
set(handles.radiobutton3,'value',0,'userdata',[]);
cla(handles.axes1,'reset');
set(handles.axes1,'XTick',[],'YTick',[]);
set(handles.text8,'string','Total Jarak Reverse-Delete
= 0 m');
```

```
set(handles.text9,'string','Total Jarak ACO = 0 m');
set(handles.text10,'string','Waktu Kom. Reverse-Delete
= 0 detik');
set(handles.text11,'string','Waktu Kom. ACO = 0
detik');
cla(handles.axes2,'reset');
axes(handles.axes2);
ylabel('Total Jarak');
set(handles.axes2,'FontSize',8,'FontWeight','bold');
set(handles.text12,'string','Konvergen saat iterasi ke-
');

% --- Executes on button press in pushbutton3.
function pushbutton3_Callback(hObject, eventdata,
handles)
% hObject    handle to pushbutton3 (see GCBO)
% eventdata  reserved - to be defined in a future
version of MATLAB
% handles    structure with handles and user data (see
GUIDATA)
clc;

DistData=get(handles.uitable2,'data');
if ~isempty(DistData)
    tic;
    set(handles.radiobutton1,'value',0);
    set(handles.radiobutton2,'value',0,'userdata',[]);
    set(handles.radiobutton3,'value',0);
    cla(handles.axes1,'reset');
    set(handles.axes1,'XTick',[],'YTick',[]);
    set(handles.text8,'string','Total Jarak Reverse-
Delete = 0 m');
    set(handles.text10,'string','Waktu Kom. Reverse-
Delete = 0 detik');
    pause(0.001);

    node=size(DistData,1); %banyak titik

    %Cari sisi
    a=1;
    for i=1:node
        for j=i:node
            if DistData(i,j)~=0
                Data(a,:)=[i,j,DistData(i,j)];
                a=a+1;
            end
        end
```

```
    end

    %Sorting bobot sisi
    [urut, ind]=sort(Data(:,3),'descend');
    Data=Data(ind,:);

    %Delete
    Dist=DistData;
    edge=size(Data,1); %banyak sisi
    for i=1:edge
        Dist(Data(i,1),Data(i,2))=0;
        Dist(Data(i,2),Data(i,1))=0;
        %cek koneksi
        connect=1;
        ss=1;
        for j=1:node
            if j~=Data(i,1)

[dist(ss),path,pred]=graphshortestpath(sparse(Dist),Dat
a(i,1),j);
                ss=ss+1;
            end
        end
        if sum(dist)==inf
            connect=0;
        end
        ss=1; dist=[];
        for j=1:node
            if j~=Data(i,2)

[dist(ss),path,pred]=graphshortestpath(sparse(Dist),Dat
a(i,2),j);
                ss=ss+1;
            end
        end
        if sum(dist)==inf
            connect=0;
        end
        if connect==0 %jika tidak terhubung
            Dist(Data(i,1),Data(i,2))=Data(i,3);
            Dist(Data(i,2),Data(i,1))=Data(i,3);
        end
    end

    %Plot
    set(handles.radiobutton1,'value',0);
```

```
set(handles.radiobutton2,'value',1,'userdata',Dist);
    set(handles.radiobutton3,'value',0);
    CoorData=get(handles.uitable1,'data');
    Coor=[CoorData(:,2) CoorData(:,1)];
    axes(handles.axes1);
    gplot(Dist,Coor,'b');
    inlat=max(Coor(:,2))-min(Coor(:,2));
    inlon=max(Coor(:,1))-min(Coor(:,1));
    for i=1:node

line(Coor(i,1),Coor(i,2),'Marker','s','MarkerEdgeColor'
,'k','MarkerFaceColor','r','MarkerSize',5);
        text(Coor(i,1),Coor(i,2),sprintf(['V'
num2str(i)]));
    end
    set(handles.axes1,'Xlim',[min(Coor(:,1))-inlon/10
max(Coor(:,1))+inlon/10],...
        'Ylim',[min(Coor(:,2))-inlat/10
max(Coor(:,2))+inlat/10],...
        'XTick',[],'YTick',[]);

    %jarak & waktu
    jarak=sum(sum(Dist))/2;
    waktu=toc;
    set(handles.text8,'string',['Total Jarak Reverse-
Delete = ' num2str(jarak) ' m']);
    set(handles.text10,'string',['Waktu Kom. Reverse-
Delete = ' num2str(waktu) ' detik']);
end

function edit1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit1 (see GCBO)
% eventdata  reserved - to be defined in a future
version of MATLAB
% handles    empty - handles not created until after
all CreateFcns called

% Hint: edit controls usually have a white background
on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
```

```
% --- Executes during object creation, after setting
all properties.
function edit2_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit2 (see GCBO)
% eventdata  reserved - to be defined in a future
version of MATLAB
% handles    empty - handles not created until after
all CreateFcns called

% Hint: edit controls usually have a white background
on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes during object creation, after setting
all properties.
function edit3_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit3 (see GCBO)
% eventdata  reserved - to be defined in a future
version of MATLAB
% handles    empty - handles not created until after
all CreateFcns called

% Hint: edit controls usually have a white background
on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes during object creation, after setting
all properties.
function edit4_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit4 (see GCBO)
% eventdata  reserved - to be defined in a future
version of MATLAB
% handles    empty - handles not created until after
all CreateFcns called

% Hint: edit controls usually have a white background
on Windows.
%       See ISPC and COMPUTER.
```

```
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes during object creation, after setting
all properties.
function edit5_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit5 (see GCBO)
% eventdata  reserved - to be defined in a future
version of MATLAB
% handles    empty - handles not created until after
all CreateFcns called

% Hint: edit controls usually have a white background
on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes during object creation, after setting
all properties.
function edit6_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit6 (see GCBO)
% eventdata  reserved - to be defined in a future
version of MATLAB
% handles    empty - handles not created until after
all CreateFcns called

% Hint: edit controls usually have a white background
on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes on button press in pushbutton4.
function pushbutton4_Callback(hObject, eventdata,
handles)
% hObject    handle to pushbutton4 (see GCBO)
% eventdata  reserved - to be defined in a future
version of MATLAB
% handles    structure with handles and user data (see
GUIDATA)
```

```
clc;

DistData=get(handles.uitable2,'data');
if ~isempty(DistData)
    tic;
    set(handles.radiobutton1,'value',0);
    set(handles.radiobutton2,'value',0);
    set(handles.radiobutton3,'value',0,'userdata',[]);
    cla(handles.axes1,'reset');
    set(handles.axes1,'XTick',[],'YTick',[]);
    set(handles.text9,'string','Total Jarak ACO = 0
m');
    set(handles.text11,'string','Waktu Kom. ACO = 0
detik');
    cla(handles.axes2,'reset');
    axes(handles.axes2);
    ylabel('Total Jarak');

set(handles.axes1,'FontSize',8,'FontWeight','bold');
    set(handles.text12,'string','Konvergen saat iterasi
ke-');
    pause(0.001);

    node=size(DistData,1); %banyak titik

    %Parameter
    m=str2num(get(handles.edit1,'string'));
    alpha=str2num(get(handles.edit2,'string'));
    beta=str2num(get(handles.edit3,'string'));
    rho=str2num(get(handles.edit4,'string'));
    T=str2num(get(handles.edit5,'string'));
    Q=str2num(get(handles.edit6,'string'));

    if ~isempty(m) && ~isempty(alpha) && ~isempty(beta)
&& ~isempty(rho) && ~isempty(T) && ~isempty(Q)
        %tau
        tau=zeros(node);
        tau(DistData~=0)=Q/max(max(DistData));
        %Cari sisi
        a=1;
        for i=1:node
            for j=i:node
                if DistData(i,j)~=0
                    Sisi(a,:)=[i,j,DistData(i,j)];
                    a=a+1;
                end
            end
```

```
        end

        %iterasi
        for t=1:T
            %pembentukan jalur
            for i=1:m
                TempDist{i}=zeros(node);
                TempSisi=Sisi;
                %Roullete wheel
                prob=[]; pr=[]; pk=[];
                for j=1:size(TempSisi,1)

prob(j)=tau(TempSisi(j,1),TempSisi(j,2))^alpha*(Q/TempS
isi(j,3))^beta;
                end
                pr=prob/sum(prob);
                for j=1:size(TempSisi,1)
                    pk(j)=sum(pr(1:j));
                end
                rn=rand;
                for j=1:size(TempSisi,1)
                    if rn<=pk(j);
                        S=j; %sisi terpilih
                        break;
                    end
                end
                st=S;
                TempNode{i}{1}=TempSisi(S,1:2);

TempDist{i}(TempSisi(S,1),TempSisi(S,2))=DistData(TempS
isi(S,1),TempSisi(S,2));

TempDist{i}(TempSisi(S,2),TempSisi(S,1))=DistData(TempS
isi(S,2),TempSisi(S,1));
                TempSisi=[TempSisi(1:S-
1,:);TempSisi(S+1:end,:)];
                ln=0;
                for j=1:size(TempNode{i},2)
                    ln=ln+length(TempNode{i}{j});
                end

                %pembentukan jalur hingga semua titik
dikunjungi
                while ln<node
                    %Roullete wheel
                    prob=[]; pr=[]; pk=[];
                    for j=1:size(TempSisi,1)
```

```
prob(j)=tau(TempSisi(j,1),TempSisi(j,2))^alpha*(Q/TempS
isi(j,3))^beta;
                    end
                    pr=prob/sum(prob);
                    for j=1:size(TempSisi,1)
                        pk(j)=sum(pr(1:j));
                    end
                    rn=rand;
                    for j=1:size(TempSisi,1)
                        if rn<=pk(j);
                            S=j; %sisi terpilih
                            break;
                        end
                    end
                    st=[st S];
                    %cek sub himpunan
                    u1=TempSisi(S,1);
                    u2=TempSisi(S,2);
                    sh1=0; sh2=0; %titik belum
dikunjungi
                    for j=1:size(TempNode{i},2)
                        if ismember(u1,TempNode{i}{j})
                            sh1=j; %titik sudah
dikunjungi
                        end
                        if ismember(u2,TempNode{i}{j})
                            sh2=j;
                        end
                    end

                    %update sub himpunan
                    if sh1~=sh2
                        if sh1~=0 && sh2==0 %titik 2
belum dikunjungi & titik 1 sudah dikunjungi

TempNode{i}{sh1}=[TempNode{i}{sh1} u2];

TempDist{i}(u1,u2)=DistData(u1,u2);

TempDist{i}(u2,u1)=DistData(u2,u1);
                        elseif sh1==0 && sh2~=0 %titik
1 belum dikunjungi & titik 2 sudah dikunjungi

TempNode{i}{sh2}=[TempNode{i}{sh2} u1];

TempDist{i}(u1,u2)=DistData(u1,u2);
```

```
TempDist{i}(u2,u1)=DistData(u2,u1);
                        else %kedua titik sudah
dikunjungi tapi berbeda sub himpunan
                            temp={};
                            b=1;
                            for j=1:size(TempNode{i},2)
                                if j~=sh1 && j~=sh2

temp{b}=TempNode{i}{j};
                                    b=b+1;
                                end
                            end
                            temp{b}=[TempNode{i}{sh1}
TempNode{i}{sh2}]; %sub himpunan digabung
                            TempNode{i}=temp;

TempDist{i}(u1,u2)=DistData(u1,u2);

TempDist{i}(u2,u1)=DistData(u2,u1);
                        end
                    elseif sh1==0 && sh2==0 %kedua
titik belum dikunjungi

TempNode{i}{size(TempNode{i},2)+1}=[u1 u2];

TempDist{i}(u1,u2)=DistData(u1,u2);

TempDist{i}(u2,u1)=DistData(u2,u1);
                    end

                    %update sisi
                    TempSisi=[TempSisi(1:S-
1,:);TempSisi(S+1:end,:)];
                    ln=0;
                    for j=1:size(TempNode{i},2)
                        ln=ln+length(TempNode{i}{j});
                    end
                end

                %pembentukan jalur hingga terbentuk
spanning tree
                while size(TempNode{i},2)>1
                    %Roullete wheel
                    prob=[]; pr=[]; pk=[];
                    for j=1:size(TempSisi,1)
```

```
prob(j)=tau(TempSisi(j,1),TempSisi(j,2))^alpha*(Q/TempS
isi(j,3))^beta;
                    end
                    pr=prob/sum(prob);
                    for j=1:size(TempSisi,1)
                        pk(j)=sum(pr(1:j));
                    end
                    rn=rand;
                    for j=1:size(TempSisi,1)
                        if rn<=pk(j);
                            S=j; %sisi terpilih
                            break;
                        end
                    end
                    st=[st S];
                    %cek sub himpunan
                    u1=TempSisi(S,1);
                    u2=TempSisi(S,2);
                    sh1=0; sh2=0;
                    for j=1:size(TempNode{i},2)
                        if ismember(u1,TempNode{i}{j})
                            sh1=j;
                        end
                        if ismember(u2,TempNode{i}{j})
                            sh2=j;
                        end
                    end

                    %update sub himpunan
                    if sh1~=sh2 %kedua titik sudah
dikunjungi tapi berbeda sub himpunan
                        temp={};
                        b=1;
                        for j=1:size(TempNode{i},2)
                            if j~=sh1 && j~=sh2
                                temp{b}=TempNode{i}{j};
                                b=b+1;
                            end
                        end
                        temp{b}=[TempNode{i}{sh1}
TempNode{i}{sh2}]; %sub himpunan digabung
                        TempNode{i}=temp;

TempDist{i}(u1,u2)=DistData(u1,u2);

TempDist{i}(u2,u1)=DistData(u2,u1);
```

```
                    end

                    %update sisi
                    TempSisi=[TempSisi(1:S-
1,:);TempSisi(S+1:end,:)];
                    ln=0;
                    for j=1:size(TempNode{i},2)
                        ln=ln+length(TempNode{i}{j});
                    end
                end
            end

            %update tau
            ch=zeros(node);
            for i=1:m

ch(TempDist{i}~=0)=ch(TempDist{i}~=0)+1;
            end
            tau(DistData~=0)=(1-
rho)*tau(DistData~=0)+ch(DistData~=0).*(Q./DistData(Dis
tData~=0));

            %best so far
            for i=1:m
                TempJarak(i)=sum(sum(TempDist{i}))/2;
            end
            bsf(t)=min(TempJarak);
            if t>1
                if bsf(t)~=bsf(t-1)
                    kon=t;
                end
            else
                kon=1;
            end

            %Plot Grafik
            plot(bsf,'r','LineWidth',2);

line(kon,bsf(kon),'Marker','s','MarkerEdgeColor','k','M
arkerFaceColor','b','MarkerSize',6);
            ylabel('Total Jarak');

set(handles.axes2,'FontSize',8,'FontWeight','bold');
            pause(0.001);
        end

        %Solusi akhir
```

```
        for i=1:m
            TempJarak(i)=sum(sum(TempDist{i}))/2;
        end
        best=find(TempJarak==min(TempJarak));
        Dist=TempDist{best(1)};

        %Plot
        set(handles.radiobutton1,'value',0);
        set(handles.radiobutton2,'value',0);

set(handles.radiobutton3,'value',1,'userdata',Dist);
        CoorData=get(handles.uitable1,'data');
        Coor=[CoorData(:,2) CoorData(:,1)];
        axes(handles.axes1);
        gplot(Dist,Coor,'b');
        inlat=max(Coor(:,2))-min(Coor(:,2));
        inlon=max(Coor(:,1))-min(Coor(:,1));
        for i=1:node

line(Coor(i,1),Coor(i,2),'Marker','s','MarkerEdgeColor'
,'k','MarkerFaceColor','r','MarkerSize',5);
            text(Coor(i,1),Coor(i,2),sprintf(['V'
num2str(i)]));
        end
        set(handles.axes1,'Xlim',[min(Coor(:,1))-
inlon/10 max(Coor(:,1))+inlon/10],...
            'Ylim',[min(Coor(:,2))-inlat/10
max(Coor(:,2))+inlat/10],...
            'XTick',[],'YTick',[]);

        %jarak & waktu
        jarak=sum(sum(Dist))/2;
        waktu=toc;
        set(handles.text9,'string',['Total Jarak ACO =
' num2str(jarak) ' m']);
        set(handles.text11,'string',['Waktu Kom. ACO =
' num2str(waktu) ' detik']);
        set(handles.text12,'string',['Konvergen saat
iterasi ke-' num2str(kon)]);
    else
        errordlg('Parameter kosong');
    end
end

% --- Executes on button press in radiobutton1.
function radiobutton1_Callback(hObject, eventdata,
handles)
```

```
% hObject    handle to radiobutton1 (see GCBO)
% eventdata  reserved - to be defined in a future
version of MATLAB
% handles    structure with handles and user data (see
GUIDATA)
set(handles.radiobutton1,'value',1);
set(handles.radiobutton2,'value',0);
set(handles.radiobutton3,'value',0);
CoorData=get(handles.uitable1,'data');
DistData=get(handles.uitable2,'data');
cla(handles.axes1,'reset');
set(handles.axes1,'XTick',[],'YTick',[]);
if ~isempty(DistData)
    node=size(CoorData,1);
    Coor=[CoorData(:,2) CoorData(:,1)];
    axes(handles.axes1);
    gplot(DistData,Coor,'b');
    inlat=max(Coor(:,2))-min(Coor(:,2));
    inlon=max(Coor(:,1))-min(Coor(:,1));
    for i=1:node

line(Coor(i,1),Coor(i,2),'Marker','s','MarkerEdgeColor'
,'k','MarkerFaceColor','r','MarkerSize',5);
        text(Coor(i,1),Coor(i,2),sprintf(['V'
num2str(i)]));
    end
    set(handles.axes1,'Xlim',[min(Coor(:,1))-inlon/10
max(Coor(:,1))+inlon/10],...
        'Ylim',[min(Coor(:,2))-inlat/10
max(Coor(:,2))+inlat/10],...
        'XTick',[],'YTick',[]);
end
% Hint: get(hObject,'Value') returns toggle state of
radiobutton1

% --- Executes on button press in radiobutton2.
function radiobutton2_Callback(hObject, eventdata,
handles)
% hObject    handle to radiobutton2 (see GCBO)
% eventdata  reserved - to be defined in a future
version of MATLAB
% handles    structure with handles and user data (see
GUIDATA)
set(handles.radiobutton1,'value',0);
set(handles.radiobutton2,'value',1);
set(handles.radiobutton3,'value',0);
CoorData=get(handles.uitable1,'data');
```

```
DistData=get(handles.radiobutton2,'userdata');
cla(handles.axes1,'reset');
set(handles.axes1,'XTick',[],'YTick',[]);
if ~isempty(DistData)
    node=size(CoorData,1);
    Coor=[CoorData(:,2) CoorData(:,1)];
    axes(handles.axes1);
    gplot(DistData,Coor,'b');
    inlat=max(Coor(:,2))-min(Coor(:,2));
    inlon=max(Coor(:,1))-min(Coor(:,1));
    for i=1:node
line(Coor(i,1),Coor(i,2),'Marker','s','MarkerEdgeColor'
,'k','MarkerFaceColor','r','MarkerSize',5);
        text(Coor(i,1),Coor(i,2),sprintf(['V'
num2str(i)]));
    end
    set(handles.axes1,'Xlim',[min(Coor(:,1))-inlon/10
max(Coor(:,1))+inlon/10],...
        'Ylim',[min(Coor(:,2))-inlat/10
max(Coor(:,2))+inlat/10],...
        'XTick',[],'YTick',[]);
end
% Hint: get(hObject,'Value') returns toggle state of
radiobutton2


% --- Executes on button press in radiobutton3.
function radiobutton3_Callback(hObject, eventdata,
handles)
% hObject    handle to radiobutton3 (see GCBO)
% eventdata  reserved - to be defined in a future
version of MATLAB
% handles    structure with handles and user data (see
GUIDATA)
set(handles.radiobutton1,'value',0);
set(handles.radiobutton2,'value',0);
set(handles.radiobutton3,'value',1);
CoorData=get(handles.uitable1,'data');
DistData=get(handles.radiobutton3,'userdata');
cla(handles.axes1,'reset');
set(handles.axes1,'XTick',[],'YTick',[]);
if ~isempty(DistData)
    node=size(CoorData,1);
    Coor=[CoorData(:,2) CoorData(:,1)];
    cla(handles.axes1,'reset');
    axes(handles.axes1);
    gplot(DistData,Coor,'b');
```

```
    inlat=max(Coor(:,2))-min(Coor(:,2));
    inlon=max(Coor(:,1))-min(Coor(:,1));
    for i=1:node

line(Coor(i,1),Coor(i,2),'Marker','s','MarkerEdgeColor'
,'k','MarkerFaceColor','r','MarkerSize',5);
        text(Coor(i,1),Coor(i,2),sprintf(['V'
num2str(i)]));
    end
    set(handles.axes1,'Xlim',[min(Coor(:,1))-inlon/10
max(Coor(:,1))+inlon/10],...
        'Ylim',[min(Coor(:,2))-inlat/10
max(Coor(:,2))+inlat/10],...
        'XTick',[],'YTick',[]);
end
% Hint: get(hObject,'Value') returns toggle state of
radiobutton3
```