OPTIMASI PROSES MUAT BARANG DALAM KONTAINER STUDI KASUS PT ANTESS MENGGUNAKAN ALGORITMA *TABU SEARCH*

SKRIPSI

Oleh:
RIZQY KURNIA RAHMAN
NIM. 09650025

JURUSAN TEKNIK INFORMATIKA
FAKULTAS SAINS DAN TEKNOLOGI
UNIVERSITAS ISLAM NEGERI MAULANA MALIK IBRAHIM
MALANG
2014

OPTIMASI PROSES MUAT BARANG DALAM KONTAINER STUDI KASUS PT ANTESS MENGGUNAKAN ALGORITMA *TABU SEARCH*

SKRIPSI

Diajukan kepada :

Fakultas Sains dan Teknologi

Universitas Islam Negeri Maulana Malik Ibrahim Malang

Untuk Memenuhi Salah Satu Persyaratan Dalam

Memperoleh Gelar Sarjana Komputer (S.Kom)

Oleh:

RIZQY KURNIA RAHMAN

NIM. 09650025

JURUSAN TEKNIK INFORMATIKA

FAKULTAS SAINS DAN TEKNOLOGI

UNIVERSITAS ISLAM NEGERI MAULANA MALIK IBRAHIM MALANG

2014

HALAMAN PERSETUJUAN

OPTIMASI PROSES MUAT BARANG DALAM KONTAINER STUDI KASUS PT ANTESS MENGGUNAKAN ALGORITMA *TABU SEARCH*

SKRIPSI

Oleh:

Nama	:	Rizqy Kurnia Rahman
NIM	:	09650025
Jurusan	:	Teknik Informatika
Fakultas	:	Sains dan Teknologi

Telah Disetujui, 11 September 2014

Dosen Pembimbing I	Dosen Pembimbing II
Fachrul Kurniawan, M. MT	**Fresy Nugroho, M.T**
NIP. 19771020 200912 1 001	**NIP. 19710722 201101 1 001**

Mengetahui,

Ketua Jurusan Teknik Informatika

Dr. Cahyo Crysdian

NIP. 19740424 200901 1 008

HALAMAN PENGESAHAN

OPTIMASI PROSES MUAT BARANG DALAM KONTAINER STUDI KASUS PT ANTESS MENGGUNAKAN ALGORITMA *TABU SEARCH*

SKRIPSI

Oleh:
RIZQY KURNIA RAHMAN
NIM : 09650025

Telah Dipertahankan Di Depan Dewan Penguji Skripsi
Dan Dinyatakan Diterima Sebagai Salah Satu Persyaratan
Untuk Memperoleh Gelar Sarjana Komputer (S.Kom)

Tanggal, 11 September 2014

Susunan Dewan Penguji			Tanda Tangan
1. Penguji Utama	:	Yunifa Miftachul Arif, M.T NIP. 19830616 201101 1 004	( )
2. Ketua	:	Dr. Suhartono, M.Kom NIP. 19680519 200312 1 001	( )
3. Sekretaris	:	Fachrul Kurniawan, M.MT NIP. 19771020 200901 1 001	( )
4. Anggota	:	Fresy Nugroho, M.MT NIP. 19710722 201101 1 001	( )

Mengetahui,
Ketua Jurusan Teknik Informatika

Dr. Cahyo Crysdian
NIP. 19740424 200901 1 008

HALAMAN PERSEMBAHAN

Ayahku Wratsongko Arif Pinudji dan Ibuku Ismunarti
yang beristiqomah
melimpahkan segala tenaga dan doa kepadaku
agar selalu berada dalam Jalan-Nya dan selalu mendapatkan
perhatian-Nya

Adikku Luthfie Surya Rahman
yang selalu melimpahkan pikiran dan perasaannya kepadaku
agar beristiqomah menjadi teladan yang baik

HALAMAN PERNYATAAN
ORISINALITAS PENELITIAN

Saya yang bertanda tangan di bawah ini :

Nama Lengkap	:	RIZQY KURNIA RAHMAN
NIM	:	09650025
Fakultas/Jurusan	:	SAINS DAN TEKNOLOGI / TEKNIK INFORMATIKA
Judul Skripsi	:	OPTIMASI PROSES MUAT BARANG DALAM KONTAINER STUDI KASUS PT ANTESS MENGGUNAKAN ALGORITMA *TABU SEARCH*

Dengan ini menyatakan bahwa:

1. Isi dari skripsi yang saya buat ini adalah benar-benar karya saya sendiri dan tidak terdapat unsur-unsur penjiplakan karya orang lain, selain nama-nama termaktub di isi dan tertulis di daftar pustaka dalam skripsi ini.
2. Apabila di kemudian hari ternyata skripsi yang saya tulis terbukti hasil jiplakan, maka saya bersedia untuk mempertanggung jawabkan, dan menanggung segala resiko, serta diproses sesuai peraturan yang berlaku.

Demikian pernyataan ini saya buat dengan penuh kesadaran.

Malang, 02 September 2014
Yang Membuat Pernyataan

Rizqy Kurnia Rahman
NIM 09650025

KATA PENGANTAR

Assalamualaikum Warrahmatullohi Wabarokatuh

Segala puji syukur kepada Allah SWT yang telah memberikan kesehatan, kekuatan, kebahagiaan dan segala nikmat-Nya kepada penulis, serta sholawat selalu tercurahkan kepada Rasulullah Muhammad SAW sehingga peneliti dapat menyelesaikan studi di Jurusan Teknik Informatika Fakultas Sains dan Teknologi Universitas Islam Negeri Maulana Malik Ibrahim Malang, sekaligus dapat menyelesaikan skripsi dengan judul "*Optimasi Proses Muat Barang Dalam Kontainer Studi Kasus PT ANTESS Menggunakan Algoritma Tabu Search*".

Tanpa adanya keterlibatan dari berbagai pihak yang telah memberikan ilmu, pengetahuan dan bantuan baik moril maupun materiil, penulis menyadari akan sulit menyelesaikan skripsi ini. Oleh karena itu, penulis sampaikan ucapan terima kasih dan doa kepada :

1. Prof. Dr. H. Mudjia Rahardjo, selaku rektor Universitas Islam Negeri Maulana Malik Ibrahim Malang.
2. Dr. drh. Hj Bayyinatul Muchtaromah, M.Si selaku Dekan Fakultas Sains dan Teknologi Universitas Islam Negeri Maulana Malik Ibrahim Malang.
3. Dr. Cahyo Crysdian selaku Ketua Jurusan Teknik Informatika Universitas Islam Negeri Maulana Malik Ibrahim Malang.
4. H. Fatchurrochman, M.Kom selaku dosen wali yang telah memberikan bimbingan dan pengetahuannya kepada penulis dalam menempuh jenjang pendidikan ini.
5. Fachrul Kurniawan, M.MT selaku dosen pembimbing skripsi yang telah memberikan arahan, bimbingan, ilmu dan pengetahuannya bagi penulis dalam proses penyelesaian skripsi ini.

6. Fresy Nugroho, M.T selaku dosen pembimbing skripsi integrasi Sains dan Islam yang telah memberikan motivasi dan bimbingannya kepada penulis dalam proses penyelesaian skripsi ini.
7. Seluruh Dosen Universitas Islam Negeri Maliki Malang, khususnya Dosen dan seluruh Staf Jurusan Teknik Informatika yang telah memberikan ilmu dan bantuannya kepada penulis dalam proses penyelesaian skripsi ini.
8. Seluruh pengasuh di Ma'had Sunan Ampel Al-Aly Universitas Islam Negeri Maliki Malang yang telah memberikan doa, arahan dan bimbingannya kepada penulis.
9. PT ANTESS beserta karyawannya yang telah memberikan bantuan kepada penulis dalam proses penyelesaian skripsi ini.
10. Bapakku Wratsongko Arif Pinudji, ibuku Ismunarti, dan adikku Luthfie Surya Rahman serta keluarga besar penulis yang selalu melimpahkan doa, dorongan, dan motivasi bagi penulis.
11. Sahabatku di rumah kontrakan Akmal Afif, Ahmad Fuad H, Ikhwan Baidlowi S, Miftah Farid A, M Khoirur Roziqin, M Hasan, M Faisol, dan Nizar Zakaria yang selalu memberikan semangat dan motivasi kepada penulis.
12. Dan kepada semua pihak yang telah memberikan ilmu dan pengetahuannya kepada penulis.

Penulis menyadari keterbatasan ilmu dan pengetahuan yang dimiliki dalam proses penyelesaian skripsi ini. Penulis berdoa skripsi ini dapat bermanfaat dan menambah ilmu dan pengetahuan bagi kita semua.

Wassalamualaikum Warrohmatullohi Wabarakotuh

Malang, 01 September 2014

Penulis,

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

ABSTRAK

Rahman, Rizqy Kurnia. 2014. 09650025. **Optimasi Proses Muat Barang Dalam Kontainer Studi Kasus PT ANTESS Menggunakan Algoritma *Tabu Search***. Jurusan Teknik Informatika Fakultas Sains dan Teknologi Universitas Islam Negeri Maulana Malik Ibrahim Malang. Pembimbing (I) Fachrul Kurniawan, M. MT, (II) Fresy Nugroho, M.T

Kata Kunci : perusahaan logistik, optimasi, muat barang, kontainer, *tabu search*

Permasalahan proses muat barang dalam kontainer yang tidak optimal membuat biaya pengeluaran sebuah perusahaan khususnya perusahaan logistik bertambah. Permasalahan proses muat barang bisa disebabkan karena belum optimalnya penggunaan ruang kosong dan penataan barang dalam kontainer. Selama ini pada PT ANTESS dalam menangani proses muat barang belum ada bantuan dari sebuah aplikasi dari komputer.

PT ANTESS (Antaran Express) merupakan perusahaan swasta yang bergerak di bidang jasa transportasi logistik. Dengan memiliki misi memberikan pelayanan dengan nilai tambah serta memberikan pendapatan yang sesuai kepada perusahaan, maka PT ANTESS membutuhkan sebuah optimasi untuk menangani proses muat barang dalam kontainer agar tercapai salah satu misi tersebut. Optimasi adalah kegiatan untuk memperoleh hasil yang terbaik sesuai dengan yang diharapkan. Optimasi proses muat barang diharapkan dapat mengurangi biaya operasional yang dikeluarkan sehingga pendapatan yang diterima sesuai dengan yang diharapkan perusahaan.

Aplikasi optimasi proses muat barang ini dibuat dengan tujuan untuk menerapkan algoritma *tabu search* dalam permasalahan proses muat barang. Aplikasi ini menggunakan data barang dan data kontainer. Data barang yang diperlukan meliputi dimensi(panjang, lebar, tinggi) barang, berat barang, jenis barang dan kota tujuan. Sementara data kontainer yang diperlukan meliputi dimensi dalam kontainer, maksimum berat kontainer, dan maksimum volume kontainer. Aplikasi yang dibuat menghasilkan *output* berupa tabel posisi barang dalam kontainer dan grafik penggambaran letak posisi barang berdasarkan dimensinya.

ABSTRACT

Rahman, Rizqy Kurnia. 2014. 09650025. **Process Optimization of Goods In Container Load Case Study PT ANTESS Using Tabu Search Algorithm**. Department of Informatics Faculty of Science and Technology of the State Islamic University of Maulana Malik Ibrahim Malang. Supervisor (I) Fachrul Kurniawan, M. MT, (II) Fresy Nugroho, MT

Keywords: logistics, optimization, unloading goods, containers, *tabu search*

The problems in unloading goods in containers which are not optimized make the expenses of companies especially logistics companies increased. And these problems can be caused by the inefficient way of using spaces and the arrangement of the goods in containers. During this time, the PT ANTESS has not used a computer application to handle the unloading process.

PT ANTESS (Express delivery) is a privately-owned company specialized in transporting logistics. Having a mission to provide services with added value and provide appropriate incomes to companies, the PT ANTESS requires an optimized process to handle the unloading process in order to achieve one of these missions. The optimized process is a way to obtain the best results as expected. The optimized process in unloading goods is expected to be able to reduce the operating costs so that the received income could be the same as expected by the company.

The optimized application in unloading goods is created with the aim to apply the *tabu search* algorithm in the problem of unloading goods. This application uses the data items and the data container. The needed data items are the dimensions (length, width, height) of goods, the weight of the goods, the type of the goods and the destination. Meanwhile the needed data containers are the dimensions in the container, the maximum weight of the container, and the maximum volume of the container. This application is made to produce the outputs which are the position table of the goods in the container and the depicted graphic of the positions of the goods based on the dimensions.

ملخص البحث

رحمن، رزقي. ك. 2014. 09650025. أمثل عملية التفريغ البضائع فى الحاوية، دراسة حالية الشركة أنتيس (**PT. ANTESS**)، بتطبيق البحث التبو الخوارزمي. قسم التقنية المعلومية، كلية العلوم والتكنولوجيا، الجامعة الإسلامية مولنا مالك إبراهيم بمالانج. المشرف الأول فخر الكورنياوان الماجستير، والثانى فريس نوغراها الماجستير.

الرئيسية : الشركة اللوجستية، الأمثل، تفريغ البضائع، الحاوية، البحث التبو.

المشكلة عن عملية تفريغ البضائع فى الحاوية تؤدى الى ازداد التصدير فى الشركة خصوصا الشركة اللوجيستية. ومن سبب هذه المشكلة هو نقص الأمثل فى تطبيق المساحة الفارغة وتصميم البضائع فى الحاوية، ولم يوجد البرمج الحسوب لهذه العملية فى هذه الشركة حتى الآن.

الشركة أنتيس (**PT. ANTESS**) أو التوصيل السريع هي الشركة الأهلية حاركة فى قطاع الحدمة لانتقال اللوجيستية بمهة اعطاء الحدمة حسنا واعطاء الدخل المناسب للشركة، فلذلك يحتاج التحسين او الأمثل لتغلب عملية تفريغ البضائع فى الحاوية لكي يُبْلَغ الى الهمة. والأمثل هو العمل لنيل أفضل النتائج تناسب بالرجية. ترجو بهذه الأمثل أن ينقص تكاليف التشغيل المخروج حتى يناسب الدخل بمأمول الشركة.

جُعل البرمج لتحسين عملية تفريغ البضائع لتطبيق البحث التبو الخوارزمية فى مشكلة تفريغ البضائع. ويستحدم هذا البرمج البيانات البضائع والبيانات الحاوية. البيانات البضائع المحتاج تتكون من أبعاد (الطويل والواسع والرافع) البضائع، وثقيلها، وانواعها، والمدينة المقصودة. وأما البيانات الحاوية المحتاج هو الأبعاد فى الحاوية، وغاية ثقيلتها، ووغاية حجمه. وينتاج البرمج المصنوع القائمة عن موقف البضائع فى الحاوية والرسوم البيانية عن تصوير موقف البضائع بحيث أبعادها.

BAB I

PENDAHULUAN

1.1 Latar Belakang

Permasalahan proses muat barang dalam kontainer yang tidak optimal membuat biaya pengeluaran sebuah perusahaan khususnya perusahaan logistik bertambah. Permasalahan proses muat barang bisa disebabkan karena belum optimalnya penataan barang dalam kontainer. Penataan yang tidak memanfaatkan ruang kosong dengan baik serta penataan barang yang tidak berdasarkan jenis barang dan tujuan barang bisa menjadi penyebabnya. Dengan optimalnya proses penataan barang dalam kontainer, diharapkan barang yang diangkut bisa maksimal dan meminimumkan jumlah kontainer yang digunakan untuk mengangkut barang.

PT ANTESS merupakan perusahaan swasta nasional yang bergerak di bidang jasa transportasi logistik. Kantor utama dari PT ANTESS bertempat di Graha Sarana-Puri Surya Jaya, J01-20 Cluster Vancouver, Gedangan Sidoarjo. Tidak hanya bergerak dalam bidang jasa transportasi logistik, PT ANTESS juga difokuskan bergerak dalam bidang *cargo management*, jasa pergudangan dan distribusi. PT ANTESS yang merupakan salah satu anak perusahaan Sarana Group, selama ini perhitungan proses muat barang yang dilakukan belum terkomputasi, proses perhitungan masih dilakukan manual (dengan menghitung di kertas). Barang yang dikirim semuanya dikumpulkan

dan dikelompokkan berdasarkan jenis dan tujuan yang sama atau tujuan yang berdekatan. Setelah barang dikumpulkan, proses muat barang dilakukan. Barang yang pertama masuk dalam kontainer adalah barang yang memiliki beban yang paling berat untuk ditaruh di dasar lantai kontainer dan penurunan barang dilakukan di akhir. Barang tidak bisa dibolak-balik. Barang yang paling dekat dengan pintu kontainer adalah barang yang mempunyai kota tujuan paling dekat dengan kota keberangkatan.

Di dalam kontainer barang-barang dipisahkan berdasarkan tujuan. Jika terdapat ruang kosong, petugas akan menambahkan barang dengan syarat memiliki tujuan yang sama atau berdekatan dan barang yang dimasukkan memiliki volume yang sesuai dengan ruang kosong yang ada. Jika terdapat barang lebih yang seharusnya barang itu masuk di kontainer dan itu hanya beberapa barang, daripada petugas menggunakan kontainer lain untuk mengangkut barang tersebut, petugas akan mencari cara agar barang itu masuk tetapi jika tidak barang-barang yang tadi sudah tertata di dalam kontainer diturunkan lagi dan ditata ulang. Dari permasalahan ini perusahaan membutuhkan sebuah optimasi untuk membantu mencari alternatif metode untuk menyelesaikan proses muat barang di kontainer.

Optimasi adalah proses pencarian nilai atau hasil yang optimal. Pencarian nilai atau hasil yang optimal bisa berguna di berbagai bidang. Kegunaan dari optimasi selain untuk mendapatkan hasil yang optimal juga berguna untuk melakukan usaha secara efektif dan efisien.

Dalam berusaha ataupun berniaga pelaku harus berlaku jujur dan amanah. Hal ini sesuai dengan Firman Alloh SWT dalam Al-Qur'an Surat Al-Isra' ayat 27 dan Sabda Rasulullah SAW yang diriwayatkan HR Tirmidzi:

إِنَّ الْمُبَذِّرِينَ كَانُوا إِخْوَانَ الشَّيَاطِينِ وَكَانَ الشَّيْطَانُ لِرَبِّهِ كَفُورًا [١٧:٢٧]

Artinya : "*Sesungguhnya orang-orang yang pemboros itu adalah saudara setan dan setan itu sangat ingkar kepada Tuhannya*" [Q.S Al-Isra' : 27]

عَنْ أَبِي سَعِيدٍ عَنْ النَّبِيِّ صَلَّى اللَّهُ عَلَيْهِ وَسَلَّمَ قَالَ التَّاجِرُ الصَّدُوقُ الْأَمِينُ مَعَ النَّبِيِّينَ وَالصِّدِّيقِينَ وَالشُّهَدَاءِ

(رواه الترمذي)

Artinya : *"Seorang pebisnis yang jujur lagi amanah, maka ia akan bersama para Nabi, Shiddiqin, dan Syuhada"* [HR. Tirmidzi].

Dalam penelitian ini, peneliti menggunakan salah satu metode algoritma *heuristic* yaitu *tabu search*. Mengingat proses muat barang merupakan problem berklasifikasi *NP-hard problem*. Dimana pengembangan model *heuristic* menjadi prioritas dibandingkan dengan model *exact*. Konsep dasar dari *tabu search* adalah pengefektifan proses pencarian solusi dengan cara mencari *best solution* pada setiap tahap pelacakan. Pada beberapa tahap pelacakan dapat dikategorikan sebagai langkah tabu(dilarang).

1.2 Rumusan Masalah

Berdasarkan uraian latar belakang diatas, maka permasalahan yang akan dibahas dalam penelitian ini adalah "Bagaimana menerapkan algoritma *tabu search* untuk optimasi proses muat barang dalam kontainer studi kasus PT ANTESS?".

1.3 Batasan Masalah

Agar penelitian ini tidak menyimpang dari akar permasalahan, maka peneliti memberikan batasan masalah, yaitu :

a. Barang berbentuk *rectangular box* (balok atau kubus) dan telah dikemas di kardus.
b. Berpusat pada proses muat barang dalam kontainer tidak bongkar barang dari kontainer.
c. Kontainer yang digunakan adalah tipe kontainer 20 feet dengan ukuran volume 33.1 m^3.
d. Kategori yang digunakan dalam penentuan optimasi adalah dimensi barang, tujuan barang, dan jenis barang.
e. Barang tidak akan terjadi perubahan bentuk jika saling tumpuk.
f. Posisi barang tidak dapat diubah-ubah saat melakukan proses penataan barang dalam kontainer.
g. Tujuan barang diambil 3 kota, yaitu : Jakarta, Bandung, dan Semarang.

h. Jenis barang yang diambil hanya 3 jenis, yaitu : Kulkas, Televisi, dan Makanan.

1.4 Tujuan

Tujuan dari penelitian ini adalah menerapkan algoritma *tabu search* untuk optimasi proses muat barang dalam kontainer studi kasus PT ANTESS.

1.5 Manfaat

1.5.1 Peneliti

1. Memberikan alternatif metode untuk menyelesaikan problem proses muat barang dalam kontainer menggunakan algoritma *tabu search*

1.5.2 Perusahaan

1. Menjadi pertimbangan untuk optimasi proses muat barang dalam kontainer.
2. Memberikan alternatif metode untuk menyelesaikan proses muat barang dalam kontainer.
3. Membantu perusahaan untuk mencari alternatif metode penghitungan penentuan biaya pemuatan.

1.6 Sistematika Penulisan

BAB I : PENDAHULUAN

Bab ini memberikan informasi tentang latar belakang melakukan penelitian ini, fokus dan ruang lingkup permasalahan yang dibahas dalam melakukan penelitian, tujuan serta manfaat melakukan penelitian ini bagi peneliti maupun perusahaan yang dijadikan sebagai studi kasus, dan sistematika penulisan yang menjelaskan masing-masing bab.

BAB II : TINJAUAN PUSTAKA

Bab ini menjelaskan tentang teori-teori yang digunakan menjadi acuan dalam optimasi proses muat barang dalam kontainer studi kasus PT ANTESS menggunakan algoritma *tabu search*.

BAB III : ANALISIS DAN PERANCANGAN SISTEM

Bab ini menjelaskan tentang kebutuhan sistem dan rancangan sistem yang digunakan untuk membuat program optimasi proses muat barang dalam kontainer serta menjelaskan rancangan dan langkah-langkah penerapan algoritma *tabu search* di program optimasi proses muat barang dalam kontainer.

BAB IV : HASIL DAN PEMBAHASAN

Bab ini menjelaskan tentang rancangan sistem dan penerapan algoritma *tabu search* untuk optimasi proses muat barang dalam kontainer yang telah diimplementasikan.

BAB V : PENUTUP

Terdapat kesimpulan dari penelitian yang telah dilakukan serta beberapa saran untuk penelitian dan pengembangan pada aplikasi optimasi proses muat barang dalam kontainer.

BAB II

TINJAUAN PUSTAKA

2.1 Optimasi

Optimasi adalah proses pencarian satu atau lebih penyelesaian layak yang berhubungan dengan nilai-nilai ekstrim dari satu atau lebih nilai objektif pada suatu masalah sampai tidak terdapat solusi ekstrim yang dapat ditemukan **(**Intan & Arifin, 2010). Optimasi sangat berguna dihampir segala bidang dalam rangka melakukan usaha secara efektif dan efisien untuk mencapai target hasil yang ingin dicapai. Tentunya hal ini akan sangat sesuai dengan prinsip ekonomi yang berorientasikan untuk senantiasa menekan pengeluaran untuk menghasilkan output yang maksimal.

Teknik optimasi secara umum dapat dibagi menjadi dua bagian, yang pertama adalah *mathematical programming,* dan yang kedua adalah *combinatorial optimization.*

2.1.1 Mathematical Programming

AMPL (berasal dari *A Mathematical Programming Language*) bahasa pemrograman tingkat tinggi yang dikembangkan di Bell Laboratories, dalam rangka untuk menggambarkan dan memecahkan masalah kompleks dan teori optimasi penjadwalan. AMPL tidak menyelesaikan masalah secara langsung, dan panggilan pemecah eksternal yang sesuai (seperti CPLEX, Minos, IPOPT, SNOPT, dll), untuk mendapatkan solusi. AMPL bekerja dengan masalah optimasi linier dan

nonlinier dengan variabel diskrit atau kontinu. Satu keuntungan dari AMPL seperti catatan sintaks matematika atas masalah optimasi yang memungkinkan untuk memberikan yang sangat singkat dan mudah untuk membaca definisi pemrograman matematis. (Ray Fernando, 2011)

2.1.2 Optimasi Kombinatorial

Optimasi kombinatorial adalah topik dalam ilmu komputer teoritis dan matematika terapan yang berfungsi untuk mencari solusi dengan biaya yang terkecil untuk masalah matematika dimana setiap solusi dikaitkan dengan *numerical cost*. Dalam beberapa permasalahan, pencarian menyeluruh tidak dapat dilakukan. Beroperasi pada daerah yang ini dioptimisasi, dimana set solusi yang layak adalah diskrit atau dapat dikurangi menjadi diskrit, dan dimana tujuannya adalah untuk mencari solusi yang terbaik. Beberapa masalah umum yang melibatkan optimasi kombinatorial adalah *travelling salesman problem* dan *the minimum spanning tree problem.* (Ray Fernando, 2011)

2.2 Kontainer

Di Indonesia kontainer dikenal dengan nama peti kemas yang terbuat dari bahan logam dan beberapa macam ukuran dan tipe. Peti kemas atau kontainer dapat dikatakan sebagai "*the moving go down*" yaitu gudang mini yang bergerak dari satu tempat ke lain tempat sebagai akibat dari adanya pengangkutan. (Herman A. Carel Lawalata, 2000)

Menurut Bambang Semedi dalam artikelnya tentang Indikator Penyalahgunaan Peti Kemas, Peti kemas merupakan satu kemasan yang dirancang secara khusus dengan ukuran tertentu, dapat dipakai berulang kali, dipergunakan untuk menyimpan dan sekaligus mengangkut muatan yang ada di dalamnya.

2.2.1 Jenis Peti Kemas

Internasional Standard Organization (ISO) telah menetapkan ukuran-ukuran dari peti kemas sebagai berikut :

		Peti kemas 20 feet		Peti kemas 40 feet		Peti kemas 45 feet	
		Inggris	Metrik	Inggris	Metrik	Inggris	metrik
Dimensi luar	**Panjang**	19' 10½"	6.058 m	40′ 0″	12.192 m	45′ 0″	13.716 m
	Lebar	8′ 0″	2.438 m	8′ 0″	2.438 m	8′ 0″	2.438 m
	Tinggi	8′ 6″	2.591 m	8′ 6″	2.591 m	9′ 6″	2.896 m
Dimensi dalam	**Panjang**	18′ 10 5/16"	5.758 m	39′ 5 45/64″	12.032 m	44′ 4″	13.556 m
	Lebar	7′ 8 19/32″	2.352 m	7′ 8 19/32″	2.352 m	7′ 8 19/32″	2.352 m
	Tinggi	7′ 9 57/64″	2.385 m	7′ 9 57/64″	2.385 m	8′ 9 15/16″	2.698 m
Bukaan pintu	**Lebar**	7′ 8 ⅛″	2.343 m	7′ 8 ⅛″	2.343 m	7′ 8 1/8″	2.343 m
	Tinggi	7′ 5 ¾″	2.280 m	7′ 5 ¾″	2.280 m	8′ 5 49/64″	2.585 m

Volume	1,169 ft^3	33.1 m^3	2,385 ft^3	67.5 m^3	3.040 ft^3	25.9 m^3
Berat kotor	52,910 lb	24.000kg	67.200kg	30.480kg	67.200 lb	30.480kg
Berat kosong	4,850 lb	2.200kg	8.380kg	3.800kg	10.580 lb	4.800 kg
Muatan bersih	48,060 lb	21.800kg	58.820kg	26.680kg	56.620lb	25.680kg

Tabel 2.1 Jenis Peti Kemas (Sumber: Bambang S, 2012)

2.2.2 Cara Pemuatan Barang dalam Kontainer

Cara pemuatan barang dalam kontainer terbagi dalam 2 sistem, yaitu :

1. **Sistem *Full Container Loaded***

   Cara pemuatan barang menggunakan sistem ini, kontainer dapat diisi dengan berbagai jenis barang dengan syarat barang-barang tersebut mempunyai 1 alamat penerima yang sama. Sistem *Full Container Loaded* si penyewa menyewa 1 kontainer penuh meskipun masih ada ruang kosong yang tersedia. Ruang kosong tidak akan diisi barang lain dengan penerima yang berbeda.

2. **Sistem *Less Than Container Loaded***

   Cara pemuatan barang menggunakan sistem ini, kontainer dapat diisi dari berbagai jenis barang yang mempunyai alamat penerima yang berbeda. Dengan sistem ini kontainer dapat terisi penuh jika barang yang dimasukkan mempunyai tujuan yang sama atau searah. Di sistem ini barang-barang dipisahkan berdasarkan

tujuan agar lebih mudah dalam proses bongkar dan proses pengiriman barang. (Nur N. 2004)

2.3 PT ANTESS (Antaran Express)

PT ANTESS merupakan perusahaan bidang jasa transportasi logistik yang bertempat di Graha Sarana – Puri Surya Jaya, J01-20 Cluster Vancouver, Gedangan Sidoarjo. PT ANTESS juga mempunyai gudang barang dan pool dari armadanya di daerah Pergudangan Sinar Buduran, B-22 Lingkar Timur Buduran – Sidoarjo. PT ANTESS mempunyai cabang di Ruko Tiga Indah, JL. Raya Jati Asih No 10B Bekasi. Tidak hanya menyediakan jasa transportasi logistik, PT ANTESS yang merupakan salah satu anak perusahaan Sarana Group, juga difokuskan bergerak dalam bidang kargo manajemen, jasa pergudangan dan distribusi sesuai kebutuhan pelanggan. Visi dari PT ANTESS adalah

1. Memiliki sumber daya manusia serta produk jasa yang berkualitas.
2. Mendukung pelanggan dalam melaksanakan strategi *Cost Leadership.*
3. Menjadi perusahaan paling diakui dalam industri jasa transportasi logistik, manajemen kargo, pergudangan dan layanan distribusi.
4. Bekerja berdasarkan kultur perusahaan yang kuat dan kesadaran terhadap kesehatan lingkungan dan kelestarian alam.

5. Melaksanakan praktik bisnis sehat serta komitmen yang kuat terhadap tanggung jawab sosial.
6. Memberikan kesempatan karir berjenjang kepada seluruh karyawan, sesuai perkembangan perusahaan.

Misi dari PT ANTESS adalah

1. Memberikan pelayanan dengan nilai tambah serta memberikan pendapatan yang sesuai kepada perusahaan.
2. Perbaikan dan pengembangan kualitas pelayanan secara berkelanjutan.
3. Memberikan keunggulan pelayanan.
4. Meningkatkan kesejahteraan spiritual sebagai landasan bagi setiap staff, sehingga setiap tujuan personal dan tujuan perusahaan tercapai dengan baik dan benar.

Kultur dari perusahaan adalah

1. peduli
2. kasih
3. efektif
4. fleksibel
5. bersikap positif
6. mengutamakan keselamatan kerja
7. pembelajaran dan perbaikan yang berkesinambungan
8. memiliki tanggung jawab pribadi dan professional

9. memiliki kesadaran terhadap kesehatan lingkungan dan kelestarian alam
10. menumbuh kembangkan ide-ide kreatif
11. mengembangkan keunggulan pelayanan secara berkelanjutan
12. memiliki rasa kebersamaan dan bangga.

2.4 Algoritma *Tabu Search*

Tabu search adalah sebuah metode optimasi yang berbasis pada *local search*. Proses pencarian bergerak dari satu solusi ke solusi berikutnya, dengan cara memilih solusi terbaik *neighborhood* solusi sekarang (*current*) yang tidak tergolong solusi terlarang (tabu). Ide dasar dari algoritma *tabu search* adalah mencegah proses pencarian dari *local search* agar tidak melakukan pencarian ulang pada ruang solusi yang sudah pernah ditelusuri, dengan memanfaatkan suatu struktur memori yang mencatat sebagian jejak proses pencarian yang telah dilakukan. (Henri P Panggabean, 2005)

Tabu Search diperkenalkan pertama kali oleh Glover pada tahun 1970-an[GLO86]. Ide dasar dari *tabu search* juga disampaikan oleh Hansen [HAN86]. Banyak eksperimen menunjukkan bahwa *tabu search* saat ini telah menjadi suatu teknik optimasi yang dapat diadu dengan hampir semua teknik optimasi yang telah dikenal.

Tabu search mempunyai *tabu list* yang berguna untuk menyimpan solusi-solusi yang memenuhi kriteria aspirasi sebagai nilai yang optimal dari proses iterasi yang telah dilakukan. *Tabu list* berguna agar pencarian nilai

optimal bisa menelusuri semua langkah-langkah yang belum terkunjungi dan agar pencarian tidak terulang kembali ke solusi-solusi yang telah dikunjungi.

Tabu Search secara iteratif menggunakan algoritma *local search* pada setiap iterasi untuk mencari solusi terbaik diantara sebagian tetangga dari solusi terbaik saat ini. Pada setiap iterasi, algoritma *local search* memilih solusi tetangga yang memberikan peningkatan kualitas tertinggi. Tetapi jika semua solusi tetangga tidak memberikan peningkatan kualitas, maka *local search* akan memilih solusi yang penurunan kualitasnya paling rendah. Kualitas disini bergantung pada masalah yang dihadapi. (Boko Susilo, dkk. 2012)

Tabu Search dalam penyelesaiannya harus melewati setiap tahapan tertentu yang telah diatur oleh Glover, adapun tahapan-tahapan tersebut adalah :

1. Membangkitkan solusi awal

   Yang dimaksudkan disini adalah sebelum kita memulai tahapan *tabu search*, kita mempunyai acuan awal sebagai pembanding ketika proses *tabu search* dimulai.

2. Menentukan kriteria aspirasi (*aspiration criteria*)

   Kriteria aspirasi ini fungsinya sebagai fungsi tujuan atau *goal* yang akan dicapai, contohnya untuk penelitian ini kriteria aspirasinya adalah minimasi jarak material *handling*.

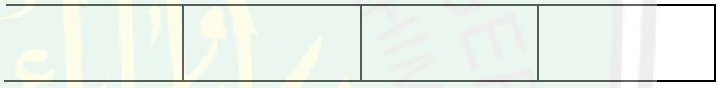
3. Melakukan *Move*

Ada beberapa macam *move* yang dapat dipilih selama proses pencarian ini berlangsung :

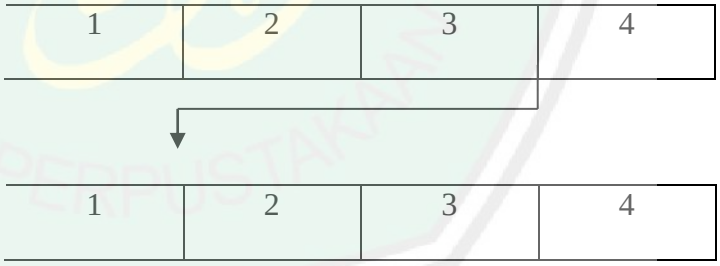
1. *Local Search*, yang terdiri dari dua macam yaitu :

a. *Insertion* : memilih secara acak satu bagian struktur untuk dipindah ke bagian yang lain

Contoh : Struktur Awal

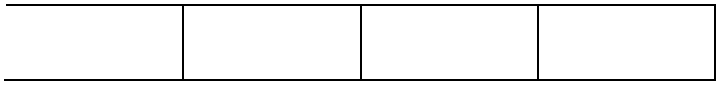


Jika dengan proses *random* didapat atribut ke-3, maka struktur dapat berubah menjadi :

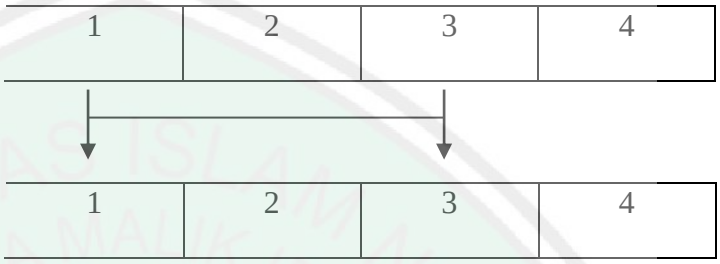


b. *Swap* : memilih secara acak dua bagian struktur untuk selanjutnya ditukar posisinya.

Contoh : Struktur Awal



Jika dengan proses *random* didapat atribut ke-3, maka struktur dapat berubah menjadi :



2. *Neighborhood Search*

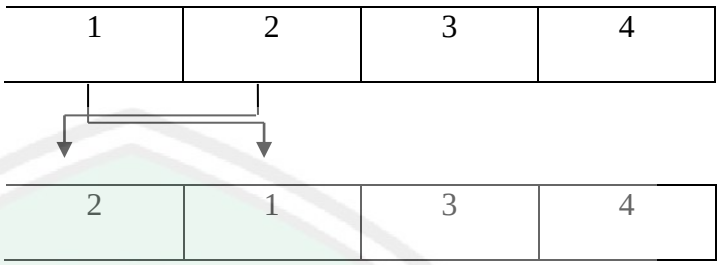
Untuk pencarian dengan teknik ini setiap kemungkinan atribut dari struktur dapat dipindah-pindah. Permutasi *n-charge neighborhood* mengambil n elemen dari matrik solusi (dimana berhubungan dengan item yang sedang diproduksi pada suatu mesin pada satu waktu), dan untuk tiap-tiap pengubahan item yang sedang diproduksi dengan item lain. Perubahan yang dipakai oleh dua *neighborhood* dengan melakukan *swap* elemen matrik atau kombinasi elemen itu dengan menukar elemen lain dalam matrik.

Contoh: Struktur Awal

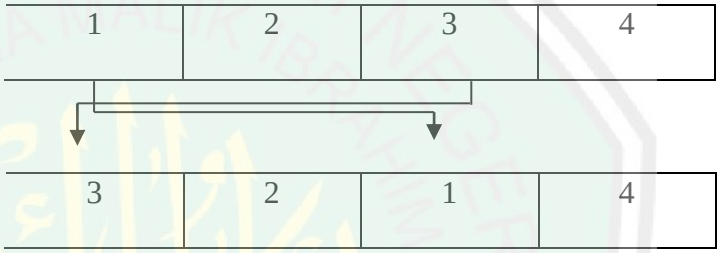
1	2	3	4

Dipilih 3 charge *neighborhood*, maka struktur atribut di atas dapat berubah menjadi:

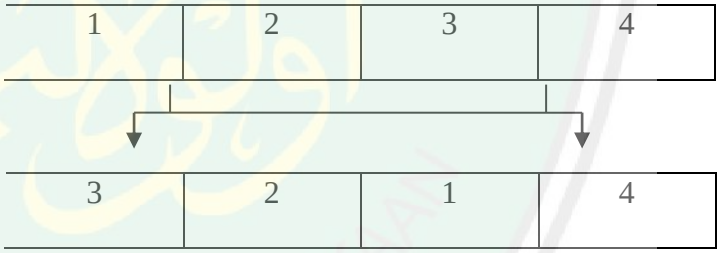
i



ii



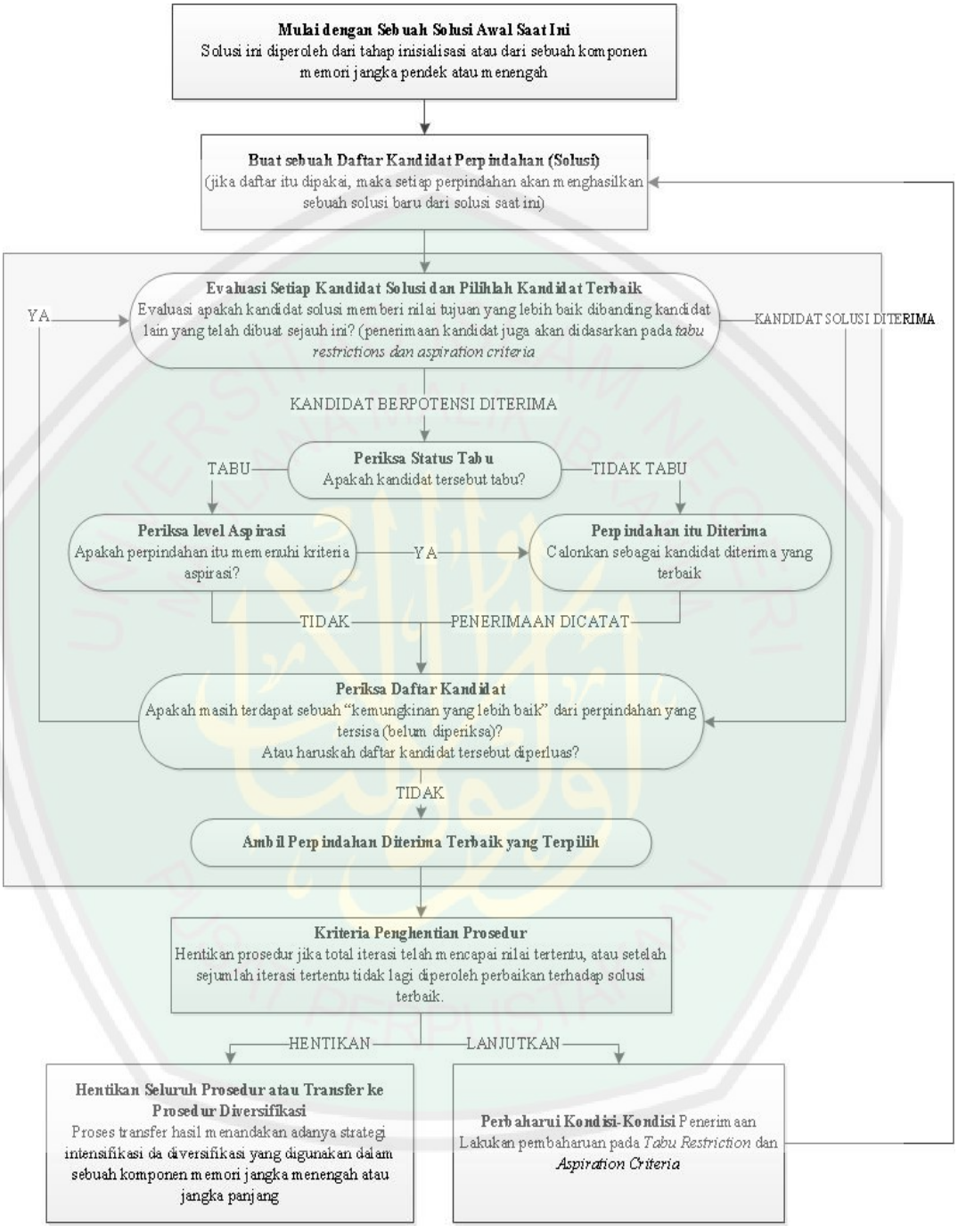
iii



4. Untuk menghindari terulangnya langkah yang diambil, maka dilakukan *tabu test*, *Tabu Test* memanfaatkan *tabu list* yang sudah ada. Tujuan sebenarnya dari *tabu list* bukan untuk mencegah terulangnya langkah yang telah diambil, tetapi lebih kepada agar tidak mundur. Untuk mencegah perulangan, daftar solusi yang telah dicapai disimpan dalam sebuah tabel.

5. *Alternatif move* yang lolos *tabu test* masih harus melewati *aspiration test*, apakah bisa melewati *aspiration threshold* atau tidak, jika tidak maka teruskan iterasi berikutnya.
6. Jika *alternatif move* mempunyai *aspiration criteria* yang lebih baik daripada *aspiration threshold* maka dilakukan eksekusi terhadap *alternatif move* tersebut dan memperbarui memori yang tidak relevan.
7. jika aturan pemberhentian sudah memenuhi syarat pemberhentian, maka pencarian berhenti. (Glover, 1986)

Secara umum algoritma *tabu search* dapat dituliskan sebagai berikut :



Gambar 2.1 Algoritma *Tabu Search* (Sumber: Priyandari, 2009)

Tabu search membutuhkan sebuah rumus fungsi untuk mengerjakan permasalahan 3D. Permasalahan dalam 3D mempertimbangkan ukuran

panjang, lebar dan tinggi. Kami mengembangkan konsep *filling function* yang didefinisikan sebagai berikut :

$$_{\varphi}(\boldsymbol{S_i}) = \propto \frac{\sum_{j \in S_i} \boldsymbol{w_j h_j d_j}}{\boldsymbol{W\ H\ D}} - \frac{|\boldsymbol{S_i}|}{\boldsymbol{n}}$$

$\boldsymbol{S_i}$ adalah kandidat solusi optimal. $\propto$ adalah parameter awal non negatif, digunakan agar hasil perhitungan selalu positif. $\boldsymbol{w}, \boldsymbol{h}, \boldsymbol{d}$ adalah dimensi barang (panjang, lebar, dan tinggi). $\boldsymbol{W}, \boldsymbol{H}, \boldsymbol{D}$ adalah dimensi kontainer bagian dalam (panjang, lebar, dan tinggi). $\boldsymbol{n}$ adalah jumlah barang yang diinputkan. (Li Pan; Z Huang Joshua; Sydney C.K Chu. 2008)

.

BAB III

ANALISIS DAN PERANCANGAN SISTEM

3.1 Analisa dan Perancangan Sistem

3.1.1 Keterangan Umum

Aplikasi optimasi proses muat barang dalam kontainer dikembangkan dalam *desktop platform* menggunakan perangkat lunak (*software*) NetBeans IDE 7.3.1. Aplikasi bertujuan mencari nilai minimum untuk proses muat barang dalam kontainer. Pengguna menginputkan data dari identitas barang, jenis barang dan dimensi barang (panjang, lebar, tinggi, dan berat). Dalam aplikasi ini jenis kontainer yang digunakan hanya satu jenis sehingga pengguna tidak dapat mengganti jenis kontainer. *Output* dari aplikasi berupa hasil dari perhitungan proses muat barang menggunakan algoritma *tabu search* dan simulasi barang yang tertata dalam kontainer.

Metode yang digunakan *Tabu Search.* Konsep dasar dari *Tabu Search* adalah pengefektifan proses pencarian solusi dengan cara mencari *best solution* pada setiap tahap pelacakan. *Tabu Search* mencegah proses pencarian dari *local search* agar tidak terjadi pencarian ulang pada ruang solusi yang telah ditelusuri.

3.2 Rancangan Penggunaan Aplikasi

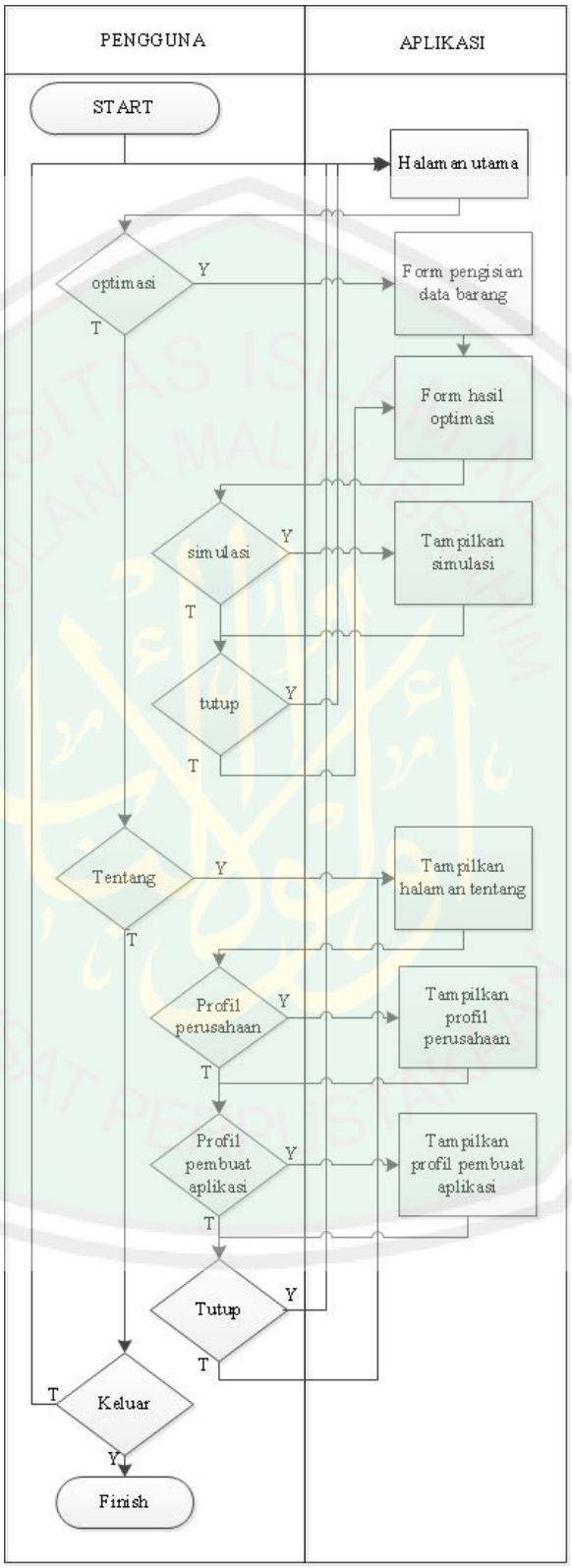
Aplikasi optimasi proses muat barang dalam kontainer mempunyai beberapa menu dalam halaman utama, yaitu : menu optimasi, menu tentang, dan menu keluar. Menu optimasi berfungsi membantu pengguna untuk melakukan optimasi proses muat barang dalam kontainer. Dalam menu optimasi pengguna melakukan pengisian data identitas barang dalam halaman pengisian data barang. Halaman pengisian data barang ditampilkan setelah pengguna menekan tombol menu optimasi. Setelah memasukkan data identitas barang yang diperlukan, pengguna dapat melihat hasil dari perhitungan aplikasi optimasi proses muat barang dalam kontainer dalam halaman hasil optimasi. Halaman hasil optimasi ditampilkan setelah pengguna menekan tombol proses dalam halaman pengisian data barang.

Halaman hasil optimasi menampilkan beberapa *tab* untuk hasil perhitungan optimasi proses muat barang dalam kontainer menggunakan algoritma *tabu search.* Pada *tab* pertama aplikasi menampilkan data semua barang yang diinputkan. Lalu pada *tab* kedua, aplikasi menampilkan data identitas barang yang tertata dalam kontainer serta koordinat penempatan barang dalam kontainer. Pada *tab* ketiga, aplikasi menampilkan data barang yang tidak tertata dalam kontainer. Selain itu aplikasi juga menampilkan hasil *filling function* dari solusi yang terpilih, jumlah barang yang tertata dan tidak tertata, jumlah isi tabu tabel, jumlah isi *tabu list*, total berat dari barang yang tertata dan volume dari barang yang tertata.

Pada *tab* ketiga, terdapat tombol simulasi, tombol cetak dan tombol tutup. Tombol simulasi digunakan untuk menampilkan gambaran barang yang tertata dalam kontainer. Penggambaran barang berupa grafik yang berdasarkan koordinat posisi barang dalam kontainer. Tombol hasil iterasi digunakan untuk menampilkan grafik perhitungan konsep *filling function* dari semua iterasi. Tombol tutup digunakan menutup halaman hasil optimasi.

Menu tentang, dalam menu ini aplikasi menampilkan profil singkat dari perusahaan logistik yang dijadikan studi kasus penelitian ini yaitu, PT ANTESS. Selain menampilkan profil perusahaan terkait, aplikasi juga menampilkan profil singkat dari pembuat aplikasi. Selanjutnya menu keluar, menu ini digunakan untuk keluar dari aplikasi optimasi proses muat barang dalam kontainer.

Flowchart untuk rancangan penggunaan aplikasi bagi pengguna digambarkan dibawah ini :



Gambar 3.1 *Flowchart* Rancangan Penggunaan Aplikasi

3.3 Rancangan Aplikasi

Rancangan aplikasi yang dibangun dalam penelitian ini adalah sebagai berikut :

Gambar 3.2 Diagram Blok Aplikasi

Penjelasan dari diagram rancangan aplikasi dijelaskan sebagai berikut :

1. Inisialisasi Identitas Barang dan Kontainer

   Melakukan pengisian data identitas barang oleh pengguna yang meliputi nama barang, dimensi barang (panjang, lebar, tinggi) dengan satuan cm, berat barang dan berat volumetrik dengan satuan Kg, tujuan barang, dan jenis barang. Untuk

pengisian berat volumetrik barang dilakukan oleh aplikasi. Berat volumetrik digunakan untuk mengetahui berat barang berdasarkan dimensi barang. Perhitungan berat volumetrik sudah diakui oleh ASPERINDO. Rumus yang digunakan dalam pengiriman barang via darat adalah

Berat(Kg) = panjang(cm) * lebar(cm) * tinggi(cm) / 4000.

Dalam penelitian ini penggunaan jenis kontainer hanya satu jenis sehingga pengisian identitas kontainer tidak dapat dilakukan oleh pengguna. Identitas kontainer terinisialisasi oleh aplikasi. Identitas kontainer meliputi tipe kontainer, dimensi dalam kontainer (panjang, lebar, tinggi), kapasitas volume kontainer, dan maksimum berat kontainer.

2. Penataan Barang

   Melakukan penataan barang dengan berbagai parameter. Parameter yang utama dalam proses penataan barang adalah parameter tujuan dan jenis.

3. Algoritma *Tabu Search*

   Mengimplementasikan algoritma *tabu search* dalam penelitian ini menggunakan konsep *filling function.*

   Konsep *filling function* bisa digunakan setelah aplikasi membangkitkan kandidat solusi optimal. Konsep *filling function* juga digunakan sebagai kriteria aspirasi dalam aplikasi ini.

4. Visualisasi Hasil Optimasi

   Memberikan gambaran penataan barang dalam kontainer dari solusi paling optimal berupa grafik.

3.4 Rancangan Penataan Barang

Proses penataan barang ini menggunakan konsep dari perusahaan yang telah digunakan dan peneliti memodifikasinya dengan algoritma *tabu search*. Dalam proses penataan barang ada beberapa hal yang perlu diperhatikan, yaitu :

1. Bilangan dan Satuan dari Dimensi Barang dan Berat Barang.

   Bilangan dimensi barang dan berat barang yang diinputkan oleh pengguna harus bilangan bulat. Pembulatan yang dilakukan pembulatan puluhan. Satuan yang digunakan dalam dimensi barang adalah cm sementara berat barang Kg.

   Contoh :

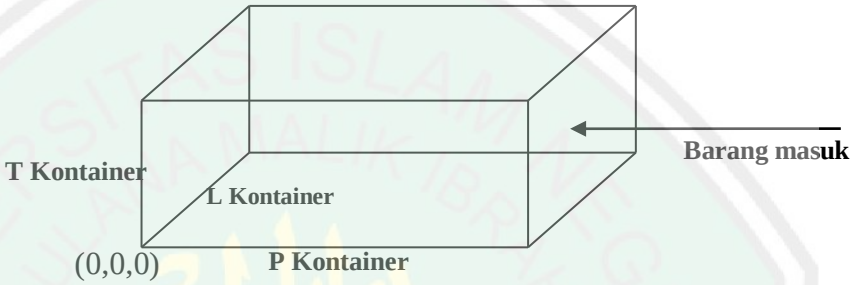
   panjang 83,5 cm menjadi 84 cm

   lebar 14.3 cm menjadi 14 cm

   tinggi 65.7 cm menjadi 66 cm

2. Penempatan Barang dalam Kontainer

   Barang yang mendapatkan urutan pertama untuk masuk ke dalam kontainer, barang tersebut menempati koordinat [0,0,0] yang dijelaskan oleh gambar dibawah ini :



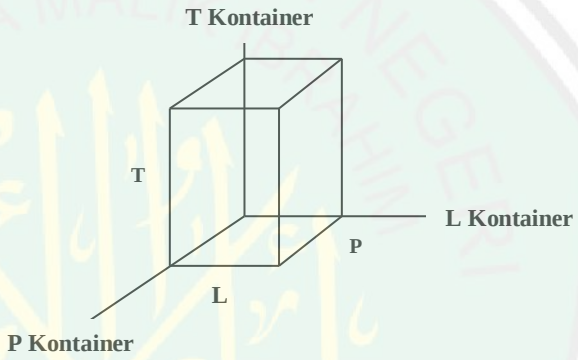
Gambar 3.3 Kontainer Tampak Samping

3. Letak Barang dalam Kontainer Berdasarkan Tujuan Barang

   Barang yang diletakkan pertama kali dalam kontainer adalah barang yang memiliki tujuan kota paling jauh dari tujuan kota barang-barang lainnya. Dalam penelitian ini tujuan yang paling jauh adalah Kota Jakarta. Setelah barang dengan tujuan Kota Jakarta telah masuk semua dalam kontainer lalu barang yang masuk berikutnya adalah barang yang memiliki kota tujuan terdekat dengan Jakarta, dalam penelitian ini adalah Bandung. Setelah barang dengan tujuan Kota Bandung telah masuk semua lalu masukkan sisa barang dengan tujuan Kota Semarang. Dalam penelitian ini hanya terdapat 3 kota tujuan, yaitu: Jakarta, Bandung, dan Semarang.

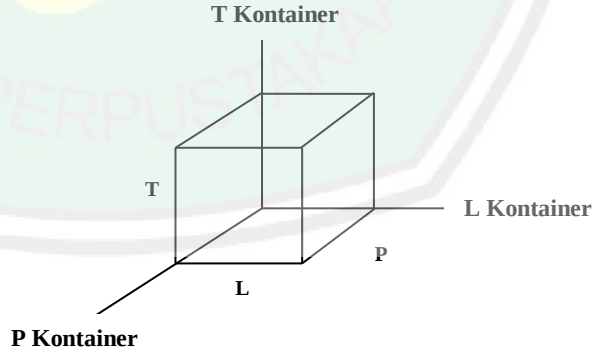
4. Posisi Barang dalam Kontainer Berdasarkan Jenis Barang

Dalam penelitian ini hanya terdapat 3 jenis barang, yaitu: kulkas, televisi, dan makanan. Barang yang akan ditata dalam kontainer posisinya tidak dapat diubah dari posisi awal barang. Posisi barang dengan jenis kulkas diletakkan berdiri tidak boleh diletakkan dalam posisi tidur.



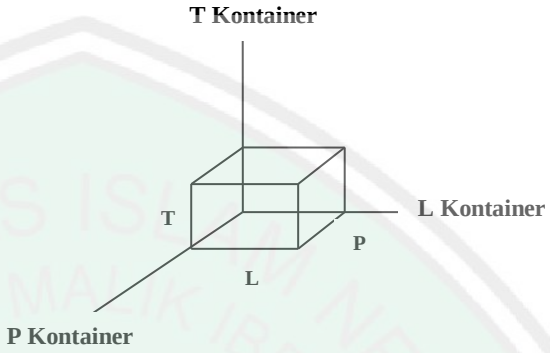
Gambar 3.4 Posisi Barang Jenis Kulkas dalam Kontainer

Posisi barang dengan jenis barang televisi dalam kontainer



Gambar 3.5 Posisi Barang Jenis Televisi dalam Kontainer

Dan posisi barang untuk jenis barang makanan dalam kontainer



Gambar 3.6 Posisi Barang Jenis Makanan dalam Kontainer

Cara penataan barang dalam kontainer terdapat beberapa langkah, yaitu:

1. Mendapatkan jumlah barang

   Menghitung jumlah barang yang tersimpan dalam tabel data_barang. Tabel data_barang digunakan untuk menyimpan identitas barang yang telah diinputkan oleh pengguna.

2. Membuat solusi berdasarkan kota tujuan barang

   Selain membuat solusi yang terisi dari berbagai kota tujuan barang. Dibuat lagi solusi yang berdasarkan kota tujuan agar pengecekan barang lebih optimal. Solusi ini juga digunakan sebagai pergantian proses penataan barang dari tujuan kota terjauh hingga terdekat dari kota keberangkatan.

3. Melakukan perulangan dengan batas maksimumnya jumlah barang yang tersimpan
4. Membuat parameter

   pada perusahaan terkait tidak terdapat aturan baku dalam penataan barang dalam kontainer. Sehingga peneliti membuat berbagai parameter untuk membantu proses penataan barang. Parameter tersebut adalah :

   - Berat barang

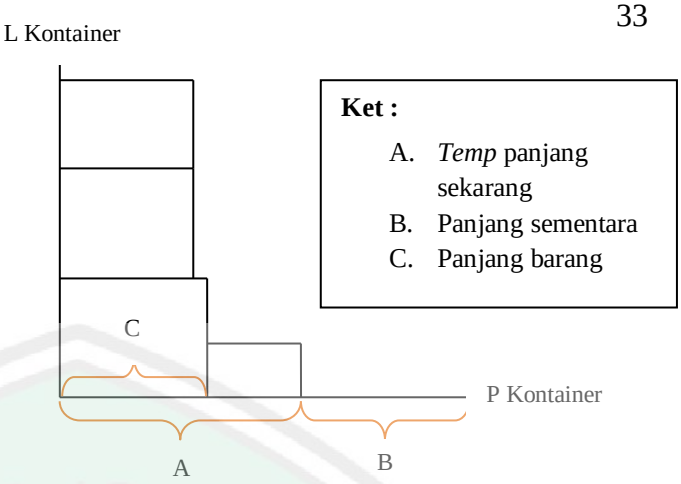
     Dilakukan proses penataan barang jika berat barang sekarang kurang dari berat kontainer maksimum. Berat barang sekarang diperoleh dari jumlah berat barang yang telah tertata dalam kontainer.

   - Tujuan barang

     Barang dengan tujuan kota terjauh dari kota keberangkatan diproses terlebih dahulu.

   - Panjang barang

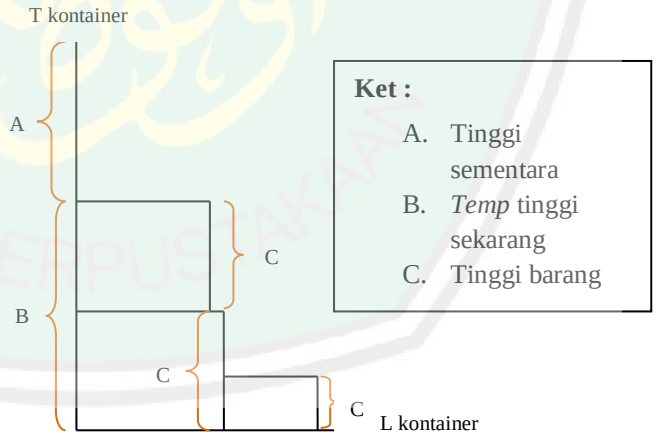
     Dilakukan proses penataan barang jika panjang barang sekarang kurang dari panjang sementara. Panjang sementara diperoleh dari maksimal panjang kontainer dikurangi *temp* panjang sekarang. *Temp* panjang sekarang diperoleh dari jumlah panjang barang yang telah tertata dalam kontainer.



Gambar 3.7 Cara Penataan Barang dengan Parameter panjang

- Tinggi barang

  Dilakukan proses penataan barang jika tinggi barang sekarang kurang dari tinggi sementara. Tinggi sementara diperoleh dari maksimal tinggi kontainer dikurangi *temp* tinggi sekarang. *Temp* tinggi sekarang diperoleh dari jumlah tinggi barang yang telah tertata.



Gambar 3.8 Cara Penataan Barang dengan Parameter Tinggi

- Lebar barang

  Dilakukan proses penataan barang jika lebar barang sekarang kurang dari lebar sementara. Lebar sementara diperoleh dari maksimum berat lebar kontainer dikurangi *temp* lebar

sekarang. *Temp* lebar sekarang diperoleh dari penambahan lebar barang yang telah tertata.

- Jenis barang

  Digunakan meletakkan barang dengan jenis tertentu. Dalam penelitian ini hanya ada tiga jenis barang, yaitu kulkas, televisi dan makanan. Peletakkan barang dengan jenis kulkas dan televisi berada dibawah dengan posisi tidak boleh tidur. Apabila diletakkan diatas maka barang yang berada di bawahnya harus barang dengan jenis yang sama. Sementara barang dengan jenis makanan akan diletakkan diatas kulkas dan televisi. Barang berjenis makanan dapat diletakkan di bawah jika sudah tidak ada lagi barang berjenis elektronik atau barang dengan jenis elektronik yang tidak dapat tertata dalam kontainer.

3.5 Rancangan Algoritma *Tabu Search*

Algoritma *Tabu Search* dalam aplikasi ini akan diimplementasikan dipencarian nilai optimal (maksimum) proses muat barang dalam kontainer. Pencarian nilai optimal menggunakan sebuah konsep, yaitu konsep *filling function*. Konsep ini dilakukan setelah solusi kandidat solusi optimal dihasilkan. Pencarian dilakukan untuk mencari nilai yang paling maksimum dari perhitungan tersebut. Dalam aplikasi ini algoritma *tabu*

search tidak digunakan diproses penataan barang. Langkah dalam perhitungan nilai optimal menggunakan algoritma *tabu search* :

1. Pengguna memasukkan data identitas barang (nama barang), jenis barang, kota tujuan barang, dimensi barang(panjang, lebar, tinggi), dan berat barang. Berat_volume barang akan dihitung oleh aplikasi setelah pengguna mengisi dimensi barang.
2. Membangkitkan kandidat solusi optimal

   Dilakukan dua tahapan untuk membangkitkan kandidat solusi optimal :

   a. Pengambilan id_barang secara acak, dengan syarat barang yang mempunyai tujuan kota terjauh diletakkan di urutan pertama. Penempatan ini dilakukan hingga barang yang mempunyai tujuan kota terdekat berada di urutan terakhir. Syarat ini dilakukan agar mudah mengambil id_barang yang digunakan dalam proses penataan barang.

   b. Melakukan proses penataan barang dari solusi yang dihasilkan secara acak. Langkah ini digunakan untuk memeriksa apakah semua barang dalam solusi yang dihasilkan secara acak bisa tertata dalam kontainer. Hasil dari langkah ini merupakan kandidat solusi optimal yang akan dicari nilai optimalnya menggunakan konsep *filling function*.
3. Mengevaluasi kandidat solusi optimal

   Terdapat beberapa evaluasi dalam tahap ini, yaitu :

a. Membandingkan nilai *filling function*

Digunakan untuk kriteria aspirasi pencarian nilai optimal dengan mengetahui "apakah nilai *filling function* kandidat solusi sekarang lebih baik (dalam kasus ini yang paling maksimum) dari nilai *filling function* maksimum sementara?". Pencarian ini menggunakan konsep *filling function*

$$_{\varphi}(\boldsymbol{S_i}) = \propto \frac{\sum_{j \in S_i} w_j h_j d_j}{W\,H\,D} - \frac{|S_i|}{n}.$$

$\boldsymbol{S_i}$ adalah kandidat solusi optimal. $\propto$ adalah parameter awal non negatif, digunakan agar hasil perhitungan selalu positif. $\boldsymbol{w, h, d}$ adalah dimensi barang (panjang, lebar, dan tinggi). $\boldsymbol{W, H, D}$ adalah dimensi kontainer bagian dalam (panjang, lebar, dan tinggi). $\boldsymbol{n}$ adalah jumlah barang yang diinputkan.

Nilai *filling function* yang dihasilkan kandidat solusi optimal di iterasi ke 1 digunakan sebagai nilai maksimum sementara dikarenakan nilai dari nilai maksimum sementara pada awalnya adalah 0. Sementara kandidat solusinya disimpan sebagai kandidat solusi optimal dan otomatis disimpan dalam tabu tabel dan *tabu list*. Proses ini dilakukan hanya pada iterasi ke 1.

b. Periksa status tabu

Pencarian status tabu dilakukan menggunakan bantuan tabu tabel. Jika terbukti solusi pernah dilakukan maka status tabu diberikan. Jika tidak terbukti maka solusi belum pernah dilakukan dan status tabu tidak diberikan.

c. Periksa kriteria aspirasi

Selanjutnya memeriksa kriteria aspirasi solusi jika kandidat solusi berstatus tabu jika solusi tidak berstatus tabu langkah ini tidak dilakukan, kriteria aspirasi solusinya adalah "apakah solusi sekarang memberikan nilai maksimum yang lebih baik dari nilai maksimum sementara". Nilai maksimum didapat dari hasil perhitungan konsep *filling function*. Jika lebih baik maka nilai maksimum sementara diganti dengan nilai maksimum dari solusi sekarang. Diambil nilai yang paling maksimum karena berdasarkan parameter perkalian dimensi barang dan jumlah barang yang terdapat pada konsep *filling function.*

4. Menyimpan kandidat solusi dalam tabu tabel

Digunakan untuk menyimpan semua kandidat solusi yang telah dihasilkan kecuali solusi yang berstatus tabu. Tabu tabel bertujuan agar solusi yang pernah dihasilkan tidak terulang dalam iterasi selanjutnya.

5. Menyimpan kandidat solusi optimal dalam *tabu list*

   Tabu list digunakan untuk menyimpan kandidat solusi optimal yang hanya memenuhi kriteria aspirasi dan tidak berstatus tabu. Jika batas maksimum *tabu list* telah mencapai batasnya tetapi masih terdapat solusi yang dapat masuk ke dalam *tabu list*, maka solusi yang berada pada akhir *tabu list* akan diganti dengan solusi yang baru.

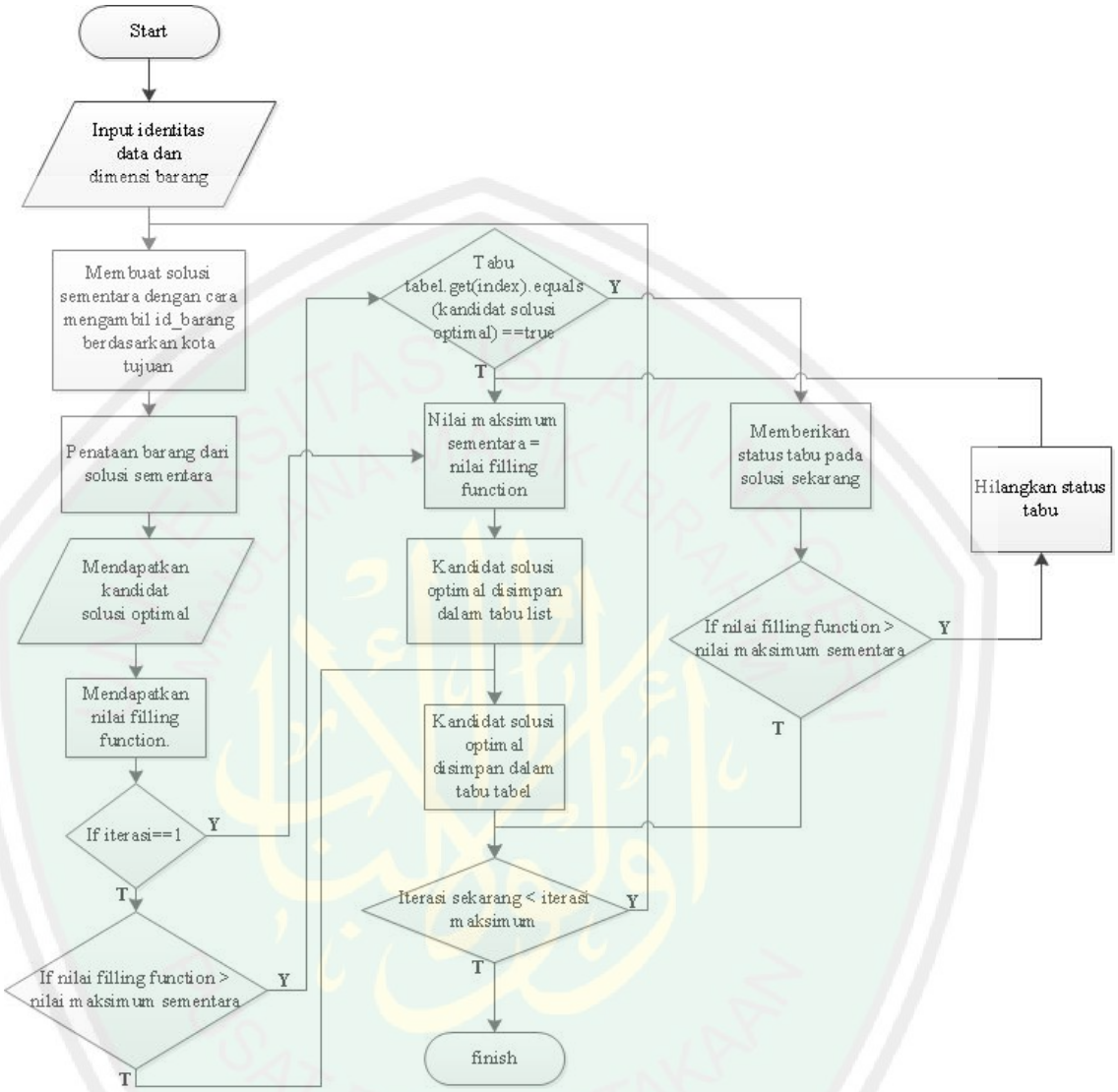
6. Memeriksa iterasi

   Memeriksa "apakah iterasi telah mencapai batas maksimum?" jika belum, lakukan proses pencarian nilai optimal lagi. Pencarian berhenti ketika iterasi sudah mencapai batas maksimum, bukan ketika *tabu list* mencapai batas maksimum.

7. Menghentikan proses

   Pemberhentian proses pencarian nilai optimal dilakukan setelah batas maksimum iterasi tercapai. Dalam aplikasi ini jumlah iterasi sama dengan jumlah isi tabu tabel.

Berikut adalah gambar rancangan *flowchart* dari algoritma *tabu search* :



Gambar 3.9 *Flowchart* Algoritma *Tabu Search* dalam Aplikasi

3.6 Rancangan Database

Dalam aplikasi optimasi ini memerlukan database untuk menyimpan data-data yang diperlukan. Data yang perlu disimpan dalam database yaitu data identitas barang dan dimensi barang yang telah diinputkan oleh pengguna dan solusi optimal yang telah dihasilkan.

3.6.1 Tabel data_barang

Tabel data_barang digunakan untuk menyimpan data-data identitas barang dan dimensi barang yang telah diinputkan oleh pengguna.

Nama	Tipe	Ukuran	Keterangan
Id_barang	Int	5	Primary key
Nama_barang	Varchar	5000	Nama barang
Panjang	int	5	Ukuran panjang barang
Lebar	Int	5	Ukuran lebar barang
Tinggi	Int	5	Ukuran tinggi barang
Berat	Int	5	Ukuran berat barang
Berat_volume	Int	5	Ukuran Berat volume barang
Tujuan	Varchar	5000	Kota tujuan barang
Jenis	Varchar	5000	Jenis barang

Tabel 3.1 Struktur Tabel data_barang

3.6.2 Tabel posisi_barang

Tabel ini digunakan untuk menyimpan identitas barang dan dimensi barang yang dihasilkan dari solusi acak berdasarkan kota tujuan barang. Barang-barang yang tersimpan dalam tabel ini belum pasti akan tertata semuanya dalam kontainer. Tabel ini berisi posisi_x_awal, posisi_y_awal, posisi_z_awal, posisi_x_akhir, posisi_y_akhir, dan posisi_z_akhir digunakan untuk menyimpan koordinat awal dari barang. dan koordinat akhir barang. Penambahan *field* ini juga digunakan agar memudahkan memvisualisasikannya dalam bentuk grafik.

Nama	**Tipe**	**Ukuran**	**Keterangan**
Id_barang	Int	5	Primary key
Panjang	Int	5	Ukuran panjang barang
Lebar	Int	5	Ukuran lebar barang
Tinggi	Int	5	Ukuran tinggi barang
Tujuan	Varchar	5000	Kota tujuan barang
Posisi_x_awal	Int	10	Koordinat x awal dalam kontainer
Posisi_y_awal	Int	10	Koordinat y awal dalam kontainer
Posisi_z_awal	Int	10	Koordinat z awal dalam kontainer
Posisi_x_akhir	Int	50	Koordinat x akhir dalam kontainer
Posisi_y_akhir	Int	50	Koordinat y akhir dalam kontainer
Posisi_z_akhir	int	50	Koordinat z akhir dalam kontainer
Berat_volume	Int	5	Ukuran Berat volume barang

Tabel 3.2 Struktur Tabel posisi_barang

3.6.3 Tabel posisi_barang_tersimpan

Tabel ini digunakan untuk menyimpan identitas barang dan dimensi barang yang telah tertata dalam kontainer. Barang-barang yang telah tertata dihasilkan dari penataan barang dari solusi acak. Terdapat *field* volume, *field* ini digunakan untuk mengetahui berapa volume yang dihasilkan barang yang tertata dalam kontainer.

Nama	Tipe	Ukuran	Keterangan
Id_barang	Int	50	Primary key
Panjang	Int	50	Ukuran panjang barang
Lebar	Int	50	Ukuran lebar barang
Tinggi	Int	50	Ukuran tinggi barang
Tujuan	Varchar	5000	Kota tujuan barang
Posisi_x_awal	Int	50	Koordinat x awal dalam kontainer
Posisi_y_awal	Int	50	Koordinat y awal dalam kontainer
Posisi_z_awal	Int	50	Koordinat z awal dalam kontainer
Posisi_x_akhir	Int	50	Koordinat x akhir dalam kontainer
Posisi_y_akhir	Int	50	Koordinat y akhir dalam kontainer
Posisi_z_akhir	int	50	Koordinat z akhir dalam kontainer
Berat_volume	Int	50	Ukuran Berat volume barang
Volume	Int	50	Ukuran volume barang

Tabel 3.3 Struktur Tabel posisi_barang_tersimpan

3.6.4 Tabel posisi_barang_taktersimpan

Tabel ini digunakan untuk menyimpan barang-barang yang tidak tertata dalam kontainer yang berasal dari solusi optimal yang dihasilkan.

Nama	Tipe	Ukuran	Keterangan
Id_barang	Int	50	Primary key
Panjang	Int	50	Ukuran panjang barang
Lebar	Int	50	Ukuran lebar barang
Tinggi	Int	50	Ukuran tinggi barang
Tujuan	Varchar	5000	Kota tujuan barang

Tabel 3.4 Struktur Tabel posisi_barang_taktersimpan

3.6.5 Tabel iterasi

Tabel ini digunakan untuk menyimpan indeks dari iterasi dan hasil perhitungan konsep *filling function* dari iterasi tersebut. Isi dari tabel ini digunakan untuk memvisualisasikan hasil konsep *filling function* per-iterasi berupa grafik.

Nama	Tipe	Ukuran	Keterangan
Id_iterasi	Int	50	Urutan iterasi
Hasil_fillingFunction	Int	50	Hasil perhitungan konsep *filling function*

Tabel 3.5 Struktur Tabel iterasi

3.7 Rancangan *Interface*

Aplikasi optimasi ini mempunyai rancangan *interface* sebagai berikut :

1. Halaman Utama

   Halaman utama memberikan 3 pilihan menu, yaitu :

   1. Optimasi

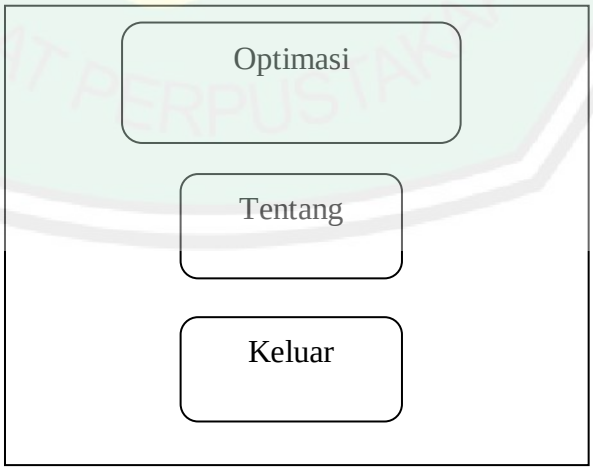
      Menuju ke halaman pengisian data barang yang digunakan sebagai data utama untuk perhitungan optimasi muat barang dalam kontainer.

   2. Tentang

      Berisi informasi profil PT ANTESS dan profil pembuat aplikasi.

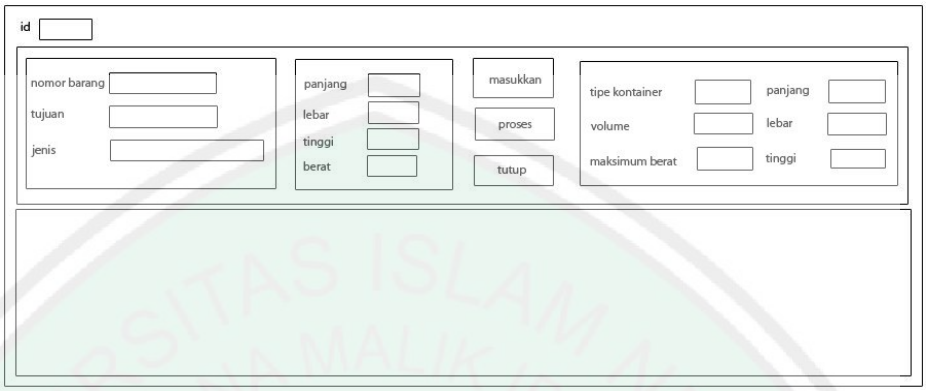
   3. Keluar

      Digunakan untuk menutup aplikasi optimasi proses muat barang dalam kontainer.



Gambar 3.10 Halaman Utama

2. Halaman Pengisian Data Barang



Gambar 3.11 Halaman Pengisian Data Barang

Terdapat tiga baris dalam menu pengisian data barang. Pada baris pertama hanya ada id yang digunakan untuk urutan barang yang diinputkan dan disimpan dalam database. Pengguna tidak dapat mengedit isi dari *textfield* disamping id. *Textfield* id terisi otomatis oleh aplikasi.

Baris kedua terdapat 4 kolom. Pada kolom pertama terdapat identitas dan jenis barang. Identitas barang berisikan nomor barang, tujuan barang dan jenis barang. Dan telah disediakan *textfield* bagi pengguna untuk menginputkan data. Pada tujuan barang dan jenis barang disediakan menu *combobox*. Pada kolom kedua terdapat dimensi barang. Pengguna disediakan *textfield* panjang, lebar, tinggi, dan berat untuk menginputkan data.

Terdapat tiga tombol pada kolom ketiga, yaitu:

1. Masukkan

   Tombol yang berfungsi untuk menginputkan data ke database dan menampilkan data di tabel data.

2. Proses

   Tombol yang berfungsi untuk memproses semua data yang telah terinputkan dan memberikan hasil perhitungan optimasi proses muat barang yang ditampilkan di halaman hasil optimasi.

3. Tutup

   Tombol yang berfungsi untuk kembali ke halaman utama.

Kolom keempat terdapat keterangan kontainer yang berisikan informasi dari kontainer yang digunakan. Terdapat beberapa keterangan :

- Tipe kontainer
- Panjang kontainer
- Volume kontainer
- Lebar kontainer
- Maks. Berat kontainer
- Tinggi kontainer

Dimensi(panjang, lebar, tinggi) kontainer yang ditampilkan merupakan dimensi dalam dari kontainer. Maksimum berat kontainer merupakan maksimum muatan bersih dari kontainer. Pengguna tidak dapat mengganti identitas kontaner.

Baris ketiga dalam menu pengisian data barang terdapat sebuah tabel. Tabel yang berisikan data-data yang telah diinputkan oleh pengguna dan tersimpan dalam database. Tabel ini selain berfungsi

sebagai menampilkan data, tabel juga berfungsi untuk mengedit dan menghapus data.

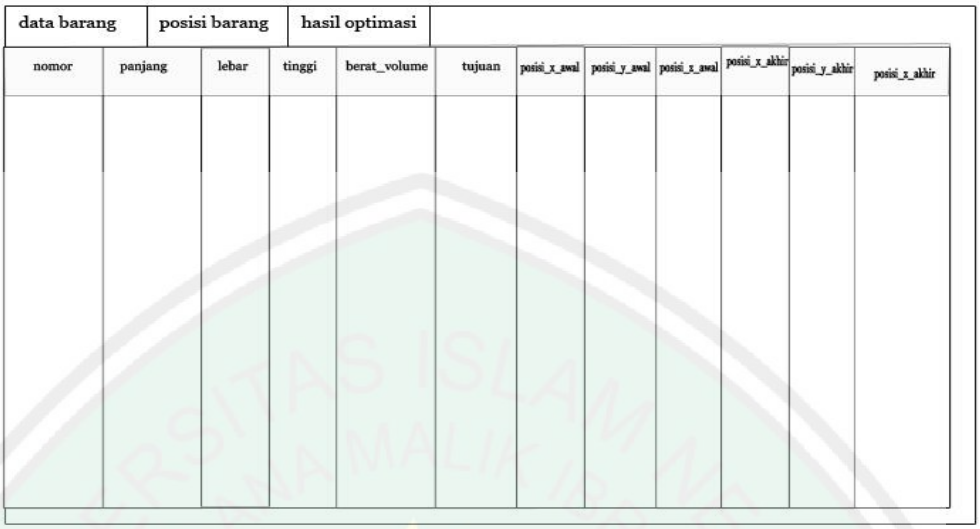
Jika ingin mengedit, pengguna memilih data yang akan diedit lalu klik 2 kali, selanjutnya pengguna mengganti data sesuai dengan yang diinginkan. Selesai mengganti data, tekan tombol enter 2 kali hingga muncul pesan yang memberi informasi bahwa data telah berhasil diedit dan tersimpan di database. Jika pengguna ingin menghapus data, user memilih baris data yang diinginkan dan tekan tombol *delete* di *keyboard* hingga muncul pesan 'peringatan' bahwa data telah berhasil dihapus.

3. Halaman Hasil Optimasi

Terdapat 3 *tabbed pane* dalam halaman hasil optimasi. Yang pertama adalah *tab* data barang. Pada *tab* ini aplikasi menampilkan data barang yang telah diinputkan oleh user berupa nama_barang, panjang, lebar, tinggi, berat, tujuan, dan jenis. Untuk *field* nomor dan berat_volume terisi oleh aplikasi.

Gambar 3.12 Halaman Hasil Optimasi *Tab* Data Barang

Pada *tab* yang kedua yaitu *tab* posisi barang, aplikasi menampilkan nomor barang, panjang, lebar, tinggi, berat_volume, tujuan, posisi awal dan posisi akhir barang yang tertata dalam kontainer.



Gambar 3.13 Halaman Hasil Optimasi *Tab* Posisi Barang

Pada *tab* yang terakhir yaitu *tab* hasil optimasi, aplikasi menampilkan beberapa informasi diantaranya data sisa barang yang berupa id_barang, panjang, lebar, tinggi, dan tujuan. Lalu *panel* hasil optimasi yang berisikan hasil konsep *filling function* yang paling optimal, jumlah barang tertata dan tak tertata dalam kontainer, total berat barang yang tertata dalam kontainer, volume barang yang terbentuk dalam kontainer, jumlah isi tabu tabel dan isi *tabu list*. Serta terdapat 3 tombol, yaitu :

a. Tombol simulasi

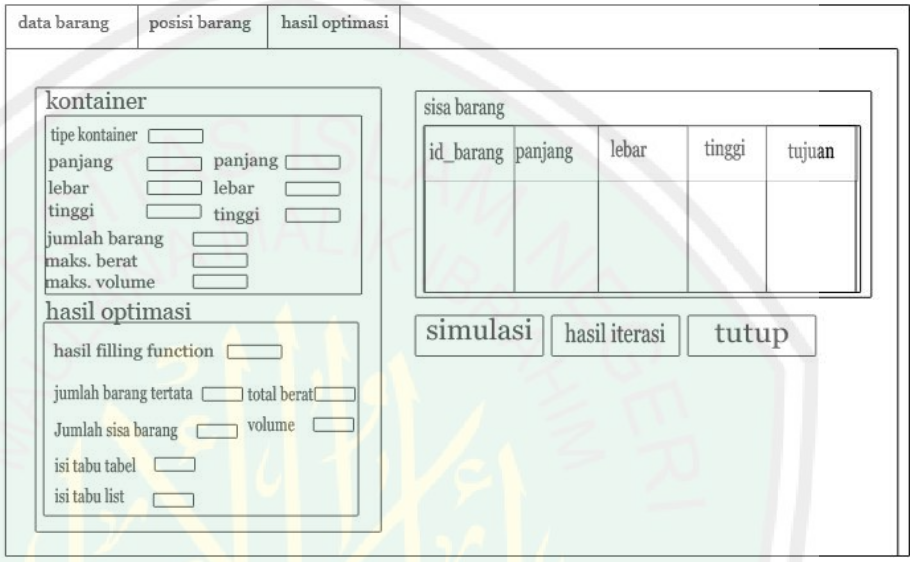
   Untuk menampilkan hasil visualisasi grafik posisi barang yang tertata dalam kontainer.

b. Tombol hasil iterasi

   Menampilkan visualisasi posisi barang berupa grafik berdasarkan iterasi dalam perhitungan algoritma *tabu search*.

c. Tombol tutup

Untuk menutup halaman hasil optimasi dan kembali ke halaman utama.



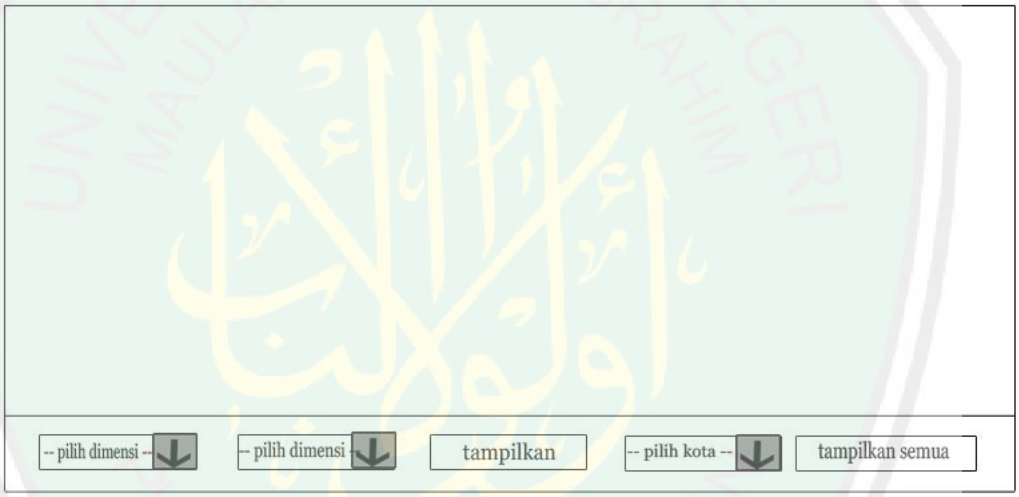
Gambar 3.14 Halaman Hasil Optimasi *Tab* Hasil Optimasi

4. Halaman Visualisasi Grafik Posisi Barang

Aplikasi memvisualisasikan hasil perhitungan optimasi algoritma *tabu search* dalam bentuk grafik. Terdapat 2 *panel*, *panel* grafik dan *panel* menu. Pada *panel* grafik akan muncul hasil penggambaran posisi barang yang digambarkan dengan 2 sumbu yaitu, sumbu x dan sumbu y. Hanya ada 3 penggambaran grafik dari gabungan dimensi barang, panjang dengan lebar, panjang dengan tinggi, dan lebar dengan tinggi.

Pada *panel* menu terdapat beberapa menu diantaranya, menu *combobox* yang pertama digunakan untuk memilih dimensi barang yang akan ditampilkan. Menu *combobox* yang kedua mempunyai

fungsi yang sama dengan menu *combobox* yang pertama. Lalu terdapat sebuah *button* yang digunakan untuk menampilkan grafik berdasarkan menu *combobox* pertama dan kedua. Selanjutnya menu *combobox* yang ketiga digunakan untuk mengetahui posisi titik berdasarkan kota yang dipilih. Yang terakhir berupa *button* "tampilkan semua". *Button* ini digunakan untuk menampilkan semua grafik posisi barang tanpa memilih dimensi mana yang akan ditampilkan.



Gambar 3.15 Halaman Visualisasi Grafik Posisi Barang

5. Halaman Visualisasi Hasil Iterasi

Pada halaman ini aplikasi menampilkan hasil perhitungan konsep *filling function* per-iterasi. Halaman ini digunakan untuk menganalisa mengapa solusi terpilih menjadi solusi optimal dibandingkan dengan kandidat solusi lain.

Aplikasi menampilkan grafik secara keseluruhan dari dimensi barang. Terdapat 2 *panel*, *panel* pertama yaitu *panel* grafik yang

digunakan untuk menampilkan grafik hasil perhitungan konsep *filling function* per-iterasi. *Panel* kedua, *panel* informasi yang digunakan untuk menampilkan informasi bahwa solusi optimal dihasilkan dari salah satu iterasi.

Gambar 3.16 Halaman Visualisasi Hasil Iterasi

6. Halaman Tentang

Halaman tentang berisikan profil singkat dari perusahaan yang dijadikan studi kasus dalam penelitian ini, yaitu PT ANTESS. Selain menampilkan profil perusahaan terkait, dalam halaman ini aplikasi juga menampilkan profil singkat dari pembuat aplikasi.

Gambar 3.17 Halaman Tentang

3.8 Kebutuhan Sistem

Berikut ini merupakan kebutuhan dari perangkat keras dan perangkat lunak yang dibutuhkan untuk pembuatan dan uji coba Aplikasi Optimasi Proses Muat Barang dalam Kontainer.

1. Perangkat Keras (*Hardware*)
   a. PC / Laptop, dengan spesifikasi minimal : *Processor* Intel® Core ™ 2 Duo T 6670 @ 2.20Ghz (2CPUs) dan *Memory* 4096MB RAM
2. Perangkat Lunak (*Software*)
   a. OS Windows 7

      Digunakan untuk pembuatan dan uji coba aplikasi.

   b. Netbeans 7.3.1

      Digunakan untuk pembuatan aplikasi. Melakukan desain dan *coding* aplikasi.

   c. Java Runtime Environment (JRE) dan Java Development Kit (JDK)

Digunakan untuk pembuatan aplikasi dan uji coba aplikasi. JRE menjadi kebutuhan utama jika dalam *PC/Laptop/Notebook* belum terpasang. JDK versi 1.6 digunakan untuk pembuatan aplikasi.

d. AppServer

Digunakan menyimpan database dari hasil perhitungan dan inputan pengguna. Jika dalam *PC/Laptop/Notebook* belum terpasang maka AppServer menjadi prioritas utama setelah JRE untuk kebutuhan aplikasi ini. AppServer yang digunakan versi 2.5.9.

BAB IV

HASIL DAN PEMBAHASAN

Pada bab ini membahas mengenai hasil uji coba Aplikasi Optimasi Proses Muat Barang dalam Kontainer menggunakan Algoritma *Tabu Search* yang telah dirancang dan dibuat. Uji coba dilakukan untuk mengetahui apakah rancangan yang telah dibangun terutama rancangan dari algoritma *tabu search* dalam aplikasi dapat berjalan sebagaimana mestinya dengan lingkungan uji coba yang telah dilakukan.

4.1 Sumber Data

Data diperoleh dengan cara melakukan wawancara dan observasi dengan salah satu pekerja administrasi di PT ANTESS. Data berupa laporan pengiriman barang, data kontainer, dan data data barang yang telah dipilih untuk dijadikan bahan penelitian.

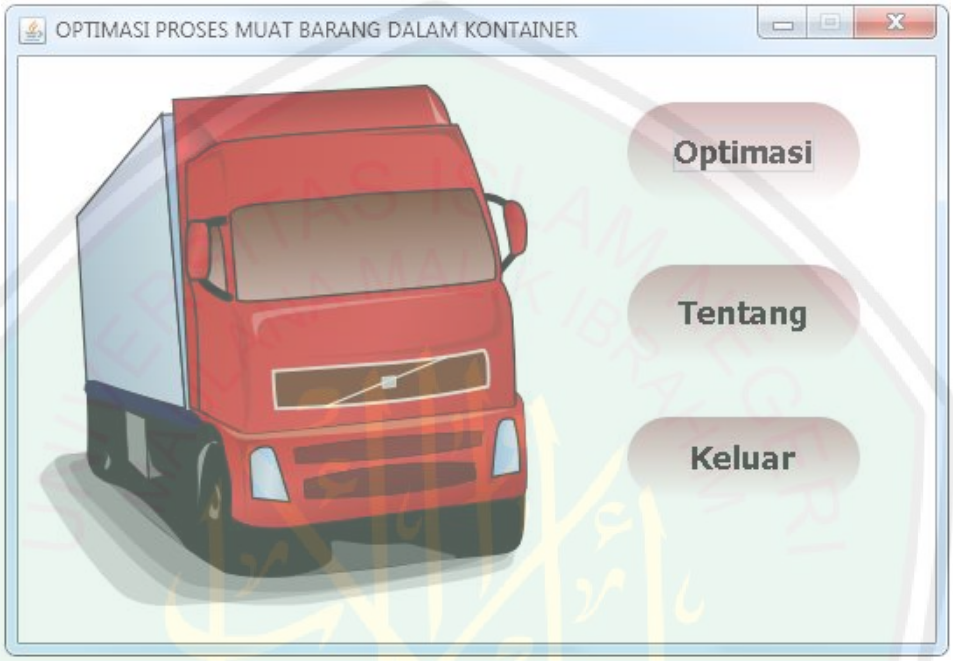
4.2 Implementasi Antarmuka dan Proses

Tahapan ini menjelaskan implementasi antarmuka dan proses yang telah dirancang dan dibangun apakah dapat berjalan sebagaimana mestinya.

4.2.1 Halaman Utama

Halaman utama adalah halaman yang muncul pertama kali bagi pengguna yang menggunakan aplikasi optimasi ini. Setelah dipanggil halaman utama akan muncul menggunakan `animationHide (Animation.`

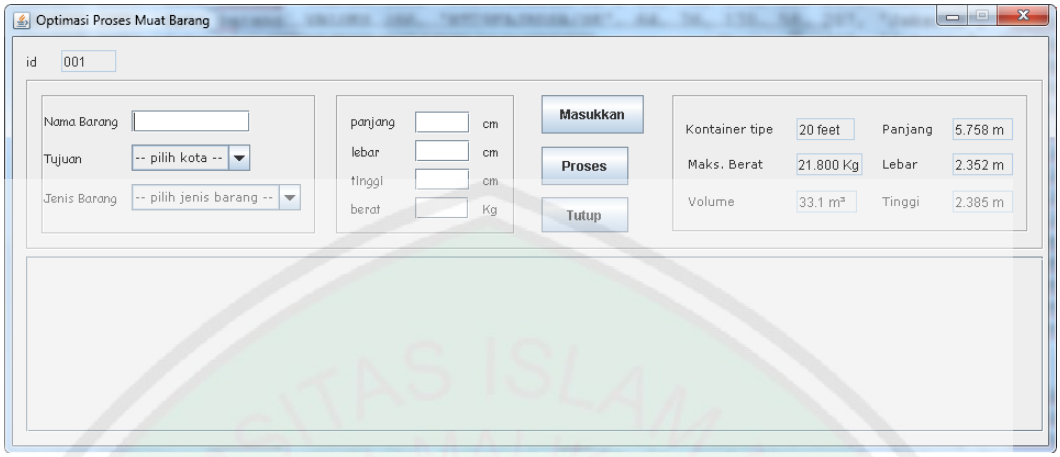
HIDE_TO_BOTTOM). *Button* yang ada pada halaman utama pembuatannya dibantu menggunakan sebuah library dari NetBeans.



Gambar 4.1 Halaman Utama

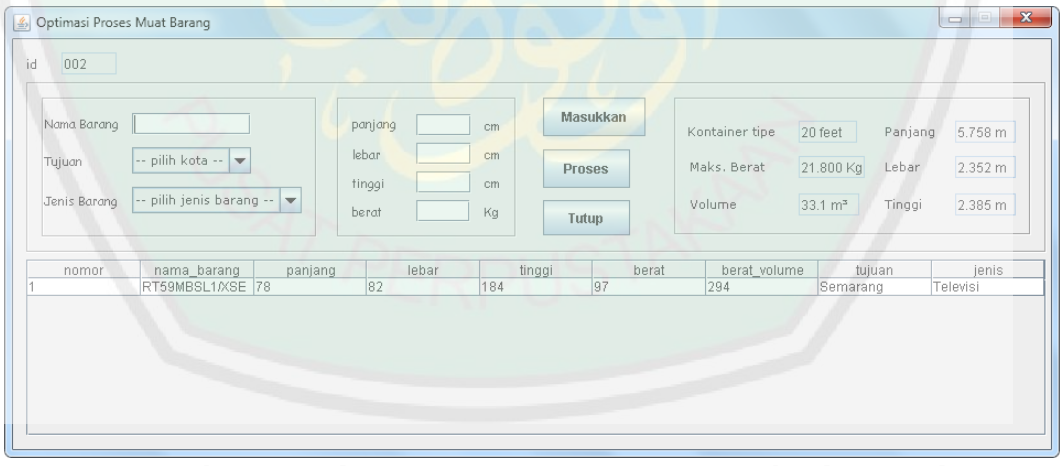
4.2.2 Halaman Pengisian Data Barang

Halaman muncul setelah pengguna mengklik *button* Optimasi pada halaman utama. Halaman ini digunakan untuk mengisi data-data barang yang akan diletakkan dalam kontainer. Data-data itu meliputi nama barang, kota tujuan barang, jenis barang, dan dimensi barang. Pengguna tidak perlu memasukkan identitas dan dimensi dari kontainer yang digunakan. Aplikasi secara otomatis mengisi data kontainer tersebut, karena dalam penelitian ini jenis kontainer yang digunakan hanya 1 jenis.



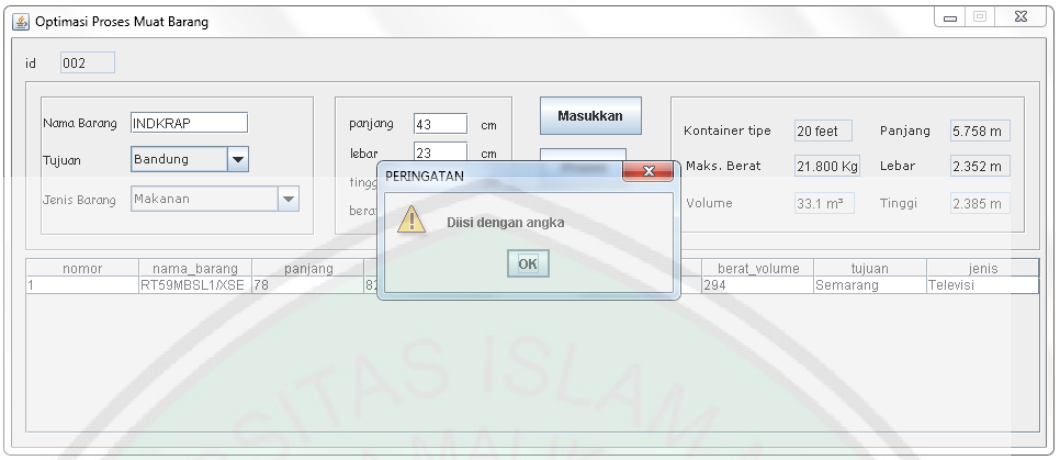
Gambar 4.2 Halaman Pengisian Data Barang

Data yang telah berhasil diinputkan bisa langsung dilihat dalam *panel* tabel dibawah *panel* pengisian data barang. Id barang akan secara otomatis bertambah setelah pengguna berhasil menginputkan barang dan tersimpan dalam database.



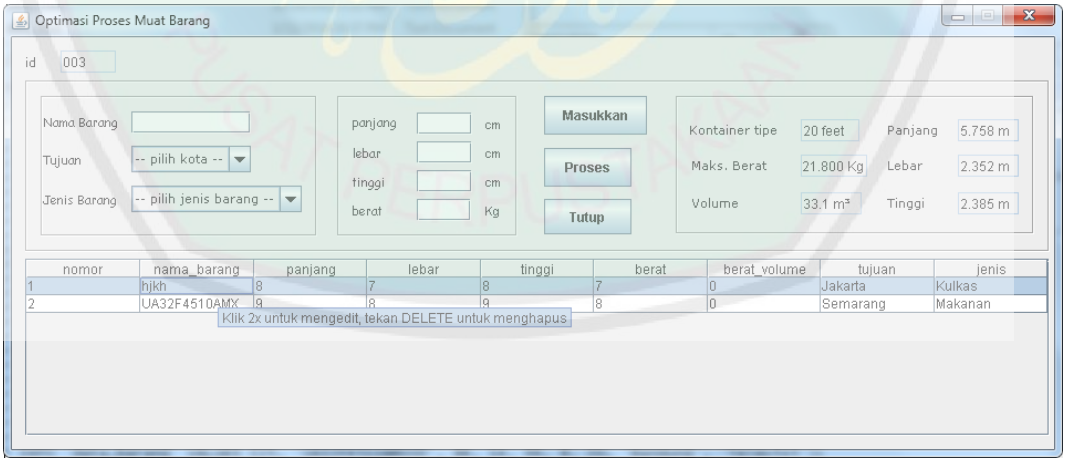
Gambar 4.3 Halaman Pengisian Data Barang Berhasil Diinputkan

Inputan pada *textfield* dimensi barang harus diisi dengan angka, jika pengguna mengisinya dengan huruf, aplikasi tidak akan menerima. Akan muncul sebuah kotak dialog peringatan bahwa *textfield* tersebut hanya bisa diisi oleh angka.



Gambar 4.4 Halaman Pengisian Data Barang Dialog Peringatan

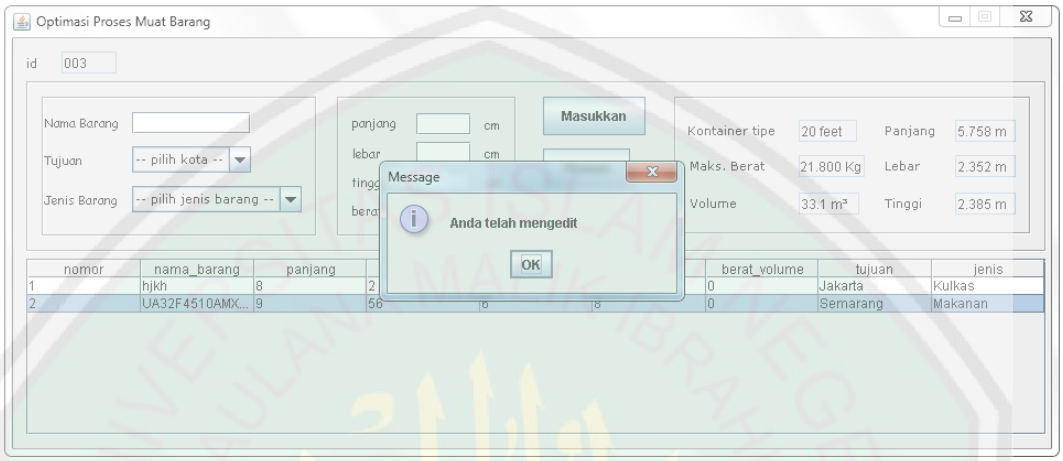
Jika pengguna ingin mengedit dan menghapus data yang telah diinputkan. Pengguna tinggal mengarahkan *pointer mouse* ke baris yang diinginkan dalam tabel akan muncul tulisan "Klik 2x untuk mengedit, tekan *Delete* untuk menghapus". Untuk mengedit tujuan dan jenis barang pengguna akan disediakan menu *combobox* untuk memilih data yang diinginkan.



Gambar 4.5 Halaman Pengisian Data Barang untuk Mengedit dan *Delete*

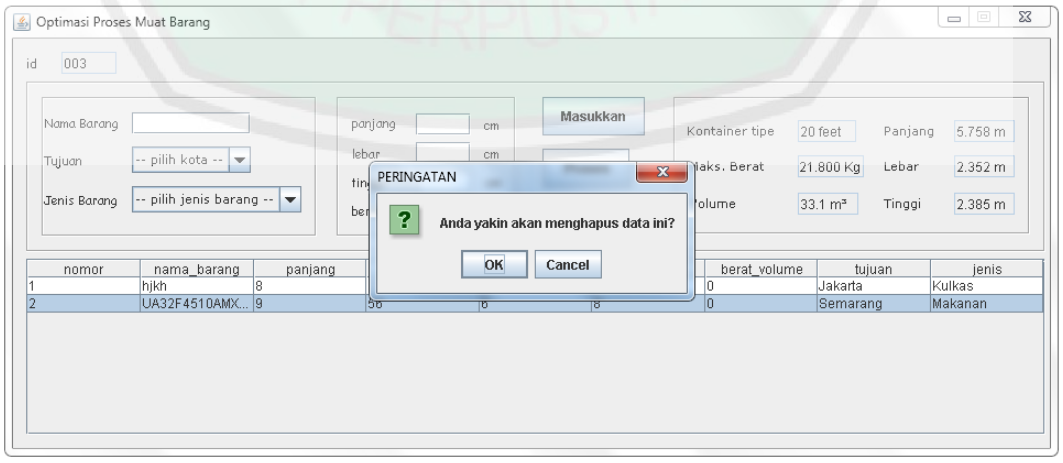
Jika ingin mengedit, pengguna memilih data yang diinginkan lalu klik 2x hingga *cursor* berkedip. Setelah mengedit sesuai dengan yang diinginkan

tekan enter 2x hingga muncul kotak dialog yang memberikan informasi bahwa identitas data telah berhasil diedit.

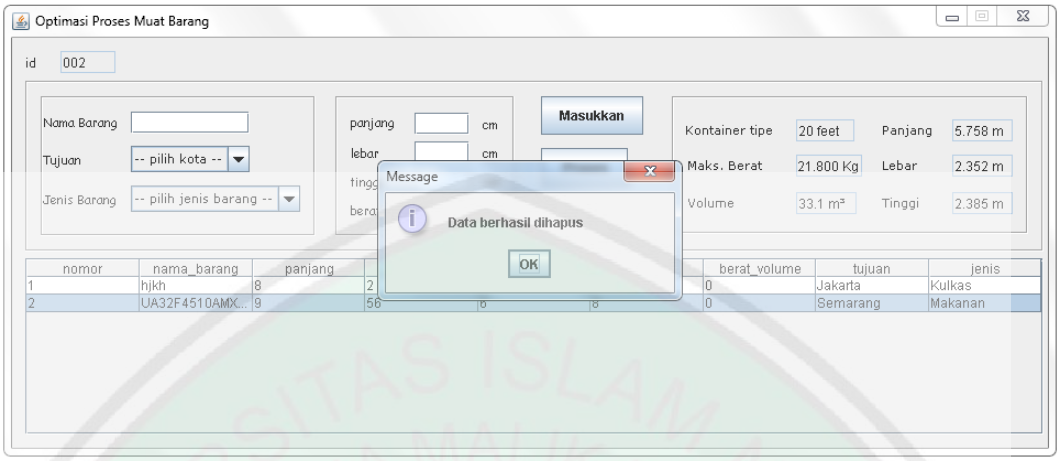


Gambar 4.6 Halaman Pengisian Data Barang Kotak Dialog Mengedit

Jika pengguna ingin menghapus data, maka arahkan *pointer mouse* ke data yang diinginkan lalu tekan dan setelah itu menekan tombol *delete* pada *keyboard*, hingga muncul kotak dialog menghapus data. Setelah pengguna menekan tombol ok pada kotak dialog menghapus data, akan muncul sebuah kotak dialog yang memberikan informasi bahwa data telah berhasil dihapus.



Gambar 4.7 Halaman Pengisian Data Barang Kotak Dialog Menghapus

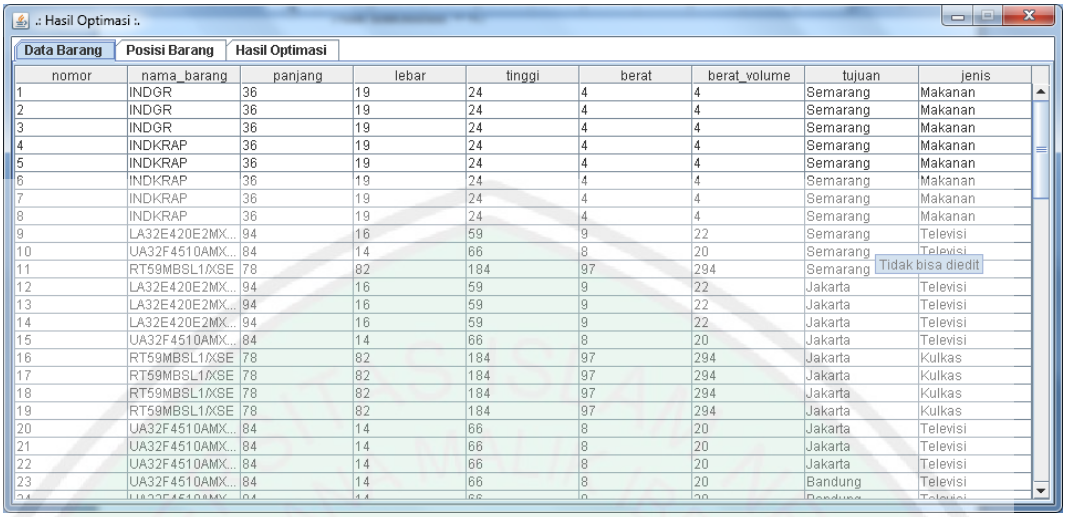


Gambar 4.8 Halaman Pengisian Data Barang Kotak Dialog Menghapus

Selesai memasukkan data yang diperlukan, pengguna menekan tombol proses untuk mengetahui bagaimana hasil aplikasi dalam mengoptimasi proses muat barang dalam kontainer.

4.2.3 Halaman Hasil Optimasi

Hasil perhitungan terdapat pada halaman hasil optimasi. Terdapat 3 *tabbed pane*, yang pertama adalah *tab* data barang. Pada *tab* data barang, terdapat informasi barang yang berupa tabel dengan isian : nomor, nama barang, panjang, lebar, tinggi, berat, berat_volume, tujuan dan jenis. Pada halaman hasil optimasi tidak disediakan fasilitas untuk mengedit dan menghapus data.



nomor	nama_barang	panjang	lebar	tinggi	berat	berat_volume	tujuan	jenis
1	INDGR	36	19	24	4	4	Semarang	Makanan
2	INDGR	36	19	24	4	4	Semarang	Makanan
3	INDGR	36	19	24	4	4	Semarang	Makanan
4	INDKRAP	36	19	24	4	4	Semarang	Makanan
5	INDKRAP	36	19	24	4	4	Semarang	Makanan
6	INDKRAP	36	19	24	4	4	Semarang	Makanan
7	INDKRAP	36	19	24	4	4	Semarang	Makanan
8	INDKRAP	36	19	24	4	4	Semarang	Makanan
9	LA32E420E2MX...	94	16	59	9	22	Semarang	Televisi
10	UA32F4510AMX...	84	14	66	8	20	Semarang	Televisi
11	RT59MBSL1/XSE	78	82	184	97	294	Semarang	Kulkas
12	LA32E420E2MX...	94	16	59	9	22	Jakarta	Televisi
13	LA32E420E2MX...	94	16	59	9	22	Jakarta	Televisi
14	LA32E420E2MX...	94	16	59	9	22	Jakarta	Televisi
15	UA32F4510AMX...	84	14	66	8	20	Jakarta	Televisi
16	RT59MBSL1/XSE	78	82	184	97	294	Jakarta	Kulkas
17	RT59MBSL1/XSE	78	82	184	97	294	Jakarta	Kulkas
18	RT59MBSL1/XSE	78	82	184	97	294	Jakarta	Kulkas
19	RT59MBSL1/XSE	78	82	184	97	294	Jakarta	Kulkas
20	UA32F4510AMX...	84	14	66	8	20	Jakarta	Televisi
21	UA32F4510AMX...	84	14	66	8	20	Jakarta	Televisi
22	UA32F4510AMX...	84	14	66	8	20	Jakarta	Televisi
23	UA32F4510AMX...	84	14	66	8	20	Bandung	Televisi

Gambar 4.9 Halaman Hasil Optimasi *Tab* Data barang

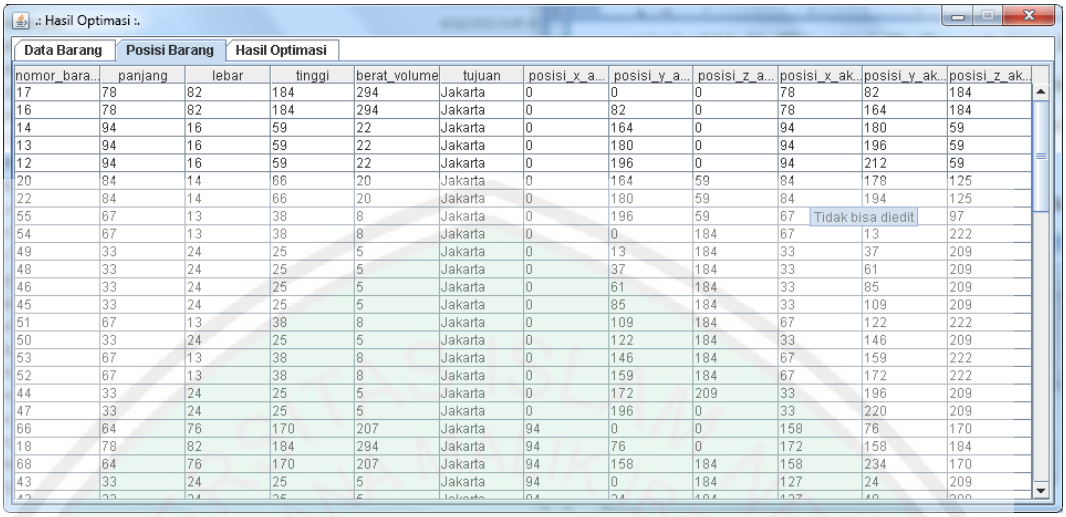
Tab selanjutnya yaitu *tab* posisi barang. Pada *tab* ini terdapat tabel data yang berisikan tentang identitas data beserta posisi barang dalam kontainer. Data yang ditampilkan merupakan solusi yang paling optimal yang dihasilkan dari perhitungan algoritma *tabu search* dengan menggunakan sebuah konsep *filling function.* Posisi barang dalam kontainer ditunjukkan oleh kolom posisi, yaitu :

1. kolom posisi_x_awal : Titik awal barang dalam kontainer dari dimensi panjang
2. kolom posisi_y_awal : Titik awal barang dalam kontainer dari dimensi lebar
3. kolom posisi_z_awal : Titik awal barang dalam kontainer dari dimensi tinggi

4. kolom posisi_x_akhir : Titik akhir barang dalam kontainer dari dimensi panjang
5. kolom posisi_y_akhir : Titik akhir barang dalam kontainer dari dimensi lebar
6. kolom posisi_z_akhir : Titik akhir barang dalam kontainer dari dimensi tinggi

Identitas data barang yang ditampilkan adalah nomor_barang, panjang, lebar, tinggi, berat volume, dan kota tujuan barang. Berat yang digunakan dalam perhitungan optimasi proses muat barang adalah berat volumetrik dari barang. Kolom nomor_barang jika dalam *tab* data barang ditunjukkan oleh kolom nomor.

1. Kolom nomor_barang : Nomor barang
2. Kolom panjang : Panjang barang
3. Kolom lebar : Lebar barang
4. Kolom tinggi : Tinggi barang
5. Kolom berat_volume : Berat volumetrik barang
6. Kolom kota tujuan : Kota tujuan barang

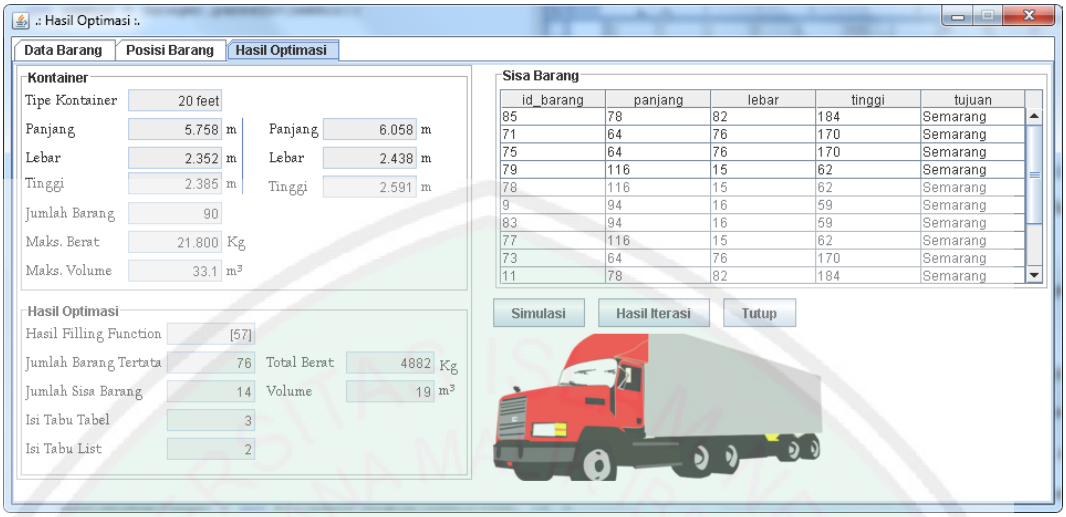


nomor_bara...	panjang	lebar	tinggi	berat_volume	tujuan	posisi_x_a...	posisi_y_a...	posisi_z_a...	posisi_x_ak...	posisi_y_ak...	posisi_z_ak...
17	78	82	184	294	Jakarta	0	0	0	78	82	184
16	78	82	184	294	Jakarta	0	82	0	78	164	184
14	94	16	59	22	Jakarta	0	164	0	94	180	59
13	94	16	59	22	Jakarta	0	180	0	94	196	59
12	94	16	59	22	Jakarta	0	196	0	94	212	59
20	84	14	66	20	Jakarta	0	164	59	84	178	125
22	84	14	66	20	Jakarta	0	180	59	84	194	125
55	67	13	38	8	Jakarta	0	196	59	67	Tidak bisa diedit	97
54	67	13	38	8	Jakarta	0	0	184	67	13	222
49	33	24	25	5	Jakarta	0	13	184	33	37	209
48	33	24	25	5	Jakarta	0	37	184	33	61	209
46	33	24	25	5	Jakarta	0	61	184	33	85	209
45	33	24	25	5	Jakarta	0	85	184	33	109	209
51	67	13	38	8	Jakarta	0	109	184	67	122	222
50	33	24	25	5	Jakarta	0	122	184	33	146	209
53	67	13	38	8	Jakarta	0	146	184	67	159	222
52	67	13	38	8	Jakarta	0	159	184	67	172	222
44	33	24	25	5	Jakarta	0	172	209	33	196	209
47	33	24	25	5	Jakarta	0	196	0	33	220	209
66	64	76	170	207	Jakarta	94	0	0	158	76	170
18	78	82	184	294	Jakarta	94	76	0	172	158	184
68	64	76	170	207	Jakarta	94	158	184	158	234	170
43	33	24	25	5	Jakarta	94	0	184	127	24	209

Gambar 4.10 Halaman Hasil Optimasi *Tab* Posisi Barang

Tab selanjutnya adalah *tab* hasil optimasi. Pada *tab* ini terdapat informasi dari identitas kontainer yang digunakan, jumlah keseluruhan barang, jumlah barang yang tertata dalam kontainer, jumlah sisa barang, hasil *filling function*, jumlah isi tabu tabel, jumlah isi *tabu list*, total berat barang yang tertata dalam kontainer, volume barang dari keseluruhan barang yang tertata dalam kontainer dan tabel yang berisikan identitas barang yang tidak tertata dalam kontainer. Selain itu juga terdapat 3 buah tombol, tombol simulasi, tombol hasil iterasi, dan tombol tutup.

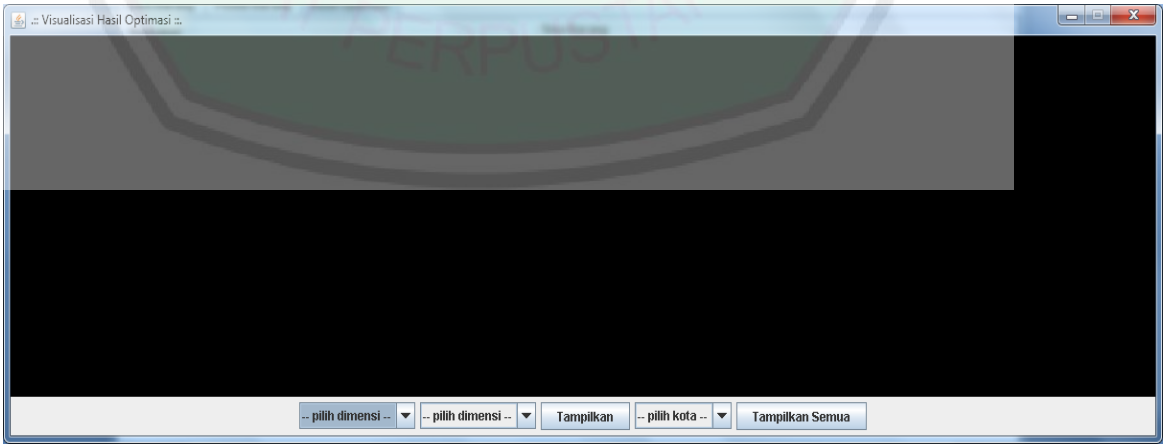
Total berat barang yang tertata dalam kontainer didapatkan dari total berat volumetrik barang yang tertata dalam kontainer. Volume barang didapatkan dengan cara menjumlah keseluruhan volume barang lalu dirubah ke satuan m^3.



Gambar 4.11 Halaman Hasil Optimasi *Tab* Hasil Optimasi

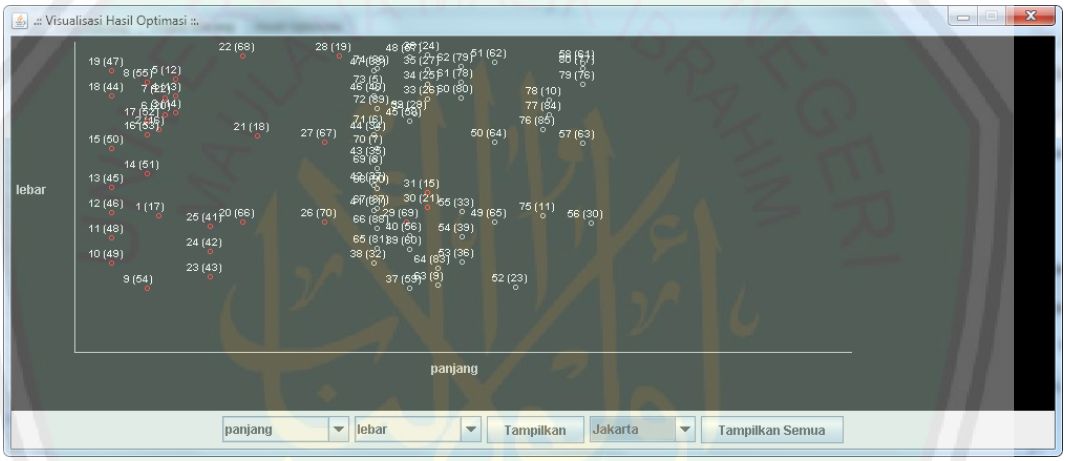
Tombol simulasi digunakan untuk menampilkan gambaran posisi barang dalam kontainer yang ditunjukkan oleh grafik. Terdapat beberapa fasilitas untuk menampilkan grafik tersebut. Tombol hasil iterasi digunakan untuk menampilkan hasil perhitungan konsep *filling function* per iterasi. Tombol tutup digunakan untuk kembali ke halaman utama.

4.2.4 Halaman Visualisasi Hasil Optimasi



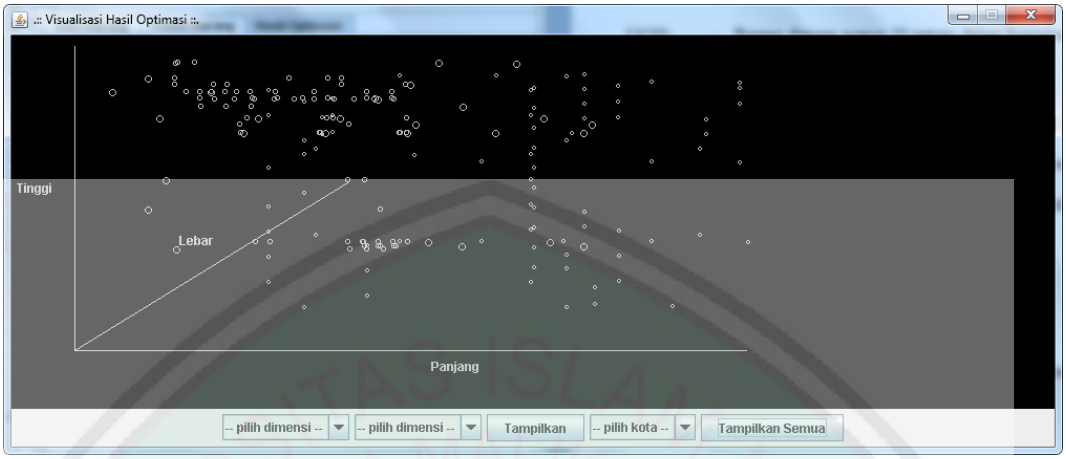
Gambar 4.12 Halaman Visualisasi Hasil Optimasi

Terdapat 2 fasilitas untuk menampilkan grafik hasil optimasi. Grafik pertama adalah dengan cara memilih dimensi barang yang akan ditampilkan. Karena grafik yang ditampilkan hanya mempunyai 2 buah sumbu sehingga pengguna harus memilih 2 dimensi barang yang ditampilkan. Pengguna memilih dimensi dari 2 menu *combobox* pilih dimensi. Setelah memilih pengguna menekan tombol tampilkan.



Gambar 4.13 Halaman Visualisasi Hasil Optimasi Berdasarkan Dimensi

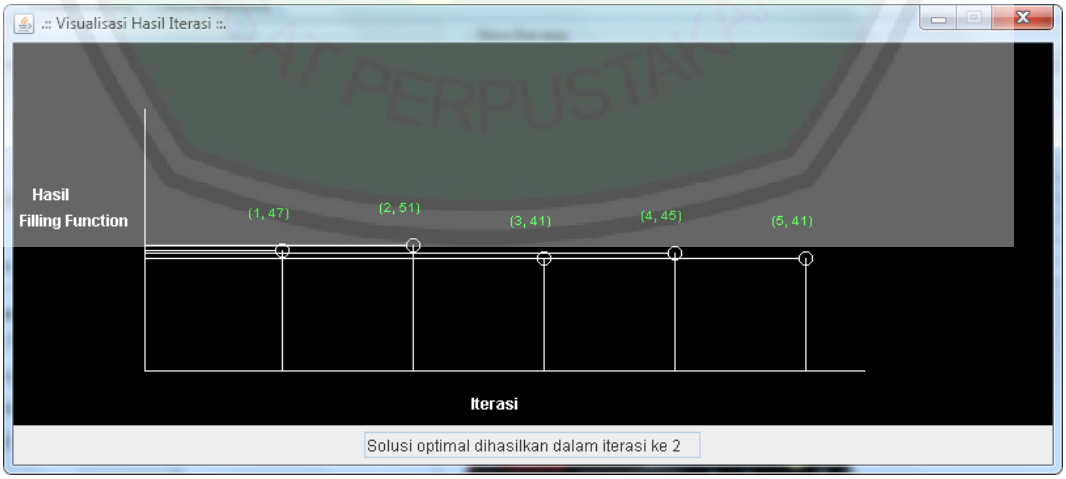
Grafik kedua ditampilkan semua dimensi barang dengan cara menekan tombol tampilkan semua. Untuk mengetahui grafik posisi barang dari tujuan kota tertentu, aplikasi menyediakan menu *combobox* pilih kota. Menu ini hanya akan mengganti warna dari titik-titik dimensi barang.



Gambar 4.14 Halaman Visualisasi Hasil Optimasi Tampilkan Semua

4.2.5 Halaman Visualisasi Hasil Iterasi

Tombol hasil iterasi digunakan untuk menampilkan hasil perhitungan konsep *filling function* per-iterasi yang digambarkan oleh grafik. Dengan sumbu x sebagai urutan iterasi, sumbu y sebagai hasil *filling function.* Dan terdapat sebuah *textfield* yang memberikan informasi bahwa solusi optimal didapat dari salah satu iterasi.



Gambar 4.15 Halaman Visualisasi Hasil Iterasi

4.2.6 Halaman Tentang

Pada halaman tentang terdapat profil singkat dari perusahaan terkait, yaitu PT ANTESS serta profil singkat dari pembuat aplikasi. Pengguna disediakan 2 tombol untuk memilih profil yang ingin ditampilkan.

Gambar 4.16 Halaman Tentang

4.3 Implementasi Algoritma *Tabu Search*

Implementasi algoritma *tabu search* menggunakan konsep *filling function* dilakukan setelah semua barang tertata dalam kontainer. Algoritma *tabu search* yang diimplementasikan tidak menangani proses penataan barang. Proses penataan barang dilakukan dengan berbagai parameter oleh peneliti.

4.3.1 Membandingkan Nilai Filling Function

Data yang diperlukan algoritma *tabu search* untuk melakukan sebuah optimasi proses muat barang dalam kontainer adalah hasil perhitungan dari konsep *filling function* .

$$_{\varphi}(S_i) = \propto \frac{\sum_{j \in S_i} w_j h_j d_j}{W\ H\ D} - \frac{|S_i|}{n}$$

Konsep *filling function* memerlukan dimensi kontainer bagian dalam (panjang, lebar, dan tinggi), dimensi barang, jumlah barang keseluruhan, jumlah barang yang telah tertata dalam kontainer, dan konstanta positif agar nilai yang dihasilkan tidak negatif. Kode program dari konsep ini terdapat dalam *class* tabuSearch.java. Berikut kode dari konsep *filling function* dalam aplikasi optimasi ini :

```
xDimensiKontainer = Math.abs(pKon*lKon*tKon);

xDimensiBarang =
Math.abs(hasilPanjang*hasilLebar*hasilTinggi);

hasil = Math.abs(alpha*((xDimensiBarang/xDimensiKontainer)
- (nBarangTertata/nBarang)));
```

Semua variabel yang digunakan dalam konsep *filling function* bertipe Integer dengan nilai *default*-nya nol (0). Agar hasil dari perhitungan ini selalu bernilai positif ditambahkan `Math.abs` di depan konsep.

Konsep ini berupa void yang nantinya dipanggil dalam `void pencarianNilai.`

```
void pencarianNilai(String nilai, JTextField tf, JTextField
tf2, JTextField tf3, ArrayList<String> ngambil ){

      . . .

tabuSearch ts = new tabuSearch();

ts.fillingFunction(alpha, pkon, lkon, tkon,
hasilFilFuncCurr);

      . . .

}
```

Variabel `alpha` adalah konstanta positif bertipe *Integer*. Selain menggunakan variabel `alpha` juga ditambahkan fungsi `Math.abs` dalam konsep ini agar nilai yang dihasilkan tidak negatif. Variabel `pkon`, `lkon`, dan `tkon` adalah variabel dari dimensi dalam kontainer yang bertipe *Integer*. Sementara untuk `hasilFilFuncCurr` merupakan sebuah *ArrayList* bertipe Integer yang digunakan untuk menampung hasil dari perhitungan konsep *filling function* yang dilakukan dalam void `fillingFunction`.

Dalam `void fillingFunction` terdapat beberapa perintah pemanggilan ke database untuk mendapatkan beberapa data. Data ini digunakan untuk variabel perhitungan pada konsep *filling function.* Perintah pemanggilan tersebut diantaranya :

1. jumlah barang yang telah diinputkan oleh pengguna
2. jumlah barang yang tertata dalam kandidat solusi optimal
3. jumlah keseluruhan dimensi panjang barang dari kandidat solusi optimal
4. jumlah keseluruhan dimensi berat barang dari kandidat solusi optimal

5. jumlah keseluruhan dimensi tinggi barang dari kandidat solusi optimal
6. jumlah keseluruhan berat volume barang dari kandidat solusi optimal

semua pemanggilan ini terdapat dalam kode dibawah ini :

```
String sql4 = "SELECT count(id_barang) as count FROM
data_barang";

ResultSet rs4 = st.executeQuery(sql4);

while(rs4.next()){

      String a = rs4.getString("count");

      nBarang = Integer.parseInt(a);

}

String sq = "SELECT count(id_barang) as idCount FROM
posisi_barang";

ResultSet r = st.executeQuery(sq);

while(r.next()){

       String a = r.getString("idCount");

       nBarangTertata = Integer.parseInt(a);

}

String sql = "SELECT sum(panjang) as Panjang FROM
posisi_barang";

ResultSet rs = st.executeQuery(sql);

while(rs.next()){

       String pjg = rs.getString("Panjang");

       hasilPanjang = Integer.parseInt(pjg);

}
```

```
String sql2 = "SELECT sum(lebar) as Lebar FROM
posisi_barang";

ResultSet rs2 = st.executeQuery(sql2);

while(rs2.next()){

       String lbr = rs2.getString("Lebar");

       hasilLebar = Integer.parseInt(lbr);

}

String sql3 = "SELECT sum(tinggi) as Tinggi FROM
posisi_barang";

ResultSet rs3 = st.executeQuery(sql3);

while(rs3.next()){

       String tnggi = rs3.getString("Tinggi");

       hasilTinggi = Integer.parseInt(tnggi);

}

String sql6 = "SELECT sum(berat_volume) as BeratVolume FROM
posisi_barang";

ResultSet rs6 = st.executeQuery(sql6);

while(rs6.next()){

       String berat = rs6.getString("BeratVolume");

       hasilBerat = Integer.parseInt(berat);

}
```

Ada ketentuan jika pada iterasi awal iterasi ke-1 maka nilai dari perhitungan konsep *filling function* yang dihasilkan dijadikan sebagai nilai maksimum sementara dan solusi yang terbentuk disimpan dalam *tabu list*. Ketentuan ini digunakan agar iterasi selanjutnya bisa menjalankan perintah kriteria aspirasi "apakah hasil konsep *filling function* yang sekarang lebih baik

dari nilai maksimum sementara?". Kode dari ketentuan tersebut dijelaskan dibawah ini :

```
for(int i=0;i<tabuTabel.length;i++){

. . .

if(i==0){

hasilFilFuncMaks.addAll(hasilFilFuncCurr);
hasilFilFuncTabuList.add(hasilFilFuncCurr.get(0));

simpanKeTabuList(stringSolusiCurr, tabuList,
indeksTabuList, tempUkuranTabuList, i, indekIterasi);

indeksTabuTabel++;

indeksTabuList++;

indeksTl++;

}

. . .

}
```

`hasilFilFuncMaks.addAll(hasilFilFuncCurr)` maksudnya adalah hasil dari perhitungan konsep *filling function* sekarang dijadikan sebagai nilai maksimum sementara yang ditampung dalam sebuah *ArrayList* bertipe *Integer* yaitu `hasilFilFuncMaks`. Sementara untuk `hasilFilFuncTabuList` digunakan untuk menyimpan hasil perhitungan konsep *filling function* yang berdasarkan urutan solusi disimpan dalam *tabu list.* Selain itu juga digunakan untuk pemanggilan hasil *filling function* yang didapat dari solusi yang tersimpan dalam *tabu list.*

`Void simpanKeTabuList` digunakan untuk menyimpan solusi pada iterasi awal dalam *tabu list. Tabu list* dalam aplikasi ini merupakan sebuah array bertipe *String*.

```
void simpanKeTabuList(String stringSolusiCurr,
String[]tabuList, int indeksTabuList, ArrayList<String>
tempUkuranTabuList, int indeksIterasi, ArrayList<Integer>
indekIterasi){

        tabuList[indeksTabuList] = stringSolusiCurr;

        tempUkuranTabuList.add(stringSolusiCurr);

        indekIterasi.add(indeksIterasi+1);

    }
```

Void ini membutuhkan solusi yang dihasilkan bertipe *String* yaitu `stringSolusiCurr`. Ini digunakan untuk penyimpanan solusi ke *array tabu list.*

```
String stringSolusiCurr = ""+solusiCurr.toString()+"";
```

Selain itu void ini juga membutuhkan indeks dari *tabu list* dan iterasi yang terjadi. Indeks *tabu list* digunakan untuk urutan penyimpanan solusi sekarang dalam *array tabu list.* Sementara indeks iterasi digunakan untuk urutan iterasi. *ArrayList* `tempUkuranTabuList` bertipe *String* yang digunakan sebagai penyimpanan solusi yang terbentuk dan mengetahui berapa jumlah isi dari *tabu list*. *ArrayList* `indekIterasi` bertipe *Integer* digunakan untuk mengetahui pada iterasi ke berapa solusi yang menghasilkan nilai *filling function* paling optimal. Lalu akan disimpan dalam *ArrayList* lain yang bertipe *Integer*.

```
ArrayList<Integer> indekIterasiSolusiOptimal = new
ArrayList<Integer>();


indekIterasiSolusiOptimal.add(indekIterasi.get(indekItera
si.size()-1));
```

4.3.2 Memeriksa Kriteria Aspirasi

Kriteria aspirasi dalam aplikasi ini adalah "apakah solusi sekarang menghasilkan nilai maksimum (perhitungan konsep *filling function*) yang lebih baik dari nilai maksimum sementara". Kriteria aspirasi dilakukan dengan cara membandingkan *ArrayList* penyimpanan hasil perhitungan *filling function* sekarang, `hasilFilFuncCurr` dengan *ArrayList* penyimpanan hasil perhitungan *filling function* sementara, `hasilFilFuncMaks`.

```
if(hasilFilFuncCurr.get(0) > hasilFilFuncMaks.get(0)){

. . . . . .

}
```

Indeks angka 0 maksudnya adalah mendapatkan nilai pada indeks pertama dari *ArrayList* tersebut. Jika terbukti nilai hasil perhitungan konsep *filling function* dari solusi sekarang lebih baik dari nilai maksimum sementara., maka ganti nilai maksimum sementara dengan hasil perhitungan konsep *filling function* sekarang.

```
hasilFilFuncMaks.removeAll(hasilFilFuncMaks);
hasilFilFuncMaks.addAll(hasilFilFuncCurr);
```

Terdapat perintah `removeAll` yang digunakan untuk menghapus semua isi dari `hasilFilFuncMaks`. Ini berfungsi agar ukuran dari

`hasilFilFuncMaks` selalu satu (1) yang memudahkan dalam pengambilan isinya.

4.3.3 Memeriksa Status Tabu

Setelah mengganti nilai maksimum langkah selanjutnya adalah memeriksa status tabu dari solusi. Pemeriksaan dilakukan agar solusi yang pernah dilakukan dan tersimpan dalam tabu tabel tidak dilanjutkan lagi dan status tabu akan diberikan kepada solusi.

```
periksaTabu( stringSolusiCurr,
stringSolusiCurrTakTersimpan, tabuTabel, tabuList,
tabuListTakTersimpan, indeksTabuList, indeksTabuTabel,
statusTabu,hasilFilFuncTabuList, hasilFilFuncCurr,
hasilFilFuncMaks, tempSolusiCurr,
tempSolusiCurrTakTersimpan, solusiCurr,
solusiCurrTakTersimpan, px_awalTersimpan, py_awalTersimpan,
pz_awalTersimpan, px_awal, py_awal, pz_awal,
px_akhirTersimpan, py_akhirTersimpan, pz_akhirTersimpan,
px_akhir, py_akhir, pz_akhir, tempUkuranTabuList, i,
indekIterasi );
```

`Void periksaTabu` digunakan untuk memeriksa status tabu pada solusi sekarang. Void ini memerlukan :

1. Nomor barang yang tersimpan dari solusi sekarang yang bertipe String, `stringSolusiCurr`. Digunakan sebagai pengambilan identitas barang dari tabel data barang.

2. Nomor barang yang tak tersimpan dari solusi sekarang yang bertipe String, `stringSolusiCurrTaktersimpan`. Digunakan sebagai pengambilan identitas barang dari tabel data barang.
3. Array bertipe String untuk tabu tabel yang digunakan untuk pemeriksaan status tabu, `tabuTabel`.
4. Array bertipe String untuk *tabu list* yang digunakan untuk menyimpan solusi sekarang, `tabuList`. Meskipun status tabu telah diberikan tetapi solusi memenuhi kriteria aspirasi sehingga status tabu dihilangkan.
5. Array bertipe String untuk *tabu list* dari barang yang tak tersimpan, `tabuListTakTersimpan`. Digunakan untuk menyimpan solusi barang yang tak tertata dalam kontainer dari solusi sekarang.
6. `indeksTabuList` dan `indeksTabuTabel` yang digunakan sebagai indeks untuk *array tabu list* dan *array* tabu tabel.
7. Boolean `statusTabu` yang digunakan untuk.memberikan nilai true jika solusi pernah dilakukan. *Default* isian dari `statusTabu` adalah *false*.
8. `hasilFilFuncTabuList`, `hasilFilFuncCurr`, dan `hasilFilFuncMaks` yang digunakan untuk menyimpan hasil perhitungan konsep *filling function.*
9. `tempSolusiCurr` dan `tempSolusiCurrTakTersimpan` yang digunakan untuk tempat menyimpan sementara dari solusi

sekarang barang yang tertata dalam kontainer dan solusi sekarang barang yang tak tertata dalam kontainer.

10. `solusiCurr` digunakan untuk memanggil solusi sekarang barang yang tertata dalam kontainer.
    `solusiCurrTakTersimpan` digunakan untuk memanggil solusi sekarang barang yang tak tersimpan dalam kontainer. `solusiCurr` dan `solusiCurrTakTersimpan` merupakan tempat penyimpanan hasil dari proses penataan barang.
11. `px_awalTersimpan`,`py_awalTersimpan`, `pz_awalTersimpan`,`px_akhirTersimpan`, `py_akhirTersimpan`, dan `pz_akhirTersimpan` digunakan untuk menyimpan posisi barang dari solusi sekarang barang yang tertata dalam kontainer.
12. `px_awal`, `py_awal`, `pz_awal`, `px_akhir`, `py_akhir`, dan `pz_akhir` yang digunakan untuk mendapatkan posisi barang yang tertata dalam kontainer.
13. `tempUkuranTabuList`, `i`, `indekIterasi` digunakan dalam `void simpanKeTabuList`.

Kode dibawah ini adalah kode dari `void periksaTabu` :

```
int tempIndeksTabuTabel = 0;

int a = 0;do{

if(tabuTabel[a].equals(stringSolusiCurr)){

      a=tabuTabel.length;

      statusTabu=true;

      }else{

      a++;

      if(a==tabuTabel.length){

      statusTabu=false;

      }}}while(a<tabuTabel.length);

if(statusTabu==true){

System.out.println("solusi pernah dilakukan, status TABU");

System.out.println("...lakukan pencarian lagi...");

if(hasilFillFuncCurr.get(0) > hasilFillFuncMaks.get(0)){

simpanKeTabuList(stringSolusiCurr, tabuList,
indeksTabuList, tempUkuranTabuList, indeksIterasi,
indekIterasi);

simpanKeTabuTabel(stringSolusiCurr, tabuTabel,
indeksTabuTabel);

tabuListTakTersimpan[indeksTabuList]=
stringSolusiCurrTakTersimpan;

px_awalTersimpan.removeAll(px_awalTersimpan);
py_awalTersimpan.removeAll(py_awalTersimpan);
pz_awalTersimpan.removeAll(pz_awalTersimpan);
px_akhirTersimpan.removeAll(px_akhirTersimpan);
py_akhirTersimpan.removeAll(py_akhirTersimpan);
pz_akhirTersimpan.removeAll(pz_akhirTersimpan);

px_awalTersimpan.addAll(px_awal);
py_awalTersimpan.addAll(py_awal);
pz_awalTersimpan.addAll(pz_awal);
px_akhirTersimpan.addAll(px_akhir);
py_akhirTersimpan.addAll(py_akhir);
pz_akhirTersimpan.addAll(pz_akhir);
```

```
px_awal.removeAll(px_awal);
py_awal.removeAll(py_awal);
pz_awal.removeAll(py_awal);
px_akhir.removeAll(px_akhir);
py_akhir.removeAll(py_akhir);
pz_akhir.removeAll(pz_akhir);

statusTabu = false;

tempIndeksTabuTabel = indeksTabuTabel-1;

indeksTabuTabel = tempIndeksTabuTabel;

System.out.println(indeksTabuTabel);

}else{

System.out.println(" belum pernah dilakukan ");

simpanKeTabuList(stringSolusiCurr, tabuList,
indeksTabuList, tempUkuranTabuList, indeksIterasi,
indekIterasi);

simpanKeTabuTabel(stringSolusiCurr, tabuTabel,
indeksTabuTabel);

tabuListTakTersimpan[indeksTabuList]=
stringSolusiCurrTakTersimpan;

tempSolusiCurrTakTersimpan.removeAll(tempSolusiCurrTakTersi
mpan);

tempSolusiCurr.removeAll(tempSolusiCurr);

px_awalTersimpan.removeAll(px_awalTersimpan);
py_awalTersimpan.removeAll(py_awalTersimpan);
pz_awalTersimpan.removeAll(pz_awalTersimpan);
px_akhirTersimpan.removeAll(px_akhirTersimpan);
py_akhirTersimpan.removeAll(py_akhirTersimpan);
pz_akhirTersimpan.removeAll(pz_akhirTersimpan);

tempSolusiCurr.addAll(solusiCurr);
tempSolusiCurrTakTersimpan.addAll(solusiCurrTakTersimpan);

px_awalTersimpan.addAll(px_awal);
py_awalTersimpan.addAll(py_awal);
pz_awalTersimpan.addAll(pz_awal);
px_akhirTersimpan.addAll(px_akhir);
py_akhirTersimpan.addAll(py_akhir);
pz_akhirTersimpan.addAll(pz_akhir);
```

```
px_awal.removeAll(px_awal);
py_awal.removeAll(py_awal);
pz_awal.removeAll(py_awal);
px_akhir.removeAll(px_akhir);
py_akhir.removeAll(py_akhir);
pz_akhir.removeAll(pz_akhir);

}
```

Terdapat perintah `removeAll` yang digunakan untuk menghapus semua isi dari *ArrayList* agar isinya selalu baru.

Setelah memeriksa status tabu dari solusi sekarang, langkah selanjutnya adalah memasukkan hasil perhitungan konsep *filling function* ke dalam sebuah *ArrayList*, `hasilFilFuncTabuList`. Ini digunakan untuk mengetahui hasil perhitungan konsep *filling function* dari solusi yang tersimpan dalam *tabu list*.

4.3.4 Memeriksa Iterasi

Peneliti menerapkan jumlah yang sama antara jumlah iterasi dengan jumlah tabu tabel. Pada prinsipnya tabu tabel harus menyimpan semua solusi yang telah terbentuk, oleh karena itu peneliti menyamakan jumlahnya.

```
for(int i=0;i<tabuTabel.length;i++){
…. Kode …
}
```

4.3.5 Menghentikan Proses

Pemberhentian proses dilakukan ketika iterasi sudah mencapai batas yang telah ditentukan. Setelah proses pencarian berhenti, yang dilakukan selanjutnya adalah mengambil nilai optimal yang dihasilkan, menyimpan

solusi barang tersimpan dan solusi barang tak tersimpan ke dalam database, dan mengambil informasi pada iterasi ke berapa nilai optimal yang terpilih dihasilkan.

4.3.5.1 Mengambil Nilai Optimal

Digunakan agar pengguna mengetahui hasil perhitungan konsep *filling function* yang paling optimal. Membutuhkan sebuah *ArrayList* baru untuk menyimpan nilai perhitungan yang terpilih.

```
ArrayList<Integer> hasilAkhirFilFuncTabuList = new
ArrayList<Integer>();


hasilAkhirFilFuncTabuList.add(hasilFilFuncTabuList.get(hasi
lFilFuncTabuList.size()-1));
```

Setelah menyimpan, lalu menampilkannya pada *textfield* hasil *filling function* yang berada pada halaman hasil optimasi *tab* hasil optimasi.

```
nilai = hasilAkhirFilFuncTabuList.toString();

tf.setText(nilai);
```

Variabel nilai bertipe *String*.

4.3.5.2 Menyimpan Solusi Tersimpan Ke Dalam Database

Penyimpanan ini digunakan untuk pemodelan hasil optimasi ke dalam grafik. Semua data disimpan dalam tabel posisi_barang_tersimpan. Data yang tersimpan dalam tabel ini adalah data barang yang telah tertata dalam kontainer.

```
for(int i=0;i<tempSolusiCurr.size();i++){

String sql = "INSERT INTO posisi_barang_tersimpan VALUES
('"+tempSolusiCurr.get(i)+"','"+dataPanjangTersimpan.get(i)
+"','"+dataLebarTersimpan.get(i)+"','"+dataTinggiTersimpan.
get(i)+"','"+dataTujuanTersimpan.get(i)+"','"+px_awal.get(i
)+"', '"+py_awal.get(i)+"', '"+pz_awal.get(i)+"',
'"+px_akhir.get(i)+"', '"+py_akhir.get(i)+"',
'"+pz_akhir.get(i)+"','"+dataBeratVolumeTersimpan.get(i)+"'
, '"+dataVolumeTersimpan.get(i)+"')";

st.executeUpdate(sql); }
```

4.3.5.3 Menyimpan Solusi Tak Tersimpan Ke Dalam Database

Penyimpanan ini digunakan agar pengguna mengetahui berapa banyak barang yang tersisa. Semua data disimpan dalam tabel posisi_barang_taktersimpan. Data yang tersimpan dalam tabel ini adalah data barang yang tersisa atau tak tertata dalam kontainer.

```
for(int i=0;i<tempSolusiCurrTakTersimpan.size();i++){

String sql = "INSERT INTO posisi_barang_taktersimpan VALUES
('"+tempSolusiCurrTakTersimpan.get(i)+"','"+dataPanjangTakT
ersimpan.get(i)+"','"+dataLebarTakTersimpan.get(i)+"','"+da
taTinggiTakTersimpan.get(i)+"','"+dataTujuanTakTersimpan.ge
t(i)+"')";

st.executeUpdate(sql); }
```

4.3.5.4 Mengambil Informasi Iterasi

Digunakan agar pengguna dapat mengetahui pada iterasi ke berapa nilai optimal dihasilkan.

```
String ngambil = "Solusi optimal dihasilkan dalam iterasi
ke "+indekIterasi.get(0);

ambil.add(ngambil);
```

IndekIterasi merupakan sebuah *ArrayList* yang berisikan data indek iterasi yang telah masuk dalam *tabu list* dan urutan terakhir dalam *tabu list.*

```
ArrayList<Integer> indekIterasiSolusiOptimal = new
ArrayList<Integer>();


indekIterasiSolusiOptimal.add(indekIterasi.get(indekIterasi
.size()-1));
```

Setelah mengambil data, lalu ditampilkan pada sebuah *textfield* yang terdapat pada halaman visualisasi hasil iterasi.

```
public void ambilStringIterasi(ArrayList<String> ambil){

        String g = String.valueOf(ambil.get(0));

        tf.setText(g);

}
```

`Void ambilStringIterasi` terdapat pada halaman hasil optimasi.

```
hasilIterasi hsl = new hasilIterasi();

hsl.ambilStringIterasi(ngambil);
```

Terdapat variabel ngambil. Variabel `ngambil` merupakan *ArrayList* bertipe *String*, yang digunakan pada `class tabuSearch` dalam `void pencarianNilai`.

```
tabuSearch ts = new tabuSearch();

ts.pencarianNilai(hasilFilling, tfHasilFillFunc,
tfIsiTabuTabel, tfIsiTabuList, ngambil);
```

4.4 Uji Coba Aplikasi

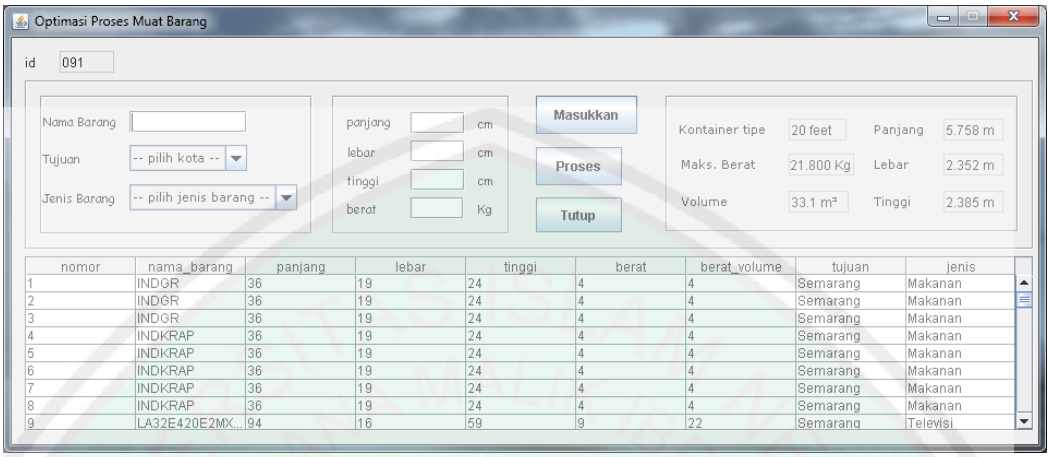
4.4.1 Uji Coba Algoritma *Tabu Search* dalam Optimasi Proses Muat Barang dalam Kontainer

Pengujian aplikasi dilakukan menggunakan perangkat keras *Laptop* Dell Vostro 1088 dengan spesifikasi :

Processor	: Intel ® Core ™ 2 Duo CPU T6670 @ 2.20Ghz (2 CPUs)
Memory	: 4096MB RAM
System Type	: 32-bit Operating System
Operating System	: Windows 7 Ultimate

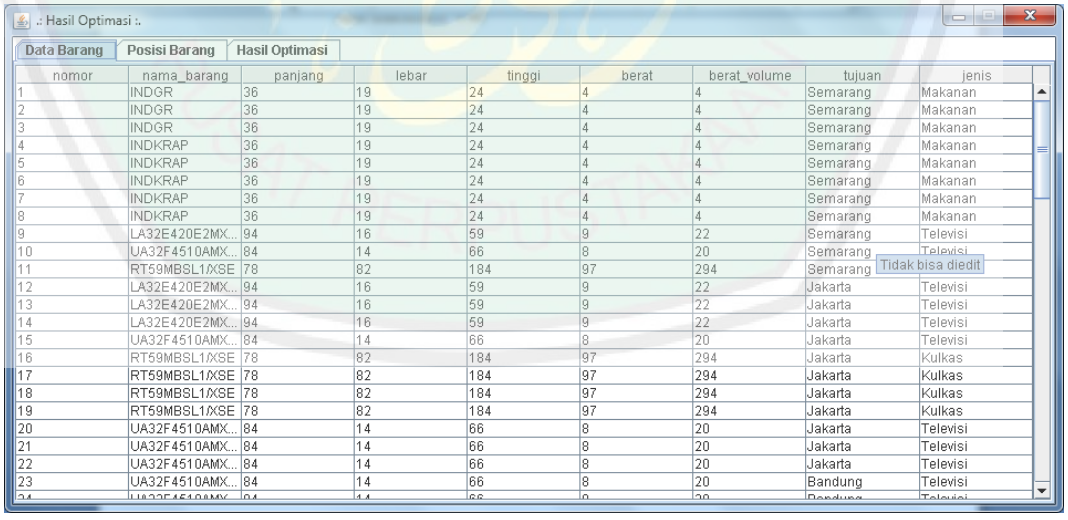
Pengujian ini menggunakan iterasi sebanyak 5 kali, konstanta α = 1, ukuran *tabu list* = 2, nilai maksimum sementara = 0, ukuran tabu tabel = 5, jumlah barang = 90, panjang kontainer = 575, lebar kontainer = 235, tinggi kontainer = 238 dan berat kontainer = 21800.

Memasukkan data identitas barang ke dalam aplikasi sebanyak 90 barang. Jumlah barang dengan tujuan Kota Jakarta = 31, tujuan Kota Bandung = 28, dan tujuan Kota Semarang = 31.



Gambar 4.17 Pengujian Memasukkan Data

Setelah memasukkan data dan menekan tombol proses, optimasi proses muat barang dalam kontainer dilakukan. Aplikasi pertama kali akan menampilkan *tab* data barang, yang berisikan data-data barang yang telah dimasukkan.



Gambar 4.18 Pengujian Hasil Optimasi (*Tab* Data Barang)

Hasil dari optimasi ini pada setiap iterasi :

Iterasi ke-1 :

Jumlah Barang	:	90
Barang yang Tertata	:	79
Barang yang Tak Tertata	:	11
Perkalian Dimensi Barang	:	1510
Perkalian Dimensi Kontainer	:	32
Hasil *Filling Function*	:	47

Nilai maksimum sementara = 0

Nilai maksimum sekarang = 47

Nilai maksimum = 47

Iterasi ke-2 :

Jumlah Barang	:	90
Barang yang Tertata	:	80
Barang yang Tak Tertata	:	10
Perkalian Dimensi Barang	:	1657
Perkalian Dimensi Kontainer	:	32
Hasil *Filling Function*	:	51

Nilai maksimum sementara = 47

Nilai maksimum sekarang = 51

Nilai maksimum = 51

Iterasi ke-3 :

Jumlah Barang	:	90
Barang yang Tertata	:	76
Barang yang Tak Tertata	:	14
Perkalian Dimensi Barang	:	1323
Perkalian Dimensi Kontainer	:	32
Hasil *Filling Function*	:	41

Nilai maksimum sementara = 51

Nilai maksimum sekarang = 41

Nilai maksimum = 51

Iterasi ke-4 :

Jumlah Barang	:	90
Barang yang Tertata	:	78
Barang yang Tak Tertata	:	12

Perkalian Dimensi Barang : 1446

Perkalian Dimensi Kontainer : 32

Hasil *Filling Function* : 45

Nilai maksimum sementara = 51

Nilai maksimum sekarang = 45

Nilai maksimum = 51

Iterasi ke-5 :

Jumlah Barang : 90

Barang yang Tertata : 76

Barang yang Tak Tertata : 14

Perkalian Dimensi Barang : 1323

Perkalian Dimensi Kontainer : 32

Hasil *Filling Function* : 41

Nilai maksimum sementara = 51

Nilai maksimum sekarang = 45

Nilai maksimum = 51

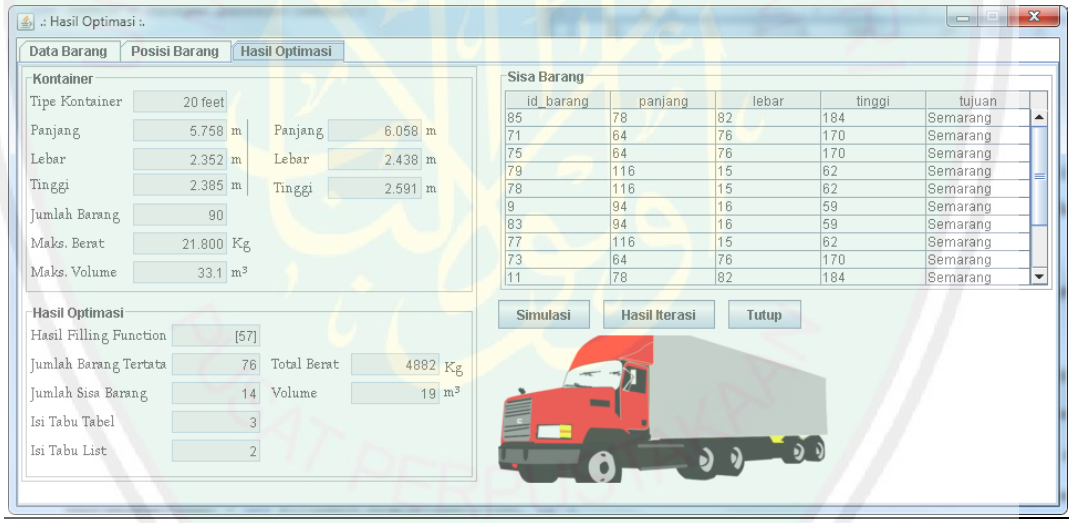
Hasil akhir :

Nilai maksimum optimal : 51

Iterasi ke : 2

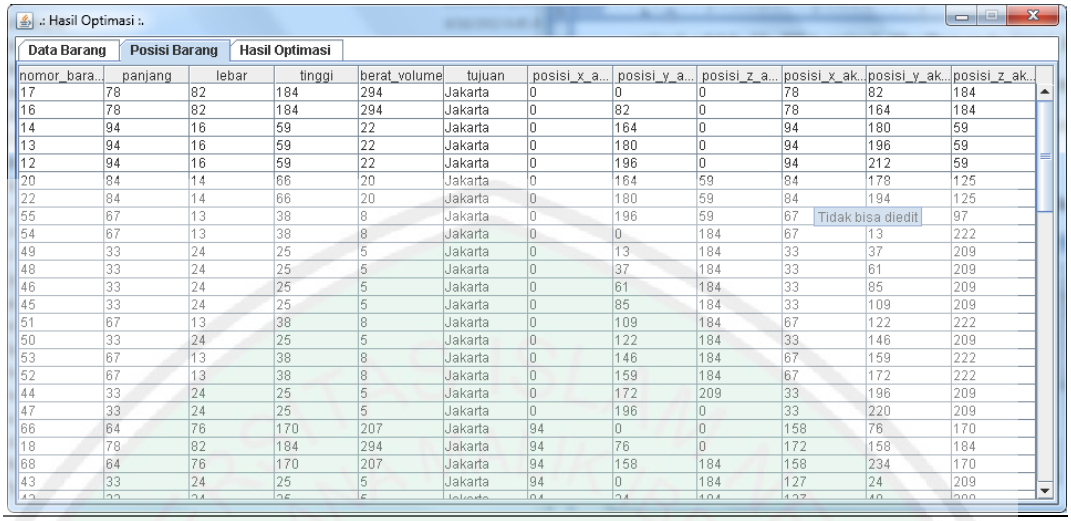
Dari hasil kelima iterasi diatas, didapatkan nilai optimasi yang paling optimal adalah 51. Nilai optimal dihasilkan pada iterasi ke-2 karena hasil perhitungan konsep *filling function* lebih baik ketimbang iterasi lainnya.

Berikut ini adalah tampilan hasil optimasi aplikasi dalam *tab* hasil optimasi.



Gambar 4.19 Pengujian Hasil Optimasi (*Tab* Hasil Optimasi)

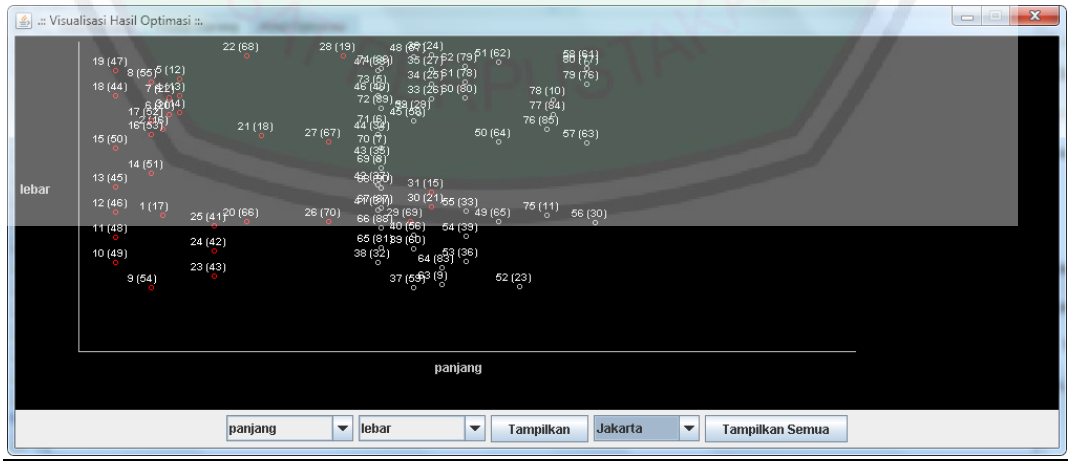
Sementara untuk posisi barang dalam kontainer ditampilkan pada *tab* posisi barang, seperti gambar di bawah ini :



nomor_bara...	panjang	lebar	tinggi	berat_volume	tujuan	posisi_x_a...	posisi_y_a...	posisi_z_a...	posisi_x_ak...	posisi_y_ak...	posisi_z_ak...
17	78	82	184	294	Jakarta	0	0	0	78	82	184
16	78	82	184	294	Jakarta	0	82	0	78	164	184
14	94	16	59	22	Jakarta	0	164	0	94	180	59
13	94	16	59	22	Jakarta	0	180	0	94	196	59
12	94	16	59	22	Jakarta	0	196	0	94	212	59
20	84	14	66	20	Jakarta	0	164	59	84	178	125
22	84	14	66	20	Jakarta	0	180	59	84	194	125
55	67	13	38	8	Jakarta	0	196	59	67	Tidak bisa diedit	97
54	67	13	38	8	Jakarta	0	0	184	67	13	222
49	33	24	25	5	Jakarta	0	13	184	33	37	209
48	33	24	25	5	Jakarta	0	37	184	33	61	209
46	33	24	25	5	Jakarta	0	61	184	33	85	209
45	33	24	25	5	Jakarta	0	85	184	33	109	209
51	67	13	38	8	Jakarta	0	109	184	67	122	222
50	33	24	25	5	Jakarta	0	122	184	33	146	209
53	67	13	38	8	Jakarta	0	146	184	67	159	222
52	67	13	38	8	Jakarta	0	159	184	67	172	222
44	33	24	25	5	Jakarta	0	172	209	33	196	209
47	33	24	25	5	Jakarta	0	196	0	33	220	209
66	64	76	170	207	Jakarta	94	0	0	158	76	170
18	78	82	184	294	Jakarta	94	76	0	172	158	184
68	64	76	170	207	Jakarta	94	158	184	158	234	170
43	33	24	25	5	Jakarta	94	0	184	127	24	209

Gambar 4.20 Pengujian Hasil Optimasi (*Tab* Posisi Barang)

Gambar di bawah merupakan hasil dari visualisasi posisi barang dengan grafik berdasarkan dimensi barang dalam kontainer. Dimensi barang yang dipanggil adalah posisi x akhir, posisi y akhir, dan posisi z akhir barang. X merepresentasikan dimensi panjang barang, Y merepresentasikan dimensi lebar barang, dan Z merepresentasikan tinggi barang.



Gambar 4.21 Pengujian Hasil Optimasi Visualisasi Posisi Barang(panjang, lebar)

Pada gambar diatas menampilkan dimensi panjang yang direpresentasikan oleh sumbu x dan dimensi lebar barang yang direpresentasikan oleh sumbu y. Angka-angka yang tertulis diatas titik sebuah barang adalah angka yang mewakili urutan barang tertata dalam kontainer dan nomor barang.

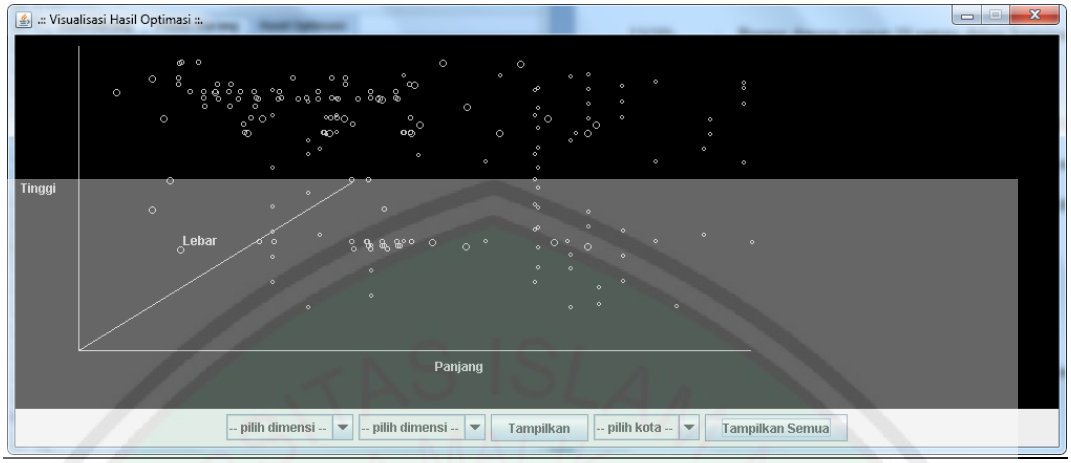
Contoh :

1(17) : Barang dengan nomor 17 tertata dalam kontainer pada urutan ke-1

9(54) : Barang dengan nomor 54 tertata dalam kontainer pada urutan ke-9

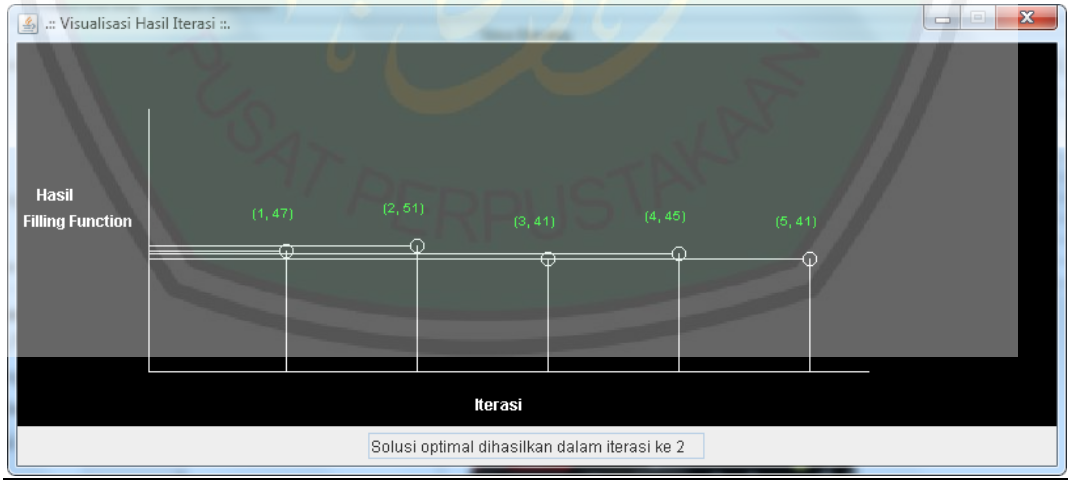
15(50) : Barang dengan nomor 50 tertata dalam kontainer pada urutan ke-15

Dan begitu seterusnya. Pembacaan yang dilakukan pada grafik lainnya juga sama dengan seperti ini, jika pengguna mengganti pilihan dimensi yang akan ditampilkan. Di bawah ini merupakan tampilan setelah pengguna menekan tombol tampilkan semua.



Gambar 4.22 Pengujian Hasil Optimasi Visualisasi Posisi Barang (semua dimensi)

Dan gambar dibawah merupakan visualisasi grafik hasil perhitungan konsep *filling function* dari setiap iterasi yang terjadi. Sumbu X merepresentasikan iterasi, sumbu Y merepresentasikan hasil *filling function.* Dan terdapat informasi bahwa solusi optimal didapat dari iterasi ke-3.



Gambar 4.23 Pengujian Hasil Optimasi Visualisasi Iterasi

4.5 Integrasi Aplikasi Optimasi Proses Muat Barang dalam Kontainer dengan Islam

Optimasi merupakan kegiatan untuk mencari nilai yang maksimal atau nilai yang diinginkan agar mendapatkan hasil yang baik. Sementara optimasi pada proses muat barang adalah kegiatan menata barang yang diinginkan ke dalam kontainer seoptimal mungkin. Optimasi pada proses penataan barang ini agar tidak menambah pengeluaran bagi perusahaan. Penggunaan ruang kosong yang ada harus dimaksimalkan, dan juga pada saat meletakkan barang harus dilakukan secara hati-hati agar tidak berubah bentuk.

Dalam Islam kegiatan optimasi proses muat barang semacam ini mengajarkan kita agar tidak terlalu menghamburkan sesuatu yang dimiliki secara berlebihan. Sesuatu yang dihamburkan akan menjadi mubazir dan manfaatnya tidak akan tampak. Ruang kosong yang tersedia tetapi tidak diisi dengan barang yang cocok dengan ruang kosong tersebut maka hal ini akan menjadi mubazir. Telah dijelaskan dalam Al-Qur'an surat Al-Isro' ayat 26 dan 27.

وَآتِ ذَا الْقُرْبَى حَقَّهُ وَالْمِسْكِينَ وَابْنَ السَّبِيلِ وَلا تُبَذِّرْ تَبْذِيرًا (٢٦)

Artinya : "Dan berikanlah haknya kepada kerabat dekat juga kepada orang miskin dan orang yang dalam perjalanan dan janganlah kamu menghambur-hamburkan (hartamu) secara boros. (26)".

Pada arti karta *"janganlah kamu menghambur-hamburkan(hartamu) secara boros",* jika dipadukan dengan proses muat barang dalam kontainer maksudnya adalah dalam proses penataannya janganlah menghamburkan ruang kosong yang tersedia. Dan juga janganlah menata barang dalam

kontainer secara paksa, maksudnya ketika ruang kosong yang tersedia telah habis lalu barang ditata secara paksa ataupun ruang kosong yang tersedia tidak sesuai dengan ukuran barang yang ada. Bisa saja barang akan berubah bentuk sehingga perusahaan akan mengganti biaya kerusakan jika terjadi hal itu.

Jika ruang kosong yang tersedia telah habis tapi terdapat sisa barang, lebih baik menggunakan kontainer lain dengan syarat mempunyai tujuan terdekat dengan tujuan kota barang yang tersisa. Kegiatan ini bukan termasuk pemborosan, karena barang yang akan ditaruh dalam kontainer lain mempunyai kota tujuan yang sama atau berdekatan dengan barang-barang yang ada dalam kontainer tersebut.

Kegiatan ini memanfaatkan ketersediaan ruang kosong dalam kontainer lain agar tidak mubazir tak terisi barang. Jika kita tetap melakukan penataan barang meskipun tak tersedianya ruang kosong, maka kita akan termasuk saudara setan karena berlebihan menyia-nyiakan ruang kosong yang ada dalam kontainer lainnya. Seperti yang dijelaskan dalam Al-Qur'an Surat Al-Isro' ayat 27 tentang ketidakbaikan orang yang memboroskan sesuatu secara berlebihan.

إِنَّ الْمُبَذِّرِينَ كَانُوا إِخْوَانَ الشَّيَاطِينِ وَكَانَ الشَّيْطَانُ لِرَبِّهِ كَفُورًا (٢٧)

Artinya : *"Sesungguhnya orang-orang yang pemboros itu adalah saudara setan dan setan itu sangat kufur kepada Tuhannya (27)*".

Dalam ayat ini terdapat kandungan bahwa siapa saja orang yang memboroskan sesuatu secara berlebihan akan menjadi saudara setan. Karena setan sangat kufur kepada Tuhannya. Jika saja kita mengabaikan kata-kata

setannya, seseorang yang pemboros adalah seseorang yang telah mengkufurkan Tuhannya. Maka dari itu kita dilarang untuk memboroskan barang secara berlebihan.

Optimasi proses muat barang dalam kontainer jika diintegrasikan dengan Hadits Nabi yang diriwayatkan HR Tirmidzi adalah melakukan sebuah bisnis jasa pengiriman barang haruslah jujur dan dapat dipercaya.

عَنْ أَبِي سَعِيدٍ عَنْ النَّبِيِّ صَلَّى اللَّهُ عَلَيْهِ وَسَلَّمَ قَالَ التَّاجِرُ الصَّدُوقُ الْأَمِينُ مَعَ النَّبِيِّينَ وَالصِّدِّيقِينَ وَالشُّهَدَاءِ

(رواه الترمذي)

Artinya : *"Seorang pebisnis yang jujur lagi amanah, maka ia akan bersama para Nabi, Shiddiqin, dan Syuhada"* [HR Tirmidzi].

Karena kita dipercaya untuk membawa sesuatu yang bukan milik kita. Dan barang yang dibawa keadaannya tetap sama dengan keadaan barang saat awal diterima.

BAB V

PENUTUP

5.1 Kesimpulan

Dari hasil implementasi dan uji coba yang telah dilakukan dapat disimpulkan:

1. Algoritma *Tabu Search* yang digunakan dalam optimasi proses muat barang dalam kontainer dapat diaplikasikan.
2. Aplikasi dapat menghemat waktu dalam proses perhitungan barang yang dapat tertata dalam kontainer jika dibandingkan dengan perhitungan manual yang dilakukan oleh perusahaan terkait, sehingga aplikasi dapat dijadikan bahan pertimbangan proses muat barang dalam kontainer di perusahaan terkait.
3. Aplikasi memberikan informasi tentang koordinat letak barang yang akan ditata dalam kontainer. Sebelum adanya aplikasi, perusahaan tidak bisa mengetahui koordinat letak barang dalam kontainer sebelum proses muat barang dilakukan.

5.2 Saran

Masih banyak kekurangan dalam penelitian optimasi proses muat barang dalam kontainer menggunakan algoritma *tabu search*. Peneliti menyarankan beberapa hal untuk penelitian dan pengembangan selanjutnya, diantaranya :

1. Pemodelan hasil aplikasi untuk posisi barang dalam kontainer tidak hanya berupa grafik, ditambahkan pula dengan pemodelan 3D agar pengguna aplikasi memahami hasil yang didapat dari aplikasi.
2. Posisi barang dapat diubah-ubah saat penataan barang dalam kontainer.
3. Mengembangkan aplikasi optimasi proses muat barang dalam kontainer pada sistem operasi lainnya seperti Linux dan Mac OS.

DAFTAR PUSTAKA

A Carel Lawalata, Herman. 2000. *Tekhnik Operasi Peti Kemas dan Perasuransinya.* Jakarta: Bina Aksara

A Wisniewski, Marco;Ritt, Marcus; Luciana S Buriol. 2011. *A Tabu Search Algortihm for The Capacitated Vehicle Routing Problem with Three-Dimensional Loading Constraints*. Instituto de Informatica - Universidade Federal do Rio Grande do Sul

Berlianty,Intan;Arifin,Miftahol. 2010. *Teknik - Teknik Optimasi Heuristik*. Yogyakarta:Graha Ilmu

Bortfeldt, A; Gehring H. *Applying Tabu Search to Container Loading Problems.* Fern Universität Hagen

Bramantya, Mahisa. 2009. *Sistem Simulasi Penentuan Ukuran Peti Kemas pada Proses Ekspor di PT. Samsung Electronics Indonesia (SEIN).* Bekasi : Sekolah Tinggi Manajemen Informatika dan Komputer Bani Saleh

Fernando, Ray. 2011. *Perancangan Program Aplikasi Optimasi Listrik Pada Industri Plastik Menggunakan Metode Sequential Dynamic Programming.* Jurusan Teknik Informatika dan Matematika, Bina Nusantara.

Gendreau, Michel;Yves P Jean. *Tabu Search.* Departement d'informatique et de recherche operationnelle and Centre de recherche sur les transports^ Universite de Montreal, Canada

Gunadi, Kartika;Kristanto J, Irwan; Beny Hariyanto. 2003. *Optimasi Pola Penyusunan Barang dalam Ruang Tiga Dimensi Menggunakan Metode Genetic Algorithms.* Jurusan Teknik Informatika : Universitas Kristen Petra

Kusumadewi,Sri;Purnomo,Hari. 2005. *Penyelesaian Masalah Optimasi Menggunakan Teknik-Teknik Heuristik.* Yogyakrta : Graha ilmu

Li Pan; Z Huang Joshua; Sydney C.K Chu. 2008. *A Tabu Search Based Algorithm for Cargo Loading Problem.* Department of Mathematics, E-Business Technology Institute - University of Hong Kong, Hong Kong, China

Nur N, M. 2004. *Manajemen Transportasi*. Jakarta:Ghalia Indonesia

Panggabean, Henri P. 2005. *Penjadwalan Job Shop Statik Dengan Algoritma Tabu Search.* Integral, Vol.10 No. 1.

Prasetyaningrum, Ira. *Pengepakan Pallet dalam Kontainer dengan Forklif Menggunakan Metode Algoritma Genetika.* Jurusan Teknik Informatika, Politeknik Elektronika Negeri Surabaya – ITS

Semedi, Bambang. 2012. *Pengenalan Ciri Fisik Peti Kemas.*

Semedi, Bambang. *Indikator Penyalahgunaan Peti Kemas.*

Susilo, Boko;Effendi, Rusdi; Ernawati; Hastini, Safta. 2012. *Rancang Bangun Aplikasi Metode Tabu Search Pada Penyelesain Assignment Problem.* Vol.1 No. 1.

Suyanto. 2010. *Algoritma Optimasi.* Yogyakarta : Graha Ilmu