PENCARIAN RUTE ANGKUTAN UMUM MENGGUNAKAN ALGORITMA ANT COLONY OPTIMIZATION

SKRIPSI

Untuk memenuhi sebagian persyaratan mencapai gelar sarjana komputer

Disusun Oleh :
CANDRA IRWANSYAH
105060807111146

PROGRAM STUDI TEKNIK INFORMATIKA
PROGRAM TEKNOLOGI INFORMASI DAN ILMU KOMPUTER
UNIVERSITAS BRAWIJAYA
2014

LEMBAR PERSETUJUAN

PENCARIAN RUTE ANGKUTAN UMUM MENGGUNAKAN ALGORITMA ANT COLONY OPTIMIZATION

SKRIPSI

LABORATORIUM
KOMPUTASI CERDAS & VISUALISASI

Untuk memenuhi sebagian persyaratan mencapai gelar Sarjana Komputer

Disusun Oleh:
CANDRA IRWANSYAH
NIM.105060807111146

Skripsi ini telah disetujui oleh dosen pembimbing pada tanggal 11 April 2014

Dosen Pembimbing I	Dosen Pembimbing II
Aryo Pinandito, S.T., M.MT.	**Wayan Firdaus Mahmudy, S.Si., MT. Phd.**
NIK. 83051916110374	**NIP. 197209191997021001**

LEMBAR PENGESAHAN

PENCARIAN RUTE ANGKUTAN UMUM MENGGUNAKAN ALGORITMA ANT COLONY OPTIMIZATION

SKRIPSI

LABORATORIUM
KOMPUTASI CERDAS & VISUALISASI

Untuk memenuhi sebagian persyaratan mencapai gelar Sarjana Komputer

Disusun Oleh :

CANDRA IRWANSYAH
NIM.105060807111146

Skripsi ini telah diuji dan dinyatakan lulus pada tanggal 08 Mei 2014

Penguji I	**Penguji II**
Muhammad Tanzil Furqon, S.Kom., MCompSc.	**Dian Eka Ratnawati, S.Si., M.Kom.**
NIP. 19820930 200801 1 004	**NIP. 19730619 200212 2 001**

Penguji III

Wijaya Kurniawan, ST., MT.
NIK. 820125 16 1 1 0418

Mengetahui
Ketua Program Studi Informatika/Ilmu Komputer

Drs. Marji, M.T.
NIP. 19670801 199203 1 001

PERNYATAAN
ORISINALITAS SKRIPSI

Saya menyatakan dengan sebenar-benarnya bahwa sepanjang pengetahuan saya, di dalam naskah SKRIPSI ini tidak terdapat karya ilmiah yang pernah diajukan oleh orang lain untuk memperoleh gelar akademik di suatu perguruan tinggi dan tidak terdapat karya atau pendapat yang pernah ditulis atau diterbitkan oleh orang lain, kecuali yang secara tertulis dikutip dalam naskah ini dan disebutkan dalam sumber kutipan dan daftar pustaka.

Apabila ternyata didalam naskah SKRIPSI ini dapat dibuktikan terdapat unsur-unsur PLAGIASI, saya bersedia SKRIPSI ini digugurkan dan gelar akademik yang telah saya peroleh (SARJANA) dibatalkan, serta diproses sesuai dengan peraturan perundang-undangan yang berlaku yaitu UU No. 20 Tahun 2003, Pasal 25 ayat 2 dan Pasal 70.

Undang-undang No. 20 Tahun 2003 Pasal 25 ayat 2 yang berisi "Lulusan perguruan tinggi yang karya ilmiahnya digunakan untuk memperoleh gelar akademik, profesi, atau vokasi terbukti merupakan jiplakan dicabut gelarnya." dan Pasal 70 yang berisi "Lulusan yang karya ilmiah yang digunakannya untuk mendapatkan gelar akademik, profesi, atau vokasi sebagaimana dimaksud dalam Pasal 25 ayat (2) terbukti merupakan jiplakan dipidana dengan pidana penjara paling lama dua tahun dan/atau pidana denda paling banyak Rp200.000.000,00 (dua ratus juta rupiah)."

Malang, 4 April 2014

Mahasiswa,

Candra Irwansyah

105060807111146

KATA PENGANTAR

Puji dan syukur Penulis panjatkan kehadirat Allah SWT, karena hanya dengan rahmat dan bimbingannya Penulis dapat menyelesaikan tugas akhir dengan judul "Pencarian Rute Angkutan Umum Menggunakan Algoritma Ant Colony Optimization" dengan baik. Tanpa rahmat dan bimbingan dari Tuhan Yang Maha Esa, maka niscaya Penulis tidak akan dapat menyelesaikan tugas akhir ini dengan baik dan tepat waktu.

Terima kasih pula Penulis sampaikan kepada pihak-pihak yang telah membantu Penulis dalam penyelesaian tugas akhir ini. Pihak-pihak tersebut antara lain:

1. Orang tua Penulis, Juhariansyah dan Jumratul Aini yang telah memberi motivasi, kasih sayang serta dukungan moril dan materil kepada Penulis. Muhammad Sabariansyah yang telah memberikan semangat dari awal sampai akhir pengerjaan tugas akhir ini.
2. Bapak Drs. Marji, M.Si. dan Issa Arwani, ST., MT.. selaku Ketua dan Sekretaris Program Studi Informatika serta segenap Bapak/Ibu Dosen, Staff Administrasi dan Perpustakaan Program Studi Teknik Informatika Universitas Brawijaya.
3. Bapak Aryo Pinandito, S.T., M.MT. dan Bapak Wayan Firdaus Mahmudy, S.Si., MT., Phd. selaku dosen pembimbing I dan dosen pembimbing II yang telah banyak memberikan bimbingan, ilmu dan saran dalam penyusunan tugas akhir ini.
4. Bapak Eriq Muhammad Adams J, ST., M.Kom. selaku dosen pembimbing akademik yang telah memberikan bimbingan, ilmu dan saran selama penulis belajar.
5. Seluruh rekan kerja dan supervisor BPTIK PTIIK UB yang telah memberi saran masukan kepada penulis hingga terselesaikannya skripsi ini.
6. Semua teman-teman di PTIIK, terimakasih atas segala bantuannya selama menjadi mahasiswa.

7. Semua pihak yang tidak dapat penulis sebutkan satu per satu yang terlibat baik secara langsung maupun tidak langsung demi terselesaikannya tugas akhir ini.

Semoga jasa dan amal baik mendapatkan balasan dari Allah SWT. Ibarat tak ada gading yang tak retak, dengan segala kerendahan hati, penulis menyadari sepenuhnya bahwa skripsi ini masih belum sempurna. Oleh karena itu kritik dan saran untuk kesempurnaan skripsi ini, senantiasa penulis harapkan dari berbagai pihak.

Malang, 4 April 2014

Penulis

ABSTRAK

Candra Irwansyah. 2014. Pencarian Rute Angkutan Umum Menggunakan Algoritma Ant Colony Optimization. Program Teknologi Informasi dan Ilmu Komputer, Universitas Brawijaya, Malang. Dosen Pembimbing : **Aryo Pinandito, S.T., M.MT. dan Bapak Wayan Firdaus Mahmudy, S.Si., MT., Phd.**

Kita sering dibingungkan pada saat memilih angkutan umum yang akan digunakan ketika ingin menuju ke suatu tempat. Hal ini disebabkan oleh kurangnya informasi rute setiap angkutan umum yang tersedia. Oleh sebab itu, sistem yang dapat menjangkau pencarian rute angkutan umum yang secara otomatis yang menggunakan diperlukan.

Algoritma Ant Colony Optimization dapat digunakan untuk mencari rute sebuah perjalanan. Algoritma ini bekerja dengan cara mencari setiap kemungkinan rute yang di pilih berdasarkan setiap lajur yang telah ditentukan di dalam sebuah matriks. Rute terbaik yang sudah didapatkan dari Algoritma Ant Colony Optimization ini kemudian di proses menggunakan Algoritma Brute Force untuk menentukan angkutan umum apa yang akan digunakan.

Data yang digunakan dalam penelitian ini adalah data rute angkutan umum yang ada dikota Malang. Sampel angkutan umum yang digunakan pada penelitian ini ada lima. Data angkutan umum kemudian dibuat menjadi sebuah matriks vertex jarak sehingga dapat diproses oleh Algoritma Ant Colony Optimization dan Algoritma Brute Force. Hasil pengujian menunjukkan bahwa waktu pemrosesan untuk pencarian jarak dekat yang memiliki jumlah kombinasi rute angkutan umum lebih banyak memiliki waktu pemrosesan yang lebih lama dibandingkan pencarian jarak jauh yang memiliki jumlah kombinasi rute angkutan umum lebih sedikit.

Kata kunci : Algoritma Ant Colony Optimization, Algoritma Brute Force, Angkutan Umum

ABSTRACT

Candra Irwansyah. 2014. ***Search For Public Transport Routes Using Ant Colony Optimization Algorithm.*** *Information Technology and Computer Science Program,* Brawijaya *University,* Malang. *Advisors*: **Aryo Pinandito, S.T., M.MT. dan Bapak Wayan Firdaus Mahmudy, S.Si., MT., Phd.**

We are often confused when choosing public transport that we will use to go to a place. This is caused by the lack of any public transport route information available. Therefore, the system that can automatically search public transport route required.

Ant Colony Optimization algorithms can be used to route a trip. This algorithm works by searching for every possible route that is selected based on each of the specified columns in a matrix. The best route that obtained from Ant Colony Optimized Algorithm is then processed using a Brute Force Algorithm to determine public transport to be used.

The Data that used in this research is public transport routes that exists in Malang. There are five public transportation samples that are used in this research. Then the public transportation data made into a matrix vertex distance so it can be processed by ant colony optimization algorithms and the brute force algorithms. The test result indicated that the processing time for a close range search that have a lot of public transportation routes combination requires more processing time than a long distance search that have fewer public transportaton route combination.

Keywords : Ant Colony Optimization Algortihm, Brute Force Algorithm, Public Transport

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

BAB I
PENDAHULUAN

1.1 Latar Belakang

Sistem informasi sebagai infrastruktur yang mampu menangani pengelolaan data dan informasi menjadi sangat penting di era globalisasi saat ini. , tak terkecuali aplikasi sistem informasi transportasi berbasis *smart phone* yang mengadopsi sistem operasi *Android* yang memudahkan para pengembang untuk membuat bahkan mengembangkan aplikasi pada *smart phone.* Android juga menyediakan platform terbuka bagi para pengembang untuk menciptakan aplikasi mereka sendiri untuk digunakan oleh bermacam peranti bergerak [KUS-11].

Pemilihan angkutan umum yang akan digunakan dari suatu tempat ke tempat yang lain hampir selalu membingungkan setiap orang karena terlalu banyak jumlah rute angkutan umum dan jalur yang dilalui pada kota tersebut. Hal ini disebabkan oleh kurangnya informasi rute setiap angkutan umum yang tersedia jika terjadi oper angkutan umum pada saat ingin menuju ke suatu tempat. Sistem pencarian rute yang sudah ada pada saat ini dibuat oleh Google dengan nama Google Maps dan untuk Indonesia sendiri tidak semua kota mendukung fitur pencarian rute angkutan umum dan pada kota lainnya baru mendukung fitur pencarian rute transportasi kendaraan pribadi dan untuk pejalan kaki.

Algoritma Ant Colony Optimization memungkinkan untuk digunakan dalam menentukan rute angkutan umum yang harus dipilih dari suatu lokasi ke lokasi yang lain serta memiliki banyak hasil kemungkinan lintasan yang dilewati. Aplikasi ini diharapkan dapat memberi kemudahan bagi pengguna untuk menentukan rute angkutan umum dari lokasi *pengguna* saat itu. Sehingga seorang *pengguna* dapat dengan cepat dan mudah menentukan angkutan umum apa saja yang harus digunakan untuk menuju lokasi yang hendak dituju.

ACO diadopsi dari perilaku koloni semut yang dikenal sebagai sistem semut [DOR-96]. Secara alamiah koloni semut mampu menemukan rute terpendek dalam perjalanan dari sarang ke tempat-tempat sumber makanan. Koloni semut dapat

menemukan rute terpendek antara sarang dan sumber makanan berdasarkan jejak kaki pada lintasan yang telah dilalui. Dalam algoritma semut terdapat beberapa parameter masukan yang berpengaruh dalam mendapatkan hasil yang maksimal.

Mengikuti perkembangan sistem informasi yang begitu pesat dan fasilitas yang tidak mudah dijangkau menyebabkan sulitnya pemenuhan kebutuhan terhadap informasi transportasi dan tidak efisien dalam pencarian secara manual untuk menentukan rute angkutan umum. Oleh karena itu, sistem yang dapat menjangkau pencarian rute angkutan umum secara otomatis yang menggunakan Algoritma Ant Colony Optimization diperlukan. Aplikasi pencarian rute angkutan umum ini merupakan aplikasi berbasis *smart phone* sederhana yang mudah digunakan oleh pengguna sebagai sarana pencarian otomatis dalam mencari rute angkutan umum.

Sistem ini dapat diimplementasikan pada berbagai jenis platform yang dapat memanfaatkan fitur *Google Maps* dan *GPS*. Dikarenakan jumlah pengguna Android di Indonesia menguasai 51.83% smartphone yang beredar di pasar Indonesia [STC-14], maka Platform yang digunakan pada penelitian ini adalah platform *Android* karena mendukung untuk penggunaan fitur *Google Maps* dan *GPS* serta mengingat banyaknya pengguna platform tersebut di Indonesia.

1.2 Rumusan Masalah

Berdasarkan uraian latar belakang di atas, maka dapat di rumuskan permasalahan pada penelitian ini yaitu bagaimana merancang dan mengimplementasikan Algoritma Ant Colony Optimization pada sebuah aplikasi untuk menentukan rute angkutan umum yang terdekat dan termurah serta menentukan parameter – parameter pendukung Algoritma Ant Colony Optimization paling baik untuk diterapkan pada *smart phone* berbasis Android.

1.3 Batasan Masalah

Agar permasalahan yang dirumuskan dapat lebih fokus, maka pada penelitian ini dibatasi dalam hal:

1. Fokus pembahasan terletak pada pencarian rute untuk angkutan umum di kota Malang dengan rute AL, CKL, ABG, GL, dan GA.

2. Rute angkutan umum hanya menyediakan 1 arah perjalanan.
3. Aplikasi tidak menentukan rute jalan kaki pengguna untuk menuju angkutan umum tersebut bila berada dilokasi yang tidak dilalui oleh angkutan umum sehingga rute jalan kaki pengguna diasumsikan secara tarik lurus dari posisi pengguna menuju titik awal dan akhir naik angkutan.
4. Proses penentuan vertex dilakukan dengan menggunakan program php yang menampilkan map dan menyimpan setiap latitude dan longitude dari rute setiap angkutan umum kedalam sebuah array yang akan langsung di simpan ke dalam aplikasi *Android* ini.

1.4 Tujuan

Tujuan yang ingin dicapai dalam penelitian ini yaitu membuat aplikasi yang dapat menentukan rute angkutan umum di daerah kota Malang menuju lokasi yang ditandai pengguna dari lokasi pengguna yang menggunakan fitur GPS pada perangkat *smart phone* berbasis *Android*.

1.5 Manfaat

Adapun manfaat yang dari penelitian ini diharapkan dapat memudahkan para pengguna perangkat *Android* untuk menentukan mana angkutan umum yang paling baik bila ingin menuju lokasi yang akan dituju. Sehingga pengguna dapat meminimalisir penggunaan waktu dan biaya yang dibutuhkan untuk menuju suatu lokasi bila menggunakan angkutan umum.

1.6 Sistematika Penulisan

Sistematika penulisan dalam skripsi ini sebagai berikut:

BAB I Pendahuluan

Memuat latar belakang, rumusan masalah, tujuan, batasan masalah, manfaat, metodologi pembahasan, dan sistematika penulisan.

BAB II **Kajian Pustaka dan Dasar teori**

Menguraikan tentang dasar teori dan referensi yang mendasari pembuatan aplikasi pencarian rute angkutan umum yang meliputi : Graph, Algoritma Ant Colony Optimization, Algoritma Brute Force dan Roulette Wheel Selection.

BAB III **Metode Penelitian**

Membahas metode yang digunakan dalam penelitian yang terdiri dari studi literatur, perancangan sistem perangkat lunak, implementasi sistem perangkat lunak, pengujian dan analisis, serta pengambilan kesimpulan.

BAB VI **Implementasi**

Membahas tentang implementasi dari pencarian rute angkutan umum manggunakan Algoritma Ant Colony Optimization sesuai dengan perancangan sistem aplikasi.

BAB V **Pengujian dan Analisa**

Memuat proses dan hasil pengujian terhadap sistem yang telah direalisasikan.

BAB VI **Penutup**

Memuat kesimpulan yang diperoleh dari pembuatan dan pengujian perangkat lunak yag dikembangkan dalam skripsi ini serta saran – saran untuk pengembangan lebih lanjut.

BAB II
KAJIAN PUSTAKA DAN DASAR TEORI

Bab ini berisi tinjauan pustaka yang meliputi kajian pustaka dan dasar teori yang diperlukan untuk penelitian. Kajian pustaka membahas penelitian yang telah ada dan yang diusulkan. Dasar teori membahas teori yang diperlukan untuk menyusun penelitian yang diusulkan.

Teori dasar yang akan dibahas pada bab ini yaitu konsep dasar :

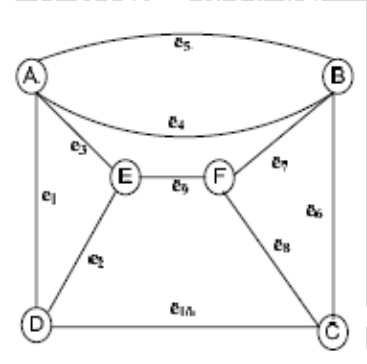
- *Graph*
- *Algoritma Ant Colony Optimization*
- *Algoritma Brute Force*

2.1 Graph

2.1.1 Definisi Graph

Definisi 2.1 [JWI-90]

Suatu graph G terdiri atas himpunan yang tidak kosong dari elemen – elemen yang disebut titik (vertek), dan suatu daftar pasangan vertek yang tidak terurut disebut sisi (edge). Himpunan vertek dari suatu graph G dinotasikan dengan V, dan daftar himpunan edge dari graph tersebut dinotasikan dengan E. Untuk selanjutnya suatu graph G dapat dinotasikan dengan G = (V, E).



Gambar 2.1 Contoh graph

Gambar 2.1, menunjukkan graph G dengan V = {V1, V2, V3 } dan E = {e1, e2, e3}.

2.1.2 Definisi Walk

Definisi 2.2 [JWI-90]

Suatu walk (jalan) dalam graph G adalah barisan vertek – vertek dan edge – edge yang dimulai dan diakhiri oleh suatu vertek. Panjang suatu walk dihitung berdasarkan jumlah edge dalam walk tersebut.

Walk juga dapat diartikan sebagai suatu perjalanan (dalam sebuah graph) dari vertek satu ke vertek lain yang terhubung dengan suatu edge.

2.1.3 Definisi Trail

Definisi 2.3 [JWI-90]

Walk yang panjangnya k pada suatu graph G adalah urutan k edge G yang berbentuk

uv, vw, wx, …, yz.

Walk ini dinotasikan dengan uvwx….yz, dan ditunjuk sebagai walk antara u dan z. Jika semua edge (tetapi tidak perlu semua vertek) suatu walk berbeda, maka walk itu disebut trail. Jika semua vertek pada trail itu berbeda, maka trail itu disebut path.

2.1.4 Graph Eulerian dan Graph Hamilton

Definisi 2.4 [JWI-90]

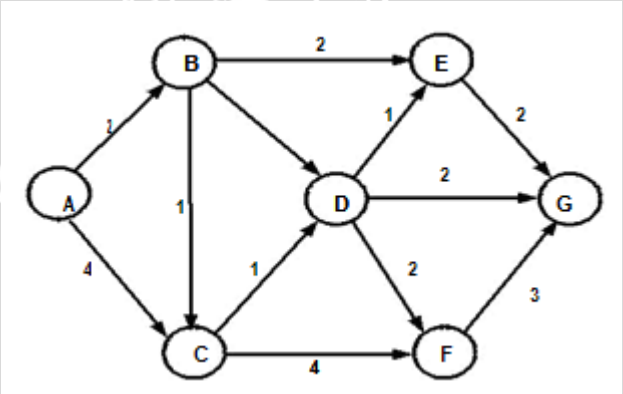
Graph terhubung G merupakan graph Euler (Eulerian) jika ada trail tertutup yang memuat setiap edge dari G. Trail semacam ini disebut trail Euler.

Graph terhubung G merupakan graph Hamilton (Hamiltonian) jika ada siklus yang memuat setiap vertek dari G. Siklur semacam ini disebut Siklus Hamilton.

2.1.5 Macam – macam Graph Menuruh Arah dan Bobotnya

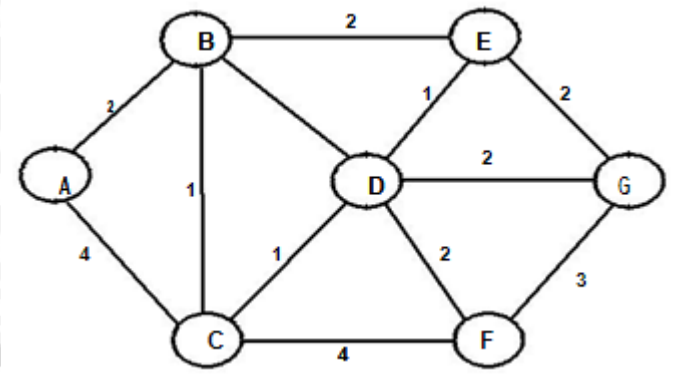
Menurut arah dan bobotnya, graph dibagi menjadi empat bagian, yaitu :

1. Graph berarah dan berbobot : setiap edge mempunyai arah (yang ditunjukkan dengan anak panah) dan bobot. Gambar 2.2 adalah contoh graph berarah dan berbobot yang terdiri dari tujuh vertek yaitu vertek A, B, C, D, E, F, G. Vertek A mempunyai dua edge yang masing – masing menuju ke vertek B dan vertek C, vertek B mempunyai tiga edge yang masing – masing menuju ke vertek C, vertek D dan vertek E. Bobot antara vertek A dan vertek B pun telah di ketahui.



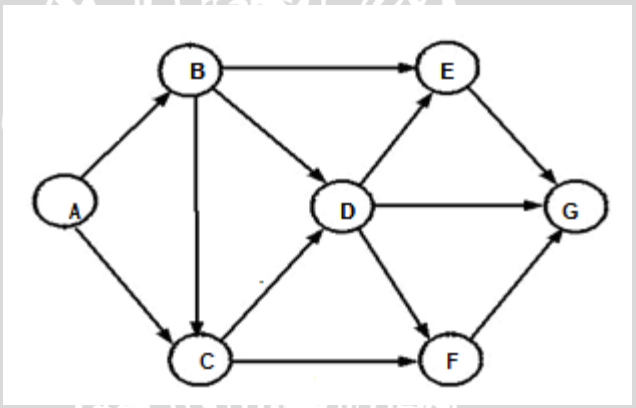
Gambar 2.2 Contoh graph berarah dan berbobot

2. Graph tidak berarah dan berbobot : setiap edge tidak mempunyai arah tetapi mempunyai bobot. Gambar 2.3 adalah contoh graph tidak berarah dan berbobot. Graph terdiri dari tujuh vertek yaitu vertek A, B, C, D, E, F, G. Vertek A mempunyai dua edge yang masing – masing berhubungan dengan vertek B dan vertek C, tetapi dari masing – masing edge tersebut tidak mempunyai arah. Edge yang menghubungkan vertek A dan vertek B mempunyai bobot yang telah diketahui begitu pula dengan edge – edge yang lain.



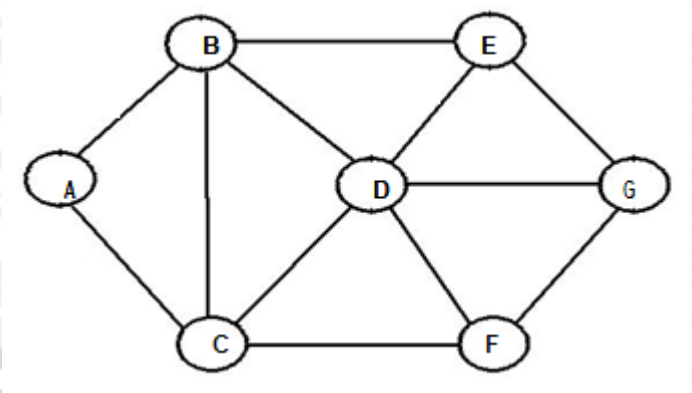
Gambar 2.3 Contoh graph tidak berarah dan berbobot

3. Graph berarah dan tidak berbobot : setiap edge mempunyai arah tetapi tidak mempunyai bobot. Gambar 2.4 adalah contoh graph berarah dan tidak berbobot.



Gambar 2.4 Contoh graph berarah dan tidak berbobot

4. Graph tidak berarah dan tidak berbobot : setiap edge tidak mempunyai arah dan tidak terbobot. Gambar 2.5 adalah contoh graph tidak berarah dan tidak berbobot.

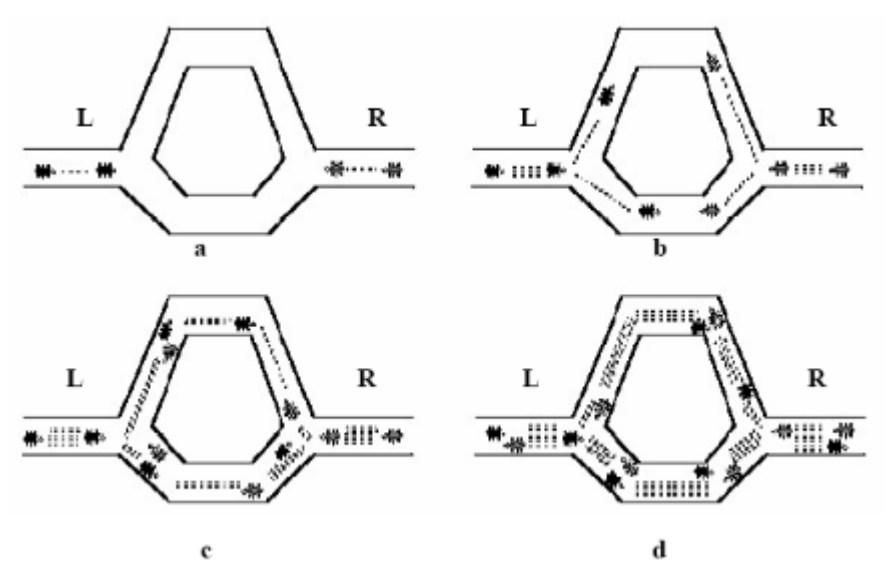


Gambar 2.5 Contoh graph tidak berarah dan tidak berbobot

2.2 Algoritma Ant Colony Optimization

2.2.1 Cara kerja Algoritma Ant Colony Optimization

Secara jelasnya cara kerja semut menemukan rute terpendek dalam Ant Colony Optimization adalah sebagai berikut : Secara alamiah semut mampu menemukan rute terpendek dalam perjalanan dari sarang ke tempat-tempat sumber makanan. Koloni semut dapat menemukan rute terpendek antara sarang dan sumber makanan berdasarkan jejak kaki pada lintasan yang telah dilalui. Semakin banyak semut yang melalui suatu lintasan, maka akan semakin jelas bekas jejak kakinya. Hal ini akan menyebabkan lintasan yang dilalui semut dalam jumlah sedikit, semakin lama akan semakin berkurang kepadatan semut yang melewatinya, atau bahkan akan tidak dilewati sama sekali. Sebaliknya lintasan yang dilalui semut dalam jumlah banyak, semakin lama akan semakin bertambah kepadatan semut yang melewatinya, atau bahkan semua semut akan melalui lintasan tersebut [DOR-91].



Gambar 2.6 Perjalanan semut dari sarang ke sumber makanan

Gambar 2.6.a di atas menunjukkan ada dua kelompok semut yang akan melakukan perjalanan. Satu kelompok bernama L yaitu kelompok yang berangkat dari arah kiri yang merupakan sarang semut dan kelompok lain yang bernama kelompok R yang berangkat dari kanan yang merupakan sumber makanan. Kedua kelompok semut dari titik awal keberangkatan sedang dalam posisi pengambilan keputusan jalan sebelah mana yang akan diambil. Kelompok semut L membagi dua kelompok lagi. Sebagian melalui jalan atas dan sebagian melalui jalan bawah. Hal ini juga berlaku pada kelompok semut R. Gambar 2.6.b dan gambar 2.6.c menunjukkan bahwa kelompok semut berjalan pada kecepatan yang sama dengan meninggalkan Pheromone (jejak kaki semut) di jalan yang telah dilalui. Pheromone yang ditinggalkan oleh semut - semut yang melalui jalan atas telah mengalami banyak penguapan karena semut yang melalui jalan atas berjumlah lebih sedikit dari pada jalan yang di bawah. Hal ini dikarenakan jarak yang ditempuh lebih panjang daripada jalan 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 2.6.d menunjukkan bahwa semut-semut yang lain pada akhirnya memutuskan untuk melewati jalan bawah karena Pheromone yang

ditinggalkan masih banyak. Sedangkan Pheromone pada jalan atas sudah banyak menguap sehingga semut-semut tidak memilih jalan atas tersebut. Semakin banyak semut yang melalui jalan bawah maka semakin banyak semut yang mengikutinya.

Demikian juga dengan jalan atas, semakin sedikit semut yang melalui jalan atas, maka Pheromone yang ditinggalkan semakin berkurang bahkan hilang. Dari sinilah kemudian terpilihlah rute terpendek antara sarang dan sumber makanan.

2.2.2 Langkah – langkah Penentuan Jalur Terpendek Algoritma Ant Colony Optimization

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

Langkah 1 :

a. Inisialisasi harga parameter-parameter algoritma.

Parameter-parameter yang di inisialisasikan adalah:

1. Intensitas jejak semut antar kota dan perubahannya (tij)
2. Banyak kota (n) termasuk x dan y (koordinat) atau dij (jarak antar kota).
3. Penentuan kota berangkat dan kota tujuan.
4. Tetapan siklus-semut (Q).
5. Tetapan pengendali intensitas jejak semut (a).
6. Tetapan pengendali visibilitas (ß).
7. Visibilitas antar kota = 1/dij (ηij)..... (1)
8. Jumlah semut (m).

9. Tetapan penguapan jejak semut (ρ).

10. Jumlah siklus maksimum (NCmax) bersifat tetap selama algoritma dijalankan, sedangkan tij 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 tij dilakukan, kemudian m semut ditempatkan pada kota pertama yang telah ditentukan.

Langkah 2 :

Pengisian kota pertama ke dalam tabu list. Tabu list merupakan sebuah wadah yang menyimpan rute perjalan yang dilakukan oleh seekor semut dari titik asal menuju titik tujuan sehingga apabila rute yang akan dipilih semut tersebut sudah terdaftar didalam tabu list nya maka perjalanan semut itu akan dihentikan dan akan dilanjutkan oleh semut selanjutnya. Hasil inisialisasi kota pertama semut pada langkah 1 harus diisikan sebagai elemen pertama tabu list. Hasil dari langkah ini adalah terisinya elemen pertama tabu list setiap semut dengan indeks kota pertama.

Langkah 3:

Penyusunan jalur kunjungan setiap semut ke setiap kota. Koloni semut yang sudah terdistribusi ke kota pertama akan mulai melakukan perjalanan dari kota pertama sebagai kota asal dan salah satu 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 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 (s) dan kota-kota lainnya dinyatakan sebagai {N- }, maka untuk menentukan kota tujuan digunakan persamaan probabilitas kota untuk dikunjungi sebagai berikut,

= untuk j .....(2)

= 0, untuk lainnya ..... (3)

dengan i sebagai indeks kota asal dan j sebagai indeks kota tujuan serta k sebagai indeks semut.

Langkah 4:

a. Perhitungan panjang rute setiap semut.

Perhitungan panjang rute tertutup (length closed tour) atau Lk setiap semut dilakukan setelah satu siklus diselesaikan oleh semua semut. Perhitungan dilakukan berdasarkan masing-masing dengan persamaan berikut:

.= ..... (4)

dengan yang merupakan rute perjalanan yang dilakukan oleh seekor semut k dan dij adalah jarak antara kota i ke kota j yang dihitung berdasarkan persamaan:

.....(5)

b. Pencarian rute terpendek.

Setelah Lk setiap semut dihitung, akan didapat harga minimal panjang rute tertutup setiap siklus atau LminNC dan harga minimal panjang rute tertutup secara keseluruhan atau Lmin.

c. Perhitungan perubahan harga intensitas jejak kaki 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 perubahan ini adalah:

.....(6)

dengan adalah perubahan harga intensitas jejak kaki semut antar kota setiap semut yang dihitung berdasarkan persamaan :

– .....(7)

untuk (i,j) vertex asal dan vertex tujuan dalam

dan .....(8)

untuk (i,j) lainnya.

Langkah 5:

a. Perhitungan harga intensitas jejak kaki semut antar kota untuk siklus selanjutnya. 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 antarkota untuk siklus selanjutnya dihitung dengan persamaan:

.....(9)

b. Atur ulang harga perubahan intensitas jejak kaki semut antar kota. intensitas jejak semut antar kota perlu diatur kembali agar memiliki nilai sama dengan nol.

Langkah 6:

Pengosongan tabu list, dan ulangi langkah 2 jika diperlukan. 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 langkah 2 dengan harga parameter intensitas jejak kaki semut antar kota yang sudah diperbaharui.

2.3 Algoritma Brute Force

Brute force adalah sebuah pendekatan yang lempang (straight forward) untuk memecahkan masalah, biasanya didasarkan pula pada pernyataan masalah (problem statement) dan definisi konsep yang dilibatkan. Algoritma brute force memecahkan masalah dengan sangat sederhana, langsung dan dengan cara yang jelas (obvious way) [MNR-06].

2.3.1 Karakteristik Algoritma Brute Force

Sebenarnya, algoritma Brute Force merupakan algoritma yang muncul karena pada dasarnya alur pikir manusia adalah Brute Force (langsung/to the point). Berikut merupakan beberapa karakteristik algoritma brute force :

1. Membutuhkan jumlah langkah yang banyak dalam menyelesaikan suatu permasalahan sehingga jika diterapkan menjadi suatu algoritma program aplikasi akan membutuhkan banyak memori.
2. Digunakan sebagai dasar dalam menemukan suatu solusi yang lebih efektif.

3. Banyak dipilih dalam penyelesaian sebuah permasalahan yang sederhana karena kemudahan cara berpikirnya.
4. Pada banyak kasus, algoritma ini banyak dipilih karena hampir dapat dipasti -kan dapat menyelesaikan banyak persoalan yang ada.
5. Digunakan sebagai dasar bagi perbandingan keefektifan sebuah algoritma.

2.3.2 Kekuatan dan Kelemahan Metode Brute Force

Berikut ini merupakan beberapa kekutan dan kelemahan metode brute force [SMR-04]:

Kekuatan

1. Metode *brute force* dapat digunakan untuk memecahkan hampir sebagian besar masalah (*wide applicability*).
2. Metode *brute force* sederhana dan mudah dimengerti.
3. Metode *brute force* menghasilkan algoritma yang layak untuk beberapa masalah penting seperti pencarian, pengurutan, pencocokan *string*, perkalian matriks.
4. Metode *brute force* menghasilkan algoritma baku (standard) untuk tugas – tugas komputasi seperti penjumlahan/perkalian n buah bilangan, menentukan elemen minimum atau maksimal di dalam tabel (*list*).

Kelemahan

1. Metode *brute force* jarang menghasilkan algoritma yang mangkus.
2. Beberapa algoritma *brute force* lambat sehingga tidak dapat diterima.
3. Tidak sekonstruktif/sekreatif teknik pemecahan masalah lainnya.

2.4 Roulette Wheel Selection

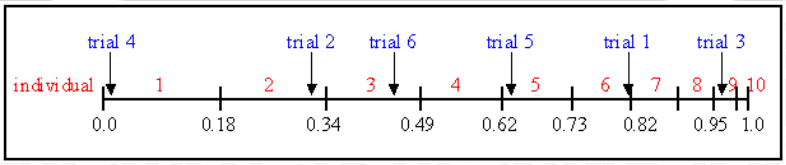
Roulete Wheel selection adalah seleksi berdasarkan kualitas individual. Semakin berkualitas individu semakin besar kemungkinan individu ini terpilih untuk menjadi anggota pada populasi yang baru. Metode ini diajukan oleh John Holland.

Sebagai contoh kita mempunyai 11 individu di dalam populasi saat ini. Dari populasi ini dilakukan proses seleksi dimana kita memilih individu yang memiliki kualitas yang terbaik dan ditempatkan di populasi terbaru. Untuk menghitung kualitas dari individu digunakan fungsi matematika tertentu. Ini berarti kualitas dari individu akan disimbolkan menggunakan angka. Berdasarkan angka-angka kualitas ini, kemudian kita menghitung kemungkinan terseleksi. Semakin besar angka kualitas yang dimiliki individu tertentu, semakin tinggi juga kemungkinan terpilih. Jika individu tidak memiliki angka kualitas maka kemungkinan terpilih bisa dipastikan tidak ada dan individu ini tidak akan digambarkan di dalam roulette-wheel.

Number of individual	1	2	3	4	5	6	7	8	9	10	11
fitness value	2.0	1.8	1.6	1.4	1.2	1.0	0.8	0.6	0.4	0.2	0.0
selection probability	0.18	0.16	0.15	0.13	0.11	0.09	0.07	0.06	0.03	0.02	0.0

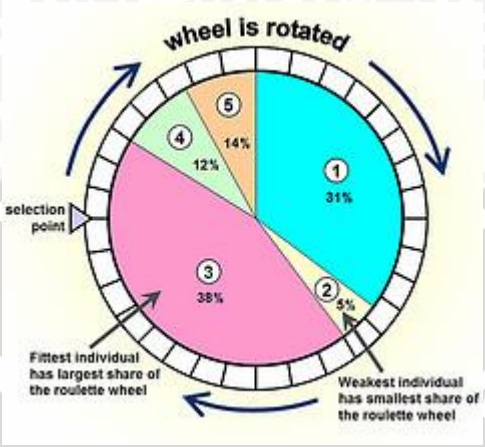
Gambar 2.7 Probabilitas seleksi dan nilai fitness

Berdasarkan tabel di atas dapat dilihat individu ke-11 tidak memiliki angka kualitas, sehingga secara otomatis kemungkinan terpilihnya individu ini tidak ada sehingga tidak akan digambarkan di dalam roulutte wheel. Walaupun nama dari pendekatan ini adalah Wheel namun bisa digambarkan dalam bentuk garis seperti gambar 1 berikut



Gambar 2.8 Roulette Wheel Selection 1

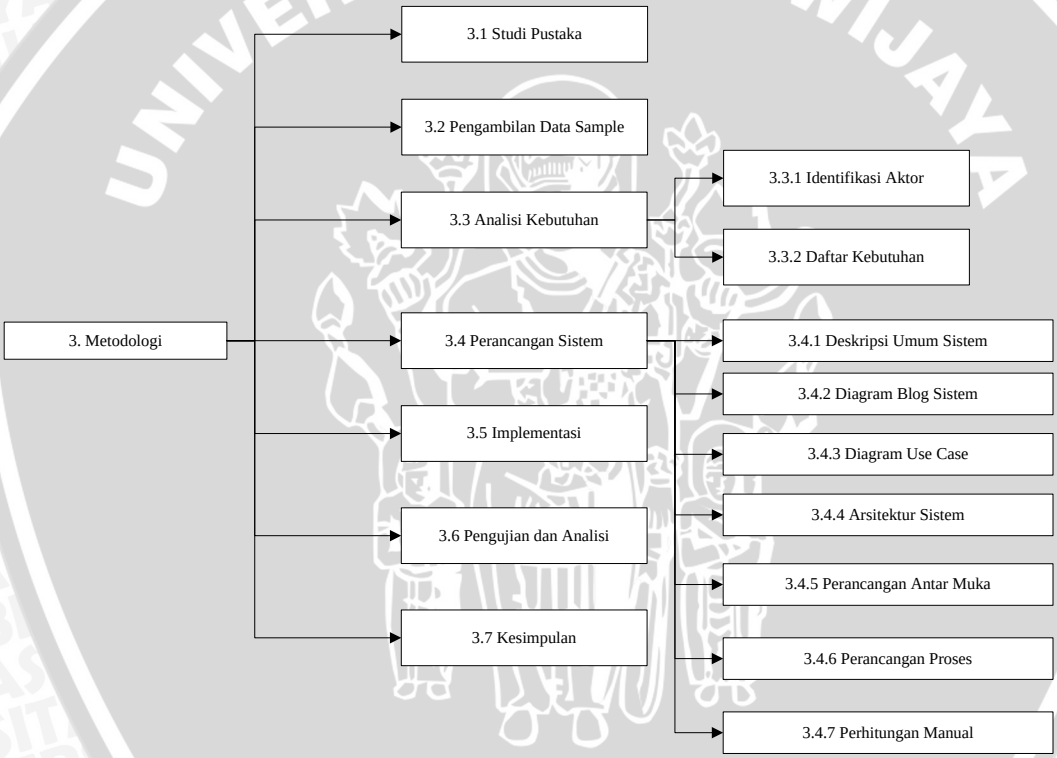
atau bisa digambarkan dalam bentuk lingkaran seperti ditunjukkan gambar berikut:



Gambar 2.8 Roulette Wheel Selection 2

BAB III
METODOLOGI PENELITIAN

Pada bab ini dijelaskan langkah-langkah yang akan dilakukan dalam pengerjaan tugas akhir, yaitu studi literatur dan penyusunan dasar teori, analisa dan perancangan, implementasi, pengujian dari aplikasi perangkat lunak yang akan dibuat, hingga penulisan laporan. Kesimpulan dan saran disertakan sebagai catatan atas aplikasi dan kemungkinan arah pengembangan aplikasi selanjutnya. Gambar 3.1 menunjukkan tahapan penelitian secara umum.



Gambar 3.1 Tahapan penelitian

3.1 Studi Pustaka

Studi literatur menjelaskan dasar teori yang digunakan untuk menunjang penulisan skripsi. Teori – teori pendukung tersebut yaitu :

1. *Graph*
2. *Algoritma Ant Colony Optimization*

3. *Algoritma Brute Force*

3.2 Pengambilan Data Sampel

Metode ini digunakan untuk mendapatkan data sampel sebagai acuan untuk pengembangan perangkat lunak. Data sampel yang dimaksud adalah data rute angkutan umum yang digunakan untuk implemetasi, serta data yang digunakan dapat menghitung tingkat kesuksesan perangkat lunak pada tindakan pencarian rute angkutan umum. Tabel untuk teknik pengambilan data sampel ditunjukkan pada tabel 3.1 berikut :

Tabel 3.1 Teknik pengambilan data sampel

Data	Jenis Data	Cara Pengambilan
Rute angkutan umum	- Rute angkutan umum - Jalur yang dilewati angkutan umum	Observasi

3.3 Analisis Kebutuhan

Analisis kebutuhan digunakan untuk mendapatkan semua kebutuhan sistem yang membangun pencarian rute angkutan umum. Metode analisis kebutuhan yang digunakan adalah *Object oriented analysis* dengan menggunakan bahasa pemrograman Java. Diagram *use case* digunakan untuk mendeskripsikan kebutuhan – kebutuhan dan fungsionalitas dari perspektif *user*. Analisis ini dilakukan dengan mengidentifikasi semua kebutuhan *requirements* sistem yang kemudian akan dimodelkan dalam diagram *use case*. Kebutuhan fungsional yang nantinya akan disediakan oleh aplikasi ini adalah :

1. Aplikasi pada Android ini menyediakan fungsi untuk menampilkan peta dan lokasi sekarang pada peta.
2. *User* menentukan titik lokasi tujuan pada map sehingga aplikasi dapat menentukan rute angkutan umum dari titik asal ke titik tujuan.

3. Aplikasi ini memiliki fitur legenda pada saat proses pencari rute angkutan umum sudah dilakukan sehingga dapat memudahkan user untuk mengetahui rute angkutan apa yang akan digunakan.

3.3.1 Identifikasi Aktor

Identifikasi aktor bertujuan untuk mengetahui aktor yang terlibat dalam penggunaan aplikasi pencarian rute angkutan umum. Aktor beserta deskripsinya ditunjukkan pada tabel 3.2 berikut :

Tabel 3.2 Identifikasi aktor

Aktor	Deskripsi Aktor
User	User merupakan aktor pengguna yang menggunakan aplikasi pencarian rute angkutan umum dengan menggunakan Algoritma Ant Colony Optimization.

3.3.2 Daftar Kebutuhan

Daftar kebutuhan terdiri dari kebutuhan fungsional dan non-fungsional. Pada tabel 3.3 kebutuhan fungsional ditunjukkan dengan penomoran F, sedangkan pada tabel 3.4 kebutuhan non-fungsional ditunjukkan dengan penomoran NF.

Tabel 3.3 Tabel kebutuhan fungsional

Identifikasi	Kebutuhan	Use case
F01	Aplikasi harus bisa menampilkan rute angkutan umum ke tujuan yang sudah dipilih dari keberadaan *user*	Menentukan rute angkutan umum
F02	Aplikasi harus bisa menampilkan legenda dari rute angkutan dan jalan kaki yang sudah digambarkan oleh aplikasi kepada *user*	Melihat legenda setiap rute

Tabel 3.4 Daftar kebutuhan non-fungsianal

Parameter	Kode	Deskripsi Kebutuhan
Usability	NF01	Aplikasi harus dapat dengan mudah digunakan oleh pengguna (Ketika pengguna menekan *icon* aplikasi pencarian rute angkutan umum yang telah terpasang

		pada *smartphone* Android, maka sistem akan langsung menampilkan keberadaan *user* dan *user* apabila menekan ke sebuah lokasi yang ada dipeta maka aplikasi akan menampilkan marker sebagai lokasi tujuan)
	NF02	Aplikasi memiliki fitur setting untuk menentukan mana default rute yang pertama ditampilkan ketika telah selesai melakukan komputasi dan berapa jarak maksimal untuk pertama kali naik ke angkutan umum dari lokasi *user* sekarang.
Compatibility	NF03	Aplikasi harus dapat digunakan pada *smartphone* dengan sistem operasi Android

3.4 Perancangan Sistem

3.4.1 Deskripsi Umum Sistem

Sistem yang dibuat adalah sistem pencarian rute angkutan umum dengan menggunakan Algoritma Ant Colony Optimization. Sistem ini membutuhkan masukan data berupa jarak antar node yang akan disimpan kedalam bentuk matriks jarak antar node yang disimpan secara statis didalam aplikasi. Selain itu sistem akan meberikan informasi angkutan umum mana yang akan dipilih setelah mendapatkan jarak antar semua node. Angkutan umum terpilih tersebut di cari menggunakan *algoritma brute force.* Secara garis besar sistem ini dirancang untuk memberikan informasi rute angkutan umum mana yang akan digunakan oleh user untuk menuju ke lokasi tujuan. Proses pencarian rute angkutan umum dengan menggunakan Algoritma Ant Colony Optimization dan Brute Force di tunjukkan di dalam gambar 3.2 berikut :

Algoritma Ant Colony Optimization dan Algoritma Brute Force pada system

```
JUDUL : Algoritma Ant Colony Optimization dan Brute Force

Deskripsi : Masukan berupa matriks jarak [jumlahNode] [jumlahNode]
kemudian matriks jarak di olah menjadi matriks visibilitas
[jumlahNode][jumlahNode] serta yang terakhir akan diperolah matriks
```

probabilitas [jumlahNode][jumlahNode] yang di olah dari matriks visibilitas. Pemilihan lintasan dilakukan dengan cara membangkitkan bilangan random dalam interval (probabilitasTerkecil, probabilitasTerbesar). Setelah itu digunakan proses lotere (roulette-wheel selection) untuk memilih di area mana bilangan random yang dibangkitkan tadi terletak dalam lingkaran lotere. Setelah mendapatkan lintasan node dari proses Algoritma Ant Colony maka proses selanjutkan yang akan dilakukan yaitu pencari rute angkutan umum yang akan dipilih dengan menggunakan Algoritma Brute Force.

ALGORITMA

MASUKAN

- Inisialisati matriks jarak[jumlahNode][jumlahNode]
- Inisialisasi harga parameter – parameter algoritma.
- Inisialisasi matriks ruteAngkutanUmum[jumlahRuteAngkutanUmum][jumlahNodeRuteAngkutanUmum]
- Inisialisasi asal dan tujuan
- Pengisian kota asal kedalam setiap tabu list semut.

PROSES

1. Membentuk nilai pada matriks probabiliatas.
2. Membangkitkan bilangan random untuk setiap semut.
3. Membentuk area lingkaran lotera dengan cara mengambil komulatif dari nilai probabilitas.
4. Melakukan proses lotere (roulette-wheel selection) dan memilih pada area mana bilangan random tadi terletak dalam lingakaran lotere.
5. Memasukkan node terpilih kedalam tabu list apabila node tersebut tidak berada dalam tabu list serta bukan menjadi node tujuan.
6. Membandingkan jarak dari setiap tabu list semut yang berhasil sampai ke node tujuan kemudian dipilih jarak terpendeknya dan dimasukkan kedalam matriks rute terpendek.
7. Melakukan pencarian rute angkutan umum terpilih dengan cara membandingkan mana jumlah node rute angkutan umum yang terbanyak pada matriks ruteAngkutanUmum dengan matriks rute terpendek menggunakan algoritma brute force.

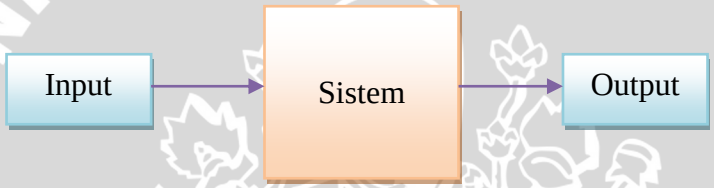
KELUARAN

- Jarak terpendek dan rute angkutan umum yang terpilih.

Gambar 3.2 Gambaran proses pada sistem

3.4.2 Diagram Blok Sistem

Diagram blok merupakan sebuah diagram yang menunjukkan gambaran dari proses pencarian rute angkutan umum menggunakan Algoritma Ant Colony Optimization. Blok diagram ini menunjukkan interasi antara pengguna sistem dengan sistem. *User* mimilih lokasi tujuan pada map sebagai inputan dan sistem mengeluarkan hasil rute angkutan umum serta angkutan umum yang terpilih. Pada gambar 3.3 merupakan diagram blok sistem pencarian rute angkutan umum memfokuskan pada input-proses-output dan komponen dari Algoritma Ant Colony Optimization.



Gambar 3.3 Diagram blok sistem pencarian rute angkutan umum

Beberapa komponen dari diagram blok pada gambar 3.2 sebagai berikut :

1. Input

   Bagi *User, input* kedalam sistem adalah memasukkan titik tujuan yang akan dituju pada map dan titik asal akan secara otomatis didapat dari current location (GPS).

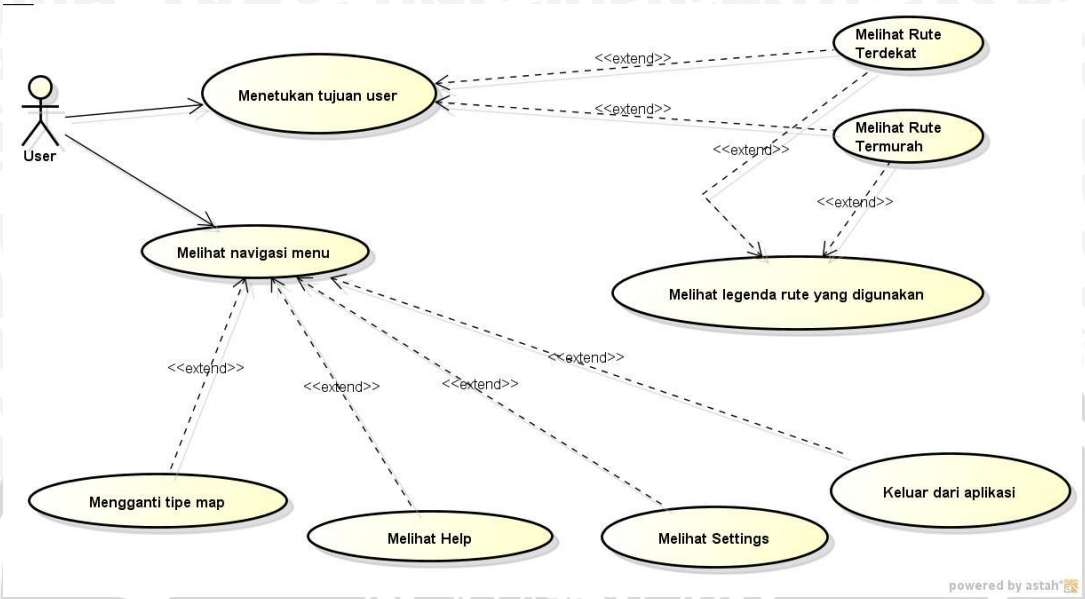
2. Output

   Bagi *User, Output* dari sistem adalah rute angkutan umum berupa polyline dan angkutan umum yang terpilih.

3.4.3 Diagram Use Case

Pemodelan *use case* sistem diperoleh dari kebutuhan fungsional yang digunakan untuk menggambarkan interaksi antara satu atau lebih aktor dengan

sistem yang akan dibuat. Gambar 3.4 menunjukkan use case sistem pencarian rute angkutan umum.



Gambar 3.4 *Use Case* sistem pencarian rute angkutan umum

Gambar 3.4 merupakan *use case* sistem pencarian rute angkutan umum yang terdiri dari 1 aktor yaitu *user*, dimana *user* dapat menetukan dimana tujuan yang ingin dikunjungin sehingga aplikasi dapat menentukan rute angkutan umum yang melalui lokasi tujuan tersebut.

- Use Case menentukan tujuan user

  Pada use case ini sebelum *user* mendapatkan rute angkutan umum dan informasi angkutan umum apa yang akan digunakan *user* harus terlebih dahulu menentukan dimana titik lokasi yang akan dikunjungi oleh *user* dengan cara menyentuh lokasi tujuan pada map dan menahannya selama kurang lebih 1 detik maka secara otomatis akan terdapat sebuah marker pada peta tersebut. Setelah marker lokasi tujuan pada peta sudah ditampilkan *user* dapat memilih apakah ingin menentukan titik lokasi tujuan tersebut sebagai lokasi tujuan atau tidak pada saat dialog pertanyaan di tampilkan pada peta.

- Use Case melihat rute terdekat

  Use case melihat rute terdekat ini akan menampilkan mana rute yang paling terdekat untuk menuju lokasi tujuan dengan pertukaran – pertukaran angkutan umum yang digunakan.

- Use Case melihat rute termurah

  Use case milihat rute termurah ini akan menampilkan mana rute yang paling murah yaitu jumlah rute angkutan umum yang paling sedikit untuk menuju lokasi tujuan meskipun jarak tempuhnya jauh.

- Use Case melihat legenda rute yang digunakan

  Use care melihat legenda merupakan acuan informasi untuk warna garis dan angkutan umum apa yang digunakan pada peta. Dengan adanya legenda ini diharapkan *user* akan lebih mengerti mengenai angkutan umum apa yang akan digunakan, dimana *user* naik angkutan umum dan pada saat ada proses pengoperan angkutan umum dimana *user* naik dan turun.

➢ Use Case melihat navigasi menu

Navigasi menu ini dibuat untuk lebih mempermudah *user* untuk melakukan pengaturan dan melihat help dengan tampilan yang *user friendly*.

- Use Case mengganti tipe map

  Pada use case ini *user* dapat memilih jenis atau tipe peta apa yang akan digunakan. Tipe map pada peta ini terdapat 3 tipe peta yaitu map dengna tipe normal, map dengan tipe terrain dan map dengan tipe satelit.

- Use Case melihat help

  Use case melihat help digunakan agar *user* dapat melihat informasi tentang penjelasan singkat dari aplikasi ini dan bagaimana cara penggunaannya.

- Use Case melihat settings

  Use case melihat setting digunakan untuk menentukan default pilihan rute angkutan yang akan ditampilkan pertama kali ketika selesai melakukan proses komputasi baik rute terdekat ataupun rute termurah dan juga digunakan untuk menentukan jarak maksimal untuk menuju lokasi naik angkutan umum terdekat dari lokasi *user* sekarang.

- Use Case keluar aplikasi

  Use case keluar aplikasi ini merupakan proses dimana ketika seorang *user* ingin mengakhiri penggunaan aplikasi ini.

3.4.4 Arsitektur Sistem

Perancangan aplikasi pencarian rute angkutan umum menggunakan Algoritma Ant Colony Optimization dapat dilihat lebih jelas pada gambar 3.5 di bawah :

Gambar 3.5 Arsitektur sistem pencarian angkutan umum

Dalam gambar diatas di jelaskan bagaimana cara aplikasi ini bekerja. Pertama, user melihat map dan terdapat current location sebagai asal dan user menentukan lokasi tujuan yang akan dituju pada map.

Sistem kemudian akan memproses pencarian rute angkutan umum dari titik asal dan titik tujuan yang telah ditentukan oleh user melalui proses yang terdiri dari

pencarian rute terpendek menggunakan *Algoritma Ant Colony Optimization* dan pemilihan angkutan umum yang akan digunakan menggunakan *Algoritma Brute Forced.*

a. User

User merupakan aktor yang bertindak secara langsung dalam penggunaan aplikasi pencarian rute angkutan umum. User yang akan menggunakan aplikasi ini harus mengerti peta malang sendiri sehingga dapat menentukan lokasi tujuan yang akan dituju.

b. Antar Muka Aplikasi

Antar muka aplikasi merupakan media interaksi user dengan aplikasi pencarian rute angkutan umum. Pada tampilan awal aplikasi ini user langsung dihadapkan kepada layout map dengan tertera current location pada map. Pada sisi navigasi user akan mendapatkan menu pemilihan sebagai berikut :

1. Type map yang terdiri dari normal, terrain dan satellite.
2. Map (menu untuk menampilkan peta).
3. Settings (menu untuk malakukan pengaturan).
4. Help (menu untuk menampilkan help).
5. Exit (menu untuk keluar dari aplikasi).

c. Lokasi Sekarang

Lokasi sekarang merupakan titik dimana user berada pada saat ini dan titik ini yang sekaligus akan dijadikan sebagai titik asal. Lokasi sekarang ini didapatkan dari proses pencarian lokasi sekarang menggunakan GPS (Global Position System).

d. Lokasi Tujuan

Lokasi tujuan merupakan titik dimana lokasi yang akan dituju oleh user. Lokasi tujuan ini didapat hasil dari pemilihan tujuan pada peta dengan cara menekan lokasi tujuan pada peta.

e. Cari Jalur

Cari jalur merupakan kelas yang bertindak sebagai penampung sementara data yang didapat dari lokasi sekarang sebagai titik asal dan lokasi tujuan sebagai titik tujuan. Setelah data lokasi sekarang dan lokasi tujuan didapat maka kelas ini akan melakukan proses pencarian rute angkutan umum dan mengembalikan hasil dari pencarian rute dan angkutan terpilih ke antar muka aplikasi.

f. Ant Colony Optimization

Ant colony Optimization disini bertindak sebagai kelas yang menjadi mesin pencari rute angkutan umum dengan berdasarkan jumlah siklus dan jumlah semut yang digunakan. Algoritma ini akan mencari jarak terdekat dari semua kemungkinan rute yang ada berdasarkan setiap rute yang dilalui oleh semua semut yang berhasil menuju lokasi tujuan dari asalnya dan akan menghasilkan rute yang paling pendek dari semua siklus.

g. Brute Force

Setelah rute sudah didapatkan melalui proses yang ada pada Algoritma Ant Colony Optimization langkah selanjutnya kita akan mencari angkutan umum yang mana yang akan terpilih secara maksimal dengan cara melakukan penelusuran satu persatu ke setiap daftar rute angkutan umum yang ada di aplikasi. Untuk melakukan proses pancarian angkutan terpilih ini kita akan menggunakan algoritma brute force sehingga dapat menghasilkan jumlah angkutan umum yang paling sedikit. Pada proses ini algoritma brute force akan melakukan

pencocokan apakah node rute yang berurutan tersebut adalah yang paling maksimal ataupun tidak untuk setiap daftar rute angkutan umum yang tersedia, proses ini akan terus dilakukan sampai node yang berada di rute tersebut habis terisi pada rute angkutan umum yang terpilih.

h. Rute Angkutan Umum dan Angkutan Terpilih

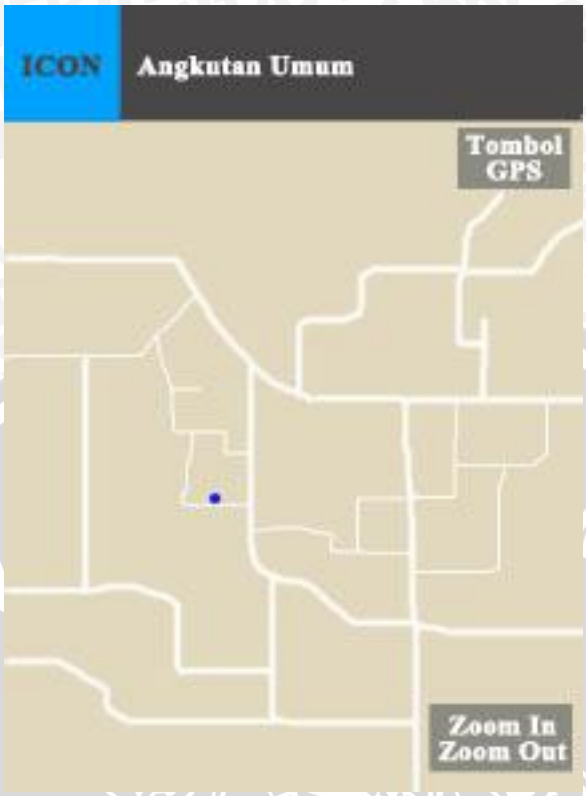
Pada proses ini aplikasi akan mencetak garis rute angkutan umum setelah rute sudah di peroleh melalui proses Algoritma Ant Colony Optimization dan setiap garis rute akan di bedakan berdasarkan nama angkutan umum yang terpilih sehingga dapat memudahkan user untuk membaca rute angkutan umum pada peta serta akan dibuat sebuah legenda yang akan membantu user untuk mengetahui nama angkutan dan mengacu kepada warna garis yang ada di peta. Setiap rute yang ada di peta akan di bedakan berdasarkan warna yang mengacu kepada nama angkutan umum yang digunakan.

3.4.5 Perancangan Antar Muka

Perancangan antarmuka bertujuan untuk mewakili keadaan sebenarnya dari sistem yang akan dibangun. Berikut adalah penjelasan dari setiap antar muka aplikasi ini :

a. Layout utama

Layout utama ini merupakan layout yang pertama ditampilkan pada saat *user* membuka aplikasi. Pada layout ini aplikasi akan menampilkan peta yang berasal dari google dengan tipe peta normal. Peta ini secara otomatis akan menampilkan lokasi sekarang dengan bantuan GPS (Global Position System), juga terdapat tombol menuju lokasi sekarang serta tombol perbesar dan perbesar peta. Perancangan layout utama ditunjukkan didalam gambar 3.6 berikut :



Gambar 3.6 Layout utama

b. Layout Navigasi Menu

Pada layout navigasi menu ini kita akan menemukan sejumlah menu. Menu yang disediakan pada aplikasi ini ialah sebagai berikut :

1. Type Map Normal

   Menu ini akan mengganti tipe peta menjadi tampilan normal.

2. Type Map Terrain

   Menu ini akan mengganti tipe peta menjadi tampilan lapang.

3. Type Map Sattelite

Menu ini akan mengganti tipe peta menjadi tampilan satelit.

4. Map

   Menu ini akan menampilkan layout peta.

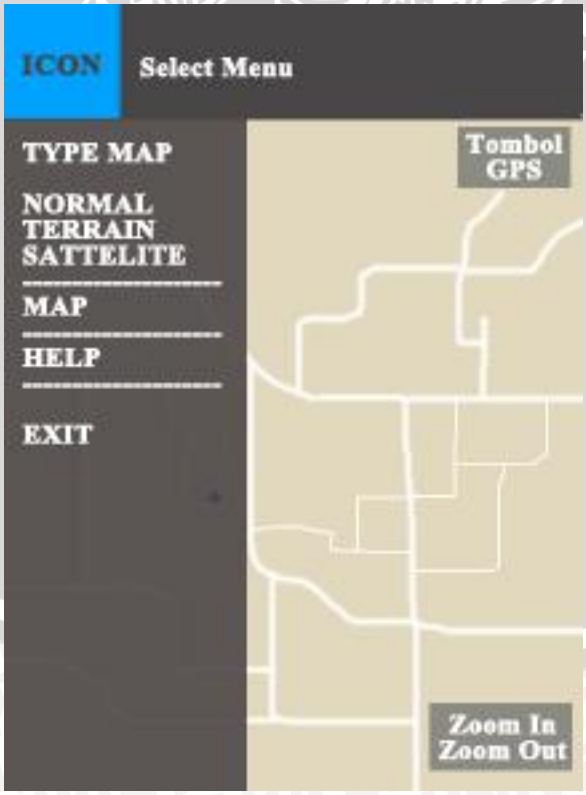
5. Help

   Menu ini akan menampilkan layout help.

6. Exit

   Menu ini akan mengakhiri aplikasi atau keluar dari aplikasi.

Perancangan layout navigasi menu ditunjukkan didalam gambar 3.7 berikut :



Gambar 3.7 Layout navigasi menu

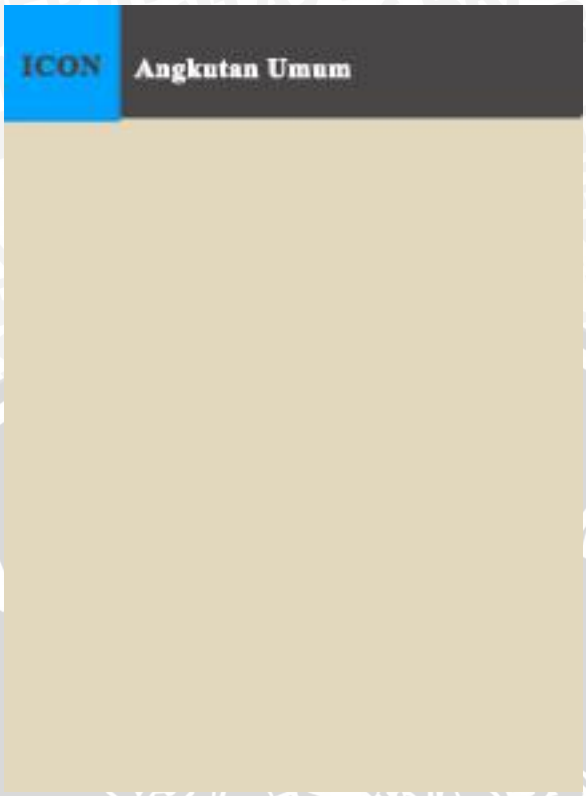
c. Layout Settings

Layout ini akan digunakan untuk melakukan pengaturan default jenis rute yang akan ditampilkan pertama kali ketika telah selesai melakukan proses komputasi baik itu rute terdekat ataupun rute termurah dan juga digunakan untuk melakukan pengaturan default jarak maksimal rute angkutan umum yang pertama kali di naiki dengan lokasi *user* sekarang. Perancangan layout setting ditunjukkan didalam gambar 3.8 berikut :

Gambar 3.8 Layout Settings

d. Layout Help

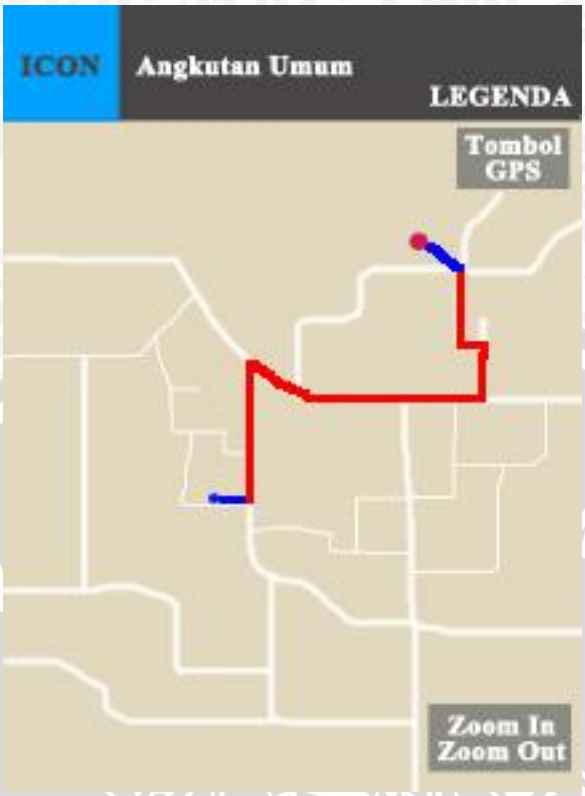
Layout ini akan memberikan informasi tentang kontak pengembang, versi aplikasi, logo dari aplikasi dan penjelasan singkat dari aplikasi ini. Perancangan layout help ditunjukkan pada gambar 3.9 berikut :



Gambar 3.9 Layout Help

e. Layout Hasil

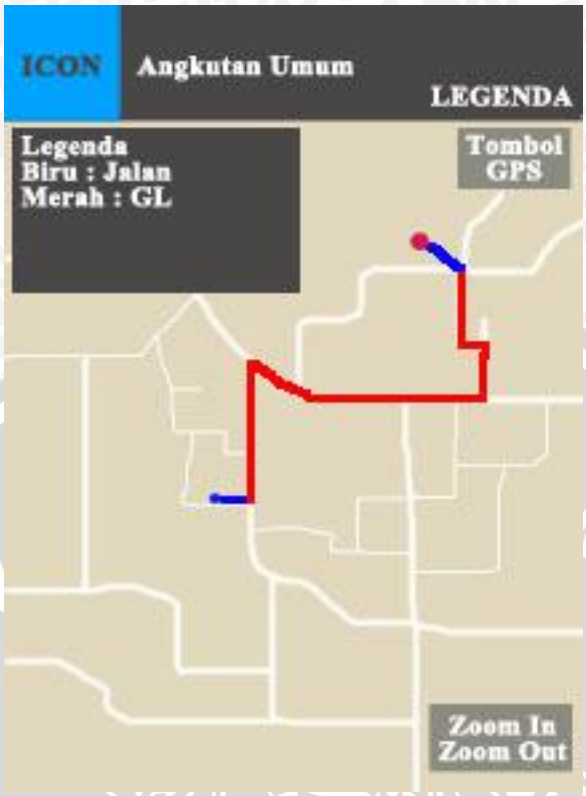
Pada layout hasil ini sebelum aplikasi akan membuat garis berwarna yang menunjukkan angkutan umum apa saja yang akan digunakan kita terlebih dahulu menyentuh lokasi tujuan pada peta maka akan terdapat marker sebagai penanda bahwa lokasi tersebut akan menjadi titik tujuan ada apa bila marker di tekan akan keluar popup pencarian apakah ingin mencari rute atau tidak. Perancangan layout hasil ditunjukkan didalam gambar 3.10 berikut :



Gambar 3.10 Layout hasil

f. Layout Hasil dan Legenda

Layout ini merupakan tampilan akhir ketika semua proses pencarian rute sudah dilakukan. Pada tampilan di layout aplikasi akan menampilkan popup legenda berdasarkan nama angkutan umum yang digunakan dengan informasi warna garisnya. Garis yang ada pada peta memiliki warna yang mengacu kepada informasi yang berada pada popup legenda. Perancangan layout hasil dan legenda ditunjukkan didalam gambar 3.11 berikut :



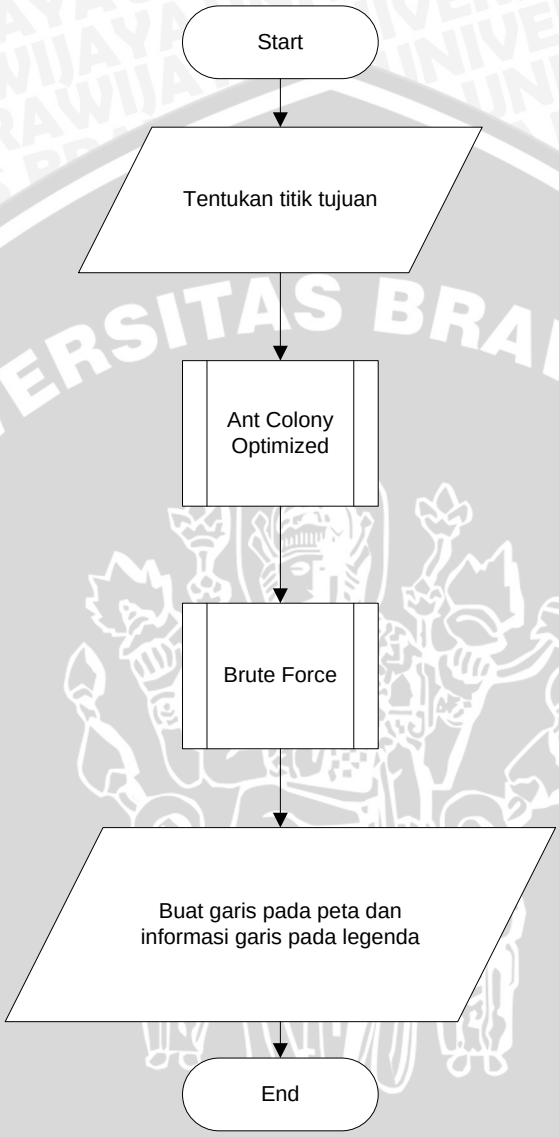
Gambar 3.11 Layout hasil dan legenda

3.4.6 Perancangan Proses

Perancangan proses merupakan perancangan tahap atau urutan sistem untuk melakukan proses pencarian rute. Pada tahapannya aplikasi ini memiliki proses pencarian titik tujuan, mencari rute, menentukan angkutan umum mana yang terpilih beserta rute terpilih yang terdapat pada angkutan tersebut dan mengeluarkan hasil akhir berupa garis rute dari titik asal ke titik tujuan dan memiliki warna yang berbeda untuk setiap angkutan yang terpilih lalu kemudian informasi tentang angkutan umum terpilih beserta warna garisnya di simpan di dalam legenda.

Proses pertama yang dilakukan oleh sistem adalah menentukan titik asal yang didapatkan dari GPS dan titik tujuan yang ditentukan oleh *user*. Titik asal dan titik tujuan tersebut akan digunakan sebagai acuan dalam menentukan rute melalui proses pencarian rute menggunakan *Algoritma Ant Colony Optimization*. Setelah mendapatkan rute dari proses pencarian rute menggunakan *Algoritma Ant Colony Optimization* data tersebut akan di jadikan acuan untuk menentukan mana angkutan

umum yang terpilih beserta rute terpilih pada angkutan umum tersebut menggunakan *Algoritma Brute Force*. Diagram alir yang menunjukkan proses aplikasi ini yang ditunjukkan didalam gambar 3.12 berikut :

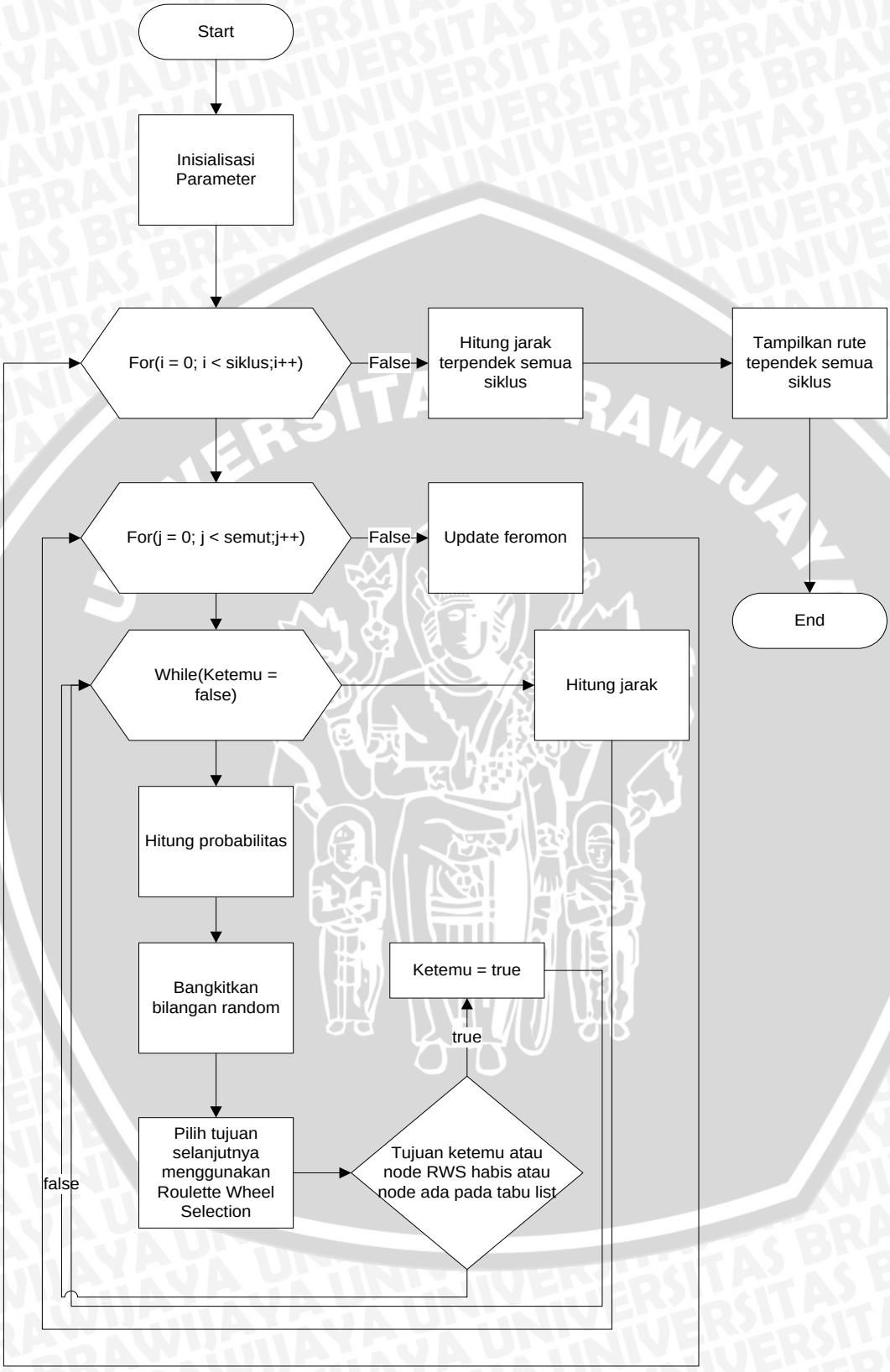


Gambar 3.12 Diagram alir sistem

3.4.6.1 Ant Colony Optimization

Pada tahap ini aplikasi akan menentukan rute angkutan umum dengan melalui proses penyebaran sejumlah semut yang sudah ditentukan dimulai dari titik asal menuju titik tujuan dengan sejumlah siklus yang sudah ditentukan pula. Setiap semut yang melakukan perjalanan dari titik asal akan menentukan rute titik

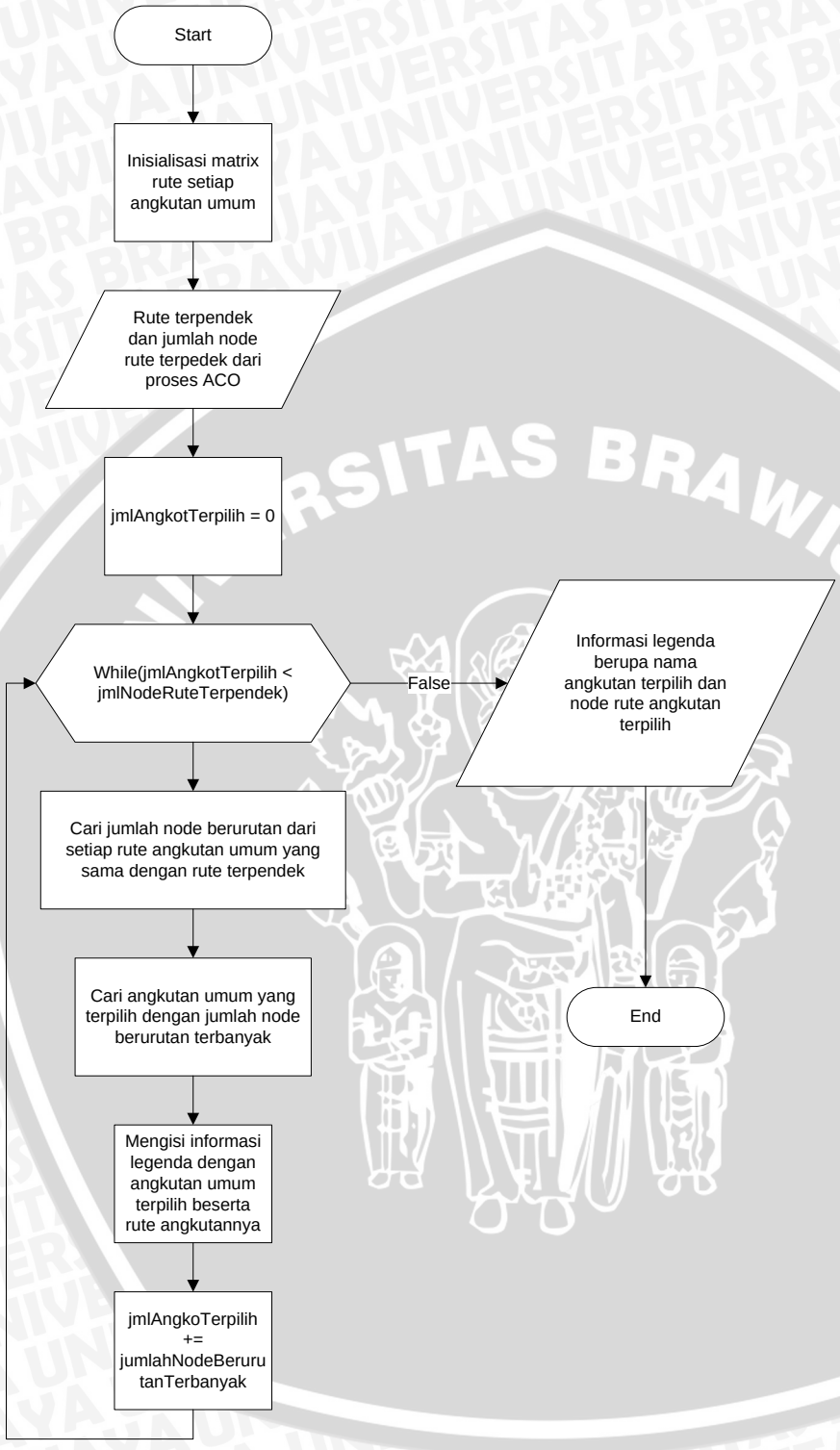
selanjutnya berdasarkan probabilitas titik selanjutnya. Setelah semua semut sudah melewati semua titik kemungkinannya dan sampai ke titik tujuan maka proses selanjutnya aplikasi akan menghitung jarak terpendek yang di dapat dari semua semut tersebut. Apabila jarak terpendek semua semut sudah didapatkan maka proses selanjutnya yaitu mengupdate nilai feromone berdasarkan titik yang dilewati oleh semut yang mencapai rute terpendek. Setelah semua siklus dilewati maka akan didapatkan total jarak terpendek dari setiap semut pada setiap siklus. Diagram alir yang menunjukkan proses pencarian rute *Ant Colony Optimization* ini ditunjukkan didalam gambar 3.13 berikut :



Gambar 3.13 Diagram alir ant colony Optimization

3.4.6.2 Brute Force

Pada tahap ini sistem akan menetukan mana angkutan yang akan terpilih berserta node terpilih pada angkutan umum tersebut. Pada permulaanya sistem ini akan menginisialisasikan node – node pada setiap angkutan umum yang terdaftar lalu dibutuhkan rute jarak terpendek yang didapat dari proses *Ant Colony Optimization.* Setelah semua data sudah didapat maka sistem akan memulai proses pencarian rute angkutan umum yang terpilih dengan cara mencoba – coba mana angkutan umum yang memiliki node yang sama terbanyak dengan rute jarak terpendek tersebut. Proses mencoba – coba yang dilakukan aplikasi ini dengan cara mencocokan node terdepan dari node rute terpendek terpendek lalu kemudian apabila node terdepan dan node rute terpendek tersebut ada yang sama sistem akan mulai menghitung berapa jumlah node yang sama tersebut dan akan di pilih angkutan umum yang memiliki node yang sama terbanyak sehingga mendapatkan jumlah angkutan umum yang paling sedikit. Diagram alir yang menunjukkan proses pencarian rute *Brute Force* ini ditunjukkan didalam gambar 3.14 berikut :



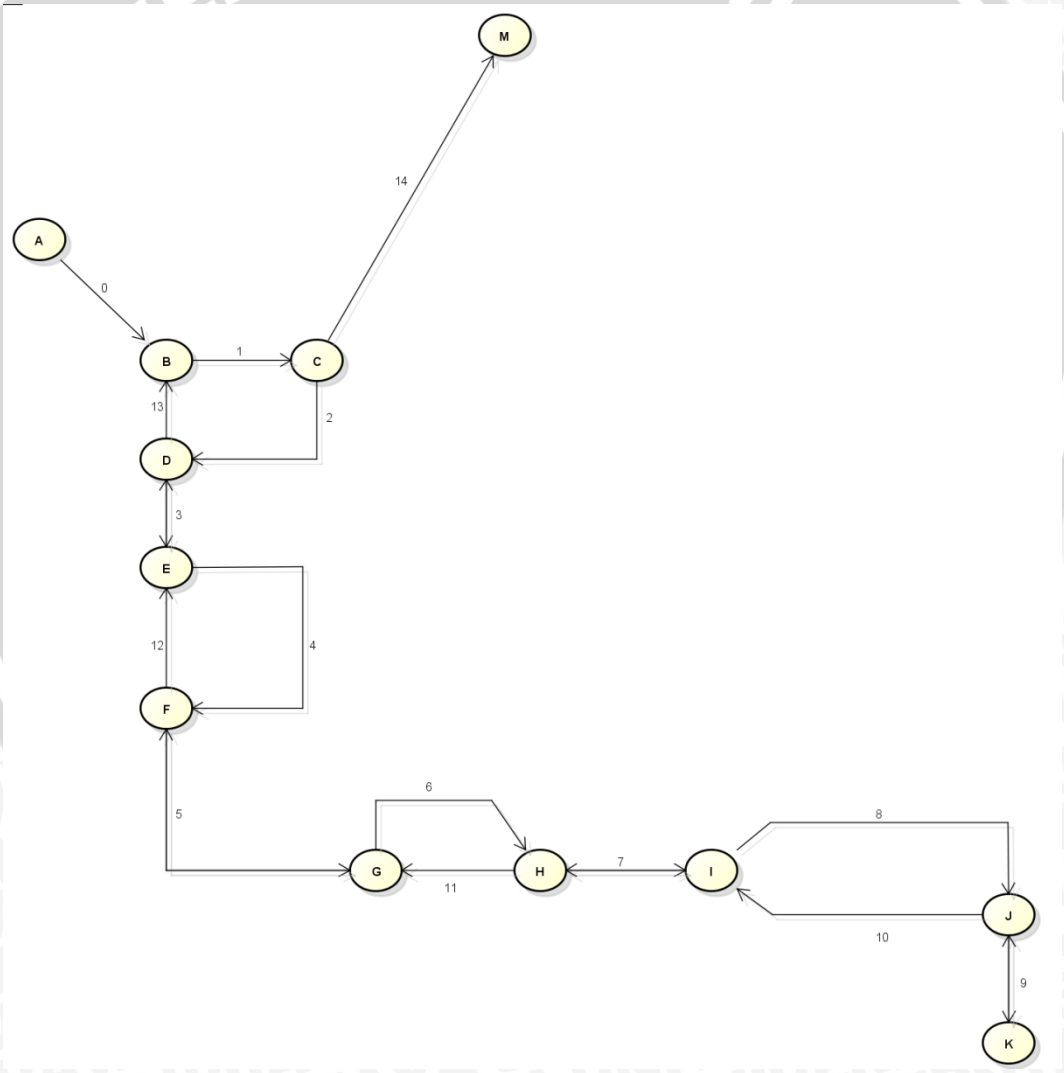
Gambar 3.14 Diagram alir brute force

3.4.7 Perhitungan Manual

Penghitungan manual berfungsi untuk memberikan gambaran umum perancangan sistem yang akan dibangun. Aplikasi pencarian rute angkutan umum ini menggunakan *Algoritma Ant Colony Optimization*. Contoh manualisasi pencarian rute terdekat pada kasus pencarian rute dari Jl. Irian Jaya menuju Jl. Doktor Cipto di kota malang dengan sampel rute angkutan umum GL dan GA satu arah adalah sebagai berikut.

3.4.7.1 Graph Rute Angkutan Umum

Gambar 3.15 merupakan graph untuk rute angkutan umum GL dan GA :



Gambar 3.15 Graph angkutan umum GL dan GA

3.4.7.2 Tahap Ant Colony Optimization

a. Tahap Inisialisasi Parameter – Parameter

Pada tahap ini sistem akan menginisialisasikan parameter – parameter yang dibutuhkan oleh *Algoritama Ant Colony Optimization*, dari parameter – parameter yang akan digunakan pada proses perhitungan ini akan dibedakan menjadi 2 yaitu parameter yang bersifat tetap dan parameter yang harus diuji. Berikut parameter – parameter yang digunakan pada proses perhitungan ini :

Parameter yang bersifat tetap :

1. Jumlah Titik = 15
2. Asal = 4
3. Tujuan = 14

Parameter yang harus di uji (disini nilai dari parameter hanya diambil sebagai sampel dan belum merupakan nilai parameter yang pasti) :

4. tijPakai = 0.5
5. a = 1
6. b = 0.99
7. rho = 0.5
8. pQ = 1
9. siklus = 15
10. semut = 15

b. Tahap Inisialisasi Matriks Jarak

Pada tahap ini sistem akan menentukan jarak yang dibuat berdasarkan hasil kompresi antar edge setiap vertex pada peta yang didapat dari sejumlah point latitude and longitude . Kompresi vertex pada peta sebelum sistem membuat matriks jarak ditunjukkan pada tabel 3.5 berikut:

Tabel 3.5 Tabel kompresi vertex

Vertex	Point Start	Point End
0	0	317
1	317	327
2	327	342
3	342	353
4	353	474
5	474	573
6	573	578
7	578	589
8	589	637
9	637	639
10	639	698
11	699	705
12	706	772
13	773	779
14	780	1077

Setelah sistem mendapatkan tabel pengandaian dari edge yang akan dirubah menjadi vertex dengan cara menghitung jarak dari point start menuju point end menggunakan fungsi distanceTo() maka sistem akan membuat sebuah matriks jarak antar vertex yang ditunjukkan pada tabel 3.6 berikut :

Tabel 3.6 Tabel matriks jarak

Matriks Jarak															
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
0	0	452													
1		0	618												8580
2			0	387										188	

3				0	3714									188	
4					0	3206							1870		
5						0	107						1870		
6							0	318				105			
7								0	1330			105			
8									0	34	1299				
9										0	1299				
10								318			0				
11						3206						0			
12				387	3714								0		
13		452												0	
14															0

c. Tahap Inisialisasi Intensitas Awal Setiap Semut

Pada tahap ini sistem akan mengisi intensitas setiap semut untuk setiap rute tujuan yang ada pada matriks jarak. Matriks intensitas awal (tijPakai) setiap semut ditunjukkan pada tabel 3.7 berikut:

Tabel 3.7 Tabel matriks intensitas jejak semut

Matriks Intensitas Jejak Semut															
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
0	0	0.5													
1		0	0.5												0.5
2			0	0.5										0.5	
3				0	0.5									0.5	
4					0	0.5							0.5		
5						0	0.5						0.5		
6							0	0.5				0.5			
7								0	0.5			0.5			
8									0	0.5	0.5				
9										0	0.5				
10								0.5			0				
11						0.5						0			
12				0.5	0.5								0		
13		0.5												0	
14															0

d. Tahap Inisialisasi Visibilitas

Dari jarak antar vertex yang sudah diketahui maka dapat dihitung visibilitas antar vertex dengan persamaan (1) yang ditunjukkan pada tabel 3.8 berikut:

Contoh perhitungan menggunakan persamaan (1) :

Visibilitas = 1/dij

[4][5] = 1/d[0][1]

[4][5] = 1/3206

[4][5] = 0.00031

Tabel 3.8 Tabel matriks visibilitas

Visibilitas															
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
0	0	0.00221													
1		0	0.00161												0.00011
2			0	0.00258										0.00531	
3				0	0.00026									0.00531	
4					0	0.00031							0.00053		
5						0	0.00934						0.00053		
6							0	0.00314				0.00952			
7								0	0.00075			0.00952			
8									0	0.02941	0.00076				
9										0	0.00076				
10								0.00314			0				
11						0.00031						0			
12				0.00258	0.00026								0		
13		0.00221												0	
14															0

e. Tahap Inisialisasi Probabilitas

Sebelum sistem dapat melakukan penyusunan kunjungan semut ke setiap vertex hal yang perlu dilakukan yaitu menentukan probabilitas untuk setiap vertex dengan menggunakan persamaan (2) dan (3) yang ditunjukkan pada tabel 3.9 berikut:

Contoh perhitungan menggunakan persamaan (2) dan (3) :

$$= \frac{\quad}{\quad}$$

$$= \frac{\quad}{\quad}$$

$$= \frac{\quad}{\quad}$$

$$= \frac{\quad}{\quad}$$

$$= 0.36965$$

Tabel 3.9 Tabel matriks probabilitas

Probabilitas															
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
0	0	1													
1		0	0.93114												0.06885
2			0	0.32854										0.67145	
3				0	0.04956									0.95043	
4					0	0.36965							0.63034		
5						0	0.94439						0		
6							0	0.25030				0.74969			
7								0	0.07491			0.92508			
8									0	0.97357	0.02642				
9										0	1				

10								1			0				
11						1						0			
12				0.90368	0.09631								0		
13		1												0	
14															0

f. Tahap Pencarian Rute Terpendek

Pada tahap ini sistem akan mulai mencari rute ke vertex tujuan dan menghitung jarak terpendeknya. Semut akan menentukan rute ke vertex selanjutnya dengan cara membentuk roulette wheel selection dari bilangan komulatif probabilitas vertex sekarang kemudian membangkitkan bilangan random dan memilih di area mana bilangan random tersebut berada. Setelah semua semut sudah berada di vertex tujuan dan menghabiskan satu siklus maka sebelum menuju siklus selanjutnya sistem akan menghitung panjang jalur tertutup (*length close tour*) atau . Perhitungan dilakukan berdasarkan masing – masing dengan persamaan (4) dan (5) yang ditunjukkan pada tabel 3.10 :

Contoh perhitungan menggunakan persamaan (4) dan (5) :

$$= d[4][12] + d[12][3] + d[3][13] + d[13][1] + d[1][14]$$

$$= 1870 + 387 + 188 + 452 + 618 + 8580$$

$$= 11470$$

Tabel 3.10 Tabel hasil pencarian rute

	Rute Node Terpendek	Jarak Tempuh
Semut 1	4 - 12 - 3 - 13 - 1 - 2 - 13	3703
Semut 2	4 - 12 - 3 - 13 - 1 - 2 - 3	3902
Semut 3	4 - 12 - 3 - 13 - 1 - 2 - 13	3703
Semut 4	4 - 12 - 3 - 13 - 1 - 14	11477
Semut 5	4 - 12 - 3 - 13 - 1 - 2 - 3	3902
Semut 6	4 - 12 - 3 - 13 - 1 - 14	11477

Semut 7	4 - 12 - 3 - 13 - 1 - 14	11477
Semut 8	4 - 12 - 3 - 13 - 1 - 14	11477
Semut 9	4 - 12 - 3 - 13 - 1 - 2 - 3	3902
Semut 10	4 - 12 - 3 - 13 - 1 - 2 - 13	3703
Semut 11	4 - 12 - 3 - 13 - 1 - 14	11477
Semut 12	4 - 12 - 3 - 13 - 1 - 14	11477
Semut 13	4 - 12 - 3 - 13 - 1 - 2 - 3	3902
Semut 14	4 - 12 - 3 - 13 - 1 - 14	11477
Semut 15	4 - 12 - 3 - 13 - 1 - 14	11477

Setelah Lk setiap semut dihitung, akan diperoleh harga minimal panjang jalur tertutup setiap siklus atau LminNC dan harga minimal panjang jalur tertutup secara keseluruhan atau Lmin. Dari daftar tabel yang ada diatas ditemukan LminNC dari vertex 4 menuju vertex 14 adalah 11477.

g. Tahap Perhitungan Perubahan Harga Intensitas Jejak Kaki Semut

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 dengan persamaan (6).

Dengan adalah perubahan harga intensitas jejak kaki semua antar vertex setiap semut yang dihitung berdasarkan persamaan (7) dan (8).

Perubahan harga intensitas jejak kaki semut untuk sekali siklus ditunjukkan pada tabel 4.1 sebagai berikut :

Contoh perhitungan menggunakan persamaan (6), (7) dan (8):

—

——

Tabel 3.11 Tabel hasil perubahan harga intensitas

	Rute Node Terpendek	Jarak Tempuh	Perubahan Harga Intentisas
Semut 1	4 - 12 - 3 - 13 - 1 - 2 - 13	3703	0
Semut 2	4 - 12 - 3 - 13 - 1 - 2 - 3	3902	0
Semut 3	4 - 12 - 3 - 13 - 1 - 2 - 13	3703	0
Semut 4	4 - 12 - 3 - 13 - 1 - 14	11477	8.71308E-05
Semut 5	4 - 12 - 3 - 13 - 1 - 2 - 3	3902	0
Semut 6	4 - 12 - 3 - 13 - 1 - 14	11477	8.71308E-05
Semut 7	4 - 12 - 3 - 13 - 1 - 14	11477	8.71308E-05
Semut 8	4 - 12 - 3 - 13 - 1 - 14	11477	8.71308E-05
Semut 9	4 - 12 - 3 - 13 - 1 - 2 - 3	3902	0
Semut 10	4 - 12 - 3 - 13 - 1 - 2 - 13	3703	0
Semut 11	4 - 12 - 3 - 13 - 1 - 14	11477	8.71308E-05
Semut 12	4 - 12 - 3 - 13 - 1 - 14	11477	8.71308E-05
Semut 13	4 - 12 - 3 - 13 - 1 - 2 - 3	3902	0
Semut 14	4 - 12 - 3 - 13 - 1 - 14	11477	8.71308E-05
Semut 15	4 - 12 - 3 - 13 - 1 - 14	11477	8.71308E-05
Total Perubahan Harga Intensitas Jejak Kaki Semut			0.000697046

h. Tahap Update Feromone

Update feromone dibutuhkan untuk perubahan harga intesitas jejak kaki semut untuk siklus selanjutnya. 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 (9).

Untuk perhitungan total harga intensitas jejak kaki semut antar kota ditunjukkan pada tabel 4.2 berikut :

Contoh perhitungan menggunakan persamaan (9) :

(i,j) vertex asal dan vertex tujuan dalam

(i,j) lainnya

Tabel 3.12 Tabel total harga intensitas

(i,j) vertex asal dan vertex tujuan dalam	0.250697046
(i,j) lainnya	0.25

Daftar tabel perubahan intensitas jejak kaki semut untuk sebuah siklusnya ditunjukkan pada tabel 4.2 :

Tabel 3.13 Tabel update feromone

Update Feromone															
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
0	0	0.25													
1		0	0.25												**0.25069**
2			0	0.25										0.25	
3				0	0.25									**0.25069**	
4					0	0.25							**0.25069**		
5						0	0.25						0.25		
6							0	0.25				0.25			
7								0	0.25			0.25			
8									0	0.25	0.25				
9										0	0.25				
10								0.25			0				
11						0.25						0			
12				**0.25069**	0.25								0		
13		**0.25069**												0	
14															0

Intensitas jejak kaki semut akan terus berubah pada setiap siklusnya sampai siklus pergerakan semut berakhir sehingga untuk (i,j) vertex asal dan vertex tujuan dalam akan memiliki probabilitas area terpiih yang lebih besar dari sebelumnya.

3.4.7.3 Tahap Brute Force

a. Tahap Inisialisasi Matriks Rute Angkutan Umum

Setelah rute terpendek didapatkan dari proses *Algoritma Ant Colony Optimization* yaitu rute turpendek dari Jl. Irian Jaya menuju Jl. Doktor Cipto di kota malang dengan vertex 4 menuju vertex 14 adalah 4 - 12 - 3 - 13 - 1 – 14.

Pada tahap ini sebelum sistem melakukan brute force pada setiap angkutan umum sistem akan menginisialisasikan daftar vertex yang tersimpan pada setiap rute angkutan umum tersebut. Daftar vertex yang

tersimpan pada setiap rute angkutan umum ditunjukkan pada tabel 4.3 berikut:

Tabel 3.14 Tabel daftar vertex setiap rute angkutan umum

Rute GL	Rute GA
0	9
1	10
2	7
3	11
4	5
5	12
6	3
7	13
8	1
9	14

b. Tahap Penentuan Rute Angkutan Umum Terpilih

Setelah maktriks rute angkutan umum selesai dibuat maka selanjutnya sistem akan mulai mencari mana rute angkutan umum yang memiliki jumlah vertex yang sama dari rute angkutan umum yang sudah di dapat dari proses pencarian rute dengan menggunakan *Algoritma Ant Colony Optimization.* Pencarian rute angkutan umum terpilih yang ditemukan oleh sistem dengan cara melakukan penelusuran ke setiap rute angkutan umum dan melakukan pencocokan dengan pattern rute angkutan umum menggunakan *Algoritma Brute Force* ditunjukkan pada tabel 4.4 berikut :

Tabel 3.15 Tabel hasil pencarian rute angkutan umum terpilih

Vertex Rute Angkutan Umum	4	12 - 3 - 13 - 1 - 14
Angkutan Umum Terpilih	GL	GA
Jumlah Vertex Terpilih Dalam Sebuah Rute Angkutan Umum	1	5

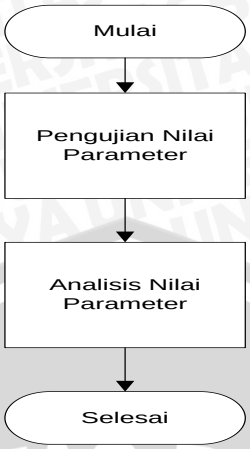
Dari hasil pencarian yang terdapat pada tabel di atas ditemukan bahwa untuk menuju Jl. Doktor Cipto dari Jl. Irian Jaya di kota malang untuk jalur 4 menggunakan angkutan umum GL dan untuk jalur 12 ke 14 menggunakan angkutan umum GA.

3.5 Implementasi

Implementasi perangkat lunak mengacu kepada hasil perancangan perangkat lunak. Pada tahap perancangan implementasi perangkat lunak diawali dengan penjabaran spesifikasi lingkungan perancangan perangkat lunak. Selanjutnya dijabarkan *use case diagram* dan *layout* saat implementasi perangkat lunak. Implementasi perangkat lunak dilakukan dengan menggunakan bahasa pemrograman Java dengan Library Software Development Kit For Android. Implementasi antarmuka berdasarkan perancangan yang telah dilakukan. Pada tahap akhir dilakukan implementasi pada *smartphone* Android secara langsung. Pemasangan aplikasi pada *smartphone* menggunakan ADB *(Android Debug Bridge).*

3.6 Pengujian dan Analisis

Pengujian perangkat lunak dilakukan untuk mengetahui apakah kinerja dan performa sistem aplikasi android telah sesuai dengan spesifikasi kebutuhan yang melandasinya. Pengujian dilakukan berdasarkan implementasi yang telah dibuat melalui perhitungan secara manual. Pengujian nilai parameter dilakukan dengan membandingkan setiap nilai parameter yang diujikan sampai mendapatkan nilai keakuratan maksimal sehingga didapatkan suatu kesimpulan. Diagram alir pengujian dan analisis ditunjukkan didalam gambar 3.16 berikut :



Gambar 3.16 Diagram alir pengujian dan analisis

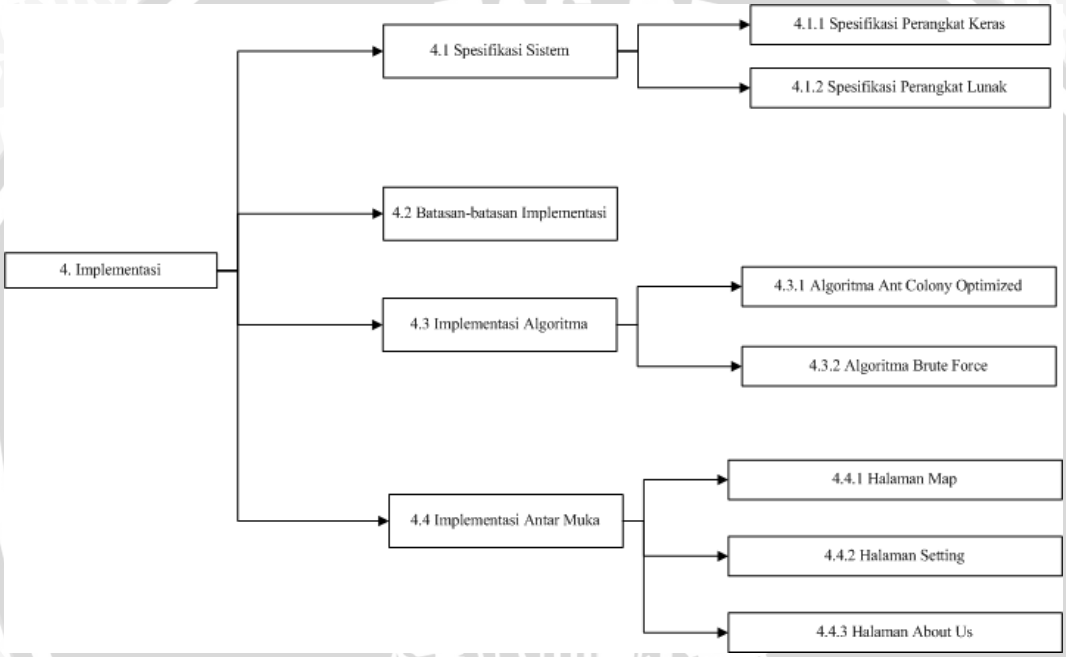
Setelah tahap pengujian, dilakukan analisis untuk mengetahui hasil dari pengujian perangkat lunak sehingga dapat didapatkan kesimpulan dari hasil rancang bangun aplikasi yang telah dibuat.

3.7 Kesimpulan dan Saran

Kesimpulan dilakukan setelah semua tahapan perancangan perangkat lunak, implementasi perangkat lunak, dan pengujian perangkat lunak telah selesai dilakukan. Kesimpulan diambil dari hasil pengujian dan analisis hasil uji terhadap sistem yang dibangun. Tahap terakhir dari penulisan adalah saran yang dimaksudkan untuk memperbaiki kesalahan–kesalahan yang terjadi dan menyempurnakan penulisan serta untuk memberikan pertimbangan atas pengembangan perangkat lunak lebih lanjut.

BAB IV
IMPLEMENTASI

Bab ini membahas mengenai implementasi perangkat lunak berdasarkan hasil yang telah didapatkan dari analisis kebutuhan dan proses perancangan perangkat lunak yang dibuat. Pembahasan terdiri dari penjelasan tentang spesifikasi sistem, batasan-batasan dalam implementasi, implementasi algoritma, dan implementasi antar muka. Tahap-tahap pembahasan implementasi yang dikerjakan digambarkan pada Gambar 4.1 berikut ini.



Gambar 4.1 Tahapan Implementasi

4.1 Spesifikasi Sistem

Hasil analisis kebutuhan yang telah diuraikan pada bab 3 menjadi acuan untuk melakukan implementasi menjadi sebuah sistem yang dapat berfungsi sesuai dengan kebutuhan. Spesifikasi sistem diimplementasikan pada spesifikasi perangkat keras dan perangkat lunak.

4.1.1 Spesifikasi Perangkat Keras

Spesifikasi perangkat keras computer untuk proses pengembangan aplikasi ini ditunjukkan pada tabel 4.1 berikut :

Tabel 4.1 Spesifikasi Perangkat Keras Pengembangan

Nama Komponen	Spesifikasi
Prosessor	Intel(R) Core(TM) i3 CPU M 370 @ 2.40GHz (4 CPUs)
Memori (RAM)	4 GB
Hardisk	320 GB

Spesifikasi perangkat keras smartphone untuk proses pengujian aplikasi ini ditunjukkan pada tabel 4.2 berikut :

Tabel 4.2 Spesifikasi Perangkat Keras Pengujian

Nama Komponen	Spesifikasi
Prosessor	Dual Core 1 GHz
RAM	1 GB
Memory	16 GB
Network	3G

4.1.2 Spesifikasi Perangkat Lunak

Pengembangan sistem menggunakan perangkat lunak dengan spesifikasi yang ditunjukkan pada tabel 4.3 berikut :

Tabel 4.3 Tabel Spesifikasi Perangkat Lunak

Nama Komponen	**Spesifikasi**
Sistem Operasi	Windows XP Professional 32 bit.
Tools pemrograman	Eclipse, Android SDK, Notepad++, XAMPP 1.7.3 dan Google Chrome

Pengujian sistem menggunakan perangkat lunak dengan spesifikasi yang ditunjukkan pada tabel 4.4 berikut :

Tabel 4.4 Tabel Spesifikasi Perangkat Lunak

Nama Komponen	Spesifikasi
Sistem Operasi	Android 4.1.2

4.2 Batasan – batasan Implementasi

Beberapa-batasan dalam mengimplementasikan sistem adalah sebagai berikut:

1. Pembuatan aplikasi pencarian rute angkutan umum ini dikerjakan dengan bahasa pemrograman *Java* dengan *Eclipse* disertai Software Development Kit for *Android*.
2. Aplikasi pembantu untuk pengumpulan data koordinat rute angkutan kedalam bentuk vertex dibuat menggunakan bahasa pemrograman *PHP*.
3. Wilayah cakupan rute angkutan umum yang digunakan terbatas di kota Malang.
4. Algoritma yang digunakan pada sistem adalah *Algoritma Ant Colony Optimization* dan *Algoritma Brute Force.*

4.3 Implementasi Algoritma

Aplikasi pancarian rute angkutan umum memiliki beberapa proses utama yang terbagi menjadi beberapa fungsi. Pada penulisan skripsi ini hanya dicantumkan algoritma dari beberapa proses saja sehingga tidak semua algoritma akan dicantumkan. Algoritma proses yang dicantumkan antara lain adalah *Algoritma Ant Colony Optimization* dan *Algoritma Brute Force*. Implementasi algoritma ini akan direpresentasikan dalam bentuk *code* dengan bahasa pemrograman Java pada Eclipse.

4.3.1 Implementasi Algoritma Ant Colony Optimization

Proses dari algoritma ini bertujuan untuk menentukan rute terpendek dari asal ke tujuan dan akan menghasilkan rute node yang paling terpendek di akhir proses.

Penjelasan Algoritma Ant Colony Optimization:

1. Inisialisasi vertex asal pada tabu list.

```
tabu[y][0][0] = layer;
tabu[y][0][1] = layer;
cobaRute = 0;
```

Gambar 4.2 Inisialisasi vertex asal

2. Proses pengisian nilai probabilitas untuk setiap vertex yang berada di dalam matrix.

```
for(i = 0; i < jmlTitik; i++){
if((matrix[layer][i] != 0) && (matrix[layer][i] !=
100000)){
      tempProbAtas = Math.pow(tij[layer][i], alpa) *
(Math.pow((1/matrix[layer][i]), beta));
      prob = tempProbAtas/tempProbBawah;
      arrTemp[jmlRWS][0] = prob;
      arrTemp[jmlRWS][1] = i;
      jmlRWS += 1;
      }
}
```

Gambar 4.3 Pengisian nilai probabilitas untuk setiap vertex

3. Proses pembuatan Roulette Wheel Selection dengan pengurutan dari probabilitas terkecil sampai terbesar.

```
for(i = 0; i < jmlRWS; i++){
      for(j = i+1; j < jmlRWS; j++){
      if(arrTemp[j][0] < arrTemp[i][0]){
            tempSort = arrTemp[i][0];
            arrTemp[i][0] = arrTemp[j][0];
            arrTemp[j][0] = tempSort;
            tempSort = arrTemp[i][1];
            arrTemp[i][1] = arrTemp[j][1];
            arrTemp[j][1] = tempSort;
            }
      }
}
for(i = 0; i < jmlRWS; i++){
      if(i == 0){
            arrRWS[i][0] = 0;
            arrRWS[i][1] = arrTemp[i][0];
            arrRWS[i][2] = arrTemp[i][1];
      }
      else{
            arrRWS[i][0] = arrRWS[i-1][1];
            arrRWS[i][1] = arrRWS[i][0] +
```

```
arrTemp[i][0];
            arrRWS[i][2] = arrTemp[i][1];
      }
}
```

Gambar 4.4 Pembuatan Roulette Wheel Selection

4. Proses pembangkitan bilangan random dan akan memilih rute selanjutnya berdasarkan berada di area mana bilangan random tersebut pada lingkaran Roulette Wheel Selection.

```
if(jmlRWS > 0){
      start = 0;
      end = arrRWS[jmlRWS-1][1];
      random = new Random().nextDouble();
      result = start + (random * (end - start));
spin = false;
i = 0;
while(spin == false){
      if((result > arrRWS[i][0]) && (result <
arrRWS[i][1])){
      int kosong = 0;
      for(j = 0; j < tempJmlIsiTabu; j++){
            if(tabu[y][j][1] == (int)arrRWS[i][2]){
                  spin = true;
                  arrKetemu[y] = "gagal";
                  ketemu = true;
                  kosong++;
            }
      }
      if(kosong == 0){
            tempJarakTerdekat =
matrix[layer][(int)arrRWS[i][2]];
            idxTempJarakTerdekat = (int)arrRWS[i][2];
            tempJmlIsiTabu++;
            tabu[y][tempJmlIsiTabu][0] = layer;
            tabu[y][tempJmlIsiTabu][1] =
idxTempJarakTerdekat;
            layer = idxTempJarakTerdekat;
            spin = true;
            if(idxTempJarakTerdekat == tujuan){
                  arrKetemu[y] = "berhasil";
                  ketemu = true;
            }
      }
      }
      i++;
      }
      layer = idxTempJarakTerdekat;
}
```

Gambar 4.5 Pembangkitan bilangan random untuk memilih rute selanjutnya

5. Proses pencarian jarak terpendek dari perjalanan semua semut.

```
for(i = 0; i < jmlSemut; i++){
      tempJarakTempuh = 0;
      for(j = 0; j <= jmlTabu[i]; j++){
            tempJarakTempuh +=
matrix[tabu[i][j][0]][tabu[i][j][1]];
      }
      if(arrKetemu[i].equals("berhasil")){
            if(tempJarakTempuh < jarakTempuh){
                  jarakTempuh = tempJarakTempuh;
                  idxTempJarakTempuh = i;
            }
      }
}
```

Gambar 4.6 Pencarian jarak terpendek dari semua semut

6. Proses perhitungan harga intensitas seluruh semut yang berhasil sampai ke vertex tujuan.

```
for(i = 0; i < jmlSemut; i++){
      if(arrKetemu[i] == "berhasil"){
            tempJarakTempuh = 0;
            for(j = 0; j <= jmlTabu[i]; j++){
                  tempJarakTempuh +=
matrix[tabu[i][j][0]][tabu[i][j][1]];
            }
            if(tempJarakTempuh == jarakTempuh){
                  himpunTij += pQ/jarakTempuh;
            }
      }
}
```

Gambar 4.7 Perhitungan harga intensitas semut yang berhasil sampai ketujuan

7. Proses perhitungan harga intensitas jejak kaki semut untuk siklus selanjutnya.

```
for(i = 0; i < jmlTitik; i++){
      for(j = 0; j < jmlTitik; j++){
            tij[i][j] = rho * tijPakai + 0;
            if(idxTempJarakTempuh > -1){
                  for(k = 0; k <=
jmlTabu[idxTempJarakTempuh]; k++){
                        if(i ==
tabu[idxTempJarakTempuh][k][0] && j ==
```

```
tabu[idxTempJarakTempuh][k][1]){
                                tij[i][j] = rho *
tijPakai + himpunTij;
                            }
                        }
                    }
                }
}
```

Gambar 4.8 Perhitungan harga intensitas untuk siklus selanjutnya

8. Proses pemilihan jarak terpendek untuk seluruh siklus dari semua semut yang berhasil sampai ke tujuan.

```
if(jarakTempuh < jalurTerpedekSemuaSiklus &&
idxTempJarakTempuh != -1){
     jalurTerpedekSemuaSiklus = jarakTempuh;
          for(k = 0; k <=
jmlTabu[idxTempJarakTempuh]; k++){
               ruteAkhir[k][0] =
tabu[idxTempJarakTempuh][k][0];
               ruteAkhir[k][1] =
tabu[idxTempJarakTempuh][k][1];
          }
     jmlRuteAkhir = jmlTabu[idxTempJarakTempuh];
}
```

Gambar 4.9 Pemilihan jarak terpendek untuk seluruh siklus

4.3.2 Implementasi Algoritma Brute Force

Setelah rute terpendek dari vertex asal ke vertex tujuan telah didapatkan dari proses yang di lakukan oleh Algoritma Ant Colony Optimization selanjutnya rute tersebut akan olah sehingga di dapatkan rute angkutan apa yang akan digunakan melalui proses yang ada di Algoritma Brute Force.

Penjelasan Algoritma Brute Force:

1. Proses pemindahan rute terpendek yang berada pada array kesebuah string.

```
for(k = 0; k <= jmlRuteAkhir; k++){
     tempRuteAkhir += String.valueOf(ruteAkhir[k][1]);
     if(k < jmlRuteAkhir)
          tempRuteAkhir += "_";
}
```

Gambar 4.10 Pemindahan rute terpendek

2. Proses penginisialisasian nilai dari setiap rute angkutan umum.

```
for(i = 0; i < 5; i++){
      arrRuteAngkotTerdekat[i][0][0] = -1;
}
```

Gambar 4.11 Penginisialisasian nilai setiap rute angkutan umum

3. Proses pencarian jumlah vertex dari rute terpendek terbanyak yang berurutan dari setiap vertex angkutan umum.

```
for(i = 0; i < 5; i++){
      tempJmlPerAngkot = 0;
      int tempCariDepan = tempAngkotTerpilih;
      for(j = 0; j < jmlNodeAngkot[i]; j++){
            for(k = tempAngkotTerpilih; k <=
jmlRuteAkhir; k++){
                  if((ruteAkhir[k][1] ==
ruteAngkot[i][j]) && (k == tempCariDepan)){
                        tempCariDepan++;
                        tempJmlPerAngkot++;
                  }
            }
      }
      jmlPerAngkot[i] = tempJmlPerAngkot;
}
```

Gambar 4.12 Pencarian jumlah vertex dari rute terpendek

4. Proses pemilihan rute angkutan umum berdasarkan jumlah vertex dari rute terpendek yang paling banyak dari seluruh angkutan umum.

```
for(i = 0; i < 5; i++){
      if(jmlPerAngkot[i] > tempPjgAngkotTerpilih){
            tempPjgAngkotTerpilih = jmlPerAngkot[i];
            idxAngkot = i;
      }
}
```

Gambar 4.13 Pemilihan rute angkutan umum

5. Proses pengisian vertex rute terpendek untuk setiap rute angkutan umum yang terpilih tersebut.

```
for(i = tempAngkotTerpilih; i <
(tempAngkotTerpilih+tempPjgAngkotTerpilih); i++){
     arrRuteAngkotTerdekat[countCariAngkot][1][x] =
ruteAkhir[i][1];
     x++;
}
```

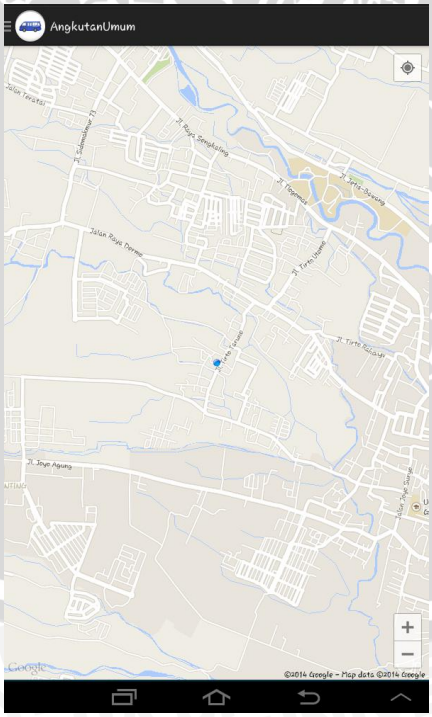
Gambar 4.14 Pengisian vertex rute angkutan umum yang terpilih

4.4 Implementasi Antar Muka

Antar muka pada aplikasi pencarian rute angkutan umum ini digunakan untuk memudahkan user berinter aksi dengan sistem. Antar muka pada aplikasi ini dibagi menjadi 3 yaitu halaman map, halaman settings dan halaman help.

4.4.1 Halaman Map

Halaman map merupakan halaman utama dari aplikasi ini yang menjadi tempat dimana semua proses pencarian rute dilakukan. Gambar 4.4 menunjukkan implementasi dari halaman map.



Gambar 4.15 Implementasi Halaman Map

4.4.2 Halaman Settings

Halaman settings digunakan untuk menentukan default pengaturan aplikasi ini. Gambar 4.5 menunjukkan implementasi dari halaman settings.

Gambar 4.16 Implementasi Halaman Settings

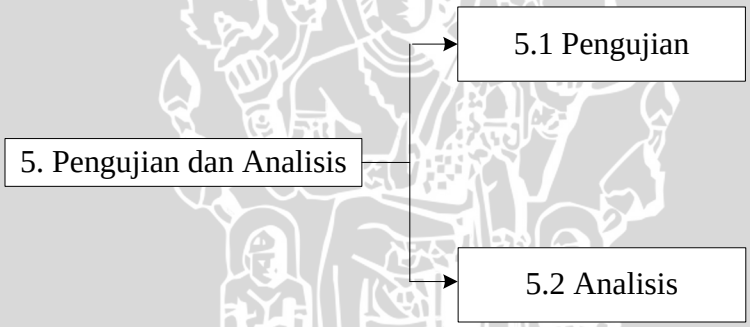
4.4.3 Halaman Help

Halaman help digunakan untuk menampilkan sedikit informasi serta cara – cara penggunaan dari aplikasi ini. Gambar 4.6 menunjukkan implementasi dari halaman help.

Gambar 4.17 Implementasi Halaman Help

BAB V
PENGUJIAN DAN ANALISIS

Bab ini membahas mengenai tahapan pengujian dan analisis Aplikasi pencarian rute angkutan umum menggunakan Algoritma Ant Colony Optimization. Proses pengujian dilakukan dengan menentukan nilai parameter yang paling maksimal dari Algoritma Ant Colony Optimization. Nilai parameter yang akan di uji pada pengujian ini adalah nilai α, , ρ, m dan NcMax. Untuk nilai β dan Q tidak akan di ujikan karena nilai β di set dengan nilai 0.0000000001 dan Q di set dengan nilai 1. Parameter β yang bernilai 0.0000000001 digunakan karena untuk mendapatkan signifikan jarak yang sama untuk setiap vertex di dalam matrix dan nilai Q tidak akan di uji karena pada penelitian [AMV-91] menyatakan bahwa perubahan nilainya tidak akan mendapatkan hasil yang signifikan.



Gambar 5.1 Tahapan Pengujian dan Analisis

5.1 Pengujian

Pada pengujian ini ada beberapa nilai yang akan diujikan untuk setiap parameternya. Untuk parameter α nilai yang akan diujikan yaitu 0, 0.5, 1, 2, 5 dan untuk parameter nilai yang akan diujikan yaitu 0.000005, 0.5, 5 dan untuk parameter ρ nilai yang akan diujikan yaitu 0.1, 0.5, 0.9 [DOR-04]. Parameter NcMax dan m disini akan diisikan nilai kombinasi yaitu 500 & 50, 500 & 25, 500 & 10, 100 & 100, 100 & 50 serta 100 & 25. Dalam menentukan nilai optimal untuk setiap parameter Algoritma Ant Colony Optimization digunakan nilai tetap yang terdiri atas α, β, , ρ, Q, NcMax dan m yaitu 0.5, 0.0000000001, 0.5, 0.5, 1, 500,

25 untuk melihat perbedaan hasil dari tiap – tiap nilai parameter yang akan di ujikan. Setiap nilai parameter akan diujikan masing – masing sebanyak 10 kali untuk jarak dekat (edge 0 – edge 9, lihat lampiran 2) pada percarian rute terpendek dan termurah dan masing – masing sebanyak 10 kali untuk jarak jauh (edge 0 – edge 18, lihat lampiran 2) pada percarian rute terpendek dan termurah.

5.1.1 Pengujian Parameter α

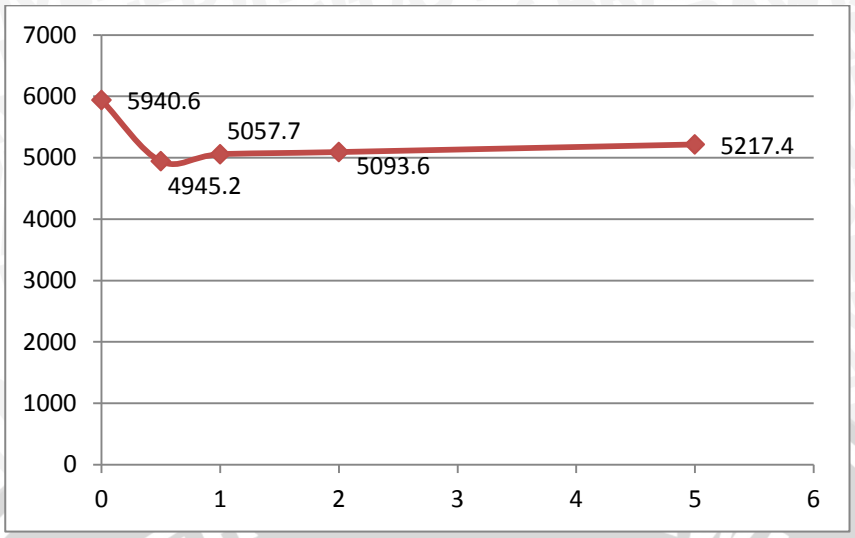
Parameter α yang paling optimal didapatkan dari rata – rata runtime paling kecil dan rata – rata menemukan tujuan paling besar, karena semakin kecil rata – rata runtime menandakan semakin sering nilai α tersebut menghasilkan waktu proses yang paling kecil dan semakin besar rata – rata menemukan tujuan menandakan semakin sering nilai α tersebut menghasilkan solusi menemukan tujuan paling banyak. Rata – rata runtime dan rata – rata menemukan tujuan yang dihasilkan oleh parameter α dapat dilihat pada tabel 5.1 dan 5.2.

Tabel 5.1 Tabel hasil pengujian parameter α jarak dekat

Nilai α	Rata – rata runtime (milidetik)	Rata – rata menemukan tujuan
0	5940.6	100%
0.5	4945.2	100%
1	5057.7	100%
2	5093.6	100%
5	5217.4	100%

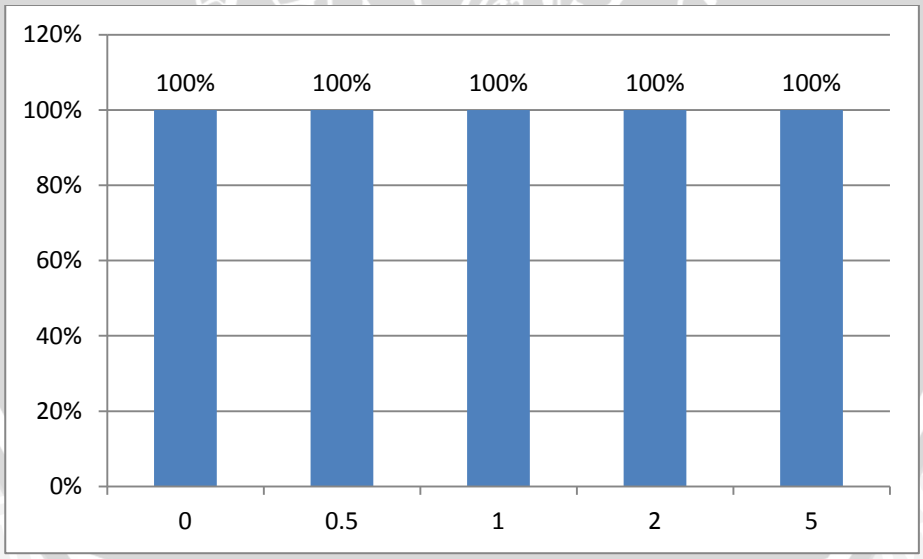
Semua nilai α mendapatkan hasil rata – rata menemukan tujuan sebesar 100% karena semua nilai dapat menemukan tujuan dari 10 kali percobaan untuk setiap masing – masing nilai α yang diberikan.

Dalam gambar 5.2 dapat dilihat grafik hubungan parameter α dengan rata – rata runtime untuk jarak dekat.



Gambar 5.2 Grafik hubungan parameter α dengan rata – rata runtime jarak dekat

Dalam gambar 5.3 dapat dilihat tingkat keberhasilan parameter α dengan rata – rata menemukan tujuan untuk jarak dekat.



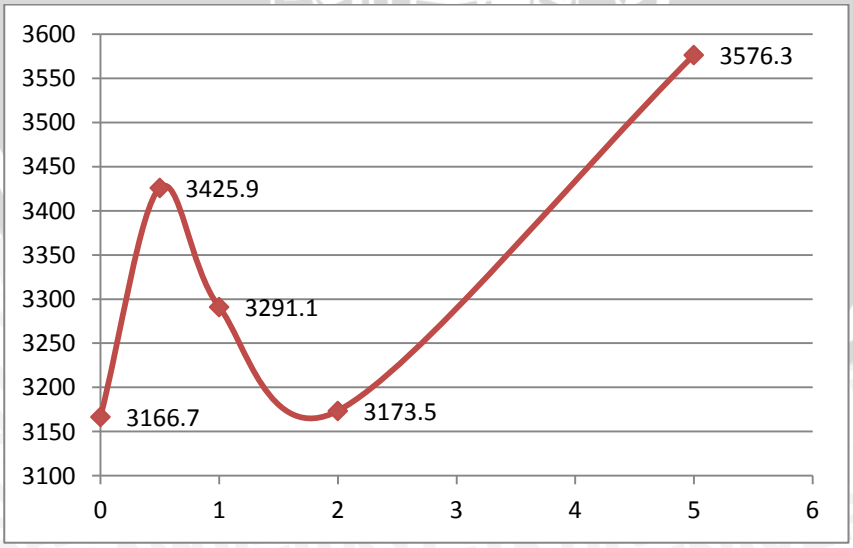
Gambar 5.3 Tingkat keberhasilan parameter α dengan rata – rata menemukan tujuan jarak dekat

Tabel 5.2 Tabel hasil pengujian parameter α jarak jauh

Nilai α	Rata – rata runtime (milidetik)	Rata – rata menemukan tujuan
0	3166.7	90%
0.5	3425.9	90%
1	3291.1	90%
2	3173.5	80%
5	3576.3	100%

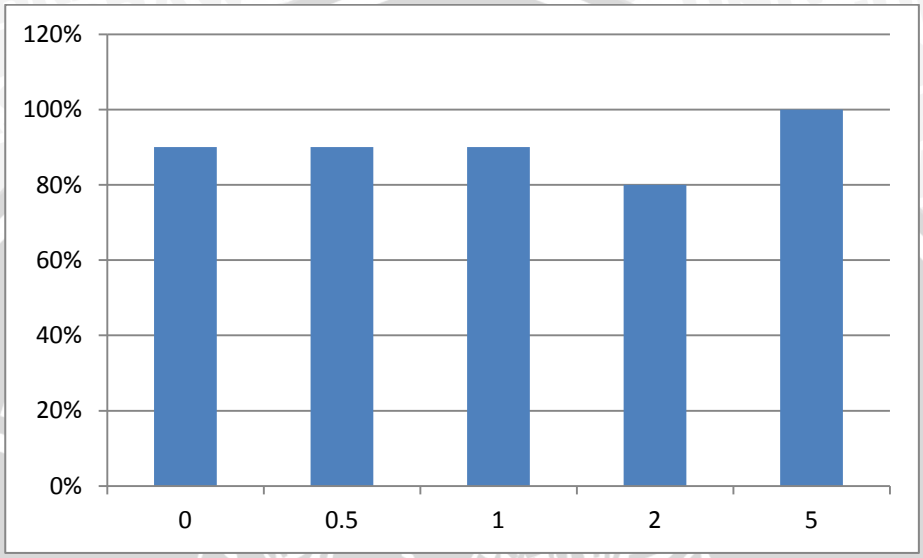
Dari setiap nilai α yang diberikan untuk masing – masing 10 kali percobaan ditunjukkan bahwa untuk nilai α = 0 menghasilkan rata – rata menemukan tujuan 90% karena ada 1 percobaan yang gagal menemukan tujuan, nilai α = 0.5 menghasilkan rata – rata menemukan tujuan 90% karena ada 1 percobaan yang gagal menemukan tujuan, nilai α = 1 menghasilkan rata – rata menemukan tujuan 90% karena ada 1 percobaan yang gagal menemukan tujuan, nilai α = 2 menghasilkan rata – rata menemukan tujuan 80% karena ada 2 percobaan yang gagal menemukan tujuan dan nilai α = 5 menghasilkan rata – rata menemukan tujuan 100% karena semua percobaan berhasil menemukan tujuan.

Dalam gambar 5.4 dapat dilihat grafik hubungan parameter α dengan rata – rata runtime untuk jarak jauh.



Gambar 5.4 Grafik hubungan parameter α dengan rata – rata runtime jarak jauh

Dalam gambar 5.5 dapat dilihat tingkat keberhasilan parameter α dengan rata – rata menemukan tujuan untuk jarak dekat.



Gambar 5.5 Tingkat keberhasilan parameter α dengan rata – rata menemukan tujuan jarak jauh

5.1.2 Pengujian Parameter

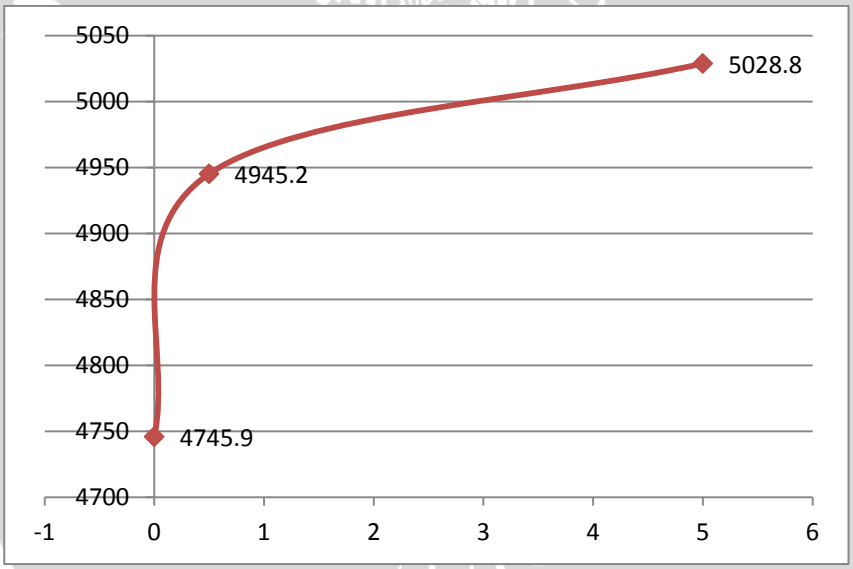
Parameter yang paling optimal didapatkan dari rata – rata runtime paling kecil dan rata – rata menemukan tujuan paling besar, karena semakin kecil rata – rata runtime menandakan semakin sering nilai tersebut menghasilkan waktu proses yang paling kecil dan semakin besar rata – rata menemukan tujuan menandakan semakin sering nilai tersebut menghasilkan solusi menemukan tujuan paling banyak. Rata – rata runtime dan rata – rata menemukan tujuan yang dihasilkan oleh parameter dapat dilihat pada tabel 5.3 dan 5.4.

Tabel 5.3 Tabel hasil pengujian parameter jarak dekat

Nilai	Rata – rata runtime (milidetik)	Rata – rata menemukan tujuan
0.000005	4745.9	100%
0.5	4945.2	100%
5	5028.8	100%

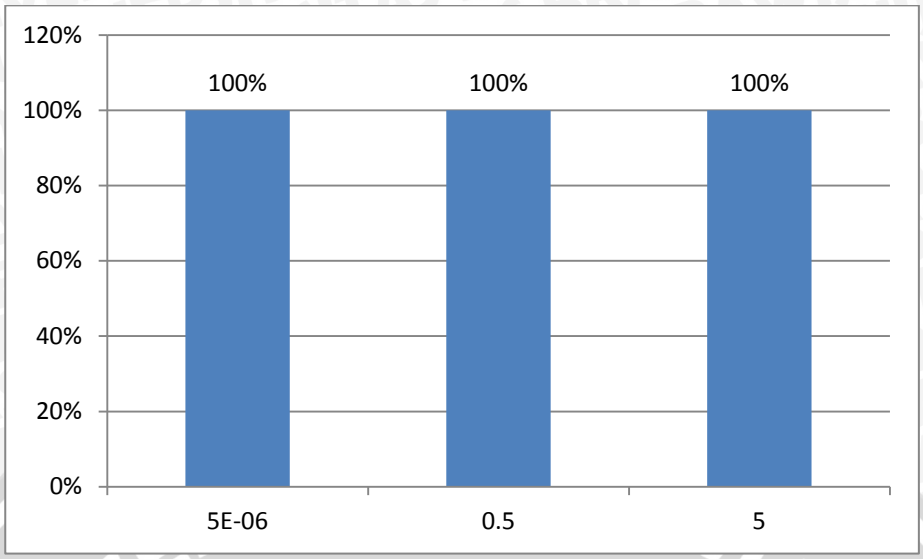
Semua nilai mendapatkan hasil rata – rata menemukan tujuan sebesar 100% karena semua nilai dapat menemukan tujuan dari 10 kali percobaan untuk setiap masing – masing nilai yang diberikan.

Dalam gambar 5.6 dapat dilihat grafik hubungan parameter dengan rata – rata runtime untuk jarak dekat.



Gambar 5.6 Grafik hubungan parameter dengan rata – rata runtime jarak dekat

Dalam gambar 5.7 dapat dilihat tingkat keberhasilan parameter dengan rata – rata menemukan tujuan untuk jarak dekat.



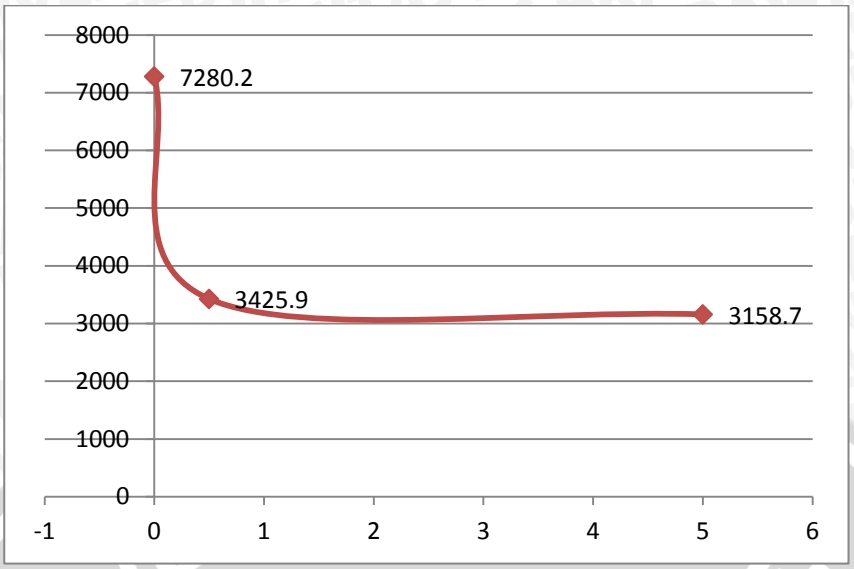
Gambar 5.7 Tingkat keberhasilan parameter dengan rata – rata menemukan tujuan jarak dekat

Tabel 5.3 Tabel hasil pengujian parameter jarak jauh

Nilai	Rata – rata runtime (milidetik)	Rata – rata menemukan tujuan
0.000005	7280.2	100%
0.5	3425.9	90%
5	3158.7	100%

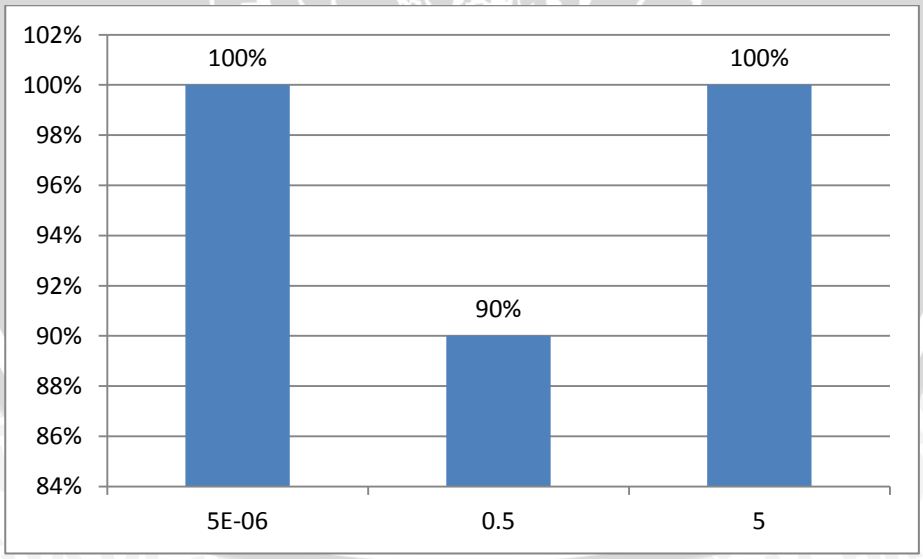
Dari setiap nilai yang diberikan untuk masing – masing 10 kali percobaan ditunjukkan bahwa untuk nilai = 0.000005 menghasilkan rata – rata menemukan tujuan 100% karena semua percobaan berhasil menemukan tujuan, nilai = 0.5 menghasilkan rata – rata menemukan tujuan 90% karena ada 1 percobaan yang gagal menemukan tujuan dan nilai = 5 menghasilkan rata – rata menemukan tujuan 100% karena semua percobaan berhasil menemukan tujuan.

Dalam gambar 5.8 dapat dilihat grafik hubungan parameter dengan rata – rata runtime untuk jarak jauh.



Gambar 5.8 Grafik hubungan parameter dengan rata – rata runtime jarak jauh

Dalam gambar 5.9 dapat dilihat tingkat keberhasilan parameter dengan rata – rata menemukan tujuan untuk jarak jauh.



Gambar 5.9 Tingkat keberhasilan parameter dengan rata – rata menemukan tujuan jarak jauh

5.1.3 Pengujian Parameter ρ

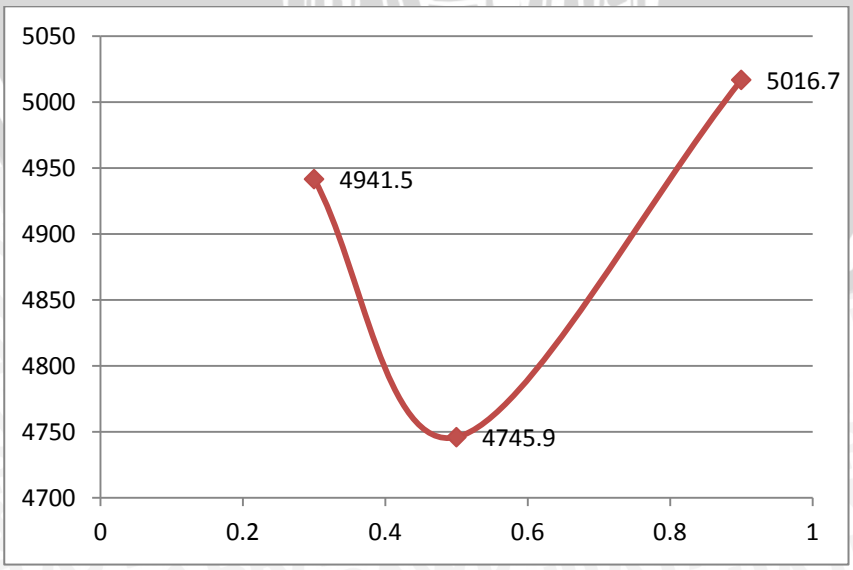
Parameter ρ yang paling optimal didapatkan dari rata – rata runtime paling kecil dan rata – rata menemukan tujuan paling besar, karena semakin kecil rata – rata runtime menandakan semakin sering nilai ρ tersebut menghasilkan waktu proses yang paling kecil dan semakin besar rata – rata menemukan tujuan menandakan semakin sering nilai ρ tersebut menghasilkan solusi menemukan tujuan paling banyak. Rata – rata runtime dan rata – rata menemukan tujuan yang dihasilkan oleh parameter ρ dapat dilihat pada tabel 5.5 dan 5.6.

Tabel 5.5 Tabel hasil pengujian parameter ρ jarak dekat

Nilai ρ	Rata – rata runtime (milidetik)	Rata – rata menemukan tujuan
0.3	4941.5	100%
0.5	4745.9	100%
0.9	5016.7	100%

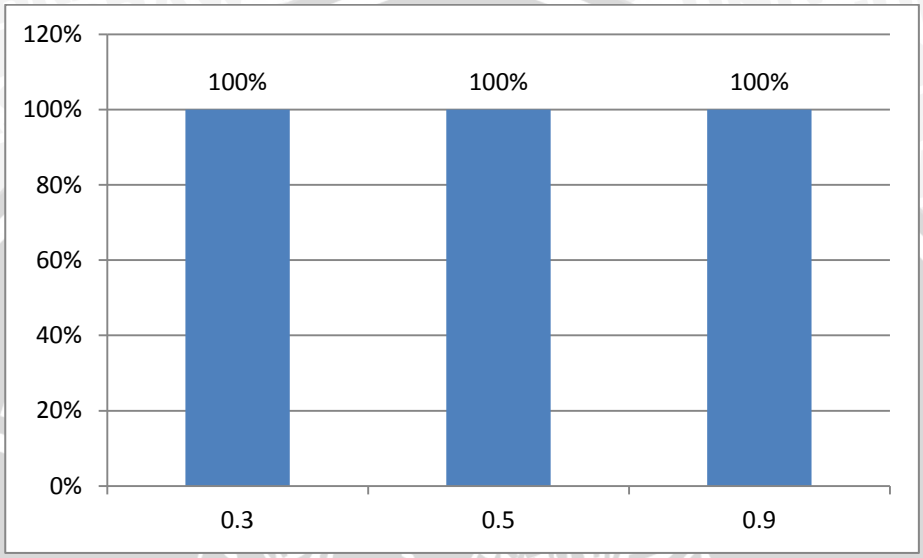
Semua nilai ρ mendapatkan hasil rata – rata menemukan tujuan sebesar 100% karena semua nilai dapat menemukan tujuan dari 10 kali percobaan untuk setiap masing – masing nilai ρ yang diberikan.

Dalam gambar 5.10 dapat dilihat grafik hubungan parameter ρ dengan rata – rata runtime untuk jarak dekat.



Gambar 5.10 Grafik hubungan parameter ρ dengan rata – rata runtime jarak dekat

Dalam gambar 5.11 dapat dilihat tingkat keberhasilan parameter ρ dengan rata – rata menemukan tujuan untuk jarak dekat.



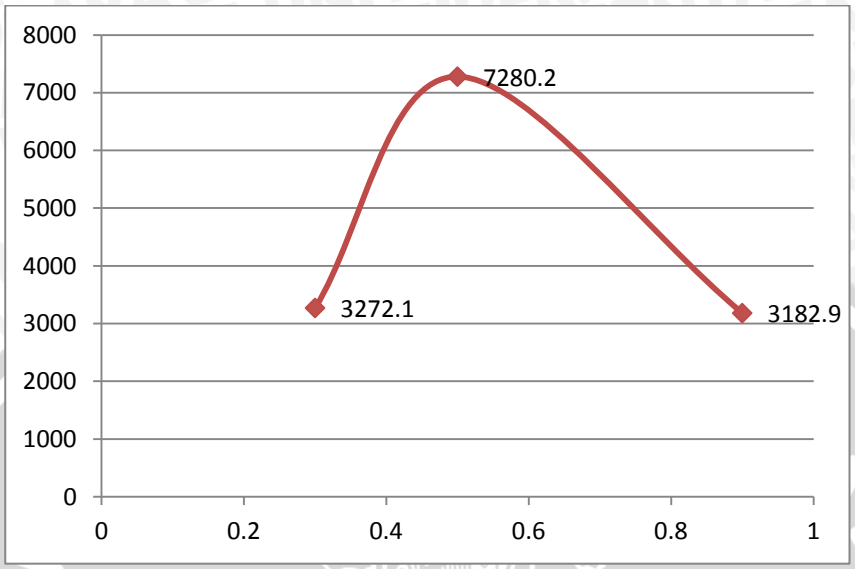
Gambar 5.11 Tingkat keberhasilan parameter ρ dengan rata – rata menemukan tujuan jarak dekat

Tabel 5.6 Tabel hasil pengujian parameter ρ jarak jauh

Nilai ρ	Rata – rata runtime (milidetik)	Rata – rata menemukan tujuan
0.3	3272.1	90%
0.5	7280.2	100%
0.9	3182.9	100%

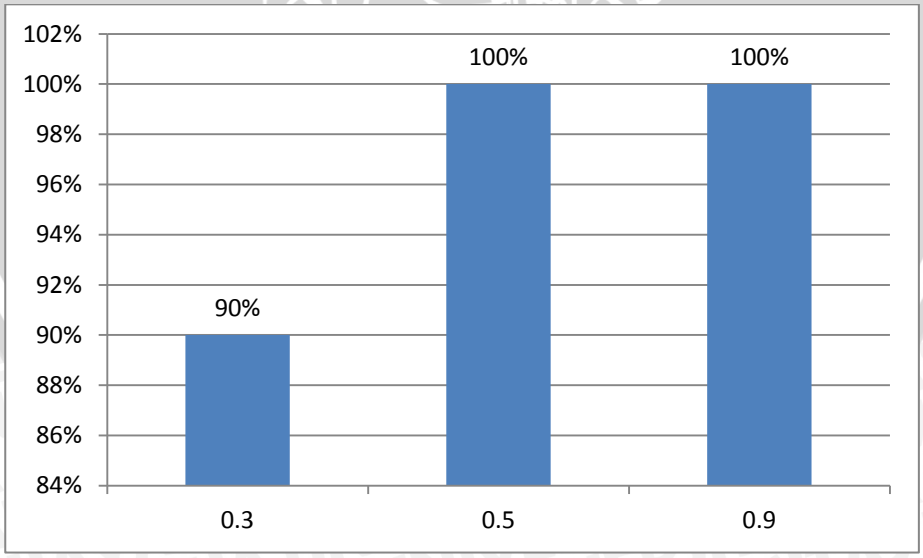
Dari setiap nilai ρ yang diberikan untuk masing – masing 10 kali percobaan ditunjukkan bahwa untuk nilai ρ = 0.3 menghasilkan rata – rata menemukan tujuan 90% karena ada 1 percobaan yang gagal menemukan tujuan dan nilai ρ = 5 atau nilai ρ = 0.9 sama – sama menghasilkan rata – rata menemukan tujuan 100% karena semua percobaan berhasil menemukan tujuan.

Dalam gambar 5.12 dapat dilihat grafik hubungan parameter ρ dengan rata – rata runtime untuk jarak jauh.



Gambar 5.12 Grafik hubungan parameter ρ dengan rata – rata runtime jarak jauh

Dalam gambar 5.13 dapat dilihat tingkat keberhasilan parameter ρ dengan rata – rata menemukan tujuan untuk jarak dekat.



Gambar 5.13 Tingkat keberhasilan parameter ρ dengan rata – rata menemukan tujuan jarak jauh

5.1.4 Pengujian Parameter *NcMax* dan *m*

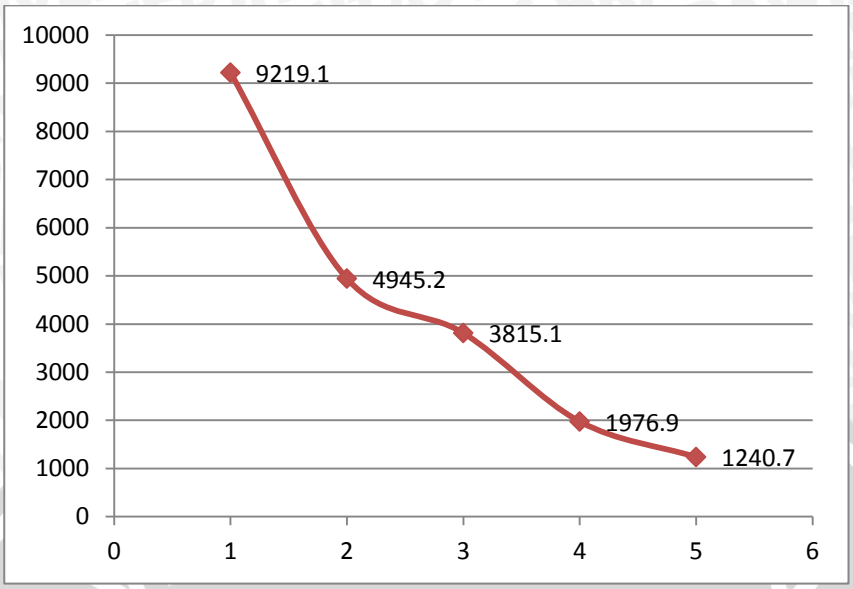
Parameter *NcMax* dan *m* yang paling optimal didapatkan dari rata – rata runtime paling kecil dan rata – rata menemukan tujuan paling besar, karena semakin kecil rata – rata runtime menandakan semakin sering nilai *NcMax* dan *m* tersebut menghasilkan waktu proses yang paling kecil dan semakin besar rata – rata menemukan tujuan menandakan semakin sering nilai *NcMax* dan *m* tersebut menghasilkan solusi menemukan tujuan paling banyak. Rata – rata runtime dan rata – rata menemukan tujuan yang dihasilkan oleh parameter *NcMax* dan *m* dapat dilihat pada tabel 5.7 dan 5.8.

Tabel 5.5 Tabel hasil pengujian parameter *NcMax* dan *m* jarak dekat

No	Nilai *NcMax* dan *m*	Rata – rata runtime (milidetik)	Rata – rata menemukan tujuan
1	500 dan 50	9219.1	100%
2	500 dan 25	4945.2	100%
3	100 dan 100	3815.1	100%
4	100 dan 50	1976.9	100%
5	100 dan 25	1240.7	100%

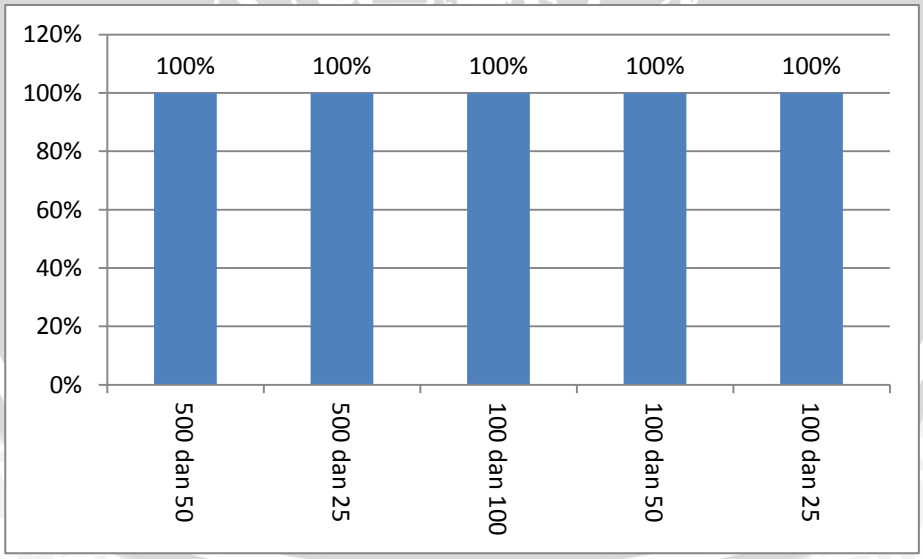
Semua nilai NcMax dan m mendapatkan hasil rata – rata menemukan tujuan sebesar 100% karena semua nilai dapat menemukan tujuan dari 10 kali percobaan untuk setiap masing – masing nilai NcMax dan m yang diberikan.

Dalam gambar 5.14 dapat dilihat grafik hubungan parameter *NcMax* dan *m* dengan rata – rata runtime untuk jarak dekat.



Gambar 5.14 Grafik hubungan parameter *NcMax* dan *m* dengan rata – rata runtime jarak dekat

Dalam gambar 5.15 dapat dilihat tingkat keberhasilan parameter *NcMax* dan *m* dengan rata – rata menemukan tujuan untuk jarak dekat.



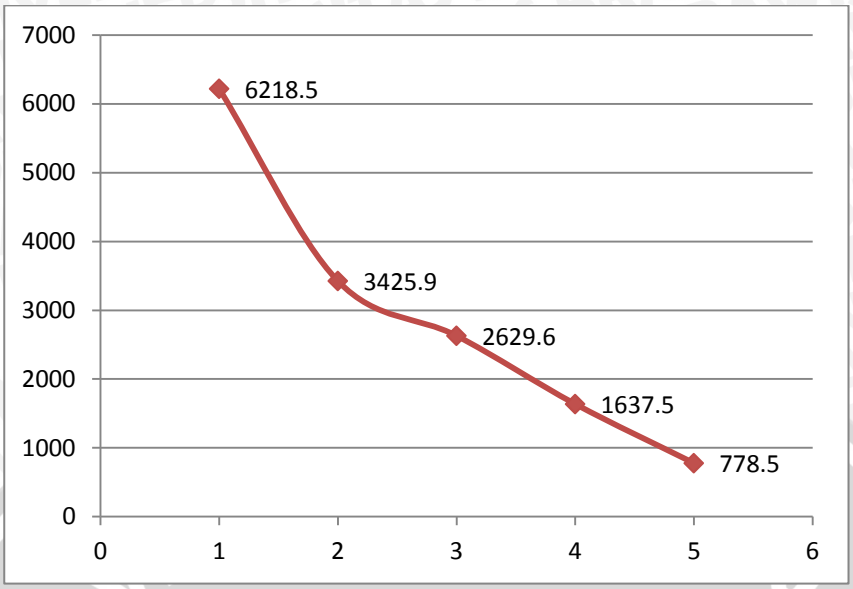
Gambar 5.15 Tingkat keberhasilan parameter *NcMax* dan *m* dengan rata – rata menemukan tujuan jarak dekat

Tabel 5.8 Tabel hasil pengujian parameter *NcMax* dan *m* jarak jauh

No	Nilai *NcMax dan m*	Rata – rata runtime (milidetik)	Rata – rata menemukan tujuan
1	500 dan 50	6218.5	100%
2	500 dan 25	3425.9	90%
3	100 dan 100	2629.6	70%
4	100 dan 50	1637.5	50%
5	100 dan 25	778.5	30%

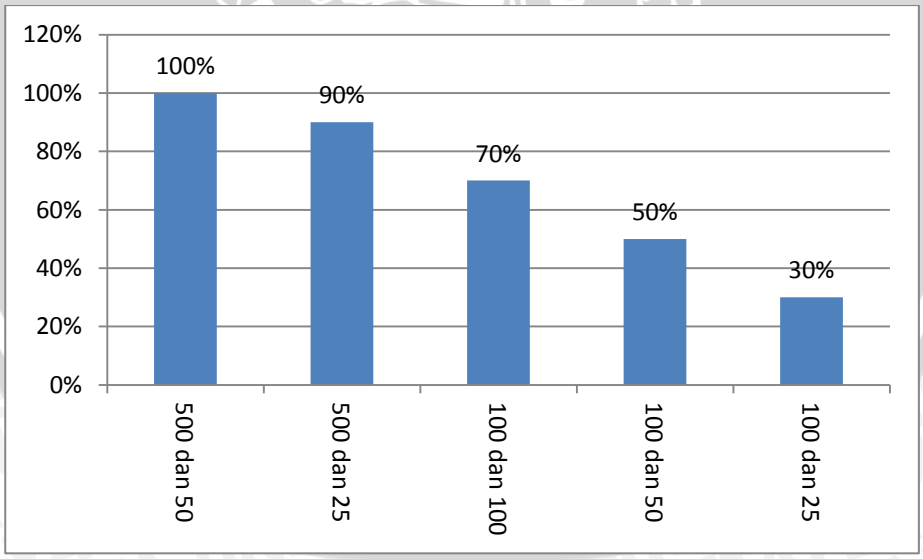
Dari setiap nilai NcMax dan m yang diberikan untuk masing – masing 10 kali percobaan ditunjukkan bahwa untuk nilai NcMax = 500 dan m = 50 menghasilkan rata – rata menemukan tujuan 100% karena semua percobaan berhasil menemukan tujuan, nilai NcMax = 500 dan m = 25 menghasilkan rata – rata menemukan tujuan 90% karena ada 1 percobaan yang gagal menemukan tujuan, nilai NcMax = 100 dan m = 100 menghasilkan rata – rata menemukan tujuan 70% karena ada 3 percobaan yang gagal menemukan tujuan, nilai NcMax = 100 dan m = 50 menghasilkan rata – rata menemukan tujuan 50% karena ada 5 percobaan yang gagal menemukan tujuan, nilai NcMax = 100 dan m = 25 menghasilkan rata – rata menemukan tujuan 30% karena ada 7 percobaan yang gagal menemukan tujuan.

Dalam gambar 5.16 dapat dilihat grafik hubungan parameter *NcMax* dan *m* dengan rata – rata runtime untuk jarak jauh.



Gambar 5.16 Grafik hubungan parameter *NcMax* dan *m* dengan rata – rata runtime jarak jauh

Dalam gambar 5.17 dapat dilihat tingkat keberhasilan parameter *NcMax* dan *m* dengan rata – rata menemukan tujuan untuk jarak dekat.



Gambar 5.17 Tingkat keberhasilan parameter *NcMax* dan *m* dengan rata – rata menemukan tujuan jarak jauh

5.2 Analisis

Proses analisis bertujuan untuk mendapatkan kesimpulan dari hasil pengujian nilai parameter – parameter *Algoritma Ant Colony Optimization* yang telah dilakukan. Analisis dilakukan terhadap hasil pengujian di setiap parameter – parameter uji yang telah ditentukan. Proses analisis yang dilakukan untuk mendapatkan nilai parameter yang mendapatkan keberhasilan menemukan tujuan paling besar dan rata – rata runtime paling kecil.

5.2.1 Analisis Parameter α

Hasil pengujian untuk parameter α pada pengujian jarak dekat semua nilai menujukkan rata – rata menemukan tujuan sebesar 100% akan tetapi pada nilai α = 0.5 menghasilkan rata – rata runtime yang paling kecil yaitu sebesar 4945.2 sedangkan pada pengujian jarak jauh nilai α = 5 menghasilkan rata – rata menemukan tujuan paling besar yaitu 100% dan pada nilai α = 5 tersebut memiliki nilai rata – rata runtime yang paling besar yaitu 3576.3. Pada pengujian ini nilai parameter yang paling baik digunakan ialah nilai parameter yang mampu menemukan tujuan yang paling baik dan nilai runtime yang paling kecil akan tetapi yang lebih di prioritaskan pada pengujian ini adalah menemukan nilai parameter yang mampu menemukan tujuan yang paling baik, jadi meskipun pada pengujian jarak dekat nilai α = 0.5 menghasilkan rata – rata runtime yang paling kecil sedangkan pada pengujian jarak jauh nilai α = 5 menghasilkan rata – rata runtime yang paling besar dari pada nilai α lainnya maka hal tersebut akan diabaikan karena pada pengujian ini nilai parameter yang paling baik lebih memprioritaskan nilai parameter yang mampu menghasilkan rata – rata menemukan tujuan yang paling baik.

5.2.2 Analisis Parameter

Hasil pengujian untuk parameter pada pengujian jarak dekat semua nilai menunjukkan rata – rata menemukan tujuan sebesar 100% akan tetapi pada nilai = 0.000005 menghasilkan rata – rata runtime yang paling kecil yaitu sebesar 4745.9 sedangkan pada pengujian jarak jauh nilai = 0.000005 dan nilai = 5 sama – sama menghasilkan rata – rata hasil menemukan tujuan sebesar 100% dan pada

nilai = 5 memiliki nilai rata – rata runtime yang paling kecil yaitu 3158.7. Pada pengujian ini nilai parameter yang paling baik digunakan ialah nilai parameter yang mampu menemukan tujuan yang paling baik dan nilai runtime yang paling kecil akan tetapi yang lebih di prioritaskan pada pengujian ini adalah menemukan nilai parameter yang mampu menemukan tujuan yang paling baik, jadi nilai parameter = 5 merupakan nilai yang paling baik karena sudah memenuhi ketentuan rata – rata menemukan tujuan yang paling besar dan rata – rata runtime yang paling kecil.

5.2.3 Analisis Parameter ρ

Hasil pengujian untuk parameter ρ pada pengujian jarak dekat semua nilai menujukkan rata – rata menemukan tujuan sebesar 100% dan rata – rata runtime nilai ρ = 0.3 sebesar 4941.5, nilai ρ = 0.5 sebesar 4745.9 dan nilai ρ = 0.9 sebesar 5016.7 jadi nilai ρ = 0.5 merupakan nilai parameter yang memiliki rata – rata runtime paling kecil sedangkan pada pengujian jarak jauh nilai ρ = 0.5 dan 0.9 sama – sama menghasilkan rata – rata tujuan sebesar 100% dan pada nilai ρ = 0.5 memiliki nilai rata – rata runtime sebesar 7280.9 dan nilai ρ = 0.9 memiliki rata – rata runtime sebesar 3182.9 jadi nilai ρ = 0.9 merupakan nilai parameter yang memiliki rata – rata runtime paling kecil. Pada pengujian ini nilai parameter yang paling baik digunakan ialah nilai parameter yang mampu menemukan tujuan yang paling baik dan nilai runtime yang paling kecil akan tetapi yang lebih di prioritaskan pada pengujian ini adalah menemukan nilai parameter yang mampu menemukan tujuan yang paling baik, jadi nilai parameter yang di pilih pada pengujian ini ialah 0.9 meskipun pada pengujian jarak dekat nilai parameter ρ = 0.5 memiliki rata – rata runtime yang paling kecil akan tetapi perbedaan rata – rata runtime dari nilai ρ = 0.5 dan ρ = 0.9 yang ada pada pengujian jarak jauh memiliki rentan yang lebih lama dibandingkan yang ada pada pengujian jarak dekat.

5.2.4 Analisis Parameter *NcMax* dan *m*

Hasil pengujian untuk parameter NcMax dan m pada pengujian jarak dekat semua nilai menunjukkan rata – rata menemukan tujuan sebesar 100% akan tetapi pada nilai NcMax = 100 dan m = 25 menghasilkan rata – rata runtime yang paling kecil yaitu sebesar 1240.7 sedangkan pada pengujian jarak jauh nilai NcMax = 500

dan m = 50 menghasilkan rata – rata menemukan tujuan paling besar yaitu 100% dan pada nilai NcMax = 500 dan m = 50 tersebut menghasilkan nilai rata – rata runtime paling besar yaitu 6218.5. Pada pengujian ini nilai parameter yang paling baik digunakan ialah nilai parameter yang mampu menemukan tujuan yang paling baik dan nilai runtime yang paling kecil akan tetapi yang lebih di prioritaskan pada pengujian ini adalah menemukan nilai parameter yang mampu menemukan tujuan yang paling baik, jadi meskipun pada pengujian jarak dekat nilai NcMax = 100 dan m = 25 menghasilkan rata – rata runtime yang paling kecil sedangkan pada pengujian jarak jauh nilai NcMax = 500 dan m = 50 menghasilkan rata – rata runtime yang paling besar dari pada nilai NcMax dan m lainnya maka hal tersebut akan diabaikan karena pada pengujian ini nilai parameter yang paling baik lebih memprioritaskan nilai parameter yang mampu menghasilkan rata – rata menemukan tujuan yang paling baik.

BAB VI
PENUTUP

6.1 Kesimpulan

Berdasarkan hasil perancangan, implementasi dan pengujian yang dilakukan, maka diambil kesimpulan sebagai berikut :

1. Aplikasi pencarian rute angkutan umum dengan menggunakan Algoritma Ant Colony Optimization telah dibuat sesuai hasil perancangan dan dapat menemukan rute angkutan umum langsung dari perangkat *smartphone* Android. Proses pencarian rute angkutan umum yang diimplementasikan dapat menghasilkan 1 rute terpendek dan 1 rute termurah.
2. Penggunaan parameter pendukung Algoritma Ant Colony Optimization yang paling baik dapat mempengaruhi tingkat akurasi dan lama proses pencarian rute angkutan umum yang terbaik dalam menentukan rute terpendek dan termurah.
3. Tingkat akurasi Algoritma Ant Colony Optimization untuk menemukan rute angkutan umum terdekat dan termurah dengan menggunakan 5 sampel rute angkutan umum memiliki tingkat akurasi sebesar 100%. Nilai parameter yang digunakan pada penelitian ini adalah α = 5, β = 0.0000000001, = 5, ρ = 0.5, Q = 1, NcMax = 500 dan m = 50.

6.2 Saran

Saran yang yang dapat dilakukan untuk penelitian selanjutnya adalah sebagai berikut :

1. Ditambahkan sebuah rute tercepat yang dibedakan berdasarkan jalan mana yang memiliki tingkat kemacetan lebih besar dan yang lebih kecil untuk waktu – waktu tertentu.
2. Untuk proses komputasi sebaiknya dilakukan menggunakan sistem client – server dikarenakan pada saat penambahan jumlah angkutan umum maka

waktu pemrosesan akan menjadi lebih lambat bila langsung di lakukan pada *smartphone*.

DAFTAR PUSTAKA

[KUS-11] Kushwaha, Vineet Kushwaha. 2011. *Location Based Services using Android Smart phone Operating Sistem*. November 2013

[STC-14] Statcounter. 2014. StatCounter Global Stats. WWW [terhubung berkala]. http://gs.statcounter.com [diakses pada tanggal 11 April 2014].

[DOR-96] Dorigo,M. 1996. The Ant System: Optimization by a colony of cooperating agents, IEEE transactions on Systems, Man, and Cybernetics–Part B, Vol.26, No.1. Januari 2014

[JWI-90] J. Wilson and J.J. Watkins John Wiley & Sons. 1990. Graphs: An Introductory Approach. Maret 2014

[DOR-91] Dorigo, M., Maniezzo, V., dan Colorni, A. 1991. Positive feedback as a search strategy. Technical report 91-016, Dipartimento di Elettronica, Politecnico di Milano, Milan. Maret 2014

[MNR-06] Munir, Rinaldi. 2006. Strategi Algoritmik, Program Studi Teknik Informatika, Sekolah Teknik Elektro dan Informatika Institut Teknologi Bandung. Maret 2014

[SMR-04] Syaroni, Mokhamad dan Munir, Rinaldi. 2004. Pencocokan String Berdasarkan Kemiripan Ucapan (Phonetic String Matching) Dalam Bahasa Inggris. Departemen Teknik Informatika, Fakultas Teknologi Industri, Institut Teknologi Bandung. Maret 2014

[AMV-91] A. Colorni, M. Dorigo et V. Maniezzo, Distributed Optimization by Ant Colonies, actes de la première conférence européenne sur la vie artificielle, Paris, France, Elsevier Publishing, 134-142, 1991. Maret 2014

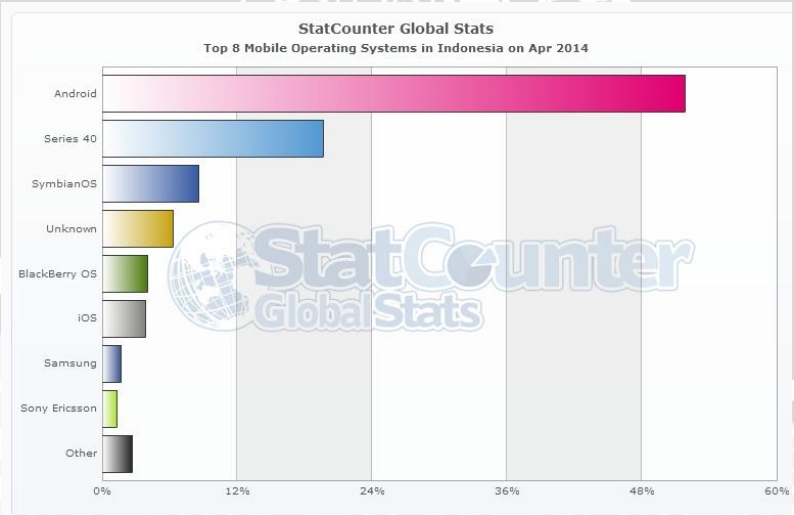
[DOR-04] Dorigo M, Stutzle T. 2004. Ant Colony Optimization. Massachusetts Institute of Technology: USA. Maret 2014

LAMPIRAN

Lampiran 1 Data statistik jumlah pengguna android di Indonesia

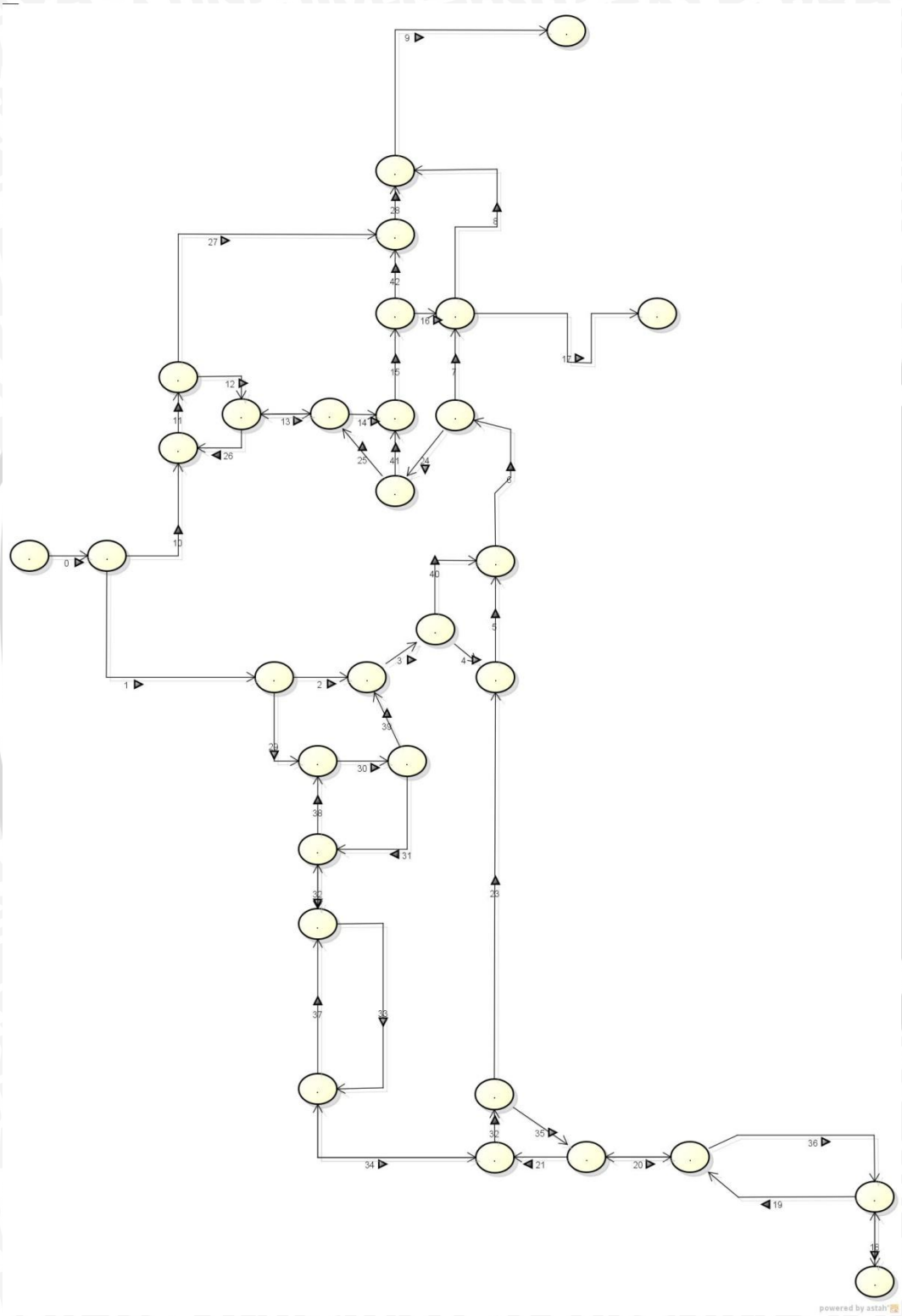
Tabel L.1 Data statistik pengguna android di Indonesia

OS	Market Share Perc. (Apr 2014)
Android	51.83
Series 40	19.65
SymbianOS	8.61
Unknown	6.3
BlackBerry OS	4.08
iOS	3.86
Samsung	1.71
Sony Ericsson	1.28
Windows Phone	1
Nokia Unknown	0.97
Linux	0.38
LG	0.13
Brew	0.09
bada	0.05
WinVista	0.05
Other	0.03
Android	51.83



Gambar L.1 Top 8 mobile pada bulan april 2014 di indonesia

Lampiran 2 Graph dari 5 angkutan umum yang digunakan



Gambar L.2 Graph 5 angkutan umum yang digunakan

Lampiran 2 Hasil pembuktian untuk nilai β = 0.0000000001

Tabel L.2 Inisialisasi parameter pembuktian

Tij Pakai	0.5
a	0.1
b	0.000000001
rho	0.7
Pq	1

Tabel L.3 Pembuktian matriks Jarak

MATRIKS JARAK															
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
0	0	452													
1		0	618												8580
2			0	387										188	
3				0	3714									188	
4					0	3206							1870		
5						0	107						1870		
6							0	318				105			
7								0	1330			105			
8									0	34	1299				
9										0	1299				
10								318			0				
11						3206						0			
12				387	3714								0		
13		452												0	
14															0

Tabel L.4 Pembuktian intensitas jejak semut

INTESITAS JEJAK SEMUT															
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
0	0	0.5													
1		0	0.5												0.5
2			0	0.5										0.5	
3				0	0.5									0.5	
4					0	0.5							0.5		

5						0	0.5						0.5		
6							0	0.5				0.5			
7								0	0.5			0.5			
8									0	0.5	0.5				
9										0	0.5				
10								0.5			0				
11						0.5						0			
12				0.5	0.5								0		
13		0.5												0	
14															0

Tabel L.5 Pembuktian visibilitas semut

VISIBILITAS															
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
0	0	0.00221													
1		0	0.00162												0.00012
2			0	0.00258										0.00532	
3				0	0.00027									0.00532	
4					0	0.00031							0.00053		
5						0	0.00935						0.00053		
6							0	0.00314				0.00952			
7								0	0.00075			0.00952			
8									0	0.02941	0.00077				
9										0	0.00077				
10								0.00314			0				
11						0.00031						0			
12				0.00258	0.00027								0		
13		0.00221												0	
14															0

Tabel L.6 Pembuktian probabilitas semut

PROBABILITAS															
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
0	0	1													
1		0	0.5												0.5
2			0	0.5										0.5	
3				0	0.5									0.5	
4					0	0.5							0.5		

5						0	0.5						0		
6							0	0.5				0.5			
7								0	0.5			0.5			
8									0	0.5	0.5				
9										0	1				
10								1			0				
11						1						0			
12				0.5	0.5								0		
13		1												0	
14															0

Lampiran 3 Hasil Pengujian Nilai α

1. Uji nilai $\alpha = 0$

   a. Jarak dekat

Tabel L.7 Hasil pengujian nilai $\alpha = 0$ untuk jarak dekat

						Jarak dekat (terdekat)		Jarak dekat (termurah)			
Ulangan	α	t0	ρ	Siklus	Semut	Solusi (meter)	Jml angkot	Solusi (meter)	Jml angkot	Menemukan tujuan	Runtime (milidetik)
1	0	0.5	0.5	500	25	13142	2	20273	1	ya	6040
2	0	0.5	0.5	500	25	13142	2	20273	1	ya	6069
3	0	0.5	0.5	500	25	13142	2	20273	1	ya	5595
4	0	0.5	0.5	500	25	13142	2	20273	1	ya	6292
5	0	0.5	0.5	500	25	13142	2	20273	1	ya	6025
6	0	0.5	0.5	500	25	13142	2	20273	1	ya	5810
7	0	0.5	0.5	500	25	13142	2	20273	1	ya	5684
8	0	0.5	0.5	500	25	13142	2	20273	1	ya	5887
9	0	0.5	0.5	500	25	13142	2	20273	1	ya	6268
10	0	0.5	0.5	500	25	13142	2	20273	1	ya	5736
Rata – rata										100%	5940.6

   b. Jarak jauh

Tabel L.8 Hasil pengujian nilai $\alpha = 0$ untuk jarak jauh

						Jarak jauh (terdekat)		Jarak jauh (termurah)			
Ulangan	α	t0	ρ	Siklus	Semut	Solusi (meter)	Jml angkot	Solusi (meter)	Jml angkot	Menemukan tujuan	Runtime (milidetik)
1	0	0.5	0.5	500	25	21110	1	21110	1	ya	3158
2	0	0.5	0.5	500	25	21110	1	21110	1	ya	3161
3	0	0.5	0.5	500	25	21110	1	21110	1	ya	3233
4	0	0.5	0.5	500	25	21110	1	21110	1	ya	3109
5	0	0.5	0.5	500	25	21110	1	21110	1	ya	3315
6	0	0.5	0.5	500	25	21110	1	21110	1	ya	3099
7	0	0.5	0.5	500	25	21110	1	21110	1	ya	3253
8	0	0.5	0.5	500	25	21110	1	21110	1	ya	3113
9	0	0.5	0.5	500	25						3004
10	0	0.5	0.5	500	25	21110	1	21110	1	ya	3222
Rata – rata										90%	3166.7

2. Uji nilai $\alpha = 0.5$

a. Jarak dekat

Tabel L.9 Hasil pengujian nilai $\alpha = 0.5$ untuk jarak dekat

						Jarak dekat (terdekat)		Jarak dekat (termurah)			
Ulangan	α	t0	ρ	Siklus	Semut	Solusi (meter)	Jml angkot	Solusi (meter)	Jml angkot	Menemukan tujuan	Runtime (milidetik)
1	0.5	0.5	0.5	500	25	13142	2	20273	1	ya	4827
2	0.5	0.5	0.5	500	25	13142	2	20273	1	ya	5285
3	0.5	0.5	0.5	500	25	13142	2	20273	1	ya	5088
4	0.5	0.5	0.5	500	25	13142	2	20273	1	ya	4981
5	0.5	0.5	0.5	500	25	13142	2	20273	1	ya	4990
6	0.5	0.5	0.5	500	25	13142	2	20273	1	ya	4850
7	0.5	0.5	0.5	500	25	13142	2	20273	1	ya	4900
8	0.5	0.5	0.5	500	25	13142	2	20273	1	ya	4977
9	0.5	0.5	0.5	500	25	13142	2	20273	1	ya	4682
10	0.5	0.5	0.5	500	25	13142	2	20273	1	ya	4872
Rata – rata										100%	4945.2

b. Jarak jauh

Tabel L.10 Hasil pengujian nilai α = 0.5 untuk jarak jauh

						Jarak jauh (terdekat)		Jarak jauh (termurah)			
Ulangan	α	t0	ρ	Siklus	Semut	Solusi (meter)	Jml angkot	Solusi (meter)	Jml angkot	Menemukan tujuan	Runtime (milidetik)
1	0.5	0.5	0.5	500	25	21110	1	21110	1	Ya	3229
2	0.5	0.5	0.5	500	25	21110	1	21110	1	Ya	3159
3	0.5	0.5	0.5	500	25	21110	1	21110	1	Ya	3230
4	0.5	0.5	0.5	500	25				1	Tidak	3228
5	0.5	0.5	0.5	500	25	21110	1	21110	1	Ya	3166
6	0.5	0.5	0.5	500	25	21110	1	21110	1	Ya	3137
7	0.5	0.5	0.5	500	25	21110	1	21110	1	Ya	5376
8	0.5	0.5	0.5	500	25	21110	1	21110	1	Ya	3067
9	0.5	0.5	0.5	500	25	21110	1	21110	1	Ya	3311
10	0.5	0.5	0.5	500	25	21110	1	21110	1	Ya	3356
Rata – rata										90%	3425.9

3. Uji nilai α = 1

a. Jarak dekat

Tabel L.11 Hasil pengujian nilai α = 1 untuk jarak dekat

						Jarak dekat (terdekat)		Jarak dekat (termurah)			
Ulangan	α	t0	ρ	Siklus	Semut	Solusi (meter)	Jml angkot	Solusi (meter)	Jml angkot	Menemukan tujuan	Runtime (milidetik)
1	1	0.5	0.5	500	25	13142	2	20273	1	ya	5166
2	1	0.5	0.5	500	25	13142	2	20273	1	ya	4958
3	1	0.5	0.5	500	25	13142	2	20273	1	ya	5039
4	1	0.5	0.5	500	25	13142	2	20273	1	ya	5250
5	1	0.5	0.5	500	25	13142	2	20273	1	ya	4920
6	1	0.5	0.5	500	25	13142	2	20273	1	ya	5172
7	1	0.5	0.5	500	25	13142	2	20273	1	ya	4757
8	1	0.5	0.5	500	25	13142	2	20273	1	ya	5086
9	1	0.5	0.5	500	25	13142	2	20273	1	ya	5107
10	1	0.5	0.5	500	25	13142	2	20273	1	ya	5122
Rata – rata										100%	5057.7

b. Jarak jauh

Tabel L.12 Hasil pengujian nilai α = 1 untuk jarak jauh

						Jarak jauh (terdekat)		Jarak jauh (termurah)			
Ulangan	α	t0	ρ	Siklus	Semut	Solusi (meter)	Jml angkot	Solusi (meter)	Jml angkot	Menemukan tujuan	Runtime (milidetik)
1	1	0.5	0.5	500	25					tidak	3020
2	1	0.5	0.5	500	25	21110	1	21110	1	ya	3486
3	1	0.5	0.5	500	25	21110	1	21110	1	ya	3459
4	1	0.5	0.5	500	25	21110	1	21110	1	ya	3112
5	1	0.5	0.5	500	25	21110	1	21110	1	ya	3281
6	1	0.5	0.5	500	25	21110	1	21110	1	ya	3177
7	1	0.5	0.5	500	25	21110	1	21110	1	ya	3536
8	1	0.5	0.5	500	25	21110	1	21110	1	ya	3256
9	1	0.5	0.5	500	25	21110	1	21110	1	ya	3336
10	1	0.5	0.5	500	25	21110	1	21110	1	ya	3248
Rata – rata										90%	3291.1

4. Uji nilai α = 2

   a. Jarak dekat

Tabel L.13 Hasil pengujian nilai α = 2 untuk jarak dekat

						Jarak dekat (terdekat)		Jarak dekat (termurah)			
Ulangan	α	t0	ρ	Siklus	Semut	Solusi (meter)	Jml angkot	Solusi (meter)	Jml angkot	Menemukan tujuan	Runtime (milidetik)
1	2	0.5	0.5	500	25	13142	2	20273	1	ya	5445
2	2	0.5	0.5	500	25	13142	2	20273	1	ya	5256
3	2	0.5	0.5	500	25	13142	2	20273	1	ya	5183
4	2	0.5	0.5	500	25	13142	2	20273	1	ya	4812
5	2	0.5	0.5	500	25	13142	2	20273	1	ya	4967
6	2	0.5	0.5	500	25	13142	2	20273	1	ya	5093
7	2	0.5	0.5	500	25	13142	2	20273	1	ya	4889
8	2	0.5	0.5	500	25	13142	2	20273	1	ya	5168
9	2	0.5	0.5	500	25	13142	2	20273	1	ya	5245
10	2	0.5	0.5	500	25	13142	2	20273	1	ya	4878
Rata – rata										100%	5093.6

   b. Jarak jauh

Tabel L.14 Hasil pengujian nilai α = 2 untuk jarak jauh

						Jarak jauh (terdekat)		Jarak jauh (termurah)			
Ulangan	α	t0	ρ	Siklus	Semut	Solusi (meter)	Jml angkot	Solusi (meter)	Jml angkot	Menemukan tujuan	Runtime (milidetik)
1	2	0.5	0.5	500	25	21110	1	21110	1	Ya	3319
2	2	0.5	0.5	500	25	21110	1	21110	1	Ya	2917
3	2	0.5	0.5	500	25	21110	1	21110	1	Ya	3048
4	2	0.5	0.5	500	25	21110	1	21110	1	Ya	3095
5	2	0.5	0.5	500	25					Tidak	3213
6	2	0.5	0.5	500	25	21110	1	21110	1	Ya	3131
7	2	0.5	0.5	500	25					Tidak	3140
8	2	0.5	0.5	500	25	21110	1	21110	1	Ya	3162
9	2	0.5	0.5	500	25	21110	1	21110	1	Ya	3401
10	2	0.5	0.5	500	25	21110	1	21110	1	Ya	3309
Rata – rata										80%	3173.5

5. Uji nilai α = 5

   a. Jarak dekat

Tabel L.15 Hasil pengujian nilai α = 5 untuk jarak dekat

						Jarak dekat (terdekat)		Jarak dekat (termurah)			
Ulangan	α	t0	ρ	Siklus	Semut	Solusi (meter)	Jml angkot	Solusi (meter)	Jml angkot	Menemukan tujuan	Runtime (milidetik)
1	5	0.5	0.5	500	25	13142	2	20273	1	Ya	5667
2	5	0.5	0.5	500	25	13142	2	20273	1	Ya	5476
3	5	0.5	0.5	500	25	13142	2	20273	1	Ya	5249
4	5	0.5	0.5	500	25	13142	2	20273	1	Ya	5136
5	5	0.5	0.5	500	25	13142	2	20273	1	Ya	5195
6	5	0.5	0.5	500	25	13142	2	20273	1	Ya	5226
7	5	0.5	0.5	500	25	13142	2	20273	1	Ya	4920
8	5	0.5	0.5	500	25	13142	2	20273	1	Ya	5063
9	5	0.5	0.5	500	25	13142	2	20273	1	Ya	5138
10	5	0.5	0.5	500	25	13142	2	20273	1	Ya	5104
Rata – rata										100%	5217.4

   b. Jarak jauh

Tabel L.16 Hasil pengujian nilai α = 5 untuk jarak jauh

						Jarak jauh (terdekat)		Jarak jauh (termurah)			
Ulangan	α	t0	ρ	Siklus	Semut	Solusi (meter)	Jml angkot	Solusi (meter)	Jml angkot	Menemukan tujuan	Runtime (milidetik)
1	5	0.5	0.5	500	25	21110	1	21110	1	Ya	3560
2	5	0.5	0.5	500	25	21110	1	21110	1	Ya	3458
3	5	0.5	0.5	500	25	21110	1	21110	1	Ya	4134
4	5	0.5	0.5	500	25	21110	1	21110	1	Ya	3478
5	5	0.5	0.5	500	25	21110	1	21110	1	Ya	3399
6	5	0.5	0.5	500	25	21110	1	21110	1	Ya	3627
7	5	0.5	0.5	500	25	21110	1	21110	1	Ya	3295
8	5	0.5	0.5	500	25	21110	1	21110	1	Ya	3489
9	5	0.5	0.5	500	25	21110	1	21110	1	Ya	3810
10	5	0.5	0.5	500	25	21110	1	21110	1	Ya	3513
Rata – rata										100%	3576.3

Lampiran 4 Hasil Pengujian Nilai

1. Uji nilai = 0.000005

   a. Jarak dekat

Tabel L.17 Hasil pengujian nilai = 0.000005 untuk jarak dekat

						Jarak dekat (terdekat)		Jarak dekat (termurah)			
Ulangan	α	t0	ρ	Siklus	Semut	Solusi (meter)	Jml angkot	Solusi (meter)	Jml angkot	Menemukan tujuan	Runtime (milidetik)
1	0.5	0.000005	0.5	500	25	13142	2	20273	1	Ya	4929
2	0.5	0.000005	0.5	500	25	13142	2	20273	1	Ya	4700
3	0.5	0.000005	0.5	500	25	13142	2	20273	1	Ya	4499
4	0.5	0.000005	0.5	500	25	13142	2	20273	1	Ya	4750
5	0.5	0.000005	0.5	500	25	13142	2	20273	1	Ya	4745
6	0.5	0.000005	0.5	500	25	13142	2	20273	1	Ya	4840
7	0.5	0.000005	0.5	500	25	13142	2	20273	1	Ya	4695
8	0.5	0.000005	0.5	500	25	13142	2	20273	1	Ya	4900
9	0.5	0.000005	0.5	500	25	13142	2	20273	1	Ya	4705
10	0.5	0.000005	0.5	500	25	13142	2	20273	1	Ya	4696
Rata – rata										100%	4745.9

b. Jarak jauh

Tabel L.18 Hasil pengujian nilai = 0.000005 untuk jarak jauh

						Jarak jauh (terdekat)		Jarak jauh (termurah)			
Ulangan	α	t0	ρ	Siklus	Semut	Solusi (meter)	Jml angkot	Solusi (meter)	Jml angkot	Menemukan tujuan	Runtime (milidetik)
1	0.5	0.000005	0.5	500	25	21110	1	21110	1	Ya	4737
2	0.5	0.000005	0.5	500	25	21110	1	21110	1	Ya	8709
3	0.5	0.000005	0.5	500	25	21110	1	21110	1	Ya	8676
4	0.5	0.000005	0.5	500	25	21110	1	21110	1	Ya	6096
5	0.5	0.000005	0.5	500	25	21110	1	21110	1	Ya	7606
6	0.5	0.000005	0.5	500	25	21110	1	21110	1	Ya	4176
7	0.5	0.000005	0.5	500	25	21110	1	21110	1	Ya	8072
8	0.5	0.000005	0.5	500	25	21110	1	21110	1	Ya	9220
9	0.5	0.000005	0.5	500	25	21110	1	21110	1	Ya	7247
10	0.5	0.000005	0.5	500	25	21110	1	21110	1	Ya	8263
Rata – rata										100%	7280.2

2. Uji nilai = 0.5

a. Jarak dekat

Tabel L.19 Hasil pengujian nilai = 0.5 untuk jarak dekat

						Jarak dekat (terdekat)		Jarak dekat (termurah)			
Ulangan	α	t0	ρ	Siklus	Semut	Solusi (meter)	Jml angkot	Solusi (meter)	Jml angkot	Menemukan tujuan	Runtime (milidetik)
1	0.5	0.5	0.5	500	25	13142	2	20273	1	Ya	4827
2	0.5	0.5	0.5	500	25	13142	2	20273	1	Ya	5285
3	0.5	0.5	0.5	500	25	13142	2	20273	1	Ya	5088
4	0.5	0.5	0.5	500	25	13142	2	20273	1	Ya	4981
5	0.5	0.5	0.5	500	25	13142	2	20273	1	Ya	4990
6	0.5	0.5	0.5	500	25	13142	2	20273	1	Ya	4850
7	0.5	0.5	0.5	500	25	13142	2	20273	1	Ya	4900
8	0.5	0.5	0.5	500	25	13142	2	20273	1	Ya	4977
9	0.5	0.5	0.5	500	25	13142	2	20273	1	Ya	4682
10	0.5	0.5	0.5	500	25	13142	2	20273	1	Ya	4872
Rata – rata										100%	4945.2

b. Jarak jauh

Tabel L.20 Hasil pengujian nilai = 0.5 untuk jarak jauh

						Jarak jauh (terdekat)		Jarak jauh (termurah)			
Ulangan	α	t0	ρ	Siklus	Semut	Solusi (meter)	Jml angkot	Solusi (meter)	Jml angkot	Menemukan tujuan	Runtime (milidetik)
1	0.5	0.5	0.5	500	25	21110	1	21110	1	ya	3229
2	0.5	0.5	0.5	500	25	21110	1	21110	1	ya	3159
3	0.5	0.5	0.5	500	25	21110	1	21110	1	ya	3230
4	0.5	0.5	0.5	500	25				1	tidak	3228
5	0.5	0.5	0.5	500	25	21110	1	21110	1	ya	3166
6	0.5	0.5	0.5	500	25	21110	1	21110	1	ya	3137
7	0.5	0.5	0.5	500	25	21110	1	21110	1	ya	5376
8	0.5	0.5	0.5	500	25	21110	1	21110	1	ya	3067
9	0.5	0.5	0.5	500	25	21110	1	21110	1	ya	3311
10	0.5	0.5	0.5	500	25	21110	1	21110	1	ya	3356
Rata – rata										90%	3425.9

3. Uji nilai = 5

a. Jarak dekat

Tabel L.21 Hasil pengujian nilai = 5 untuk jarak dekat

						Jarak dekat (terdekat)		Jarak dekat (termurah)			
Ulangan	α	t0	ρ	Siklus	Semut	Solusi (meter)	Jml angkot	Solusi (meter)	Jml angkot	Menemukan tujuan	Runtime (milidetik)
1	0.5	5	0.5	500	25	13142	2	20273	1	Ya	4919
2	0.5	5	0.5	500	25	13142	2	20273	1	Ya	5076
3	0.5	5	0.5	500	25	13142	2	20273	1	Ya	4851
4	0.5	5	0.5	500	25	13142	2	20273	1	Ya	4949
5	0.5	5	0.5	500	25	13142	2	20273	1	Ya	5938
6	0.5	5	0.5	500	25	13142	2	20273	1	Ya	4792
7	0.5	5	0.5	500	25	13142	2	20273	1	Ya	4936
8	0.5	5	0.5	500	25	13142	2	20273	1	Ya	4737
9	0.5	5	0.5	500	25	13142	2	20273	1	Ya	4801
10	0.5	5	0.5	500	25	13142	2	20273	1	Ya	5289
Rata – rata										100%	5028.8

b. Jarak jauh

Tabel L.22 Hasil pengujian nilai = 5 untuk jarak jauh

						Jarak jauh (terdekat)		Jarak jauh (termurah)			
Ulangan	α	t0	ρ	Siklus	Semut	Solusi (meter)	Jml angkot	Solusi (meter)	Jml angkot	Menemukan tujuan	Runtime (milidetik)
1	0.5	5	0.5	500	25	21110	1	21110	1	Ya	3031
2	0.5	5	0.5	500	25	21110	1	21110	1	Ya	3261
3	0.5	5	0.5	500	25	21110	1	21110	1	Ya	3200
4	0.5	5	0.5	500	25	21110	1	21110	1	Ya	3087
5	0.5	5	0.5	500	25	21110	1	21110	1	ya	3026
6	0.5	5	0.5	500	25	21110	1	21110	1	ya	3223
7	0.5	5	0.5	500	25	21110	1	21110	1	ya	3005
8	0.5	5	0.5	500	25	21110	1	21110	1	ya	3086
9	0.5	5	0.5	500	25	21110	1	21110	1	ya	3280
10	0.5	5	0.5	500	25	21110	1	21110	1	ya	3388
Rata – rata										100%	3158.7

Lampiran 5 Hasil Pengujian Nilai ρ

1. Uji nilai ρ = 0.3

   a. Jarak dekat

Tabel L.23 Hasil pengujian nilai ρ = 0.3 untuk jarak dekat

						Jarak dekat (terdekat)		Jarak dekat (termurah)			
Ulangan	α	t0	ρ	Siklus	Semut	Solusi (meter)	Jml angkot	Solusi (meter)	Jml angkot	Menemukan tujuan	Runtime (milidetik)
1	0.5	0.5	0.1	500	25	13142	2	20273	1	Ya	4830
2	0.5	0.5	0.1	500	25	13142	2	20273	1	Ya	4776
3	0.5	0.5	0.1	500	25	13142	2	20273	1	Ya	5003
4	0.5	0.5	0.1	500	25	13142	2	20273	1	Ya	4854
5	0.5	0.5	0.1	500	25	13142	2	20273	1	Ya	4956
6	0.5	0.5	0.1	500	25	13142	2	20273	1	Ya	5194
7	0.5	0.5	0.1	500	25	13142	2	20273	1	Ya	5023
8	0.5	0.5	0.1	500	25	13142	2	20273	1	Ya	4829
9	0.5	0.5	0.1	500	25	13142	2	20273	1	Ya	5018
10	0.5	0.5	0.1	500	25	13142	2	20273	1	Ya	4932
Rata – rata										100%	4941.5

b. Jarak jauh

Tabel L.24 Hasil pengujian nilai ρ = 0.3 untuk jarak jauh

						Jarak jauh (terdekat)		Jarak jauh (termurah)			
Ulangan	α	t0	ρ	Siklus	Semut	Solusi (meter)	Jml angkot	Solusi (meter)	Jml angkot	Menemukan tujuan	Runtime (milidetik)
1	0.5	0.5	0.1	500	25	21110	1	21110	1	Ya	3631
2	0.5	0.5	0.1	500	25	21110	1	21110	1	Ya	3251
3	0.5	0.5	0.1	500	25	21110	1	21110	1	Ya	3135
4	0.5	0.5	0.1	500	25	21110	1	21110	1	Ya	3110
5	0.5	0.5	0.1	500	25	21110	1	21110	1	Ya	3487
6	0.5	0.5	0.1	500	25	21110	1	21110	1	Ya	3173
7	0.5	0.5	0.1	500	25	21110	1	21110	1	Ya	3082
8	0.5	0.5	0.1	500	25	21110	1	21110	1	Ya	3195
9	0.5	0.5	0.1	500	25					Tidak	3431
10	0.5	0.5	0.1	500	25	21110	1	21110	1	Ya	3226
Rata – rata										90%	3272.1

2. Uji nilai ρ = 0.5

a. Jarak dekat

Tabel L.25 Hasil pengujian nilai ρ = 0.5 untuk jarak dekat

						Jarak dekat (terdekat)		Jarak dekat (termurah)			
Ulangan	α	t0	ρ	Siklus	Semut	Solusi (meter)	Jml angkot	Solusi (meter)	Jml angkot	Menemukan tujuan	Runtime (milidetik)
1	0.5	0.5	0.5	500	25	13142	2	20273	1	Ya	4929
2	0.5	0.5	0.5	500	25	13142	2	20273	1	Ya	4700
3	0.5	0.5	0.5	500	25	13142	2	20273	1	Ya	4499
4	0.5	0.5	0.5	500	25	13142	2	20273	1	Ya	4750
5	0.5	0.5	0.5	500	25	13142	2	20273	1	Ya	4745
6	0.5	0.5	0.5	500	25	13142	2	20273	1	Ya	4840
7	0.5	0.5	0.5	500	25	13142	2	20273	1	Ya	4695
8	0.5	0.5	0.5	500	25	13142	2	20273	1	Ya	4900
9	0.5	0.5	0.5	500	25	13142	2	20273	1	Ya	4705
10	0.5	0.5	0.5	500	25	13142	2	20273	1	Ya	4696
Rata – rata										100%	4745.9

b. Jarak jauh

Tabel L.26 Hasil pengujian nilai ρ = 0.5 untuk jarak jauh

						Jarak jauh (terdekat)		Jarak jauh (termurah)			
Ulangan	α	t0	ρ	Siklus	Semut	Solusi (meter)	Jml angkot	Solusi (meter)	Jml angkot	Menemukan tujuan	Runtime (milidetik)
1	0.5	0.5	0.5	500	25	21110	1	21110	1	Ya	4737
2	0.5	0.5	0.5	500	25	21110	1	21110	1	Ya	8709
3	0.5	0.5	0.5	500	25	21110	1	21110	1	Ya	8676
4	0.5	0.5	0.5	500	25	21110	1	21110	1	Ya	6096
5	0.5	0.5	0.5	500	25	21110	1	21110	1	Ya	7606
6	0.5	0.5	0.5	500	25	21110	1	21110	1	Ya	4176
7	0.5	0.5	0.5	500	25	21110	1	21110	1	Ya	8072
8	0.5	0.5	0.5	500	25	21110	1	21110	1	Ya	9220
9	0.5	0.5	0.5	500	25	21110	1	21110	1	Ya	7247
10	0.5	0.5	0.5	500	25	21110	1	21110	1	Ya	8263
Rata – rata										100%	7280.2

3. Uji nilai ρ = 0.9

a. Jarak dekat

Tabel L.27 Hasil pengujian nilai ρ = 0.9 untuk jarak dekat

						Jarak dekat (terdekat)		Jarak dekat (termurah)			
Ulangan	α	t0	ρ	Siklus	Semut	Solusi (meter)	Jml angkot	Solusi (meter)	Jml angkot	Menemukan tujuan	Runtime (milidetik)
1	0.5	0.5	0.9	500	25	13142	2	20273	1	Ya	5096
2	0.5	0.5	0.9	500	25	13142	2	20273	1	Ya	5107
3	0.5	0.5	0.9	500	25	13142	2	20273	1	Ya	5427
4	0.5	0.5	0.9	500	25	13142	2	20273	1	Ya	4902
5	0.5	0.5	0.9	500	25	13142	2	20273	1	Ya	5273
6	0.5	0.5	0.9	500	25	13142	2	20273	1	Ya	4768
7	0.5	0.5	0.9	500	25	13142	2	20273	1	Ya	4963
8	0.5	0.5	0.9	500	25	13142	2	20273	1	Ya	4833
9	0.5	0.5	0.9	500	25	13142	2	20273	1	Ya	4829
10	0.5	0.5	0.9	500	25	13142	2	20273	1	Ya	4969
Rata – rata										100%	5016.7

b. Jarak jauh

Tabel L.28 Hasil pengujian nilai ρ = 0.9 untuk jarak jauh

						Jarak jauh (terdekat)		Jarak jauh (termurah)			
Ulangan	α	t0	ρ	Siklus	Semut	Solusi (meter)	Jml angkot	Solusi (meter)	Jml angkot	Menemukan tujuan	Runtime (milidetik)
1	0.5	0.5	0.9	500	25	21110	1	21110	1	Ya	3354
2	0.5	0.5	0.9	500	25	21110	1	21110	1	ya	3320
3	0.5	0.5	0.9	500	25	21110	1	21110	1	ya	3304
4	0.5	0.5	0.9	500	25	21110	1	21110	1	ya	3113
5	0.5	0.5	0.9	500	25	21110	1	21110	1	ya	3128
6	0.5	0.5	0.9	500	25	21110	1	21110	1	ya	3045
7	0.5	0.5	0.9	500	25	21110	1	21110	1	ya	2919
8	0.5	0.5	0.9	500	25	21110	1	21110	1	ya	3054
9	0.5	0.5	0.9	500	25	21110	1	21110	1	ya	3248
10	0.5	0.5	0.9	500	25	21110	1	21110	1	ya	3344
Rata – rata										100%	3182.9

Lampiran 6 Hasil Pengujian Nilai NcMax dan m

1. Uji nilai NcMax = 500 dan m = 50

   a. Jarak dekat

Tabel L.29 Hasil pengujian nilai NcMax = 500 dan m = 50 untuk jarak dekat

						Jarak dekat (terdekat)		Jarak dekat (termurah)			
Ulangan	α	t0	ρ	Siklus	Semut	Solusi (meter)	Jml angkot	Solusi (meter)	Jml angkot	Menemukan tujuan	Runtime (milidetik)
1	0.5	0.5	0.5	500	25	13142	2	20273	1	ya	4827
2	0.5	0.5	0.5	500	25	13142	2	20273	1	ya	5285
3	0.5	0.5	0.5	500	25	13142	2	20273	1	ya	5088
4	0.5	0.5	0.5	500	25	13142	2	20273	1	ya	4981
5	0.5	0.5	0.5	500	25	13142	2	20273	1	ya	4990
6	0.5	0.5	0.5	500	25	13142	2	20273	1	ya	4850
7	0.5	0.5	0.5	500	25	13142	2	20273	1	ya	4900
8	0.5	0.5	0.5	500	25	13142	2	20273	1	ya	4977

9	0.5	0.5	0.5	500	25	13142	2	20273	1	ya	4682
10	0.5	0.5	0.5	500	25	13142	2	20273	1	ya	4872
Rata – rata										100%	4945.2

b. Jarak jauh

Tabel L.30 Hasil pengujian nilai NcMax = 500 dan m = 50 untuk jarak jauh

						Jarak jauh (terdekat)		Jarak jauh (termurah)			
Ulangan	α	t0	ρ	Siklus	Semut	Solusi (meter)	Jml angkot	Solusi (meter)	Jml angkot	Menemukan tujuan	Runtime (milidetik)
1	0.5	0.5	0.5	500	25	21110	1	21110	1	ya	3229
2	0.5	0.5	0.5	500	25	21110	1	21110	1	ya	3159
3	0.5	0.5	0.5	500	25	21110	1	21110	1	ya	3230
4	0.5	0.5	0.5	500	25					tidak	3228
5	0.5	0.5	0.5	500	25	21110	1	21110	1	ya	3166
6	0.5	0.5	0.5	500	25	21110	1	21110	1	ya	3137
7	0.5	0.5	0.5	500	25	21110	1	21110	1	ya	5376
8	0.5	0.5	0.5	500	25	21110	1	21110	1	ya	3067
9	0.5	0.5	0.5	500	25	21110	1	21110	1	ya	3311
10	0.5	0.5	0.5	500	25	21110	1	21110	1	ya	3356
Rata – rata										90%	3425.9

2. Uji nilai NcMax = 500 dan m = 25

a. Jarak dekat

Tabel L.31 Hasil pengujian nilai NcMax = 500 dan m = 25 untuk jarak dekat

						Jarak dekat (terdekat)		Jarak dekat (termurah)			
Ulangan	α	t0	ρ	Siklus	Semut	Solusi (meter)	Jml angkot	Solusi (meter)	Jml angkot	Menemukan tujuan	Runtime (milidetik)
1	0.5	0.5	0.5	500	50	13142	2	20273	1	ya	9456
2	0.5	0.5	0.5	500	50	13142	2	20273	1	ya	8984
3	0.5	0.5	0.5	500	50	13142	2	20273	1	ya	9270
4	0.5	0.5	0.5	500	50	13142	2	20273	1	ya	9257
5	0.5	0.5	0.5	500	50	13142	2	20273	1	ya	9478
6	0.5	0.5	0.5	500	50	13142	2	20273	1	ya	9319

7	0.5	0.5	0.5	500	50	13142	2	20273	1	ya	9143
8	0.5	0.5	0.5	500	50	13142	2	20273	1	ya	9041
9	0.5	0.5	0.5	500	50	13142	2	20273	1	ya	9434
10	0.5	0.5	0.5	500	50	13142	2	20273	1	ya	8809
Rata – rata										100%	9219.1

b. Jarak jauh

Tabel L.32 Hasil pengujian nilai NcMax = 500 dan m = 25 untuk jarak jauh

						Jarak jauh (terdekat)		Jarak jauh (termurah)			
Ulangan	α	t0	ρ	Siklus	Semut	Solusi (meter)	Jml angkot	Solusi (meter)	Jml angkot	Menemukan tujuan	Runtime (milidetik)
1	0.5	0.5	0.5	500	50	21110	1	21110	1	ya	6746
2	0.5	0.5	0.5	500	50	21110	1	21110	1	ya	6088
3	0.5	0.5	0.5	500	50	21110	1	21110	1	ya	6150
4	0.5	0.5	0.5	500	50	21110	1	21110	1	ya	6082
5	0.5	0.5	0.5	500	50	21110	1	21110	1	ya	5993
6	0.5	0.5	0.5	500	50	21110	1	21110	1	ya	6183
7	0.5	0.5	0.5	500	50	21110	1	21110	1	ya	6339
8	0.5	0.5	0.5	500	50	21110	1	21110	1	ya	6041
9	0.5	0.5	0.5	500	50	21110	1	21110	1	ya	6156
10	0.5	0.5	0.5	500	50	21110	1	21110	1	ya	6407
Rata – rata										100%	6218.5

3. Uji nilai NcMax = 100 dan m = 100

a. Jarak dekat

Tabel L.33 Hasil pengujian nilai NcMax = 100 dan m = 100 untuk jarak dekat

						Jarak dekat (terdekat)		Jarak dekat (termurah)			
Ulangan	α	t0	ρ	Siklus	Semut	Solusi (meter)	Jml angkot	Solusi (meter)	Jml angkot	Menemukan tujuan	Runtime (milidetik)
1	0.5	0.5	0.5	100	25	13142	2	20273	1	ya	1106
2	0.5	0.5	0.5	100	25	13142	2	20273	1	ya	1220
3	0.5	0.5	0.5	100	25	13142	2	20273	1	ya	1062
4	0.5	0.5	0.5	100	25	13142	2	20273	1	ya	1193

5	0.5	0.5	0.5	100	25	13142	2	20273	1	ya	1212
6	0.5	0.5	0.5	100	25	13142	2	20273	1	ya	1278
7	0.5	0.5	0.5	100	25	13142	2	20273	1	ya	1572
8	0.5	0.5	0.5	100	25	13142	2	20273	1	ya	1364
9	0.5	0.5	0.5	100	25	13142	2	20273	1	ya	1135
10	0.5	0.5	0.5	100	25	13142	2	20273	1	ya	1265
Rata – rata										100%	1240.7

b. Jarak jauh

Tabel L.33 Hasil pengujian nilai NcMax = 100 dan m = 100 untuk jarak jauh

						Jarak jauh (terdekat)		Jarak jauh (termurah)			
Ulangan	α	t0	ρ	Siklus	Semut	Solusi (meter)	Jml angkot	Solusi (meter)	Jml angkot	Menemukan tujuan	Runtime (milidetik)
1	0.5	0.5	0.5	100	25					tidak	667
2	0.5	0.5	0.5	100	25	21110	1	21110	1	ya	909
3	0.5	0.5	0.5	100	25	21110	1	21110	1	ya	806
4	0.5	0.5	0.5	100	25					tidak	846
5	0.5	0.5	0.5	100	25					tidak	691
6	0.5	0.5	0.5	100	25					tidak	805
7	0.5	0.5	0.5	100	25					tidak	752
8	0.5	0.5	0.5	100	25					tidak	718
9	0.5	0.5	0.5	100	25	21110	1	21110	1	ya	844
10	0.5	0.5	0.5	100	25					tidak	747
Rata – rata										30%	778.5

4. Uji nilai NcMax = 100 dan m = 50

a. Jarak dekat

Tabel L.34 Hasil pengujian nilai NcMax = 100 dan m = 50 untuk jarak dekat

						Jarak dekat (terdekat)		Jarak dekat (termurah)			
Ulangan	α	t0	ρ	Siklus	Semut	Solusi (meter)	Jml angkot	Solusi (meter)	Jml angkot	Menemukan tujuan	Runtime (milidetik)
1	0.5	0.5	0.5	100	50	13142	2	20273	1	ya	2093
2	0.5	0.5	0.5	100	50	13142	2	20273	1	ya	2042

3	0.5	0.5	0.5	100	50	13142	2	20273	1	ya	1921
4	0.5	0.5	0.5	100	50	13142	2	20273	1	ya	1975
5	0.5	0.5	0.5	100	50	13142	2	20273	1	ya	1938
6	0.5	0.5	0.5	100	50	13142	2	20273	1	ya	2009
7	0.5	0.5	0.5	100	50	13142	2	20273	1	ya	2052
8	0.5	0.5	0.5	100	50	13142	2	20273	1	ya	1840
9	0.5	0.5	0.5	100	50	13142	2	20273	1	ya	1962
10	0.5	0.5	0.5	100	50	13142	2	20273	1	ya	1937
Rata – rata										100%	1976.9

b. Jarak jauh

Tabel L.35 Hasil pengujian nilai NcMax = 100 dan m = 50 untuk jarak jauh

						Jarak jauh (terdekat)		Jarak jauh (termurah)			
Ulangan	α	t0	ρ	Siklus	Semut	Solusi (meter)	Jml angkot	Solusi (meter)	Jml angkot	Menemukan tujuan	Runtime (milidetik)
1	0.5	0.5	0.5	100	50	21110	1	21110	1	ya	1522
2	0.5	0.5	0.5	100	50					tidak	1575
3	0.5	0.5	0.5	100	50					tidak	1403
4	0.5	0.5	0.5	100	50					tidak	1439
5	0.5	0.5	0.5	100	50	21110	1	21110	1	ya	1577
6	0.5	0.5	0.5	100	50					tidak	1324
7	0.5	0.5	0.5	100	50					tidak	1412
8	0.5	0.5	0.5	100	50	21110	1	21110	1	ya	1389
9	0.5	0.5	0.5	100	50	21110	1	21110	1	ya	1629
10	0.5	0.5	0.5	100	50	21110	1	21110	1	ya	3105
Rata – rata										50%	1637.5

5. Uji nilai NcMax = 100 dan m = 25

a. Jarak dekat

Tabel L.36 Hasil pengujian nilai NcMax = 100 dan m = 25 untuk jarak dekat

						Jarak dekat (terdekat)		Jarak dekat (termurah)			
Ulangan	α	t0	ρ	Siklus	Semut	Solusi (meter)	Jml angkot	Solusi (meter)	Jml angkot	Menemukan tujuan	Runtime (milidetik)

1	0.5	0.5	0.5	100	100	13142	2	20273	1	ya	3957
2	0.5	0.5	0.5	100	100	13142	2	20273	1	ya	3356
3	0.5	0.5	0.5	100	100	13142	2	20273	1	ya	3759
4	0.5	0.5	0.5	100	100	13142	2	20273	1	ya	4336
5	0.5	0.5	0.5	100	100	13142	2	20273	1	ya	3812
6	0.5	0.5	0.5	100	100	13142	2	20273	1	ya	3872
7	0.5	0.5	0.5	100	100	13142	2	20273	1	ya	3587
8	0.5	0.5	0.5	100	100	13142	2	20273	1	ya	3521
9	0.5	0.5	0.5	100	100	13142	2	20273	1	ya	3774
10	0.5	0.5	0.5	100	100	13142	2	20273	1	ya	4177
Rata – rata										100%	3815.1

b. Jarak jauh

Tabel L.37 Hasil pengujian nilai NcMax = 100 dan m = 25 untuk jarak jauh

						Jarak jauh (terdekat)		Jarak jauh (termurah)			
Ulangan	α	t0	ρ	Siklus	Semut	Solusi (meter)	Jml angkot	Solusi (meter)	Jml angkot	Menemukan tujuan	Runtime (milidetik)
1	0.5	0.5	0.5	100	100	21110	1	21110	1	ya	2559
2	0.5	0.5	0.5	100	100					tidak	2447
3	0.5	0.5	0.5	100	100					tidak	2579
4	0.5	0.5	0.5	100	100	21110	1	21110	1	ya	2829
5	0.5	0.5	0.5	100	100					tidak	2430
6	0.5	0.5	0.5	100	100	21110	1	21110	1	ya	2678
7	0.5	0.5	0.5	100	100	21110	1	21110	1	ya	2788
8	0.5	0.5	0.5	100	100	21110	1	21110	1	ya	2516
9	0.5	0.5	0.5	100	100	21110	1	21110	1	ya	2569
10	0.5	0.5	0.5	100	100	21110	1	21110	1	ya	2901
Rata – rata										70%	2629.6