PENENTUAN RUTE TERPENDEK BERSEPEDA DI AREA KOTA MALANG MENGGUNAKAN ALGORITMA SEMUT

HALAMAN JUDUL

SKRIPSI

Oleh:
SINDY YUDI PRAKOSO
0710963017-96

PROGRAM STUDI ILMU KOMPUTER
JURUSAN MATEMATIKA
FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM
UNIVERSITAS BRAWIJAYA
2012


UNIVERSITAS BRAWIJAYA

PENENTUAN RUTE TERPENDEK BERSEPEDA DI AREA KOTA MALANG MENGGUNAKAN ALGORITMA SEMUT

HALAMAN JUDUL

Skripsi

Sebagai salah satu syarat untuk memperoleh gelar
Sarjana Komputer dalam bidang Ilmu Komputer

Oleh:
SINDY YUDI PRAKOSO
0710963017-96

PROGRAM STUDI ILMU KOMPUTER
JURUSAN MATEMATIKA
FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM
UNIVERSITAS BRAWIJAYA

2012


UNIVERSITAS BRAWIJAYA

LEMBAR PENGESAHAN SKRIPSI

PENENTUAN RUTE TERPENDEK BERSEPEDA DI AREA KOTA MALANG MENGGUNAKAN ALGORITMA SEMUT

oleh:
Sindy Yudi Prakoso
0710963017-96

Setelah dipertahankan di depan Majelis Penguji
Pada tanggal 20 Juni 2012
Dan dinyatakan memenuhi syarat untuk memperoleh gelar
Sarjana Komputer dalam bidang Ilmu Komputer

Pembimbing I,	Pembimbing II,
Dian Eka Ratnawati,S.Si.,M.Kom	**Yusi Tyroni M, S.Kom., MS.**
NIP. 19730619 2002122 001	**NIP. 19800228 2006041 001**

Mengetahui,
Ketua Jurusan Matematika
Fakultas MIPA Universitas Brawijaya

Dr. Abdul Rouf Alghofari, M.Sc
NIP. 19670907 1992031 001


UNIVERSITAS BRAWIJAYA

LEMBAR PERNYATAAN
Saya yang bertanda tangan di bawah ini :

Nama **:** Sindy Yudi Prakoso
NIM **:** 0710963017-96
Jurusan **:** Matematika
Program Studi **:** Ilmu Komputer
Penulis skripsi berjudul : Penentuan Rute Terpendek Bersepeda di Area Kota Malang menggunakan Algoritma Semut

Dengan ini menyatakan bahwa :

1. **Isi dari Skripsi yang saya buat adalah benar-benar karya sendiri dan tidak menjiplak karya orang lain, selain nama-nama yang termaktub di isi dan tertulis di daftar pustaka dalam Skripsi ini.**
2. **Apabila dikemudian hari ternyata Skripsi yang saya tulis terbukti hasil jiplakan, maka saya akan bersedia menanggung segala resiko yang akan saya terima.**

Demikian pernyataan ini dibuat dengan segala kesadaran.

Malang, 20 Juni 2012
Yang menyatakan,

Sindy Yudi Prakoso
NIM. 0710963017-96

PENENTUAN RUTE TERPENDEK BERSEPEDA DI AREA KOTA MALANG MENGGUNAKAN ALGORITMA SEMUT

ABSTRAK

Transportasi saat ini merupakan kebutuhan yang esensial dalam kehidupan manusia. Seiring dengan perkembangan suatu daerah, maka transportasi akan memiliki berbagai masalah serta dapat menimbulkan polusi berlebih dengan semakin padatnya ruas-ruas jalan yang digunakan. Bersepeda dianggap sebagai transportasi bebas polusi dan menyehatkan, namun diperlukan adanya suatu informasi perjalanan. Informasi perjalanan ini sangat erat kaitannya dengan rute yang dipilih sehingga dapat menghasilkan solusi yang optimal, dalam hal ini adalah meminimalkan jarak perjalanan dari tempat asal menuju tempat tujuannya.

Penelitian ini bertujuan membangun sebuah sistem penentuan jalur terpendek bersepeda di area kota Malang menggunakan algortima semut, dengan input berupa sebuah titik awal dan sebuah titik tujuan. Untuk mengetahui tingkat optimasi algoritma semut, maka dilakukan pengujian terhadap pengaruh nCmax, m, $\tau 0$, α, β, ρ, dan q0. Dari uji coba yang dilakukan, diketahui bahwa semakin banyak m maka akan mempengaruhi jumlah iterasi untuk mencapai konvergensi. Nilai paramater α dan β berbanding lurus dengan nilai Φ. Nilai ρ berbanding terbalik dengan nilai $\tau 0$, dan diperoleh nilai parameter yang optimal adalah nCmax=20, m=500, $\tau 0$=0.5, α=4, β=4, ρ=0.9, dan q0=0.3. Solusi yang dihasilkan oleh algoritma semut dibandingkan dengan solusi yang dihasilkan algoritma *Dijkstra*, diperoleh nilai MSE 0.490746 yang membuktikan bahwa algoritma semut memiliki tingkat akurasi yang cukup tinggi.

Kata Kunci : Algoritma Semut, Algoritma *Dijkstra*, Rute Terpendek

THE DETERMINATION OF SHORTEST ROUTES BIKE IN MALANG CITY USING ANT ALGORITHM

ABSTRACT

Transportation is now a requirement that is essential in human life. Along with the development of an area, transportation will have a variety of problems and can cause excessive pollution in the crowded streets that are used. Cycling is considered as a pollution-free transportation and healthy, but travel information is required. Travel information is closely related with the selected route to produce an optimal solution, in this case is to minimize the distance traveled from starting point to destination point.

This study was intent to build a system of determining cycling shortest path in Malang City area using ant algorithms, with starting point and destination point as an input. To knowing the optimization level of ant algorithm, the influence of nCmax, m, $\tau 0$, α, β, ρ, and q0 are tested. From the experiments, it is known that the more m it will affect the number of iterations to achieve convergence. Parameter α and β is directly proportional to the value of Φ. P is inversely proportional to the value of $\tau 0$, and the optimal parameter values are nCmax = 20, m = 500, $\tau 0 = 0.5$, $\alpha = 4$, $\beta = 4$, $\rho = 0.9$, and $q0 = 0.3$. The Comparison of resulting solutions between ant algorithm and Dijkstra algorithm produced MSE value 0.490746, which proves that the ant algorithm has high degree of accuracy

Keywords : Ant Algorithm, Dijkstra Algorithm, Shortest Path

KATA PENGANTAR

Alhamdulillahi rabbil 'alamin. Puji syukur penulis panjatkan kehadirat Allah SWT, karena atas segala rahmat dan limpahan hidayahNya, penulis masih dapat belajar dan mengerjakan skripsi yang berjudul "PENENTUAN RUTE TERPENDEK BERSEPEDA DI AREA KOTA MALANG MENGGUNAKAN ALGORITMA SEMUT ".

Skripsi ini disusun dan diajukan sebagai syarat untuk memperoleh gelar sarjana pada program studi Ilmu Komputer, jurusan Matematika, fakultas MIPA, universitas Brawijaya.

Dalam penyelesaian skripsi ini, penulis telah mendapat begitu banyak bantuan baik moral maupun materiil dari banyak pihak. Atas bantuan yang telah diberikan, penulis ingin menyampaikan penghargaan dan ucapan terima kasih kepada:

1. Dian Eka Ratnawati, S.Si., M.Kom sebagai pembimbing I dan Yusi Tyroni Mursityo,S.Kom.,MS selaku pembimbing II. Terima kasih atas semua waktu dan bimbingan yang telah diberikan.
2. Dr. Abdul Rouf Alghofari, M.Sc., selaku Ketua Jurusan Matematika Fakultas Matematika Dan Ilmu Pengetahuan Alam.
3. Drs. Marji, M.T., selaku Ketua Program Studi Ilmu Komputer Jurusan Matematika.
4. Segenap bapak dan ibu dosen yang telah mendidik dan mengamalkan ilmunya kepada penulis.
5. Segenap staf dan karyawan di Jurusan Matematika FMIPA Universitas Brawijaya
6. Papa, Mama, Mbak dan Eyang. Terima kasih atas cinta, kasih sayang, doa, dukungan dan semangat yang tiada henti.
7. Sahabat-sahabat ilkomers angkatan 2007 dan seluruh warga Program Studi Ilmu Komputer Universitas Brawijaya.
8. Pihak lain yang telah membantu terselesaikannnya skripsi ini yang tidak bisa penulis sebutkan satu-persatu.

Penulis sadari bahwa masih banyak kekurangan dalam laporan ini disebabkan oleh keterbatasan kemampuan dan pengalaman. Oleh karena itu Penulis sangat menghargai saran dan

kritik yang sifatnya membangun demi perbaikan penulisan dan mutu isi skripsi ini untuk kelanjutan penelitian serupa di masa mendatang.

Penulis berharap semoga skripsi ini dapat memberikan manfaat kepada pembaca dan bisa diambil manfaatnya, baik oleh Penulis selaku mahasiswa maupun pihak-pihak lainnya.

Malang, 20 Juni 2012

Penulis

DAFTAR ISI

HALAMAN JUDUL i
HALAMAN JUDUL i
LEMBAR PENGESAHAN SKRIPSI iii
LEMBAR PERNYATAAN v
ABSTRAK vii
ABSTRACT ix
KATA PENGANTAR xi
DAFTAR ISI xiii
DAFTAR GAMBAR xvii
DAFTAR TABEL xix
DAFTAR *SOURCECODE* xxiii

BAB I 1
PENDAHULUAN 1
1.1 Latar Belakang 1
1.2 Rumusan Masalah 2
1.3 Tujuan 2
1.4 Batasan Masalah 3
1.5 Manfaat 3
1.6 Metode Penyelesaian Masalah 3
1.7 Sistematika penulisan 3
BAB II 5
TINJAUAN PUSTAKA 5
2.1 Teori Graf 5
2.2.1 Definisi Graf 5
2.2.2 Jenis-jenis Graf 5
2.2 Algoritma *Dijkstra* 7
2.3 Optimasi 9
2.4 Konsep Dasar Algoritma Koloni Semut (AntCo) 9
2.4.1 Inisialisasi Parameter 11
2.4.2 Pengisian *Tabu List* 11
2.4.3 Pemilihan Rute Perjalanan Semut 11
2.4.4 Perhitungan panjang jalur setiap semut 12
2.4.5 Perhitungan perubahan harga intensitas *pheromone* semut antar kota 13
2.4.6 Pengosongan *tabu list* 14
2.5 *Ant Colony System* (ACS) 15

2.5.1 *State Transition Rule* (Aturan Transisi Status).....16
2.5.2 *Update Pheromone* Lokal 16
2.5.3 *Update Pheromone Global*.....17
2.6 Akurasi 18
BAB III.....21
METODOLOGI.....21
3.1 Analisis Perancangan Sistem..... 22
3.1.1 Deskripsi Umum Sistem 22
3.1.2 Analisis Data.....22
3.2 Perancangan Sistem..... 23
3.2.1 Perancangan Proses Algoritma Semut 24
3.2.2 Perancangan Proses *Simulate Ant* 24
3.2.3 Perancangan Proses Penentuan Rute Selanjutnya.....25
3.2.4 Perancangan Proses Update Pheromone lokal 28
3.2.5 Perancangan Proses Update Pheromone global 29
3.3 Perancangan Basis Data 30
3.4 Perhitungan Manual Menggunakan *Ant Colony System*..... 32
3.4.1 Inisialisasi Parameter 36
3.4.2 Pengisian Tabu List.....38
3.4.3 Penyusunan Rute Perjalanan Semut.....38
3.4.4 Perhitungan Panjang Jalur Setiap Semut 43
3.4.5 Perhitungan *Update Pheromone* Global 44
3.4.6 Pengosongan *Tabu List* 47
3.4.7 Perhitungan Pada Iterasi Kedua 47
3.5 Perancangan Antarmuka..... 48
3.5.1 *Form* Utama 49
3.5.2 Inisialisasi *Pheromone* 50
3.6 Perancangan Uji Coba dan Evaluasi..... 51
BAB IV.....57
IMPLEMENTASI DAN PEMBAHASAN 57
4.1 Lingkungan Implementasi 57
4.1.1 Lingkungan Perangkat Keras 57
4.1.2 Lingkungan Perangkat Lunak 57
4.2 Implementasi Basis Data 57
4.3 Implementasi Program 58
4.3.1 Struktur Data.....58
4.3.2 Penentuan Rute Perjalanan Semut 59
4.3.3 Pemilihan Rute Oleh Semut.....60

4.3.4 Penyimpanan Rute Perjalanan ke TabuList 63
4.3.5 Proses *Update Pheromone* 63
4.3.6 Perhitungan Panjang Jalur semut 65
4.3.7 Penerapan Rumus Pitagoras 66
4.4 Implementasi Antarmuka Program Pencarian Rute Terpendek Bersepeda Menggunakan Algoritma Semut 67
4.5 Implementasi Uji Coba Perangkat Lunak 70
4.5.1 Hasil dan Analisa Uji Coba Parameter Algoritma *Ant Colony System* 70
4.5.1.1. Pengujian Parameter Jumlah Semut (m) dan *Pheromone* Awal (τ0) 71
4.5.1.2. Pengujian Parameter Tetapan Pengendali Intensitas (α) dan Tetapan Pengendali Visibilitas (β) 73
4.5.1.3. Pengujian Parameter Tetapan Penguapan Jejak Semut (ρ) 75
4.5.1.4. Pengujian Parameter Q0 77
4.5.1.5. Pengujian Generasi 78
4.5.2 Uji coba Tingkat Optimasi Algoritma *Ant Colony System* 80
4.5.2.1. Pengujian Jarak Dekat 80
4.5.2.2. Pengujian Jarak Menengah 82
4.5.2.3. Pengujian Jarak Jauh 84
4.5.2.4. Pengujian Menggunakan Titik Singgah 86
BAB V 89
KESIMPULAN DAN SARAN 89
5.1 Kesimpulan 89
5.2 Saran 89
Daftar Pustaka 91
Lampiran 1 93
Lampiran 2 97

DAFTAR GAMBAR

Gambar 2. 1 Perjalanan semut menemukan sumber makanan 10
Gambar 3. 1 Diagram Sistem 21
Gambar 3. 3 *Flowchart* Membangun Rute Perjalanan 23
Gambar 3. 4 *Flowchart Simulate Ant* 25
Gambar 3. 5 *Flowchart* Penentuan Rute Perjalanan Selanjutnya 28
Gambar 3. 6 *Flowchart Update Pheromone lokal* 28
Gambar 3. 7 *Flowchart Update Pheromone Global* 29
Gambar 3. 8 Ilustrasi Graph dengan 14 *Node* 33
Gambar 3. 9 Tabu List Semut Pertama 38
Gambar 3. 10 Isi *Tabu List* Pertama Perjalanan Semut (A) Pada Iterasi Pertama 43
Gambar 3. 11 Rancangan Antarmuka *Input data* untuk Pencarian Rute Terpendek 49
Gambar 3. 12 Rancangan Antarmuka Menu Perhitungan *Pheromone* 51
Gambar 4. 1 *Form* Utama Program 68
Gambar 4. 2 *Form tab* Rute Rekomendasi Sistem 69
Gambar 4. 3 *Form tab* Grafik *Pheromone* Semut 70
Gambar 4. 4 Grafik Pengaruh Perubahan Parameter Semut dan *Pheromone* Awal 72
Gambar 4. 5 Grafik Uji Coba Tetapan Pengendali Intensitas (α) 74
Gambar 4. 6 Grafik Uji Coba Tetapan Pengendali Visibilitas (β) 75
Gambar 4. 7 Grafik Uji Tetapan Penguapan Jejak Semut (ρ) 76
Gambar 4. 8 Grafik Uji Q0 78
Gambar 4. 9 Grafik Uji Konvergensi 79
Gambar 4. 10 Pengujian Untuk Jarak Dekat 81
Gambar 4. 11 *Running Time* untuk Pengujian Jarak Dekat 81
Gambar 4. 12 Rute yang dihasilkan oleh ACS dan *Dijkstra* pada jarak dekat 81
Gambar 4. 13 Pengujian Untuk Jarak Menengah 82
Gambar 4. 14 *Running Time* untuk Pengujian Jarak Menengah 83
Gambar 4. 15 Rute yang dihasilkan oleh ACS pada jarak menengah 83
Gambar 4. 16 Rute yang dihasilkan oleh *Dijkstra* pada jarak menengah 83
Gambar 4. 17 Pengujian Untuk Jarak Jauh 84
Gambar 4. 18 *Running Time* untuk Pengujian Jarak Jauh 85

Gambar 4. 19 Rute yang dihasilkan oleh ACS pada jarak jauh.....85
Gambar 4. 20 Rute yang dihasilkan oleh *Dijkstra* pada jarak jauh ..85
Gambar 4. 21 Pengujian Menggunakan Titik Singgah.....86
Gambar 4. 22 *Running Time* untuk Pengujian Menggunakan Titik Singgah.....87
Gambar 4. 23 Rute yang dihasilkan oleh ACS dengan titik singgah 87
Gambar 4. 24 Rute yang dihasilkan oleh *Dijkstra* dengan titik singgah.....87

DAFTAR TABEL

Tabel 3. 1 Tabel Jalan.....30
Tabel 3. 2 Tabel Cabang.....31
Tabel 3. 3 Tabel KoordinatPenyusun.....31
Tabel 3. 4 Data Nama Jalan.....33
Tabel 3. 5 Jarak Antar Jalan.....35
Tabel 3. 6 Inisialisasi Parameter ACS.....36
Tabel 3. 7 Invers jarak *(η(i,j))*.....37
Tabel 3. 8 Probabilitas Komulatif Semut ke-1 Pada Generasi ke-1 .39
Tabel 3. 9 Probabilitas Komulatif Semut ke-1 Pada Generasi ke-1 .40
Tabel 3. 10 Probabilitas Komulatif Semut ke-1 Pada Generasi ke-141
Tabel 3. 11 Probabilitas Komulatif Semut ke-1 Pada Generasi ke-142
Tabel 3. 12 Pemilihan Rute Perjalanan Oleh semut ke-B Iterasi 1 ..43
Tabel 3. 13 Pemilihan Rute Perjalanan Oleh semut ke-C Iterasi 1 ..43
Tabel 3. 14 Hasil Panjang Jalur Semut Pada Iterasi Pertama.....44
Tabel 3. 15 Hasil *Update Pheromone Global* Iterasi Pertama.....46
Tabel 3. 16 Pemilihan Rute Oleh Semut ke-A Iterasi 2.....47
Tabel 3. 17 Pemilihan Rute Oleh Semut ke-B Iterasi 2.....47
Tabel 3. 18 Pemilihan Rute Oleh Semut ke-C Iterasi 2.....48
Tabel 3. 19 Hasil Panjang Jalur Semut Pada Iterasi Kedua.....48
Tabel 3. 20 *Default* nilai parameter yang digunakan dalam uji coba51
Tabel 3. 21 Rancangan Uji Coba Jalur dengan Perubahan Parameter Jumlah Semut.....52
Tabel 3. 22 Rancangan Uji Coba Jalur dengan Perubahan Parameter intensitas *pheromone* (τ_{ij}).....52
Tabel 3. 23 Rancangan Uji Coba Jalur dengan Perubahan Parameter α.....53
Tabel 3. 24 Rancangan Uji Coba Jalur dengan Perubahan Parameter β.....53
Tabel 3. 25 Rancangan Uji Coba Jalur dengan Perubahan Parameter *ρ*.....53
Tabel 3. 26 Rancangan Uji Coba Jalur dengan Perubahan Parameter Jumlah Iterasi.....54
Tabel 3. 27 Rancangan Uji Coba Jalur Tanpa Titik Singgah Jarak Dekat.....54
Tabel 3. 28 Rancangan Uji Coba Jalur Tanpa Titik Singgah Jarak Menengah.....54

Tabel 3. 29 Rancangan Uji Coba Jalur Tanpa Titik Singgah Jarak Jauh.....55
Tabel 3. 30 Rancangan Uji Coba Jalur Dengan Titik Singgah.....55
Tabel 4. 1 Definisi Variabel.....58
Tabel 4. 2 Hasil Uji *Pheromone* Awal dengan 100 Semut.....71
Tabel 4. 3 Hasil Uji *Pheromone* Awal dengan 200 Semut.....71
Tabel 4. 4 Hasil Uji *Pheromone* Awal dengan 300 Semut.....71
Tabel 4. 5 Hasil Uji *Pheromone* Awal dengan 400 Semut.....72
Tabel 4. 6 Hasil Uji *Pheromone* Awal dengan 500 Semut.....72
Tabel 4. 7 Hasil Uji Tetapan Pengendali Intensitas (α).....73
Tabel 4. 8 Hasil Uji Tetapan Pengendali Visibilitas (β).....74
Tabel 4. 9 Hasil Uji Tetapan Penguapan Jejak Semut.....76
Tabel 4. 10 Hasil Uji Q0.....77
Tabel 4. 11 Hasil Uji Konvergensi.....79
Tabel 4. 12 Hasil Pengujian Jarak Dekat.....80
Tabel 4. 13 Hasil Pengujian Jarak Menengah.....82
Tabel 4. 14 Hasil Pengujian Jarak Jauh.....84
Tabel 4. 15 Hasil Pengujian Menggunakan Titik Singgah.....86

DAFTAR PERSAMAAN

Persamaan (2. 1) 12
Persamaan (2. 2) 12
Persamaan (2. 3) 13
Persamaan (2. 4) 13
Persamaan (2. 5) 13
Persamaan (2. 6) 14
Persamaan (2. 7) 14
Persamaan (2. 8) 16
Persamaan (2. 9) 16
Persamaan (2. 10) 17
Persamaan (2. 11) 17
Persamaan (2. 12) 17
Persamaan (2. 13) 19

DAFTAR *SOURCECODE*

Source Code 4. 1 Struktur Data 58
Source Code 4. 2 Membangun Rute Perjalanan 60
Source Code 4. 3 Menentukan Rute Selanjuntnya 62
Source Code 4. 4 Penyimpanan Rute dalam TabuList 63
Source Code 4. 5 Proses *Update Pheromone* Lokal 64
Source Code 4. 6 Proses *Update Pheromone Global* 65
Source Code 4. 7 Proses Mendapatkan Jalur Terbaik 66
Source Code 4. 8 Fungsi Pitagoras 67

BAB I
PENDAHULUAN

1.1 Latar Belakang

Seiring dengan perkembangan era globalisasi yang semakin pesat, sebagian besar masyarakat merasakan keberadaan akses transportasi darat telah menjadi salah satu kebutuhan pokok dalam menjalani rutinitas sehari-hari. Kebutuhan terhadap transportasi yang lancar, aman nyaman dan sesuai lingkungan pada suatu kota adalah keinginan seluruh masyarakat kota. Transportasi telah berubah menjadi suatu kebutuhan utama yang dapat mempengaruhi produktivitas pekerjaan sehari-hari, kendaraan pribadi baik berupa kendaraan roda dua ataupun roda empat dinilai mempunyai keuntungan yang sangat besar bagi setiap individu terutama dalam hal mobilitas. Penggunaan kendaraan pribadi ini akan meningkatkan kesempatan seseorang untuk bekerja, rekreasi dan melakukan aktivitas sosial. Namun, kondisi ini akan mengakibatkan kepemilikan kendaraan pribadi semakin meningkat, sehingga berdampak pada pertumbuhan jumlah kendaraan yang tidak sebanding dengan pertumbuhan prasarana jalan yang ada.

Berdasarkan data (Dishub pada tahun 2006 sampai 2008) keberadaan transportasi darat khususnya kendaraan pribadi lebih mendominasi dalam bidang perhubungan dan transportasi, tercatat 98% adalah kendaraan pribadi sedangkan sisanya adalah kendaraan umum dan kendaraan dinas. Dari kendaraan yang melintas di jalan raya adalah kendaraan pribadi dan dari jumlah ini ternyata 47% bermuatan 1 (satu) orang saja, sehingga menyebabkan tingginya volume pergerakan, makin rendahnya kecepatan bahkan cenderung macet sehingga penggunaan kendaraan pribadi dinilai tidak efisien lagi.

Salah satu inovasi untuk mengurangi polusi dalam transportasi darat yaitu dengan adanya program *car free day* yang diharapkan bisa memberikan ruang bagi masyarakat Kota Malang untuk bisa berolahraga serta menikmati udara segar bebas asap (surya, 2011) . Dengan ditunjang dengan slogan *Bike to Work* ternyata banyak diminati oleh warga kota Malang. Mulai dari kalangan anak-anak hingga dewasa banyak menggunakan sepeda sebagai sarana transportasi untuk mencapai tempat tujuan yang diinginkan.

Kebutuhan akan suatu informasi atau petunjuk tentang suatu lokasi baik itu lokasi jalan ataupun lokasi obyek-obyek tertentu sangat diperlukan bagi pengguna sepeda untuk menentukan jalur terpendek dari suatu tempat ke tempat lainnya. *Shortest path problem* merupakan salah satu permasalahan optimasi yang dapat diselesaikan dengan menggunakan metode heuristik, seperti algoritma genetika (*Genetic Algorithm*, GA), algoritma semut (*Ant Colony*, AntCo) (Saptono, 2007). Algoritma semut adalah solusi universal dan fleksibel yang awalnya digunakan pada permasalahan optimasi *Traveling Salesman Problem*. Analogi pencarian rute terpendek oleh semut-semut, telah menjadi stimulus terciptanya suatu metode untuk menentukan jarak terpendek dari suatu tempat ke tempat lain, Jarak terpendek yang dimaksud adalah hasil dari perhitungan beberapa parameter melalui *Ant Colony Optimization* (Dorigo, 1996).

Dalam skripsi ini, pencarian rute terpendek mempunyai tujuan membantu dalam menentukan jalur terpendek dari dan menuju suatu tempat tertentu di Malang khususnya Malang Kota, dengan ditentukannya jalur terpendek maka diharapkan dapat meminimalisasi waktu tempuh perjalanan menggunakan metode algoritma semut. Berdasarkan latar belakang yang telah dipaparkan, maka judul yang diambil dalam penelitian ini adalah **"Penentuan Rute Terpendek Bersepeda Area Kota Malang Menggunakan Algoritma Semut"**.

1.2 Rumusan Masalah

Berdasarkan latar belakang, maka rumusan masalah yang akan dibahas dalam penelitian ini adalah :

1. Bagaimana mengimplementasikan Algoritma *Ant Colony System (ACS)* untuk menentukan rute terpendek perjalanan di kota Malang?
2. Bagaimana jarak terpendek yang dihasilkan Algoritma Semut dibandingkan dengan Algoritma *Dijkstra*?

1.3 Tujuan

Tujuan dari penelitian ini adalah :

1. Mengimplementasikan Algoritma *Ant Colony System (ACS)* dalam pencarian rute terpendek sebagai solusi dalam perjalanan bersepeda.

2. Mengetahui nilai keakuratan antara Algoritma Semut dan Algoritma *Dijkstra*.

1.4 Batasan Masalah

Batasan masalah pada skripsi ini adalah sebagai berikut :

1. Lokasi yang dibahas hanya seputar jalan *protocol* kota malang.
2. Jalan di semua rute diasumsikan dalam kondisi baik.
3. Tidak memperhatikan jalanan menanjak ataupun menurun.

1.5 Manfaat

Manfaat yang akan dicapai dari penulisan skripsi ini adalah untuk memberikan solusi terutama bagi pengguna sepeda sebagai alat transportasi dalam menetukan jalur terpendek saat melakukan perjalanan sehingga dapat menghemat waktu, tenaga dan biaya.

1.6 Metode Penyelesaian Masalah

Metode penyelesaian masalah yang dilakukan pada penelitian ini, yaitu :

1. Studi Literatur

   Membaca dan mempelajari beberapa literatur (jurnal, buku dan artikel dari *website*) mengenai *ACS*, dan jenis-jenis graf dalam (*shortest path)*.

2. Perancangan dan implementasi perangkat lunak

   Merancang dan membangun sebuah perangkat lunak berbasis C# yang mengimplementasikan proses penentuan rute terpendek menggunakan algoritma semut.

3. Uji coba dan analisis hasil implementasi

   Menganalisis hasil implementasi, yaitu hasil-hasil perhitungan dari beberapa jarak yang tersedia.

1.7 Sistematika penulisan

Skripsi ini akan menjabarkan keseluruhan penelitian yang dikelompokkan secara sistematis menjadi lima bab. Adapun pembagian bab-bab tersebut adalah:

BAB I PENDAHULUAN

Dalam bab ini membahas mengenai latar belakang, rumusan masalah, batasan masalah, tujuan, metode penyelesaian masalah, manfaat, serta sistematika penulisan.

BAB II TINJAUAN PUSTAKA

Menjelaskan tentang dasar teori yang digunakan dalam menyusun skripsi. Hal tersebut, meliputi penjelasan Graf, Algoritma Semut.

BAB III METODOLIGI

Membahas Perancangan sistem, arsitektur sistem, diagram alur proses sistem, rancangan penelitian, dan contoh perhitungan manual.

BAB IV IMPLEMENTASI DAN PEMBAHASAN

Bab ini membahas tentang lingkungan implementasi, implementasi sistem yang terbagi atas implementasi program dan implementasi antarmuka, serta analisa hasil uji coba.

BAB V KESIMPULAN DAN SARAN

Bab ini berisi tentang kesimpulan dari pembahasan bab sebelumnya serta saran dari keseluruhan penelitian.

BAB II
TINJAUAN PUSTAKA

2.1 Teori Graf

Sejak pertama kali diperkenalkan oleh seorang matematikawan Swiss bernama Leonhard Euler pada tahun 1736 (Munir, 2005), teori graf mulai berkembang pesat bahkan hingga saat ini. Teori graf merupakan salah satu cabang dari ilmu matematika diskrit yang meskipun usianya sudah cukup tua namun masih banyak mendapat perhatian terutama oleh para ilmuan. Hal ini disebabkan karena model-model yang ada pada teori graf berguna untuk aplikasi yang luas di berbagai bidang, seperti kelistrikan, jaringan komunikasi, ilmu komputer, kimia, biologi, dan lain-lain.

Graf digunakan untuk merepresentasikan objek-objek diskrit dan hubungan antara objek-objek tersebut (Munir, 2005). Representasi visual dari graf adalah dengan menyatakan objek sebagai noktah, bulatan atau titik, sedangkan hubungan antara objek dinyatakan dengan garis

2.2.1 Definisi Graf

Graf juga disebutkan sebagai pasangan himpunan (V,E), ditulis dengan notasi G=(V,E), yang dalam hal ini V adalah himpunan tidak kosong dari simpul-simpul (*vertices* atau *nodes*) yang menghubungkan sepasang simpul (Munir, 2005). Jadi, sebuah graf dimungkinkan untuk tidak mempunyai sisi sama sekali tetapi simpulnya harus ada walaupun tanpa penghubung.

Sumber lain mengatakan bahwa graf G mengandung tiga poin, yaitu V(G), E(G), dan μg dimana V(G) adalah set dari *vertex* atau simpul, E(G) adalah set dari *edge* atau penghubung yang memisahkan *vertex* V(G) dan fungsi μg yang menentukan masing-masing *edge* dari graf G dimana *edge-edge* tersebut menghubungkan *vertex* yang tidak beraturan (terdapat kemungkinan untuk sebuah *edge* menghubungkan *vertex* yang sama) (Bondy & Murty, 1976).

2.2.2 Jenis-jenis Graf

Graf dapat dikelompokkan menjadi beberapa jenis bergantung pada sudut pandang pengelompokannya. Pengelompokan graf dapat dipandang dari ada tidaknya sisi ganda atau sisi kalang, berdasarkan

jumlah simpul atau berdasarkan orientasi arah pada sisi (Munir, 2005).

Berdasarkan ada tidaknya gelang atau sisi ganda pada suatu graf, maka secara umum graf dapat digolongkan menjadi dua jenis : (Munir, 2005)

1. **Graf sederhana** (*simple graph*)
   Yaitu graf yang tidak mengandung gelang (*loop*) maupun sisi-ganda dinamakan graf sederhana. Pada graf sederhana, sisi adalah pasangan tak-terurut (*unordered pairs*).
2. **Graf tak-sederhana** (*unsimple graph*)
   Graf yang mengandung sisi ganda atau gelang dinamakan graf tak-sederhana (*unsimple graph*). Ada dua macam graf tak-sederhana, yaitu **graf ganda** (*multigraph*) yang mengandung sisi ganda dan **graf semu** (*pseudograph*) yang mengandung gelang (*loop*).

Berdasarkan jumlah simpul pada suatu graf, maka secara umum graf dapat digolongkan menjadi dua jenis: (Munir, 2005)

1. **Graf berhingga** (*limited graph*)
   Graf berhingga adalah graf yang jumlah simpulnya, *n*, berhingga.
2. **Graf tak-berhingga** (*unlimited graph*)
   Graf yang jumlah simpulnya, *n*, tidak berhingga banyaknya.

Sisi pada graf dapat mempunyai orientasi arah. Berdasarkan orientasi arah pada sisi, maka graf dapat dibedakan menjadi dua jenis : (Munir, 2005)

1. **Graf tak-berarah** (*undirected graph*)
   Yaitu graf yang sisinya tidak mempunyai orientasi arah.
2. **Graf berarah** (*directed graph*)
   Yaitu graf yang setiap sisinya diberikan orientasi arah.

Sebuah struktur graf bisa dikembangkan dengan memberi bobot pada tiap *edge*. Graf ini disebut dengan graf berbobot, yaitu graf yang setiap sisinya diberi sebuah harga (bobot) (Munir, 2005).

Bobot pada tiap sisi dapat berbeda-beda bergantung pada masalah yang dimodelkan dengan graf. Bobot dapat menyatakan

jarak antara dua buah kota, biaya perjalanan antara dua kota, waktu tempuh pesan (*message*) dari sebuah simpul ke simpul yang lain (dalam jaringan komputer), ongkos produksi, dan sebagainya (Munir, 2005).

2.2 Algoritma *Dijkstra*

Algoritma *Dijkstra* merupakan salah satu varian dari algoritma greedy, yaitu salah satu algoritma untuk pemecahan persoalan yang terkait dengan masalah optimasi, beberapa kasus pencarian lintasan terpendek yang diselesaikan menggunakan algoritma *dijkstra*, yaitu : Pencarian lintasan terpendek antara dua buah simpul tertentu (*a pair shortest path*), pencarian lintasan terpendek antara semua pasangan simpul (*all pairs shortest path*), pencarian lintasan terpendek dari simpul tertentu ke semua simpul yang lain (*single-source shortest path*), serta pencarian lintasan terpendek antara dua buah simpul yang melalui beberapa simpul tertentu (*intermediate shortest path*).

Algoritma greedy ini hanya memikirkan solusi terbaik yang akan diambil pada setiap langkah tanpa memikirkan konsekuensi ke depan. Algoritma *greedy* ini berupaya membuat pilihan nilai optimum lokal pada setiap langkah dan berharap agar nilai optimum lokal ini mengarah kepada nilai optimum global (Novandi, 2007).

Elemen-elemen penyusun algoritma *greedy* adalah:

1. Himpunan kandidat (C)
   Himpunan ini berisi elemen-elemen yang memiliki peluang untuk membentuk solusi. Pada persoalan lintasan terpendek dalam graf, himpunan kandidat ini adalah himpunan simpul pada graf tersebut.
2. Himpunan solusi (S)
   Himpunan ini berisi solusi dari permasalahan yang diselesaikan dan elemennya terdiri dari elemen dalam himpunan kandidat namun tidak semuanya atau dengan kata lain himpunan solusi ini adalah bagian dari himpunan kandidat.
3. Fungsi seleksi
   Fungsi seleksi adalah fungsi yang akan memilih setiap kandidat yang yang memungkinkan untuk menghasilkan solusi optimal pada setiap langkahnya.
4. Fungsi kelayakan

Fungsi kelayakan akan memeriksa apakah suatu kandidat yang telah terpilih (terseleksi) melanggar *constraint* atau tidak. Apabila kandidat melanggar constraint maka kandidat tidak akan dimasukkan ke dalam himpunan solusi.

5. Fungsi objektif
   Fungsi objektif akan memaksimalkan atau meminimalkan nilai solusi. Tujuannya adalah memilih satu saja solusi terbaik dari masing-masing anggota himpunan solusi

Penggunaan strategi *greedy* pada algoritma Dijkstra adalah Pada setiap langkah, ambil sisi berbobot minimum yang menghubungkan sebuah simpul yang sudah terpilih dengan sebuah simpul lain yang belum terpilih. Lintasan dari simpul asal ke simpul yang baru haruslah merupakan lintasan yang terpendek di antara semua lintasannya ke simpul-simpul yang belum terpilih.

Pada algoritma berikut Mencari lintasan terpendek dari simpul awal a ke semua simpul lainnya.. Berikut adalah *pseudocode* dari algoritma Dijkstra

```
1   procedure Dijkstra(INPUT m: matriks, a : simpul awal)
2
3   { Mencari lintasan terpendek dari simpul awal a
4   ke semua simpul lainnya.
5   Masukan: matriks ketetanggaan (m) dari graf
6   berbobot G dan simpul awal a
7   Keluaran: lintasan terpendek dari a ke semua
8   simpul lainnya
9   }
10
11  Kamus:
12  s : array [1..n] of integer
13  d : array [1..n] of integer
14  i : integer
15  Algoritma:
16    { Langkah 0 (inisialisasi: }
17    traversal [1..n]
18      s_i ← 0
19      d_i ← m_ai { Langkah 1: }
20    s_a ← 1
21    d_a ← ∞
22
23    { Langkah 2, 3, ..., n-1: }
24    traversal [2..n-1]
25      cari j sedemikian sehingga s_j= 0 dan
```

```
    dj = min {d1, d2, ..., dn}
    sj ← 1 {simpul j sudah terpilih}
    perbarui di, untuk i = 1, 2, 3, s.d. n
dengan:
    di(baru) = min{di(lama),dj + mji}
```

Kompleksitas algoritma Dijkstra adalah $O(n^2)$, dengan n adalah jumlah simpul pada graf. Kompleksitas ini bisa diperbaiki dengan penggunaan struktur data senarai ketetanggaan (*adjacency list*) atau antrian prioritas (*priority queue*) untuk memperoleh kompleksitas O(*(m+n) log n*).

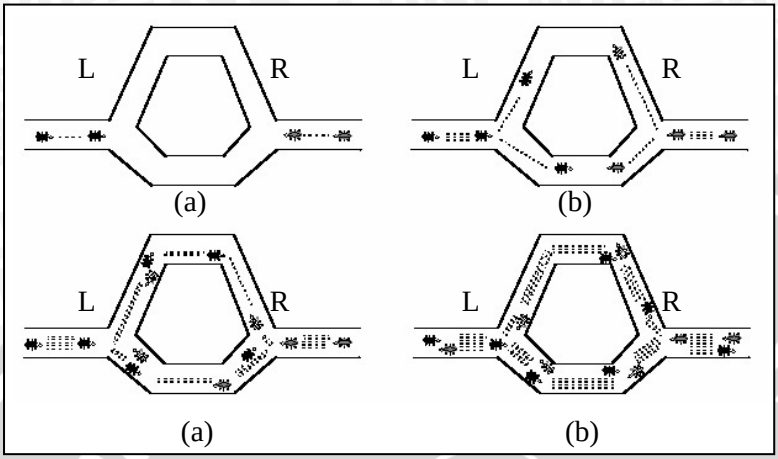
2.3 Optimasi

Optimasi merupakan masalah memaksimalkan atau meminimalkan suatu besaran tertentu, yang disebut dengan fungsi objektif. Fungsi objektif ini bergantung pada sejumlah variabel. Variabel-variabel ini tidak saling berhubungan atau saling bergantung melalui satu atau lebih kendala (Bronson, 1983).

Ada pula yang beranggapan bahwa suatu nilai dikatakan optimal adalah nilai yang didapat melalui suatu proses dan dianggap menjadi solusi jawaban yang paling baik dari semua solusi yang ada (Wardy, I.S, 2007).

2.4 Konsep Dasar Algoritma Koloni Semut (AntCo)

Koloni semut merupakan algorima yang bersifat heuristik untuk menyelesaikan masalah optimasi. Algoritma ini diinspirasikan oleh lingkungan koloni semut pada saat mencari makanan (Dorigo, 1996). Semut dapat mencari makanan, semut dapat mencari lintasan terpendek dari suatu sumber makanan menuju sarangnya, tanpa harus melihatnya secara langsung. Karena terinspirasi dari semut asli dinamakan algoritma koloni semut. Semut-semut mempunyai penyelesaian yang unik dan sangat maju yaitu dengan menggunakan jejak *pheromone*, proses peninggalan *pheromone* ini dikenal sebagai *stigmergy*, yaitu sebuah proses memodifikasi lingkungan yang tidak hanya bertujuan untuk mengingat jalan pulang ke sarang, tetapi juga memungkinkan para semut berkomunikasi dengan koloninya pada suatu jalur dan membangun solusi, semakin banyak jejak pheromone ditinggalkan, maka jalur tersebut akan diikuti oleh semut lain. Gambar 2.1 menujukkan perjalanan semut dalam menemukan jalur terpendek dari sarang ke sumber makanan.



Gambar 2. 1 Perjalanan semut menemukan sumber makanan (Sumber : Mutakhiroh, 2007)

Dari gambar 2.1(a) perjalanan semut dalam menemukan jalur terpendek dari sarang ke sumber makanan, terdapat dua kelompok semut yang melakukan perjalanan. Kelompok semut L berangkat dari arah kiri ke kanan dan kelompok semut R berangkat dari kanan ke kiri. Kedua kelompok berangkat dari titik yang sama dan dalam posisi pengambilan keputusan jalan sebelah mana yang akan diambil. Kelompok L membagi dua kelompok lagi. Sebagian melalui jalan atas dan sebagian melalui jalan bawah, hal ini berlaku juga untuk kelompok semut R. Pada gambar 2.1(b) dan gambar 2.1(c) menunjukkan bahwa kelompok semut berjalan pada kecepatan yang sama dengan meninggalkan *pheromone* atau jejak kaki di jalan yang telah dilalui. *Pheromone* yang ditinggalkan oleh kumpulan semut yang melalui jalan atas telah mengalami banyak penguapan karena semut yang melalui jalan atas berjumlah lebih sedikit dari pada jalan yang di bawah. Sedangkan *pheromone* yang berada di jalan bawah penguapannya cenderung lebih lama. Karena semut yang melalui jalan bawah lebih banyak daripada semut yang melalui jalan atas. Gambar 2.1(d) menunjukkan bahwa semut-semut yang lain memutuskan untuk melewati jalan bawah karena *pheromone* yang ditinggalkan masih banyak. Semakin banyak semut yang melalui jalan yang sama maka semakin banyak semut yang mengikutinya. Dari sinilah kemudian terpilihlah jalur terpendek antara sarang dan sumber makanan. (Mutakhiroh, 2007)

Dalam algoritma semut, diperlukan beberapa variabel dan langkah-langkah untuk menentukan jalur terpendek, yaitu:

2.4.1 Inisialisasi Parameter

Dalam ACO terdapat beberapa parameter masukan sebagai inisialisasi awal untuk melakukan proses optimasi. Beberapa parameter tersebut adalah

a. Inisialisasi harga parameter-parameter algoritma.

Parameter-parameter yang di inisialisasikan adalah :

1. τ_{ij} : Intensitas jejak semut antar kota dan perubahannya
2. n : Banyak titik atau d_{ij} (jarak antar kota)
3. Penentuan kota berangkat dan kota tujuan
4. Q : Tetapan siklus semut
5. α : Tetapan pengendali intensitas jejak semut
6. β : Tetapan pengendali visibilitas
7. η_{ij} : Visibilitas antar titik ($1/d_{ij}$)
8. m : Jumlah semut
9. ρ : Tetapan penguapan jejak semut
10. Jumlah siklus maksimum (NC_{max}) bersifat tetap selama algoritma dijalankan, sedangkan τ_{ij} akan selalu diperbaharui harganya pada setiap siklus algoritma mulai dari siklus pertama (NC=1) sampai tercapai jumlah siklus maksimum (NC=NCmax) atau sampai terjadi konvergensi.

b. Inisialisasi kota pertama setiap semut.

Setelah inisialisasi τ_{ij} dilakukan, kemudian m semut ditempatkan pada kota pertama yang telah ditentukan.

2.4.2 Pengisian *Tabu List*

Pengisian kota pertama ke dalam *tabu list*. Hasil inisialisasi parameter semut diisikan sebagai elemen pertama *tabu list*. Hasil dari langkah ini adalah terisinya elemen *tabu list* oleh setiap semut dengan indeks titik antara 1 sampai n titik sesuai dengan inisialisasi parameter awal.

2.4.3 Pemilihan Rute Perjalanan Semut

Koloni semut yang sudah terdistribusi ke kota pertama akan mulai melakukan perjalanan dari kota pertama sebagai kota asal dan salah satu kota-kota lainnya sebagai kota tujuan. Kemudian dari kota kedua, masing-masing koloni semut akan melanjutkan perjalanan

dengan memilih salah satu dari kota-kota yang tidak terdapat pada *tabu*$_k$ sebagai kota tujuan selanjutnya.Perjalanan koloni semut berlangsung terus menerus hingga mencapai kota yang telah ditentukan. Jika s menyatakan indeks urutan kunjungan, kota asal dinyatakan sebagai *tabu*$_k$(s) dan kota-kota lainnya dinyatakan sebagai {*N-tabu*$_k$}, maka untuk menentukan kota tujuan digunakan persamaan 2.1 sebagai probabilitas kota untuk dikunjungi sebagai berikut,

$$P_{ij}^{k} = \frac{[\tau_{ij}]^{\alpha} . [\eta_{ij}]^{\beta}}{\sum_{k' \in \{N - tabu_k\}} [\tau_{ik'}]^{\alpha} . [\eta_{ik'}]^{\beta}} \text{ untuk } j \in \{N\text{-}tabu_k\} \quad (2.1)$$

$$P_{ij}^{k} = 0, \text{ untuk j lainnya}$$

dimana :

P_{ij}	: probabilitas verteks i ke verteks j
i	: verteks ke-i
j	: verteks ke-j
τ_{ij}	: *pheromone* dari verteks i ke verteks j
β	: tetapan pengendali visibilitas
α	: tetapan pengendali intensitas jejak semut
η_{ij}	: visibilitas dari verteks i ke verteks j
k	: jumlah jalur kemungkinan yang dilalui

dengan i sebagai indeks kota asal dan j sebagai indeks kota tujuan. Setelah diperoleh nilai probabilitas antar kota, maka akan dihitung probabilitas komulatif dan range. Probabilitas komulatif didapat dengan persamaan 2.2, dengan membangkitkan bilangan *random r* dan kondisi $Pk(n-1) \leq rand \leq Pk(n)$, maka jalur yang dipilih adalah jalur Pk(n).

$$Pk(n) = Pij(n) + Pk(n-1) \quad (2.2)$$

dimana :

Pk	: nilai probabilitas komulatif
Pij	: nilai probabilitas antar jarak
n	: indeks rute yg dilalui semut

2.4.4 Perhitungan panjang jalur setiap semut

Setelah koloni semut menyelesaikan perjalanannya, maka akan didapat rute dari setiap semut pada siklus tersebut. Dari rute hasil perjalanan semut dapat dihitung panjang jalur perjalanan semut.

Perhitungan panjang jalur tertutup (*length closed tour*) atau L_k setiap semut dilakukan setelah satu siklus diselesaikan oleh semua semut. Perhitungan dilakukan berdasarkan *tabu*$_k$ masing-masing dengan persamaan 2.3 berikut:

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

dimana :

L_k : panjang lintasan yang dilalui semut k

n : banyak titik

$d_{tabu_k(s),tabu_k(s+1)}$: bobot antara titik ke s dan s+1 pada *tabu list*

dengan d_{ij} adalah jarak antara kota i ke kota j yang dihitung berdasarkan persamaan 2.4 berikut:

$$d_{ij} = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2} \quad (2.4)$$

Setelah L_k setiap semut dihitung, maka dapat ditentukan nilai minimal dari perjalanan setiap siklus semut atau L_{minNC} dan harga minimal dari perjalanan semut secara keseluruhan adalah atau L_{min}.

2.4.5 Perhitungan perubahan harga intensitas *pheromone* semut antar kota

Koloni semut akan meninggalkan jejak-jejak kaki pada lintasan antar kota yang dilaluinya. Adanya penguapan dan perbedaan jumlah semut yang lewat, menyebabkan kemungkinan terjadinya perubahan harga intensitas jejak kaki semut antar kota. Persamaan 2.5 adalah persamaan yang digunakan untuk menghitung perubahan intensitas *pheromone* semut

$$\Delta\tau_{ij} = \sum_{k=1}^{m} \Delta\tau_{ij}^{k} \quad (2.5)$$

dimana :

$\Delta\tau_{ij}$: Intensitas jejak semut antar kota dan perubahannya

$\Delta\tau_{ij}^{k}$: perubahan harga intensitas jejak kaki semut antar kota setiap semut
m : jumlah semut
k : jumlah jalur yang mungkin dilewati

dengan $\Delta\tau_{ij}^{k}$ dihitung berdasarkan persamaan 2.6 dibawah ini

$$\Delta\tau_{ij}^{k} = \frac{Q}{L_k} \quad (2.6)$$

dimana :
$\Delta\tau_{ij}^{k}$: perubahan harga intensitas jejak kaki semut antar kota setiap semut
Q : tetapan siklus semut
L_k : panjang lintasan yang dilalui semut k

Harga intensitas jejak kaki semut antar kota pada semua lintasan antar kota ada kemungkinan berubah karena adanya penguapan dan perbedaan jumlah semut yang melewati. Untuk siklus selanjutnya, semut yang akan melewati lintasan tersebut harga intensitasnya telah berubah. Harga intensitas jejak kaki semut antar kota untuk siklus selanjutnya dihitung dengan persamaan 2.7 :

$$\tau_{ij} = \rho.\tau_{ij} + \Delta\tau_{ij} \quad (2.7)$$

dimana :
τ_{ij} : intensitas jejak feromon pada garis (i,j)
ρ : tetapan penguapan jejak semut
$\Delta\tau_{ij}$: Intensitas jejak semut antar kota dan perubahannya

Untuk siklus selanjutnya perubahan harga intensitas jejak semut antar kota perlu diatur kembali agar memiliki nilai sama dengan nol.

2.4.6 Pengosongan *tabu list*

Tabu list perlu dikosongkan untuk diisi lagi dengan urutan kota yang baru pada siklus selanjutnya, jika jumlah siklus maksimum belum tercapai atau belum terjadi konvergensi. Algoritma diulang lagi dari pengisian tabu list dengan harga parameter intensitas jejak kaki semut antar kota yang sudah diperbaharui.

2.5 *Ant Colony System* (ACS)

Algoritma *Ant Colony System* (ACS) merupakan pengembangan dari Algoritma ACO, setelah beberapa algoritma seperti *Ant System(AS), Elitist Ant System(EAS), Rank-Based Ant System(AS_{rank}), MAX-MIN Ant System(MMAS)*. Algoritma ini tersusun atas sejumlah m semut yang bekerjasama dan berkomunikasi secara tidak langsung melalui komunikasi *Pheromone*. (Dorigo & Stützle, 2004)

Secara informal, ACS bekerja sebagai berikut: pertama kali, sejumlah *m* semut ditempatkan pada sejumlah *n* titik berdasarkan beberapa aturan inisialisasi (misalnya, secara acak). Setiap semut membuat sebuah tour (yaitu, sebuah solusi TSP yang mungkin) dengan menerapkan sebuah aturan transisi status secara berulang kali. Selagi membangun tournya, setiap semut juga memodifikasi jumlah *Pheromone* pada edge-edge yang dikunjunginya dengan menerapkan aturan pembaruan *Pheromone* lokal yang telah disebutkan tadi. Setelah semua semut mengakhiri tour atau perjalanan, jumlah *Pheromone* yang ada pada edge-edge dimodifikasi kembali (dengan menerapkan aturan pembaruan *Pheromone* global). Seperti yang terjadi pada *Ant system*, Setiap semut memiliki sebuah memori, dinamai *tabulist*, yang berisi semua titik yang telah dikunjunginya pada setiap tour. *Tabulist* ini mencegah semut untuk mengunjungi titik-titik yang sebelumnya telah dikunjungi selama tour tersebut berlangsung, yang membuat solusinya mendekati optimal. dalam membuat tour, semut 'dipandu' oleh informasi *heuristic* (mereka lebih memilih edge-edge yang pendek) dan oleh informasi *Pheromone*. Sebuah edge dengan jumlah *Pheromone* yang tinggi (Dorigo & Gambardella, 1997) merupakan pilihan yang sangat diinginkan. Kedua aturan pembaruan *Pheromone* itu dirancang agar semut cenderung untuk memberi lebih banyak *Pheromone* pada edge-edge yang harus mereka lewati. Berikutnya akan dibahas mengenai tiga karakteristik utama dari ACS, yaitu aturan transisi status, aturan pembaharuan *Pheromone* global, dan aturan pembaharuan *Pheromone* lokal.

Peranan utama dari penguapan *Pheromone* adalah untuk mencegah stagnasi, yaitu situasi dimana semua semut berakhir dengan melakukan tour yang sama. Proses di atas kemudian diulangi sampai tour-tour yang dilakukan mencapai jumlah maksimum atau

sistem ini menghasilkan perilaku stagnasi dimana sistem ini berhenti untuk mencari solusi alternatif.

2.5.1 *State Transition Rule* (Aturan Transisi Status)

Pemilihan kota-kota yang akan dilalui oleh semut didasarkan pada suatu fungsi probabilitas yang dinamakan aturan transisi status. Aturan transisi status yang berlaku pada ACS (Dorigo & Gambardella, 1997) yang ditunjukkan oleh persamaan 2.8. semut k yang berada dititik r, akan memilih titik berikutnya s, menurut persamaan:

$$j = \begin{cases} \arg\max\{[\tau_{i,u}].[\eta_{i,u}]^{\beta}\}, & jika\ q \le q0\ (eksploitasi) \\ J, & jika\ tidak\ \ (eksplorasi) \end{cases} \quad (2.8)$$

dimana :

- j : titik yang akan dituju. Jumlah *Pheromone* yang terdapat pada edge antara titik r dan titik s.
- q : sebuah bilangan random.
- β : parameter pengendali jarak ($\beta > 0$)

Aturan transisi status yang berlaku pada ACS adalah seekor semut yang ditempatkan pada titik *i* memilih untuk menuju ke titik *j*, kemudian diberikan bilangan pecahan acak q dimana $0 \le q \le 1$, *q0* adalah sebuah parameter yaitu Probabilitas semut melakukan eksplorasi pada setiap tahapan, dimana ($0 \le q0 \le 1$) dan *pk (t,v)* adalah probabilitas dimana semut k memilih untuk bergerak dari titik i ke titik j. Jika $q \le q0$ maka pemilihan titik yang akan dituju menerapkan aturan yang ditunjukkan oleh persamaan 2.8 sedangkan jika $q > q0$ digunakan persamaan 2.1.

2.5.2 *Update Pheromone* Lokal

Ketika membangun solusi (tour) dari TSP, semut mengaplikasikan *lokal updating rule* dengan mengubah tingkat *pheromone* pada edge-edge yang telah dikunjungi semut (Dorigo & Gambardella, 1997) berdasarkan pada persamaan 2.9

$$\tau_{ij} \leftarrow (1-\rho).\tau_{ij} + \rho.\tau_0 \quad (2.9)$$

dimana :

ρ : parameter *evaporasi lokal*.
τ_0 : nilai awal jejak *pheromone*.
dengan

$$\tau_0 = \frac{1}{n.C^{nn}} \quad (2.10)$$

dimana :
n : jumlah titik.
C^{nn} : panjang sebuah tour terbaik yang diperoleh dari metode *nearest neighbourhood heuristic*.

Persamaan *update Pheromone* lokal ini digunakan saat semut membangun tour, peranan dari aturan *update Pheromone lokal* ini adalah untuk mengacak arah *tour* yang sedang dibangun, sehingga *edge* yang telah dilewati oleh seekor semut sebelumnya mungkin akan dilewati kemudian oleh *tour* semut yang lain. Dengan kata lain, pengaruh dari *update Pheromone* lokal ini adalah untuk membuat tingkat ketertarikan ruas-ruas yang ada berubah secara dinamis dimana setiap seekor semut menggunakan sebuah ruas maka ruas ini dengan segera akan berkurang tingkat ketertarikannya (karena ruas tersebut kehilangan sejumlah *pheromone*-nya), secara tidak langsung semut yang lain akan memilih ruas-ruas lain yang belum dikunjungi. Konsekuensinya, semut tidak akan memiliki kecenderungan untuk berkumpul pada jalur yang sama. Fakta ini, telah diamati dengan melakukan percobaan (Dorigo dan Gambardella, 1997), merupakan sifat yang diharapkan bahwa jika semut membuat *tour* yang berbeda maka akan terdapat kemungkinan yang lebih tinggi dimana salah satu dari semut akan menemukan solusi yang lebih baik daripada jika semua semut berkumpul dalam *tour* yang sama.

2.5.3 *Update Pheromone Global*

Setelah semua semut menyelesaikan tour-nya masing – masing maka *Pheromone* di-*update* dengan mengaplikasikan *global updating rule* (Dorigo & Gambardella, 1997). Menurut persamaan 2.11sebagai berikut:

$$\tau_{ij} \leftarrow (1-\rho).\tau_{ij} + \rho.\Delta\tau_{ij}^{k} \quad (2.11)$$

dengan:

(2. 12)

$$\Delta\tau_{ij}^{k} = \begin{cases} \frac{1}{C^{bs}} & jika(r,s)\epsilon\ tour\ terbaik\ \ keseluruhan \\ 0 & sebaliknya \end{cases}$$

dimana :

ρ : parameter *evaporasi global*.
C^{bs} : panjang lintasan terbaik keseluruhan.
τ_{ij} : intensitas jejak *pheromone* pada garis (i,j)

Persamaan *update* jejak *Pheromone* secara *offline* ini, dilakukan pada akhir sebuah iterasi algoritma, saat semua semut telah menyelesaikan sebuah tour. Persamaan diaplikasikan ke edge yang digunakan semut menemukan lintasan keseluruhan yang terbaik sejak awal percobaan. Tujuan pemberian nilai ini adalah memberi sejumlah jejak *Pheromone* pada lintasan terpendek, dimana tour terbaik (lintasan dengan panjang terpendek) mendapat penguatan. Bersama dengan *pseudo-random-proportional rule* dimaksudkan untuk membuat pencarian lebih terarah.

Untuk memastikan bahwa semut mengunjungi n titik yang berbeda, diberikan *tabu list* pada masing – masing semut, yaitu sebuah struktur data yang menyimpan titik – titik yang telah dikunjungi semut dan melarang semut mengunjungi kembali titik – titik tersebut sebelum mereka menyelesaikan sebuah tour. Ketika sebuah tour selesai, *tabulist* digunakan untuk menghitung solusi yang ditemukan semut pada tour tersebut. *Tabulist* kemudian dikosongkan dan semut kembali bebas memilih titik tujuannya pada tour berikutnya. $Tabu_k$ adalah *tabu list* untuk semut k. $Tabu_k$ (r) adalah elemen ke-r dari $Tabu_k$, yaitu titik ke-r yang dikunjungi semut k pada suatu tour.

2.6 Akurasi

Akurasi merupakan seberapa dekat suatu angka hasil pengukuran terhadap angka sebenarnya (*true value* atau *reference value*). Dalam penelitian ini akurasi jarak dihitung menggunakan metode *Mean Square Error* (MSE). Semakin kecil nilai MSE mengindikasikan bahwa hasil prediksi semakin akurat (MohammadniaOranj.2009). Nilai MSE diperoleh dengan perhitungan sesuai persamaan 2.13

$$MSE = \frac{\sum_{i=1}^{n} (D_{ig} - D_{ik})^2}{n} \quad (2.\ 13)$$

dimana :

D_{ig} : jarak aktual semut pada periode *i*

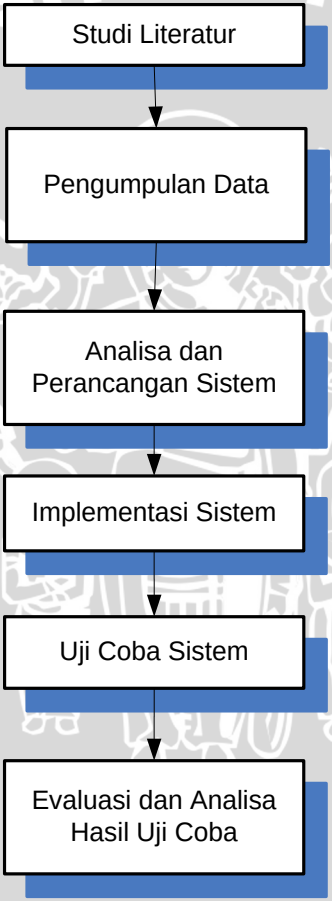
D_{ik} : jarak aktual *dijkstra* pada periode *i*

n : jumlah percobaan


UNIVERSITAS BRAWIJAYA

BAB III
METODOLOGI

Pada bab ini berisi tentang pembahasan metodologi yang digunakan dan langkah-langkah dalam penelitian tentang pencarian rute terpendek bersepeda menggunakan algoritma semut (*ant colony*). Sistem akan diterapkan pada mesin komputer personal (PC) dengan aplikasi berbasis C#. Pada sistem ini tahapan dari penelitian yang akan dilakukan ditunjukkan pada Gambar 3.1.



Gambar 3. 1 Diagram Sistem

Berdasarkan diagram sistem pada gambar 3.1, tahapan dari penelitian ini adalah sebagai berikut :

1. Studi literatur

   Pencarian informasi mengenai segala sesuatu yang berhubungan dengan algoritma semut dengan membaca buku dan *browsing*.
2. Pengumpulan data

   Pada tahap ini, pengumpulan data diambil dari data dinas pehubungan kota Malang. Serta melakukan wawancara dengan pihak-pihak terkait dengan penulisan skripsi ini.
3. Analisa dan Perancangan Sistem

   Pada tahap ini, dilakukan analisa dan perancangan sistem perangkat lunak untuk menerapkan algoritma semut.
4. Implementasi sistem

   Pada tahap ini, adalah tahap mengimplementasikan perangkat lunak berdasarkan analisa dan perancangan yang telah dilakukan.
5. Uji coba sistem

   Pada tahap ini, dilakukan pengujian terhadap perangkat lunak dengan memasukkan data latih dan data uji.
6. Evaluasi sistem

   Tahap ini adalah tahapan untuk melakukan evaluasi terhadap perangkat lunak dan data hasil uji coba yang diperoleh.

3.1 Analisis Perancangan Sistem

3.1.1 Deskripsi Umum Sistem

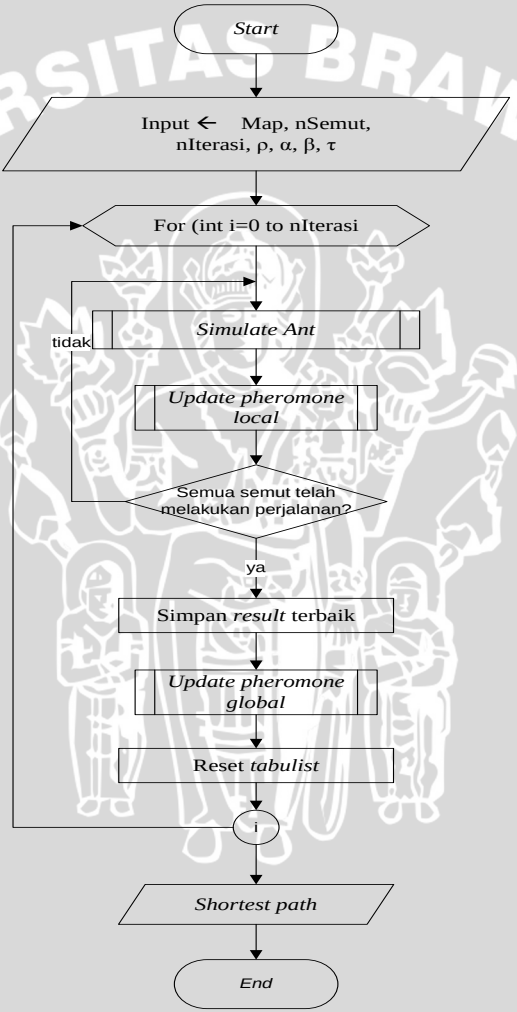
Sistem yang akan dibuat dalam skripsi ini adalah sebuah sistem yang mengimplementasikan algoritma semut dalam permasalahan penentuan rute terpendek untuk perjalanan bersepeda. Dalam mencapai tempat tujuan, terdapat banyak rute yang bisa dipilih. Pengguna sepeda akan cenderung mencari rute terpendek dengan tujuan agar cepat sampai ke tempat tujuan. Algoritma semut diharapkan dapat menemukan solusi dalam pencarian rute yang optimal, yaitu rute yang memiliki jarak tempuh terpendek.

3.1.2 Analisis Data

Data yang digunakan dalam penelitian ini merupakan data jalan yang diperoleh dari Dinas Perhubungan kota Malang dan *googlemaps*, yang diperoleh dari *website* www.maps.google.com.

3.2 Perancangan Sistem

Pada subbab ini akan dibahas mengenai perancangan proses secara keseluruhan dari sistem. Proses algoritma semut terdiri dari beberapa fungsi yaitu *simulate ant*, rute selanjutnya, *update pheromone lokal*, dan *update pheromone global*. *Flowchart* untuk proses algortima semut yang digunakan terdapat pada gambar 3.2



Gambar 3. 2 *Flowchart* Membangun Rute Perjalanan

3.2.1 Perancangan Proses Algoritma Semut

Dari *flowchart* pada gambar 3.2 dapat dijelaskan langkah-langkah proses pencarian rute terpendek menggunakan algoritma semut adalah sebagai berikut :

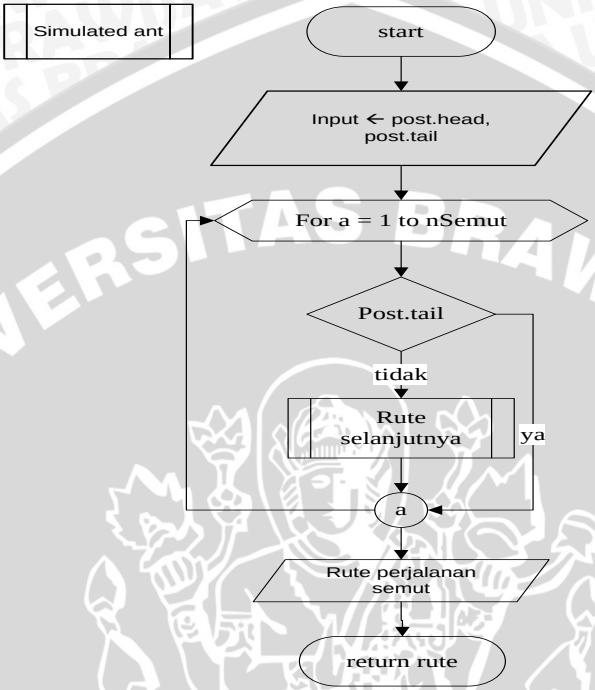
1. Inisialisasi parameter awal, yaitu:
   a. Parameter yang digunakan dalam penentuan rute terpendek yaitu nama jalan, titik penyusun jalan, dan panjang jalan
   b. Parameter algoritma semut, yaitu :
      1. Jumlah semut yang akan digunakan (m)
      2. Jumlah iterasi yang akan digunakan (NC_{max})
      3. Tetapan penguapan jejak semut (ρ)
      4. Tetapan pengendali intensitas jejak semut (α)
      5. Tetapan pengendali visibilitas (β)
      6. Intensitas jejak semut antar kota (τ)
2. Hasil dari inisialisasi tersebut akan dimasukkan ke dalam *tabulist* sebagai node awal semut.
3. Proses *simulate ant* yaitu mensimulasikan perjalanan semut sampai semua semut tiba pada tujuan tertentu secara random.
4. *Update pheromone lokal* dimana setiap semut setelah menuju titik selanjutnya akan selalu melakukan update *pheromone* untuk membuat tingkat ketertarikan ruas-ruas yang ada berubah secara dinamis.
5. Simpan *result* terbaik dari setiap perjalanan semut kedalam *tabulist.*
6. *Update pheromone global.*
7. *Reset tabulist.*
8. Jika kondisi terpenuhi, berhenti dan hasilnya adalah solusi rute terpendek

3.2.2 Perancangan Proses *Simulate Ant*

Proses *simulate ant* adalah proses bagaimana algoritma semut dapat membangun rute perjalanan setiap semut untuk menemukan tujuan yang telah ditentukan sebelumnya. Langkah - langkah dari proses *simulate ant* adalah sebagai berikut :

1. Telah ditentukan posisi awal dan posisi tujuan.
2. Lakukan proses *simulate ant* sebanyak jumlah semut yang telah ditentukan.

Gambar 3.3 merupakan deskripsi dari flowchart proses *simulate ant*



Gambar 3. 3 *Flowchart Simulate Ant*

3.2.3 Perancangan Proses Penentuan Rute Selanjutnya

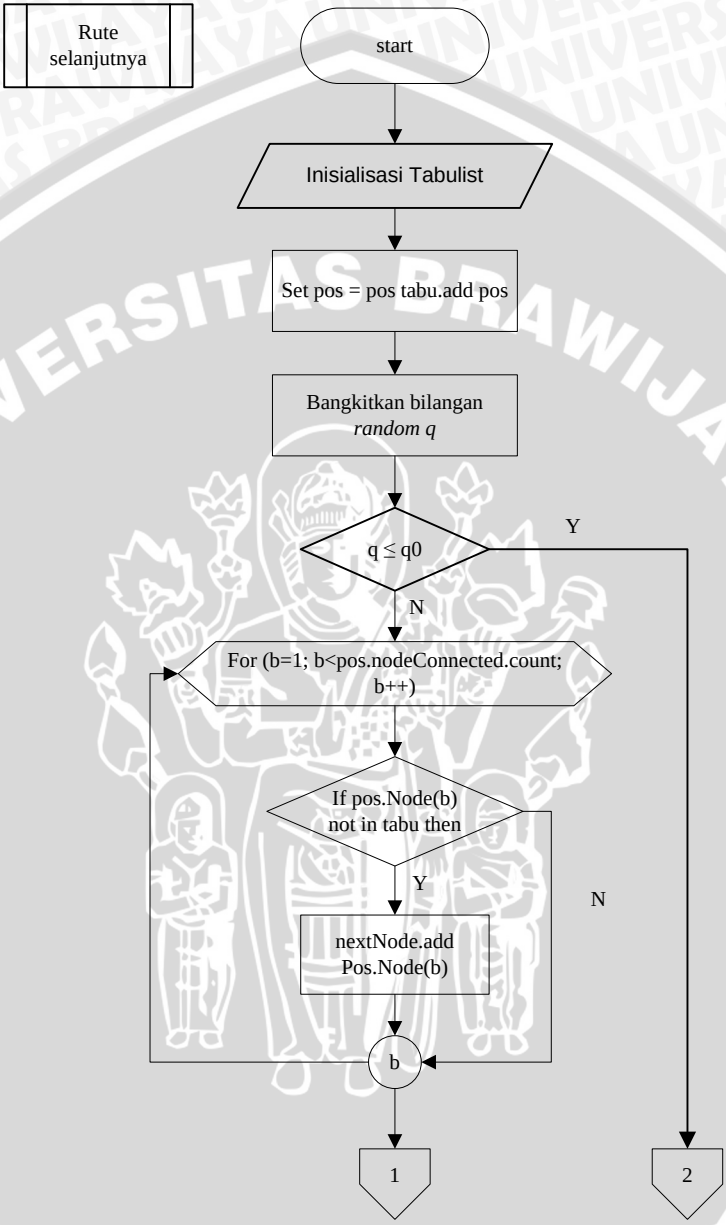
Pada proses ini akan menunjukkan bagaimana setiap semut melakukan perhitungan probabilitas untuk memilih jalur yang harus ditempuh dalam membangun perjalanannya. Langkah – langkah pada proses penentuan rute selanjutnya adalah sebagai berikut :

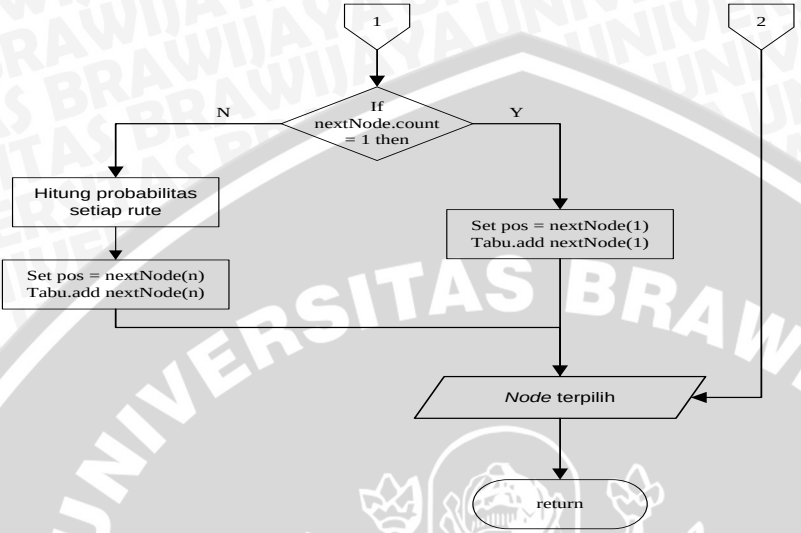
1. Mengisikan posisi awal semut kedalam *tabulist*. Inisialisasi *tabulist* yang telah ditentukan pertama, inisialisasi *tabulist* ini berfungsi sebagai *memory* semut untuk menyimpan informasi jalan yang pernah dilalui semut.
2. Bangkitkan bilangan random q untuk menentukan arah perjalanan semut selanjutnya.
3. Jika nilai $q \leq q0$ maka semut akan langsung menentukan rute perjalanan berikutnya dengan melakukan eksploitasi

dalam menentukan rute perjalanan selanjutnya dengan menggunakan persamaan 2.8

4. Jika q > q0 maka semut akan melakukan eksplorasi untuk memilih rute yang akan dilewati selanjutnya dengan menggunakan probabilitas pemilihan jalan dihitung dengan persamaan 2.1
5. Semut akan melakukan pengecekan kedalam *tabulist* apakah jalan yang akan dilewati tersebut sudah terlewati (terisi dalam *tabulist*) atau belum.
6. Semut akan melakukan pengecekan terhadap percabangan jalan.
7. Jika jalan tidak memiliki cabang maka semut akan langsung melanjutkan perjalanannya kemudian mengisikan kedalam *tabulist.*
8. Jika jalan memiliki cabang maka akan dilakukan perhitungan terhadap probabilitas jalan yang akan dipilih sebagai perjalanan berikutnya kemudian disimpan kedalam *tabulist.*
9. Terpilihnya rute selanjutnya.

Flowchart dari proses penentuan rute selanjutnya dapat dilihat pada gambar 3.4

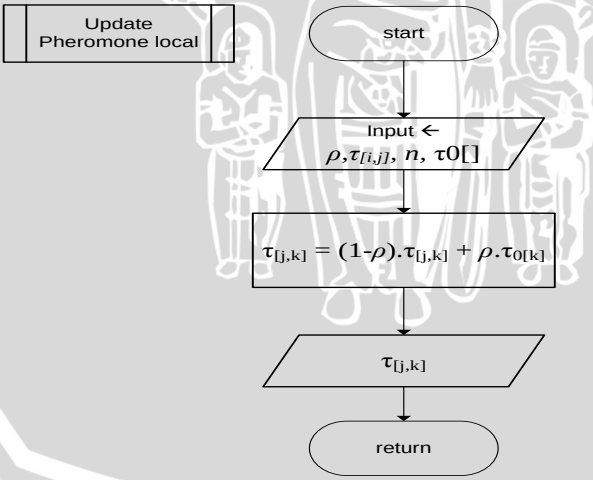

Rute selanjutnya
start
Inisialisasi Tabulist
Set pos = pos tabu.add pos
Bangkitkan bilangan *random q*
q ≤ q0
Y
N
For (b=1; b<pos.nodeConnected.count; b++)
If pos.Node(b) not in tabu then
Y
N
nextNode.add Pos.Node(b)
b
1
2



Gambar 3. 4 *Flowchart* Penentuan Rute Perjalanan Selanjutnya

3.2.4 Perancangan Proses Update Pheromone lokal

Proses ini terlihat pada gambar 3.5 bagaimana semut menandai rute jalan yang pernah atau sudah dilewati dengan menambahkan sejumlah *pheromone* ke jalur yang hanya dilewati ketika membangun awal membangun perjalanannya.

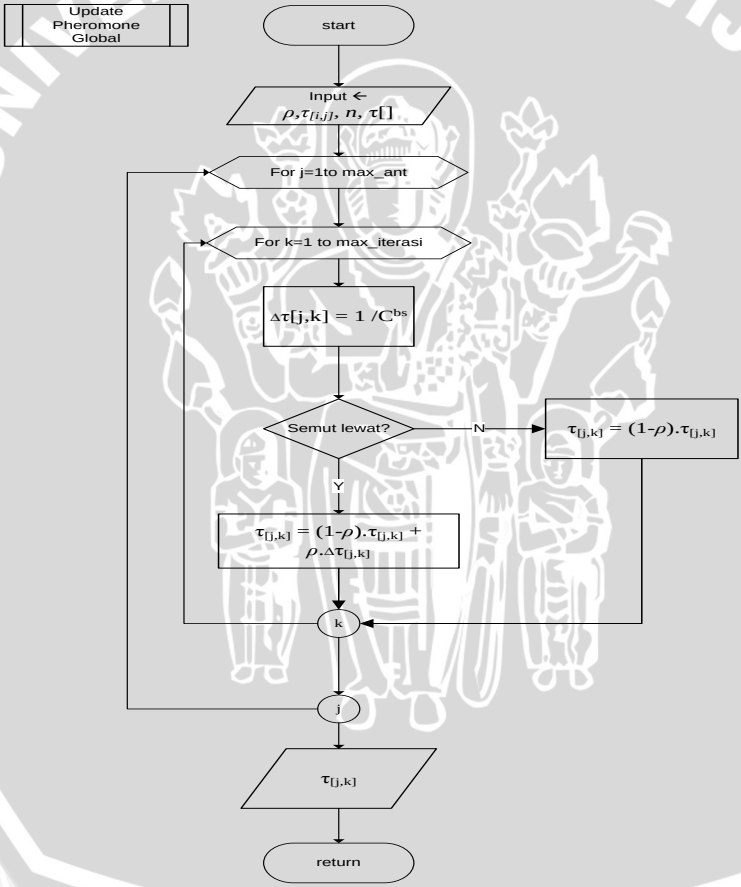


Gambar 3. 5 *Flowchart Update Pheromone lokal*

Pheromone ini digunakan sebagai petunjuk bagi semut lainnya untuk memilih jalan yang akan ditempuh dan juga terdapat proses *eveporation* atau penguapan *pheromone* yang membuat kadar *pheromone* pada suatu jalur semakin menipis bahkan menghilang karena semakin sedikitnya semut yang melewati jalur tersebut. Proses penguapan ini terjadi disetiap jalan.

3.2.5 Perancangan Proses Update Pheromone global

Proses ini dilakukan setelah semua semut telah selesai melakukan *tour* perjalanannya sampai menuju titik tujuan. Gambar 3.6 merupakan deskripsi dari proses *update pheromone global*



Gambar 3. 6 *Flowchart Update Pheromone Global*

Update pheromone global ini digunakan sebagai perhitungan nilai *pheromone* awal pada iterasi berikutnya dan juga sebagai petunjuk bagi semut lainnya untuk memilih jalan yang akan ditempuh pada iterasi berikutnya, nilai *update pheromone global* bergantung kepada rute terpendek yang telah dihasilkan oleh semut pada iterasi sebelumnya. Proses *eveporation* atau penguapan *pheromone* yang membuat kadar *pheromone* pada suatu jalur semakin menipis bahkan menghilang karena semakin sedikitnya semut yang melewati jalur tersebut.

3.3 Perancangan Basis Data

Tabel-tabel dalam *database* digunakan untuk menyimpan seluruh data yang digunakan oleh sistem. Tabel yang dapat disusun berdasarkan kebutuhan sistem ini yaitu tabel Jalan, tabel Cabang, tabel LampuLantas, dan tabel KoordinatPenyusun.

Pada tabel 3. 1 sampai dengan tabel 3. 3 akan dijelaskan tentang isi dari setiap tabel pada *database* yang akan diimplementasikan untuk program.

Tabel 3. 1 Tabel Jalan

No	Nama Atribut	Tipe Data	Keterangan
1	KodeJalan	*Long integer*	*Primary key, Notnull*
2	NamaJalan	*Text*(255)	*Notnull*
3	PanjangJalan	*Double*	*Notnull*
4	Point1	*Long integer*	*Notnull*
5	Point2	*Long integer*	*Notnull*

Keterangan :

- KodeJalan : Informasi tentang nomor identitas jalan.
- NamaJalan : Informasi tentang nama jalan.
- PanjangJalan : Informasi tentang panjang jalan.
- Point1 : Informasi tentang titik awal dari jalan x.
- Point2 : Informasi tentang titik akhir dari jalan x.

Tabel 3. 2 Tabel Cabang

No	Nama Atribut	Tipe Data	Keterangan
1	KodeCabang	*Long integer*	*Primary key, Notnull*
2	TitikCabang	*Long integer*	*Notnull*
3	KodeJalan	*Long integer*	*Notnull*

Keterangan :

- KodeCabang : Informasi tentang nomor identitas cabang.
- TitikCAbang : Informasi tentang posisi percabangan.
- KodeJalan : Informasi tentang nomor identitas jalan.

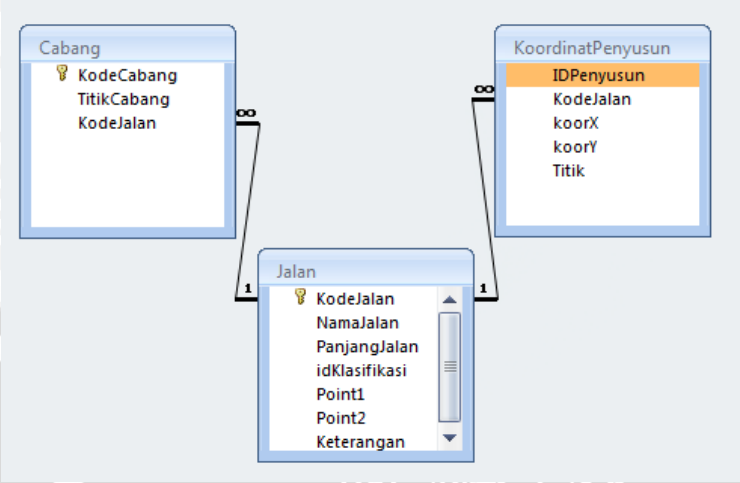
Tabel 3. 3 Tabel KoordinatPenyusun

No	Nama Atribut	Tipe Data	Keterangan
1	IDPenyusun	*Long integer*	*Notnull*
2	KodeJalan	*Long integer*	*Notnull*
3	KoorX	*Long integer*	*Notnull*
4	KoorY	*Long integer*	*Notnull*
5	Titik	*Long integer*	*Notnull*

Keterangan :

- IDPenyusun : Informasi tentang nomor identitas penyusun koordinat.
- KodeJalan : Informasi tentang nomor identitas jalan.
- KoorX : Informasi tentang koordinat pada sumbu x.
- KoorY : Informasi tentang koordinat pada sumbu y.
- Titik : Informasi titik jalan pada peta.

Pada gambar 3. 7 dibawah ini merupakan *entity relationship diagram* dalam database.



Gambar 3. 7 *Entity Relationship Diagram*

3.4 Perhitungan Manual Menggunakan *Ant Colony System*

Untuk dapat mengetahui dan mengimplementasikan kebenaran jalannya *ant colony system (ACS)* maka diberikan satu kasus sederhana, yang akan dicari rute terpendeknya. Kasus tersebut dapat dilihat pada Gambar 3.8 dengan

Gambar 3. 8 Ilustrasi Graph dengan 14 *Node*
(Sumber : Peta Wisata Kota Malang)

Pada kasus Gambar 3.8 diberikan studi kasus sederhana dengan 15 titik pada map. Data jarak jalan diberikan dalam tabel 3.4

Tabel 3. 4 Data Nama Jalan

No	Nama Jalan	Panjang Jalan (Km)	*Point* Jalan
1	Soekarno Hatta 1	1,15	4 dan 5
2	Soekarno Hatta 2	0,92	5 dan 6
3	Soekarno Hatta 3	0,92	6 dan 7
4	Soekarno Hatta 4	1,61	7 dan 8
5	Borobudur	1,34	8 dan 9
6	Letjen Suparman 1	0,65	23 dan 24
7	Letjen Suparman 2	0,65	9 dan 24

8	Lj Sutoyo 1	0,35	21 dan 22
9	LJ. Sutoyo 2	1	22 dan 23
10	Terusan Candi Mendut	0,55	6 dan 25
11	Kalpataru	0,8	26 dan 27
12	Cengkeh	0,5	27 dan 28
13	Coklat	0,17	5 dan 28
14	Kedawung	0,85	23 dan 26
15	Sarangan	0,65	22 dan 29
16	Cengger ayam	1,2	25 dan 26
17	Melati	1	26 dan 29
18	Candi Telogo Wangi	1,3	25 dan 24

Dari Gambar 3.8 akan dicari jarak terpendek antara jl. Soekarno hatta 3 (7) menuju ke jl. Sarangan (22) Tabel 3.5 merupakan matriks W untuk menyatakan graf pada Gambar 3.8

Tabel 3. 5 Jarak Antar Jalan

	4	5	6	7	8	9	22	23	24	25	26	27	28	29
4	∞	1,15	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
5	1,15	∞	0,92	∞	∞	∞	∞	∞	∞	∞	∞	∞	0,17	∞
6	∞	0,92	∞	0,92	∞	∞	∞	∞	∞	0,55	∞	∞	∞	∞
7	∞	∞	0,92	∞	1,61	∞	∞	∞	∞	∞	∞	∞	∞	∞
8	∞	∞	∞	1,61	∞	1,34	∞	∞	∞	∞	∞	∞	∞	∞
9	∞	∞	∞	∞	1,34	∞	∞	∞	0,45	∞	∞	∞	∞	∞
22	∞	∞	∞	∞	∞	∞	∞	1	∞	∞	∞	∞	∞	0,65
23	∞	∞	∞	∞	∞	∞	1	∞	0,65	∞	0,85	∞	∞	∞
24	∞	∞	∞	∞	∞	∞	∞	0,65	∞	1,3	∞	∞	∞	∞
25	∞	∞	0,55	∞	∞	∞	∞	∞	1,3	∞	1,2	∞	∞	∞
26	∞	∞	∞	∞	∞	∞	∞	0,85	∞	1,2	∞	0,8	∞	1
27	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	0,8	∞	0,5	∞
28	∞	0,17	∞	∞	∞	∞	∞	∞	∞	∞	∞	0,5	∞	∞
29	∞	∞	∞	∞	∞	∞	0,65	∞	∞	∞	1	∞	∞	∞

3.4.1 Inisialisasi Parameter

Langkah awal sebelum memasuki perhitungan pada tahap pertama dalam perhitungan algoritma *Ant Colony System* (ACS) ini terlebih dahulu dilakukan perhitungan awal untuk menghitung nilai invers jarak antar titik dengan titik lainnya menggunakan rumus $\eta(i,j)=\frac{1}{jarak\ (i,j)}$ sehingga dari perhitungan invers jarak tersebut akan didapatkan hasil seperti pada tabel 3.7

Kemudian dilakukan tahapan inisialisasi nilai parameter-paramater yang akan digunakan dalam perhitungan ACS yang diperlukan, yaitu pada tabel 3.6

Tabel 3. 6 Inisialisasi Parameter ACS

Parameter	Keterangan
Jumlah semut (m)	3
NCmax	2
ρ	0.9
β	0.9
α	0.1
τ_{ij}	0.5
q0	0.9

Nilai dari semua pheromone (τ_{ij}) pada awal perhitungan ditetapkan dengan angka awal yang sangat kecil. Pada contoh perhitungan penelitian ini nilai pheromone awal menggunakan nilai τ awal sebesar 0,5. Penetapan nilai pheromone awal dimaksudkan agar tiap-tiap ruas memiliki nilai ketertarikan untuk dikunjungi oleh tiap-tiap semut. Sedangkan jumlah semut yang akan digunakan dalam perhitunga manual ini sebanyak 3 semut dengan mengalami iterasi sebanya 2 kali.

Tabel 3. 7 Invers jarak *(η(i,j))*

	4	5	6	7	8	9	22	23	24	25	26	27	28	29
4	∞	0,87	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
5	0,87	∞	1,09	∞	∞	∞	∞	∞	∞	∞	∞	∞	5,88	∞
6	∞	1,09	∞	1,09	∞	∞	∞	∞	∞	1,82	∞	∞	∞	∞
7	∞	∞	1,09	∞	0,62	∞	∞	∞	∞	∞	∞	∞	∞	∞
8	∞	∞	∞	0,62	∞	0,75	∞	∞	∞	∞	∞	∞	∞	∞
9	∞	∞	∞	∞	0,75	∞	∞	∞	2,22	∞	∞	∞	∞	∞
22	∞	∞	∞	∞	∞	∞	∞	1,00	∞	∞	∞	∞	∞	1,54
23	∞	∞	∞	∞	∞	∞	∞	∞	1,54	∞	1,18	∞	∞	∞
24	∞	∞	∞	∞	∞	∞	∞	1,54	∞	0,77	∞	∞	∞	∞
25	∞	∞	1,82	∞	∞	∞	∞	∞	0,77	∞	0,83	∞	∞	∞
26	∞	∞	∞	∞	∞	∞	∞	1,18	∞	0,83	∞	1,25	∞	1,00
27	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	1,25	∞	2,00	∞
28	∞	5,88	∞	∞	∞	∞	∞	∞	∞	∞	∞	2,00	∞	∞
29	∞	∞	∞	∞	∞	∞	1,54	∞	∞	∞	1,00	∞	∞	∞

3.4.2 Pengisian Tabu List

Pengisian *tabu list* pertama atau posisi awal semut dilakukan secara random. Pada Gambar 3.9 merupakan gambar *tabulist* pertama kali. Awalnya *tabulist* tidak berisi atau kosong seperti ketika semut melakukan kunjungan ke kota berikutnya maka kota akan bertambah 1 pada *tabu list* dan seterusnya sampai *tabu list* terisi penuh. *Tabu list* dikatakan penuh ketika semua semut telah selesai melakukan tournya. Pengisian *tabulist* untuk ini berfungsi untuk menyimpan urutan kota yang telah dikunjungi, sehingga dapat juga diketahui kota yang belum dikunjungi oleh semut.

T_1				

Gambar 3. 9 Tabu List Semut Pertama

3.4.3 Penyusunan Rute Perjalanan Semut

Langkah selanjutnya adalah penyusunan rute perjalanan semut dimana semut akan mulai melakukan perjalanan dari kota pertama sebagai kota asal dan kota lainnya sebagai kota tujuan. Kemudian dari kota asal tersebut semut akan memilih kota yang belum pernah dilewati atau tidak terdapat dalam *tabulist* sebagai rute perjalanan selanjutnya. Semut akan melakukan perjalanan secara terus menerus sampai semua kota telah dikunjungi atau telah mengisi kedalam *tabulist*. Dalam pemilihan rute selanjutnya, pertama-tama dilakukan penetapan dari nilai $\beta \geq 0$.

Berikut ini adalah perhitungan dari penyusunan rute perjalanan semut berdasarkan nilai probabilitas dari kota awal sampai kota selanjutnya yang belum dilalui oleh semut pertama dalam satu siklus sesuai dengan persamaan 2.1. Dalam melakukan perhitungan terhadap probabilitas semut ke-1 pada iterasi pertama dalam memilih rute perjalanan selanjutnya. P_{76}^{1} menyatakan probabilitas semut ke-1 dari rute asal 7 menuju ke titik 6, hal tersebut sama seperti P_{78}^{1} yang juga menyatakan probabilitas dari semut ke-1 dari rute asal 7 menuju ke titik 8.

$$P_{(7,6)}^{1} = \frac{[0.5]^{0.1}[1/0.92]^{0.9}}{[0.5]^{0.1}[1/0.92]^{0.9} + [0.5]^{0.1}[1/1.61]^{0.9}}$$

$$= \frac{1.01}{1.01 + 0.61} = 0.62$$

$$P^{1}_{(7,8)} = \frac{[0.5]^{0.1}[1/1.61]^{0.9}}{[0.5]^{0.9}[1/0.92]^{0.9} + [0.5]^{0.1}[1/1.61]^{0.9}}$$
$$= \frac{0.61}{1{,}61} = 0{,}38$$

Setelah diperoleh nilai probabilitas antar jarak, selanjutnya akan dihitung nilai probabilitas komulatif dan *range* dari probabilitas tersebut dengan menggunakan persamaan 2.2. Hasil dari perhitungan nilai probabilitas seperti pada tabel 3.8

Tabel 3. 8 Probabilitas Komulatif Semut ke-1 Pada Generasi ke-1

Rute	Probabilitas (P_{ij})	Probabilitas P_{ij} Komulatif	*Range*
$P_{(7,7)}$	0	0	0
$P_{(7,6)}$	0.62	0.62	0 – 0.62
$P_{(7,8)}$	0.38	1	0.62 - 1

Pada perhitungan probabilitas pertama telah diperoleh nilai probabilitas dan *range*, kemudian untuk memilih rute perjalanan selanjutnya maka akan dibangkitkan angka desimal *random (q)* antara 0 sampai 1. Dengan bilangan *random q* dan kondisi $q \leq q0$, maka penentuan lokasi yang akan dituju berdasarkan hasil temporary yang paling besar. Jadi dengan nilai *random* = 0.4 hasilnya adalah $0.4 \leq 0.9$ yang artinya semut melakukan proses eksploitasi dengan probabilitas 90% dan proses eksplorasi 40%. Karena $q \leq q0$, maka rute yang dipilih berdasarkan persamaan 2.8 adalah $P_{(7,6)}$ dimana semut pertama berasal dari *edge* 7 akan menuju ke *edge* 6. Kemudian diisikan kedalam *tabu list* A(1) sementara adalah = 7 – 6.

Setelah semut menentukan rute perjalanan selanjutnya, kemudian dilakukan proses *update pheromone* (τ) secara lokal dengan menggunakan persamaan 2.9. berikut adalah contoh perhitungan dari proses *update pheromone* lokal

$$\Delta\tau_{ij} \leftarrow (1-\rho).\tau_{ij} + \rho.\tau_0$$
$$\Delta\tau_{(7,6)} \leftarrow (1-0.9)(0.5) + 0.9\left(\frac{1}{0.92 * 14}\right)$$
$$\Delta\tau_{(7,6)} \leftarrow (0.05) + (0.07)$$

$$\Delta\tau_{(7,6)} \leftarrow 0.12$$

Rute berikutnya yang belum di kunjungi adalah P_{25} dan P_5, jika nilai random q = 0.95 maka akan dilakukan eksplorasi dengan perhitungan persamaan 2.1 terhadap nilai probabilitas terhadap rute $P_{(6,25)}$, dan $P_{(6,5)}$

$$P_{(6,25)} = \frac{[0.5]^{0.1}[1/0.55]^{0.9}}{[0.5]^{0.1}[1/0.55]^{0.9} + [0.5]^{0.1}[1/0.92]^{0.9}}$$
$$= \frac{1.6}{2.6} = 0.61$$

$$P_{(6,5)} = \frac{[0.5]^{0.1}[1/0.92]^{0.9}}{[0.5]^{0.1}[1/0.55]^{0.9} + [0.5]^{0.1}[1/0.92]^{0.9}}$$
$$= \frac{1.01}{2.6} = 0.39$$

Tabel 3. 9 Probabilitas Komulatif Semut ke-1 Pada Generasi ke-1

Rute	Probabilitas (P_{ij})	Probabilitas P_{ij} Komulatif	*Range*
$P_{(6,6)}$	0	0	0
$P_{(6,25)}$	0.61	0.61	0 – 0.61
$P_{(6,5)}$	0.39	1	0.61 - 1

Jika nilai *random r,* dan kondisi $P_k(n\text{-}1) \leq r \leq P_k(n)$ dipenuhi,maka yang terpilih nilai *random r* adalah angka 0.96, maka berdasarkan range probabilitas komulatif, rute yang memenuhi adalah rute $P_{(6,5)}$ sehingga semut akan berjalan ke *node* 5.

Maka isi dari *tabu list* A(2) adalah 7-6-5 kemudian dilakukan proses *update pheromone* (τ) secara lokal

$$\Delta\tau_{(6,5)} \leftarrow (1 - 0.9)(0.5) + 0.9\left(\frac{1}{0.92 * 14}\right)$$
$$\Delta\tau_{(6,5)} \leftarrow 0.12$$

Rute yang belum terlewati adalah *edge* 4 dan *edge* 28, memilih rute selanjutnya menggunakan *state transition rule* dengan nilai *random q* = 0,5 maka eksploitasi yang terpilih

$$P_{(5,4)} = \frac{[0.5]^{0.1}[1/1.15]^{0.9}}{[0.5]^{0.1}[1/1.15]^{0.9} + [0.5]^{0.1}[1/0.17]^{0.9}}$$
$$= \frac{0.82}{5.42} = 0.15$$
$$P_{(5,28)} = \frac{[0.5]^{0.1}[1/0.17]^{0.9}}{[0.5]^{0.1}[1/1.15]^{0.9} + [0.5]^{0.1}[1/0.17]^{0.9}}$$
$$= \frac{4.6}{5.42} = 0.85$$

Dengan nilai maksimal adalah 4.6, maka rute selanjutnya adalah P_{28}. Kemudian dilakukan *update pheromone lokal*

$$\Delta\tau_{(5,28)} \leftarrow (1 - 0.9)(0.5) + 0.9\left(\frac{1}{0.17 * 14}\right)$$
$$\Delta\tau_{(5,28)} \leftarrow 0.43$$

Tabel 3. 10 Probabilitas Komulatif Semut ke-1 Pada Generasi ke-1

Rute	Probabilitas (P_{ij})	Probabilitas P_{ij} Komulatif	*Range*
$P_{(5,5)}$	0	0	0
$P_{(5,4)}$	0.15	0.15	0 – 0.15
$P_{(5,28)}$	0.85	1	0.15 - 1

Maka isi *tabu list* A(3) adalah 7-6-5-28 dan *edge* yang belum terlewati adalah *edge* 27. Karena pada *edge* 27 tidak memiliki percabangan maka semut akan langsung melanjutkan perjalanannya menuju ke *edge* 26, maka *tabu list* A(4)dan(5) berisi 7-6-5-28-27-26, perhitungan update pheromone lokal pada tabu list (4) dan (5) adalah

$$\Delta\tau_{(28,27)} \leftarrow (1 - 0.9)(0.5) + 0.9\left(\frac{1}{0.5 * 14}\right)$$
$$\Delta\tau_{(28,27)} \leftarrow 0.18$$
$$\Delta\tau_{(27,26)} \leftarrow (1 - 0.9)(0.5) + 0.9\left(\frac{1}{0.8 * 14}\right)$$
$$\Delta\tau_{(27,26)} \leftarrow 0.13$$

node yang belum terlewati adalah *node* 23,25 dan 29. Jika nilai *random* yang terpilih adalah angka 0.3, maka penentuan lokasi

yang akan dituju yaitu dengan melihat hasil temporary yang memiliki nilai paling besar.

$$P_{(26,23)} = \frac{[0.5]^{0.1}[1/0.85]^{0.9}}{[0.5]^{0.1}[1/0.85]^{0.9} + [0.5]^{0.1}[1/1.2]^{0.9} + [0.5]^{0.1}[1/1]^{0.9}} = \frac{1.08}{2.8} = 0.39$$

$$P_{(26,25)} = \frac{[0.5]^{0.1}[1/1.2]^{0.9}}{[0.5]^{0.1}[1/0.85]^{0.9} + [0.5]^{0.1}[1/1.2]^{0.9} + [0.5]^{0.1}[1/1]^{0.9}} = \frac{0.79}{2.8} = 0.28$$

$$P_{(26,29)} = \frac{[0.5]^{0.1}[1/1]^{0.9}}{[0.5]^{0.1}[1/0.85]^{0.9} + [0.5]^{0.1}[1/1.2]^{0.9} + [0.5]^{0.1}[1/1]^{0.9}} = \frac{0.93}{2.8} = 0.33$$

Tabel 3. 11 Probabilitas Komulatif Semut ke-1 Pada Generasi ke-1

Rute	Probabilitas (P_{ij})	Probabilitas P_{ij} Komulatif	*Range*
$P_{(26,26)}$	0	0	0
$P_{(26,23)}$	0.39	0.39	0 – 0.39
$P_{(26,25)}$	0.28	0.67	0.39 – 0.67
$P_{(26,29)}$	0.33	1	0.67 – 1

Berdasarkan perhitungan diatas maka nilai temporary terbesar adalah 1.08, maka rute yang terpilih adalah rute $P_{(26,23)}$ sehingga semut akan berjalan ke *edge* 23. Isi *tabu list* A(6) adalah 7-6-5-28-27-26-23 dan dilakukan *update pheromone lokal*

$$\Delta\tau_{(26,23)} \leftarrow (1 - 0.9)(0.5) + 0.9\left(\frac{1}{0.85 * 14}\right)$$

$$\Delta\tau_{(26,23)} \leftarrow 0.13$$

node yang belum terlewati adalah *node* 22 atau tujuan dari perjalanan semut. Karena pada *node* 22 adalah tujuan maka secara langsung semut akan menuju ke *node* 22, sehingga *tabu list* A(7) akan berisi 7-6-5-28-27-26-23-22.

$$\Delta\tau_{(23,22)} \leftarrow (1-0.9)(0.5) + 0.9\left(\frac{1}{1*14}\right)$$

$$\Delta\tau_{(23,22)} \leftarrow 0.11$$

T_7	T_6	T_5	T_{28}	T_{27}	T_{26}	T_{23}	T_{22}

Gambar 3. 10 Isi *Tabu List* Pertama Perjalanan Semut (A) Pada Iterasi Pertama

Pada semut ke-B dan ke-C dilakukan proses yang sama dilakukan seperti pada semut ke-1, untuk singkatnya proses pemilihan rute oleh semut ke-B dan ke-C ditunjukkan pada Tabel 3.12 dan Tabel 3.13.

Tabel 3. 12 Pemilihan Rute Perjalanan Oleh semut ke-B Iterasi 1

No	Nilai q	Keterangan	Pilih Rute	*Tabulist*
1	-	pemilihan acak	8	7,8
2	0,8	eksploitasi	9	7,8,9
3	0,5	eksploitasi	24	7,8,9,24
4	0,6	eksploitasi	23	7,8,9,24,23
5	-	tujuan	22	7,8,9,24,23,22

Tabel 3. 13 Pemilihan Rute Perjalanan Oleh semut ke-C Iterasi 1

No	Nilai q	Keterangan	Pilih Rute	*Tabulist*
1	-	pemilihan acak	6	7,6
2	0,46	eksploitasi	25	7,6,25
3	0,59	eksploitasi	26	7,6,25,26
4	0,98	eksplorasi, r=0,54	29	7,6,25,26,29
5	-	tujuan	22	7,6,25,26,29,22

3.4.4 Perhitungan Panjang Jalur Setiap Semut

Perhitungan panjang jalur perjalanan setiap semut ini dilakukan setelah satu siklus perjalanan diselesaikan oleh semua semut, panjang jalur atau L_k dihitung berdasarkan memori perjalanan

semut yang tersimpan dalam *tabu list*. Perhitungan panjang rute menggunakan persamaan 2.2 dengan menambahkan bobot jarak pada setiap *edge* berdasarkan rute yang dilewati semut dan tersimpan dalam *tabu list.* Seperti pada gambar 3.10 maka panjang rute perjalanan semut pada iterasi pertama tersebut adalah :

Tabel 3. 14 Hasil Panjang Jalur Semut Pada Iterasi Pertama

semut ke-	Rute								panjang rute
A	7	6	5	28	27	26	23	22	5,16
B	7	8	9	24	23	22	-	-	5,25
C	7	6	25	26	29	22	-	-	4,32

Selanjutnya setelah panjang jalur atau L_k setiap semut dihitung, maka untuk mendapatkan rute perjalanan terpendek dari setiap siklus atau L_{minNC} dan harga minimal panjang jalur secara keseluruhan adalah atau L_{min}. Sehingga hasil dari iterasi pertama perjalanan semut dapat ketahui rute perjalanan terpendek semut pada iterasi pertama adalah semut C

3.4.5 Perhitungan *Update Pheromone* Global

Perjalanan koloni semut akan meninggalkan jejak-jejak kaki pada semua lintasan antar *node* yang dilalui. Adanya 'penguapan' dan perbedaan jumlah semut yang melewati rute tersebut menyebabkan kemungkinan terjadinya perubahan harga intensitas jejak kaki semut antar *node*. Sebelum menghitung *delta pheromone* semua semut, akan dihitung *delta pheromone* untuk masing-masing *edge* yang dilalui oleh semut yang mempunyai *tabu list* dengan hasil rute optimal. Perhitungan menggunakan persamaan 2.11.

Tabu list dengan rute terbaik pada tabel 3.14 didapatkan dari perjalanan semut (C) dengan memiliki nilai panjang jalur terpendek pada akhir iterasi, maka rute inilah yang akan digunakan sebagai *global best tour*,berdasarkan persamaan 2.11 :

- Untuk *edge* yang dilewati semut :

$$\Delta\tau_{ij} = \frac{1}{C^{bs}} = \frac{1}{4.32} = 0.23$$

- Untuk edge yang tidak dilewati semut :
  $\Delta\tau_{ij} = 0$

Oleh karena itu *pheromone* yang lain dilakukan evaporasi *pheromone* global untuk urutan *i* dan *j* yang bukan elemen dari rute semut terbaik yaitu:

- Untuk edge yang dilewati semut
  $\tau_{ij} \leftarrow (1-\rho).\tau_{ij} + \rho.\Delta\tau_{ij}^{bs}$
  $\tau_{ij} \leftarrow (1-0.9) * 0.5 + 0.9 * 0.23$
  $= 0.257$

- Untuk edge yang tidak dilewati semut
  $\tau_{ij} \leftarrow (1-0.9) * 0.5 + 0$
  $= 0.05$

Tabel 3. 15 Hasil *Update Pheromone Global* Iterasi Pertama

τ	4	5	6	7	8	9	22	23	24	25	26	27	28	29
4	0,05	0,05	0,05	0,05	0,05	0,05	0,05	0,05	0,05	0,05	0,05	0,05	0,05	0,05
5	0,05	0,05	0,05	0,05	0,05	0,05	0,05	0,05	0,05	0,05	0,05	0,05	0,05	0,05
6	0,05	0,05	0,05	**0.257**	0,05	0,05	0,05	0,05	0,05	**0.257**	0,05	0,05	0,05	0,05
7	0,05	0,05	**0.257**	0,05	0,05	0,05	0,05	0,05	0,05	0,05	0,05	0,05	0,05	0,05
8	0,05	0,05	0,05	0,05	0,05	0,05	0,05	0,05	0,05	0,05	0,05	0,05	0,05	0,05
9	0,05	0,05	0,05	0,05	0,05	0,05	0,05	0,05	0,05	0,05	0,05	0,05	0,05	0,05
22	0,05	0,05	0,05	0,05	0,05	0,05	0,05	0,05	0,05	0,05	0,05	0,05	0,05	**0.257**
23	0,05	0,05	0,05	0,05	0,05	0,05	0,05	0,05	0,05	0,05	0,05	0,05	0,05	0,05
24	0,05	0,05	0,05	0,05	0,05	0,05	0,05	0,05	0,05	0,05	0,05	0,05	0,05	0,05
25	0,05	0,05	**0.257**	0,05	0,05	0,05	0,05	0,05	0,05	0,05	**0.257**	0,05	0,05	0,05
26	0,05	0,05	0,05	0,05	0,05	0,05	0,05	0,05	0,05	**0.257**	0,05	0,05	0,05	**0.257**
27	0,05	0,05	0,05	0,05	0,05	0,05	0,05	0,05	0,05	0,05	0,05	0,05	0,05	0,05
28	0,05	0,05	0,05	0,05	0,05	0,05	0,05	0,05	0,05	0,05	0,05	0,05	0,05	0,05
29	0,05	0,05	0,05	0,05	0,05	0,05	**0.257**	0,05	0,05	0,05	**0.257**	0,05	0,05	0,05

3.4.6 Pengosongan *Tabu List*

Pengosongan *tabu list* dilakukan apabila belum mencapai iterasi maksimum dari inisialisasi parameter awal atau belum mencapai nilai konvergen, maka ulangi langkah perhitungan mulai dari pengisian *tabu list* dengan *pheromone* yang telah diperbarui. *Tabu list* perlu dikosongkan untuk diisi lagi dengan urutan *node* yang baru pada iterasi selanjutnya. Proses ini akan diulang lagi hingga mencapai jumlah iterasi maksimum atau telah berada dalam kondisi konvergen.

3.4.7 Perhitungan Pada Iterasi Kedua

Pada iterasi kedua, posisi semut di atur kembali ke *edge* awal yang akan dipilih kembali secara *random*, pengerjaannya sama dengan iterasi pertama. Hanya saja, nilai dari *pheromone* masing-masing *edge* telah berubah. Perubahan nilai *pheromone* yang telah di proses sebelumnya tersebut terdapat pada tabel 3.15

Tabel 3. 16 Pemilihan Rute Oleh Semut ke-A Iterasi 2

No	Nilai q	Keterangan	Pilih Rute	*Tabulist*
1	-	pemilihan acak	6	7,6
2	0,95	eksplorasi, r=0,56	25	7,6,25
3	0,94	eksplorasi, r=0,56	26	7,6,25,26
4	0,93	eksplorasi, r=0,23	23	7,6,25,26,23
5	-	tujuan	22	7,6,25,26,23,22

Tabel 3. 17 Pemilihan Rute Oleh Semut ke-B Iterasi 2

No	Nilai q	Keterangan	Pilih Rute	*Tabulist*
1	-	pemilihan acak	6	7,6
2	0,8	eksploitasi	25	7,6,25
3	0,95	eksplorasi, r=0,3	24	7,6,25,24
4	0,6	eksploitasi	23	7,6,25,24,23
5	-	tujuan	22	7,6,25,24,23,22

Tabel 3. 18 Pemilihan Rute Oleh Semut ke-C Iterasi 2

No	Nilai q	Keterangan	Pilih Rute	*Tabulist*
1	-	pemilihan acak	6	7,6
2	0,46	eksploitasi	25	7,6,25
3	0,59	eksploitasi	26	7,6,25,26
4	0,98	eksplorasi, r=0,54	29	7,6,25,26,29
5	-	tujuan	22	7,6,25,26,29,22

Setelah semua semut sampai pada *edge* akhir, maka dilakukan perhitungan panjang rute perjalanan terhadap semua semut pada iterasi kedua sampai tujuan dan *tabulist* penuh, sehingga didapatkan tabel 3.19 sebagai hasil perjalanan semua semut pada iterasi kedua

Tabel 3. 19 Hasil Panjang Jalur Semut Pada Iterasi Kedua

semut ke-	Rute						panjang rute
A	7	6	25	26	23	22	4,52
B	7	6	25	24	23	22	4,42
C	7	6	25	26	29	22	4,32

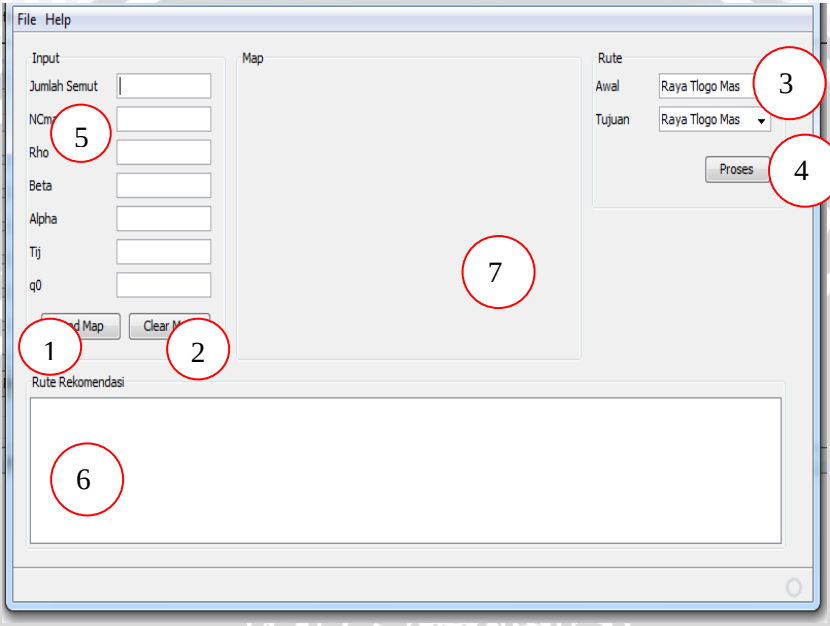
Dari perhitungan manual menggunakan ACS (*Ant Colony System)* dengan menggunakan 2 iterasi maka dapat dikatakan bahwa rute perjalanan oleh semut C merupakan rute terpendek yang memiliki rute perjalanan yang dilewati adalah 7-6-25-26-29-22 dengan panjang rute 4,32Km dengan waktu tempuh

3.5 Perancangan Antarmuka

Pada subbab ini akan dijelaskan mengenai perancangan antar muka yang digunakan dalam aplikasi pencarian rute terpendek bersepeda dengan algoritma ACS (*Ant Colony System*). Berdasarkan hasil analisis secara keseluruhan terdapat beberapa bagian yang dibutuhkan dalam antarmuka sistem ini, yaitu :

3.5.1 *Form* Utama

Pada rancangan antarmuka menu utama ini, semua bagian dan komponen-komponen yang dibutuhkan pada menu utama akan disajikan dalam satu halaman antarmuka. Rancangan antarmuka dari menu utama ini dapat dilihat pada Gambar 3.11.



Gambar 3. 11 Rancangan Antarmuka *Input data* untuk Pencarian Rute Terpendek

Keterangan:

1. Tombol *Load* Map untuk mencari direktori dari *folder* yang terdapat map, map hanya berformat .jpg
2. Tombol *Clear* Map untuk menghapus map dan mereset ulang proses perhitungan ACS
3. *Combo Box* yang digunakan untuk menentukan titik awal dan titik tujuan.
4. Tombol Proses adalah untuk menghitung menggunakan algoritma ACS
5. *Text field* Inputan Parameter ACS

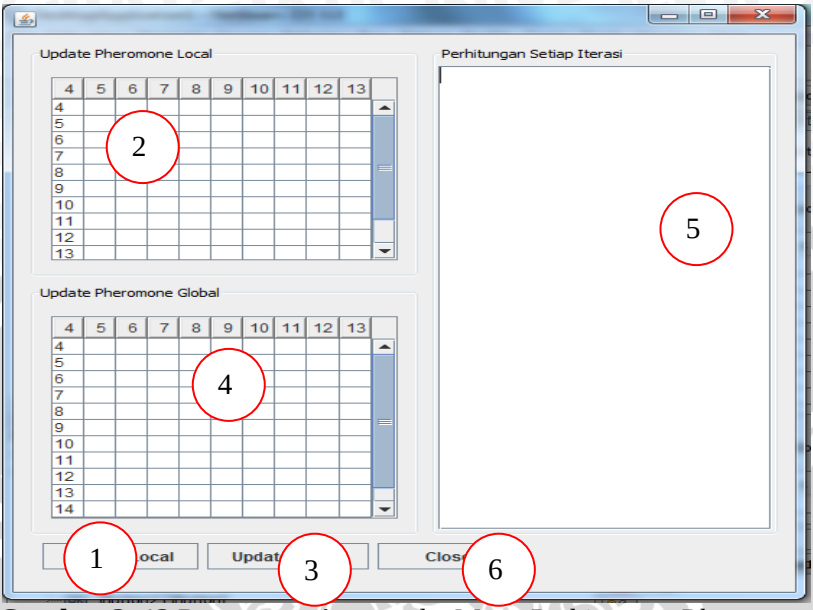
6. *Text field output* dari perhitungan menggunakan algoritma ACS berupa rekomendasi rute dan waktu tempuh perjalanan dengan bersepeda
7. Berisikan Map yang telah di *load*

3.5.2 Inisialisasi *Pheromone*

Ada beberapa bagian yang dibutuhkan dalam rancangan antarmuka pada halaman ini, yaitu :

1. Bagian untuk memproses nilai *pheromone* lokal. Dengan menekan tombol ini hasil perhitungan *pheromone* lokal akan tampil pada area *update pheromone lokal.*
2. Bagian untuk menampilkan hasil *updating pheromone* lokal.
3. Bagian untuk memproses nilai *pheromone* global maupun hasil dari perhitungan sistem. Dengan menekan tombol ini hasil perhitungan global *pheromone* akan tampil pada area *update pheromone global* dan pada area perhitungan setiap iterasi akan memberikan informasi dari perhitungan dengan ACS dengan hasil perhitungan dari tiap generasi dan hasil akhir berupa generasi terbaik.
4. Bagian untuk menampilkan hasil *updating* global *pheromone*
5. Bagian untuk menampilkan hasil perhitungan dari tiap generasi dan hasil akhir berupa generasi terbaik.
6. Bagian untuk keluar dari menu ini. Dengan menekan tombol ini *user* akan kembali ke halaman utama.

Rancangan antarmuka dari menu ini dapat dilihat pada gambar 3.12



Gambar 3. 12 Rancangan Antarmuka Menu Perhitungan *Pheromone*

3.6 Perancangan Uji Coba dan Evaluasi

Setelah perangkat lunak telah terbentuk, langkah selanjutnya adalah melakukan pengujian dan juga evaluasi sistem. Uji coba dilakukan untuk mengevaluasi keakuratan solusi yang diberikan oleh sistem. Dari pengujian ini akan digunakan beberapa parameter tetap ketika parameter tersebut tidak mengalami pengujian atau perubahan nilai yaitu :

Tabel 3. 20 *Default* nilai parameter yang digunakan dalam uji coba

Parameter	Keterangan
Jumlah semut (m)	10
NCmax	5
ρ	0.5
β	0.5
α	0.1
$\tau 0$	0.1
$q0$	0.5

Dari nilai *default* parameter ini akan dilakukan perubahan nilai parameter seperti pada tabel 3.18 nilai parameter yang mengalami perubahan adalah nilai parameter jumlah semut sehingga nilai parameter lainnya bernilai seperti *default*, dan seterusnya. Beberapa parameter yang akan diuji dengan melakukan perubahan nilai parameter adalah jumlah semut, intensitas *pheromone* awal(τ_{ij}), α, β, ρ, dan jumlah iterasi semut. Pada tabel 3.21 sampai dengan tabel 3.26 merupakan perancangan uji coba sistem dengan perubahan nilai dari parameter semut.

Tabel 3. 21 Rancangan Uji Coba Jalur dengan Perubahan Parameter Jumlah Semut

Semut	Uji Coba					Jarak Rata-Rata
	1	2	3	4	5	
10						
20						
30						
40						
50						

Tabel 3. 22 Rancangan Uji Coba Jalur dengan Perubahan Parameter intensitas *pheromone* (τ_{ij})

τ_{ij}	Uji Coba					Jarak Rata-Rata
	1	2	3	4	5	
0,1						
0,2						
0,3						
0,4						
0,5						

Tabel 3. 23 Rancangan Uji Coba Jalur dengan Perubahan Parameter α

α	Uji Coba					Jarak Rata-Rata
	1	2	3	4	5	
0,1						
0,3						
0,5						
0,7						
0,9						

Tabel 3. 24 Rancangan Uji Coba Jalur dengan Perubahan Parameter β

β	Uji Coba					Jarak Rata-Rata
	1	2	3	4	5	
0,1						
0,3						
0,5						
0,7						
0,9						

Tabel 3. 25 Rancangan Uji Coba Jalur dengan Perubahan Parameter *ρ*

ρ	Uji Coba					Jarak Rata-Rata
	1	2	3	4	5	
0,1						
0,3						
0,5						
0,7						
0,9						

Tabel 3. 26 Rancangan Uji Coba Jalur dengan Perubahan Parameter Jumlah Iterasi

NC max	Uji Coba					Jarak Rata-Rata
	1	2	3	4	5	
5						
10						
15						
20						
50						

Pengujian kedua yang akan dilakukan adalah membandingkan kinerja dari algoritma semut dan *Dijkstra* berdasarkan jarak dan waktu proses dari kedua algoritma tersebut. Pengujian dilakukan tanpa menambahkan titik singgah dengan dibagi menjadi tiga kategori yaitu jauh, menengah, dekat. Parameter algoritma semut yang digunakan merupakan nilai terbaik yang telah di dapat pada pengujian sebelumnya. Pada tabel 3.27 sampai dengan tabel 3.29 merupakan perancangan uji coba sistem perbandingan algoritma semut dan *Dijkstra* tanpa titik singgah.

Tabel 3. 27 Rancangan Uji Coba Jalur Tanpa Titik Singgah Jarak Dekat

Algoritma	Percobaan Ke-					Rata-rata
	1	2	3	4	5	
ACS						
Dijkstra						

Tabel 3. 28 Rancangan Uji Coba Jalur Tanpa Titik Singgah Jarak Menengah

Algoritma	Percobaan Ke-					Rata-rata
	1	2	3	4	5	
ACS						
Dijkstra						

Tabel 3. 29 Rancangan Uji Coba Jalur Tanpa Titik Singgah Jarak Jauh

Algoritma	Percobaan Ke-					Rata-rata
	1	2	3	4	5	
ACS						
Dijkstra						

Pengujian ketiga sama dengan penguian kedua hanya saja pada pengujian ketiga dengan menambahkan titik singgah, titik singgah ini adalah titik dimana kedua algoritma baik algoritma semut maupun algoritma *Dijkstra* harus melewati titik tersebut sebelum mencapai titik tujuan. Pada tabel 3.30 merupakan perancangan uji coba sistem perbandingan algoritma semut dan *Dijkstra* dengan menggunakan titik singgah.

Tabel 3. 30 Rancangan Uji Coba Jalur Dengan Titik Singgah

Algoritma	Percobaan Ke-					Rata-rata
	1	2	3	4	5	
ACS						
Dijkstra						

BAB IV
IMPLEMENTASI DAN PEMBAHASAN

4.1 Lingkungan Implementasi

Proses implementasi merupakan tahapan penerapan rancangan sistem pada bahasa pemrograman dan siap dioperasikan pada keadaan yang sebenarnya, sehingga akan diketahui apakah sistem yang dibuat telah menghasilkan tujuan yang diinginkan. Lingkungan implementasi yang akan dijelaskan pada bab ini meliputi lingkungan implementasi perangkat keras dan perangkat lunak.

4.1.1 Lingkungan Perangkat Keras

Dalam penelitian ini, perangkat keras yang digunakan dalam membangun sistem adalah sebuah notebook dengan spesifikasi sebagai berikut:

1. Prosesor Intel(R) Core(TM) I3-390M @2.67GHz
2. Memori 4GB DDR2
3. *Harddisk* dengan kapasitas 640 GB
4. Monitor 14”
5. Keyboard
6. Mouse

4.1.2 Lingkungan Perangkat Lunak

Perangkat lunak yang digunakan dalam penelitian dan pembangunan sistem ini adalah:

1. Sistem Operasi Microsoft Windows 7 Ultimate Edition 64 bit.
2. Microsoft Visual C# 2010 Ultimate Edition sebagai *software development* dalam implementasi perancangan sistem.
3. MS Access 2003 sebagai *software* pengolah *database*.

4.2 Implementasi Basis Data

Berdasarkan analisa dan perancangan proses yang terdapat pada bab 3 telah dijelaskan mengenai rancangan tabel dan relasi antar tabel dari basis data yang terdiri dari 3 tabel. Tabel-tabel tersebut yaitu tabel Jalan, tabel Cabang, tabel KoordinatPenyusun. Definisi dari masing-masing tabel dapat dilihat pada bab 3.

Tabel-tabel tersebut beserta relasinya diimplementasikan dengan menggunakan Microsoft Access 2007.

4.3 Implementasi Program

Penyusunan program untuk penentuan rute terpendek jalur bersepeda menggunakan algoritma semut memiliki beberapa bagian proses yang telah dirancang dan dijelaskan pada bab 3. Perangkat lunak yang dibangun adalah untuk aplikasi program berbasis *dekstop.*

Implementasi dari proses-proses ini memanfaatkan struktur data berupa *array* dan *arraylist*. Maka pada subbab ini akan dijelaskan implementasi proses-proses tersebut.

4.3.1 Struktur Data

Struktur data yang digunakan dalam implementasi pencarian rute terpendek menggunakan algoritma semut dapat dilihat pada *sourcecode* 4. 1

```
1   //parameter Jalan
2   Jalan[] dataJalan;
3   Koordinat[] dataKoordinat;
4   Cabang[] dataCabang;
5   string awal, akhir;
6   ArrayList titikTersedia;
7
8   //parameter Algoritma Semut
9   Int semut;
10  double FeromonAwal;
11  double Q0;
12  double Rho;
13  double Beta;
14  double Alpha;
15  int[] TotalTitik;
16  double[] JarakTotal;
17  static Random acak = new Random();
18  ArrayList Titik;
19  ArrayList Jalur;
20  ArrayList tabuList;
```

***Source Code* 4. 1** Struktur Data

Tabel 4. 1 Definisi Variabel

Variabel	Tipe	Keterangan
dataJalan	*array*	variabel yang berfungsi untuk menyimpan parameter jalan pada tabel jalan.

dataKoordinat	*array*	variabel yang berfungsi untuk menyimpan parameter pada tabel koordinat
dataCabang	*array*	variabel yang berfungsi untuk menyimpan parameter pada tabel cabang
lokasiAwal	*string*	berfungsi untuk menyimpan nilai titik awal
lokasiTujuan	*string*	berfungsi untuk menyimpan nilai titik tujuan
titikTersedia	*arraylist*	berfungsi untuk menyimpan titik yang terhubung dengan cabang x
semut	*integer*	berfungsi untuk mengisikan jumlah semut
FeromonAwal	*double*	berfungsi untuk mengisikan nilai *pheromone* awal disetiap titik jalan
TotalTitik	*integer*	berfungsi untuk menyimpan jumlah titik dari peta
JarakTotal	*double*	berfungsi untuk menyimpan jarak tempuh perjalanan semut paling optimal dari tiap generasi
Titik	*list*	berfungsi untuk menyimpan daftar titik yang akan digunakan dalam perjalanan
Jalur	*arraylist*	berfungsi untuk menyimpan jalur perjalanan terbaik dalam perjalanan dari tiap generasi
tabuList	*list*	berfungsi untuk menyimpan data jalur yang telah dilewati oleh semut dalam perjalanan dari tiap generasi

4.3.2 Penentuan Rute Perjalanan Semut

Pada proses membangun rute perjalanan awal, semut akan berjalan bebas sampai mencapai titik tujuan kemudian dilakukan proses pengecekan terhadap setiap semut yang telah mencapai titik tujuan. Jumlah semut awal berdasarkan pada jumlah yang diinputkan oleh user. Proses membangun rute perjalanan dapat dilihat pada *source code* 4. 2

```
1  . . . . . .
2  public bool punya(Titik akhir)
3    {
4        if (NamaJalan == akhir.NamaJalan)
5        {
6            return true;
7        }
8        if (Lanjutan.Any(j => j.NamaJalan ==
9  akhir.NamaJalan))
10       {
11           return true;
12       }
13       else
14       {
15           return Lanjutan.Any(j =>
16 j.punya(akhir));
17       }
18   }
19 Titik target;
20 . . . . . .
21 public void bersihkan()
22       {
23           Lanjutan.RemoveAll(x =>
24 !x.punya(target))
25           foreach (var lanjutan in Lanjutan)
26               {
27                   lanjutan.bersihkan();
28               }
29       }
30 . . . . . .
```

***Source Code* 4. 2** Membangun Rute Perjalanan

4.3.3 Pemilihan Rute Semut

Pada proses menentukan rute selanjutnya Semut diletakkan pada titik awal yang telah ditentukan *user* dan akan dilakukan inisialisasi titik awal semut kedalam suatu *tabulist.* Untuk menentukan titik mana yang akan dikunjungi selanjutnya akan dibangkitkan nilai random yang menentukan terjadinya proses eksploitasi atau eksplorasi. Jika $q \leq q0$ maka akan dilakukan proses eksploitasi sesuai dengan persamaan 2.8, jika nilai $q > q0$ maka akan dilakukan proses eksplorasi sesuai dengan persamaan 2.1. Setelah titik selanjutnya terpilih maka dilakukan *update pheromone* lokal

terhadap jalur yang terpilih. Titik awal dan titik tujuan sesuai dengan inputan *user*. Kemudian semut melakukan perjalanan dari titik awal sampai titik tujuan. Proses menentukan rute selanjutnya dapat dilihat pada *sourcecode* 4. 3

```
. . . . . .
if (!Dijkstra)
{
   if (titikTersedia.Contains(akhir) &&
(mampir== null ||
titikTersedia.Contains(mampir)))
   {
      var titikYangMungkin =
titikTersedia.Where(tT =>tT.titik1 ==
awal.titik1 || tT.titik1 == awal.titik2 ||
tT.titik2 == awal.titik1 || tT.titik2 ==
awal.titik2).ToDictionary(x => x, x => 0);
if (titikYangMungkin.Count == 0)
{
   return;
}
   if (mampir!=null &&
titikYangMungkin.ContainsKey(mampir))
   {
      titikYangMungkin = new[] {
titikYangMungkin[mampir] }.ToDictionary(x =>
mampir, x => x);
      mampir = null;
      titikTersedia = Titik.daftar.Where(x =>
x != titikYangMungkin.Keys.First()).ToList();
   }
foreach (var t in titikYangMungkin.Keys)
{
   if (t==akhir)
   {
      continue;
   }
   titikTersedia.Remove(t);
}
foreach (var fer in titikYangMungkin.Keys)
   {
      fer.FeromonLokal = fer.FeromonGlobal;
   }
```

```
// double nilai=0;

for (int index = 0; index < jumlahSemut;
index++)
{
   var q = acak.NextDouble();
//eksplorasi
   if (q > Q0)
   {
      var total = titikYangMungkin.Keys.Sum(x
=> Math.Pow((1 / x.PanjangJalan), Jalur.Beta)
* Math.Pow(x.FeromonLokal, Alpha));
      var nilai = acak.NextDouble() * total;
      foreach (var item in
titikYangMungkin.Keys)
      {
         nilai -= Math.Pow((1 /
item.PanjangJalan), Jalur.Beta) *
Math.Pow(item.FeromonLokal, Alpha);
         if (nilai <= 0)
         {
            item.FeromonLokal = (1 - Rho) *
item.FeromonLokal + Rho * (1 /
(item.PanjangJalan * TotalTitik));
            titikYangMungkin[item]++;
            break;
         }
      }
   }
//eksploitasi
   else
      {
         var titikTerpilih =
titikYangMungkin.OrderBy(x => Math.Pow((1 /
x.Key.PanjangJalan), Beta) *
x.Key.FeromonLokal).First().Key;
         titikYangMungkin[titikTerpilih]++;
         titikTerpilih.FeromonLokal = (1 -
Rho) * titikTerpilih.FeromonLokal + Rho * (1 /
(titikTerpilih.PanjangJalan * TotalTitik));
      }
}
```

***Source Code* 4. 3** Menentukan Rute Selanjuntnya

4.3.4 Penyimpanan Rute Perjalanan ke TabuList

Pada proses pengisian ke dalam tabulist setiap titik jalan yang terpilih dimasukkan dalam tabu list/daftar jalur semut sehingga dapat diketahui rute terbaik serta dapat diketahui jarak perjalanan setiap semut pada setiap generasi. Proses penyimpanan data dapat dilihat pada Sourcecode 4.4.

```
1   public List<Titik> tabuList = new
2   List<Titik>();
3   public double JarakTotal
4   {
5      get
6      {
7         return tabuList.Sum(x => x.PanjangJalan);
8      }
9   }
10  public static Jalur Terbaik;
11  public Jalur(Titik awal, Titik akhir,
12  List<Titik> titikTersedia, Jalur sebelumnya,
13  int jumlahSemut, bool Dijkstra)
14     {
15        target = akhir;
16        NamaJalan = awal.NamaJalan;
17        Jarak = awal.PanjangJalan;
18        //isi tabulist
19        if (sebelumnya != null)
20           {
21              tabuList =
22  sebelumnya.tabuList.ToList();
23           }
24           tabuList.Add(awal);
25           Lanjutan = new List<Jalur>();
26  //jalur terbaik
27        if (awal == akhir)
28        {
29           if (Terbaik == null ||
30  Terbaik.WaktuTempuh > WaktuTempuh)
31           {
32              Terbaik = this;
33           }
34        return;
35        }
```

***Source Code* 4. 4** Penyimpanan Rute dalam TabuList

4.3.5 Proses *Update Pheromone*

Setelah semut telah mencapai tujuan perjalanan, selanjutnya dilakukan proses *update pheromone*, proses *update pheromone* dilakukan 2 kali, pertama ketika setiap kali semut melakukan perpindahan langkah dari titik r (sebagai titik awal) menuju ke titik s (sebagai titik selanjutnya) proses ini dinamakan *update pheromone* lokal. Proses *update pheromone* lokal dapat dilihat pada *sourcecode* 4. 5

```
1   . . . . . .
2   //update lokal
3   foreach (var item in titikYangMungkin.Keys)
4     {
5        nilai -= Math.Pow((1 / item.PanjangJalan),
6   Jalur.Beta) * Math.Pow(item.FeromonGlobal,
7   Alpha);
8
9        if (nilai <= 0)
10       {
11          item.FeromonLokal = (1 - Rho) * item.
12  FeromonGlobal + Rho * (1 /(item.PanjangJalan *
13  TotalTitik));
14          titikYangMungkin[item]++;
15          break;
16       }
17    }
18  . . . . . .
```

***Source Code* 4. 5** Proses *Update Pheromone* Lokal

Kedua adalah setelah semua semut telah menyelesaikan perjalanannya dalam satu generasi dilakukan *update pheromone global*. Untuk melakukan *update pheromone* global, dicari rute terbaik dari semua semut pada satu generasi yang akan dipakai sebagai *global best tour*. *Update pheromone* akan dilakukan pada *pheromone* yang merupakan elemen rute terbaik. Proses *update pheromone global* dapat dilihat pada *sourcecode* 4.6

```
1   . . . . . .
2   public static void UpdateGlobal()
3      {
4         foreach (var item in daftar)
5         {
6            item.FeromonGlobal = (1 - Jalur.Rho) *
```

```
item.FeromonGlobal;
        }
         var jarMin = Jalur.Terbaik.WaktuTempuh;
         foreach (var item in
Jalur.Terbaik.yangDilewati)
         {
         item.FeromonGlobal = (1 - Jalur.Rho) *
item.FeromonGlobal + Jalur.Rho / jarMin;
         }
    }
. . . . . .
```

***Source Code* 4. 6** Proses *Update Pheromone Global*

4.3.6 Perhitungan Panjang Jalur semut

Pada perhitungan panjang jalur semut ini digunakan untuk mengetahui panjang jalur perjalanan terbaik dari suatu koloni semut, dengan mengetahui panjang jalur perjalanan maka akan didapat rute dari setiap semut pada generasi tersebut. Dari rute hasil perjalanan semut dapat dihitung panjang jalur perjalanan semut. Perhitungan panjang jalur tertutup (*length closed tour*) atau L_k setiap semut dilakukan setelah satu siklus diselesaikan oleh semua semut.

```
. . . . . .
public double JarakTotal
{
   get
   {
      return tabuList.Sum(x =>x.PanjangJalan);
   }
}
Public static Jalur Terbaik;
public Jalur(Titik awal, Titik akhir,
List<Titik> titikTersedia, Jalur sebelumnya,
int jumlahSemut, bool Dijkstra)
   {
     target = akhir;
     NamaJalan = awal.NamaJalan;
     Jarak = awal.PanjangJalan;
     if (sebelumnya != null)
       {
          yangDilewati =
sebelumnya.yangDilewati.ToList();
        }
```

```
22          yangDilewati.Add(awal);
23          Lanjutan = new List<Jalur>();
24          if (awal == akhir)
25            {
26                if (Terbaik == null ||
27  Terbaik.WaktuTempuh > WaktuTempuh)
28                {
29                    Terbaik = this;
30                }
31                return;
32            }
33        }
34  . . . . . .
```

***Source Code* 4. 7** Proses Mendapatkan Jalur Terbaik

4.3.7 Penerapan Rumus Pitagoras

Fungsi dari rumus pitagoras yang digunakan pada aplikasi ini adalah untuk mencari titik terdekat ketika *mouse* di klik, diharapkan untuk mempermudah dalam menggunakan aplikasi serta mengurangi tingkat kesalahan oleh *user* saat menginputkan titik dari dan titik tujuan perjalanan.

```
1   . . . . . .
2   double jarak(Point t, Point p1, Point p2)
3       {
4       return FindDistanceToSegment(new
5   PointF(t.X, t.Y), new PointF(p1.X, p1.Y), new
6   PointF(p2.X, p2.Y));
7       }
8   private double FindDistanceToSegment(PointF
9   pt, PointF p1, PointF p2)
10      {
11          PointF closest;
12          float dx = p2.X - p1.X;
13          float dy = p2.Y - p1.Y;
14          if ((dx == 0) && (dy == 0))
15          {
16              // It's a point not a line segment.
17              closest = p1;
18              dx = pt.X - p1.X;
19              dy = pt.Y - p1.Y;
20              return Math.Sqrt(dx * dx + dy * dy);
21          }
```

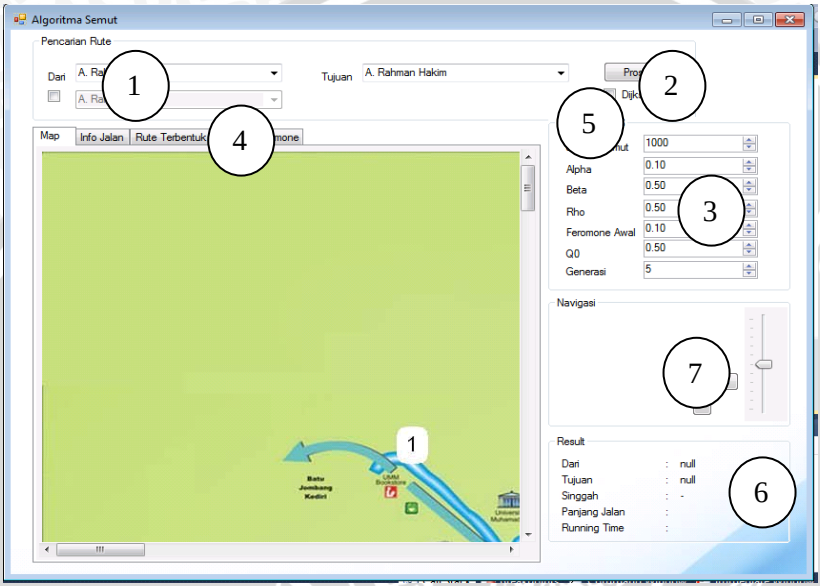
```
22          // Calculate the t that minimizes the
23      distance.
24          float t = ((pt.X - p1.X) * dx + (pt.Y -
25      p1.Y) * dy) / (dx * dx + dy * dy);
26          // See if this represents one of the
27      segment's
28          // end points or a point in the middle.
29          if (t < 0)
30              {
31                  closest = new PointF(p1.X, p1.Y);
32                  dx = pt.X - p1.X;
33                  dy = pt.Y - p1.Y;
34              }
35          else if (t > 1)
36          {
37              closest = new PointF(p2.X, p2.Y);
38              dx = pt.X - p2.X;
39              dy = pt.Y - p2.Y;
40          }
41      else
42          {
43              closest = new PointF(p1.X + t * dx,
44      p1.Y + t * dy);
45              dx = pt.X - closest.X;
46              dy = pt.Y - closest.Y;
47          }
48          return Math.Sqrt(dx * dx + dy * dy);
49      }
50      public double jarakTerdekat(int x, int y)
51      {
52          return Enumerable.Range(1,
53      DaftarTitik.Count - 1).Min(m => jarak(new
54      Point(x, y), DaftarTitik[m - 1],
55      DaftarTitik[m]));
56      }
57      . . . . . .
```

***Source Code* 4. 8** Fungsi Pitagoras

4.4 Implementasi Antarmuka Program Pencarian Rute Terpendek Bersepeda Menggunakan Algoritma Semut

Berdasarkan rancangan antarmuka aplikasi pada bab 3, maka dibuatlah antarmuka yang hanya terdiri dari 1 *form*, namun memiliki

beberapa tab untuk masing-masing fungsinya. Yaitu tab untuk Map, Rekomendasi Sistem dan Hasil *Update Pheromone*. Tampilan antarmuka utama dari aplikasi dapat dilihat pada gambar 4. 1



Gambar 4. 1 *Form* Utama Program

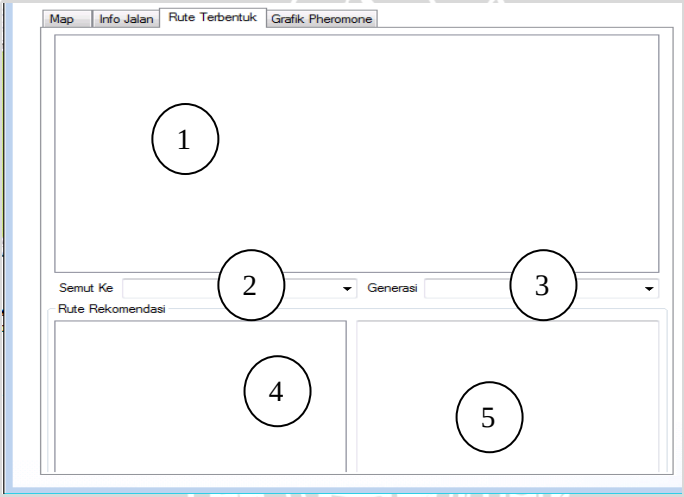
Aplikasi pada Gambar 4.1 terdiri dari beberapa komponen yang mempunyai fungsi sebagai berikut.

1. Parameter Rute yang berfungsi sebagai inputan perjalanan yang terdiri dari inputan lokasi awal dan inputan tujuan, serta titik singgah perjalanan.
2. *Button* Proses yang digunakan untuk memproses perhitungan pencarian rute baik menggunakan algoritma semut maupun algoritma *Dijkstra*.
3. Parameter ACS (*Ant Colony System*) yang berfungsi sebagai inputan dari *user* yang terdiri dari :
   - Jumlah semut
   - Alpha
   - Beta
   - Rho

- *Pheromone* awal
- Q0
- Generasi

4. Tab yang berfungsi untuk menampilkan hasil dari proses algoritma semut dan *Dijkstra*.
5. *Chekcbox* yang berfungsi untuk penyelesaian dengan algoritma *Dijkstra*.
6. Hasil pencarian sistem.
7. *Groupbox* Navigasi peta.

Gambar 4.2 menunjukkan tampilan aplikasi tab rekomendasi sistem sebagai tempat ditampilkannya hasil dari rekomendasi perjalanan oleh sistem dengan menggunakan algoritma semut

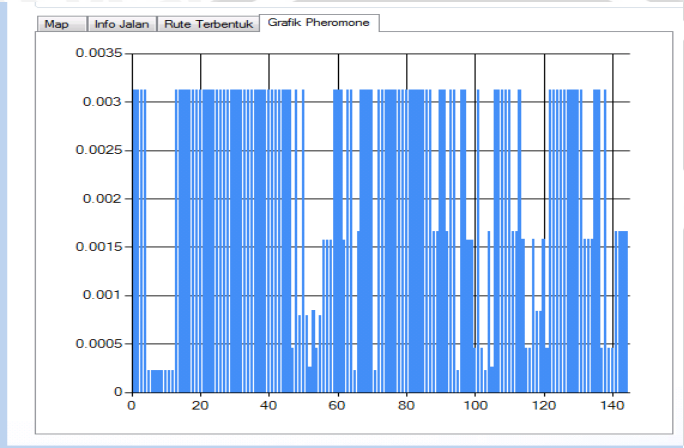


Gambar 4. 2 *Form tab* Rute Rekomendasi Sistem

Aplikasi terdiri dari beberapa komponen yang mempunyai fungsi sebagai berikut :

1. Informasi semua rute dari titik awal menuju titik akhir.
2. *Combobox* untuk menampilkan semut ke-x
3. *Combobox* untuk menampilkan generasi ke-x
4. *Listbox* hasil rute yang direkomendasikan oleh sistem
5. *Textbox* rute perjalanan semut ke-x

Gambar 4.3 menunjukkan tampilan aplikasi tab hasil *update pheromone* semut



Gambar 4. 3 *Form tab* Grafik *Pheromone* Semut

4.5 Implementasi Uji Coba Perangkat Lunak

Perangkat lunak hasil implementasi program dan implementasi antarmuka digunakan untuk pengujian seperti yang telah dijelaskan pada sub Bab 3.6. Nilai *default* pengujian perangkat lunak dapat dilihat pada tabel 3.20, pengujian meliputi evaluasi jalur tiap siklus dan evaluasi jalur pada beberapa kali *running* program dengan parameter algoritma semut yang berbeda.

4.5.1 Hasil dan Analisa Uji Coba Parameter Algoritma *Ant Colony System*

Pada uji coba yang pertama dilakukan pengujian pengaruh parameter semut untuk mendapatkan nilai parameter terbaik dari algoritma semut. Data yang digunakan pada pengujian pertama adalah data jalan kota malang. Uji coba ini dilakukan sebanyak 5 kali pada setiap perubahan nilai parameter. Dari uji coba tersebut akan diperoleh nilai terbaik yang kemudian akan di rata-rata sebagai nilai parameter yang optimal.

Program dijalankan sesuai dengan data dan parameter awal yang telah ditentukan. Uji coba dilakukan sebanyak 5 kali percobaan untuk masing-masing parameter yang akan diujikan.

4.5.1.1. Pengujian Parameter Jumlah Semut (m) dan *Pheromone* Awal (τ0)

Pengujian parameter semut (m) dan τ0 digunakan untuk mencari jumlah semut dan pheromone awal terbaik. Tabel 4.2, 4.3, 4.4, 4.5, dan 4.6 merupakan data hasil pengujian rata-rata dari 5 percobaan. Nilai yang dihasilkan masing-masing percobaan dapat dilihat pada lampiran 1.

Tabel 4. 2 Hasil Uji *Pheromone* Awal dengan 100 Semut

m	τ0	Jarak Rata-Rata
100	0.1	17.03
100	0.3	19.336
100	0.5	17.334
100	0.7	16.554
100	0.9	17.132

Tabel 4. 3 Hasil Uji *Pheromone* Awal dengan 200 Semut

m	τ0	Jarak Rata-Rata
200	0.1	17.158
200	0.3	15.4
200	0.5	14.282
200	0.7	14.556
200	0.9	14.244

Tabel 4. 4 Hasil Uji *Pheromone* Awal dengan 300 Semut

m	τ0	Jarak Rata-Rata
300	0.1	13.29
300	0.3	15.322
300	0.5	13.336
300	0.7	13.604
300	0.9	13.938

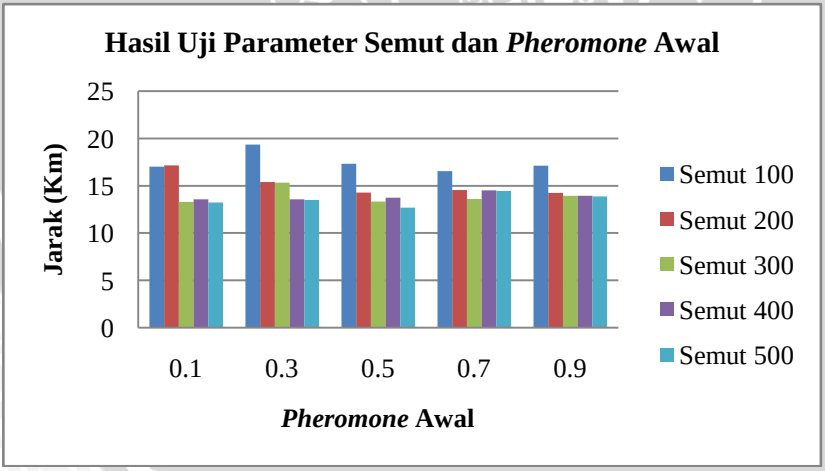
Tabel 4. 5 Hasil Uji *Pheromone* Awal dengan 400 Semut

m	τ0	Jarak Rata-Rata
400	0.1	13.578
400	0.3	13.55
400	0.5	13.75
400	0.7	14.524
400	0.9	13.924

Tabel 4. 6 Hasil Uji *Pheromone* Awal dengan 500 Semut

m	τ0	Jarak Rata-Rata
500	0.1	13.236
500	0.3	13.506
500	0.5	12.686
500	0.7	14.46
500	0.9	13.854

Berdasarkan data hasil pengujian parameter m dan τ0 pada tabel 4.2 sampai tabel 4.6, dapat dilihat bahwa parameter m dan τ0 yang menghasilkan solusi terbaik adalah 500 dan 0,5. Perbandingan nilai parameter dari masing-masing percobaan dapat dilihat pada gambar 4.4.



Gambar 4. 4 Grafik Pengaruh Perubahan Parameter Semut dan *Pheromone* Awal

Pengujian untuk mencari nilai m dan τ0 terbaik. Nilai m antara 100 sampai dengan 500 dan nilai τ0 antara 0,1 sampai dengan 0,9. Setelah 5 kali pengujian, didapatkan hasil nilai m terbaik adalah 500 seperti yang ditunjukkan pada gambar 4.4. Hal tersebut terjadi karena semakin banyak semut yang digunakan akan semakin baik hasil solusi yang diberikan sehingga rute yang dihasilkan akan lebih optimal. Penambahan jumlah semut sangat berpengaruh pada waktu proses sistem untuk melakukan optimasi. Semakin banyak semut, akan semakin sulit untuk mencapai kondisi stagnasi, sehingga jumlah iterasi meningkat untuk mencapai konvergensi. Nilai *pheromone* awal (τ0) terbaik yang menghasilkan solusi lebih optimal dengan jarak rute yang minimal adalah 0,3. Nilai parameter ini akan digunakan pada uji coba terhadap parameter lainnya untuk menghasilkan suatu parameter terbaik dari algoritma semut yang diimplementasikan pada perangkat lunak.

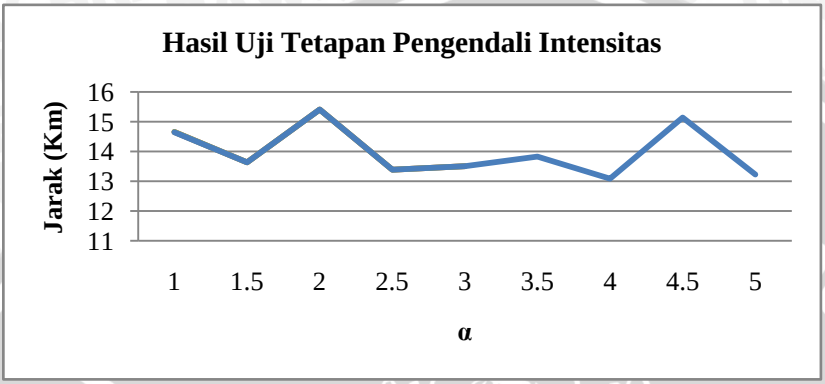
4.5.1.2. Pengujian Parameter Tetapan Pengendali Intensitas (α) dan Tetapan Pengendali Visibilitas (β)

Pengujian parameter α digunakan untuk mengetahui pengaruh tetapan pengendali intensitas jejak semut terhadap solusi yang dihasilkan. Tabel 4.7 merupakan data hasil pengujian parameter α menggunakan rata-rata dari 5 percobaan. Nilai yang dihasilkan masing-masing percobaan dapat dilihat pada lampiran 1.

Tabel 4. 7 Hasil Uji Tetapan Pengendali Intensitas (α)

α	β	ρ	m	nC	τ0	Q0	Jarak
1	1	0.5	500	5	0.5	0.5	14.652
1.5	1	0.5	500	5	0.5	0.5	13.636
2	1	0.5	500	5	0.5	0.5	15.41
2.5	1	0.5	500	5	0.5	0.5	13.388
3	1	0.5	500	5	0.5	0.5	13.508
3.5	1	0.5	500	5	0.5	0.5	13.83
4	1	0.5	500	5	0.5	0.5	13.088
4.5	1	0.5	500	5	0.5	0.5	15.142
5	1	0.5	500	5	0.5	0.5	13.228

Perbandingan nilai parameter dari masing-masing percobaan dapat dilihat pada gambar 4.5.



Gambar 4. 5 Grafik Uji Coba Tetapan Pengendali Intensitas (α)

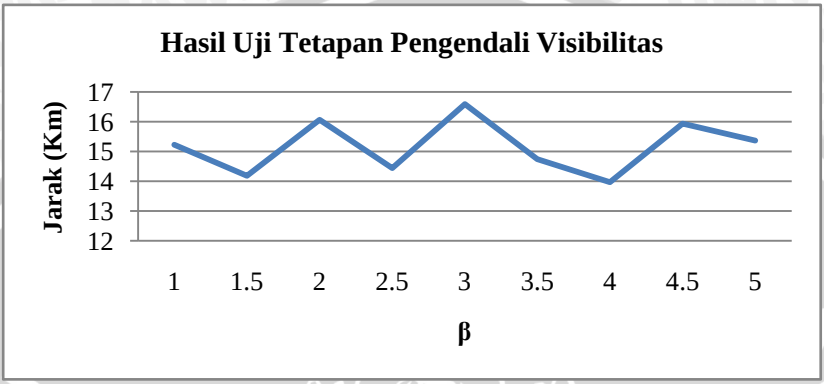
Pengujian untuk mencari nilai α yang terbaik dari nilai 1 sampai dengan 5. Setelah 5 kali pengujian, didapatkan hasil nilai β terbaik adalah 4 seperti yang ditunjukkan pada gambar 4.5. Sehingga nilai m, τ0, dan α yang sudah didapatkan akan digunakan sebagai parameter uji untuk parameter lainnya.

Pengujian parameter β digunakan untuk mengetahui pengaruh tetapan pengendali intensitas jejak semut terhadap solusi yang dihasilkan. Tabel 4.8 merupakan data hasil pengujian parameter β menggunakan rata-rata dari 5 percobaan. Nilai yang dihasilkan masing-masing percobaan dapat dilihat pada lampiran 1.

Tabel 4. 8 Hasil Uji Tetapan Pengendali Visibilitas (β)

α	β	ρ	m	nC	τ0	Q0	Jarak
4	1	0.5	500	5	0.5	0.5	15.222
4	1.5	0.5	500	5	0.5	0.5	14.19
4	2	0.5	500	5	0.5	0.5	16.062
4	2.5	0.5	500	5	0.5	0.5	14.448
4	3	0.5	500	5	0.5	0.5	16.588
4	3.5	0.5	500	5	0.5	0.5	14.748
4	4	0.5	500	5	0.5	0.5	13.972
4	4.5	0.5	500	5	0.5	0.5	15.936
4	5	0.5	500	5	0.5	0.5	15.37

Perbandingan nilai parameter dari masing-masing percobaan dapat dilihat pada gambar 4.6.



Gambar 4. 6 Grafik Uji Coba Tetapan Pengendali Visibilitas (β)

Pengujian untuk mencari nilai β yang terbaik dari nilai 1 sampai dengan 5. Setelah 5 kali pengujian, didapatkan hasil nilai β terbaik adalah 4 seperti yang ditunjukkan pada gambar 4.6. Sehingga nilai m, τ0, α, dan β yang sudah didapatkan akan digunakan sebagai parameter uji untuk parameter lainnya.

Pengujian dilakukan dengan mengubah nilai alpha sedangkan nilai beta tetap bernilai *default*. Nilai alpha yang digunakan bervariasi mulai dari 1 sampai dengan 5 untuk mendapatkan nilai alpha yang terbaik. Kemudian pengujian dilanjutkan terhadap nilai beta dengan cara yang sama dilakukan terhadap nilai alpha untuk mendapatkan nilai beta yang terbaik. Sehingga didapatkan komposisi terbaik untuk parameter alpha (α) dan beta (β) adalah 4 dan 4 seperti ditunjukkan pada gambar 4.5 dan 4.6

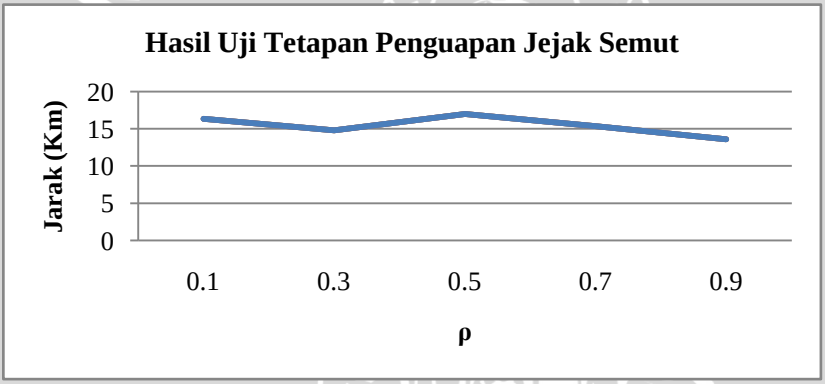
4.5.1.3. Pengujian Parameter Tetapan Penguapan Jejak Semut (ρ)

Pengujian parameter ρ digunakan untuk mengetahui pengaruh tetapan penguapan jejak semut terhadap solusi yang dihasilkan. Tabel 4.9 merupakan data hasil pengujian parameter ρ menggunakan rata-rata dari 5 percobaan. Nilai yang dihasilkan masing-masing percobaan dapat dilihat pada lampiran.

Tabel 4. 9 Hasil Uji Tetapan Penguapan Jejak Semut

α	β	ρ	m	nC	τ0	Q0	Jarak
4	4	0.1	500	5	0.3	0.5	16.326
4	4	0.3	500	5	0.3	0.5	14.776
4	4	0.5	500	5	0.3	0.5	16.986
4	4	0.7	500	5	0.3	0.5	15.332
4	4	0.9	500	5	0.3	0.5	13.592

Berdasarkan tabel 4.9, dapat dilihat bahwa nilai ρ terbaik yang menghasilkan jarak minimal adalah 0,9. Sehingga nilai m, τ0, α, β, dan ρ yang sudah didapatkan akan digunakan sebagai parameter uji untuk parameter lainnya. Perbandingan nilai parameter dari masing-masing percobaan dapat dilihat pada gambar 4.7.



Gambar 4. 7 Grafik Uji Tetapan Penguapan Jejak Semut (ρ)

Pengujian untuk mencari nilai rho yang terbaik dari nilai 0.1 sampai dengan 0.9. Setelah 5 kali pengujian, didapatkan hasil nilai rho terbaik adalah 0.9 seperti yang ditunjukkan pada gambar 4.7

Setelah dilakukan beberapa pengujian terhadap perangkat lunak yang telah dibuat, dapat dilihat pada gambar 4.5 dan 4.6 bahwa parameter-parameter yang digunakan saling berhubungan satu dengan lainnya.

Nilai paramater α dan β mempengaruhi nilai Φ , dimana Φ merupakan probabilitas dari titik i ke titik j. Semakin besar nilai keduanya, semakin besar pula probabilitas dari titik yang sekarang ke titik berikutnya. Ini berarti nilai paramater α dan β berbanding lurus

dengan nilai Φ . Jika salah satu parameter yang digunakan mendekati nol berarti hanya mengandalkan *pheromone* saja atau visibilitas.
Sedangkan pada gambar 4.7, Nilai parameter ρ akan mempengaruhi nilai τ0 , dimana τ0 merupakan nilai *pheromone* awal. Setiap iterasi yang dilakukan menyebabkan perubahan nilai pada *pheromone* awal tersebut. Jadi pada setiap iterasi diadakan pembaharuan nilai *pheromone* tersebut. Semakin besar nilai ρ akan memperkecil nilai τ0, sedangkan semakin kecil nilai ρ akan memperbesar nilai τ0 . Ini berarti nilai ρ berbanding terbalik dengan nilai τ0.

4.5.1.4. Pengujian Parameter Q0

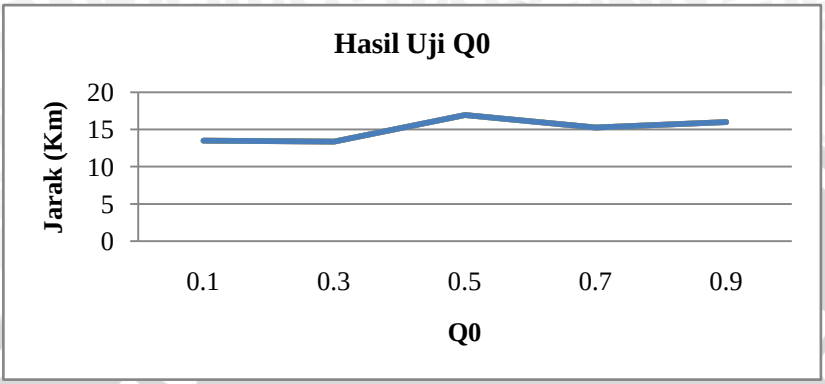
Pengujian parameter Q0 digunakan untuk mengetahui pengaruh tetapan penguapan jejak semut terhadap solusi yang dihasilkan. Tabel 4.10 merupakan data hasil pengujian parameter ρ menggunakan rata-rata dari 5 percobaan. Nilai yang dihasilkan masing-masing percobaan dapat dilihat pada lampiran.

Tabel 4. 10 Hasil Uji Q0

α	β	ρ	m	nC	τ0	Q0	Jarak
4	4	0.1	500	5	0.5	0.1	13.504
4	4	0.3	500	5	0.5	0.3	13.362
4	4	0.5	500	5	0.5	0.5	16.95
4	4	0.7	500	5	0.5	0.7	15.27
4	4	0.9	500	5	0.5	0.9	15.996

Berdasarkan tabel 4.10, dapat dilihat bahwa nilai Q0 terbaik yang menghasilkan jarak minimal adalah 0,3. Perbandingan nilai parameter dari masing-masing percobaan dapat dilihat pada gambar 4.8.

.



Gambar 4. 8 Grafik Uji Q0

Pada gambar 4.8 menunjukkan pengaruh aturan transisi status yang berlaku pada ACS pada selang $0.1 \leq Q \leq 0.9$. Parameter ini merupakan konstanta yang digunakan untuk pemilihan titik-titik yang akan dilalui oleh semut dengan membandingkan nilai q yang didapatkan dengan random terhadap nilai Q0. jika $q \leq Q0$ pemilihan rute dengan cara eksploitasi sedangkan jika $1 > Q0$ maka pemilihan rute dilakukan dengan cara eksplorasi. Nilai konstanta Q0 yang terbaik adalah 0.3.

4.5.1.5. Pengujian Generasi

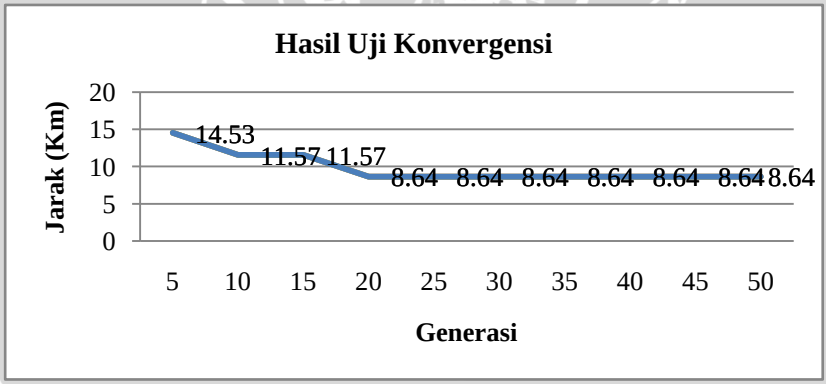
Pengujian dengan beberapa generasi dilakukan untuk menghasilkan solusi yang konvergen. Parameter yang digunakan untuk menguji konvergensi algoritma semut adalah m = 500, $\tau 0$ = 0.5, α = 4, β = 4, ρ = 0.9, dan Q0 = 0.3. Hasil dari pengujian ini digunakan untuk mencari siklus awal terjadinya kestabilan nilai / jarak dengan pengecekan 20 iterasi terakhir yang menghasilkan jarak yang sama.

Jarak yang dihasilkan pada saat terjadi konvergensi merupakan solusi yang relatif optimal. Tabel 4.11 merupakan data hasil pengujian algoritma semut dengan generasi antara 5 sampai dengan 50 generasi.

Tabel 4. 11 Hasil Uji Konvergensi

Generasi	Jarak
5	14.53
10	11.57
15	11.57
20	8.64
25	8.64
30	8.64
35	8.64
40	8.64
45	8.64
50	8.64

Berdasarkan hasil uji coba pada tabel 4.11,dapat diketahui bahwa pada generasi ke 20 sampai 50 memiliki nilai yang sama. Dapat dikatakan bahwa pada generasi ke 20 telah didapatkan hasil yang konvergen. Gambar 4.9 merupakan representasi grafik dari pengujian generasi



Gambar 4. 9 Grafik Uji Konvergensi

Gambar 4.9 menunjukkan grafik jarak(Km) yang dihasilkan oleh algoritma *ant colony system* dengan beberapa generasi. Pada gambar 4.9 dapat dilihat bahwa terjadinya konvergensi pada generasi ke- 20 yang menghasilkan jarak sebesar 8.64Km. Nilai tersebut

merupakan solusi yang relatif optimal yang dihasilkan oleh algoritma *ant colony system*. Untuk mengetahui tingkat keoptimalan algoritma *ant colony system*, maka akan dilakukan perbandingan antara algoritma *ant colony system* dengan algoritma *Dijkstra.*

4.5.2 Uji coba Tingkat Optimasi Algoritma *Ant Colony System*

Untuk mengetahui tingkat keoptimalan algoritma *ant colony system*, dilakukan perbandingan antara algoritma *ant colony system* dengan algoritma *Dijkstra*. Dalam melakukan perbandingan ini, parameter yang digunakan algoritma *ant colony system* adalah nCmax=20, m=500, $\tau 0=0.5$, $\alpha=4$, $\beta=4$, $\rho=0.9$, dan $Q0=0.3$. Tabel 4.12 merupakan tabel yang menunjukkan jarak yang dihasilkan oleh kedua algoritma pada 5 kali percobaan yang dapat dilihat pada lampiran.

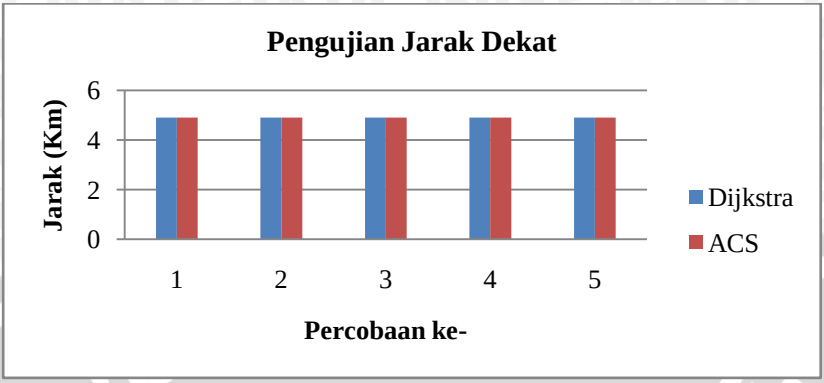
4.5.2.1. Pengujian Jarak Dekat

Pengujian jarak dekat ini dilakukan dari jl. L.A Sucipto 2 menuju ke jl. Hamid Rusdi. Hasil dari pengujian jarak dekat ini dapat dilihat pada tabel 4.12

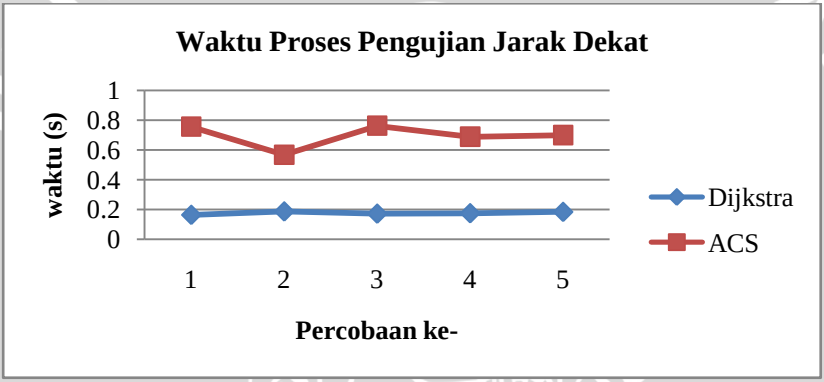
Tabel 4. 12 Hasil Pengujian Jarak Dekat

Algoritma	Percobaan Ke-					Rata-Rata
	1	2	3	4	5	
ACS	6	6	6	6	6	6
Dijkstra	6	6	6	6	6	6

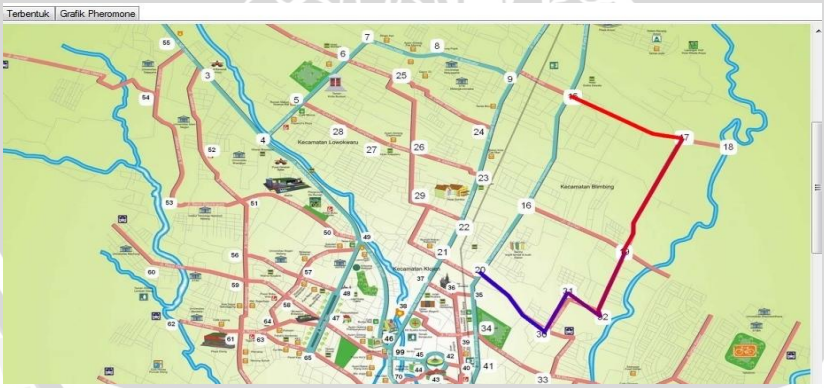
Dari hasil pengujian yang ditunjukkan pada gambar 4.10, jarak yang dihasilkan oleh ACS dan *Dijkstra* adalah sama bernilai 6Km. ACS memerlukan waktu antara 0,5 sampai 0,8 detik untuk menghasilkan rute terpendek, sedangakan pada *Dijkstra* waktu yang dibutuhkan antara 0,1 sampai 0,2 detik. Rata-rata waktu untuk ACS adalah 0,694878 detik dan *Dijkstra* 0,175409 detik. Gambar 4.11 merupakan grafik waktu yang diperlukan oleh masing-masing algoritma dalam menghasilkan rute optimal. Rute yang terbentuk dari ACS dan *Dijkstra* adalah sama, rute ditunjukkan pada gambar 4.12



Gambar 4. 10 Pengujian Untuk Jarak Dekat



Gambar 4. 11 *Running Time* untuk Pengujian Jarak Dekat



Gambar 4. 12 Rute yang dihasilkan oleh ACS dan *Dijkstra* pada jarak dekat

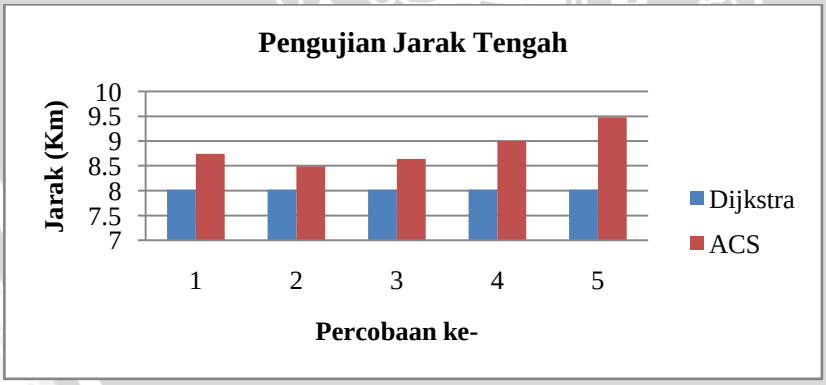
4.5.2.2. Pengujian Jarak Menengah

Pengujian jarak Menengah ini dilakukan dari jl. A.Rahman Hakim menuju ke jl. Balearjosari. Hasil dari pengujian jarak menengah ini dapat dilihat pada tabel 4.13

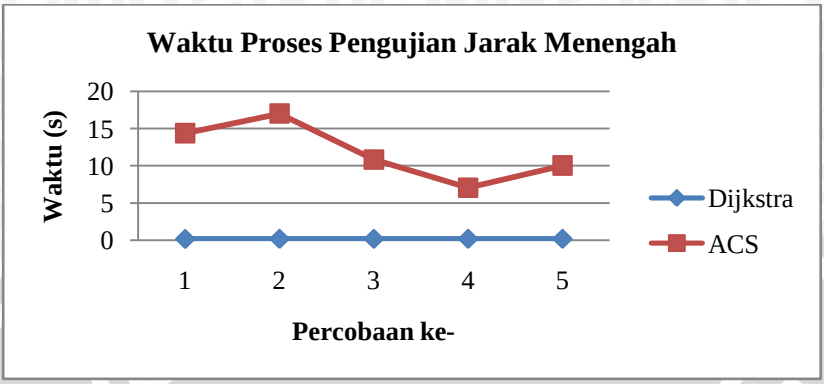
Tabel 4. 13 Hasil Pengujian Jarak Menengah

Algoritma	Percobaan Ke-					Rata-Rata
	1	2	3	4	5	
ACS	8.74	8.49	8.64	9	9.48	8.87
Dijkstra	8.02	8.02	8.02	8.02	8.02	8.02

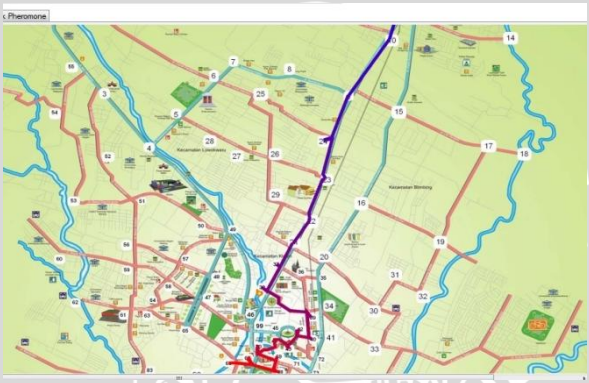
Pada gambar 4.15 dan 4.16 dapat dilihat rute terbentuk dari ACS dan *Dijkstra*, dimana rute yang terbentuk antara kedua algoritma berbeda. Dari hasil pengujian yang ditunjukkan pada gambar 4.13, rute optimal yang dihasilkan oleh ACS adalah 8.49Km dan *Dijkstra* adalah 8.02Km. ACS memerlukan waktu antara 7 sampai 17 detik untuk menghasilkan rute terpendek, sedangakan pada *Dijkstra* waktu yang dibutuhkan antara 0,1 sampai 0,2 detik. Rata-rata waktu untuk ACS adalah 11,86734 detik dan *Dijkstra* 0.17161 detik. Gambar 4.14 merupakan grafik waktu yang diperlukan oleh masing-masing algoritma dalam menghasilkan rute optimal



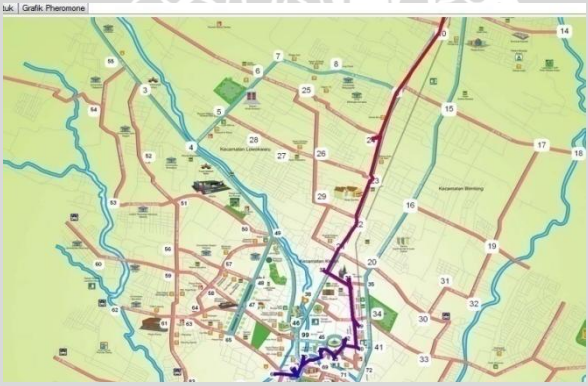
Gambar 4. 13 Pengujian Untuk Jarak Menengah



Gambar 4. 14 *Running Time* untuk Pengujian Jarak Menengah



Gambar 4. 15 Rute yang dihasilkan oleh ACS pada jarak menengah



Gambar 4. 16 Rute yang dihasilkan oleh *Dijkstra* pada jarak menengah

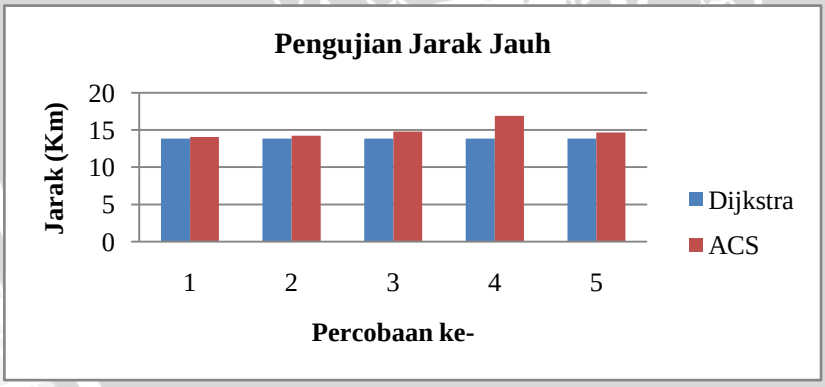
4.5.2.3. Pengujian Jarak Jauh

Pengujian jarak jauh ini dilakukan dari jl. Raya Tlogomas menuju ke jl. Kol. Sugiono. Hasil dari pengujian jarak jauh ini dapat dilihat pada tabel 4.14

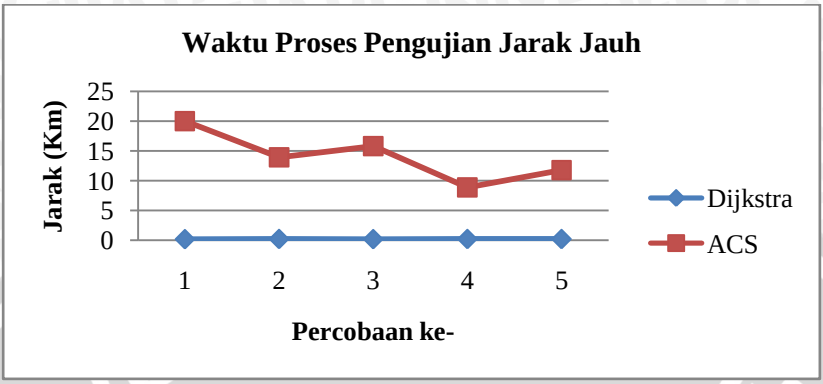
Tabel 4. 14 Hasil Pengujian Jarak Jauh

Algoritma	Percobaan Ke-					Rata-Rata
	1	2	3	4	5	
ACS	14.07	14.23	14.79	16.92	14.68	14.938
Dijkstra	13.86	13.86	13.86	13.86	13.86	13.86

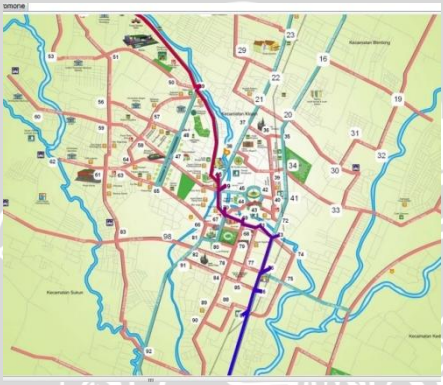
Pada gambar 4.19 dan 4.20 dapat dilihat rute terbentuk dari ACS dan *Dijkstra*, dimana rute yang terbentuk antara kedua algoritma berbeda. Dari hasil pengujian yang ditunjukkan pada gambar 4.8, rute optimal yang dihasilkan oleh ACS adalah 14.07Km dan *Dijkstra* adalah 13.86Km. ACS memerlukan waktu antara 8 sampai 19 detik untuk menghasilkan rute terpendek, sedangakan pada *Dijkstra* waktu yang dibutuhkan antara 0,1 sampai 0,2 detik. Rata-rata waktu untuk ACS adalah 14.05698 detik dan *Dijkstra* 0.177447 detik. Gambar 4.9 merupakan grafik waktu yang diperlukan oleh masing-masing algoritma dalam menghasilkan rute optimal.



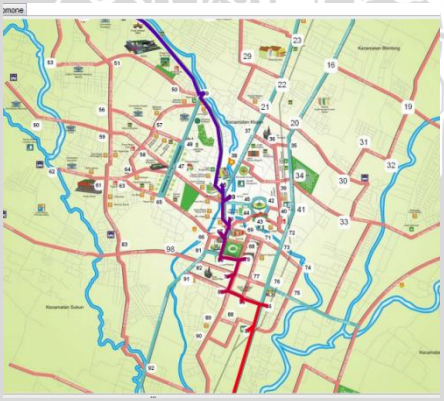
Gambar 4. 17 Pengujian Untuk Jarak Jauh



Gambar 4. 18 *Running Time* untuk Pengujian Jarak Jauh



Gambar 4. 19 Rute yang dihasilkan oleh ACS pada jarak jauh



Gambar 4. 20 Rute yang dihasilkan oleh *Dijkstra* pada jarak jauh

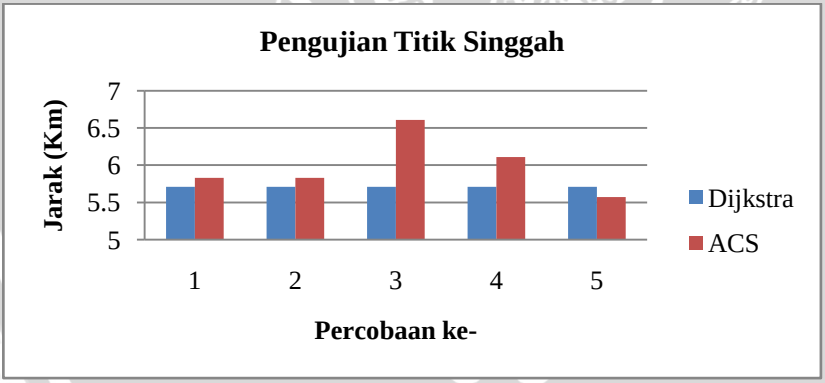
4.5.2.4. Pengujian Menggunakan Titik Singgah

Pengujian menggunakan titik singgah ini dilakukan dari jl. Zainul zakse menuju ke jl. Ijen 1 singgah ke jl. Ijen 3. Hasil dari pengujian ini dapat dilihat pada tabel 4.15

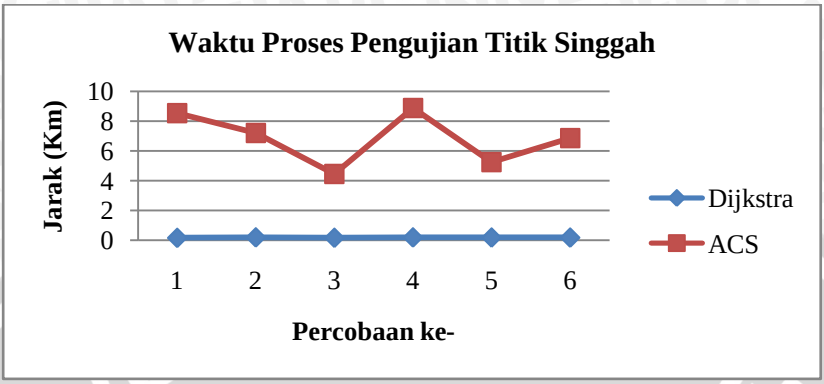
Tabel 4. 15 Hasil Pengujian Menggunakan Titik Singgah

Algoritma	Percobaan Ke-					Rata-Rata
	1	2	3	4	5	
ACS	5.83	5.83	6.61	6.11	5.57	5.99
Dijkstra	5.71	5.71	5.71	5.71	5.71	5.71

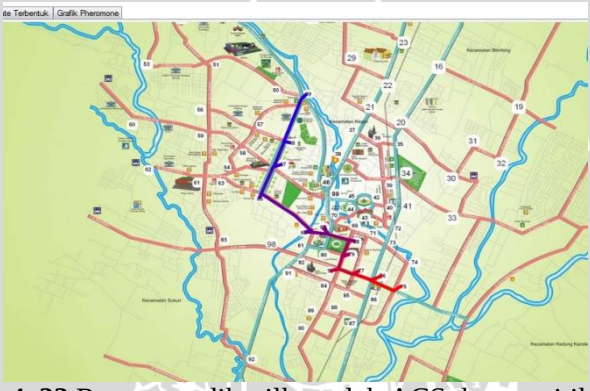
Dari hasil pengujian yang ditunjukkan pada gambar 4.21, rute optimal yang dihasilkan oleh ACS adalah pada percobaan ke-5 yaitu 5.57Km dan *Dijkstra* adalah 5.71Km. ACS memerlukan waktu antara 4 sampai 8 detik untuk menghasilkan rute terpendek, sedangakan pada *Dijkstra* waktu yang dibutuhkan antara 0,1 sampai 0,2 detik. Rata-rata waktu untuk ACS adalah 13.8253 detik dan *Dijkstra* 0.178247 detik. Gambar 4.22 merupakan grafik waktu yang diperlukan oleh masing-masing algoritma dalam menghasilkan rute optimal. Rute terbentuk dapat dilihat pada gambar 4.23 dan 4.24



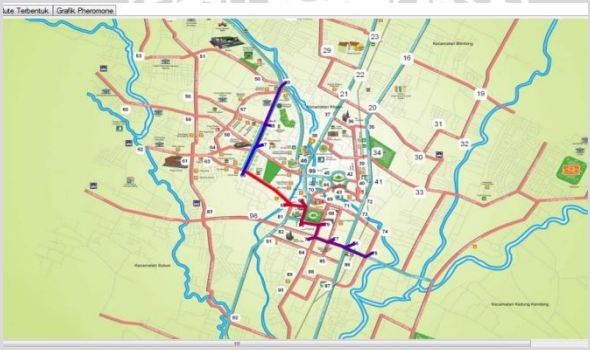
Gambar 4. 21 Pengujian Menggunakan Titik Singgah



Gambar 4. 22 *Running Time* untuk Pengujian Menggunakan Titik Singgah



Gambar 4. 23 Rute yang dihasilkan oleh ACS dengan titik singgah



Gambar 4. 24 Rute yang dihasilkan oleh *Dijkstra* dengan titik singgah

Dari beberapa percobaan, algoritma *Dijkstra* selalu mendapat rute yang optimal untuk semua data uji karena algoritma hanya memikirkan solusi terbaik dengan langsung mengambil jarak terpendek pada saat itu tanpa memikirkan jarak total yang dihasilkan. Sedangkan algoritma semut, untuk pengujian jarak dekat, nilai yang dihasilkan selalu optimal. Sedangkan pengujian jarak menengah atau jarak jauh, nilai yang dihasilkan mendekati optimal. Hal ini disebabkan semakin banyak data, maka semakin besar ruang lingkup pencarian sehingga kemungkinan algoritma semut untuk terjebak atau tidak dapat menemukan tujuan dalam menyelesaikan permasalahan. Sesuai dengan persamaan 2.12, maka nilai MSE dapat dihitung sebagai berikut.

$$MSE = \frac{\sum_{i=1}^{n}(D_{ig} - D_{ik})^2}{n}$$

$$MSE = \frac{(6-6)^2 + (8.87-8.02)^2 + (14.938-13.86)^2 + (5.99-5.71)^2}{4}$$

$$MSE = \frac{0 + 0.7225 + 1.162084 + 0.0784}{4}$$

$$MSE = 0.490746$$

Dari hasil nilai MSE tersebut membuktikan bahwa algoritma semut memiliki tingkat akurasi yang cukup tinggi untuk penyelesaian masalah. Semakin kecil nilai MSE, maka semakin tinggi tingkat akurasi algoritma semut dalam penyelesaian pencarian rute terpendek.

BAB V
KESIMPULAN DAN SARAN

5.1 Kesimpulan

Berdasarkan penelitian dengan melakukan uji coba dan evaluasi, maka kesimpulan yang diperoleh adalah sebagai berikut :

1. Pencarian rute terpendek dengan metode *Ant Colony System* dapat diimplementasikan. Proses pencarian rute terpendek di mulai dengan menyebarkan semut ke suatu titik, kemudian semut akan berjalan bebas menuju tujuan perjalanan. Dalam pemilihan jalur terpendek, metode ini tergantung dari nilai parameter yang dimasukkan antara lain jumlah semut, $\tau 0$, α, β, ρ, dan Q0.
2. Perbandingan jarak terpendek yang dihasilkan algoritma *Ant Colony System* dengan *Dijkstra* menghasilkan jarak yang *relative* sama. Hanya saja rute yang dihasilkan oleh algoritma *Ant Colony System* lebih bervariatif, karena dipengaruhi oleh beberapa parameter. Nilai parameter terbaik yang diperoleh dari penelitian adalah jumlah semut = 500, $\tau 0 = 0.5$, $\alpha = 4$, $\beta = 4$, $\rho = 0.9$, dan $Q0 = 0.3$. Dari beberapa percobaan diperoleh nilai MSE sebesar 0.490746, yang berarti algoritma *Ant Colony System* memiliki tingkat akurasi yang cukup tinggi.

5.2 Saran

Untuk pengembangan lebih lanjut, saran yang dapat diberikan untuk penelitian lebih lanjut adalah :

1. Aplikasi ini dapat dikembangkan untuk menyelesaikan permasalahan pencarian rute perjalanan terpendek dengan menambahkan jalan kecil (gang) yang dapat mempengaruhi rute yang akan dipilih dalam perjalanan.
2. Dapat dikembangkan dengan menggunakan peta digital agar mempermudah dalam pengaksesan aplikasi.

Daftar Pustaka

Dorigo, M., & Gambardella, L. M. (1997). *Ant colonies for the traveling salesman problem.* Belgium: Tech.Rep/IRIDIA/1996-003, Université Libre de Bruxelles.

Dorigo, M., & Stützle, T. (2004). *Ant Colony Optimization.* England: The MIT Press Cambridge, Massachusetts Institute of Technology.

Handaka, M. S. (2010). *Perbandingan Algoritma Dijkstra (Greedy), Bellman-Ford(BFS-DFS), dan Floyd-Warshall (Dynamic Programming) dalam Pengaplikasian Lintasan Terpendek pada Link-State Routing Protocol.* Bandung: Institut Teknologi Bandung.

M. Dorigo, V. Maniezzo, dan A. Colorni. (1996). The Ant System: Optimization by a colony of cooperating agents. *IEEE Transactions on Systems, Man,and Cybernetics—Part B, 26(1),* , pp.1-13.

Munir, R. (2005). *Matematika Diskrit Edisi Ketiga.* Bandung: Informatika.

Mutakhiroh, I., Saptono, F., Hasanah, N., & Wiryadinata, R. (2007). *Pemanfaatan Metode Heuristik Dalam Pencarian Jalur Terpendek Dengan Algoritma Semut dan Algoritma Genetik.* Yogyakarta: Seminar Nasional Aplikasi Teknologi Informasi. ISSN: 1907-5022.

Novandi, R. A. (2007). *Perbandingan Algoritma Dijkstra dan Algoritma Floyd-Warshall dalam Penentuan Lintasan Terpendek (Single Pair Shortest Path).* Bandung: Institut Teknologi Bandung.

Nugraha, D., & dkk. (2006). *Diagnosis Gangguan Sistem Urinari pada Anjing dan Kucing menggunakan VFI 5.* Bandung: IPB.

Surya Post, 30 November 2011

Wardy, I. S. (2007). *Penggunaan graph dalam algoritma semut untuk melakukan optimasi.* Bandung : Institut Teknologi Bandung.

Anonymous. Bersepeda. http://kamusbahasaindonesia.org/Bersepeda diakses tanggal 9 Desember 2011.

Bike2Work Indonesia. 60 Manfaat Bersepeda. http://b2w-indonesia.or.id/bacanote/60_manfaat_bersepeda diakses tanggal 30 November 2011

Lampiran 1

Data Hasil Pengujian Parameter Algoritma *Ant Colony System*

1. Data pengujian parameter jumlah semut dan *pheromone* awal dari jalan AR. Hakim menuju ke jalan Bale Arjosari

semut		jarak				
	T0	1	2	3	4	5
100	0.1	16.9	19.7	15.98	16.78	15.79
100	0.3	18.6	16.81	19.05	17.39	24.83
100	0.5	20.29	19.41	18.8	14.44	13.73
100	0.7	13.76	12.34	17.79	21.26	17.62
100	0.9	13.98	15.95	16.76	22.15	16.82
200	0.1	14.82	14.61	20.99	17.84	17.53
200	0.3	18.03	17.4	13.56	13.47	14.54
200	0.5	11	15.23	20.55	12.5	12.13
200	0.7	15.36	13.79	15.12	13.02	15.49
200	0.9	15.89	10.27	21.03	14.55	9.48
300	0.1	12.56	12	14.81	12.93	14.15
300	0.3	21.33	15.38	14.52	13.19	12.19
300	0.5	14.85	13.85	13.73	12.3	11.95
300	0.7	11.53	12.88	15.08	13.71	14.82
300	0.9	14.52	13.03	15.38	13.63	13.13
400	0.1	12.88	11.73	15.92	15.02	12.34
400	0.3	13.73	11.31	16.81	14.36	11.54
400	0.5	14.27	14.69	13.31	12.74	13.74
400	0.7	15.48	11.54	15.21	14.14	16.25
400	0.9	13.73	13.46	15.98	14.57	11.88
500	0.1	14.66	12.18	15.24	14.27	9.83
500	0.3	14.27	15.39	10.41	13.73	13.73
500	0.5	13.81	14.27	12.19	13.73	9.43
500	0.7	15.75	12.2	17.71	14.16	12.48
500	0.9	12.55	16.27	14.56	14.35	11.54

2. Data pengujian parameter α dengan menggunakan m=500, τ0=0.5, ρ=0.5, Q0=0.5, dan nCmax=5

α	β	Jarak				
		1	2	3	4	5
1	1	18.58	15.63	11.59	13.73	13.73
1.5	1	15.29	10.27	14.31	16.41	11.9
2	1	16.03	14.22	15.53	14.88	16.39
2.5	1	13.73	13.97	13.03	10.92	15.29
3	1	12.26	13.73	14.39	12.34	14.82
3.5	1	13.87	13.15	13.96	15.38	12.79
4	1	12.45	13.75	14.32	13.18	11.74
4.5	1	14.27	14.27	19.71	13.73	13.73
5	1	11.51	15.22	11.22	10.92	17.27

3. Data pengujian parameter β dengan menggunakan m=500, τ0=0.5, ρ=0.5, Q0=0.3, dan nCmax=5

α	β	Jarak				
		1	2	3	4	5
4	1	16.97	12.45	18.07	12.2	16.42
4	1.5	11.07	15.41	14.41	14.27	15.79
4	2	15.64	15.57	14.7	16.03	18.37
4	2.5	17.46	16.17	13.92	12.49	12.2
4	3	14.67	18.25	19.38	16.52	14.12
4	3.5	14.15	13.3	16.22	12.76	17.31
4	4	11.13	14.12	15.9	12.19	16.52
4	4.5	14.27	14.22	18.67	16.05	16.47
4	5	17.62	18.64	16.26	11.73	12.6

4. Data pengujian parameter ρ dengan menggunakan m=500, τ0=0.5, α=4, β=4, Q0=0.3, dan nCmax=5

ρ	Jarak				
	1	2	3	4	5
0.1	14.28	13.16	17.56	14.58	22.05
0.3	14.49	15.12	12.83	17.62	13.82
0.5	20.56	14.27	16.41	19.3	14.39
0.7	15.36	17.63	13.06	13.27	17.34
0.9	12.2	13.06	12.87	14.47	15.36

5. Data pengujian parameter ρ dengan menggunakan m=500, τ0=0.5, α=4, β=4, ρ=0.3, dan nCmax=5

Q0	Jarak				
	1	2	3	4	5
0.1	14.21	12.93	14.98	13.92	11.48
0.3	11.1	16.04	11.5	12.9	15.27
0.5	16.77	14.95	14.16	19.02	19.85
0.7	14.67	16.83	14.27	14.22	16.36
0.9	13.39	16.36	15.5	18.37	16.36

Lampiran 2

Pengujian Algoritma Semut dan *Dijkstra*

1. Pengujian Jarak Dekat

Algoritma	Percobaan Ke-				
	1	2	3	4	5
Dijkstra	4.9	4.9	4.9	4.9	4.9
ACS	4.9	4.9	4.9	4.9	4.9

2. Pengujian Jarak Menengah

Algoritma	Percobaan Ke-				
	1	2	3	4	5
Dijkstra	8.02	8.02	8.02	8.02	8.02
ACS	8.74	8.49	8.64	9	9.48

3. Pengujian Jarak Jauh

Algoritma	Percobaan Ke-				
	1	2	3	4	5
Dijkstra	13.86	13.86	13.86	13.86	13.86
ACS	22.12	21.1	22.96	21.67	22.65