# SISTEM OPTIMASI RUTE TERPENDEK PENGANGKUTAN SAMPAH DI SURABAYA MENGGUNAKAN *ANT COLONY OPTIMIZATION* (ACO)

**SKRIPSI**

Oleh

**Desi Wulandari**

**NIM 102410101052**

**PROGRAM STUDI SISTEM INFORMASI**

**UNIVERSITAS JEMBER**

**2015**

# SISTEM OPTIMASI RUTE TERPENDEK PENGANGKUTAN SAMPAH DI SURABAYA MENGGUNAKAN *ANT COLONY OPTIMIZATION* (ACO)

**SKRIPSI**

diajukan guna melengkapi tugas akhir dan memenuhi salah satu syarat untuk menyelesaikan pendidikan di Program Studi Sistem Informasi Universitas Jember dan mendapat gelar Sarjana Sistem Informasi

Oleh

**Desi Wulandari**

**NIM 102410101052**

**PROGRAM STUDI SISTEM INFORMASI**

**UNIVERSITAS JEMBER**

**2015**

## PERSEMBAHAN

Skripsi ini saya persembahkan untuk :

1. Ayahanda Suparlan dan Ibunda tercinta Sri Mulyani.
2. Adikku tersayang Riesmawan Bima Saputra.
3. Om Hadi Sujitno yang selalu memberi motivasi.
4. Guru-guruku sejak taman kanak-kanak sampai dengan perguruan tinggi.
5. Almamater Program Studi Sistem Informasi Universitas Jember.

**PERNYATAAN**

Saya yang bertanda tangan di bawah ini:

Nama : Desi Wulandari

NIM : 102410101052

menyatakan sesungguhnya bahwa karya ilmiah yang berjudul "Sistem Optimasi Rute Terpendek Pengangkutan Sampah di Surabaya Menggunakan *Ant Colony Optimization* (ACO)" adalah benar-benar hasil karya sendiri, kecuali kutipan yang sudah saya sebutkan sumbernya, belum pernah diajukan pada institusi mana pun dan bukan karya jiplakan. Saya bertanggung jawab atas keabsahan dan kebenaran isisnya sesuai dengan sikap ilmiah yang harus dijunjung tinggi.

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

Jember, Juni 2015

Yang menyatakan,

Desi Wulandari

NIM. 102410101052

**SKRIPSI**

# SISTEM OPTIMASI RUTE TERPENDEK PENGANGKUTAN SAMPAH DI SURABAYA MENGGUNAKAN *ANT COLONY OPTIMIZATION* (ACO)

Oleh:

DESI WULANDARI

NIM. 102410101052

Menyetujui

| Pembimbing Utama | Pembimbing Anggota |
| --- | --- |
| Dr. Saiful Bukhori, ST., M.Kom | Windi Eka Retnani, S.Kom., MT |
| NIP. 196811131994121001 | NIP.198403052010122002 |

**PENGESAHAN**

Skripsi berjudul **"Sistem Optimasi Rute Terpendek Pengangkutan Sampah di Surabaya Menggunakan *Ant Colony Optimization* (ACO)"**, telah diuji dan disahkan pada:

Hari tanggal : Senin, 29 Juni 2015

Tempat : Program Studi Sistem Informasi Universitas Jember

| Penguji 1, | Penguji 2, |
|---|---|
| Yanuar Nurdiansyah, ST,. M.Cs. | Anang Andrianto, ST,. MT |
| NIP. 198201012010121004 | NIP. 196906151997021002 |

Mengesahkan

Ketua Program Studi

Prof. Drs. Slamin, M.Comp.Sc.,Ph.D

NIP. 19670420 1992011001

## RINGKASAN

Rancang Bangun Sistem Informasi Tes Kemampuan Baca, Tulis, dan Hitung (CALISTUNG) Untuk Siswa Sekolah Dasar Berbasis Android; Indra Yusuf Kinarta, 102410101112 2014, 81 HALAMAN; Program Studi Sistem Informasi Universitas Jember.

Pendidikan merupakan hal yang sangat penting dalam kehidupan. Manusia membutuhkan pendidikan untuk bisa mendapatkan ilmu pengetahuan untuk berkembang menjadi tahap seorang anak untuk mulai mengembangkan kemampuannya dalam bidang yang ditekuni selama hidupnya. Pendidikan selalu diberikan sedini mungkin. Pendidikan yang paling mendasar adalah pendidikan yang diajarkan oleh orang tua dirumah. Orang tua bertindak sebagai pendidik awal, dengan memberi rangsangan pendidikan untuk membantu pertumbuhan dan perkembangan rohani dan jasmani agar anak memiliki kesiapan dalam memasuki pendidikan lebih lanjut. Biasanya pendidikan dari orang tua dimulai dari usia 0-6 tahun, setelah itu maka akan dilanjutkan ke jenjang TK, SD, SMP, SMA dan Perguruan Tinggi. Pada jenjang pendidikan SD anak sudah diajarkan mata pelajaran dasar mulai dari membaca, menulis, dan berhitung, kegiatan diatas biasa disingkat dengan CALISTUNG. Calistung merupakan dasar untuk bisa menguasai mata pelajaran selanjutnya, seperti berhitung dalam matematika, membaca dalam bahasa indonesia, dan menulis dalam semua kegiatan. Untuk mengukur seberapa tingkat kemampuan seorang anak maka dibuatlah tes calistung, sehingga diharapkan dapat membauat guru, orang tua maupun anak bisa mengetahui seberapa besar kemampuan calistung. Teknologi yang semakin pesat dibidang jaringan dan *mobile* menjadikan tes calistung juga bisa dibuat dalam bentuk aplikasi yang berbasis android. Kemudahan dan efektifitas yang diberikan kepada *user* sangat banyak, karena bisa di akses kapan saja dan dimana saja. Dengan aplikasi tes calistung yang berbasis android guru bisa dengan mudah membuat soal dan bisa dengan mudah melihat nilai tes siswa.

# PRAKATA

Puji syukur ke hadirat Allah SWT atas segala rahmat dan karunia-Nya, sehingga penulis dapat menyelesaikan skripsi dengan judul "Pengembangan Sistem Informasi Swamedikasi Menggunakan Metode TOPSIS". Skripsi ini disusun untuk memenuhi salah satu syarat menyelesaikan pendidikan Strata Satu (S1) pada Program Studi Sistem Informasi Universitas Jember.

Penyusunan skripsi ini tidak lepas dari bantuan berbagai pihak. Oleh karena itu, penulis menyampaikan terima kasih kepada:

1. Prof. Drs. Slamin, M.CompSc., Ph.D., selaku Ketua Program Studi Sistem Informasi Universitas Jember.
2. Prof. Drs. Slamin, M.CompSc., Ph.D., selaku Dosen Pembimbing Utama dan M. Arief Hidayat, S.Kom., M.Kom selaku Dosen Pembimbing Anggota yang telah meluangkan waktu, pikiran, dan perhatian dalam penulisan skripsi ini.
3. Windi Eka Retnani, S.Kom., MT., selaku Dosen Pembimbing Akademik yang telah membimbing selama penulis menjadi mahasiswa.
4. Seluruh Bapak dan Ibu dosen beserta staf karyawan di Program Studi Sistem Informasi Universitas Jember.
5. Ayahanda Sholikin dan Suhartatik yang telah sangat memotivasi penulis.
6. Adikku Indri Yunia Kinarti yang selalu memberi semangat.
7. Ayu Priyanti yang selalu memberi motivasi penulis.
8. Keluarga besar Uklam Foundation Brian, Awang, Rasya, Syafiq, Indra, Doni, Hamdan, Yusa, Nay, Hani, Anggi, Ain, Pipit.
9. Keluarga Besar Himpunan Mahasiswa Sistem Informasi (HIMASIF) periode 2011-2012 dan UKM Kesenian.
10. Teman-teman mahasiswa Program Studi Sistem Informasi Universitas Jember terkhusus zerone.

11. SDN Jember Lor 1.
12. *Basecamp* Perum Mastrip E22 Jember Pak Budiarto sekeluarga.
13. Semua pihak yang tidak dapat disebutkan satu-persatu.

Dengan harapan bahwa penelitian ini nantinya akan terus berlanjut dan berkembang kelak, penulis juga menerima segala kritik dan saran dari semua pihak demi kesempurnaan skripsi ini. Akhirnya penulis berharap, semoga skripsi ini dapat bermanfaat.

Jember, November 2014

Penulis

# DAFTAR ISI

# DAFTAR GAMBAR

# DAFTAR TABEL

# BAB 1. PENDAHULUAN

Bab ini merupakan bab awal dari tugas akhir yang didalamnya terdapat latar belakang, rumusan masalah, tujuan, manfaat, dan sistematika penulisan.

## 1.1 Latar Belakang

Sampah adalah salah satu permasalahan yang sampai saat ini belum terpecahkan khususnya bagi kota-kota besar di Indonesia seperti Surabaya. Permasalahan ini timbul terutama karena besarnya volume sampah dan keterbatasan lahan untuk pembuangan akhir. Diiringi dengan pertumbuhan penduduk yang cukup tinggi. Dimana hal ini ditunjang pula oleh adanya teknis pengelolaan sampah yang masih konvensional.

Umumnya pengumpulan sampah di Tempat Pembuangan Sementara (TPS) dilakukan secara tercampur. Sedangkan pengangkutan sampah baik dari sumber maupun dari TPS ke Tempat Pembuangan Akhir (TPA) tidak dilakukan setiap hari. Timbunan sampah lebih dari dua hari ini menimbulkan bau tak sedap, lalat, dan lindi yang dapat meluber ke jalan.

Permasalahan ini dapat disebabkan karena letak TPA yang sangat jauh dari TPS. Bahkan di kota-kota besar cukup sulit untuk menemukan lahan yang bisa dijadikan TPA. Tidak jarang penempatan TPA harus dilakukan di pinggiran kota bahkan luar kota. Di Surabaya dimana wilayahnya terbagi atas 5 bagian yaitu Surabaya Pusat, Utara, Barat, Timur, dan Selatan, penempatan TPA dilakukan di wilayah Surabaya bagian barat (TPA Benowo). Karena masih tersedianya lahan yang cukup luas untuk pembangunan TPA dan letaknya yang berjauhan dengan daerah pemukiman. Hal ini menimbulkan masalah mengenai jauhnya jarak yang harus ditempuh untuk mengangkut sampah. Contohnya dari TPS di wilayah Surabaya Timur seperti kecamatan Sukolilo yang letaknya terhitung cukup jauh dari TPA.

Belum optimalnya sistem pengangkutan sampah akan berdampak langsung terhadap biaya yang dibutuhkan. Untuk masalah jalur perjalanan pengangkutan

sampah di Kota Surabaya, biasanya sopir hanya memperkirakan sendiri rute yang ditempuh yang menurutnya terdekat. Sehingga tidak akan diketahui secara pasti jumlah biaya perjalanan yang dikeluarkan selama perjalanan. Secara tidak langsung akan terjadi pembengkakan anggaran yang dikeluarkan oleh Dinas Kebersihan Kota Surabaya.

Usaha perbaikan sistem pengangkutan sampah agar menjadi lebih efisien dan ekonomis maka perlu diterapkan sebuah sistem. Mempelajari pola pengangkutan sampah yang diterapkan saat ini untuk dianalisis tingkat optimalitasnya dan pemakaian suatu algoritma untuk menentukan rute pengangkutan sampah. Ditinjau berdasarkan rute yang paling ekonomis seperti misalnya, ACO.

ACO adalah salah satu algoritma alternatif yang dapat digunakan untuk penentuan jalur terdekat. Selain prosesnya cepat dan memberikan hasil yang bisa diterima, ACO juga mampu memberikan suatu solusi pada waktu kapanpun. Mengingat prinsip algoritma yang didasarkan pada perilaku koloni semut dalam menemukan jarak perjalanan paling pendek tersebut, ACO sangat tepat digunakan untuk diterapkan dalam penyelesaian masalah optimasi, salah satunya adalah untuk penentuan jarak terdekat, yang dalam masalah ini mencari rute terpendek untuk memangkas waktu dan biaya yang diperlukan.

## 1.2 Perumusan Masalah

Berdasarkan latar belakang diatas, maka ditemukan beberapa permasalahan yang antara lain yaitu :

1. Bagaimana mencari rute terpendek pada sistem pengangkutan sampah untuk efisiensi waktu dan biaya dengan menggunakan ACO ?
2. Bagaimana membangun sistem untuk mencari rute terpendek dengan menggunakan ACO ?

### 1.3 Tujuan

Tujuan dari penelitian ini adalah sebagai berikut :

1. Menemukan rute terpendek untuk sistem pengangkutan sampah sehingga diharapkan meminimalisir waktu dan biaya dengan menggunakan ACO.
2. Membangun sistem untuk mencari rute terpendek dengan menggunakan ACO.

### 1.4 Batasan Masalah

Batasan masalah pada tugas akhir ini yaitu:

1. Rute yang dihasilkan tidak berdasarkan pada jalan yang saat itu mengalami kemacetan.
2. Parameter optimal berdasarkan pada jarak yang ditempuh.

### 1.5 Sistematika Penulisan Skripsi

1. Pendahuluan

   Bab ini berisi latar belakang, perumusan masalah, tujuan dan manfaat, metodologi penelitian dan sistematika penulisan skripsi.

2. Tinjauan Pustaka

   Bab ini berisi tentang kajian materi, penelitian terdahulu dan informasi apa saja yang digunakan dalam penelitian ini. Dimulai dari kajian pustaka mengenai definisi dari rute terpendek hingga ACO.

3. Metode Penelitian

   Bab ini metode yang akan digunakan selama penelitian. Mulai dari tahap pengumpulan data, perancangan desain sistem, implementasi dan evaluasi sistem.

4. Desain dan Perancangan Sistem

   Bab ini menjelaskan tentang hasil dan pembahsan dari penelitian yang telah dilakukan. Menggambarkan dampak apa yang terjadi pada saat sebelum penggunaan sistem dan setelah penggunaan sistem.

5. Hasil dan Pembahasan

Bab ini menjelaskan tentang hasil penelitian yang telah dilakukan beserta pembahasannya.

6. Penutup

   Bab ini berisi kesimpulan penelitian serta saran untuk penelitian selanjutnya.

# BAB 2. TINJAUAN PUSTAKA

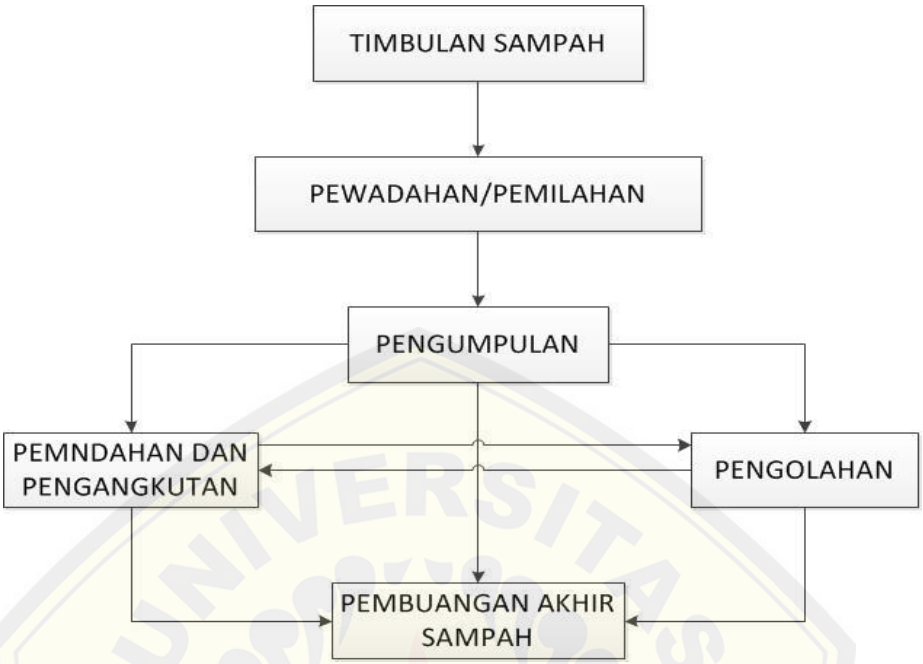
Pustaka yang digunakan pada tugas akhir ini di uraikan pada bab ini. Semua teori yang digunakan diambil dari jurnal, buku, dan internet.

## 2.1 Definisi Sampah

Sampah adalah limbah yang bersifat padat terdiri dari bahan organik dan bahan anorganik yang dianggap tidak berguna lagi dan harus dikelola agar tidak membahayakan lingkungan dan melindungi investasi pembangunan (SNI 19-2454-2000). Sampah padat adalah semua barang sisa yang ditimbulkan dari aktivitas manusia dan binatang yang secara normal padat dan dibuang ketika tak dikehendaki atau sia-sia (Tchobanoglous, Theisen dan Vigil,1993). Sampah adalah limbah yang berbentuk padat dan juga setengah padat, dari bahan organik dan atau anorganik, baik benda logam maupun benda bukan logam, yang dapat terbakar dan yang tidak dapat terbakar. Bentuk fisik benda-benda tersebut dapat berubah menurut cara pengangkutannya atau cara pengolahnnya.

## 2.2 Pengelolaan Sampah di Surabaya

Pengelolaan sampah adalah suatu bidang yang berhubungan dengan pengendalian bagaimana sampah dihasilkan, penyimpanan, pengumpulan, pengangkutan, pengolahan dan pembuangan sampah yang menggunakan suatu cara yang sesuai dengan prinsip-prinsip pewadahan, pengumpulan, TPS. Bila salah satu kegiatan tersebut terputus atau tidak tertangani dengan baik, maka akan menimbulkan masalah kesehatan, banjir atau genangan, pencemarahan air tanah, dan estetika. Diagram teknik operasional pengelolaan persampahan dapat dilihat pada Gambar 2.1




Gambar 2.1 Diagram Teknik Operasional Pengelolaan Persampahan (SK. SNI 19-2454-2002)

**2.3 Peraturan Terkait dengan Pengangkutan Sampah**

Pengangkutan sampah menurut UU no 18 Tahun 2008 tentang Pengelolaan Sampah, merupakan bagian dari penanganan sampah. Pengangkutan didefinisikan sebagai dalam bentuk membawa sampah dari sumber dan atau dari tempat penampungan sampah sementara atau dari TPS 3R (reduce, reuse, recycle) menuju ke tempat pengolahan sampah terpadu atau tempat pemrosesan akhir. Beberapa acuan normatif juga mencantumkan tentang pengaturan pengangkutan sampah, antara lain:

1. Pedoman Standar Pelayanan Minimal Pedoman Penentuan Standar Pelayanan Minimal Bidang Penataan Ruang, Perumahan dan Pemukiman dan Pekerjaan Umum (Keputusan Menteri Permukiman Dan Prasarana Wilayah No. 534/KPTS/M/2001). Pedoman ini mencakup pelayanan minimal untuk pengelolaan sampah secara umum dalam wilayah pemukiman perkotaan dimana 80% dari total jumlah penduduk terlayani terkait dengan pengelolaan sampah.

Khusus untuk pengangkutan dicantumkan bahwa jenis alat angkut mempengaruhi pelayanan, sebagai berikut:

a. *Dump truck* dengan kapasitas 6 $m^3$ dapat melayani pengangkutan untuk 700 kk – 1000 kk sedangkan dengan kapasitas 8 $m^3$ untuk 1500 kk – 2000 kk (jumlah ritasi 2-3/ hari).
b. *Armroll truck* dengan kontainer 8 $m^3$ juga dapat melayani 2000 kk – 3000 kk (jumlah ritasi 3-5/hari). *Armroll truck* merupakan kendaraan angkut yang dilengkapi sistem hidrolis untuk mengangkat bak dan membongkar muatannya. Pengisian muatan masih tetap secara manual dengan tenaga kerja. Truk ini memiliki kapasitas yang bervariasi, yaitu 6$m^3$, 8$m^3$, dan 10$m^3$. Dalam pengangkutan sampah, efisiensi penggunaan *armroll truck* dapat dicapai apabila memenuhi beberapa kriteria yaitu jumlah trip atau ritasi perhari minimum 5 dan jumlah kru maksimum 1 orang. Agar tidak mengganggu lingkungan selama perjalanan ke TPA, kontainer sebaiknya memiliki tutup dan tidak rembes sehingga lindi tidak mudah tercecer. Kontainer yang tidak memiliki tutup sebaiknya dilengkapi dengan tutup terpal selama pengangkutan.
c. *Compactor truck* 8 $m^3$ mampu melayani 2500 kk

2. SNI 19-2454-2002, tata cara teknik operasional pengelolaan sampah perkotaan. SNI ini mengatur tentang pola pengangkutan dan operasional pengangkutan.
3. SNI 03-3243-2008, pengelolaan sampah pemukiman. SNI mengatur tentang kebutuhan sarana untuk pengangkutan sampah yang dipengaruhi oleh tipe rumah dan tingkat pelayanan serta jenis alat angkut.

**2.4 Sistem Pengangkutan Sampah**

Menurut materi pelatihan berbasis kompetensi bidang persampahan pada tahun 2010 pengangkutan dimaksudkan sebagai kegiatan operasi yang dimulai dari titik pengumpulan terakhir dari suatu siklus pengumpulan sampai ke TPA atau TPS pada pengumpulan dengan pola individual langsung atau dari tempat pemindahan, penampungan sementara atau tempat penampungan komunal sampai ke tempat

pengolahan atau pembuangan akhir. Sehubungan dengan hal tersebut, metode pengangkutan serta peralatan yang akan digunakan tergantung dari pola pengumpulan yang digunakan. Permasalahan yang dihadapi dalam pengangkutan sampah adalah sebagai berikut :

1. Penggunaan waktu kerja yang tidak efisien.
2. Penggunaan kapasitas muat kendaraan yang tidak tepat.
3. Rute pengangkutan yang tidak efisien.
4. Tingkah laku petugas.
5. Aksesbilitas yang kurang baik.

Berdasarkan atas operasional pengelolaan sampah, maka pengangkutan ini merupakan tanggung jawab dari pemerintah kota atau kabupaten. Sedangkan pelaksana adalah pengelola kebersihan dalam suatu kawasan atau wilayah, badan usaha dan kemitraan. Sangat tergantung dari struktur organisasi di wilayah yang bersangkutan. Sebagai contoh, misal dalam suatu wilayah kota terdapat Dinas Kebersihan Dan Pertamanan, maka tanggung jawab pengelolaan sampah ada dibawah dinas ini.

Pengangkutan sampah adalah subsistem yang membawa sampah dari tempat pemindahan atau sumber sampah secara langsung TPA. Pengangkutan sampah merupakan komponen yang paling penting dan membutuhkan perhitungan yang cukup teliti untuk mendapatkan sistem pengangkutan yang efisien dan efektif maka pengangkutan sampah sebaiknya mengikuti kriteria sebagai berikut:

1. Menggunakan rute pengangkutan sampah yang sependek mungkin dan dengan hambatan yang sekecil mungkin.
2. Menggunakan kendaraan angkut dengan kapasitas atau daya tampung yang semaksimal mungkin.
3. Menggunakan kendaraan angkut yang hemat bahan bakar.
4. Dapat memanfaatkan waktu kerja semaksimal mungkin dengan meningkatkan jumlah beban kerja atau ritasi pengangkutan.

## 2.5 Rute Pengangkutan

Rute pengangkutan dibuat agar pekerja dan peralatan dapat digunakan secra efektif. Umumnya rute pengumpulan dicoba-coba, karena rute tidak dapat digunakan pada semua kondisi. Pedoman yang dapat digunakan dalam membuat rute sangat tergantung dari beberapa faktor yaitu:

1. Peraturan lalu lintas yang ada
2. Pekerja, ukuran dan tipe alat angkut
3. Jika memungkinkan, rute dibuat mulai dan berakhir di dekat jalan utama
4. Pada daerah berbukit, usahakan rute dimulai dari atas dan berakhir dibawah
5. Rute dibuat agar kontainer/TPS terakhir yang akan diangkut yang terdekat ke TPA
6. Timbulan sampah pada daerah sibuk/lalu lintas padat diangkut sepagi mungkin
7. Daerah yang menghasilkan timbulan sampah terbnayak, diangkut lebih dahulu
8. Daerah yang menghasilkan timbulan sampah sedikit, diusahakan terangkut dalam hari yang sama.

## 2.6 Operasional Pengangkutan

Pengaturan rute pengangkutan sangat penting dalam penanganan sampah di pemukiman karena terkait dengan penyimpanan sampah di TPS. Jika pengangkutan mengalami kendala dan tidak dapat mengangkut sampah sesuai dengan jadwal pengangkutan, maka akan terjadi penumpukan sampah di TPS dan secara langsung akan mempengaruhi kondisi lingkungan sekitar TPS. Terkait dengan permasalahan rute pengangkutan maka perlu adanya upaya untuk membuat rute secara efisien. Selain itu operasional pengangkutan juga akan mempengaruhi waktu pengangkutan sampah. Ada beberapa faktor yang mempengaruhi operasional pengangkutan yaitu:

1. Pola pengangkutan yang digunakan
2. Alat angkut yang digunakan
3. Jumlah personil
4. Lokasi TPS atau TPA

Operasional untuk sistem kontainer tetap yaitu pola pengumpulan tidak langsung dan pola *transfer station*. Untuk pola pengumpulan tidak langsung adalah pola yang berkaitan dengan pengumpulan tidak langsung baik individual maupun komunal. Berikut adalah pola nya :

1. Petugas menyiapkan kendaraan sesuai ketentuan
2. Petugas mendatangi lokasi TPS, menerima muatan sampah dari gerobak pengumpul sampai penuh
3. Truk menuju TPA untuk membongkar sampahnya
4. Truk menuju ke lokasi TPS atau transfer depo selanjutnya sesuai rute yang direncanakan dan melanjutkan operasinya
5. Setelah seluruh rute diselesaikan, truk dicuci dan kembali ke pool

Untuk pola *transfer station*, pola ini muncul karena jarak dari TPS menuju ke TPA sangat jauh, sehingga untuk membantu pola pengangkutan dari TPS menuju ke *transfer station* kemudian baru menuju ke TPA. Truk untuk mengangkut menuju ke TPS yang mempunyai ukuran kontainer lebih kecil antara 6-10 $m^3$ kemudian di transfer station truk trailer dengan kapasitas 80-100 $m^3$ digunakan untuk mengangkut sampah ke TPA. Operasional pola ini adalah :

1. Trailer bergerak menuju ke lokasi *transfer station*
2. Trailer menerima muatan sampah berupa kontainer kapasitas besar
3. Trailer membawa kontainer ke TPA untuk dibongkar
4. Trailer kembali ke lokasi transfer, demikian seterusnya sampai pengangkutan diselesaikan.

**2.7 Efisiensi Waktu dan Biaya**

Pengertian efisiensi menurut Mulyamah (1987) yaitu: “efisiensi merupakan suatu ukuran dalam membandingkan rencana penggunaan masukan dengan penggunaan yang direalisasikan atau perkataan lain penggunaan yang sebenarnya”. Sedangkan pengertian efisiensi menurut SP. Hasibuan (1984) yang mengutip pernyataan H.Emerson adalah : “efisiensi adalah perbandingan yang terbaik antara

*input* (masukan) dan *output* (hasil antara keuntungan dengan sumber-sumber yang dipergunakan), seperti halnya juga hasil optimal yang dicapai dengan penggunaan sumber yang terbatas. Dengan kata lain hubungan antara apa yang telah diselesaikan". Maka, efisiensi adalah perbandingan *output* dan *input*. Disebut memiliki sebuah efisiensi 100% jika *output* (hasil) sesuai dengan *input* (sumber daya) yang diberikan.

Menurut walikota Surabaya Tri Rismaharini, diperlukan anggaran yang cukup besar untuk biaya angkut sampah ke TPA, yang dapat ditekan bila sampah dapat ditekan mulai dari sumbernya. Menurutnya jika dilihat dari pengelolaan sampah, sebetulnya justru yang paling besar itu adalah untuk biaya angkut. Biaya angkutan bisa mencapai 50 persen dari anggaran, karena itu jika merujuk pada rencana menyelesaikan sampah dari sumbernya maka biaya angkut dapat dipangkas (mongabay,2014).

Bahan bakar yang disediakan setiap harinya untuk *dump truck* adalah 14 liter solar, sehingga jika dirupiahkan dengan asumsi harga solar Rp. 5.500/L yaitu sebesar Rp. 63.000,-. Dengan bahan bakar ini, setiap harinya truk hanya mampu menempuh jarak dari pangkalan ke daerah pelayanan lalu menuju ke TPA dan kembali lagi ke pangkalan, rata-rata dua kali ritasi per hari. Setiap hari *dump truck* beroperasi pada jam 06.30 – 13.30 WIB. Rata-rata jarak tempuh pulang – pergi yaitu 22 km, sehingga setiap liter solar dapat menempuh 1,57 Km.

Sedangkan untuk *armroll truck,* bahan bakar yang disediakan setiap harinya adalah 60 liter solar, sehingga jika dirupiahkan dengan asumsi harga solar Rp5.500/L yaitu sebesar Rp 270.000,-. Dengan bahan bakar ini, setiap harinya truk mampu menempuh jarak dari pangkalan ke TPS pelayanan lalu menuju ke TPA kemudian ke TPS berikutnya, kembali ke TPA, dan seterusnya hingga 6 kali ritasi/hari dan kembali lagi ke pangkalan atau setara dengan kurang lebih 125,40 Km. dengan jarak tempuh tersebut, setiap liter solar dapat menempuh 2,09 Km. Daerah pelayanan *armroll truck* dan *dump truck* dapat dilihat pada Tabel 2.1. Dengan waktu pelayanan 06.00 – 14.00

WIB setiap harinya. Daftar lokasi TPA dan TPS serta jumlah ritasi truk per hari dapat dilihat pada Tabel 2.1.

*Tabel 2.1 Lokasi dan Ritasi* Armroll Truck *dan* Dump Truck

| **Lokasi TPS** | **Jumlah Rit** | **Rute/Tujuan** |
|---|---|---|
| TPS Kalijudan | 2 rit/hari | Pool Tanjungsari |
| | | TPS Kalijudan |
| | | TPA |
| | | TPS Pacar Keling |
| | | TPA |
| | | Pool Tanjungsari |
| TPS Wisma Permai | 2 rit/hari | Pool Tanjungsari |
| | | TPS Kedinding |
| | | TPA |
| | | TPS Wisma Permai |
| | | TPA |
| | | Pool Tanjungsari |
| Super Depo Organik | 2 rit/hari | Pool Tanjungsari |
| | | TPS Kayoon |
| | | TPA |
| | | Super Depo Organik |
| | | Rumah Kompos |
| | | Pool Tanjungsari |
| Super Depo Residu | 1 rit/hari | Pool Tanjungsari |
| | | Super Depo Residu |
| | | TPA |
| TPS Darmahusada | 2 rit/hari | Pool Tanjungsari |
| | | TPS Lidah |
| | | TPA |
| | | TPS Darmahusada |
| | | TPA |
| | | Pool Tanjungsari |
| TPS Manyar Sabrangan | 2 rit/hari | Pool Tanjungsari |
| | | TPS Manyar Sabrangan |
| | | TPA |
| | | TPS Kalibokor |
| | | TPA |
| | | Pool Tanjungsari |
| TPS Kejawan Putih Tambak | 2 rit/hari | Pool Tanjungsari |
| | | TPS Kejawan Putih |

| | | |
|---|---|---|
| | | Tambak |
| | | TPA |
| | | TPS Simorukun |
| | | TPA |
| | | Pool Tanjungsari |

Sumber : Hasil Analisa (2014)

**2.8 Rute Terpendek**

Rute terpendek adalah lintasan minimum yang diperlukan untuk mencapai suatu tempat dari tempat tertentu. Lintasan minimum yang dimaksud dapat dicari dengan menggunakan graf. Graf yang digunakan adalah graf yang berbobot, yaitu graf yang setiap sisinya diberikan suatu nilai atau bobot. Dalam kasus ini, bobot yang dimaksud berupa jarak dan waktu yang diperlukan. Terdapat beberapa algoritma yang sering digunakan untuk menemukan rute terpendek, antara lain:

1. Algoritma Greedy

Algoritma Greedy adalah algoritma yang memecahkan masalah langkah demi langkah. Algoritma Greedy membentuk solusi langkah per langkah sebagai berikut :

a. Terdapat banyak pilihan yang perlu dieksplorasi pada setiap langkah solusi. Oleh karena itu, pada setiap langkah harus dibuat keputusan yang terbaik dalam menentukan pilihan. Keputusan yang telah diambil pada suatu langkah tidak dapat diubah lagi pada langkah selanjutnya.
b. Pendekatan yang digunakan di dalam algoritma Greedy adalah membuat pilihan yang terlihat memberikan perolehan terbaik, yaitu dengan membuat pilihan optimum lokal pada setiap langkah dan diharapkan akan mendapatkan solusi optimum global.

Algoritma Greedy didasarkan pada pemindahan *edge* per *edge* dan pada setiap langkah yang diambil tidak memikirkan konsekuensi ke depan, Greedy tidak beroperasi secara menyeluruh terhadap semua alternatif solusi yang ada serta sebagian *masalah Greedy* tidak selalu berhasil memberikan solusi yang benar-benar optimum tapi pasti memberikan solusi yang mendekati nilai optimum (Defindal,2003).

2. Algoritma Djikstra

Algoritma Djikstra ditemukan oleh Edsger W. Dijkstra yang merupakan salah satu varian bentuk algoritma populer dalam pemecahan persoalan yang terkait dengan masalah optimisasi dan bersifat sederhana. Algoritma ini menyelesaikan masalah mencari sebuah lintasan terpendek (sebuah lintasan yang mempunyai panjang minimum) dari *verteks* a ke *verteks* z dalam graf berbobot, bobot tersebut adalah bilangan posistif jadi tidak dapat dilalui oleh node negatif, namun jika terjadi demikian, maka penyelesaian yang diberikan adalah *infinity*.

Algoritma Djikstra melibatkan pemasangan label pada *verteks*. Misalkan L(v) menyatakan label dari *verteks* v. Pada setiap pembahasan, beberapa *verteks* mempunyai label sementara dan yang lain mempunyai label tetap. Misalkan T menyatakan himpunan *verteks* yang mempunyai label sementara. Dalam menggambarkan algoritma tersebut *verteks-verteks* yang mempunyai label tetap akan dilingkari. Selanjutnya, jika L(v) adalah label tetap dari *verteks* v, maka L(v) merupakan panjang lintasan terpendek dari a ke v. Sebelumnya semua *verteks* mempunyai label sementara. Setiap iterasi dari algoritma tersebut mengubah status satu label dari sementara ke tetap, sehingga algoritma dapat berakhir ketika z menerima sebuah label tetap. Pada bagian ini L(z) merupakan panjang lintasan terpendek dari a ke z. Pada algoritma Dijkstra node digunakan, karena algoritma Dijkstra menggunakan diagram pohon untuk penentuan jalur lintasan terpendek dan menggunakan graf yang berarah (Muttaqin,2012).

3. Algoritma A*

Algortima ini pertama kali ditemukan pada tahun 1968 Peter Hart, Nils Nilsson dan Bertram Raphael. A* sebenarnya merupakan algoritma pengembangan dari BFS (*Breadth-First-Search*) untuk menemukan jalan dengan biaya terkecil dari titik awal ke titik tujuan (bisa lebih dari 1 titik tujuan) (Rudy Adipranata,2011). Algoritma A* diterapkan untuk mencari lintasan terpendek pada graf berarah. Namun, algoritma ini tetap benar untuk graf yang tak berarah. Pada setiap langkah, ambil sisi yang berbobot minimum yang menghubungkan sebuah simpul yang telah dipilih dengan

sebuah simpul lain yang belum terpilih. Lintasan dari simpul asal ke simpul yang baru haruslah merupakan lintasan yang terpendek diantara semua lintasannya ke simpul-simpul yang belum terpilih. Misalkan kita tentukan S adalah simpul awal dan T adalah simpul akhir, akan dicari lintasan terpendek antara simpul S dan simpul T.

4. ACO

ACO diadopsi dari perilaku koloni semut yang dikenal sebagai sistem semut (Dorigo,1996). 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. Semakin banyak semut yang melalui suatu lintasan, maka akan semakin jelas bekas jejak kaki nya. Hal ini akan menyebabkan lintasan yang dilalui semut dalam jumlah sedikit, semakin lama akan semakin berkurang kepadatan semut yang melewatinya, atau bahkan semua semut akan melalui lintasan tersebut.

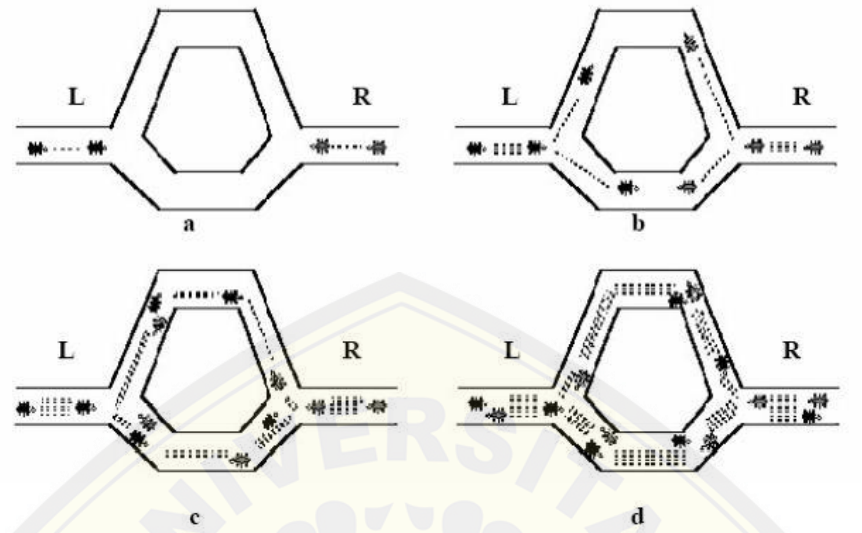
Algoritma ACO telah banyak diaplikasikan dalam berbagai bidang yang mencakup beberapa persoalan, yaitu :

a. *Traveling Salesman Problem* (TSP), yaitu mencari rute terpendek dalam sebuah graf menggunakan rute Hamilton.
b. *Quadratic Assigment Problem* (QAP), yaitu menugaskan sejumlah *n resources* untuk ditempatkan pada sejumlah *m* lokasi dengan meminimalisasi biaya penugasan (*assigment*).
c. *Job-Shop Scheduling Problem* (JSP) juga salah satu contoh aplikasi ACO, yaitu untuk mencari lintasan sejumlah *n* pekerjaan menggunakan sejumlah *m* mesin demikian sehingga seluruh pekerjaan diselesaikan dalam waktu yang seminimal mungkin.
d. dll.

ACO diperkenalkan oleh Moyson dan Manderick dan secara meluas dikembangkan oleh Marco Dorigo, merupakan teknik probabilistik untuk menyelesaikan masalah komputasi dengan menemukan jalur terbaik melalui grafik.

Optimasi ini terinspirasi oleh perilaku semut dalam menemukan jalur dari koloninya menuju makanan (Wardy,2007). Cara kerja semut untuk mencari jalur optimal adalah koloni semut dapat menemukan rute terpendek antara sarang dan sumber makanan berdasarkan jejak kaki pada lintasan yang telah dilalui. Semut selalu meninggalkan cairan yang bernama feromon pada tiap jejak kakinya. Semakin banyak semut yang melewati suatu lintasan maka akan semakin jelas bekas jejak kakinya dan sebaiknya lintasan yang dilalui semut dalam jumlah sedikit semakin lama akan semakin berkurang kepadatan semut yang melewatinya. Algoritma dari ACO ini yaitu :

a. Semut berkeliling secara acak hingga menemukan makanan.
b. Ketika sudah menemukan makanan mereka akan kembali pada koloninya sambil memberikan tanda dengan jejak feromonnya.
c. Jika semut-semut lain menemukan jalur tersebut, mereka tidak akan bepergian dengan acak lagi, melainkan akan mengikuti jejak tersebut.
d. Mereka kembali dan menguatkannya jika pada akhirnya mereka pun menemukan makanan.
e. Bila seekor semut yang secara tidak sengaja menemukan jalur optimal akan menempuh jalur ini lebih cepat dari rekan-rekannya, melakukan *round-trip* lebih sering, dan dengan sendirinya meninggalkan feromon lebih banyak dari jalur-jalur yang lebih lambat ditempuh.
f. Feromon yang berkonsentrasi tinggi pada akhirnya akan menarik semut-semut lain untuk berpindah jalur, menuju jalur paling optimal, sedangkan jalur lainnya akan ditinggalkan.
g. Pada akhirnya semua semut yang tadinya menempuh jalur yang berbeda-beda akan beralih ke sebuah jalur tunggal yang ternyata paling optimal dari sarang menuju ketempat makanan. Cara kerja ACO dapat dilihat pada Gambar 2.2.




Gambar 2.2 ACO (dorigo, 2004)

Pada Gambar 2.2.a 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 2.2.b dan 2.2.c menunjukkan bahwa kelompok semut berjalan pada kecepatan yang sama dengan meninggalkan feromon di jalan yang telah dilalui. Feromon yang ditinggalkan oleh semut-semut yang melalui jalan atas telah mengalami banyak penguapan karena semut yang melalui jalan atas berjumlah lebih sedikit dari pada yang jalan bawah. Hal ini dikarenakan jarak yang ditempuh lebih panjang daripada yang jalan di bawah. Sedangkan feromon yang berada di jalan bawah, penguapannya cenderung lebih lama. Karena semut yang melalui jalan bawah lebih banyak daripada semut yang melalui jalan atas. Gambar 2.2.d menunjukkan bahwa semut-semut yang lain pada akhirnya memutuskan untuk melewati jalan bawah karena feromon yang ditinggalkan masih banyak. Sedangkan

feromon 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 feromon yang ditinggalkan semakin berkurang bahkan hilang. Dari sinilah kemudian terpilihlah rute terpendek antara sarang dan sumber makanan.

Algoritma ini merupakan algoritma yang paling terkenal untuk mencari lintasan terpendek. Dari sinilah kemudian terpilih jalur terpendek antara sarang dan sumber makanan. Mengingat prinsip algoritma yang didasarkan pada perilaku koloni semut dalam menemukan jarak perjalanan paling pendek tersebut maka algoritma ini sangat tepat digunakan untuk diterapkan dalam penyelesaian masalah optimasi, salah satunya adalah untuk menemukan rute terpendek. Dalam algoritma semut, diperlukan beberapa variabel dan langkah-langkah untuk menentukan rute terpendek (Mutakhiroh, I., Indranto, Hidayat, T., 2007), yaitu :

**Langkah 1 :**

a. Inisialisasi harga parameter-parameter algoritma.

   Parameter-parameter yang di inisialisasikan adalah :

   1. Intensitas jejak semut antar kota dan perubahannya ($\tau$ij)
   2. Banyak kota (n) termasuk koordinat (x,y) atau jarak antar kota (dij)
   3. Kota berangkat dan kota tujuan
   4. Tetapan siklus semut (Q)
   5. Tetapan pengendali intensitas jejak semut ($\alpha$) , nilai $\alpha \geq 0$
   6. Tetapan pengendali visibilitas ($\beta$), nilai $\beta \geq 0$
   7. Visibilitas antar kota = 1/dij (ηij)
   8. Banyak semut (m)
   9. Tetapan penguapan jejak semut ($\rho$), nilai $\rho$ harus > 0 dan < 1 untuk mencegah jejak feromon yang tak terhingga

10. Jumlah siklus maksimum (NCmax) bersifat tetap selama algoritma dijalankan, sedangkan $\tau$ij akan selalu diperbaharui harganya pada setiap siklus algoritma mulai dari siklus pertama (NC=1) sampai tercapai jumlah siklus maksimum (NC=NCmax) atau sampai terjadi konvergasi.

b. Inisialisasi kota pertama setiap semut.

Setelah di inisislaisasi $\tau$ij dilakukan, kemudian m semut ditempatkan pada kota pertama tertentu secara acak.

**Langkah 2 :**

Pengisian kota pertama ke dalam *tabu list*. Hasil inisialisasi kota pertama setiap semut dalam langkah 1 harus diisikan sebagai elemen pertama *tabu list.* Hasil dari langkah ini adalah terisinya elemen pertama *tabu list* setiap semut dengan indeks kota tertentu, yang berarti bahwa setiap *tabu (l)* bisa berisi indeks kota antara *l* sampai *n* sebagaimana hasil inisialisasi pada langkah 1.

**Langkah 3 :**

Penyusunan rute kunjungan setiap semut ke setiap kota. Koloni semut yang sudah terdistribusi ke sejumlah atau setiap kota, akan mulai melakukan perjalanan dari kota pertama masing-masing sebagai kota asal dan salah satu kota-kota linnya sebagai kota tujuan. Kemudian dari kota kedua masing-masing, koloni semut akan melanjutkan perjalanan dengan memilih salah satu dari kota-kota yang tidak terdapat pada $tabu_k$ sebagai kota tujuan selanjutnya. Perjalanan koloni semut berlangsung terus- menerus sampai semua kota satu persatu dikunjungi atau telah menempati $tabu_k$ . jika *s* menyatakan indeks urutan kunjungan, kota asal dinyatakan sebagai $tabu_k(s)$ dan kota-kota lainnya dinyatakan sebagai $\{N\text{-}tabu_k\}$, maka untuk menentukan kota tujuan digunakan persamaan probabilitas kota untuk dikunjungi dapat dilihat pada persamaan 2.1 :

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

Dengan *i* sebagai indeks titik awal dan *j* sebagai indeks titik tujuan.

**Langkah 4 :**

a. Perhitungan panjang rute setiap semut.

Perhitungan panjang rute tertutup *(length closed tour)* atau $L_k$ setiap semut dilakukan setelah satu siklus diselesaikan oleh semua semut. Perhitungan ini dilakukan berdasarkan $tabu_k$ masing-masing dengan persamaan 2.2 :

$$L_k = d_{tabu_k(n),tabu_{k(1)}} + \sum_{s=1}^{n-1} d_{tabu_k(s),tabu_k(s+1)} \ldots..( 2.2)$$

Dengan $d_{ij}$ adalah jarak antara kota *i* ke kota *j* yang dihitung berdasarkan persamaan 2.3 :

$$d_{ij} = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2} \ldots..( 2.3)$$

b. Pencarian rute terpendek

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

c. Perhitungan perubahan harga intensitas jejak kaki pada lintasan 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 dapat dilihat pada persamaan 2.4:

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

Dengan k $\Delta\tau ij$ adalah perubahan harga intensitas jejak kaki semut antar kota setiap semut yang dihitung berdasarkan persamaan 2.5:

$$\Delta\tau_{ij}^{Q} = \frac{Q}{L_k} \ldots..( 2.5)$$

Untuk (i,j) ∈ kota asal dan kota tujuan adalah m tabu$_k$ $\Delta\tau ij$ = , 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 melewatinya. Untuk siklus selanjutnya, semut yang akan melewati lintasan tersebut harga intensitasnya telah berubah. Harga intensitas jejak kaki semut antar kota untuk siklus selanjutnya dihitung dengan persamaan 2.6:

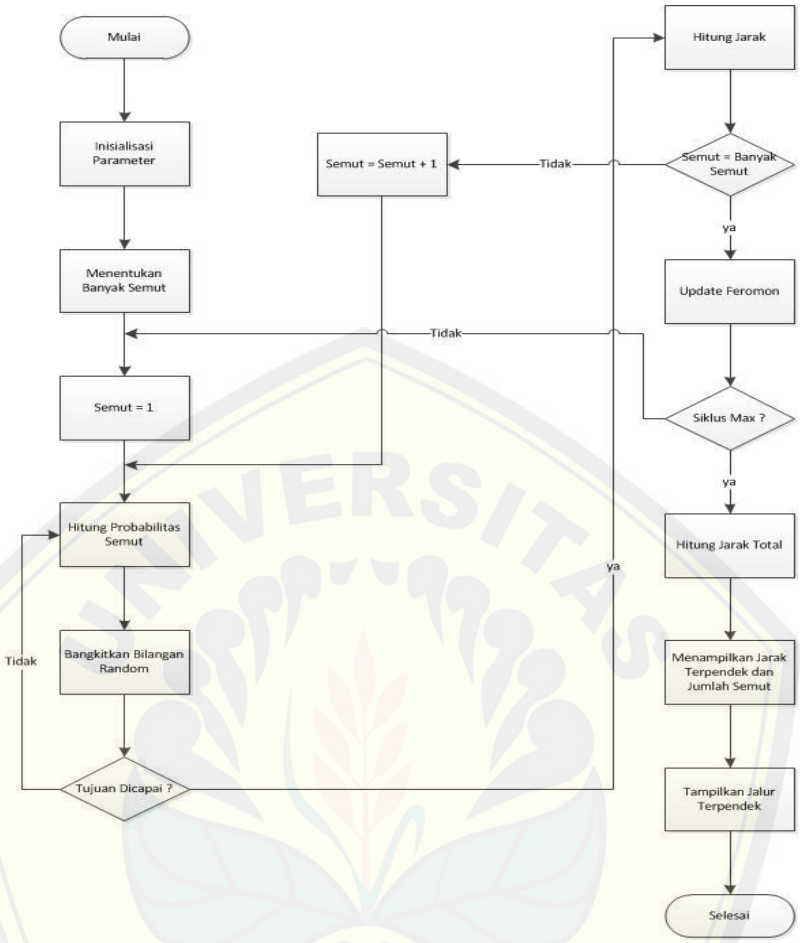
$$\tau ij = \rho.\tau ij + \Delta\tau i \ldots..(2.6)$$

b. Atur ulang harga perubahan intensitas jejak kaki semut antar kota.

Untuk siklus selanjutnya perubahan harga 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. *Flowchart* cara kerja ACO dapat dilihat pada Gambar 2.3.

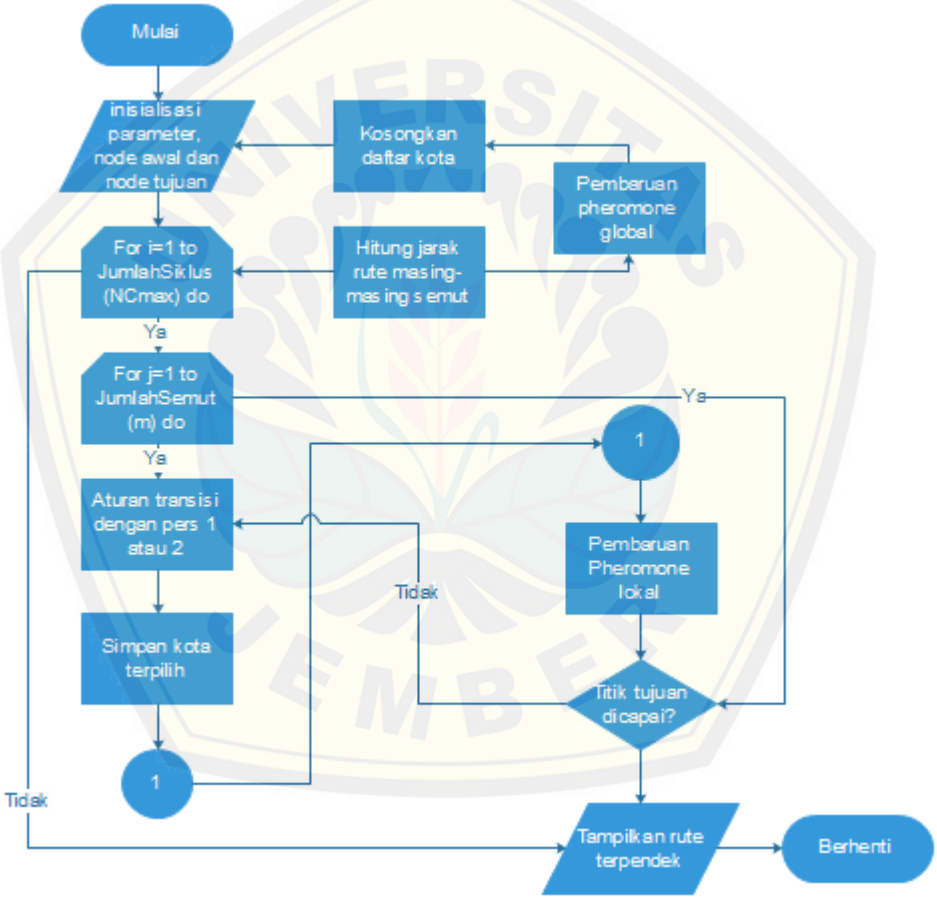



Gambar *2.3 Flowchart* ACO

## 2.9 Ant Colony System

ACS merupakan pengembangan dari ACO. Cara kerja algoritma ini adalah sebagai berikut: sejumlah *m* semut ditempatkan pada sejumlah *n* titik berdasarkan beberapa aturan inisialisasi. Setiap semut membuat sebuah tur dengan menerapkan sebuah aturan transisi status secara berulang kali. Selagi membangun turnya, setiap semut juga memodifikasi jumlah feromon pada *edge-edge* yang dikunjunginya dengan menerapkan aturan pembaruan feromon lokal yang telah disebutkan tadi. Setelah semua semut mengakhiri tur mereka, jumlah feromon yang ada pada *edge-edge* dimodifikasi kembali dengan menerapkan aturan pembaruan feromon global. Dalam membuat tur, semut dipandu oleh informasi *heuristic* (mereka lebih memilih

*edge-edge* yang pendek) dan oleh informasi feromon. Sebuah *edge* dengan jumlah feromon yang tinggi merupakan pilihan yang sangat diinginkan. Kedua aturan pembaruan feromon itu dirancang agar semut cenderung untuk memberi lebih banyak feromon pada *edge-edge* yang harus mereka lewati. Tiga karakteristik utama dari ACS adalah aturan transisi status, aturan pembaruan feromon lokal dan aturan pembaruan feromon global (Verdianto,2013). Gambar 2.4 menggambarkan tentang alur kerja ACS.




Gambar 2.4 *Flowchart* ACS

1. Aturan Transisi Status

Aturan transisi status adalah aturan yang digunakan dalam memilih titik tujuan berikutnya dengan melakukan perhitungan probabilitas masing-masing titik tujuan yang mungkin (Dorigo, 1996). Aturan transisi status yang berlaku pada ACS adalah

sebagai berikut: seekor semut yang ditempatkan pada node *i* memilih untuk menuju ke node *j*. Aturan transisi status digunakan oleh sistem semut, disebut sebagai *random-proportional rule* diberikan oleh persamaan 2.7, yang memberikan probabilitas semut *k* di kota *i* memilih untuk pindah ke node *j*.

$$P_{ij}^{k}=\frac{[\tau_{ij}]^{\alpha}.[\eta_{ij}]^{\beta}}{\Sigma_{k\in[N-tabu_k]}\left[\tau_{ik}i\right]^{\alpha}.\left[\eta_{ik}j\right]^{\beta}} \text{ untuk } j \in \{N-tabu_k\} \ldots (2.7)$$

Dimana:

$P_{ij}^{k}$ = Probabilitas semut *k* memilih untuk berpindah dari node *i* ke node *j*

$\tau_{ij}$= Jumlah feromon pada sisi simpul dari simpul *i* ke simpul *j*

$\eta_{ij}$ = Panjang sisi dari simpul *i* ke simpul *j*

$\tau_{ik}$ = Jumlah feromon pada sisi simpul dari simpul *i* ke simpul *k*

$\eta_{ik}$ = Panjang sisi dari simpul *i* ke simpul *j*

Setelah hasil perhitungan probabilitas kota yang akan dipilih berikutnya selesai, kemudian dicari probabilitas kumulatifnya ($q_k$) dimana q1 = $P_1$ sedangkan $q_k = q_k - 1 + P_k$ untuk k = 2,3,4, ..., n. Kemudian dibangkitkan bilangan acak (v) antara 0 sampai titik ke –k akan terpilih jika $q_k - 1 < v \leq q_k$ .

2. Pembaruan Feromon Lokal

Ketika membangun solusi (tur) dari TSP , semut mengaplikasikan *local updating rule* (pembaruan feromon lokal) (Dorigo, 1996) yang ditunjukkan oleh persamaan 2.8:

$$\tau(i,s) \leftarrow (1-\rho).\tau(i,s) + \rho.\Delta\tau(i,s) \ldots (2.8)$$

Dimana:

$\tau(i,s)$ = tetapan penguapan feromon

$\rho$ = tetapan penguapan feromon

$\Delta\tau$ = perubahan intensitas feromon

Persamaan pembaruan feromon lokal ini diaplikasikan saat semut membangun tur TSP, yaitu ketika melewati *edge* dan mengubah tingkat feromon pada *edge*.

Tujuannya untuk membantu melewati sebuah *edge*, *edge* ini menjadi kurang diinginkan (karena berkurangnya jejak feromon pada *edge* yang bersesuaian).

3. Pembaruan Feromon Global

Pada algoritma ini, pembaruan feromon secara global hanya akan dilakukan oleh semut yang membuat tur terpendek sejak permulaan percobaan. Pada akhir sebuah iterasi, setelah semua semut menyelesaikan tur mereka, sejumlah feromon ditaruh pada ruas-ruas yang lain tidak diubah). Tingkat feromon itu diperbarui dengan menerapkan aturan pembaruan feromon global yang ditunjukkan ole persamaan 2.9:

$$\tau(i,j) \leftarrow (1-\alpha).\tau(i,j) + \alpha.\Delta\tau(i,j) \ldots (2.9)$$

Dimana:

$\tau(i,j)$ = nilai feromon akhir setelah mengalami pembaruan

$\alpha$ = tetapan pengendali feromon

$\Delta\tau$ = perubahan intensitas feromon

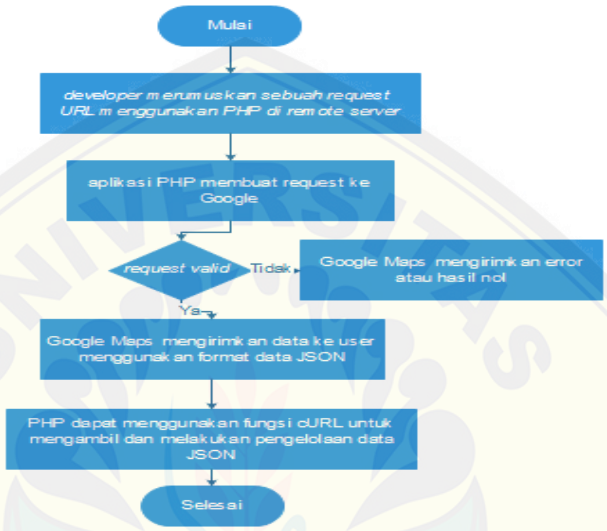
Seperti halnya dalam sistem semut, pembaruan feromon global dimaksudkan untuk menyediakan sejumlah besar feromon untuk kunjungan terpendek. Persamaan 2.10 menyatakan bahwa hanya sebuah *edge* tur terbaik secara global akan menerima penguatan. Jenis lain aturan pembaruan feromon global, yang disebut *iteration-best*, sebagai lawan diatas yang disebut *global-best.* Dalam persamaan juga dengan iterasi terbaik *edge* yang menerima penguatan adalah yang termasuk tur terbaik dari iterasi saat ini. Percobaan telah menunjukkan bahwa perbedaan antara dua skema minimal, dengan preferensi sedikit untuk global terbaik, karena yang digunakan dalam percobaan berikutnya.

## 2.10 Google Map API

Google Maps API merupakan aplikasi antarmuka yang dapat diakses lewat *javascript* agar Google Maps dapat ditampilkan pada halaman web yang sedang dibangun. Ada dua cara untuk mengakses data Google Maps, tergantung dari data yang ingin diambil dan diuraikan dari Google Maps, yaitu:

1. Mengakses data Google Maps tanpa menggunakan API *key*
2. Mengakses data Google Maps menggunakan API *key*

   Pendaftaran API *key* dilakukan dengan data pendaftaran berupa nama domain web yang kita bangun. Gambar 2.5 menggambarkan tentang *flowchart request* URL Google Maps.




Gambar *2.5 Flowchart request* URL Google Maps

# BAB 3. METODOLOGI PENELITIAN

Metode penelitian yang digunakan untuk melakukan penelitian ini meliputi metode pengumpulan data dan metode pengembangan sistem.

## 3.1 Metode Pengumpulan Data

Teknik pengumpulan data yang digunakan untuk mendapatkan kebutuhan sistem yang sesuai dengan kriteria dari *user* maka digunakan teknik pengumpulan data dengan cara observasi. Teknik observasi, wawancara dan studi waktu dan gerak, dilakukan secara pengamatan langsung di studi kasus dan di lapangan. Observasi (*observation*) merupakan teknik atau pendekatan untuk mendapatkan data primer dengan cara mengamati langsung obyek datanya. Pendekatan observasi dapat diklasifikasikan kedalam observasi perilaku (*bahavioral observation*) dan observasi non-perilaku (*nonbehavioral observation*) (Jogiyanto, 2008).

Sesuai dengan data yang ingin diperoleh maka metode pengumpulan data menggunakan observasi non-perilaku (*nonbehavioral observation*). Observasi non-perilaku (*nonbehavioral observation*) terdiri dari analisis catatan (*record analisys*) dapat berupa pengumpulan data baik dari catatan data sekarang atau catatan data historis, analisis kondisi fisik (*physical condition analisys*) dilakukan pada data kondisi fisik seperti fisik sediaan, kondisi keamanan pabrik, dan analisis proses fisik (*physical procss analisys*) dapat berupa observasi pada *time and motion* dari suatu proses, prosedur-prosedur akuntansi dan lain sebagainya.

Penelitian dilakukan di wilayah Surabaya. Pada penelitian ini objek yang diteliti adalah TPS dan TPA, serta rute awal yang dilewati oleh kendaraan angkut sampah seperti *armroll truck*. Wawancara merupakan salah satu cara untuk mengumpulkan data dengan mengajukan pertanyaan secara langsung kepada narasumber. Dalam penelitian ini penulis melakukan wawancara langsung kepada narasumber terkait untuk memperoleh data yang dibutuhkan dalam penyelesaian penelitian.

Pengumpulan data dilakukan untuk memperoleh informasi yang dibutuhkan dalam penelitian. Data-data yang dibutuhkan meliputi:

1. Data Primer

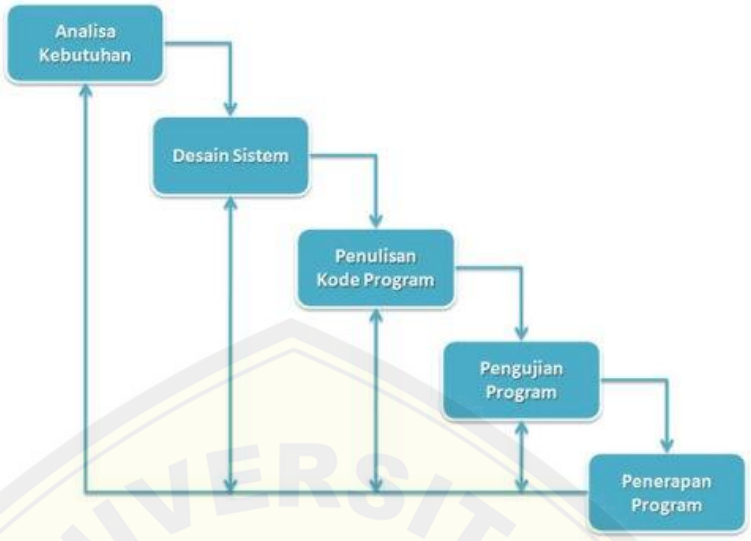
   Data primer adalah data yang diperoleh langsung dari sumbernya dengan melakukan wawancara dan survei secara langsung di tempat penelitian. Data primer yang dibutuhkan pada penelitian ini seperti data TPS yang ada di Kota Surabaya, data sopir dan data kendaraan untuk menentukan fitur-fitur yang akan dibangun pada system, serta sebagai data koordinat yang kan digunakan sebagai data yang dipanggil dalam penerapan algoritma ACO untuk menentukan rute terpendek yang dicari.

2. Data Sekunder

   Data sekunder adalah data yang diperoleh dari sumber lain selain tempat penelitian, tetapi mempunyai keterkaitan yang sangat erat dengan tema penelitian contohnya literature yang berhubungan dengan penelitian.

### 3.2 Metode Pengembangan Sistem

Metode pengembangan sistem yang digunakan dalam penelitian ini adalah perancangan model *waterfall*. Model *waterfall* adalah model yang sederhana dengan aliran sistem yang linier. Dengan modelnya yang sederhana pengaplikasian menggunakan model ini mudah dan mempunyai kelebihan prosesnya teratur dan jadwal pengerjaan lebih menentu. Semua kebutuhan sistem dapat di definisikan secara utuh, eksplisit, dan benar di awal *project* sehingga pembuatan sistem dapat berjalan dengan baik dan teratur. Adapun tahapan dalam perancangan model *waterfall* yaitu analisa kebutuhan, desain sistem, penulisan kode program, pengujian program dan *maintenance* atau penerapan program. Setiap tahap pada model ini dilakukan secara sistematis dan urut seperti pada Gambar 3.1.




Gambar 3.1 Model *Waterfall*

### 3.2.1 Analisa Kebutuhan

Tahap analisis dimulai dengan menelaah data secara keseluruhan yang telah dikumpulkan dari tahap pengumpulan data baik itu studi literatur dan wawancara. Langkah selanjutnya adalah menganalisa data dengan menggunakan ACO. Data input yang digunakan untuk menentukan jalur terpendek adalah data lokasi TPS dan TPA serta data jalan di Kota Surabaya. Data input untuk ACO ada dua yaitu titik awal dan titik tujuan. Output yang dihasilkan dari proses algoritma berupa jalur terpendek dari titik awal menuju titik tujuan, serta info mengenai waktu dan biaya.

### 3.2.2 Desain Sistem

Pada tahap kedua dari model *waterfall* adalah desain sistem. Model desain yang digunakan adalah bahasa pemodelan *Unified Modelling Language* (UML). Penulis menggunakan UML karena sudah mendukung konsep pemodelan *Programming* berbasis *Object Oriented Programming* (OOP) seperti yang akan diterapkan pada tahap penulisan kode program. Dalam UML ada beberapa diagram yang harus dibuat untuk memodelkan sistem sebelum penulisan kode program

menggunakan *software Visual Paradigma for* UML. Adapun diagram-diagram adalah sebagai berikut :

1. *Business Process*
2. *Usecase Diagram*
3. *Usecase Skenario*
4. *Sequence Diagram*
5. *Activity Diagram*
6. *Class Diagram*
7. *Entity Relationship Diagram*

### 3.2.3 Penulisan Kode Program

Sistem yang dibangun akan ditulis dengan bahasa pemrograman *Page Hypertext pro-Processor* (PHP) dengan bantuan *framework Code Igniter* (CI). *Database* yang digunakan adalah MySql.

### 3.2.4 Pengujian Program

Pengujian program akan dilakukan dengan menggunakan dua metode pengujian yaitu *white box testing* dan *black box testing.*

1. *White box testing*

*White box testing* merupakan pengujian pada modul pengkodean program yang dilakukan oleh peneliti. Pengujian ini dilakukan dengan menghitung *independent path* dengan menggunakan *cyclomatic complexity*. Untuk menghitung *cyclomatic complexity* digunakan rumus 3.1:

$$V(G) = E - N + 2 \ldots\ldots ( 3.1)$$

Keterangan :

E = jumlah *edge* grafik alir

N = jumlah node grafik alir

2. *Black Box Testing*

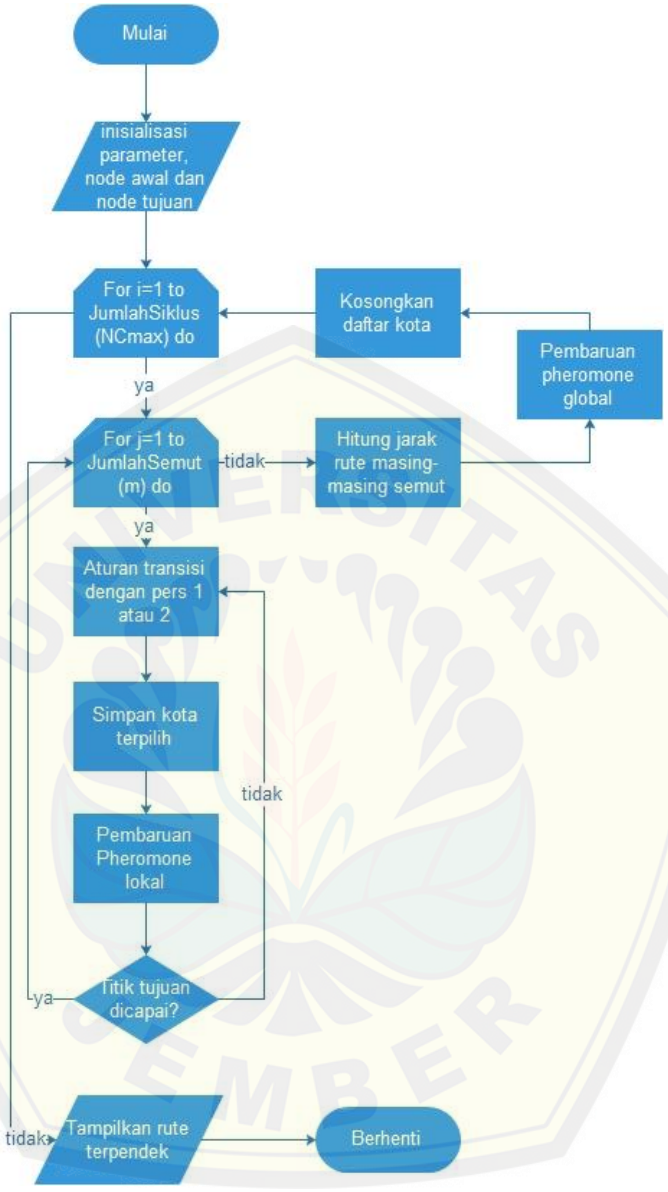
*Black box testing* merupakan pengujian yang menitik beratkan pada uji fungsionalitas dari program yang dibuat. Pengujian ini digunakan untuk menemukan ketidaksesuaian program dengan kebutuhan fungsional maupun non-fungsional. Hal yang perlu dilakukan dalam pengujian ini adalah menguji *interface* dari program untuk memastikan suatu masukan diproses oleh sistem dengan benar dan menghasilkan keluaran yang sesuai dengan perancangan.

### 3.2.5 Penerapan Program

Tahap ini merupakan tahap akhir dari perancangan model *waterfall*. Penerapan program ini dilakukan setelah program yang dibuat oleh penulis selesai dan telah melalui tahap pengujian terlebih dahulu. Selanjutnya dilakukan proses *maintenance* atau perawatan mengatasi masalah *bug* yang muncul setelah aplikasi diserahkan pada *user*.

## 3.3 Penerapan Algoritma ACO untuk Mencari Rute Terpendek

Sistem optimasi rute terpendek ini memiliki beberapa fitur yang diharapkan dapat membantu pekerjaan pada Dinas Kebersihan Kota Surabaya. Salah satu fitur info jalan yang dapat mencari rute terpendek. Mencari rute terpendek tersebut menggunakan titik koordinat yang telah diinputkan sebelumnya. Flowchart penerapan algoritma ACO dapat dilihat pada Gambar 3.2




Gambar 3.2 *Flowchart* ACO dalam Sistem

Penjelasan dari Gambar 3.2 penerapan algoritma ACO pada fitur info jalan sesuai dengan flowchart diatas. Terdapat beberapa tahapan, yang pertama adalah input parameter awal algoritma, input titik awal, dan titik tujuan. Kemudian data tersebut akan digunakan dalam proses perhitungan probabilitas node selanjutnya yang akan dikunjungi. Tahap selanjutnya setelah input parameter yaitu melakukan siklus

pertama. Siklus pertama adalah tur yang dilakukan oleh semut selama satu kali perjalanan. Dalam satu siklus tur semut terdapat beberapa semut yang akan melakukan perjalanan. Setelah semua semut melakukan perjalanan nya maka satu siklus sudah terselesaikan.

Semut dalam mencari keputusan harus melewat node mana yang harus dikunjungi selanjutnya menggunkaan aturan transisi status. Ketika semut telah mengetahui node mana yang harus dikunjungi nya, pada saat yang sama ketika mereka berjalan menuju node tersebut maka mereka juga akan melepaskan feromon nya. Disinilah dihitung tingkat perubahan feromon nya menggunakan perhitungan update feromon lokal. Setelah semua semut melakukan perjalanannya, dan satu siklus telah terpenuhi maka seluruh jalur yang telah dilalui oleh semut akan dihitung nilai feromon final nya. Proses ini yang disebut dengan update feromon global. Dari sinilah akan data dilihat bahwa semut yang memiliki tingkat kenaikan feromon lah yang mempunyai jalur terpendek. Dan yang tidak memiliki kenaikan tingkat feromon tidak akan terpilih menjadi rute terpendek.

# BAB 4. ANALISIS DAN PERANCANGAN SISTEM

Bab ini akan menjelaskan mengenai proses perancangan untuk membuat Sistem Optimasi Rute Terpendek Pengangkutan Sampah di Surabaya Menggunakan Algoritma *Ant Colony Optimization* (ACO). Proses perancangan sistem dimulai dari analisis kebutuhan fungsional dan non-fungsional sistem, dilanjutkan dengan pembuatan *usecase diagram,* skenario, *activity diagram, sequence diagram, class diagram* dan *entity relation diagram* (ERD).

## 4.1 Pengumpulan Data

Pengumpulan data dilakukan dengan beberapa cara diantaranya adalah dengan digitasi, download data spasial dari situs *open source*, wawancara dan observasi atau *tracking* langsung ke lapangan. Data-data yang perlu dikumpulkan untuk sistem optimasi yang akan dibangun adalah data batas Kota Surabaya, data jalan dan data TPS yang berada di Kota Surabaya.

## 4.2 Analisis Data

Analisis data diperlukan untuk mengetahui data yang digunakan pada sistem dapat diimplementasikan dengan benar. Pada tahap analisis data terdapat dua tahap yaitu pengolahan data dan analisis menggunakan ACO.

## 4.3 Perancangan Sistem

Perancangan sistem yang akan digunakan adalah perancangan model *waterfall*. Pada model *waterfall* terdapat lima tahapan perancangan yaitu analisa kebutuhan, desain sistem, penulisan kode program, pengujian program dan *maintenance* program.

### 4.3.1 Analisa Kebutuhan

Tahap analisa kebutuhan adala tahap pengumpulan kebutuhan yang harus dimiliki oleh sistem untuk memenuhi fungsi yang diinginkan oleh pengguna dan kemudian dijabarkan ke dalam sebuah deskripsi yang jelas dan lengkap. Analisa

kebutuhan yang dilakukan adalah menganalisis kebutuhan dengan mengelompokkan ke dalam kebutuhan fungsional dan non-fungsional.

a. Kebutuhan Fungsional

Kebutuhan fungsional merupakan fitur dasar yang harus dimiliki oleh perangkat lunak untuk menerima inputan data kemudian diproses hingga menghasilkan sutau output. Sistem optimasi yang dirancang ini memiliki kebutuhan fungsional yang dapat dilihat pada Tabel 4.1 :

Tabel 4.1 Kebutuhan Fungsional Sistem

| **SRS_ID** | **Identifikasi** |
|---|---|
| SRS_01 | Sistem dapat mencari rute terpendek dari titik awal ke titik tujuan. |
| SRS_02 | Sistem dapat menampilkan peta beserta polyline rute jalan sebagai hasil dari proses algoritma. |
| SRS_03 | Sistem dapat menampilkan arah jalan per tahap sesuai dengan rute jalan yang telah ditemukan. |
| SRS_04 | Sistem dapat menampilkan total waktu dan biaya sebagai bahan perbandingan dengan rute awal yang biasa digunakan. |
| SRS_05 | Sistem dapat memanajemen data *user* seperti tambah, edit, dan hapus data. |
| SRS_06 | Sistem dapat membedakan hak akses *user* melalui proses login. |
| SRS_07 | Sistem dapat mengedit data harga bahan bakar. |
| SRS_08 | Sistem dapat memanajemen data TPS dan TPA seperti tambah, edit, dan hapus data. |
| SRS_09 | Sistem dapat memanajemen data sopir dan kendaraan seperti tambah, edit, dan hapus data. |
| SRS_10 | Sistem dapat menginput data parameter algoritma yan diinginkan. |

b. Kebutuhan Non-Fungsional

Kebutuhan non-fungsional merupakan fungsi layanan pada sistem yang tidak secara langsung terkait pada fungsi sistem. Sistem optimasi ini memiliki kebutuhan non-fungsional yang dapat dilihat pada Tabel 4.2

Tabel 4.2 Kebutuhan Non-Fungsional Sistem

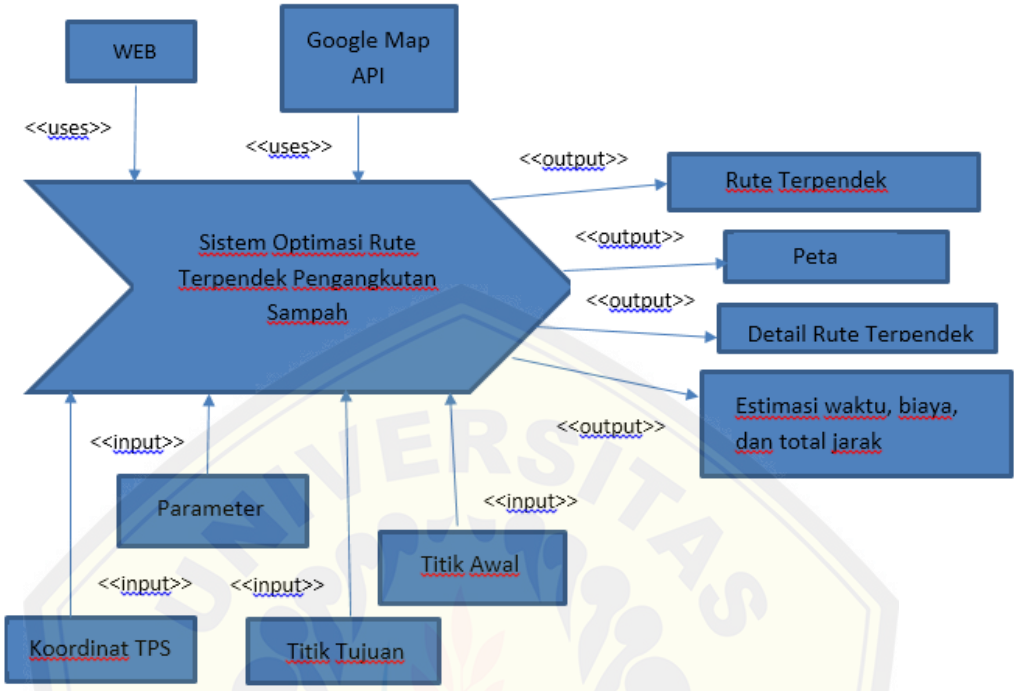
| SRS_ID | Identifikasi |
|---|---|
| SRS_01 | Sistem dapat diakses secara bersamaan oleh *user* yang berbeda |
| SRS_02 | Sistem dapat berjalan pada platform atau sistem operasi apa saja yang mendukung aplikasi berbasis web |
| SRS_03 | Sistem menggunakan enkripsi password pengguna untuk keamanan saat autentifikasi login |
| SRS_04 | Sistem optimasi berbasis web ini mudah digunakan (*user friendly*) |

4.3.2 Desain Sistem

Desain sistem yang digunakan adalah bahasa pemodelan *Unified Modeling Language* (UML). Dalam UML terdapat beberapa diagram yang harus digambarkan untuk mendukung pembuatan program. Diagram-diagram tersebut adalah sebagai berikut :

*a. Business Process*

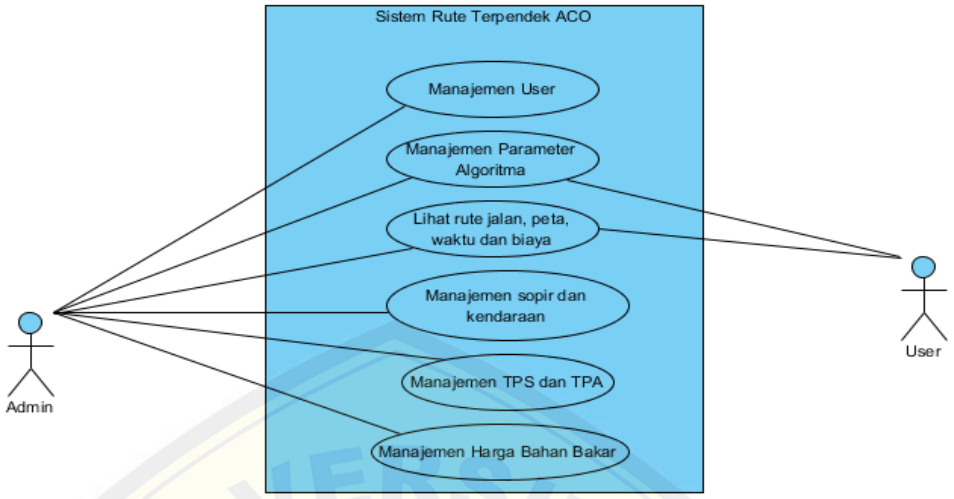
*Business process* merupakan sebuah model diagram yang menggambarkan proses keseluruhan pada sistem yang terdiri dari *trigger, uses, input, supply, output*, dan *goal*. Input pada sistem optimasi ini adalah data TPA dan TPS beserta titik koordinat nya. Informasi waktu dan biaya sebagai data pendukung atau *supply*. Keluaran dari sistem berupa peta rute jalan yang dibangun menggunakan Google Map API, dan petunjuk *path* jalan menuju TPA yang dituju. Tujuannya adalah untuk mengetahui rute terpendek menuju TPA. Sistem optimasi ini akan dibangun berbasis web. Gambaran *business process* sistem dapat dilihat pada Gambar 4.1




Gambar 4.1 *Business Process* Sistem Optimasi

*b. Usecase Diagram*

*Usecase diagram* merupakan identifikasi fungsi atau fitur yang ada pada sistem digambarkan berinteraksi dengan *user* sebagai akses fitur yang bisa digunakan oleh *user* tersebut. Fitur-fitur pada sistem ini terdapat 5 fitur yang digambarkan dengan elips dan terdapat 2 tipe *user*. Use case diagram pada sistem optimasi ini dapat digambarkan pada Gambar 4.2




Gambar 4.2 *Usecase Diagram* Sistem Rute Terpendek

1. Definisi Aktor

Definisi aktor menjelaskan tentang aktor yang berinteraksi dengan sistem dan menjelaskan hak akses fitur yang dapat digunakan oleh aktor tersebut. Penjelasan definisi aktor dapat dilihat pada Tabel 4.3

Tabel 4.3 Definisi Aktor *Usecase*

| **No.** | **Aktor** | **Deskripsi** |
|---|---|---|
| **1.** | Admin | Aktor yang memiliki hak akses penuh terhadap sistem. Aktor ini bisa menggunakan semua fitur yang ada pada sistem mulai mengelola data *user*, data sopir dan kendaraan, data TPS dan TPA, data harga bahan bakar,serta data parameter algoritma. Pengelolaan tersebut dapat berupa tambah, edit, dan hapus data. Admin juga dapat melihat fitur fungsionalitas sistem. |
| **2.** | *User* | Aktor yang memiliki akses terbatas pada sistem. Aktor ini hanya dapat mencari dan melihat informasi detail rute jalan, peta, serta informasi waktu dan biaya serta |

| | |
|---|---|
| | *input* parameter algoritma. |

2. Definisi *Usecase*

Definisi *usecase* merupakan penjelasan dari setiap *usecase* diagram yang berisi fitur-fitur dari sistem. Penjelasan definisi *usecase* dapat dilihat pada Tabel 4.4

Tabel 4.4 Definisi *Usecase*

| No. | *Usecase* | Deskripsi |
|---|---|---|
| **1.** | Manajemen *User* | *Usecase* yang menggambarkan proses pengelolaan data *user*. |
| **2.** | Lihat Info Jalan | *Usecase* yang menampilkan rute jalan yang telah didapat, peta rute jalan, arah jalan yang sesuai dengan peta serta informasi waktu dan biaya sebagai hasil dari proses ACO. |
| **3.** | Manajemen Parameter Algoritma | *Usecase* yang menampilkan input titik awal dan titik tujuan, serta kolom untuk input parameter algoritma yaitu alfa, beta, rho, *Asymptote Factor*, jumlah semut, dan jumlah jalan. Selain itu juga akan ditampilkan peta rute jalan, arah jalan yang sesuai dengan peta, dan estimasi waktu dan biaya sebagai hasil dari proses algoritma. |
| **4.** | Manajemen Sopir dan Kendaraan | *Usecase* yang menggambarkan proses pengelolaan data sopir dan kendaraan. |
| **5.** | Manajemen Harga Bahan Bakar | *Usecase* yang menggambarkan proses pengelolaan data harga bahan bakar sesuai dengan harga bahan bakar saat ini. |
| **6.** | Manajemen TPA | *Usecase* yang menggambarkan proses pengelolaan data TPS dan TPA. |

*c.* *Usecase* Skenario

*Usecase* Skenario merupakan penjelasan alur skenario dari sistem yang akan dibuat berdasarkan masing-masing fitur. Setiap *usecase* diagram pada Tabel 4.4 akan dibuat skenario baik itu skenario normal maupun skenario alternatif.

1. *Usecase* skenario manajemen *user*

*Usecase* skenario ini menjelaskan alur untuk memanajemen data *user* seperti melakukan tambah, edit, dan hapus data. Detail penjelasan urutan reaksi aktor dan reaksi sistem pada skenario normal dan skenario alternatif dapat dilihat pada Tabel 4.5.

Tabel 4.5 *Usecase* Skenario Manajemen *User*

| | |
|---|---|
| Nama Usecase : Manajemen *User*<br>ID usecase : 01<br>Aktor : Admin<br>Deskripsi : Admin dapat menambah, mengedit, dan menghapus data *user* | |
| ***Normal flow* "Tambah Data *User*"** | |
| ***User*** | **Sistem** |
| 1. Memilih menu *user* | |
| | 2. Menampilkan halaman data *user* berupa data *user*. Pada setiap baris data *user* terdapat dua pilihan yaitu edit dan hapus serta pilihan tambah diatas tabel yang digunakan untuk menambahkan data *user* baru. |
| 3. Klik tambah | |
| | 4. Menampilkan *form* data *user* |
| 5. Mengisi *form* data *user* | |
| 6. Klik submit | |
| | 7. Memeriksa kelengkapan data *user* |

| | |
|---|---|
| | 8. Menyimpan data *user* baru |
| | 9. Mengembalikan pada halaman data *user* |
| ***Normal flow* "Edit Data *User*"** | |
| 10. Klik edit pada baris data *user* yang ingin di edit | |
| | 11. Menampilkan data *user* yang akan di edit pada form data *user* |
| 12. Mengedit data *user* pada form data *user* | |
| 13. Klik submit | |
| | 14. Memeriksa kelengkapan data |
| | 15. Menyimpan data *user* |
| | 16. Mengembalikan ke halaman data *user* |
| ***Normal flow* "Hapus Data *User*"** | |
| 17. Klik hapus pada baris data *user* yang ingin dihapus | |
| | 18. Menampilkan *alert* "anda yakin ingin menghapus data ini? " |
| 19. Klik "Ok" | |
| | 20. Menghapus data *user* |
| | 21. Mengembalikan ke halaman data *user* |
| ***Alternative flow* "Kelengkapan Data"** | |
| 22. Apabila *user* memasukkan data tidak lengkap | |
| 23. Klik submit | |

| | |
|---|---|
| | 24. Menampilkan *alert* "*Please fill this field*" |
| ***Alternative flow* "Batal Menghapus Data *User*"** | |
| 25. Apabila *user* batal menghapus data *user* | |
| 26. Klik *Cancel* | |
| | 27. Mengembalikan ke halaman data *user* |

2. *Usecase* skenario lihat info jalan

*Usecase* skenario ini menjelaskan alur untuk melihat peta rute jalan TPS ke TPA, melihat detail rute jalan, serta melihat informasi waktu dan biaya. Detail penjelasan urutan reaksi aktor dan reaksi sistem pada skenario normal dapat dilihat pada Tabel 4.6.

Tabel 4.6 *Usecase* Skenario Lihat Info Jalan

| | |
|---|---|
| Nama usecase : Lihat Info Jalan<br>ID usecase : 02<br>Aktor : Admin dan *User*<br>Deskripsi : Admin dan *user* dapat menginput titik awal dan titik tujuan, melihat rute jalan, peta, serta info waktu dan biaya | |
| ***Normal flow* "Lihat Info Jalan"** | |
| ***User*** | **Sistem** |
| 1. Klik menu info jalan | |
| | 2. Menampilkan halaman info jalan |
| 3. Memilih titik awal | |
| | 4. Menonaktifkan titik awal yang telah dipilih agar tidak dipilih kembali pada titik tujuan |

| | |
|---|---|
| 5. Memilih titik tujuan | |
| | 6. Menampilkan hasil proses pencarian rute terpendek, hasil waktu dan biaya, serta peta rute jalan |
| ***Alternative Flow* "Kelengkapan Data"** | |
| 7. Apabila user memasukkan data tidak lengkap | |
| 8. Klik enter | |
| | 9. Menampilkan *alert* "*Please fill this field*" |

3. *Usecase* skenario manajemen parameter algoritma

*Usecase* skenario ini menjelaskan alur untuk memanajemen data parameter algoritma, seperti input parameter dasar dari algoritma ACO. Detail penjelasan urutan reaksi aktor dan reaksi sistem pada skenario normal dan skenario alternatif dapat dilihat pada Tabel 4.7.

Tabel 4.7 *Usecase* skenario lihat manajemen parameter algoritma

| | |
|---|---|
| Nama usecase : Lihat Manajemen Parameter Algoritma<br>ID usecase : 03<br>Aktor : Admin dan *User*<br>Deskripsi : Admin dan *user* dapat menginput titik awal dan titik tujuan, input parameter dasar dari algoritma, melihat rute jalan, peta, serta info waktu dan biaya | |
| ***Normal flow* "lihat manajemen parameter algoritma"** | |
| ***User*** | **Sistem** |
| 1. Klik menu Algoritma | |
| | 2. Menampilkan halaman algoritma |
| 3. Input parameter | |

| | |
|---|---|
| 4. Memilih titik awal | |
| | 5. Menonaktifkan titik awal yang telah dipilih agar tidak dipilih kembali pada titik tujuan |
| 6. Memilih titik tujuan | |
| | 7. Menampilkan hasil proses pencarian rute terpendek, hasil waktu dan biaya, serta peta rute jalan |
| ***Alternative flow* "kelengkapan data"** | |
| 8. Apabila *user* memasukkan data tidak lengkap | |
| 9. Klik enter | |
| | 10. Menampilkan *alert* "*please fill this field*" |

4. *Usecase skenario* manajemen sopir

*Usecase skenario* ini menjelaskan alur untuk memanajemen data sopir seperti melakukan tambah, edit, dan hapus data. Detail penjelasan urutan reaksi aktor dan reaksi sistem pada skenario normal dan skenario alternatif dapat dilihat pada Tabel 4.8.

Tabel 4.8 *Usecase* skenario Manajemen Data Sopir dan Kendaraan

| | |
|---|---|
| Nama Usecase : Manajemen Data Sopir<br>ID usecase : 04<br>Aktor : Admin<br>Deskripsi : Admin dapat menambah, mengedit, dan menghapus data sopir dan kendaraan | |
| ***Normal flow* "Tambah Data Sopir dan kendaraan"** | |
| ***User*** | **Sistem** |

| | |
|---|---|
| 1. Memilih menu sopir | |
| | 2. Menampilkan halaman data admin berupa data sopir dan kendaraan. Pada setiap baris data sopir terdapat dua pilihan yaitu edit dan hapus serta pilihan tambah diatas tabel yang digunakan untuk menambahkan data sopir dan kendaraan baru. |
| 3. Klik tambah | |
| | 4. Menampilkan form data sopir dan kendaraan |
| 5. Mengisi form data sopir dan kendaraan | |
| 6. Klik submit | |
| | 7. Memeriksa kelengkapan data sopir dan kendaraan |
| | 8. Menyimpan data sopir baru dan kendaraan |
| | 9. Mengembalikan pada halaman data sopir |
| ***Normal flow* "Edit Data Sopir dan Kendaraan"** | |
| 10. Klik edit pada baris data sopir dan kendaraan yang ingin di edit | |
| | 11. Menampilkan data sopir dan kendaraan yang akan di edit pada form data sopir |
| 12. Mengedit data sopir pada form data sopir dan kendaraan | |

| | |
|---|---|
| 13. Klik submit | |
| | 14. Memeriksa kelengkapan data |
| | 15. Menyimpan data sopir |
| | 16. Mengembalikan ke halaman data sopir |
| ***Normal flow* "Hapus Data Sopir dan Kendaraan"** | |
| 17. Klik hapus pada baris data sopir yang ingin dihapus | |
| | 18. Menampilkan *alert* "anda yakin ingin menghapus data ini? " |
| 19. Klik "Ok" | |
| | 20. Menghapus data sopir |
| | 21. Mengembalikan ke halaman data sopir |
| ***Alternative flow* "Kelengkapan Data"** | |
| 22. Apabila *user* memasukkan data tidak lengkap | |
| 23. Klik submit | |
| | 24. Menampilkan *alert* "*Please fill this field*" |
| ***Alternative flow* "Batal Menghapus Data Sopir dan Kendaraan"** | |
| 25. Apabila *user* batal menghapus data sopir | |
| 26. Klik *Cancel* | |
| | 27. Mengembalikan ke halaman data sopir |

5. *Usecase* skenario manajemen TPS dan TPA

*Usecase* skenario ini menjelaskan alur untuk memanajemen data TPS dan TPA seperti melakukan tambah, edit, dan hapus data. Detail penjelasan urutan reaksi aktor dan reaksi sistem pada skenario normal dan skenario alternatif dapat dilihat pada Tabel 4.9.

Tabel 4.9 *Usecase* skenario Manajemen Data TPS dan TPA

| | |
|---|---|
| Nama Usecase : Manajemen Data TPS dan TPA<br>ID usecase : 05<br>Aktor : Admin<br>Deskripsi : Admin dapat menambah, mengedit, dan menghapus data TPS dan TPA | |
| ***Normal flow* "Tambah Data TPS"** | |
| ***User*** | **Sistem** |
| 1. Memilih menu TPS dan TPA | |
| | 2. Menampilkan halaman data admin berupa data TPS dan TPA. Pada setiap baris data *user* terdapat dua pilihan yaitu edit dan hapus serta pilihan tambah diatas tabel yang digunakan untuk menambahkan data TPS dan TPA baru. |
| 3. Klik tambah | |
| | 4. Menampilkan form data TPS dan TPA |
| 5. Mengisi form data TPS dan TPA | |
| 6. Klik submit | |
| | 7. Memeriksa kelengkapan data TPS dan TPA |

| | |
|---|---|
| | 8. Menyimpan data TPS dan TPA baru |
| | 9. Mengembalikan pada halaman data TPS dan TPA |
| ***Normal flow* "Edit Data TPS dan TPA"** | |
| 10. Klik edit pada baris data TPS dan TPA yang ingin di edit | |
| | 11. Menampilkan data TPS dan TPA yang akan di edit pada form data TPS dan TPA |
| 12. Mengedit data TPS dan TPA pada form data TPS dan TPA | |
| 13. Klik submit | |
| | 14. Memeriksa kelengkapan data |
| | 15. Menyimpan data TPS dan TPA |
| | 16. Mengembalikan ke halaman data TPS dan TPA |
| ***Normal flow* "Hapus Data TPS dan TPA"** | |
| 17. Klik hapus pada baris data TPS dan TPA yang ingin dihapus | |
| | 18. Menampilkan *alert* "anda yakin ingin menghapus data ini? " |
| 19. Klik "Ok" | |
| | 20. Menghapus data TPS dan TPA |
| | 21. Mengembalikan ke halaman data TPS dan TPA |
| ***Alternative flow* "Kelengkapan Data"** | |
| 22. Apabila *user* memasukkan data tidak | |

| | |
|---|---|
| lengkap | |
| 23. Klik submit | |
| | 24. Menampilkan *alert* "Please fill this field" |
| ***Alternative flow* "Batal Menghapus Data TPS dan TPA"** | |
| 25. Apabila *user* batal menghapus data TPS dan TPA | |
| 26. Klik *Cancel* | |
| | 27. Mengembalikan ke halaman data TPS dan TPA |

6. *Usecase* skenario manejemen harga bahan bakar

*Usecase* skenario ini menjelaskan alur untuk memanajemen data harga bahan bakar seperti melakukan tambah, edit, dan hapus data. Detail penjelasan urutan reaksi aktor dan reaksi sistem pada skenario normal dan skenario alternatif dapat dilihat pada Tabel 4.10.

Tabel 4.10 *Usecase* skenario manajemen harga bahan bakar

| | |
|---|---|
| Nama Usecase : Manajemen Data Harga Bahan Bakar<br>ID usecase : 06<br>Aktor : Admin<br>Deskripsi : Admin dapat menambah, mengedit, dan menghapus data harga bahan bakar | |
| ***Normal flow* "Tambah Data Harga Bahan Bakar"** | |
| ***User*** | **Sistem** |
| 1. Memilih menu Harga Bahan Bakar | |
| | 2. Menampilkan halaman data admin berupa data harga bahan bakar. Pada setiap baris data harga bahan |

| | |
|---|---|
| | bakar terdapat dua pilihan yaitu edit dan hapus serta pilihan tambah diatas tabel yang digunakan untuk menambahkan data harga bahan bakar baru. |
| 3. Klik tambah | |
| | 4. Menampilkan form data harga bahan bakar |
| 5. Mengisi form data harga bahan bakar | |
| 6. Klik submit | |
| | 7. Memeriksa kelengkapan data harga bahan bakar |
| | 8. Menyimpan data harga bahan bakar baru |
| | 9. Mengembalikan pada halaman data harga bahan bakar |
| ***Normal flow* "Edit Data Harga Bahan Bakar"** | |
| 10. Klik edit pada baris data harga bahan bakar yang ingin di edit | |
| | 11. Menampilkan data harga bahan bakar yang akan di edit pada form data kendaraan |
| 12. Mengedit data harga bahan bakar pada form data harga bahan bakar | |
| 13. Klik submit | |
| | 14. Memeriksa kelengkapan data |
| | 15. Menyimpan data harga bahan bakar |

| | |
|---|---|
| | 16. Mengembalikan ke halaman data harga bahan bakar |
| ***Normal flow* "Hapus Data Harga Bahan Bakar"** | |
| 17. Klik hapus pada baris data harga bahan bakar yang ingin dihapus | |
| | 18. Menampilkan *alert* "anda yakin ingin menghapus data ini? " |
| 19. Klik "Ok" | |
| | 20. Menghapus data harga bahan bakar |
| | 21. Mengembalikan ke halaman data harga bahan bakar |
| ***Alternative flow* "Kelengkapan Data"** | |
| 22. Apabila *user* memasukkan data tidak lengkap | |
| 23. Klik submit | |
| | 24. Menampilkan *alert* "*Please fill this field*" |
| ***Alternative flow* "Batal Menghapus Data Harga Bahan Bakar"** | |
| 25. Apabila *user* batal menghapus data harga bahan bakar | |
| 26. Klik *Cancel* | |
| | 27. Mengembalikan ke halaman data harga bahan bakar |

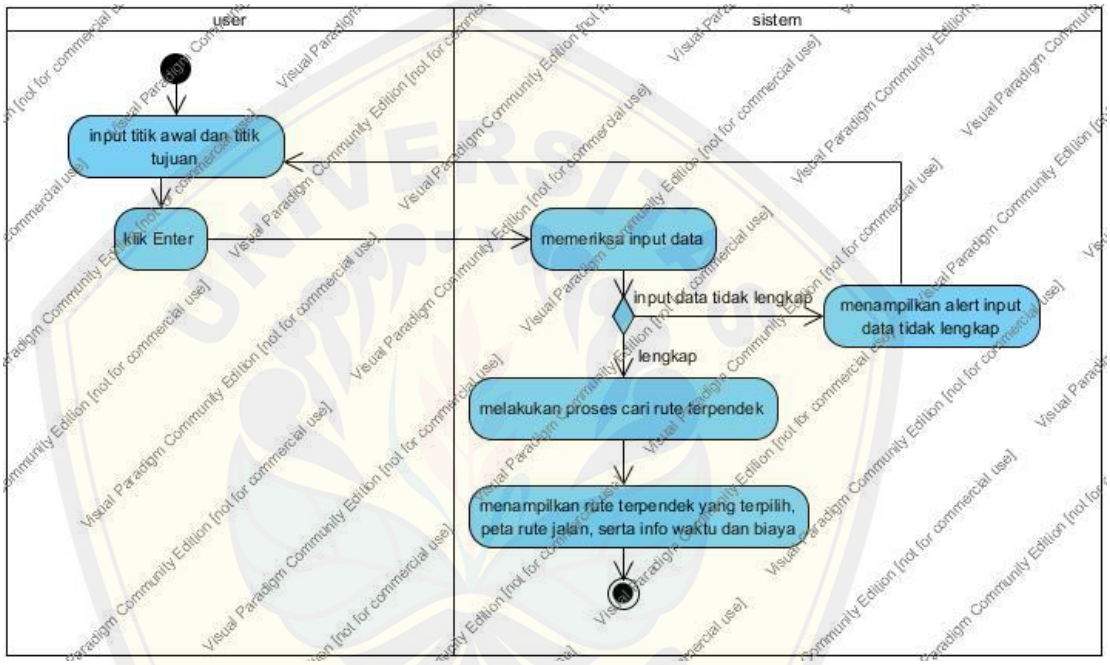
*d. Activity Diagram*

*Activity diagram* merupakan penggambaran alur kerja aktifitas pada setiap fungsi fitur yang ada pada sistem. Pada sistem optimasi ini terdapat 6 *usecase* yang akan

digambarkan ke dalam *activity diagram*. Masing-masing diagram akan digambarkan bagaimana suatu *activity* dimulai kemudian terdapat *decision* dan bagaimana *activity* tersebut berakhir.

1. *Activity Diagram* Lihat Info Jalan

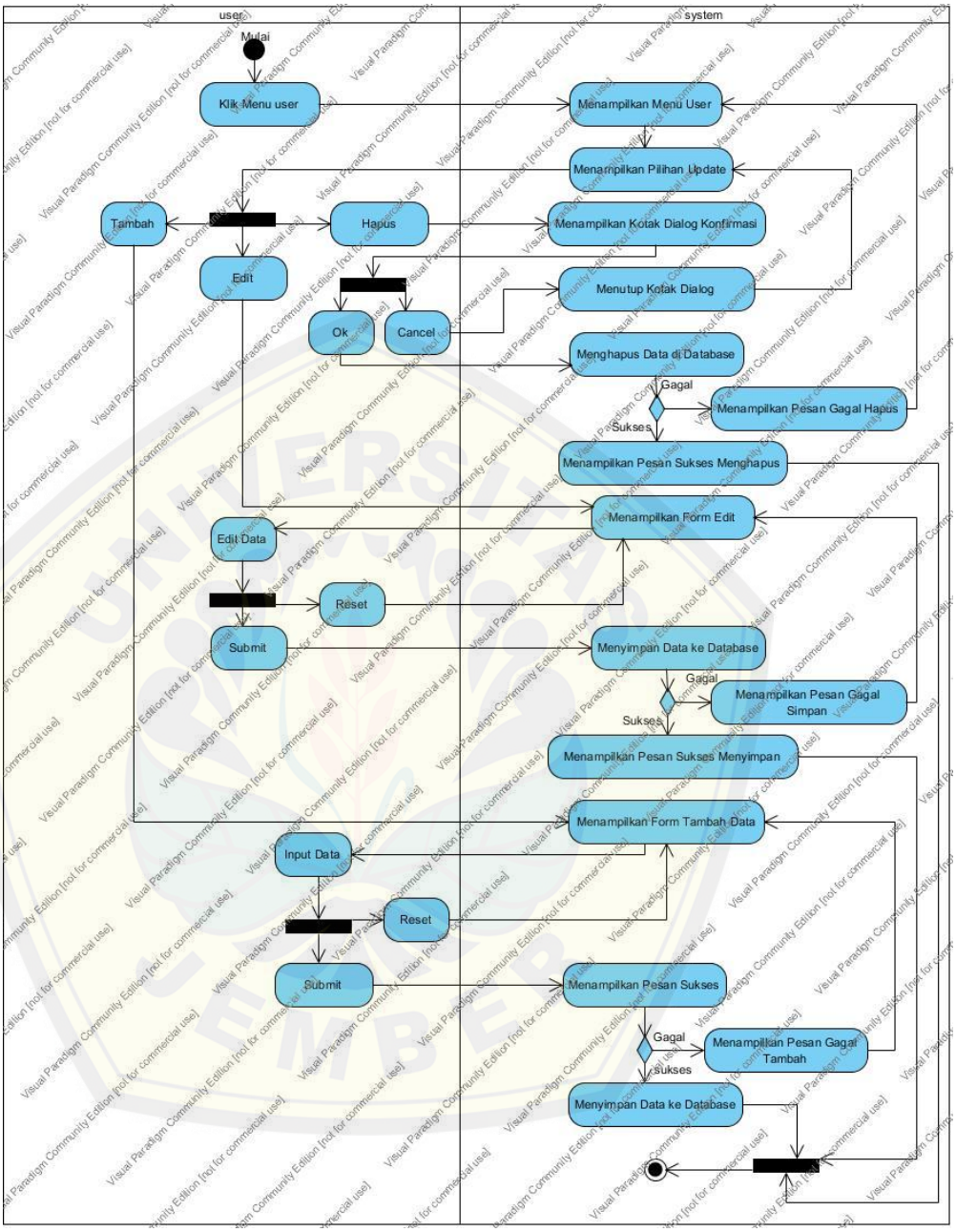
*Activity diagram* ini menjelaskan alur kerja aktifitas pada sistem dari proses Lihat Info Jalan. Alur kerja aktifitas lihat info jalan dapat dilihat pada Gambar 4.3.




Gambar 4.3 *Activity diagram* Lihat Info Jalan

2. *Activity Diagram* Manajemen *User*

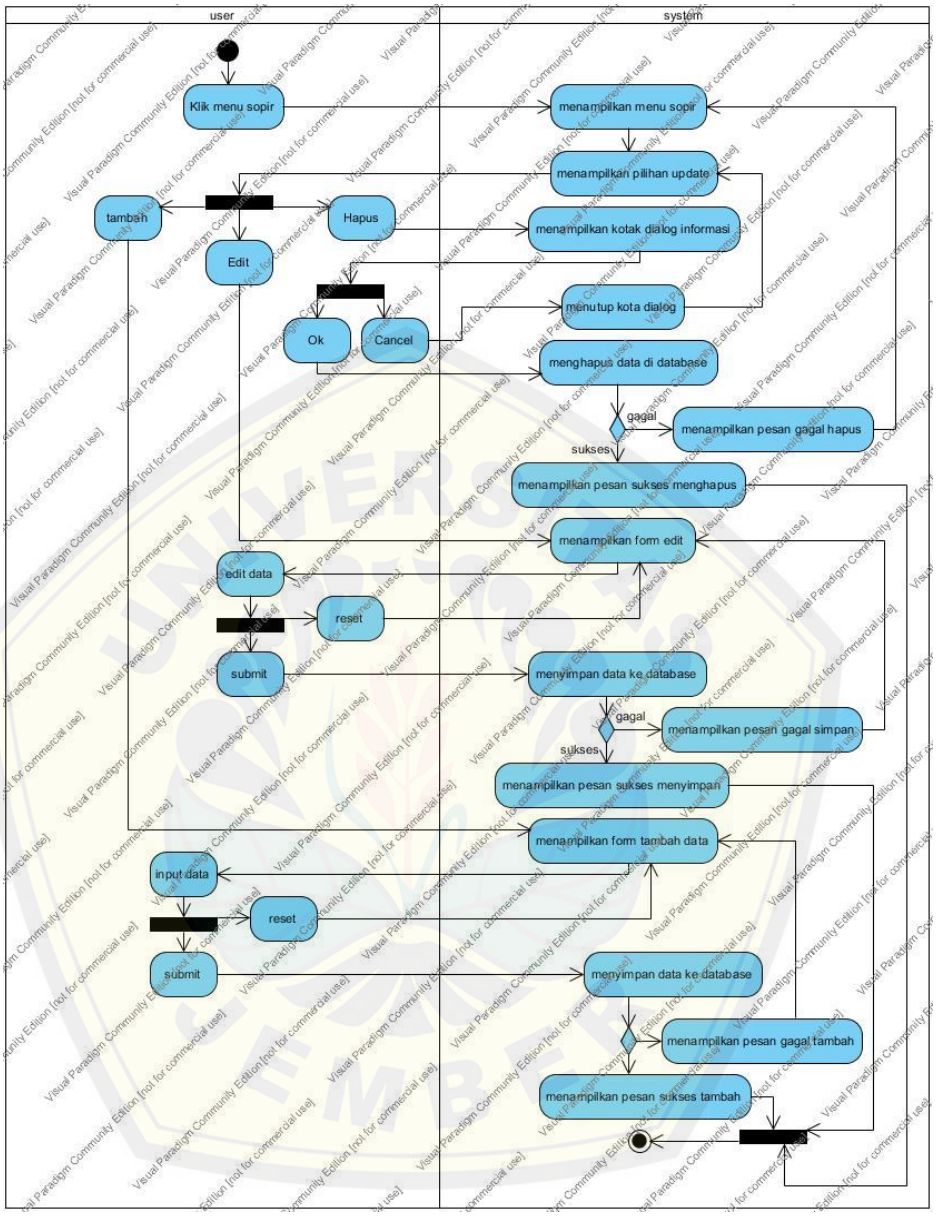
*Activity diagram* ini menjelaskan alur kerja aktifitas pada sistem dari proses manajemen *user*. Alur kerja aktifitas manajemen data *user* dapat dilihat pada Gambar 4.4.




Gambar 4.4 *Activity diagram* Manajemen *User*

3. *Activity Diagram* Manajemen Sopir

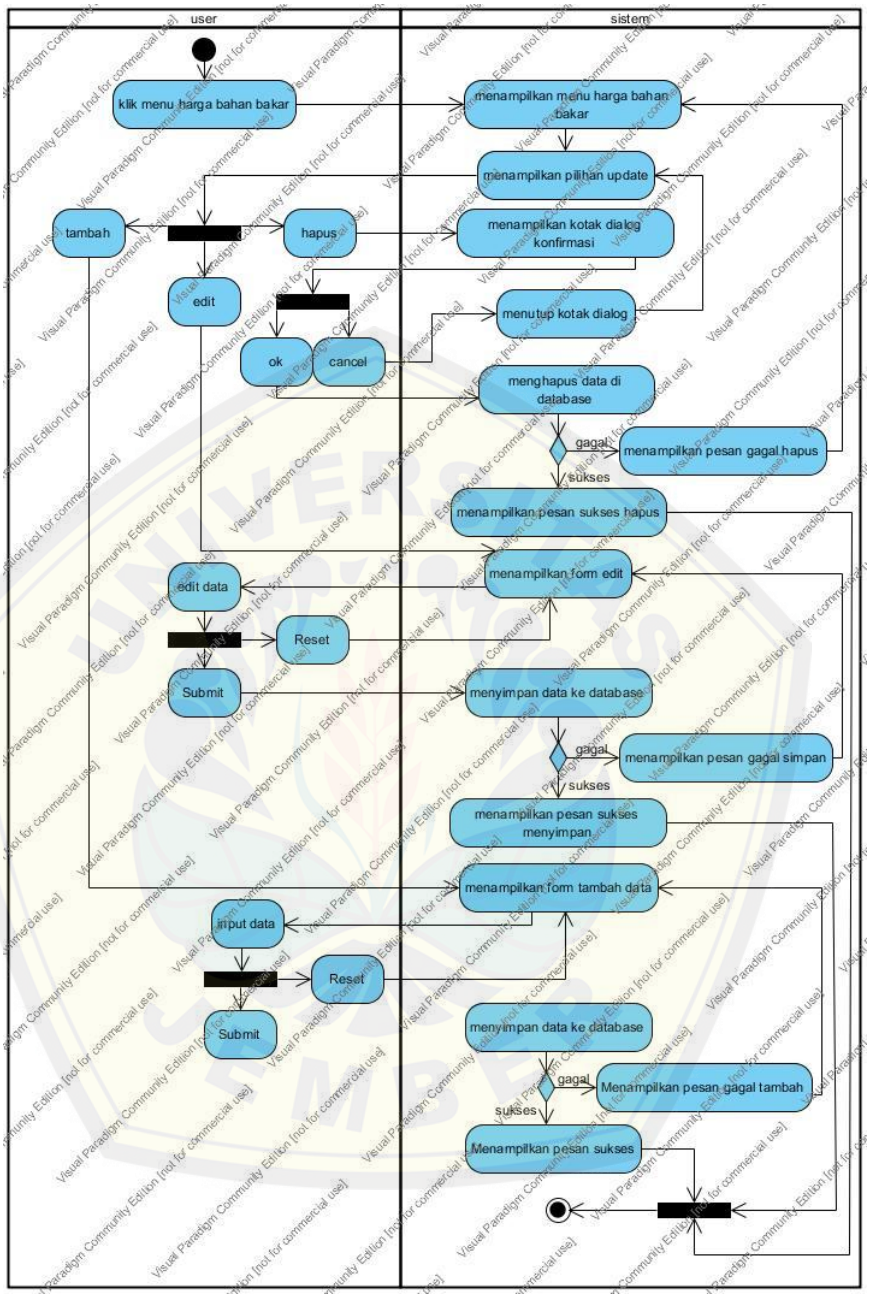
*Activity diagram* ini menjelaskan alur kerja aktifitas pada sistem dari proses manajemen data sopir. Alur kerja aktifitas manajemen sopir dapat dilihat pada Gambar 4.5.




Gambar 4.5 *Activity diagram* manajemen data sopir

4. *Activity Diagram* Manajemen Harga Bahan Bakar

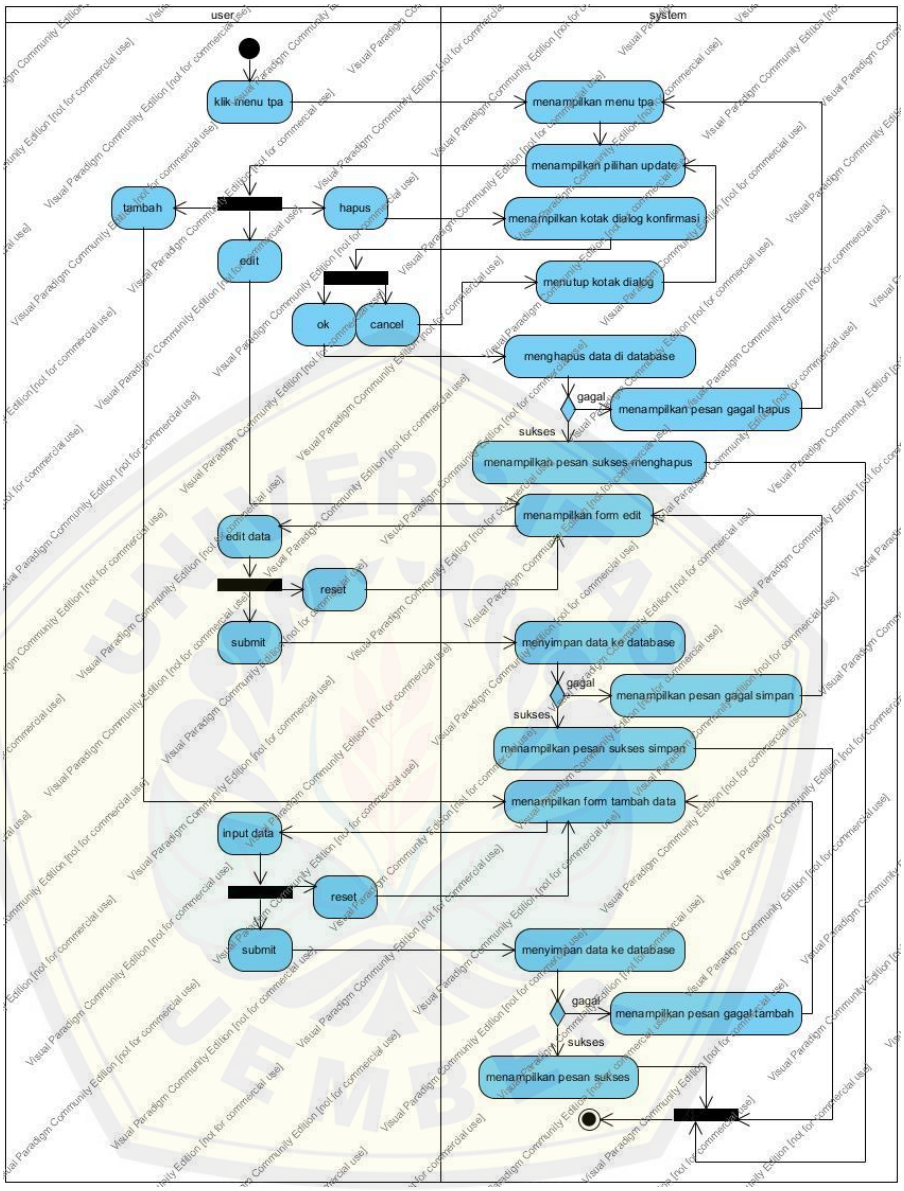
*Activity diagram* ini menjelaskan alur kerja aktifitas pada sistem dari proses manajemen data harga bahan bakar. Alur kerja aktifitas manajemen harga bahan bakar dapat dilihat pada Gambar 4.6.




Gambar 4.6 *Activity diagram* manajemen data harga bahan bakar

5. *Activity Diagram* Manajemen TPA dan TPS

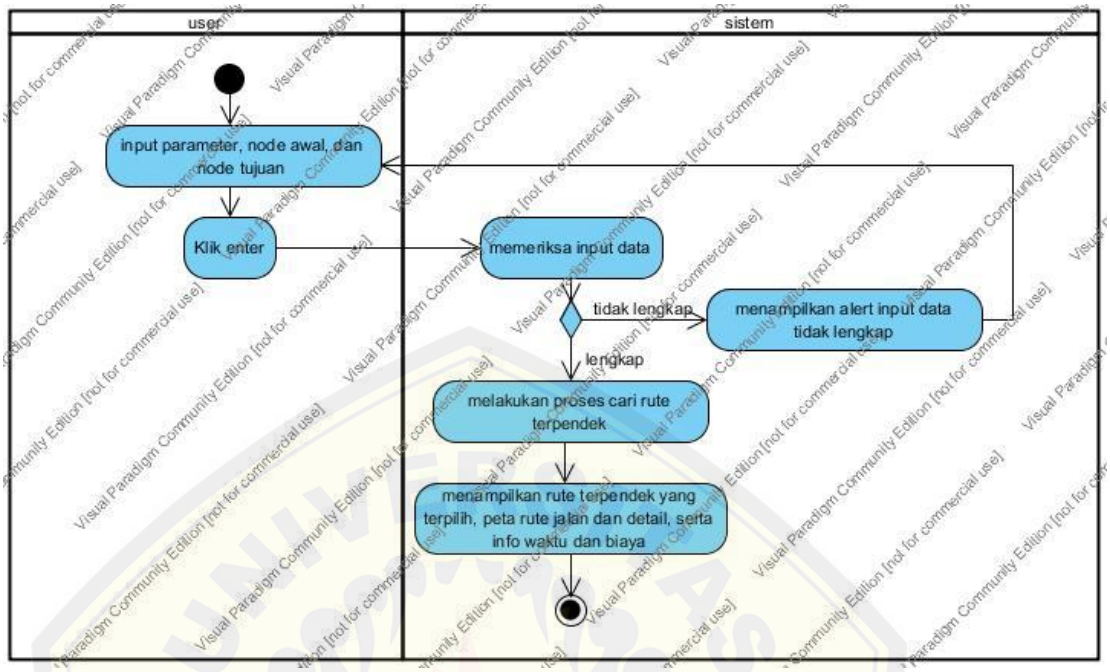
*Activity diagram* ini menjelaskan alur kerja aktifitas pada sistem dari proses manajemen data TPS dan TPA. Alur kerja aktifitas manajemen TPS dan TPA dapat dilihat pada Gambar 4.7.




Gambar 4.7 *Activity Diagram* Manajemen TPA dan TPS

6. *Activity Diagram* Manajemen Parameter Algoritma

*Activity diagram* ini menjelaskan alur kerja aktifitas pada sistem dari proses manajemen parameter algoritma ACO. Alur kerja aktifitas manajemen parameter algoritma ACO dapat dilihat pada Gambar 4.8.



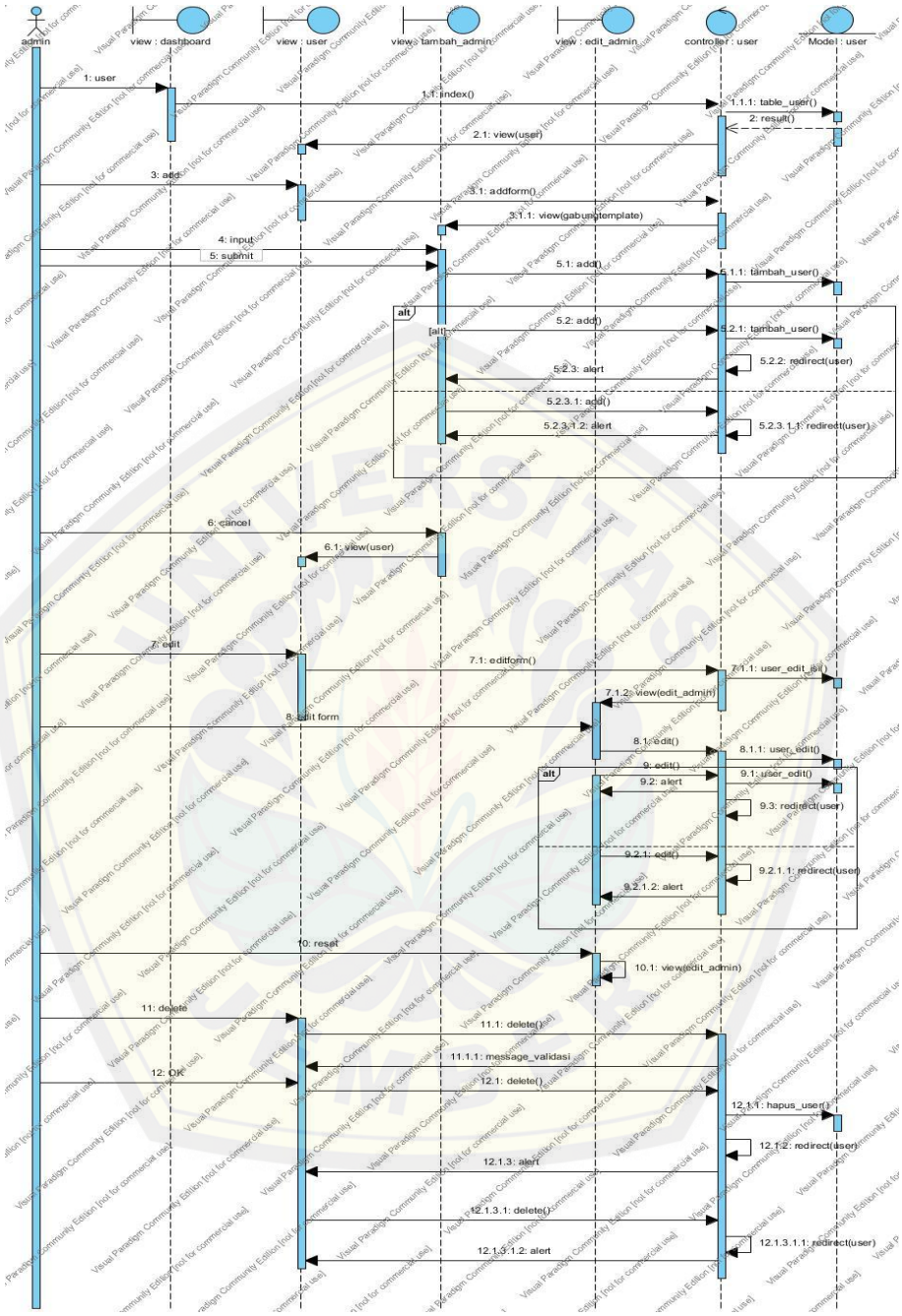

Gambar 4.8 *Activity diagram* Parameter Algoritma ACO

e. *Sequence Diagram*

*Sequence diagram* menggambarkan skenario dan memodelkan aliran logika sistem. Pada subbab ini, *sequence diagram* menggambarkan skenario dan memodelkan aliran logika pada fitur yang dibangun.

1. *Sequence User*

*Sequence* diagram data *user* merupakan penggambaran skenario dan pemodelan aliran logika sistem pada fitur data *user*. Penggambaran *sequence* diagram data *user* dapat dilihat pada Gambar 4.9.

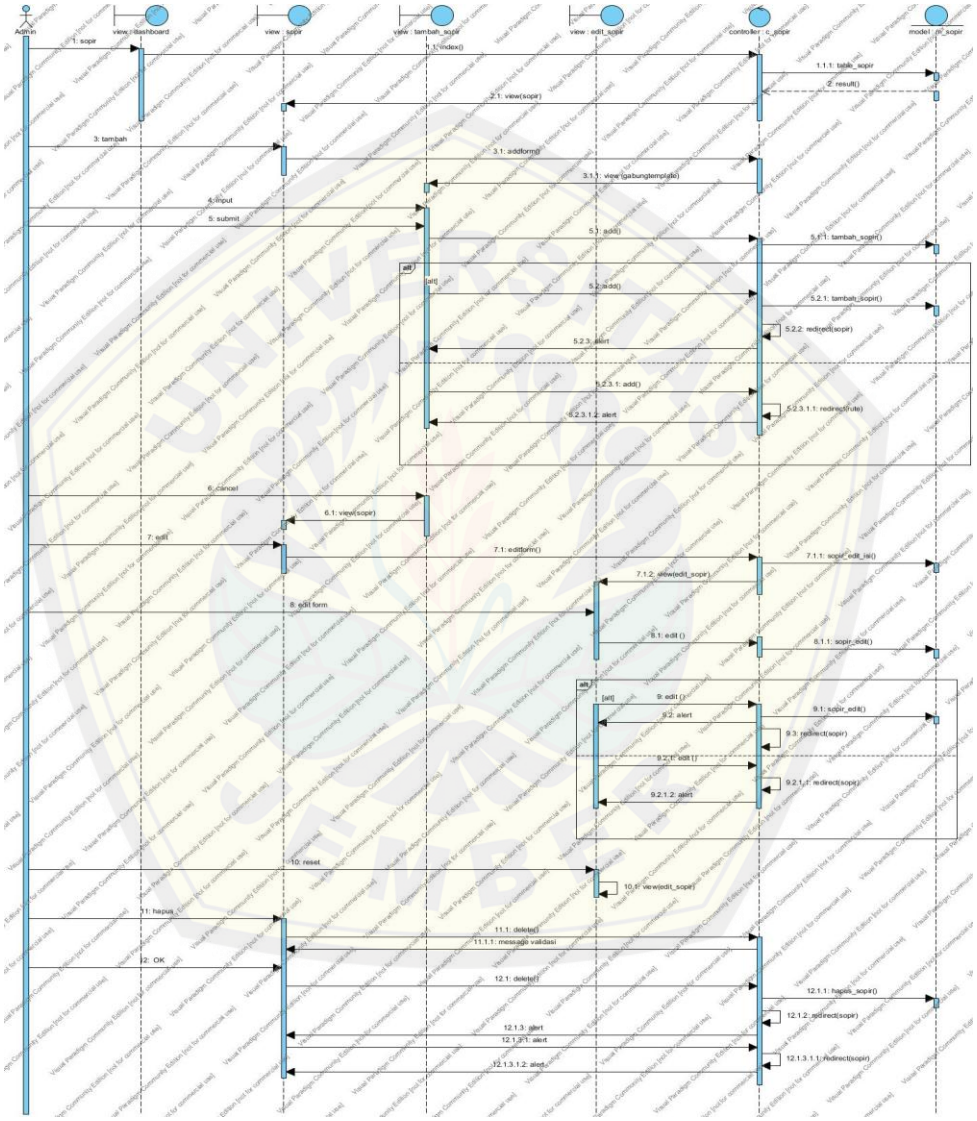



Gambar 4.9 *Sequence Diagram* manajemen *user*

Gambar 4.9 menggambarkan skenario dan memodelkan alian logika pada fitur data *user*. Terdapat empat *view*, satu *controller*, dan satu model yang digunakan pada fitur data *user*.

2. Sequence Sopir

Sequence diagram sopir merupakan penggambaran skenario dan pemodelan aliran logika sistem pada fitur sopir. Penggambaran *sequence* diagram sopir dapat dilihat pada Gambar 4.10.

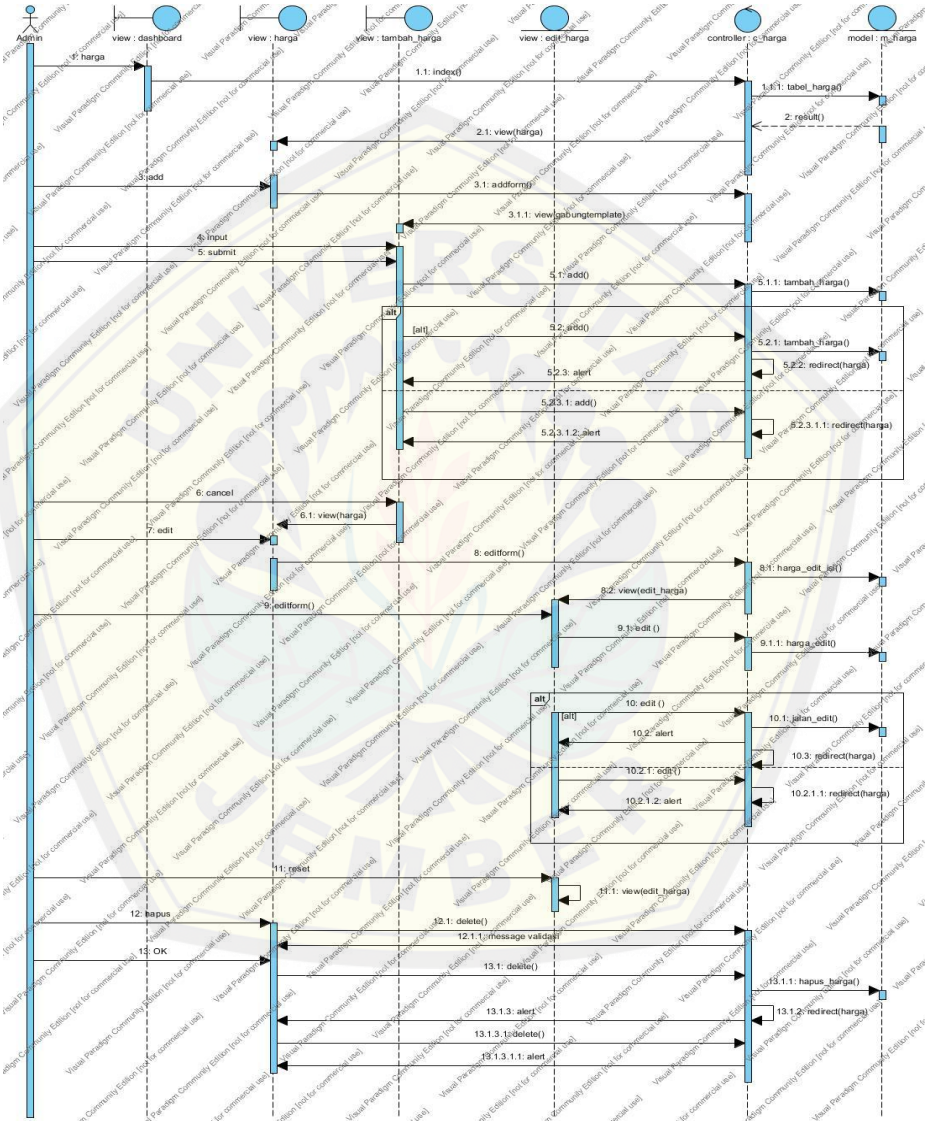



Gambar 4.10 *Sequence Diagram* manajemen data sopir

Gambar 4.10 menggambarkan skenario dan memodelkan aliran logika pada fitur data sopir. Terdapat empat *view*, satu *controller* dan satu model yang digunakan pada fitur data sopir.

3. *Sequence* harga bahan bakar

*Sequence* diagram harga bahan bakar merupakan penggambaran skenario dan pemodelan aliran logika sistem pada fitur kendaraan. Penggambaran *sequence* diagram harga bahan bakar dapat dilihat pada Gambar 4.11.

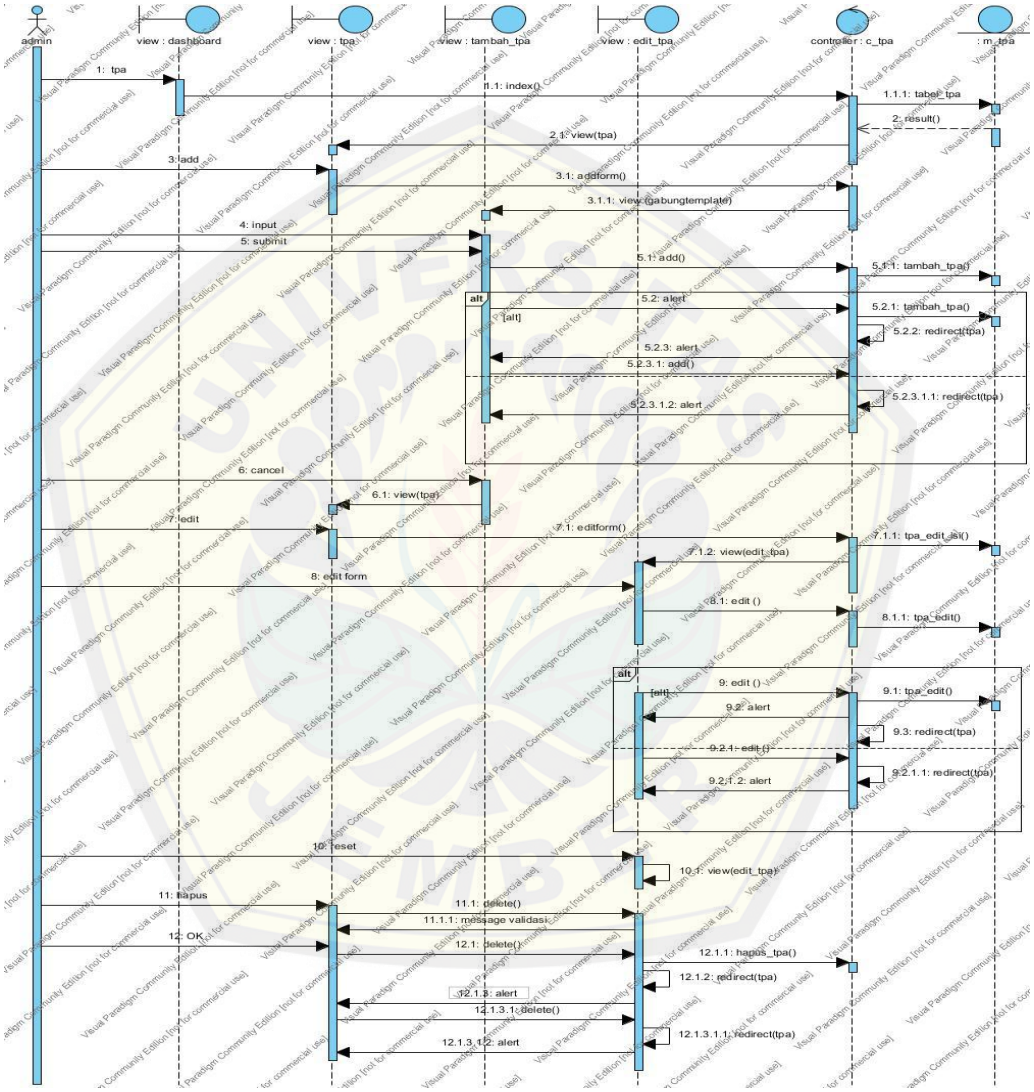



Gambar 4.11 *Sequence Diagram* manajemen data harga bahan bakar

Gambar 4.11 menggambarkan skenario dan memodelkan aliran logika pada fitur data harga bahan bakar. Terdapat empat *view*, satu *controller*, dan satu model yang digunakan pada fitur data harga bahan bakar.

4. *Sequence* TPA

*Sequence* diagram TPA merupakan penggambaran skenario dan pemodelan aliran logika sistem pada fitur TPA. Penggambaran *sequence* diagram TPA dapat dilihat pada Gambar 4.12.

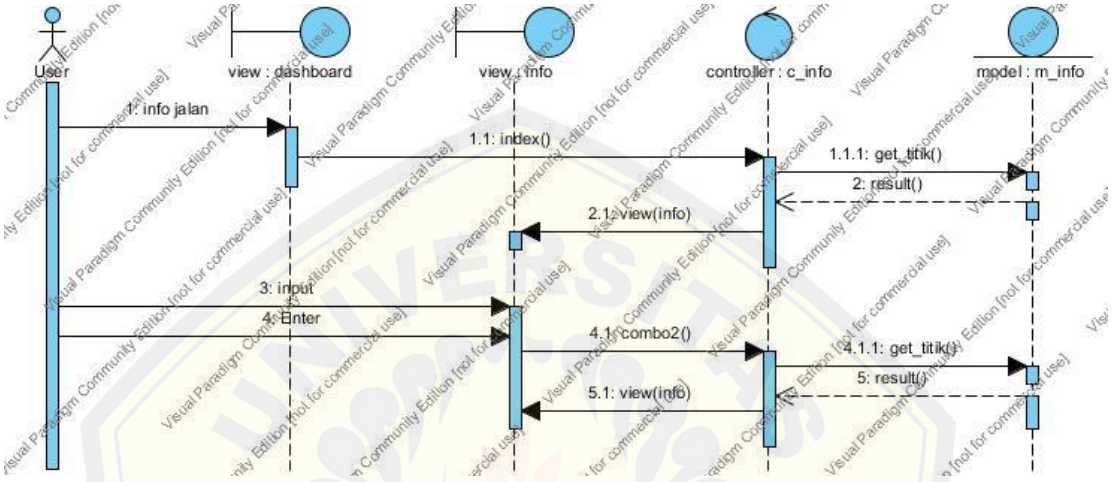



Gambar 4.12 *Sequence Diagram* manajemen data TPA dan TPS

Gambar menggambarkan skenario dan memodelkan aliran logika pada fitur data TPA. Terdapat empat *view*, satu *controller*, dan satu model yang digunakan pada fitur data TPA.

5. Sequence Info Jalan

Sequence diagram info jalan merupakan penggambaran skenario dan pemodelan aliran logika sistem pada fitur info jalan. Penggambaran sequence diagram info jalan dapat dilihat pada Gambar 4.13.

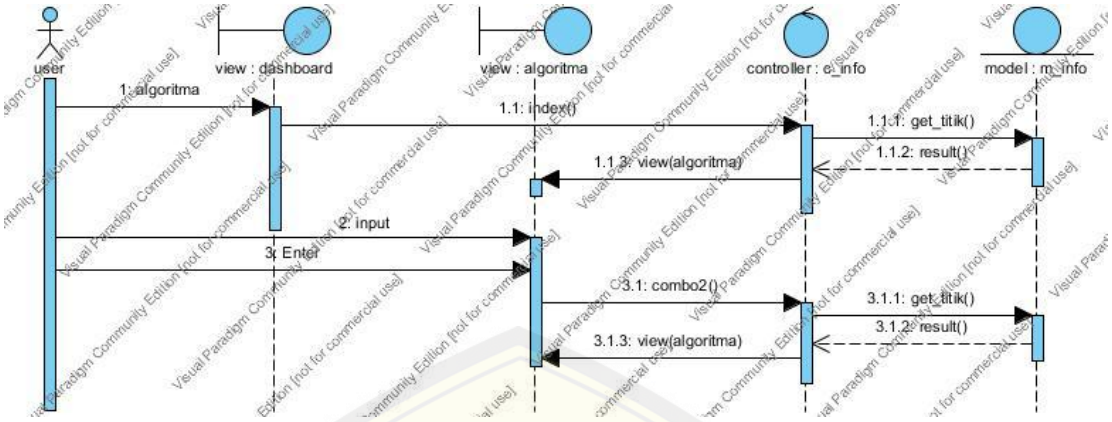



Gambar 4.13 *Sequence diagram* Lihat Info Jalan

Gambar 4.13 menggambarkan skenario dan memodelkan aliran logika pada fitur data info jalan. Terdapat empat *view*, satu *controller*, dan satu model yang digunakan pada fitur data info jalan.

6. *Sequence* Parameter Algoritma

*Sequence* diagram parameter algoritma merupakan penggambaran skenario dan pemodelan aliran logika sistem pada fitur parameter algoritma. Penggambaran *sequence* diagram parameter algoritma dapat dilihat pada Gambar 4.14.

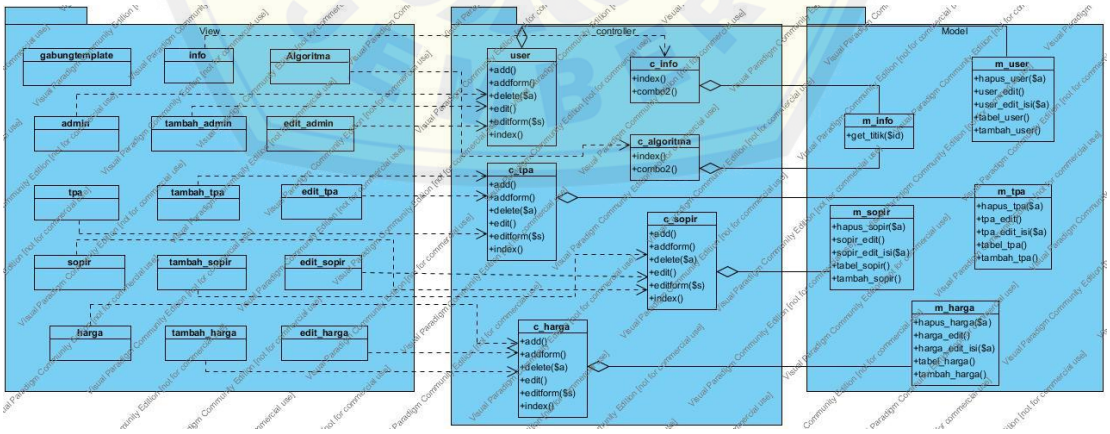



Gambar 4.14 *Sequence Diagram* Parameter Algoritma

Gambar 4.14 menggambarkan skenario dan memodelkan aliran logika pada fitur data parameter algoritma. Terdapat dua *view*, satu *controller*, dan satu model yang digunakan pada fitur data parameter algoritma.

f. *Class Diagram*

*Class diagram* menggambarkan kelas-kelas pada sistem yang dibangun dan hubungan antara kelas satu dan lainnya serta berisi atribut dan method apa saja yang ada didalamnya. Pada subbab ini, class diagram menggambarkan kelas-kelas yang digunakan pada fitur yang dibangun.

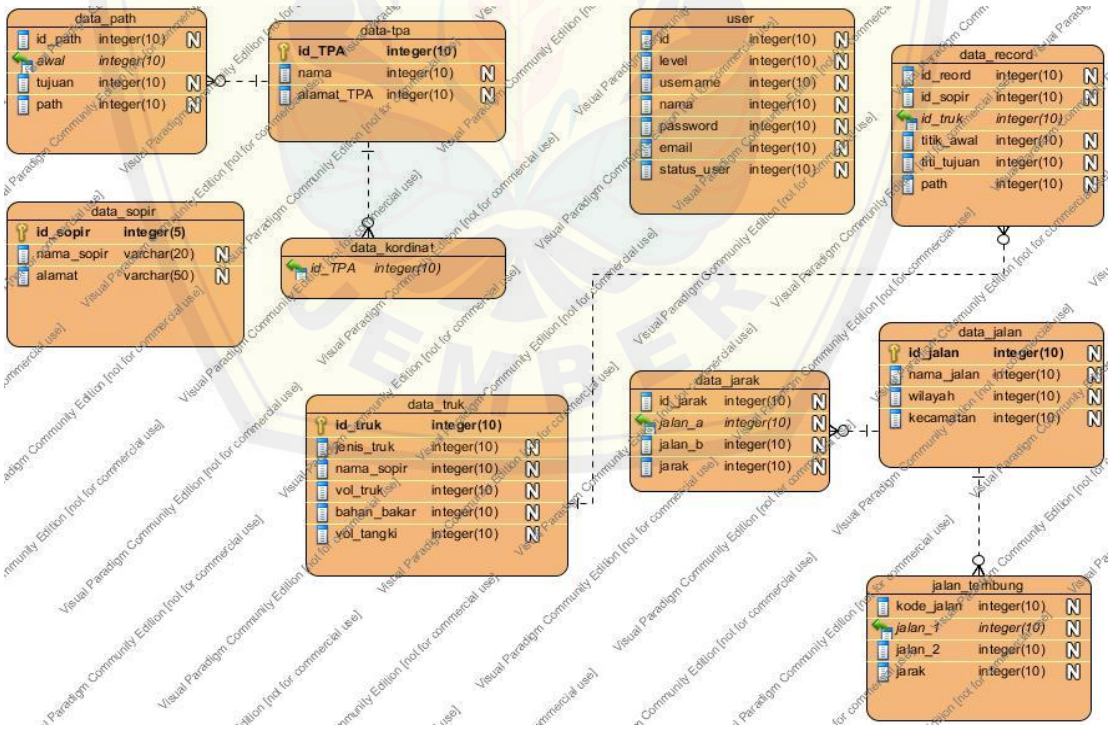



Gambar 4.15 *Class Diagram* Sistem
Sumber : (Hasil Analisis, 2015)

Gambar 4.15 merupakan *class* diagram dari sistem. Terdapat lima belas *view*, enam *controller*, dan lima model. Tiga *view* memiliki hubungan *dependency* dengan *controller* c_harga, tiga *view* memiliki hubungan *dependancy* dengan *controller user*, tiga *view* memiliki hubungan *dependancy* dengan *controller* c_tpa, tiga *view* memiliki hubungan *dependancy* dengan *controller* c_sopir, satu *view* memiliki hubungan *dependancy* dengan *controller* c_info dan satu *view* memiliki hubungan *dependency* dengan *controller* c_algoritma, sedangkan masing-masing model memiliki hubungan *aggregat* terhadap masing-masing *controller*.

g. *Entity relationship diagram (ERD)*

*Entity relationship diagram* menggambarkan hubungan antar entitas dalam sebuah basis data. Pada subbab ini, *entity relationship diagram* menggambarkan hubungan antar entitas pada sistem. ERD sistem akan ditunjukkan pada Gambar 4.16.




Gambar 4.16 *Entity Relationship Diagram*
Sumber : (Hasil Analisis, 2015)

Gambar 4.16 merupakan *entity relationship diagram* untuk sistem optimasi rute terpendek. Sesuai dengan fitur-fitur yang dibuat pada sistem optimasi ini, entitas yang dibutuhkan data_path, data_tpa, data_koordinat, data_sopir, user, data_record, data_truk, data_jarak, data_jalan, dan jalan_terhubung. Entitas data_tpa memiliki relasi *one-to-many* dengan entitas data_path. Data_tpa memiliki relasi *one-to-many* dengan entitas data_koordinat, dengan id_tpa sebagai *foreign key*. Data_truk memiliki relasi *one-to-many* dengan data_record, dengan id_truk sebagai *foreign key*. Data_jalan memiliki relasi *one-to-many* dengan data_jarak, dengan id_jalan sebagai *foreign key*. Data_jalan memiliki relasi *one-to-many* dengan entitas jalan_terhubung, dengan id_jalan sebagai *foreign key*.

## 4.4 Pengkodean

Pengkodean atau *coding* merupakan proses menulis kode dalam membangun sebuah program. Dalam proses pengkodean sistem optimasi rute terpendek pengangkutan sampah, penulis menggunakan bahasa pemrograman *Page Hyper Text Pre-processor* (PHP), *cascading Style Sheet* (CSS), dan *Javascript*. MySQL sebagai sistem manajemen database. Pengkodean sistem menggunakan konsep *object oriented programming* (OOP) menggunakan *framework codeigniter*. Berikut potongan kode program untuk fitur info jalan yang terdapat pada kelas *controller* yang digambarkan pada Tabel 4.11

Tabel 4.11 potongan kode program untuk fitur info jalan (*controller*)

```
function combo2() {
    $id = $_POST['combo'];
    $isi = $this->m_info->get_titik($id);
    $data = '<option value=""> -- Pilih Opsi -- </option>';
    for ($i = 0, $ln = count($isi); $i < $ln; $i++) {
        $data .= '<option value="' . $isi[$i]['latitude'] . ',' .
                $isi[$i]['longitude'] . '">' . $isi[$i]['nama'] . '</option>';
    }
    echo $data;
}
```

Tabel 4.11 merupakan potongan kode program fitur info_ jalan pada kelas *contoller*. Nama *function* yang digunakan adalah combo2(), didalamnya terdapat beberapa variabel dan beberapa kode program untuk memanggil *function* yang berada pada kelas model m_info yaitu get_titik(). Potongan kode program pada *function* get_titik() dapat dilihat Tabel 4.12

Tabel 4.12 potongan kode program pada *function* get_titik()

```
function get_titik($id = NULL) {
    $query = $this->db->query('SELECT a.id_TPA, a.nama, b.latitude,
            b.longitude FROM data_tpa a, data_koordinat b ' .
            (($id != NULL) ? (' WHERE a.id_TPA <> ' . $id . ' AND a.id_TPA = b.id_TPA') :
        'WHERE a.id_TPA = b.id_TPA'));
    return $query->result_array();
}
```

Tabel 4.12 merupakan potongan kode program model m_info yaitu *function* get_titik(). Terdapat beberapa logika dalam menentukan pemanggilan titik koordinat latitude dan longitude dari database TPA. Kode program fitur-fitur lain yang dibangun pada sistem optimasi rute terpendek sistem pengangkutan sampah dapat dilihat pada Lampiran C.

# BAB 5. HASIL DAN PEMBAHASAN

Bab ini menjelaskan tentang hasil dan pembahasan dari sistem optimasi rute terpendek pengangkutan sampah di Surabaya menggunakan ACS.

## 5.1 Implementasi Algoritma

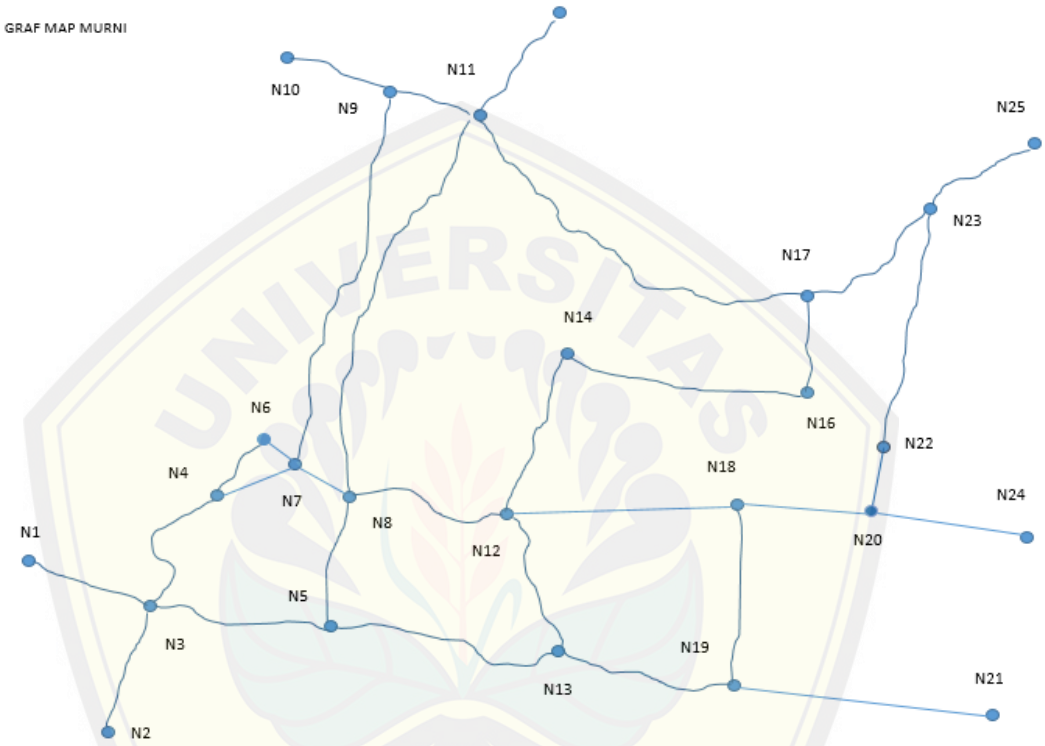
Berikut adalah simulasi proses pencarian rute terpendek untuk digunakan sebagai jalur pengangkutan sampah dari TPS ke TPA dengan algoritma ACS. Simulasi dilakukan dengan melakukan perhitungan secara manual dengan mengambil sampel contoh dari studi kasus TPS Kalijudan ke TPS Pacar Keling. Algoritma ACS akan menentukan rute terpendek dari jalan-jalan yang ada di Surabaya untuk dijadikan jalur pengangkutan sampah menuju ke tempat pembuangan selanjutnya. Untuk pengimplementasian Algoritma ACS, konsep graf diterapkan pada jalan-jalan di Surabaya. Jalan-jalan di Surabaya akan dijadikan *edge*s untuk pencarian dan persimpangan antara jalan akan dijadikan node atau titik pertemuan jalan. Setiap node akan diberi identifikasi dengan nama "nomornode" contohnya N2. Algoritma ACS akan melakukan perhitungan setiap jalan untuk menentukan rute terpendek berdasarkan *edge*s dan node tersebut.

Proses yang dilakukan pertama kali pada algoritma ini adalah menentukan titik awal dan titik tujuan. Sebagai contoh TPS Kalijudan dipilih sebagai titik awal dan TPS Pacar Keling dipilih sebagai titik tujuan. Koordinat tiap TPS yang telah disimpan sebagai database akan dipanggil sebagai parameter awal perhitungan. Data koordinat dapat dilihat pada Tabel 5.1

Tabel 5.1 Data koordinat TPS Kalijudan dan TPS Pacar Keling

| No. | Nama TPS | Lat | Long |
|---|---|---|---|
| 1. | TPS Kalijudan | -7.262475 | 112.775769 |
| 2. | TPS Pacar Keling | -7.25941 | 112.75517 |

Jika direpresentasikan kedalam graf, tampilan peta area TPS Kalijudan dan TPS Pacar Keling dapat dilihat pada Gambar. TPS Kalijudan dengan node N22 dan TPS Pacar Keling menggunakan node N6. Gambar 5.1 merupakan gambar yang menunjukkan area TPS Kalijudan dan TPS Pacar Keling dalam graf.




Gambar 5.1 Area TPS Kalijudan dan TPS Pacar Keling dalam graf

Dari graf diatas, jarak antar node dapat ditampilkan dan diinputkan kedalam sebuah tabel. Kemudian akan dicari rute terpendeknya. Tabel 5.1 merupakan tabel jarak. Dari jarak antar kota yang telah diketahui dapat dihitung visibilitas antar node yaitu $\eta_{ij} = 1/d_{ij}$ . Sedangkan untuk intensitas feromon antar node akan ditunjukkan oleh kolom $\boldsymbol{\tau ij}$. Nilai dari $\boldsymbol{\tau ij}$ adalah 0.01. Karena $\boldsymbol{\tau ij}$ adalah intensitas awal feromon yang belum mengalami perubahan. Nilai dari parameter visibilitas dan intensitas feromon ini nantinya akan digunakan dalam persamaan probabilitas dan merupakan

parameter yang mempengaruhi semut dalam pemilihan titik berikutnya (aturan transisi).

Berikut ini merupakan langkah-langkah perhitungan rute terpendek menggunakan algoritma ACS. Parameter-parameter yang digunakan ditunjukkan oleh Tabel 5.2.

Tabel 5.2 Parameter Algoritma

| No. | Parameter | Nilai |
|---|---|---|
| 1. | Alfa ($\alpha$) | 0.1 |
| 2. | Beta ($\beta$) | 1.0 |
| 3. | Rho ($\rho$) | 0.1 |
| 4. | Tetapan siklus semut (Q) | 1 |
| 5. | Siklus maksimum (NCmax) | 1 |
| 6. | Jumlah semut (m) | 3 |
| 7. | Intensitas Feromon awal | 0.01 |

Sumber : (Hasil Analisis, 2015)

Mencari node tujuan berikutnya dengan perhitungan probabilitas:

Siklus ke-1

**a. Semut 1**

Pada semut 1 ini terdapat pencarian untuk menuju ke node selanjutnya dari node awal yaitu node N22 ke node tujuan akhir yaitu node N6. Dalam pencarian probabilitas tersebut terdapat beberapa tahap. Tahap-tahap tersebut adalah:

1) Semut 1 dari N22 ke N6

Semut 1 → tujuan

Titik awal = N22. Titik-titik yang terhubung dengan N22 adalah N20 dan N23. Berikut merupakan tabel jarak antara node N22, N20, dan N23. Nilai intensitas feromon = 0.01. Maka nilai $\eta_{ij}$ adalah

| Node ke- | N22 | N20 | N23 |
|---|---|---|---|
| **N22** | 0 | 0.2 | 0.6 |
| **N20** | 0.2 | 0 | 0 |
| **N23** | 0.6 | 0 | 0 |

| Node | $\eta_{ij}$ |
|---|---|
| N22 | 5 |
| N23 | 1.67 |

2) Probabilitas node selanjutnya

Probabilitas dari N22 ke setiap node berikutnya dapat dihitung menggunakan persamaan

$$P_{ij}^{k}=\frac{[\tau_{ij}]^{\alpha}.[\eta_{ij}]^{\beta}}{\Sigma_{k\in[N-tabu_k]}[\tau_{ik^i}]^{\alpha}.[\eta_{ikj}]^{\beta}}$$

$\sum [\tau_{ik^i}]^{\alpha}.[\eta_{ikj}]^{\beta}$ = (0.63096*0) + ( 0.63096*5) + (0.63096*1.67)

= 0 + 3.1548 + 1.0537

= 4.2085

Dengan demikian dapat dihitung probabilitas dari node N22 menuju ke tiap node adalah

Node N22 = 0.00

Node N20 = (0.63096 * 5) / 4.2085= 3.1548 / 4.2085 = 0.7497

Node N23 = (0.63096 * 1.67) / 4.2085= 1.0537 /4.2085 = 0.2503

3) Probabilitas Kumulatif

Probabilitas kumulatif nya yaitu:

Node N22 = 0.00

Node N20 = 0.7497

Node N23 = 1

4) Bilangan Acak dan Pengecekan

Bilangan acak nya = 1.9

Memeriksa qk-1 < r ≤ qk untuk:

qk = 0.7497, maka (0.7497– 1) < 1.9≤ 0.7497

= -0.2503 < 1.9 ≤ 0.7497

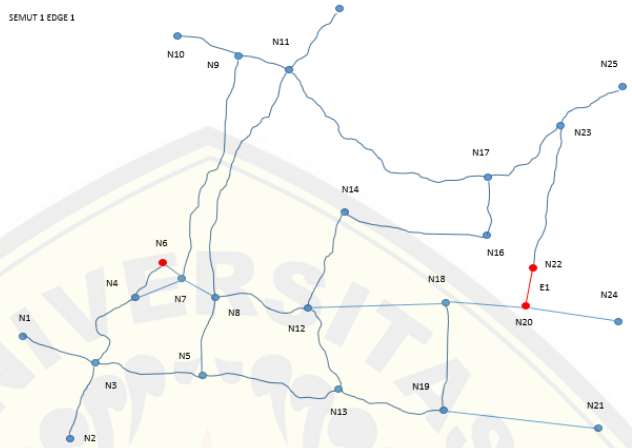
qk = 1, maka (1 – 1) < 1.9 ≤ 1

= 0 < 1.9 ≤ 1

Karena nilai qk sama maka akan diambil acak node selanjutnya, yaitu N20 sebagai titik berikutnya.

5) Rute Selanjutnya

Rute yang didapat → N22 N20. Gambar rute yang didapat dapat dilihat pada Gambar 5.2.




Gambar 5.2 Semut 1 untuk *edge* 1

6) Pembaruan Feromon Lokal

Pembaruan feromon lokal untuk *edge*s yang menghubungkan node N22 dan N20 dengan persamaan

$$\tau(i,s) \leftarrow (1-\rho).\tau(i,s) + \rho.\Delta\tau(i,s)$$

$$\Delta\tau(i,s) = \frac{1}{L_{nn}.C}$$

$$\Delta\tau\ (1,2) = \frac{1}{0.2 * 3} = \frac{1}{0.6} = 1.67$$

$$\rho.\Delta\tau(i,s) = 0.1 * 1.67 = 0.167$$

$$\tau(1,2) \leftarrow ((1-0.1) * (0.01)) + 0.167$$

$$\tau(1,2) \leftarrow \big((0.9) * (0.01)\big) + (0.167)$$

$$\tau(1,2) \leftarrow 0.009 + 0.167 = 0.176$$

Didapatkan N20 sebagai node selanjutnya pada perhitungan diatas. Kemudian akan dilanjutkan menghitung probabilitas pencarian node selanjutnya. Dalam pencarian probabilitas tersebut terdapat beberapa tahap. Tahap-tahap tersebut adalah:

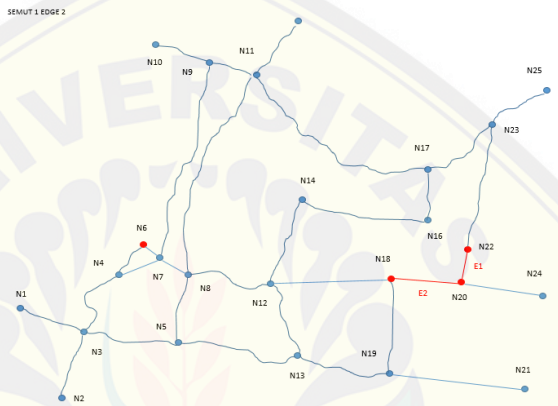
1) Semut 1 dari N20 ke node selanjutnya

Semut 1 → N22 N20

Titik awal = N20. Titik-titik yang terhubung dengan N20 adalah N18 saja. Maka dapat diperoleh titik selanjutnya adalah N18

2) Rute selanjutnya

Rute yang didapat → N22 N20 N18. Gambar rute yang didapat dapat dilihat pada Gambar 5.3.




Gambar 5.3 Semut 1 untuk *edge* 2

3) Pembaruan Feromon Lokal

Pembaruan feromon lokal untuk *edge*s yang menghubungkan node N20 dan N18 dengan persamaan

$$\tau(i,s) \leftarrow (1-\rho).\tau(i,s) + \rho.\Delta\tau(i,s)$$

$$\Delta\tau(i,s) = \frac{1}{L_{nn}.C}$$

$$\Delta\tau\ (2,3) = \frac{1}{0.5 * 3} = \frac{1}{1.5} = 0.67$$

$$\rho.\Delta\tau(i,s) = 0.1 * 0.67 = 0.067$$

$$\tau(2,3) \leftarrow ((1-0.1) * (0.01)) + 0.067$$

$$\tau(2,3) \leftarrow \big((0.9) * (0.01)\big) + (0.067)$$

$$\tau(2,3) \leftarrow 0.009 + 0.067 = 0.076$$

Didapatkan N18 sebagai node selanjutnya pada perhitungan diatas. Kemudian akan dilanjutkan menghitung probabilitas pencarian node selanjutnya. Dalam pencarian probabilitas tersebut terdapat beberapa tahap. Tahap-tahap tersebut adalah:

1) Semut 1 dari N18 ke node selanjutnya

Semut 1 → N22 N20 N18

Titik awal = N18. Titik-titik yang terhubung dengan N18 adalah N19 dan N12. Berikut merupakan tabel jarak antara node N18, N12, dan N19. Nilai intensitas feromon = 0.01. Maka nilai $\eta_{ij}$ adalah

| **Node ke-** | **N18** | **N12** | **N19** |
|---|---|---|---|
| **N18** | 0 | 1.4 | 0.7 |
| **N12** | 1.4 | 0 | 0 |
| **N19** | 0.7 | 0 | 0 |

| **Node** | $\eta_{ij}$ |
|---|---|
| N12 | 0.714 |
| N19 | 1.428 |

2) Menghitung probabilitas dari node N18 ke node selanjutnya

Probabilitas dari N18 ke setiap node berikutnya dapat dihitung menggunakan persamaan

$$P_{ij}^{k}=\frac{[\tau_{ij}]^{\alpha}.[\eta_{ij}]^{\beta}}{\Sigma_{k\in[N-tabu_k]}[\tau_{ik^i}]^{\alpha}.[\eta_{ik^j}]^{\beta}}$$

$$\sum [\tau_{ik^i}]^{\alpha}.[\eta_{ik^j}]^{\beta} = (0.63096*0) + ( 0.63096*0.714) + (0.63096*1.428)$$

$$= 0 + 0.4479 + 0.8959$$

$$= 1.3438$$

Dengan demikian dapat dihitung probabilitas dari node N18 menuju ke tiap node adalah

Node N18 = 0.00

Node N12 = ( 0.63096*0.714) / 1.3438= 0.4479 /1.3438 = 0.3333

Node N19 = (0.63096*1.428)/ 1.3438= 0.8959 / 1.3438 = 0.6666

3) Probabilitas Kumulatif

Probabilitas kumulatif nya yaitu:

Node N15 = 0.00

Node N12 = 0.3333

Node N19 = 0.9999

4) Bilangan acak dan Pengecekan

Bilangan acak nya = 0.5

Memeriksa qk-1 < r ≤ qk untuk:

qk = 0.3333, maka (0.3333– 1) < 0.5≤ 0.3333

= -0.6667< 56.777 ≤ 0.3333

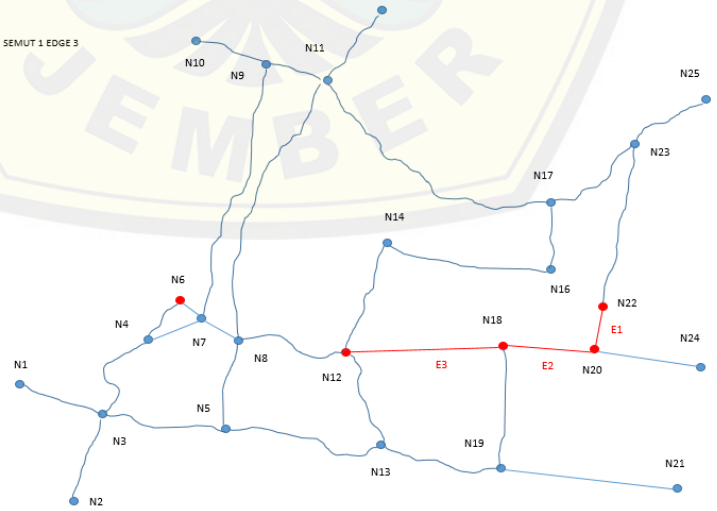
qk = 0.9999, maka (0.9999– 1) < 0.5≤ 0.9999

= -0.0001< 56.777≤ 0.9999

Karena nilai qk untuk N12 dan N19 bernilai benar maka dipilihlah N12 sebagai node selanjutnya.

5) Rute Selanjutnya

Rute yang didapat → N22 N20 N18 N12. Gambar rute yang didapat dapat dilihat pada Gambar 5.4.




Gambar 5.4 Semut 1 untuk *edge* 3

6) Pembaruan feromon lokal

Pembaruan feromon lokal untuk *edge*s yang menghubungkan node N18 dan N12 dengan persamaan

$$\tau(i,s) \leftarrow (1-\rho).\tau(i,s) + \rho.\Delta\tau(i,s)$$

$\Delta\tau(i,s) = \frac{1}{L_{nn}.C}$

$\Delta\tau$ (3,4) = $\frac{1}{1.4 * 3} = \frac{1}{4.2}$ = 0.238

$\rho.\Delta\tau(i,s)$ = 0.1 * 0.238 = 0.0238

$\tau(3,4) \leftarrow ((1-0.1) * (0.01)) + 0.0238$

$\tau(3,4) \leftarrow \big((0.9) * (0.01)\big) + (0.0238)$

$\tau(3,4) \leftarrow 0.009 + 0.0238 = 0.0328$

Didapatkan N12 sebagai node selanjutnya pada perhitungan diatas. Kemudian akan dilanjutkan menghitung probabilitas pencarian node selanjutnya. Dalam pencarian probabilitas tersebut terdapat beberapa tahap. Tahap-tahap tersebut adalah:

1) Semut 1 dari N12 ke node selanjutnya

Semut 1 → N22 N20 N18 N12

Titik awal = N12. Titik-titik yang terhubung dengan N12 adalah N13, N14, dan N8. Berikut merupakan tabel jarak antara node N12, N13, N14 dan N8. Nilai intensitas feromon = 0.01. Maka nilai $\eta_{ij}$ adalah

| **Node ke-** | **N12** | **N13** | **N14** | **N8** |
|---|---|---|---|---|
| **N12** | 0 | 0.1 | 0.7 | 0.4 |
| **N13** | 0.1 | 0 | 0 | 0 |
| **N14** | 0.7 | 0 | 0 | 0 |
| **N8** | 0.4 | 0 | 0 | 0 |

| **Node** | $\eta_{ij}$ |
|---|---|
| N13 | 10 |
| N14 | 1.428 |
| N8 | 2.5 |

2) Menghitung probabilitas dari node N12 ke node berikutnya

Probabilitas dari N12 ke setiap node berikutnya dapat dihitung menggunakan persamaan

$$P_{ij}^{k}=\frac{[\tau_{ij}]^{\alpha}.[\eta_{ij}]^{\beta}}{\Sigma_{k\in[N-tabu_k]}[\tau_{ik}i]^{\alpha}.[\eta_{ikj}]^{\beta}}$$

$\sum [\tau_{ik}i]^{\alpha}.[\eta_{ikj}]^{\beta}$ = (0.63096*0) + ( 0.63096*10) + (0.63096*1.42) + (0.63096*2.5)

= 0 + 6.3096 + 0.8959 + 1.5773

= 8.7828

Dengan demikian dapat dihitung probabilitas dari node N12 menuju ke tiap node adalah

Node N12 = 0.00

Node N13 = ( 0.63096*10) / 8.7828= 0.7184

Node N14 = (0.63096*1.42) / 8.7828= 0.1020

Node N8 = (0.63096*2.5)/ 8.7828= 0.1795

3) Probabilitas Kumulatif

Probabilitas kumulatif nya yaitu:

Node N12 = 0.00

Node N13 = 0.7184

Node N14 = 0.8204

Node N8 = 0.9999

4) Bilangan acak dan Pengecekan

Bilangan acak nya = 0.94

Memeriksa qk-1 < r ≤ qk untuk:

qk = 0.7184, maka (0.7184– 1) < 0.94≤ 0.7184

= -0.2816< 0.94≤ 0.7184

qk = 0.8204, maka (0.8204– 1) < 0.94≤ 0.8204

= -0.1796< 0.94≤ 82.049

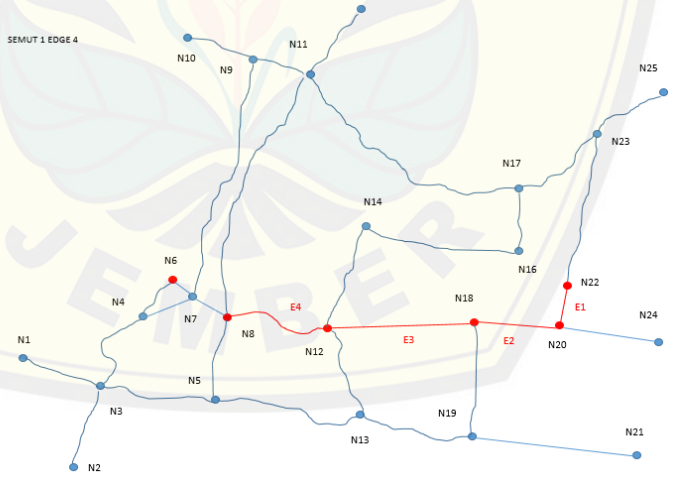
qk = 0.9999, maka (0.9999– 1) < 0.94≤ 0.9999

= -0.0001< 0.94≤ 99.998

Karena nilai qk untuk N13,N14, dan N8 bernilai benar maka diambil N8 sebagai node selanjutnya.

5) Rute Selanjutnya

Rute yang didapat → N22 N20 N18 N12 N8. Gambar rute yang didapat dapat dilihat pada Gambar 5.5.




Gambar 5.5 Semut 1 untuk *edge* 4

6) Pembaruan feromon lokal

Pembaruan feromon lokal untuk *edge*s yang menghubungkan node N12 dan N8 dengan persamaan

$$\tau(i,s) \leftarrow (1-\rho).\tau(i,s) + \rho.\Delta\tau(i,s)$$

$\Delta\tau(i,s) = \frac{1}{L_{nn}.C}$

$\Delta\tau$ (5,6) = $\frac{1}{0.4 * 3} = \frac{1}{1.2}$ = 0.833

$\rho.\Delta\tau(i,s)$ = 0.1 * 0.833 = 0.0833

$\tau(5,6) \leftarrow ((1-0.1) * (0.01)) + 0.0833$

$\tau(5,6) \leftarrow \big((0.9) * (0.01)\big) + (0.0833)$

$\tau(5,6) \leftarrow 0.009 + 0.0833$ = 0.0923

Didapatkan N8 sebagai node selanjutnya pada perhitungan diatas. Kemudian akan dilanjutkan menghitung probabilitas pencarian node selanjutnya. Dalam pencarian probabilitas tersebut terdapat beberapa tahap. Tahap-tahap tersebut adalah:

1) Semut 1 dari N8 ke node selanjutnya

Semut 1 → N22 N20 N18 N12 N8

Titik awal = N8. Titik-titik yang terhubung dengan N8 adalah N5, N7, dan N11. Berikut merupakan tabel jarak antara node N8, N5, N7 dan N11. Nilai intensitas feromon = 0.01. Maka nilai $\eta_{ij}$ adalah

| **Node ke-** | **N8** | **N5** | **N7** | **N11** |
|---|---|---|---|---|
| **N8** | 0 | 0.3 | 0.041 | 1.2 |
| **N5** | 0.3 | 0 | 0 | 0 |
| **N7** | 0.041 | 0 | 0 | 0 |
| **N11** | 1.2 | 0 | 0 | 0 |

| **Node** | $\eta_{ij}$ |
|---|---|
| N5 | 3.33 |
| N7 | 24.39 |
| N11 | 5 |

2) Menghitung probabilitas dari node N8 ke node berikutnya

Probabilitas dari N8 ke setiap node berikutnya dapat dihitung menggunakan persamaan

$$P_{ij}^{k}=\frac{[\tau_{ij}]^{\alpha}.[\eta_{ij}]^{\beta}}{\Sigma_{k\in[N-tabu_k]}[\tau_{ik^i}]^{\alpha}.[\eta_{ik^j}]^{\beta}}$$

$\sum [\tau_{ik^i}]^{\alpha}.[\eta_{ik^j}]^{\beta}$ = (0.63096*0) + ( 0.63096*3.33) + (0.63096*24.3) + (0.63096*5)

= 0 + 2.1010 + 15.332 + 3.1547

= 20.5877

Dengan demikian dapat dihitung probabilitas dari node N8 menuju ke tiap node adalah

Node N8 = 0.00

Node N5 = ( 0.63096*3.33) / 20.5877= 0.1020

Node N7 = (0.63096*24.3) / 20.5877= 0.7447

Node N11 = (0.63096*5) / 20.5877= 0.1532

3) Probabilitas kumulatif

Probabilitas kumulatif nya yaitu:

Node N8 = 0.00

Node N5 = 0.1020

Node N7 = 0.8467

Node N11 = 0.9999

4) Bilangan acak dan Pengecekan

Bilangan acak nya = 0.50

Memeriksa qk-1 < r ≤ qk untuk:

qk = 0.1020, maka (0.1020– 1) < 0.50≤ 0.1020

= -0.898< 0.50≤ 0.1020

qk = 0.8467, maka (0.8467– 1) < 50.123≤ 0.8467

= -0.1533< 0.50≤ 0.8467

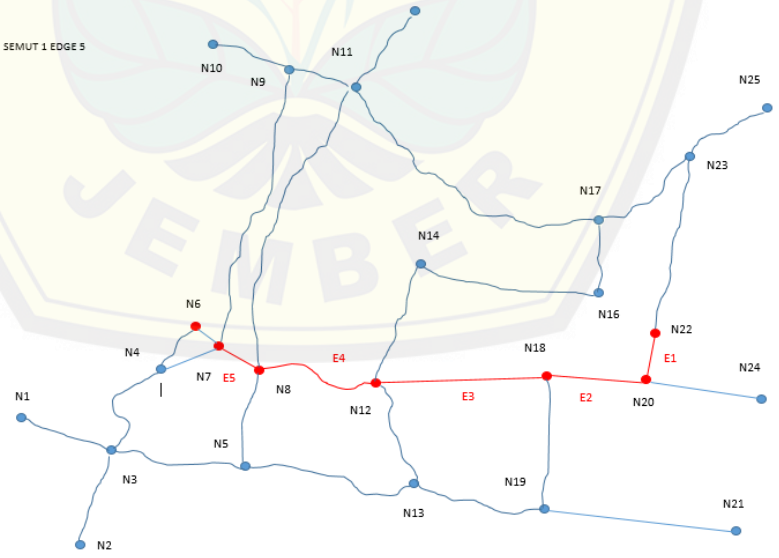
qk = 0.9999, maka (0.9999– 1) < 0.50≤ 0.9999

= -0.0001< 0.50≤ 52.16

Karena nilai qk untuk N5, N11 dan N7 bernilai benar maka diambil N7 sebagai node selanjutnya.

5) Rute selanjutnya

Rute yang didapat → N22 N20 N18 N12 N8 N7. Gambar rute yang didapat dapat dilihat pada Gambar 5.6.




Gambar 5.6 Semut 1 untuk *edge* 5

6) Pembaruan feromon lokal

Pembaruan feromon lokal untuk *edge*s yang menghubungkan node N8 dan N7 dengan persamaan

$$\tau(i,s) \leftarrow (1-\rho).\tau(i,s) + \rho.\Delta\tau(i,s)$$

$\Delta\tau(i,s) = \frac{1}{L_{nn}.C}$

$\Delta\tau\ (6,7) = \frac{1}{0.041 * 3} = \frac{1}{0.123} = 8.13$

$\rho.\Delta\tau(i,s) = 0.1 * 8.13 = 0.813$

$\tau(6,7) \leftarrow ((1-0.1) * (\ 0.01)) + \ 0.813$

$\tau(6,7) \leftarrow \big((0.9) * (0.01)\big) + (0.813)$

$\tau(6,7) \leftarrow 0.009 + 0.813 = 0.822$

Didapatkan N7 sebagai node selanjutnya pada perhitungan diatas. Kemudian akan dilanjutkan menghitung probabilitas pencarian node selanjutnya. Dalam pencarian probabilitas tersebut terdapat beberapa tahap. Tahap-tahap tersebut adalah:

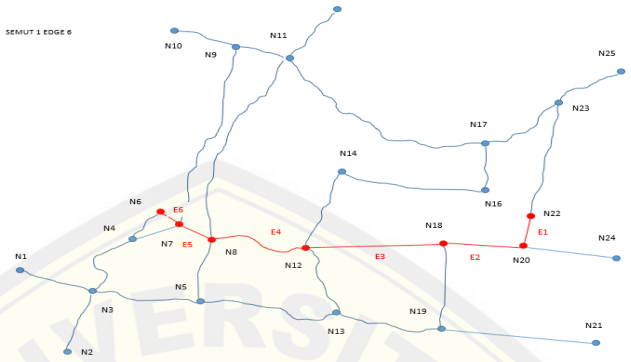
1) Semut 1 dari N7 ke node selanjutnya

Semut 1 → N22 N20 N18 N12 N8 N7

Titik awal = N7. Titik-titik yang terhubung dengan N7 adalah N4 dan N6. Node N6 merupakan node tujuan yang dicari maka tidak perlu dicari nilai probabilitas nya. N6 merupakan node selanjutnya.

2) Rute selanjutnya

Rute yang didapat → N22 N20 N18 N12 N8 N7 N6. Gambar rute yang didapat dapat dilihat pada Gambar 5.7.




Gambar 5.7 Semut 1 untuk *edge* 6

3) Pembaruan feromon lokal

Pembaruan feromon lokal untuk *edge*s yang menghubungkan node N7 dan N6 dengan persamaan

$$\tau(i,s) \leftarrow (1-\rho).\tau(i,s) + \rho.\Delta\tau(i,s)$$

$$\Delta\tau(i,s) = \frac{1}{L_{nn}.C}$$

$$\Delta\tau\ (7,8) = \frac{1}{0.08 * 3} = \frac{1}{0.3} = 0.24$$

$$\rho.\Delta\tau(i,s) = 0.1 * 0.24 = 0.024$$

$$\tau(7,8) \leftarrow ((1-0.1) * (0.01)) + 0.024$$

$$\tau(7,8) \leftarrow \big((0.9) * (0.01)\big) + (0.024)$$

$$\tau(7,8) \leftarrow 0.009 + 0.024 = 0.033$$

Dari perhitungan semut 1 diatas didapatkan rute N22 N20 N18 N12 N8 N7 N6. Dengan total jarak 2,621 Km. Perhitungan akan dilanjutkan menggunakan semut 2. Terdapat beberapa tahap untuk mencari probabilitas untuk menemukan rute dari N22 menuju N6.

**b. Semut 2**

Pada semut 2 ini terdapat pencarian untuk menuju ke node selanjutnya dari node awal yaitu node N22 ke node tujuan akhir yaitu node N6. Dalam pencarian probabilitas tersebut terdapat beberapa tahap. Tahap-tahap tersebut adalah:

1) Semut ke-2 dari N22 ke N6

Semut 2 → tujuan

Titik awal = N22. Titik-titik yang terhubung dengan N22 adalah N20 dan N23. Berikut merupakan tabel jarak antara node N22, N20, dan N23. Nilai intensitas feromon = 0.01. Maka nilai $\eta_{ij}$ adalah

| Node ke- | N22 | N20 | N23 |
|---|---|---|---|
| **N22** | 0 | 0.2 | 0.6 |
| **N20** | 0.2 | 0 | 0 |
| **N23** | 0.6 | 0 | 0 |

| Node | $\eta_{ij}$ |
|---|---|
| N22 | 5 |
| N23 | 1.67 |

2) Probabilitas dari N22 ke node selanjutnya

Probabilitas dari N22 ke setiap node berikutnya dapat dihitung menggunakan persamaan

$$P_{ij}^{k}=\frac{[\tau_{ij}]^{\alpha}.[\eta_{ij}]^{\beta}}{\sum_{k\in[N-tabu_k]}[\tau_{ik^i}]^{\alpha}.[\eta_{ik^j}]^{\beta}}$$

$\sum [\tau_{ik^i}]^{\alpha}.[\eta_{ik^j}]^{\beta}$ = (0.63096*0) + (0.63096*5) + (0.63096*1.67)

= 0 + 3.1547 + 1.0536

= 4.2083

Dengan demikian dapat dihitung probabilitas dari node N22 menuju ke tiap node adalah

Node N22 = 0.00

Node N20 = (0.63096*5) / 4.2083= 0.7496

Node N23 = (0.63096*1.67) / 4.2083= 0.2503

3) Probabilitas kumulatif

Probabilitas kumulatif nya yaitu:

Node N22 = 0.00

Node N20 = 0.7496

Node N23 = 0.9999

4) Bilangan acak dan Pengecekan

Bilangan acak nya = 1.79

Memeriksa qk-1 < r ≤ qk untuk:

qk = 0.7496, maka (0.7496– 1) < 1.79≤ 0.7496

= -0.2504< 1.79≤ 0.7496

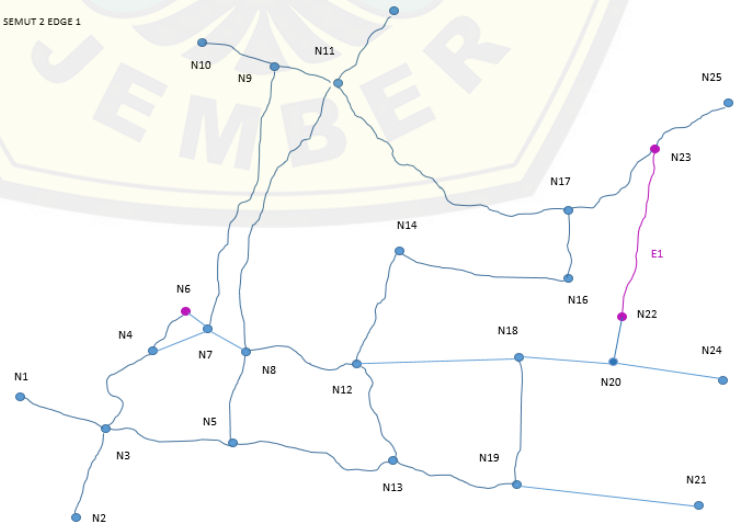
qk = 0.9999, maka (0.9999– 1) < 1.79≤ 0.9999

= -0.0001< 1.79≤ 0.9999

Karena nilai qk sama maka akan diambil acak node selanjutnya, yaitu N23 sebagai titik berikutnya.

5) Rute selanjutnya

Rute yang didapat → N22 N23. Gambar rute yang didapat dapat dilihat pada Gambar 5.8.




Gambar 5.8 Semut 2 untuk *edge* 1

6) Pembaruan Feromon Lokal

Pembaruan feromon lokal untuk *edge*s yang menghubungkan node N22 dan N23 dengan persamaan

$$\tau(i,s) \leftarrow (1-\rho).\tau(i,s) + \rho.\Delta\tau(i,s)$$

$\Delta\tau(i,s) = \frac{1}{L_{nn}.C}$

$\Delta\tau\ (1,2) = \frac{1}{0.6 * 3} = \frac{1}{1.8} = 0.56$

$\rho.\Delta\tau(i,s) = 0.1 * 0.56 = 0.056$

$\tau(1,2) \leftarrow ((1-0.1) * (\ 0.01)) + \ 0.056$

$\tau(1,2) \leftarrow \big((0.9) * (0.01)\big) + (0.056)$

$\tau(1,2) \leftarrow 0.009 + 0.056 = 0.065$

Didapatkan N23 sebagai node selanjutnya pada perhitungan diatas. Kemudian akan dilanjutkan menghitung probabilitas pencarian node selanjutnya. Dalam pencarian probabilitas tersebut terdapat beberapa tahap. Tahap-tahap tersebut adalah:

1) Semut ke-2 dari N22 ke N6

Semut 2 → N22 N23

Titik awal = N23. Titik-titik yang terhubung dengan N23 adalah N25 dan N17. Berikut merupakan tabel jarak antara node N23, N25, dan N17. Nilai intensitas feromon = 0.01. Maka nilai $\eta_{ij}$ adalah

| **Node ke-** | **N23** | **N25** | **N17** |
|---|---|---|---|
| **N23** | 0 | 1.2 | 0.2 |
| **N25** | 1.2 | 0 | 0 |
| **N17** | 0.2 | 0 | 0 |

| **Node** | $\eta_{ij}$ |
|---|---|
| N25 | 0.833 |
| N17 | 5 |

2) Probabilitas dari N23 ke node selanjutnya

Probabilitas dari N23 ke setiap node berikutnya dapat dihitung menggunakan persamaan

$$P_{ij}^{k}=\frac{[\tau_{ij}]^{\alpha}.[\eta_{ij}]^{\beta}}{\Sigma_{k\in[N-tabu_k]}[\tau_{ik}i]^{\alpha}.[\eta_{ikj}]^{\beta}}$$

$\sum [\tau_{ik}i]^{\alpha}.[\eta_{ikj}]^{\beta}$ = (0.63096*0) + (0.63096*0.83) + (0.63096*5)

= 0 + 0.5236 + 3.1547

= 3.6783

Dengan demikian dapat dihitung probabilitas dari node N22 menuju ke tiap node adalah

Node N22 = 0.00

Node N25 = (0.63096*0.83) / 3.6783= 0.1423

Node N17 = (0.63096*5) / 3.6783= 0.8576

3) Probabilitas kumulatif

Probabilitas kumulatif nya yaitu:

Node N22 = 0.00

Node N25 = 0.1423

Node N17 = 0.9999

4) Bilangan acak dan Pengecekan

Bilangan acak nya = 0.79

Memeriksa qk-1 < r ≤ qk untuk:

qk = 0.1423, maka (0.1423– 1) < 0.79≤ 0.1423

= -0.8577< 0.79≤ 0.1423

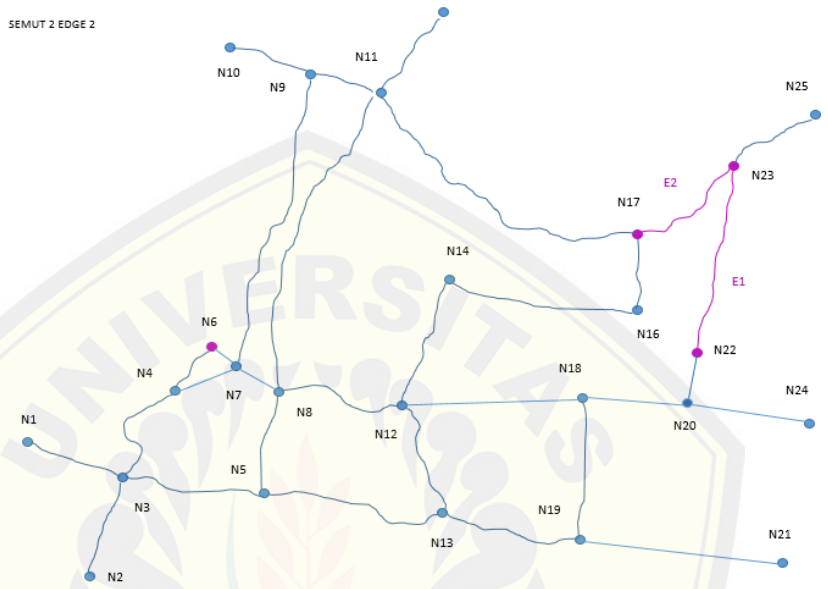
qk = 0.9999, maka (0.9999– 1) < 0.79≤ 0.9999

= -0.0001< 0.79≤ 0.9999

Karena nilai qk sama maka akan diambil acak node selanjutnya, yaitu N17 sebagai titik berikutnya.

5) Rute selanjutnya

Rute yang didapat → N22 N23 N17. Gambar rute yang didapat dapat dilihat pada Gambar 5.9.




Gambar 5.9 Semut 2 untuk *edge* 2

6) Pembaruan feromon lokal

Pembaruan feromon lokal untuk *edge*s yang menghubungkan node N23 dan N17 dengan persamaan

$$\tau(i,s) \leftarrow (1-\rho).\tau(i,s) + \rho.\Delta\tau(i,s)$$

$$\Delta\tau(i,s) = \frac{1}{L_{nn}.C}$$

$$\Delta\tau\ (2,3) = \frac{1}{0.2 * 3} = \frac{1}{0.6} = 1.67$$

$$\rho.\Delta\tau(i,s) = 0.1 * 1.67 = 0.167$$

$$\tau(2,3) \leftarrow ((1-0.1) * (\ 0.01)) + \ 0.167$$

$$\tau(2,3) \leftarrow \big((0.9) * (0.01)\big) + (0.167)$$

$$\tau(2,3) \leftarrow 0.009 + 0.167 = 0.176$$

Didapatkan N17 sebagai node selanjutnya pada perhitungan diatas. Kemudian akan dilanjutkan menghitung probabilitas pencarian node selanjutnya. Dalam pencarian probabilitas tersebut terdapat beberapa tahap. Tahap-tahap tersebut adalah:

1) Semut ke-2 dari N22 ke N6

Semut 2 → N22 N23 N17

Titik awal = N17. Titik-titik yang terhubung dengan N17 adalah N11 dan N16. Berikut merupakan tabel jarak antara node N17, N11, dan N16. Nilai intensitas feromon = 0.01. Maka nilai $\eta_{ij}$ adalah

| Node ke- | N17 | N11 | N16 |
|---|---|---|---|
| **N17** | 0 | 0.1 | 1.3 |
| **N11** | 0.1 | 0 | 0 |
| **N16** | 1.3 | 0 | 0 |

| Node | $\eta_{ij}$ |
|---|---|
| N11 | 10 |
| N16 | 3.33 |

2) Probabilitas dari N17 ke node selanjutnya

Probabilitas dari N17 ke setiap node berikutnya dapat dihitung menggunakan persamaan

$$P_{ij}^{k}=\frac{[\tau_{ij}]^{\alpha}.[\eta_{ij}]^{\beta}}{\Sigma_{k\in[N-tabu_k]}[\tau_{ik^i}]^{\alpha}.[\eta_{ik^j}]^{\beta}}$$

$\sum [\tau_{ik^i}]^{\alpha}.[\eta_{ik^j}]^{\beta}$ = (0.63096*0) + (0.63096*10) + (0.63096*3.33)

= 0 + 6.3095 + 2.1010

= 8.4105

Dengan demikian dapat dihitung probabilitas dari node N17 menuju ke tiap node adalah

Node N17 = 0.00

Node N11 = (0.63096*10)/ 8.4105= 0.7501

Node N16 = (0.63096*3.33)/ 8.4105= 0.2498

3) Probabilitas kumulatif

Probabilitas kumulatif nya yaitu:

Node N11 = 0.00

Node N16 = 0.7501

Node N17 = 0.9999

4) Bilangan acak dan Pengecekan

Bilangan acak nya = 0.5

Memeriksa qk-1 < r ≤ qk untuk:

qk = 0.7501, maka (0.7501– 1) < 0.5 ≤ 0.7501

= -0.2499< 0.5 ≤ 0.7501

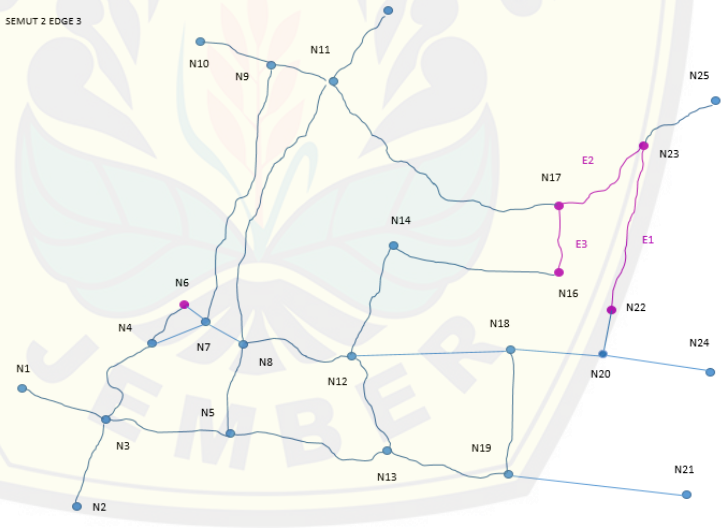
qk = 0.9999, maka (0.9999– 1) < 0.5 ≤ 0.9999

= -0.0001< 0.5 ≤ 0.9999

Karena nilai qk sama maka akan diambil acak node selanjutnya, yaitu N16 sebagai titik berikutnya.

5) Rute selanjutnya

Rute yang didapat → N22 N23 N17 N16. Gambar rute yang didapat dapat dilihat pada Gambar 5.10.




Gambar 5.10 Semut 2 untuk *edge* 3

6) Pembaruan feromon lokal

Pembaruan feromon lokal untuk *edge*s yang menghubungkan node N17 dan N16 dengan persamaan

$$\tau(i,s) \leftarrow (1-\rho).\tau(i,s) + \rho.\Delta\tau(i,s)$$

$$\Delta\tau(i,s) = \frac{1}{L_{nn}.C}$$

$$\Delta\tau\ (3,4) = \frac{1}{0.1 * 3} = \frac{1}{0.3} = 3.33$$

$$\rho.\Delta\tau(i,s) = 0.1 * 3.33 = 0.333$$

$$\tau(3,4) \leftarrow ((1-0.1) * (0.01)) + 0.333$$

$$\tau(3,4) \leftarrow \big((0.9) * (0.01)\big) + (0.333)$$

$$\tau(3,4) \leftarrow 0.009 + 0.333 = 0.342$$

Didapatkan N16 sebagai node selanjutnya pada perhitungan diatas. Kemudian akan dilanjutkan menghitung probabilitas pencarian node selanjutnya. Dalam pencarian probabilitas tersebut terdapat beberapa tahap. Tahap-tahap tersebut adalah:

1) Semut ke-2 dari N22 ke N6

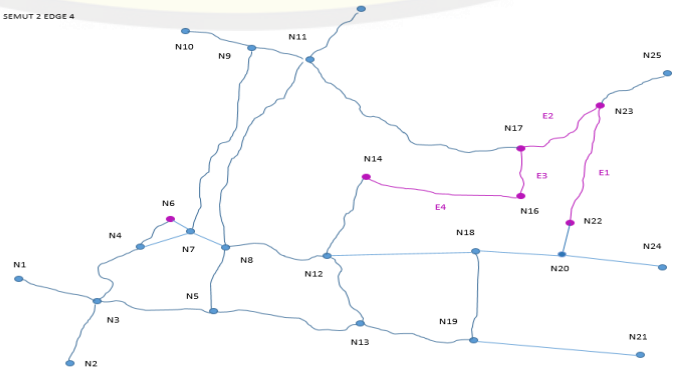
Semut 2 → N22 N23 N17 N16

Titik awal = N16. Titik-titik yang terhubung dengan N16 adalah N14 saja.

Maka dapat diperoleh titik selanjutnya adalah N14

2) Rute selanjutnya

Rute yang didapat → N22 N23 N17 N16 N14. Gambar rute yang didapat dapat dilihat pada Gambar 5.11.




Gambar 5.11 Semut 2 untuk *edge* 4

3) Pembaruan feromon lokal

Pembaruan feromon lokal untuk *edge*s yang menghubungkan node N16 dan N14 dengan persamaan

$$\tau(i,s) \leftarrow (1-\rho).\tau(i,s) + \rho.\Delta\tau(i,s)$$

$$\Delta\tau(i,s) = \frac{1}{L_{nn}.C}$$

$$\Delta\tau\ (4,5) = \frac{1}{0.8 * 3} = \frac{1}{2.4} = 0.416$$

$$\rho.\Delta\tau(i,s) = 0.1 * 0.416 = 0.0416$$

$$\tau(4,5) \leftarrow ((1-0.1) * (\ 0.01)) + \ 0.0416$$

$$\tau(4,5) \leftarrow \big((0.9) * (0.01)\big) + (0.0416)$$

$$\tau(4,5) \leftarrow 0.009 + 0.0416 = 0.0506$$

Didapatkan N14 sebagai node selanjutnya pada perhitungan diatas. Kemudian akan dilanjutkan menghitung probabilitas pencarian node selanjutnya. Dalam pencarian probabilitas tersebut terdapat beberapa tahap. Tahap-tahap tersebut adalah:

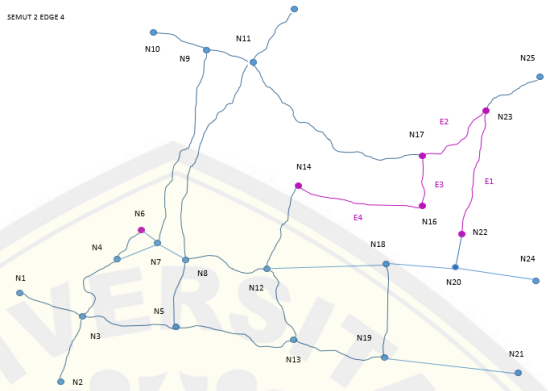
1) Semut ke-2 dari N22 ke N6

Semut 2 → N22 N23 N17 N16 N14

Titik awal = N14. Titik-titik yang terhubung dengan N14 adalah N12 saja. Maka dapat diperoleh titik selanjutnya adalah N12.

2) Rute selanjutnya

Rute yang didapat → N22 N23 N17 N16 N14 N12. Gambar rute yang didapat dapat dilihat pada Gambar 5.12.




Gambar 5.12 Semut 2 untuk *edge* 5

3) Pembaruan feromon lokal

Pembaruan feromon lokal untuk *edge*s yang menghubungkan node N14 dan N12 dengan persamaan

$$\tau(i,s) \leftarrow (1-\rho).\tau(i,s) + \rho.\Delta\tau(i,s)$$

$$\Delta\tau(i,s) = \frac{1}{L_{nn}.C}$$

$$\Delta\tau\ (5,6) = \frac{1}{0.7 * 3} = \frac{1}{2.1} = 0.476$$

$$\rho.\Delta\tau(i,s) = 0.1 * 0.476 = 0.0476$$

$$\tau(5,6) \leftarrow ((1-0.1) * (0.01)) + 0.0476$$

$$\tau(5,6) \leftarrow \big((0.9) * (0.01)\big) + (0.0476)$$

$$\tau(5,6) \leftarrow 0.009 + 0.0476 = 0.01376$$

Didapatkan N12 sebagai node selanjutnya pada perhitungan diatas. Kemudian akan dilanjutkan menghitung probabilitas pencarian node selanjutnya. Dalam pencarian probabilitas tersebut terdapat beberapa tahap. Tahap-tahap tersebut adalah:

1) Semut ke 2 dari N22 ke node selanjutnya

Semut 1 → N22 N23 N17 N16 N14 N12

Titik awal = N12. Titik-titik yang terhubung dengan N12 adalah N13 dan N8. Berikut merupakan tabel jarak antara node N12, N13, dan N8. Nilai intensitas feromon = 0.01. Maka nilai $\eta_{ij}$ adalah

| **Node ke-** | **N12** | **N13** | **N8** |
|---|---|---|---|
| **N12** | 0 | 0.1 | 0.4 |
| **N13** | 0.1 | 0 | 0 |
| **N8** | 0.4 | 0 | 0 |

| **Node** | $\eta_{ij}$ |
|---|---|
| N13 | 10 |
| N8 | 2.5 |

2) Probabilitas dari node N22 ke node selanjutnya

Probabilitas dari N12 ke setiap node berikutnya dapat dihitung menggunakan persamaan

$$P_{ij}^{k}=\frac{[\tau_{ij}]^{\alpha}.[\eta_{ij}]^{\beta}}{\sum_{k\in[N-tabu_k]}[\tau_{ik}{}^{i}]^{\alpha}.[\eta_{ik}{}^{j}]^{\beta}}$$

$\sum [\tau_{ik}{}^{i}]^{\alpha}.[\eta_{ik}{}^{j}]^{\beta}$ = (0.63096*0) + (0.63096*10) + (0.63096*2.5)

= 0 + 6.3096+ 1.5774

= 7.887

Dengan demikian dapat dihitung probabilitas dari node N12 menuju ke tiap node adalah

Node N12 = 0.00

Node N13 = (0.63096*10) / 7.887= 0.8

Node N8 = (0.63096*2.5) / 7.887= 0.2

3) Probabilitas kumulatif

Probabilitas kumulatif nya yaitu:

Node N12 = 0.00

Node N13 = 0.8

Node N8 = 0.10

4) Bilangan acak dan Pengecekan

Bilangan acak nya = 12.0

Memeriksa qk-1 < r ≤ qk untuk:

qk = 0.8, maka (0.8– 1) < 12.0≤ 0.8

= -0.2< 12.0≤ 0.8

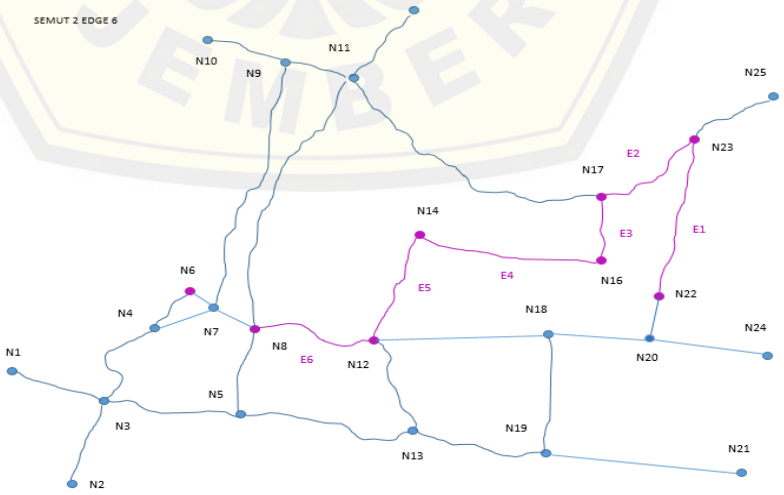
qk = 0.10, maka (0.10– 1) < 12.0≤ 0.10

= -0.9 < 12.0 ≤ 0.10

Karena nilai qk untuk N13 dan N8 bernilai benar maka diambil N8 sebagai node selanjutnya.

5) Rute selanjutnya

Rute yang didapat → N22 N23 N17 N16 N14 N12 N8. Gambar rute yang didapat dapat dilihat pada Gambar 5.13.




Gambar 5.13 Semut 2 untuk *edge* 6

6) Pembaruan feromon lokal

Pembaruan feromon lokal untuk *edge*s yang menghubungkan node N12 dan N8 dengan persamaan

$$\tau(i,s) \leftarrow (1-\rho).\tau(i,s) + \rho.\Delta\tau(i,s)$$

$$\Delta\tau(i,s) = \frac{1}{L_{nn}.C}$$

$$\Delta\tau\ (6,7) = \frac{1}{0.4 * 3} = \frac{1}{1.2} = 0.833$$

$$\rho.\Delta\tau(i,s) = 0.1 * 0.833 = 0.0833$$

$$\tau(6,7) \leftarrow ((1-0.1) * (\ 0.01)) + \ 0.0833$$

$$\tau(6,7) \leftarrow \big((0.9) * (0.01)\big) + (0.0833)$$

$$\tau(6,7) \leftarrow 0.009 + 0.0833 = 0.0923$$

Didapatkan N8 sebagai node selanjutnya pada perhitungan diatas. Kemudian akan dilanjutkan menghitung probabilitas pencarian node selanjutnya. Dalam pencarian probabilitas tersebut terdapat beberapa tahap. Tahap-tahap tersebut adalah:

1) Semut 2 dari N22 ke node selanjutnya

Semut 2 → N22 N23 N17 N16 N14 N12 N8

Titik awal = N8. Titik-titik yang terhubung dengan N8 adalah N5, N7, dan N11. Berikut merupakan tabel jarak antara node N8, N5, N7 dan N11. Nilai intensitas feromon = 0.01. Maka nilai $\eta_{ij}$ adalah

| **Node ke-** | **N8** | **N5** | **N7** | **N11** |
|---|---|---|---|---|
| **N8** | 0 | 0.3 | 0.041 | 1.2 |
| **N5** | 0.3 | 0 | 0 | 0 |
| **N7** | 0.041 | 0 | 0 | 0 |
| **N11** | 1.2 | 0 | 0 | 0 |

| **Node** | $\eta_{ij}$ |
|---|---|
| N5 | 3.33 |
| N7 | 24.39 |
| N11 | 5 |

2) Menghitung probabilitas dari node N8 ke node berikutnya

Probabilitas dari N8 ke setiap node berikutnya dapat dihitung menggunakan persamaan

$$P_{ij}^{k}=\frac{[\tau_{ij}]^{\alpha}.[\eta_{ij}]^{\beta}}{\Sigma_{k\in[N-tabu_k]}[\tau_{ik^i}]^{\alpha}.[\eta_{ik^j}]^{\beta}}$$

$\sum [\tau_{ik^i}]^{\alpha}.[\eta_{ik^j}]^{\beta}$ = (0.63096*0) + (0.63096*3.33) + (0.63096*24.3) + (0.63096*5)

= 0 + 2.1010+ 15.3322+ 3.1547

= 20.5879

Dengan demikian dapat dihitung probabilitas dari node N8 menuju ke tiap node adalah

Node N8 = 0.00

Node N5 = (0.63096*3.33) / 20.5879= 0.1020

Node N7 = (0.63096*24.3) / 20.5879= 0.7447

Node N11 = (0.63096*5)/ 20.5879= 0.1532

3) Probabilitas kumulatif

Probabilitas kumulatif nya yaitu:

Node N8 = 0.00

Node N5 = 0.1020

Node N7 = 0.8467

Node N11 = 0.9999

4) Bilangan acak dan Pengecekan

Bilangan acak nya = 50.123

Memeriksa qk-1 < r ≤ qk untuk:

qk = 0.1020, maka (0.1020– 1) < 50.123≤ 0.1020

= -0.898< 50.123≤ 0.1020

qk = 0.8467, maka (0.8467– 1) < 50.123≤ 0.8467

= -0.1533< 50.123≤ 0.8467

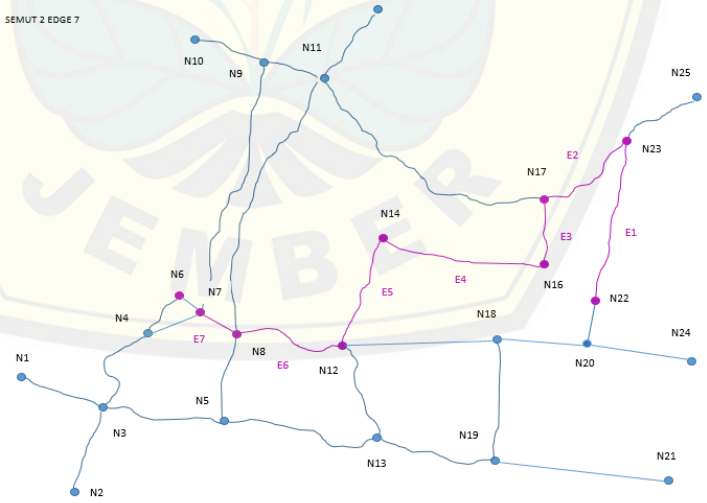
qk = 0.9999, maka (0.9999– 1) < 50.123≤ 0.9999

= -0.0001< 50.123≤ 0.9999

Karena nilai qk untuk N5 dan N7 bernilai benar maka diambil N7 sebagai node selanjutnya.

5) Rute selanjutnya

Rute yang didapat → N22 N23 N17 N16 N14 N12 N8 N7. Gambar rute yang didapat dapat dilihat pada Gambar 5.14.




Gambar 5.14 Semut 2 untuk *edge* 7

6) Pembaruan feromon lokal

Pembaruan feromon lokal untuk *edge*s yang menghubungkan node N8 dan N7 dengan persamaan

$$\tau(i,s) \leftarrow (1-\rho).\tau(i,s) + \rho.\Delta\tau(i,s)$$

$$\Delta\tau(i,s) = \frac{1}{L_{nn}.C}$$

$$\Delta\tau\ (7,8) = \frac{1}{0.041 * 3} = \frac{1}{0.123} = 8.13$$

$$\rho.\Delta\tau(i,s) = 0.1 * 8.13 = 0.813$$

$$\tau(7,8) \leftarrow ((1-0.1) * (\ 0.01)) + \ 0.813$$

$$\tau(7,8) \leftarrow \big((0.9) * (0.01)\big) + (0.813)$$

$$\tau(7,8) \leftarrow 0.009 + 0.813 = 0.822$$

Didapatkan N7 sebagai node selanjutnya pada perhitungan diatas. Kemudian akan dilanjutkan menghitung probabilitas pencarian node selanjutnya. Dalam pencarian probabilitas tersebut terdapat beberapa tahap. Tahap-tahap tersebut adalah:

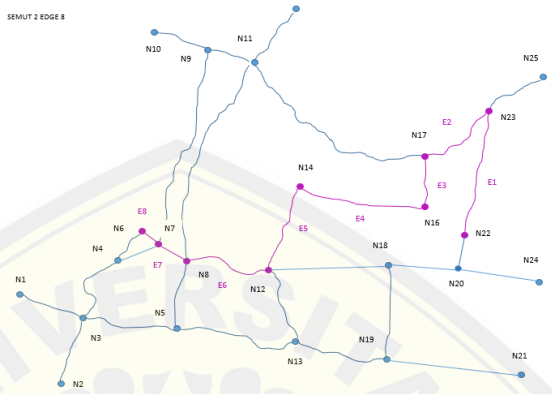
1) Probabilitas ke node selanjutnya

Semut 2 → N22 N23 N17 N16 N14 N12 N8 N7

Titik awal = N7. Titik-titik yang terhubung dengan N7 adalah N4 dan N6. Node N6 merupakan node tujuan yang dicari maka tidak perlu dicari nilai probabilitas nya. N6 merupakan node selanjutnya.

2) Rute selanjutnya

Rute yang didapat → N22 N23 N17 N16 N14 N12 N8 N7 N6. Gambar rute yang didapat dapat dilihat pada Gambar 5.15.




Gambar 5.15 Semut 2 untuk *edge* 8

3) Pembaruan feromon lokal

Pembaruan feromon lokal untuk *edge*s yang menghubungkan node N7 dan N6 dengan persamaan

$$\tau(i,s) \leftarrow (1-\rho).\tau(i,s) + \rho.\Delta\tau(i,s)$$

$$\Delta\tau(i,s) = \frac{1}{L_{nn}.C}$$

$$\Delta\tau\ (8,9) = \frac{1}{0.08 * 3} = \frac{1}{0.3} = 0.24$$

$$\rho.\Delta\tau(i,s) = 0.1 * 0.24 = 0.024$$

$$\tau(8,9) \leftarrow ((1-0.1) * (\ 0.01)) + \ 0.024$$

$$\tau(8,9) \leftarrow \big((0.9) * (0.01)\big) + (0.024)$$

$$\tau(8,9) \leftarrow 0.009 + 0.024 = 0.033$$

Dari perhitungan semut 2 diatas didapatkan rute N22 N23 N17 N16 N14 N12 N8 N7 N6. Dengan total jarak 3,000 Km. Perhitungan akan dilanjutkan menggunakan semut 3. Terdapat beberapa tahap untuk mencari probabilitas untuk menemukan rute dari N22 menuju N6.

**c. Semut 3**

Pada semut 3 ini terdapat pencarian untuk menuju ke node selanjutnya dari node awal yaitu node N22 ke node tujuan akhir yaitu node N6. Dalam pencarian probabilitas tersebut terdapat beberapa tahap. Tahap-tahap tersebut adalah:

1) Semut ke-3 dari N22 ke N6

Semut 3 → tujuan

Titik awal = N22. Titik-titik yang terhubung dengan N22 adalah N20 dan N23. Berikut merupakan tabel jarak antara node N22, N20, dan N23. Nilai intensitas feromon = 0.01

| **Node ke-** | **N22** | **N20** | **N23** |
|---|---|---|---|
| **N22** | 0 | 0.2 | 0.6 |
| **N20** | 0.2 | 0 | 0 |
| **N23** | 0.6 | 0 | 0 |

Maka nilai $\eta_{ij}$ adalah

| **Node** | **$\eta_{ij}$** |
|---|---|
| N22 | 5 |
| N23 | 1.67 |

2) Probabilitas ke node selanjutnya

Probabilitas dari N22 ke setiap node berikutnya dapat dihitung menggunakan persamaan

$$P_{ij}^{k}=\frac{[\tau_{ij}]^{\alpha}.[\eta_{ij}]^{\beta}}{\Sigma_{k\in[N-tabu_k]}[\tau_{ik^i}]^{\alpha}.[\eta_{ikj}]^{\beta}}$$

$\sum [\tau_{ik^i}]^{\alpha}.[\eta_{ikj}]^{\beta}$ = (0.63096*0) + (0.63096*5) + (0.63096*1.67)

= 0 + 3.1548+ 1.0537

= 4.2085

Dengan demikian dapat dihitung probabilitas dari node N22 menuju ke tiap node adalah

Node N22 = 0.00

Node N20 = (0.63096*5) / 4.2085= 0.7496

Node N23 = (0.63096*1.67)/ 4.2085= 0.2503

3) Probabilitas kumulatif

Probabilitas kumulatif nya yaitu:

Node N22 = 0.00

Node N20 = 0.7496

Node N23 = 0.9999

4) Bilangan acak dan Pengecekan

Bilangan acak nya = 79.000

Memeriksa qk-1 < r ≤ qk untuk:

qk = 0.7496, maka (0.7496– 1) < 52.00 ≤ 0.7496

= -0.2504< 52.00 ≤ 0.7496

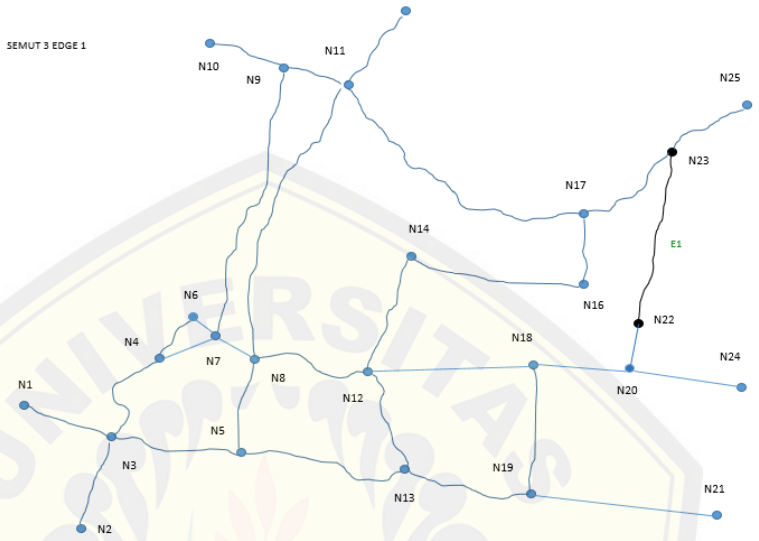
qk = 0.9999, maka (0.9999– 1) < 52.00 ≤ 0.9999

= -0.0001< 52.00 ≤ 0.9999

Karena nilai qk sama maka akan diambil acak node selanjutnya, yaitu N23 sebagai titik berikutnya.

5) Rute selanjutnya

Rute yang didapat → N22 N23. Gambar rute yang didapat dapat dilihat pada Gambar 5.16.




Gambar 5.16 Semut 3 untuk *edge* 1

6) Pembaruan feromon lokal

Pembaruan feromon lokal untuk *edge*s yang menghubungkan node N22 dan N23 dengan persamaan

$$\tau(i,s) \leftarrow (1-\rho).\tau(i,s) + \rho.\Delta\tau(i,s)$$

$$\Delta\tau(i,s) = \frac{1}{L_{nn}.C}$$

$$\Delta\tau\ (1,2) = \frac{1}{0.6 * 3} = \frac{1}{1.8} = 0.56$$

$$\rho.\Delta\tau(i,s) = 0.1 * 0.56 = 0.056$$

$$\tau(1,2) \leftarrow ((1-0.1) * (0.01)) + 0.056$$

$$\tau(1,2) \leftarrow \big((0.9) * (0.01)\big) + (0.056)$$

$$\tau(1,2) \leftarrow 0.009 + 0.056 = 0.065$$

Didapatkan N23 sebagai node selanjutnya pada perhitungan diatas. Kemudian akan dilanjutkan menghitung probabilitas pencarian node selanjutnya. Dalam pencarian probabilitas tersebut terdapat beberapa tahap. Tahap-tahap tersebut adalah:

1) Semut ke-3 dari N22 ke N6

Semut 3 → N22 N23

Titik awal = N23. Titik-titik yang terhubung dengan N23 adalah N25 dan N17. Berikut merupakan tabel jarak antara node N23, N25, dan N17. Nilai intensitas feromon = 0.01. Maka nilai $\eta_{ij}$ adalah

| **Node ke-** | **N23** | **N25** | **N17** |
|---|---|---|---|
| **N23** | 0 | 1.2 | 0.2 |
| **N25** | 1.2 | 0 | 0 |
| **N17** | 0.2 | 0 | 0 |

| **Node** | $\eta_{ij}$ |
|---|---|
| N25 | 0.833 |
| N17 | 5 |

2) Probabilitas dari N23 ke node selanjutnya

Probabilitas dari N23 ke setiap node berikutnya dapat dihitung menggunakan persamaan

$$P_{ij}^{k}=\frac{[\tau_{ij}]^{\alpha}.[\eta_{ij}]^{\beta}}{\sum_{k\in[N-tabu_k]}[\tau_{ik}i]^{\alpha}.[\eta_{ikj}]^{\beta}}$$

$\sum [\tau_{ik}i]^{\alpha}.[\eta_{ikj}]^{\beta}$ = (0.63096*0) + (0.63096*0.833) + (0.63096*5)

= 0 + 0.5255+ 3.1548

= 3.6803

Dengan demikian dapat dihitung probabilitas dari node N22 menuju ke tiap node adalah

Node N22 = 0.00

Node N25 = (0.63096*0.833) / 3.6803= 0.1427

Node N17 = (0.63096*5) / 3.6803= 0.8572

3) Probabilitas kumulatif

Probabilitas kumulatif nya yaitu:

Node N22 = 0.00

Node N25 = 0.1427

Node N17 = 0.9999

4) Bilangan acak dan Pengecekan

Bilangan acak nya = 79.000

Memeriksa qk-1 < r ≤ qk untuk:

qk = 0.1427, maka (0.1427– 1) < 79.000≤ 0.1427

= -0.8573< 79.000≤ 0.1427

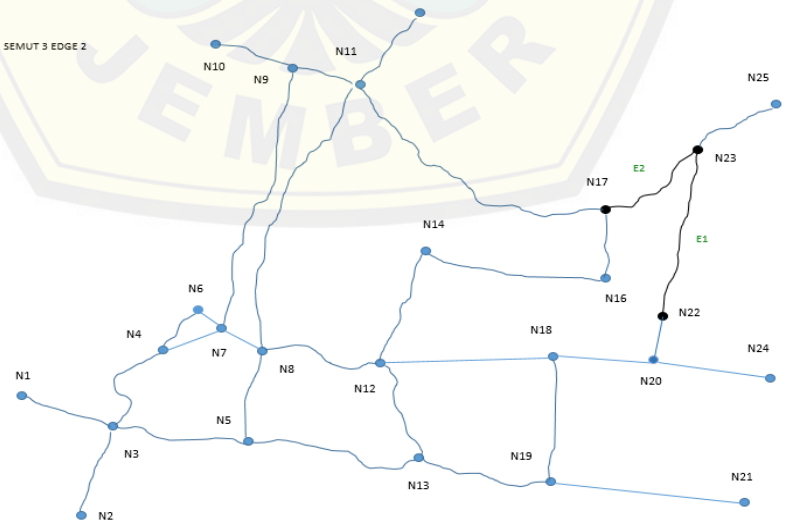
qk = 0.9999, maka (0.9999– 1) < 79.000≤ 0.9999

= -0.0001< 79.000≤ 0.9999

Karena nilai qk sama maka akan diambil acak node selanjutnya, yaitu N17 sebagai titik berikutnya.

5) Rute selanjutnya

Rute yang didapat → N22 N23 N17. Gambar rute yang didapat dapat dilihat pada Gambar 5.17.




Gambar 5.17 Semut 3 untuk *edge* 2

6) Pembaruan feromon lokal

Pembaruan feromon lokal untuk *edge*s yang menghubungkan node N23 dan N17 dengan persamaan

$$\tau(i,s) \leftarrow (1-\rho).\tau(i,s) + \rho.\Delta\tau(i,s)$$

$\Delta\tau(i,s) = \frac{1}{L_{nn}.C}$

$\Delta\tau$ (2,3) = $\frac{1}{0.2 * 3} = \frac{1}{0.6}$ = 1.67

$\rho.\Delta\tau(i,s)$ = 0.1 * 1.67 = 0.167

$\tau(2,3) \leftarrow ((1-0.1) * (0.01)) + 0.167$

$\tau(2,3) \leftarrow \big((0.9) * (0.01)\big) + (0.167)$

$\tau(2,3) \leftarrow 0.009 + 0.167 = 0.176$

Didapatkan N17 sebagai node selanjutnya pada perhitungan diatas. Kemudian akan dilanjutkan menghitung probabilitas pencarian node selanjutnya. Dalam pencarian probabilitas tersebut terdapat beberapa tahap. Tahap-tahap tersebut adalah:

1) Semut ke-3 dari N22 ke N6

Semut 3 → N22 N23 N17

Titik awal = N17. Titik – titik yang terhubung dengan N17 adalah N11 dan N16. Berikut merupakan tabel jarak antara node N17, N11, dan N16. Nilai intensitas feromonn = 0.01. Maka nilai $\eta_{ij}$ adalah

| **Node ke-** | **N17** | **N11** | **N16** |
|---|---|---|---|
| **N17** | 0 | 0.1 | 1.3 |
| **N11** | 0.1 | 0 | 0 |
| **N16** | 1.3 | 0 | 0 |

| **Node** | **$\eta_{ij}$** |
|---|---|
| N11 | 10 |
| N16 | 3.33 |

2) Probabilitas dari N17 ke node selanjutnya

Probabilitas dari N17 ke setiap node berikutnya dapat dihitung menggunakan persamaan

$$P_{ij}^{k}=\frac{[\tau_{ij}]^{\alpha}.[\eta_{ij}]^{\beta}}{\Sigma_{k\in[N-tabu_k]}[\tau_{ik}i]^{\alpha}.[\eta_{ik}j]^{\beta}}$$

$\sum [\tau_{ik}i]^{\alpha}.[\eta_{ik}j]^{\beta}$ = (0.63096*0) + (0.63096*10) + (0.63096*3.33)

= 0 + 6.3096+ 2.1010

= 8.4106

Dengan demikian dapat dihitung probabilitas dari node N17 menuju ke tiap node adalah

Node N17 = 0.00

Node N11 = (0.63096*10) / 8.4106= 0.75019

Node N16 = (0.63096*3.33) / 8.4106= 0.2498

3) Probabilitas kumulatif

Probabilitas kumulatif nya yaitu:

Node N17 = 0.00

Node N11 = 0.75019

Node N16 = 0.99999

4) Bilangan acak dan Pengecekan

Bilangan acak nya = 77.777

Memeriksa qk-1 < r ≤ qk untuk:

qk = 0.75019, maka (0.75019– 1) < 0.5 ≤ 0.75019

= -0.24981< 0.5 ≤ 0.75019

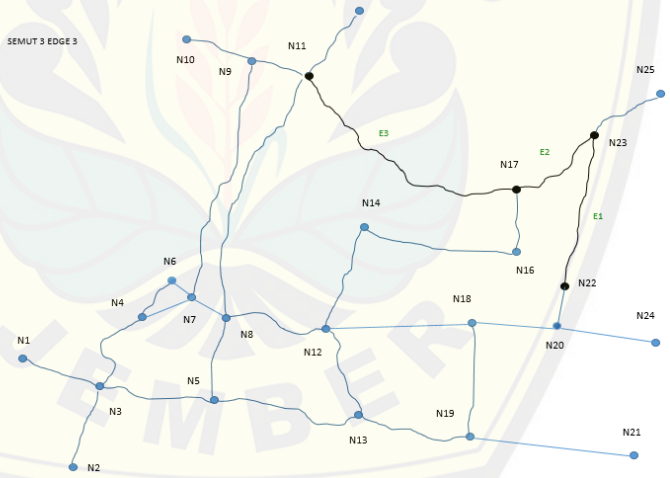
qk = 0.99999, maka (0.99999– 1) < 0.5 ≤ 0.99999

= -0.00001< 0.5 ≤ 0.99999

Karena nilai qk untuk 75.187 bernilai benar maka diambil N11 sebagai node selanjutnya.

5) Rute selanjutnya

Rute yang didapat → N22 N23 N17 N11. Gambar rute yang didapat dapat dilihat pada Gambar 5.18.




Gambar 5.18 Semut 3 untuk *edge* 3

6) Pembaruan feromon lokal

Pembaruan feromon lokal untuk *edge*s yang menghubungkan node N17 dan N11 dengan persamaan

$$\tau(i,s) \leftarrow (1-\rho).\tau(i,s) + \rho.\Delta\tau(i,s)$$

$$\Delta\tau(i,s) = \frac{1}{L_{nn}.C}$$

$$\Delta\tau\ (3,4) = \frac{1}{1.3 * 3} = \frac{1}{0.39} = 2.564$$

$$\rho.\Delta\tau(i,s) = 0.1 * 2.564 = 0.2564$$

$$\tau(3,4) \leftarrow ((1-0.1) * (0.01)) + 0.2564$$

$$\tau(3,4) \leftarrow \big((0.9) * (0.01)\big) + (0.2564)$$

$$\tau(3,4) \leftarrow 0.009 + 0.2564 = 0.2636$$

Didapatkan N11 sebagai node selanjutnya pada perhitungan diatas. Kemudian akan dilanjutkan menghitung probabilitas pencarian node selanjutnya. Dalam pencarian probabilitas tersebut terdapat beberapa tahap. Tahap-tahap tersebut adalah:

1) Semut ke-3 dari N22 ke N6

Semut 3 → N22 N23 N17 N11

Titik awal = N11. Titik-titik yang terhubung dengan N11 adalah N15, N8 dan N9. Berikut merupakan tabel jarak antara node N11, N15, N8 dan N9. Nilai intensitas feromon = 0.01. Maka nilai $\eta_{ij}$ adalah

| **Node ke-** | **N11** | **N15** | **N8** | **N9** |
|---|---|---|---|---|
| **N11** | 0 | 0.5 | 1.2 | 0.1 |
| **N15** | 0.5 | 0 | 0 | 0 |
| **N8** | 1.2 | 0 | 0 | 0 |
| **N9** | 0.1 | 0 | 0 | 0 |

| **Node** | **$\eta_{ij}$** |
|---|---|
| N15 | 2 |
| N8 | 0.83 |
| N9 | 10 |

2) Probabilitas dari N11 ke node selanjutnya

Probabilitas dari N11 ke setiap node berikutnya dapat dihitung menggunakan persamaan

$$P_{ij}^{k}=\frac{[\tau_{ij}]^{\alpha}.[\eta_{ij}]^{\beta}}{\Sigma_{k\in[N-tabu_k]}[\tau_{ik^i}]^{\alpha}.[\eta_{ik^j}]^{\beta}}$$

$\Sigma\ [\tau_{ik^i}]^{\alpha}.[\eta_{ik^j}]^{\beta}$ = (0.63096*0) + (0.63096*2) + (0.63096*0.83) + (0.63096*10)

= 0 + 1.2619 + 0.5236+ 6.3096

= 8.0951

Dengan demikian dapat dihitung probabilitas dari node N11 menuju ke tiap node adalah

Node N11 = 0.00

Node N15 = (0.63096*2) / 8.0951= 0.1558

Node N8 = (0.63096*0.83) / 8.0951= 0.0646

Node N9 = (0.63096*10)/ 8.0951= 0.779

3) Probabilitas kumulatif

Probabilitas kumulatif nya yaitu:

Node N11 = 0.00

Node N15 = 0.1558

Node N8 = 0.2204

Node N9 = 0.9994

4) Bilangan acak dan Pengecekan

Bilangan acak nya = 0.234

Memeriksa qk-1 < r ≤ qk untuk:

qk = 0.1558, maka (0.1558– 1) < 0.234 ≤ 0.1558

= -0.902< 0.234 ≤ 0.1558

qk = 0.2204, maka (0.2204– 1) < 0.234 ≤ 0.2204

= -0.862< 0.234 ≤ 0.2204

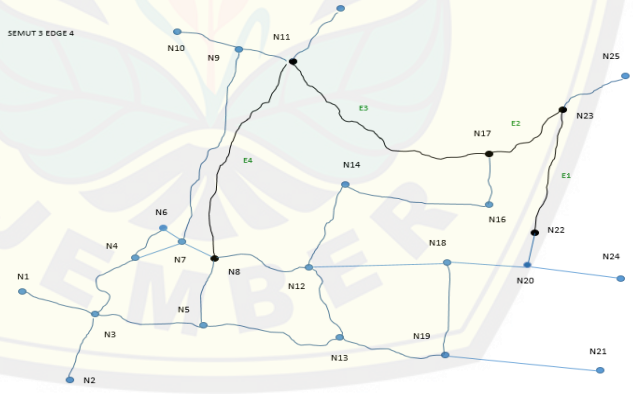
qk = 0.9994, maka (0.9994– 1) < 0.234 ≤ 0.9994

= -0.37 < 0.234 ≤ 0.9994

Karena nilai qk bernilai benar maka akan diambil node selanjutnya secara acak, yaitu N8 sebagai node selanjutnya.

5) Rute selanjutnya

Rute yang didapat → N22 N23 N17 N11 N8. Gambar rute yang didapat dapat dilihat pada Gambar 5.19.




Gambar 5.19 Semut 3 untuk *edge* 4

6) Pembaruan feromon lokal

Pembaruan feromon lokal untuk *edge*s yang menghubungkan node N11 dan N8 dengan persamaan

$$\tau(i,s) \leftarrow (1-\rho).\tau(i,s) + \rho.\Delta\tau(i,s)$$

$$\Delta\tau(i,s) = \frac{1}{L_{nn}.C}$$

$$\Delta\tau\ (4,5) = \frac{1}{1.2*3} = \frac{1}{3.6} = 0.277$$

$$\rho.\Delta\tau(i,s) = 0.1 * 0.277 = 0.0277$$

$$\tau(4,5) \leftarrow ((1-0.1) * (0.01)) + 0.0277$$

$$\tau(4,5) \leftarrow \big((0.9) * (0.01)\big) + (0.0277)$$

$$\tau(4,5) \leftarrow 0.009 + 0.0277 = 0.0367$$

Didapatkan N8 sebagai node selanjutnya pada perhitungan diatas. Kemudian akan dilanjutkan menghitung probabilitas pencarian node selanjutnya. Dalam pencarian probabilitas tersebut terdapat beberapa tahap. Tahap-tahap tersebut adalah:

1) Semut 3 dari N22 ke node selajutnya

Semut 3 → N22 N23 N17 N11 N8

Titik awal = N8. Titik-titik yang terhubung dengan N8 adalah N5, N7, dan N12. Berikut merupakan tabel jarak antara node N8, N5, N7 dan N12. Nilai intensitas feromon = 0.01. Maka nilai $\eta_{ij}$ adalah

| **Node ke-** | **N8** | **N5** | **N7** | **N12** |
|---|---|---|---|---|
| **N8** | 0 | 0.3 | 0.041 | 0.4 |
| **N5** | 0.3 | 0 | 0 | 0 |
| **N7** | 0.041 | 0 | 0 | 0 |
| **N12** | 0.4 | 0 | 0 | 0 |

| **Node** | $\eta_{ij}$ |
|---|---|
| N5 | 3.33 |
| N7 | 24.39 |
| N12 | 2.5 |

2) Menghitung probabilitas dari node N8 ke node berikutnya

Probabilitas dari N8 ke setiap node berikutnya dapat dihitung menggunakan persamaan

$$P_{ij}^{k}=\frac{[\tau_{ij}]^{\alpha}.[\eta_{ij}]^{\beta}}{\Sigma_{k\in[N-tabu_k]}[\tau_{ik^i}]^{\alpha}.[\eta_{ik^j}]^{\beta}}$$

$\sum [\tau_{ik^i}]^{\alpha}.[\eta_{ik^j}]^{\beta}$ = (0.63096*0) + (0.63096*3.33) + (0.63096*24.39) + (0.63096*2.5)

= 0 + 2.1010+ 15.3891+ 1.5774

= 19.0675

Dengan demikian dapat dihitung probabilitas dari node N8 menuju ke tiap node adalah

Node N8 = 0.00

Node N5 = (0.63096*3.33) / 19.0675= 0.1101

Node N7 = (0.63096*24.39) / 19.0675= 0.8070

Node N12 = (0.63096*2.5)/ 19.0675= 0.0827

3) Probabilitas kumulatif

Probabilitas kumulatif nya yaitu:

Node N8 = 0.00

Node N5 = 0.1101

Node N7 = 0.9171

Node N11 = 0.9998

4) Bilangan acak dan Pengecekan

Bilangan acak nya = 0.52

Memeriksa qk-1 < r ≤ qk untuk:

qk = 0.1101, maka (0.1101– 1) < 0.52≤ 0.1101

= -0.8899< 0.52≤ 0.1101

qk = 0.9171, maka (0.9171– 1) < 0.52≤ 0.9171

= -0.0829< 0.52≤ 0.9171

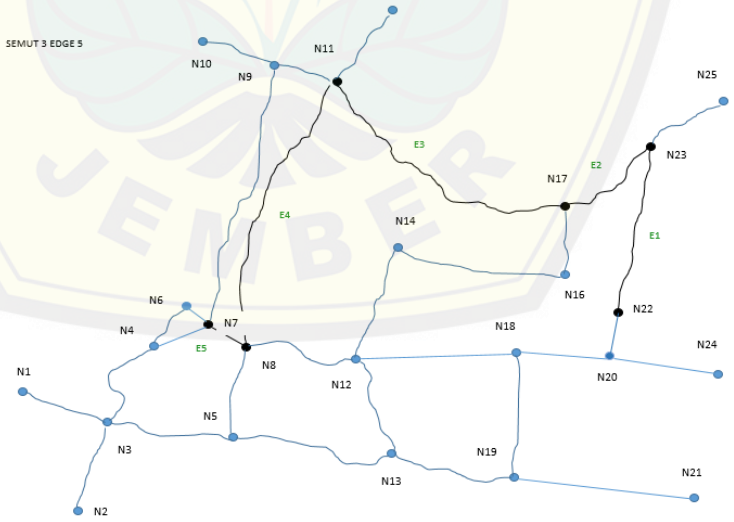
qk = 0.9998, maka (0.9998– 1) < 0.52≤ 0.9998

= -0.0002< 0.52≤ 0.9998

Karena nilai qk bernilai salah maka node selanjutnya akan diambil secara acak yaitu, diambil N7 sebagai node selanjutnya.

5) Rute selanjutnya

Rute yang didapat → N22 N23 N17 N11 N8 N7. Gambar rute yang didapat dapat dilihat pada Gambar 5.20.




Gambar 5.20 Semut 3 untuk *edge* 5

6) Pembaruan feromon lokal

Pembaruan feromon lokal untuk *edge*s yang menghubungkan node N8 dan N7 dengan persamaan

$$\tau(i,s) \leftarrow (1-\rho).\tau(i,s) + \rho.\Delta\tau(i,s)$$

$$\Delta\tau(i,s) = \frac{1}{L_{nn}.C}$$

$$\Delta\tau\ (5,6) = \frac{1}{0.041 * 3} = \frac{1}{0.123} = 8.13$$

$$\rho.\Delta\tau(i,s) = 0.1 * 8.13 = 0.813$$

$$\tau(5,6) \leftarrow ((1-0.1) * (\ 0.01)) + \ 0.813$$

$$\tau(5,6) \leftarrow \big((0.9) * (0.01)\big) + (0.813)$$

$$\tau(5,6) \leftarrow 0.009 + 0.813 = 0.822$$

Didapatkan N8 sebagai node selanjutnya pada perhitungan diatas. Kemudian akan dilanjutkan menghitung probabilitas pencarian node selanjutnya. Dalam pencarian probabilitas tersebut terdapat beberapa tahap. Tahap-tahap tersebut adalah:

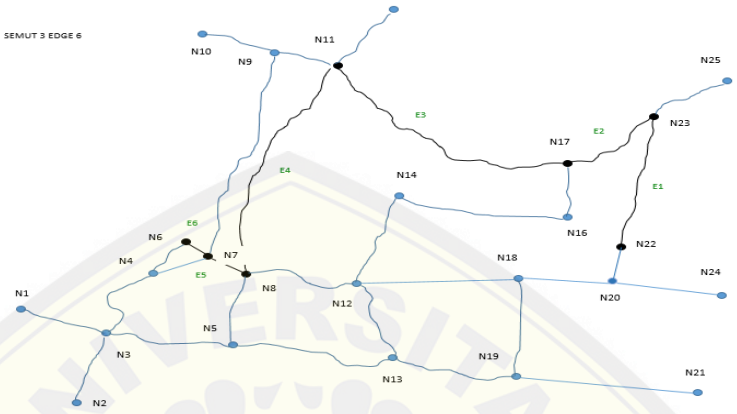
1) Probabilitas ke node selanjutnya

Semut 3 → N22 N23 N17 N11 N8 N7

Titik awal = N7. Titik-titik yang terhubung dengan N7 adalah N4 dan N6. Node N6 merupakan node tujuan yang dicari maka tidak perlu dicari nilai probabilitas nya. N6 merupakan node selanjutnya.

2) Rute selanjutnya

Rute yang didapat → N22 N23 N17 N11 N8 N7 N6. Gambar rute yang didapat dapat dilihat pada Gambar 5.16.




Gambar 5.21 Semut 3 untuk *edge* 6

3) Pembaruan feromon lokal

Pembaruan feromon lokal untuk *edge*s yang menghubungkan node N7 dan N6 dengan persamaan

$$\tau(i,s) \leftarrow (1-\rho).\tau(i,s) + \rho.\Delta\tau(i,s)$$

$$\Delta\tau(i,s) = \frac{1}{L_{nn}.C}$$

$$\Delta\tau\ (6,7) = \frac{1}{0.08 * 3} = \frac{1}{0.3} = 0.24$$

$$\rho.\Delta\tau(i,s) = 0.1 * 0.24 = 0.024$$

$$\tau(6,7) \leftarrow ((1-0.1) * (\ 0.01)) + \ 0.024$$

$$\tau(6,7) \leftarrow \big((0.9) * (0.01)\big) + (0.024)$$

$$\tau(6,7) \leftarrow 0.009 + 0.024 = 0.033$$

**d. Hasil rute semut pada siklus 1**

1) Hasil pencarian dari siklus 1

Hasil pencarian dan perhitungan dari siklus pertama dapat dilihat pada Tabel 5.3

Tabel 5.3 Hasil siklus 1 algoritma ACS

| Semut | Rute | Panjang |
|---|---|---|
| Semut 1 | N22 N20 N18 N12 N8 N7 N6 | 2.621 Km |
| Semut 2 | N22 N23 N17 N16 N14 N12 N8 N7 N6 | 3.000 Km |
| Semut 3 | N22 N23 N17 N11 N8 N7 N6 | 3.500 Km |

Dari Tabel 5.3 dapat dilihat bahwa rute terpendek dimiliki oleh semut 1 dengan total jarak 2,621 Km. Rute tersebut melewati node N22 N20 N18 N12 N8 N7 N6. Setelah diketahui bahwa semut 1 memiliki rute terpendek maka akan dihitung nilai pembaruan feromon global nya.

**2) Pembaruan Feromon Global**

Melakukan perhitungan feromon global menggunakan persamaan

$$\tau(i,j) \leftarrow (1-\alpha).\tau(i,j) + \alpha.\Delta\tau(i,j)$$

$\Delta\tau(i,j) = L_{gb}^{-1}$ jika (i,j) rute terbaik

Rute terbaik dari siklus 1 adalah pada semut 1 yaitu N22 N20 N18 N12 N8 N7 N6 dengan panjang $L_{gb}$ = 2.621 Km, sehingga diperoleh

$\Delta\tau(i,j) = \frac{1}{2.621}$ = 0.3815

$\tau(i,j) \leftarrow (1-\alpha).\tau(i,j) + \alpha.\Delta\tau(i,j)$

Feromon global Node N22 N20 untuk *Edge* 1

$\tau(1,2) \leftarrow (1-0.1) * (0.176) + (0.1) * (0.3815)$

$\tau(1,2) \leftarrow 0.19655$

Feromon global Node N20 N18 untuk *Edge* 2

$\tau(2,3) \leftarrow (1-0.1) * (0.076) + (0.1) * (0.3815)$

$\tau(2,3) \leftarrow 0.10655$

Feromon global Node N18 N12 untuk *Edge* 3

$\tau(3,4) \leftarrow (1-0.1) * (0.0328) + (0.1) * (0.3815)$

$\tau(3,4) \leftarrow 0.06767$

Feromon global Node N12 N8 untuk *Edge* 4

$\tau(4,5) \leftarrow (1-0.1) * (0.0923) + (0.1) * (0.3815)$

$\tau(4,5) \leftarrow 0.12122$

Feromon global Node N8 N7 untuk *Edge* 5

$\tau(5,6) \leftarrow (1-0.1) * (0.802) + (0.1) * (0.3815)$

$\tau(5,6) \leftarrow 0.75995$

Feromon global Node N7 N6 untuk *Edge* 6

$\tau(6,7) \leftarrow (1-0.1) * (0.033) + (0.1) * (0.3815)$

$\tau(6,7) \leftarrow 0.06785$

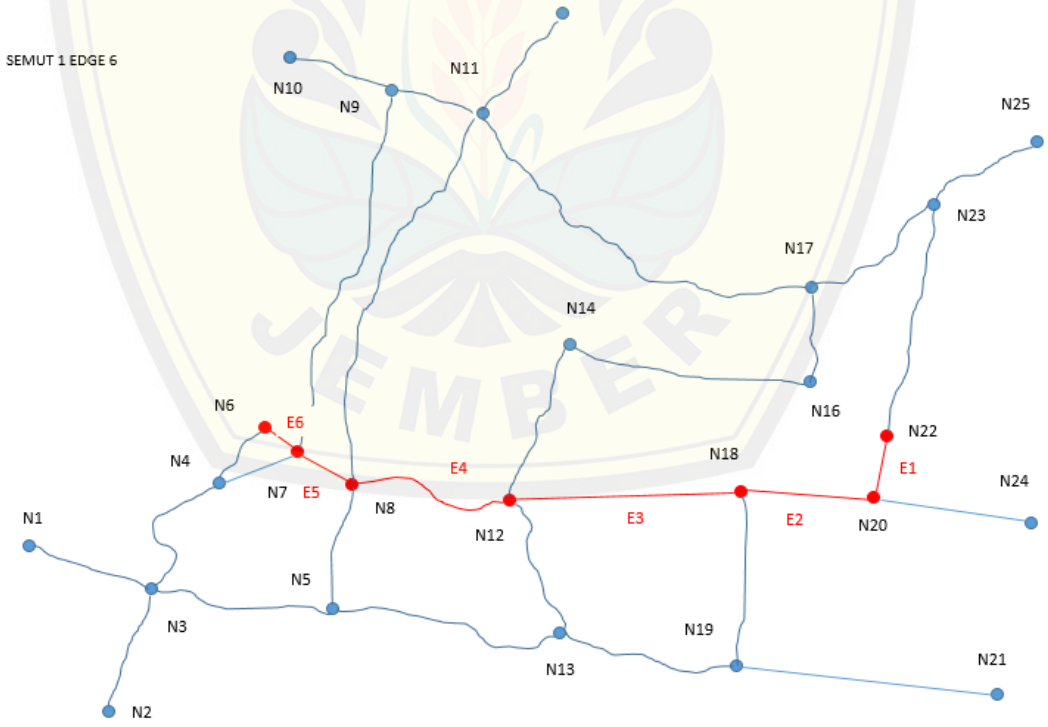
Intensitas feromon pada setiap titik setelah diperbarui atau $\tau(i,j)$ dari hasil siklus 1 akan ditunjukkan oleh Tabel 5.4

Tabel 5.4 Perubahan intensitas feromon per titik

| Semut 1 | $\tau(i,j)$ | $\Delta\tau(i,j)$ | Semut 2 | $\tau(i,j)$ | $\Delta\tau(i,j)$ | Semut 3 | $\tau(i,j)$ | $\Delta\tau(i,j)$ |
|---|---|---|---|---|---|---|---|---|
| E1 | 0.176 | 0.19655 | E1 | 0.065 | 0.0585 | E1 | 0.065 | 0.0585 |
| E2 | 0.076 | 0.10655 | E2 | 0.176 | 0.1584 | E2 | 0.176 | 0.1584 |
| E3 | 0.0328 | 0.06767 | E3 | 0.342 | 0.3078 | E3 | 0.2636 | 0.2372 |
| E4 | 0.0923 | 0.12122 | E4 | 0.0506 | 0.04554 | E4 | 0.0367 | 0.0330 |
| E5 | 0.802 | 0.75995 | E5 | 0.01376 | 0.012384 | E5 | 0.822 | 0.7398 |

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| E6 | 0.033 | 0.06785 | E6 | 0.0923 | 0.08307 | | | |
| | | | E7 | 0.822 | 0.7398 | | | |
| | | | E8 | 0.033 | 0.0297 | | | |

Dari Tabel 5.4 terlihat bahwa terjadi perubahan nilai feromon. Jalan yang sering dikunjungi semut akan terjadi peningkatan nilai feromon. Sedangkan pada jalan yang jarang dikunjungi semut terjadi pengurangan nilai feromon. Nilai feromon yang baru inilah yang akan digunakan pada perhitungan siklus berikutnya. Pada contoh perhitungan kita, banyak siklus yang kita inisialisasi adalah satu, maka perhitungan berhenti dari rute terpendek yang didapatkan adalah melalui node N22 N20 N18 N12 N8 N7 N6. Gambar 5.22 menunjukkan gambar peta dari rute yang dihasilkan oleh semut 1 dalam bentuk graf.




Gambar 5.22 Peta rute terpendek yang dihasilkan oleh semut 1 dalam bentuk graf

Berdasarkan data yang didapatkan selama penelitian di Dinas Kebersihan dan Pertamanan kota Surabaya dapat dihitung perbandingan jarak tempuh rute layanan armada kebersihan seperti pada Tabel 5.5

Tabel 5.5 Tabel Perbandingan Total Jarak

| No. | Lokasi TPS | Jarak Sebelumnya (Km) | Jarak Hasil Perhitungan ACO (Km) | Selisih |
|---|---|---|---|---|
| 1. | TPS Kalijudan | 54,694 | 42,85 | 11,844 |
| 2. | Super Depo Organik | 41,42 | 39,245 | 2,175 |
| 3. | Super Depo Residu | 62,46 | 54,57 | 7,83 |
| 4. | TPS Wisma Permai | 61,036 | 57,68 | 3,356 |
| 5. | TPS Dharmahusada | 60,84 | 54,88 | 5,96 |
| 6. | TPS Manyar Sabrangan | 67,31 | 60,21 | 7,10 |
| 7. | TPS Kejawan Putih Tambak | 66,662 | 58,52 | 8,142 |
| Total Jarak | | 414,422 | 367,955 | 46,467 |

Tabel 5.5 diatas menunjukkan bahwa jarak tempuh rancangan rute terpendek dengan menggunakan Ant Colony Optimization merupakan rancangan rute yang optimal. Penghematan jarak = (Total Jarak Awal – Total Jarak Hasil Perhitungan ACO) Km.

Penghematan jarak = (414,422 – 367,955) Km = 46,467 Km

Penghematan (%) = (46,467 / 414,422) * 100% = 11,212%

Dengan adanya penghematan total jarak sebesar 11,212% maka akan berdampak secara langsung pada waktu dan biaya yang diperlukan selama proses pengangkutan sampah. Sebagai contoh, waktu yang diperlukan selama proses pengangkutan pada TPS Kalijudan akan ditunjukkan oleh Tabel 5.6

Tabel 5.6 Perbandingan Waktu dan Biaya

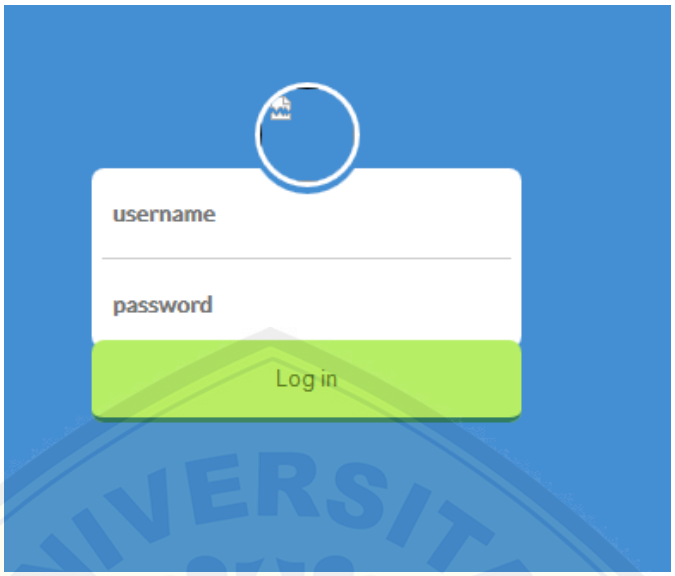
| TPS Kalijudan | Waktu Awal | Waktu Hasil Perhitungan ACO |
|---|---|---|
| Biaya | Rp 87.510 | Rp 68.560 |
| Waktu | 2 jam 24 menit | 1 jam 52 menit |

## 5.2 Implementasi Sistem

Implementasi sistem merupakan tahap pengkodean sistem berdasarkan pada perancangan yang telah dibuat ke dalam bahasa pemrograman. Penulis melakukan pengkodean menggunakan bahasa pemrograman PHP, HTML, CSS dan *Javascript*. Tahap pengkodean menghasilkan bebrapa tampilan atau *interface*. Berikut beberapa implementasi yang dibuat oleh penulis.

*a. Login*

Tampilan menu *login* ini digunakan untuk setiap *user* agar dapat mengakses fitur yang telah disediakan sesuai dengan hal akses masing-masing *user*. Sistem rute terpendek ini dapat diakses oleh admin dan *user*. Tampilan menu *login* dapat dilihat pada Gambar 5.23

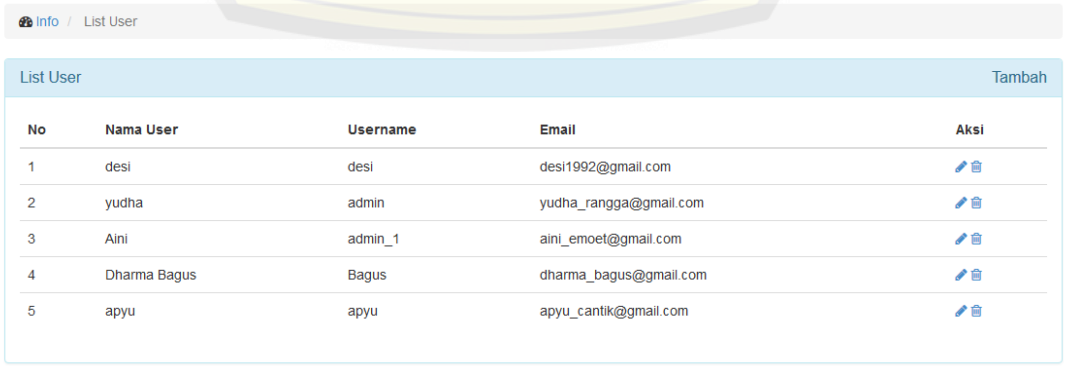



Gambar 5.23 Tampilan menu *Login*

Gambar 5.23 merupakan tampilan menu *login* pada sistem rute terpendek. Terdapat *form login* berisi *username* dan *password* yang dapat diisi dengan *username* dan *password* yang telah dimiliki *user* serta tombol *login* untuk melakukan proses *login*.

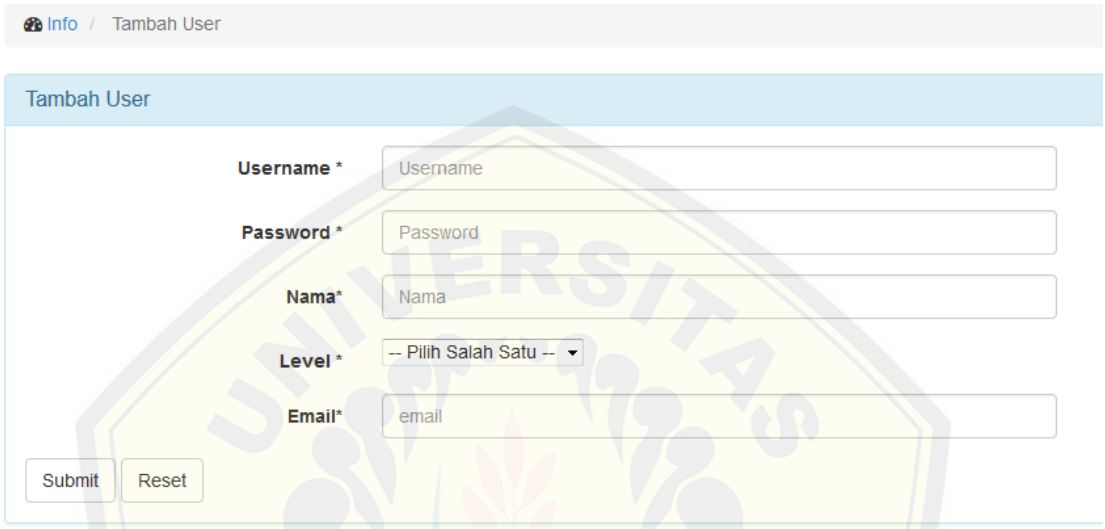
b. Data *User*

Tampilan menu data *user* ini hanya dapat diakses oleh admin untuk menambah, mengedit, serta menghapus data *user*. Tampilan menu data *user* dapat dilihat pada Gambar 5.24



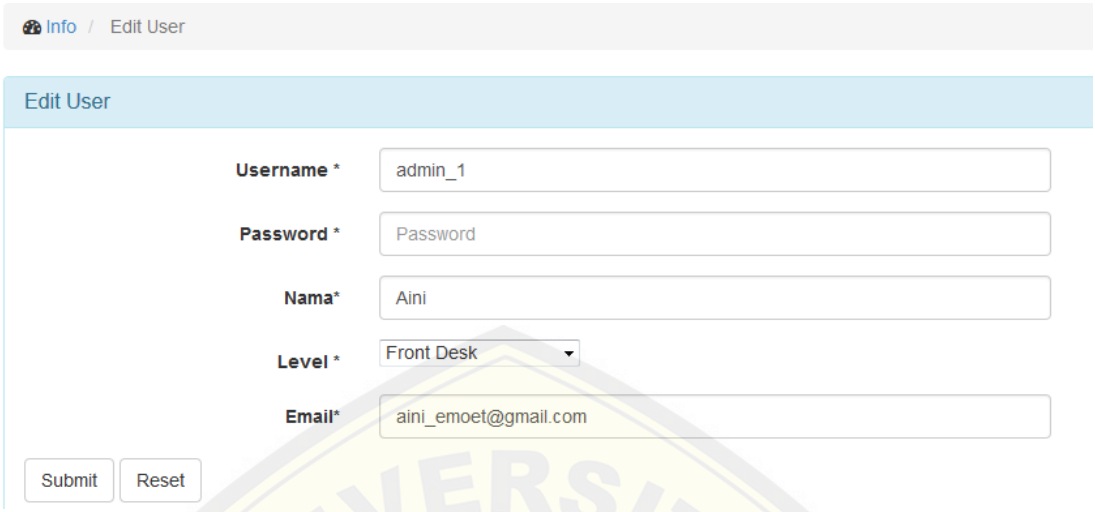

Gambar 5.24 Tampilan menu data *user*

Gambar 5.24 merupakan tampilan menu data *user* yang berisi tabel data user, tombol tambah data, edit, dan hapus. Untuk menambah data *user*, admin dapat menekan tombol tambah data dan kemudian sistem akan menampilkan *form* tambah data *user*. Tampilan *form* tambah data user dapat dilihat pada Gambar 5.25




Gambar 5.25 Tampilan *form* tambah data *user*

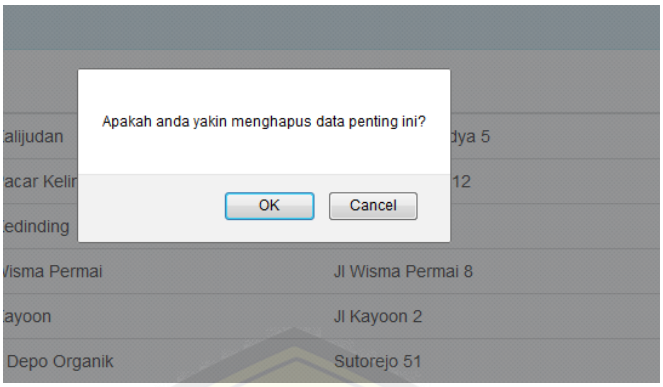
Gambar 5.25 merupakan tampilan *form* tambah data *user* yang berisi nama, *username*, *password*, *level user*, email serta tombol *submit* dan *reset*. Tombol *submit* untuk melakukan proses menambahkan data ke dalam *database*. Sedangkan tombol *reset* untuk membatalkan tambah data.Admin juga dapat mengedit data *user* dengan menekan tombol edit pada tabel data *user* yang ingin di edit datanya. Tampilan *form* edit data user dapat dilihat pada Gambar 5.26




Gambar 5.26 Tampilan *form* edit data *user*

Gambar 5.25 merupakan tampilan *form* edit data *user*. *User* admin hanya mengedit bagian yang ingin di *edit* kemudian klik *submit* agar sistem melakukan proses *update* pada *database*. Sedangkan tombol *reset* untuk membatalkan aktifitas edit data user.

Tombol hapus pada menu data *user* adalah tombol untuk menghapus data *user* yang ingin dihapus. Dengan menekan tombol hapus maka sistem akan menampilkan *alert* berisi "Apakah anda yakin ingin menghapus data penting ini?", jika admin ingin menghapus maka menekan tombol *ok* pada *alert* dan sistem akan menghapus data pada *database* dan mengembalikan ke tampilan menu data *user*. Gambar tampilan *alert* dapat dilihat pada Gambar 5.27



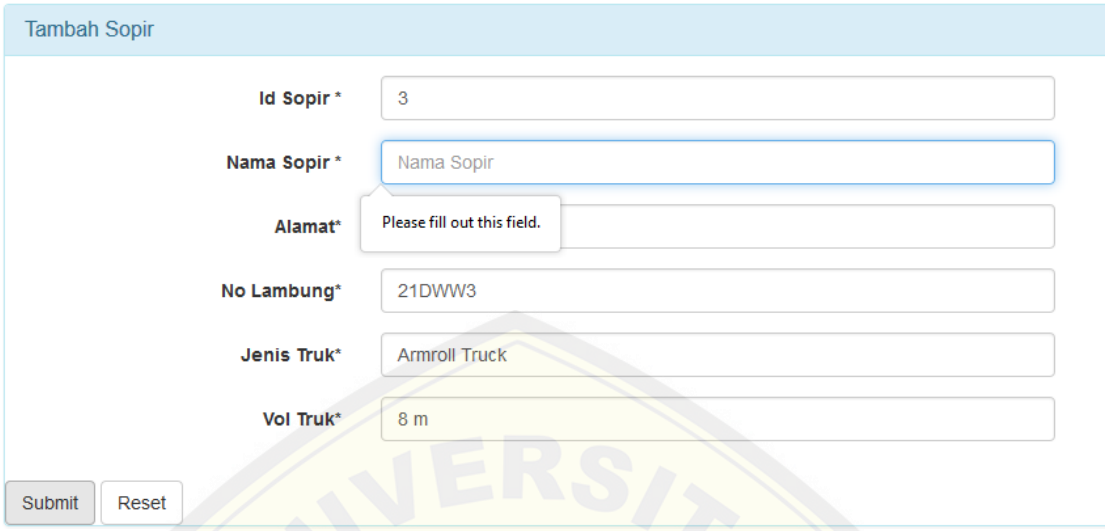

Gambar 5.27 Tampilan *alert* hapus data *user*

c. Data Sopir

Tampilan menu data sopir ini hanya dapat diakses oleh admin untuk menambah, mengedit, serta menghapus data sopir. Tampilan menu data sopir dapat dilihat pada Gambar 5.28

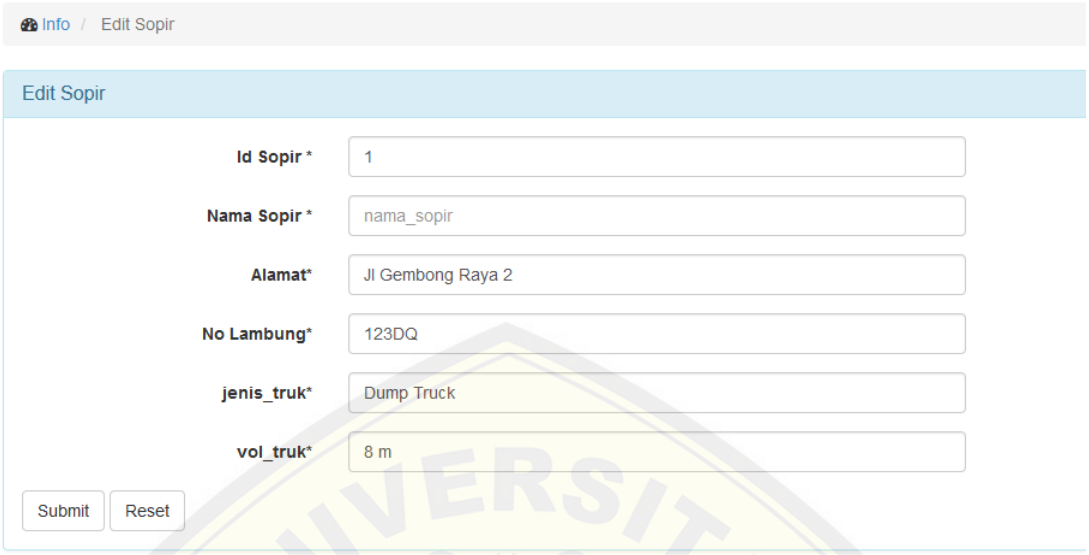

Gambar 5.28 Tampilan menu data sopir

Gambar 5.28 merupakan tampilan menu data sopir yang berisi tabel data sopir, tombol tambah data, edit, dan hapus. Untuk menambah data sopir, admin dapat menekan tombol tambah data dan kemudian sistem akan menampilkan *form* tambah data sopir. Tampilan *form* tambah data sopir dapat dilihat pada Gambar 5.29




Gambar 5.29 Tampilan form Tambah Sopir

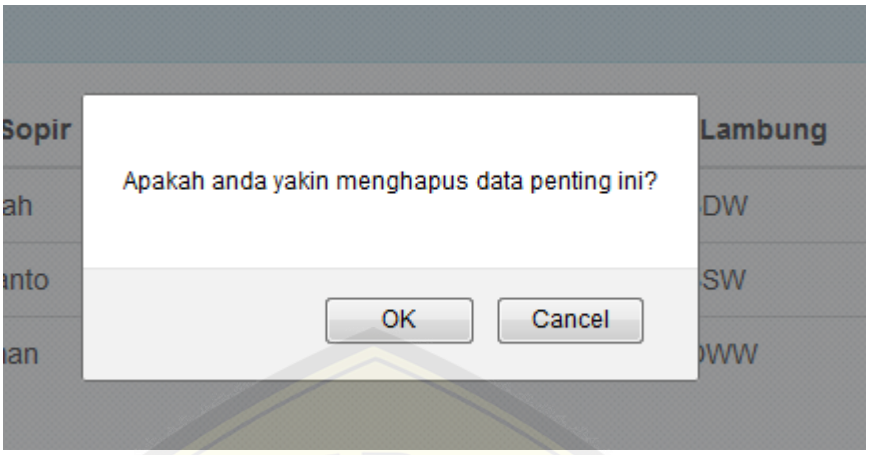
Gambar 5.29 merupakan tampilan *form* tambah data sopir yang berisi id sopir, nama sopir, alamat, no lambung, jenis truk, volume truk serta tombol *submit* dan *reset*. Tombol *submit* untuk melakukan proses menambahkan data ke dalam *database*. Sedangkan tombol *reset* untuk membatalkan tambah data. Admin juga dapat mengedit data sopir dengan menekan tombol edit pada tabel data sopir yang ingin di edit datanya. Tampilan *form* edit data sopir dapat dilihat pada Gambar 5.30




Gambar 5.30 Tampilan *form* edit sopir

Gambar 5.30 merupakan tampilan *form* edit data sopir. Admin hanya mengedit bagian yang ingin di *edit* kemudian klik *submit* agar sistem melakukan proses *update* pada *database*. Sedangkan tombol *reset* untuk membatalkan aktifitas edit data sopir.

Tombol hapus pada menu data *user* adalah tombol untuk menghapus data sopir yang ingin dihapus. Dengan menekan tombol hapus maka sistem akan menampilkan *alert* berisi "Apakah anda yakin ingin menghapus data penting ini?", jika admin ingin menghapus maka menekan tombol *ok* pada *alert* dan sistem akan menghapus data pada *database* dan mengembalikan ke tampilan menu data sopir. Gambar tampilan *alert* dapat dilihat pada Gambar 5.31




Gambar 5.31 Tampilan *Alert* Hapus Sopir

d. Data TPA

Tampilan menu data TPA ini hanya dapat diakses oleh admin untuk menambah, mengedit, serta menghapus data TPA. Tampilan menu data TPA dapat dilihat pada Gambar 5.32


| No | ID TPA | Nama | Alamat | Kecamatan | Aksi |
|---|---|---|---|---|---|
| 1 | 1 | TPS Kalijudan | Jl Kalijudan Madya 5 | Mulyorejo | |
| 2 | 2 | TPS Pacar Keling | Jl Pacar Keling 12 | Tambaksari | |
| 3 | 3 | TPS Kedinding | Jl Kedinding 46 | Kenjeran | |
| 4 | 4 | TPS Wisma Permai | Jl Wisma Permai 8 | Wisma Permai | |
| 5 | 5 | TPS Kayoon | Jl Kayoon 2 | Kayoon | |
| 6 | 6 | Super Depo Organik | Sutorejo 51 | Mulyorejo | |
| 7 | 7 | Rumah Kompos | | | |
| 8 | 8 | Super Depo Residu | Jl Sutorejo 51 | Mulyorejo | |
| 9 | 9 | TPS Lidah | Jl Lidah Kulon 42 | Lakar Santri | |
| 10 | 10 | TPS Dharmahusada | Jl Dharmahusada Indah 20 | Mulyorejo | |
| 11 | 11 | TPS Manyar Sabrangan | Jl Manyar 43 | Mulyorejo | |
| 12 | 12 | TPS Kalibokor | Jl Kalibokor 37 | | |

Gambar 5.32 Tampilan menu data TPA

Gambar 5.32 merupakan tampilan menu data TPA yang berisi tabel data TPA, tombol tambah data, edit, dan hapus. Untuk menambah data TPA, admin dapat menekan tombol tambah data dan kemudian sistem akan menampilkan *form* tambah data TPA. Tampilan *form* tambah data TPA dapat dilihat pada Gambar 5.33

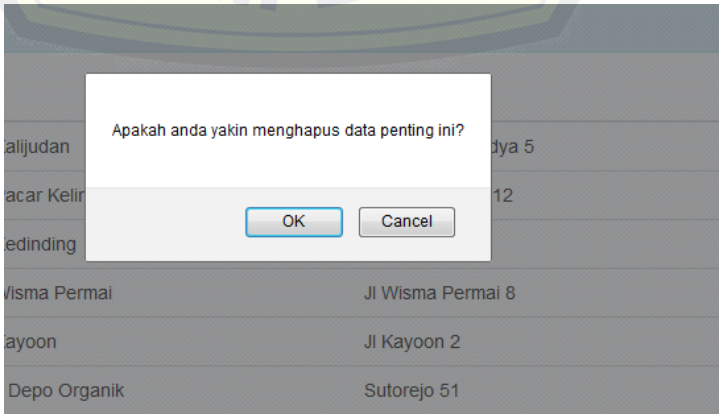

Gambar 5.33 Tampilan *form* tambah data TPA

Gambar 5.33 merupakan tampilan *form* tambah data TPA yang berisi id TPS, nama TPS, alamat, kecamatan, serta tombol *submit* dan *reset*. Tombol *submit* untuk melakukan proses menambahkan data ke dalam *database*. Sedangkan tombol *reset* untuk membatalkan tambah data.Admin juga dapat mengedit data TPA dengan menekan tombol edit pada tabel data TPA yang ingin di edit datanya. Tampilan *form* edit data TPA dapat dilihat pada Gambar 5.34


Gambar 5.34 Tampilan *form* edit data TPA

Gambar 5.34 merupakan tampilan *form* edit data *user*. Admin hanya mengedit bagian yang ingin di *edit* kemudian klik *submit* agar sistem melakukan proses update pada *database*. Sedangkan tombol *reset* untuk membatalkan aktifitas edit data TPA.

Tombol hapus pada menu data TPA adalah tombol untuk menghapus data TPA yang ingin dihapus. Dengan menekan tombol hapus maka sistem akan menampilkan *alert* berisi "Apakah anda yakin ingin menghapus data penting ini?", jika admin ingin menghapus maka menekan tombol *ok* pada *alert* dan sistem akan menghapus data pada *database* dan mengembalikan ke tampilan menu data TPA. Gambar tampilan *alert* dapat dilihat pada Gambar 5.35




Gambar 5.35 Tampilan *alert* hapus data TPA

e. Data Harga

Tampilan menu data harga bahan bakar ini hanya dapat diakses oleh admin untuk menambah, mengedit, serta menghapus data TPA. Tampilan menu data harga bahan bakar dapat dilihat pada Gambar 5.36


Gambar 5.36 Tampilan *Form* Data Harga Bahan Bakar

Gambar 5.36 merupakan tampilan menu data harga bahan bakar yang berisi tabel data harga bahan bakar, tombol tambah data, edit, dan hapus. Untuk menambah data harga bahan bakar, admin dapat menekan tombol tambah data dan kemudian sistem akan menampilkan *form* tambah data harga bahan bakar. Tampilan *form* tambah data harga bahan bakar dapat dilihat pada Gambar 5.37

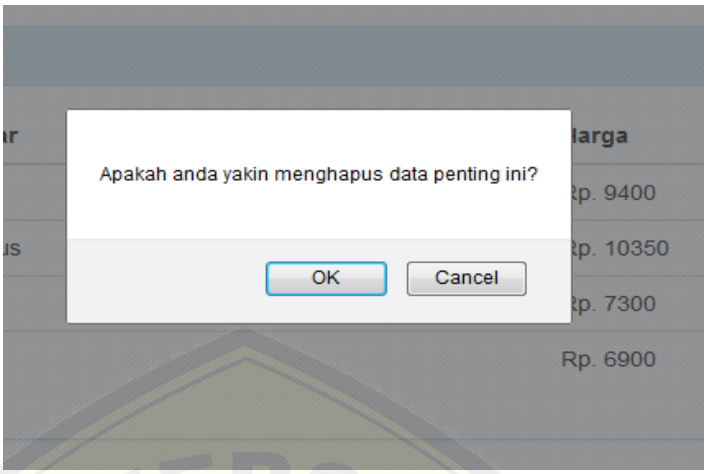

Gambar 5.37 Tampilan *Form* Tambah Bahan Bakar

Gambar 5.37 merupakan tampilan *form* tambah data TPA yang berisi id bahan bakar, bahan bakar, harga serta tombol *submit* dan *reset*. Tombol *submit* untuk melakukan proses menambahkan data ke dalam *database*. Sedangkan tombol *reset* untuk membatalkan tambah data. Admin juga dapat mengedit data harga bahan bakar dengan menekan tombol edit pada tabel data harga bahan bakar yang ingin di edit datanya. Tampilan *form* edit data harga bahan bakar dapat dilihat pada Gambar 5.38


Gambar 5.38 Tampilan *Form* Edit Data Harga Bahan Bakar

Gambar 5.38 merupakan tampilan *form* edit data harga bahan bakar. Admin hanya mengedit bagian yang ingin di *edit* kemudian klik *submit* agar sistem melakukan proses *update* pada *database*. Sedangkan tombol *reset* untuk membatalkan aktifitas edit data harga bahan bakar.

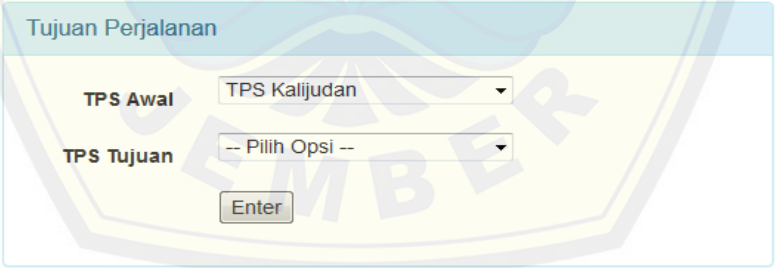
Tombol hapus pada menu data harga bahan bakar adalah tombol untuk menghapus data harga bahan bakar yang ingin dihapus. Dengan menekan tombol hapus maka sistem akan menampilkan *alert* berisi "Apakah anda yakin ingin menghapus data penting ini?", jka admin ingin menghapus maka menekan tombol *ok* pada *alert* dan sistem akan menghapus data pada *database* dan mengembalikan ke tampilan menu data harga bahan bakar. Gambar tampilan *alert* dapat dilihat pada Gambar 5.39




Gambar 5.39 Tampilan *alert* hapus data harga bahan bakar

f. Lihat Info Jalan

Tampilan menu info jalan ini dapat dilihat baik oleh admin maupun oleh *user*. Tampilan menu info jalan ini hanya dapat mencari rute terpendek dari TPS awal menuju ke TPS tujuan maupun ke TPA. Tampilan menu info jalan ini dapat menampilkan rute terpendek yang telah ditemukan, total jarak, biaya, serta peta perjalanan dan detail arah perjalanan yang harus dilewati. Tampilan menu info jalan dapat dilihat pada Gambar 5.40

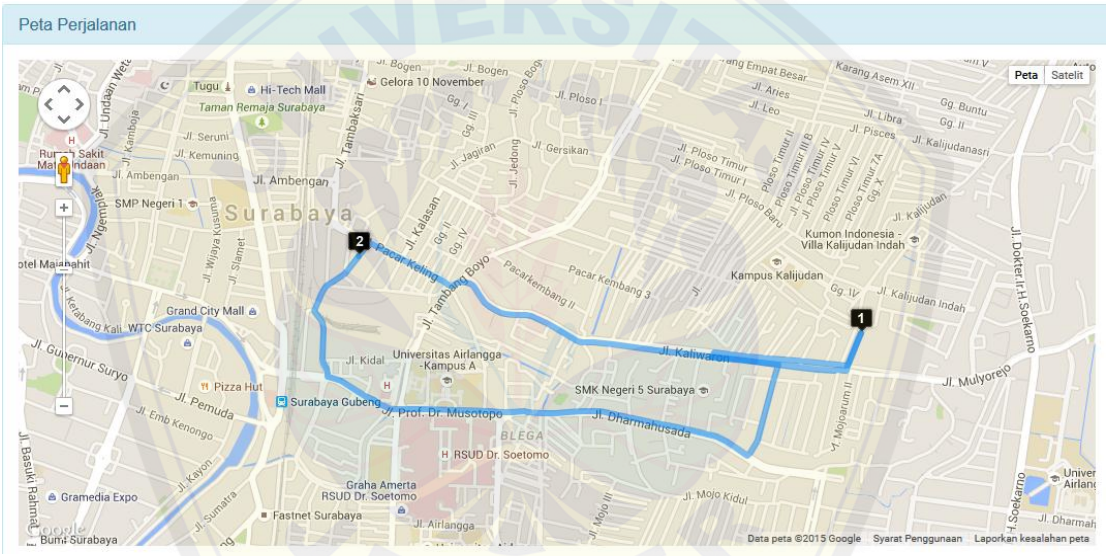



Gambar 5.40 Tampilan Menu Info Jalan *Input* titik

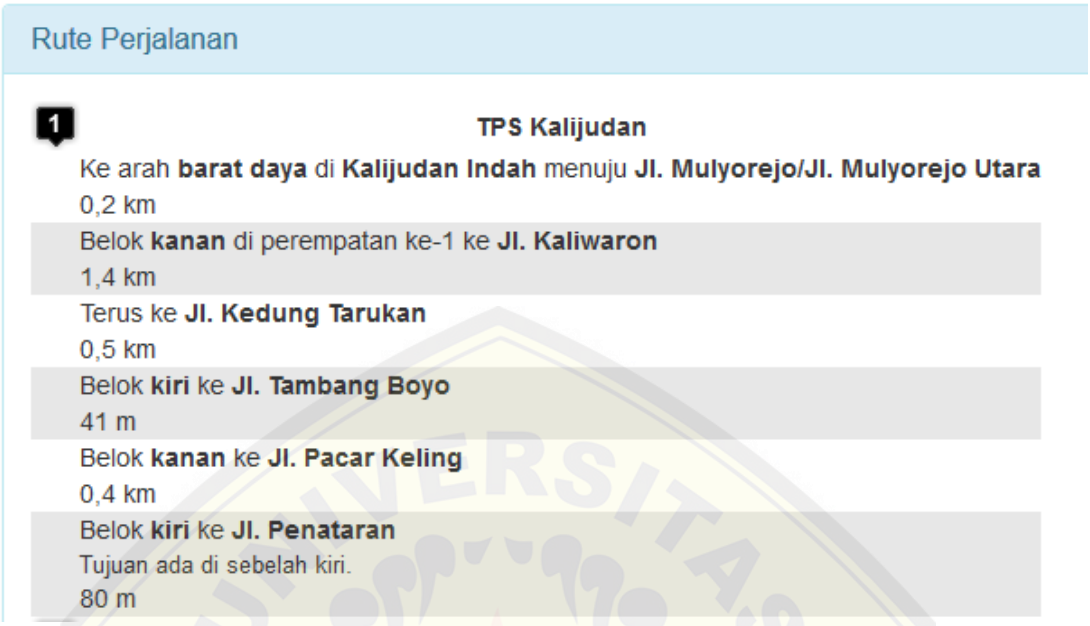
Gambar 5.40 merupakan tampilan menu info jalan untuk input titik awal dan titik tujuan. Setelah input dilakukan oleh *user* dan *user* telah menekan tombol *Enter*, maka akan dilakukan pencarian rute terpendek. Hasil dari pencarian berupa info waktu dan biaya akan ditunjukkan oleh Gambar 5.41


Gambar 5.41 Tampilan Hasil Info Waktu dan Biaya

Gambar 5.42 dan Gambar 5.43 merupakan tampilan peta perjalanan dan detail arah peta perjalanan.



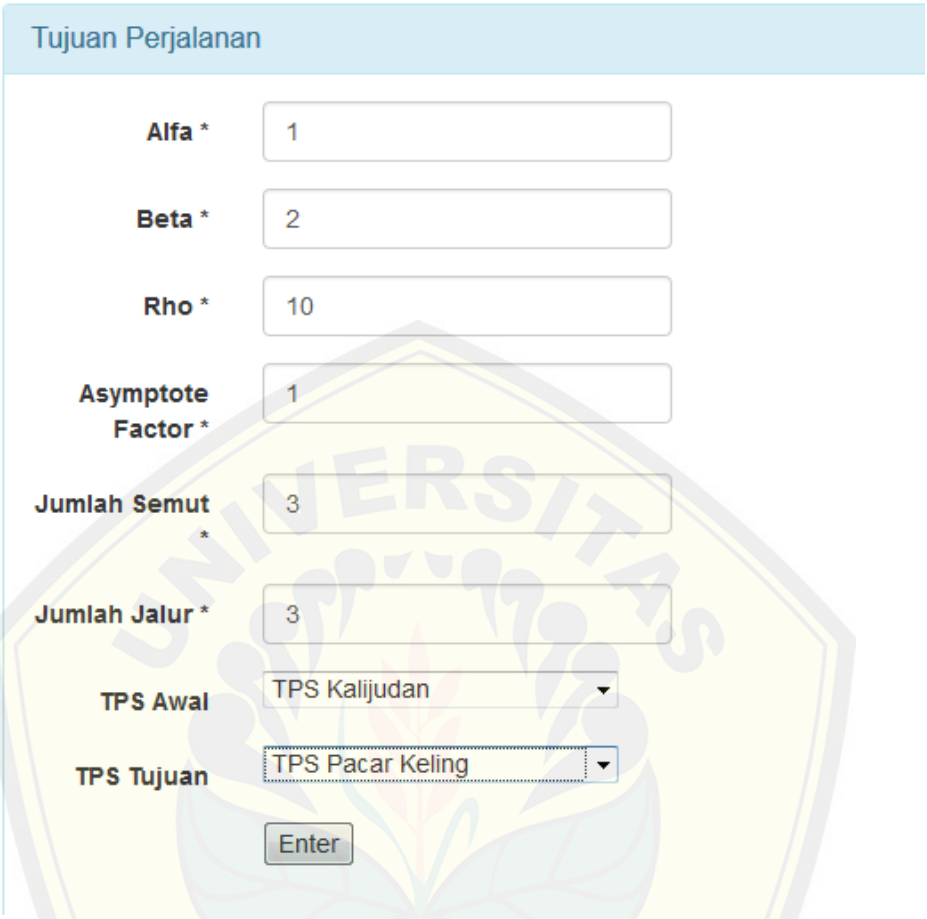

Gambar 5.42 Tampilan Peta Perjalanan




Gambar 5.43 Tampilan Detail Rute Perjalanan

g. Parameter Algoritma

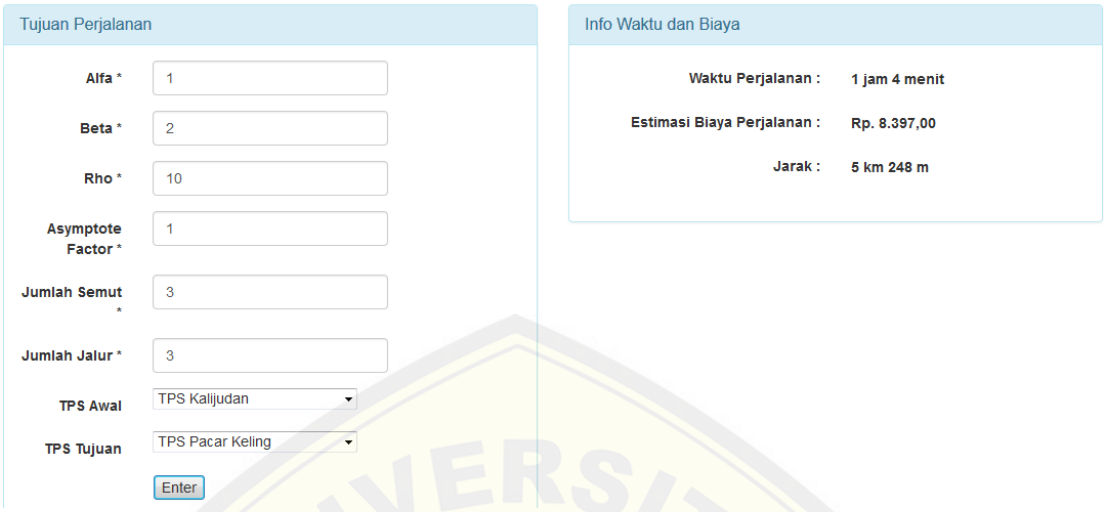
Tampilan menu parameter algoritma ini dapat dilihat baik oleh admin maupun oleh user. Tampilan menu parameter algoritma ini hanya dapat mencari rute terpendek dari TPS awal menuju ke TPS tujuan maupun ke TPA berdasarkan parameter yang telah diinputkan sebelumnya . Tampilan menu info jalan ini dapat menampilkan rute terpendek yang telah ditemukan, total jarak, biaya, serta peta perjalanan dan detail arah perjalanan yang harus dilewati. Tampilan menu parameter algoritma dapat dilihat pada Gambar 5.44




Gambar 5.44 Tampilan Menu Parameter Algoritma

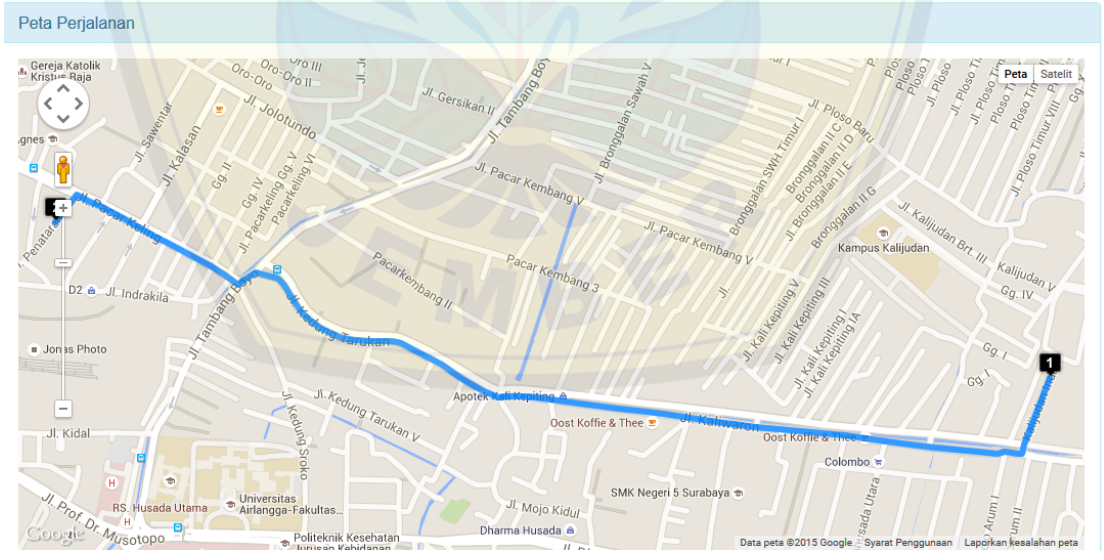
Gambar 5.44 merupakan tampilan menu parameter algoritma. Terdapat kolom *alfa, beta, rho, asymptote factor*, jumlah semut, jumlah jalur, TPS awal, TPS tujuan serta tombol *Enter*.

Setelah user menekan tombol *Enter* maka akan dicari rute terpendeknya. Tampilan hasil dari pencarian ini akan digambarkan oleh Gambar 5.45




Gambar 5.45 Tampilan Hasil Pencarian Rute Terpendek dengan Perbedaan Parameter

Sedangkan Gambar 5.46 dan 5.47 merupakan gambar peta perjalanan dan detail arah jalan sebagai hasil dari pencarian rute terpendek sebelumnya.




Gambar 5.46 Peta Perjalanan dengan Perbedaan Parameter


Gambar 5.47 Tampilan Detail Peta Perjalanan

## 5.3 Pengujian Sistem

Tahap selanjutnya adalah pengujian aplikasi yang telah dibuat. Tahap pengujian dimulai dari pengujian *whitebox* terlebih dahulu, kemudian akan dilanjutkan dengan pengujian *blackbox*. Langkah pertama yang dilakukan dalam pengujian *whitebox* adalah dengan membuat diagram alir dari *listing* yang diujikan.

### 5.3.1 Whitebox Testing

*Whitebox testing* merupakan pengujian program dengan melihat kode program. Pengujian whitebox dilakukan dengan metode *cyclomatic complexity* untuk mengetahui tingkat kompleksitas program. Berikut pengujian program untuk fitur info jalan.

**A. *Controller* c_info**

1. *Listing* Program

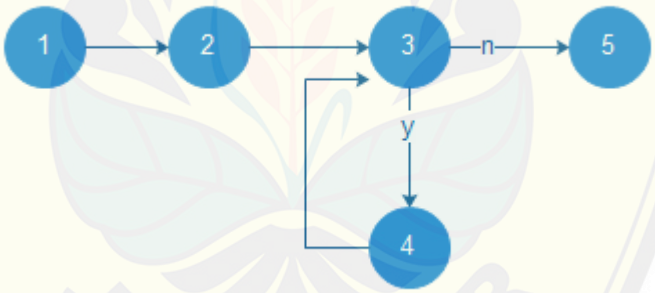
Gambar 5.48 merupakan potongan *listing* program *controller* c_info

```
function combo2() {
    $id = $_POST['combo'];
    $isi = $this->m_info->get_titik($id);
    $data = '<option value=""> -- Pilih Opsi -- </option>';
    for ($i = 0, $ln = count($isi); $i < $ln; $i++) {
        $data .= '<option value="' . $isi[$i]['latitude'] . ',' . $isi[$i]['longitude'] .
            '">' . $isi[$i]['nama'] . '</option>';
    }
    echo $data;
}
```

Gambar 5.48 Potongan Listing Program Fitur Info Jalan Bagian *Controller* c_info

2. Grafik Alir

Menggambarkan grafik alir controller fitur info jalan berdasarkan listing program diatas. Grafik alir bisa dilihat pada Gambar 5.49 dibawah ini.




Gambar 5.49 Grafik Alir *Controller* c_info fitur Info Jalan

3. Kompleksitas siklomatis dari grafik alir dapat diperoleh dengan perhitungan :

   V(G) = E – N + 2 = 5 – 5 + 2 = 2

4. *Basis set* yang dihasilkan dari jalur independen secara linier adalah dua jalur, yaitu :

   Jalur 1 : 1 – 2 – 3 – 4 – 3

   Jalur 2 : 1 – 2 – 3 – 5

5. *Test case, basis set* dicoba dan basis set sukses yang dihasilkan 1 – 2 – 3 – 4 – 3, dapat dilihat bahwa simpul telah dieksekusi satu kali. Berdasarkan ketentuan tersebut sistem telah memenuhi syarat kelayakan.

**B. Model m_info**

1. *Listing* Program

Gambar 5.50 merupakan potongan *listing* program model m_info

```
function get_titik($id = NULL) {
    $query = $this->db->query('SELECT a.id_TPA, a.nama, b.latitude, b.longitude FROM data_tpa a, data_koordinat b ' .
            (($id != NULL) ? (' WHERE a.id_TPA <> ' . $id . ' AND a.id_TPA = b.id_TPA') : 'WHERE a.id_TPA = b.id_TPA'));
    return $query->result_array();
}
```

1

2

Gambar 5.50 Potongan Listing Program Fitur Info Jalan Bagian Model m_info

*2.* Grafik Alir

Menggambarkan grafik alir model fitur info jalan berdasarkan listing program diatas. Grafik alir bisa dilihat pada Gambar 5.51 dibawah ini.


Gambar 5.51 Grafik Alir Model m_info fitur Info Jalan

3. Kompleksitas siklomatis dari grafik alir dapat diperoleh dengan perhitungan :

   V(G) = E – N + 2 = 1 – 2 + 2 = 1

4. *Basis set* yang dihasilkan dari jalur independen secara linier adalah satu jalur, yaitu :

   Jalur 1 : 1 – 2

5. *Test case, basis set* dicoba dan basis set sukses yang dihasilkan 1 – 2 dapat dilihat bahwa simpul telah dieksekusi satu kali. Berdasarkan ketentuan tersebut sistem telah memenuhi syarat kelayakan.

*Whitebox Testing* dilakukan pada setiap fitur. Penjelasan diatas merupakan *whitebox testing* untuk fitur info jalan, sedangkan fitur-fitur lainnya dapat dilihat pada Lampiran A.

### 5.3.2 BlackBox Testing

Proses pengujian yang selanjutnya adalah proses pengujian *blackbox*. Pada penelitian ini pengujian *blackbox* dapat dilihat pada lampiran B.

# BAB 6. PENUTUP

Bab ini merupakan bagian akhir dari penulisan skripsi yang berisi tentang kesimpulan dan saran. Kesimpulan berisi tentang hasil penelitian yang telah dilakukan dan saran untuk dilanjutkan pada penelitian yang selanjutnya.

## 6.1 Kesimpulan

Kesimpulan dari sistem yang dibangun pada penelitian ini adalah sebagai berikut:

1. Fitur info jalan dan parameter algoritma yang dibangun dengan menerapkan algoritma *Ant Colony Optimization* dapat menghasilkan rute terpendek (dibandingkan dengan rute awal) sehingga berdampak pada efiensi waktu dan biaya yang diperlukan selama proses pengangkutan sampah.
2. Pembangunan rute menggunakan *Ant Colony Optimzation* berdasarkan letak titik yang sudah ditentukan lebih optimal daripada rute yang selama ini dilalui oleh armada kebersihan.
3. Penghematan jarak yang diperoleh dari hasil pengolahan data menggunakan *Ant Colony Optimization* adalah 11.212% dari jarak tempuh rute yang digunakan saat ini.
4. Sistem optimasi rute terpendek dibangun sesuai kebutuhan dari objek penelitian dan dapat membantu kegiatan proses pengangkutan sampah di kota Surabaya.

## 6.2 Saran

Hasil yang dicapai dari penelitian ini belum sempurna, oleh karena itu untuk meningkatkan hasil yang dicapai beberapa saran untuk pengembangan penelitian ini diantaranya:

1. Pada penelitian ini, sistem optimasi rute terpendek ini belum memperhatikan data masukan titik koordinat untuk persimpangan jalan atau pengaturan jalan yang harus dilalui. Oleh karena itu, penulis menyarankan kepada pembaca yang tertarik

dengan masalah optimasi ini agar dapat menyertakan titik koordinat dari persimpangan jalan tersebut.

2. Diharapkan pada penelitian selanjutnya dapat membandingkan antar metode heuristik yang lainnya.
3. Perlu sebuah pengembangan yang dapat memuat masukan laporan kondisi jalan secara *up to date* sehingga menghasilkan informasi rute yang lebih baik.

# DAFTAR PUSTAKA

Dorigo,M., dan Stuzle,T. 2004, "*Ant Colony Optimization*". A Bradford book. The MIT Press Cambridge, Massachussets London, England.

Dorigo,M., dan Gambardella, L.M. 1997, "Ant Colony for the Travelling Salesman Problem". A Bradford book. The MIT Press Cambridge, Massachussets London, England.

Dorigo,M. 1996, "The Ant Colony Optimization Metaheuristic: Algorythms, Applications, and Advances". Universite Libre de Bruxelles.

Jogiyanto, HM. 2005. *Analisis & Disain Sistem* Informasi *: Pendekatan Terstruktur Teori dan Praktik Aplikasi Bisnis.* Yogyakarta : Andi

Materi Pelatihan Berbasis Kompetensi Bidang Persampahan (Balai Teknik Air Minum dan Sanitasi Wilayah 2, Wiyung – Surabaya,2010)

Mutakhiroh I., Saptono F., Hasanah N., dan Wiryadinata R. 2007. Pemnafaatn Metode Heuristik dalam Pencarian Jalur Terpendek dengan Algoritma Semut dan Algoritma Semut dan Algoritma Genetik. Seminar Nasional Aplikasi Tknologi Informasi. ISSN: 1997-5022. Yogyakarta

Peraturan Menteri Pekerjaan Umum (PERMEN PU) No: 21/PRT/M/2006

Peraturan Daerah Kota Surabaya (PERDA) Nomor 4 tahun 2000

Pranata R. A., Prasetyaningrum I., Fariza A., dan Martiana E. 2011. Perancangan Sistem Optimasi Rute Distribusi Pengangkutan Sampah di Surabaya Secara Adaptif Menggunakan Algoritma Koloni Semut. Jurusan Teknik Informatika, PENS-ITS. Surabaya

Sofwan Fauzi, Pencarian Jalur Terpendek Travelling Salesman problem Menggunakan Algoritma Ant Colony System, Fakultas Teknik dan Ilmu Komputer, Universitas Komputer Indonesia, 2010

Tchobanoglous, G, Theisen, H, Vigil, S.A. 1993. "Integrated Solid Waste Management". Mc. Graw Hill Publishing Company.

Wardy, I. S. 2007. Penggunaan Graph dalam Algoritma Semut untuk Melakukan Optimisasi. Program Studi Teknik Informatika, Institut Teknologi Bandung, Bandung.Siallagan, Sariadin. 2009. *Pemrograman Java Dasar-Dasar Pengenalan & Pemahaman.* Yogyakarta : Andi

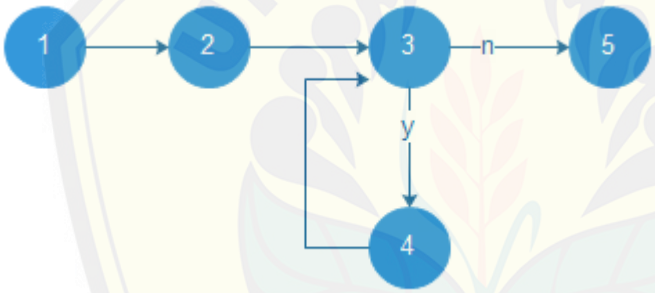
# LAMPIRAN

## A. Pengujian *White Box*

A. *Whitebox Testing* Parameter Algoritma

*Controller* c_algoritma

```
function combo2() {
    $id = $_POST['combo'];
    $isi = $this->m_info->get_titik($id);
    $data = '<option value=""> -- Pilih Opsi -- </option>';
    for ($i = 0, $ln = count($isi); $i < $ln; $i++) {
        $data .= '<option value="' . $isi[$i]['latitude'] . ',' .
            $isi[$i]['longitude'] . '">' . $isi[$i]['nama'] . '</option>';
    }
    echo $data;
}
```




V(G) = E – N + 2

V(G) = 5 – 5 + 2

V(G) = 2

*Basis set* menghasilkan dua jalur independen dari perhitungan kompleksitas siklomatik, yaitu :

Jalur 1 : 1 – 2 – 3 – 5

Jalur 2 : 1 – 2 – 3 – 4 – 3

*Test Case, basis set* dicoba dan basis set sukses yang dihasilkan 1 – 2 – 3 – 4 – 3, dapat dilihat bahwa simpul telah dieksekusi satu kali. Berdasarkan ketentuan tersebut sistem telah memenuhi syarat kelayakan.

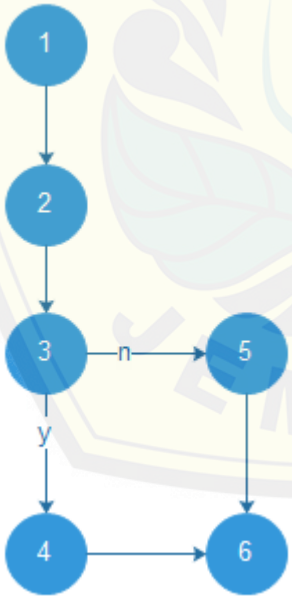
*B. Whitebox Testing User*

1) Tambah Data User

a. *Controller* Tambah Data *User*

```
function add() {
    $a = array
        (
        'username' => $_POST['username']
        , 'password' => md5($_POST['password'])
        , 'nama' => $_POST['nama']
        , 'email' => $_POST['email']
        , 'level' => $_POST['level']
    );

    if ($this->m_user->tambah_user($a)) {
        $this->session->set_userdata('pesan_sistem', 'Selamat! Penambahan user baru, SUKSES!');
        $this->session->set_userdata('tipe_pesan', 'Sukses');
        redirect('user');
    } else {
        $this->session->set_userdata('pesan_sistem', 'Maaf! Penambahan user baru,
            GAGAL! Silahkan periksa dan coba kembali');
        $this->session->set_userdata('tipe_pesan', 'Gagal');
        redirect('user');
    }
}
```




V(G) = E – N + 2

V(G) = 6 – 6 + 2

V(G) = 2

Basis set menghasilkan dua jalur independen dari perhitungan kompleksitas siklomatik, yaitu :

Jalur 1 : 1 – 2 – 3 – 5 – 6

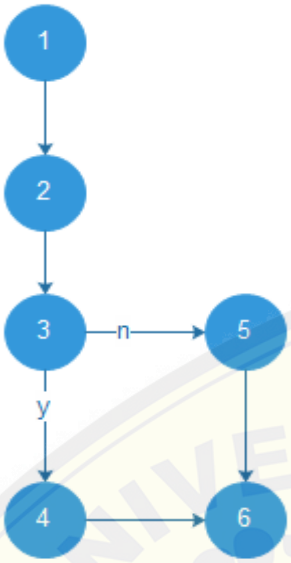
Jalur 2 : 1 – 2 – 3 – 4 – 6

Test Case, basis set dicoba dan basis set sukses yang dihasilkan 1 – 2 – 3 – 4 – 6, dapat dilihat bahwa simpul telah dieksekusi satu kali. Berdasarkan ketentuan tersebut sistem telah memenuhi syarat kelayakan.

b. Model Tambah Data User

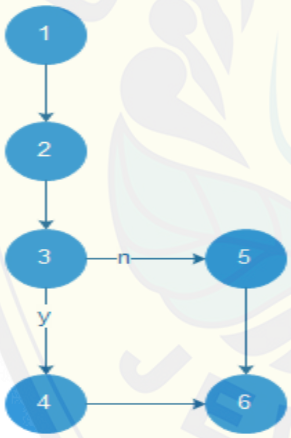
```
function tambah_user($a) {  1
   2 $query = $this->db->query
            ('
                    INSERT INTO user VALUES
                    (
                        ' . "'" . "'" . '
                        , ' . "'" . $a['level'] . "'" . '
                        , ' . "'" . $a['username'] . "'" . '
                        , ' . "'" . $a['nama'] . "'" . '
                        , ' . "'" . $a['password'] . "'" . '
                        , ' . "'" . $a['email'] . "'" . '
                        , 1
                    )
            ');

   3 if ($query) {
        return TRUE;  4
     } else {
        return FALSE;  5
     }
   6 }
}
```

<table>
<tr><td> 
</td><td>V(G) = E – N + 2<br>V(G) = 6 – 6 + 2<br>V(G) = 2</td></tr>
<tr><td colspan="2">Basis set menghasilkan dua jalur independen dari perhitungan kompleksitas siklomatik, yaitu :<br>Jalur 1 : 1 – 2 – 3 – 5 – 6<br>Jalur 2 : 1 – 2 – 3 – 4 – 6</td></tr>
<tr><td colspan="2">Test Case, basis set dicoba dan basis set sukses yang dihasilkan 1 – 2 – 3 – 4 – 6, dapat dilihat bahwa simpul telah dieksekusi satu kali. Berdasarkan ketentuan tersebut sistem telah memenuhi syarat kelayakan.</td></tr>
<tr><td colspan="2">c. Controller Edit User</td></tr>
</table>

```
function edit() {
    $a = array                                    1
 2      (
        'id' => $_POST['id']
        , 'username' => $_POST['username']
        , 'password' => md5($_POST['password'])
        , 'nama' => $_POST['nama']
        , 'email' => $_POST['email']
        , 'level' => $_POST['level']
    );

 3  if ($this->m_user->user_edit($a)) {
        $this->session->set_userdata('pesan_sistem', 'Selamat! edit user baru, SUKSES!');
  4     $this->session->set_userdata('tipe_pesan', 'Sukses');
        redirect('user');
    } else {
        $this->session->set_userdata('pesan_sistem', 'Maaf! edit user baru, GAGAL!
 5          Silahkan periksa dan coba kembali');
        $this->session->set_userdata('tipe_pesan', 'Gagal');
        redirect('user');
 6  }
}
```




V(G) = E – N + 2

V(G) = 6 – 6 + 2

V(G) = 2

Basis set menghasilkan dua jalur independen dari perhitungan kompleksitas siklomatik, yaitu :

Jalur 1 : 1 – 2 – 3 – 5 – 6

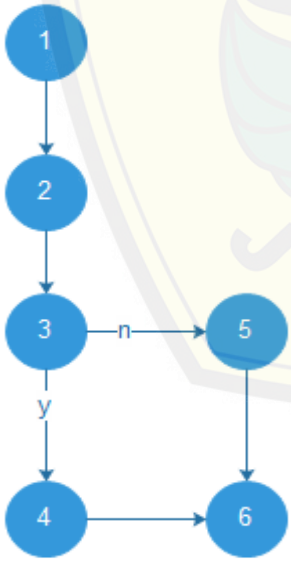
Jalur 2 : 1 – 2 – 3 – 4 – 6

Test Case, basis set dicoba dan basis set sukses yang dihasilkan 1 – 2 – 3 – 4 – 6, dapat dilihat bahwa simpul telah dieksekusi satu kali. Berdasarkan ketentuan tersebut sistem telah memenuhi syarat kelayakan.

d. Model Edit Data User

```
function user_edit($a) {  1
    $query = $this->db->query
             ('
                    UPDATE user SET
                        level = ' . "'" . $a['level'] . "'" . '
                        , username = ' . "'" . $a['username'] . "'" . '
                        , nama = ' . "'" . $a['nama'] . "'" . '
                        , password = ' . "'" . $a['password'] . "'" . '
                        , email = ' . "'" . $a['email'] . "'" . '
                        , status_user = ' . "'" . $a['status_user'] . "'" . '
                    WHERE
                        id = ' . $a['id']
             );

  3 if ($query) {
        return TRUE;  4
    } else {  5
        return FALSE;
    }
}  6
```




V(G) = E – N + 2

V(G) = 6 – 6 + 2

V(G) = 2

Basis set menghasilkan dua jalur independen dari perhitungan kompleksitas siklomatik, yaitu :

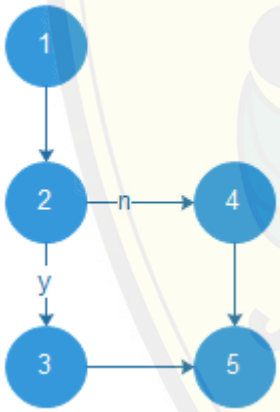
Jalur 1 : 1 – 2 – 3 – 5 – 6

Jalur 2 : 1 – 2 – 3 – 4 – 6

Test Case, basis set dicoba dan basis set sukses yang dihasilkan 1 – 2 – 3 – 4 – 6, dapat dilihat bahwa simpul telah dieksekusi satu kali. Berdasarkan ketentuan tersebut sistem telah memenuhi syarat kelayakan.

e. Controller Hapus Data User

```
function delete($a) {
    if ($this->m_user->hapus_user($a)) {
        $this->session->set_userdata('pesan_sistem', 'Selamat! User telah dihapus!');
        $this->session->set_userdata('tipe_pesan', 'Sukses');
        redirect('user');
    } else {
        $this->session->set_userdata('pesan_sistem', 'Maaf! User tidak terhapus! Silahkan periksa dan coba kembali');
        $this->session->set_userdata('tipe_pesan', 'Gagal');
        redirect('user');
    }
}
```




V(G) = E – N + 2

V(G) = 5 – 5 + 2

V(G) = 2

Basis set menghasilkan dua jalur independen dari perhitungan kompleksitas siklomatik, yaitu :

Jalur 1 : 1 – 2 – 3 – 5

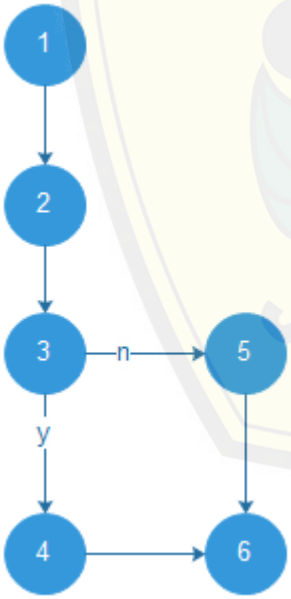
Jalur 2 : 1 – 2 – 4 – 5

Test Case, basis set dicoba dan basis set sukses yang dihasilkan 1 – 2 – 3 – 5, dapat dilihat bahwa simpul telah dieksekusi satu kali. Berdasarkan ketentuan tersebut sistem telah memenuhi syarat kelayakan.

f. Model Hapus Data User

```
1 function hapus_user($a) {
      $query = $this->db->query   2
               ('
                        UPDATE user SET
                            status_user = 0
                        WHERE
                            id = ' . $a
               );

3     if ($query) {
          return TRUE;   4
5     } else {
          return FALSE;
      }
  }
  6
```




V(G) = E – N + 2

V(G) = 6 – 6 + 2

V(G) = 2

Basis set menghasilkan dua jalur independen dari perhitungan kompleksitas siklomatik, yaitu :

Jalur 1 : 1 – 2 – 3 – 5 – 6

Jalur 2 : 1 – 2 – 3 – 4 – 6

Test Case, basis set dicoba dan basis set sukses yang dihasilkan 1 – 2 – 3 – 4 – 6, dapat dilihat bahwa simpul telah dieksekusi satu kali. Berdasarkan ketentuan tersebut sistem telah memenuhi syarat kelayakan.

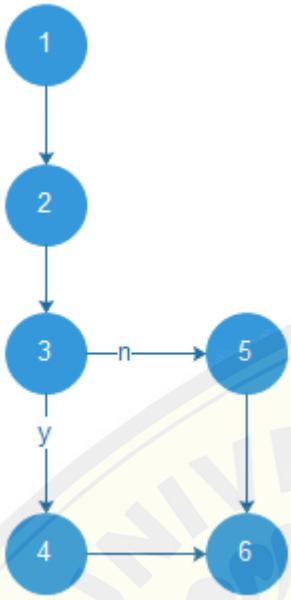
*C. Whitebox Testing* Sopir

1) Tambah Data Sopir

a. *Controller* Tambah Data Sopir

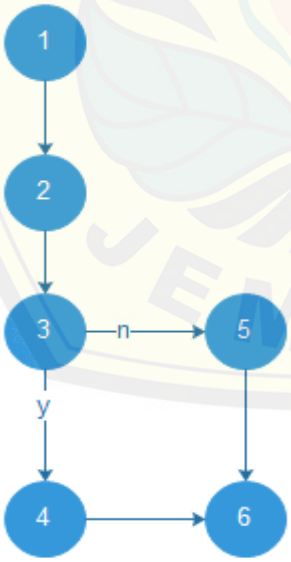
```
function add() {
    $a = array
        (
        'id_sopir' => $_POST['id_sopir']
        , 'nama_sopir' => $_POST['nama_sopir']
        , 'alamat' => $_POST['alamat']
        , 'no_lambung' => $_POST ['no_lambung']
        , 'jenis_truk' => $_POST ['jenis_truk']
        , 'vol_truk' => $_POST ['vol_truk']
    );

    if ($this->m_sopir->tambah_sopir($a)) {
        $this->session->set_userdata('pesan_sistem', 'Selamat! Penambahan sopir baru, SUKSES!');
        $this->session->set_userdata('tipe_pesan', 'Sukses');
        redirect('c_sopir');
    } else {
        $this->session->set_userdata('pesan_sistem', 'Maaf! Penambahan jalan baru, GAGAL!
            Silahkan periksa dan coba kembali');
        $this->session->set_userdata('tipe_pesan', 'Gagal');
        redirect('c_sopir');
    }
}
```

<table>
<tr>
<td> 
</td>
<td>V(G) = E – N + 2<br>V(G) = 6 – 6 + 2<br>V(G) = 2</td>
</tr>
<tr>
<td colspan="2">Basis set menghasilkan dua jalur independen dari perhitungan kompleksitas siklomatik, yaitu :<br>Jalur 1 : 1 – 2 – 3 – 5 – 6<br>Jalur 2 : 1 – 2 – 3 – 4 – 6</td>
</tr>
<tr>
<td colspan="2">Test Case, basis set dicoba dan basis set sukses yang dihasilkan 1 – 2 – 3 – 4 – 6, dapat dilihat bahwa simpul telah dieksekusi satu kali. Berdasarkan ketentuan tersebut sistem telah memenuhi syarat kelayakan.</td>
</tr>
<tr>
<td colspan="2">b. Model Tambah Data Sopir</td>
</tr>
<tr>
<td colspan="2"></td>
</tr>
</table>

```
function tambah_sopir($a) {
    $query = $this->db->query
            ('
                    INSERT INTO data_sopir VALUES
                    (
                        ' . $a['id_sopir'] . '
                        , ' . "'" . $a['nama_sopir'] . "'" . '
                        , ' . "'" . $a['alamat'] . "'" . '
                        , ' . "'" . $a['no_lambung'] . "'" . '
                        , ' . "'" . $a['jenis_truk'] . "'" . '
                        , ' . "'" . $a['vol_truk'] . "'" . '
                    )
            ');

    if ($query) {
        return TRUE;
    } else {
        return FALSE;
    }
}
```




V(G) = E – N + 2

V(G) = 6 – 6 + 2

V(G) = 2

Basis set menghasilkan dua jalur independen dari perhitungan kompleksitas siklomatik, yaitu :

Jalur 1 : 1 – 2 – 3 – 5 – 6

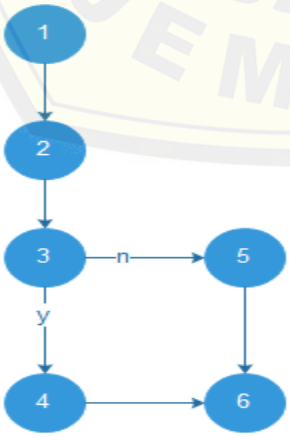
Jalur 2 : 1 – 2 – 3 – 4 – 6

Test Case, basis set dicoba dan basis set sukses yang dihasilkan 1 – 2 – 3 – 4 – 6, dapat dilihat bahwa simpul telah dieksekusi satu kali. Berdasarkan ketentuan tersebut sistem telah memenuhi syarat kelayakan.

c. Controller Edit Sopir

```
function edit() {
    $a = array
        (
        'id_sopir' => $_POST['id_sopir']
        , 'nama_sopir' => $_POST['nama_sopir']
        , 'alamat' => ($_POST['alamat'])
        , 'no_lambung' => $_POST ['no_lambung']
        , 'jenis_truk' => $_POST ['jenis_truk']
        , 'vol_truk' => $_POST ['vol_truk']
    );

    if ($this->m_sopir->sopir_edit($a)) {
        $this->session->set_userdata('pesan_sistem', 'Selamat! edit sopir baru, SUKSE
        $this->session->set_userdata('tipe_pesan', 'Sukses');
        redirect('c_sopir');
    } else {
        $this->session->set_userdata('pesan_sistem', 'Maaf! edit sopir baru, GAGAL!
            Silahkan periksa dan coba kembali');
        $this->session->set_userdata('tipe_pesan', 'Gagal');
        redirect('c_sopir');
    }
}
```




V(G) = E – N + 2

V(G) = 6 – 6 + 2

V(G) = 2

Basis set menghasilkan dua jalur independen dari perhitungan kompleksitas

siklomatik, yaitu :

Jalur 1 : 1 – 2 – 3 – 5 – 6

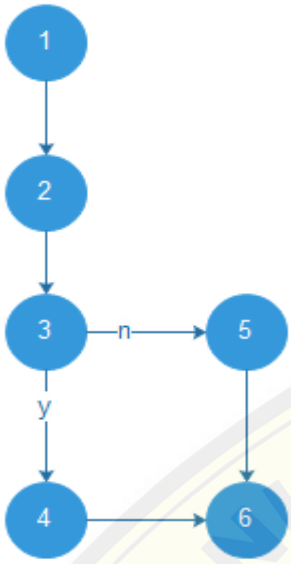
Jalur 2 : 1 – 2 – 3 – 4 – 6

Test Case, basis set dicoba dan basis set sukses yang dihasilkan 1 – 2 – 3 – 4 – 6, dapat dilihat bahwa simpul telah dieksekusi satu kali. Berdasarkan ketentuan tersebut sistem telah memenuhi syarat kelayakan.

d. Model Edit Data Sopir

```
function sopir_edit($a) {   1
    $query = $this->db->query
 2          ('
                UPDATE data_sopir SET
 3                  nama_sopir = ' . "'" . $a['nama_sopir'] . "'" . '
                    , alamat = ' . "'" . $a['alamat'] . "'" . '
                    , no_lambung = ' . "'" . $a['no_lambung'] . "'" .
                    , jenis_truk = ' . "'" . $a['jenis_truk'] . "'" .
                    , vol_truk = ' . "'" . $a['vol_truk'] . "'" . '
                WHERE
                    id_sopir = ' . $a['id_sopir']
            );

 4  if ($query) {
        return TRUE;
 5  } else {
        return FALSE;
    }
}   6
```




V(G) = E – N + 2

V(G) = 6 – 6 + 2

V(G) = 2

Basis set menghasilkan dua jalur independen dari perhitungan kompleksitas siklomatik, yaitu :

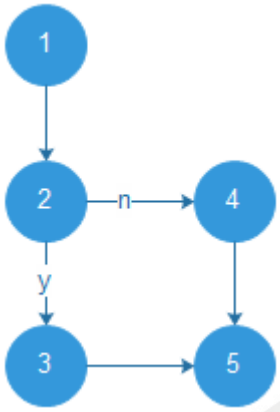
Jalur 1 : 1 – 2 – 3 – 5 – 6

Jalur 2 : 1 – 2 – 3 – 4 – 6

Test Case, basis set dicoba dan basis set sukses yang dihasilkan 1 – 2 – 3 – 4 – 6, dapat dilihat bahwa simpul telah dieksekusi satu kali. Berdasarkan ketentuan tersebut sistem telah memenuhi syarat kelayakan.

e. Controller Hapus Data Sopir

```
1 function delete($a) {
2   if ($this->m_sopir->hapus_sopir($a)) {
3       $this->session->set_userdata('pesan_sistem', 'Selamat! sopir telah dihapus!');
        $this->session->set_userdata('tipe_pesan', 'Sukses');
        redirect('c_sopir');
4   } else {
        $this->session->set_userdata('pesan_sistem', 'Maaf! sopir tidak terhapus!
            Silahkan periksa dan coba kembali');
        $this->session->set_userdata('tipe_pesan', 'Gagal');
        redirect('c_sopir');
    }
}
5
```




V(G) = E – N + 2

V(G) = 5 – 5 + 2

V(G) = 2

Basis set menghasilkan dua jalur independen dari perhitungan kompleksitas siklomatik, yaitu :

Jalur 1 : 1 – 2 – 3 – 5

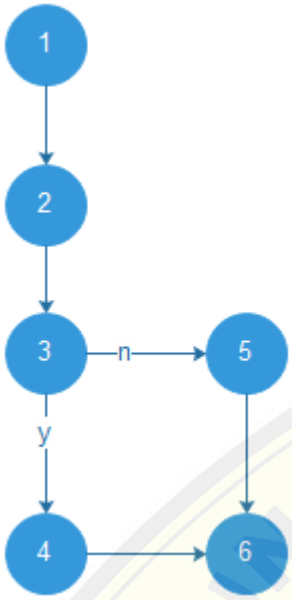
Jalur 2 : 1 – 2 – 4 – 5

Test Case, basis set dicoba dan basis set sukses yang dihasilkan 1 – 2 – 3 – 5, dapat dilihat bahwa simpul telah dieksekusi satu kali. Berdasarkan ketentuan tersebut sistem telah memenuhi syarat kelayakan.

f. Model Hapus Data Sopir

```
1 function hapus_sopir($a) {
      $query = $this->db->query   2
               ('
                        DELETE FROM data_sopir
                        WHERE
                            id_sopir = ' . $a
               );

   3 if ($query) {
           return TRUE;   4
     } else {
   5       return FALSE;
     }
}
       6
```

| | |
|---|---|
|  | V(G) = E – N + 2<br>V(G) = 6 – 6 + 2<br>V(G) = 2 |
| Basis set menghasilkan dua jalur independen dari perhitungan kompleksitas siklomatik, yaitu :<br>Jalur 1 : 1 – 2 – 3 – 5 – 6<br>Jalur 2 : 1 – 2 – 3 – 4 – 6 | |
| Test Case, basis set dicoba dan basis set sukses yang dihasilkan 1 – 2 – 3 – 4 – 6, dapat dilihat bahwa simpul telah dieksekusi satu kali. Berdasarkan ketentuan tersebut sistem telah memenuhi syarat kelayakan. | |

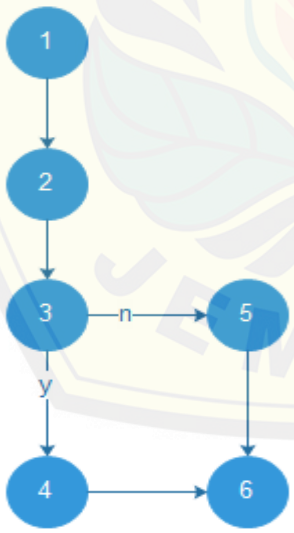
*D. Whitebox Testing* TPA

| |
|---|
| 1) Tambah Data TPA |
| a. Model Tambah Data TPA |

```
function tambah_tpa($a) {
    $query = $this->db->query
            ('
                    INSERT INTO data_TPA VALUES
                    (
                        ' . $a['id_TPA'] . '
                        , ' . "'" . $a['nama'] . "'" . '
                        , ' . "'" . $a['alamat_TPA'] . "'" . '
                        , ' . "'" . $a['kecamatan'] . "'" . '

                    )
            ');

    if ($query) {
        return TRUE;
    } else {
        return FALSE;
    }
}
```




V(G) = E – N + 2

V(G) = 6 – 6 + 2

V(G) = 2

Basis set menghasilkan dua jalur independen dari perhitungan kompleksitas siklomatik, yaitu :

Jalur 1 : 1 – 2 – 3 – 5 – 6

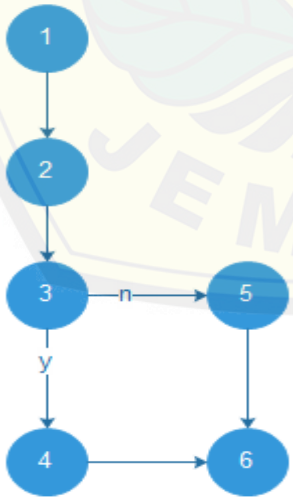
Jalur 2 : 1 – 2 – 3 – 4 – 6

Test Case, basis set dicoba dan basis set sukses yang dihasilkan 1 – 2 – 3 – 4 – 6, dapat dilihat bahwa simpul telah dieksekusi satu kali. Berdasarkan ketentuan tersebut sistem telah memenuhi syarat kelayakan.

b. Controller Tambah Data TPA

```
function add() {
    $a = array
    (
        'id_TPA' => $_POST['id_TPA']
        , 'nama' => ($_POST['nama'])
        , 'alamat_TPA' => $_POST['alamat_TPA']
        , 'kecamatan' => $_POST['kecamatan']
    );

    if ($this->m_tpa->tambah_tpa($a)) {
        $this->session->set_userdata('pesan_sistem', 'Selamat! Penambahan TPS baru, SUKSES!');
        $this->session->set_userdata('tipe_pesan', 'Sukses');
        redirect('c_tpa');
    } else {
        $this->session->set_userdata('pesan_sistem', 'Maaf! Penambahan TPS baru, GAGAL!
            Silahkan periksa dan coba kembali');
        $this->session->set_userdata('tipe_pesan', 'Gagal');
        redirect('c_tpa');
    }
}
```




V(G) = E – N + 2

V(G) = 6 – 6 + 2

V(G) = 2

Basis set menghasilkan dua jalur independen dari perhitungan kompleksitas siklomatik, yaitu :

Jalur 1 : 1 – 2 – 3 – 5 – 6

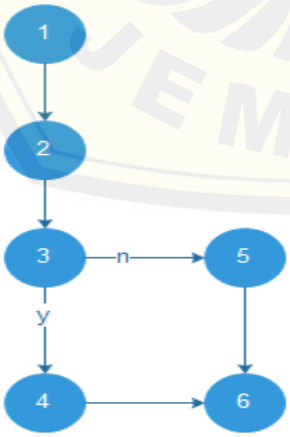
Jalur 2 : 1 – 2 – 3 – 4 – 6

Test Case, basis set dicoba dan basis set sukses yang dihasilkan 1 – 2 – 3 – 4 – 6, dapat dilihat bahwa simpul telah dieksekusi satu kali. Berdasarkan ketentuan tersebut sistem telah memenuhi syarat kelayakan.

c. Controller Edit TPA

```
1  function edit() {
2      $a = array
           (
           'id_TPA' => $_POST['id_TPA']
           , 'nama' => $_POST['nama']
           , 'alamat_TPA' => ($_POST['alamat_TPA'])
           , 'kecamatan' => $_POST['kecamatan']
       );

3      if ($this->m_tpa->tpa_edit($a)) {
           $this->session->set_userdata('pesan_sistem', 'Selamat! edit TPS baru, SUKSES!
4          $this->session->set_userdata('tipe_pesan', 'Sukses');
           redirect('c_tpa');
       } else {
           $this->session->set_userdata('pesan_sistem', 'Maaf! edit TPS baru, GAGAL!
5              Silahkan periksa dan coba kembali');
           $this->session->set_userdata('tipe_pesan', 'Gagal');
           redirect('c_tpa');
       }
   }
6
```




V(G) = E – N + 2

V(G) = 6 – 6 + 2

V(G) = 2

Basis set menghasilkan dua jalur independen dari perhitungan kompleksitas siklomatik, yaitu :

Jalur 1 : 1 – 2 – 3 – 5 – 6

Jalur 2 : 1 – 2 – 3 – 4 – 6

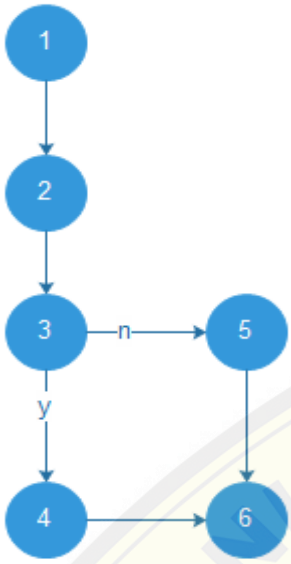
Test Case, basis set dicoba dan basis set sukses yang dihasilkan 1 – 2 – 3 – 4 – 6, dapat dilihat bahwa simpul telah dieksekusi satu kali. Berdasarkan ketentuan tersebut sistem telah memenuhi syarat kelayakan.

d. Model Edit Data TPA

```
1  function tpa_edit($a) {
2      $query = $this->db->query
           ('
               UPDATE data_tpa SET
                   nama = ' . "'" . $a['nama'] . "'" . '
                   , alamat_TPA = ' . "'" . $a['alamat_TPA'] . "'" .
                   , kecamatan = ' . "'" . $a['kecamatan'] . "'" . '

               WHERE
                   id_TPA = ' . $a['id_TPA']
           );

3      if ($query) {
           return TRUE;
4      } else {
5          return FALSE;
       }
   }
6
```




V(G) = E – N + 2

V(G) = 6 – 6 + 2

V(G) = 2

Basis set menghasilkan dua jalur independen dari perhitungan kompleksitas siklomatik, yaitu :

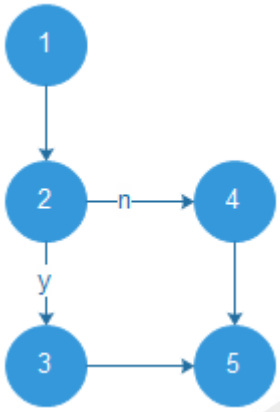
Jalur 1 : 1 – 2 – 3 – 5 – 6

Jalur 2 : 1 – 2 – 3 – 4 – 6

Test Case, basis set dicoba dan basis set sukses yang dihasilkan 1 – 2 – 3 – 4 – 6, dapat dilihat bahwa simpul telah dieksekusi satu kali. Berdasarkan ketentuan tersebut sistem telah memenuhi syarat kelayakan.

e. Controller Hapus Data TPA

```
1 function delete($a) {
  2 if ($this->m_tpa->hapus_tpa($a)) {
      3 $this->session->set_userdata('pesan_sistem', 'Selamat! tpa telah dihapus!')
        $this->session->set_userdata('tipe_pesan', 'Sukses');
        redirect('c_tpa');
   4 } else {
        $this->session->set_userdata('pesan_sistem', 'Maaf! TPS tidak terhapus!
            Silahkan periksa dan coba kembali');
        $this->session->set_userdata('tipe_pesan', 'Gagal');
        redirect('c_tpa');
    }
      5
}
```




V(G) = E – N + 2

V(G) = 5 – 5 + 2

V(G) = 2

Basis set menghasilkan dua jalur independen dari perhitungan kompleksitas siklomatik, yaitu :

Jalur 1 : 1 – 2 – 3 – 5

Jalur 2 : 1 – 2 – 4 – 5

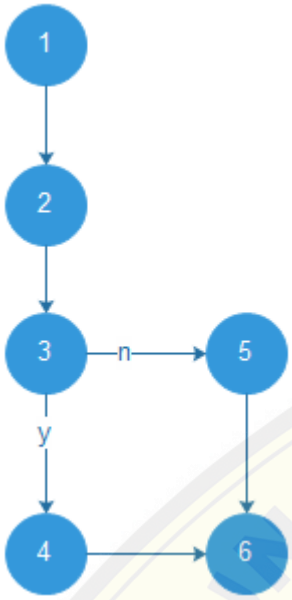
Test Case, basis set dicoba dan basis set sukses yang dihasilkan 1 – 2 – 3 – 5, dapat dilihat bahwa simpul telah dieksekusi satu kali. Berdasarkan ketentuan tersebut sistem telah memenuhi syarat kelayakan.

f. Model Hapus Data TPA

```
function hapus_tpa($a) {
    $query = $this->db->query
        ('
            DELETE FROM data_tpa
            WHERE
                id_TPA = ' . $a
        );

    if ($query) {
        return TRUE;
    } else {
        return FALSE;
    }
}
```

1 2 3 4 5 6

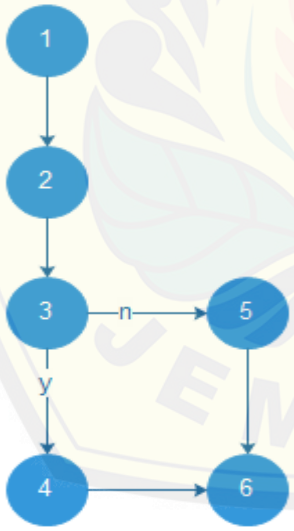
<table>
<tr><td> 
</td><td>V(G) = E – N + 2<br>V(G) = 6 – 6 + 2<br>V(G) = 2</td></tr>
<tr><td colspan="2">Basis set menghasilkan dua jalur independen dari perhitungan kompleksitas siklomatik, yaitu :<br>Jalur 1 : 1 – 2 – 3 – 5 – 6<br>Jalur 2 : 1 – 2 – 3 – 4 – 6</td></tr>
<tr><td colspan="2">Test Case, basis set dicoba dan basis set sukses yang dihasilkan 1 – 2 – 3 – 4 – 6, dapat dilihat bahwa simpul telah dieksekusi satu kali. Berdasarkan ketentuan tersebut sistem telah memenuhi syarat kelayakan.</td></tr>
</table>

*E. Whitebox Testing* Harga

| |
|---|
| 1) Tambah Data Harga |
| a. Model Tambah Data Harga |

```
function tambah_harga($a)    1
    $query = $this->db->query
  2          ('
                    INSERT INTO data_harga VALUES
                    (
                        ' . $a['id_harga'] . '
                        , ' . "'" . $a['bahan_bakar'] . "'" . '
                        , ' . "'" . $a['harga'] . "'" . '
                    )
            ');

  3 if ($query) {
        return TRUE;
    } else {                    4
  5     return FALSE;
    }
}   6
```




V(G) = E – N + 2

V(G) = 6 – 6 + 2

V(G) = 2

Basis set menghasilkan dua jalur independen dari perhitungan kompleksitas siklomatik, yaitu :

Jalur 1 : 1 – 2 – 3 – 5 – 6

Jalur 2 : 1 – 2 – 3 – 4 – 6

Test Case, basis set dicoba dan basis set sukses yang dihasilkan 1 – 2 – 3 – 4 – 6, dapat dilihat bahwa simpul telah dieksekusi satu kali. Berdasarkan ketentuan

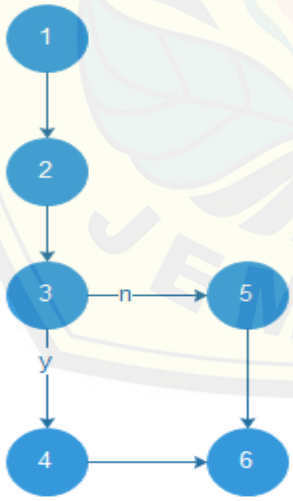
tersebut sistem telah memenuhi syarat kelayakan.

b. Controller Tambah Data Harga

```
function add() {                                    1
      = array                                       2
        (
        'id_harga' => $_POST['id_harga']
        , 'bahan_bakar' => $_POST['bahan_bakar']
        , 'harga' => $_POST['harga']

    );

    if ($this->m_harga->tambah_harga($a)) {         3
        $this->session->set_userdata('pesan_sistem', 'Selamat! Penambahan harga baru, SUKSES!');   4
        $this->session->set_userdata('tipe_pesan', 'Sukses');
        redirect('c_harga');                        5
    } else {
        $this->session->set_userdata('pesan_sistem', 'Maaf! Penambahan harga baru, GAGAL!
            Silahkan periksa dan coba kembali');
        $this->session->set_userdata('tipe_pesan', 'Gagal');
        redirect('c_harga');
    }
}                                                   6
```




V(G) = E – N + 2

V(G) = 6 – 6 + 2

V(G) = 2

Basis set menghasilkan dua jalur independen dari perhitungan kompleksitas siklomatik, yaitu :

Jalur 1 : 1 – 2 – 3 – 5 – 6

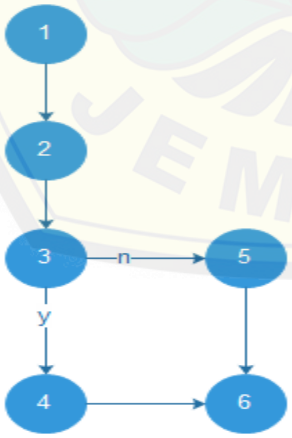
Jalur 2 : 1 – 2 – 3 – 4 – 6

Test Case, basis set dicoba dan basis set sukses yang dihasilkan 1 – 2 – 3 – 4 – 6, dapat dilihat bahwa simpul telah dieksekusi satu kali. Berdasarkan ketentuan tersebut sistem telah memenuhi syarat kelayakan.

c. Controller Edit Harga

```
function edit() {
    $a = array
        (
        'id_harga' => $_POST['id_harga']
        , 'bahan_bakar' => $_POST['bahan_bakar']
        , 'harga' => ($_POST['harga'])

    );
    if ($this->m_harga->harga_edit($a)) {
        $this->session->set_userdata('pesan_sistem', 'Selamat! edit harga baru, SUKSES!
        $this->session->set_userdata('tipe_pesan', 'Sukses');
        redirect('c_harga');
    } else {
        $this->session->set_userdata('pesan_sistem', 'Maaf! edit harga baru, GAGAL!
            Silahkan periksa dan coba kembali');
        $this->session->set_userdata('tipe_pesan', 'Gagal');
        redirect('c_harga');
    }
}
```




V(G) = E – N + 2

V(G) = 6 – 6 + 2

V(G) = 2

Basis set menghasilkan dua jalur independen dari perhitungan kompleksitas siklomatik, yaitu :

Jalur 1 : 1 – 2 – 3 – 5 – 6

Jalur 2 : 1 – 2 – 3 – 4 – 6

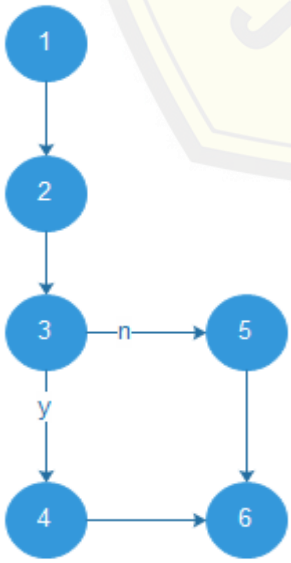
Test Case, basis set dicoba dan basis set sukses yang dihasilkan 1 – 2 – 3 – 4 – 6, dapat dilihat bahwa simpul telah dieksekusi satu kali. Berdasarkan ketentuan tersebut sistem telah memenuhi syarat kelayakan.

d. Model Edit Data Harga

```
function harga_edit($a)
    $query = $this->db->query
            ('
                    UPDATE data_harga SET
                        bahan_bakar = ' . "'" . $a['bahan_bakar'] . "'" .
                        , harga = ' . "'" . $a['harga'] . "'" . '

                    WHERE
                        id_harga = ' . $a['id_harga']
            );

    if ($query) {
        return TRUE;
    } else {
        return FALSE;
    }
}
```




V(G) = E – N + 2

V(G) = 6 – 6 + 2

V(G) = 2

Basis set menghasilkan dua jalur independen dari perhitungan kompleksitas siklomatik, yaitu :

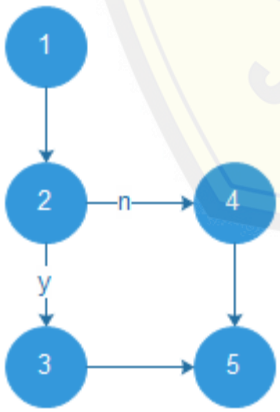
Jalur 1 : 1 – 2 – 3 – 5 – 6

Jalur 2 : 1 – 2 – 3 – 4 – 6

Test Case, basis set dicoba dan basis set sukses yang dihasilkan 1 – 2 – 3 – 4 – 6, dapat dilihat bahwa simpul telah dieksekusi satu kali. Berdasarkan ketentuan tersebut sistem telah memenuhi syarat kelayakan.

e. Controller Hapus Data Harga

```
1 function delete($a) {
  2 if ($this->m_harga->hapus_harga($a)) {
      3 $this->session->set_userdata('pesan_sistem', 'Selamat! harga telah dihapus!')
        $this->session->set_userdata('tipe_pesan', 'Sukses');
        redirect('c_harga');
    } else {
        $this->session->set_userdata('pesan_sistem', 'Maaf! harga tidak terhapus!
            Silahkan periksa dan coba kembali');
    4
        $this->session->set_userdata('tipe_pesan', 'Gagal');
        redirect('c_harga');
    }
}     5
```




V(G) = E – N + 2

V(G) = 5 – 5 + 2

V(G) = 2

Basis set menghasilkan dua jalur independen dari perhitungan kompleksitas siklomatik, yaitu :

Jalur 1 : 1 – 2 – 3 – 5

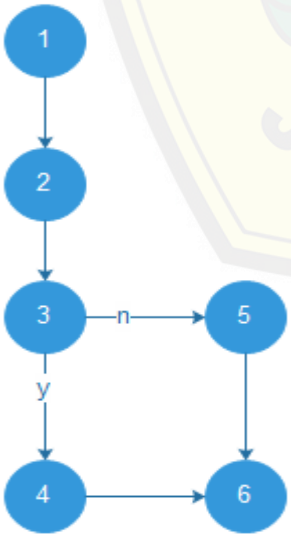
Jalur 2 : 1 – 2 – 4 – 5

Test Case, basis set dicoba dan basis set sukses yang dihasilkan 1 – 2 – 3 – 5, dapat dilihat bahwa simpul telah dieksekusi satu kali. Berdasarkan ketentuan tersebut sistem telah memenuhi syarat kelayakan.

f. Model Hapus Data TPA

```
function hapus_harga($a)
    $query = $this->db->query
             ('
                    DELETE FROM data_harga
                    WHERE
                        id_harga = ' . $a
             );

    if ($query) {
        return TRUE;
    } else {
        return FALSE;
    }
}
```




V(G) = E – N + 2

V(G) = 6 – 6 + 2

V(G) = 2

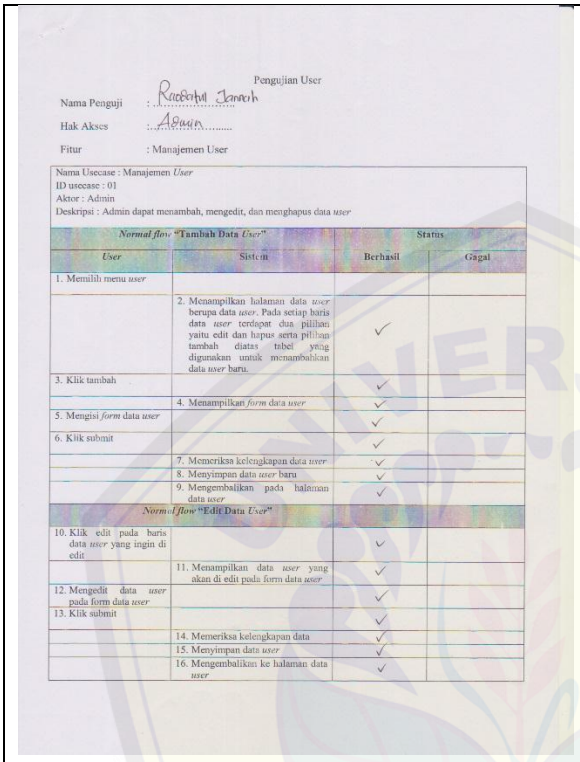
Basis set menghasilkan dua jalur independen dari perhitungan kompleksitas siklomatik, yaitu :

| |
|---|
| Jalur 1 : 1 – 2 – 3 – 5 – 6<br>Jalur 2 : 1 – 2 – 3 – 4 – 6 |
| Test Case, basis set dicoba dan basis set sukses yang dihasilkan 1 – 2 – 3 – 4 – 6, dapat dilihat bahwa simpul telah dieksekusi satu kali. Berdasarkan ketentuan tersebut sistem telah memenuhi syarat kelayakan. |

## B. Pengujian *Black Box*

1. Admin

a. Data User



Pengujian User

Nama Penguji : Raodatul Jannah

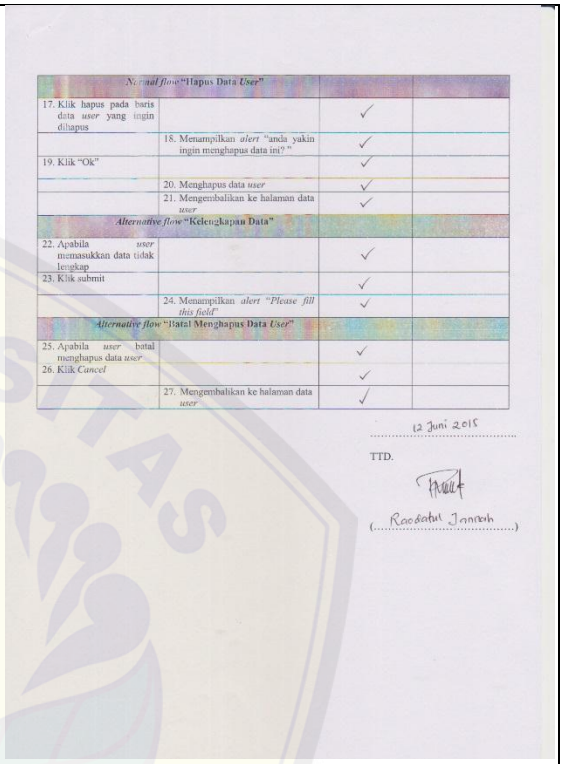
Hak Akses : Admin

Fitur : Manajemen User

Nama Usecase : Manajemen *User*
ID usecase : 01
Aktor : Admin
Deskripsi : Admin dapat menambah, mengedit, dan menghapus data *user*

| *Normal flow* "Tambah Data *User*" | | Status | |
|---|---|---|---|
| *User* | Sistem | Berhasil | Gagal |
| 1. Memilih menu *user* | | | |
| | 2. Menampilkan halaman data *user* berupa data *user*. Pada setiap baris data *user* terdapat dua pilihan yaitu edit dan hapus serta pilihan tambah diatas tabel yang digunakan untuk menambahkan data *user* baru. | ✓ | |
| 3. Klik tambah | | ✓ | |
| | 4. Menampilkan *form* data *user* | ✓ | |
| 5. Mengisi *form* data *user* | | ✓ | |
| 6. Klik submit | | ✓ | |
| | 7. Memeriksa kelengkapan data *user* | ✓ | |
| | 8. Menyimpan data *user* baru | ✓ | |
| | 9. Mengembalikan pada halaman data *user* | ✓ | |
| *Normal flow* "Edit Data *User*" | | | |
| 10. Klik edit pada baris data *user* yang ingin di edit | | ✓ | |
| | 11. Menampilkan data *user* yang akan di edit pada form data *user* | ✓ | |
| 12. Mengedit data *user* pada form data *user* | | ✓ | |
| 13. Klik submit | | ✓ | |
| | 14. Memeriksa kelengkapan data | ✓ | |
| | 15. Menyimpan data *user* | ✓ | |
| | 16. Mengembalikan ke halaman data *user* | ✓ | |

Pengujian fitur data user Hal. 1



| *Normal flow* "Hapus Data *User*" | | | |
|---|---|---|---|
| 17. Klik hapus pada baris data *user* yang ingin dihapus | | ✓ | |
| | 18. Menampilkan *alert* "anda yakin ingin menghapus data ini? " | ✓ | |
| 19. Klik "Ok" | | ✓ | |
| | 20. Menghapus data *user* | ✓ | |
| | 21. Mengembalikan ke halaman data *user* | ✓ | |
| *Alternative flow* "Kelengkapan Data" | | | |
| 22. Apabila *user* memasukkan data tidak lengkap | | ✓ | |
| 23. Klik submit | | ✓ | |
| | 24. Menampilkan *alert "Please fill this field"* | ✓ | |
| *Alternative flow* "Batal Menghapus Data *User*" | | | |
| 25. Apabila *user* batal menghapus data *user* | | ✓ | |
| 26. Klik *Cancel* | | ✓ | |
| | 27. Mengembalikan ke halaman data *user* | ✓ | |

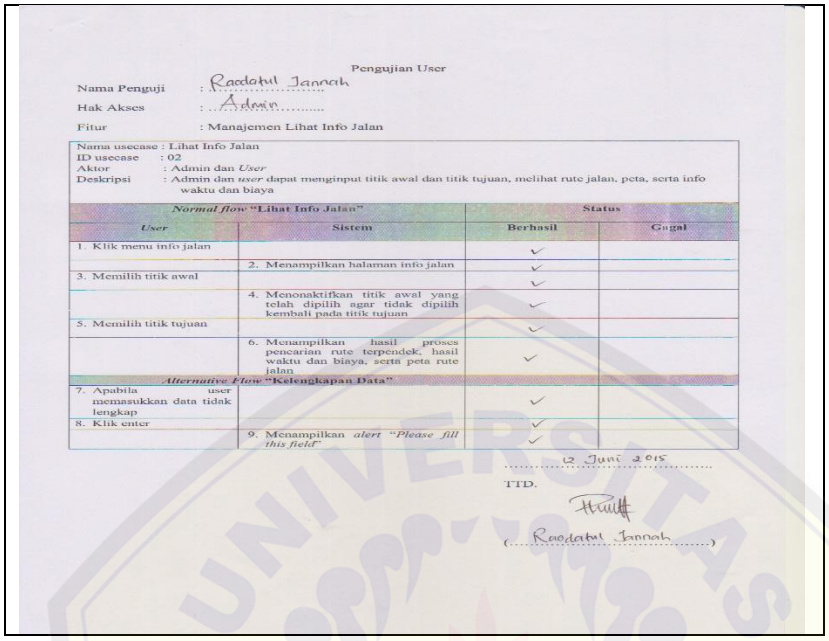
12 Juni 2015

TTD.

(Raodatul Jannah)

Pengujian fitur data user Hal. 2

b. Lihat Info Jalan



Pengujian User

Nama Penguji : Raodatul Jannah
Hak Akses : Admin
Fitur : Manajemen Lihat Info Jalan

Nama usecase : Lihat Info Jalan
ID usecase : 02
Aktor : Admin dan *User*
Deskripsi : Admin dan *user* dapat menginput titik awal dan titik tujuan, melihat rute jalan, peta, serta info waktu dan biaya

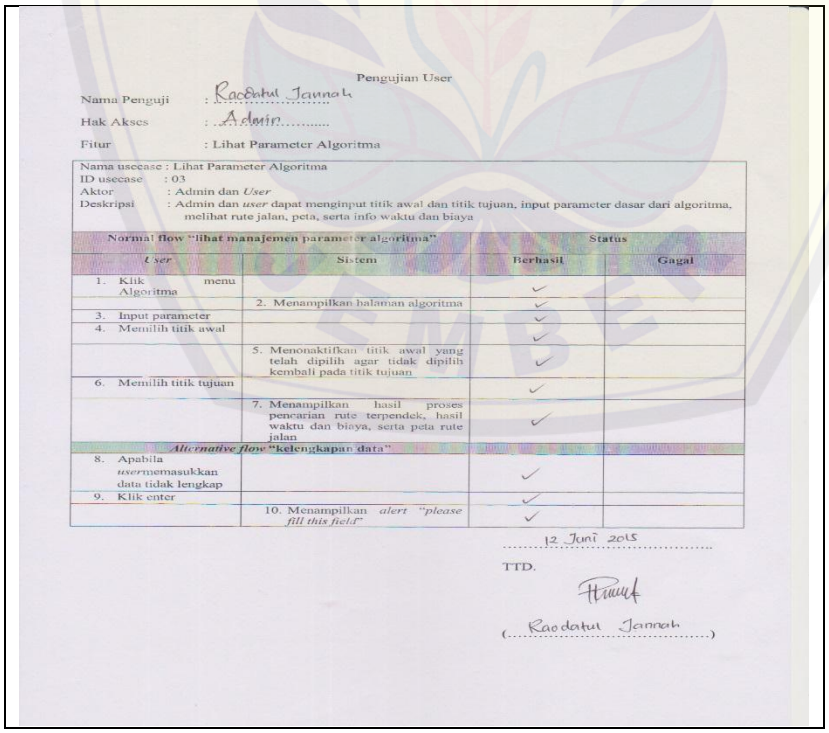
| *Normal flow* "Lihat Info Jalan" | | Status | |
|---|---|---|---|
| *User* | Sistem | Berhasil | Gagal |
| 1. Klik menu info jalan | | ✓ | |
| | 2. Menampilkan halaman info jalan | ✓ | |
| 3. Memilih titik awal | | ✓ | |
| | 4. Menonaktifkan titik awal yang telah dipilih agar tidak dipilih kembali pada titik tujuan | ✓ | |
| 5. Memilih titik tujuan | | ✓ | |
| | 6. Menampilkan hasil proses pencarian rute terpendek, hasil waktu dan biaya, serta peta rute jalan | ✓ | |
| *Alternative Flow* "Kelengkapan Data" | | | |
| 7. Apabila user memasukkan data tidak lengkap | | ✓ | |
| 8. Klik enter | | ✓ | |
| | 9. Menampilkan *alert* "*Please fill this field*" | ✓ | |

12 Juni 2015

TTD.

( Raodatul Jannah )

c. Parameter Algoritma



Pengujian User

Nama Penguji : Raodatul Jannah
Hak Akses : Admin
Fitur : Lihat Parameter Algoritma

Nama usecase : Lihat Parameter Algoritma
ID usecase : 03
Aktor : Admin dan *User*
Deskripsi : Admin dan *user* dapat menginput titik awal dan titik tujuan, input parameter dasar dari algoritma, melihat rute jalan, peta, serta info waktu dan biaya

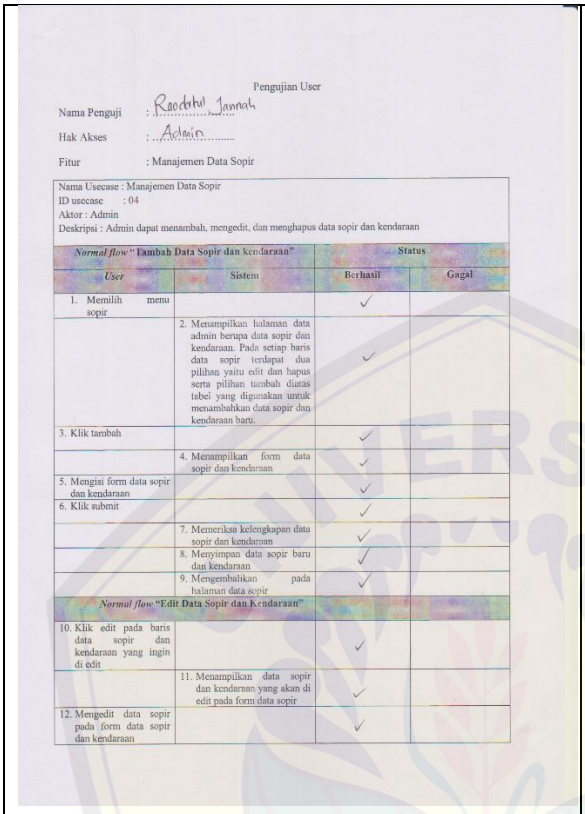
| Normal flow "lihat manajemen parameter algoritma" | | Status | |
|---|---|---|---|
| *User* | Sistem | Berhasil | Gagal |
| 1. Klik menu Algoritma | | ✓ | |
| | 2. Menampilkan halaman algoritma | ✓ | |
| 3. Input parameter | | ✓ | |
| 4. Memilih titik awal | | ✓ | |
| | 5. Menonaktifkan titik awal yang telah dipilih agar tidak dipilih kembali pada titik tujuan | ✓ | |
| 6. Memilih titik tujuan | | ✓ | |
| | 7. Menampilkan hasil proses pencarian rute terpendek, hasil waktu dan biaya, serta peta rute jalan | ✓ | |
| *Alternative flow* "kelengkapan data" | | | |
| 8. Apabila *user*memasukkan data tidak lengkap | | ✓ | |
| 9. Klik enter | | ✓ | |
| | 10. Menampilkan *alert* "*please fill this field*" | ✓ | |

12 Juni 2015

TTD.

( Raodatul Jannah )

d. Manajemen Data Sopir



Pengujian User

Nama Penguji : Raodatul Jannah

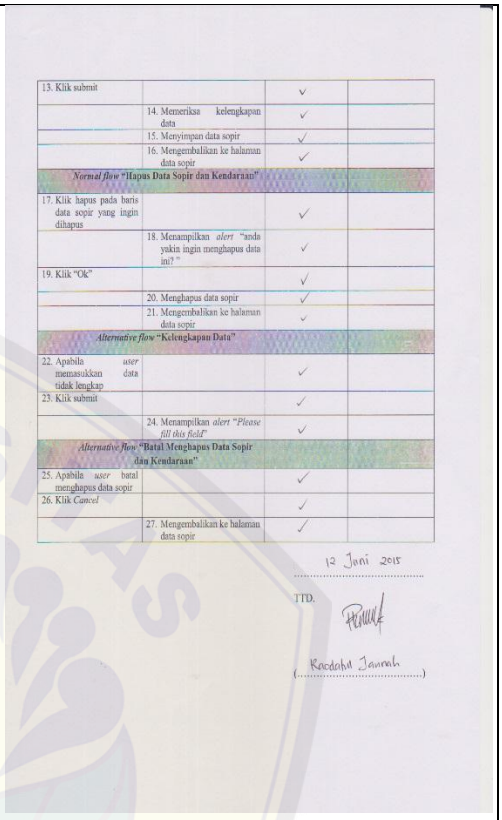
Hak Akses : Admin

Fitur : Manajemen Data Sopir

Nama Usecase : Manajemen Data Sopir
ID usecase : 04
Aktor : Admin
Deskripsi : Admin dapat menambah, mengedit, dan menghapus data sopir dan kendaraan

| *Normal flow* "Tambah Data Sopir dan Kendaraan" | | Status | |
|---|---|---|---|
| *User* | Sistem | Berhasil | Gagal |
| 1. Memilih menu sopir | | ✓ | |
| | 2. Menampilkan halaman data admin berupa data sopir dan kendaraan. Pada setiap baris data sopir terdapat dua pilihan yaitu edit dan hapus serta pilihan tambah diatas tabel yang digunakan untuk menambahkan data sopir dan kendaraan baru. | ✓ | |
| 3. Klik tambah | | ✓ | |
| | 4. Menampilkan form data sopir dan kendaraan | ✓ | |
| 5. Mengisi form data sopir dan kendaraan | | ✓ | |
| 6. Klik submit | | ✓ | |
| | 7. Memeriksa kelengkapan data sopir dan kendaraan | ✓ | |
| | 8. Menyimpan data sopir baru dan kendaraan | ✓ | |
| | 9. Mengembalikan pada halaman data sopir | ✓ | |
| *Normal flow* "Edit Data Sopir dan Kendaraan" | | | |
| 10. Klik edit pada baris data sopir dan kendaraan yang ingin di edit | | ✓ | |
| | 11. Menampilkan data sopir dan kendaraan yang akan di edit pada form data sopir | ✓ | |
| 12. Mengedit data sopir pada form data sopir dan kendaraan | | ✓ | |

Pengujian fitur data sopir Hal. 1



| | | | |
|---|---|---|---|
| 13. Klik submit | | ✓ | |
| | 14. Memeriksa kelengkapan data | ✓ | |
| | 15. Menyimpan data sopir | ✓ | |
| | 16. Mengembalikan ke halaman data sopir | ✓ | |
| *Normal flow* "Hapus Data Sopir dan Kendaraan" | | | |
| 17. Klik hapus pada baris data sopir yang ingin dihapus | | ✓ | |
| | 18. Menampilkan *alert* "anda yakin ingin menghapus data ini? " | ✓ | |
| 19. Klik "Ok" | | ✓ | |
| | 20. Menghapus data sopir | ✓ | |
| | 21. Mengembalikan ke halaman data sopir | ✓ | |
| *Alternative flow* "Kelengkapan Data" | | | |
| 22. Apabila *user* memasukkan data tidak lengkap | | ✓ | |
| 23. Klik submit | | ✓ | |
| | 24. Menampilkan *alert "Please fill this field"* | ✓ | |
| *Alternative flow* "Batal Menghapus Data Sopir dan Kendaraan" | | | |
| 25. Apabila *user* batal menghapus data sopir | | ✓ | |
| 26. Klik *Cancel* | | ✓ | |
| | 27. Mengembalikan ke halaman data sopir | ✓ | |

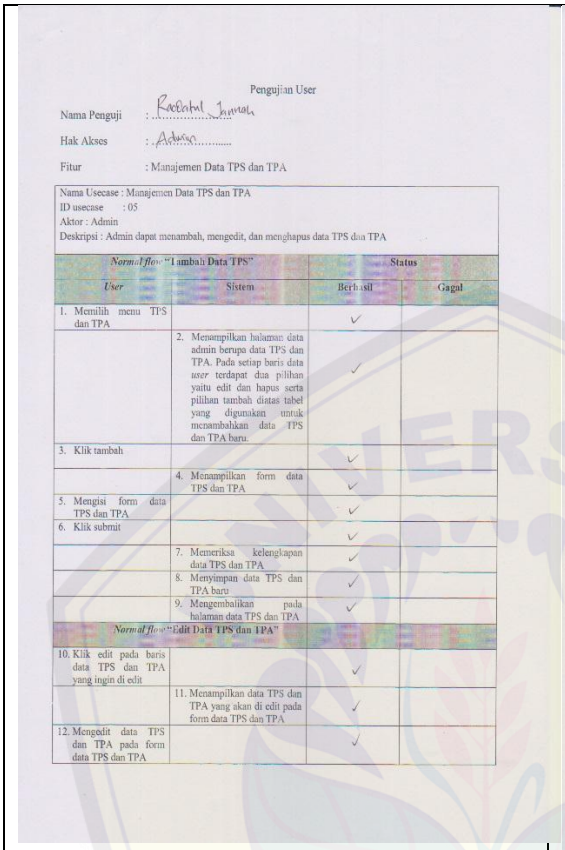
12 Juni 2015

TTD.

(Raodatul Jannah)

Pengujian fitur data sopir Hal. 2

e. Manajemen Data TPS dan TPA



Pengujian User

Nama Penguji : Raodatul Jannah

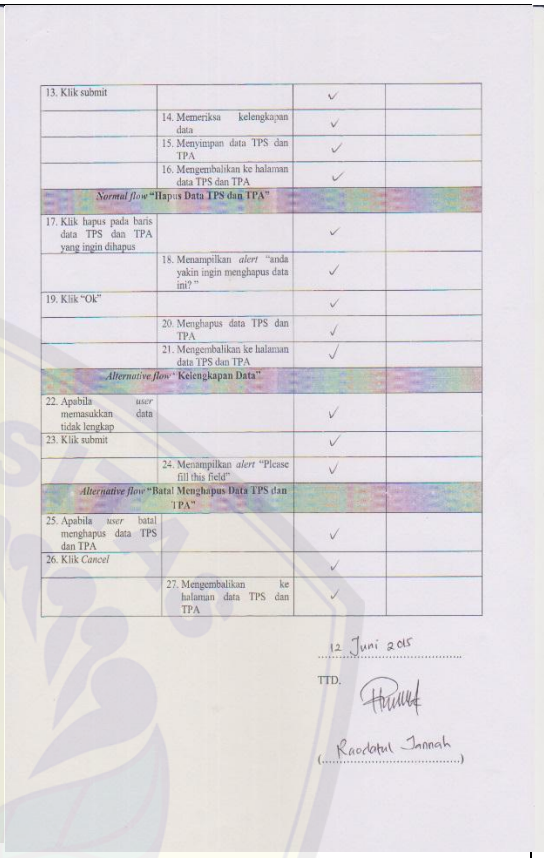
Hak Akses : Admin

Fitur : Manajemen Data TPS dan TPA

Nama Usecase : Manajemen Data TPS dan TPA
ID usecase : 05
Aktor : Admin
Deskripsi : Admin dapat menambah, mengedit, dan menghapus data TPS dan TPA

| *Normal flow* "Tambah Data TPS" | | Status | |
|---|---|---|---|
| *User* | Sistem | Berhasil | Gagal |
| 1. Memilih menu TPS dan TPA | | ✓ | |
| | 2. Menampilkan halaman data admin berupa data TPS dan TPA. Pada setiap baris data *user* terdapat dua pilihan yaitu edit dan hapus serta pilihan tambah diatas tabel yang digunakan untuk menambahkan data TPS dan TPA baru. | ✓ | |
| 3. Klik tambah | | ✓ | |
| | 4. Menampilkan form data TPS dan TPA | ✓ | |
| 5. Mengisi form data TPS dan TPA | | ✓ | |
| 6. Klik submit | | ✓ | |
| | 7. Memeriksa kelengkapan data TPS dan TPA | ✓ | |
| | 8. Menyimpan data TPS dan TPA baru | ✓ | |
| | 9. Mengembalikan pada halaman data TPS dan TPA | ✓ | |
| *Normal flow* "Edit Data TPS dan TPA" | | | |
| 10. Klik edit pada baris data TPS dan TPA yang ingin di edit | | ✓ | |
| | 11. Menampilkan data TPS dan TPA yang akan di edit pada form data TPS dan TPA | ✓ | |
| 12. Mengedit data TPS dan TPA pada form data TPS dan TPA | | ✓ | |

Pengujian fitur TPA Hal. 1



| | | | |
|---|---|---|---|
| 13. Klik submit | | ✓ | |
| | 14. Memeriksa kelengkapan data | ✓ | |
| | 15. Menyimpan data TPS dan TPA | ✓ | |
| | 16. Mengembalikan ke halaman data TPS dan TPA | ✓ | |
| *Normal flow* "Hapus Data TPS dan TPA" | | | |
| 17. Klik hapus pada baris data TPS dan TPA yang ingin dihapus | | ✓ | |
| | 18. Menampilkan *alert* "anda yakin ingin menghapus data ini? " | ✓ | |
| 19. Klik "Ok" | | ✓ | |
| | 20. Menghapus data TPS dan TPA | ✓ | |
| | 21. Mengembalikan ke halaman data TPS dan TPA | ✓ | |
| *Alternative flow* "Kelengkapan Data" | | | |
| 22. Apabila *user* memasukkan data tidak lengkap | | ✓ | |
| 23. Klik submit | | ✓ | |
| | 24. Menampilkan *alert* "Please fill this field" | ✓ | |
| *Alternative flow* "Batal Menghapus Data TPS dan TPA" | | | |
| 25. Apabila *user* batal menghapus data TPS dan TPA | | ✓ | |
| 26. Klik *Cancel* | | ✓ | |
| | 27. Mengembalikan ke halaman data TPS dan TPA | ✓ | |

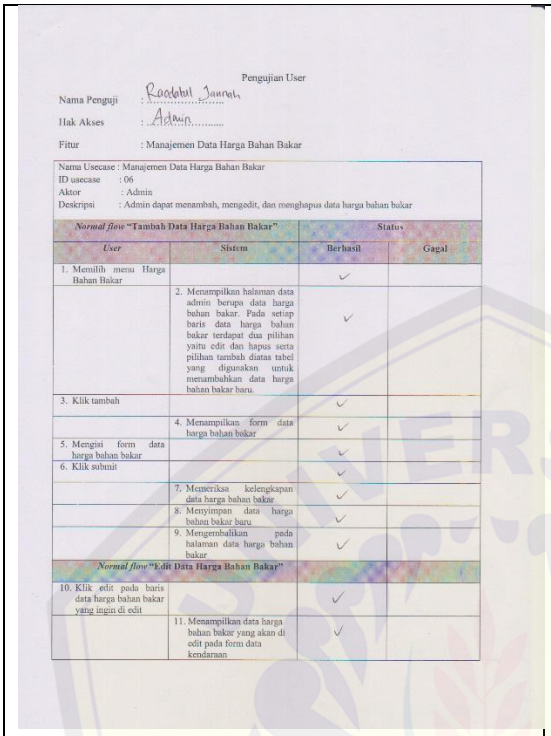
12 Juni 2015

TTD.

Raodatul Jannah

Pengujian fitur TPA Hal. 2

f. Manajemen Harga Bahan Bakar



Pengujian User

Nama Penguji : Raodatul Jannah

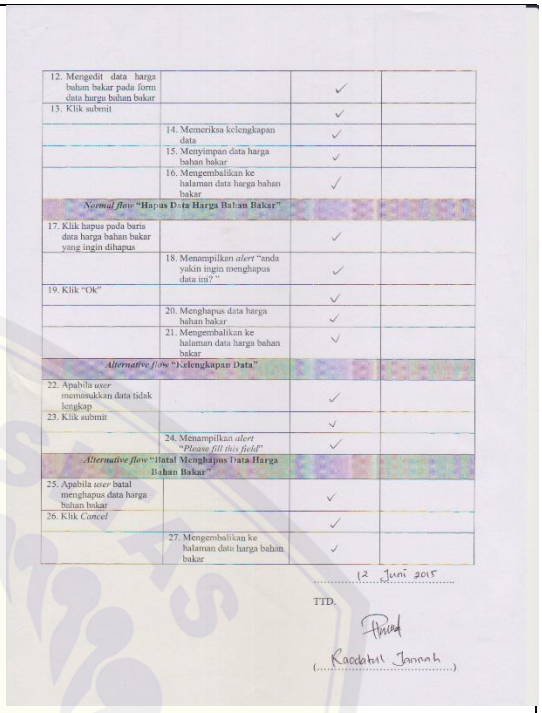
Hak Akses : Admin

Fitur : Manajemen Data Harga Bahan Bakar

Nama Usecase : Manajemen Data Harga Bahan Bakar
ID usecase : 06
Aktor : Admin
Deskripsi : Admin dapat menambah, mengedit, dan menghapus data harga bahan bakar

| *Normal flow* "Tambah Data Harga Bahan Bakar" | | Status | |
|---|---|---|---|
| *User* | Sistem | Berhasil | Gagal |
| 1. Memilih menu Harga Bahan Bakar | | ✓ | |
| | 2. Menampilkan halaman data admin berupa data harga bahan bakar. Pada setiap baris data harga bahan bakar terdapat dua pilihan yaitu edit dan hapus serta pilihan tambah diatas tabel yang digunakan untuk menambahkan data harga bahan bakar baru. | ✓ | |
| 3. Klik tambah | | ✓ | |
| | 4. Menampilkan form data harga bahan bakar | ✓ | |
| 5. Mengisi form data harga bahan bakar | | ✓ | |
| 6. Klik submit | | ✓ | |
| | 7. Memeriksa kelengkapan data harga bahan bakar | ✓ | |
| | 8. Menyimpan data harga bahan bakar baru | ✓ | |
| | 9. Mengembalikan pada halaman data harga bahan bakar | ✓ | |
| *Normal flow* "Edit Data Harga Bahan Bakar" | | | |
| 10. Klik edit pada baris data harga bahan bakar yang ingin di edit | | ✓ | |
| | 11. Menampilkan data harga bahan bakar yang akan di edit pada form data kendaraan | ✓ | |

Pengujian fitur Harga Hal. 1



| | | | |
|---|---|---|---|
| 12. Mengedit data harga bahan bakar pada form data harga bahan bakar | | ✓ | |
| 13. Klik submit | | ✓ | |
| | 14. Memeriksa kelengkapan data | ✓ | |
| | 15. Menyimpan data harga bahan bakar | ✓ | |
| | 16. Mengembalikan ke halaman data harga bahan bakar | ✓ | |
| *Normal flow* "Hapus Data Harga Bahan Bakar" | | | |
| 17. Klik hapus pada baris data harga bahan bakar yang ingin dihapus | | ✓ | |
| | 18. Menampilkan *alert* "anda yakin ingin menghapus data ini? " | ✓ | |
| 19. Klik "Ok" | | ✓ | |
| | 20. Menghapus data harga bahan bakar | ✓ | |
| | 21. Mengembalikan ke halaman data harga bahan bakar | ✓ | |
| *Alternative flow* "Kelengkapan Data" | | | |
| 22. Apabila *user* memasukkan data tidak lengkap | | ✓ | |
| 23. Klik submit | | ✓ | |
| | 24. Menampilkan *alert* "*Please fill this field*" | ✓ | |
| *Alternative flow* "Batal Menghapus Data Harga Bahan Bakar" | | | |
| 25. Apabila *user* batal menghapus data harga bahan bakar | | ✓ | |
| 26. Klik *Cancel* | | ✓ | |
| | 27. Mengembalikan ke halaman data harga bahan bakar | ✓ | |

12 Juni 2015

TTD.

(Raodatul Jannah)

Pengujian fitur harga Hal. 2

2. User 1

Pengujian User

Nama Penguji : Prima Fajarahkta

Hak Akses : User

Fitur : Manajemen Lihat Info Jalan

Nama usecase : Lihat Info Jalan
ID usecase : 02
Aktor : Admin dan *User*
Deskripsi : Admin dan *user* dapat menginput titik awal dan titik tujuan, melihat rute jalan, peta, serta info waktu dan biaya

| *Normal flow* "Lihat Info Jalan" | | Status | |
|---|---|---|---|
| *User* | Sistem | Berhasil | Gagal |
| 1. Klik menu info jalan | | ✓ | |
| | 2. Menampilkan halaman info jalan | ✓ | |
| 3. Memilih titik awal | | ✓ | |
| | 4. Menonaktifkan titik awal yang telah dipilih agar tidak dipilih kembali pada titik tujuan | ✓ | |
| 5. Memilih titik tujuan | | ✓ | |
| | 6. Menampilkan hasil proses pencarian rute terpendek, hasil waktu dan biaya, serta peta rute jalan | ✓ | |
| *Alternative Flow* "Kelengkapan Data" | | | |
| 7. Apabila user memasukkan data tidak lengkap | | ✓ | |
| 8. Klik enter | | ✓ | |
| | 9. Menampilkan *alert* "*Please fill this field*" | ✓ | |

7 Juni '15

TTD.

(Prima Fajarahkta)

Pengujian fitur Lihat Info Jalan

Pengujian User

Nama Penguji : Prima Fajarahkta

Hak Akses : User

Fitur : Lihat Parameter Algoritma

Nama usecase : Lihat Parameter Algoritma
ID usecase : 03
Aktor : Admin dan *User*
Deskripsi : Admin dan *user* dapat menginput titik awal dan titik tujuan, input parameter dasar dari algoritma, melihat rute jalan, peta, serta info waktu dan biaya

| Normal flow "lihat manajemen parameter algoritma" | | Status | |
|---|---|---|---|
| *User* | Sistem | Berhasil | Gagal |
| 1. Klik menu Algoritma | | ✓ | |
| | 2. Menampilkan halaman algoritma | ✓ | |
| 3. Input parameter | | ✓ | |
| 4. Memilih titik awal | | ✓ | |
| | 5. Menonaktifkan titik awal yang telah dipilih agar tidak dipilih kembali pada titik tujuan | ✓ | |
| 6. Memilih titik tujuan | | ✓ | |
| | 7. Menampilkan hasil proses pencarian rute terpendek, hasil waktu dan biaya, serta peta rute jalan | ✓ | |
| *Alternative flow* "kelengkapan data" | | | |
| 8. Apabila *user*memasukkan data tidak lengkap | | ✓ | |
| 9. Klik enter | | ✓ | |
| | 10. Menampilkan *alert* "*please fill this field*" | ✓ | |

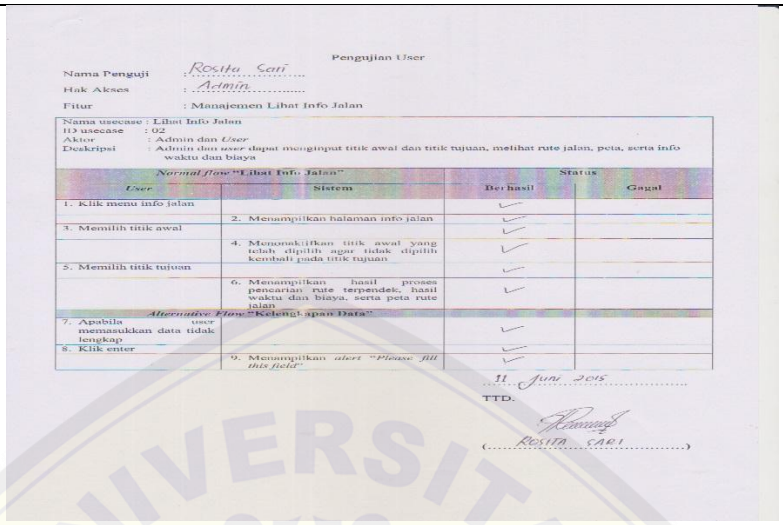
7 Juni '15

TTD.

(Prima Fajarahkta)

Pengujian Parameter Algoritma

3. User 2



Pengujian User

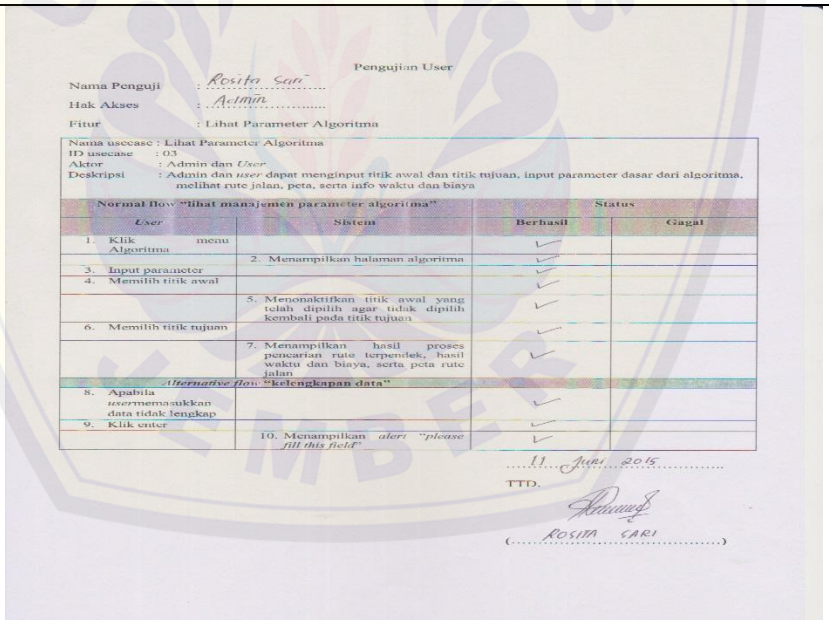
Nama Penguji : Rosita Sari
Hak Akses : Admin
Fitur : Manajemen Lihat Info Jalan

Nama usecase : Lihat Info Jalan
ID usecase : 02
Aktor : Admin dan *User*
Deskripsi : Admin dan *user* dapat menginput titik awal dan titik tujuan, melihat rute jalan, peta, serta info waktu dan biaya

| *Normal flow* "Lihat Info Jalan" | | Status | |
|---|---|---|---|
| *User* | Sistem | Berhasil | Gagal |
| 1. Klik menu info jalan | | ✓ | |
| | 2. Menampilkan halaman info jalan | ✓ | |
| 3. Memilih titik awal | | ✓ | |
| | 4. Menonaktifkan titik awal yang telah dipilih agar tidak dipilih kembali pada titik tujuan | ✓ | |
| 5. Memilih titik tujuan | | ✓ | |
| | 6. Menampilkan hasil proses pencarian rute terpendek, hasil waktu dan biaya, serta peta rute jalan | ✓ | |
| *Alternative Flow* "Kelengkapan Data" | | | |
| 7. Apabila user memasukkan data tidak lengkap | | ✓ | |
| 8. Klik enter | | ✓ | |
| | 9. Menampilkan *alert* "*Please fill this field*" | ✓ | |

11 Juni 2015
TTD.
(ROSITA SARI)

Penguiian fitur Lihat Info JAlan



Pengujian User

Nama Penguji : Rosita Sari
Hak Akses : Admin
Fitur : Lihat Parameter Algoritma

Nama usecase : Lihat Parameter Algoritma
ID usecase : 03
Aktor : Admin dan *User*
Deskripsi : Admin dan *user* dapat menginput titik awal dan titik tujuan, input parameter dasar dari algoritma, melihat rute jalan, peta, serta info waktu dan biaya

| Normal flow "lihat manajemen parameter algoritma" | | Status | |
|---|---|---|---|
| *User* | Sistem | Berhasil | Gagal |
| 1. Klik menu Algoritma | | ✓ | |
| | 2. Menampilkan halaman algoritma | ✓ | |
| 3. Input parameter | | ✓ | |
| 4. Memilih titik awal | | ✓ | |
| | 5. Menonaktifkan titik awal yang telah dipilih agar tidak dipilih kembali pada titik tujuan | ✓ | |
| 6. Memilih titik tujuan | | ✓ | |
| | 7. Menampilkan hasil proses pencarian rute terpendek, hasil waktu dan biaya, serta peta rute jalan | ✓ | |
| *Alternative flow* "kelengkapan data" | | | |
| 8. Apabila *user*memasukkan data tidak lengkap | | ✓ | |
| 9. Klik enter | | ✓ | |
| | 10. Menampilkan *alert* "*please fill this field*" | ✓ | |

11 Juni 2015
TTD.
(ROSITA SARI)

Penguiian fitur Parameter Algoritma

**C. Penulisan Kode Program**

1. Data User

a. Tambah Data User

1) Controller Tambah Data User

```
function add() {
    $a = array
      (
      'username' => $_POST['username']
      , 'password' => md5($_POST['password'])
      , 'nama' => $_POST['nama']
      , 'email' => $_POST['email']
      , 'level' => $_POST['level']
    );
    if ($this->m_user->tambah_user($a)) {
      $this->session->set_userdata('pesan_sistem','Selamat! Penambahan user
     baru, SUKSES!');
      $this->session->set_userdata('tipe_pesan', 'Sukses');
      redirect('user');
    } else {
      $this->session->set_userdata('pesan_sistem', 'Maaf! Penambahan user
     baru,
          GAGAL! Silahkan periksa dan coba kembali');
      $this->session->set_userdata('tipe_pesan', 'Gagal');
      redirect('user');
    } }
```

2) Model Tambah Data User

```
function tambah_user($a) {
```

```
    $query = $this->db->query
        ('
            INSERT INTO user VALUES
            (
              ' . "''" . "''" . '
              , ' . "'" . $a['level'] . "'" . '
              , ' . "'" . $a['username'] . "'" . '
              , ' . "'" . $a['nama'] . "'" . '
              , ' . "'" . $a['password'] . "'" . '
              , ' . "'" . $a['email'] . "'" . '
              , 1
            )
      ');

    if ($query) {
      return TRUE;
    } else {
      return FALSE;
    }
  }
```

b. Edit Data User

1) Controller Edit Data User

```
function edit() {
    $a = array
      (
      'id' => $_POST['id']
      , 'username' => $_POST['username']
```

```
            , 'password' => md5($_POST['password'])
            , 'nama' => $_POST['nama']
            , 'email' => $_POST['email']
            , 'level' => $_POST['level']
        );

        if ($this->m_user->user_edit($a)) {
            $this->session->set_userdata('pesan_sistem', 'Selamat! edit user baru,
          SUKSES!');
            $this->session->set_userdata('tipe_pesan', 'Sukses');
            redirect('user');
        } else {
            $this->session->set_userdata('pesan_sistem', 'Maaf! edit user baru,
          GAGAL!
              Silahkan periksa dan coba kembali');
            $this->session->set_userdata('tipe_pesan', 'Gagal');
            redirect('user');
        }
    }
```

2) Model Edit Data User

```
function user_edit_isi($a) {
    $query = $this->db->query
        ('
                SELECT
            id
            , username
            , password
```

```
                , nama
                , status_user
                , email
                , level
            FROM
                user
                    WHERE
                        id = ' . $a
            );

    return $query->result_array();
}

function user_edit($a) {
    $query = $this->db->query
        ('
                    UPDATE user SET
                level = ' . "'" . $a['level'] . "'" . '
                , username = ' . "'" . $a['username'] . "'" . '
                , nama = ' . "'" . $a['nama'] . "'" . '
                , password = ' . "'" . $a['password'] . "'" . '
                , email = ' . "'" . $a['email'] . "'" . '
                , status_user = ' . "'" . $a['status_user'] . "'" . '
                    WHERE
                        id = ' . $a['id']
            );

    if ($query) {
```

```
            return TRUE;
        } else {
            return FALSE;
        }
    }
```

c. Hapus Data User

1) Controller Hapus Data User

```
function delete($a) {
        if ($this->m_user->hapus_user($a)) {
            $this->session->set_userdata('pesan_sistem', 'Selamat! User telah
            dihapus!');
            $this->session->set_userdata('tipe_pesan', 'Sukses');
            redirect('user');
        } else {
            $this->session->set_userdata('pesan_sistem', 'Maaf! User tidak terhapus!
            Silahkan periksa dan coba kembali');
            $this->session->set_userdata('tipe_pesan', 'Gagal');
            redirect('user');
        }
    }
```

2) Model Hapus Data User

```
function hapus_user($a) {
        $query = $this->db->query
                ('
                        UPDATE user SET
                    status_user = 0
```

```
                    WHERE
                        id = ' . $a
                );


        if ($query) {
            return TRUE;
        } else {
            return FALSE;
        }
    }
```

2. Data Sopir

a. Tambah Data User

1) Controller Tambah Data Sopir

```
function add() {
        $a = array
          (
          'id_sopir' => $_POST['id_sopir']
          , 'nama_sopir' => $_POST['nama_sopir']
          , 'alamat' => $_POST['alamat']
          , 'no_lambung' => $_POST ['no_lambung']
          , 'jenis_truk' => $_POST ['jenis_truk']
          , 'vol_truk' => $_POST ['vol_truk']
        );

        if ($this->m_sopir->tambah_sopir($a)) {
            $this->session->set_userdata('pesan_sistem', 'Selamat! Penambahan sopir
```

```
            baru, SUKSES!');
             $this->session->set_userdata('tipe_pesan', 'Sukses');
             redirect('c_sopir');
          } else {
             $this->session->set_userdata('pesan_sistem', 'Maaf! Penambahan jalan
            baru, GAGAL!
                Silahkan periksa dan coba kembali');
             $this->session->set_userdata('tipe_pesan', 'Gagal');
             redirect('c_sopir');
          }
      }
```

2) Model Tambah Data Sopir

```
function tambah_sopir($a) {
      $query = $this->db->query
            ('
                  INSERT INTO data_sopir VALUES
                  (
                     ' . $a['id_sopir'] . '
                     , ' . "'" . $a['nama_sopir'] . "'" . '
                     , ' . "'" . $a['alamat'] . "'" . '
                     , ' . "'" . $a['no_lambung'] . "'" . '
                     , ' . "'" . $a['jenis_truk'] . "'" . '
                     , ' . "'" . $a['vol_truk'] . "'" . '
                  )
            ');

      if ($query) {
```

```
            return TRUE;
        } else {
            return FALSE;
        }
    }
```

b. Edit Data Sopir

1) Controller Edit Data Sopir

```
function edit() {
        $a = array
            (
            'id_sopir' => $_POST['id_sopir']
            , 'nama_sopir' => $_POST['nama_sopir']
            , 'alamat' => ($_POST['alamat'])
            , 'no_lambung' => $_POST ['no_lambung']
            , 'jenis_truk' => $_POST ['jenis_truk']
            , 'vol_truk' => $_POST ['vol_truk']
        );

        if ($this->m_sopir->sopir_edit($a)) {
            $this->session->set_userdata('pesan_sistem', 'Selamat! edit sopir baru,
          SUKSES!');
            $this->session->set_userdata('tipe_pesan', 'Sukses');
            redirect('c_sopir');
        } else {
            $this->session->set_userdata('pesan_sistem', 'Maaf! edit sopir baru,
          GAGAL!
              Silahkan periksa dan coba kembali');
```

```
            $this->session->set_userdata('tipe_pesan', 'Gagal');
            redirect('c_sopir');
        }
    }
```

2) Model Edit Data Sopir

```
function sopir_edit_isi($a) {
    $query = $this->db->query
        ('
                SELECT
            id_sopir
            , nama_sopir
            , alamat
            , no_lambung
            ,jenis_truk
            ,vol_truk

        FROM
            data_sopir
                WHERE
                    id_sopir = ' . $a
        );

    return $query->result_array();
}

function sopir_edit($a) {
    $query = $this->db->query
```

```
            ('
                        UPDATE data_sopir SET
                    nama_sopir = ' . "'" . $a['nama_sopir'] . "'" . '
                    , alamat = ' . "'" . $a['alamat'] . "'" . '
                    , no_lambung = ' . "'" . $a['no_lambung'] . "'" . '
                    , jenis_truk = ' . "'" . $a['jenis_truk'] . "'" . '
                    , vol_truk = ' . "'" . $a['vol_truk'] . "'" . '
                        WHERE
                            id_sopir = ' . $a['id_sopir']
                );

    if ($query) {
      return TRUE;
    } else {
      return FALSE;
    }
  }
```

c. Hapus Data Sopir

1) Controller Hapus Data Sopir

```
function delete($a) {
      if ($this->m_sopir->hapus_sopir($a)) {
        $this->session->set_userdata('pesan_sistem', 'Selamat! sopir telah
        dihapus!');
        $this->session->set_userdata('tipe_pesan', 'Sukses');
        redirect('c_sopir');
      } else {
        $this->session->set_userdata('pesan_sistem', 'Maaf! sopir tidak terhapus!
```

```
            Silahkan periksa dan coba kembali');
        $this->session->set_userdata('tipe_pesan', 'Gagal');
        redirect('c_sopir');
    }
}
```

2) Model Hapus Data Sopir

```
function hapus_sopir($a) {
    $query = $this->db->query
        ('
            DELETE FROM data_sopir
            WHERE
                id_sopir = ' . $a
        );

    if ($query) {
        return TRUE;
    } else {
        return FALSE;
    }
}
```

3. Data TPA

a.Tambah Data TPA

1) Controller Tambah Data TPA

```
function add() {
    $a = array
        (
```

```
            'id_TPA' => $_POST['id_TPA']
            , 'nama' => ($_POST['nama'])
            , 'alamat_TPA' => $_POST['alamat_TPA']
            , 'kecamatan' => $_POST['kecamatan']
        );

        if ($this->m_tpa->tambah_tpa($a)) {
            $this->session->set_userdata('pesan_sistem', 'Selamat! Penambahan TPS
          baru, SUKSES!');
            $this->session->set_userdata('tipe_pesan', 'Sukses');
            redirect('c_tpa');
        } else {
            $this->session->set_userdata('pesan_sistem', 'Maaf! Penambahan TPS
          baru, GAGAL!
              Silahkan periksa dan coba kembali');
            $this->session->set_userdata('tipe_pesan', 'Gagal');
            redirect('c_tpa');
        }
    }
```

2) Model Tambah Data TPA

```
function tambah_tpa($a) {
    $query = $this->db->query
            ('
                INSERT INTO data_TPA VALUES
                (
                  ' . $a['id_TPA'] . '
                  , ' . "'" . $a['nama'] . "'" . '
```

```
                    , ' . """ . $a['alamat_TPA'] . """ . '
                    , ' . """ . $a['kecamatan'] . """ . '

                )
        ');

    if ($query) {
        return TRUE;
    } else {
        return FALSE;
    }
}
```

b. Edit Data TPA

1) Controller Edit Data TPA

```
function edit() {
    $a = array
        (
        'id_TPA' => $_POST['id_TPA']
        , 'nama' => $_POST['nama']
        , 'alamat_TPA' => ($_POST['alamat_TPA'])
        , 'kecamatan' => $_POST['kecamatan']
    );

    if ($this->m_tpa->tpa_edit($a)) {
        $this->session->set_userdata('pesan_sistem', 'Selamat! edit TPS baru,
        SUKSES!');
        $this->session->set_userdata('tipe_pesan', 'Sukses');
```

```
            redirect('c_tpa');
        } else {
            $this->session->set_userdata('pesan_sistem', 'Maaf! edit TPS baru,
          GAGAL!
              Silahkan periksa dan coba kembali');
            $this->session->set_userdata('tipe_pesan', 'Gagal');
            redirect('c_tpa');
        }
    }
```

2) Model Edit Data Sopir

```
function tpa_edit_isi($a) {
    $query = $this->db->query
        ('
                SELECT
            id_TPA
            , nama
            , alamat_TPA
            , kecamatan

          FROM
            data_tpa
                WHERE
                  id_TPA = ' . $a
          );

    return $query->result_array();
  }
```

```
function tpa_edit($a) {
    $query = $this->db->query
        ('
                UPDATE data_tpa SET
            nama = ' . "'" . $a['nama'] . "'" . '
            , alamat_TPA = ' . "'" . $a['alamat_TPA'] . "'" . '
            , kecamatan = ' . "'" . $a['kecamatan'] . "'" . '

                WHERE
                  id_TPA = ' . $a['id_TPA']
        );

    if ($query) {
      return TRUE;
    } else {
      return FALSE;
    }
}
```

c. Hapus Data TPA

1) Controller Hapus Data TPA

```
function delete($a) {
    if ($this->m_tpa->hapus_tpa($a)) {
      $this->session->set_userdata('pesan_sistem', 'Selamat! tpa telah
      dihapus!');
      $this->session->set_userdata('tipe_pesan', 'Sukses');
      redirect('c_tpa');
```

```
        } else {
            $this->session->set_userdata('pesan_sistem', 'Maaf! TPS tidak terhapus!
                Silahkan periksa dan coba kembali');
            $this->session->set_userdata('tipe_pesan', 'Gagal');
            redirect('c_tpa');
        }
    }
```

2) Model Hapus Data TPA

```
function hapus_tpa($a) {
        $query = $this->db->query
            ('
                    DELETE FROM data_tpa
                    WHERE
                      id_TPA = ' . $a
                );

        if ($query) {
            return TRUE;
        } else {
            return FALSE;
        }
    }
```

4. Data Harga Bahan Bakar

a. Tambah Data Harga Bahan Bakar

1) Controller Tambah Data Harga

```
function add() {
```

```
        $a = array
            (
            'id_harga' => $_POST['id_harga']
            , 'bahan_bakar' => $_POST['bahan_bakar']
            , 'harga' => $_POST['harga']

        );

        if ($this->m_harga->tambah_harga($a)) {
            $this->session->set_userdata('pesan_sistem', 'Selamat! Penambahan
          harga baru, SUKSES!');
            $this->session->set_userdata('tipe_pesan', 'Sukses');
            redirect('c_harga');
        } else {
            $this->session->set_userdata('pesan_sistem', 'Maaf! Penambahan harga
          baru, GAGAL!
              Silahkan periksa dan coba kembali');
            $this->session->set_userdata('tipe_pesan', 'Gagal');
            redirect('c_harga');
        }
    }
```

3) Model Tambah Data Harga

```
function tambah_harga($a) {
    $query = $this->db->query
        ('
            INSERT INTO data_harga VALUES
            (
```

```
                    ' . $a['id_harga'] . '
                    , ' . "'" . $a['bahan_bakar'] . "'" . '
                    , ' . "'" . $a['harga'] . "'" . '
                )
        ');

    if ($query) {
        return TRUE;
    } else {
        return FALSE;
    }
}
```

b. Edit Data Harga

1) Controller Edit Data Harga

```
function edit() {
    $a = array
        (
        'id_harga' => $_POST['id_harga']
        , 'bahan_bakar' => $_POST['bahan_bakar']
        , 'harga' => ($_POST['harga'])

    );

    if ($this->m_harga->harga_edit($a)) {
        $this->session->set_userdata('pesan_sistem', 'Selamat! edit harga baru,
        SUKSES!');
        $this->session->set_userdata('tipe_pesan', 'Sukses');
```

```
            redirect('c_harga');
        } else {
            $this->session->set_userdata('pesan_sistem', 'Maaf! edit harga baru,
          GAGAL!
              Silahkan periksa dan coba kembali');
            $this->session->set_userdata('tipe_pesan', 'Gagal');
            redirect('c_harga');
        }
    }
```

2) Model Edit Data Harga

```
function harga_edit_isi($a) {
    $query = $this->db->query
        ('
                    SELECT
                id_harga
                , bahan_bakar
                , harga

            FROM
              data_harga
                  WHERE
                      id_harga = ' . $a
            );

    return $query->result_array();
}
```

```
function harga_edit($a) {
    $query = $this->db->query
        ('
                UPDATE data_harga SET
            bahan_bakar = ' . "'" . $a['bahan_bakar'] . "'" . '
            , harga = ' . "'" . $a['harga'] . "'" . '

                WHERE
                    id_harga = ' . $a['id_harga']
            );

    if ($query) {
        return TRUE;
    } else {
        return FALSE;
    }
}
```

c. Hapus Data Harga

1) Controller Hapus Data Harga

```
function delete($a) {
    if ($this->m_harga->hapus_harga($a)) {
        $this->session->set_userdata('pesan_sistem', 'Selamat! harga telah
        dihapus!');
        $this->session->set_userdata('tipe_pesan', 'Sukses');
        redirect('c_harga');
    } else {
        $this->session->set_userdata('pesan_sistem', 'Maaf! harga tidak terhapus!
```

```
            Silahkan periksa dan coba kembali');
        $this->session->set_userdata('tipe_pesan', 'Gagal');
        redirect('c_harga');
    }
}
```

2) Model Hapus Data Harga

```
function hapus_harga($a) {
    $query = $this->db->query
        ('
            DELETE FROM data_harga
            WHERE
                id_harga = ' . $a
        );

    if ($query) {
        return TRUE;
    } else {
        return FALSE;
    }
}
```

5. Lihat Info Jalan

a. Controller Lihat Info Jalan

```
function index() {
    $data['page'] = 'info';
    $data['breadcrumb'] = 'Info Jalan';
```

```
        $isi['combo'] = $this->m_info->get_titik();
        $data['content'] = $this->load->view('info', $isi, true);

        $this->load->view('gabungtemplate', $data);
    }

    function combo2() {
        $id = $_POST['combo'];
        $isi = $this->m_info->get_titik($id);
        $data = '<option value=""> -- Pilih Opsi -- </option>';
        for ($i = 0, $ln = count($isi); $i < $ln; $i++) {
            $data .= '<option value="' . $isi[$i]['latitude'] . ',' . $isi[$i]['longitude'] .
                    '">' . $isi[$i]['nama'] . '</option>';
        }
        echo $data;
    }
}
```

b. Model Lihat Info Jalan

```
function get_titik($id = NULL) {
    $query = $this->db->query('SELECT a.id_TPA, a.nama, b.latitude,
      b.longitude FROM data_tpa a, data_koordinat b ' .
          (($id != NULL) ? (' WHERE a.id_TPA <> ' . $id . ' AND a.id_TPA =
      b.id_TPA') : 'WHERE a.id_TPA = b.id_TPA'));
    return $query->result_array();
  }
```

6. Parameter Algoritma

a. Controller Parameter Algoritma

```
function index() {
    $data['page'] = 'algoritma';
    $data['breadcrumb'] = 'Algoritma';

    $isi['combo'] = $this->m_info->get_titik();
    $data['content'] = $this->load->view('algoritma', $isi, true);

    $this->load->view('gabungtemplate', $data);
  }

  function combo2() {
    $id = $_POST['combo'];
    $isi = $this->m_info->get_titik($id);
    $data = '<option value=""> -- Pilih Opsi -- </option>';
    for ($i = 0, $ln = count($isi); $i < $ln; $i++) {
      $data .= '<option value="' . $isi[$i]['latitude'] . ',' .
          $isi[$i]['longitude'] . '">' . $isi[$i]['nama'] . '</option>';
    }
    echo $data;
  }
}
```

7. Algoritma *Ant Colony System*

a. AntCo

```
function AntColony(mode) {
    var alfa = 0.1; // Pentingnya jalan sebelumnya
```

```
var beta = 2.0; // Pentingnya jangka waktu
var rho = 0.1; //  Tingkat  pembusukan  jalan feromon
var asymptoteFactor = 0.9; //  Ketajaman  dari  reward  sebagai solusi mendekati solusi terbaik

var pher = new Array();
var nextPher = new Array();
var prob = new Array();
var numAnts = 20;
var numWaves = 20;
for (var i = 0; i < numActive; ++i) {
   pher[i] = new Array();
   nextPher[i] = new Array();
}

for (var i = 0; i < numActive; ++i) {
   for (var j = 0; j < numActive; ++j) {

      pher[i][j] = 1;
      nextPher[i][j] = 0.0;
   }
}

var lastNode = 0;
var startNode = 0;
var numSteps = numActive - 1;
var numValidDests = numActive;
if (mode == 1) {
```

```
    lastNode = numActive - 1;
    numSteps = numActive - 2;
    numValidDests = numActive - 1;
}

/*perhitungan ACO sesuai jumlah semut dan siklus*/
for (var wave = 0; wave < numWaves; ++wave) {

    for (var ant = 0; ant < numAnts; ++ant) {
      var curr = startNode;
      var currDist = 0;
      for (var i = 0; i < numActive; ++i) {
        visited[i] = false;
      }
      currPath[0] = curr;
      for (var step = 0; step < numSteps; ++step) {
        visited[curr] = true;
        var cumProb = 0.0;
        /* probabiitas semut*/
        for (var next = 1; next < numValidDests; ++next)

          {
            if (!visited[next]) {

                prob[next] = Math.pow(pher[curr][next], alfa) *
Math.pow(dur[curr][next], 0.0 - beta);
                cumProb += prob[next];
```

```
        }
      }

      /*Mengambil bilangan random untuk menentukan kota yg
akan ditempuh selanjutnya*/

      var guess = Math.random() * cumProb;
      var nextI = -1;
      for (var next = 1; next < numValidDests; ++next)

      {

        if (!visited[next]) {
          nextI = next;
          guess -= prob[next];
          if (guess < 0) {

            nextI = next;
            break;
          }

        }
      }

      currDist += dur[curr][nextI];
      currPath[step + 1] = nextI;
      curr = nextI;
    }
```

```
            currPath[numSteps + 1] = lastNode;
            currDist += dur[curr][lastNode];
// k2-rewire:

            var lastStep = numActive;
            if (mode == 1) {

              lastStep = numActive - 1;
            }
            var changed = true;
            var i = 0;
            while (changed) {

              changed = false;
              for (; i < lastStep - 2 && !changed; ++i) {
                var cost =
                    dur[currPath[i + 1]][currPath[i + 2]];
                var revCost =
                    dur[currPath[i + 2]][currPath[i + 1]];
                var iCost =
                    dur[currPath[i]][currPath[i + 1]];
                var tmp, nowCost, newCost;
                for (var j = i + 2; j < lastStep && !changed;
                    ++j) {
                  nowCost = cost + iCost +
                      dur[currPath[j]][currPath[j + 1]];
                  newCost = revCost +
```

```
                    dur[currPath[i]][currPath[j]] + dur[currPath[i +
1]][currPath[j + 1]];
                if (nowCost > newCost) {

                  currDist += newCost - nowCost;
// Membalikkan ruas jalan terpisah

                  for (var k = 0; k < Math.floor((j -
                      i) / 2); ++k) {
                    tmp = currPath[i + 1 + k];
                    currPath[i + 1 + k] = currPath[j - k];
                    currPath[j - k] = tmp;
                  }

                  changed = true;
                  --i;
                }

                cost += dur[currPath[j]][currPath[j + 1]];
                revCost += dur[currPath[j + 1]][currPath[j]];
              }

            }
          }

          if (currDist < bestTrip) {
            bestPath = currPath;
            bestTrip = currDist;
```

```
                }

                /*Local Update And Decay Pheromone*/
                for (var i = 0; i <= numSteps; ++i) {

                    nextPher[currPath[i]][currPath[i + 1]] += (bestTrip -
asymptoteFactor * bestTrip) / (numAnts * (currDist - asymptoteFactor *
bestTrip));
                }
            }

            /*Apply global pheromone update */
            for (var i = 0; i < numActive; ++i) {

                for (var j = 0; j < numActive; ++j) {

                    pher[i][j] = pher[i][j] * (1.0 - rho) + rho *
                        nextPher[i][j];
                    nextPher[i][j] = 0.0;
                }

            }
        }
}
```

b. BtspSolver

```
function formatTime(seconds) {
  var days;
```

```
    var hours;
    var minutes;
    days = parseInt(seconds / (24*3600));
    seconds -= days * 24 * 3600;
    hours = parseInt(seconds / 3600);
    seconds -= hours * 3600;
    minutes = parseInt(seconds / 60);
    seconds -= minutes * 60;
    var ret = "";
    if (days > 0)
        ret += days + " hari ";
    if (days > 0 || hours > 0)
        ret += hours + " jam ";
    if (days > 0 || hours > 0 || minutes > 0)
        ret += minutes + " menit ";
    if (days == 0 && hours == 0)
        ret += seconds + " detik";
    return(ret);
}
function formatLength(meters) {
    var km = parseInt(meters / 1000);
    meters -= km * 1000;
    var ret = "";
    if (km > 0)
        ret += km + " km ";
    if (km < 10)
        ret += meters + " m";
    return(ret);
```

```
}

function formatLengthMiles(meters) {
  var sMeters = meters * 0.621371192;
  var miles = parseInt(sMeters / 1000);
  var commaMiles = parseInt((sMeters - miles * 1000 + 50) / 100);
  var ret = miles + "." + commaMiles + " mil";
  return(ret);
}
function formatDirections(gdir, mode) {
    var addr = tsp.getAddresses();
    var labels = tsp.getLabels();
    var order = tsp.getOrder();
    var retStr = "<table class='gebddir' border=0 cell-spacing=0>\n";
    var dragStr = "Drag to re-order stops:<br><ul class='unsortable'>";
    var retArr = new Array();
    for (var i = 0; i < gdir.legs.length; ++i) {
        var route = gdir.legs[i];
        var colour = "g";
        var number = i+1;
        retStr += "\t<tr><td>"
            +                    "<div                    class='centered-directions'><img
        src='http://localhost/RT1/assets/black"
            + number + ".png'></div></td>"
            + "<td><div class='centered-directions'>";
        var headerStr;
        if (labels[order[i]] != null && labels[order[i]] != "") {
            headerStr = labels[order[i]];
```

```
} else if (addr[order[i]] != null) {
    headerStr = addr[order[i]];
} else {
    var prevI = (i == 0) ? gdir.legs.length - 1 : i-1;
    var latLng = gdir.legs[prevI].end_location;
    headerStr = gdir.legs[i].start_location.toString();
}
dragStr += "<li id='" + i + "' class='ui-state-"
  + (i ? "default" : "disabled") + "'>"
  +        "<table        class='dragTable'><tr><td        class='left'><img
src='http://localhost/RT1/assets/black"
  + number + ".png' /></td><td class='middle'>" + headerStr + "</td><td
class='right'>"
  + (i ? "<button id='dragClose" + i + "' value='" + i + "'></button>" : "")
  + "</td></tr></table></li>";
if (i == 0) {
  dragStr += "</ul><ul id='sortable'>";
}

retStr += headerStr + "</div></td></tr>\n";
for (var j = 0; j < route.steps.length; ++j) {
    var classStr = "odd";
    if (j % 2 == 0) classStr = "even";
    retStr += "\t<tr class='text'><td class='" + classStr + "'></td>"
          + "<td class='" + classStr + "'>"
          + route.steps[j].instructions + "<div class='left-shift'>"
          + route.steps[j].distance.text + "</div></td></tr>\n";
}
```

```
}
dragStr += "</ul><ul class='unsortable'>";
if (mode == 0) {
    var headerStr;
    if (labels[order[0]] != null && labels[order[0]] != "") {
       headerStr = labels[order[0]];
    } else if (addr[order[0]] != null) {
       headerStr = addr[order[0]];
    } else {
       var prevI = gdir.legs.length - 1;
       var latLng = gdir.legs[prevI].end_location;
       headerStr = latLng.toString();
    }
    dragStr += "<li id='" + 0 + "' class='ui-state-disabled'>"
     +                "<table                class='dragTable'><tr><td><img
    src='http://localhost/RT1/assets/black"
     + 1 + ".png' /></td><td>" + headerStr
     + "</td></tr></table></li>";
    retStr += "\t<tr class='heading'><td class='heading'>"
       +                "<div                class='centered-directions'><img
    src='http://localhost/RT1/assets/black1.png'></div></td>"
       + "<td class='heading'>"
       + "<div class='centered-directions'>"
       + headerStr + "</div></td></tr>\n";
} else if (mode == 1) {
    var headerStr;
    if        (labels[order[gdir.legs.length]]        !=        null        &&
    labels[order[gdir.legs.length]] != "") {
```

```
            headerStr = labels[order[gdir.legs.length]];
        } else if (addr[order[gdir.legs.length]] == null) {
            var latLng = gdir.legs[gdir.legs.length - 1].end_location;
            headerStr = latLng.toString();
        } else {
            headerStr = addr[order[gdir.legs.length]];
        }
        dragStr += "<li id='" + gdir.legs.length + "' class='ui-state-disabled'>"
          +                  "<table                  class='dragTable'><tr><td><img
        src='http://localhost/RT1/assets/black"
          + (gdir.legs.length + 1) + ".png' /></td><td>"
          + headerStr + "</td></tr></table></li>";
        retStr += "\t<tr class='heading'><td class='heading'>"
            +                  "<div                  class='centered-directions'><img
        src='http://localhost/RT1/assets/black"
            + (gdir.legs.length + 1) + ".png'></div></td>"
            + "<td class='heading'>"
            + "<div class='centered-directions'>"
            + headerStr + "</div></td></tr>\n";
    }
    dragStr += "</ul>";
    retStr += "</table>";
    retArr[0] = dragStr;
    retArr[1] = retStr;
    return(retArr);
}
```