TUGAS AKHIR – TE141599

OPTIMASI *VEHICLE ROUTING PROBLEM* DENGAN *PACKING CONSTRAINTS* MENGGUNAKAN METODE ALGORITMA GENETIKA

Moh. Yasya Bahrul Ulum
NRP 2213 100 025

Dosen Pembimbing
Nurlita Gamayanti, ST., MT.

DEPARTEMEN TEKNIK ELEKTRO
Fakultas Teknologi Elektro
Institut Teknologi Sepuluh Nopember
Surabaya 2017

--halaman ini sengaja dikosongkan--

FINAL PROJECT – **TE141599**

OPTIMIZATION APPROACH OF THE VEHICLE ROUTING PROBLEM WITH PACKING CONSTRAINTS USING GENETIC ALGORITHM METHOD

Moh. Yasya Bahrul Ulum
NRP 2213 100 025

Supervisor
Nurlita Gamayanti, ST., MT.

ELECTRICAL ENGINEERING DEPARTMENT
Faculty of Electrical Technology
Institut Teknologi Sepuluh Nopember
Surabaya 2017

--halaman ini sengaja dikosongkan--

PERNYATAAN KEASLIAN
TUGAS AKHIR

Dengan ini saya menyatakan bahwa isi sebagian maupun keseluruhan Tugas Akhir saya dengan judul "Optimasi *Vehicle Routing Problem* dengan *Packing Constraints* Menggunakan Metode Algoritma Genetika" adalah benar-benar hasil karya intelektual mandiri, diselesaikan tanpa menggunakan bahan-bahan yang tidak diizinkan dan bukan merupakan karya pihak lain yang saya akui sebagai karya saya pribadi.

Semua referensi yang dikutip maupun dirujuk telah ditulis secara lengkap pada daftar pustaka. Apabila ternyata pernyataan ini tidak benar, saya bersedia menerima sanksi sesuai peraturan yang berlaku.

Surabaya, 5 Juni 2016

Moh. Yasya Bahrul Ulum
Nrp 2213 100 025

--halaman ini sengaja dikosongkan--

OPTIMASI *VEHICLE ROUTING PROBLEM* DENGAN *PACKING CONSTRAINTS* MENGGUNAKAN METODE ALGORITMA GENETIKA

TUGAS AKHIR

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

Menyetujui

Dosen Pembimbing

Nurlita Gamayanti, ST., MT.
NIP : 19781201 200212 2 002

KEMENTERIAN RISET, TEKNOLOGI, DAN PENDIDIKAN TINGGI
INSTITUT TEKNOLOGI SEPULUH NOPEMBER
FAKULTAS TEKNOLOGI ELEKTRO
DEPARTEMEN TEKNIK ELEKTRO

SURABAYA
JULI, 2017

--halaman ini sengaja dikosongkan--

Optimasi *Vehicle Routing Problem* dengan *Packing Constraints* Menggunakan Metode Algoritma Genetika

Nama : Moh. Yasya Bahrul Ulum
Pembimbing : Nurlita Gamayanti, ST., MT.

ABSTRAK

Vehicle Routing Problem adalah suatu permasalahan dalam pengiriman barang dari depot ke beberapa outlet atau pelanggan menggunakan beberapa kendaraan yang memiliki kapasitas terbatas dengan tujuan meminimalkan biaya pengiriman. Adanya *packing constraints* dikarenakan pada umumnya kendaraan yang digunakan dalam pengiriman barang mempunyai kontainer berbentuk balok. Sehingga diperlukan cara-cara penyusunan agar kendaraan dapat memuat semua barang, tidak merusak barang dan memudahkan dalam mengeluarkan barang Dalam tugas akhir ini akan dikembangkan model dan algoritma untuk menyelesaikan *vehicle routing problem with packing contraints* dengan metode metaheuristik yaitu algoritma genetika dengan tujuan meminimalkan total jarak tempuh pengiriman. Selain itu gabungan algoritma genetika dan algoritma *bottom-left fill* digunakan untuk untuk melakukan proses *packing* agar didapat proses *packing* yang optimal. Algoritma genetika mengeluarkan hasil 0,08% lebih buruk dari *ant colony optimization* dan 2,93% lebih baik dari *tabu search.* Selain itu metode algoritma genetika juga berhasil diterapkan pada suatu perusahaan retail pada jaringan jalan kota Surabaya.

Kata Kunci : *Routing*, Optimasi, Algoritma Genetika, *Packing*, *Bottom-Left Fill*

--halaman ini sengaja dikosongkan--

Optimization Approach of the Vehicle Routing Problem with Packing Constraints Using Genetic Algorithm Method

Name : Moh. Yasya Bahrul Ulum
Supervisor : Nurlita Gamayanti, S.T., M.T.,

ABSTRACT

Vehicle Routing Problem is a problem in delivering item from depot to all customers using several vehicles whose have limited capacity with purpose minimizing transportation cost. The packing constraints exist because the vehicles usually used in delivering item have a rectangular-box shaped container. The items itself also commonly have a shape of rectangular-box. Therefore, it needs a way to pack or load so that containers could load all of the items, not damage the items and make it easy to unload the items. The aim of this final project is developing a model and method as approach solution of vehicle routing problem with packing constraints using genetic algorithm to minimize the transportation mileage. In addition, a hybrid genetic algorithm and bottom-left fill algorithm also take place to solve the packing proses. This algorithm deliver average solution 0.08% worse than ant colony optimization but has 2.93% better than tabu search.

***Keyword**: Routing, Optimization, Genetic Algortihm, Packing, Bottom-Left Fill*

--halaman ini sengaja dikosongkan--

KATA PENGANTAR

Puji dan syukur kehadirat Allah SWT yang telah memberikan petunjuk dan kesempatan sehingga penulis dapat menyelesaikan buku Tugas Akhir ini. Sholawat dan salam senantiasa tercurah ke junjungan Nabi Muhammad SAW.

Buku Tugas Akhir ini merupakan salah satu perjuangan dari penulis demi melengkapi syarat memperoleh gelar Sarjana Teknik di Departemen Teknik Elektro ITS. Buku ini berjudul "Optimasi *Vehicle Routing Problem* dengan *Packing Constraints* Menggunakan Metode Algoritma Genetika" dipersembahkan untuk kemajuan riset pada bidang pengembangan algoritma dan implementasinya. Khususnya bagi ITS dan Indonesia.

Dalam pengerjaannya penulis menemukan berbagai kendala namun berkat dukungan berbagai pihak dan alhamdulillah buku ini dapat terselesaikan tepat pada waktunya. Ucapan terima kasih penulis sampaikan kepada

1. Allah SWT yang telah memberikan petunjuk dan kesempatan untuk mengerjakan buku Tugas Akhir ini
2. Orang Tua yang selalu memberikan dukungan moral, doa dan finansial sehingga buku ini dapat terselesaikan
3. Dosen Pembimbing Ibu Nurlita Gamayanti yang dengan sabar selalu memberikan saya semangat dalam pengerjaan meskipun dalam akhir pengumpulan masih dalam keadaan tertinggal.
4. Teman-teman rumah belajar Arafat yang telah memberi semangat, mengajak bermain *online game* dan mau memberi hutang ketika tidak punya uang.
5. Teman-teman Laboratorium Sistem dan Sibernetik yang sering menemani belajar dan memberi masukan ketika berada di Laboratorium.

Penulis menyadari banyaknya kekurangan dan memohon maaf. Kritik dan saran selalu penulis nantikan agar dapat menjadi kemajuan dalam penelitian selanjutnya. Akhir kata penulis berharap Tugas Akhir ini dapat menajadi acuan untuk penelitian selanjutnya.

--halaman ini sengaja dikosongkan--

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

BAB 1
PENDAHULUAN

1.1 Latar Belakang

Transportasi adalah salah satu komponen yang penting dalam pengiriman barang atau manajemen logistik. Peningkatan efisiensi dilakukan karena dalam pengiriman memerlukan biaya yang berbanding dengan jarak ataupun waktu tempuh kendaraan. Selain itu peningkatan utilitas dari kendaraan diperlukan agar proses pengiriman menjadi lebih cepat. Dalam hal ini, perlu dicari jalur dan rute terbaik, yang dapat meminimalkan waktu dan biaya maupun meningkatkan utilitas masing-masing kendaraan. Permasalahan yang bertujuan membuat suatu rute yang terbaik pada sekelompok kendaraan pengiriman barang disebut *vehicle routing problem*(VRP).

Secara umum VRP dimodelkan pada teori *graph* yaitu mendesain *node* yang berbentuk depot ataupun pelanggan dan *arc* pada jalan penghubung antar *node.* Kemudian dibentuk aliran pada *node* depot ke sekumpulan *node* pelanggan (atau konsumen) yang tersebar, dengan biaya termurah. Hasil aliran tersebut harus dibuat sedemikian sehingga setiap *node* pelanggan dilayani sesuai dengan permintaan dan total barang yang dibawa tidak boleh melebihi kapasitas dari kendaraan.

Pada studi kasus diasumsikan kendaran berupa *box-truck* dengan *box* berupa balok yang berisi palet-palet untuk menata barang. Barang-barang yang dikirim harus disusun sedemikian sehingga dapat tertampung pada kendaraan, tidak merusak barang-barang dan memudahkan dalam pengeluaran barang (barang dari pelanggan yang dikunjungi lebih awal terletak lebih dekat dengan pintu). Batasan seperti ini disebut *packing contraints (*PC).

Metode algoritma genetika merupakan salah satu metode yang dapat membangkitkan permutasi-permutasi secara berkelanjutan dan menjadi lebih baik kemudian diambil hasil terbaik dengan tujuan mendapatkan hasil (*objective function)* paling optimal.

1.2 Permasalahan

Penjadawalan, pembuatan rute dan proses pengepakan barang bukan proses yang mudah untuk dikerjakan secara manual agar mendapat hasil yang optimal. Permasalahan yang dibuat adalah bagaimana

mengembangkan algoritma genetika dalam permasalahan pembuatan rute pengiriman beberapa barang memperhatikan beberapa aspek yaitu pelanggan yang memesan barang harus dipenuhi dengan barang-barang harus termuat dalam kendaraan.

1.3 Batasan Masalah

Dalam penelitian tugas akhir ini yang menjadi batasan permasalahan adalah

1. Barang dan kontainer kendaraan berbentuk balok.
2. Penataan barang ke kontainer harus tegak lurus.
3. Untuk implementasi memperhatikan 208 unit pelanggan, maksimal 30 kendaraan dan 19 jenis barang.
4. Untuk perbandingan hasil digunakan data sekunder [1]
5. *Transportaion cost* pada jaringan jalan berupa jarak tempuh perjalanan.
6. Tidak ada batasan *time-window* pada pelanggan maupun pada kendaraan.

1.4 Tujuan

Penelitian Tugas Akhir ini memiliki tujuan untuk mengembangkan model dan algoritma guna menghasilkan rute kendaraan-kendaraan yang akan dikirim untuk mengirim barang ke pelanggan-pelanggan beserta bentuk penataan barang ke dalam kontainer kendaraan. Hasil yang diharapkan adalah mendapatkan rute beserta penataan barang dimana meminimalkan total jarak tempuh semua kendaraan.

1.5 Metodologi

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

1. Studi Literatur & Pengumpulan Data
   Kegiatan pengumpulan dan pengkajian terhadap referensi terkait topik tugas akhir yang diusulkan. Data tersebut berupa segmentasi jalan, alamat pelanggan, dimensi kontainer kendaraan, biaya operasi kendaraan dan karateristik barang yang dikirim.
2. Identifikasi dan Permodelan VRP dengan PC
   Dari data yang telah dikumpulkan kemudian dilakukan identifikasi parameter-parameter yang penting untuk melakukan pemodelan jalanan dalam bentuk *adjacency-list* dengan parameter lama waktu dan biaya yang dikeluarkan. Kemudian identifikasi parameter berhubungan dengan melakukan *fitting* untuk proses *packing* seperti karateristik barang dan truk.
3. Penyelesaian dengan Metode Algoritma Genetika
   Melakukan penyelesaian dengan metode metaheuristik. Sehingga didapat nilai-nilai terbaik yang meminimalkan *objective function* berupa biaya yang dikeluarkan dalam proses pengiriman.
4. Perbandingan Hasil
   Setelah didapatkan hasil kemudian dibandingkan nilai optimal yang dihasilkan, kurva nilai optimal per generasi dan waktu yang dibutuhkan dalam mencapai nilai optimal.
5. Penarikan Kesimpulan dan Saran
   Penarikan kesimpulan mengacu pada data pengujian, analisis data, dan referensi terkait. Kesimpulan menunjukkan hasil kerja secara garis besar sesuai rumusan masalah yang telah dibuat. Selanjutnya, penarikan saran juga perlu dilakukan sebagai bentuk koreksi terhadap penelitian yang telah dilakukan dan pengembangan penelitian selanjutnya terkait topik serupa.
6. Penyusunan Buku Tugas Akhir
   Tahap ini merupakan tahap akhir dari serangkaian pelaksanaan tugas akhir. Penyusunan buku tugas akhir dilakukan sebagai bentuk laporan tertulis dari proses dan hasil kerja terkait topik yang diusulkan.

1.6 Sistematika Penulisan

Tahapan terakhir dari sebuah penelitian adalah penulisan laporan. Pada penulisan laporan atau Tugas Akhir ini disusun dalam 5 bab, dimana setiap bab berisi mengenai permasalahan dan penelitian. Bab tersebut ialah sebagai berikut,

BAB 1 PENDAHULUAN

Berisi mengenai latar belakang permasalahan, batasan masalah, tujuan penelitian, metodologi masalah, sistematika penulisan dan relevansi tugas akhir ini.

BAB 2 TEORI PENUNJANG

Berisi konsep dasar dan teori-teori yang menjadi rujukan penelitian meliputi teori *vehicle routing problem*, *bin packing problem* dan algoritma genetika.

BAB 3 PERANCANGAN SISTEM

Berisi rancangan sistem, model sistem beserta perancangan algoritma-algoritma penyelesaian yang mengacu teori pada Bab 2.

BAB 4 IMPLEMENTASI DAN ANALISIS

Berisi pengujian dan perbandingan hasil beserta analisis data mengenai rata-rata nilai fungsi tujuan dengan algoritma pembanding dan hasil implementasi menggunakan data riil perusahaan.

1.7 Relevansi

Tugas Akhir ini diharapkan dapat menjadi rekomendasi untuk perusahaan agar memudahkan proses pembuatan rute beserta penataan barang-barang pada kontainer kendaraan.

BAB 2
TEORI PENUNJANG

Pada Bab ini akan dibahas beberapa uraian teori yang digunakan sebagai penunjang dan dasar dalam pengerjaan Tugas Akhir.

2.1 Permasalahan Lintasan Terpendek [2]

Permasalahan lintasan terpendek dapat digambarkan sebagai upaya pencarian lintasan yang mempunyai biaya minimum pada suatu *graph* dari sebuah *node* ke *node* yang lain. Biaya lintasan adalah jumlah biaya semua *arc* yang membentuk lintasan tersebut.

Ada beberapa asumsi yang digunakan dalam perhitungan lintasan terpendek, yaitu

1. *Graph* memiliki arah (*directed graph)*
2. Pasti ada lintasan berarah dari setiap *node* ke *node* yang lain
3. Tidak ada siklus negatif, atau siklus dengan biaya negatif

Algoritma penyelesaian lintasan terpendek dari *node a* ke *node* b pada suatu jaringan adalah sebagai berikut,

1. Mulai inisiasi $S := \{a\}$, $d(i) := \infty$ untuk $i \in S, d(s) := 0, pred(s) := 0$
2. Ambil $i \in S'$ dengan $d(i) = \min_{j \in S'} d(j)$
3. $S := S \cup \{i\}$
4. $\forall\, i, j \in A(i)$ jika $d(j) > d(i) + c_{ij}$ maka $d(j) := d(i) + c_{ij}$ dan $pred(j) := i$
5. Kembali ke 2 sampai $S' = \emptyset$

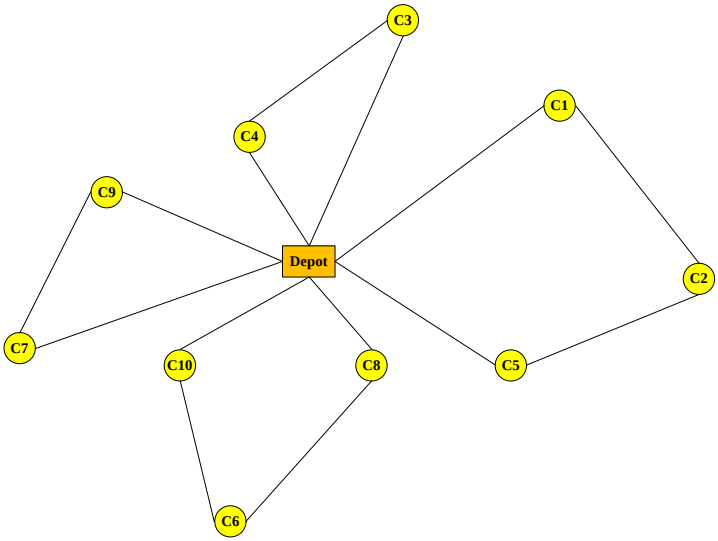
Bila perhitungan lintasan terpendek telah mencapai *node* yang dituju maka algoritma dapat dihentikan.

2.2 *Vehicle Routing Problem* (VRP) [2]

Vehicle routing problem adalah suatu permasalahan dalam pengiriman barang dari depot ke beberapa pelanggan atau pelanggan menggunakan beberapa kendaraan yang memiliki kapasitas terbatas dengan tujuan meminimalkan biaya dalam pengiriman atau memeratakan waktu pengiriman antar kendaraan. Secara umum VRP dapat dipisah menjadi dua pokok permasalahan yaitu *scheduling* dan *routing*

2.2.1 *Scheduling* [3]

Scheduling atau penjadwalan dalam VRP merupakan suatu proses perencaaan kemana saja suatu kendaraan akan beroperasi. *Scheduling* memberi perintah ke kendaraan bedasarkan tujuan seperti meminimalkan biaya dengan memperhatikan kendala-kendala yang ada seperti permintaan pelanggan atau kapasitas kendaraan.



Gambar 2.1 Contoh Gambar *Vehicle Routing Problem*

Scheduling merupakan lanjutan dari *travelling salesman problem.* Yaitu permasalahan mencari jarak terkecil untuk mengunjungi semua *node* tepat satu kali pada suatu *graph* dan kembali ke *node* mula dengan satu kendaraan. Kemudian muncul permasalahan *scheduling* yang memiliki lebih dari satu kendaraan seperti Gambar 2.1 diharuskan memenuhi permintaan masing-masing *node*. Setiap kendaraan dari *node* depot mengunjungi beberapa *node* dan kembali lagi ke *node* depot yang disebut *tour.* Adapun kendala tambahan yang umum untuk digunakan.

Kendala kapasitas, yaitu untuk setiap kendaraan memiliki batasan kapasitas sehingga jumlah atau berat barang yang dibawa suatu

kendaraan tidak lebih dari kapasitas yang ada. VRP dengan konstrain kapasitas biasa disebut *capacitated vehicle routing problem* (CRVP).

1. Kendala kunjungan minimal, yaitu pada setiap *tour* harus mengunjungi sejumlah *node* yang lebih dari atau sama dengan minimal *node* yang ditetapkan.
2. Kendala total waktu, yaitu total waktu dalam melakukan suatu *tour* tidak boleh melebihi batas yang ditentukan
3. *Time windows*, yaitu setiap *node* memiliki jam buka atau tutup sehingga hanya bias dikunjungi oleh kendaraan pada saat-saat tertentu saja
4. *Presedence relation* yaitu adanya kendala pada beberapa pasangan pelanggan i dan j sehingga pelanggan i harus dikunjungi terlebih dahulu daripada pelanggan j dalam satu *tour*.

Permasalahan ini dapat dinyatakan dalam bentuk *directed graph* yaitu pasangan himpunan *node* dan a*rc* (N, A). Dimana $N = \{0,1,2, \dots, n\}$ himpunan *node* merepresentasikan pelanggan, kota, tujuan atau tempat yang harus dipenuhi permintaanya dan A himpunan *arc* merepresentasikan jarak atau waktu tempuh dari dua buah *node*. Kemudian setiap kendaraannya membentuk jaringan yang berupa *cycle* bermula di depot dan kembali ke depot.

Permasalahan *scheduling* dalam VRP masuk dalam klasifikasi permasalahan dengan kompleksitas waktu komputasi NP (*nondeterministic polynomial time*). Sehingga dalam menghasilkan solusi paling optimal belum ditemukan algoritma dengan kompleksitas waktu polinom. Beberapa algoritma heuristik dan metaheuristik pun muncul meskipun belum tentu menghasilkan nilai optimal namun mendekati hasil optimal.

Pada VRP dengan kapasitas kendaraan terbatas memiliki model matematika sebagai berikut,

ψ_{ijk} : Variabel biner yang bernilai "1"apabila kendaraan k bergerak dari pelanggan i ke pelanggan j dan bernilai "0" apabila tidak.

c_{ij} : Jarak pelanggan i ke pelanggan j.

u_i : Variabel yang menyatakan apabila $u_i > u_j$ maka tidak ada kendaraan yang mengunjungi pelanggan i terlebih dahulu sebelum mengunjungi pelanggan j.

nm : Banyak *node* pelanggan

V : Himpunan semua pelanggan, $V = \{1,2,3, \dots, nm\}$.
V_0 : Himpunan semua pelanggan ditambah depot, $V_0 = \{0,1,2,3, \dots, nm\}$.
m_i : Total berat permintaan pada pelanggan i.
Q_k : Kapasitas berat kendaraan k.

Minimize $$z = \sum_i \sum_j \sum_k c_{ij} \psi_{ijk} \quad (2.1)$$

Subject to

$$\sum_{j \in V_0} \sum_k \psi_{ijk} = 1 \qquad \forall i \in V \quad (2.2)$$

$$\sum_{i \in V} \psi_{i0k} \leq 1 \qquad \forall k \quad (2.3)$$

$$\sum_{i \in V_0} \psi_{ihk} - \sum_{j \in V_0} \psi_{hjk} = 0 \qquad \forall h \in V_0, k \quad (2.4)$$

$$\sum_{j \in V} \sum_k \psi_{0jk} \geq 1 \quad (2.5)$$

$$u_i - u_j + nm \sum_k \psi_{ijk} = nm + 1 \qquad \forall i, j \in V \quad (2.6)$$

$$\sum_{i \in V} \sum_{j \in V_0} m_i \psi_{ijk} \leq Q_k \qquad \forall k \quad (2.7)$$

Fungsi tujuan yang ada pada Persamaan (2.1) bertujuan meminimalkan biaya pengiriman. Persamaan (2.2) menyatakan bahwa setiap *node* pelanggan dilayani oleh tepat satu kendaraan. Persamaan (2.3) menyatakan bahwa setiap kendaraan melakukan *tour* paling banyak sekali. Persamaan (2.4) menyatakan setiap kendaraan dari sebuah *node* hanya bisa berpindah ke satu *node* saja. Persamaan (2.5) menyatakan setidaknya ada sebuah kendaraan yang beroperasi. Persamaan (2.6) digunakan untuk menghilangkan *subtour* yang tidak melewati depot. Persamaan (2.7) menyatakan berat yang dibawa kendaraan untuk melayani *node* pelanggan tidak boleh melebihi kapasitasnya.

2.2.2 *Routing* [2]

Routing merupakan lanjutan dari *scheduling* namun memiliki perbedaan adanya *node* atau beberapa *node* yang tidak perlu dikunjungi atau memiliki permintaan. *Node* yang perlu dikunjungi dan dipenuhi permintaanya berbentuk depot atau pelanggan sedangkan *node* yang tidak

perlu dikunjungi berbentuk persimpangan jalan, bundaran atau putar balik. Permasalahan ini lebih mengarah ke implementasi riil.

Permasalahan *Routing* juga masuk dalam klasifikasi permasalahan dengan kompleksitas waktu komputasi NP. Namun permasalahan ini dapat dibagi menjadi dua permasalahan yaitu *scheduling* dan pencarian jarak terpendek antar pelanggan dan depot. Sehingga dilakukan perhitungan pencarian jarak terpendek masing-masing antar *node* depot dan *node* pelanggan terlebih dahulu kemudian permasalahan akan berubah menjadi *scheduling*.

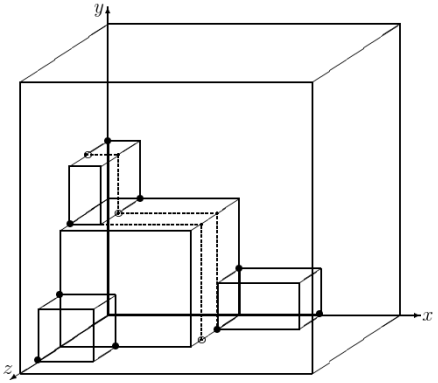
Pencarian jarak terpendek merupakan permasalahan dalam *graph theory* dimana mencari *path* yang menghubungkan dua *node* dengan jumlah bobot *arc* terkecil. Algoritma yang paling terkenal dalam menyelesaikan permasalahan ini adalah algoritma *dijkstra.*

2.3 *Bin Packing Problem* (BPP) [4]

Bin Packing Problem merupakan permasalahan menata beberapa objek ke dalam suatu tempat/kontainer yang terbatas dengan tujuan tertentu seperti semua objek dapat masuk ke dalam tempat tersebut sesuai dengan kendala yang ada ataupun memaksimalkan barang yang masuk sesuai perioritas barang dan kendala yang ada *(knapsack problem)*. Pada proses perhitungannya permasalahan ini masuk dalam klasifikasi permasalahan *NP-hard* [5]. Sehingga muncul berbagai algoritma heuristik seperti *first-fit* dimana memasukkan barang dengan urutan tertentu namun tidak menjamin hasil yang dicapai adalah optimum.

2.3.1 *Three-Dimensional Bin Packing Problem* (3D-BPP) [4]

Three-dimensional bin packing problem merupakan bagian dari BPP dimana objek berbentuk balok yang dapat dirotasi 90 derajat secara horizontal dengan kontainer berbentuk balok dan sisi-sisi semua objek sejajar dengan sisi kontainer. Permasalahan ini mirip dengan 2D-BPP. Beberapa algoritma yang digunakan untuk menyelesaikan 2D-BPP dapat digunakan untuk menyelesaikan 3D-BPP. Namun pada kondisi riilnya setiap barang terpengaruh oleh gravitasi sehingga perlu adanya luas penopang minimal, kapasitas penopang dan keseimbangan objek. Berikut contoh *three-dimensional bin packing problem* pada Gambar 2.2.



Gambar 2.2 *Three-Dimensional Bin Packing Problem* **[6]**

Permasalahan ini dapat didefinisikan sebagai permasalahan pengepakan dan penataan barang dari himpunan $\{1,2,3,\dots,n\}$ dimana setiap barang i memiliki lebar w_i, panjang l_i dan tinggi h_i ke suatu kontainer yang memiliki lebar W, panjang L dan tinggi H. Selanjutnya variabel-variabel yang dapat diputuskan dari proses pengepakan ini adalah koordinat-koordinat barang. Misalkan sumbu x, y, z berturut-turut sejajar dengan panjang, lebar dan tinggi kontainer dengan *origin* terletak pada sudut kiri bawah belakang dari kontainer. Kemudian variabel koordinat dari barang i adalah (x_i, y_i, z_i). Pada umumnya fungsi tujuan dari permasalahan ini hanyalah mencari solusi yang mungkin atau *feasible solution.* Namun adapula yang memiliki tujuan memaksimalkan volume okupasi dari barang-barang yang ada. Berikut adalah model *mixed integer liniear programming* untuk 3D-BPP,

$p_\alpha, q_\alpha, r_\alpha$: Berturut-turut panjang, lebar dan tinggi barang α.

L, W, H : Berturut-turut panjang, lebar dan tinggi kontainer kendaraan.

$x_\alpha, y_\alpha, z_\alpha$: Variabel yang merupakan koordinat RLB (*Rear, Left, Bottom*) dari barang α di dalam kontainer.

$a_{\alpha\beta}$: Variabel biner yang bernilai "1" apabila barang α di kiri barang β dan bernilai "0" apabila tidak.

$b_{\alpha\beta}$: Variabel biner yang bernilai"1" apabila barang α di kanan barang βdan bernilai "0" apabila tidak.

$c_{\alpha\beta}$: Variabel biner yang bernilai"1" apabila barang α di depan barang βdan bernilai "0" apabila tidak

$d_{\alpha\beta}$: Variabel biner yang bernilai"1" apabila barang α di belakang barang βdan bernilai "0" apabila tidak.

$e_{\alpha\beta}$: Variabel biner yang bernilai"1" apabila barang α di bawah barang βdan bernilai "0" apabila tidak.

$f_{\alpha\beta}$: Variabel biner yang bernilai"1" apabila barang α di atas barang βdan bernilai "0" apabila tidak.

$lx_\alpha, ly_\alpha, lz_\alpha$: Variabel biner yang bernilai"1" apabila panjang dari barang α sejajar dengan sumbu x, y, z berturut-turut dan bernilai "0" apabila tidak.

$wx_\alpha, wy_\alpha, wz_\alpha$: Variabel biner yang bernilai"1" apabila lebar dari barang α sejajar dengan sumbu x, y, z berturut-turut dan bernilai "0" apabila tidak.

$hx_\alpha, hy_\alpha, hz_\alpha$: Variabel biner yang bernilai"1" apabila tinggi dari barang α sejajar dengan sumbu x, y, z berturut-turut dan bernilai "0" apabila tidak.

M : Bilangan konstan yang sangat besar

$$x_\alpha + p_\alpha lx_\alpha + q_\alpha wx_\alpha + r_\alpha hx_\alpha \le x_\beta + (1 - a_{\alpha\beta})M \quad \forall \alpha, \beta \tag{2.8}$$

$$x_\beta + p_\beta lx_\beta + q_\beta wx_\beta + r_\beta hx_\beta \le x_\alpha + (1 - b_{\alpha\beta})M \quad \forall \alpha, \beta \tag{2.9}$$

$$y_\alpha + p_\alpha ly_\alpha + q_\alpha wy_\alpha + r_\alpha hy_\alpha \le y_\beta + (1 - c_{\alpha\beta})M \quad \forall \alpha, \beta \tag{2.10}$$

$$y_\beta + p_\beta ly_\beta + q_\beta wy_\beta + r_\beta hy_\beta \le y_\alpha + (1 - d_{\alpha\beta})M \quad \forall \alpha, \beta \tag{2.11}$$

$$z_\alpha + p_\alpha lz_\alpha + q_\alpha wz_\alpha + r_\alpha hz_\alpha \le z_\beta + (1 - e_{\alpha\beta})M \quad \forall \alpha, \beta \tag{2.12}$$

$$z_\beta + p_\beta lz_\beta + q_\beta wz_\beta + r_\beta hz_\beta \le z_\alpha + (1 - f_{\alpha\beta})M \quad \forall \alpha, \beta \tag{2.13}$$

$$a_{\alpha\beta} + b_{\alpha\beta} + c_{\alpha\beta} + d_{\alpha\beta} + e_{\alpha\beta} + f_{\alpha\beta} \ge 1 \qquad \forall \alpha, \beta \tag{2.14}$$

$$x_\alpha + p_\alpha lx_\alpha + q_\alpha wx_\alpha + r_\alpha hx_\alpha \le L \qquad \forall \alpha, \beta \tag{2.15}$$

$$y_\alpha + p_\alpha ly_\alpha + q_\alpha wy_\alpha + r_\alpha hy_\alpha \le W \qquad \forall \alpha, \beta \qquad (2.16)$$

$$z_\alpha + p_\alpha lz_\alpha + q_\alpha wz_\alpha + r_\alpha hz_\alpha \le H \qquad \forall a \qquad (2.17)$$

$$lx_\alpha + ly_\alpha + lz_\alpha = 1 \qquad \forall a \qquad (2.18)$$

$$wx_\alpha + wy_\alpha + wz_\alpha = 1 \qquad \forall a \qquad (2.19)$$

$$hx_\alpha + hy_\alpha + hz_\alpha = 1 \qquad \forall a \qquad (2.20)$$

$$lx_\alpha + wx_\alpha + hx_\alpha = 1 \qquad \forall a \qquad (2.21)$$

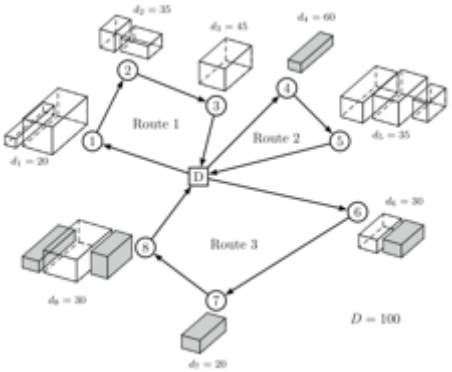
$$ly_\alpha + wy_\alpha + hy_\alpha = 1 \qquad \forall a \qquad (2.22)$$

$$lz_\alpha + wz_\alpha + hz_\alpha = 1 \qquad \forall a \qquad (2.23)$$

Pertidaksamaan (2.8), (2.9), (2.10), (2.11), (2.12), (2.13) dan (2.14) menjelaskan hubungan antara dua buah barang agar tidak ada barang yang *overlapping*. Pertidaksamaan (2.15),(2.16) dan (2.17) menyatakan agar barang muat didalam kontainer. Persamaan (2.18), (2.19), (2.20), (2.21), (2.22), (2.23) menyatakan posisi barang yang diletakkan.

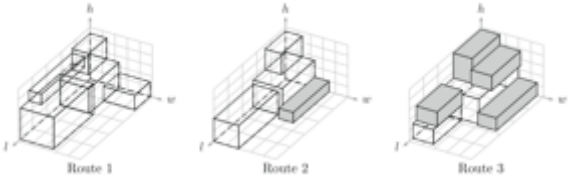
2.4 *Vehicle Routing Problem with Packing Constraint* (VRP-PC) [7]

Vehicle routing problem with packing constraint (VRP-PC) merupakan suatu permasalahan yang merupakan bagian dari VRP namun ditambah dengan kendala yang berupa BPP. Permasalahan ini pertama kali dikenalkan untuk menyelesaikan penentuan rute kendaraan untuk mengirimkan barang kepada pelanggan dengan meminimalkan total biaya tranportasi [8]. Barang yang dikirim berbentuk balok dimana barang memiliki berat dan ukuran berbeda-beda dan barang tersebut harus dapat dimuat dalam kendaraan. Secara teori permasalahan ini merupakan gabungan dari *capacitated vehicle routing problem* dengan *three dimensional bin packing problem.* Berikut Gambar 2.3 merupakan contoh sederhana dari VRP-PC.



Gambar 2.3 Contoh Gambar VRP-PC **[1]**

Berikut Gambar 2.4 merupakan salah satu penyelesaian dari permasalahan Gambar 2.3.



Gambar 2.4 Contoh Hasil Penyelesaian VRP-PC **[1]**

Secara matematis VRP-PC dapat deskripsikan sebagai berikut. Didefiniskan $G = (N, A)$ merupakan *graph* dengan $N = \{0,1,2,3, \dots, nm\}$ merupakan himpunan *node* dengan *node* 0 disebut depot dan *node* lainnya disebut pelanggan. Setiap pelanggan i memiliki permintaan barang sejumlah n_i dengan ukuran dan berat yang dapat berbeda-beda. Kemudian kendaraan k memiliki kontainer dengan ukuran panjang, lebar dan tinggi berturut-turut L_k, W_k dan H_k. Apabila kendaraan k harus memenuhi pelanggan-pelanggan dalam himpunan V_k dalam *tour*-nya maka barang-barang yang ada dalam himpunan V_k harus termuat dalam kendaraan k. Berikut model *mixed integer liniear programming* untuk VRP-PC.

2.5 Algoritma Genetika [9]

Algoritma genetika adalah teknik untuk mendapatkan nilai yang mendekati nilai terbaik dari solusi-solusi yang ada. Algoritma genetika

terinspirasi dari teori biologi revolusioner dimana generasi selanjutnya didapat dari rekombinasi DNA dari generasi sebelumnya. Dimana setiap generasi memiliki suatu populasi dan populasi adalah kumpulan individu.

Pada permasalahan inidividu didefinisikan sebagai sebuah solusi dari permasalahan. Komponen terpenting dari individu adalah kromosom. Kromosom ini berbetuk variabel solusi dari permasalahan. Nilai pembanding baik tidaknya sebuah solusi disebut *fitness*. Sehingga *fitness* merupakan fungsi dari variabel-variabel solusi.

Prosedur algoritma genetika pertama-tama adalah pemasukan data dan parameter. Kemudian pembangkitan generasi awal. Pembangkitan generasi awal ini dilakukan secara acak namun dapat pula dilakukan dengan bantuan algoritma heuristik agar mendapat hasil awal yang lebih baik. Kemudian dilakukan kawin silang. Proses kawin silang ini dilakukan agar mendapat individu baru dengan kromosom yang mirip dengan kromosom induk-induknya. Lalu dilakukan proses mutasi dengan tujuan semua titik yang ada tidak tertuju pada satu nilai optimal lokal. Kemudian dilakukan elitisme dengan tujuan mempertahankan solusi dengan *fitness* terbaik.

2.5.1 Algoritma Genetika untuk Optimasi Permutasi [9]

Permasalahan permutasi adalah suatu permasalahan penyusunan urutan dengan tujuan tertentu dan memperhatikan kendala-kendala yang ada. Salah satu contoh permasalahan permutasi adalah TSP. Pada permasalahan ini sangat sulit untuk menyatakan ke dalam beberapa variabel independen. Apabila dibawa ke dalam bentuk biner tentu variabel akan menjadi sangat banyak.

Pada algoritma genetika individu menyatakan solusi. Maka pada permasalahan permutasi ada dua teknik dalam merepresentasikan individu. Pertama, individu berbentuk *array* yaitu kromosom icr menyatakan urutan objek ke icr. Kedua, individu berbentuk *list* yaitu kromosom icr menyatakan objek pada urutan ke icr. Pada teknik pertama memiliki kelebihan akses objek yang cepat karena tidak perlu mencari keberadaan objek dalam urutan namun memiliki kelemahan apabila terdapat beberapa objek yang identik.

2.5.2 Algorita Genetika untuk VRP [10]

Pada permasalahan *vehicle routing problem* (VRP) secara naif kita dapat menyelesaikannya dengan menggunakan algoritma genetika dengan variabel dan kendala seperti model yang ada. Namun pada

kenyataannya variabel-variabel tersebut bersifat biner dengan banyak sekali kemungkinan dan memiliki kendala-kendala yang sangat mengikat variabel-variabel tersebut. Maka dalam pendekatan penyelesaiannya digunakan pengembangan dari algoritma genetika untuk optimasi permutasi.

Pada algoritma genetika individu menyatakan solusi. Dan pada kasus VRP terdapat *node* yang dikunjungi lebih dari sekali yaitu *node* depot. Sehingga dapat dilakukan representasi individu dalam bentuk *list*. Pada bentuk *list* ini hanya membentuk satu kromosom dengan banyak gen sebesar banyak *node* yang dikunjungi. Sebagai contoh representasi solusi untuk solusi VRP pada Gambar 2.1 ditampilkan pada Gambar 2.5.

Gambar 2.5 Contoh Representasi Solusi VRP dalam Bentuk *List*

Pada algoritma genetika untuk VRP ini memiliki urutan algoritma antara lain pembangkitan generasi awal, perhitungan *fitness*, kawin silang, mutasi, elitisme dan kembali ke perhitungan *fitness* hingga iterasi selesai pada kondisi yang ditentukan.

2.5.3 Pembangkitan Generasi

Pada pembangkitan generasi awal pada VRP dengan konstrain seperti CVRP dapat dilakukan algoritma sederhana seperti berikut.

1. Bangkitkan permutasi acak σ dari himpunan $\{1,2,3,\dots,n\}$ dan substitusi sebuah indeks $j := 1$
2. Untuk setiap i anggota σ dari kiri ke kanan masukkan i ke daftar kunjungan kendaraan j
3. Apabila i masuk ke j tidak dapat memenuhi konstrain maka sisipkan 0 sebelum i dan masukkan i ke $j + 1$
4. Kembali ke langkah 2

2.5.4 Kawin Silang

Kawin silang atau *crossover* merupakan proses pertukaran gen dalam algoritma genetika. Pada kawin silang algoritma genetika ini memeiliki kendala apabila digunakan algoritma sederhana seperti *uniform crossover* karena hasil kawin silangnya juga harus membentuk solusi dengan konstrain yang diberikan.

Ada dua macam kawin silang yaitu *sequence-based crossover* (SBX) dan *route-based crossover* (RBX) [10]. Dua kawin silang ini merupakan penyempurnaan *crossover* permutasi namun memperbaiki nilai keturunan sehingga dapat mewariskan sifat dari orang tua. Misal x_{p1} dan x_{p2} menyatakan kedua *parent* dan x_{o1} dan x_{o2} menyatakan kedua *offspring*

Berikut Algoritma untuk SBX,

1. $x'_{o1} = x_{p1}$
2. Ambil rute acak s_{r1} dari x_{p1} dan rute acak s_{r2} dari x_{p2}
3. Buatlah rute baru s_{new} dengan menambah *task* dari s_{r1} hingga suatu titik acak lalu tambahkan *task* dari sr_2 dari titik acak hingga akhir
4. Hilangkan duplikat pada s_{new} apabila ada
5. Hilangkan *task* pada x'_{o1} yang ada pada s_{new}
6. Hilangkan s_{r1} dan pindahkan semua *task* ke T_{temp}
7. Tambahkan s_{new} ke x'_{o1}
8. Buatlah x_{o1} dari x'_{o1} dengan menambahkan *task* yang ada pada T_{temp} dengan mengevaluasi fungsi objektif dengan metode *cheapest insertion* dan mempertahankan konstrain.
9. Buatlah x_{o2} dengan menukar *parent* dan melakukan pengulangan langkah 1 sampai 8.

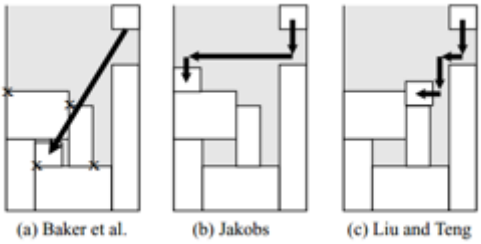
Berikut Algoritma untuk RBX,

1. $x'_{o1} = x_{p1}$
2. Ambil rute acak s_{r2} dari x_{p2}
3. Hilangkan semua *task* dari x'_{o1} yang ada pada s_{r2}
4. Hilangkan satu rute acak s_{r1} dari x'_{o1} dan pindahkan semua *task* ke T_{temp}
5. Tambahkan rute s_{r2} ke x'_{o1}
6. Buatlah x_{o1} dari x'_{o1} dengan menambahkan *task* yang ada pada T_{temp} dengan mengevaluasi fungsi objektif dengan metode *cheapest insertion* dan mempertahankan konstrain

Buatlah x_{o2} dengan menukar *parent* dan melakukan pengulangan langkah 1 sampai 6.

2.6 Algoritma *Bottom-Left Fill* (BLF) [11]

Bottom-left fill merupakan metode heuristik dalam menyelesaikan *bin packing problem*.. Karateristik utama algoritma ini adalah meletakkan satu per satu barang dengan urutan yang telah ditentukan dari sudut kiri bawah yang mungkin.



Gambar 2.6 Contoh Algoritma *Bottom Left* [11]

Ada beberapa jenis algoritma *bottom-left* seperti yang ditampilkan pada Gambar 2.6. Pada awalnya algoritma menggunakan titik paling kiri diantara posisi paling rendah yang mungkin [5]. Berikut tanda x pada Gambar 2.6 merupakan kemungkinan peletakan barang. Berikut urutan dari algoritma BLF,

1. Buat *list* kosong L dan masukkan koordinat $(0,0,0)$
2. Apabila barang i muat pada l (l_x, l_y, l_z) anggota *list* L letakkan barang i di l.
3. Kemudian hapus l di L.
4. Tambahkan $(l_x + length(i), l_y, l_z)$, $(l_x, l_y + width(i), l_z)$ dan $\left(l_x, l_y, l_z + height(i)\right)$ pada L.
5. Apabila tidak ada barang yang muat maka keluarkan hasil tidak memenuhi
6. Kembali ke langkah 2 hingga semua barang habis.

BAB 3
PERANCANGAN DAN IMPLEMENTASI SISTEM

Pada bab ini akan dijelaskan tentang pengembangan model *vehicle routing problem with packing constaints*, pemodelan jaringan jalan, dan tahapan penyelesaian beserta penjelasan algoritma yang digunakan.

3.1 Pengembangan Model *Vehicle Routing Problem with Packing Constaints*

Pada penjadwalan dari *vehicle routing problem with packing constaints* dibuat sebuah model yang dapat mendeskripsikan masalah dan mempermudah dalam penggunaan algoritma-algoritma. Konsep model ini bertujuan untuk menentukan *tour* setiap kendaraan yang melakukan pengiriman barang dari depot ke titik-titik pelanggan kemudian kembali lagi ke depot beserta penentuan cara menata barang-barang yang akan dikirimkan. Kriteria penentuan *tour* setiap kendaraan adalah meminimalkan jarak yang ditempuh. Kontrol *input* dan variabel tindakan yang dapat digunakan adalah penentuan rute setiap *tour* dan peletakan barang kedalam kendaraan. Alternatif tindakan yang dapat dilakukan adalah penambahan jumlah kendaraan.

Pengembangan model yang ada dari model yang tercantum pada studi literatur adalah penambahan kendala pada proses pengepakan barang yang berupa luas penopang minimal, keseimbangan barang, maksimal berat yang ditopang dan kendala *last-in first-out* (LIFO). Pemberian kendala ini bertujuan model dapat lebih memungkinkan untuk diimplementasikan. Pemberian kendala berbentuk luas penopang minimal dan keseimbangan bertujuan agar barang tidak jatuh saat ditumpuk. Kemudian kendala maksimum berat yang ditopang bertujuan agar barang yang menopang tidak penyok ataupun tidak pecah. Kendala *last-in first-out* (LIFO) bertujuan untuk memudahkan petugas dalam proses pengeluaran barang. Selain itu barang hanya boleh dirotasi secara horizontal dan asumsi semua truk adalah identik.

Salah satu model yang dapat dibentuk bedasar konsep model diatas adalah model matematis. Model matematis merupakan sebuah ekspresi kuantitatif yang menyatakan hubungan-hubungan antar komponen dari sistem yang relevan. Model matematis dibuat bedasarkan

konsep model yang telah dibuat. Berikut ini merupakan model matematis dari permasalahan *vehicle routing problem with packing constaints*,

ψ_{ijk} : Variabel biner yang bernilai "1"apabila kendaraan k bergerak dari pelanggan i ke pelanggan j dan bernilai "0" apabila tidak.

c_{ij} : Jarak pelanggan i ke pelanggan j.

u_i : Variabel yang menyatakan apabila $u_i > u_j$ maka tidak ada kendaraan yang mengunjungi pelanggan i terlebih dahulu sebelum mengunjungi pelanggan j.

nm : Jumlah pelanggan ditambah depot.

V : Himpunan semua pelanggan, $V = \{1,2,3, \dots, nm\}$.

V_0 : Himpunan semua pelanggan ditambah depot, $V_0 = \{0,1,2,3, \dots, nm\}$.

$s_{\alpha k}$: Variabel biner yang bernilai "1" apabila barang α diangkut kendaraan k dan bernilai "0" apabila tidak.

K_i : Himpunan barang-barang yang dikirim ke pelanggan i.

n_i : Banyak barang yang dikirim ke pelanggan i.

$p_\alpha , q_\alpha , r_\alpha$: Berturut-turut panjang, lebar dan tinggi barang α.

L, W, H : Berturut-turut panjang, lebar dan tinggi kontainer kendaraan.

$x_\alpha, y_\alpha, z_\alpha$: Variabel yang merupakan koordinat RLB (*Rear, Left, Bottom*) dari barang α di dalam kontainer.

$xt_\alpha, yt_\alpha, zt_\alpha$: Variabel yang merupakan koordinat FRT (*Front,Right,Top)* dari barang α di dalam kontainer.

$a_{\alpha\beta}$: Variabel biner yang bernilai "1" apabila barang α di kiri barang β dan bernilai "0" apabila tidak.

$b_{\alpha\beta}$: Variabel biner yang bernilai"1" apabila barang α di kanan barang βdan bernilai "0" apabila tidak.

$c_{\alpha\beta}$: Variabel biner yang bernilai"1" apabila barang α di depan barang βdan bernilai "0" apabila tidak

$d_{\alpha\beta}$: Variabel biner yang bernilai"1" apabila barang α di belakang barang βdan bernilai "0" apabila tidak.

$e_{\alpha\beta}$: Variabel biner yang bernilai"1" apabila barang α di bawah barang βdan bernilai "0" apabila tidak.

$f_{\alpha\beta}$: Variabel biner yang bernilai"1" apabila barang α di atas barang βdan bernilai "0" apabila tidak.

lx_α , ly_α : Variabel biner yang bernilai"1" apabila panjang dari barang α sejajar dengan sumbu x, y berturut-turut dan bernilai "0" apabila tidak.

wx_α , wy_α : Variabel biner yang bernilai"1" apabila lebar dari barang α sejajar dengan sumbu x, y berturut-turut dan bernilai "0" apabila tidak.

$g_{\alpha\beta}$: Variabel biner yang bernilai "1" apabila α ditopang β dan bernilai "0" apabila tidak

gh_α : Variabel biner yang bernilai "1" apabila α berada di dasar kontainer.

F : Perbandingan minimal luas penopang dengan luas permukaan yang ditopang.

m_α : Berat barang α.

tm_α : Total berat yang ditumpu oleh barang α

qm_α : Total berat maksimal yang dapat ditumpu oleh barang α

vx_α, vy_α : Koordinat horizontal berturut-turut sumbu x dan y dari pusat berat dari barang-barang yang ditumpu oleh barang α

ε_α : Jarak koordinat horizontal pusat berat dari barang yang ditumpu barang α tidak boleh berada kurang dari ε_α dari tepi sisi barang α

fr_α : Bernilai 1 apabila barang α bersifat *fragile* atau tidak dapat diberi tumpukan

Q : Kapasitas berat kendaraan .

M : Bilangan yang sangat besar.

Minimize $$z = \sum_i \sum_j \sum_k c_{ij} \psi_{ijk} \qquad (3.1)$$

Subject to

$$\sum_{j \in V_0} \sum_k \psi_{ijk} = 1 \qquad \forall i \in V \qquad (3.2)$$

$$\sum_{i \in V} \psi_{i0k} \leq 1 \qquad \forall k \qquad (3.3)$$

$$\sum_{i \in V_0} \psi_{ihk} - \sum_{i \in V_0} \psi_{hjk} = 0 \qquad \forall h \in V_0, k \qquad (3.4)$$

$$\sum_{j \in V} \sum_{k} \psi_{0jk} \geq 1 \qquad (3.5)$$

$$u_i - u_j + nm \sum_{k} \psi_{ijk} = nm + 1 \qquad \forall i, j \in V \qquad (3.6)$$

$$\sum_{i \in V} \sum_{j \in V_0} \sum_{\alpha \in K_i} m_\alpha \psi_{ijk} \leq Q \qquad \forall k \qquad (3.7)$$

$$xt_\alpha = x_\alpha + p_\alpha lx_\alpha + q_\alpha wx_\alpha \qquad \forall \alpha \qquad (3.8)$$

$$yt_\alpha = y_\alpha + p_\alpha ly_\alpha + q_\alpha wy_\alpha \qquad \forall \alpha \qquad (3.9)$$

$$zt_\alpha = z_\alpha + r_\alpha \qquad (3.10)$$

$$xt_\alpha \leq x_\beta + (1 - a_{\alpha\beta})M$$
$$\forall k, \alpha, \beta, \alpha \neq \beta, s_{ak} = s_{\beta k} = 1 \qquad (3.11)$$

$$xt_\beta \leq x_\alpha + (1 - b_{\alpha\beta})M$$
$$\forall k, \alpha, \beta, \alpha \neq \beta, s_{ak} = s_{\beta k} = 1 \qquad (3.12)$$

$$yt_\alpha \leq y_\beta + (1 - c_{\alpha\beta})M$$
$$\forall k, \alpha, \beta, \alpha \neq \beta, s_{ak} = s_{\beta k} = 1 \qquad (3.13)$$

$$yt_\beta \leq y_\alpha + (1 - d_{\alpha\beta})M$$
$$\forall k, \alpha, \beta, \alpha \neq \beta, s_{ak} = s_{\beta k} = 1 \qquad (3.14)$$

$$zt_\alpha \leq z_\beta + (1 - e_{\alpha\beta})M$$
$$\forall k, \alpha, \beta, \alpha \neq \beta, s_{ak} = s_{\beta k} = 1 \qquad (3.15)$$

$$zt_\beta \leq z_\alpha + (1 - f_{\alpha\beta})M$$
$$\forall k, \alpha, \beta, \alpha \neq \beta, s_{ak} = s_{\beta k} = 1 \qquad (3.16)$$

$$a_{\alpha\beta} + b_{\alpha\beta} + c_{\alpha\beta} + d_{\alpha\beta} + e_{\alpha\beta} + f_{\alpha\beta} \geq 1$$
$$\forall k, \alpha, \beta, \alpha \neq \beta, s_{ak} = s_{\beta k} = 1 \qquad (3.17)$$

$$xt_\alpha \leq L + (1 - s_{ak})M \qquad \forall a, k \qquad (3.18)$$

$$yt_\alpha \leq W + (1 - s_{ak})M \qquad \forall a, k \qquad (3.19)$$

$$zt_\alpha \leq H + (1 - s_{ak})M \qquad \forall a, k \qquad (3.20)$$

$$\sum_{a\in K_i} s_{ak} - n_i \sum_{j\in V} \psi_{ijk} = 0 \qquad \forall i,k \qquad (3.21)$$

$$lx_\alpha + ly_\alpha = 1 \qquad \forall a \qquad (3.22)$$

$$wx_\alpha + wy_\alpha = 1 \qquad \forall a \qquad (3.23)$$

$$lx_\alpha + wx_\alpha = 1 \qquad \forall a \qquad (3.24)$$

$$ly_\alpha + wy_\alpha = 1 \qquad \forall a \qquad (3.25)$$

$$\min\left(xt_\alpha, xt_\beta\right) - \max\left(x_\alpha, x_\beta\right) > \left(g_{\alpha\beta} - 1\right)M$$
$$\forall k, \alpha, \beta, \alpha \neq \beta, s_{ak} = s_{\beta k} = 1 \qquad (3.26)$$

$$\min\left(yt_\alpha, yt_\beta\right) - \max\left(y_\alpha, y_\beta\right) > \left(g_{\alpha\beta} - 1\right)M$$
$$\forall k, \alpha, \beta, \alpha \neq \beta, s_{ak} = s_{\beta k} = 1 \qquad (3.27)$$

$$\max\left(\left|z_\alpha - z_\beta - r_\beta\right|, 0\right) \leq \left(1 - g_{\alpha\beta}\right)M$$
$$\forall k, \alpha, \beta, \alpha \neq \beta, s_{ak} = s_{\beta k} = 1 \qquad (3.28)$$

$$\max\left(z_\alpha, 0\right) \leq \left(1 - gh_\alpha\right)M \qquad \forall \alpha \qquad (3.29)$$

$$\sum_\beta g_{\alpha\beta} + gh_\alpha \geq 1 \qquad \forall \alpha \qquad (3.30)$$

$$\sum_\beta g_{\alpha\beta}\left(\min\left(xt_\alpha, xt_\beta\right) - \max\left(x_\alpha, x_\beta\right)\right)$$
$$\times\left(\min\left(yt_\alpha, yt_\beta\right) - \max\left(y_\alpha, y_\beta\right)\right) \qquad \forall \alpha \qquad (3.31)$$
$$\geq F\left(1 - gh_\alpha\right)p_\alpha q_\alpha$$

$$tm_\beta = \sum_\alpha \left(g_{\alpha\beta} tm_\alpha + m_\alpha\right) \qquad \forall \beta \qquad (3.32)$$

$$tm_\beta \leq qm_\beta \qquad \forall \beta \qquad (3.33)$$

$$vx_\beta = \begin{cases} \dfrac{\sum_\alpha \left(g_{\alpha\beta} vx_\alpha tm_\alpha + \left(x_\alpha + \dfrac{q_\alpha}{2}\right)m_\alpha\right)}{\sum_\alpha \left(g_{\alpha\beta} tm_\alpha + m_\alpha\right)}, \sum_\alpha \left(g_{\alpha\beta} tm_\alpha + m_\alpha\right) > 0 \\ x_\beta + \dfrac{q_\beta}{2}, otherwise \end{cases}$$
$$\forall \beta \qquad (3.34)$$

$$vy_\beta = \begin{cases} \dfrac{\sum_\alpha (g_{\alpha\beta} vy_\alpha tm_\alpha + (y_\alpha + \frac{q_\alpha}{2}) m_\alpha)}{\sum_\alpha (g_{\alpha\beta} tm_\alpha + m_\alpha)}, \sum_\alpha (g_{\alpha\beta} tm_\alpha + m_\alpha) > 0 \\ y_\beta + \dfrac{q_\beta}{2}, otherwise \end{cases}$$

$$\forall \beta \tag{3.35}$$

$$vx_\beta \le xt_\beta - \varepsilon_\beta \qquad \forall \beta \tag{3.36}$$

$$vx_\beta \ge x_\beta + \varepsilon_\beta \qquad \forall \beta \tag{3.37}$$

$$vy_\beta \le yt_\beta - \varepsilon_\beta \qquad \forall \beta \tag{3.38}$$

$$vy_\beta \ge y_\beta + \varepsilon_\beta \qquad \forall \beta \tag{3.39}$$

$$\max\left(\min\left(yt_\alpha, yt_\beta\right) - \max\left(y_\alpha, y_\beta\right), 0\right)$$
$$\times \max\left(\min\left(zt_\alpha, zt_\beta\right) - \max\left(z_\alpha, z_\beta\right), 0\right) \times \left(x_\alpha - x_\beta\right) \ge 0$$
$$\forall i, j, k, \alpha \in K_i, \beta \in K_j \text{ dengan } u_i \le u_j \text{ dan } s_{\alpha k} = s_{\beta k} = 1 \tag{3.40}$$

$$\max\left(\min\left(xt_\alpha, xt_\beta\right) - \max\left(x_\alpha, x_\beta\right), 0\right)$$
$$\times \max\left(\min\left(yt_\alpha, yt_\beta\right) - \max\left(y_\alpha, y_\beta\right), 0\right)$$
$$\times \left(z_\alpha - z_\beta\right) \ge 0$$
$$\forall i, j, k, \alpha \in K_i, \beta \in K_j \text{ dengan } u_i \le u_j \text{ dan } s_{\alpha k} = s_{\beta k} = 1 \tag{3.41}$$

$$\sum_\alpha g_{\alpha\beta} \le (1 - fr_\beta) M \qquad \forall \beta \tag{3.42}$$

Model yang dibentuk memiliki fungsi tujuan meminimalkan jarak tempuh yang dinyatakan pada Persamaan (3.1). Persamaan (3.2) menyatakan bahwa setiap *node* pelanggan dilayani oleh tepat satu kendaraan. Pertidaksamaan (3.3) menyatakan bahwa setiap kendaraan melakukan *tour* paling banyak sekali. Persamaan (3.4) menyatakan setiap kendaraan dari sebuah *node* hanya bisa berpindah ke satu *node* saja. Pertidaksamaan (3.5) menyatakan setidaknya ada sebuah kendaraan yang beroperasi. Persamaan (3.6) digunakan untuk menghilangkan *subtour* yang tidak melewati depot. Pertidakksamaan (3.7) menyatakan berat yang dibawa kendaraan untuk melayani *node* pelanggan tidak boleh melebihi kapasitasnya. Persamaan (3.8) ,(3.9) ,(3.10) memberikan hubungan

koordinat RLB dengan koordinat FRT setiap barang. Pernyataan (3.11), (3.12), (3.13), (3.14), (3.15), (3.16) dan (3.17) diberikan agar tidak ada barang-barang yang terletak pada satu kendaraan yang *overlapping*. Pertidaksamaan (3.18), (3.19) dan (3.20) diberikan agar barang-barang terletak didalam kontainer. Persamaan (3.21) menyatakan banyak barang yang dimasukkan kontainer kendaraan sama dengan banyak barang yang diminta oleh setiap pelanggan. Persamaan (3.22), (3.23), (3.24) dan (3.25) menyatakan posisi barang dalam hal ini posisi dibatasi untuk horizontal saja. Pernyataan (3.26), (3.27) dan (3.28) menyatakan posisi barang yang menopang barang lainnya. Pernyataan (3.29) menyatakan bahwa barang ditopang oleh kontainer kendaraan. Pernyataan (3.30) memberikan kepastian bahwa barang pasti ditopang oleh barang yang lain atau ditopang langsung oleh kontainer. Pernyataan (3.31) menyatakan pemberian kendala luas penopang minimal. Pernyataan (3.32) memberikan nilai pada variabel total berat yang ditumpu setiap barang. Pernyataan (3.33) memberikan kendala berat maksimal yang dapat ditumpu barang. Persamaan (3.34) dan (3.35) memberikan nilai untuk koordinat horizontal dari pusat masa beban yang ditopang untuk setiap barang. Pertidaksamaan (3.36), (3.37), (3.38), (3.39) memberikan kendala keseimbangan barang. Pertidaksamaan (3.40) dan (3.41) memberikan kendala LIFO. Pertidaksamaan (3.42) memberikan kendala *fragility*.

3.2 Pemodelan Jaringan pada Jaringan Jalan kota Surabaya

Jalan yang dilalui kendaraan dimodelkan dalam bentuk *graph*. Dimana setiap persimpangan, titik tujuan direpresentasikan sebagai *node* dan ruas-ruas jalan yang menghubungkan antar *node* menjadi *arc*. Pada kota-kota besar jalanan sangat diatur sehingga sering dijumpainya jalan searah dan jalan kembar. Dalam kasus ini menggunakan *directed graph* dimana setiap *arc* memiliki arah dari satu *node* ke *node* yang lain.

3.2.1 Pengambilan Data

Data yang diambil berupa *node* dan *arc* dari *database* aplikasi Microsoft Maps. *Node* disimpan dalam koordinat lintang dan bujur bumi. Pada implementasinya akan digunakan peta kota Surabaya dimana memiliki koordinat bujur 112,601° BT- 112,822° BT dan memiliki koordinat lintang -7,413° LS - -7,202° LS. Sedangkan *arc* diberi bobot jarak terpendek permukaan bumi antar dua buah *node*. Dalam

memberikan bobot ini karena Surabaya dekat dengan khatulistiwa maka dapat digunakan pendekatan

$$z = \sqrt{\left(110.574 \times (\psi_1 - \psi_2)\right)^2 + \left(111.32 \times (\theta_1 cos\psi_1 - \theta_2 cos\psi_2)\right)^2}$$

Dengan z menyatakan jarak antar *node* dengan satuan kilometer.

3.2.2 Model *Graph*

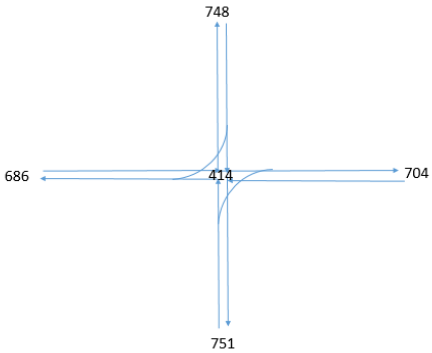
Setelah jalan dimodelkan dalam bentuk *graph* maka akan diselesaikan dengan algoritma pencari lintasan terpendek. Algoritma yang umum digunakan adalah algoritma *dijkstra*. Pada jalanan di kota-kota besar sering dijumpai adanya aturan larangan-larangan berbelok terutama pada jalan satu arah dan beberapa belok kanan.

Untuk mengatasi permasalah tersebut dilakukan perubahan dalam model *graph*. Perubahan tersebut berupa ruas jalan kita nyatakan sebagai *node* dan persimpangan-persimpangan kita nyatakan sebagai *arc* yang menghubungkan antara ruas jalan.

Tabel 3.1 Contoh Pemodelan Jaringan Jalan

Di	Dari	ke
414	704	686
414	704	751
414	686	748
414	686	704
414	751	686
414	751	704
414	751	748
414	748	686
414	748	751
414	748	704

Perhatikan pada Tabel 3.1 dapat digambarkan pada Gambar 3.1,



Gambar 3.1 Contoh Jaringan Jalan

Dari Gambar 3.1 berupa perempatan namun karena adanya larangan belok kanan dari 704 ke 748 dan 686 ke 751 maka pada Tabel 3.1 tidak ditulis *arc* yang menghubungkan ruas jalan 704 ke 414 ke 748 dan 686 ke 414 ke 751.

3.3 Pemodelan Jaringan Data Sekunder

Pada data jalan sekunder [1] berbentuk koordinat-koordinat *node* pelanggan dan depot. Jarak jalan penghubung antar *node* sebesar *euclidian distance* antar kedua *node* tersebut.

$$z = \sqrt{(x_1 - x_2)^2 + (y_2 - y_2)^2}$$

3.4 Tahapan Penyelesaian Masalah

Pada bagian ini akan dijelaskan tahapan penyelesaian masalah. Ada 3 tahap penyelesaian dalam kasus ini. Pertama kali akan dilakukan penggunaan algoritma *dijkstra* untuk mendapatkan jarak antar pelanggan-pelanggan dan depot. Kemudian pada tahap kedua menggunakan algoritma genetika untuk menghasilkan rute. Tahap ketiga adalah penggunaan algoritma genetika untuk melakukan penyusunan barang dengan algoritma gabungan algoritma genetika dengan algoritma BLF (GA-BLF).

3.4.1 Penggunaan Algoritma *Dijkstra*

Algoritma dijkstra merupakan algoritma pencarian lintasan terpendek. Penggunaan algortima dijkstra pada permasalahan ini adalah mendapatkan lintasan terpendek antar *node* pelanggan dan *node* depot. Algoritma ini memerlukan *input* data jalan yang telah dimodelkan dalam bentuk *graph* seperti pada Gambar 3.1. Kemudian hasil dari pencarian lintasan terpendek akan digunakan pada tahapan selanjutnya. Berikut tahapan algoritma *dijkstra*

Buatlah matriks jarak D untuk menyimpan jarak antar *node* depot dan pelanggan ke *node* yang lain dan beri nilai tak terhingga

Untuk setiap *node* i simpan semua *node* yang terhubung *arc* dengan i dalam bentuk *list* A_i secara terurut dari yang terpendek hingga terpanjang,

1. Untuk setiap *node* depot dan pelanggan i
2. Buatlah *list* kosong dengan aturan *first-in first-out* L dan masukkan i ke dalam L
3. Buatlah array *flag* dengan ukuran sama dengan banyak *node* total dan beri nilai mula 0 kecuali i beri nilai 1
4. Selama *list* tidak kosong ambil *node* paling awal j
5. Untuk setiap *node* k yang ada di A_j
6. $D(i,k) := \min\big(D(i,j) + length(j), D(i,k)\big)$
7. Apabila k belum diberi *flag* masukkan k ke dalam L dan beri *flag* pada k
8. Kembali ke 7
9. Kembali ke 6
10. Kembali ke 3.

3.4.2 Algoritma Genetika untuk *Vehicle Routing Problem*

Pada tahap ini dilakukan pengembangan algoritma genetika untuk menyelesaikan *vehicle routing problem with packing constraints*. Pengembangan ini berasal dari pengembangan algoritma genetika yang disesuaikan dengan permasalahan VRP kemudian dilanjutkan untuk menyelesaikan *packing constraints* pada tahap selanjutnya.

3.4.2.1 Pengumpulan Data

Data yang diperlukan antara lain jarak antar pelanggan dan depot yang telah didapat dari algoritma *dijkstra*, banyak kendaraan, permintaan

barang-barang setiap pelanggan beserta dimensi barang dan data dimensi kontainer kendaraan.

.

3.4.2.3 Parameter Algoritma

Dalam algoritma genetika banyak individu dan lama generasi merupakan parameter yang dapat dikontrol dan diperlukan untuk mendapat kualitas algoritma yang baik. Selain parameter tersebut adapula parameter yang tidak dapat dikontrol yaitu lebar kromosom yang ada. Misal kita definisikan parameter tersebut,

n : Banyak *node* pelanggan dengan indeks i, dan $i = 0$ menyatakan *node* depot

m : Banyak kendaraan dengan indeks k

ncr : Banyak kromosom dalam setiap individu dengan indeks icr

$npop$: Banyak individu dalam populasi dengan indeks $ipop$

$ngen$: Banyak generasi yang dibangkitkan dengan indeks $igen$.

3.4.2.3 Struktur Kromosom pada Individu

Setiap individu dalam algoritma ini merupakan salah satu solusi dari permasalahan yang ada. Untuk kasus VRP struktur kromosom yang ada pada setiap individu berupa barisan yang disimpan dalam bentuk *list.* Karena terdapat m kendaraan yang bergerak ke n buah *node* kita dapat nyatakan kromosom dalam *array* berukuran $m + n$ dimana berupa bilangan dari himpunan $\{0,1,2, \dots, n\}$ dimana memiliki sifat berikut

1. Memiliki n kromosom dari himpunan $\{1,2, \dots, n\}$ dengan setiap nilai muncul tepat satu kali
2. Memiliki m kromosom bernilai 0, sehingga
3. $ncr = n + m$
4. Urutan bilangan menyatakan urutan *tour* yang dilakukan dengan kromosom 0 yang ke k menyatakan kendaraan k kembali ke depot. Sebagai contoh (1 2 3 0 4 5 0) menyatakan kendaraan pertama dari depot menuju pelanggan 1 kemudian pelanggan 2 kemudian pelanggan 3 kemudian kembali ke depot dan dilanjutkan kendaraan kedua dari depot ke pelanggan 4 kemudian pelanggan 5 kemudian kembali lagi ke depot.

3.4.2.4 Solusi Terbaik

Solusi terbaik pada algoritma genetika adalah berupa himpunan solusi yang setiap solusinya memiliki nilai *fitness* yang terbaik. Nilai *fitness* adalah sebuah fungsi yang memetakan nilai kromosom ke bilangan *real*. Nilai *fitness* digunakan sebagai pembanding beberapa kromosom. Untuk kasus VRP ini nilai *fitness* yang digunakan adalah meminimalkan total jarak tempuh kendaraan.

Dikarenakan waktu komputasi untuk melakukan *packing constraints* sangat tinggi untuk mempersingkat waktu kita lakukan pembagian konstrain menjadi konstrain primer dan konstrain sekunder. Konstrain primer dalam kasus VRP-PC ini adalah konstrain berat dan volume barang sedangkan konstrain sekunder adalah konstrain penataan atau *packing contstraints*.

Konstrain primer membatasi semua solusi yang ada dalam individu. Sedangkan konstrain sekunder hanya membatasi semua solusi yang ada pada himpunan solusi terbaik.

3.4.2.5 Prosedur Algoritma

Pada tahap ini akan dibahas tentang urutan atau prosedur algoritma genetika untuk VRP yang akan dilakukan. Berikut urutan algoritmanya,

1. Inisialisasi data dan nilai parameter
2. Pembangkitan sebuah individu terbaik mula, individu terbaik harus memenuhi konstrain primer dan sekunder
3. Pembangkitan himpunan individu mula, semua individu dalam populasi harus memenuhi konstrain primer, ada dua jenis, pembangkitan individu mula yaitu *cluster generation* dan *random generation*
4. Pemberian nilai *fitness* dan uji konstrain sekunder, apabila ada individu yang memenuhi konstrain sekunder dengan nilai *fitness* lebih baik. Gantilah individu terbaik dengan individu dengan *fitness* lebih baik tersebut.
5. Kawin silang, menggunakan RBX dan SBX dengan menjamin terpenuhinya konstrain primer, apabila individu anak hasil kawin silang menghasilkan individu dengan *fitness* lebih buruk atau tidak memenuhi konstrain primer maka pertahankan orang tuanya dalam populasi, apabila tidak maka masukkan individu anak hasil kawin silang ke dalam populasi menggantikan individu orang tua.

6. Mutasi, Ada 2 jenis mutasi yang digunakan yaitu pembangkitan individu baru dan algoritma *2-opt*. Pembangkitan individu baru digunakan agar hasil tidak terjebak pada lokal optimal. Sedangkan algoritma *2-opt* digunakan untuk menghasilkan rute suatu individu menjadi lebih baik
7. Elitisme, masukkan individu terbaik ke dalam populasi
8. Pemberian nilai *fitness* dan uji konstrain sekunder, apabila ada individu yang memenuhi konstrain sekunder dengan nilai *fitness* lebih baik. Gantilah individu terbaik dengan individu dengan nilai *fitness* lebih baik tersebut.
9. Lakukan Pengulangan langkah 5 sampai 8.

3.4.2.6 Random Generation untuk VRP

Random generation adalah pembangkitan solusi atau individu dimana melakukan pembangkitan permutasi dari himpunan $\{1,2,3, \dots, n\}$ secara acak. Kemudian melakukan penyisipan nilai 0 ke dalam permutasi yang telah dibangkitkan. Penyisipan nilai 0 ke dalam permutasi dilakukan dengan memperhatikan kontrain primernya. Berikut algoritma lengkapnya,

1. Bangkitkan permutasi acak σ dari himpunan {1,2,3,…,n}
2. Buat variabel $tmp1 \coloneqq 0$ dan $tmp2 \coloneqq 0$ berturut-turut untuk menyimpan berat dan volume sementara
3. Untuk setiap i anggota σ dari kiri ke kanan
4. Substitusi $tmp1 \coloneqq tmp1 + berat(i)$ dan $tmp2 \coloneqq tmp2 + volume(i)$
5. Apabila $tmp1$ atau $tmp2$ melebihi kapasitas substitusi $tmp1 \coloneqq 0$ dan $tmp2 \coloneqq 0$, sisipkan angka 0 sebelum i di σ kemudian ulangi ke langkah 4
6. Kembali ke langkah 3.

3.4.2.7 Cluster Generation untuk VRP

Cluster generation adalah pembangkitan solusi atau individu dimana melakukan pembangkitan bedasarkan *cluster* sehingga diharapkan kendaraan akan melayani pelanggan-pelanggan yang berdekatan sehingga nilai $fitness$ menjadi lebih dekat. Algoritma ini diadopsi dari penelitian tentang *solution representation* [9].

Berikut algoritma *cluster generation,*

1. Untuk setiap kendaraan letakkan pada satu pelanggan secara acak
2. Buat *array* $tmp1 \coloneqq 0$ dan $tmp2 \coloneqq 0$ ukuran m menyatakan total berat yang ada pada masing-masing kendaraan
3. Buat matriks $priority$ ukuran $n \times m$ untuk meletakan prioritas kendaraan pada masing-masing *node* pelanggan
4. Bangkitkan permutasi acak σ dari $\{1,2,3, \dots, n\}$ menyatakan urutan prioritas.
5. Untuk setiap i anggota σ lakukan
6. Ukur jarak i ke masing-masing kendaraan dan beri urutan rendah ke tinggi dan masukkan ke $priority(i)$
7. Kembali ke 5
8. Untuk setiap i anggota σ dari kiri ke kanan lakukan
9. Pilih kendaraan bedasarkan $priority(i)$
10. Apabila $tmp1\big(priority(i)\big) + berat(i) > kapasitasberat$ atau $tmp2\big(priority(i)\big) + volume(i) > kapasitasvolume$ maka kembali ke langkah 9
11. Substitusi $tmp1\big(priority(i)\big) \coloneqq tmp1\big(priority(i)\big) + berat(i)$ dan $tmp2\big(priority(i)\big) \coloneqq tmp2\big(priority(i)\big) + volume(i)$
12. Kembali ke langkah 8.

3.4.2.8 Improved Sequence-Based Crossover (ISBX) dan Improved Route-Based Crossover (IRBX)

Pada *sequence-based crossover* (SBX) dan *route-based crossover* (RBX) memiliki kelemahan yaitu pemilihan gen-gen *tour* yang akan ditukarkan dilakukan secara acak tanpa memperhatikan nilai *fitness*-nya. Maka dapat dilakukan pengembangan algoritma untuk menghasilkan solusi yang lebih baik dengan mengganti faktor acak pada kedua SBX dan RBX dengan faktor *roulette wheel*. Misal x_{p1} dan x_{p2} menyatakan kedua *parent* dan x_{o1} dan x_{o2} menyatakan kedua *offspring*

Berikut Algoritma untuk ISBX,

1. $x'_{o1} = x_{p1}$
2. Ambil rute acak s_{r1} dari x_{p1} dan rute acak s_{r2} dari x_{p2} dengan faktor *roulette wheel* setiap rutenya $\sum_{y_i \in sr_1} iy_i$

dengan y_i keuntungan *fitness* apabila gen y_i pada x_{p1} dipindah ke x_{p2}

3. Buatlah rute baru s_{new} dengan menambah *task* dari s_{r1} hingga suatu titik acak lalu tambahkan *task* dari sr_2 dari titik acak hingga akhir
4. Hilangkan duplikat pada s_{new} apabila ada
5. Hilangkan *task* pada x'_{o1} yang ada pada s_{new}
6. Hilangkan s_{r1} dan pindahkan semua *task* ke T_{temp}
7. Tambahkan s_{new} ke x'_{o1}
8. Buatlah x_{o1} dari x'_{o1} dengan menambahkan *task* yang ada pada T_{temp} dengan mengevaluasi fungsi objektif dengan metode *cheapest insertion* dan mempertahankan konstrain primer.
9. Buatlah x_{o2} dengan menukar *parent* dan melakukan pengulangan langkah 1 sampai 8.

Berikut Algoritma untuk RBX,

1. $x'_{o1} = x_{p1}$
2. Ambil rute acak s_{r2} dari x_{p2} dengan faktor *roulette wheel* rata-rata keuntungan setiap gen apabila gen-gen *tour* s_{r2}dipindah ke x_{p1}
3. Hilangkan semua *task* dari x'_{o1} yang ada pada s_{r2}
4. Hilangkan satu rute acak s_{r1} dengan faktor *roulette wheel* rata-rata kerugian setiap gen apabila s_{r1} dihilangkan dari x'_{o1} dan pindahkan semua *task* ke T_{temp}
5. Tambahkan rute s_{r2} ke x'_{o1}
6. Buatlah x_{o1} dari x'_{o1} dengan menambahkan *task* yang ada pada T_{temp} dengan mengevaluasi fungsi objektif dengan metode *cheapest insertion* dan mempertahankan konstrain primer
7. Buatlah x_{o2} dengan menukar *parent* dan melakukan pengulangan langkah 1 sampai 6.

3.4.2.9 Algoritma 2-opt

Algoritma *2-opt* merupakan algoritma heuristik untuk menyelesaikan *travelling salesman problem.* Algoritma ini memiliki hal

unik yaitu mengganti subrute yang saling silang dan menggantinya dengan yang lebih dekat dengan cara membalik rute yang digantinya.

Algoritma ini pada VRP digunakan untuk memperbaiki nilai *fitness* masing-masing *tour* yang ada. Berikut algoritma lengkapnya

Ambil barisan s dari salah satu *tour* kendaraan dan berikan permulaan dan penutupan s dari depot atau *node* 0 sehingga seolah s menjadi barisan sebuah *tour* yang bergerak bermula depot dan berakhir di depot,

1. Untuk setiap i barisan s dari kiri ke kanan
2. Untuk setiap j dari barisan s dari $i + 2$ ke kanan
3. Simpan jarak *subtour* dari $i + 1$ ke j yaitu penjumlahan *subtour* $i + 1$ ke $j - 1$ dan *subtour* $j - 1$ ke j
4. Simpan jarak *subtour* dari j ke $i + 1$ yaitu penjumlahan *subtour* $j - 1$ ke $i + 1$ dan *subtour* j *ke* $j - 1$
5. Apabila jarak i ke $i + 1$ditambah jarak *subtour* $i + 1$ ke j ditambah jarak j ke $j + 1$ lebih dari jarak i ke j ditambah jarak j ke $i + 1$ ditambah jarak $i + 1$ ke $j + 1$ maka balik urutan dari $i + 1$ hingga j dari s kemudian ulangi langkah 2
6. Kembali ke langkah 3
7. Kembali ke langkah 2.

3.4.2.10 Metode Cheapest Insertion

Metode ini adalah merupakan metode penambahan suatu *node* ke dalam sebuah *tour* dengan memperhatikan fungsi objektif. Berikut langkah algoritma ini untuk memasukkan sebuah *node* i ke dalam sebuah *tour*,

1. Untuk setiap *node* j dan $j + 1$ pada sebuah *tour* evaluasi semua jarak j ke i ditambah i ke $j + 1$
2. Ambil j dengan evaluasi jarak terpendek dan sisipkan i di antara j dan $j + 1$ pada *tour* tersebut.

3.4.3 Penggunaan Algoritma GA-BLF

Pada tahap ini dilakukan pengembangan algoritma genetika yang digabung dengan algoritma *bottom-left fill*. Algoritma genetika dalam hal ini digunakan untuk menentukan urutan penyusunan barang kemudian dilanjutkan ke algoritma *bottom-left fill* untuk dilakukan proses

penataan barang. Apabila ditemukan sebuah solusi yang memenuhi maka algoritma ini dihentikan.

3.4.3.1 Input Data Algoritma Genetika

Input dari algoritma ini berbentuk barang-barang yang akan dikirim pada sebuah *tour* yang telah ditentukan pada algoritma genetika untuk VRP.

3.4.3.2 Parameter Algoritma Genetika, Struktur Data Kromosom dan Nilai Fitness

Dalam algoritma genetika banyak individu dan lama generasi merupakan parameter yang dapat dikontrol dan diperlukan untuk mendapat kualitas algoritma yang baik. Selain parameter tersebut adapula parameter yang tidak dapat dikontrol yaitu lebar kromosom yang ada. Misal kita definisikan parameter tersebut,

ncr : Banyak kromosom dalam setiap individu dengan indeks icr

$npop$: Banyak individu dalam populasi dengan indeks $ipop$

$ngen$: Banyak generasi yang dibangkitkan dengan indeks $igen$.

Setiap individu merupakan solusi yang bisa jadi belum mungkin untuk sepenuhnya ditata pada algoritma BLF. Ada dua buah kromosom yang membentuk setiap individu. Kromosom pertama membentuk *list* urutan. *List* tersebut menyimpan urutan barang yang akan ditata pada algoritma BLF. Kemudian kromosom kedua berbentuk *array* yang menyatakan posisi barang. Kromosom tersebut bernilai 1 apabila barang dirotasi dan bernilai 0 apabila barang tidak di rotasi.

Solusi terbaik ditentukan oleh nilai *fitness*. Nilai *fitness* berupa total okupasi yang dapat diisi oleh beberapa barang terakhir. Prosedur nilai *fitness* diletakkan pada algoritma BLF.

3.4.3.4 Prosedur Algoritma Genetika

Pada tahap ini akan dibahas tentang urutan tahapan untuk proses algoritma genetika untuk melakukan urutan barang. Berikut urutan algoritma genetika,

1. Inisialisasi data dan nilai parameter, berikan *flag* tidak berhasil
2. Pembangkitan himpunan individu mula, ada beberapa jenis pembangkitan yang dilakukan antara lain LAF-*priority*, LIFO-*priority*, LAF-LIFO-*priority*, *area-based random generation* untuk kromosom urutan dan *size-based random generation* untuk kromosom posisi
3. Pemberian nilai *fitness*, apabila ada individu yang memenuhi, berikan *flag* berhasil dan hentikan algoritma ini, masukkan individu dengan *fitness* terbaik ke dalam himpunan individu terbaik
4. Kawin silang, ada dua kawin silang yang dapat dilakukan yaitu kawin silang kromosom jenis dan kawin silang kromosom posisi. Untuk kawin silang jenis digunakan *arithmatic-decoded* kawin silang sedangkan untuk kawin silang kromosom posisi digunakan *uniform crossover*
5. Mutasi yang digunakan adalah pembangkitan individu baru. Pembangkitan individu baru digunakan agar hasil tidak terjebak pada *local optimum*.
6. Masukkan individu terbaik ke dalam populasi
7. Pemberian nilai *fitness*, apabila ada individu yang memenuhi, berikan *s* berhasil dan hentikan algoritma ini, masukkan individu dengan *fitness* terbaik ke dalam himpunan individu terbaik
8. Lakukan pengulangan langkah 3 sampai 7.

3.4.3.5 LAF-Priority, Area-Based, LIFO-priority, LA-LIFO-priority Generations

Beberapa algoritma BLF yang dapat digunakan adalah algoritma LAF-BLF atau *large-area-first bottom-left fill*. Algoritma ini membentuk urutan barang bedasar luas penampang barang yang diurutkan dari besar ke yang terkecil. Berdasarkan algoritma ini, dapat dilakukan pembangkitan sebuah individu dengan kromosom urutan bedasarkan luas penampang yang diurutkan dari besar ke yang kecil.

Selain itu untuk melakukan *generation* urutan barang yang acak namun tetap memperhatikan luas penampang barang yang disebut *area-based generation.* Algoritma ini memiliki langkah sebagai berikut,

1. Setiap barang diberikan nilai prioritas sebesar $rand * LuasPenampang$
2. Kemudian nilai-nilai prioritas yang ada diurutkan dan dibentuk individu baru dari urutan tersebut.

Kemudian karena ada konstrain *last-in first-out* (LIFO) maka perlu adanya pembangkitan individu yang berdasarkan urutan barang yaitu LIFO-*priority generations*. Pembangkitan ini memerlukan *input* urutan sebuah *tour* dengan urutan terbalik atau dari akhir pengiriman ke awal pengiriman. Kemudian barang-barang yang dikirim pada pelanggan yang sama dilakukan pengacakan seperti halnya *area-based generations*.

Gabungan dari LAF-*priority* dan LIFO-*priority* dapat dihasilkan sebuah individu yang memenuhi konstrain LIFO kemudian untuk barang-barang pada pelanggan yang sama diurutkan bedasarkan urutan barang dari luas penampang barang dari yang paling besar ke paling kecil. Pembangkitan individu ini disebut LAF-LIFO-*priority generations.*

3.4.3.6 Size-Based Generations

Algoritma ini digunakan untuk membangkitkan kromosom posisi setiap barang. Dasar algoritma ini adalah penggunaan *roulette wheel* dengan memberikan prioritas barang yang memiliki luas daerah yang tidak memiliki koordinat yang sama dengan posisi tersebut.

Misalkan pada barang α memiliki panjang l_α dan lebar w_α sedangkan kontainer memiliki ukuran panjang L dan lebar W. Maka prosedur algoritmanya adalah,

1. Untuk setiap barang α
2. Masukkan nilai untuk membatasi *roulette wheel* $\alpha = \frac{\max((L-l_\alpha)(W-w_\alpha),0)}{\max((L-l_\alpha)(W-w_\alpha),0)+\max((L-w_\alpha)(W-l_\alpha),0)}$
3. Bangkitkan bilangan acak $rand \in [0,1]$
4. Apabila $rand < \alpha$ maka kromosom di *array* α diberi nilai 0 dan apabila $rand \geq \alpha$ maka kromosom di *array* α diberi nilai 1
5. Kembali ke langkah 1.

3.4.3.7 Arithmetic-Decoded Crossover

Pada algoritma ini akan dijelaskan salah satu kawin silang untuk kromosom urutan. Kawin silang ini pada umumnya digunakan untuk permasalahan dengan nilai-nilai kontinyu. Namun pada permasalahan

optimasi permutasi dapat dilakukan dengan menambahkan algoritma *decode* yang berupa algoritma *sorting* atau pengurutan kembali. Misal x_{p1} dan x_{p2} adalah kromosom orang tua dan x_{c1} dan x_{c2} adalah kromosom anak yang berbentuk barisan urutan barang. Maka langkah algoritmanya adalah sebagai berikut,

1. Bangkitkan bilangan acak $rand$
2. $x_{c1} \coloneqq rand \times x_{p1} + (1 - rand) \times x_{p2}$
3. $x_{c2} \coloneqq rand \times x_{p2} + (1 - rand) \times x_{p1}$
4. Kemudian buat *list* urutan (berbentuk permutasi dari $\{1,2,3, \dots, ncr\}$) dari x_{c1} dan x_{c2} masukkan *list* tersebut kedalam x_{c1} dan x_{c2}.

3.4.3.8 Uniform Crossover

Kawin silang ini digunakan pada kromosom posisi yang berbentuk biner. Prosedur kromosom ini adalah menukar posisi pada barang yang sama di dua individu.

Misal x_{p1} dan x_{p2} adalah kromosom orang tua dan x_{c1} dan x_{c2} adalah kromosom anak yang berbentuk *array* posisi barang. Langkah dari *uniform-crossover* adalah sebagai berikut,

1. Untuk setiap barang i
2. Bangkitkan bilangan acak $rand \in [0,1]$
3. Apabila $rand < 0.5$ maka masukkan kromosom *array* i x_{c1} dari x_{p2} dan x_{c2} dari x_{p1}
4. Apabila $rand \geq 0.5$ maka masukkan kromosom *array* i x_{c1} dari x_{p1} dan x_{c2} dari x_{p2}
5. Kembali ke langka 1.

3.4.3.9 Complete-BLF

Complete-BLF merupakan algortima pengembangan dari BLF. Untuk kasus 3D-BPP, BLF menyimpan 3 buah titik ke dalam *list* setiap kali ada barang yang dapat ditata. Algortima ini memerlukan *input* urutan dan posisi barang dari algoritma genetika. Berikut adalah algoritma *complete*-BLF,

1. Lakukan rotasi horizontal pada barang-barang yang memiliki kromosom rotasi bernilai 1
2. Buat *list* kosong L dan masukkan koordinat $(0,0,0)$
3. Masukkan nilai $fitness \coloneqq L * W * H$

4. Bedasarkan kromosom urutan barang. Lakukan langkah berikut untuk barang i
5. Untuk setiap barang j yang sudah diletakkan periksa dengan barang i untuk konstrain *overlaping*, luas penopang minimal, keseimbangan, berat maksimal dan LIFO. Apabila barang i muat pada $l\ (l_x, l_y, l_z)$ anggota *list L* lanjutkan langkah. Apabila tidak ulangi langkah ini.
6. Kemudian hapus l di L.
7. Untuk setiap barang j yang sudah diletakkan simpan semua *bottom-left point* yang dibentuk barang i dan j ke dalam *list.*
8. Urutkan semua koordinat yang ada pada *list* dengan prioritas sumbu x terkecil, kemudian apabila ada yang bernilai sama urutkan berdasarkan prioritas sumbu *z* terkecil dan apabila ada yang sama lagi urutkan berdasarkan prioritas sumbu y terkecil.
9. Letakkan barang i kemudian kurangi nilai *fitness* sebesar volume barang yang ada. $fitness \coloneqq fitness - l(i) \times w(i) \times h(i)$
10. Apabila tidak ada barang yang muat maka keluarkan hasil tidak memenuhi
11. Kembali ke langkah 4 hingga semua barang habis
12. Apabila semua barang muat maka keluarkanlah hasil yang memenuhi.

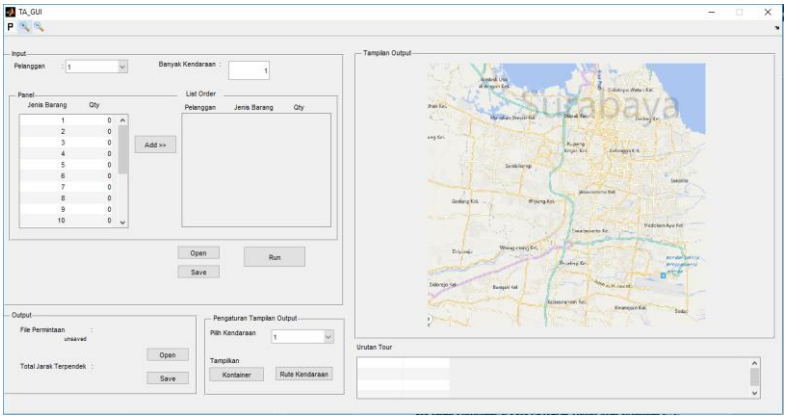
3.5 Implementasi Algoritma

Algoritma-algoritma yang telah dirancang diimplementasikan dalam bentuk *software* dengan GUI dan tanpa GUI pada MATLAB. Dikarenakan ada dua kategori implementasi yaitu untuk perbandingan algoritma dan implementasi pada suatu perusahaan di Surabaya maka digunakan digunakan konstrain dan data yang berbeda. Untuk perbandingan algoritma digunakan konstrain dan data yang sama dengan penelitian [1] sedangkan untuk implementasi pada perusahaan digunakan konstrain-konstrain seperti yang telah dimodelkan. Berikut perbedaan pada kedua implementasi ditampilkan pada Tabel 3.2.

Tabel 3.2. Perbedaan pada Kedua Implementasi

Perbandingan Algoritma	Perusahaan
Menggunakan data sekunder	Menggunakan hasil pemodelan jaringan data jalan kota Surabaya
Menggunakan konstrain seperti pada model selain konstrain keseimbangan dan maksimal beban ditopang [1]	Menggunakan konstrain seperti pada model dengan asumsi tidak ada barang yang bersifat *fragile*
Tidak menggunakan GUI	Menggunakan GUI

Berikut Gambar 3.2 adalah tampilan GUI dari program yang telah dibuat



Gambar 3.2 Tampilan GUI Program

Dari GUI pada Gambar 3.2 memiliki 4 panel utama yaitu

1. *Input*, berisi masukan barang tiap pelanggan, *list* pemesanan dari data yang dimasukan, *save* untuk menyimpan data yang telah dimasukan dan *open* untuk mengambil data yang telah disimpan. Serta *run* untuk menjalankan program
2. *Output*, berisi hasil dari *running* program, lokasi data, *save* untuk menyimpan hasil dan *open* untuk mengambil data dari program yang telah dijalankan.

3. Tampilan *output*, sebuah gambar yang akan mengeluarkan gambar rute kendaraan pada peta dan penataan barang pada kontainer beserta tabel berisi urutan pelanggan yang akan dikunjungi.
4. Pengaturan tampilan *output*, berisi *listbox* untuk memilih kendaraan dan tombol-tombol untuk menampilkan rute kendaraan dan penataan barang.

3.6 Verifikasi dan Validasi

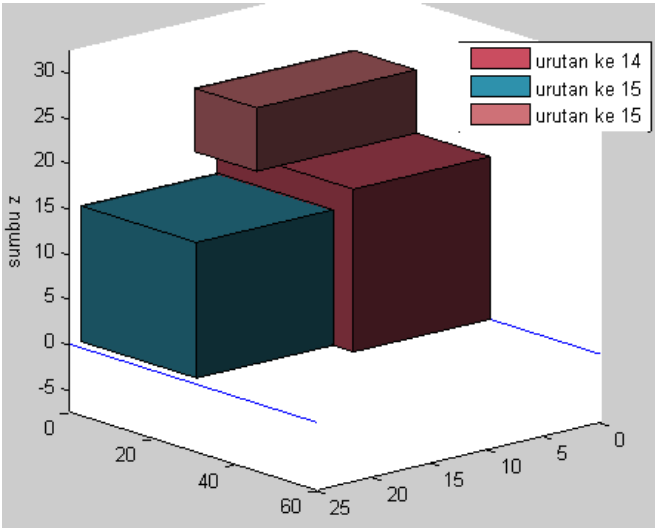
Verifikasi dan validasi pada hasil penelitian ini digunakan sebagai syarat bahwa program yang diminta memenuhi konstrain yang berlaku dan hasil yang dihasilkan mendekati optimal. Dalam proses verifikasi dan validasi berikut digunakan data sekunder [1].

3.6.1 Verifikasi

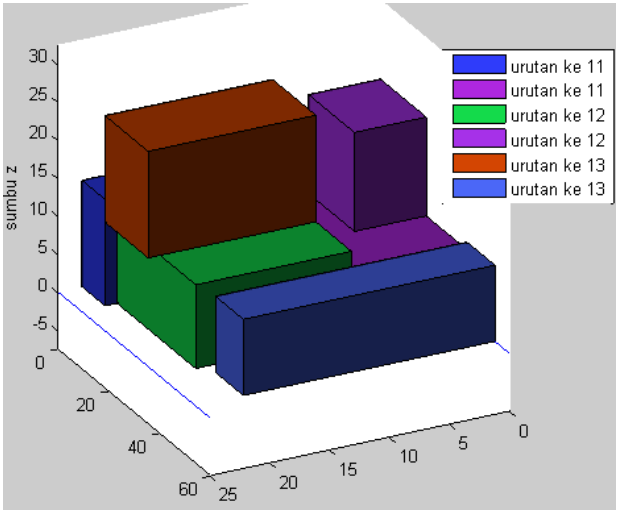
Pada verifikasi algoritma berikut menggunakan data kedua pada penelitian [1] sebagai syarat bahwa solusi yang dihasilkan memenuhi konstrain-konstrain yang ada. Karena mengikuti penelitian [1] maka digunakan konstrain-konstrain yang ada dalam penelitiannya. Satuan yang digunakan menggunakan satuan *unit.* Pada MATLAB sendiri tidak ditemukan *error* ketika dijalankan. Berikut hasil verifikasi ditampilkan pada Tabel 3.3 dan Gambar 3.3 sampai Gambar 3.7.

Tabel 3.3. Hasil *Running* pada Data Kedua Penelitian [1]

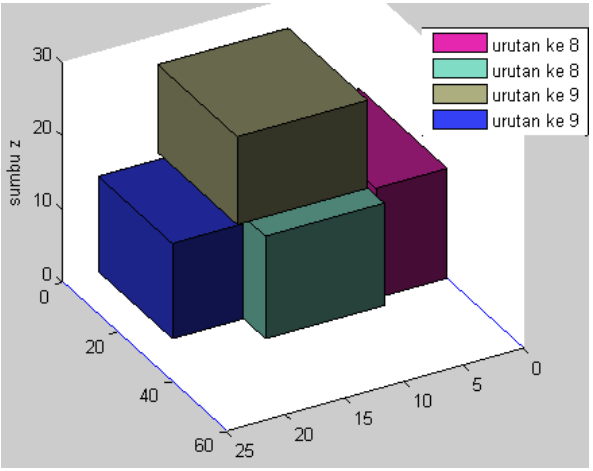
Kendaraan	Kunjungan 1	Kunjungan 2	Kunjungan 3	Kunjungan 4
1	12	5	-	-
2	8	4	13	-
3	6	7	14	-
4	1	3	2	-
5	11	9	10	15



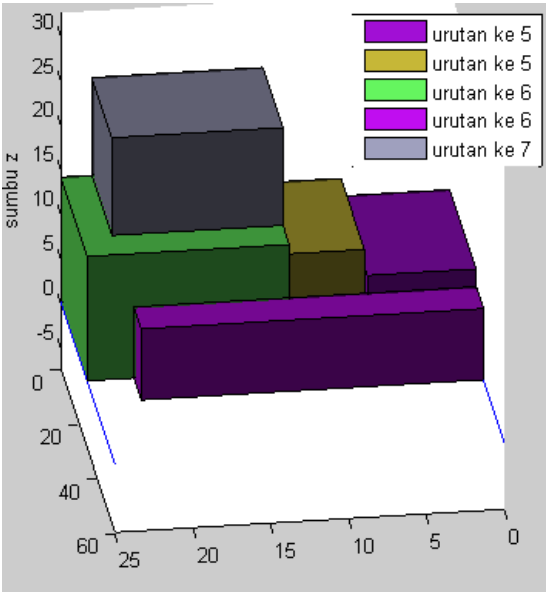
Gambar 3.3 Penataan Barang pada Kendaraan 1 dalam Satuan *Unit*



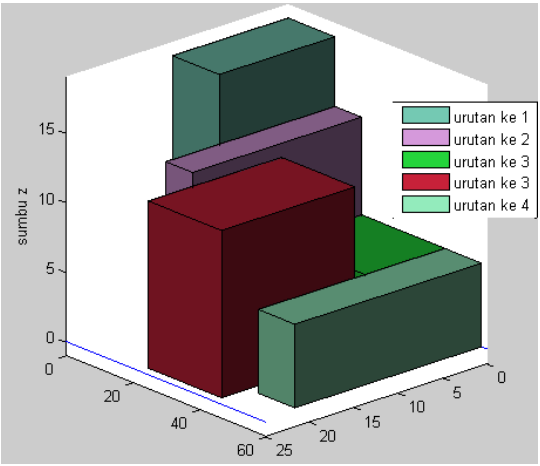
Gambar 3.4 Penataan Barang pada Kendaraan 2 dalam Satuan *Unit*



Gambar 3.5 Penataan Barang pada Kendaraan 3 dalam Satuan *Unit*



Gambar 3.6 Penataan Barang pada Kendaraan 4 dalam Satuan *Unit*



Gambar 3.7 Penataan Barang pada Kendaraan 5 dalam Satuan *Unit*

Dari hasil 5 kendaraan tersebut kita ujikan barang-barang pada konstrain-konstrain yang ada. Setelah dilakukan pengujian didapat hasil yang sesuai model maka hasil telah terverifikasi oleh model.

3.6.1 Validasi

Pada bagian ini akan dibahas mengenai validasi dari hasil yang telah ada. Kemudian pada Bab 4 Hasil dan Analisa akan dilakukan perbandingan algoritma dimana apakah hasilnya mendekati nilai sesuai pada penelitian [1] apa tidak. Sehingga validasi dapat dilakukan bersamaan dengan perbandingan algoritma yang ada pada Bab 4.

BAB 4
HASIL DAN ANALISA

Pada bab ini akan dijelaskan mengenai implementasi perangkat lunak yang telah dirancang dan dibuat sebelumnya. Sebelum itu dilakukan pengumpulan data-data yang telah dibutuhkan. Ada 2 data yang akan digunakan yang pertama adalah data sekunder dari penelitian [1]. Terdapat 5 data yang akan diuji dan ata tersebut meliputi koordinat depot dan pelanggan dengan menggunakan jarak berupa *euclidian distance* beserta barang-barang permintaan setiap pelanggan, spesifikasi ukuran barang dan total berat yang diminta suatu pelanggan. Dari data tersebut diharapkan mendapat perbandingan hasil dari penelitian [1]. Kemudian data selanjutnya adalah data riil dari perusahaan dan hasil pengamatan jalan. Data tersebut berupa koordinat depot dan pelanggan, segmentasi jalan suatu kota dan jenis-jenis barang yang dapat diminta oleh pelanggan sehingga untuk permintaan barang menggunakan asumsi. Pengujian pertama dengan data sekunder dengan perangkat lunak dilakukan beberapa kali per pengujian. Setiap pengujian membandingkan nilai-nilai parameter yang berbeda-beda antara lain parameter pembangkitan generasi, kawin silang, mutasi baik pada algoritma genetika untuk VRP ataupun untuk pengurutan barang. Dari pengujian tersebut akan ditampilkan nilai-nilai *fitness* beserta identifikasi statistik seperti rata-rata, solusi terbaik dan terburuk dan standar deviasi. Pengujian dengan hasil terbaik akan dibandingkan dengan penelitian [1] yang menggunakan algoritma *ant colony optimization* (ACO) dan *tabu search*. Untuk implementasi menggunakan data riil perusahaan akan digunakan beberapa pemesanan setiap pelanggan selama 7 kali dan akan ditampilkan *routing* kendaraan beserta barang yang ditata.

4.1 Pengumpulan dan Pengolahan Data Sekunder Penelitian

Untuk menerapkan rancangan sistem yang telah dibuat diperlukan sejumlah data diantaranya koordinat pelanggan dan data pesanan barang beserta ukuran barang dan kontainer kendaraan. Selanjutnya dari data koordinat akan diubah menjadi data jarak antara pelanggan-pelanggan dan depot dengan menggunakan *euclidian distance*. Hal ini dilakukan agar sesuai dengan konsep yang dirancang dari penelitian [1].

4.2 Pengujian dengan Beberapa Parameter

Dari data pada yang telah diperoleh akan dilakukan beberapa pengujian beberapa parameter untuk membangkinkan hasil-hasil. Parameter-parameter tersebut adalah,

1. Populasi dan generasi / iterasi algoritma genetika untuk VRP
    a. $npop$: Banyak populasi dengan nilai *default* $npop \coloneqq 200$
    b. $ngen$: Banyak generasi atau iterasi dengan nilai *default* $ngen \coloneqq 40$
2. Parameter pembangkitan generasi awal algoritma genetika untuk VRP
    a. pgc : Menggunakan *cluster generation* dengan nilai *default* $pgc \coloneqq 0{,}6$
    b. pgr : Menggunakan *random generation*, memiliki nilai $pgr \coloneqq 1 - pgc$
3. Parameter kawin silang algoritma genetika untuk VRP
    a. $pcsbx$: Menggunakan *improved sequence-based crossover* dengan nilai default $pcsbx \coloneqq 0{,}4$
    b. $pcrbx$: Menggunakan *improved route-based crossover* memiliki nlai $pcrbx \coloneqq 1 - pcsbx$
4. Parameter mutasi algoritma genetika untuk VRP
    a. pm : Menggunakan algoritma pembangkitan generasi awal dengan nilai *default* $pm \coloneqq 0{,}1$
    b. $pm2opt$: Menggunakan algoritma *2-opt* dengan nilai *default* $pm2opt \coloneqq 0{,}2$
5. Populasi dan generasi / iterasi untuk GA-BLF
    a. $nbpop$: Banyak populasi dengan nilai *default* $nbpop \coloneqq 100$
    b. $nbgen$: Banyak generasi atau iterasi dengan nilai *default* $nbgen \coloneqq 20$
6. Parameter pembangkitan generasi awal algoritma genetika untuk VRP
    a. $pbgab$: Menggunakan *area-based generation* dengan nilai *default* $pbgab \coloneqq 0{,}1$

b. $pbglifo$: Menggunakan LIFO-*priority generation*, memiliki nilai $pbglifo \coloneqq 1 - pbgab$

7. Parameter kawin silang GA-BLF
   a. $pbca$: Menggunakan *arithmetic-decoded crossover* pada kromosom urutan dengan nilai *default* $pbca \coloneqq 0{,}5$
   b. $pbcu$: Menggunakan *uniform crossover* pada kromosom posisi memiliki nlai $pbca \coloneqq 1 - pbcu$
8. Parameter mutasi GA-BLF
   a. pbm : Menggunakan algoritma pembangkitan generasi awal dengan nilai *default* $pm \coloneqq 0{,}15$

Nilai *default* merupakan nilai yang digunakan ketika menguji parameter yang lain. Untuk konstrain pada penataan barang yang digunakan mengikuti konstrain yang ada di penelitian [1]. Untuk data yang digunakan dalam pengujian ini digunakan data 2.

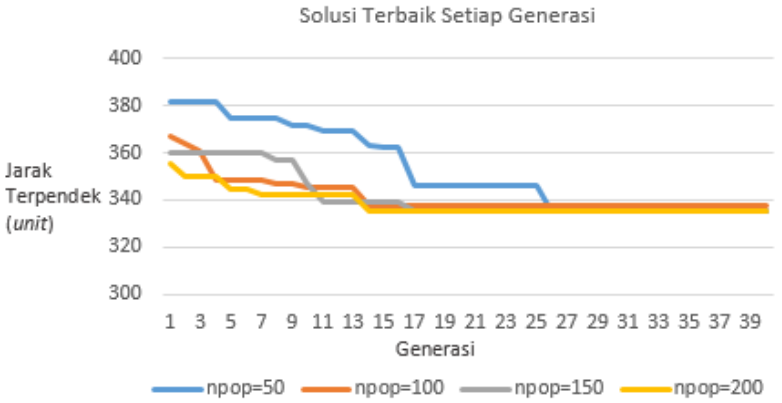
4.2.1 Pengujian Parameter Populasi

Pada tahap ini akan diuji dengan beberapa nilai untuk parameter populasi dan nilai *default* untuk parameter yang lain. Nilai-nilai yang diuji antara lain $npop = 50, 100, 150$ dan 200 masing-masing selama 10 kali dan akan disajikan pada Tabel 4.1 dan Gambar 4.1.

Tabel 4.1 Pengujian Paremeter Populasi dalam Satuan *Unit*

No.	$npop$			
	50	100	150	200
1	343,8639	344,7179	337,9513	337,9513
2	348,7711	344,9859	341,2119	346,6744
3	342,4307	349,4052	345,0076	334,9639
4	345,725	336,5181	334,9639	334,9639
5	345,725	341,9984	334,9639	341,9984
6	334,9639	336,5181	334,9639	334,9639
7	350,1453	337,9513	334,9639	334,9639
8	350,5766	347,098	341,9984	337,9513
9	346,6744	341,9984	345,3604	334,9639

No.	*npop*			
	50	100	150	200
10	350,6305	349,9505	347,995	334,9639
mean	345,9506	343,1142	339,938	337,4359
stdev	4,795013	4,993552	5,042048	3,991133
best	334,9639	336,5181	334,9639	334,9639
worst	350,6305	349,9505	347,995	346,6744



Gambar 4.1 Perbandingan Nilai Terbaik Setiap Generasi

Dari hasil pada Gambar 4.1 dapat disimpulkan semakin banyak individu dalam populasi dapat menghasilkan hasil yang lebih baik.

4.2.2 Pengujian Parameter Pembangkitan Populasi Awal

Pada tahap ini dilakukan pengujian nilai untuk pgc yaitu peluang dibangkitkan populasi dengan *cluster generation* atau dengan pgr yaitu peluang dibangkitkan populasi dengan *random generation*. Nilai-nilai yang diuji untuk pgc antara lain $pgc = 0, 0{,}2\ , 0{,}4\ , 0{,}6, 0{,}8, 1$. Masing-masing akan dijalankan 10 kali dan akan disajikan dalam Tabel 4.2.

Tabel 4.2 Pengujian Parameter Pembangkitan Populasi Awal dalam Satuan *Unit*

No.	pgc					
	0	0,2	0,4	0,6	0,8	1
1	358,94	347,91	334,96	336,52	334,96	334,96
2	363,21	337,95	334,96	334,96	334,96	334,96
3	391,10	342,00	334,96	342,00	334,96	334,96
4	377,9	351,67	337,95	334,96	334,96	334,96
5	357,14	356,17	342,00	334,96	344,72	336,52
6	368,15	349,93	337,95	334,96	334,96	334,96
7	367,86	352,66	349,95	345,01	334,96	334,96
8	358,35	349,41	336,52	337,95	334,96	334,96
9	369,94	334,96	334,96	337,95	334,96	334,96
10	376,43	336,52	349,41	334,96	334,96	334,96
mean	368,90	345,92	339,36	337,42	335,93	335,11
stdev	10,618	7,4757	5,863	3,4972	3,0845	0,4915
best	357,14	334,96	334,96	334,96	334,96	334,96
worst	391,10	356,17	349,95	345,01	344,72	336,52

Dari hasil pada Tabel 4.2 terlihat penggunaan *cluster based generation* sangat berpengaruh terhadap hasil terbaiknya. Maka menurut data pada Tabel 4.2 nilai $pgc = 0{,}8$ dan $pgc = 1$ memiliki hasil yang hampir sama hanya berbeda pada satu data saja. Maka akan digunakan nilai $pgc = 0{,}8$.

4.2.3 Pengujian Parameter Kawin Silang

Pada tahap ini dilakukan pengujian pada nilai $psbx$ yaitu peluang melakukan kawin silang dengan ISBX dan $prbx$ yaitu peluang melakukan kawin silang dengan IRBX. Nilai-nilai yang diuji antara lain $psbx = 0, 0{,}2\ , 0{,}4\ , 0{,}6\ , 0{,}8\ , 1$. Berikut hasil yang dihasilkan 10 kali per nilai ditampilkan pada Tabel 4.3.

Tabel 4.3 Pengujian Parameter Kawin Silang dalam Satuan *Unit*

No.	$psbx$					
	0	0,2	0,4	0,6	0,8	1
1	334,96	336,52	334,96	334,96	345,57	334,96
2	336,52	337,95	337,95	337,95	342	354,96
3	336,52	344,11	334,96	334,96	342	344,11
4	339,66	334,96	334,96	334,96	334,96	345,36
5	339,66	334,96	334,96	334,96	334,96	340,48
6	336,52	334,96	337,95	344,72	344,99	334,96
7	334,96	344,11	339,66	337,95	334,96	334,96
8	334,96	342	334,96	337,95	341,21	339,5
9	337,95	334,96	334,96	337,95	337,95	337,95
10	334,96	334,96	337,95	337,95	345,36	336,52
mean	336,667	337,949	336,327	337,43	340,39	340,37
stdev	1,86436	3,9305	1,83253	2,9608	4,3837	6,3334
best	334,96	334,96	334,96	334,96	334,96	334,96
worst	339,66	344,11	339,66	344,72	345,57	354,96

Dari Hasil pada Tabel 4.3 untuk $psbx = 0$ memiliki standar deviasi lebih baik namun tidak sering mendapat hasil paling optimal. Sedangkan untuk $psbx = 0{,}2\ , 0{,}4, 0{,}6$ memiliki hasil yang mirip dan lebih sering mendapat hasil yang optimal sedangkan untuk $psbx = 0{,}8\ , 1$ memiliki rata-rata yang lebih buruk. Dalam kasus ini kita ambil rata-rata terkecilnya yaitu ketika $psbx = 0{,}4$.

4.2.4 Pengujian Parameter Mutasi

Pada tahap ini akan dilakukan pengujian parameter mutasi pm yaitu peluang terjadi mutasi pada suatu individu dengan algoritma pembangkitan individu ulang. Nilai-nilai yang diuji antara lain $pm = 0, 0{,}1\ , 0{,}2, 0{,}3, 0{,}5, 0{,}8$. Berikut hasil dari pengujian pada Tabel 4.4.

Tabel 4.4 Pengujian Parameter Mutasi dalam Satuan *Unit*

No.	pm					
	0	0,1	0,2	0,3	0,5	0,8
1	345	342	334,96	342	334,96	339,66
2	344,11	336,52	341,21	339,51	344,99	344,72
3	337,95	334,96	339,51	337,95	342	336,52
4	337,95	334,96	334,96	348,73	334,96	339,66
5	337,95	334,96	337,95	337,95	334,96	337,96
6	337,95	334,96	339,51	337,95	346,96	339,52
7	339,66	336,82	342	334,96	336,52	339,66
8	334,96	334,96	337,95	348,68	339,66	341,21
9	334,96	337,95	336,52	334,96	345,01	336,52
10	334,96	337,95	342	336,52	344,72	339,51
mean	338,54	336,6	338,66	339,92	340,47	339,49
stdev	3,5567	2,268	2,65050	5,0690	4,8408	2,3660
best	334,96	334,96	334,96	334,96	334,96	336,52
worst	345	342	342	348,73	346,96	344,72

Dari hasil pada Tabel 4.4 dapat di lihat untuk $pm = 0, 0{,}3, 0{,}5$ menghasilkan standar deviasi yang cukup tinggi. Untuk $pm = 0{,}1\ , 0{,}2$ menghasilkan rata-rata nilai rendah dan standar deviasi rendah. Untuk $pm = 0{,}8$ dalam kasus ini tidak ada kawin silang namun tidak menghasilkan solusi bernilai $334{,}96$ *unit*. Maka dalam pertimbangan pada Tabel 4.4 nilai yang baik adalah $pm = 0{,}1$.

4.2.5 Pengujian Parameter Mutasi dengan Algoritma *2-opt*

Pada bagian ini dilakukan pengujian parameter mutasi dengan algoritma *2-opt* $pm2opt$ yaitu peluang terjadinya mutasi dengan algortima *2-opt*. Nilai-nilai yang diuji antara lain $0, 0{,}2, 0{,}4, 0{,}6, 0{,}8, 0{,}9$. Hasil pengujian disajikan pada Tabel 4.5 berikut.

Tabel 4.5 Pengujian Parameter Mutasi dengan Algortima *2-opt* dalam Satuan *Unit*

No.	$pm2opt$					
	0	0,2	0,4	0,6	0,8	0,9
1	348,68	342	334,96	334,96	334,96	334,96
2	356,56	336,52	337,95	336,52	334,96	334,96
3	342	334,96	334,96	334,96	334,96	334,96
4	334,96	334,96	334,96	334,96	334,96	334,96
5	339,51	334,96	334,96	334,96	334,96	334,96
6	348	334,96	334,96	334,96	334,96	334,96
7	344,11	336,82	334,96	334,96	334,96	334,96
8	347,73	334,96	334,96	334,96	334,96	334,96
9	347,1	337,95	334,96	334,96	334,96	334,96
10	337,95	337,95	339,51	334,96	334,96	334,96
mean	344,66	336,6	335,71	335,12	334,96	334,96
stdev	6,3009	2,2683	1,6315	0,4933	0	0
best	334,96	334,96	334,96	334,96	334,96	334,96
worst	356,56	342	339,51	336,52	334,96	334,96

Dari hasil pada Tabel 4.5 pada perbandingan $pm2opt = 0, 0{,}2, 0{,}4$ terlihat bahwa mutasi ini memberikan hasil yang sangat signifikan bahkan ketika $pm2opt = 0{,}8$ memberikan nilai yang minimal. Maka parameter yang dipilih adalah $pm2opt = 0{,}8$.

4.2.6 Pemilihan Parameter Bedasarkan Pengujian

Bedasarkan pengujian diatas didapatkan parameter-parameter sebagai berikut

1. Parameter pembangkitan generasi awal algoritma genetika untuk VRP
    a. $pgc = 0{,}8$
    b. $pgr = 0{,}2$
2. Parameter kawin silang algoritma genetika untuk VRP
    a. $pcsbx = 0{,}4$
    b. $pcrbx = 0{,}6$
3. Parameter mutasi algoritma genetika untuk VRP
    a. $pm = 0{,}1$
    b. $pm2opt = 0{,}8$

Pada pengujian selanjutnya yaitu perbandingan algoritma dan implementasi pada perusahaan akan digunakan nilai-nilai di atas dan digunakan individu *npop* sebanyak 400 dan iterasi sebanyak 50.

4.2.3 Perbandingan Algoritma

Perbandingan Algoritma dalam hal ini akan dibandingkan dengan algoritma ACO [1] dan TS [8]. Dari data-data penelitian [1] penulis mengambil 5 data dari semua data diantaranya data 2,4,6,16 dan 17. Berikut hasil dari menjalankan program dengan dimasukan setiap data masing-masing sepuluh kali menggunakan parameter yang telah diuji dan akan disajikan pada Tabel 4.6 dan Tabel 4.7.

Tabel 4.6 Hasil *Running* dari Program dalam Satuan *Unit*

No.	Data yang digunakan				
	2	4	6	16	17
1	334,96	440,68	498,32	698,92	870,33
2	334,96	440,68	498,32	698,92	870,33
3	334,96	440,68	504,39	698,92	874,24
4	334,96	440,68	517,21	698,92	870,33
5	334,96	440,68	498,32	698,92	876,21
6	334,96	440,68	504,39	698,92	874,24
7	334,96	440,68	498,32	698,92	870,33
8	334,96	440,68	498,32	698,92	876,21
9	334,96	440,68	498,32	698,92	874,24
10	334,96	440,68	498,32	698,92	870,33
best	334,96	440,68	498,32	698,92	870,33
worst	334,96	440,68	517,21	698,92	876,21
mean	334,96	440,68	501,423	698,92	872,679
stdev	0	0	6,094039	0	2,578438

Tabel 4.7 Perbandingan Algoritma Genetika dengan ACO dan TS dalam Satuan *Unit*

	Data yang digunakan				
	2	4	6	16	17
GA *mean*	334,96	440,68	501,42	698,92	872,68
ACO *mean*	334,96	440,68	501,47	698,92	870,33
TS *mean*	350,58	448,48	504,46	707,85	920,87

Dari kelima data yang ditampilkan pada Tabel 4.6 dan Tabel 4.7 secara umum algoritma genetika memiliki hasil yang hampir sama dengan ACO dan lebih baik daripada *tabu search*. Dari semua data tersebut ACO memiliki nilai lebih baik yaitu 569,27 *unit* dan algoritma genetika terpaut 0,08% dari ACO yaitu sebesar 569,73 *unit* kemudan algoritma genetika terpaut 2,93% dari *tabu search* yang memiliki rata-rata hasil 586,45 *unit*.

4.2.4 Hasil Implementasi pada suatu Perusahaan

Setelah didapat jaringan data jalan kota Surabaya. Maka di masukkan pula pesanan-pesanan pelanggan didapat hasil-hasil selama 6 hari. Setiap data perhari program dijalankan 8 kali ditampilkan pada Tabel 4.8 dan Tabel 4.9.

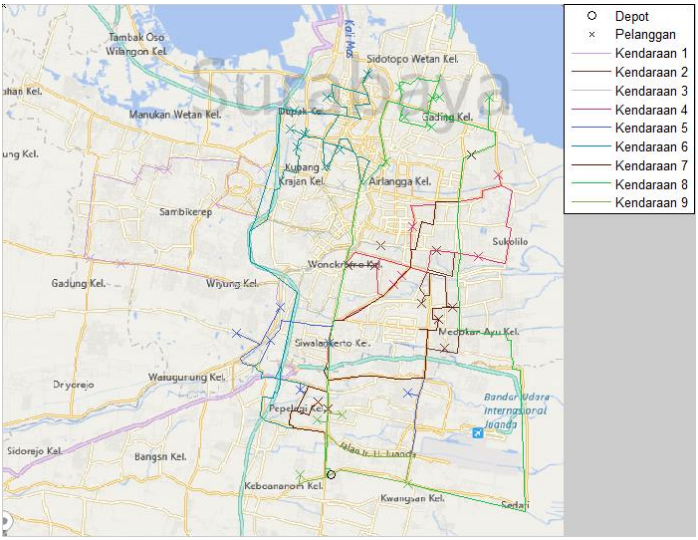
Tabel 4.8 Data Implementasi yang Digunakan

	Data yang digunakan					
	Hari 1	Hari 2	Hari 3	Hari 4	Hari 5	Hari 6
Banyak pelanggan	58	63	81	84	98	99
Banyak kendaraan	9	10	15	20	25	27
Banyak barang	2090	2466	3300	5188	5871	6609

Tabel 4.9 Hasil Implementasi Berupa Total Jarak Terpendek (km)

No.	Data yang digunakan					
	Hari 1	Hari 2	Hari 3	Hari 4	Hari 5	Hari 6
1	458,28	501,22	716,71	901,69	1133,4	1265,8
2	461,15	496,09	693,13	903,8	1144,3	1298,8
3	450,8	486,76	717,03	895,68	1132,5	1255,5
4	456,8	499,83	720,58	901,54	1157,4	1262,9
5	452,09	497,62	710,15	892,74	1145	1286,5
6	453,2	485,81	681,91	897,64	1138,6	1286,2
7	456,41	475,32	690,20	894,34	1124,9	1298,8
8	464,45	499,43	720,93	913,46	1151,7	1296
mean	456,65	492,76	706,33	900,11	1141	1281,3
stdev	4,6366	9,1728	15,511	6,6529	10,728	17,403
best	450,8	475,32	681,91	892,74	1124,9	1255,5
worst	464,45	501,22	720,93	913,46	1157,4	1298,8

Berikut pada Gambar 4.2 ditampilkan rute pada setiap kendaraanya dan pada Tabel 4.10 ditampilkan urutan pelanggan yang dilayani setiap kendaraanya.



Gambar 4.2 Hasil *Routing* Data Pertama Hari 1

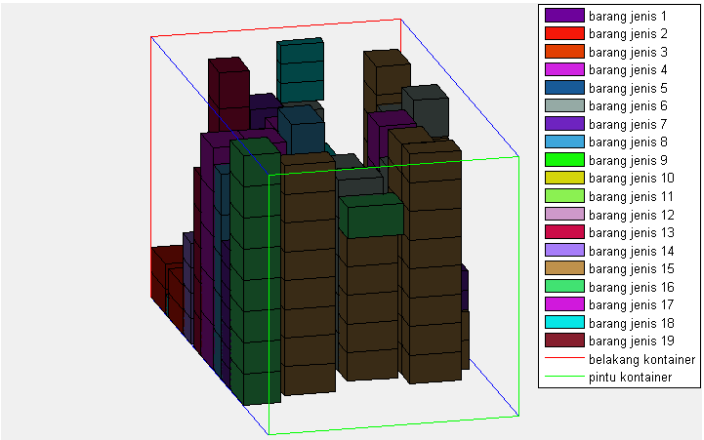
Tabel 4.10 Hasil *Scheduling* Setiap Kendaraan

	Urutan Pelanggan								
Kendaraan 1	92	118	120	110	102	117	98		
Kendaraan 2	205	60	184	61	190	71	196		
Kendaraan 3	33	115	75	116	76	82	84	87	195
Kendaraan 4	70	182	175	187	64				
Kendaraan 5	22	111	42	46	29	51			
Kendaraan 6	126	135	132	124	121	91	147	96	
Kendaraan 7	17	57	172	16					
Kendaraan 8	171	168	160	142	137	167	173	152	
Kendaraan 9	5	13	19	8					

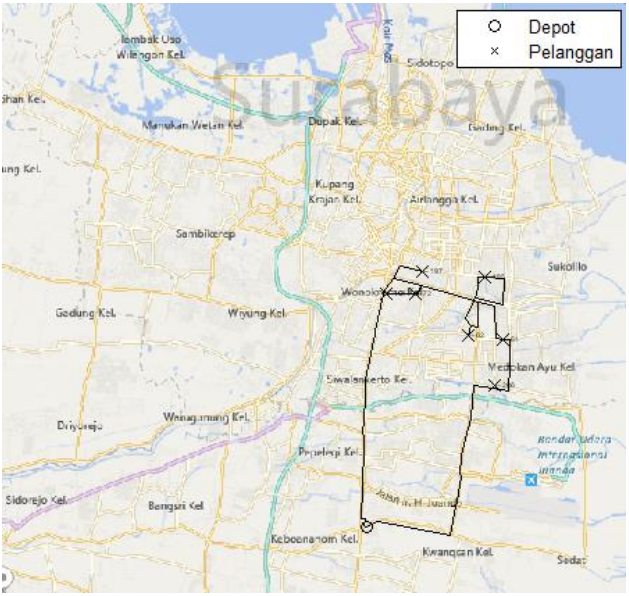
Dari Gambar 4.2 dan Tabel 4.10 didapatkan hasil rute setiap kendaraan dengan rentang *node* yang dilayani setiap kendaraan adalah 4

pelanggan sampai 8 pelanggan. Berikut akan ditampilkan detail rute dan penataan barang setiap kendaraan pada Gambar 4.3 sampai Gambar 4.20.

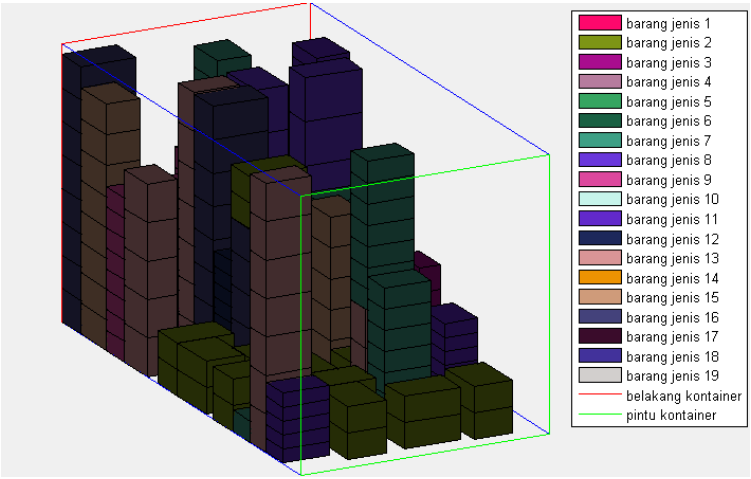
Gambar 4.3 *Tour* Kendaraan 1 Data Hari 1



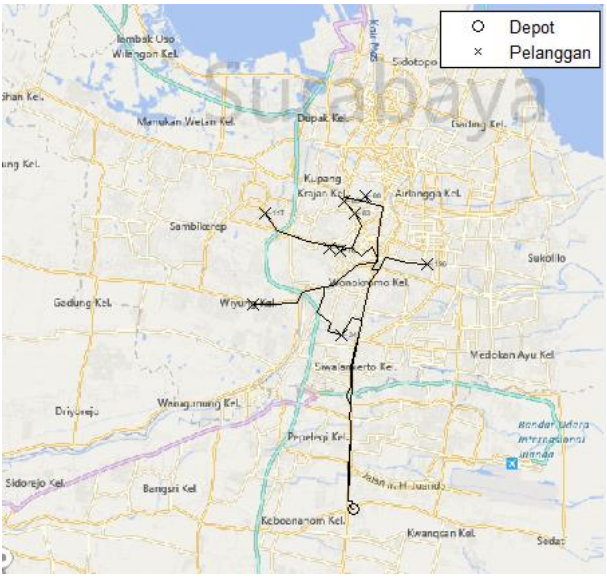
Gambar 4.4 Penataan Barang-barang pada Kontainer Kendaraan 1 Data Hari 1



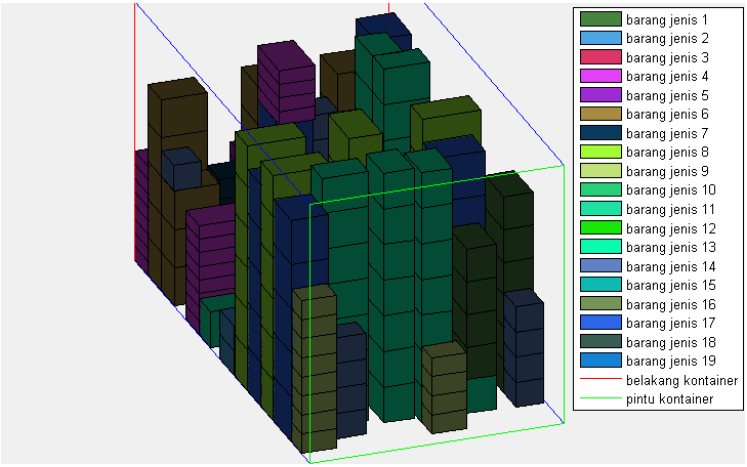
Gambar 4.5 *Tour* Kendaraan 2 Data Hari 1



Gambar 4.6 Penataan Barang-barang pada Kontainer Kendaraan 2 Data Hari 1

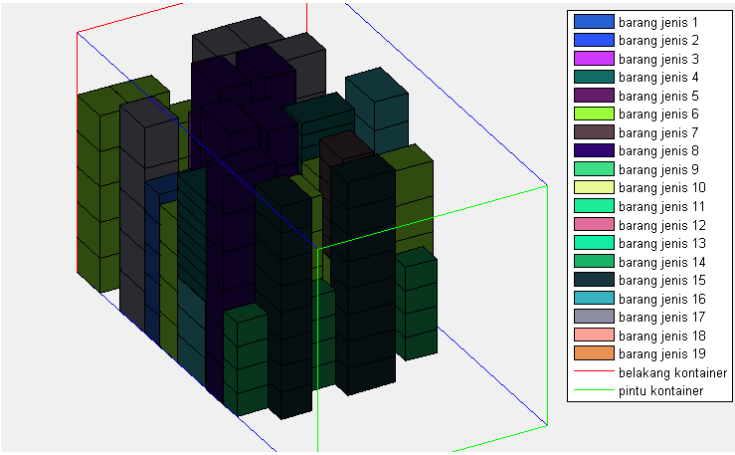


Gambar 4.7 *Tour* Kendaraan 3 Data Hari 1

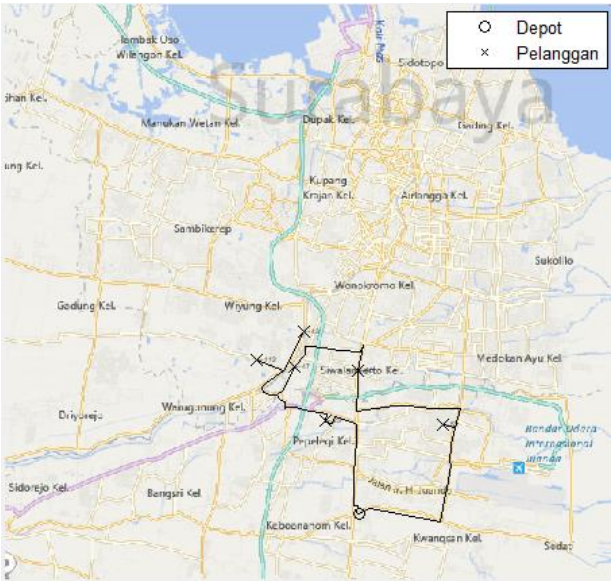


Gambar 4.8 Penataan Barang-barang pada Kontainer Kendaraan 3 Data Hari 1

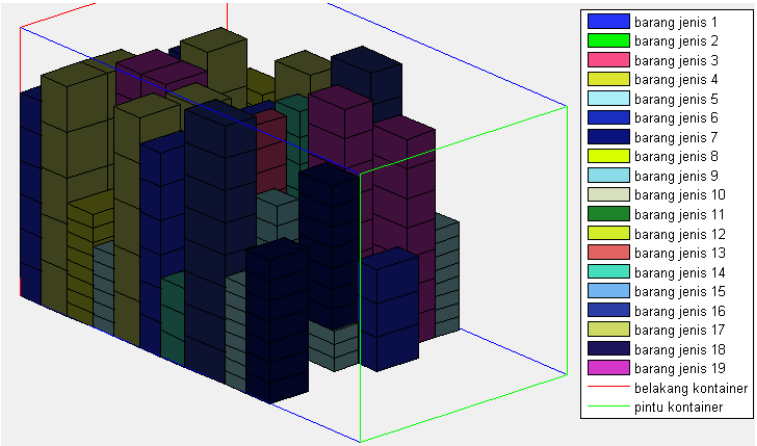
Gambar 4.9 *Tour* Kendaraan 4 Data Hari 1



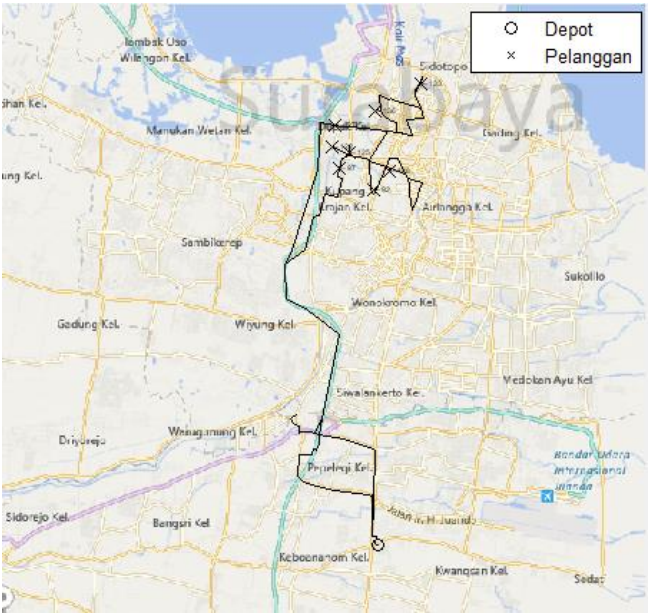
Gambar 4.10 Penataan Barang-barang pada Kontainer Kendaraan 4 Data Hari 1



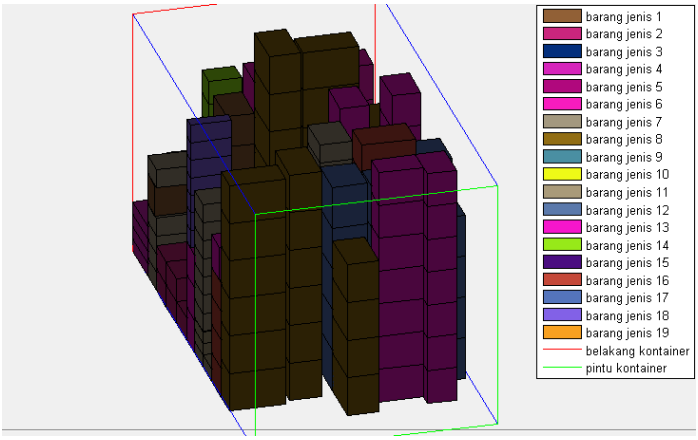
Gambar 4.11 *Tour* Kendaraan 1 Data Hari 1



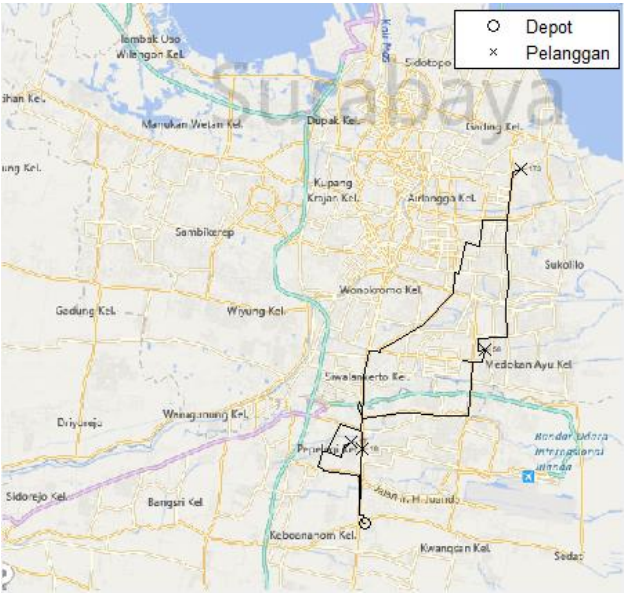
Gambar 4.12 Penataan Barang-barang pada Kontainer Kendaraan 5 Data Hari 1



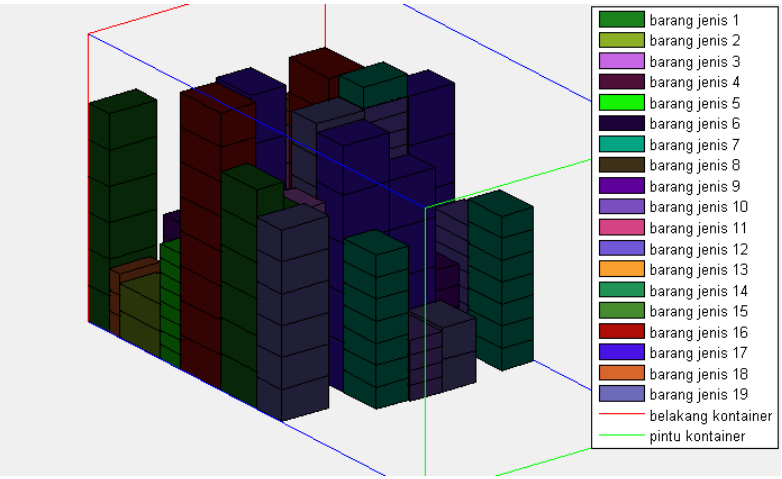
Gambar 4.13 *Tour* Kendaraan 6 Data Hari 1



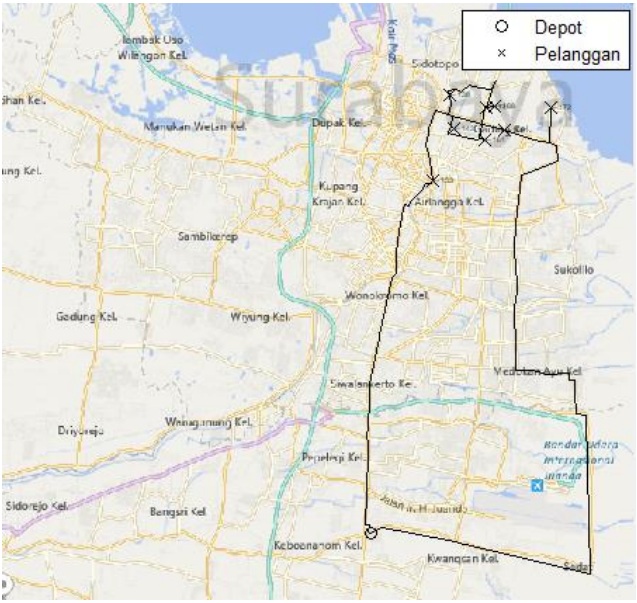
Gambar 4.14 Penataan Barang-barang pada Kontainer Kendaraan 6 Data Hari 1



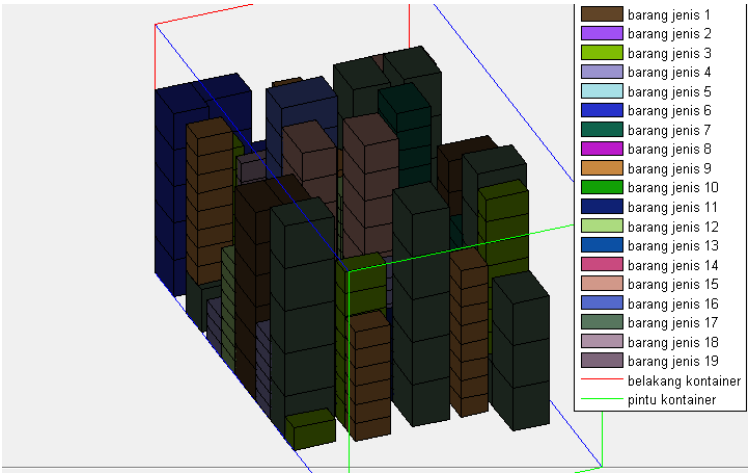
Gambar 4.15 *Tour* Kendaraan 7 Data Hari 1



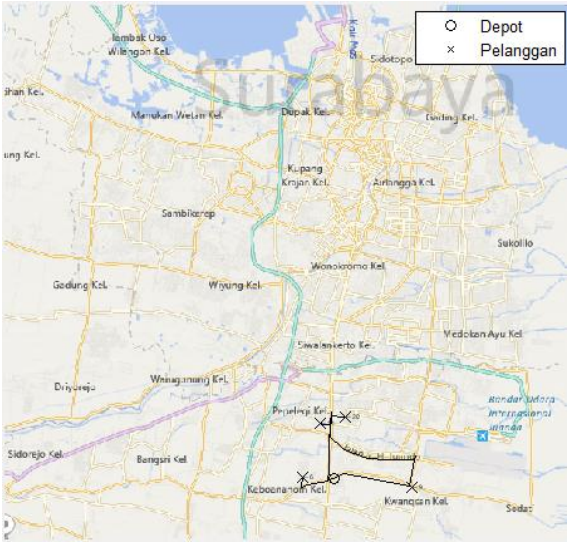
Gambar 4.16 Penataan Barang-barang pada Kontainer Kendaraan 7 Data Hari 1



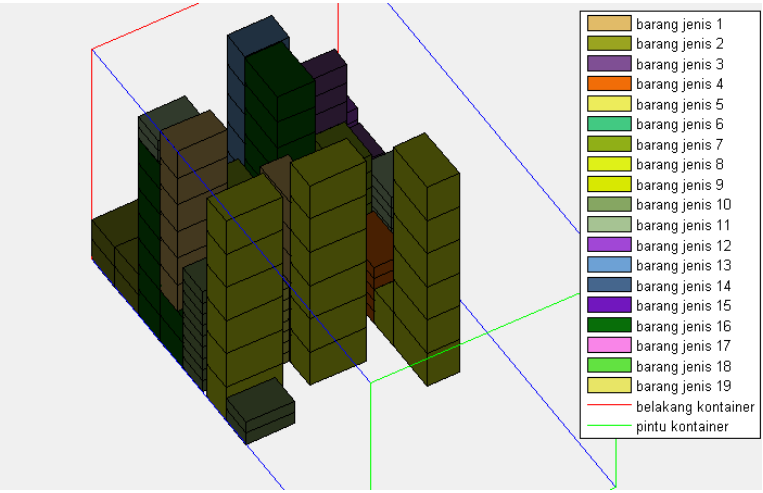
Gambar 4.17 *Tour* Kendaraan 8 Data Hari 1



Gambar 4.18 Penataan Barang-barang pada Kontainer Kendaraan 8 Data Hari 1



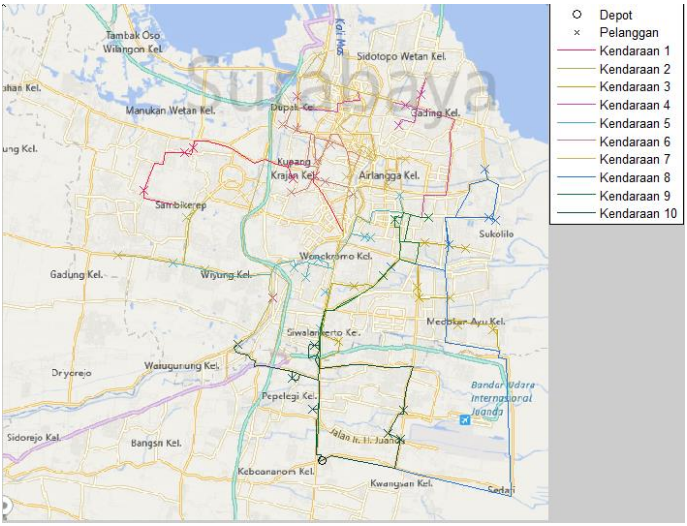
Gambar 4.19 *Tour* Kendaraan 9 Data Hari 1



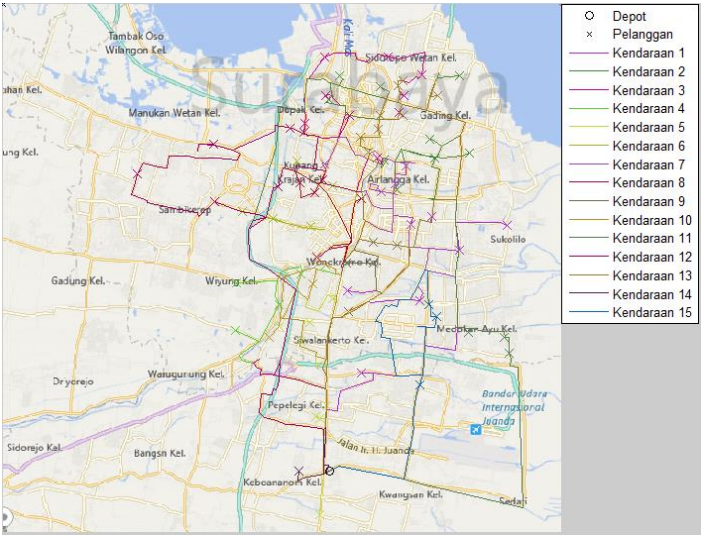
Gambar 4.20 Penataan Barang-barang pada Kontainer Kendaraan 9 Data Hari 1

Dari Gambar 4.3 sampai Gambar 4.20 didapatkan hasil rute yang cenderung mengelompok pada setiap kendaraannya hal ini disebabkan adanya pembangkitan dengan metode *cluster generation.* Sedangkan karena algortima *2-opt* hasil rute cenderung berbebentuk konveks. Adapun hasil penataan pada Gambar 4.3 dan Gambar 4.20 merupakan penataan setiap barang pada masing-masing kontainer kendaraan sebagai bukti bahwa barang-barang yang dikirim dapat ditata pada kontainer kendaraan tersebut.

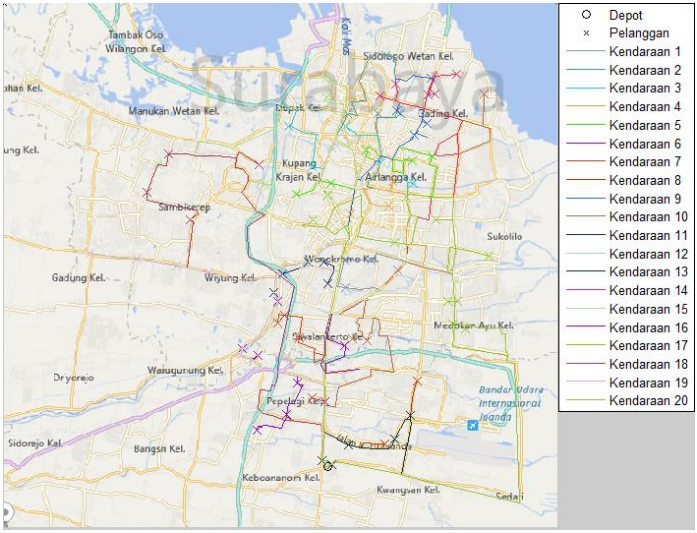
Untuk data pemesanan yang lain yaitu pemesanan hari 2 hingga hari 6 pada *running* pertama berikut dilampirkan pada Gambar 4.21 sampai Gambar 4.25.



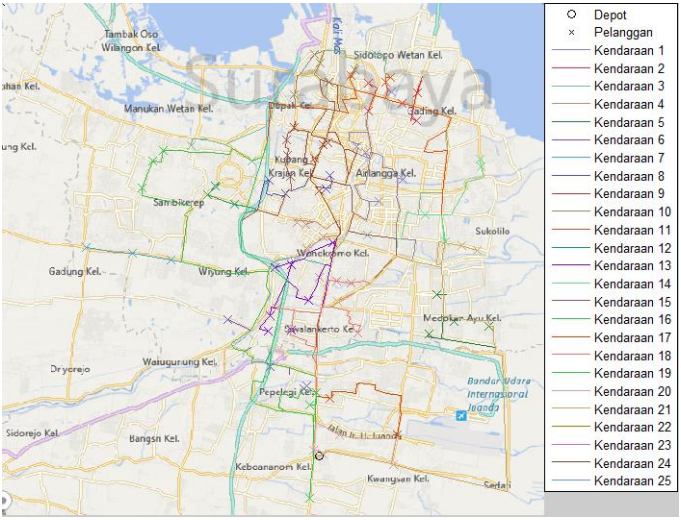
Gambar 4.21 Hasil *Routing* Data Pertama Hari 2



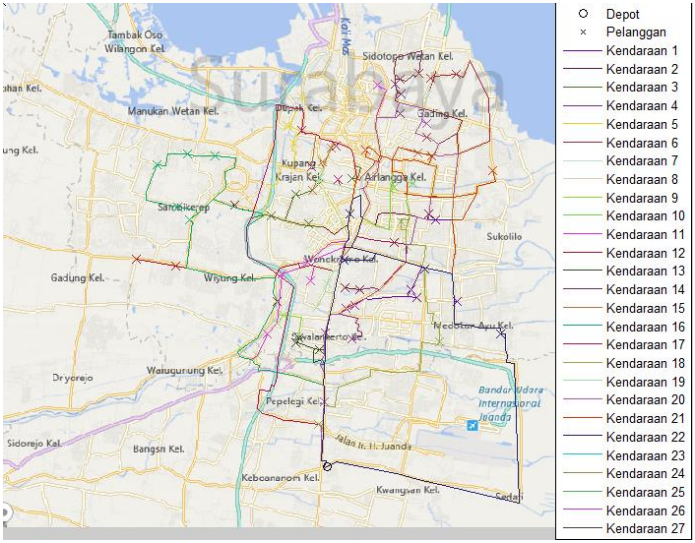
Gambar 4.22 Hasil *Routing* Data Pertama Hari 3



Gambar 4.23 Hasil *Routing* Data Pertama Hari 4



Gambar 4.24 Hasil *Routing* Data Pertama Hari 5



Gambar 4.25 Hasil *Routing* Data Pertama Hari 6

Pada Gambar 4.21 sampai Gambar 4.25 didapat hasil sama seperti pada data pertama hari 1 yaitu setiap kendaraan membentuk rute pada pelanggan-pelanggan yang cenderung mengelompok dengan bentuk yang cenderung konveks. Adanya *cluster generation* dan algoritma *2-opt* yang memiliki pengaruh signifikan memberikan hasil yang mengelompok dan konveks.

BAB 5
PENUTUP

Hasil dari perancangan dan penelitian Tugas Akhir dirangkum dan dirumuskan kesimpulan. Kesimpulan ini menerangkan hasil dari pengujian yang telah dilaksanakan

Selama proses perancangan dan pengujian, terdapat banyak kendala yang hadapi. Kendala tersebut telah dirangkum dan dirumuskan dalam bentuk saran untuk penyempurnaan lebih lanjut

5.1 Kesimpulan

Bedasarkan hasil pengujian dapat diperoleh kesimpulan sebagai berikut

1. Parameter-parameter yang mempengaruhi kualitas solusi terbaik dari algoritma genetika antara lain parameter banyak populasi, parameter pembangkitan generasi awal, parameter kawin silang, parameter mutasi. Parameter yang menghasilkan nilai terbaik secara signifikan adalah mutasi dengan algoritma *2-opt* dan *cluster generation*
2. Algoritma genetika menghasilkan total jarak terpendek 0,08% lebih buruk dibandingkan ACO dan 2,93% lebih baik dari TS.
3. Pada implementasi di Perusahaan retail dengan data jalan kota Surabaya algoritma genetika terlihat dapat menghasilkan rute beserta penataan barang yang diangkut.

5.2 Saran

Dari hasil penelitian yang dilakukan, untuk pengembangan berikutnya, disarankan beberapa hal berikut

1. Pemodelan keseimbangan yang melibatkan kendala tinggi, berat barang dan sandaran barang
2. Perlu adanya bantuan dari penelitian tentang pengambilan data jalan yang lebih detail.
3. Penelitian kedepan diharapkan dapat merancang dalam bentuk *software* yang terpadu
4. Penggunaan algoritma GA untuk VRP mudah untuk dikembangkan namun sangat susah untuk mendapat logika heuristik yang lebih diterima.

DAFTAR PUSTAKA

[1] G. Fuellerer, K. F. Doerner, R. F. Hartl and M. Iori, "Metaheuristic for vehicle routing problems with three-dimensional loading constraints," *European Journal of Operational Research,* pp. 751-759, 2010.

[2] R. K. Ahuja, T. L. Magnanti and J. B. Orlin, *Network Flows*, Pearson, 1993.

[3] G. Laporte, "The Vehicle Routing Problem : An overview of exact and approximate algorithms," *European Journal of Operational Research,* pp. 345-358, 1992.

[4] M. Hifi, I. Kacem, S. Negre and L. Wu, "A Linear Programming Approach for the Three-Dimensional Bin Packing Problem," *Electronic Notes in Discrete Mathematics,* pp. 993-1000, 2010.

[5] B. S. Baker, E. Coffman, JR and R. L. Rivest, "Orthogonal Packings in Two Dimensions," *SIAM Journal on Computing,* pp. 846-855, 1980.

[6] M. Gendreau, M. Iori, G. Laporte and S. Martello, "A Tabu Search heuristic for the vehicle routing problem with two dimensional loading constraints," *Networks,* p. Forthcoming, 2006.

[7] A. Turkay and E. Emel, "Vehicle Routing Problem with Packing Contraints," in *5th Euro/Informs Joint International Meeting*, Istanbul, 2003.

[8] M. Gendreau, M. Iori, G. Laporte and S. Martello, "A Tabu Search Algorithm for a Routing and Container Loading Problem," *Transportation Science,* pp. 342-350, 2006.

[9] B. Santosa and P. Willy, *Metoda Metaheuristik Konsep dan Implementasi*, Surabaya: Guna Widya, 2011.

[10] G. Vaira and O. Kurasova, "Genetic Algorithms and VRP: the Behaviour of a Crossover Operator," *Baltic Journal of Modern Computing,* 2013.

[11] D. Liu and H. Teng, "An Improved BL-algorithm for genetic algorithm of the orthogonal packing of rectangles," *European Journal of Operational Research,* pp. 413-420, 1999.

[12] J.-P. Chiou and C.-F. Chang, "A Novel Evolutionary Algorithm for Capacitor Placement in Distribution Systems," *GSTF Journal Engineering Technology,* pp. 9-13, 2013.

[13] S. Jakobs, "On genetic algorithms for the packing of polygons," *European Journal of Operational Research,* pp. 165-181, 1996.

[14] S. Martello, D. Pisinger and D. Vigo, "The Three-Dimensional Bin Packing Problem," *Operations Research,* pp. 256-267, 2000.

[15] M. G. Omran, "CODEQ: an effective metaheuristic for continuous global optimisation," *Int. J. Metaheuristic,* pp. 108-131, 2010.

[16] M. G. Omran and A. Salman, "Constrained optimization using CODEQ," *Chaos, Solitons, and Fractals,* pp. 662-668, 2009.

[17] Q. Ruan, Z. Zhang, L. Miao and H. Shen, "A hybrid approach for the vehicle routing problem with three-dimensional loading constraints," *Computers & Operations Research,* pp. 1579-1589, 2013.

[18] R. Storn and K. Price, "Differential Evolution - A Simple and Efficient Heuristic for Global Optimization over Continuous Space," *Journal of Global Optimization,* pp. 341-359, 1997.

RIWAYAT HIDUP

Moh. Yasya Bahrul Ulum lahir pada 12 Agustus 1994 di Kediri. Pada masa kecilnya penulis tinggal di kabupaten Kediri dan memulai pendidikannya di SDI Al-Huda Kota Kediri. Kemudian melanjutkan di SMPN 1 kota Kediri pada tahun 2007 dan mendapatkan beasiswa untuk bersekolah di SBBS Senior High School hingga lulus pada 2013. Sampai penulisan buku ini penulis melanjutkan studi di Institut teknologi Sepuluh Nopember, Surabaya.

Riwayat hidup penulis banyak melakoni hobinya dalam Olimpiade Matematika. Sejak SD sering sekali menjuarai lomba matematika tingkat kabupaten atau kota. Kemudian SMP mendapatkan medali perunggu pada OSN Matematika jenjang SMP di Jakarta pada tahun 2009. Dan melanjutkannya lagi pada SMA mendapat medali emas pada kompetisi yang sama jenjang SMA. Setelah mengikuti pelatihan nasional hingga tahap 3 penulis gagal melangkah menjadi wakil Indonesia di IMO. Namun, ketika memasuki dunia perkuliahan penulis berhasil mendapatakan medali emas IMC 2013 di Bulgaria sebagai perwakilan dari Indonesia.

Sampai buku ini di tulis penulis masih menekuni dunia pendidikan terutama olimpiade. Penulis aktif dalam pengajaran olimpiade terutama bidang SMA. Penulis berharap adanya perubahan pada pelajar Indonesia dalam belajar. Sehingga harus ada pacuan semangat dalam belajar. Pacuan tersebut akan membentuk ide-ide yang dapat mengubah kondisi Indonesia.