TUGAS AKHIR – KI141502

Identifikasi Parameter yang Berpengaruh pada Ant Colony Optimization yang Dimodifikasi pada Penyelesaian Travelling Salesman Problem

Andalani Diri Astami
NRP. 5110 100 065

Dosen Pembimbing 1
Dr. Eng. Chastine Fatichah, S.Kom., M.Kom.

Dosen Pembimbing 2
Victor Hariadi, S.Si., M.Kom.

JURUSAN TEKNIK INFORMATIKA
Fakultas Teknologi Informasi
Institut Teknologi Sepuluh Nopember
Surabaya 2015

FINAL PROJECT – KI141502

Identification of the Parameters that Affect on the Ant Colony Optimization for Solving Travelling Salesman Problem

Andalani Diri Astami
NRP. 5110 100 065

Advisor 1
Dr. Eng. Chastine Fatichah, S.Kom., M.Kom.

Advisor 2
Victor Hariadi, S.Si., M.Kom.

DEPARTMENT OF INFORMATICS
Faculty of Information Technology
Sepuluh Nopember Institute of Technology
Surabaya 2015

LEMBAR PENGESAHAN

IDENTIFIKASI PARAMETER YANG BERPENGARUH PADA ANT COLONY OPTIMIZATION YANG DIMODIFIKASI PADA PENYELESAIAN TRAVELLING SALESMAN PROBLEM

TUGAS AKHIR
Diajukan Untuk Memenuhi Salah Satu Syarat
Memperoleh Gelar Sarjana Komputer
pada
Bidang Studi Komputasi Cerdas dan Visualisasi
Program Studi S-1 Jurusan Teknik Informatika
Fakultas Teknologi Informasi
Institut Teknologi Sepuluh Nopember

Oleh:
ANDALANI DIRI ASTAMI
NRP: 5110 100 065

Disetujui oleh Pembimbing Tugas Akhir:

1. Dr. Eng. Chastine Fatichah, S.Kom, M.Kom.....
   NIP: *197512202001122002* (Pembimbing 1)

2. Victor Hariadi, S.Si., M.Kom.
   NIP: *196912281994121001* (Pembimbing 2)

SURABAYA
Januari, 2015

Identifikasi Parameter yang Berpengaruh pada Ant Colony Optimization yang Dimodifikasi pada Penyelesaian Travelling Salesman Problem

Nama	: Andalani Diri Astami
NRP	: 5110100065
Jurusan	: Teknik Informatika – FTIF ITS
Dosen Pembimbing I	: Dr. Eng. Chastine Fatichah, S.Kom., M.Kom.
Dosen Pembimbing II	: Victor Hariadi, S.Si., M.Kom.

ABSTRAK

Travelling Salesman Problem (TSP) merupakan permasalahan dalam mencari jarak minimal sebuah perjalanan pada sejumlah kota. Dimana setiap kota hanya dikunjungi sekali dan kota awal merupakan kota tujuan. Tujuan utama dari TSP adalah untuk meminimalkan total jarak yang ditempuh.

Pada Tugas Akhir ini, TSP diselesaikan menggunakan metode Ant Colony Optimization (ACO) yang sudah dimodifikasi. Ada dua modifikasi yang akan dilakukan yaitu optimalisasi routing dan individual variation dengan menggunakan metode Routing Optimization and Individual Variation (ROIVA). Optimalisasi routing dapat mengurangi frekuensi routing dan kompleksitas waktu. Individual variation dapat meningkatkan konvergensi dari algortima ACO.

Dari hasil uji coba pada beberapa dataset dapat disimpulkan bahwa Algortima ACO yang sudah dimodifikasi mampu mengurangi kompleksitas waktu dan meningkatkan konvergensi pada algortima ACO konvensional.

Kata kunci: ant colony optimization (ACO), travelling salesman problem (TSP), routing optimization and individual variation (ROIVA)

Identification of the Parameters that Affect on the Ant Colony Optimization for Solving Travelling Salesman Problem.

Name	: Andalani Diri Astami
NRP	: 5110100065
Department	: Informatics Engineering – FTIF ITS
Advisor I	: Dr. Eng. Chastine Fatichah, S.Kom., M.Kom.
Advisor II	: Victor Hariadi, S.Si., M.Kom.

ABSTARCT

Travelling Salesman Problem (TSP) is a problem that is looking for a minimum distance of a trip to a number of cities. Where each city is visited only once and the initial city is also the final destination. The main goal of this problem is to minimize the total distance traveled.

In this Final Project, TSP will be solved using Ant Colony Optimization (ACO) that has been modified. There are two modification that will be made. There are Routing Optimization and Individual Variation (ROIVA). Routing optimization can reduce the frequency of routing and running time in TSP. Individual Variation can improve the convergence of ACO Algorithm.

From the test result on several datasets, it can be concluded that the modified ACO algorithm can reduce the running and the speed of convergence of ACO Algorithm could be enhanced greatly.

Keywords : ant colony optimization (ACO), travelling salesman problem (TSP), routing optimization and individual variation (ROIVA)

KATA PENGANTAR

Alhamdulillahirabbil'alamin, segala puji bagi Allah SWT, yang telah melimpahkan rahmat dan hidayah-Nya sehingga penulis dapat menyelesaikan Tugas Akhir yang berjudul: **"IDENTIFIKASI PARAMETER YANG BERPENGARUH PADA ANT COLONY OPTIMIZATION PADA PENYELESAIAN TRAVELLING SALESMAN PROBLEM"**

Pengerjaan Tugas Akhir ini merupakan suatu kesempatan yang sangat baik bagi penulis. Dengan pengerjaan Tugas Akhir ini, penulis bisa belajar lebih banyak untuk memperdalam dan meningkatkan apa yang telah didapatkan penulis selama menempuh perkuliahan di Teknik Informatika ITS. Dengan Tugas Akhir ini penulis juga dapat menghasilkan suatu implementasi dari apa yang telah penulis pelajari.

Selesainya Tugas Akhir ini tidak lepas dari bantuan dan dukungan beberapa pihak. Sehingga pada kesempatan ini penulis mengucapkan terima kasih kepada:

1. Allah SWT atas limpahan rahmat-Nya sehingga penulis dapat menyelesaikan Tugas Akhir ini dengan baik.
2. Bapak, Alm. Ibu, Dhya, Dini yang sudah menjadi motivasi penulis selama ini hingga terselesaikannya Tugas Akhir ini.
3. Ibu Dr. Eng. Chastine Fatichah, S.Kom., M.Kom. selaku dosen pembimbing I yang telah memberikan nasihat, arahan, dan bantuan dalam menyelesaikan Tugas Akhir ini.
4. Bapak Victor Hariadi, S.Si., M.Kom. selaku dosen pembimbing II yang telah membantu dan membimbing penulis dalam menyelesaikan Tugas Akhir ini.
5. Ibu Dr. Ir. Siti Rochimah, M.T. selaku dosen wali penulis, Ibu Dr. Eng. Nanik Suciati, S.Kom., M.Kom. selaku Kepala Jurusan Teknik Informatika ITS, dan

segenap dosen Teknik Informatika yang telah memberikan ilmunya.

6. Pak Yudi, Pak Sugeng, Mbak Fathin, Bu Wartani, Pak Sholeh, Pak Pri dan segenap staf Tata Usaha yang telah memberikan segala bantuan dan kemudahan kepada penulis selama menjalani kuliah di Teknik Informatika ITS.
7. Teman-teman angkatan 2010 yang selalu menjaga kebersamaan, kakak-kakak angkatan 2007, 2008, dan 2009 serta adik-adik angkatan 2011 dan 2012 yang membuat penulis untuk selalu belajar.
8. Serta semua pihak yang yang telah turut membantu penulis dalam menyelesaikan Tugas Akhir ini.

Penulis menyadari bahwa Tugas Akhir ini masih memiliki banyak kekurangan. Sehingga dengan kerendahan hati, penulis mengharapkan kritik dan saran dari pembaca untuk perbaikan ke depan.

Surabaya, Januari 2015

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

DAFTAR KODE SUMBER

DAFTAR PUSTAKA

[1] M. Dorigo and L.M Gambardella, "Ant Colonies for The Travelling Salesman Problem," *IEEE Transaction on Evolutionary Computation*, vol. 1, no. 1, pp. 53-66, 1997.

[2] S.Lin, "Computer Solutions for the Traveling Salesman Problem," *Bell System Technology Journal*, vol. 44, pp. 2245-2269, 1965.

[3] J.M Kan and Y. Zhang, "Application of an Improved Ant Colony Optimization on Generelized Travelling Salesman Proble," *Energy Procedia*, vol. 17, pp. 319-325, 2012.

[4] Z. L.Pei, J. Yang, and Y. Liang Y. Zhang, "An Improved Ant Colony Optimization Algorithm Based on Route Optimization and Its Application in Travelling Salesman Problem," *BIBE*, pp. 693-698, 2007.

[5] Vittorio Maniezzo, Alberto Colomi Dorigo M, "The Ant System: optimization by a colony of cooperating agents," *IEEE Transaction on System, Man, and Cybernetics*, vol. Part B, no. 26(1), p. 1 13, 1996.

[6] H. Wang, "Comparison of several intelligent algorithms for solving TSP problem in industrial engineering," *System Engineering Procedia*, vol. 4, pp. 226-235, 2012.

BIODATA PENULIS

Andalani Diri Astami yang akrab dipanggil Tami, lahir pada 10 Oktober 1992 di Jakarta. Penulis merupakan anak pertama dari tiga bersaudara. Pendidikan yang ditempuh penulis dari SD Islam Al-Azhar Syifa Budi Legenda Bekasi (1998-2004), SMP Islam Al-Azhar 9 Kemang Pratama Bekasi (2004-2005), SMPN 11 Bekasi (2005-2007), SMA Islam PB.Soedirman 1 Bekasi (2007-2009), SMA Islam PB.Soedirman Jakarta (2009-2010), dan S1 Teknik Informatika ITS Surabaya (2010-2014). Di Teknik Informatika ITS, penulis mengambil bidang minat Komputasi Cerdas dan Visualisasi (KCV). Selama masa perkuliahan, penulis aktif sebagai anggota Himpunan Mahasiswa Teknik Computer-Informatika (HMTC). Semasa di Himpunan, penulis pernah menjadi staf departemen kewirausahaan (2011-2012). Penulis dapat dihubungi melalui email: andalani1010@yahoo.com.

BAB 1
PENDAHULUAN

Pada bagian pendahuluan dijelaskan hal dasar mengenai Tugas Akhir ini yang meliputi latar belakang, tujuan, manfaat, rumusan masalah, batasan masalah, metodologi, serta sistematika penulisan Tugas Akhir.

1.1 Latar Belakang

Travelling Salesman Problem (TSP) dinyatakan sebagai permasalahan dalam mencari jarak minimal sebuah perjalanan pada sejumlah *n* kota. Setiap kota hanya dikunjungi sekali dimana kota awal juga merupakan kota akhir. Persoalan TSP sudah banyak diaplikasikan dibeberapa perusahaan antara lain untuk rute pengambilan surat dari kotak pos, rute patroli polisi, rute penjemputan siswa sekolah, dsb. Oleh karena itu, solusi optimal dari permasalahan TSP ini akan sangat membantu perusahaan untuk mengefesienkan proses yang terjadi baik dari segi waktu maupun biaya.

Ant Colony Optimization (ACO) diadopsi dari perilaku koloni semut yang dikenal sebagai sistem semut [1]. Dalam dunia nyata, semut dapat menemukan jalur terpendek dari sarang ke sumber makanan tanpa menggunakan isyarat visual. Mereka juga dapat beradaptasi dengan perubahan lingkungan. Misalnya mencari jalur terpendek baru setelah jalur lama bukan lagi menjadi jalur terpendek saat ini. Hal ini dikarenakan semut bergerak mengikuti jejak *pheromone*.

Pheromone adalah zat kimia yang berasal dari kelenjar endokrin dan digunakan oleh makhluk hidup lain untuk mengenali sesama jenis, individu lain, dan kelompok. Saat semut berjalan, mereka meninggalkan sejumlah *pheromone* pada jalur yang dilalui. Masing-masing semut lebih memilih jalur yang memiliki jumlah *pheromone* lebih banyak.

Dengan prinsip algoritma yang didasarkan pada perilaku koloni semut dalam menemukan rute terpendek, maka ACO dapat diterapkan pada permasalahan TSP. Berdasarkan hasil penelitian yang dilakukan oleh M. Dorigo dan L. M Gambardella (1997) dalam penyelesaian permasalahan TSP, terbukti bahwa algoritma *Ant Colony Optimization* (ACO) mampu mendapatkan hasil perjalanan terbaik dibandingkan dengan *Genetic Algorithm* (GA), *Evolutionary Programming* (EP), *Simulated Annealing* (SA), dan *Annealing-Genetic Algorithm* (AG) [1].

Tugas Akhir ini mengimplementasikan modifikasi dari algoritma ACO menggunakan metode *Routing Optimization and Individual Variation* (ROIVA) [4]. Dengan *Routing Optimization Strategy*, ACO dapat dioptimalkan dengan mengurangi frekuensi dari *routing* dan kompleksitas waktu. *Individual Variation* digunakan untuk meningkatkan konvergensi dari ACO konvensional. Algoritma ACO yang sudah dimodifikasi akan dibandingkan dengan algoritma ACO konvensional dalam menyelesaikan masalah TSP.

1.2 Rumusan Masalah

Rumusan masalah yang diangkat dalam pembuatan Tugas Akhir ini adalah sebagai berikut:

1. Pengimplementasian algoritma *Ant Colony Optimization* (ACO) dalam menyelesaikan permasalahan *Travelling Salesman Problem* (TSP) menggunakan metode *Routing Optimization and Individual Variation* (ROIVA).
2. Pengidentifikasian parameter yang berpengaruh pada ACO dengan metode ROIVA dalam menyelesaikan TSP.
3. Penyusunan skenario uji coba dari algoritma ACO dengan metode ROIVA dalam menyelesaikan TSP yang melibatkan beberapa parameter dengan beberapa *dataset*.

1.3 Batasan Masalah

Permasalahan yang dibahas dalam Tugas Akhir ini memiliki batasan sebagai berikut:

1. Implementasi dan uji coba dilakukan dengan menggunakan MATLAB R2014a.
2. Graf yang digunakan pada uji coba yang dilakukan adalah graf lengkap.
3. Data uji coba yang digunakan diambil dari TSPLIB (http://www.iwr.uni-heidelberg.de/groups/comopt/software/TSPLIB95/tsp/.
4. Data uji coba yang digunakan adalah:
   a. Ulysses16 yang terdiri dari 16 *nodes*.
   b. Eil76 yang terdiri dari 51 *nodes*.
   c. Gr96 yang terdiri dari 96 *nodes*.
   d. Gr137 yang terdiri dari 137 *nodes*.
   e. Ch150 yang terdiri dari 150 *nodes*.
   f. D198 yang terdiri dari 198 *nodes*.
   g. Pr299 yang terdiri dari 299 *nodes*.
   h. D493 yang terdiri dari 493 *nodes*.
5. Pengujian akan dilakukan dengan membandingkan ACO menggunakan metode ROIVA dengan ACO konvensional.

1.4 Tujuan

Tujuan dari pembuatan Tugas Akhir ini adalah:

1. Menerapkan konsep dan cara kerja algoritma *Ant Colony Optimization* (ACO) menggunakan metode *Routing Optimization and Individual Variation* (ROIVA) dalam menyelesaikan *Travelling Salesman Problem* (TSP).
2. Mengidentifikasi parameter yang berpengaruh pada ACO menggunakan metode ROIVA dalam menyelesaikan permasalahan TSP.
3. Mengevaluasi kinerja algoritma ACO dengan metode ROIVA dengan melakukan analisis uji coba.

1.5 Manfaat

Tugas Akhir ini diharapkan akan bermanfaat dalam membantu perusahaan terutama yang berhubungan dengan pengiriman barang atau surat, transportasi seperti perusahaan bus dan travel dalam menentukan jalur terpendek saat mereka melaksanakan perjalanan sehingga dapat mengefesiensikan waktu, tenaga, dan biaya.

1.6 Metodologi

Pembuatan Tugas Akhir ini dilakukan dengan menggunakan metodologi sebagai berikut:

1. Penyusunan Proposal Tugas Akhir

   Tahap awal untuk memulai pengerjaan Tugas Akhir adalah penyusunan Proposal Tugas Akhir. Proposal Tugas Akhir yang diajukan memiliki gagasan untuk mengimplementasikan algoritma *Ant Colony Optimization* (ACO) yang dimodifikasi menggunakan metode *Routing Optimization and Individual Variation* (ROIVA) dalam menyelesaikan persoalan TSP.

2. Studi Literatur

   Pada tahap ini dilakukan pencarian, pengumpulan, pembelajaran dan pemahaman informasi dan literatur yang diperlukan untuk mengiimplementasikan algoritma ACO yang dimodifikasi menggunakan metode ROIVA dalam menyelesaikan persoalan TSP. Informasi dan literatur didapatkan dari jurnal-jurnal di internet dan buku.

3. Perancangan Perangkat Lunak

   Pada tahap ini dilakukan proses perancangan perangkat lunak, berdasarkan literatur yang telah dikaji kemudian dibuat desain model, dan diagram alir proses-proses yang ada. Kemudian dilakukan implementasi.

4. Pengimplementasian Perangkat Lunak

   Implementasi merupakan tahap membangun rancangan sistem yang telah dibuat. Pada tahapan ini merealisasikan apa yang terdapat pada tahapan sebelumnya, sehingga menjadi sebuah sistem yang sesuai dengan apa yang telah direncanakan.

5. Uji Coba dan Evaluasi

Pada tahapan ini dilakukan uji coba terhadap perangkat lunak yang telah dibuat. Tahap ini dimaksudkan juga untuk mengevaluasi jalannya sistem, mencari masalah yang mungkin timbul dan mengadakan perbaikan jika terdapat kesalahan.

6. Penyusunan Laporan Tugas Akhir

Pada tahap ini disusun buku sebagai dokumentasi dari pelaksanaan Tugas Akhir yang mencangkup seluruh konsep, teori, implementasi, serta hasil yang telah dikerjakan.

1.7 Sistematika Penulisan

Laporan Tugas Akhir ini dibagi menjadi beberapa bab, sebagai berikut:

Bab I Pendahuluan

Bab yang berisi mengenai latar belakang, tujuan, dan manfaat dari pembuatan Tugas Akhir. Selain itu permasalahan, batasan masalah, metodologi yang digunakan, dan sistematika penulisan juga merupakan bagian dari bab ini.

Bab II Dasar Teori

Bab ini berisi penjelasan secara detail mengenai dasar-dasar penunjang dan teori-teori yang digunakan untuk mendukung pembuatan Tugas Akhir ini.

Bab III Perancangan Perangkat Lunak

Bab ini berisi pembahasan mengenai desain yang digunakan dalam implementasi perangkat lunak. Pada bab ini juga dibahas mengenai perancangan data yang terdiri dari data proses, data masukan, dan data keluaran.

Bab IV Implementasi

Bab ini akan membahas implementasi dari aplikasi yang telah dibuat, akan dilakukan pembuatan aplikasi yang dibangun dengan komponen-komponen yang telah ada yang sesuai dengan permasalahan dan batasannya yang telah dijabarkan pada bab pertama.

Bab V Uji Coba dan Evaluasi

Bab ini menjelaskan kemampuan perangkat lunak dengan melakukan pengujian kebenaran dan pengujian kinerja dari sistem yang telah dibuat.

Bab VI Kesimpulan dan Saran

Bab ini berupa hasil penelitian yang menjawab permasalahan atau yang berupa konsep, program, dan karya rancangan. Selain itu, pada bab ini diberikan saran-saran yang berisi hal-hal yang masih dapat dikerjakan dengan lebih baik dan dapat dikembangkan lebih lanjut, atau berisi masalah-masalah yang dialami pada proses pengerjaan Tugas Akhir.

.

BAB 2
DASAR TEORI

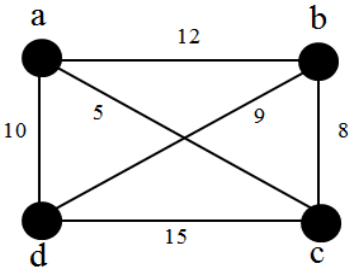
Bab ini membahas tentang teori dasar yang menunjang penyusunan Tugas Akhir mengenai permasalahan *Travelling Salesman Problem* (TSP), algoritma *Ant Colony Optimization* (ACO), dan algoritma ACO menggunakan metode ROIVA.

2.1 *Travelling Salesman Problem*

Travelling Salesman Problem (TSP) merupakan salah satu persoalan yang terkenal dalam teori graf. TSP adalah suatu masalah yang ditemukan dari seorang pedagang yang berkeliling mengunjungi sejumlah kota. Inti permasalahan TSP adalah menemukan lintasan terpendek yang harus dilalui oleh seorang pedagang [2]. Pedagang tersebut berangkat dari satu kota dan menyinggahi setiap kota tepat satu kali dan kembali ke kota asal keberangkatan.

Dari penjelasan diatas, dapat ditarik kesimpulan bahwa ada dua batasan dalam permasalahan TSP ini. Batasan pertama adalah setiap kota hanya boleh dikunjungi tepat satu kali dan pada akhir perjalanan harus kembali ke kota asal keberangkatan. Batasan yang kedua adalah lintasan yang ditempuh adalah lintasan yang paling terpendek. Data yang diketahui dalam permasalahan ini adalah jumlah kota yang harus dikunjungi beserta titik koordinatnya.

Kota dapat dinyatakan sebagai simpul graf, sedangkan sisi menyatakan jalan yang menghubungkan antar dua kota. Bobot pada sisi menyatakan jarak antara dua kota. Permasalahan TSP tidak lain menentukan sirkuit *Hamilton* yang memiliki bobot minimum pada sebuah graf terhubung. Pada persoalan TSP ini, sebuah graf lengkap dengan n buah simpul ($n > 2$), jumlah sirkuit *Hamilton* yang berbeda adalah $(n - 1)!/2$ [2]. Contoh kasus dalam TSP dijelaskan pada Gambar 2.1.



Gambar 2.1 Graf Lengkap

Graf lengkap diatas mempunyai *n* = 4 simpul seperti yang ditujukan pada Gambar 2.1. Graf tersebut memiliki sirkuit *Hamilton* sebanyak (4-1)!/2 = 3, yaitu:

I_1= (a,b,c,d,a) atau (a,d,c,b,a) dengan panjang rute = 10 + 12 + 8 + 15 = 45

I_2 = (a,c,d,b,a) atau (a,b,d,c,a) dengan panjang rute = 12 + 5 + 9 + 15 = 41

I_3 = (a,c,b,d,a) atau (a,d,b,c,a) dengan panjang rute = 10 + 5 + 9 + 8 = 32

Jadi sirkuit *Hamilton* terpendek adalah I_3 = (a,c,b,d,a) atau (a,d,b,c,a) dengan panjang rute = 10 + 5 + 9 + 8 = 32

2.2 *Ant Colony Optimization*

Ant Colony Optimization (ACO) diadopsi dari perilaku koloni semut yang dikenal sebagai sistem semut [1]. Dalam dunia nyata, semut dapat menemukan jalur terpendek dari sarang ke sumber makanan tanpa menggunakan isyarat visual. Mereka juga dapat beradaptasi dengan perubahan lingkungan. Semut bergerak secara acak dalam mencari makanan. Saat menemukan makanan dan kembali ke koloni, mereka meninggalkan jejak *pheromone*.

Pheromone adalah zat kimia yang berasal dari kelenjar endokrin dan digunakan oleh makhluk hidup untuk mengenali sesama jenis, individu lain, dan kelompok. Saat semut lain menemukan jejak tersebut maka mereka cenderung mengikuti jejak tersebut dan memperbarui jejak tersebut jika menemukan makanan.

Semut yang melewati lintasan yang pendek akan meninggalkan aroma *pheromone* yang lebih tajam daripada yang menempuh lintasan yang lebih panjang. Hal ini dikarenakan *pheromone* dapat menguap sehingga kekuatan daya tariknya akan berkurang. Semakin lama waktu yang dibutuhkan seekor semut dalam mencari makanan dan kembali ke koloni, maka semakin banyak juga *pheromone* tersebut menguap. Penguapan *pheromone* ini bertujuan untuk menghindari terjadinya solusi optimum lokal. Jika sama sekali tidak terjadi penguapan *pheromone*, maka rute yang dipilih oleh semut pertama cenderung akan dipilih oleh semut lainnya. Hal ini akan membatasi ruang penjelajahannya.

Proses dalam ACO bisa dijelaskan sebagai berikut, misalkan ada *m* semut dalam satu koloni. Semut-semut akan memulai dari sarang mereka menuju tujuan akhir melalui beberapa simpul dan berakhir pada simpul tujuan di akhir setiap siklus atau iterasi. Jika semua semut sudah menyelesaikan lintasannya, jumlah *pheromone* pada lintasan terbaik secara global akan diperbarui. Lintasan global terbaik artinya terbaik diantara semua semut. Pada awal proses yaitu pada iterasi pertama, semua ruas dari simpul awal akan diberi jumlah pheromone yang sama. Sehingga pada iterasi pertama, semua semut akan mulai dari simpul awal dan berakhir pada simpul tujuan. Proses berakhir jika jumlah iterasi maksimum sudah tercapai atau tidak ada lagi solusi yang lebih baik yang bisa didapat dalam beberapa iterasi.

2.3 ACO untuk TSP

Pertama kali ACO diperkenalkan, ACO sudah diterapkan untuk menyelesaikan persoalan *Traveling Salesman Problem* (TSP) [3]. Dalam TSP, jika terdapat *n* kota akan ada (*n* (*n* - 1)/2) buah ruas, dan juga memiliki (*n* - 1)!/2 rute yang mungkin. Semut memulai dan mengakhiri rutenya di kota yang sama. Pada setiap tahap, semut memilih untuk berpindah dari satu kota ke kota lain menurut beberapa aturan:

1. Semut harus mengelilingi setiap kota tepat satu kali.
2. Sebuah kota yang jauh memiliki kesempatan yang lebih kecil untuk dipilih.
3. Semakin kuat jejak *pheromone* yang ada di jalur antara dua kota, semakin besar kemungkinan jalur tersebut akan dipilih.
4. Setelah setiap iterasi selesai, jejak *pheromone* akan menguap.

Setiap semut memiliki sebuah memori yang disebut dengan *tabulist* atau $tabu_k$. *Tabulist* berisi semua kota yang telah dikunjungi semut. *Tabulist* mencegah semut untuk mengunjungi kota-kota yang sebelumnya telah dikunjungi sehingga tidak akan ada satu kota yang dikunjungi lebih dari sekali. Kota pertama semut sebagai kota awal perjalanan harus diisikan sebagai elemen pertama pada tabel tabu. Setiap $tabu_k(1)$ bisa berisi indeks kota antara 1 sampai *n*. Semut akan melakukan perjalanan dengan memilih salah satu dari kota-kota yang tidak terdapat pada $tabu_k$ sebagai kota tujuan selanjutnya.

Secara berulang, setiap semut akan memilih kota berikutnya sampai semua kota satu persatu dikunjungi atau telah menempati $tabu_k$ kemudian kembali ke kota awal. Pemilihan kota-kota yang akan dilaluinya berdasarkan suatu fungsi probabilitas dengan mempertimbangkan *visibility* (*invers* dari jarak) kota tersebut dan jumlah *pheromone* yang terdapat pada ruas antar kota. Semut-semut akan memilih jalur yang lebih pendek dan atau memiliki tingkat *pheromone* yang tinggi. Untuk menentukan kota tujuan digunakan Persamaan 2.1.

$$P_{ij}^{k} = \begin{cases} \frac{[\tau_{ij}]^{\alpha}\,[\eta_{ij}]^{\beta}}{\sum_{l \in \mathcal{N}_i^k}[\tau_{il}]^{\alpha}[\eta_{il}]^{\beta}}, & \text{jika } j \in \mathcal{N}_i^k \\ 0, & \text{lainnya} \end{cases} \quad (2.1)$$

Dimana P_{ij}^{k} adalah probabilitas semut k dari titik i ke titik j. τ_{ij} adalah jumlah *pheromone* yang terdapat pada jalur antara titik *i* dan titik *j*. $\eta_{ij} = \frac{1}{d_{ij}}$ adalah *invers* jarak titik *i* ke titik *j* (d_{ij}). α adalah sebuah parameter pengendali *pheromone* dan β adalah parameter pengendali jarak (α > 0 dan β > 0). $\mathcal{N}_i^k$ adalah himpunan yang berisi titik-titik yang akan dikunjungi oleh semut *k*. *l* adalah titik yang berada dalam $\mathcal{N}_i^k$. d_{ij} dapat dihitung berdasarkan Persamaan 2.2.

$$d_{ij} = \sqrt{(x_i + x_j)^2 + (y_i + y_j)^2} \quad (2.2)$$

Dengan hasil perhitungan probabilitas sesuai Persamaan 2.1, semut-semut akan memilih kota yang memiliki probabilitas yang paling besar. Pada rute yang pendek waktu perjalanan akan lebih cepat, sehingga ketebalan *pheromone* tetap tinggi.

Setelah satu siklus terselesaikan oleh semua semut, dilakukan perhitungan panjang rute atau L_k setiap semut. Perhitungan dilakukan berdasarkan $tabu_k$ berdasarkan Persamaan 2.3.

$$L_k = d_{tabu_k(n),tabu_k(1)} + \sum_{s=1}^{n-1} d_{tabu_k(s),tabu_k(s+1)} \quad (2.3)$$

Dimana *n* menyatakan jumlah kota yang harus dikunjungi dan *s* menyatakan indeks urutan kunjungan. Setelah L_k setiap semut dihitung bisa didapatkan panjang rute minimum tertutup setiap siklus (L_{minNC}) dan panjang rute minimum keseluruhan (L_{min}).

Koloni semut akan meninggalkan jejak *pheromone* pada lintasan antar kota yang dilalui. Adanya penguapan dan perbedaan jumlah semut yang lewat menyebabkan kemungkinan terjadinya perubahan jumlah intensitas jejak *pheromone* semut antar kota. Perubahan jumlah intensitas ini dihitung berdasarkan Persamaan 2.4.

$$\Delta\tau_{ij} = \sum_{k=1}^{m} \Delta\tau_{ij}^{k} \quad (2.4)$$

Dimana $\Delta\tau_{ij}^{k}$ adalah jumlah penambahan jejak *pheromone* antar kota setiap semut dapat dihitung berdasarkan Persamaan 2.5.

$$\Delta\tau_{ij}^{k} = \begin{cases} \frac{1}{L_k}, & \text{untuk } (i,j) \in \text{tabu}_k \\ 0, & \text{lainnya} \end{cases} \quad (2.5)$$

Dimana L_k merupakan panjang lintasan yang dilalui semut k. N Setelah satu siklus terselesaikan atau semua semut sudah selesai melakukan perjalanan, terjadi pembaruan *pheromone* pada setiap kota. Pembaruan *pheromone* dihitung berdasarkan Persamaan 2.6.

$$\tau_{ij} = (1-\rho)\tau_{ij} + \Delta\tau_{ij} \quad (2.6)$$

Dimana ρ tingkat evaporasi *pheromone.* Setelah satu siklus terselesaikan, *tabulist* perlu dikosongkan untuk diisi lagi dengan urutan kota yang baru pada siklus selanjutnya jika jumlah siklus maksimum belum tercapai atau belum terjadi konvergensi.

2.4 *Routing Optimization and Individual Variation* (ROIVA)

2.4.1 *Routing Optimization Strategy*

Routing pada algoritma *Ant Colony Optimization* (ACO) adalah kondisi semut mencari kota yang akan disinggahi selanjutnya. *Routing* merupakan operasi yang paling sering

dilakukan saat menjalankan ACO. Kompleksitas waktu dari ACO juga bergantung pada proses *routing*. Semakin banyak jumlah kota yang harus dikunjungi oleh semut, semakin tinggi juga proses *routing* yang dijalankan, yang berarti akan semakin tinggi juga kompleksitas waktu. *Routing Optimizatin Strategy* (ROS) dapat mengoptimalkan algoritma ACO dengan mengurangi frekuensi dari *routing*.

Jumlah *routing* dalam proses ACO dapat dihitung menggunakan *m * n *(n - 1)/2* dalam satu iterasi [4]. Dimana *m* adalah jumlah semut dan *n* adalah jumlah kota. Untuk mengurangi frekuensi dari *routing,* metode ini mempunyai suatu variable *Lmin* yang bertugas untuk menyimpan solusi minimum saat ini. *Lmin* awalnya diinisialisasikan dengan ∞. Pada saat semut selesai melakukan *routing,* akan dibandingkan panjang jalur saat ini dengan *Lmin,* jika *Lmin* lebih kecil nilainya maka semut yang melakukan perjalanan langsung diberhentikan perjalanannya dari siklus saat ini, jika tidak maka dicari semut minimum lagi.

2.4.2 *Individual Variation*

Dalam ACO, parameter α dan β adalah statik dan semua menggunakan nilai yang sama di setiap prosesnya. Dorigo menyarankan untuk nilai $\alpha = 1$ dan $\beta = 5$ adalah tepat untuk dibanyak situasi [5]. Dalam ACO, awalnya jarak memiliki dampak yang lebih kuat pada proses *routing.* Setelah algoritma berjalan, dampak dari *pheromone* harus diperkuat karena informasi selanjutnya tentang jalur yang lebih baik disimpan dalam *pheromone*. Dari pengamatan ini, ACO dengan *Individual Variation* menggunakan nilai parameter α dan β yang berbeda dan dapat berubah.

Di awal siklus ditentukan nilai α dan β sebagai nilai awal parameter. Pada akhir setiap siklus, semut pemenang atau semut yang memiliki panjang jalur minimum diperbolehkan untuk memodifikasi parameter dengan aturan sebagai berikut:

- α meningkat untuk meningkatkan pengaruh *pheromone*.
- β menurun untuk mengurangi pengaruh jarak.

Strategi ini dinamakan *encouragement* [4]. $P_{ij}^{k}(\alpha,\beta)$ menunjukkan probabilitas semut k dalam memilih kota mana yang akan dikunjungi dari kota i ke j berdasarkan parameter α dan β. Dengan x adalah sebuah angka *real*, $0 < x < \tau_{max}$, yang membuat persamaan berikut menjadi Persamaan 2.7.

$$\frac{x^{\alpha}}{\sum_{l \in \mathcal{N}_i^k} [\tau_{i,l}(t)]^{\alpha}[\eta_{i,l}]^{\beta}} = \frac{x^{\alpha+1}}{\sum_{l \in \mathcal{N}_i^k} [\tau_{i,l}(t)]^{\alpha+1}[\eta_{i,l}]^{\beta}} \tag{2.7}$$

Oleh karena itu Persamaan 2.7 menjadi Persamaan 2.8.

$$x = \frac{\sum_{l \in \mathcal{N}_i^k} [\tau_{i,l}(t)]^{\alpha+1}[\eta_{i,l}]^{\beta}}{\sum_{l \in \mathcal{N}_i^k} [\tau_{i,l}(t)]^{\alpha}[\eta_{i,l}]^{\beta}} \tag{2.8}$$

Untuk kota K, jika $\tau_{i,K}(t) \geq x$ maka persamaan menjadi Persamaan 2.9.

$$\tau_{i,K}(t) \geq \frac{\sum_{l \in \mathcal{N}_i^k} [\tau_{i,l}(t)]^{\alpha+1}[\eta_{i,l}]^{\beta}}{\sum_{l \in \mathcal{N}_i^k} [\tau_{i,l}(t)]^{\alpha}[\eta_{i,l}]^{\beta}} \tag{2.9}$$

Dengan demikian Persamaan 2.9 menjadi Persamaan 2.10.

$$\frac{[\tau_{i,K}(t)]^{\alpha+1}[\eta_{i,K}]^{\beta}}{\sum_{l \in \mathcal{N}_i^k}[\tau_{i,l}(t)]^{\alpha+1}[\eta_{i,l}]^{\beta}} \geq \frac{[\tau_{i,K}(t)]^{\alpha}[\eta_{i,K}]^{\beta}}{\sum_{l \in \mathcal{N}_i^k}[\tau_{i,l}(t)]^{\alpha}[\eta_{i,l}]^{\beta}} \tag{2.10}$$

Berdasarkan Persamaan 2.1, maka menjadi Persamaan 2.11 dan 2.12.

$$P_{i,K}^{t}(\alpha+1,\beta) \geq P_{ij}^{t}(\alpha,\beta) \tag{2.11}$$

$$P_{i,K}^{t}(\alpha, \beta - 1) \geq P_{ij}^{t}(\alpha, \beta) \quad (2.12)$$

Probabilitas dari kota *K* untuk dipilih oleh semut *k* meningkat jika $\tau_{i,K}(t)$ lebih besar daripada sebuah nilai, ketika α dinaikkan. Begitu juga untuk β dengan analisis yang sama.

BAB 3
PERANCANGAN PERANGKAT LUNAK

Bab ini membahas mengenai perancangan perangkat lunak yang akan digunakan untuk menyelesaikan Tugas Akhir. Pembahasan perancangan perangkat lunak akan meliputi, penjelasan tentang *dataset* yang digunakan untuk uji coba, perancangan data, serta perancangan proses-proses yang ada dalam Tugas Akhir ini secara lebih jelas. Perancangan sistem pada bagian ini meliputi dua bagian penting, yaitu perancangan data yang akan digunakan dalam sistem dan algoritma yang akan digunakan dalam sistem.

3.1 Perancangan Data

Bagian ini menjelaskan perancangan data yang digunakan dalam proses penyelesaian TSP menggunakan algoritma ACO yang dimodifikasi dengan metode ROIVA (*Routing Optimization and Individual Variaton*) [4]. Perancangan data sangat diperlukan untuk menentukan data yang tepat untuk perangkat lunak sehingga dapat dioperasikan dengan benar. Data yang diperlukan adalah data masukan (*input*) yang didapatkan dari pengguna, data proses yang dibutuhkan dan dihasilkan selama proses eksekusi, dan data keluaran (*output*) yang memberikan hasil proses eksekusi.

3.1.1 Data Masukan

Data masukan (*input*) merupakan data yang dimasukan ke dalam perangkat lunak. Pada Tugas Akhir ini, terdapat 5 *dataset* yang digunakan sebagai data masukan pada uji coba evaluasi, yaitu *dataset* yang didapat pada situs TSPLIB, yaitu *Ulysses16, Eil76, Gr96, Gr137, Ch150. Dataset* terdiri dari data kota yang berisi nomor kota dan titik koordinat kota. Contoh *dataset* dapat dilihat pada Tabel 3.1. Selain *dataset* diatas, terdapat data masukan lain yang dapat dilihat pada Tabel 3.2.

Tabel 3.1 Contoh *Dataset* 16 Kota

Nomor kota	Sumbu x	Sumbu y
1	38.24	20.42
2	39.57	26.15
3	40.56	25.32
4	36.26	23.12
5	33.48	10.54
6	37.56	12.19
7	38.42	13.11
8	37.52	20.44
9	41.23	9.10
10	41.17	13.05
11	36.08	-5.21
12	38.47	15.13
13	38.15	15.35
14	37.51	15.17
15	35.49	14.32
16	39.36	19.56

Tabel 3.2 Data Masukan

No	Nama Data	Keterangan
1	`alpha`	Konstanta yang menunjukkan tingkat kepentingan *pheromone.*
3	`beta`	Konstanta yang menunjukkan tingkat kepentingan jarak atau *distance.*
4	`rho`	Konstanta yang menunjukkan penguapan *pheromone.*
5	`MaxITime`	Jumlah maksimal perulangan yang dilakukan.
6	`AntNum`	Jumlah semut yang melakukan perjalanan.

3.1.2 Data Proses

Data proses adalah data-data yang digunakan dalam proses penyelesaian TSP menggunakan algortima ACO dengan metode ROIVA. Data proses dapat dilihat pada Tabel 3.3 dan Tabel 3.4.

Tabel 3.3 Data Proses (1)

No	Nama Data	Keterangan
1	`Dimension`	Jumlah kota dalam TSP.
2	`NodeCoord`	Menyimpan titik koordinat kota.
3	`NodeWeight`	Menyimpan bobot kota.
4	`CityMatrix`	Matriks kota.
5	`nextnode`	Tujuan kota berikutnya.
6	`CurrentNode`	Menunjukkan kota yang saat ini sedang ditempuh.
7	`VisitedNodes`	Menunjukkan kota yang telah dilewati.
8	`TourLength`	Menyimpan jarak tempuh tiap rute semut.
9	`tau_i`	Menunjukkan nilai *pheromone* sesuai kota saat ini.
10	`MatrixTau`	Menyimpan nilai *pheromone* yang terbaru untuk perbandingan *next state*.
11	`sumdtau`	Menunjukkan jumlah *pheromone* yang ditambahkan.
12	`tau`	Menunjukkan *pheromone* yang sudah ter*update*.

Tabel 3.4 Data Proses (2)

13	`dis_i`	Menunjukkan nilai jarak sesuai kota saat ini.
14	`Distances`	Menyimpan jarak antar kota.
15	`Ant.lengths`	Menyimpan panjang rute semut.
16	`Ant.tours`	Menyimpan rute perjalanan semut.
17	`jmlTour`	Menyimpan jumlah *routing* semut.
18	`Probs`	Menunjukkan nilai Probabilitas.
19	`IBLength`	Menyimpan panjang rute per iterasi.
20	`IBTour`	Menyimpan panjang rute per iterasi.

3.1.3 Data Keluaran

Data keluaran yang dihasilkan dari proses penyelesaian TSP terdapat tiga bagian yaitu jarak tempuh terpendek, jumlah *routing*, dan kompleksitas waktu program. Jarak tempuh terpendek adalah hasil penjumlahan jarak yang ditempuh semut. Jumlah *routing* adalah total *routing* yang dilakukan semut. Total kompleksitas waktu adalah waktu yang dibutuhkan untuk menyelesaikan sebuah permasalahan TSP. Data keluaran dapat dilihat pada Tabel 3.5.

Tabel 3.5 Data Keluaran

No	Nama Data	Keterangan
1	`GBLength`	Jarak tempuh terpendek semut.
2	`GBTour`	Jumlah *routing*.
3	`t`	Waktu yang dibutuhkan untuk menyelesaikan sebuah permasalahan TSP.

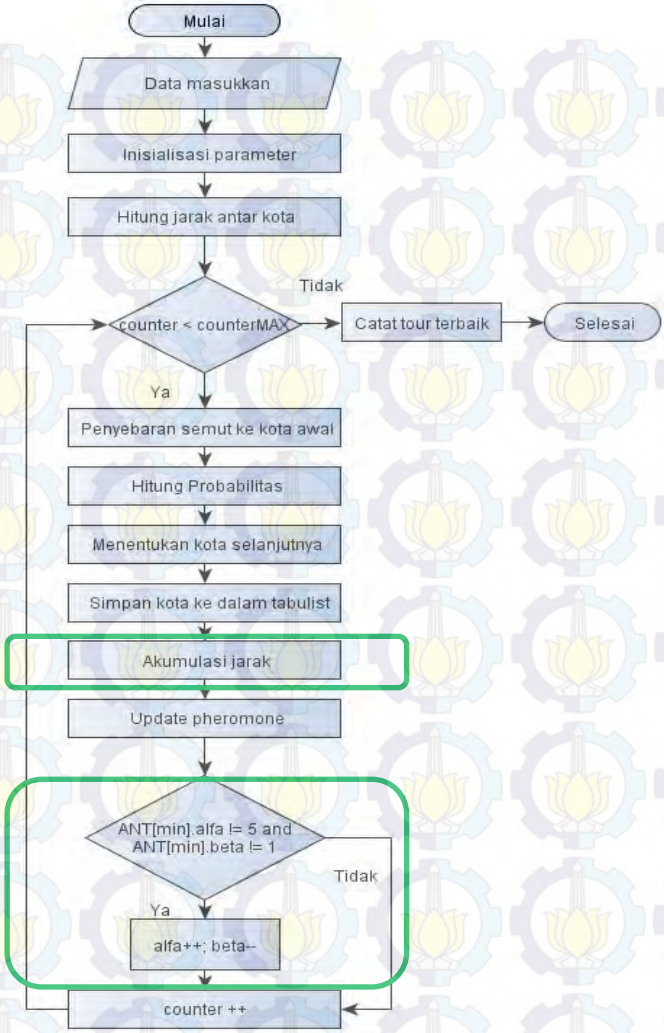
3.2 Proses Penyelesaian TSP Menggunakan Algoritma ACO dengan Metode ROIVA

Bagian ini menjelaskan bagaimana algoritma ACO dengan menggunakan metode ROIVA dapat menyelesaikan permasalahan TSP.

3.2.1 Desain Proses Metode ROIVA

Dalam proses penyelesaian TSP ada beberapa tahap pemrosesan. Tahapan pertama adalah melakukan inisialisasi data awal. Inisialisasi data awal adalah proses pengolahan data masukan. Setelah dilakukan inisialisasi data awal dilakukan inisialisasi permasalahan dengan menghitung jarak antar tiap kota dan dilakukan penyebaran semut-semut di kota awal perjalanan. Kota awal ini juga sebagai indeks pertama yang disimpan ke dalam *tabulist*. Kemudian semut dijalankan untuk mengelilingi kota. Pemilihan kota dipilih berdasarkan nilai probabilitas. Setelah semut berpindah kota, maka jarak dihitung.

Dari sejumlah *m* semut dijalankan, ditemukan satu semut yang memiliki panjang perjalanan terpendek. Setelah ditemukan semut yang memiliki jalur terpendek, solusi terbaik yang awalnya diinisialisasikan dengan tak hingga kemudian digantikan dengan jalur semut minimum. Jika semut yang melakukan perjalanan belum kembali ke kota awal perjalanan, yang artinya semut belum selesai melakukan perjalanan, maka semut memilih untuk menentukan kota mana yang akan dikunjungi selanjutnya. Kota yang disinggahi selanjutnya akan disimpan ke dalam *tabulist* sampai semut tersebut selesai mengunjungi seluruh kota atau *tabulist* terisi penuh. Jika semut sudah selesai melakukan perjalanan, dilakukan *update pheromone* untuk setiap jalur. Proses tersebut dilakukan sampai kondisi terpenuhi. Diagram alir proses penyelesaian TSP menggunakan algoritma ACO dengan metode ROIVA dapat dilihat pada Gambar 3.1.



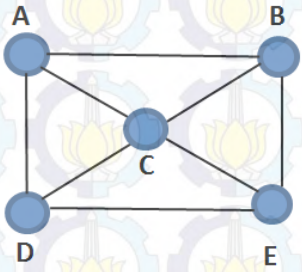
Gambar 3.1 Diagram Alir Proses Penyelesaian TSP Menggunakan Algoritma ACO dengan Metode ROIVA

Pada Gambar 3.1 terdapat bagian yang diberi tanda kotak menunjukkan proses metode ROIVA yang juga membedakan dengan ACO konvensional. Pada proses akumulasi jarak, didalamnya terdapat proses pencarian semut minimum. Pada ACO konvensional, proses pencarian semut minimum dilakukan saat semut telah selesai mengelilingi semua kota. Ketika semut telah mengelilingi semua kota, kemudian jarak total dihitung lalu dibandingkan dengan solusi minimum saat ini. Pada ACO ROIVA, pencarian semut minimum tidak menunggu sampai semut telah mengelilingi semua kota, tetapi saat setiap semut sudah berpindah kota jarak yang sudah diakumulasikan langsung dibandingkan dengan solusi minimum saat ini. Jika solusi saat ini ternyata lebih kecil nilainya dibandingkan dengan jarak semut tersebut, maka semut tersebut langsung diberhentikan perjalanannya pada iterasi saat ini. Cara tersebut yang dinamakan dengan *Routing Optimization*.

Proses *Individual Variaton* ditunjukkan pada kotak hijau yang kedua. Pada ACO konvensional, parameter α dan β bersifat statis. Pada ACO ROIVA, parameter α dan β bisa berubah di setiap iterasinya. Pada awalnya ditetapkan $\alpha = 1$dan $\beta = 5$, setelah proses berjalan maka ditemukan semut yang memiliki solusi minimum. Semut tersebut pada iterasi berikutnya berhak untuk mengubah parameter α dan β sesuai aturan yang sudah dijelaskan pada bab sebelumnya. Agar lebih mudah memahami perbedaan antara ACO konvensional dengan ACO menggunakan metode ROIVA dapat dilihat pada contoh kasus berikut. Gambar 3.2 merupakan graf lengkap, yaitu graf yang memiliki bobot disetiap sisinya (*edge*). Bobot yang dimiliki graf pada Gambar 3.2 adalah:

1. AB = 100
2. AD = 50
3. AE = 70
4. AC = 60
5. BD = 70
6. BE = 60
7. BC = 60

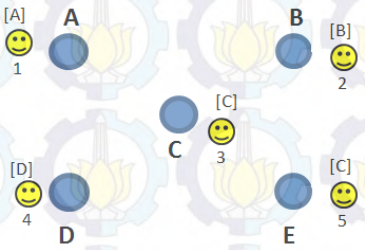
8. CD = 90
9. CE = 40
10. DE = 150



Gambar 3.2 Graf Berbobot

A. Iterasi pertama

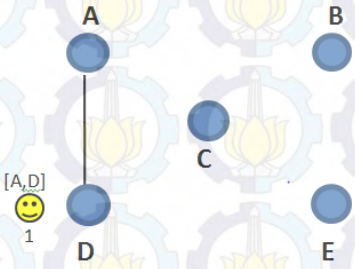
Pada iterasi pertama, langkah awal yang dilakukan adalah melakukan penyebaran semut disetiap kota yang artinya jumlah semut sama dengan jumlah kota. Masing-masing semut ditetapkan memiliki nilai parameter $\alpha = 1$, $\beta = 5$, dan $\rho = 0.6$. Kota awal semut langsung disimpan ke dalam *tabulist* pada setiap semut. Seperti yang dijelaskan pada Gambar 3.3.



Gambar 3.3 Penyebaran Semut di Setiap Kota

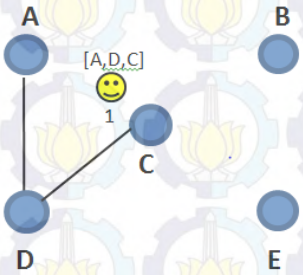
Selanjutnya, masing-masing semut akan melakukan perjalanan ke semua kota. Semut 1 memiliki 4 jalur berbeda yang

harus dipilih yaitu jalur AB = 100, AC = 60, AD = 50, dan AE = 70. Semut 1 akan memilih jalur berdasarkan sebuah nilai probabilitas yang dihitung dari kekuatan *pheromone* dan *inverse* dari jarak sesuai dengan Persamaan 2.1. Diasumsikan Semut 1 memilih jalur AD dan kota kedua yang dikunjungi semut disimpan ke dalam *tabulist*. Seperti yang dijelaskan pada Gambar 3.4.



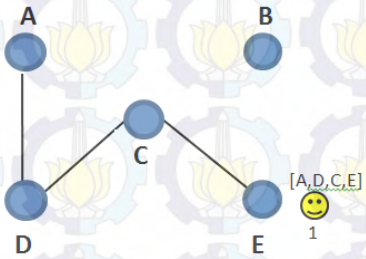
Gambar 3.4 Jalur Pertama Semut 1 (Iterasi 1)

Saat ini Semut 1 berada di kota D. Semut 1 memiliki 3 jalur berbeda untuk dipilih yaitu jalur DB = 70, DC = 90, DE = 150. Berdasarkan nilai probabilitas, diasumsikan Semut 1 memilih jalur DC dan kota ketiga yang dikunjungi disimpan ke dalam *tabulist.* Seperti yang dijelaskan pada Gambar 3.5.



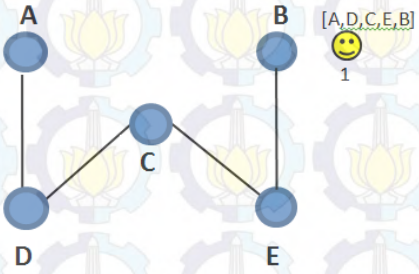
Gambar 3.5 Jalur Kedua Semut 1 (Iterasi 1)

Saat ini Semut 1 berada di kota C . Semut 1 memiliki 2 jalur berbeda untuk dipilih yaitu jalur CE = 40 dan CB = 60. Berdasarkan nilai probabilitas, diasumsikan Semut 1 memilih jalur CE dan kota keempat yang dikunjungi semut disimpan ke dalam *tabulist.* Seperti yang dijelaskan pada Gambar 3.6.



Gambar 3.6 Jalur Ketiga Semut 1 (Iterasi 1)

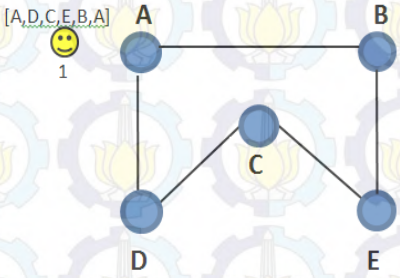
Saat ini Semut 1 berada di kota E. Semut 1 hanya tinggal memiliki 1 jalur untuk dipilih yaitu jalur EB = 60 karena kota lainnya sudah dikunjungi. Kota yang dikunjungi selanjutnya disimpan ke dalam *tabulist.* Seperti yang dijelaskan pada Gambar 3.7.



Gambar 3.7 Jalur Keempat Semut 1 (Iterasi 1)

Semut 1 berada di kota B. Semua kota sudah dikunjungi oleh Semut 1 dan Semut 1 harus kembali lagi ke kota awal

perjalanan. Maka jalur yang harus dilalui adalah jalur BA = 100. Jalur keseluruhan dijelaskan pada Gambar 3.8.



Gambar 3.8 Jalur Keseluruhan Semut 1 (Iterasi 1)

Cara yang sama dilakukan untuk semut lainnya. Setelah semua semut selesai melakukan perjalanan dan sudah kembali ke kota awal, maka hasil dari akumulasi jarak dari masing-masing semut adalah:

1. Semut 1 = A, D, C, E, B, A
   L1 = 50 + 90 + 40 + 60 + 100 = 340
2. Semut 2 = B, C, D, A, E, B
   L2 = 60 + 90 + 50 + 70 + 60 = 330
3. Semut 3 = C, B, E, D, A, C
   L3 = 60 + 60 + 150 + 50 + 60 = 380
4. Semut 4 = D, E, A, B, C, D
   L4 = 150 + 70 + 100 + 60 + 90 = 470
5. Semut 5 = E, A, B, C, D, E
   L5 = 70 + 100 + 60 + 90 + 150 = 470

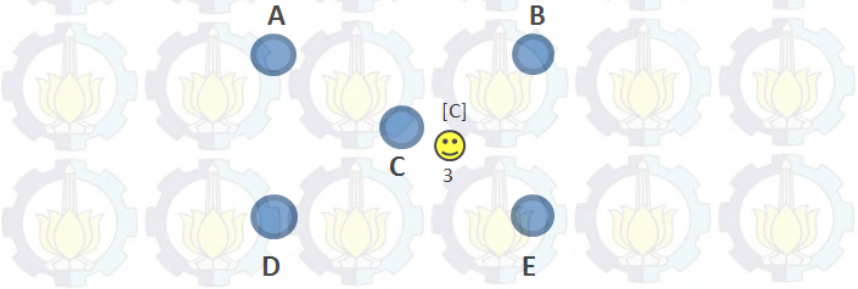
Dari akumulasi jarak diatas, maka didapat hasil solusi minimum lokal adalah jarak yang didapat oleh Semut 2 yaitu sebesar 330. Maka solusi minimum global untuk saat ini adalah Lmin = 330. Setelah didapat solusi minimum global, dilakukan *update pheromone* pada setiap jalur. Jalur yang dilewati,

pheromone akan diperkuat. Jalur yang tidak dilewati semut maka *pheromone* akan menguap.

B. Iterasi kedua

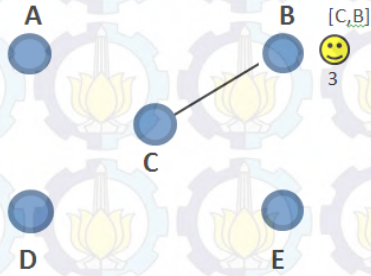
Pada iterasi yang kedua, cara yang sama dilakukan seperti pada iterasi pertama pada ACO konvensional. Setelah semua semut selesai melakukan perjalanan ke semua kota, maka akan didapat solusi minimum lokal pada iterasi kedua. Jika solusi minimum lokal nilainya lebih kecil daripada nilai pada iterasi sebelumnya maka solusi minimum global di *update.* Jika tidak, maka nilai solusi minimum global tetap.

Ada dua perbedaan dengan ACO ROIVA pada iterasi kedua. Perbedaan yang pertama adalah semut yang memiliki jalur minimum pada iterasi sebelumnya yaitu Semut 2 akan mengubah nilai parameter α dan β nya yaitu menjadi $\alpha = 2$ dan $\beta = 4$. Sementara semut lainnya memiliki nilai parameter α dan β sama dengan iterasi sebelumnya. Perbedaan kedua adalah setiap semut berpindah kota jarak yang diakumulasikan akan langsung dibandingkan dengan solusi minimum global. Jika semut tersebut memiliki nilai akumulasi jarak lebih besar daripada solusi minimum global maka semut tersebut langsung diberhentikan perjalanannya. Jika tidak, maka dilakukan pencarian semut minimum kembali. Untuk lebih jelasnya, akan dijelaskan pada graf berikut. Gambar 3.9 menunjukkan penempatan semut di kota awal.



Gambar 3.9 Penempatan Semut 3 di Kota C

Semut 3 berada di kota C. Semut 3 memiliki 4 jalur berbeda yang harus dipilih yaitu jalur CA = 60, CB = 60, CD = 90, dan CE = 40. Semut 3 akan memilih jalur berdasarkan sebuah nilai probabilitas yang dihitung dari kekuatan *pheromone* dan *inverse* dari jarak sesuai dengan Persamaan 2.1. Diasumsikan Semut 3 memilih jalur CB dan kota kedua yang dikunjungi semut disimpan ke dalam *tabulist*. Seperti yang dijelaskan pada Gambar 3.10.

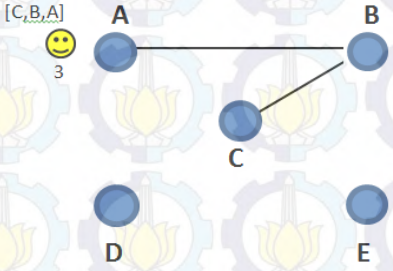


Gambar 3.10 Jalur Pertama Semut 3 (Iterasi 2)

Setelah berpindah kota dari kota C ke kota B, Semut 3 memiliki total jarak L3 = 60. Karena total jarak Semut 3 nilainya masih lebih kecil jika dibandingkan dengan solusi minimum global maka Semut 3 bisa melanjutkan perjalanan ke kota selanjutnya. Sekarang Semut 3 berada di kota B. Semut 3 memiliki 3 jalur berbeda untuk dipilih yaitu jalur BA = 100, BD = 70, dan BE = 60. Berdasarkan nilai probabilitas, diasumsikan Semut 3 memilih jalur BA dan kota ketiga yang dikunjungi semut disimpan ke dalam *tabulist.* Seperti yang dijelaskan pada Gambar 3.11.

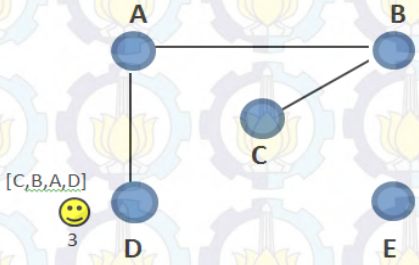
Setelah berpindah kota dari kota B ke kota A, Semut 3 memiliki total jarak L3 = 60 + 100 = 160. Karena total jarak yang ditempuh oleh Semut 3 nilainya masih lebih kecil jika dibandingkan dengan solusi minimum global maka Semut 3 bisa melanjutkan perjalanan ke kota selanjutnya. Sekarang Semut 3 berada di kota A. Semut 3 memiliki 2 jalur berbeda untuk dipilih

yaitu jalur AD = 60, dan AE = 70. Berdasarkan nilai probabilitas, diasumsikan Semut 3 memilih jalur AD dan kota keempat yang dikunjungi semut disimpan ke dalam *tabulist.* Seperti yang dijelaskan pada Gambar 3.12

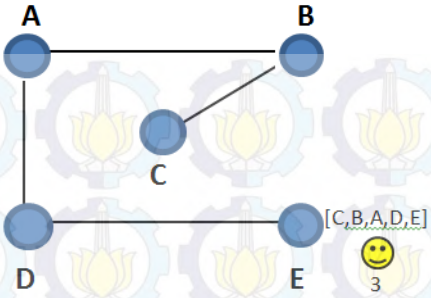


Gambar 3.11 Jalur Kedua Semut 3 (Iterasi 2)

Setelah berpindah kota dari kota A ke kota D, Semut 3 memiliki total jarak L3 = 60 + 100 + 50 = 210. Karena total jarak Semut 3 nilainya masih lebih kecil dibandingkan dengan solusi minimum global maka Semut 3 bisa melanjutkan perjalanan ke kota selanjutnya. Sekarang Semut 3 berada di kota D. Semut 3 hanya tinggal memiliki 1 jalur untuk dipilih yaitu jalur DE = 150 karena kota lainnya sudah dikunjungi. Kota yang dikunjungi selanjutnya disimpan ke dalam *tabulist*. Seperti yang dijelaskan pada Gambar 3.13.



Gambar 3.12 Jalur Ketiga Semut 3 (Iterasi 2)



Gambar 3.13 Jalur Keempat Semut 3 (Iterasi 2)

Setelah berpindah kota dari kota D ke kota E, Semut 3 memiliki total jarak tempuh L3 = 60 + 100 + 50 + 150 = 370. Karena total jarak tempuh Semut 3 nilainya ternyata sudah lebih besar jika dibandingkan dengan solusi minimum global maka Semut 3 diberhentikan perjalanannya sampai pada kota E saja dan tidak bias melanjutkan perjalanan ke kota selanjutnya.

Cara yang sama dilakukan pada semut lainnya yaitu Semut 1, Semut 2, Semut 4, dan Semut 5. Maka didapat akumulasi jarak dari masing-masing semut sebagai berikut:

1. Semut 1 = A, D, B, E, C, A
   L1 = 50 + 70 + 60 + 40 + 60 = 280
2. Semut 2 = B, A, E, C, D,B
   L2 = 100 + 70 + 40 + 90 + 70 = 370
3. Semut 3 = C, B, A, D, E
   L3 = 60 + 100 + 50 + 150 = 370
4. Semut 4 = D, C, E, B, A, D
   L4 = 90 + 40 + 60 + 100 + 50 = 340
5. Semut 5 = E, A, D, C, B, E
   L5 = 70 + 90 + 50 + 60 + 60 = 330

Dari akumulasi jarak diatas, maka didapat hasil solusi minimum lokal yaitu solusi minimum dari semua semut pada satu iterasi adalah jarak yang didapat oleh Semut 1 yaitu sebesar 280. Maka solusi minimum global atau solusi minimum untuk

keseluruhan dari seluruh iterasi untuk saat ini adalah Lmin = 280. Setelah didapat solusi minimum global, dilakukan *update pheromone* pada setiap jalur. Jalur yang dilewati oleh semut, maka *pheromone* akan diperkuat. Untuk jalur yang tidak dilewati oleh semut maka *pheromone* akan menguap. Semakin banyak semut yang melewati jalur tersebut maka akan semakin kuat juga *pheromone* yang ditinggalkan. Semakin jalur tersebut tidak dilewati oleh semut maka *pheromone* akan semakin banyak menguap.

BAB 4
IMPLEMENTASI

Bab ini berisi penjelasan mengenai implementasi perangkat lunak yang meliputi algoritma dan kode sumber yang terdapat dalam perangkat lunak.

4.1 Lingkungan Implementasi Program

Lingkungan implementasi perangkat lunak yang digunakan dalam pembuatan Tugas Akhir ini meliputi perangkat lunak dan perangkat keras yang dijelaskan sebagai berikut:

1. Perangkat keras
   a. Prosesor: Intel® Core™ i5-3210M CPU @ 2.50 GHz
   b. Memory (RAM): 4 GB
   c. Tipe sistem: 64-bit sistem operasi
2. Perangkat lunak
   a. Sistem operasi: Windows 8.1 Single Language
   b. Perangkat pengembang: MATLAB 8.3.0 (R2014a)

4.2 Implementasi

Implementasi dilakukan sesuai dengan proses yang dijelaskan pada bab sebelumnya.

4.2.1 Proses Pembacaan Data Masukan

Tahap ini akan dilakukan proses bagaimana data masukan *file.tsp* diproses. Variabel *Dimension* menyimpan jumlah *nodes*. Variabel *NodeCoord* menyimpan koordinat sumbu *x* dan sumbu *y* dari *nodes*. Variabel *NodeWeight* menyimpan bobot *nodes*. Dan variabel *Name* menyimpan nama *nodes*. Proses implementasi dapat dilihat pada Kode Sumber 4.1.

4.2.2 Proses Inisialisasi Data

Pada tahapan ini dilakukan inisialisasi data awal, yaitu menentukan nilai awal untuk solusi minimum, menentukan data

```
1 [Dimension,NodeCoord,NodeWeight,Name]=
  FileInput(inputfile);
2 disp([num2str(Dimension),'nodes
  in',Name]);
3 C =
  sprintf('%s%s%s%s%c',C,num2str(Dimension),
  'nodes in',Name);
4 set(handles.edProses,'string',C)
5 drawnow;
```

Kode Sumber 4.1 Proses Pembacaan *File*

jumlah iterasi maksimum, data jumlah semut, data koefisien nilai penguapan *pheromone*, data nilai α, dan data nilai β.

4.2.3 Proses Inisialisasi Data

Pada tahapan ini akan dilakukan inisialisasi data awal, yaitu menentukan nilai awal untuk solusi minimum, menentukan data jumlah iterasi maksimum, data jumlah semut, data koefisien nilai penguapan *pheromone*, data nilai α, dan data nilai β.

Nilai awal untuk solusi minimum ditentukan dengan nilai ∞ dan disimpan dalam variabel *Lmin*. Data masukan awal merupakan data masukan dengan format tsp (*filename.tsp*) yang terdiri dari urutan *nodes*, koordinat *x*, dan koordinat *y* dari setiap *nodes*. Data jumlah iterasi merupakan nilai yang dimasukan untuk menentukan jumlah perulangan yang diinginkan, dan disimpan dalam variable *MaxITime*. Data jumlah semut merupakan nilai yang dimasukan untuk menentukan jumlah semut yang diinginkan untuk melakukan perjalanan, dan disimpan dalam variabel *AntNum*. Banyaknya jumlah semut ditentukan berdasarkan jumlah kota yang akan dikelilingi. Data koefisien nilai penguapan *pheromone* merupakan nilai yang dibutuhkan pada proses pembaruan *pheromone*, dan disimpan dalam variabel *rho*. Data koefisien nilai α adalah nilai koefisien dari α yang berarti tingkat kepentingan dari *pheromone*. Koefisien α ditentukan bernilai 1, dan disimpan dalam variabel *alpha*. Data

nilai β adalah nilai koefisien dari β yang berarti tingkat kepentingan dari *distance* atau jarak. Koefisien β ditentukan bernilai 5, dan disimpan dalam variabel *beta*. Proses implementasi dari inisialisasi data dapat dilihat pada Kode Sumber 4.2 berikut:

```
1  MaxITime=30;
2  AntNum=Dimension;
3  alpha=1;
4  beta=5;
5  rho=0.9;
6  LMin = 999999;
```

Kode Sumber 4.2 Inisialiasi Data Awal

4.2.4 Proses Inisialisasi Awal untuk Tiap Semut

Kode Sumber 4.3 menunjukkan proses inisialisasi awal untuk tiap semut sebelum melakukan perjalanan meliputi rute tiap semut dan jarak tempuh tiap semut. Saat awal rute tiap semut dan jarak tempuh tiap semut diinisialisasikan dengan 0.

```
1  function Ant = InitAnt()
2  global ASOption
4  AntTours = zeros(ASOption.m,ASOption.n+1);
6  ToursLength = zeros(ASOption.m,1);
8  Ant =
   struct('tours',AntTours,'lengths',ToursLength
   );
```

Kode Sumber 4.3 Inisialisasi Awal Tiap Semut

4.2.5 Inisialisasi Permasalahan

Tahap ini menunjukkan proses inisialisasi permasalahan. Jumlah kota atau *node* disimpan dalam variabel *n*. Jarak antar kota disimpan dalam matriks *Distances*. Proses inisialisasi permasalahan dapat dilihat pada Kode Sumber 4.4.

```
1  function Problem =
   InitProblem(Nodes,WeightMatrix)
2  global ASOption
3  n = length(Nodes(:,1));
4  Distances = WeightMatrix;
5  SymmetryFlag = false;
6  if isempty(WeightMatrix)
7      Distances = CalculateDistance(Nodes);
8     SymmetryFlag = true;
9  end
```

Kode Sumber 4.4 Inisialisasi Permasalahan

4.2.6 Proses Perhitungan Jarak Antar Kota

Tahap ini melakukan perhitungan jarak antar kota (*nodes*) yang didapatkan dari data masukan. Perhitungan menggunakan persamaan *Euclidean* seperti yang sudah dijelaskan pada Persamaan 2.2. Proses perhitungan dapat dilihat pada Kode Sumber 4.5 dan Kode Sumber 4.6.

```
1  function Distances = CalculateDistance(Nodes)
2  global ASOption
3  Nodes(:,1)=[];
4  Distances=zeros(ASOption.n,ASOption.n);
5  for i=2:ASOption.n
6      for j=1:i
7          if(i==j)
8              continue;
9          Else
10       dij=Nodes(i,:)-Nodes(j,:);
11   Distances(i,j)=sqrt(dij(1)^2+dij(2)^2);
```

Kode Sumber 4.5 Proses Perhitungan Jarak antar Kota (1)

```
12          Distances(j,i)=Distances(i,j);
13            End
14        End
15  End
```

Kode Sumber 4.6 Proses Perhitungan Jarak antar Kota (2)

4.2.7 Proses Inisialisasi *Tabulist*

Tabulist menyimpan kota-kota yang sudah dilalui semut. Tiap semut yang sudah berpindah kota maka kota tersebut disimpan ke dalam *tabulis*t. *Tabulist* digunakan untuk menghindari semut mengunjungi kota yang sudah dikunjungi. Proses inisialisasi awal untuk *tabulist* dapat dilihat pada Kode Sumber 4.7.

```
1  function RefreshTabu(step_i,ant_k,nextnode)
2  global Ant
3  Ant.tours(ant_k,step_i) = nextnode;
```

Kode Sumber 4.7 Inisialisasi Awal *Tabulist*

4.2.8 Proses Penempatan Semut di Kota Awal

Kode Sumber 4.8 menunjukkan proses penempatan semut di kota awal, dimana semut akan memulai dan mengakhiri perjalanan di kota ini. Baris 3 - 4 menunjukkan inisialisasi matriks rute semut. Baris 5 menunjukkan inisialisasi nilai random untuk matriks rute semut. Baris 6 menunjukkan inisialisasi matriks jarak tempuh dari tiap rute semut.

```
1  function InitStartPoint()
2  global Ant ASOption
3  Ant.tours = zeros(ASOption.m,ASOption.n+1);
4  rand('state',sum(100*clock));
5  Ant.tours(:,1) =
   randint(ASOption.m,1,[1,ASOption.n]);
6  Ant.lengths = zeros(ASOption.m,1);
```

Kode Sumber 4.8 Penempatan Semut di Kota Awal

4.2.9 Perhitungan Nilai Probabilitas

Pada tahapan ini akan dilakukan proses perhitungan nilai probabilitas, dimana dimulainya proses perjalanan semut. Nilai probabilitas digunakan untuk menentukan kota tujuan berikutnya pada semut jika masih dalam perjalanan atau belum sampai pada tujuan (kota awal). Jalur yang memiliki nilai *pheromone* lebih besar memiliki kemungkinan yang lebih besar untuk dilewati. Baris 3 menentukan *node* yang saat ini sedang ditempuh. Baris 4 menentukkan *node* yang telah dilewati. Baris 5 – 6 menentukkan nilai pembaruan *pheromone* sesuai *node* saat ini. Baris 7 – 8 menentukkan nilai jarak sesuai *node* saat ini. Baris 9 merupakan rumus untuk menghitung nilai probabilitas yang sudah dijelaskan pada Persamaan 2.1. Proses perhitungan probabilitas dapat dilihat pada Kode Sumber 4.9 dan Kode Sumber 4.10.

```
1  function Probs = CaculateShiftProb(step_i, ant_k)
2  global Ant ASOption Problem
3  CurrentNode = Ant.tours(ant_k, step_i-1);
4  VisitedNodes = Ant.tours(ant_k, 1:step_i-1);
5  tau_i = Problem.tau(CurrentNode,:);
6  tau_i(1,VisitedNodes) = 0;
7  dis_i = Problem.dis(CurrentNode,:);
8  dis_i(1,CurrentNode) = 1;
10 Probs = (tau_i.^ASOption.alpha).*((1./dis_i).^ASOption.beta);
11 if sum(Probs) ~= 0
12     Probs = Probs/sum(Probs);
13 else
14 NoVisitedNodes = setdiff(1:ASOption.n,VisitedNodes);
15     Probs(1,NoVisitedNodes) = 1/length(NoVisitedNodes);
16 End
```

Kode Sumber 4.9 Perhitungan Nilai Probabilitas (1)

```
17  global Ant ASOption Problem
18  CurrentNode = Ant.tours(ant_k, step_i-1);
19  VisitedNodes = Ant.tours(ant_k, 1:step_i-1);
20  tau_i = Problem.tau(CurrentNode,:);
21  tau_i(1,VisitedNodes) = 0;
```

Kode Sumber 4.10 Perhitungan Nilai Probabilitas (2)

4.2.10 Perhitungan *Cost*

Pada tahapan ini akan dilakukan proses perhitungan *cost*, yaitu proses menghitung total jarak tempuh pada jalursemut untuk menyelesaikan TSP. Proses perhitungan dapat dilihat pada Kode Sumber 4.11.

```
1   function CaculateToursLength()
2   global Ant ASOption Problem
3   Lengths = zeros(ASOption.m,1);
4   for k=1:ASOption.m
5       for i=1:ASOption.n
6           Lengths(k)=Lengths(k)+...
7   Problem.dis(Ant.tours(k,i),Ant.tours(k,i+
    1));
8       End
9   End
10  Ant.lengths = Lengths;
```

Kode Sumber 4.11 Perhitungan *Cost*

4.2.11 Proses Pembaruan Nilai *Pheromone*

Pada tahapan ini dilakukan pembaruan nilai *pheromone*. Nilai *pheromone* mengalami peningkatan pada jalur yang dilewati semut. Jalur yang tidak dilewati semut, *pheromone* akan mengalami penguapan. Semakin jalur tersebut banyak dilewati oleh semut maka nilai *pheromone* akan semakin tinggi. Proses pembaruan nilai *pheromone* dapat dilihat pada Kode Sumber 4.12.

```
function GlobleRefreshPheromone()
global Ant ASOption Problem
AT = Ant.tours;
TL = Ant.lengths;
sumdtau=zeros(ASOption.n,ASOption.n);
for k=1:ASOption.m
    for i=1:ASOption.n
sumdtau(AT(k,i),AT(k,i+1))=sumdtau(AT(k,i),AT(k,i+1))+1/TL(k);
        if Problem.symmetry
sumdtau(AT(k,i+1),AT(k,i))=sumdtau(AT(k,i),AT(k,i+1));
        End
    End
End
Problem.tau=Problem.tau*(1-ASOption.rho)+sumdtau;
```

Kode Sumber 4.12 Pembaruan *Pheromone*

4.2.9 Proses *Routing Optimization and Individual Variation (ROIVA)*

Pada tahapan ini dilakukan proses metode ROIVA. Nilai α dan β memungkinkan untuk berubah pada iterasi berikutnya pada semut minimum. Semut yang memiliki jarak tempuh minimum memungkinkan untuk menaikkan nilai parameter α dan menurunkan nilai parameter β. Penjelasan lengkap mengenai metode ROIVA sudah dijelaskan pada Bab 3. Implementasi metode dapat dilihat pada Kode Sumber 4.13 dan Kode Sumber 4.14.

```
if(iv==1)
if(ANB < LMin) LMin = ANB;
if(ASOption.alpha ~= beta && ASOption.beta ~= alpha)
    ASOption.alpha = ASOption.alpha + 1;
```

Kode Sumber 4.13 Implementasi ROIVA (1)

```
5           ASOption.beta = ASOption.beta - 1;
6       end;
7       Else
8       if Terminate(ITime,ANB)
9       break;
10              End
7           end;
8       end;
```

Kode Sumber 4.14 Implementasi ROIVA (2)

4.2.10 Proses Pencarian Jalur Terpendek

Pada tahapan ini akan dilakukan proses pencarian jalur terpendek, yaitu proses menentukan sebuah jalur dari beberapa jalur yang sudah dibuat oleh semut. Jalur dengan nilai *cost* terendah dipilih sebagai jalur terpendek. Proses pencarian jalur terpendek dapat dilihat pada Kode Sumber 4.15.

```
1   function [GBTour,GBLength,Record] =
    GetResults(ITime,ANB)
2   global Ant ASOption
3   [IBLength,AntIndex] = min(Ant.lengths);
4   IBTour = Ant.tours(AntIndex,:);
5   if IBLength<=ASOption.GBLength
6       ASOption.GBLength = IBLength;
7       ASOption.GBTour = IBTour;
8       ASOption.OptITime = ITime;
9   End
10  GBTour = ASOption.GBTour';
11  GBLength = ASOption.GBLength;
12  Record = [IBLength,ANB,IBTour]';
```

Kode Sumber 4.15 Pencarian Jalur Terpendek

BAB 5
UJI COBA DAN EVALUASI

Bab ini menjelaskan uji coba yang dilakukan pada aplikasi yang telah dikerjakan serta analisis dari uji coba yang telah dilakukan. Pembahasan pengujian meliputi lingkungan uji coba, data uji coba, dan skenario uji coba yang meliputi uji perbandingan dan analisis setiap pengujian.

5.1 Lingkungan Uji Coba

Lingkungan uji coba yang digunakan dalam pembuatan Tugas Akhir ini meliputi perangkat lunak dan perangkat keras. Lingkungan uji coba merupakan komputer tempat uji coba yang digunakan pada Tugas Akhir ini yang dijelaskan sebagai berikut:

1. Perangkat keras
   a. Prosesor: Intel® Core™ i5-3210M CPU @ 2.50 GHz
   b. Memory (RAM): 4 GB
   c. Tipe sistem: 64-bit sistem operasi
2. Perangkat lunak
   a. Sistem operasi: Windows 8.1 Single Language
   b. Perangkat pengembang: MATLAB 8.3.0 (R2014a)

5.2 Data Uji Coba

Data yang digunakan pada uji coba ini adalah sebuah graf yang setiap *nodes*-nya terhubung satu sama lain (*fully connected graph*). Dalam pengujian aplikasi ini data yang digunakan didapatkan dari TSPLIB. Terdapat 8 *dataset* yang akan digunakan dalam pengujian aplikasi Tugas Akhir ini. *Dataset* tersebut adalah:

a. Ulysses16

Dataset ulysses16 memiliki 16 *nodes* yang harus dikunjungi seluruhnya. Titik koordinat dari ulysses16 dapat dilihat pada Tabel A.1.

b. Eil76

Dataset eil76 memiliki 76 *nodes* yang harus dikunjungi seluruhnya. Titik koordinat dari eil76 dapat dilihat pada Tabel A.2.

c. Gr96

Dataset gr96 memiliki 96 *nodes* yang harus dikunjungi seluruhnya. Titik koordinat dari gr96 dapat dilihat pada Tabel A.3.

d. Gr137

Dataset gr137 memiliki 137 *nodes* yang harus dikunjungi seluruhnya. Titik koordinat dari gr137 dapat dilihat pada Tabel A.4.

e. Ch150

Dataset ch150 memiliki 150 *nodes* yang harus dikunjungi seluruhnya. Titik koordinat dari ch150 dapat dilihat pada Tabel A.5.

f. D198

Dataset d198 memiliki 198 *nodes* yang harus dikunjungi seluruhnya.

g. Pr299

Dataset d198 memiliki 198 *nodes* yang harus dikunjungi seluruhnya.

h. D493

Dataset d198 memiliki 198 *nodes* yang harus dikunjungi seluruhnya.

5.3 Skenario Uji Coba

Uji coba dilakukan untuk menguji apakah fungsionalitas aplikasi telah diimplementasikan dengan benar dan berjalan sebagaimana mestinya. Uji coba akan didasarkan pada beberapa skenario untuk menguji kesesuaian dan kinerja aplikasi.

Terdapat tiga skenario yang diujicobakan. Ketiga skenario tersebut adalah uji coba pengubahan jumlah *nodes* data masukan, parameter ρ (tingkat penguapan *pheromone*), dan

jumlah iterasi yang dilakukan. Pengubahan jumlah *nodes* dilakukan dengan 16 *nodes*, 76 *nodes*, 96 *nodes*, 137 *nodes*, 150 *nodes*, 198 *nodes*, 299 *nodes*, dan 493 *nodes*. Parameter ρ yang diubah meliputi 0.1 [5], 0.6, dan 0.9 [4]. Perubahan jumlah iterasi yang dilakukan adalah sebanyak 10 iterasi, 20 iterasi, dan 30 iterasi. Setiap skenario akan diuji sebanyak 10 kali percobaan.

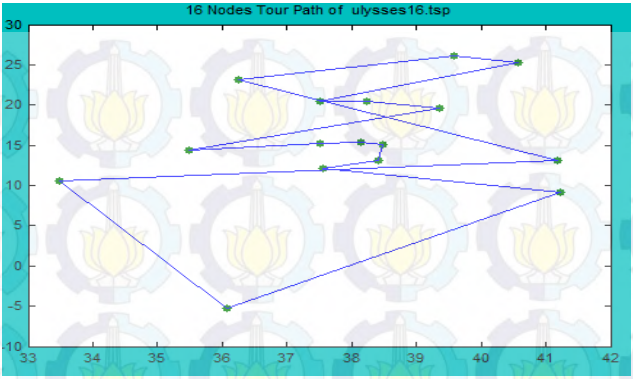
Pada uji coba kali ini hasil uji coba akan dibandingkan dengan hasil dari TSPLIB (http://www.iwr.uni-heidelberg.de/groups/comopt/software/TSPLIB95/STSP.html). Hasil dari uji coba untuk *dataset* D198 dapat dilihat pada Tabel B.21, Pr299 dapat dilihat pada Tabel B.22 dan Tabel B.23, dan D493 dapat dilihat pada Tabel B.24 dan untuk gambar jalur terbaik dapat dilihat pada Gambar B.1, Gambar B.2, dan Gambar B.3.

5.3.1 Uji Coba Ulysses16

Pada permasalahan ulysses16 terdapat 16 kota yang akan dilalui. Koordinat dari ulysses16 terdapat pada Tabel A.1. Hasil uji coba dapat dilihat pada Tabel 5.1. Hasil pengujian akan dibandingkan dengan hasil dari ACO konvensional. Hasil jalur terbaik ditunjukkan pada Gambar 5.1.

Tabel 5.1 Hasil Uji Coba untuk Ulysses16

		Solusi Terbaik	Jumlah Routing	Waktu (s)
Iterasi 10	ρ = 0.10	75.37	139	2.19
	ρ = 0.60	74.87	138	2.33
	ρ = 0.90	74.95	139	2.06
Iterasi 20	ρ = 0.10	74.84	138	4.46
	ρ = 0.60	74.69	138	2.81
	ρ = 0.90	74.61	137	1.57
Iterasi 30	ρ = 0.10	75.60	138	6.53
	ρ = 0.60	75.18	139	7.56
	ρ = 0.90	75.30	137	3.81



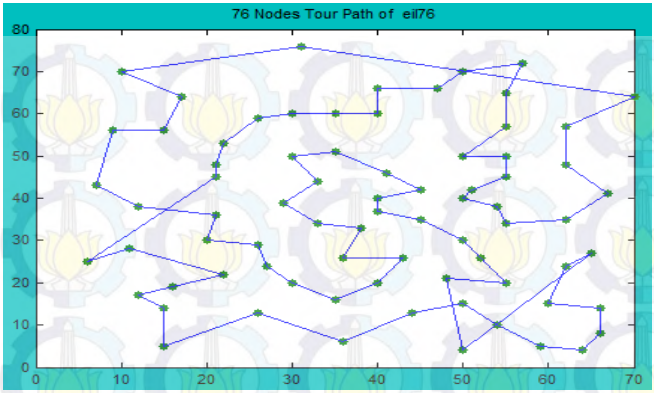
Gambar 5.1 Jalur Terbaik Ulysses16

Solusi terbaik didapatkan pada uji coba dengan parameter ρ = 0.9 dan dilakukan sebanyak 20 iterasi dengan nilai 74.61. Pada solusi terbaik, jumlah *routing* yang dilakukan sebanyak 137 kali dan memiliki kompleksitas waktu sebesar 1.57 detik.

5.3.2 Uji Coba Eil76

Pada permasalahan eil76 ini terdapat 76 kota yang akan dilalui seluruhnya. Koordinat dari eil76 terdapat pada Tabel A.2. Hasil percobaan penyelesaian TSP dapat dilihat pada Tabel 5.2. Hasil pengujian nantinya akan dibandingkan dengan hasil dari algoritma ACO konvensional. Hasil jalur tempuh terbaik ditunjukkan pada Gambar 5.2.

Dapat dilihat pada Tabel 5.2 bahwa solusi terbaik terdapat pada uji coba dengan parameter ρ = 0.9 dan dilakukan sebanyak 20 iterasi dengan nilai 570.06. Pada solusi terbaik, jumlah *routing* yang dilakukan sebanyak 2934 kali dan memiliki kompleksitas waktu sebesar 29.91 detik.



Gambar 5.2 Jalur Terbaik Eil76

Tabel 5.2 Hasil Uji Coba untuk Eil76

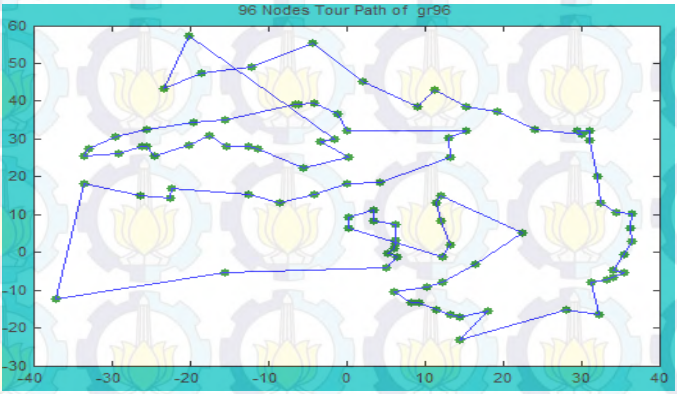
		Solusi Terbaik	Jumlah Routing	Waktu (s)
Iterasi 10	ρ = 0.10	661.95	2931	14.63
	ρ = 0.60	585.95	2932	15.16
	ρ = 0.90	572.45	2960	15.24
Iterasi 20	ρ = 0.10	646.70	2939	29.57
	ρ = 0.60	576.58	2945	29.73
	ρ = 0.90	570.06	2934	29.91
Iterasi 30	ρ = 0.10	626.65	2939	46.44
	ρ = 0.60	577.63	2954	22.49
	ρ = 0.90	579.54	2948	14.94

5.3.3 Uji Coba Gr96

Pada permasalahan gr96 ini terdapat 96 kota yang akan dilalui seluruhnya. Koordinat dari gr96 terdapat pada Tabel A.3. Hasil percobaan penyelesaian TSP dapat dilihat pada Tabel 5.3. Hasil pengujian nantinya akan dibandingkan dengan hasil dari algoritma ACO konvensional. Hasil jalur tempuh terbaik ditunjukkan pada Gambar 5.3.

Tabel 5.3 Hasil Uji Coba untuk Gr96

		Solusi Terbaik	Jumlah Routing	Waktu (s)
Iterasi 10	ρ = 0.10	625.63	4685	20.71
	ρ = 0.60	564.98	4711	23.28
	ρ = 0.90	558.41	4668	17.13
Iterasi 20	ρ = 0.10	618.00	4695	45.37
	ρ = 0.60	550.78	4713	29.94
	ρ = 0.90	558.86	4683	19.42
Iterasi 30	ρ = 0.10	609.55	4689	70.95
	ρ = 0.60	558.71	4719	27.77
	ρ = 0.90	554.77	4671	17.23

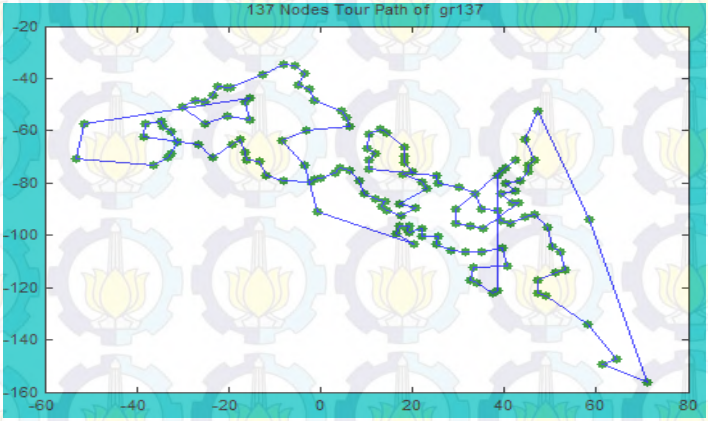


Gambar 5.3 Jalur Terbaik Gr96

Dapat dilihat pada Tabel 5.3 bahwa solusi terbaik terdapat pada uji coba dengan parameter ρ = 0.6 dan dilakukan sebanyak 20 iterasi dengan nilai 550.78. Pada solusi terbaik, jumlah *routing* yang dilakukan sebanyak 4713 kali dan memiliki kompleksitas waktu sebesar 29.94 detik.

5.3.4 Uji Coba Gr137

Pada permasalahan gr137 ini terdapat 137 kota yang akan dilalui seluruhnya. Koordinat dari gr137 terdapat pada Tabel A.4 dan Tabel A.5. Hasil percobaan penyelesaian TSP dapat dilihat pada Tabel 5.4. Hasil pengujian nantinya akan dibandingkan dengan hasil dari algoritma ACO konvensional. Hasil jalur tempuh terbaik ditunjukkan pada Gambar 5.4.



Gambar 5.4 Jalur Terbaik Gr137

Tabel 5.4 Hasil Uji Coba untuk Gr137

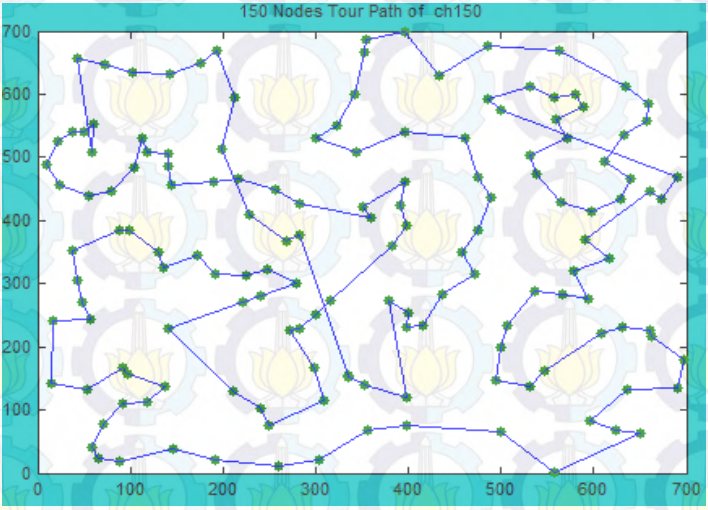
		Solusi Terbaik	Jumlah Routing	Waktu (s)
Iterasi 10	ρ = 0.10	930.72	9474	35.80
	ρ = 0.60	844.56	9496	39.25
	ρ = 0.90	825.35	9476	37.56
Iterasi 20	ρ = 0.10	920.49	9454	77.87
	ρ = 0.60	818.21	9472	74.58
	ρ = 0.90	838.05	9480	40.61
Iterasi 30	ρ = 0.10	932.30	9461	116.47
	ρ = 0.60	816.32	9475	100.06
	ρ = 0.90	844.40	9470	98.10

Solusi terbaik didapatkan pada uji coba dengan parameter ρ = 0.6 dan dilakukan sebanyak 30 iterasi dengan nilai 816.32. Pada solusi terbaik, jumlah *routing* yang dilakukan sebanyak 9475 kali dan memiliki kompleksitas waktu sebesar 100.06 detik.

5.3.5 Uji Coba Ch150

Pada permasalahan ch150 ini terdapat 150 kota yang akan dilalui seluruhnya. Koordinat dari ch150 terdapat padaTabel A.6 dan Tabel A.7. Hasil percobaan penyelesaian TSP dapat dilihat pada Tabel 5.5. Hasil pengujian nantinya akan dibandingkan dengan hasil dari algoritma ACO konvensional. Hasil jalur tempuh terbaik ditunjukkan pada Gambar 5.5.

Solusi terbaik didapatkan pada uji coba dengan parameter ρ = 0.6 dan dilakukan sebanyak 20 iterasi dengan nilai 6762.64. Pada solusi terbaik, jumlah *routing* yang dilakukan sebanyak 11368 kali dan memiliki kompleksitas waktu sebesar 91.16 detik.



Gambar 5.5 Jalur Terbaik Ch150

Tabel 5.5 Hasil Uji Coba untuk Ch150

		Solusi Terbaik	Jumlah Routing	Waktu (s)
Iterasi 10	ρ = 0.10	8278.76	11371	44.65
	ρ = 0.60	6977.43	11378	45.62
	ρ = 0.90	6841.00	11382	40.67
Iterasi 20	ρ = 0.10	8257.83	11370	89.61
	ρ = 0.60	6762.64	11368	91.16
	ρ = 0.90	6817.48	11378	86.97
Iterasi 30	ρ = 0.10	8298.31	11355	133.38
	ρ = 0.60	6859.73	11368	118.17
	ρ = 0.90	6773.49	11349	69.41

5.4 Evaluasi Uji Coba

Pada subbab ini akan dilakukan evaluasi dari hasil semua uji coba yang sudah dilakukan pada subbab sebelumnya. Hasil yang didapat dari uji coba akan dibandingkan dengan hasil dari algoritma ACO konvensional.

5.4.1 Perbandingan Solusi Terbaik

Pada tahap ini solusi terbaik akan dievaluasi. Hasil solusi terbaik akan dibandingkan dengan hasil yang didapatkan dari TSPLIB dan ACO konvensional. Berdasarkan hasil uji coba yang sudah dilakukan, hasil dikatakan lebih baik jika memiliki nilai lebih kecil. Secara keseluruhan perbandingan solusi terbaik ditunjukkan pada Tabel 5.6 dan Gambar 5.6.

Solusi terbaik yang dihasilkan dari uji coba menggunakan ACO ROIVA memiliki nilai yang masih lebih besar jika dibandingkan dengan hasil yang didapatkan dari TSPLIB. Jika dibandingkan dengan solusi terbaik yang dihasilkan algoritma ACO konvensional, maka solusi terbaik pada ACO ROIVA pada beberapa *dataset* memiliki nilai yang lebih besar. Dan di beberapa *dataset* lainnya memiliki nilai yang lebih kecil. Jika dilihat perbedaannya, hasil solusi terbaik yang dihasilkan ACO ROIVA

memiliki perbedaannya yang tidak berbeda jauh dengan hasil dari algoritma ACO.

Tabel 5.6 Perbandingan Solusi Terbaik

Dataset	Solusi Terbaik		
	TSPLIB	ACO	ACO ROIVA
Ulysses16	68.59	74.62	74.61
Eil76	538.00	563.51	570.06
Gr96	552.09	544.31	550.78
Gr137	698.53	815.15	816.32
Ch150	6528.00	6675.82	6762.64
D198	15780.00	17323.76	17269.52
Pr299	48191.00	55804.45	55568.16
D493	35002.00	40119.01	39976.03

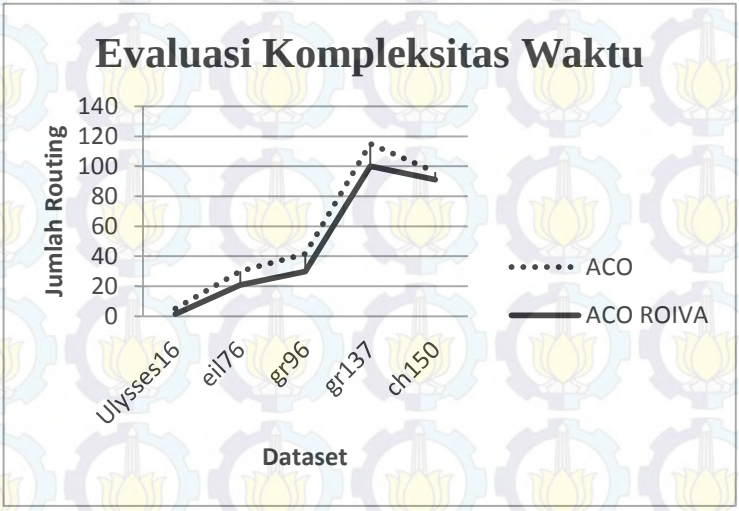
5.4.2 Evaluasi Kompleksitas Waktu

Pada tahap ini kompleksitas waktu akan dievaluasi. Berdasarkan hasil uji coba yang sudah dilakukan pada, hasil dikatakan lebih baik jika memiliki nilai lebih kecil. Evaluasi kompleksitas waktu dilakukan berdasarkan pada solusi terbaik. Secara keseluruhan hasil kompleksitas waktu ditunjukkan pada Tabel 5.7 dan Gambar 5.7.

Dapat dilihat pada Tabel 5.7 dan Gambar 5.6 bahwa kompleksitas waktu berbanding lurus dengan jumlah *nodes* pada permasalahan TSP. Semakin banyak *nodes*, maka semakin besar waktu yang dibutuhkan untuk menyelesaikan permasalahan tersebut. Kompleksitas waktu dari ACO ROIVA dalam menyelesaikan permasalahan TSP jika dibandingkan dengan kompleksitas waktu dari ACO konvensional memiliki waktu yang lebih sedikit dalam menyelesaikan permasalahan TSP. Dan memiliki rata-rata penurunan sebesar 24.80%.

Tabel 5.7 Evaluasi Kompleksitas Waktu

Dataset	ACO (s)	ACO ROIVA (s)	Penurunan (%)
Ulysses16	5.04	1.57	68.85
Eil76	30.12	20.91	30.58
Gr96	41.57	29.94	27.98
Gr137	115.05	100.06	13.03
Ch150	96.21	91.16	5.25
D198	620.07	524.16	15.47
Pr299	907.45	736.11	18.88
D493	1367.09	1115.85	18.38
		Rata-rata	24.80



Gambar 5.6 Grafik Evaluasi Kompleksitas Waktu

5.4.3 Evaluasi Jumlah *Routing*

Pada tahap ini jumlah *routing* akan dievaluasi. Berdasarkan hasil uji coba yang sudah dilakukan pada subbab sebelumnya, hasil dikatakan lebih baik jika memiliki nilai lebih

kecil. Jumlah *routing* dilakukan berdasarkan pada solusi terbaik Secara keseluruhan jumlah *routing* ditunjukkan pada Tabel 5.8.

Tabel 5.8 Evaluasi Jumlah *Routing*

Dataset	ACO	ACO ROIVA	Penurunan (%)
Ulysses16	147	137	6.80
Eil76	2975	2948	0.91
Gr96	4751	4671	1.68
Gr137	9538	9470	0.71
Ch150	11430	11349	0.71
D198	20193	19883	1.54
Pr299	45623	45107	1.13
D493	122548	121875	0.55
		Rata-rata	1.75

Dari data diatas diketahui bahwa jumlah *routing* berbanding lurus dengan jumlah *nodes*. Semakin banyak jumlah *nodes* maka akan semakin banyak *routing* yang dilakukan. Jumlah *routing* yang dihasilkan dari uji coba menggunakan ACO ROIVA jika dibandingkan dengan jumlah *routing* ACO memiliki jumlah yang lebih sedikit dengan rata-rata penurunan sebesar 1.75%. Karena persentase penurunan memliki persentase yang cukup kecil, maka dilakukan pengujian signifikansi menggunakan uji T untuk mengetahui apakah memiliki perbedaan yang signifikan atau tidak. Hasil dari uji T dapat dilihat pada Tabel 5.9.

Tabel 5.9 Hasil Uji T

	ACO	ACO ROIVA
Mean	14475.22	14472.12
Hypotesis Mean	0.00	
Alpha	0.05	
P-value	0.01	

Berdasarkan data yang terdapat pada Tabel 5.9, dapat dilihat bahwa rata-rata jumlah *routing* dari ACO konvensional adalah sebesar 14475.22 dan rata-rata jumlah *routing* dari ACO ROIVA adalah sebesar 14472.12. Nilai *p-value* yang didapatkan dari hasil uji T memiliki nilai kurang dari nilai *alpha* yang telah ditentukan maka *H0* ditolak sehingga jumlah *routing* dikatakan memiliki perbedaan yang signifikan.

BAB 6
KESIMPULAN DAN SARAN

Bab ini berisi pembahasan mengenai kesimpulan yang dapat diambil dari hasil uji coba yang telah dilakukan sebagai jawaban dari rumusan masalah yang dikemukakan. Selain kesimpulan, bab ini berisi saran yang ditujukan untuk pengembangan perangkat lunak lebih lanjut.

6.1 Kesimpulan

Berdasarkan algoritma yang telah dibuat beserta uji coba yang telah dilakukan, maka dapat ditarik kesimpulan sebagai berikut:

1. Algoritma *Ant Colony Optimization* (ACO) dengan metode ROIVA memiliki solusi terbaik yang perbedaannya tidak berbeda jauh jika dibandingkan dengan ACO konvensional, tetapi ACO dengan metode ROIVA menurunkan rata-rata kompleksitas waktu sebesar 24.80% dan jumlah *routing* sebesar 1.75% jika dibandingkan dengan ACO konvensional.
2. Setiap *dataset* memiliki nilai parameter ρ (tingkat penguapan *pheromone*) yang berbeda-beda untuk mendapatkan hasil yang optimal.
3. Iterasi mempengaruhi hasil dari solusi terbaik. Berdasarkan hasil uji coba yang sudah dilakukan, solusi optimum didapatkan pada uji coba sebanyak 20 iterasi dan 30 iterasi.

6.2 Saran

Tugas Akhir ini masih bisa dikembangkan lebih jauh lagi sehingga diperlukan saran-saran yang membangun, yaitu mengoptimalkan kembali algoritma sehingga hasil dari solusi terbaik bisa lebih baik dari algoritma sebelumnya.