TUGAS AKHIR – TE141599

PENERAPAN *FIREFLY ALGORITHM* PADA PROSES PENENTUAN RUTE DAN PEMBERANGKATAN KENDARAAN DI PT PERTAMINA TBBM SURABAYA *GROUP*

Edo Rizaldi
NRP 2212 100 139

Dosen Pembimbing
Prof. Ir. Abdullah Alkaff, M.Sc., Ph.D
Mochammad Sahal, ST., M.Sc.

JURUSAN TEKNIK ELEKTRO
Fakultas Teknologi Industri
Institut Teknologi Sepuluh Nopember
Surabaya 2016

FINAL PROJECT – TE141599

APPLICATION OF FIREFLY ALGORITHM IN VEHICLE ROUTING AND DISPATCHING PROCESS AT PT. PERTAMINA TBBM SURABAYA GROUP

Edo Rizaldi
NRP 2212 100 139

Supervisor
Prof. Ir. Abdullah Alkaff, M.Sc., Ph.D
Mochammad Sahal, ST., M.Sc.

ELECTRICAL ENGINEERING DEPARTMENT
Faculty of Industrial Technology
Institut Teknologi Sepuluh Nopember
Surabaya 2016

PENERAPAN *FIREFLY ALGORITHM* PADA PROSES PENENTUAN RUTE DAN PEMBERANGKATAN KENDARAAN DI PT PERTAMINA TBBM SURABAYA *GROUP*

TUGAS AKHIR

Diajukan Guna Memenuhi Sebagian Persyaratan
Untuk Memperoleh Gelar Sarjana Teknik
Pada
Bidang Studi Teknik Sistem Pengaturan
Jurusan Teknik Elektro
Institut Teknologi Sepuluh Nopember

Menyetujui,

Dosen Pembimbing I,	Dosen Pembimbing II,
Prof. Ir. Abdullah Alkaff, M.Sc., Ph.D	**Mochammad Sahal, S.T., M.Sc.**
NIP. 195501231980031002	**NIP. 197011191998021002**

SURABAYA
JUNI 2016

PENERAPAN *FIREFLY ALGORITHM* PADA PROSES PENENTUAN RUTE DAN PEMBERANGKATAN KENDARAAN DI PT PERTAMINA TBBM SURABAYA *GROUP*

Nama : Edo Rizaldi
Pembimbing I : Prof. Ir. Abdullah Alkaff, M.Sc., Ph.D
Pembimbing II : Mochammad Sahal, S.T., M.Sc.

ABSTRAK

Proses *vehicle routing and dispatching* sangat menentukan ketersediaan Bahan Bakar Minyak (BBM) di tiap-tiap Stasiun Pengisian Bahan Bakar Umum (SPBU). Proses *vehicle routing* akan membuat rute distribusi tiap SPBU dengan mempertimbangkan permintaan (*demand*) BBM tiap SPBU, jarak tempuh dari depo ke SPBU, serta jarak tempuh antar SPBU. Rute-rute yang terbentuk akan digunakan sebagai data masukan untuk proses *vehicle dispatching*, yaitu proses pengalokasian kendaraan per nomor kendaraan untuk menjalankan tiap rute yang terbentuk. Kendaraan yang digunakan dalam proses distribusi terbagi dalam kelompok (*cluster*) sesuai dengan kapasitas maksimalnya yang menandakan permasalahan ini termasuk *capacitated vehicle routing problem*. Pada penelitian Tugas Akhir ini, digunakan *firefly algorithm* untuk menyelesaikan permasalahan *vehicle routing and dispatching* dengan mempertimbangkan deviasi jarak tempuh tiap kendaraan yang terpakai. Dari hasil perbandingan menggunakan metode *nearest neighbor*, didapatkan hasil bahwa *firefly algorithm* mampu menghasilkan jumlah rute dan nilai deviasi jarak tempuh yang lebih baik, yaitu 24 rute dengan deviasi jarak tempuh sebesar 1.677 jam.

Kata Kunci: *Capacitated Vehicle Routing Problem*, *Firefly Algorithm*, Proses Distribusi, *Vehicle Dispatching*.

APPLICATION OF FIREFLY ALGORITHM IN VEHICLE ROUTING AND DISPATCHING PROCESS AT PT. PERTAMINA TBBM SURABAYA GROUP

Name **: Edo Rizaldi**
Supervisor I **: Prof. Ir. Abdullah Alkaff, M.Sc., Ph.D**
Supervisor II **: Mochammad Sahal, S.T., M.Sc.**

ABSTRACT

Vehicle routing and dispatching process will determine the availability of fuel oil (BBM) in each of the Gas Station. Vehicle routing process will make the distribution route of each gas station by considering the request (demand) for each gas station, the distance from depot to gas stations, and the distance between gas stations. The routes that created by vehicle routing process will be used as input data for vehicle dispatching process, a process that allocate a number of vehicles to take each route. The vehicle that used for distribution process devided into groups (clusters) in accordance with their maximum capacity that indicates this problem into the capacitated vehicle routing problem. In this final assignment research, we used firefly algorithm to solve vehicle routing and dispatching process with considering the deviation of distance for each vehicle were used. From the comparison result with nearest neighbor method, firefly algorithm can produce a minimum number of routes and the deviation of distance, which is 24 routes and 1.677 hour for deviation of distance.

Key Word: Capacitated Vehicle Routing Problem, Distribution Process, Firefly Algorithm, Vehicle Dispatching.

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

BAB I
PENDAHULUAN

Tugas Akhir adalah suatu penelitian yang bersifat m*and*iri yang dilakukan sebagai persyaratan akademik untuk mendapatkan gelar sarjana teknik di Institut Teknologi Sepuluh Nopember (ITS) Surabaya. Topik yang dibahas dalam Tugas Akhir ini ialah mengenai Penerapan *Firefly Algorithm* Pada Proses *Vehicle Routing and Dispatching* di PT. Pertamina TBBM Surabaya *Group*.

Pada Bab ini membahas mengenai hal-hal yang mendahului pelaksanaan Tugas Akhir. Hal tersebut meliputi latar belakang, permasalahan, batasan masalah, tujuan, metodologi, sistematika penulisan, dan relevansi.

1.1 Latar Belakang

Distribusi merupakan domain penting dalam kehidupan kita sehari-hari. Hal itu dikarenakan distribusi sangat berpengaruh pada aspek sosial maupun ekonomi. Lebih jauh lagi, distribusi juga memegang peranan penting pada ketersediaan bahan baku. Penurunan kecil dalam jarak tempuh operasi logistik secara harian dapat berdampak secara langsung pada pengurangan biaya distribusi.

Kebutuhan akan pasokan Bahan Bakar Minyak (BBM) yang tepat waktu pada tiap-tiap Stasiun Pengisian Bahan Bakar Umum (SPBU) merupakan bagian dari permasalahan distribusi. PT Pertamina sebagai perusahaan utama dalam negeri yang memasok BBM dari Terminal BBM (TBBM) ke tiap-tiap SPBU, menyadari hal tersebut sebagai suatu permasalahan utama. Beberapa komponen yang sangat mempengaruhi kecepatan pemenuhan BBM ke tiap SPBU adalah ketersediaan mobil tangki, rute pendistribusian dan penjadwalan, yang dalam hal ini diartikan sebagai *Vehicle Routing*. Jika sistem *Vehicle Routing* itu tidak tertata dengan baik, maka sistem distribusi BBM tidak akan berjalan dengan baik dan mengakibatkan harga satuan BBM di masyarakat akan naik apabila terjadi kelangkaan BBM. Lebih jauh lagi, kejadian tersebut mengakibatkan turunnya citra Pertamina di mata masyarakat sehingga dikhawatirkan masyarakat akan berpindah membeli BBM ke perusahaan asing lainnya yang mampu memberikan pelayanan yang lebih baik.

PT Pertamina TBBM Surabaya Group, sebagai perwakilan PT Pertamina pusat dalam hal distribusi BBM di area Surabaya dan

sekitarnya, sebenarnya telah melakukan berbagai upaya untuk meningkatkan tingkat efisiensi sistem distribusi BBM. Salah satunya dengan menerapkan sistem sewa dalam pengelolaan mobil tangki pengangkut BBM. Dalam hal ini, PT. Pertamina menyewa mobil tangki dari pihak ketiga, untuk selanjutnya dioperasikan oleh PT. Pertamina melalui anak perusahaan, yaitu PT. Patra Niaga. Jumlah mobil yang disewa diusahakan seminimal mungkin sehingga pihak pertamina dapat memaksimalkan pemanfaatan mobil tangki. Namun upaya efisiensi ini tidak diikuti dengan sistem pengelolaan manajemen transportasi yang baik atau yang bisa disebut sebagai *vehicle dispatching*. Seringkali ditemui di lapangan SPBU mengalami keterlambatan pengiriman BBM yang mengakibatkan kekosongan stok di SPBU tertentu. Hal ini menimbulkan ketidakpuasan pihak SPBU atas pelayanan pengiriman BBM yang diberikan oleh Pertamina melalui PT. Patra Niaga. Selain itu, permasalahan jarak tempuh tiap kendaraan yang tidak merata juga menjadi masalah yang besar bagi pihak Pertamina. Hal itu dapat menyebabkan ketidakpuasan pada sopir truk.

Berdasarkan beberapa permasalahan tersebut, Pertamina TBBM Surabaya Group mutlak perlu melakukan pembenahan pada sistem penentuan rute (*vehicle routing*) dan penjadwalan kendaraan (*Vehicle dispatching*) sehingga dapat mengoptimalkan sistem distribusi yang ada agar performansi pengiriman BBM ke depan lebih baik. Selama ini, pihak Pertamina TBBM Surabaya Group masih belum mengoptimalkan penjadwalan pengiriman kendaraan yang mengacu pada ketersediaan kendaraan dan kecepatan habis SPBU (*throughput*). Seringkali ditemui kasus kendaraan tertentu menempuh jarak melebihi jarak tempuh normal. Hal itu menimbulkan tingkat kerusakan pada kendaraan tersebut menjadi tinggi. Namun di satu sisi, adapula kendaraan yang jarak tempuhnya jauh dibawah batas normal. Permasalahan itulah yang nantinya akan menjadi fokusan pada pengerjaan tugas akhir ini. Diharapkan, hasil dari penelitian ini mampu mengoptimalkan sistem distribusi BBM yang ada pada PT. Pertamina TBBM Surabaya Group.

1.2 Permasalahan

Berangkat dari latar belakang permasalahan yang telah disebutkan sebelumnya, fokus permasalahan yang dijadikan sumber pengerjaan tugas akhir ini adalah bagaimana menyelesaikan permasalahan penentuan rute dan penjadwalan pengiriman mobil tangki per nomor kendaraan dari TBBM (dalam hal ini bisa dikatakan depot) ke

tiap SPBU dengan mempertimbangkan deviasi jarak tempuh tiap-tiap kendaraan.

Pembuatan sistem tersebut juga mempertimbangkan tipe kendaraan distribusi yang ada, tipe permintaan (*demand*) tiap-tiap SPBU, serta sistem giliran (*shift*) yang ada pada pengiriman BBM dari depo ke SPBU.

1.3 Batasan Masalah

Dalam pengerjaan tugas akhir ini, permasalahan diatas dibatasi oleh hal-hal sebagai berikut:

1. SPBU yang digunakan dalam pengerjaan tugas akhir ini adalah SPBU wilayah Surabaya yang berjumlah 95.
2. Tipe BBM yang diangkut oleh truk adalah tipe Premium.
3. Data jarak tempuh dari Depo ke SPBU diperoleh dari pihak Pertamina TBBM Surabaya Group.
4. Data jarak tempuh SPBU antar SPBU dibuat acak dengan batasan jarak SPBU terdekat dari depot dan SPBU terjauh dari depot.
5. Satuan *demand* BBM adalah kiloliter (KL).
6. Jenis *demand* untuk tiap SPBU dibuat acak antara nilai 16 KL, 24 KL, dan 32 KL.
7. Pembagian *shift* difungsikan untuk mengelompokkan tiap-tiap SPBU kedalam slot pengiriman.
8. Jumlah *shift* yang digunakan ada 3. *Shift* 1 adalah jam 24.01-08.00. *Shift* 2 adalah 08.01-16.00. *Shift* 3 adalah 16.01-24.00.
9. Tipe kendaraan terbagi menjadi 4 tipe, yaitu tipe 16 KL, 24 KL, 32 KL, dan 40 KL.
10. Tiap tipe kendaraan memiliki jumlah unit yang berbeda-beda sesuai data yang dimiliki pihak Pertamina TBBM Surabaya Group.
11. Semua kendaraan diasumsikan tidak berada pada masa perawatan (*maintenance*).

1.4 Tujuan

Tujuan yang ingin dicapai dalam tugas akhir ini adalah menerapkan *Firefly Algorithm* pada permasalahan *vehicle routing and dispatching* sehingga dapat digunakan untuk mencari nilai minimum dari deviasi jarak tempuh tiap kendaraan yang terpakai.

1.5 Metodologi

Dalam penelitian Tugas Akhir ini diperlukan suatu tahapan yang merepresentasikan urutan yang harus dilaksanakan agar sesuai dengan tujuan penelitian. Tahapan tersebut ialah sebagai berikut:

1. Studi literatur

   Mempelajari buku-buku dan referensi-referensi yang berkaitan dengan metode yang akan digunakan, yaitu tentang optimasi pengiriman dan *Vehicle Routing Problem* (VRP). Buku yang dipelajari yaitu *Introduction to Operation Research* dari Sheldon M. Ross yang membahas secara detil mengenai berbagai macam bentuk optimasi dan *scheduling*. Sedangkan untuk metode, yaitu penerapan *firefly Algorithm*, digunakan buku *Nature-Inspired Optimization Algorithm*s dari Xin-She Yang.

   Referensi-referensi berupa jurnal dan tesis juga membantu dalam menyelesaikan metode yang akan dikerjakan pada tugas akhir ini. Jurnal yang digunakan yaitu berasal dari IEEE, researchgate, dan beberapa sumber yang dapat dipercaya. Tesis acuan dalam pembuatan tugas akhir ini adalah *Storage & Transportation Scheduling Model and Algorithm of Petroleum Products Based on Network* oleh Xiaoqiang Zhao dan Wei Li. Selain itu, untuk memperkuat pengetahuan mengenai penerapan *firefly Algorithm* dan *Vehicle Routing Problem*, penulis membaca tesis dari *Solution Methods for the Periodic Petrol Station Replenishment Problem* dari C. Triki.

2. Pengumpulan Data

   Pengumpulan data untuk tugas akhir ini diperoleh dengan observasi langsung ke Terminal BBM (TBBM) Perak melalui proses magang yang berlangsung dari tanggal 26 Januari 2016 sampai 15 Maret 2016. Data-data yang diambil meliputi tipe kendaraan, jumlah kendaraan total, serta jarak tempuh dari depo ke SPBU dan dari SPBU ke SPBU. Data-data tersebut ada yang bersifat umum sehingga dapat dikonsumsi oleh publik sebagai bahan riset atau pengembangan. Namun ada pula yang bersifat confidential atau tidak bisa dikonsumsi oleh publik.

3. Pemodelan Sistem

   Pemodelan sistem *Vehicle Routing and Dispatching* meliputi: penjadwalan kendaraan, batasan jumlah kunjungan kendaraan,

alokasi jumlah SPBU per *shift*, perancangan rute per kendaraan untuk meminimalkan deviasi jarak tempuh, serta perancangan *firefly algorithm.*

4. Simulasi dan Evaluasi
   Software yang digunakan dalam proses pembuatan dan evaluasi kinerja sistem adalah MatLAB.
5. Penulisan buku Tugas Akhir
   Tahap yang terakhir ialah penulisan laporan/buku Tugas Akhir. Penulisan dilakukan secara berbarengan dengan proses perancangan sistem.

1.6 Sistematika Penulisan

Tahap terakhir dari sebuah penelitian adalah penulisan laporan. Pada penulisan laporan/buku Tugas Akhir ini disusun berdasarkan 5 bab, di mana setiap bab berisi mengenai permasalahan dalam penelitian. Bab tersebut ialah sebagai berikut:

BAB I PENDAHULUAN
Berisi mengenai latar belakang, permasalahan, pembatasan masalah, tujuan, metodologi, sistematika penulisan, dan relevansi pembahasan tugas akhir ini.

BAB II DASAR TEORI
Berisi mengenai konsep dasar dan teori yang mendasari perancangan tugas akhir ini, meliputi teori dasar *Vehicle Routing* Problem, Dinamika *Vehicle Routing and Dispatching*, spesifikasi pengiriman BBM dari Depo ke SPBU, dan penggunaan *Firefly Algorithm* pada sistem *Vehicle Routing and Dispatching*.

BAB III PERANCANGAN SISTEM
Berisi mengenai spesifikasi sistem, identifikasi parameter, perancangan model matematis sistem *Routing and Dispatching* dengan *Firefly Algorithm* yang mengacu teori pada BAB II.

BAB IV IMPLEMENTASI SISTEM
Berisi prosedur pelaksanaan pengujian model matematis sistem *Vehicle Routing and Dispatching* menggunakan *Firefly Algorithm*.

BAB V PENGUJIAN DAN HASIL ANALISIS PENGUJIAN

Berisi analisa performa sistem dan perbandingan hasil sistem dengan mengubah nilai dari parameter sistem maupun menggunakan metode lain.

BAB VI KESIMPULAN

Berisi mengenai kesimpulan dari penelitian Tugas Akhir dan saran untuk dapat digunakan untuk pengembangan tugas akhir ini untuk lebih lanjut.

1.7 Relevansi

Tugas akhir ini diharapkan dapat menjadi referensi untuk pengembangan sistem pengiriman BBM dari Depo ke SPBU milik PT Pertamina TBBM Surabaya *Group* sehingga nantinya bisa mengoptimalkan sistem pengiriman yang ada.

BAB II
DASAR TEORI

Suatu penelitian memerlukan teori-teori yang sudah ada sebelumnya untuk dikaji lebih dalam memperkuat argumen penulis. Teori tersebut digunakan untuk membantu penulis dan sebagai dasar dalam membuat suatu penelitian.

Pada bab ini terdapat beberapa teori dasar yang menjadi landasan untuk merumuskan dan menyelesaikan masalah yang akan dibahas pada penelitian ini. Pada bagian awal, terdapat penjelasan mengenai teorema umum yang mendasari kasus *Vehicle Routing problem*, yaitu *Operation Research*, Pemrograman Linier, serta *Vehicle Dispathing* Pada bagian selanjutnya, membahas mengenai teori-teori pendukung, meliputi fungsi standar deviasi dan *Firefly Algorithm*.

2.1 *Operation Research* [1]

Menurut *Operation Research Society of Great Britain*, *operation research* adalah penerapan metode-metode ilmiah dalam masalah yang kompleks. Dapat pula diartikan sebagai suatu pengelolaan sistem manajemen yang besar, baik yang menyangkut manusia, mesin, bahan dan uang dalam skala industri, bisnis, maupun pemerintahan. Pendekatan ini menggabungkan dan menerapkan metode ilmiah yang sangat kompleks dalam suatu pengelolaan manajemen dengan menggunakan faktor-faktor produksi yang ada dan digunakan secara efektif dan efesien untuk membantu pengambilan keputusan dalam kebijakan suatu perusahaan. Definisi lain menurut *Operation Research Society of America* (ORSA), *operation research* berkaitan dengan pengambilan keputusan secara ilmiah dan bagaimana membuat suatu model yang baik dalam merancang dan menjalankan sistem yang melalui alokasi sumber daya yang terbatas. Inti dari beberapa kesimpulan di atas adalah bagaimana proses pengambilan keputusan yang optimal dengan menggunakan sumber daya yang ada.

2.2 *Linear Programming* [1], [2]

Linear Programming (LP) atau pemrograman linier merupakan suatu teknik riset operasional (*operation research technique*) yang telah dipergunakan secara luas dalam berbagai jenis masalah manajemen. *Linear Programming* memakai model matematis untuk memperoleh

hasil terbaik dari suatu fungsi tujuan dan kendala yang bersifat linier terhadap variabel keputusan. Komponen utama penyusun program linear antara lain:

1. Fungsi Tujuan (*Objective Function*)
   Fungsi tujuan adalah fungsi yang menggambarkan tujuan/sasaran dari dalam permasalahan *integer linear programming* yang berkaitan dengan pengaturan secara optimal sumber daya-sumber daya untuk mencapai hasil yang optimal.
2. Fungsi Pembatas (*Constraint*)
   Fungsi pembatas merupakan bentuk penyajian secara matematis batasan-batasan kapasitas yang tersedia yang akan dialokasikan secara optimal ke berbagai kegiatan.
3. Variabel Keputusan (*Decision Variable*)
   Variabel keputusan merupakan aspek dalam model yang dapat dikendalikan.

Bentuk umum atau standar dari *linear programming* adalah sebagai berikut:

$$Max \quad C_1X_1 + C_2X_2 + \ldots + C_nX_n \tag{2.1}$$

Subject to:

$$a_{11}X_{11} + a_{12}X_{12} + \ldots + a_{1n}X_n \leq b_1 \tag{2.2}$$

$$a_{21}X_{21} + a_{22}X_2 + \ldots + a_{2n}X_n \leq b_2 \tag{2.3}$$

$$a_{m1}X_1 + a_{m2}X_2 + \ldots + a_{mn}X_n \leq b_m \tag{2.4}$$

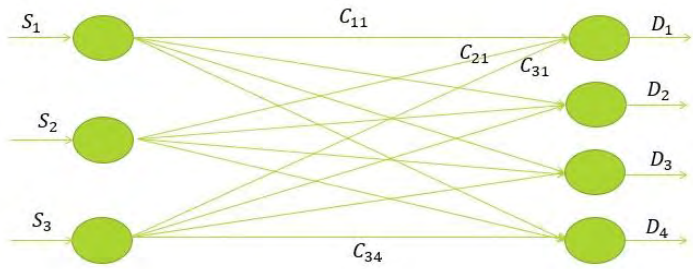
$$X_1, X_2, X_3, \ldots, X_n \geq 0 \tag{2.5}$$

2.3 Pemrograman Transportasi [1], [2]

Pemrograman Transportasi adalah variasi masalah dari *Linear Programming*. Fungsi tujuan pemrograman trasnportasi secara umum adalah meminimumkan biaya dari sejumlah pasokan ke sejumlah permintaan. Biaya total pengiriman adalah sebanding dengan jumlah barang yang dikirim dengan koefisien kesebandingan yang menyatakan biaya pengiriman per unit dari suatu pasokan tertentu ke permintaan tertentu. Dalam pemrograman transportasi, tiap node hanya dipasangkan untuk satu node tertentu.

Dalam pemrograman transportasi dikenal 2 tipe bentuk, yaitu bentuk standard dan bentuk tidak standard. Bentuk standard adalah ketika dimana jumlah pasokan samadengan jumlah permintaan. Bentuk standard bisa diselesaikan dengan metode simpleks. Bentuk tidak standard adalah ketika dimana jumlah pasokan tidak samadengan jumlah permintaan. Dalam bentuk tidak standard, diperlukan suatu analisa khusus sehingga pasokan tetap bisa memenuhi jumlah permintaan.

Representasi pemrograman transportasi dapat dilihat pada gambar 2.1.



Gambar 2.1 Representasi Pemrograman Transportasi

Model matematis dari permasalahan pemrograman transportasi dapat direpresentasikan sebagai berikut:

$$Min \quad \sum_i \sum_j C_{ij} X_{ij} \tag{2.6}$$

Subject to:

$$\sum_j X_{ij} = S_i \quad \forall i \tag{2.7}$$

$$\sum_j X_{ij} = S_i \quad \forall j \tag{2.8}$$

$$X_{ij} = D_j \tag{2.9}$$

$$X_{ij} \geq 0 \tag{2.10}$$

Fungsi tujuan dari permasalahan pemrograman transportasi adalah meminimumkan jarak atau biaya tempuh dari *node i* ke *node j*. Variabel jarak dan biaya tempuh direpresentasikan sebagai nilai dari C_{ij}. Pada batasan (2.7), jumlah total pasokan yang dilayani *node i* untuk semua *j* yang dikunjungi adalah sebanding dengan kapasitas dari *node i*. Batasan (2.8) adalah jumlah total pasokan yang dilayani *node j* untuk semua *i* yang dikunjungi adalah sebanding dengan kapasitas dari *node j*. Batasan (2.9) menjelaskan bahwa pasokan dari *node i* ke *j* adalah permintaan *node j.* Dan yang terakhir, batasan (2.10) menunjukkan pasokan dari *node i* ke *j* harus lebih besar atau sama dengan nol.

2.4 *Transhipment Problem* [3]

Transhipment Problem adalah suatu permasalahan pengiriman barang ke suatu tujuan menengah, dan kemudian menuju ke tujuan akhir. Salah satu fungsi dari permasalahan ini adalah pemindahan sarana distribusi (sebagai contoh dari transportasi laut menuju transportasi darat), yang umumnya disebut *transloading*. Tujuan yang lain adalah untuk menggabngkan beberapa pengiriman kecil menjadi satu pengiriman besar (*consolidation*). Atau juga bisa membagi satu pengiriman besar menjadi beberapa pengiriman kecil (*deconsolidation*). Dalam *Transhipment Problem* terdapat beberapa asumsi yang lazim dipakai, yaitu:

1. Sistem terdiri dari *m* sumber dan *n* tujuan dengan index *i*=1,2,…,*m* dan *j*=1,2,…,*n*
2. Terdiri dari satu macam *demand*
3. Kebutuhan *demand* dari beberapa tujuan sama dengan kapasitas total dari sumber
4. Sistem Transportasi berawal dari sumber dan memungkinkan mengunjungi beberapa tujuan (juga untuk kasus dari tujuan ke sumber)
5. Biaya transportasi tergantung dari besarnya *demand* yang dikirim

Fungsi objektif dari kasus ini adalah untuk meminimumkan waktu tempuh dari tiap rute trasportasi yang terbentuk. Hal ini bisa direpresentasikan dengan model matematis sebagai berikut:

$$\sum_{i=1}^{m}\sum_{j=1}^{n} t_{ij} x_{ij} \tag{2.11}$$

Notasi:

$t_{r,s}$ = Waktu tempuh dari *node r* ke *node s*
a_i = Barang yang tersedia di *node i*
b_{m+j} = *Demand* barang di *node* (*m+j*)
$x_{r,s}$ = Jumlah barang yang terkirim dari *node r* ke *node s*

Subject to:

$$x_{rs} \geq 0, \quad \forall r = 1,\ldots,n \quad s = 1,\ldots,n \tag{2.12}$$

$$\sum_{s=1}^{m+n} x_{i,s} - \sum_{r=1}^{m+n} x_{r,i} = a_i, \quad \forall i = 1,\ldots,m \tag{2.13}$$

$$\sum_{s=1}^{m+n} x_{r,m+j} - \sum_{s=1}^{m+n} x_{m+j,s} = b_{m+j}, \quad \forall i = 1,\ldots,m \tag{2.14}$$

$$\sum_{i=1}^{m} a_i = \sum_{j=1}^{n} b_{m+j} \tag{2.15}$$

Pada batasan (2.12), jumlah barang yang terkirim dari *node r* ke *node s* adalah harus lebih besar atau sama dengan nol. Pada batasan (2.13), jumlah barang yang tersedia pada *node i* adalah pengurangan dari barang yang terkirim dari *node r* ke *i* dengan barang yang terkirim dari *node i* ke *s*. Pada batasan (2.14) dan (2.15) menunjukkan bahwa *demand* yang diminta oleh *node m+j* harus sebanding dengan ketersediaan barang yang ada di *node i*.

2.5 *Vehicle Routing Problem*

2.5.1 Definisi *Vehicle Routing Problem* [4], [5]

Vehicle Routing Problem (VRP) adalah contoh kasus *Combinatorial Optimization* dan *Integer Linear Programming* yang bertujuan mencari kumpulan rute yang optimal untuk sejumlah kendaraan dengan kapasitas tertentu dari satu atau lebih depot untuk melayani konsumen. Komponen-komponen penyusun permasalahan VRP antara lain:

1. Jaringan jalan
   Jaringan jalan biasanya dideskripsikan dalam sebuah graf yang terdiri dari *edge* (rusuk) yang merepresentasikan bagian jalan

yang digunakan, dan *vertex* (simpul) yang merepresentasikan pelanggan atau depo

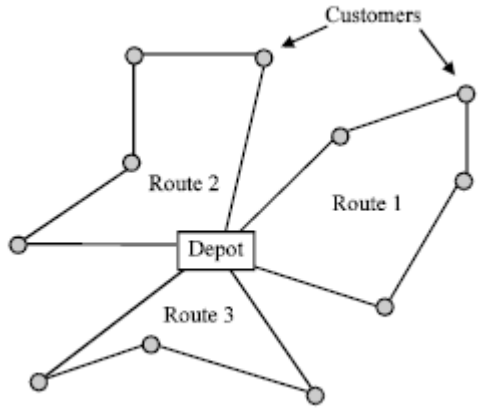
2. Pelanggan
   Dalam menyelesaikan masalah VRP, terlebih dahulu harus menetapkan lokasi semua pelanggan yang ada. Kemudian diperhatikan pula permintaan yang dibutuhkan pelanggan tersebut. Banyaknya permintaan yang dibutuhkan oleh pelanggan sangat mempengaruhi lamanya waktu bongkar-muat (loading-unloading) barang. Selain itu diperhatikan juga apakah ada rentang waktu yang diisyaratkan dalam melayani pelanggan-pelanggan tersebut.
3. Depo
   Depo merupakan tempat awal dan berakhirnya suatu kendaraan dalam proses pendistribusian barang. Lokasi depo menjadi penting dalam permasalahan VRP dikarenakan lokasi tersebut dapat mempengaruhi jarak tempuh kendaraan untuk menyuplai *demand* pelanggan. Semakin jauh lokasi depo dengan pelanggan, maka biaya distribusi atau waktu tempuh yang diperlukan juga semakin besar. Selain itu, perlu diketahui juga jumlah kendaraan yang ada pada depot serta jam operasional yang ditentukan pada depot. Tujuannya adalah untuk membatasi waktu kinerja kendaraan dalam proses distribusi.
4. Pengemudi
   Pengemudi memiliki kendala jam kerja harian, durasi maksimum perjalanan, dan tambahan jam lembur jika diperlukan.
5. Kendaraan
   Komponen yang perlu diperhatikan dari kendaraan yaitu jumlah dan kapasitas kendaraan yang digunakan. Kapasitas kendaraan tersebut membatasi permintaan pelanggan, artinya jumlah permintaan pelanggan tidak boleh melebihi kapasitas kendaraan yang digunakan. Kemudian ditentukan pula bahwa dalam satu rute hanya dilayani oleh satu kendaraan.

Tujuan yang ingin dicapai dalam VRP, diantaranya:

1. Meminimalkan ongkos perjalanan secara keseluruhan yang dipengaruhi oleh keseluruhan jarak yang ditempuh dan jumlah kendaraan yang digunakan
2. Meminimalkan jumlah kendaraan yang digunakan untuk melayani semua konsumen
3. Menyeimbangkan rute atau jarak tempuh tiap kendaraan

4. Meminimalkan keluhan pelanggan

Permasalahan VRP sendiri biasanya direpresentasikandalam sebuah *graph* atau grafik. Grafik tersebut menggambarkan permasalahan yang terjadi, yaitu berupa penyebaran konsumen yang harus dilayani oleh depot yang merupakan pusat pendistribusian berlangsung. *Vertex* $(V_0, \dots, V_n)$ merupakan titik yang menunjukkan posisi depot dan konsumen berada. *Vertex* depot ditunjukkan oleh V_0 dan *vertex* menunjukkan konsumen yang berjumlah n. Garis yang menghubungkan antar *vertex* disebut *edge*. *Edge* menunjukkan waktu, ongkos perjalanan ataupun jarak yang digunakan untuk perjalanan dari satu titik ke titik yang lain.



Gambar 2.2 Representasi *Vehicle Routing Problem*

Solusi VRP tersebut dianggap layak jika memenuhi beberapa syarat, antara lain:

1. Total rute yang terbentuk harus dapat melayani semua permintaan konsumen
2. Semua konsumen hanya bisa dikunjungi satu kali oleh tiap kendaraan
3. Semua rute harus dimulai dan selesai di depo
4. *Demand* yang ada tidak boleh melebihi dari kapasitas total pemasok (depo).

2.5.2 Fungsi Tujuan *Vehicle Routing Problem*

Fungsi tujuan dari permasalahan *Vehicle Routing Problem* adalah gabungan dari fungsi tujuan yang ada pada permasalahan pemrograman transportasi. Namun dalam perkembangannya, VRP memiliki suatu keunikan yang mana dia dapat menyuplai beberapa *demand* sekaligus. Kasus ini identik dengan permasalahan *Transhipment Problem*.

2.5.3 Jenis-jenis *Vehicle Routing Problem* [6]

Dilihat dari kendalanya, VRP dapat dikelompokkan menjadi beberapa varian, antara lain:

1. *Capacitated VRP* (CVRP)
   Faktor: Setiap kendaraan memiliki kapasitas yang terbatas
2. *VRP with Time Windows* (VRPTW)
   Faktor: Pelanggan harus dilayani dalam zona waktu tertentu
3. *Multiple Depot VRP* (MDVRP)
   Faktor: Distributor memiliki banyak depot
4. *VRP with Pick-Up and Delivering* (VRPPD)
   Faktor: Pelanggan diperbolehkan mengembalikan barang ke depot asal
5. *Split Delivery VRP* (SDVRP)
   Faktor: Pelanggan dilayani dengan kendaraan berbeda
6. *Stochastic VRP* (SVRP)
   Faktor: Munculnya *random value* (seperti jumlah pelanggan, jumlah permintaan, waktu perjalanan, dan waktu pelayanan)
7. *Periodic VRP*
   Faktor: Pendistribusian *Demand* hanya dilakukan di hari tertentu

2.5.4 Metode Penyelesaian *Vehicle Routing Problem* [6]

Pada dasarnya, terdapat 3 metode untuk menyelesaikan permasalahan VRP, antara lain:

1. Metode Eksak
   Pada metode eksak, dilakukan pendekatan dengan menghitung setiap solusi yang mungkin sehingga solusi terbaik dapat diperoleh. Contoh dari penyelesaian ini adalah metode *branch and bound* dan *branch and cut.*
2. Metode Heuristik
   Metode ini memberikan satu cara untuk menyelesaikan permasalahan optimasi yang lebih sulit dan waktu penyelesaian

yang lebih cepat daripada metode eksak. Contoh metode ini adalah metode *saving*, metode *nearest neighbor*, *multiroute improvement heuristic*, dll.

3. Metode Metaheuristik
   Algoritma heuristik modern atau yang lebih dikenal dengan meta-heuristik memecahkan masalah penjadwalan produksi dengan melakukan perbaikan mulai dengan satu atau lebih solusi awal. Solusi awal ini dapat dihasilkan secara acak, dapat pula dihasilkan berdasarkan hasil dari metode heuristik tertentu. Contoh dari metode metaheuristik adalah *genetic Algorithm*, *particle swarm Algorithm*, *firefly Algorithm*, dll.

2.6 *Bin Packing Problem* (BPP) [1], [7]

Bin Packing Problem adalah suatu kasus dimana beberapa obyek yang punya volume masing-masing, harus dimasukkan kedalam suatu *bin* atau kotak yang punya keterbatasan kapasitas. Fungsi objektif dari permasalah BPP adalah untuk meminimumkan jumlah penggunaan dari kotak tersebut. BPP sendiri adalah kasus dengan karakteristik mempunyai 1 distributor, dengan kapasitas yang homogen, dan nilai dari matriks biaya $c_{ij} = 0$ untuk setiap rute dari pelanggan *i* ke pelanggan *j*. Dalam kasus BPP, diberikan sejumlah *n* barang dan *m* kotak dengan:

$w_j =$ berat dari barang j

$c =$ Kapasitas dari tiap kotak

Masukkan setiap barang kedalam 1 kotak sehingga berat total dari semua barang yang masuk kotak tersebut tidak melebihi dari nilai *C* tiap kotak. Selain itu, permasalahan BPP selalu difungsikan untuk meminimumkan jumlah penggunaan kotak. Model matematis permasalahan ini adalah sebagai berikut:

$$Min \quad Z = \sum_{i=1}^{n} y_i \tag{2.16}$$

Subject to:

$$\sum_{i=1}^{n} w_j x_{ij} \le c y_i, \quad i \in N = \{1, \ldots, n\} \tag{2.17}$$

$$\sum_{i=1}^{n} x_{ij} = 1, \quad j \in N \tag{2.18}$$

$$y_i = \begin{cases} 0 \\ 1 \end{cases} \quad i \in N \tag{2.19}$$

$$x_{ij} = \begin{cases} 0 \\ 1 \end{cases} \quad i \in N, j \in N \tag{2.20}$$

Dimana:

$$y_i = \begin{cases} 1 & \text{Jika kotak } i \text{ dipakai} \\ 0 & \text{Jika kotak } i \text{ tidak dipakai} \end{cases} \tag{2.21}$$

$$x_{ij} = \begin{cases} 1 & \text{Jika barang } j \text{ dimasukkan ke kotak } i \\ 0 & \text{Jika barang } j \text{ tidak dimasukkan ke kotak } i \end{cases} \tag{2.22}$$

Dengan asumsi biasa, bahwa besarnya w_j adalah integer bernilai positif. Apabila nilai w_j tidak integer positif, maka nilai *c* bisa diganti dengan $\lfloor c \rfloor$. Jika sebuah ada permasalahan diluar batas itu, maka solusinya adalah *infeasible.*

2.7 *Capacitated Vehicle Routing Problem* (CVRP) [5], [8]

Kasus CVRP adalah suatu permasalahan gabungan dari kasus *Vehicle Routing Problem* dengan *Bin Packing Problem (BPP).*

CVRP adalah varian yang paling umum dan dasar dari kasus *vehicle routing problem*. Dalam varian ini, kendaraan berkapasitas tetap *m* (m ≥ 1) harus melayani *demand* pelanggan, dengan tambahan batasan bahwa tiap kendaraan ini punya kapasitas tertentu. Kendaraan ini dapat memuat suatu barang (*demand* dari pelanggan) sampai dengan kapasitas maksimalnya.

Dalam kasus *Homogeneous* CVRP, tiap kendaraan yang ada memiliki kapasitas yang seragam, yaitu *Q*. perbedaan satu-satunya dalam definisi umum dari VRP adalah tiap rute terdefinisi *feasible* apabila total *demand* dari semua pelanggan dalam rute *R* tidak melebihi kapasitas kendaraan *Q*. Bentuk ini dapat ditulis dengan model matematis:

$$\left(\sum_{j=1}^{K} d_j \right) \leq Q \tag{2.23}$$

Dimana:

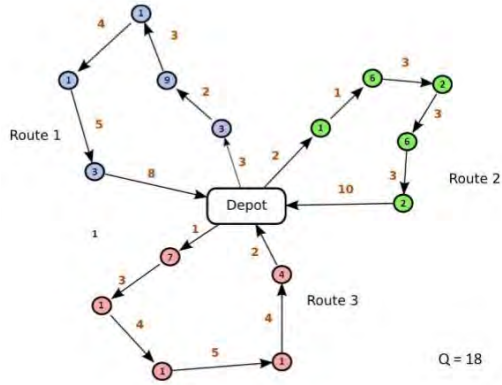
d_j = *demand* dari pelanggan v_j

Tentu saja dengan batasan bahwa *demand* setiap pelanggan tidak pernah lebih besar dari kapasitas kendaraan.

$$d_j \leq Q, \quad (1 \leq j \leq n) \tag{2.24}$$

Selain itu, total *demand* dari semua pelanggan tidak boleh lebih besar dari kapasitas total semua kendaraan.

$$\left(\sum_{j=1}^{n} d_j \right) \leq m * Q \tag{2.25}$$



Gambar 2.3 Representasi *Capacitated Vehicle Routing Problem*

Dalam gambar 2.3, diilustrasikan suatu permasalahan *Capacitated Vehicle Routing Problem* yang terdiri dari 1 depo, 15 pelanggan, 3 kendaraan, serta *demand* pelanggan (disediakan didalam

node). Jarak antar node diberikan ditiap garisnya. Kapasitas tiap kendaraan adalah Q=18. Dari ketiga kendaraan tersebut, dihasilkan 3 rute dengan kendaraan 1 (biru) menyuplai 17 *demand*, kendaraan 2 (pink) menyuplai 17 *demand*, dan kendaraan 3 (hijau) menyplai 14 *demand*.

Kasus lain dari permasalahan CVRP adalah *Heterogeneous* CVRP (atau *Mixed Fleet* CVRP). *Heterogeneous* CVRP mempunyai beberapa tipe kendaraan yang tiap tipenya memiliki kapasitas tertentu. Sifat dari CVRP *Heterogeneous Fleet* inilah yang mendasari terciptanya konsep *vehicle routing and dispatching*. Model permasalahan CVRP *Heterogeneous Fleet* bisa dituliskan sebagai berikut:

$$Q_f \left(1 \le f \le m\right) \tag{2.26}$$

Sebuah rute dalam permasalahan ini didefinisikan sebagai (*R, f*), dimana *R* adalah vektor rute dan *f* adalah tipe kendaraan. Batasan yang dipakai dalam permasalahan ini adalah serupa dengan batasan yang ada dalam tipe *homogeneous* CVRP dipakai untuk maksimum *demand* tiap rute. Bahwa maksimum *demand* per rute adalah samadengan kapasitas maksimal kendaraan yang menyuplainya. Sedangkan total *demand* adalah sebanding dengan total kapasitas dar tiap tipe kendaraan. Dalam beberapa penelitian, tipe kendaraan bisa dilambangkan juga dengan biaya tiap kendaraan F_k . Contoh kasusnya adalah dalam proses sewa mobil.

2.8 Standar Deviasi [9]

Standar deviasi adalah suatu ukuran yang digunakan untuk menganalisa suatu sebaran nilai dari sekumpulan data tertentu. Nilai standar deviasi yang rendah menunjukkan bahwa tiap nilai yang ada pada sekumpulan data tersebut hampir mendekati nilai rata-ratanya. Sedangkan apabila nilai standar deviasinya rendah, maka hal itu dapat menunjukkan bahwa tiap nilai yang ada pada kumpulan data tersebut berada pada kisaran nilai yang jauh dari nilai rata-ratanya. Fungsi matematis dari standar deviasi adalah sebagai berikut:

$$\sigma = \sqrt{\mathrm{E}\left[\left(x - \mu\right)^2\right]} \tag{2.27}$$

Dimana:

σ	= Standar deviasi
X	= Variabel acak
μ	= Nilai rata-rata
E	= Operator matematis yang mengeluarkan nilai rata-rata dari X

2.9 Metode *Sorting* [9], [10]

Sorting atau pengurutan data adalah proses yang sering dan harus dilakukan dalam pengolahan data. *Sort* dalam hal ini diartikan mengurutkan data yang berada dalam suatu tempat penyimpanan, dengan urutan tertentu baik urut menaik (*ascending*) dari nilai terkecil sampai dengan nilai terbesar, atau urut menurun (*descending*) dari nilai terbesar sampai dengan nilai terkecil.

Terdapat dua macam pengurutan pada kasus *sorting*, yaitu:

1. Pengurutan internal (*internal sort*)
   Pengurutan terhadap sekumpulan data yang disimpan dalam media internal komputer yang dapat diakses setiap elemennya secara langsung. Dapat dikatakan sebagai pengurutan tabel
2. Pengurutan eksternal (*external sort*)
   Pengurutan data yang disimpan dalam memori sekunder, biasanya data bervolume besar sehingga tidak mampu untuk dimuat semuanya dalam memori.

2.9.1 *Selection Sort* [9]

Algoritma *Selection sort* bertujuan untuk memilih elemen maksimum/minimum dari suatu data *array*, lalu menempatkan elemen maksimum/minimum itu pada awal atau akhir *array* (tergantung pada urutannya ascending/descending). Selanjutnya elemen tersebut tidak disertakan pada proses selanjutnya. Dalam algoritma *selection sort*, proses pemilihan nilai maksimum/minimum dilakukan pada setiap pass/data. Jika *array* berukuran N, maka jumlah *pass* adalah N-1. Terdapat dua pendekatan dalam metode pengurutan dengan *Selection Sort* :

1. Algoritma pengurutan maksimum (*maximum selection sort*),
   Memilih elemen maksimum sebagai basis pengurutan.
2. Algoritma pengurutan minimum (*minimum selection sort*)
   Memilih elemen minimum sebagai basis pengurutan

Tabel 2.1 adalah *Pseudocode* dari prosedur *Minimum Selection Sort.*

Tabel 2.1 *Pseudocode Selection Sort*

```
Begin
     Input   : Array of A with n elements
     Output  : Permutation of A such that
      A[1] ≤ A[2] ≤ A[3] ≤ … ≤ A[n]
     n       : numData
     Procedure Selection Sort:
     If n ≤ 1
      Return
     Else
      For i=1: n-1
            If (A[i]>A[n])
                  Swap (A[i], A[n])
            End
            Selection Sort (A[ ], n-1)
      i=i+1
     end
end
```

2.9.2 *Insertion Sort* [10]

Insertion sort adalah sebuah algoritma pengurutan yang membandingkan dua elemen data pertama, mengurutkannya, kemudian mengecek elemen data berikutnya satu persatu dan membandingkannya dengan elemen data yang telah diurutkan. Karena algoritma ini bekerja dengan membandingkan elemen-elemen data yang akan diurutkan, algoritma ini termasuk pula dalam *comparison-based sort*.

Ide dasar dari algoritma *Insertion Sort* ini adalah mencari tempat yang "tepat" untuk setiap elemen *array*, dengan cara *sequential search*. Proses ini kemudian menyisipkan sebuah elemen *array* yang diproses ke tempat yang seharusnya. Proses dilakukan sebanyak N-1 tahapan (dalam *sorting* disebut sebagai "*pass*"), dengan indeks dimulai dari 0.

Proses pengurutan dengan menggunakan algoritma *Insertion Sort* dilakukan dengan cara membandingkan data ke-i (dimana i dimulai dari data ke-2 sampai dengan data terakhir) dengan data berikutnya. Jika

ditemukan data yang lebih kecil maka data tersebut disisipkan ke depan sesuai dengan posisi yang seharusnya

Tabel 2.2 adalah *Pseudocode* dari prosedur Insertion Sort.

Tabel 2.2 *Pseudocode Insertion Sort*

```
Begin

     Input  : Array A[1…n] of n elements
     Output       :      A[1…n]sorted      in
nondecreasing order
     n      : numData
     Procedure Insertion Sort:
     For 1=2: n
      X=A[i]
      J=i-1
      While (j>0) and (A[j]>x)
            A[j+1]=A[j]
            J=j-1
      End
      A[j+1]=x
     End
End
```

2.10 *Sequential Insertion Algorithm* [9], [11]

Dalam membentuk solusi VRP terdapat dua macam pendekatan, yaitu menggabungkan rute yang ada dengan menggunakan kriteria penghematan (*savings criterion*) dan mencoba secara berurutan memasukkan pelanggan dalam rute kendaraan dengan menggunakan kriteria biaya penyisipan (*cost insertion*). Metode yang kedua telah terbukti menjadi metode yang populer digunakan untuk menyelesaikan permasalahan rute dan penjadwalan kendaraan.

Metode *sequential insertion* pada dasarnya melakukan pembentukan rute dengan melakukan penyisipan suatu node/pelanggan yang belum ditugaskan atau dikunjungi ke dalam suatu rute. Pada tabel 2.3 dijelaskan mengenai *pseudocode* dari metode tersebut.

Adapun langkah-langkah yang dilakukan dalam algoritma *sequential insertion* dalam kasus VRP adalah sebagai berikut:

1. Langkah 1

   Pilih satu titik awal sebagai titik awal (depot) yang dipilih berdasarkan jarak terpendek dari depot menuju ke pelanggan pertama kembali ke depot .Selanjutnya ke langkah 2.

2. Langkah 2

   Hitung jarak tempuh yang dilalui depot ke tiap pelanggan dalam mengirimkan barang ke tiap pelanggan, lanjut ke langkah 3.

3. Langkah 3

   Hitung sisa kapasitas kendaraan, jika sisa kapasitas kendaraan memenuhi untuk mengirimkan barang sesuai permintaan pelanggan maka lanjut ke langkah 4, jika tidak lanjut ke langkah 9.

4. Langkah 4

   Jika telah memasuki pelanggan ke-2 atau seterusnya maka lanjut ke langkah 5, jika tidak lanjut ke langkah 6.

5. Langkah 5

   Sisipkan pelanggan berikutnya ke dalam urutan rute yang telah terbentuk, lanjut ke langkah 6.

6. Langkah 6

   Pilih pelanggan yang memiliki jarak paling pendek, lanjut langkah 7.

7. Langkah 7

   Hitung jarak tur (perjalanan), dan list rute pelanggan yang telah dilayani.Lanjut ke langkah 8.

8. Langkah 8

   Jika permintaan barang yang akan dikirimkan ke pelanggan belum semua terpenuhi maka lanjut ke langkah 2, jika sudah lanjut ke langkah 10.

9. Langkah 9

   Kembali ke depot, buat tur baru, kembali ke langkah 2.

10. Langkah 10

    Program berakhir apabila semua pelanggan telah dikunjungi

Tabel 2.3 *Pseudocode Sequential Insertion*

```
Begin
     Input  : ListInsert (L, A, O)
     Output : L.insert (A)
If L.Size () = maxlistsize then
     L.sortByName ()
```

```
     %Look for pairs, move unmatched to start
     of list
     (i, o) ← (1, 1)
     While i ≤ maxlistsize do
            If i+1 ≤ maxlistsize and is Pair
            (L[i], L[i+1]) then
                  O.insert ((L[i], L[i+1]))
                  i ← i+2
                  else
                  L[o] ← L[i]
                  i ← i+1
                  o ← o+1
            endif
            o ← 0-1
      Endwhile
      %Put  unmatched  entries  to  temporary
file
      If o > 0 then
            L.PutToTempFile(1,o)
            L ← empty list
     Endif
Endif
End
```

2.11 *Firefly Algorithm* [12], [13], [14]

Firefly Algorithm atau algoritma kunang-kunang merupakan salah satu algoritma pada bidang Artificial Intelligence atau kecerdasan buatan. *Firefly Algorithm* dibentuk dan dikembangkan oleh Dr. Xin She Yang di Cambridge University pada tahun 2007 untuk memecahkan masalah optimasi. *Firefly Algorithm* merupakan algoritma yang didasarkan pada perilaku kunang-kunang. Kunang-kunang pada umumnya menghasilkan sinar dalam durasi yang pendek dan memiliki ritme tertentu. Terdapat dua fungsi dasar sinar kunang-kunang, yaitu untuk menarik perhatian kunang-kunang yang lain atau bertahan dari serangan pemangsa.

Fungsi tambahan lain pada sinar kunang-kunang sebagai mekanisme t*and*a peringatan bahaya. Ritme dari sinar dan durasi waktu

penyinaran dari kunang-kunang merupakan bagian sinyal dari sistem kunang-kunang yang dibawa oleh kunang-kunang baik yang jantan maupun betina. *Firefly Algorithm* dibentuk dari beberapa karakteristik dari kebiasaan dan pola hidup kunang-kunang. Beberapa aturan yang diadopsi dan disintesis untuk membentuk *Firefly Algorithm* adalah:

1. Semua kunang-kunang bersifat unisex, dimana kunang-kunang tertarik antar satu dengan yang lain tanpa menghiraukan jenis kelamin.
2. Daya pikat dari kunang-kunang bersifat proporsional, bergantung pada tingkat terang dari sinar yang dipancarkan. Daya pikat kunang-kunang semakin berkurang pada saat jarak semakin bertambah. Jika diantara kunang-kunang tidak ada yang bersinar lebih terang, maka kunang-kunang bergerak secara *random.*
3. Terang yang ditimbulkan kunang-kunang dipengaruhi atau ditentukan oleh susunan dari fungsi objektif atau fungsi tujuan.
   Berikut adalah *pseudo-code* dari *firefly algorithm*
   Tabel 2.4 menjelaskan mengenai *pseudocode firefly algorithm.*

Tabel 2.4 *Pseudocode Firefly Algorithm*

```
begin
Objective function f(x),x = (x1,...,xd)T
Generate initial population of fireflies x_i
     (i = 1,2,...,n)
Light intensity I_i at xi is determined by
f(x_i)
     Define light absorption coefficient γ
While (t <MaxGeneration)
     for i = 1: n all n fireflies
            for j = 1: i all n fireflies
            if (I_j> I_i)
                  Move firefly i towards j
            end if
     %Attractiveness varies with distance
      r via exp[-γr2]
     %Evaluate new solutions and update light
     intensity
            end for j
      end for i
```

```
     %Rank the fireflies and find the current
     best
end while
     %Postprocess results and visualization
end
```

2.11.1 Keatraktifan *Firefly Algorithm*

Ada dua hal yang berkaitan dan sangat penting dalam *Firefly Algorithm*, yaitu intensitas cahaya dan fungsi keatraktifan. Dalam hal ini, kita asumsikan bahwa keatraktifan dipengaruhi oleh tingkat intensitas cahaya. Untuk kasus yang paling sederhana, contohnya adalah kasus optimasi maksimum, tingkat intensitas cahaya pada sebuah kunang-kunang x dapat dilihat sebagai

$$I(x) = f(x) \tag{2.28}$$

Dengan nilai I merupakan tingkat intensitas cahaya pada x kunang-kunang yang seb*and*ing terhadap solusi fungsi tujuan f(x) permasalahan yang dihadapi. Keatraktifan *firefly* dilambangkan dengan lambang β. Nilai β bersifat relatif karena intensitas cahaya dari tiap-tiap *firefly* berbeda-beda. Dan intensitas cahaya sebanding dengan keatraktifan *firefly*. Dengan demikian, hasil penelitian berbeda-beda tergantung dari jarak antara kunang-kunang yang satu dengan yang lainnya. Selain itu, intensitas cahaya menurun dilihat dari sumbernya dikarenakan terserap oleh media. Dalam hal ini, penyerapan cahaya disimbolkan dengan lambang ɣ. Fungsi keatraktifan *firefly* ialah sebagai berikut:

$$\beta(r) = \beta_0 * \exp\left(-\gamma * r^m\right), \quad m \geq 1 \tag{2.29}$$

Dimana:

$\beta(r)$: Nilai keatraktifan pada jarak r

$\beta(0)$: Nilai keatraktidan pada jarak 0

γ : Konstanta penyerapan cahaya

2.11.2 Jarak Antar *Firefly*

Jarak antar *firefly* i dan j pada lokasi x, xi, dan xj dapat ditentukan ketika dilakukannya peletakan titik dimana *firefly* tersebut tersebar secara *random* dalam diagram kartesius dengan rumus:

$$r_{ij} = \sqrt{\left(x_i - x_j\right)^2 + \left(y_i - y_j\right)^2} \tag{2.30}$$

Dimana selisih dari koordinat lokasi kunang-kunang i terhadap kunang-kunang j merupakan jarak diantara keduanya (r_{ij}).

2.11.3 Pergerakan *Firefly*

Pergerakan *firefly* i yang bergerak menuju tingkat intensitas cahaya terbaik dapat dilihat dari persamaan berikut:

$$x_i = x_i + \beta_0 * \exp\left(-\gamma * r_{ij}{}^2\right) * \left(x_j - x_i\right) + a * \left(rand - \frac{1}{2}\right) \tag{2.31}$$

Dimana variabel awal xi menunjukkan posisi awal kunang-kunang yang berada pada lokasi x. Kemudian persamaan kedua yang terdiri dari variabel $\beta(0)$=1, adalah nilai keatraktifan awal *firefly*. Variabel (exp) yaitu bilangan eksponensial, variabel γ=1 merupakan nilai untuk tingkat penyerapan pada lingkungan sekitar *firefly*, yaitu udara. Semua variabel pada persamaan kedua tersebut diberikan dari fungsi keatraktifan *firefly*, yang mana menentukan tingkat kecerahan, Kemudian fungsi pergerakan *firefly* diberikan secara *random* yang nilainya berkisar antara 0.1 Variabel α yang nilainya berkisar antara 0-1 biasa diberikan di nilai 0.2.

2.12 *Nearest Neighbor* [9], [11]

Metode *Nearest Neighbor* pertama kali diperkenalkan pada tahun 1983 dan merupakan metode yang sangat sederhana. Pada setiap iterasinya, dilakukan pencarian pelanggan terdekat dengan pelanggan yang terakhir untuk ditambahkan pada akhir rute tersebut. Rute baru dimulai dengan metode yang serupa, jika tidak terdapat posisi yang fisibel untuk menempatkan pelanggan baru karena kendala kapasitas atau *time windows*. Cara kerja metode ini adalah pertama-tama, semua

rute kendaraan masih kosong. Dimulai dari rute kendaraan pertama, metode ini memasukkan *(insert*) satu persatu pelanggan terdekat (*Nearest Neighbor*) yang belum dikunjungi ke dalam rute, selama memasukkan pelanggan tersebut ke dalam rute kendaraan tidak melanggar batasan kapasitas maksimum kendaraan tersebut. Kemudian proses yang serupa juga dilakukan untuk kendaraan-kendaraan berikutnya, sampai semua kendaraan telah penuh atau semua pelanggan telah dikunjungi.

Langkah-langkah metode *Nearest Neighbor* adalah sebagai berikut:

1. Langkah 1

   Berawal dari depot, kemudian mencari pelanggan yang belum dikunjungi yang memiliki jarak terpendek dari depot sebagai lokasi pertama.

2. Langkah 2

   Ke pelanggan lain yang memiliki jarak terdekat dari pelanggan yang terpilih sebelumnya dan jumlah pengiriman tidak melebihi kapasitas kendaraan.

a. Apabila ada pelanggan yang terpilih sebagai pelanggan berikutnya dan terdapat sisa kapasitas kendaraan, kembali ke langkah (2).
b. Bila kendaraan tidak memiliki sisa kapasitas, kembali ke langkah (1).
c. Bila tidak ada lokasi yang terpilih karena jumlah pengiriman melebihi kapasitas kendaraan, maka kembali ke langkah (1). Dimulai lagi dari depot dan mengunjungi pelanggan yang belum dikunjungi yang memiliki jarak terdekat.

3. Langkah 3

   Bila semua pelanggan telah dikunjungi tepat satu kali maka algoritma berakhir.

Tabel 2.5 menjelaskan mengenai *pseudocode* dari *nearest neighbor.*

Tabel 2.5 *Pseudocode Nearest Neighbor Method*

Input : E = {e1, e2, …, en} Set of entities to be clustered K (number of clusters) maxIters (limit of iterations) Output : C = {c1, c2, …, ck} (set of cluster centroids) L = {l(e) | e=1, 2, …, n} (set of cluster labels of E)

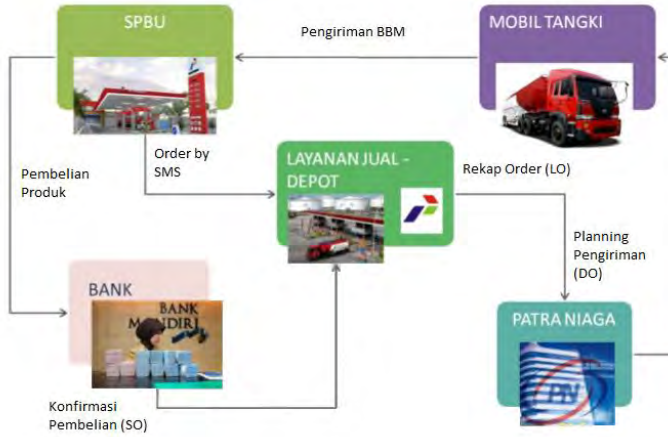
```
Foreach c1 ϵ C do
        Ci ←ej ϵ E
End
Foreach ei ϵ E do
        L(ei) ←argminDistance (ei, cj)j ϵ {1…k}
End

Changed ← False;
Iter ← 0;
Repeat
        Foreach ci ϵ C do
          UpdateCluster (ci);
        End
        Foreach ei ϵ E do
          minDist ← argminDistance (ei,cj)j ϵ {1…k};
          If minDist ≠ (ei) then
                  L(ei) ← minDist;
                  Changed ← True;
          End
        End
        Iter++;
Until changed = true and iter ≤ MaxIters;
```

BAB III
PERANCANGAN SISTEM

3.1 Proses Distribusi BBM [15]

Secara umum, aktivitas pemesanan dan distribusi BBM ke SPBU dapat digambarkan dalam ilustrasi berikut :



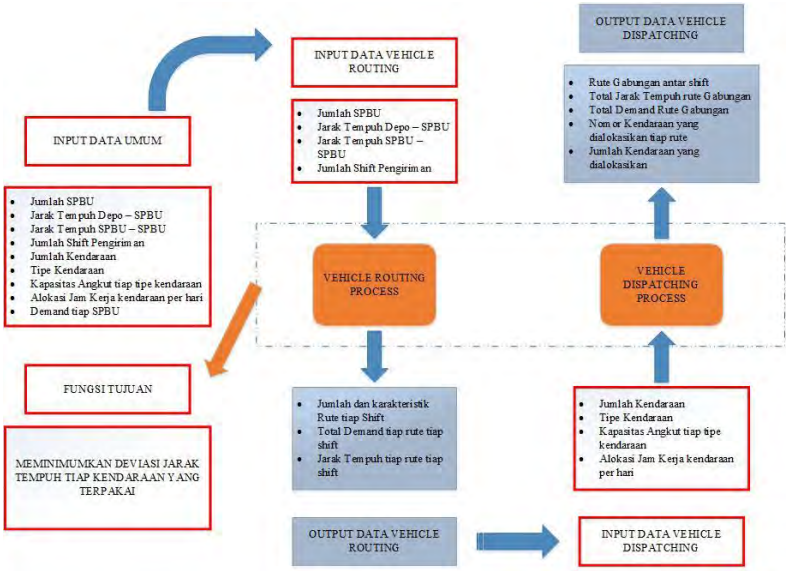
Gambar 3.1 Sistem Pemesanan dan Distribusi BBM

Penjabaran proses pemesanan dan distribusi BBM di PT Pertamina TBBM Surabaya Group adalah sebagai berikut :

1. Pihak SPBU melakukan pelaporan stok serta permintaan BBM (Premium dan Solar) melalui SMS 1 hari sebelum pengiriman BBM dilakukan
2. SMS dan SPBU akan diterima oleh server Pertamina. Selanjutnya dari data stok dan permintaan BBM tersebut, pihak Layanan Jual Depot akan menentukan jumlah BBM yang akan dikirim untuk masing-masing SPBU. Penentuan jumlah pengiriman BBM ke masing-masing SPBU didasarkan pada : sisa stok SPBU, sisa *Loading Order* (LO) yang dimiliki SPBU serta *Daily Objective Truput* (DOT) yang diizinkan. DOT adalah angka rata-rata penyaluran harian yang ditetapkan oleh Pertamina dalam rangka pengendalian BBM subsidi.

3. Rencana pengiriman BBM yang telah disusun oleh Layanan Jual Depot selanjutnya diserahkan kepada pihak Patra Niaga selaku pengelola mobil tangki untuk selanjutnya dikirim ke SPBU.
4. Pihak Patra Niaga akan mengatur pengiriman ke masing-masing SPBU, termasuk mengatur penugasan mobil tangki dengan tujuan seluruh perencanaan pengiriman yang dibuat oleh Layanan Jual.

3.2 Deskripsi Sistem



Gambar 3.2 Rancangan Sistem

Pada penelitian ini, sistem berusaha mencari nilai minimum dari deviasi jarak tempuh tiap kendaraan. Nilai tersebut dapat diperoleh dari hasil *Vehicle Routing Process* dan *Vehicle Dispatching Process*.

Sistem diawali dengan mengambil data umum. Data umum ini adalah data yang digunakan sebagai data masukan untuk proses *Vehicle Routing Process* dan *Vehicle Dispatching Process*. Data ini ada yang diperoleh langsung dari pihak PT Pertamina TBBM Surabaya Group, dan juga ada data yang berdasarkan asumsi penulis. Data yang bersifat asumsi penulis adalah data jarak tempuh SPBU – SPBU dan Depo – SPBU.

Setelah diperoleh data umum, proses selanjutnya adalah pembuatan rute atau yang bisa disebut sebagai *Vehicle Routing Process*. Tujuan utama dari proses ini adalah menghasilkan rute yang akan dijalankan oleh tiap kendaraan sehingga dapat memenuhi *Demand* tiap SPBU. Rute yang dibuat pada proses ini masih dalam tahap rute tiap *shift*. Pembuatan rute ini mempertimbangkan jumlah *shift* pengiriman, jumlah SPBU tiap *shift*, serta jarak tempuh tiap SPBU. Pembuatan rute ini juga memperhatikan *total Demand* tiap rute. Apabila total *Demand* melebihi kapasitas kendaraan, maka rute tersebut tidak dapat dilayani. Rute tidak dilayani mengakibatkan *Demand* SPBU yang ada di rute tersebut tidak dapat dipenuhi.

Setelah dihasilkan rute tiap *shift* sebagai hasil dari *Vehicle Routing Process*, sistem akan menjalankan proses selanjutnya, yaitu *Vehicle Dispatching Process.* Proses ini akan mengalokasikan kendaraan untuk tiap rute yang ada dengan memperhatikan total *Demand* tiap rute tersebut. Proses pertama yang dilakukan adalah menggabungkan rute tiap *shift* dengan memperhatikan kesesuaian *Demand* rute. Maksudnya adalah rute yang dapat digabung antar *shift* adalah rute yang memiliki jumlah *Demand* yang sama. Hal ini berfungsi untuk mempermudah pengalokasian tipe kendaraan dengan *Demand* rute tersebut. Selain mempertimbangkan total *Demand* tiap rute, *Vehicle Dispatching Process* juga mempertimbangkan jarak tempuh tiap kendaraan. Hal ini dikarenakan fungsi tujuan dari sistem ini adalah meminimumkan deviasi jarak tempuh tiap kendaraan yang terpakai sehingga mendekati nilai jam kerja optimal, yaitu 8 jam.

3.3 Perancangan Konseptual

3.3.1 Parameter Sistem

1. Menggunakan sistem 1 Depo
2. Tipe kendaraan terbagi dalam 4 macam, yaitu tipe 16 KL, 24 KL, 32 KL, dan 40 KL. KL adalah akronim dari Kilo Liter.
3. Tiap tipe kendaraan memiliki jumlah yang fix atau tetap
4. Jarak depo dengan SPBU dan SPBU dengan SPBU adalah fix
5. Tipe *Demand* SPBU pada sistem ini terbagi kedalam nilai 16 KL, 24 Kl, dan 32 KL.
6. Terbagi dalam 3 *shift* pengiriman
7. Kecepatan rata-rata kendaraan adalah 25 km per jam
8. Waktu tempuh maksimal kendaraan adalah 8 jam per hari

3.3.2 Variabel Sistem

1. Jumlah SPBU dalam 1 trip kendaraan
2. Jumlah kendaraan yang digunakan untuk mensuplai keseluruhan *Demand*
3. Jarak tempuh tiap kendaraan
4. *Demand* tiap SPBU

3.3.3 Batasan Sistem

1. *Demand* yang terpenuhi dalam 1 rute tidak boleh melebihi kapasitas angkut kendaraan
2. Rute selalu dimulai dan diakhiri dari depo
3. Tiap SPBU hanya bisa dikunjungi 1 kali
4. Tiap kendaraan dapat mensuplai SPBU lain asalkan kapasitas dan jarak tempuhnya masih mencukupi
5. Semua Demand SPBU harus terpenuhi
6. 1 Kompartemen kendaraan berisi 8 KL
7. *Demand* SPBU adalah kelipatan 8
8. Minimum permintaan tiap SPBU adalah 16 KL dan maksimal 32 KL.
9. Semua SPBU bisa dilayani oleh semua tipe kendaraan

3.3.4 Kriteria Sistem

Meminimumkan deviasi jarak tempuh per kendaraan yang terpakai.

3.4 Model Matematis *Vehicle Routing and Dispatching*

Model matematis digunakan untuk menginterpretasikan hasil perancangan konseptual dalam bentuk yang lebih detil. Dalam model matematis ini, akan dijelaskan mengenai fungsi objektif, parameter, variabel, dan *constraint* yang ada.

3.4.1 Fungsi Objekif

Fungsi objektif dalam permasalahan *vehicle routing and dispatching* adalah untuk meminimumkan deviasi jarak tempuh tiap kendaraan agar mendekati suatu nilai jarak tempuh tertentu. Dalam penelitian ini, digunakan nilai 200 km sebagai nilai jarak tempuh yang ingin didekati. Fungsi deviasi diberikan pada persamaan berikut :

$$MinZ = \left(\sum_{k=1}^{K} \sum_{i=1}^{N} \sum_{j=1}^{N} d_{ij} X_{ijk} - Y \right)^2 \quad (3.1)$$

3.4.2 Parameter

K : Jumlah total kendaraan
N : Jumlah Total SPBU
d_{ij} : Jarak tempuh dari SPBU *i* ke SPBU *j*
m_i : *Demand* SPBU *i*
q_k : Kapasitas kendaraan *k*
Y : Waktu tempuh maksimal kendaraan

3.4.3 Variabel Keputusan

$$X_{ijk} = \begin{cases} 1 & kendaraan\ k\ mengunjungi\ SPBU\ i\ dan\ SPBU\ j \\ 0 & kendaraan\ k\ hanya\ mengunjungi\ salah\ satu\ SPBU \end{cases}$$

3.4.4 Batasan *(constraint)*

$$\sum_{k=1}^{K} \sum_{j=1}^{N} x_{ijk} \leq K, \quad for \quad i = 0 \quad (3.2)$$

$$\sum_{j=1}^{N} x_{ijk} = 1, \quad for \quad i = 0 \quad and \quad k \in \{1, \ldots, K\} \quad (3.3)$$

$$\sum_{j=1}^{N} x_{jik} = 1, \quad for \quad i = 0 \quad and \quad k \in \{1, \ldots, K\} \quad (3.4)$$

$$\sum_{k=1}^{K} \sum_{j=0}^{N} x_{ijk} = 1, \quad for \quad i \in \{1, \ldots, N\} \quad (3.5)$$

$$\sum_{k=1}^{K} \sum_{i=0}^{N} x_{ijk} = 1, \quad for \quad j \in \{1, \ldots, N\} \quad (3.6)$$

$$\sum_{i=1}^{N} m_i \sum_{j=0}^{N} x_{ijk} = q_k, \quad for \quad k \in \{1, \ldots, K\} \quad (3.7)$$

Fungsi Objektif di persamaan (3.1) adalah untuk meminimumkan deviasi jarak tempuh tiap kendaraan yang terpakai. Deviasi yang dimaksud adalah perbedaan waktu tempuh dari waktu tempuh yang seharusnya dipenuhi oleh tiap kendaraan, yaitu 8 jam. Disamping harus memenuhi fungsi objektif, sistem ini juga harus memenuhi semua batasan yang ada.

Pada batasan (3.2) menjelaskan bahwa untuk setiap k kendaraan, jumlah *demand* pada SPBU yang disuplai tidak boleh melebihi kapasitas angkut kendaraan. Batasan (3.3) menjelaskan untuk setiap SPBU i yang dilayani, kendaraan hanya bisa mengunjungi tepat satu kali. Batasan (3.4) ini menjelaskan untuk setiap k kendaraan yang mengunjungi j SPBU,, kendaraan tersebut harus memenuhi *demand* dari SPBU i. Batasan (3.5) menjelaskan kebalikan dari batasan (3.4), yaitu setiap k kendaraan yang mengunjungi i SPBU, kendaraan tersebut harus memenuhi *demand* dari SPBU j. Pada batasan (3.6) dijelaskan setiap rute yang terbentuk selalu berawal dari depo. Sedangkan pada batasan (3.7) dijelaskan bahwa setiap rute yang terbentuk akan berakhir di depo.

3.5 Perancangan Fungsional

Tahap perancangan fungsional adalah menjabarkan semua fungsi operasional sistem, yaitu semua fungsi yang dijalankan oleh sistem ketika sistem dioperasikan. Fungsi operasional perangkat lunak yang dirancang dan dikembangkan adalah sebagai berikut :

1. Penentuan rute
   Sistem dapat mengeluarkan rute dari total SPBU yang ada
2. Penggabungan rute
   Sistem dapat memilih rute mana yang diambil untuk tiap kendaraan
3. Pemilihan Kendaraan
   Sistem dapat mengalokasikan gabungan rute untuk tiap kendaraan. Fungsi ini digunakan untuk meminimumkan deviasi jarak tempuh tiap kendaraan yang terpakai

3.5.1 Inisiasi Data

Sistem diawali dengan inisiasi data umum seperti yang telah dijelaskan di subbab 3.2. Data umum terbagi menjadi 2, data umum bersifat fix dan data umum bersifat asumsi. Data umum yang bersifat fix adalah data yang didapat langsung dari pihak Pertamina TBBM Surabaya

Group. Sedangkan data umum bersifat asumsi adalah data yang dibuat oleh penulis sebagai penunjang data fix. Data umum yang bersifat fix adalah :

1. Jumlah *shift* pengiriman

Shift pengiriman digunakan untuk mengelompokkan (cluster) SPBU kedalam slot-slot pengiriman. Hal ini bertujuan mengoptimalkan pengiriman BBM per hari.

Tabel 3.1 Tabel Pembagian *Shift*

Shift 1	*Shift* 2	*Shift* 3
Cluster SPBU 1	Cluster SPBU 2	Cluster SPBU 3

2. Tipe dan Jumlah Kendaraan

Tipe Kendaraan angkut terbagi menjadi 4 tipe. Dalam kasus kendaraan angkut, dikenal istilah kompartemen. Kompartemen adalah kapasitas terkecil dari tangki kendaraan angkut. Nilai dari 1 kompartemen adalah 8 KL. Sehingga apabila kendaraa tersebut bertipe 16 KL, berarti dia memiliki 2 kompartemen. Begitupula seterusnya. Dalam kasus pengiriman BBM, 1 kompartemen juga melambangkan satuan terkecil dari *demand* SPBU.

Tabel 3.2 Tabel Ketersediaan Kendaraan

Tipe Kendaraan	Keterangan
16 KL	Mobil Kapasitas 16 Kiloliter
24 KL	Mobil Kapasitas 24 Kiloliter
32 KL	Mobil Kapasitas 32 Kiloliter
40 KL	Mobil Kapasitas 40 Kiloliter

3. Alokasi Jam Kerja Kendaraan

Sistem yang dibuat pada tugas akhir ini menggunakan batasan berupa jam kerja kendaraan. Penetapan jam kerja adalah sesuai dengan peraturan pihak Pertamina TBBM Surabaya Group terkait jam operasional kendaraan per hari.

4. Jumlah SPBU

Sistem ini menggunakan SPBU yang ada di wilayah Surabaya, yang tersebar di 95 lokasi yang berbeda. Perlu diketahui bahwa Pertamina TBBM Surabaya Group tidak hanya menyuplai BBM untuk daerah Surabaya saja, melainkan untuk area Jatim dan Bali.

5. Jarak Tempuh Depo – SPBU

Jarak tempuh tiap SPBU terhadap Depo adalah berupa array 1x95. Satuan dari jarak tempuh ini adalah KM.

Sedangkan data umum yang bersifat asumsi adalah :

5.1 Jarak Tempuh SPBU - SPBU

Jarak tempuh antar SPBU ini berupa matriks 95x95. Satuan jarak tempuh ini adalah KM. Data ini diperoleh dengan melihat wilayah SPBU terdekat dan terjauh dari Depo.

5.2 *Demand* tiap SPBU

Demand untuk tiap SPBU dibuat secara acak diantara nilai 16, 24, dan 32 KL. Hal ini didasarkan pada pembatasan masalah pada pengerjaan tugas akhir ini. Selain itu, hal ini juga digunakan untuk mempermudah proses *vehicle dispatching* yang ada pada sistem ini.

3.5.2 Penentuan Rute

Proses penentuan rute atau penggabungan SPBU bertujuan untuk meminimumkan jarak tempuh tiap rute sehingga dapat mencapai fungsi tujuan yang diinginkan. Proses penyediaan rute atau *vehicle routing process* membutuhkan data masukan berupa:

1. Jumlah SPBU
2. Jarak tempuh Depo – SPBU
3. Jarak tempuh SPBU – SPBU
4. *Demand* tiap SPBU
5. Jumlah *shift* pengiriman

Proses diawali dengan menentukan secara random urutan pengiriman dari 95 SPBU yang ada. Setelah didapat urutan random dari ke-95 SPBU, proses selanjutnya adalah menentukan di *shift* mana SPBU tersebut dilayani. Untuk SPBU yang mendapat urutan 1-29, maka SPBU tersebut dilayani di *shift* 1. SPBU yang mendapat urutan 30-57, maka SPBU tersebut dilayani di *shift* 2. Dan untuk SPBU yang mendapat urutan 58-85, akan dilayni di *shift* 3. SPBU yang mendapat urutan 86 sampai 95 diasumsikan tidak dilayani. Rute pertama yang terbentuk dalam setiap *shift* adalah rute dari depo ke SPBU yang mendapat urutan random nomor 1. SPBU urutan 30 dan 58 di *shift* 2 dan 3 diasumsikan mendapat urutan random nomor 1 di *shift* tersebut.

Setelah inisiasi urutan pengiriman dan *plotting shift* dilakukan, proses selanjutnya adalah penambahan SPBU kedalam rute. Proses ini dilakukan dengan menambahkan SPBU yang masih *available* kedalam

suatu rute sesuai dengan urutan yang dibentuk dari urutan random. Proses ini juga mempertimbangkan total *demand* yang terbentuk dari setiap rute. Apabila penambahan SPBU baru kedalam suatu rute mengakibatkan *total demand* melebihi kapasitas angkut kendaraan, maka SPBU baru tersebut tidak bisa ditambahkan ke dalam rute tersebut. Ketika SPBU tersebut tidak bisa ditambahkan pada suatu rute, maka SPBU tersebut menginisiasi terbentuknya rute baru. Selain itu, perlu diingat bahwa semua rute berawal dan berakhir di depo sesuai dengan batasan (3.6) dan (3.7).

3.5.3 Penggabungan Rute

Proses penggabungan rute baru bisa dijalankan apabila semua *shift* telah melakukan proses *vehicle routing*. Sehingga proses penggabungan rute ini memerlukan data masukan berupa :

1. Rute distribusi tiap *shift*
2. Total demand tiap rute
3. Jarak tempuh tiap rute

Proses ini dilakukan untuk *balancing* atau pemerataan beban kerja kendaraan sehingga dapat mencapai fungsi objektif yang diinginkan, yaitu meminimumkan deviasi jarak tempuh kendaraan yang terpakai.

Proses ini diawali dengan mengelompokkan tiap rute berdasarkan total *demand*. Selain itu, proses ini hanya bisa dilakukan antar *shift* yang berbeda. Sehingga apabila ditemukan kasus rute dengan total *demand* sama pada satu *shift*, maka rute tersebut tidak bisa digabungkan. Setelah proses pengelompokkan rute, proses selanjutnya adalah penggabungan rute. Penggabungan rute diawali dengan mengambil rute yang paling awal pada *shift n* untuk ditambahkan dengan rute yang paling awal di *shift n+1*. Apabila dengan penambahan rute tersebut jarak tempuh yang didapat melebihi 200 km, maka penambahan rute tersebut tidak bisa dilakukan. Proses selanjutnya adalah mengganti rute yang tidak bisa ditambahkan tadi dengan rute selanjutnya dengan ketentuan total *demand*-nya sama. Sistem seperti ini dikenal dengan sebutan *sequential insertion.*

Prosedur untuk melakukan proses penggabungan rute adalah sebagai berikut :

1. Data masukan berupa rute distribusi tiap *shift*, total demand tiap rute, dan jarak tempuh tiap rute
2. Lakukan pengelompokan untuk tiap rute yang memiiki kesesuaian demand
3. Lakukan penggabungan rute untuk tiap rute yang bersesuaian total demand antar *shift* yang berbeda

4. Apabila penggabungan rute menghasilkan jarak tempuh melebihi 200 km, maka penggabungan rute tersebut tidak bisa dilakukan.
5. Mencari rute lain dalam kelompok rute yang sama di *shift* yang berbeda
6. Proses penggabungan rute selesai ketika semua rute yang ada telah diproses

3.5.4 Pemilihan Kendaraan

Proses Pemilihan kendaraan berjalan beriringan dengan proses penentuan rute. Hal ini dikarenakan pembuatan rute tidak bisa mengesampingkan kapasitas kendaraan. Data masukan yang diperlukan dalam proses ini antara lain :

1. Data tipe kendaraan
2. Data jumlah kendaraan per tipe kendaraan
3. *Demand* tiap rute dalam 1 *shift*
4. Jarak tempuh tiap rute dalam 1 *shift*
5. Jarak tempuh tiap rute gabungan

Proses ini bertugas untuk mengalokasikan kendaraan untuk menjalankan rute gabungan yang telah terbentuk. Pengalokasian kendaraan ini dilakukan per nomor kendaraan yang tersedia. Sehingga tidak semua kendaraan yang tersedia diberangkatkan untuk menjalankan setiap rute yang ada. Proses ini diawali ketika rute tiap *shift* telah terbentuk. Rute yang ada memiliki total demand yang beragam. Total demand itu nantinya digunakan sebagai sarana acuan penentuan tiep kendaraan. Kendaraan nomor 1 pada tiap *shift* selalu ditugaskan lebih awal dari nomor yang lain. Ketika kendaraan tersebut sudah mencapai batas 200 km atau mendekati, barulah berganti dengan nomor yang lain di tipe kendaraan yang sama. Penambahan jarak tempuh tiap nomor kendaraan juga sangat bergantung pada hasil *routing*. Sehingga kedua sistem ini tidak berjalan terpisah, namun beriringan.

Langkah-langkah dalam proses pemilihan adalah sebagai berikut :

1. Data masukan berupa hasil rute gabungan yang didapat di proses penggabungan rute
2. Total *demand* tiap rute menentukan tipe kendaraan yang digunakan untuk menjalankan rute tersebut
3. Kendaraan yang ditugaskan untuk setiap tipe selalu diawali dari kendaraan nomor 1
4. Ketersediaan kendaraan bernilai fix atau konstan

5. Apabila dalam pemilihan kendaraan didapat suatu kasus depo kehabisan stok kendaraan di tipe tertentu, maka rute yang mendapat jatah kendaraan tersebut tidak bisa dilayani

3.5.5 Fungsi *Firefly Algorithm*

Seperti yang telah dijelaskan pada bab 2, algoritma ini berjalan layaknya kehidupan *firefly* dalam mencari makan. Masukan dari algoritma ini berupa jumlah populasi *firefly*, fungsi keatraktifan *firefly*, dan maksimal generasi atau iterasi. *Firefly Algorithm* yang digunakan dalam sistem *vehicle routing and dispatching* adalah berfungsi untuk menentukan nilai deviasi total terbaik dari segala kemungkinan yang ada. Kemungkinan yang ada didapat dari proses *looping* atau perulangan yang ada pada *firefly algorithm*. Prosedur *firefly algorithm* dalam menyelesaikan sistem *vehicle routing and dispatching adalah* sebagai berikut:

1. Inisiasi parameter *Firefly*

   Dari buku karangan Xin-She Yang yang dijadikan literatur pengerjaan program ini, didapat data sebagai berikut:

Tabel 3.3 Tabel Keterangan *Firefly Algorithm*

Parameter *Firefly*	Keterangan	Nilai
Max Generation	Jumlah iterasi yang digunakan	0 - ∞
N Populasi	Jumlah populasi *firefly*.	0 - ∞
Alpha (α)	Konstanta kecepatan perpindahan *firefly*	0 – 1
Beta (β)	Intensitas cahaya *firefly*	0 – 1
Gamma (γ)	Konstanta Penyerapan cahaya yang tingkat keatraktifan *firefly*	1 - ∞

2. Nilai gamma adalah nilai yang melambangkan tingkat penyerapan cahaya disekitar *firefly*. Semakin tinggi tingkat penyerapan cahaya, maka lingkungan disekitar *firefly* menjadi semakin terang yang menyebabkan *firefly* akan bergerak secara *random* dalam

menemukan solusi terbaiknya. Nilai gamma dalam proses *vehicle routing and dispatching* ditentukan sebagai nilai yang mendekati *infini*t dikarenakan pada kasus pemilihan rute, sistem *firefly* menggunakan nilai gamma mendekati infinit. Persamaan matematis (2.29) ketika nilai gamma yang dipilih adalah ∞ menjadi sebagai berikut:

$$\beta(r) = \beta_0 * \exp\left(-\gamma * r^m\right), \quad m \geq 1$$
$$\beta(r) = \beta_0 * \exp\left(-1000 * r^m\right)$$
$$\beta(r) = 0 \tag{3.8}$$

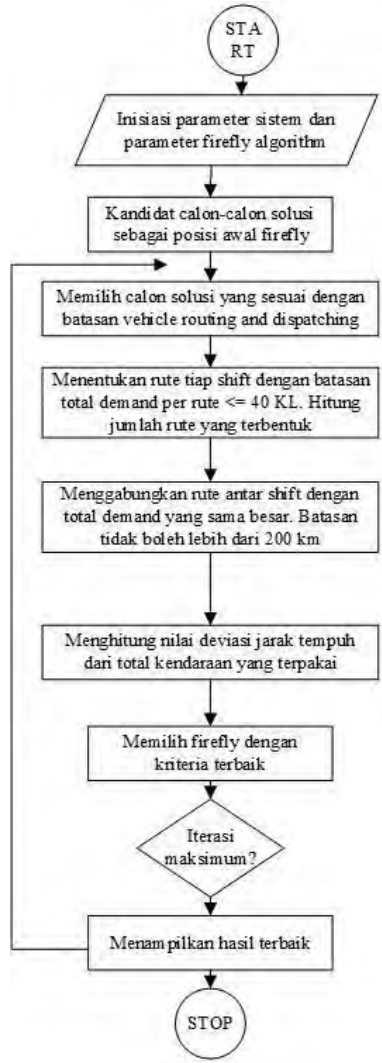
Sedangkan Persamaan (2.31) menjadi sebagai berikut:

$$x_i = x_i + \beta_0 * \exp\left(-\gamma * r_{ij}^{\ 2}\right) * \left(x_j - x_i\right) + a * \left(rand - \frac{1}{2}\right)$$
$$x_i = x_i + a * \left(rand - \frac{1}{2}\right) \tag{3.9}$$

Persamaan (3.9) juga bisa dikatakan bahwa *firefly i* ketika dibandingkan dengan *firefly j* tidak berusaha mendekati satu sama lain. Namun bergerak secara random atau bisa dikatakan hanya membandingkan nilai keluaran satu sama lain.

3. Tiap individu *firefly* melambangkan nilai deviasi yang bertindak sebagai fungsi objektif. Tiap individu ini nantinya akan dibandingkan dengan individu lain yang bertujuan untuk melihat individu mana yang paling terang (benar).

Dalam proses pembuatan sistem *vehicle routing and dispatching* menggunakan *firefly algorithm*, diperlukan suatu diagram alir khusus yang menunjukkan keterikatan antar tiap proses. *Firefly algorithm* bertujuan untuk mencari nilai objektif sesuai pada keterangan nomor 3. Berikut ini adalah diagram alir dari proses *vehicle routing and dispatching* menggunakan *firefly algorithm*.



Gambar 3.3 Diagram Alir *Firefly Algorithm* Pada Kasus *Vehicle Routing and Dispatching*

Halaman ini sengaja dikosongkan

BAB IV
IMPLEMENTASI SISTEM

Berdasarkan spesifikasi dan perancangan sistem yang telah dijelakan pada bab sebelumnya, maka langkah selanjutnya adalah melakukan implementasi sistem. Tujuan dari implementasi sistem ini adalah untuk mengetahui performansi sistem hasil perancangan. Jika performansi yang diharapkan belum sesuai, maka perlu diadakan analisis untuk penyempurnaan kinerja sistem sekaligus dapat digunakan untuk pengembangan lebih lanjut.

Pada bab ini dipaparkan mengenai perancangan detil dari sistem *Vehicle Routing and Dispatching* menggunakan *Firefly Algorithm*.

4.1 Karakterisitik Perangkat

Perangkat yang digunakan dalam proses pembuatan sistem *vehicle routing and dispatching* ini terbagi menjadi 2 perangkat, yaitu perangkat keras (*hardware*) dan perangkat lunak (*software*). Pemilihan *software* dan *hardware* pada sistem ini berdasarkan tingkat ketersediaan dan pemahaman penulis.

4.1.1 Karakteristik *Hardware*

Dalam proses pengerjaan sistem ini, digunakan perangkat keras berupa *Notebook*. Spesifikasi *notebook* yang digunakan dapat dilihat pada tabel 4.1.

Tabel 4.1 Tabel Spesifikasi *Hardware*

No.	Spesifikasi	Keterangan
1.	Merk	Hewlett Packard
2.	Tipe	Pavilion 14
3.	Operating System	Windows 10
4.	Processor	Intel Core i7 @2 GHz
5.	RAM	4 Gigabyte

4.1.2 Karakteristik *Software*

Dalam proses pengerjaan sistem ini, digunakan sebuah perangkat lunak dalam memproses data dan algoritma yang digunakan. Pemilihan perangkat lunak ini didasarkan pada kemampuan penulis dalam

menggunakannya. Perangkat lunak yang dipakai adalah MatLab dengan spesifikasi seperti pada tabel 4.2.

Tabel 4.2 Tabel Spesifikasi *Software*

No.	Spesifikasi	Keterangan
1.	Merk	MATLAB
2.	Tipe	R2015a

4.2 Pengumpulan Data

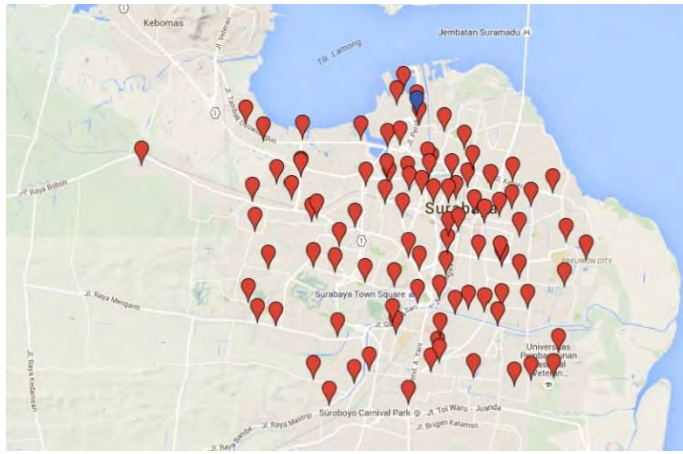
Seperti pada penjelasan subbab 3.5.1 mengenai inisiasi data, bahwa proses pengumpulan data adalah proses paling utama pada keseluruhan proses. Pengumpulan data ini dibagi kedalam dua tipe, yaitu data *intern* dan data *ekstern*.

4.2.1 Data *Intern*

Data *Intern* adalah data penunjang pengerjaan sistem *vehicle routing and dispatching* yang diperoleh langsung dari pihak Pertamina TBBM Surabaya Group. Adapun data-data tersebut antara lain:

1. Data SPBU wilayah Surabaya

   SPBU yang dipakai dalam proses pengerjaan sistem ini adalah SPBU wilayah Surabaya dengan jumlah 95 Unit dengan 1 Unit Depo Peta wilayah sebaran SPBU dapat diihat pada gambar 4.1



Gambar 4.1 Peta Wilayah Sebaran SPBU Surabaya

2. Data Jumlah dan Tipe Kendaraan

Tipe kendaraan terbagi ke dalam 4 tipe, yaitu tipe 16 KL, 24 KL, 32 KL, dan 40 KL. Tiap-tiap tipe memiliki jumlah masing-masing yang dijelaskan dalam tabel 4.3.

Tabel 4.3 Tabel Jumlah Kendaraan

Tipe Kendaraan	Jumlah Kendaraan
16 KL	17 Unit
24 KL	66 Unit
32 KL	56 Unit
40 KL	14 Unit

3. Data Nomor Kendaraan

Data nomor kendaraan ini diperlukan dalam proses *vehicle dispatching*. Nomor kendaraan ini adalah data fix sesuai ketersediaan kendaraan di Pertamina TBBM Surabaya Group. Adapun data nomor kendaraan dapat dilihat pada tabel 4.4 sampai 4.7.

Tabel 4.4 Tabel Nomor Polisi Tipe 16 KL

TIPE	NOMOR	NOMOR POLISI	TIPE	NOMOR	NOMOR POLISI
16	1	L9086UH	16	10	L8030UW
	2	L9457UR		11	L9227UH
	3	L9208UU		12	L8194UG
	4	L9330UR		13	L9733UP
	5	L8752UE		14	L8991J
	6	L8752UD		15	M8746UN
	7	L8407UC		16	L8221UC
	8	L8752UC		17	L9711UV
	9	B9508SFU			

Tabel 4.5 Tabel Nomor Polisi Tipe 24 KL

TIPE	NOMOR	NOMOR POLISI	TIPE	NOMOR	NOMOR POLISI
24	1	L9126UZ	24	11	B9556SEH

TIPE	NOMOR	NOMOR POLISI	TIPE	NOMOR	NOMOR POLISI
24	2	L8565UZ	24	12	AG9043US
	3	L8970UZ		13	L8448UV
	4	L8410UU		14	L9032UU
	5	AG8734R		15	L9031UU
	6	M8516UH		16	L9124UU
	7	M8591UH		17	L9123UU
	8	L9261UE		18	N8122UC
	9	L9144UE		19	N8123UC
	10	B9557SEH		20	N8447UC
24	21	L9288UC	24	31	E9262YB
	22	L9452UA		32	E9301YB
	23	L8506UP		33	E9302YB
	24	L8637UD		34	L9771UQ
	25	B9484SEH		35	L9770UQ
	26	B9358SEH		36	L9148UR
	27	B9481SEH		37	L9640UR
	28	B9424UO		38	L9677UR
	29	E9224YB		39	L9712UG
	30	E9223YB		40	L9561UG
24	41	L9178UH	24	51	W8496UP
	42	L9762UH		52	W8617UP
	43	N8445UC		53	L9346UU
	44	N8526UJ		54	L9794UG
	45	L9315US		55	L9793UG
	46	L9214UT		56	L9048UH

TIPE	NOMOR	NOMOR POLISI	TIPE	NOMOR	NOMOR POLISI
24	47	L9497UP	24	57	L9049UH
	48	L8566UP		58	L9082UH
	49	L8565UP		59	L9083UH
	50	W8375UP		60	L9144UH
24	61	L9143UH			
	62	L9605UR			
	63	L9270US			
	64	L8567UP			
	65	L8331UW			
	66	L9433UR			

Tabel 4.6 Tabel Nomor Polisi Tipe 32 KL

TIPE	NOMOR	NOMOR POLISI	TIPE	NOMOR	NOMOR POLISI
32	1	AG9044US	32	11	L9160UD
	2	AG9748US		12	L9275UF
	3	N8164UA		13	L9251UF
	4	N8442UC		14	E9238YB
	5	AG8956US		15	E9512YB
	6	L9434UB		16	E9514YB
	7	L9442UB		17	L9911UX
	8	L9508UB		18	L9377UQ
	9	L9527UB		19	L9378UQ
	10	L9528UB		20	L9466UQ
32	21	L9579UG	32	31	L9259UQ
	22	L9578UG		32	L9361UQ

TIPE	NOMOR	NOMOR POLISI	TIPE	NOMOR	NOMOR POLISI
32	23	N8502UB	32	33	L9388UQ
	24	N8537UB		34	L9389UQ
	25	N8524UB		35	L9160UR
	26	N8078UA		36	L9416UN
	27	N8506UJ		37	L9659UP
	28	N8571UJ		38	L9660UP
	29	L9193UQ		39	L9385UO
	30	L9260UQ		40	W8124UP
32	41	W8238UP	32	51	L9152UH
	42	W8239UP		52	L9445UH
	43	W8315UP		53	L9174UL
	44	W8316UP		54	L9000UZ
	45	L9783UG		55	L9000UY
	46	L8296UU		56	L8246UU
	47	L9513UG			
	48	L9492UG			
	49	L9723UG			
	50	L9040UH			

Tabel 4.7 Tabel Nomor Polisi Tipe 40 KL

TIPE	NOMOR	NOMOR POLISI	TIPE	NOMOR	NOMOR POLISI
40	1	L9501UB	40	11	L8562UX
	2	L9585UG		12	L8157UU
	3	N8540UB		13	L9573UG
	4	N9098UF		14	L9123UH
	5	N8024UE			

TIPE	NOMOR	NOMOR POLISI	TIPE	NOMOR	NOMOR POLISI
40	6	N8467UE			
	7	L9396UQ			
	8	L9582UP			
	9	L9293UO			
	10	W8123UP			

4. Jumlah *Shift* Pengiriman

Shift pengiriman dalam proses distribusi BBM adalah slot waktu yang disediakan oleh pihak Pertamina dalam kurun waktu satu hari. Pembagian *shift* pengiriman didasarkan pada tingkat ketersediaan kendaraan dan efektifitas pengiriman. Adapun pembagian *shift* pengiriman dalam waktu satu hari ditunjukkan dalam tabel 4.8.

Tabel 4.8 Tabel Pembagian Waktu Per *Shift*

Shift	Slot Waktu
Shift 1	24.01 – 08.00
Shift 2	08.01 – 16.00
Shift 3	16.01 – 24.00

5. Jarak Tempuh Depo ke tiap SPBU

Data Jarak tempuh ini berukuran matriks array 1x95. Data ini merupakan jarak per satuan KM dari depo ke tiap-tiap SPBU. Data ini disediakan oleh penulis dalam lampiran.

6. Alokasi Jam Kerja Per Hari

Jam kerja per hari adalah suatu nilai yang merepresentasikan jam kerja aktif pekerja Pertamina TBBM Surabaya Group. Jam kerja aktif para pekerja Pertamina TBBM Surabaya Group adalah 8 jam per hari .

4.2.2 Data *Ekstern*

Data *Ekstern* adalah data penunjang pengerjaan sistem *vehicle routing and dispatching* yang ditentukan oleh penulis sendiri. Data-data tersebut diperoleh dengan proses tertentu sesuai dengan kebutuhan sistem *vehicle routing and dispatching* yang ada. Adapun data-data tersebut antara lain:

1. *Demand* Tiap SPBU

Tabel 4.9 Pseudocode Penentuan *Random Demand*

```
spbuNUM = 95;

spbu Demand = zeros();
for i = 1:95
    spbu Demand(i) = randi([2 4]);
 end
```

Dalam tabel 4.9,dijelaskan bahwa pada sistem *vehicle routing and dispatching* yang dibuat, *demand* tiap-tiap SPBU dibuat acak dikisaran nilai 16 KL, 24 KL, dan 32 KL. Dalam penulisan demand 16, 24, dan 32, penulis merepresentasikannya kedalam satuan yang melambangkan kompartemennya. 16 KL dilambangkan dengan nilai 2. 24 KL dilambangkan dengan nilai 3. Dan 32 KL dilambangkan dengan nilai 4.

2. Jarak Tempuh Antar SPBU

Jarak tempuh antar SPBU adalah sebuah matriks berukuran 95x95. Data ini dibuat dengan melakukan prosedur pengacakan dengan batas atas berupa SPBU terjauh dari depo dan batas bawah berupa SPBU terdekat dari depo.

4.3 *Firefly Algorithm* untuk menyelesaikan proses *Vehicle routing and dispatching*

Firefly algorithm digunakan untuk mencari nilai objektif yang paling baik dari sejumlah iterasi yang telah dibuat. Dalam sistem ini dicari nilai deviasi jarak tempuh paling minimum dari keseluruhan iterasi. Sesuai prosedur yang telah dijelaskan pada subbab 3.5.2 sampai 3.5.4 serta menggunakan sistem *firefly algorithm* yang ada pada subbab 3.5.5, didapatkan suatu nilai deviasi jarak tempuh minimum yang berada pada nilai 1.677 jam. Pada tabel 4.13 dijelaskan parameter *firefly* yang digunakan dalam proses pengerjaan sistem *vehicle routing* and *dispatching.*

Tabel 4.10 Tabel Parameter *Firefly Algorithm Vehicle Routing and Dispatching*

Parameter *Firefly*	Nilai
MaxGeneration (iterasi)	50
Populasi *Firefly* (n)	200
Gamma (γ)	1000

Dalam penjelasan selanjutnya akan dijelaskan mengenai prosedur *firefly algorithm* yang digunakan. Proses diawali dengan mendeklarasikan variabel yang akan menentukan dan mempengaruhi nilai dari deviasi total. Variabel tersebut antara lain jarak tempuh tiap kendaraan yang terpakai, jarak tempuh tiap rute dalam satu *shift*, *demand* yang terpenuhi dalam satu rute, dan *demand* yang terpenuhi dalam satu kendaraan yang terpilih. Nilai-nilai yang dihasilkan dalam tiap variabel tersebut nantinya akan digunakan untuk mencari nilai deviasi terbaik dalam satu iterasi. Nilai deviasi terbaik total adalah nilai deviasi terbaik dari sejumlah iterasi yang digunakan. Keterangan mengenai variabel yang digunakan dapat dilihat dalam tabel 4.11. Sedangkan *pseudocode* untuk menemukan nilai deviasi terbaik dapat dilihat pada tabel 4.12.

Tabel 4.11 Tabel Variabel *Firefly Algorithm Vehicle Routing and Dispatching*

Variabel	Keterangan
CleanLoad	Jarak tempuh tiap kendaraan yang terpakai
Beban	Jarak tempuh semua kendaraan
Sch1, Sch2, Sch3	Jarak tempuh kendaraan *shift*1, *shift*2, *shift*3
Stdevi	Deviasi Total
*Shift*1, *Shift*2, *Shift*3	Data Jarak tempuh, demand, dan rute tiap kendaraan tiap *shift*
Used1, Used2, Used3	Demand yang terlayani di *shift*1, *shift*2, *shift*3
BestStdevi	Deviasi Total terbaik yang telah direcord sistem
CarSchedule	Beban kerja per kendaraan

Tabel 4.12 *Pseudocode Firefly Algorithm Vehicle Routing and Dispatching*

```
carSchedule(i).cleanLoad=cleanLoad;
            carSchedule(i).beban=beban;
            carSchedule(i).sch1=sch1;
            carSchedule(i).sch2=sch2;
            carSchedule(i).sch3=sch3;
            carSchedule(i).stdevi=stdevi;
            carSchedule(i).shift1=initial1(i);
            carSchedule(i).shift2=initial2(i);
            carSchedule(i).shift3=initial3(i);
            carSchedule(i).used1=used1;
            carSchedule(i).used2=used2;
            carSchedule(i).used3=used3;

if(carSchedule(i).stdevi<best.stdevi)
best.cleanLoad=carSchedule(i).cleanLoad;
            best.beban=carSchedule(i).beban;
            best.sch1=carSchedule(i).sch1;
            best.sch2=carSchedule(i).sch2;
            best.sch3=carSchedule(i).sch3;
            best.stdevi=carSchedule(i).stdevi;
            best.shift1=carSchedule(i).shift1;
            best.shift2=carSchedule(i).shift2;
            best.shift3=carSchedule(i).shift3;
            best.used1=used1;
            best.used2=used2;
            best.used3=used3;
            end
```

Untuk menemukan nilai fungsi objektif terbaik pada *firefly algorithm*, perlu dilakukan proses *vehicle routing and dispatching* didalamnya. Proses *vehicle routing* sendiri terbagi menjadi 2 bahasan utama, yaitu pengelompokan SPBU dan pembuatan rute tiap *shift*.

Sedangkan pada *vehicle dispatching* juga terbagi 2 menjadi 2 bahasan utama, yaitu penggabungan rute dan pemilihan kendaraan.

4.3.1 Pengelompokan SPBU

Tabel 4.13 *Pseudocode* Penentuan SPBU Per *Shift*

```
%Buat Komposisi SPBU Tiap Shift Baru

Begin
I=1:95
newPath = randperm(95);
shift1(1:29)=newPath(1:29);
shift2(1:28)=newPath(30:57);
shift3(1:28)=newPath(58:85);
shift4(1:10)=newPath(86:95);          %Tidak Disuplai

finalSpbu=zeros();
finalSpbu(1:29)=shift1(1:29);
finalSpbu(30:57)=shift2(1:28);
finalSpbu(58:85)=shift3(1:28);

%% Read Demand SPBU
   order = strcat('spbuOrder.csv');
   spbuOrder = dlmread(order);
   newPath = spbuOrder;
%% Read initial Distance
   init = strcat('distFromDepo.csv');
   initDist = dlmread(init);
```

Proses *vehicle routing* diawali dengan mengelompokkan SPBU ke dalam *shift-shift* yang telah disediakan. Pengelompokan SPBU didasarkan pada penomoran SPBU yang dilakukan secara acak. Nomor urut 1-29 masuk kedalam *shift* 1. Nomor urut 30-57 masuk kedalam *shift* 2. Dan nomor urut 58-85 masuk kedalam *shift* 3. Sedangkan nomor urut 86-95 tidak dimasukkan dalam *shift* manapun karena diasumsikan tidak disuplai pada hari itu.Total dalam pengerjaan sistem ini, SPBU yang dilayani berjumlah 85 SPBU. 10 SPBU yang tersisa dianggap tidak disuplai. Sehingga secara total, ada 85 SPBU yang disuplai dan 10 yang tidak.

4.3.2 Pembuatan Rute Tiap *Shift*

Tabel 4.14 *Pseudocode* Pembuatan Rute Tiap *Shift*

```
%Inisialisasi Rute Tiap Shift
Num ←jumlah spbu
Urutan ← random permutasi(num)
threshold ← 200
dist ← jarak SPBU ke SPBU
distDepo ← dist SPBU from depo

%Read Shift SPBU Awal
    name1 = strcat('shift1.csv');
    shift1 = dlmread(name1);
    name2 = strcat('shift2.csv');
    shift2 = dlmread(name2);
    name3 = strcat('shift3.csv');
    shift3 = dlmread(name3);

%Record Data Ke Fungsi Firefly
answer = getPath(shift n);
initial1(i).pathEachCar=answer.pathEachCar;
initial1(i).distEachCar=answer.distEachCar;
initial1(i).weightEachCar=answer.weightEachCa
r
initial1(i).carSize=answer.carSize;
initial1(i).totDist=answer.totDist;
```

Seperti yang telah dijelaskan ada subbab 3.5.2 mengenai penyediaan rute, bahwa pembuatan rute tiap *shift* dimulai dengan merandom urutan SPBU. Tiap *shift* memiliki komposisi SPBU yang telah diatur pada sistem pengelompokan SPBU. SPBU dengan urutan terkecil disuplai terlebih dahulu dan berlanjut sampai SPBU dengan urutan paling akhir. Dalam prosesnya, pembentukan rute baru dimulai ketika *total demand* pada suatu rute telah melebihi kapasitas kendaraan dan jarak tempuh masksimal, yaitu 200 km. Proses pembuatan rute tiap *shift* akan berakhir apabila semua *shift* telah terbentuk rutenya.

```
%Fungsi getPath
n ←urutan

inisialisasi start

pathCar(1,1) ← n(1)           %    SPBU    per
Kendaraan
weightCar(1,1) ← weight(n(1)) %   Demand   SPBU
per Kendaraan
distCar(1,1) ← distDepo(n(1)) %  Jarak  Tempuh
per Kendaraan

x=1
y=1

for i←2 to num
     add ← weight(n(i))
     addJarak ← dist(n(i),n(i-1))

if (weightCar + add ≤5 and distCar+addJarak ≤
threshold)
      pathCar(x,y) ← n(i)
      weightCar ← weightCar + add
      weightCar(x,y) ← weight(n(1))
      distCar(x,y) ← addJarak
      distCar=distCar+addJarak
      y+1  %Ke SPBU selanjutnya
     else
      x=x+1  %Gunakan kendaraan baru
      y=1
      pathCar(x,y) ←n(i)
      distCar(x,y) = distDepo(n(i))
      weightCar(x,y) = weight(n(i))
      y=y+1
      distCar = distDepo(n(i))
      weightCar = weight(n(i))
endif
end
```

4.3.3 Penggabungan Rute Antar *Shift*

Seperti yang telah dijelaskan pada subbab 3.5.3 bahwa penggabungan rute hanya bisa dilakukan pada tiap total *demand* yang nilainya sama besar. Proses ini nantinya juga akan menghasilkan tipe kendaraan mana dan nomor berapa yang akan menyuplai rute tertentu. Rute tersebut juga harus memenuhi batasan yang ada pada persamaan 3.2,

Tabel 4.15 *Pseudocode* Penggabungan Rute Antar *Shift*

```
begin
initialDist → Jarak tempuh shift sebelumnya
dist → Jarak tempuh shift saat ini
weight → Demand yg dilayani tiap kendaraan
shift saat ini
flag → true
a → Jumlah Kendaraan yang tersedia di shift
saat ini

type2 → 1;        %Nomor Kendaraan 1-17
type3 → 18;       %Nomor Kendaraan 18-83
type4 → 84;       %Nomor Kendaraan 84-139
type5 → 140;      %Nomor Kendaraan 140-153

    used → list kode rute tiap mobil
    schedule → jarak shift sekarang tiap mobil

[a,b] → size (dist);

while z ≤ a
     if type2 ≥18 or type3≥84 or type4≥140 or
     type5≥154
      break       ←Tidak ada solusi
      z=a;
     else
```

```
if weight(z) == 2
      if initialDist(type2) + dist(i) ≤200
            schedule(type2)→dist(z);
            used(type2) → z;
            z→z+1;
            type2→type2+1;
            else
            type2→type2+1;
      end
if weight(z) == 3
     if initialDist(type3) + dist(i)≤200
            schedule(type3)→dist(z);
            used(type3) → z;
            z → z+1;
            type3 → type3+1;
            else
            type3 → type3+1;
     end
if weight(z) == 4
      if initialDist(type4) + dist(i)≤200
            schedule(type4)→dist(z);
            used(type4) → z;
            z → z+1;
            type4 → type4+1;
            else
            type4 → type4+1;
            end
if weight(z) == 5
      if initialDist(type5) + dist(i) <=200
            schedule(type5)=dist(z);
            used(type5)->z;
            z->z+1;
            type5->type5+1;
            else
            type5->type5+1;
endwhile
end
```

4.3.4 Pemilihan Kendaraan

Seperti yang telah dijelaskan pada subbab 3.5.4, pemilihan kendaraan dapat dilakukan apabila kita telah mendapatkan rute gabungan antar *shift*. Hal ini mengindikasikan bahwa tiap rute hanya dilayani oleh 1 kendaraan saja. Proses pemilihan kendaraan adalah mencocokkan data rute yang ada dengan ketersediaan kendaraan per nomor kendaraan.

Tabel 4.16 *Pseudocode* Pemilihan Kendaraan

```
%Read Car Capacity Constrain
    carAvail = dlmread('carCapacity.csv');
    bestNow=zeros;

%Penjadwalan kendaraan
schedule = zeros(153,3);

    %Urutan Nomor Kendaraan Berdasarkan Tipe
            type2=1;
            type3=18;
            type4=84;
            type5=140;
    %Inisialisasi Jarak Tempuh
            [a,b]=size(initial1(i).totDist);

CleanLoad = CarAvail;

%Untuk Shift 1
   sch1s =
initSchedule(initial1(i).totDist,initial1(i).
carSize);
            sch1=sch1s.schedule;
            used1=sch1s.used;
            beban=zeros(153,1);
            for r=1:153
                beban(r)=sch1(r);
            end
```

```
%Untuk Shift 2
      sch2s =
initSchedule2(beban,initial2(i).totDist,initia
l2(i).carSize);
            sch2 = sch2s.schedule;
            used2 = sch2s.used;
            beban=zeros(153,1);
            for r=1:153
                beban(r)=sch1(r)+sch2(r);
            end
%Untuk Shift 3
            sch3s =
initSchedule2(beban,initial3(i).totDist,initia
l3(i).carSize);
            sch3 = sch3s.schedule;
            used3 = sch3s.used;
            beban=zeros(153,1);
for r=1:153
     beban(r)=sch1(r)+sch2(r)+sch3(r);
end
            cleanLoad = zeros();
            idx=1;
            for r=1:153
                if(beban(r)>0)
                cleanLoad(idx)=beban(r);
                idx=idx+1;
                end
            end
```

4.4 Hasil *Running* Program

Hasil *running* program diperoleh ketika kita sudah melakukan prosedur yang ada pada subbab 4.3 dan 4.4. Dari hasil *running* menggunakan *software* MatLab yang berdasarkan pada perancangan sistem yang telah dibuat, dapat diambil suatu hasil bahwa terdapat 24 rute yang melayani 85 SPBU dengan total jarak tempuh seluruh kendaraan yang terpakai mencapai 3938.601 km. Sedangkan deviasi jarak tempuh

kendaraan yang terpakai adalah 1.677 jam. Parameter *firefly* yang digunakan sesuai dengan yang ada pada tabel 4.10.

4.4.1 Hasil dalam bentuk tabel

Ada beberapa data yang dihasilkan dari proses pengerjaan *vehicle routing and dispatching* menggunakan *firefly algorithm*. Pada tabel 4.17, 4.18, dan 4.19 adalah hasil dari proses pembuatan rute per *shift* subbab 4.3.2. Pada tabel 4.20 adalah hasil dari proses penggabungan rute subbab 4.3.3. Dan yang terakhir adalah tabel 4.21 yang merupakan hasil dari proses pemilihan kendaraan subbab 4.3. Tiap rute meiliki karakteristik masing-masing yang terdiri dari jarak tempuh, *total demand*, dan SPBU yang dilayani.

Tabel 4.17 Rute *Shift* 1

RUTE	SPBU	JARAK TEMPUH (Km)	TOTAL *DEMAND*	KENDARAAN
1	DEPO – 30 - DEPO	42.888	3	24(1)
2	DEPO – 51 – DEPO	70.332	4	32(1)
3	DEPO – 4 – 92 – DEPO	65.034	2+2	32(2)
4	DEPO – 13 – 90 – DEPO	57.857	3+2	40(1)
5	DEPO – 23 – DEPO	30.06	2	16(1)
6	DEPO – 75 – DEPO	52.062	4	32(3)
7	DEPO – 42 – DEPO	75.59	4	32(4)
8	DEPO – 16 – 32 – DEPO	84.973	2+2	32(5)
9	DEPO – 94 – 76 – DEPO	70.461	2+3	40(2)
10	DEPO – 65 – 22 – DEPO	63.366	2+3	40(3)

RUTE	SPBU	JARAK TEMPUH (Km)	TOTAL *DEMAND*	KENDARAAN
11	DEPO – 95 – 83 – DEPO	80.789	3+2	40(4)
12	DEPO – 85 – DEPO	73.712	3	24(2)
13	DEPO – 50 – DEPO	60.634	4	32(6)
14	DEPO – 64 – 54 – DEPO	85.835	3+2	40(5)
15	DEPO – 58 – DEPO	66.262	4	32(7)
16	DEPO – 56 – DEPO	75.786	4	32(8)
17	DEPO – 80 – 15 – DEPO	81.937	3+2	40(6)
18	DEPO – 37 – DEPO	40.24	4	32(9)
19	DEPO – 86 – DEPO	74.366	3	24(3)
20	DEPO – 35 – DEPO	41.914	4	32(10)
21	DEPO – 73 – DEPO	58.31	3	24(4)

Tabel 4.18 Rute *Shift* 2

RUTE	SPBU	JARAK TEMPUH (Km)	TOTAL *DEMAND*	KENDARAAN
1	DEPO – 40 – 84 – DEPO	79.352	2+3	40(1)
2	DEPO – 41 – DEPO	42.148	4	32(1)
3	DEPO – 88 – DEPO	64.688	3	24(1)

RUTE	SPBU	JARAK TEMPUH (Km)	TOTAL *DEMAND*	KENDARAAN
4	DEPO – 68 – 19 – DEPO	80.6331	3+2	40(2)
5	DEPO – 55 – DEPO	57.026	3	24(2)
6	DEPO – 33 – DEPO	49.13	3	24(3)
7	DEPO – 45 – 77 – DEPO	50.9704	3+2	40(3)
8	DEPO – 28 – DEPO	41.414	4	32(2)
9	DEPO – 26 – DEPO	55.872	4	32(3)
10	DEPO – 60 – 57 – DEPO	53.335	3+2	40(4)
11	DEPO – 66 – 20 – DEPO	83.511	2+2	32(4)
12	DEPO – 39 – DEPO	52.322	4	32(5)
13	DEPO – 12 – DEPO	74.908	4	32(6)
14	DEPO – 93 – DEPO	49.384	4	32(7)
15	DEPO – 27 – DEPO	58.976	4	32(8)
16	DEPO – 7 – 79 – DEPO	67.725	2+3	40(5)
17	DEPO – 43 – DEPO	64.944	2	16(1)
18	DEPO – 11 – DEPO	60.246	4	32(9)

RUTE	SPBU	JARAK TEMPUH (Km)	TOTAL *DEMAND*	KENDARAAN
19	DEPO – 18 – DEPO	41.162	3	24(4)
20	DEPO – 46 – DEPO	49.584	3	24(5)
21	DEPO – 48 – DEPO	68.936	3	24(6)
22	DEPO – 47 - DEPO	70.148	3	24(7)

Tabel 4.19 Rute *Shift* 3

RUTE	SPBU	JARAK TEMPUH (Km)	TOTAL *DEMAND*	KENDARAAN
1	DEPO – 72 – DEPO	42.396	4	32(1)
2	DEPO – 29 – 63 – DEPO	60.2151	3+2	40(1)
3	DEPO – 82 – 49 – DEPO	55.89	2+2	32(2)
4	DEPO – 1 – 25 – DEPO	89.225	3+2	32(3)
5	DEPO – 53 – DEPO	49.176	3	24(1)
6	DEPO – 14 – DEPO	67.814	4	32(5)
7	DEPO – 71 – 59 – DEPO	59.976	3+2	40(2)
8	DEPO – 36 – DEPO	52.408	3	24(2)
9	DEPO – 69 – DEPO	75.118	3	24(3)

RUTE	SPBU	JARAK TEMPUH (Km)	TOTAL *DEMAND*	KENDARAAN
10	DEPO – 70 – DEPO	70.338	4	32(6)
11	DEPO – 2 – DEPO	64.458	4	32(7)
12	DEPO – 3 – DEPO	51.696	4	32(8)
13	DEPO – 91 – DEPO	74.07	3	24(4)
14	DEPO – 78 – DEPO	75.696	3	24(5)
15	DEPO – 9 – 17 – DEPO	56.4003	3+2	40(3)
16	DEPO – 81 – DEPO	45.266	2	16(1)
17	DEPO – 31 – DEPO	49.196	4	32(9)
18	DEPO – 38 – 34 – DEPO	81.025	2+2	32(10)
19	DEPO – 52 – DEPO	44.966	3	24(6)
20	DEPO – 67 – 89 – DEPO	79.4995	3+2	40(4)
21	DEPO – 21 – DEPO	49.5	3	24(7)

Tabel 4.20 Tabel Rute Gabungan

DEMAND	SPBU (*SHIFT* 1)	SPBU (*SHIFT* 2)	SPBU (*SHIFT* 3)
16 (1)	DEPO – 23 – DEPO	DEPO – 43 – DEPO	DEPO – 81 – DEPO

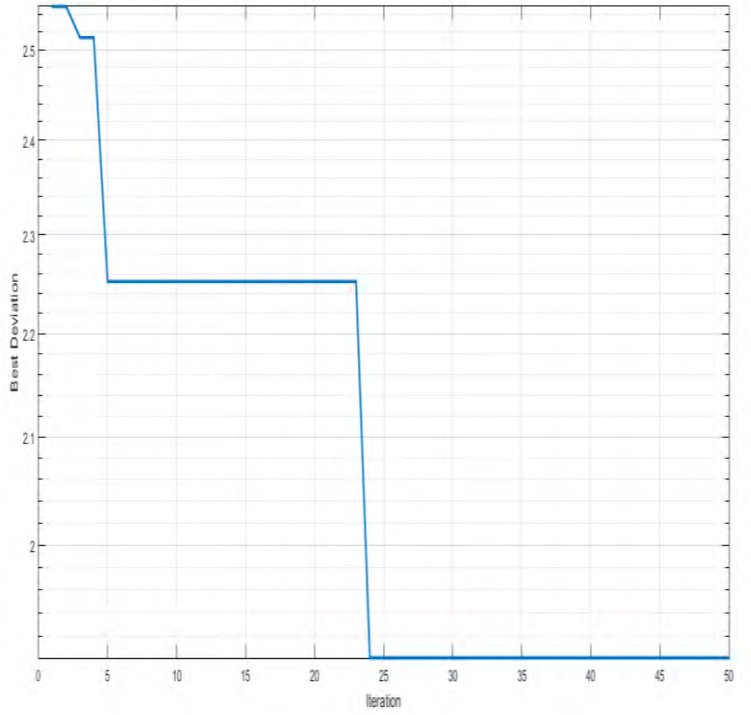
DEMAND	SPBU (*SHIFT* 1)	SPBU (*SHIFT* 2)	SPBU (*SHIFT* 3)
24 (1)	DEPO – 30 – DEPO	DEPO – 88 – DEPO	DEPO – 53 – DEPO
24 (2)	DEPO – 85 – DEPO	DEPO – 55 – DEPO	DEPO – 36 – DEPO
24 (3)	DEPO – 86 – DEPO	DEPO – 33 – DEPO	DEPO – 69 – DEPO
24 (4)	DEPO – 73 – DEPO	DEPO – 18 – DEPO	DEPO – 91 – DEPO
24 (5)	-	DEPO – 46 – DEPO	DEPO – 78 – DEPO
24 (6)	-	DEPO – 48 – DEPO	DEPO – 52 – DEPO
24 (7)	-	DEPO – 47 - DEPO	DEPO – 21 – DEPO
32 (1)	DEPO – 51 – DEPO	DEPO – 41 – DEPO	DEPO – 72 – DEPO
32 (2)	DEPO – 4 – 92 – DEPO	DEPO – 28 – DEPO	DEPO – 82 – 49 – DEPO
32 (3)	DEPO – 75 – DEPO	DEPO – 26 – DEPO	DEPO – 1 – 25 – DEPO
32 (4)	DEPO – 42 – DEPO	DEPO – 66 – 20 – DEPO	-
32 (5)	DEPO – 16 – 32 – DEPO	DEPO – 39 – DEPO	DEPO – 14 – DEPO

DEMAND	SPBU (*SHIFT* 1)	SPBU (*SHIFT* 2)	SPBU (*SHIFT* 3)
32 (6)	DEPO – 50 – DEPO	DEPO – 12 – DEPO	DEPO – 70 – DEPO
32 (7)	DEPO – 58 – DEPO	DEPO – 93 – DEPO	DEPO – 2 – DEPO
32 (8)	DEPO – 56 – DEPO	DEPO – 27 – DEPO	DEPO – 3 – DEPO
32 (9)	DEPO – 37 – DEPO	DEPO – 11 – DEPO	DEPO – 31 – DEPO
32 (10)	DEPO – 35 – DEPO	-	DEPO – 38 – 34 – DEPO
40 (1)	DEPO – 13 – 90 – DEPO	DEPO - 40 - 84 - DEPO	DEPO – 29 – 63 – DEPO
40 (2)	DEPO – 94 – 76 – DEPO	DEPO – 68 – 19 – DEPO	DEPO – 71 – 59 – DEPO
40 (3)	DEPO – 65 – 22 – DEPO	DEPO – 45 – 77 – DEPO	DEPO – 9 – 17 – DEPO
40 (4)	DEPO – 95 – 83 – DEPO	DEPO – 60 – 57 – DEPO	DEPO – 67 – 89 – DEPO
40 (5)	DEPO – 64 – 54 – DEPO	DEPO – 7 – 79 – DEPO	-
40 (6)	DEPO – 80 – 15 – DEPO	-	-

Tabel 4.21 Tabel Nomor Kendaraan dan Deviasi Total

TIPE	NOMOR	TOTAL JARAK TEMPUH (Km)	DEVIASI (Jam)
16	L9086UH	160.720	
24	L9126UZ	156.752	
	L8565UZ	183.146	
	L8970UZ	198.614	
	L8410UU	173.542	
	AG8734R	125.28	
	M8516UH	113.902	
	M8591UH	105.098	
32	AG9044US	154.876	
	AG9748US	162.338	
	N8164UA	197.159	
	N8442UC	159.101	
	AG8956US	205.109	1.677
	L9434UB	205.88	
	L9442UB	180.104	
	L9508UB	186.458	
	L9527UB	149.682	
	L9528UB	122.929	
40	L9501UB	197.4241	
	L9585UG	151.0941	
	N8540UB	174.3124	
	N9098UF	190.5243	
	N8024UE	153.56	
	N8467UE	161.4365	
TOTAL	24	3938.601	

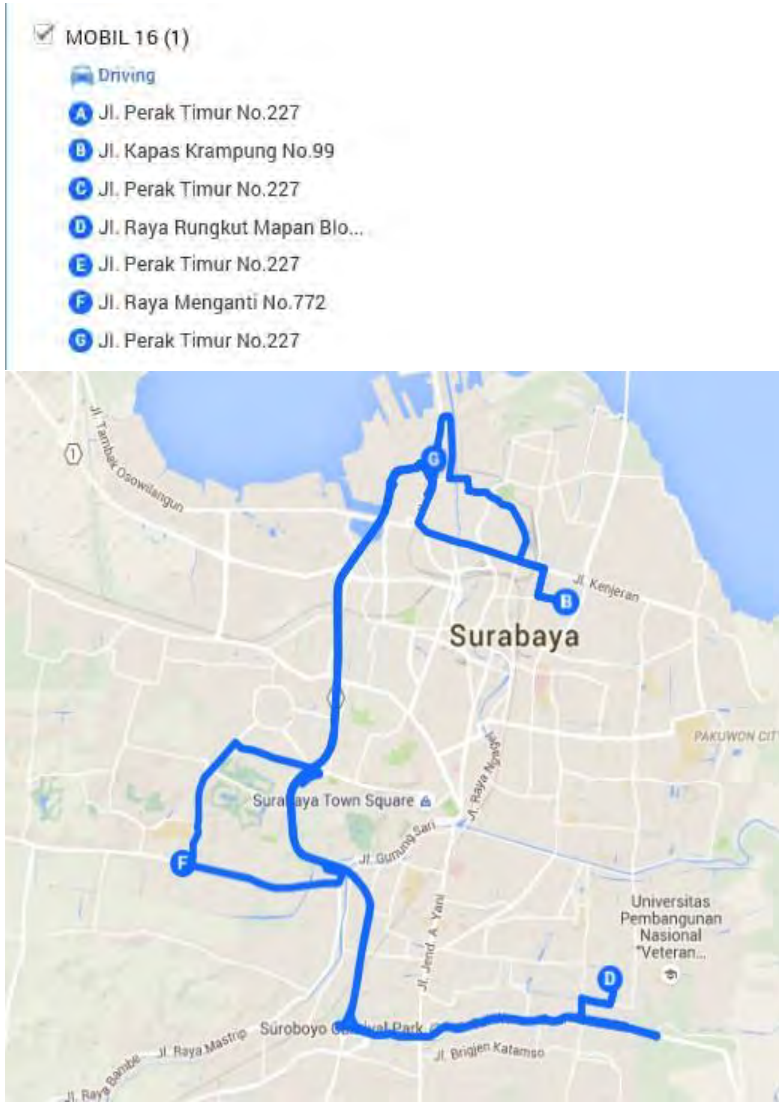
Dalam hasil *running* program, juga didapatkan kurva perbandingan antara nilai deviasi dengan jumlah iterasi. Dan dari program tersebut, didapatkan satu kesimpulan bahwa pada iterasi ke 24, program sudah dapat menemukan nilai deviasi terbaik, yaitu pada nilai 1.677. Grafik kurva tersebut dapat dilihat pada gambar 4.2.



Gambar 4.2 Gambar Konvergensi Nilai Deviasi Total

4.4.2 Hasil dalam bentuk *Plotting Map*

Hasil yang ada pada subbab 4.4.1 tabel 4.20, kemudian di plot pada suatu peta/*map*. Piranti lunak yang digunakan penulis dalam proses ini adalah *googlemymaps* yang disediakan oleh *google maps*. Dari hasil program yang ada, terdapat total 24 rute untuk memenuhi semua *demand*.



Gambar 4.3 *Plotting map* mobil Tipe 16 KL nomor 1

24(1)

Driving

A Jl. Perak Timur No.227

B SPBU Darmo Indah Timur

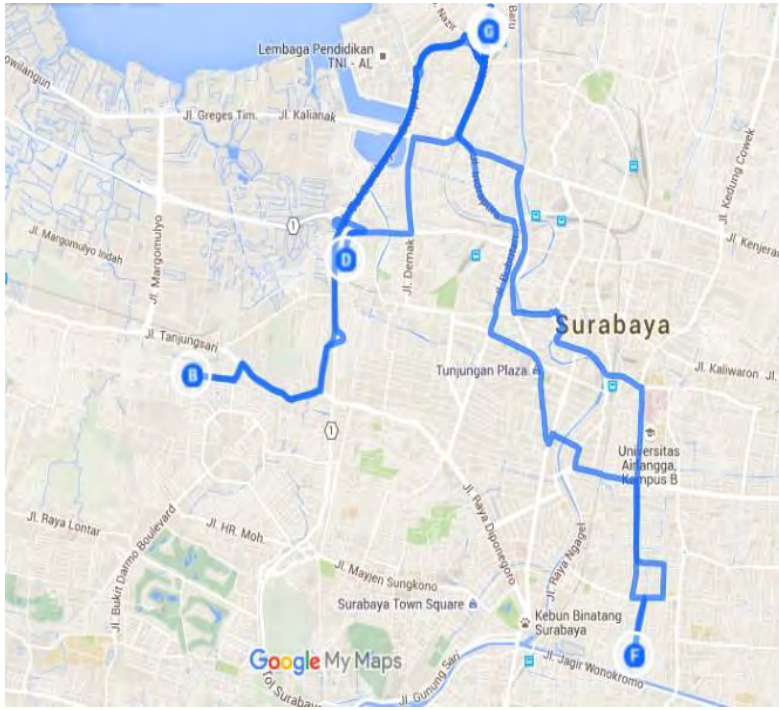
C Jl. Perak Timur No.227

D Jl. Dupak Rukun No.108

E Jl. Perak Timur No.227

F Jl. Bratang Jaya No.59

G Jl. Perak Timur No.227



Gambar 4.4 *Plotting map* mobil Tipe 24 KL nomor 1

24(2)

Driving

A Jl. Perak Timur No.227

B Jl. Raya Mastrip No.116

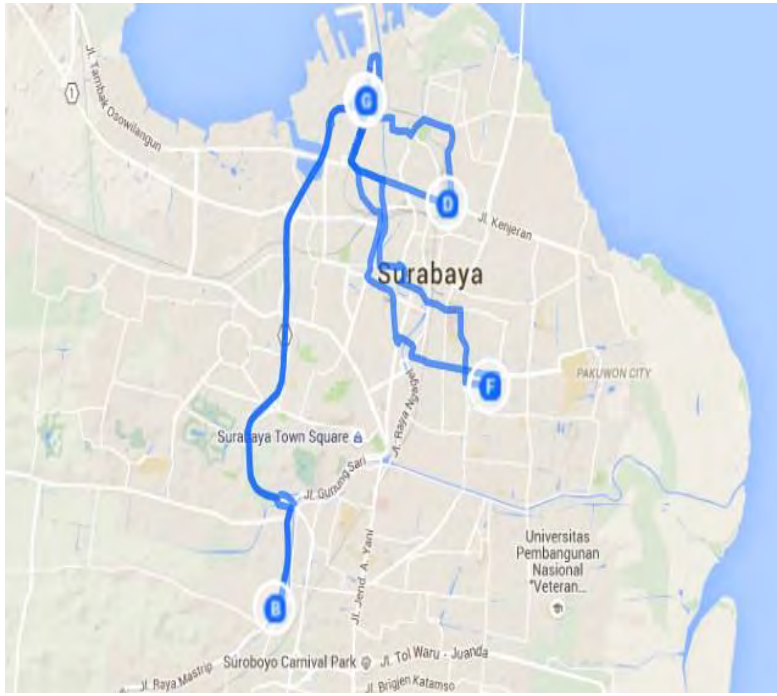
C Jl. Perak Timur No.227

D Jl. Kenjeran No.99

E Jl. Perak Timur No.227

F Jalan Manyar Sabrangan X

G Jl. Perak Timur No.227



Gambar 4.5 *Plotting map* mobil Tipe 24 KL nomor 2

24(3)

Driving

A Jl. Perak Timur No.227

B Jl. Raya Ngagel No.185

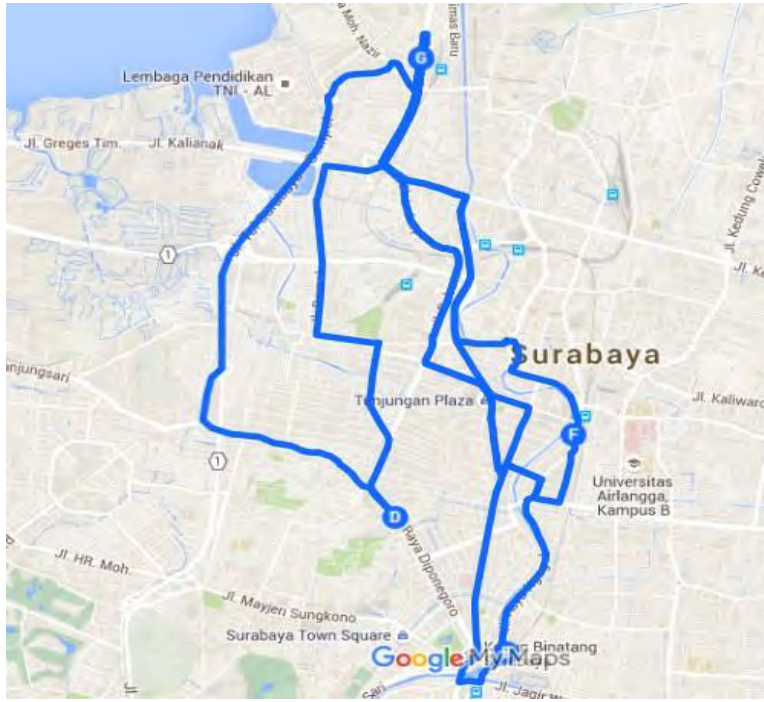
C Jl. Perak Timur No.227

D Jl. Raya Diponegoro No.221

E Jl. Perak Timur No.227

F Jl. Sumatra No.25-29

G Jl. Perak Timur No.227



Gambar 4.6 *Plotting map* mobil Tipe 24 KL nomor 3

24(4)

Driving

A Jl. Perak Timur No.227

B Jalan Manyar Kertoarjo

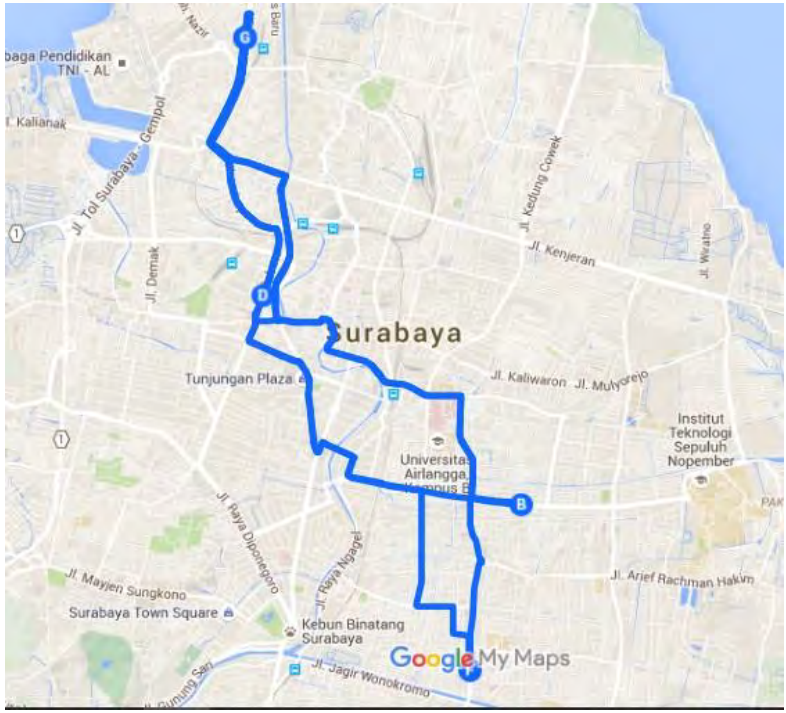
C Jl. Perak Timur No.227

D Jl. Bubutan No.72-76

E Jl. Perak Timur No.227

F Jalan Raya Nginden

G Jl. Perak Timur No.227



Gambar 4.7 *Plotting map* mobil Tipe 24 KL nomor 4

24(5)

Driving

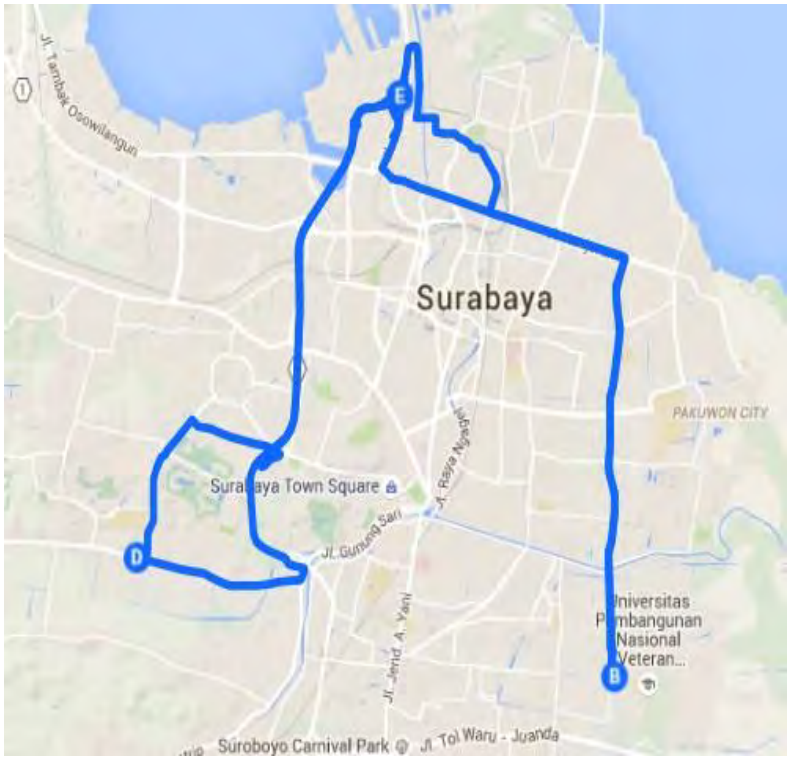
A Jl. Perak Timur No.227

B Jalan Raya Rungkut Madya

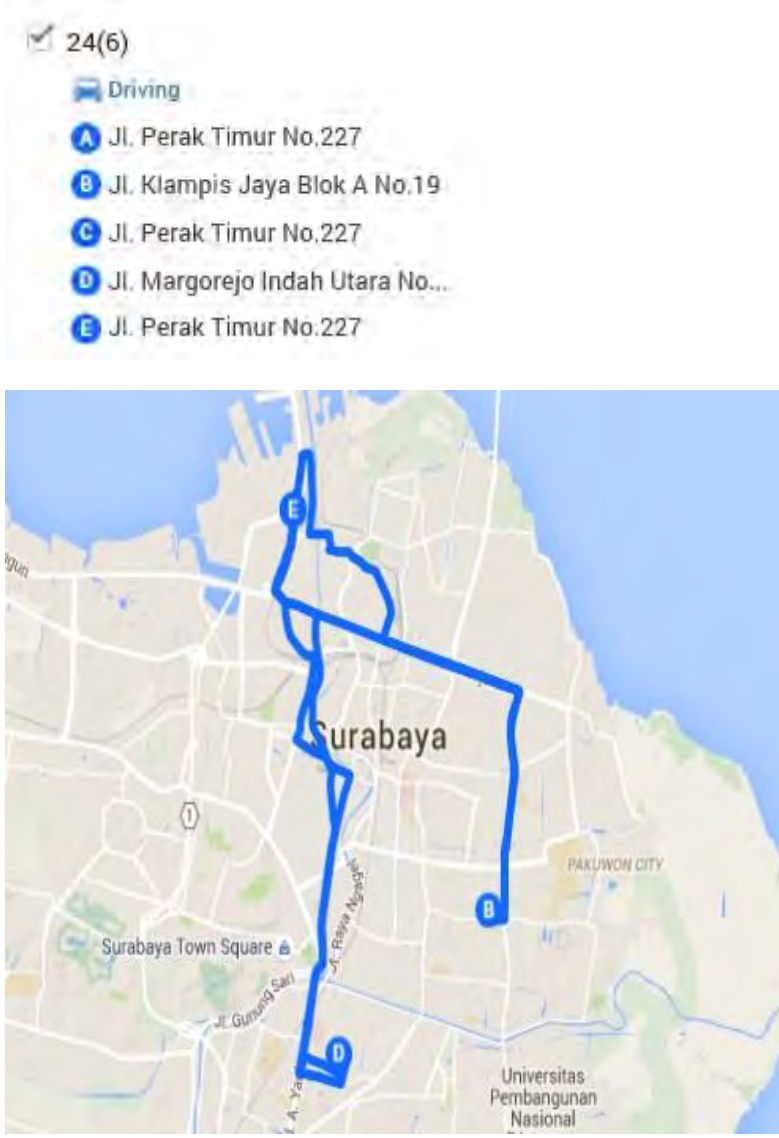
C Jl. Perak Timur No.227

D Jl. Raya Menganti No.772

E Jl. Perak Timur No.227



Gambar 4.8 *Plotting map* mobil Tipe 24 KL nomor 5



Gambar 4.9 *Plotting map* mobil Tipe 24 KL nomor 6

24(7)

Driving

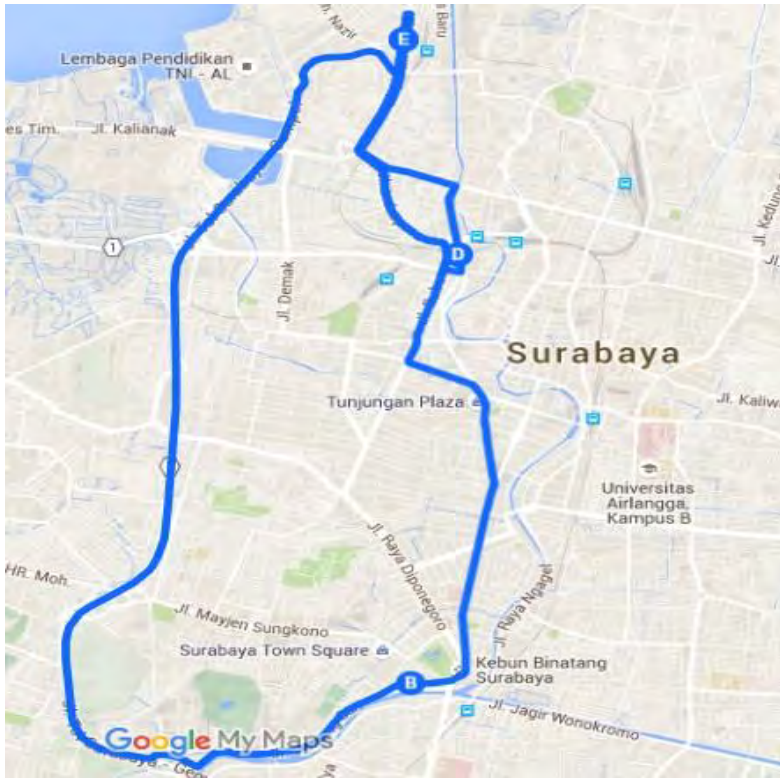
A Jl. Perak Timur No.227

B Jalan Joyoboyo

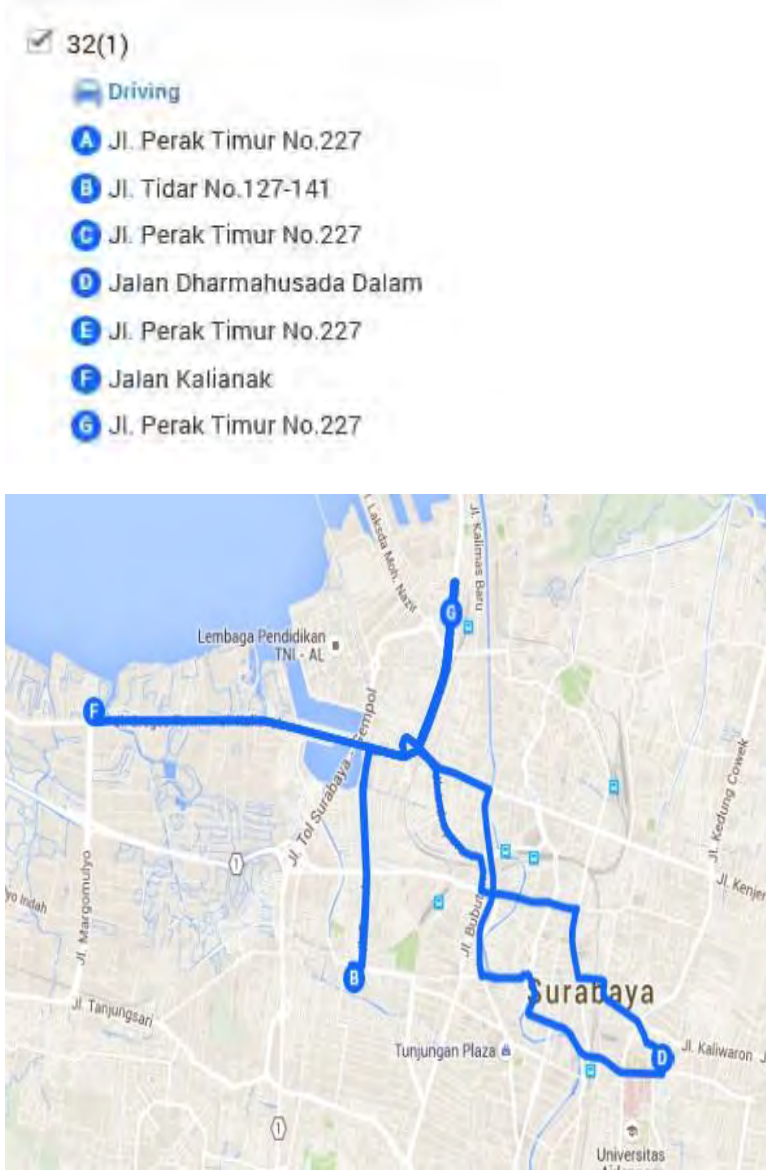
C Jl. Perak Timur No.227

D Jalan Pahlawan

E Jl. Perak Timur No.227



Gambar 4.10 *Plotting map* mobil Tipe 24 KL nomor 7



Gambar 4.11 *Plotting map* mobil Tipe 32 KL nomor 1

32(2)

Driving

A Jl. Perak Timur No.227

B Jl. Ketintang Madya No.46

C Jl. Pecindilan No.50

D Jl. Perak Timur No.227

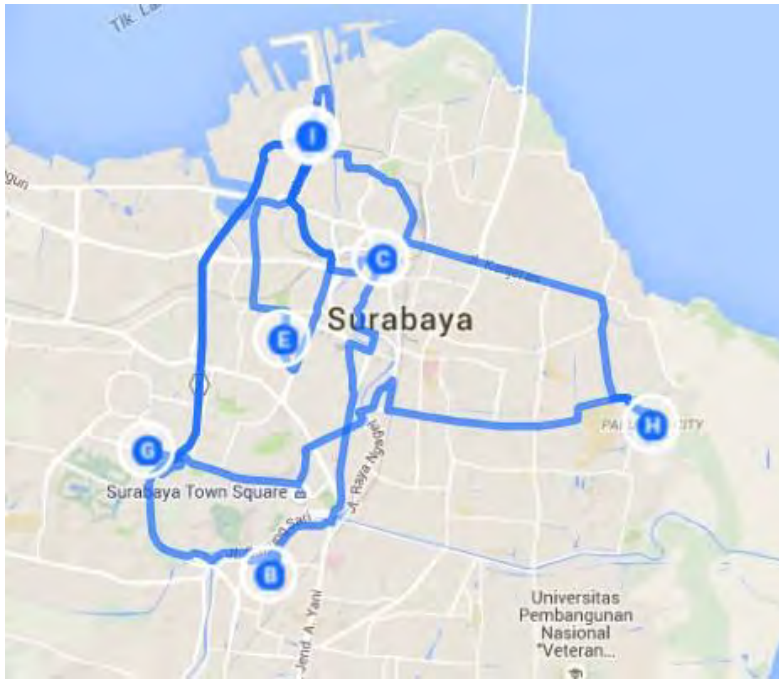
E Jalan Raya Arjuno

F Jl. Perak Timur No.227

G Pertamina Spbu May Jend H...

H Pakuwon City

I Jl. Perak Timur No.227



Gambar 4.12 *Plotting map* mobil Tipe 32 KL nomor 2

32(3)

Driving

A Jl. Perak Timur No.227

B Jalan Jagir Wonokromo

C Jl. Perak Timur No.227

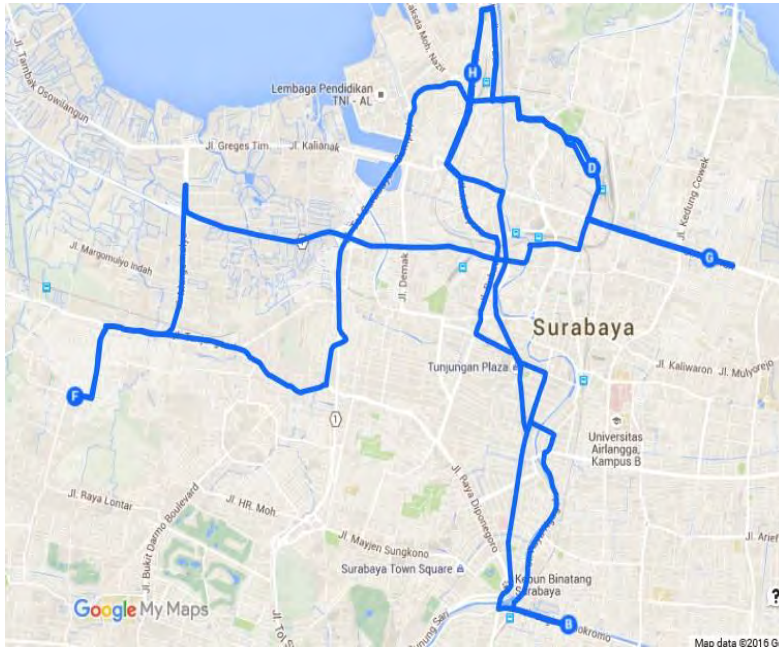
D Jl. Sidotopo Lor No.49-51

E Jl. Perak Timur No.227

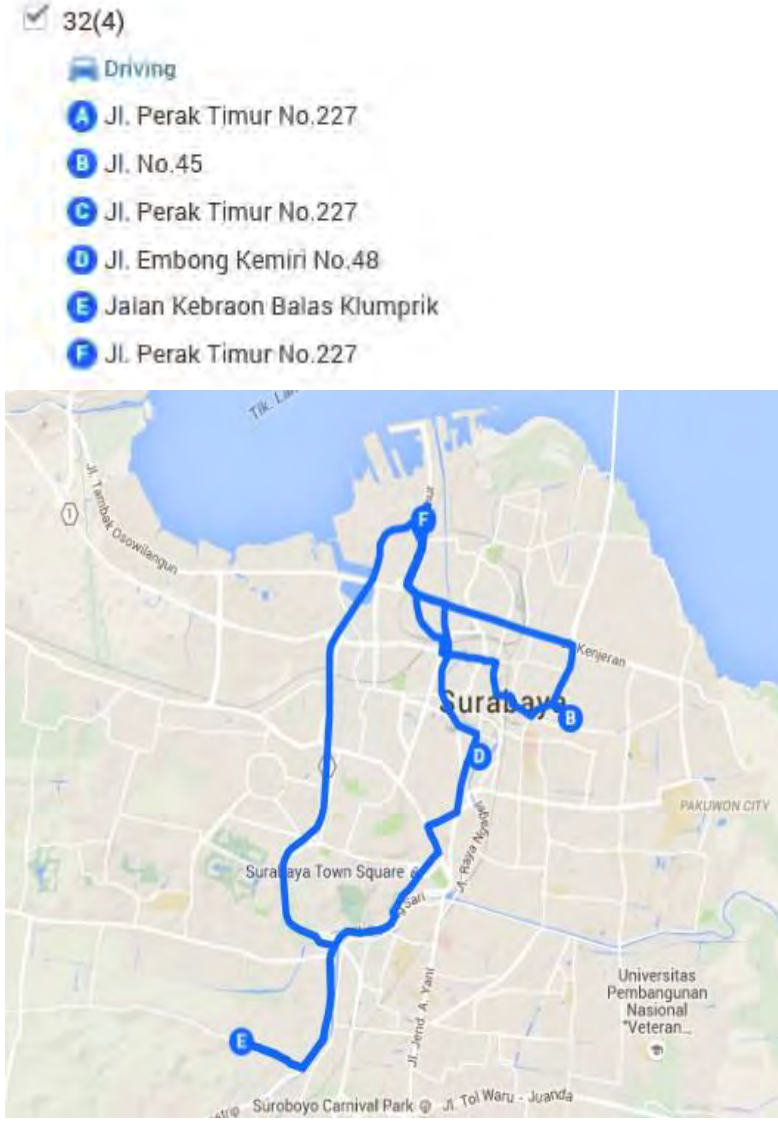
F Jl. Wonorejo Blok I No.30

G Jalan Kenjeran

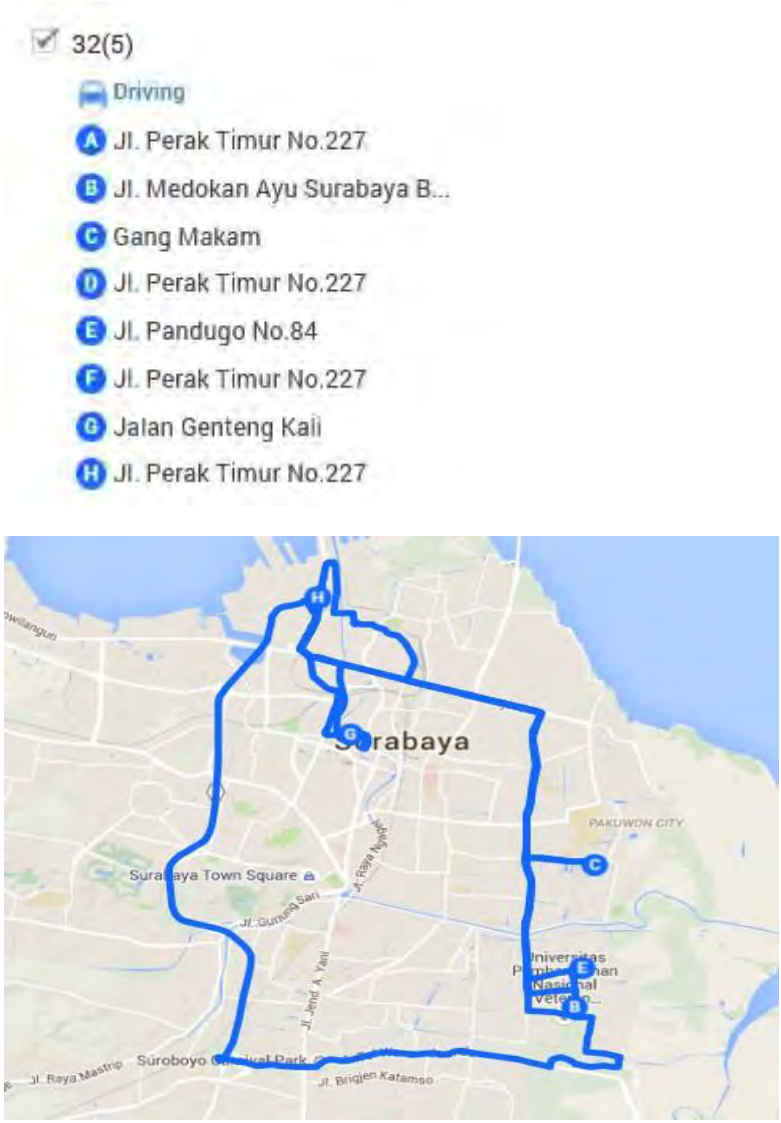
H Jl. Perak Timur No.227



Gambar 4.13 *Plotting map* mobil Tipe 32 KL nomor 3



Gambar 4.14 *Plotting map* mobil Tipe 32 KL nomor 4



Gambar 4.15 *Plotting map* mobil Tipe 32 KL nomor 5

32(6)

Driving

A Jl. Perak Timur No.227

B Jl. Nginden Semolo No.80

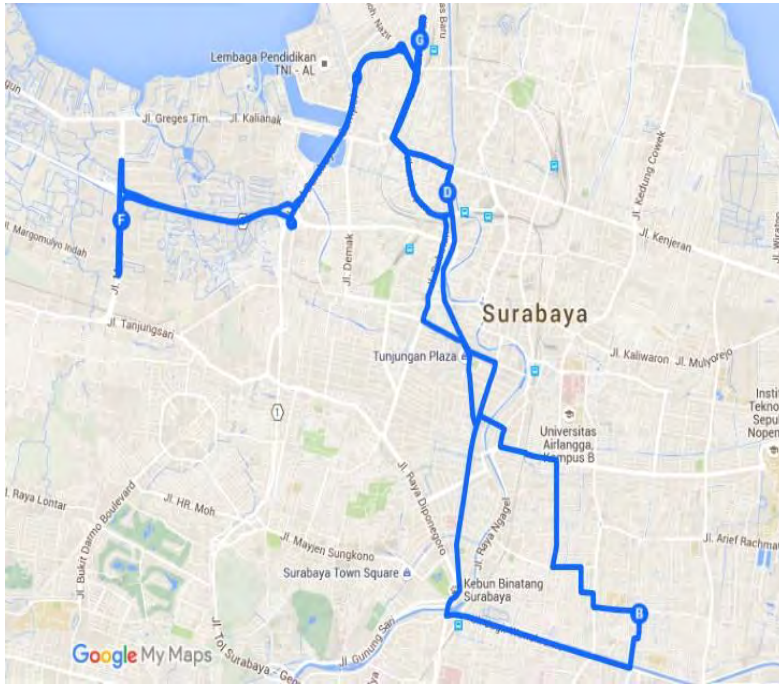
C Jl. Perak Timur No.227

D Jalan Veteran

E Jl. Perak Timur No.227

F Jalan Margomulyo

G Jl. Perak Timur No.227



Gambar 4.16 *Plotting map* mobil Tipe 32 KL nomor 6

32(7)

Driving

A Jl. Perak Timur No.227

B SPBU 54-60105

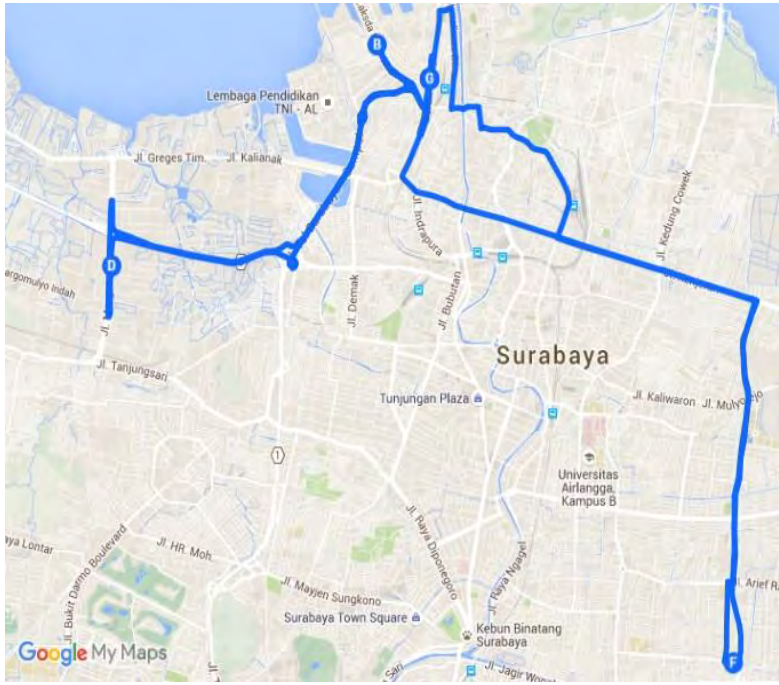
C Jl. Perak Timur No.227

D Jl. Margomulyo No.30

E Jl. Perak Timur No.227

F Jl. Raya Semolowaru No.276

G Jl. Perak Timur No.227



Gambar 4.17 *Plotting map* mobil Tipe 32 KL nomor 7

32(8)

Driving

A Jl. Perak Timur No.227

B Jl. Dr. Soetomo No.87-89

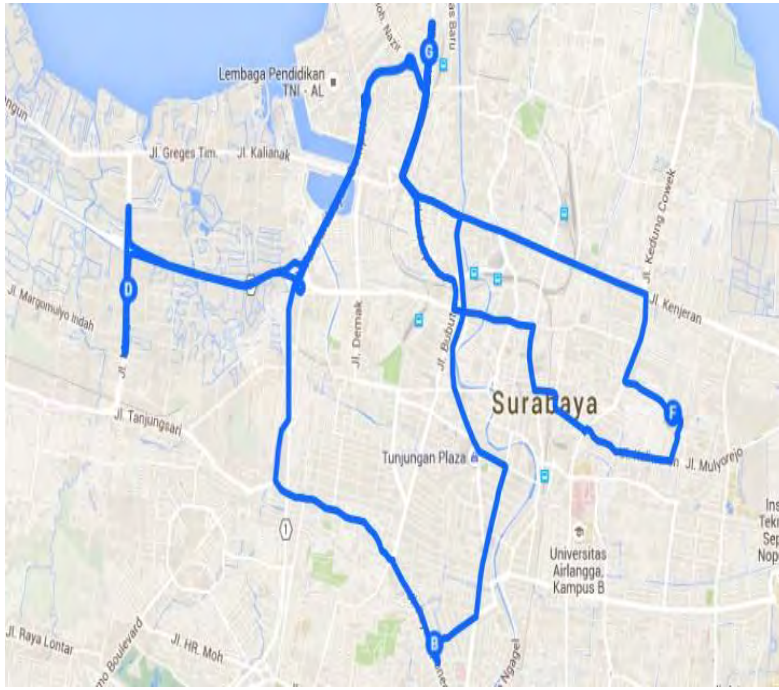
C Jl. Perak Timur No.227

D Jl. Margomulyo No.30

E Jl. Perak Timur No.227

F Jl. Ploso Baru No.183

G Jl. Perak Timur No.227



Gambar 4.18 *Plotting map* mobil Tipe 32 KL nomor 8

32(9)

Driving

A Jl. Perak Timur No.227

B Jl. Raya Ngagel No.185

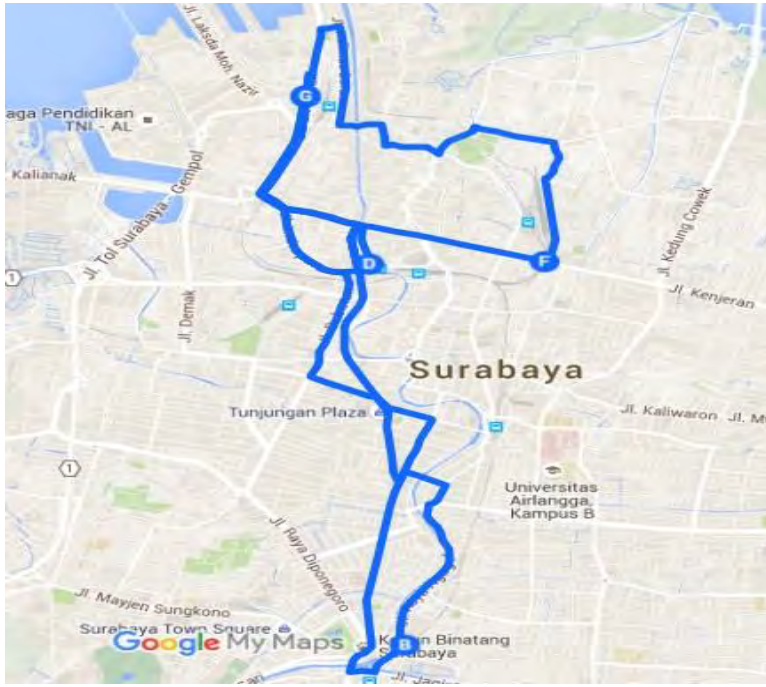
C Jl. Perak Timur No.227

D Jl. St. Kota No.62

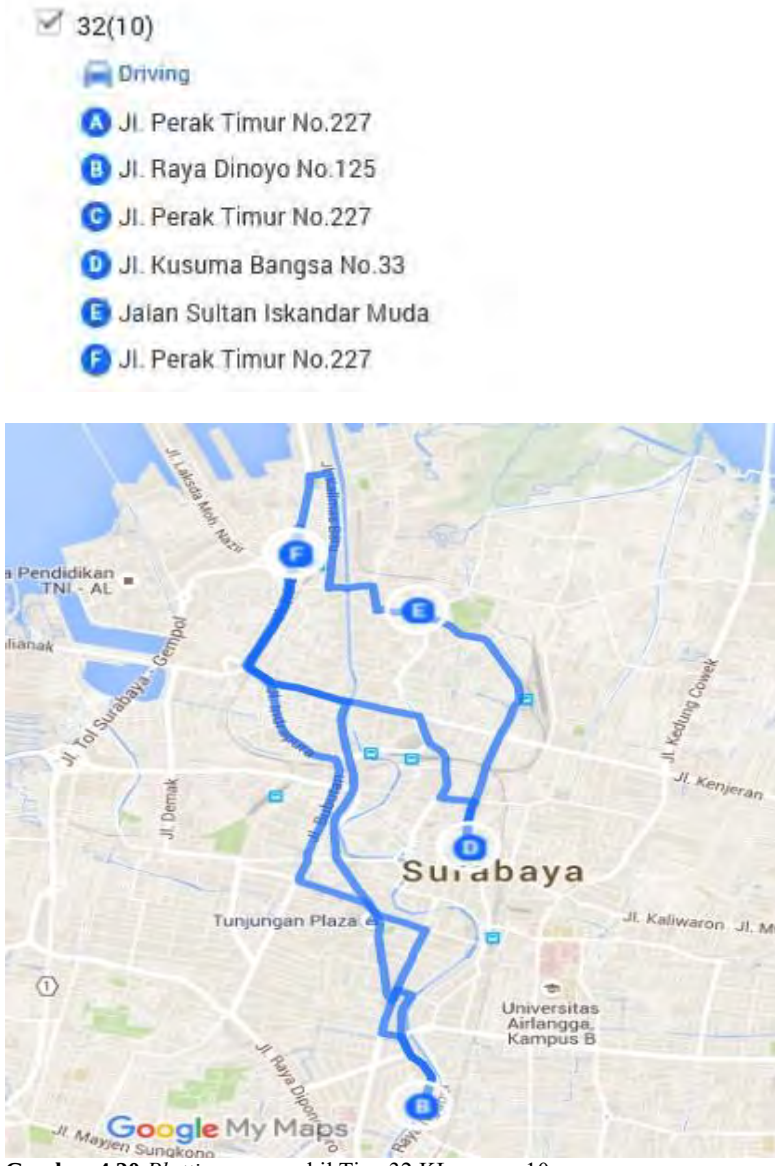
E Jl. Perak Timur No.227

F Jl. Kenjeran No.99

G Jl. Perak Timur No.227



Gambar 4.19 *Plotting map* mobil Tipe 32 KL nomor 9



Gambar 4.20 *Plotting map* mobil Tipe 32 KL nomor 10

40(1)

Driving

A Jl. Perak Timur No.227

B Jalan Kertajaya X - B

C Jl. Menganti Lidah Wetan No...

D Jl. Perak Timur No.227

E Jl. Sulawesi No.8

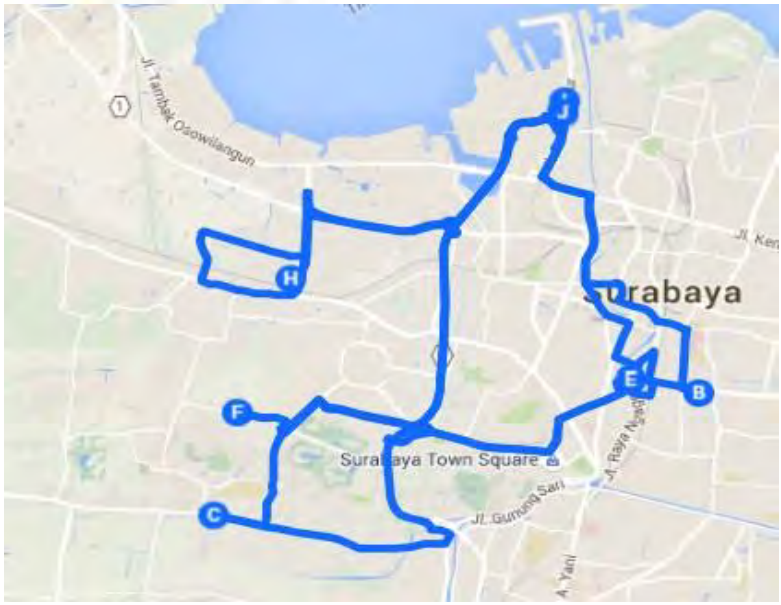
F Jl. Raya Lontar No.123

G Jl. Perak Timur No.227

H Balongsari

I Jalan Perak Barat

J Jl. Perak Timur No.227



Gambar 4.21 *Plotting map* mobil Tipe 40 KL nomor 1

40(2)

Driving

A Jl. Perak Timur No.227

B Jalan Merr Kalijudan

C Jl. Raya Mastrip No.116

D Jl. Perak Timur No.227

E Jl. Ikan Kakap No.21

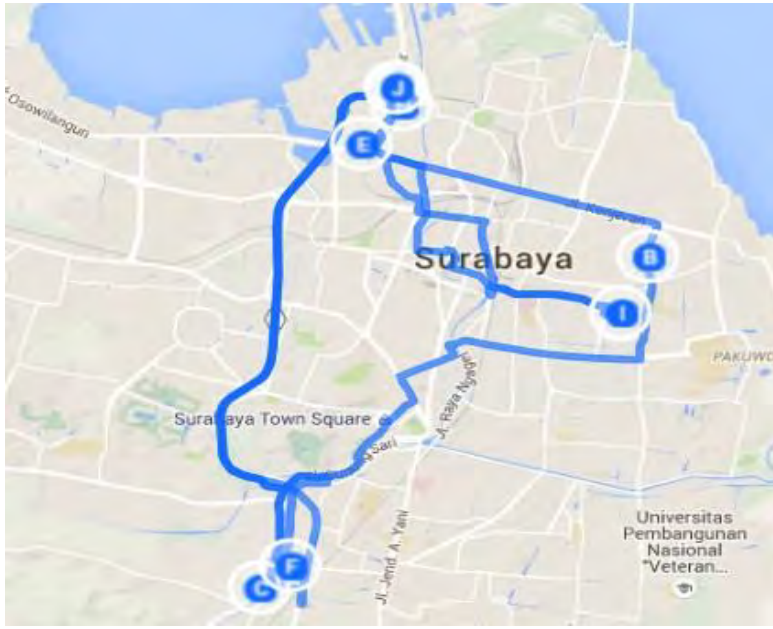
F Jl. Kebonsari Tengah No.65

G Jl. Perak Timur No.227

H Jalan Sisingamangaraja

I Jl. Dharmahusada Indah Uta...

J Jl. Perak Timur No.227



Gambar 4.22 *Plotting map* mobil Tipe 40 KL nomor 2

40(3)

Driving

A Jl. Perak Timur No.227

B Jalan Tanjungsari Indah

C Jalan Jemursari Timur II

D Jl. Perak Timur No.227

E Jl. Jemur Andayani No.35

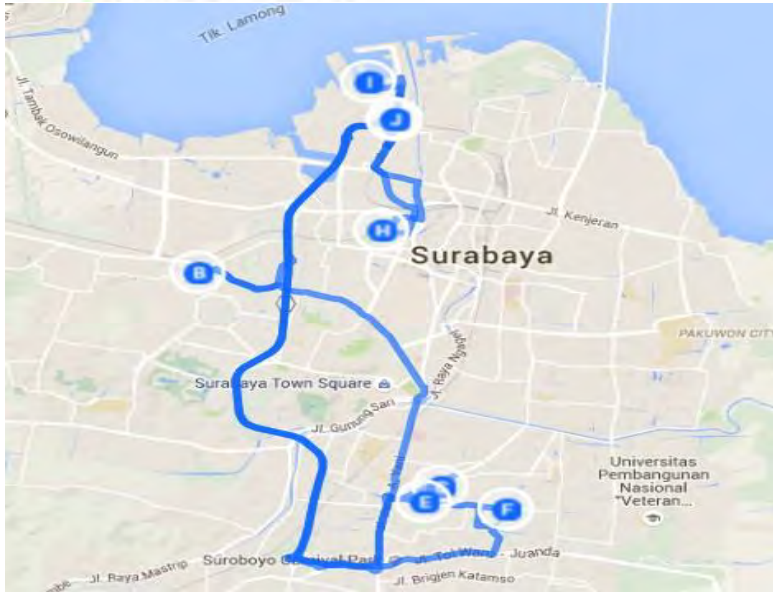
F Jl. Rungkut Industri Raya No...

G Jl. Perak Timur No.227

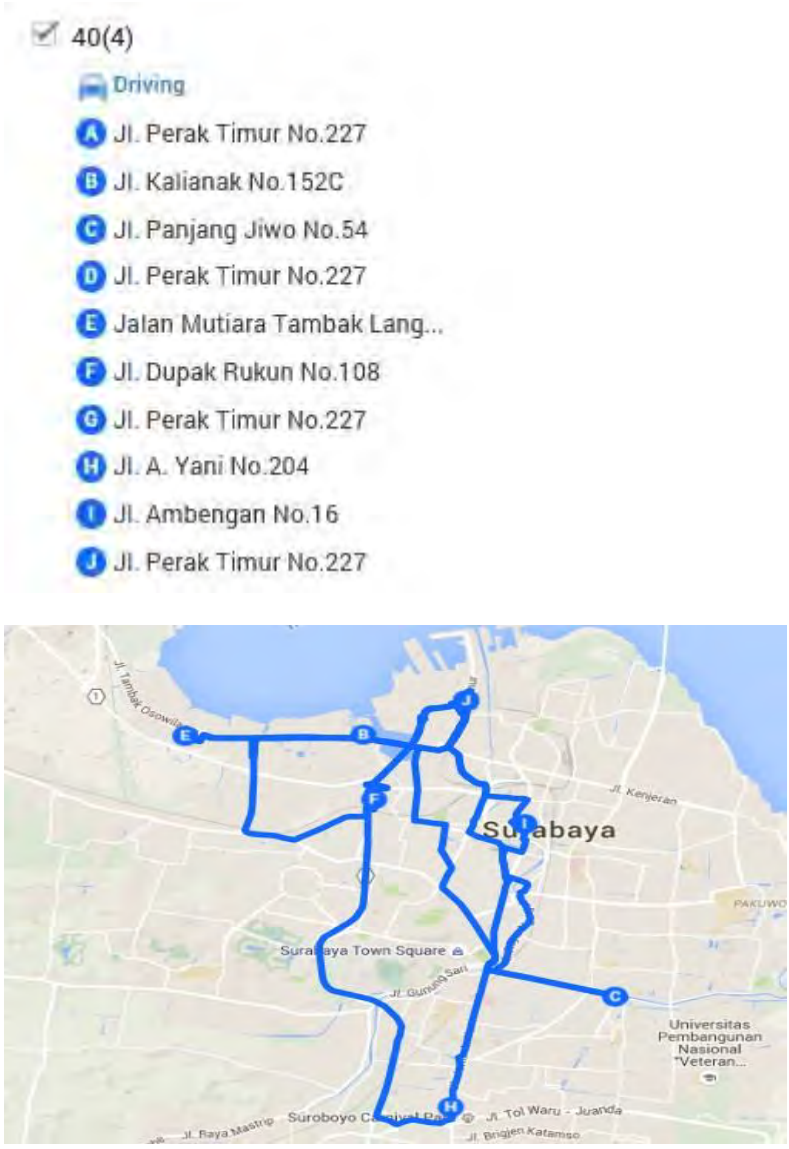
H Jalan Semarang

I Pelabuhan Tanjung Perak

J Jl. Perak Timur No.227



Gambar 4.23 *Plotting map* mobil Tipe 40 KL nomor 3



Gambar 4.24 *Plotting map* mobil Tipe 40 KL nomor 4

40(5)

Driving

A Jl. Perak Timur No.227

B Jl. Bukit Pakal VIII No.104

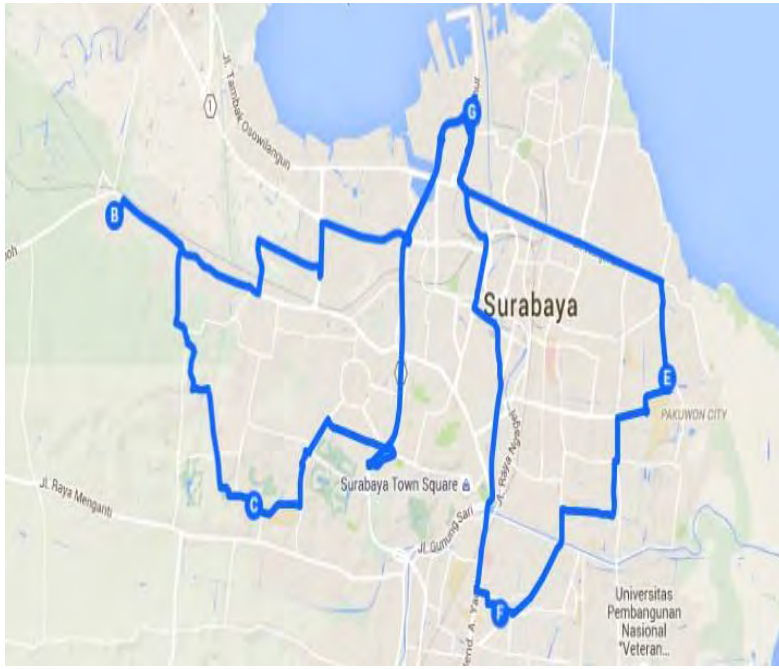
C Honda Citra Cakra

D Jl. Perak Timur No.227

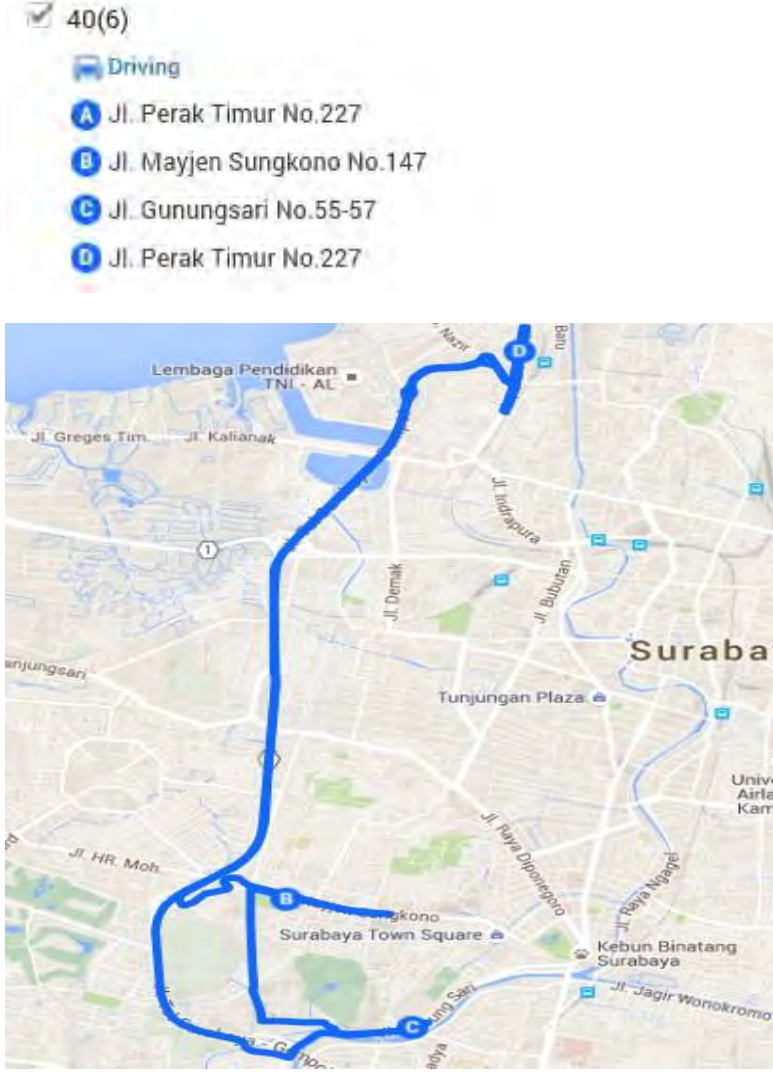
E Jl. Raya Mulyosari No.388

F Jl. Jemursari V No.113

G Jl. Perak Timur No.227



Gambar 4.25 *Plotting map* mobil Tipe 40 KL nomor 5



Gambar 4.26 *Plotting map* mobil Tipe 40 KL no

BAB V
PENGUJIAN DAN HASIL ANALISIS PENGUJIAN

Berdasarkan hasil implementasi sistem yang ada di bab 4, maka langkah selanjutnya adalah melakukan analisis pengujian. Dalam bab ini, hasil yang didapat pada bab 4 akan diuji hasilnya menggunakan beberapa metode pembanding. Ada 2 Metode pembanding yang dilakukan, yaitu dengan menguji parameter *firefly* dan menggunakan metode *nearest neigbor*. Tujuan dari analisa sitem ini adalah untuk menguji sensitifitas dari algoritma *firefly*. Jika performansi metode *firefly* ketika dibandingkan dengan metode lain menghasilkan hasil yang leih bagus, maka algoritma *firefly* ini dapat digunakan maupun dikembangkan.

Langkah awal yang dilakukan adalah dengan melakukan pengujian parameter *firefly*.

5.1 Pengujian Parameter *Firefly*

Pada proses pengujian ini, dilakukan perubahan pada parameter *firefly*. Parameter yang diubah adalah banyaknya iterasi, jumlah populasi *firefly*, dan besarnya nilai gamma atau tingkat penyerapan cahaya. Ketiga parameter tersebut menentukan tingkat pencarian nilai optimal dari permasalahan *vehicle routing and dispatching.*

Tabel 5.1 Parameter SPBU Per *Shift*

Parameter SPBU Per *Shift*	Nilai
SPBU *Shift* 1	29 Unit
SPBU *Shift* 2	28 Unit
SPBU *Shift* 3	28 Unit

5.1.1 Pengujian Jumlah Iterasi

Pada pengujian jumlah iterasi, parameter jumlah iterasi dibuat bervariasi nilainya di angka 10 iterasi, 25 iterasi, dan 50 iterasi. Pada parameter *firefly algorithm* yang lain, yaitu jumlah populasi *firefly* dan nilai gamma, kedua parameter tersebut dibuat fix nilainya. Besarnya jumlah populasi dan nilai gamma berada di nilai 200 dan 1000.

5.1.1.1 Pengujian 10 Iterasi

Pada percobaan menggunakan 10 iterasi, didapatkan nilai deviasi per kendaraan adalah 2.1 jam. Selain itu, total kendaraan yang terpakai

adalah sebanyak 25 unit dengan jarak tempuh mencapai 3939.227 Km. Parameter *firefly* yang digunakan adalah sebagai berikut:

Tabel 5.2 Parameter Pengujian *Firefly* 10 Iterasi

Parameter *Firefly*	Nilai
MaxGeneration	10
Populasi *Firefly*	200
Gamma (γ)	1000

Sedangkan hasil dari pengujian sistem menggunakan 10 iterasi adalah sebagai berikut:

Tabel 5.3 Hasil Uji Parameter 10 Iterasi

TIPE	NOMOR	TOTAL JARAK TEMPUH (Km)	DEVIASI (Jam)
16	1	154.842	2.187
	2	148.994	
	3	138.75	
24	1	165.444	
	2	174.178	
	3	198.162	
	4	187.576	
	5	184.288	
	6	123.2	
	7	36.752	
32	1	183.266	
	2	194.357	
	3	189.942	
	4	192.489	
	5	162.8827	
	6	160.916	
	7	147.464	

TIPE	NOMOR	TOTAL JARAK TEMPUH (Km)	DEVIASI (Jam)
32	8	139.78	2.187
	9	116.268	
	10	122.034	
40	1	181.989	
	2	162.9721	
	3	189.8364	
	4	133.393	
	5	149.452	
TOTAL	25	3939.227	

5.1.1.2 Pengujian 25 Iterasi

Pada percobaan menggunakan 25 iterasi, didapatkan nilai deviasi per kendaraan adalah 1.95384 jam. Selain itu, total kendaraan yang terpakai adalah sebanyak 24 unit dengan jarak tempuh mencapai 3833.405 Km. Parameter *firefly* yang digunakan adalah sebagai berikut:

Tabel 5.4 Parameter Pengujian *Firefly* 25 Iterasi

Parameter *Firefly*	Nilai
MaxGeneration	25
Populasi *Firefly*	200
Gamma (γ)	1000

Sedangkan hasil dari pengujian sistem menggunakan 25 iterasi adalah sebagai berikut:

Tabel 5.5 Hasil Uji Parameter 25 Iterasi

TIPE	NOMOR	TOTAL JARAK TEMPUH (Km)	DEVIASI (Jam)
16	1	140.234	1.95384
	2	143.904	
24	1	177.374	
	2	168.33	

TIPE	NOMOR	TOTAL JARAK TEMPUH (Km)	DEVIASI (Jam)
24	3	163.522	1.95384
	4	164.014	
	5	132.468	
	6	186.326	
	7	106.17	
32	1	156.996	
	2	162.044	
	3	151.0831	
	4	201.509	
	5	147.454	
	6	208.468	
	7	165.947	
	8	198.887	
	9	148.572	
	10	123.81	
	11	112.308	
40	1	202.8222	
	2	153.3904	
	3	186.0177	
	4	131.755	
TOTAL	24	3833.405	

5.1.1.3 Pengujian 50 Iterasi

Pada percobaan menggunakan 50 iterasi, didapatkan nilai deviasi per kendaraan adalah 1.6778 jam. Selain itu, total kendaraan yang terpakai adalah sebanyak 24 unit dengan jarak tempuh mencapai 3843.503 Km. Parameter *firefly* yang digunakan adalah sebagai berikut:

Tabel 5.6 Parameter Pengujian *Firefly* 50 Iterasi

Parameter *Firefly*	Nilai
MaxGeneration	50
Populasi *Firefly*	200
Gamma (γ)	1000

Sedangkan hasil dari pengujian sistem menggunakan 50 iterasi adalah sebagai berikut:

Tabel 5.7 Hasil Uji Parameter 50 Iterasi

TIPE	NOMOR	TOTAL JARAK TEMPUH (Km)	DEVIASI (Jam)
16	1	140.27	1.6778
	2	156.752	
24	1	183.146	
	2	198.614	
	3	173.542	
	4	125.28	
	5	113.902	
	6	154.876	
32	1	162.338	
	2	197.159	
	3	159.101	
	4	205.109	
	5	205.88	
	6	180.104	
	7	186.458	
	8	149.682	
	9	122.939	
	10	197.4241	
40	1	151.0941	

TIPE	NOMOR	TOTAL JARAK TEMPUH (Km)	DEVIASI (Jam)
40	2	174.3124	1.6778
	3	190.5243	
	4	153.56	
	5	161.4365	
	6	140.27	
TOTAL	23	3843.503	

5.1.2 Pengujian Jumlah *Firefly*

Pada pengujian jumlah *firefly*, parameter jumlah *firefly* dibuat bervariasi nilainya di angka 100, 150, 200 populasi. Pada parameter *firefly algorithm* yang lain, yaitu jumlah iterasi dan nilai gamma, kedua parameter tersebut dibuat fix nilainya. Besarnya jumlah iterasi dan nilai gamma berada di nilai 50 dan 1000.

5.1.2.1 Pengujian 100 *Firefly*

Pada percobaan menggunakan 100 *Firefly*, didapatkan nilai deviasi per kendaraan adalah 1.936619 jam. Selain itu, total kendaraan yang terpakai adalah sebanyak 24 unit dengan jarak tempuh mencapai 3931.941 Km. Parameter *firefly* yang digunakan adalah sebagai berikut:

Tabel 5.8 Parameter Pengujian 100 *Firefly*

Parameter *Firefly*	Nilai
MaxGeneration	50
Populasi *Firefly*	100
Gamma (γ)	1000

Sedangkan hasil dari pengujian sistem menggunakan 100 *firefly* adalah sebagai berikut:

Tabel 5.9 Hasil Uji Parameter 100 *Firefly*

TIPE	NOMOR	TOTAL JARAK TEMPUH (Km)	DEVIASI (Jam)
16	1	164.104	1.936619

TIPE	NOMOR	TOTAL JARAK TEMPUH (Km)	DEVIASI (Jam)
16	2	182.104	1.936619
	3	146.136	
	4	116.75	
24	1	181.398	
	2	175.722	
	3	193.73	
	4	116.684	
	5	121.344	
32	1	158.27	
	2	175.278	
	3	153.342	
	4	127.936	
	5	182.182	
	6	177.187	
	7	194.78	
	8	168.994	
	9	143.406	
	10	75.59	
40	1	199.427	
	2	194.792	
	3	187.9732	
	4	209.6129	
	5	185.199	
TOTAL	24	3931.941	

5.1.2.2 Pengujian 150 *Firefly*

Pada percobaan menggunakan 150 *Firefly*, didapatkan nilai deviasi per kendaraan adalah 1.893886 jam. Selain itu, total kendaraan yang

terpakai adalah sebanyak 23 unit dengan jarak tempuh mencapai 3812.755 Km. Parameter *firefly* yang digunakan adalah sebagai berikut:

Tabel 5.10 Parameter Pengujian 150 *Firefly*

Parameter *Firefly*	Nilai
MaxGeneration	50
Populasi *Firefly*	150
Gamma (γ)	1000

Sedangkan hasil dari pengujian sistem menggunakan 200 *firefly* adalah sebagai berikut:

Tabel 5.11 Hasil Uji Parameter 150 *Firefly*

TIPE	NOMOR	TOTAL JARAK TEMPUH (Km)	DEVIASI (Jam)
16	1	187.628	1.893886
	2	71.894	
24	1	161.37	
	2	209.284	
	3	169.56	
	4	174.218	
	5	156.046	
	6	187.846	
32	1	180.8603	
	2	150.784	
	3	178.732	
	4	175.169	
	5	189.1604	
	6	171.598	
	7	162.532	
	8	190.438	
	9	136.248	

TIPE	NOMOR	TOTAL JARAK TEMPUH (Km)	DEVIASI (Jam)
32	10	74.908	1.893886
40	1	159.4209	
	2	203.879	
	3	170.5056	
	4	177.4826	
	5	173.1907	
TOTAL	23	3812.755	

5.1.2.3 Pengujian 200 *Firefly*

Pada percobaan menggunakan 200 *Firefly*, didapatkan nilai deviasi per kendaraan adalah 1.677628 jam. Selain itu, total kendaraan yang terpakai adalah sebanyak 23 unit dengan jarak tempuh mencapai 3843.503 Km. Parameter *firefly* yang digunakan adalah sebagai berikut:

Tabel 5.12 Parameter Pengujian 200 *Firefly*

Parameter *Firefly*	Nilai
MaxGeneration	50
Populasi *Firefly*	200
Gamma (γ)	1000

Sedangkan hasil dari pengujian sistem menggunakan 200 *firefly* adalah sebagai berikut:

Tabel 5.13 Hasil Uji Parameter 200 *Firefly*

TIPE	NOMOR	TOTAL JARAK TEMPUH (Km)	DEVIASI (Jam)
16	1	140.27	1.677628
24	1	156.752	
	2	183.146	
	3	198.614	
	4	173.542	
	5	125.28	

TIPE	NOMOR	TOTAL JARAK TEMPUH (Km)	DEVIASI (Jam)
24	6	113.902	1.677628
32	1	154.876	
	2	162.338	
	3	197.159	
	4	159.101	
	5	205.109	
	6	205.88	
	7	180.104	
	8	186.458	
	9	149.682	
	10	122.939	
40	1	197.4241	
	2	151.0941	
	3	174.3124	
	4	190.5243	
	5	153.56	
	6	161.4365	
TOTAL	23	3843.503	

5.1.3 Pengujian Nilai Gamma

Pada pengujian nilai gamma, parameter nilai gamma dibuat bervariasi nilainya di angka 100, 500, dan 1000. Pada parameter *firefly algorithm* yang lain, yaitu jumlah iterasi dan jumlah populasi *firefly,* kedua parameter tersebut dibuat fix nilainya. Besarnya jumlah populasi dan nilai gamma berada di nilai 50 dan 200.

5.1.3.1 Pengujian Nilai Gamma di Angka 100

Pada percobaan menggunakan nilai gamma di angka 100, didapatkan nilai deviasi per kendaraan adalah 1.9549 jam. Selain itu, total kendaraan yang terpakai adalah sebanyak 25 unit dengan jarak tempuh

mencapai 4150.714 Km. Parameter *firefly* yang digunakan adalah sebagai berikut:

Tabel 5.14 Parameter Pengujian Nilai Gamma di Angka 100

Parameter *Firefly*	Nilai
MaxGeneration	50
Populasi *Firefly*	200
Gamma (γ)	100

Sedangkan hasil dari pengujian sistem menggunakan nilai gamma di angka 100 adalah sebagai berikut:

Tabel 5.15 Hasil Uji Parameter Nilai Gamma di Angka 100

TIPE	NOMOR	TOTAL JARAK TEMPUH (Km)	DEVIASI (Jam)
16	1	170.232	1.9549
	2	118.73	
24	1	114.176	
	2	180.078	
	3	170.376	
	4	190.454	
	5	148.704	
	6	184.578	
32	1	185.984	
	2	146.354	
	3	178.454	
	4	164.854	
	5	164.78	
	6	190.9544	
	7	193.524	
	8	193.541	
	9	147.004	
	10	81.846	

TIPE	NOMOR	TOTAL JARAK TEMPUH (Km)	DEVIASI (Jam)
32	11	92.358	1.9549
	12	206.558	
40	1	203.101	
	2	204.645	
	3	200.715	
	4	122.582	
	5	196.132	
TOTAL	25	4150.714	

5.1.3.2 Pengujian Nilai Gamma di Angka 500

Pada percobaan menggunakan nilai gamma di angka 500, didapatkan nilai deviasi per kendaraan adalah 1.7704 jam. Selain itu, total kendaraan yang terpakai adalah sebanyak 24 unit dengan jarak tempuh mencapai 4132.760 Km. Parameter *firefly* yang digunakan adalah sebagai berikut:

Tabel 5.16 Parameter Pengujian Nilai Gamma di Angka 500

Parameter *Firefly*	Nilai
MaxGeneration	50
Populasi *Firefly*	200
Gamma (γ)	500

Sedangkan hasil dari pengujian sistem menggunakan nilai gamma di angka 100 adalah sebagai berikut:

Tabel 5.17 Hasil Uji Parameter Nilai Gamma di Angka 500

TIPE	NOMOR	TOTAL JARAK TEMPUH (Km)	DEVIASI (Jam)
16	1	170.232	1.7704
24	1	118.73	
	2	114.176	
	3	180.078	

TIPE	NOMOR	TOTAL JARAK TEMPUH (Km)	DEVIASI (Jam)
24	4	170.376	1.7704
	5	190.454	
	6	148.704	
32	1	184.578	
	2	185.984	
	3	146.354	
	4	178.454	
	5	164.854	
	6	164.78	
	7	190.9544	
	8	193.524	
	9	193.541	
	10	147.004	
40	1	81.846	
	2	92.358	
	3	206.558	
	4	203.101	
	5	204.645	
	6	200.715	
	7	122.582	
TOTAL	24	4132.760	

5.1.3.3 Pengujian Nilai Gamma di Angka 1000

Pada percobaan menggunakan nilai gamma di angka 1000, didapatkan nilai deviasi per kendaraan adalah 1.677628 jam. Selain itu, total kendaraan yang terpakai adalah sebanyak 23 unit dengan jarak tempuh mencapai 3843.503 Km. Parameter *firefly* yang digunakan adalah sebagai berikut:

Tabel 5.18 Parameter Pengujian Nilai Gamma di Angka 1000

Parameter *Firefly*	Nilai
MaxGeneration	50
Populasi *Firefly*	200
Gamma (γ)	1000

Sedangkan hasil dari pengujian sistem menggunakan nilai gamma di angka 1000 adalah sebagai berikut:

Tabel 5.19 Hasil Uji Parameter Nilai Gamma di Angka 1000

TIPE	NOMOR	TOTAL JARAK TEMPUH (Km)	DEVIASI (Jam)
16	1	140.27	1.677628
24	1	156.752	
	2	183.146	
	3	198.614	
	4	173.542	
	5	125.28	
	6	113.902	
32	1	154.876	
	2	162.338	
	3	197.159	
	4	159.101	
	5	205.109	
	6	205.88	
	7	180.104	
	8	186.458	
	9	149.682	
	10	122.939	
40	1	197.4241	
	2	151.0941	

TIPE	NOMOR	TOTAL JARAK TEMPUH (Km)	DEVIASI (Jam)
40	3	174.3124	1.677628
	4	190.5243	
	5	153.56	
	6	161.4365	
TOTAL	23	3843.503	

5.2 Pengujian Komposisi SPBU per *Shift*

Pada proses pengujian ini, dilakukan perubahan komposisi SPBU per *shift*. Parameter yang diubah adalah banyaknya SPBU per *shift* serta jumlah total dari SPBU yang disuplai. Dalam hal ini, ada 3 perbandingan yang dilakukan. Yang pertama dengan melakukan suplai ke 85 SPBU. Kemudian menggunakan 90 SPBU. Dan yang terakhir melakukan suplai ke 95 SPBU. Parameter *firefly* yang digunakan untuk melakukan kedua proses tersebut adalah sesuai dengan yang ada pada tabel 5.14

Tabel 5.20 Parameter *firefly* Pengujian SPBU Per *Shift*

Parameter	Nilai
MaxGeneration	50
Populasi *Firefly*	200
Gamma (γ)	1000

5.2.1 Pengujian 85 SPBU

Pada percobaan menggunakan 85 SPBU, didapatkan nilai deviasi per kendaraan adalah 1.677628 jam. Selain itu, total kendaraan yang terpakai adalah sebanyak 23 unit dengan jarak tempuh mencapai 3843.503 Km. Pembagian SPBU per *shift* adalah sebagai berikut:

Tabel 5.21 Pembagian SPBU per *Shift* dengan 85 SPBU Total

Nomor *Shift*	Jumlah SPBU
Shift 1	29 Unit
Shift 2	28 Unit
Shift 3	28 Unit

Sedangkan hasil dari pengujian menggunakan 85 SPBU adalah sebagai berikut:

Tabel 5.22 Hasil Uji Penggunaan 85 SPBU

TIPE	NOMOR	TOTAL JARAK TEMPUH (Km)	DEVIASI (Jam)
16	1	140.27	
24	1	156.752	
	2	183.146	
	3	198.614	
	4	173.542	
	5	125.28	
	6	113.902	
32	1	154.876	
	2	162.338	
	3	197.159	
	4	159.101	
	5	205.109	
	6	205.88	1.677628
	7	180.104	
	8	186.458	
	9	149.682	
	10	122.939	
40	1	197.4241	
	2	151.0941	
	3	174.3124	
	4	190.5243	
	5	153.56	
	6	161.4365	
TOTAL	23	3843.503	

5.2.2 Penggunaan 90 SPBU

Pada percobaan menggunakan 90 SPBU, didapatkan nilai deviasi per kendaraan adalah 1.712544 jam. Selain itu, total kendaraan yang terpakai adalah sebanyak 26 unit dengan jarak tempuh mencapai 4354.56 Km. Pembagian SPBU per *shift* adalah sebagai berikut:

Tabel 5.23 Pembagian SPBU Per *Shift* dengan 90 SPBU Total

Nomor *Shift*	Jumlah SPBU
Shift 1	30 Unit
Shift 2	30 Unit
Shift 3	30 Unit

Sedangkan hasil dari pengujian menggunakan 90 SPBU adalah sebagai berikut:

Tabel 5.24 Hasil Uji Penggunaan 90 SPBU

TIPE	NOMOR	TOTAL JARAK TEMPUH (Km)	DEVIASI (Jam)
16	1	160.686	1.712544
	2	132.076	
24	1	157.474	
	2	208.706	
	3	184.948	
	4	157.528	
	5	178.654	
	6	181.862	
	7	149.754	
32	1	195.976	
	2	188.5528	
	3	162.546	
	4	178.8981	
	5	173.168	
	6	149.2418	

TIPE	NOMOR	TOTAL JARAK TEMPUH (Km)	DEVIASI (Jam)
32	7	155.198	1.712544
	8	139.664	
	9	163.2397	
	10	134.71	
	11	194.442	
	12	128.448	
40	1	202.41	
	2	201.9083	
	3	137.33	
	4	193.6183	
	5	143.521	
TOTAL	26	4354.56	

5.2.3 Penggunaan 95 SPBU

Pada percobaan menggunakan 95 SPBU, didapatkan nilai deviasi per kendaraan adalah 1.846402 jam. Selain itu, total kendaraan yang terpakai adalah sebanyak 29 unit dengan jarak tempuh mencapai 4798.938 Km. Pembagian SPBU per *shift* adalah sebagai berikut:

Tabel 5.25 Pembagian SPBU per *Shift* dengan 95 SPBU Total

Nomor *Shift*	Jumlah SPBU
Shift 1	32 Unit
Shift 2	32 Unit
Shift 3	31 Unit

Sedangkan hasil dari pengujian menggunakan 95 SPBU adalah sebagai berikut:

Tabel 5.26 Hasil Uji Penggunaan 95 SPBU

TIPE	NOMOR	TOTAL JARAK TEMPUH (Km)	DEVIASI (Jam)

TIPE	NOMOR	TOTAL JARAK TEMPUH (Km)	DEVIASI (Jam)
16	1	206.574	1.846402
	2	124.206	
24	1	185.026	
	2	185.866	
	3	188.5	
	4	178.912	
	5	193.43	
	6	118.436	
	7	116.094	
32	1	172.559	
	2	134.218	
	3	172.144	
	4	177.596	
	5	140.728	
	6	177.666	
	7	197.308	
	8	158.042	
	9	171.549	
	10	187.746	
	11	150.876	
	12	196.871	
	13	119.274	
40	1	157.804	
	2	181.54	
	3	182.784	
	4	180.938	
	5	193.466	

TIPE	NOMOR	TOTAL JARAK TEMPUH (Km)	DEVIASI (Jam)
40	6	174.257	1.846402
	7	74.528	
TOTAL	29	4798.938	

5.3 Pengujian dengan Metode *Nearest Neigbor*

Pada percobaan menggunakan metode *Nearest Neigbor*, didapatkan nilai deviasi per kendaraan adalah 5.051 jam. Selain itu, total kendaraan yang terpakai adalah sebanyak 27 unit dengan jarak tempuh mencapai 2193.9177 Km.

Tabel 5.27 Hasil Uji Coba Menggunakan Metode *Nearest Neigbor*

TIPE	NOMOR	TOTAL JARAK TEMPUH (Km)	DEVIASI (Jam)
16	1	122.136	3.7792
	2	36.1	
24	1	122.88	
	2	151.812	
	3	161.636	
	4	182.26	
	5	136.234	
	6	145.844	
	7	74.932	
	8	98.869	
32	1	114.124	
	2	121.267	
	3	134.585	
	4	161.136	
	5	176.28	
	6	192.366	
	7	204.097	
	8	55.872	
	9	67.814	
	10	70.322	

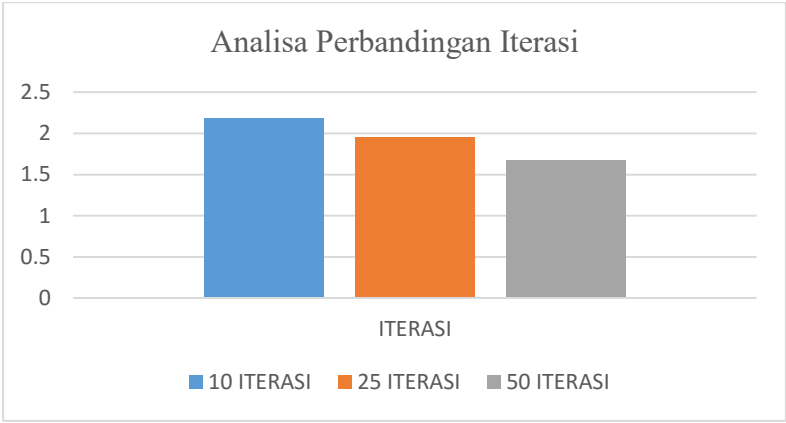
TIPE	NOMOR	TOTAL JARAK TEMPUH (Km)	DEVIASI (Jam)
32	11	74.908	3.7792
	12	78.854	
	13	91.454	
	14	113.4721	
40	1	133.785	
	2	79.334	
	3	44.693	
	4	80.246	
	5	102.4	
TOTAL	29	3329.712	

5.4 Analisa Perbandingan

Pada subbab ini akan dijelaskan mengenai perandingan hasil ujicoba parameter dan perbandingan metode. Nilai acuan yang dijadikan perbandingan adalah nilai deviasi yang dihasilkan oleh *firefly algorithm* dengan parameter *MaxGeneration* sebesar 50, banyaknya populasi *firefly* yang digunakan adalah 200, dan nilai gamma yang digunakan sebesar 1000. Analisa perbandingan ini dibutuhkan untuk mengukur tingkat keoptimalan dan keakuratan *firefly algorithm* serta dapat digunakan sebagai sarana pembelajaran metode di kemudian hari.

5.4.1 Analisa Menggunakan Perbandingan Iterasi

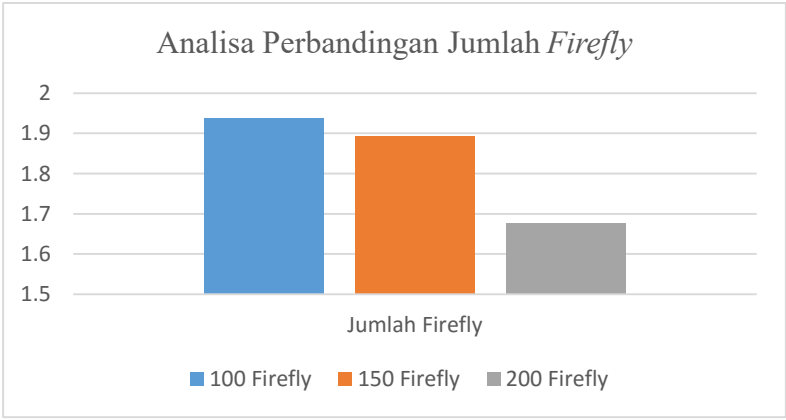
Dari hasil analisa, didaptkan suatu kesimpulan bahwa semakin banyak iterasi yang digunakan, maka nilai deviasi yang didapatkan akan jauh lebih baik. Hal ini didasarkan pada karakteristik *firefly algorithm* yang membandingkan nilai objektif yang dihasilkan dari setiap iterasi. Semakin banyak iterasi yang digunakan, maka nilai objektif yang dihasilkan akan semakin banyak sesuai banyaknya iterasi. Banyaknya iterasi akan membuat sarana pembanding nilai objektif yang jauh lebih variatif. Namun karena *firefly algorithm* merupakan suatu proses komputasi, maka seiring bertambahnya iterasi akan menimbulkan proses komputasi data yang lebih lama. Hasil perbandingan iterasi dapat dilihat pada gambar 5.1.



Gambar 5.1 Analisa Perbandingan Iterasi

5.4.2 Analisa Menggunakan Perbandingan Jumlah *Firefly*

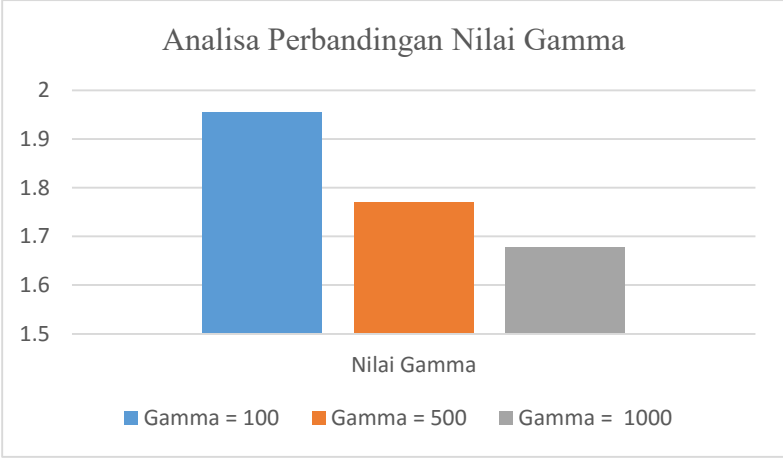
Dari hasil analisa, didapatkan suatu kesimpulan bahwa semakin banyak jumlah *firefly* yang digunakan, maka nilai deviasi yang didapat akan jauh lebih baik. Siste *firefly algorithm* membandingkan nilai tiap individu *firefly* yang ada dalam satu iterasi. Apabila jumlah *firefly* yang digunakan semakin banyak, maka sarana pembanding nilai objektif akan semakin bervariasi. Hasil analisa perbandingan jumlah *firefly* dapat dilihat pada gambar 5.2.



Gambar 5.2 Analisa Perbandingan Jumlah *Firefly*

5.4.3 Analisa Menggunakan Perbandingan Nilai Gamma

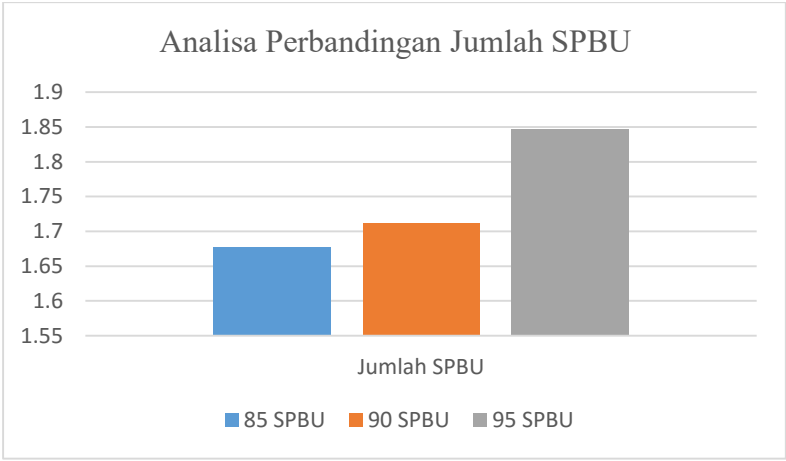
Dari hasil analisa, didapatkan suatu kesimpulan bahwa semakin besar nilai gamma yang digunakan, maka nilai deviasi yang dihasilkan semakin baik. Hal ini didasarkan pada arti nilai gamma itu sendiri. Semakin besar nilai gamma, maka pergerakan *firefly* akan semakin berkurang yang menyebabkan hanya ada perbandingan kecerahan saja pada tiap *firefly*. Hasil analisa perbandingan nilai gamma dapat dilihat pada gambar 5.3.



Gambar 5.3 Analisa Perbandingan Nilai Gamma

5.4.4 Analisa Menggunakan Perbandingan Jumlah SPBU

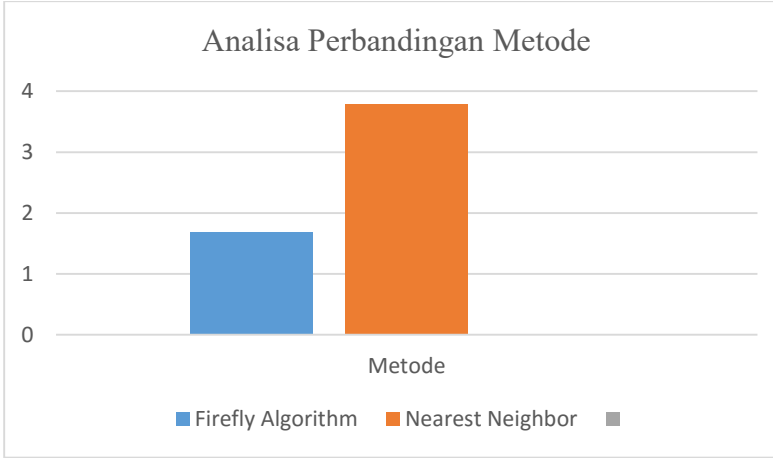
Dari hasil analisa, didapatkan suatu kesimpulan bahwa semakin banyak jumlah SPBU yang digunakan, maka nilai deviasi yang didapat akan jauh lebih buruk. Dalam perbandingan jumlah SPBU, digunakan kasus ekstrim yaitu dalam satu hari, pihak Pertamina mendistribusikan BBM ke semua SPBU yang ada di Surabaya. Hal ini difungsikan untuk melihat seberapa banyak kebutuhan kendaraan apabila semua SPBU yang ada di Surabaya membutuhkan BBM pada hari yang sama. Selain itu, dari hasil analisa didapatkan kesimpulan semakin banyak SPBU yang digunakan, maka kebutuhan akan kendaraan semakin meningkat. Dengan meningkatnya kendaraan, maka data jarak tempuh sebagai nilai masukan deviasi akan bertambah pula. Hasil analisa perbandingan jumlah SPBU dapat dilihat pada gambar 5.4.



Gambar 5.4 Analisa Perbandingan Jumlah SPBU

5.4.5 Analisa Menggunakan Perbandingan Metode

Dari hasil analisa didapatkan suatu kesimpulan bahwa *firefly algorithm* menghasilkan nilai deviasi yang jauh lebih baik dari *nearest neighbor.* Hal ini dikarenakan metode *nearest neighbor* menggunakan jarak terpendek pada proses penentuan rutenya. Sehingga pada rute-rute awal, jarak tempuh yang didapat memiliki *range* yang pendek dan mengakibatkan jarak tempuhnya menjadi jauh dari nilai 200 Km. Karakteristik ini berbanding terbalik dengan metode *firefly algorithm* yang notabenenya adalah metode metaheuristik yang menggunakan nilai random sebagai input datanya. Pada setiap iterasi, akan menghasilkan satu nilai terbaik. Dan dari banyaknya iterasi yang dipakai, akan diambil satu nilai terbaik lagi dari beberapa nilai terbaik. Hal inilah yang membuat nilai deviasi yang dihasilkan oleh *firefly algorithm* jauh lebih baik. Hasil analisa perbandingan kedua metode tersebut dapat dilihat pada gambar 5.5.



Gambar 5.5 Analisa Perbandingan Metode

Halaman ini sengaja dikosongkan

BAB VI
PENUTUP

Hasil dari keseluruhan proses pengerjaan Tugas Akhir ini dirangkum dan dapat dirumuskan suatu kesimpulan. Kesimpulan ini menerangkan hasil dari pengujian dan simulasi yang telah dilaksanakan.

Selama proses perancangan dan penelitian, terdapat banyak kendala yang dihadapi. Kendala tersebut telah penulis rangkum dan dirumuskan dalam bentuk saran untuk penyempuranaan dan penelitian lebih lanjut

6.1 Kesimpulan

1. Ada 2 tujuan utama dari proses *vehicle routing,* yaitu meminimumkan total jarak tempuh yang terpakai, yang mana adalah hasil dari proses penentuan rute, serta *balancing* atau penyamarataan beban kerja tiap kendaraan, yang mana sebagai hasil dari proses penggabungan rute.
2. *Firefly Algorithm* dapat digunakan sebagai suatu metode penyelesaian permasalahan *vehicle routing and dispatching* dengan penambahan kemampuan penyimpanan nilai terbaik di setiap iterasi.
3. Dalam penyelesaian permasalahan *vehicle routing and dispatching* dengan dataset 85 pelanggan, *firefly algorithm* mampu menghasilkan 24 rute dengan nilai deviasi jarak tempuh kendaraan yang terpakai sebesar 1.677 jam.
4. Dalam kasus ekstrem ketika jumlah SPBU yang dilayani adalah 95, *firefly algorithm* mampu menghasilkan 29 rute dengan nilai deviasi jarak tempuh kendaraan yang terpakai sebesar 1.846 jam.
5. Dari analisa poin (2) dan (3) didapatkan kecenderungan kebutuhan kendaraan meningkat seiring bertambahnya jumlah SPBU yang dilayani.
6. Dari hasil analisa parameter jumlah iterasi, jumlah populasi, maupun penentuan nilai gamma, didapatkan suatu kecenderungan nilai deviasi menurun seiring meningkatnya nilai parameter tersebut.
7. Berangkat dari analisa poin (6), Kecepatan pemrosesan data pada *firefly algorithm* berbanding terbalik dengan meningkatnya nilai parameter *firefly*. Semakin kecil nilai parameter yang dipakai,

maka kecepatan pemrosesan data akan jauh lebih cepat. Namun apabila digunakan nilai parameter yang jauh lebih besar, maka kecepatan pemrosesan data akan jauh lebih lambat, namun dapat menghasilkan suatu penyelesaian yang lebih baik.

8. *Firefly algorithm* mampu menghasilkan nilai deviasi jarak tempuh yang lebih baik dari metode *nearest neighbor*.
9. *Firefly algorithm* mampu menghasilkan nilai deviasi yang lebih baik dari *nearest neighbor* dikarenakan *firefly algorithm* sebagai salah satu varian metaheuristik, menggunakan proses *random* di setiap iterasi dalam proses pencarian nilai terbaik. Sedangkan *nearest neighbor* adalah varian dari metode heuristik yang berangkat dari karakteristik tiap-tiap algoritma dalam pencarian solusi terbaik.

6.2 Saran

Saran-saran yang dapat diusulkan oleh penulis perihal permasalahan *vehicle routing problem* menggunakan *firefly algorithm* adalah sebagai berikut:

1. Perlu adanya pemahaman lebih dalam terkait penggunaan metode metaheuristik pada kasus *vehicle routing problem*
2. Adanya pengembangan metode metaheuristik yang lain pada kasus *vehicle routing problem*, yang dapat menemukan solusi penyelesaian masalah dengan kecepatan pemrosesan data yang lebih baik dari *firefly algorithm*.
3. Kompleksitas permasalahan dapat dikembangkan dengan menambahkan penyimpanan data jarak tempuh yang telah dilalui oleh tiap kendaraan serta karakteristik kendaraan di tiap SPBU.

DAFTAR PUSTAKA

[1] Hillier, F. Lieberman, G. “Introduction to Operation Research”, McGraw-Hill, New York, 2001.

[2] Alkaff, A., Gamayanti, N. “Diktat Kuliah Penyelidikan Operasi”, Surabaya.

[3] Wu, N., R, Coppins. “Linear Programming and Extensions”, McGraw-Hill, New York, 1981.

[4] Carić, Tonči, Gold, Hrvoje. "Vehicle Routing Problem," *Faculty of Traffic and Transport Sciences*, Uniaversity of Zagreb, 2008.

[5] Toth, P., Vigo, D. “The Vehicle Routing Problem”, SIAM Monographs on Discrete Mathematics and Applications. SIAM, Philadelphia, 2002.

[6] Immanuel, Randi Mangatas. “Algoritma Particle Swarm Optimization Untuk Menyelesaikan Multi Depot Vehicle Routing Problem Dengan Variabel Travel Time”, Institut Teknologi Sepuluh Nopember, *Thesis*, 2015.

[7] A. Borodin and R. El-Yaniv. “Online Computation and Competitive Analysis”, Cambridge University Press, 1998.

[8] Ralphs, Ted, “Capacitated Vehicle Routing and Some Related Problems,” *Industrial and Systems Engineering Lehigh University,* 2005.

[9] Bird, John. “Engineering Mathematics”, Elsevier, Oxford, 2007.

[10] Al-Kharabsheh, Khalid. “Review on Sorting Algorithms: A Comparative Study”, International Journal of Computer Science and Security (IJCSS) Volume (7), 2013.

[11] Anggara Putra, Rian. “Efektifitas Metode Sequential Insertion Dan Metode Nearest Neighbour Dalam Penentuan Rute Kendaraan Pengangkut Sampah Di Kota Yogyakarta”, *Thesis*, Fakultas Matematika dan Ilmu Pengetahuan Alam, Universitas Negeri Yogyakarta, 2014.

[12] Sudirwan, J., Fadlilah, S.N. “Aplikasi Hybrid *Firefly* Algorithm Untuk Pemecahan Masalah Traveling Salesman: Studi Kasus Pada PT Anugerah Mandiri Success”, *ComTech Vol. 5 No. 2*, 2014.

[13] Zainal Abidin, A.F., Ansor, M.A. “A Preliminary Study on *Firefly* Algorithm Approach for Travelling Salesman Problem,” *Science & Engineering Technology National Conference*, 2013.

[14] Jati, G.K., Suyanto, S. “Evolutionary discrete *firefly* algorithm for traveling salesman problem”. ICAIS, 2011.

[15] Wibowo, R.P. “Optimasi Pengiriman BBM Ke SPBU Pada Kasus Multi Depot dan Multi Product Di Sales Area Bandung*”, Thesis*, Fakultas Teknik, Universitas Indonesia, 2011.

LAMPIRAN A

A.1 Wilayah SPBU Surabaya

No.	No SPBU	JARAK TEMPUH KE DEPO	ALAMAT
1	5460134	16.7	Jl. Wonorejo I Tandes
2	5460147	19.8	Jl. Semolowaru No. 276
3	5460198	13.8	Jl. Ploso Baru No. 183-185
4	51601124	17.7	Jl. Ketintang Madya No. 46
5	54601117	11.2	Jl. Raya Ngaglik No. 73-75
6	5160166	10.4	Jl. Dupak No. 15
7	5460102	17.6	Jl. Mulyosari 388 Kec. Sukolilo
8	5460119	7.9	Jl. Gresik No. 97
9	5460132	10.6	Jl. Semarang
10	5460143	13.8	Jl. Indrakila No. 1A
11	5460144	6.9	Jl. Stasiun Kota No. 62A
12	5460193	23.8	Jl. Veteran
13	5460230	26.7	Jl. Kertajaya
14	5460240	14.0	Jl. Gentengkali
15	5460253	21.5	Jl. Gunungsari No. 53
16	5460260	24.1	Jl. Medokan Ayu 20 – Ruugkut
17	5760111	3.1	Jl. Pelabuhan Tanjung Perak
18	53601125	11.2	Jl. Bubutan No. 72-76
19	54601107	21.2	Jl. Kebonsari Tengah No. 64-65
20	54601114	21.9	Jl. Balas Klumprik
21	54601121	9.7	Jl. Pahlawan Kec. Bubutan
22	5160165	26.4	Jl. Jemursari No. 113 – 123
23	5460103	12.1	Jl. Kapas Krampung No. 99

24	5460104	15.4	Jl. Kupang Jaya No. 2A
25	5460115	14.2	Jl. Kenjeran No. 661
26	5460117	10.3	Jl. Sidotopo Lor 49-51
27	5460121	13.9	Jl. Margomulyo Tandes
28	5460123	11.5	Jl. Arjuno
29	5460133	14.2	Jl. Balongsari – Tandes
30	5460135	13.7	Jl. Darmo Indah Timur 12 – 14
31	5460142	10.5	Jl. Kenjeran
32	5460146	19.5	Jl. Arief Rachman Hakim
33	5460167	14.2	Jl. Diponegoro No. 221
34	5460176	7.2	Jl. Iskandar Muda
35	5460181	17.2	Jl. Dinoyo No. 125
36	5460182	25.9	Jl. Manyar Sabrangan
37	5460183	9.7	Jl. Demak No. 152
38	5460184	12.9	Jl. Kusuma Bangsa No. 33
39	5460185	25.9	Jl. Pandugo No. 84
40	5460188	14.4	Jl. Sulawesi No. 8
41	5460191	7.9	Jl. Dharmahusada 24
42	5460192	14.1	Jl. Tegalsari No. 43 – 45
43	5460199	23.7	Jl. Raya Rungkut Mapan Blok FC/7-8
44	5460213	16.9	Jl. Mayjend Sungkono No. 47
45	5460254	30.6	Jl. Jemur Andayani No. 35
46	5460256	28.6	Jl. Raya Rungkut
47	5460268	19.8	Jl. Joyoboyo
48	5460269	18.1	Jl. Klampis Jaya No. 19
49	51601118	17.3	Jl. Pakuwon City
50	54601102	22.8	Jl. Nginden Semolo 80
51	54601106	10.8	Jl. Tidar No. 127 - 141

52	54601111	24.3	Jl. Raya Margorejo Indah 146A
53	54601115	21.4	Jl. Raya Bratang 59
54	54601119	20.9	Jl. Citra Raya Boulevard Blok TX-1
55	54601120	10.3	Jl. Kenjeran 99
56	5160177	14.8	Jl. Dr. Sutomo No. 87 – 89
57	5460101	8.8	Jl. Dupak Rukun 72A – B
58	5460105	2.3	Jl. Laksda M. Nasir No. 27
59	5460106	21.6	Jl. Raya Dharmahusada 114 – 116
60	5460108	15.2	Jl. Desa Tambak Langon Tandes
61	5460112	21.8	Jl. Tambak Osowilangun Sememi
62	5460116	11.2	Jl. Simopomahan 23 P
63	5460118	5.1	Jl. Perak Barat
64	5460173	21.5	Jl. Raya Pakal 104 Kec. Benowo
65	5460175	13.3	Jl. Tanjungsari
66	5460179	13.6	Jl. Embong Kemiri No. 48
67	5460180	24.4	Jl. A. Yani 204
68	5460189	7.6	Jl. Ikan Kakap No. 21 Krembangan
69	5460190	13.0	Jl. Sumatra No. 25 – 29
70	5460194	14.2	Jl. Raya Margomulyo No. 33
71	5460195	7.9	Jl. Sisingamangaraja
72	5460211	14.0	Jl. Desa Greges Kalianak
73	5460239	18.0	Jl. Manyar Kertoarjo
74	5460245	16.2	Jl. HR. Muhammad No. 113 Pradah Kali
75	5460248	20.3	Jl. Jagir Wonokromo
76	5460250	21.0	Jl. Raya Mastrip 237 Karangpilang
77	5460252	31.8	Jl. Rungkut Industri Raya No. 10
78	5460255	20.7	Jl. Raya Menganti 772
79	5460257	28.1	Jl. Raya Jemursari 194

80	5460259	19.3	Jl. Mayjend Sungkono 147
81	5460262	23.3	Jl. Raya Menganti 250
82	5460263	18.1	Jl. Mayjend HR. Muhammad 48 – 50
83	5460264	27.9	Jl. Raya Panjang Jiwo No. 54
84	5460272	17.1	Jl. Lontar 123
85	51601108	18.5	Jl. Mastrip Kebraon No. 116
86	54601100	17.3	Jl. Raya Ngagel 185 Kec. Wonokromo
87	54601101	16.1	Jl. Banjar Sugian No. 1
88	54601103	10.9	Jl. Dupak Rukun 108
89	54601104	13.1	Jl. Ambengan 16
90	54601105	21.7	Jl. Raya Menganti Lidah Wetan 139
91	54601109	21.2	Jl. Raya Nginden Kec. Gubeng
92	54601110	10.1	Jl. Raya Pecindilan 50 Kapasari
93	54601112	12.8	Jl. Raya Margomulyo No. 30
94	54601113	14.4	Jl. Raya Merr – Kalijudan
95	54601123	9.4	Jl Kalianak No. 152C Morokrembangan

LAMPIRAN B

B.1 *Random Distance*

```
distance = zeros(95,95);
 for i=1:95
     for j=i:95
         if(i==j)
         distance(i,j)=0;
         distance(j,i)=0;
         else
         distance(i,j)=rand()*26+1;
         distance(j,i)=distance(i,j);
         end
     end
 end
dlmwrite('distance.csv',distance);
```

B.2 *Random Demand*

```
spbuNUM = 95;
spbuOrder = zeros();
spbuDemand = zeros();
idx=1;
for i = 1:95
    spbuOrder(i) = idx;
    spbuDemand(i) = randi([2 4]);
    idx=idx+1;
    if(idx==5)
        idx=1;
    end
end
dlmwrite('spbuOrder.csv',spbuOrder);
dlmwrite('spbuDemand.csv',spbuDemand);
```

B.3 Fungsi Pengelompokkan SPBU

```
newPath = randperm(95);

shift1(1:29)=newPath(1:29);
shift2(1:28)=newPath(30:57);
shift3(1:28)=newPath(58:85);
```

```
shift4(1:10)=newPath(86:95);

finalSpbu=zeros();
finalSpbu(1:29)=shift1(1:29);
finalSpbu(30:57)=shift2(1:28);
finalSpbu(58:85)=shift3(1:28);
```

B.4 Fungsi Pembuatan Rute

```
function path = getPath(shift)
num = numel(shift);
path = randperm(num);
data1 = zeros();
demandSpbu=dlmread('spbuDemand.csv');
dist = dlmread('distance.csv');
distDepo = dlmread('distFromDepo.csv');

%Inisiasi data nomor spbu dan demand -----------
for i=1:num
    data1(1,i)=demandSpbu(shift(i));
    data1(2,i)=shift(i);
end

%Urutan diacak sesuai jumlah SPBU --------------
randomize=randperm(num);
data=zeros();
for i=1:num
    data(1,i)=data1(1,randomize(i));
    data(2,i)=data1(2,randomize(i));
end

%Pembuatan Rute --------------------------------
%Setiap Rute Berawal Dari Depo
initialDist = distDepo(data(2,1));
tempuh = initialDist;

pathEachCar=zeros();
distEachCar=zeros();
weightEachCar=zeros();
```

```
weight=data(1,1);
weightEachCar(1,1)=weight;
distEachCar(1,1)=distDepo(data(2,1));
idx=1;
idy=1;
pathEachCar(idx,idy)=data(2,1);
idy=2;
jarak=zeros();

for i=2:num
    add = data(1,i);
    addJarak = dist(data(2,i-1),data(2,i));
    jarak(i-1)=addJarak;

    if(weight+add<=5 && tempuh+addJarak<=190)
        pathEachCar(idx,idy)=data(2,i);
        distEachCar(idx,idy)=addJarak;
        weightEachCar(idx,idy)=add;
        weight=weight+add;
        tempuh=tempuh+addJarak;
        idy=idy+1;
    else
        distEachCar(idx,idy)=distDepo(data(2,i-
1));
        idy=1;
        idx=idx+1;
        pathEachCar(idx,idy)=data(2,i);
        weightEachCar(idx,idy)=add;
       distEachCar(idx,idy)=distDepo(data(2,i));
        tempuh = distDepo(data(2,i));
        weight = add;
        idy=2;
    end

    %Setiap Rute Berakhir di Depo --------------
    if(i==num)
       distEachCar(idx,idy)=distDepo(data(2,i));
    end
end
```

```
%Record Hasil Pembuatan Rute -------------------
path.pathEachCar = pathEachCar;
path.distEachCar = distEachCar;
path.Jarak = jarak;
path.weightEachCar = weightEachCar;
path.data = data;
path.carSize = carSize;
path.totDist = totalDist;

end
```

B.5 Fungsi Penggabungan Rute

```
function schedules =
initSchedule2(shift1,dist,weight)

%Tipekendaraan. Tipe 2=16. Tipe 3=24 dst -------
    type2=1;
    type3=18;
    type4=84;
    type5=140;

%Penggabungan Rute -----------------------------
if(type2>=18||type3>=84||type4>=140||type5>=154)
      z=a;
      flag=0;

%Dilihat dari tipe kendaraan -------------------
      elseif(weight(z)==2)
      if(shift1(type2)+dist(z)<=210);
         schedule(type2)=dist(z);
         used(type2)=z;
         z=z+1;
         type2=type2+1;
         else
         type2=type2+1;
      end
      elseif(weight(z)==3)
      if(shift1(type3)+dist(z)<=210)
         schedule(type3)=dist(z);
```

```
            used(type3)=z;
            z=z+1;
            type3=type3+1;
            else
            type3=type3+1;
        end
        elseif(weight(z)==4)
        if(shift1(type4)+dist(z)<=210)
            schedule(type4)=dist(z);
            used(type4)=z;
            z=z+1;
            type4=type4+1;
            else
            type4=type4+1;
        end
        elseif(weight(z)==5)
        if(shift1(type5)+dist(z)<=210)
            schedule(type5)=dist(z);
            used(type5)=z;
            z=z+1;
            type5=type5+1;
            else
            type5=type5+1;
        end
    end
end

%Tidak Ada Kendaraan yang Menyuplai ------------
        if(flag==0)
          for i=1:153
             schedule(i)=inf;
          end
        end

%Record Hasil Penggabungan Rute ----------------
        schedules.schedule =schedule;
        schedules.used = used;
end
```

B.6 Fungsi Deviasi

```
function stdevi = distStdev(load)
z=numel(load);
dev=0;

for i=1:z
    faktor = load(i)/25;
    simpangan = (faktor-8);
    simpanganSquare = simpangan*simpangan;
    dev = dev+simpanganSquare;
end
stdevi = dev/z;
stdevi = sqrt(stdevi);
end
```

B.7 Pemilihan Kendaraan

```
Clear;
Clc;

%Ambi Data Kendaraan --------------------------
    carAvail = dlmread('carCapacity.csv');
    bestNow=zeros;

%Ambil Data Tiap Shift -------------------------
    name1 = strcat('shift1.csv');
    shift1 = dlmread(name1);
    name2 = strcat('shift2.csv');
    shift2 = dlmread(name2);
    name3 = strcat('shift3.csv');
    shift3 = dlmread(name3);
    name4 = strcat('shift4.csv');
    shift4 = dlmread(name4);

%Ambil Data Demand Tiap SPBU -------------------
    order = strcat('spbuOrder.csv');
    spbuOrder = dlmread(order);
    newPath = spbuOrder;
```

```
%Ambil Data Jarak ------------------------------
    init = strcat('distFromDepo.csv');
    initDist = dlmread(init);

%Initiate Solusi Awal --------------------------
    best=zeros();
    initial1=struct;
    initial2=struct;
    initial3=struct;
    carSchedule=struct;

%initiate bestAll ------------------------------
    bestAll.stdevi=inf;
    bestpath=zeros;

%% FIREFLY ALGORITHM BEGIN ---------------------
localBest=zeros();

%For MaxGeneration ------------------------------
for s=1:20
    best.stdevi=inf;

%initiate first generation ----------------------
for i =1:100
      answer = getPath(shift1);
      initial1(i).pathEachCar=answer.pathEachCar
      ;
      initial1(i).distEachCar=answer.distEachCar
      ;
      initial1(i).weightEachCar=answer.weightEac
      hCar;
      initial1(i).carSize=answer.carSize;
      initial1(i).totDist=answer.totDist;

      answer = getPath(shift2);
      initial2(i).pathEachCar=answer.pathEachCar
      ;
      initial2(i).distEachCar=answer.distEachCar
      ;
```

```
      initial2(i).weightEachCar=answer.weightEac
      hCar;
      initial2(i).carSize=answer.carSize;
      initial2(i).totDist=answer.totDist;

      answer = getPath(shift3);
      initial3(i).pathEachCar=answer.pathEachCar
      ;
      initial3(i).distEachCar=answer.distEachCar
      ;
      initial3(i).weightEachCar=answer.weightEac
      hCar;
      initial3(i).carSize=answer.carSize;
      initial3(i).totDist=answer.totDist;
end
schedule = zeros(153,3);

%Rute Per Shift --------------------------------
%Untuk Shift 1 --------------------------------
sch1s =
initSchedule(initial1(i).totDist,initial1(i).car
Size);
sch1=sch1s.schedule;
used1=sch1s.used;
beban=zeros(153,1);
for r=1:153
   beban(r)=sch1(r);
end

%Untuk Shift 2 --------------------------------
sch2s =
initSchedule2(beban,initial2(i).totDist,initial2
(i).carSize);
sch2 = sch2s.schedule;
used2 = sch2s.used;
beban=zeros(153,1);
for r=1:153
   beban(r)=sch1(r)+sch2(r);
end
```

```
%Untuk Shift 3 ---------------------------------
sch3s =
initSchedule2(beban,initial3(i).totDist,initial3
(i).carSize);
sch3 = sch3s.schedule;
used3 = sch3s.used;
beban=zeros(153,1);
for r=1:153
   beban(r)=sch1(r)+sch2(r)+sch3(r);
end

cleanLoad = zeros();
idx=1;
for r=1:153
  if(beban(r)>0)
     cleanLoad(idx)=beban(r);
     idx=idx+1;
  end
end

stdevi = distStdev(cleanLoad);
carSchedule(i).cleanLoad=cleanLoad;
carSchedule(i).beban=beban;
carSchedule(i).sch1=sch1;
carSchedule(i).sch2=sch2;
carSchedule(i).sch3=sch3;
carSchedule(i).stdevi=stdevi;
carSchedule(i).shift1=initial1(i);
carSchedule(i).shift2=initial2(i);
carSchedule(i).shift3=initial3(i);
carSchedule(i).used1=used1;
carSchedule(i).used2=used2;
carSchedule(i).used3=used3;
if(carSchedule(i).stdevi<best.stdevi)
best.cleanLoad=carSchedule(i).cleanLoad;
best.beban=carSchedule(i).beban;
best.sch1=carSchedule(i).sch1;
best.sch2=carSchedule(i).sch2;
```

```
best.sch3=carSchedule(i).sch3;
best.stdevi=carSchedule(i).stdevi;
best.shift1=carSchedule(i).shift1;
best.shift2=carSchedule(i).shift2;
best.shift3=carSchedule(i).shift3;
best.used1=used1;
best.used2=used2;
best.used3=used3;
end

end

%Perbandingan Kecerahan Firefly ----------------
for x=1:100
    for y=1:100
        if(carSchedule(i).stdevi<best.stdevi)
               select = randi([1 6]);
            if(select==1)
            carSchedule(i).sch1=best.sch1;
            carSchedule(i).shift1=best.initial1(
            i);
            carSchedule(i).used1=best.used1;

            elseif(select==2)
            carSchedule(i).sch2=best.sch2;
            carSchedule(i).shift2=best.initial2(
            i);
            carSchedule(i).used2=best.used2;

            elseif(select==3)
            carSchedule(i).sch3=best.sch3;
            carSchedule(i).shift3=best.initial3(
            i);
            carSchedule(i).used3=best.used3;
            elseif(select==4)
            carSchedule(i).sch3=best.sch3;
            carSchedule(i).shift3=best.initial3(
            i);
            carSchedule(i).used3=best.used3;
```

```
carSchedule(i).sch2=best.sch2;

elseif(select==5)
carSchedule(i).sch2=best.sch2;
carSchedule(i).shift2=best.initial2(
i);
carSchedule(i).used2=best.used2;
carSchedule(i).sch1=best.sch1;

elseif(select==6)
carSchedule(i).sch3=best.sch3;
carSchedule(i).shift3=best.initial3(
i);
carSchedule(i).used3=best.used3;
carSchedule(i).sch1=best.sch1;

carSchedule(i).shift1=best.initial1(
i);
carSchedule(i).used1=best.used1;
end

%Pilih Kendaraan -------------------
for r=1:153
      beban(r)=carSchedule(i).sch1(r
      )+carSchedule(i).sch2(r)+carSc
      hedule(i).sch3(r);
end

%Beban Total Tiap Kendaraan Terpilih
cleanLoad = zeros();
idx=1;
    for r=1:153
        if(beban(r)>0)
         cleanLoad(idx)=beban(r);
         idx=idx+1;
        end
    end

%Record Hasil ----------------------
```

```
            carSchedule(i).beban=beban;
            carSchedule(i).cleanLoad=cleanLoad;
            carSchedule(i).stdevi=distStdev(clea
            nLoad);

            elseif(carSchedule(i).stdevi<best.st
            devi)
            best.cleanLoad=carSchedule(i).cleanL
            oad;
            best.beban=carSchedule(i).beban;
            best.sch1=carSchedule(i).sch1;
            best.sch2=carSchedule(i).sch2;
            best.sch3=carSchedule(i).sch3;
            best.stdevi=carSchedule(i).stdevi;
            best.shift1=carSchedule(i).shift1;
            best.shift2=carSchedule(i).shift2;
            best.shift3=carSchedule(i).shift3;
            best.used1=used1;
            best.used2=used2;
            best.used3=used3;
        end
    end
end

            %Apabila Firefly i > Firefly j -----
            if(bestAll.stdevi>best.stdevi)
            bestAll.cleanLoad=best.cleanLoad;
            bestAll.beban=best.beban;
            bestAll.sch1=best.sch1;
            bestAll.sch2=best.sch2;
            bestAll.sch3=best.sch3;
            bestAll.stdevi=best.stdevi;
            bestAll.shift1=best.shift1;
            bestAll.shift2=best.shift2;
            bestAll.shift3=best.shift3;
            bestAll.used1=best.used1;
            bestAll.used2=best.used2;
            bestAll.used3=best.used3;
            bestpath=newPath;
```

```
            end
localBest(s)=bestAll.stdevi;

%Buat Plot Random Untuk Iterasi Selanjutnya ----
newPath = randperm(85);
shift1=zeros();
shift2=zeros();
shift3=zeros();
shift4=zeros();

for i=1:29
    shift1(i)=finalSpbu(newPath(i));
end
for i=1:28
    shift2(i)=finalSpbu(newPath(29+i));
end
for i=1:28
    shift3(i)=finalSpbu(newPath(29+28+i));
end

%% Plot Hasil Ke Dalam Grafik -----------------
semilogy(localBest,'LineWidth',2);
xlabel('Iteration');
ylabel('Best Cost');
grid on;
```

Halaman ini sengaja dikosongkan

RIWAYAT PENULIS

Edo Rizaldi yang biasa dipanggil Edo oleh teman-temannya lahir di Surabaya, 29 September 1994. Edo merupakan anak pertama dari pasangan Budi Endro Cahyono dan Rita Utami Dewi. Lulus dari SD Muhammadiyah 6 Surabaya pada tahun 2006, kemudian melanjutkan studi ke jenjang lebih lanjut di SMPN 13 Surabaya dan lulus pada tahun 2009. Kemudian melanjutkan ke SMAN 5 Surabaya dan lulus pada tahun 2012. Setelah menempuh studi pada tingkat SMA, penulis melanjutkan ketingkat lebih lanjut, yaitu di Institut Teknologi Sepuluh Nopember (ITS) Surabaya jurusan Teknik Elektro pada tahun 2012. Pada saat menempuh pendidikan di ITS, penulis aktif di berbagai kegiatan. Pada tahun 2013, penulis diamanahi sebagai Ketua Pelaksana GERIGI ITS 2013. Di skala jurusan, penulis aktif di kegiatan kaderisasi Mahasiswa Elektro ITS selama 2 periode kepengurusan. Selain itu, penulis juga menjabat sebagai koordinator asisten Laboratorium B405 pada periode 2015-2016. Pada bulan Juni 2016 penulis mengikuti seminar dan ujian Tugas Akhir sebagai salah satu syarat untuk memperoleh gelar Sarjana Teknik Elektro dari Institut Teknologi Sepuluh Nopember Surabaya.

Penulis dapat dihubungi di edo_rizaldi@yahoo.com