PEMANFAATAN ALGORITMA GENETIKA UNTUK OPTIMASI 0/1 *MULTI-DIMENSIONAL KNAPSACK PROBLEM* DALAM PENDISTRIBUSIAN PRODUK (STUDI KASUS UD.TOSA)

SKRIPSI

Untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Komputer

Disusun oleh:
Ryan Iriany
NIM: 135150201111106

PROGRAM STUDI TEKNIK INFORMATIKA
JURUSAN TEKNIK INFORMATIKA
FAKULTAS ILMU KOMPUTER
UNIVERSITAS BRAWIJAYA
MALANG
2017

PENGESAHAN

PEMANFAATAN ALGORITMA GENETIKA UNTUK OPTIMASI 0/1 *MULTI-DIMENSIONAL KNAPSACK PROBLEM* DALAM PENDISTRIBUSIAN PRODUK

(STUDI KASUS UD.TOSA)

SKRIPSI

Diajukan untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Komputer

Disusun Oleh:
Ryan Iriany
NIM: 135150201111106

Skripsi ini telah diuji dan dinyatakan lulus pada 20 April 2017
Telah diperiksa dan disetujui oleh:

Dosen Pembimbing I	Dosen Pembimbing II
Agus Wahyu Widodo, S.T, M.Cs	Wayan Firdaus Mahmudy, S.Si, M.T, Ph.D
NIP: 197408052001121001	NIP: 197209191997021001

Mengetahui
Ketua Jurusan Teknik Informatika

Tri Astoto Kurniawan, S.T., M.T., Ph.D
NIP: 197105182003121001

PERNYATAAN ORISINALITAS

Saya menyatakan bahwa penulisan Skripsi ini berdasarkan pemikiran dan pemaparan asli dari saya sendiri. Pada nasksah skripsi ini tidak terdapat karya ilmiah orang lain yang pernah diajukan untuk memperoleh gelar akademik di suatu perguruan tinggi, kecuali karya orang lain yang secara tertulis sudah disitasi dalam naskah ini dan disebutkan dalam daftar.

Demikian pernyataan ini saya buat dengan sebenar-benarnya dan apabila dikemudian hari terbukti terdapat unsur-unsur plagiasi, maka saya bersedia menerima sanksi akademik berupa pencabutan gelar sarjana yang saya peroleh, serta sanksi lain sesuai dengan peraturan yang berlaku di Fakultas Ilmu Komputer, Universitas Brawijaya.

Malang, 20 April 2017

Ryan Iriany
NIM: 135150201111106

KATA PENGANTAR

Bismillahirrahmanirrahim. Segala puji bagi Allah SWT Rabb semesta alam, berkat rahmat dan kasih sayang-Nya sehingga penulis dapat menyelesaikan skripsi ini. Sholawat serta salam selalu tercurah kepada tauladan sepanjang masa, Nabi Muhammad SAW, beserta para keluarga, sahabat, dan para pengikutnya yang senantiasa istiqomah dalam sunnahnya hingga akhir jaman.

Skripsi ini disusun sebagai salah satu syarat memperoleh gelar Sarjana Komputer pada Fakultas Ilmu Komputer Universitas Brawijaya, dengan judul "PEMANFAATAN ALGORITMA GENETIKA UNTUK OPTIMASI 0/1 *MULTI-DIMENSIONAL KNAPSACK PROBLEM* DALAM PENDISTRIBUSIAN PRODUK (STUDI KASUS UD.TOSA)". Penulis menyadari sepenuhnya bahwa begitu banyak pihak yang telah turut membantu dalam penyelesaian skripsi ini. Melalui kesempatan ini, dengan segala kerendahan hati, penulis ingin mengucapkan terima kasih yang sebesar-besarnya kepada :

1. Bapak Agus Wahyu Widodo, S.T, M.Cs selaku Ketua Prodi Informatika Fakultas Ilmu Komputer sekaligus sebagai Pembimbing I, yang telah bersedia meluangkan waktu dan membimbing penulis sehingga mampu menyelesaikan skripsi ini dengan baik.
2. Bapak Wayan Firdaus Mahmudy, S.Si., M.T., Ph.D. selaku Dekan Fakultas Ilmu Komputer sekaligus Pembimbing II, yang telah banyak memberikan bimbingan dan dukungan selama penulis menjadi mahasiswa di Fakultas Ilmu Komputer Universitas Brawijaya, terlebih lagi dalam proses penyusunan skripsi ini.
3. Bapak Tri Astoto Kurniawan, S.T, M.T, Ph.D, selaku ketua jurusan Teknik Informatika Universitas Brawijaya Malang.
4. Bapak Eko setiawan selaku pembimbing akademik yang telah memberikan masukan dan saran kepada penulis selama masa perkuliahan di Fakultas Ilmu Komputer.
5. Ibunda Jarwati dan ayahanda Slamet Aji Sumarah yang tercinta, terima kasih yang tak terhingga atas doa, semangat, kasih sayang, pengorbanan, dan ketulusannya dalam mendampingi penulis. Semoga Allah SWT senantiasa melimpahkan rahmat dan ridho-Nya kepada keduanya. Serta kepada dek Jabat adiku tersayang yang selalu mampu menjadi tempat beristirahat dan melepas penat yang luar biasa dan seluruh saudara-saudaraku yang tidak henti-hentinya memberikan semangat dan dukungannya.
6. Sahabatku tercinta Nanda Putri yang senantiasa menemaniku dalam suka maupun duka sedari awal masuk perkuliahan dan selalu memotivasi untuk segera menyelesaikan skripsi ini. Jazzakillah khoiran katsiran kepada "keluarga keduaku", mama nanda, ayah nanda, dek puput, dan dek sawa, yang selalu menerima penulis dengan hangat dan selalu bisa menciptakan

senyum di tengah-tengah proses “penat” dalam pengerjaan skripsi ini. Keluarga yang selalu mensuport dan memberikan “warna” yang indah dalam perjalanan akhir masa perkuliahan. Fashtabiqul khoirot. Semoga Allah SWT selalu memberikan keistiqomahan kepada kita semua hingga akhir nanti.

7. Untuk Bayu syukron atas kebersamaan dan kesediaanmu menjadi sahabat sedaerah yang telah begitu sering saya repotkan dan selalu menjagaku selama hampir 4 tahun berjuang bersama menempuh studi dikota ini.
8. Untuk sahabat “GESREK” mas bryan, mbak tanti, dan dera terimakasih atas warna-warni kehidupan yang kalian bagi bersamaku, kenangan-kenangan indah yang semoga dapat selalu menjadi penginggat bagi kita untuk bisa kembali bersua.
9. Teman-teman seperjuangan di berbagai kepanitiaan dan organisasi begitu banyak hal berharga yang sudah sama-sama kita lewati selama ini.
10. Teman-teman Informatika ‘13, terima kasih untuk kebersamaannya selama ini dalam perjuangan kita menggapai impian. Apa yang terjadi selama hamper 4 tahun perkuliahan akan selalu menjadi pengalaman yang dikenang.
11. Dan kepada pihak-pihak lain yang telah begitu banyak membantu namun tidak dapat disebutkan satu persatu.

Semoga Allah SWT senantiasa melimpahkan berkah dan rahmat-Nya bagi kita semua, terima kasih untuk bantuannya selama ini, semoga juga dapat menjadi amal ibadah di hadapan-Nya. Amin.

Penulis menyadari bahwa masih banyak terdapat kesalahan dalam penyusunan skripsi ini, oleh sebab itu kritik dan saran yang membangun sangat penulis harapkan guna perbaikan di kemudian hari. Akhir kata, semoga skripsi ini dapat bermanfaat bagi perkembangan ilmu pengetahuan khususnya di bidang teknologi informasi.

Malang, 20 April 2017

Penulis
ryaniriany@gmail.com

ABSTRAK

Sebagai perusahaan *distributor*, biaya distribusi merupakan hal yang sangat berpengaruh terhadap keuntungan yang akan didapatkan UD.TOSA. Hal yang mempengaruhi biaya distribusi selain jarak distribusi adalah frekuensi *maintenance* alat muat yang digunakan dalam proses distribusi. Semakin sering terjadinya kerusakan alat muat maka akan menambah frekuensi *maintenance* sehingga menambah biaya distribusi. Salah satu penyebab kerusakan alat muat adalah karena alat muat sering mengalami kelebihan beban muatan (*overtonase*). Muatan yang berlebihan juga dapat meningkatkan potensi kecelakaan yang dapat mengakibatkan kerusakan produk maupun alat muat itu sendiri. Hal tersebut akan mengakibatkan berkurangnya keuntungan yang didapatkan. Produk yang didistribusikan dan alat muat yang digunakan memiliki karakteristik masing-masing. Setiap alat muat memiliki kapasitas terbatas sehingga tidak semua produk dapat dimuat, *distributor* dapat melakukan kombinasi produk apa saja yang seharusnya dimuat agar dapat memaksimalkan volume muatan tanpa melebihi kapasitas alat muat. Kombinasi produk dalam proses distribusi merupakan masalah kombinatorial yang kompleks, permasalahan kombinasi ini masuk ke dalam *multi-dimensional knapsack problem* (MKDP). Pemanfaatan algoritma genetika dalam *multi-dimensional knapsack problem* adalah untuk menlakukan optimasi daya muat pada proses distribusi. Parameter algoritma yang digunakan pada penelitian ini adalah populasi sebanyak 200, generasi sebanyak 100, nilai cr sebesar 0.9, dan nilai mr sebesar 0.1. Kelebihan beban pada solusi yang dihasilkan oleh sistem adalah sebesar 0% dari kapasitas beban maksimal alat muat. Solusi yang dihasilkan oleh sistem dapat dipastikan tidak melebihi kapasitas, baik dari ruang maksimal alat muat maupun beban maksimal alat muat. Hal ini dapat mengurangi resiko kerusakan alat muat sehingga frekuensi *maintenance* tidak terlalu sering.

Kata kunci: distribusi, *overtonase*, *multidimensional knapsack problem*, algoritma genetika

ABSTRACT

As a distributor company, the cost of distribution is very influential on the benefits to be obtained UD.TOSA. The affect of distribution cost is the distance distribution. Besides affected by distance, cost is also influenced by the frequency of maintenance vehicles used in the distribution process. The more frequent occurrence of damage to the vehicle, it will increase the frequency of maintenance so that adds to the cost of distribution. One cause damage to the vehicle is because vehicles are often excess payload (overtonase). Excessive loads can also increase the potential for accidents that could result in damage to the product as well as the vehicle itself. This will result in reduced profits obtained. Products are distributed and used vehicles have their respective characteristics. Each vehicle has a limited capacity, so not all products can be loaded, the distributor can perform any combination of products that should be loaded in order to maximize cargo volume without exceeding the capacity of the vehicle. The combination of products in the distribution process is a complex combinatorial problems, problems of this combination into the multi-dimensional knapsack problem (MKDP). Utilization of genetic algorithms in multi-dimensional knapsack problem is to perform such optimization of capacity in the distribution process.The algorithm parameters used in this study is a population of 200, the generation of 100, cr by 0.9 and 0.1 mr. Excess load on the solutions produced by the system is equal to 0% of the maximum load capacity of the vehicle. Solutions generated by the system can be ensured not exceed the capacity, both of maximum space vehicles as well as the maximum load of the vehicle. It can reduce the risk of damage to the vehicle so that the frequency of maintenance is not too often.

Keywords: distribution, overtonase, multidimensional knapsack problem, genetic algorithm

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

SOURCE CODE

BAB 1 PENDAHULUAN

1.1 Latar belakang

UD.TOSA merupakan distributor air minum kemasan yang berlokasi di Malang, Jawa Timur. Aktifitas bisnis yang dilakukan sehari-hari adalah distribusi air minum kemasan. Distribusi merupakan kegiatan menyalurkan hasil produksi dari produsen ke konsumen guna memenuhi kebutuhan (Ardiansyah, 2014). Terdapat dua tim yang bertugas dalam distribusi UD.TOSA, yaitu tim *droping* dan tim kanvas. Tim *droping* bertugas melakukan proses *droping*, yaitu proses pengambilan barang dari pabrik produksi untuk didistribusikan ke semua cabang yang dimiliki UD.TOSA, yaitu 3 di kota Malang dan 1 di Blitar. Sedangkan tim kanvas bertugas melakukan distribusi produk dari UD.TOSA langsung ke tangan konsumen. Penelitian ini akan difokuskan pada proses distribusi yang dilakukan oleh tim *droping*. Berbagai produk yang yang didistribusikan oleh UD.TOSA dikirimkan ke seluruh kota Malang. Pendistribusian produk dilakukan dengan menggunakan berbagai transportasi darat. Transportasi yang digunakan sebagai alat muat diantaranya adalah pick up L300, truk sedang, dan truk besar. Produk-produk tersebut diantaranya adalah cleo 115, cleo 550ml, cleo 1500ml, ale-ale, teh gelas, ake gelas, dan lain-lain.

Sebagai perusahaan *distributor*, biaya distribusi merupakan hal yang sangat berpengaruh terhadap keuntungan yang akan didapatkan UD.TOSA. Hal yang mempengaruhi biaya distribusi selain jarak distribusi adalah frekuensi *maintenance* alat muat yang digunakan dalam proses distribusi. Semakin sering terjadinya kerusakan alat muat maka akan menambah frekuensi *maintenance* sehingga mengurangi keuntungan yang akan didapatkan. Salah satu penyebab kerusakan alat muat adalah karena alat muat sering mengalami kelebihan beban muatan (*overtonase*). *Overtonase* juga dapat meningkatkan potensi kecelakaan yang dapat mengakibatkan kerusakan produk maupun alat muat itu sendiri. Hal tersebut mengakibatkan frekuensi *maintenance* meningkat dan biaya distribusi semakin bertambah (Triatmojo, 2017). Oleh karena itu frekuensi *maintenance* harus dijaga agar biaya distribusi tidak semakin bertambah. Setiap alat muat memiliki kapasitas terbatas sehingga tidak semua produk dapat dimuat, *distributor* dapat melakukan pemilihan terhadap produk apa saja yang seharusnya dimuat agar dapat memaksimalkan volume muatan tanpa melebihi kapasitas alat muat.

Setiap produk yang akan didistribusikan memiliki bobot dan volume yang berbeda-beda. Misalnya untuk air minum kemasan *merk* cleo memiliki dua jenis kemasan berbeda yaitu kemasan 600ml dan 1500ml, sehingga ruang yang dibutuhkan juga berbeda. Alat muat yang digunakan juga memiliki karakteristik masing-masing, kapasitas ruang maksimum berbeda-beda pada setiap alat muat. Selain ruang yang terbatas, alat muat juga memiliki kapasitas beban maksimum dalam mengangkut produk. Produk yang didistribusikan oleh UD.TOSA merupakan air minum kemasan yang memiliki berat yang cukup besar dibandingkan dengan produk lain misalnya makanan ringan kemasan, oleh

karena itu perlu diperhatikan mengenai kapasitas beban maksimum dari alat muat. Apabila alat muat tidak mampu menampung beban muatan maka produk tidak dapat diangkut meskipun memiliki sisa ruang yang cukup. Pemanfaatan kapasitas ruang alat muat dalam proses distribusi UD.TOSA sejauh ini sudah cukup baik karena produk yang dimuat tidak pernah melebihi kapasitas ruang. Namun kelebihan beban muatan sering terjadi pada setiap proses distribusi yang dilakukan oleh UD.TOSA. Permasalahan ini adalah permasalahan kombinasi produk untuk optimasi daya muat agar mencapai muatan maksimal tanpa melebihi kapasitas maksimal alat muat dalam pendistribusian produk.

Kombinasi produk dalam proses distribusi merupakan masalah kombinatorial yang kompleks, untuk pemecahan masalah tersebut maka diperlukan perhitungan yang tepat agar kapasitas alat muat yang tersedia dapat dimanfaatkan secara optimal. Dilihat dari karakteristik produk yang akan didistribusikan dan alat muat yang digunakan dalam proses distribusi UD.TOSA, dapat disimpulkan bahwa permasalahan distribusi ini masuk ke dalam *Multi-dimensional Knapsack Problem*. Menurut Gen dan Ceng (2000), *Multi-dimensional Knapsack Problem* (MKDP) sering dikenal sebagai *multi-constraint knapsack problem* karenai memiliki beberapa *constraint* seperti ukuran, konsistensi, dll. MKDP memiliki m knapsack dengan w1,w2,...,wm dan n objek yang memiliki *cost* berbeda-beda (Tyas, Rahman, & Dewi, 2013). *Multi-dimensional knapsack problem* juga dianggap sebagai masalah kombinatorial yang sesuai untuk merepresentasikan permasalahan muatan kargo (Unal, 2013). MKDP yang digunakan dalam penelitian ini adalah 0/1 *multi-dimensional knapsack problem*, dimana dimensi produk yang dimuat tidak bisa dipisahkan sehingga hanya memiliki dua kondisi yaitu dimuat atau tidak dimuat sama sekali. Pemilihan dilakukan dengan memperhatikan beberapa konstrain, seperti berat produk dan volume produk. Produk n akan dimasukkan kedalam knapsack m yang memiliki daya tambung terbatas, dengan konstrain kapasitas ruang maksimum dan beban maksimum.

Untuk penyelesaian permasalahan optimasi *Multi-dimensional knapsack problem* dibutuhkan algoritma yang tepat dengan estimasi waktu yang cepat dan solusi yang terbaik. Salah satu algoritma optimasi yang cukup populer adalah Algoritma Genetika. Algoritma genetika telah diterapkan dalam berbagai masalah optimasi yang memiliki model matematika kompleks dengan ukuran data yang relatif besar dan sulit untuk dibangun (Mahmudy, 2015). Pemanfaatan algoritma genetika untuk optimasi *knapsack problem* telah dibahas dalam beberapa penelitian sebelumnya. Misalnya, penelitian yang membahas penyelesaian 0/1 *multi-dimensional knapsack problem* guna menentukan komposisi makanan sehat yang optimal menggunakan algoritma genetika yang dilakukan oleh Tyas (2013). Penelitian yang serupa juga dilakukan oleh Ariastuti (2015) yaitu pemanfaatan algoritma genetika dan *harmony* search dalam penyelesaian 0/1 *knapsack problem* yang berguna untuk menentukan barang-barang apa saja yang harus dibeli serta melakukan perbandingan kinerja algoritma genetika dan *harmony search*. Dari kedua penelitian diatas, disimpulkan bahwa algoritma genetika cocok digunakan dalam permasalahan

kombinatotial seperti 0/1 *knapsack problem*. Algoritma genetika juga dikatakan lebih baik dari pada algoritma *harmony seach* dalam penyelesaian 0/1 *knapsack problem*.

Berdasarkan permasalahan UD.TOSA dan hasil penelitian yang pernah dilakukan diatas, penulis mengajukan penelitian berjudul "PEMANFAATAN ALGORITMA GENETIKA UNTUK OPTIMASI 0/1 *MULTI-DIMENSIONAL KNAPSACK PROBLEM* DALAM PENDISTRIBUSIAN PRODUK (STUDI KASUS UD TOSA)". Pada penelitian ini algoritma genetika akan digunakan dalam penyelesaian 0-1 *multi-dimensional knapsack problem* untuk optimasi menejemen beban angkut kendaraan agar dapat memaksimalkan ruang tanpa melebihi beban maksimum pada proses disribusi di UD.TOSA.

1.2 Rumusan masalah

Berdasarkan masalah yang telah dijelaskan pada latar belakang, maka berikut merupakan rumusan masalah yang akan dibahas dalam penelitian ini:

1. Bagaimana penerapan algoritma genetika untuk kasus optimasi 0/1 *multi-dimensional knapsack problem* dalam pendistribusian produk?
2. Bagaimana pengaruh parameter algoritma genetika yang digunakan pada kasus optimasi 0/1 *multi-dimensional knapsack problem*?
3. Bagaimana kualitas solusi yang dihasilkan oleh sistem dalam pemanfaatan algoritma genetika untuk optimasi 0/1 *multi-dimensional knapsack problem*?

1.3 Tujuan

Adapun beberapa tujuan penyelesain penelitian ini diantaranya adalah sebagai berikut:

1. Untuk mengetahui bagaimana penerapan algoritma genetika pada kasus optimasi 0-1 *multi-dimensional knapsack problem* dalam pendistribusian produk pada UD.TOSA
2. Untuk mengetahui parameter algoritma genetika dan pengaruhnya jika digunakan pada kasus optimasi 0-1 *multi-dimensional knapsack problem* dalam pendistribusian produk pada UD.TOSA.
3. Untuk mengetahui tingkat kualitas solusi yang dihasilkan oleh sistem menggunakan algoritma genetika.

1.4 Manfaat

Dengan dibuatnya penelitian ini, diharapkan dapat memberikan manfaat sebagai berikut:

1. Bagi penulis

Menerapkan ilmu yang diterima selama proses perkuliahan kedalam sebuah sistem untuk mengatasi masalah tertentu.

2. Bagi Perguruan Tinggi

Sesuai dengan fungsi Perguruan Tinggi yaitu mencetak lulusan terbaik agar bisa memberikan manfaat kepada sesama dari ilmu pengetahuan dan teknologi yang telah diberikan selama di perguruan tinggi.

3. Bagi Masyarakat

Dapat digunakan sebagai sumber ilmu untuk kepentingan dan kemajuan mahasiswa di masa depan bagi yang mendalami cabang ilmu komputasi cerdas kususnya berfokus pada Algoritma Genetika.

4. Bagi UD.TOSA

Dengan sistem yang dibuat ini diharapkan membantu UD.TOSA untuk dapat menjaga kondisi kendaraan agar biaya distribusi tidak semakin bertambah dikarenakan *maintenance* kendaraan yang terlalu sering.

1.5 Batasan masalah

Batasan-batasan masalah yang menjadi batasan dalam pembuatan aplikasi ini yaitu:

1. Penelitian ini digunakan untuk mengoptimalkan permasalahan 0/1 *multidimensional knapsack problem* dalam distribusi produk.
2. Karakteristik yang yang dimiliki setiap produk berbeda-beda, karakteristik tersebut adalah berat dan volume produk.
3. Karakteristik yang yang dimiliki setiap alat muat berbeda-beda, karakteristik tersebut adalah beban maksimal dan ruang maksimal alat muat.
4. Barang hanya meliliki dua kondisi, yaitu dimuat atau tidak dimuat sama sekali.
5. Alat muat dianggap selalu siap untuk melakukan distribusi produk.
6. Pengambilan produk pada satu agen (*supplier* tetap).
7. Produk yang diangkut bermacam-macam pada setiap alat muat.
8. Penelitian ini tidak memperhatikan tata letak produk dalam alat muat.
9. Penelitian ini hanya melakukan optimasi daya muat sehingga tidak memperhatikan target produk.

1.6 Sistematika pembahasan

Berikut merupakan sistematika pembahasan yang akan dilakukan pada penelitian ini:

BAB 1 PENDAHULUAN

Memuat sistematika penulisan dan jadwal pelaksanaan penelitian, latar belekang penelitian, rumusan masalah, tujuan penelitian, manfaat penelitian, batasan penelitian.

BAB 2 LANDASAN KEPUSTAKAAN

Menguraikan teori-teori yang berhubungan dengan distribusi UD.TOSA dan Algoritma Genetika yang digunakan untuk pemecahan masalah yang berhubungan dengan optimasi *knapsack problem*.

BAB 3 METODOLOGI PENELITIAN

Berisi tentang penjelasan penerapan Algoritma Genetika, metode implementasi, eksperimen, desain, survei, analisis, modeling dan langkah-langkah yang dilakukan dalam penelitian untuk optimasi 0/1 *multi-dimensional knapsack problem*.

BAB 4 ANALISIS DAN PERANCANGAN

Berisi analisis kebutuhan dari data yang telah diperoleh untuk keperluan optimasi *knapsack problem* dalam distribusi produk. Pada bab ini juga dibuat pemodelan untuk perancangan sistem optimasi daya angkut distribusi produk guna memaksimalkan volume muatan serta dibuat perancangan pengujian yang akan dilakukan pada sistem.

BAB 5 IMPLEMENTASI

Berisi implementasi perangkat lunak yang dibangun untuk keperluan optimasi 0/1 *multi-dimensional knapsack problem* dalam distribusi produk menggunakan algoritma genetika.

BAB 6 PENGUJIAN DAN ANALISIS

Berisi tentang pengujian parameter algoritma genetika dan analisa hasil pengujian berupa penggunaan populasi, kombinasi *crossover* dan mutasi, serta generasi pada sistem optimasi 0/1 *multidimensional knapsack problem* dalam distribusi produk menggunakan algoritma genetika.

BAB 7 KESIMPULAN DAN SARAN

Berisi kesimpulan dari penelitian yang dilakukan dan saran untuk pengembangan sistem yang lebih baik yang akan dilakukan di masa depan yang diambil dari kesimpulan yang diperoleh pada penelitian.

BAB 2 LANDASAN KEPUSTAKAAN

Pada bab ini berisi tinjauan pustaka dan dasar teori yang didigunakan dalam melakukan penelitian ini. Teori-teori yang berhubungan dengan penelitian yang dilakukan dibahas dalam poin dasar teori. Dasar teori yang akan dibahas dalam penelitian ini adalah mengenai *knapsack problem*, algoritma genetika, proses distribusi, dan pengaruh *overtonase*.

2.1 Tinjauan Pustaka

Tinjauan pustaka yang dilakukan mengarah pada 2 buah penelitian yang mengangkat topik mengenai optimasi 0/1 *knapsack propblem* menggunakan Algoritma Genetik. Penelitian pertama adalah "Implemetasi Algoritma Genetika Untuk Optimasi 0/1 *Multi-Dimensional Knapsack Problem* Dalam Penentuan Menu Makanan Sehat". Pada penelitian tersebut dihasilkan kesimpulan bahwa algoritma genetika cocok digunakan dalam permasalahan 0/1 *multi-dimensional knapsack problem* dengan individu terbaik didapat pada Pc 70% dan Pm 50%, dengan generasi sebanyak 100 (Tyas, Rahman, & Dewi, 2013). Kemudian penelitian kedua dilakukan oleh Putri Rukmana Ariastuti yang berjudul "Penerapan Algoritma Genetika dan Algoritma *Harmony Search* pada Permasalahan *KNAPSACK* 0-1". Pada penelitian tersebut diperoleh kesimpulan bahwa algoritma genetika dapat memberikan profit yang setabil daripada *harmony search*. Dilihat dari segi konvergensinya disimpulkan bahwa algoritma genetika bekerja lebih baik dalam permasalahan *knapsack* 0-1 (Ariastuti, 2015).

Pada Tabel 2.1 menunjukan beberapa studi literatur yang digunakan penulis untuk menunjang penelitian.

Tabel 2.1 Studi Literatur

No	Judul	Tujuan Penelitian	Algoritma	Hasil
1	Implemetasi Algoritma Genetika Untuk Optimasi 0/1 *Multi-Dimensional Knapsack Problem* Dalam Penentuan Menu Makanan Sehat (Tyas, Rahman, & Dewi, 2013)	Pemanfaatan lagoritma genetika untuk optimasi dalam penentuan menu makanan sehat	Algoritma Genetika	Algoritma genetika cocok digunakan dalam permasalahan 0/1 Multi-Dimensional Knapsack Problem dengan individu terbaik didapat pada Pc 70% dan Pm 50%, generasi 100

Tabel 2.1 (lanjutan) Studi Literatur

No	Judul	Tujuan Penelitian	Algoritma	Hasil
2	Penerapan Algoritma Genetika dan Algoritma *Harmony Search* pada Permasalahan *KNAPSACK* 0-1 (Ariastuti, 2015)	Membandingkan kinerja *Harmony Search* dan Algoritma Genetika dalam optimasi 0/1 *Multi-Dimensional Knapsack Problem*	Algoritma genetika dan *Harmony Search*	Algoritma genetika dapat memberikan profit yang stabil daripada *harmony search*. Dilihat dari segi konvergensinya disimpulkan bahwa algoritma genetika bekerja lebih baik dalam permasalahan knapsack 0-1
3	*A Genetic Algorithm for the Multiple Knapsack Problem in Dynamic Environment* (Unal, 2013)	Membandingkan pendekatan *Random Imigrants Based GA* (RIGA) dan *Memory Based GA* (MBGA)	Algoritma Genetika	menunjukkan bahwa MBGA lebih efektif dari pada RIGA untuk 0/1 *Knapsack Problem*
4	*Meta-heuristics for Multidimensional Knapsack Problems* (Mian, 2012)	Penggunaan teknik *constraint handling* serta membandingkan penggunaan operator dan parameter algoritma genetik	Algoritma Genetika	Menunjukan bahwa algoritma genetika mampu memecahkan masalah MKP dan masalah *capital budgeting*
5	*Genetic Algorithms With Decomposition Procedures for Multidimensional 0-1 Knapsack Problems With Block Angular Structures* (Kato, 2003)	Pemanfaatan *block angular struktur* yaitu *block constraint* dan *coupling constraint* untuk menghasilkan solusi yang lebih potensial dan memuaskan	Algoritma genetika	Menunjukan bahwa penggunaan *block angular structure* dalam algoritma genetika menghasilkan individu potensial.

Penelitian ini membahas mengenai optimasi 0/1 *Multi-Dimensional Knapsack Problem* dalam pendistribusian produk. Penelitian ini bertujuan melakukan kombnasi produk untuk optimasi 0/1 *Multi-Dimensional Knapsack Problem* dengan memaksimalkan muatan tanpa melebihi kapasitas beban maksimal guna menjaga frekuensi *maintenance* alat muat.

2.2 Knapsack Problem

Knapsack problem merupakan salah satu permasalahan optimasi kombinatorik yang banyak dipelajari baik secara teoritis ataupun praktis (Aristoteles, 2015). *Knapsack problem* adalah permasalahan pemilihan objek dari beberapa objek yang tersedia, kemudian objek tersebut akan disimpan atau dimasukkan kedalam sebuah tas atau wadah terbatas. Dengan memperhatikan karakteristik beberapa objek tersebut dimana setiap objek memiliki bobot, volume, dan nilai profit yang berbeda serta memperhatikan kapasitas dari media penyimpanan diharapkan diperoleh suatu penyimpanan yang optimal (Ariastuti, 2015). *Knapsack problem* dapat dijumpai pada bidang jasa distribusi barang seperti pada perusahaan distributor atau pengiriman peti kemas melalui pelabuhan. Dalam proses distribusi diharapkan keuntungan yang maksimal untuk mengangkut semua barang yang tersedia tanpa melebihi kapasitas yang ada. Dari persoalan diatas diharapkan sebuah solusi yang dapat mengatasi masalah tersebut secara optimal (Dimyanti, 2004).

2.2.1 Multiple Dimensional Knapsack Problem

Multiple Dimensional Knapsak Problem (MKDP) merupakan permasalahan *knapsack* yang memiliki beberapa batasan (*constraint*), misalnya berat, volume, harga, rasa, profit dan lain-lain. MKDP mempunyai pendekatan dengan algoritma genetika sebagai solusi optimasinya seperti halnya permasalahan kombinatorial yang lainnya (Tyas, Rahman, & Dewi, 2013). Berikut merupakan formulasi *multiple dimensional knapsack problem* menurut Gen & Cheng (2000):

$$\max \sum_{j=1}^{ni} C_j X_j \tag{2.1}$$

$$\text{s.t} \sum_{j=1}^{ni} W_{ij} X_{ij} \leq W \tag{2.2}$$

$$X_{ij} \in \{0,1\} \tag{2.3}$$

Dimana:

j = indeks item

ni = jumlah item di seluruh kelas

c_j = *cost* item j

W_{ij} = berat item j

X_{ij} = solusi yang mungkin

W = berat maksimas wadah (*knapsack*)

Pada *Multi-Dimensional Knapsack Problem* terdapat beberapa kapasitas *knapsack*: W1, W2, W3,..,Wn serta terdapat n objek yang masing-masing memiliki *cost* yang berbeda (cj), j = 1,2,..,n (Tyas, Rahman, & Dewi, 2013).

2.2.2 Jenis Knapsack problem

Menurut Martello (2006), *knapsack* terdiri dari beberapa permasalahan yang bisa diselesaikan, berikut merupakan beberapa jenis permasalahan *knapsack* tersebut:

1. *Integer Knapsack* (0-1 *Knapsack*)
   Pilihan yang dimiliki objek hanya dua, dimensinya dimasukkan semua atau tidak dimasukkan sama sekali kedalam *knapsack*.
2. *Bounded Knapsack* (*Knapsack* Terbatas)
   Permasalahan dimana bagian objek bisa dipisahkan, yaitu dimensi objek bisa dimasukkan sebagian atau seluruhnya ke dalam *knapsack*.
3. *Unbounded Knapsack* (*Knapsack* Tak Terbatas)
   Yaitu objek yang akan dimasukkan ke dalam *knapsack* dimana *knapsack* memiliki kapasitas tidak terbatas.

Pada penelitian ini knapsak problem yang digunakan adalah *integer knapsack* dimana solusi yang dihasilkan harus bernilai bulat yaitu nol atau satu (Ariastuti, 2015). Hal tersebut menggambarkan kondisi yang dialami produk, yaitu produk tidak bisa dipisahkan, hanya ada dua kondisi yaitu *take it or leave it*.

2.3 Algoritma Genetika

Algoritma Genetika merupakan algoritma bagian dari algoritma evolusi yang sangat populer. Algoritma Genetika sering diterapkan pada penyelesaian kasus-kasus optimasi. Hal ini dikarenakan kemampuan Algoritma Genetika dalam menyelesaikan kasus optimasi yang memiliki data besar dan komplek serta model matematika yang sulit untuk dibangun. Penerapan Algoritma Genetika diantaranya adalah untuk optimasi penjadawalan, kompresi citra, alokasi ruang, distribusi produk, persediaan barang, dan penyusunan rute serta maih banyak kasus lainnya (Mahmudy, 2015).

Algoritma Genetika merupakan algoritma yang memanfaatkan proses seleksi alamiah sehingga dikenal sebagai proses evolusi. Seleksi alam yang dimaksud yaitu proses penyesuaian gen secara terus-menerus dengan lingkungan hidupnya sehingga pada akhirnya hanya individu-individu yang kuat yang mampu bertahan. Dalam Algoritma Genetika, permasalahan direpresentasikan kedalam bentuk kromosom. Terdapat tiga aspek pendukung dalam kinerja Algoritma Genetika yaitu nilai fitness, representasi genetika, dan operasi genetika (Nugraha & Mahmudy, 2015).

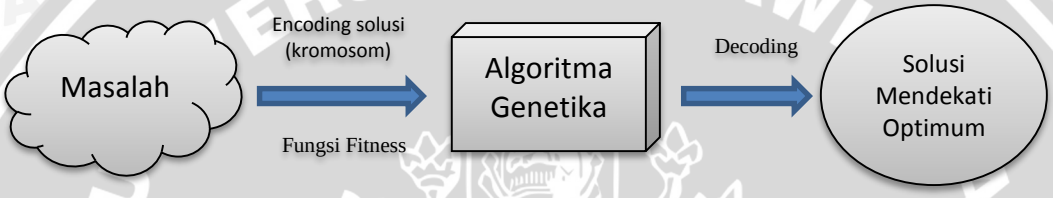
2.3.1 Proses Algoritma Genetika

Dalam proses penyelesaian suatu masalah menggunakan Algoritma Genetika dibutuhkan alur yang lengkap. Proses yang diterapkan dalam

penyelesaian masalah hingga nantinya mendekati solusi optimum dapat digambarkan pada Gambar 2.1.

Dari Gambar 2.1 dapat dipahami bahwa langkah awal yang harus dilakukan adalah meng-*encoding* suatu masalah menjadi *string* kromosom. *String* kromosom merupakan susunan gen-gen yang merepresentasikan variabel keputusan yang digunakakan dalam pencarian solusi. Dalam *string* kromosom disertai dengan nilai *fitness* yang berfungsi untuk mengukur seberapa baik kromosom yang akan diproses ke algoritma genetika. Prosedur yang dijalankan dalam algoritma genetika dapat dilihat pada Gambar 2.2.

Keterangan: *P(t)* : populasi *parents*
C(t) : populasi *offspring* (anak)



Gambar 2.1 Proses Mencari Solusi dengan Algoritma Genetika
Sumber: (Mahmudy, 2015)

```
procedure AlgoritmaGenetika
begin
  t = 0
  inisialisasi P(t)
  while (bukan kondisi berhenti) do
      reproduksi C(t) dari P(t)
      evaluasi P(t) dan C(t)
      seleksi P(t+1) dari P(t) dan C(t)
      t = t + 1
  end while
end
```

Gambar 2.2 Alur Kerja Algoritma Genetika

Algoritma genetika diawali dengan penciptaan individu secara acak yang memiliki susunan kromosom tertentu. Kemudian dilakukan reproduksi untuk menghasilkan keturunan (*offspring*) dari individu-individu yang terdapat dalam populasi. Setelah itu dilakukan penghitungan nilai *fitness* pada setiap kromosom pada proses evaluasi. Kemudian seleksi alam dilakukan untuk memilih individu mana saja yang pantas dipertahankan pada generasi berikutnya. Semakin baik individu maka semakin baik pula nilai *fitness* yang dimiliki (Mahmudy, 2015).

2.3.2 Parameter Algoritma Genetika

Berikut merupakan parameter-parameter yang digunakan dalam algoritma genetika:

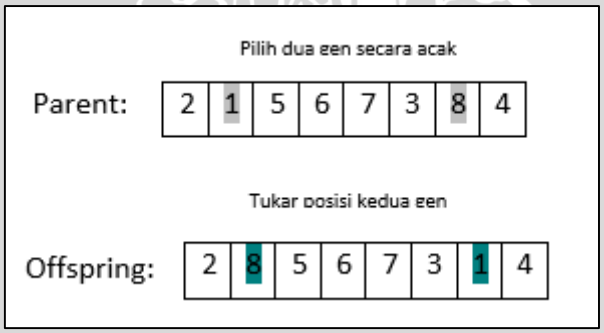
2.3.2.1 Probabilitas *Crossover*

Crossover merupakan proses menghasilkan individu baru yang nantinya memiliki sifar-sifat yang diwariskan induknya (Nugraha & Mahmudy, 2015). Proses *crossover* dilakukan dengan pemilihan acak yaitu 2 individu berbeda dari populasi sebagai *parent* (induk). Metode yang digunakan untuk melakukan *crossover* adalah *one-cut-point*, yaitu memilih satu titik potong secara acak dan menukarkan bagian kanan setiap induk guna menghasilkan individu baru (*offspring*) (Mahmudy, 2015).

Pada probabilitas *crossover* dinyatakan bahwa semakin besar nilai probabilitas *crossover* maka kemungkinan eksplorasi ruang pencarian semakin besar dan mempercepat ditemukannya solusi optimum. Nilai probabilitas *crossover* pada umumnya berkisar pada $0 \leq Pc \leq 1$, namun menurut Gen & Cheng (1997) nilai probabilitas *crossover* yang baik berkisar antara 0,6 sampai 1 (Ariastuti, 2015).

2.3.2.2 Probabilitas Mutasi

Mutasi adalah perubahan suatu operator genetika yang hasilnya menyebabkan perubahan acak pada satu kromosom. *Repciprocal exchange mutation* dan *insertion mutation* merupakan teknik yang sering digunakan sebagai teknik mutasi. *Repciprocal exchange mutation* befungsi dengan cara memilih dua posisi secara acak kemudian menukarkan dengan nilai pada posisi tempatnya sehingga menghasilkan individu yang baru (*offspring*). Gambar 2.3 menunjukan berjalannya metode *repciprocal exchange mutation*.



Gambar 2.3 Metode *Repciprocal Exchange Mutation*
Sumber: (Sulistyorini & Mahmudy, 2015)

Probabilitas mutasi merupakan prosentase yang dihasilkan dari kromosom yang akan mengalami mutasi dari seluruh kromosom yang terdapat pada populasi. Nilai probabilitas mutasi pada umumnya berkisar pada $0 \leq Pm \leq 1$, namun menurut Gen & Cheng (1997) nilai probabilitas mutasi yang baik berkisar antara 0,001 sampai 0,2 (Ariastuti, 2015).

2.3.2.3 Kriteria Pemberhentian

Kriteria pemberhentian merepresentasikan jumlah perulangan yang dilakukan pada algoritma genetika. Kriteria pemberhentian digunakan untuk menentukan pada perulangan keberapa sistem harus berhenti (Ariastuti, 2015).

2.3.3 Komponen Algoritma Genetika

Pada penelitian yang bertujuan untuk optimasi distribusi terdapat beberapa komponen penyusun algoritma genetika, diantaranya adalah representasi kromosom, evalusai, *penalty*, nilai *fitness,* dan seleksi.

2.3.3.1 Representasi Kromosom

Representasi kromosom merupakan proses penyelesaian suatu masalah dengan cara mengkodekan solusi permasalahan kedalam kromosom. Dalam Algoritma Genetika terdapat berbagai jenis representasi kromosom untuk setiap permasalahan yang berbeda (Sulistyorini & Mahmudy, 2015). Beberapa jenis representasi kromosom diantaranya adalah sebagai berikut:

1. Representasi biner
2. Representasi *integer*
3. Representasi *real*
4. Representasi permutasi

Representasi kromosom yang digunakan pada penelitian ini adalah representasi *integer* yaitu dimana setiap gen merupakan deretan bilangan utuh. Berikut pada Gambar 2.4 merupakan contoh representasi *integer*.

Individu	*Kromosom*
P1	45 78 20 48 90 10 11 105 40 22 46 29 30
P2	45 78 20 48 90 10 71 61 914 87 88 81 40
P3	40 22 46 29 30 23 67 97 102 27 10 55 64

Gambar 2.4 Representasi *Integer*

2.3.3.2 Pembangkitan Populasi

Populasi awal pada algoritma genetika dibangkitkan secara *random*. Populasi terdiri dari kromosom-kromosom yang merepresentasikan solusi yang diinginkan (Wahid & Mahmudy, 2015). Ukuran populasi merupakan ukuran dari populasi yang menyatakan jumlah individu atau kromosom (Ariastuti, 2015).

2.3.3.3 Evaluasi

Evaluasi adalah fungsi yang digunakan untuk mengetahui ketahanan kromosom untuk menentukan apakah kromosom masih pantas diteruskan eksistensinya (Tyas, Rahman, & Dewi, 2013). Untuk menentukan fungsi *fitness* diperlukan suatu fungsi khusus untuk dijadikan ukuran nilai *fitness* pada suatu

kromosom. Apabila solusi yang diinginkan adalah solusi optimum maka fungsi yang digunakan adalah f = h (Suyanto, 2007).

2.3.3.4 Nilai *Fitness*

Nilai *fitness* merupakan nilai yang menentukan kecocokan suatu masalah yang ingin dipecahkan. Nilai *fitness* melekat pada masing-masing individu yang akan dijadikan solusi. Individu yang dianggap layak untuk dijadikan solusi adalah indvidu yang memiliki nilai *fitness* tinggi, sehingga semakin tinggi nilai *fitness* yang dihasilkan maka kemungkinan individu menjadi calon solusi akan semakin besar (Nugraha & Mahmudy, 2015). Menurut Mahmudy (2015), fungsi *fitness* untuk mencari nilai maksimum dapat dihitung dengan fungsi sebagai berikut:

$$fitness = f(x) \quad (2.4)$$

Keterangan :
f = Fungsi yang digunakan dalam pencarian *Fitness*
x = Kromosom (individu)

2.3.3.5 Seleksi

Seleksi adalah memilih *parent* yang unggul untuk menghasilkan individu yang unggul. Seleksi dilakukan dengan cara memilih individu yang berkualitas baik dan pantas untuk dijadikan *parent* pada generasi berikutnya. Salah satu teknik seleksi yang dapat digunakan adalah seleksi *elitism*. Pada seleksi *elitism* dilakukan pengurutan individu pada populasi. Individu yang akan bertahan pada generasi selanjutnya adalah individu-individu pada urutan teratas sejumlah *popSize*. Dalam seleksi *elitism*, menjamin bahwa individu-individu yang terpilih merupakan individu yang memiliki nilai *fitness* yang baik. Kelemahan yang dimiliki seleksi *elitism* adalah individu yang memiliki nilai *fitness* rendah tidak memiliki kesempatan untuk bertahan pada generasi selanjutnya (Mahmudy, 2015). Pada Gambar 2.5 ditunjukan *pseudocode* proses seleksi *elitism.*

```
PROCEDURE ElitismSelection
Input :
   POP:        himpunan individu pada populasi
   Pop_size:   ukuran populasi
   OS:         himpunan individu anak (offspring) hasil reproduksi
               menggunakan crossover dan mutasi

Output:
   POP:                    himpunan individu pada populasi setelah
   proses seleksi selesai

/* gabungkan individu pada POP dan OS ke dalam TEMP */
TEMP ← Merge (POP, OS)
/* urutkan individu berdasarkan fitness secara ascending */
OrderAscending (Temp)
/* copy pop_size individu terbaik ke POP */
POP ← CopyBest (Temp, pop_size)

END PROCEDURE
```

Gambar 2.5 *Pseudocode Elitism Selection*
Sumber: (Mahmudy, 2015)

2.4 Distribusi

Distribusi merupakan kegiatan yang selalu dilakukan dalam dunia industri. Terdapat beberapa pengertian distribusi menurut para ahli, diantaranya adalah sebagai berikut:

1. Woodward. Distribusi berarti proses pengangkutan dan pengiriman barang dari pabrik produksi ke tangan pelanggan.
2. Devo Avidanto. Distribusi merupakan kegiatan menyalurkan hasil produksi dari produsen dan konsumen guna memenuhi kebutuhan manusia.
3. Indroyono. Distribusi merupakan kegiatan wajib bagi pelaku usaha untuk menyebarkan barang pasarannya ke tangan konsumen.

Dari berbagai pernyataan diatas dapat disimpulkan bahwa tujuan utama distribusi adalah untuk menyalurkan hasil produksi sampai ke tangan konsumen namun tetap harus memperhatikan sistem distribusi yang baik agar mendukung konsumen dan produsen (Ardiansyah, 2014).

2.4.1 Sistem Distribusi UD.TOSA

Dalam penelitian yang dilakukan (Ardiansyah, 2014) sistem distribusi merupakan cara yang dilakukan dalam proses distribusi hasil produksi dari produsen ke konsumen. Secara umum distribusi dibedakan menjadi dua, yaitu distribusi langsung atau distribusi tidak langsung. Berikut merupakan perbedaan dari distribusi langsung dan distribusi tidak langsung:

1. Distribusi Langsung

   Merupakan proses distribusi yang dilakukan secara langsung dari pabrik menuju konsumen tanpa melalui perantara. Contohnya ada penyaluran hasil tambak oleh nelayan ke pasar langsung.

2. Distribusi Tidak langsung

   Merupakan proses penyaluran hasil produksi melalu perantara yaitu agen distributor dalam kegiatan pengirimannya. Contohnya adalah pemasaran produk elektronik.

Pada contoh sistem distribusi di atas, UD.TOSA berperan sebagai perantara (*distributor*) dalam proses distribusi tidak langsung. UD.TOSA memilih produk untuk didistribusikan dari pabrik produksi (agen tetap) kemudian disimpan dalam gudang penyimpanan UD.TOSA. Proses tersebut dilakukan oleh tim *droping*, proses distribusi selanjutnya dilakukan oleh tim kanvas yaitu mengirimkan produk ke tangan konsumen. Konsumen yang dimaksud adalah toko-toko dan instansi-instansi yang melakukan pemesanan. Pada penelitian ini akan difokuskan terhadap proses distribusi yang dilakukan oleh tim *droping* . Proses *droping* oleh UD.TOSA dilakukan 2 sampai 3 kali per minggu atau hampir setiap hari apabila banyak pesanan dari pelanggan. Alat muat yang digunakan dalam proses *droping* adalah jenis DYNA 110 FT, COLT DIESEL FE74S (4X2) BUT, dan COLT DIESEL FE74S (4X2) M/T.

2.5 Dampak Kelebihan Beban Muatan (Overtonase)

Overtonase sering dikenal dengan kelebihan beban muatan dimana barang yang dimuat melebihi kapasitas maksimal alat muat. *Overtonase* sering terjadi karena semakin tinggi persaingan pertransportasian khususnya dalam dunia pendistribusian barang. Barang-barang yang dimuat sehingga menimbulkan kelebihan beban diantaranya merupakan kebutuhan pokok yang dibutuhkan banyak orang seperti air minum, batu bata, semen, beras, dan lain-lain (Triatmojo, 2017). Contoh fenomena *overtonase* dapat dilihat pada Gambar 2.6 dan Gambar 2.7.

Gambar 2.6 Kelebihan Muatan pada Distribusi Tebu
Sumber: indotrucker.blogspot.com

Gambar 2.7 Kelebihan Muatan pada Distribusi Gabah
Sumber: indotrucker.blogspot.com

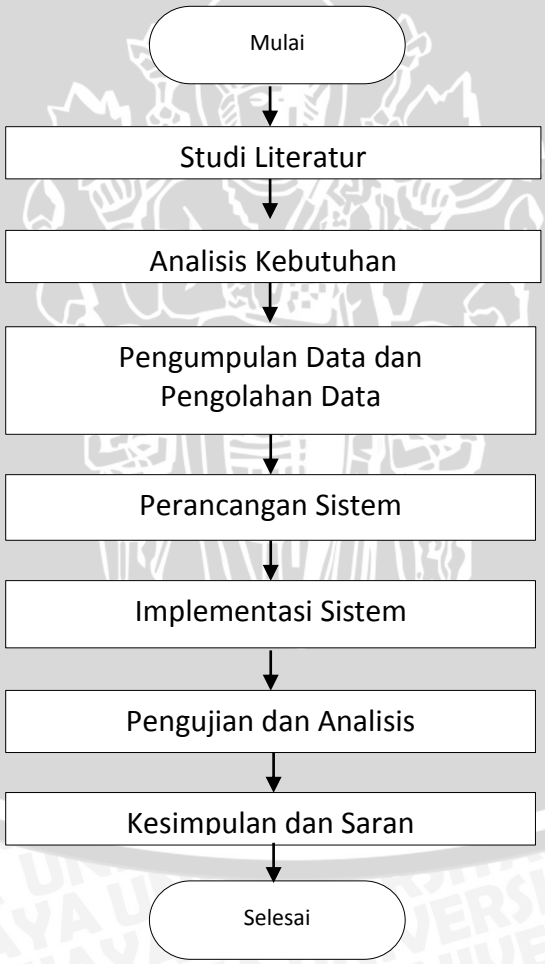
Memuat barang semaksimal mungkin merupakan hal yang dilakukan dalam setiap pengiriman barang, hal ini dimaksudkan untuk mengurangi biaya perjalanan. Meskipun begitu banyak dampak negatif yang ditimbulkan jika terlalu sering terjadi kelebihan muatan. Dampak kelebihan muatan bagi kendaraan yang

digunakan diantaranya dapat menimbulkan kerusakan pada komponen-komponen kendaraan seperti kerusakan *shockbreaker*, kerusakan *tiero* & *balljoint*, kerusakan *bushing arm*, kerusakan *bearing,* pecah ban, dan bahkan dapat menyebabkan kerusakan mesin. Selain kerusakan kendaraan, *overtonase* juga meningkatkan potensi kecelakaan yang dapat terjadi. Apabila terjadi kecelakaan maka akan mengakibatkan kerusakan produk yang dimuat. Oleh karena hal tersebut maka biaya yang harus dikeluarkan untuk perbaikan kendaraan dan penggantian produk akan mengakibatkan bertambahnya biaya distribusi dan berkurangnya keuntungan (Triatmojo, 2017). Dampak kelebihan beban muatan dapat dilihat pada Gambar 6.3.

Gambar 2.8 Dampak Kelebihan Beban Muatan
Sumber: indotrucker.blogspot.com

BAB 3 METODOLOGI

Pada bab ini akan dibahas mengenai metode yang dilakukan selama pembuatan sistem untuk optimasi 0/1 *multi-dimensional knapsack problem* dalam pendistribusian produk menggunakan algoritma genetika. Metode yang dilakukan akan dijelaskan sesuai urutan pelaksanaan yang dilakukan saat penelitian. Identifikasi masalah dan studi literatur yang menjadi dasar pembuatan sistem akan dijelaskan prosesnya pada bab ini. Kemudian untuk mengetahui fitur apa saja yang harus ada pada sistem dibahas mengenai proses pengumpulan data dan analisis kebutuhan sistem. Proses perancangan dan implementasi sistem adalah langkah yang dilakukan setelah tahap analisis kebutuhan. Setelah sistem jadi maka akan dilakukan proses pengujian dan analisis hasil pengujian untuk menarik kesimpulan dan saran dari sistem optimasi 0/1 *multi-dimensional knapsack problem* dalam pendistribusian produk menggunakan algoritma genetika. Pada Gambar 3.1 merupakan diagram blok yang merepresentasikan alur metodologi penelitian.



Gambar 3.1 Diagram Blok Metodologi Penelitian

3.1 Studi Literatur

Studi literatur merupakan tahapan lebih lanjut dalam proses penggalian informasi. Pada tahap ini mempelajari dasar teori dari berbagai literatur yang berhubungan dengan penelitian mengenai penerapan algoritma genetika untuk kasus optimasi 0-1 *multi-dimensional knapsack problem* dalam pendistribusian produk pada UD.TOSA. Pada tahap ini juga dilakukan pengkajian penelitian yang telah dilakukan sebelumnya untuk digunakan sebagai referensi pendukung yang dapat dijadikan acuan untuk penelitian yang akan dilakukan penulis. Dalam studi literatur ini dipelajari beberapa pokok bahasan yang mendukung penelitian, diantaranya adalah:

1. *Knapsack roblem*
2. *Multiple dimensional knapsack problem*
3. Algoritma genetika
4. Struktur algoritma genetika
5. Operator algoritma genetika
6. Proses distribusi UD.TOSA dan pengaruh kelebihan muatan

3.2 Analisis Kebutuhan

Untuk mengetahui kebutuhan apa saja yang harus dipenuhi dalam pembuatan sistem dilakukan Analisis kebutuhan. Sekenario aliran data yang akan dijalankan dalam sistem optimasi 0-1 *multi-dimensional knapsack problem* dalam pendistribusian produk menggunakan algoritma genetika adalah sebagai berikut:

1. Sesuai *field* yang disediakan oleh sistem, pengguna melakukan *input* data yang dibutuhkan untuk menghasilkan optimasi 0-1 *multi-dimensional knapsack problem.*
2. Sistem akan melakukan proses perhitungan Algoritma Genetik terhadap masukkan yang diterima.
3. Sistem akan mengeluarkan *output* yaitu sebuah solusi permasalahan yang mendekati optimal.

Sebelum pembuatan sistem optimasi 0-1 *multi-dimensional knapsack problem* pendistribusian produk menggunakan algoritma genetika dibutuhkan analisis kebutuhan fungsional dan non fungsional.

3.2.1 Analisis Kebutuhan Fungsional

Kebutuhan fungsional merupakan fungsional sistem yang disediakan, kinerja sistem ketika menerima *input* dan kinerja sistem dalam menghasilkan *output* sistem pada kondisi tertentu. Berikut merupakan kebutuhan fungsional pada penelitian ini:

1. Sistem dapat menerima *input* data yang dibutuhkan sistem optimasi.
2. Sistem dapat memproses masukkan menggunakan algoritma genetika dengan benar.
3. Sistem dapat menghasilkan *ouput* berupa solusi yang mendekati optimal.

3.2.2 Analisis Kebutuhan Non Fungsional

Kebutuhan non fungsional adalah kebutuhan pendukung sistem yang apabila tidak terpenuhi tidak akan mempengaruhi kinerja sistem, dan apabila terpenuhi maka akan memberikan fitur tambahan yang membuat sistem lebih baik.

1. Data kebutuhan *hardware*:
   a. Laptop
   b. *Processor* Intel Core i5-4200U CPU 1.60GHz 2.30GHz
   c. RAM 4.00 GB
   d. *System type* 64-bit

2. Kebutuhan *software* untuk mendukung proses pembangunan sistem:
   a. OS Windows 8.1 Pro
   b. *Browser* (Google Chrome, Mozila Firefox, dll)
   c. *Database* berupa *file handling* (jxl *library*)
   d. Netbeans 8.0.2

3.3 Pengumpulan Data dan Pengolahan Data

Proses pengumpulan data dalam penelitian ini dilakukan di UD.TOSA. Data yang digunakan dalam penelitian pemanfaatan algoritma genetika untuk optimasi 0/1 *multi-dimensional knapsack problem* dalam pendistribusian produk adalah variabel yang mempengaruhi optimasi distribusi. Hipotesis pada penelitian ini adalah membuat sistem yang dapat mengoptimasikan 0/1 *multi-dimensional knapsack problem* dalam pendistribusian produk di UD.TOSA. Terdapat dua cara untuk melakukan proses pengumpulan data, yaitu data skunder dan data primer. Data yang diperoleh dari pengumpulan data oleh orang lain adalah yang dimaksud dengan data sekunder, sedangkan data yang dihasilkan langsung dari responden penelitian merupakan data primer (Alfinda, Soebroto, & Regasari, 2014). Cara pengumpulan data yang digunakan pada penelitian ini adalah dengan menggunakan data sekunder.

Tabel 3.1 adalah tabel kebutuhan data dan kegunaan data dalam penelitian sistem optimasi distribusi kereta api menggunakan algoritma genetika:

Tabel 3.1 Kebutuhan dan Kegunaan Data

No	Data kebutuhan	Fungsi data	Sumber data	Metode
1	Jenis Produk	Digunakan untuk mengetahui nama produk sehingga menemukan karakteristiknya	UD.TOSA	Wawancara
2	Target produk yang akan didistribusikan	Digunakan untuk mengetahui ketepatan sistem	UD.TOSA	Observasi

Tabel 3.1 (lanjutan) Kebutuhan dan Kegunaan Data

No	Data kebutuhan	Fungsi data	Sumber data	Metode
3	Berat Produk	Digunakan untuk mengetahui bobot produk	UD.TOSA	Observasi dan wawancara
4	Volume produk	Digunakan untuk mengetahui dimensi ruang yang dibutuhkan setiap produk	UD.TOSA	Observasi dan wawancara
5	Jenis kendaraan	Digunakan untuk mengetahui alat muat apa saja yang digunakan	UD.TOSA	Wawancara
6	Kapasitas kendaraan	Digunakan untuk mengetahui kapasitas muat tiap alat transportasi	UD.TOSA	Observasi dan wawancara

Tahap selanjutnya setelah tahap pengumpulan data, adalah tahap pengolahan data. Pada tahap ini data yang telah di peroleh saat pengumpulan data akan diproses sesuai dengan kebutuhan sistem. Dalam optimasi 0/1 *multi-dimensional knapsack problem* dalam pendistribusian produk menggunakan algoritma genetika, berikut merupakan pengolahan data yang akan dilakukan:

1. Data-data yang diperoleh akan digunakan senagai nilai karakteristik produk dan alat muat.
2. Mengolah data sampai menghasilkan jumlah produk yang dimuat.
3. Memproses data menggunakan algoritma genetika untuk memperoleh solusi-solusi yang mendekata optimal.

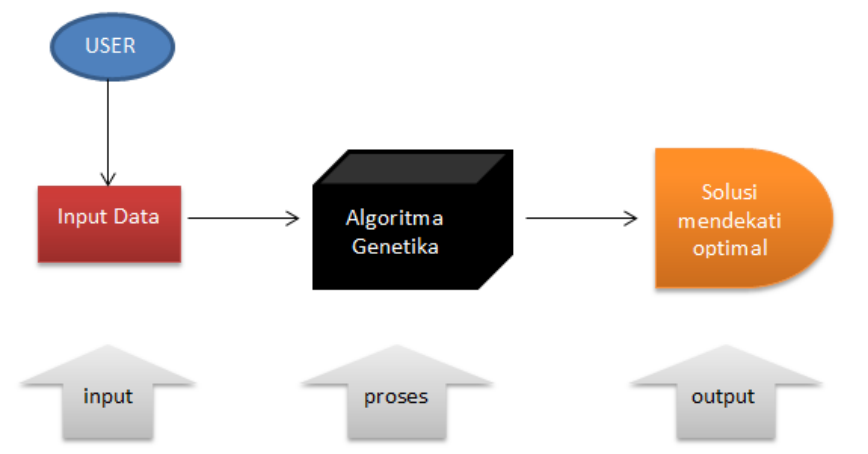
3.4 Perancangan Sistem

Seperti yang dijelaskan pada sekenario aliran data, pengguna akan memasukkan data yang nantinya data tersebut akan digunakan dalam penyelesaian *knapsack problem* sehingga menghasilkan kromosom yang optimal sebagai solusi yang dihasilkan oleh sistem. *String* kromosom memiliki nilai *fitness* masing-masing yang berfungsi untuk mengetahui kualitas kromosom tersebut. Kromosom tersebut akan diseleksi oleh sistem menggunakan algoritma genetika. *Output* yang dihasilkan sistem berupa solusi-solusi permasalahan yang mendekati optimal. Untuk menggambarkan perancangan sistem dapat dilihat pada Gambar 3.2.

3.5 Implementasi

Proses implementasi sistem dilakukan dengan mengacu pada perancangan sistem yang telah dibuat sebelumnya. Penerapan hasil perancangan ke dalam kode (*code*) sesuai dengan sintaks dari bahasa pemrograman yang digunakan untuk membangun sistem adalah proses implementasi. Untuk melakukan

implementasi sistem optimasi 0/1 *multi-dimensional knapsack problem* dalam pendistribusian produk menggunakan algoritma genetika, digunakan bahasa pemrograma java. *Database* yang digunakan adalah teknik *file handling* berupa data excel.



Gambar 3.2 Arsitektur Perancangan Sistem

3.6 Pengujian dan Analisis Sistem

Pengujian yang dilakukan pada sistem optimasi ini adalah pengujian yang ditujukan untuk mengetahui jalannya sistem sudah sesuai kebutuhan atau tidak. Selain itu pengujian dilakukan untuk mengetahui jalannya algoritma genetika yang digunakan dalam sistem optimasi. Pengujian terhadap algoritma genetika terdapat pada penggunaan ukuran populasi, banyaknya generasi, nilai *crossover rate*, dan nilai *mutation rate*. Setelah pengujian jalannya algoritma, hal selanjutnya yang diuji adalah *output* yang dihasilkan sistem. Hal ini dilakukan untuk mengetahui kualitas solusi yang dihasilkan oleh sistem. Solusi yang dihasilkan oleh sistem akan dibandingkan dengan kenyataan yang ada.

3.7 Kesimpulan dan Saran

Kesimpulan dapat dibuat setelah semua tahap dalam pembangunan sistem diselesaikan, dimulai dari proses perancangan, implementasi dan pengujian. Penarikan kesimpulan didasarkan pada analisis hasil pengujian sistem merujuk pada hasil penerapan algoritma genetika. Proses yang dilakukan setelah penarikan kesimpulan adalah pemberian saran, pada proses ini dilakukan penjabaran kekurangan yang terdapat pada sistem mulai dari pemilihan metode, proses perancangan, implementasi, dan hasil pengujian sistem. Pemberian solusi yang lebih baik dilakukan berdasarkan kekurangan yang terdapat pada sistem guna menciptakan sistem yang lebih baik di masa depan.

BAB 4 PERANCANGAN

Pada bab ini akan diulas mengenai perancangan sistem dan bagaimana pemanfaatan algoritma genetika untuk kasus optimasi 0/1 *Multi-dimensional Knapsack Problem* dalam pendistribusian produk.

4.1 Formulasi Permasalahan

4.1.1 Deskripsi Masalah

Multi-dimensional knapsack problem (MKDP) pada proses distribusi produk (proses *droping*) merupakan representasi permasalahan yang terjadi. Dalam pendistribusian produk masalah yang muncul adalah beban muatan yang selalu berlebihan sehingga sering terjadi kerusakan pada komponen alat muat yang digunakan. Hal ini mengakibatkan bertambahnya frekuensi *maintenance* alat muat sehingga mengakibatkan berkurangnya keuntungan yang didapat. Dari permasalahan tersebut dibutuhkan solusi agar daya muat alat transportasi dapat dimaksimalkan tanpa melebihi kapasitas sehingga performa kendaraan terjaga dan tidak memerlukan *maintenance* terlalu sering. Salah satu cara untuk mengatasi masalah tersebut yaitu harus dilakukan menejemen beban muat sehingga tidak terjadi kelebihan muatan saat proses *droping*. Pada kasus ini produk yang akan dimuat memiliki karakteristik yang berbeda-beda, setiap produk memiliki berat dan volume tersendiri. Alat muat yang digunakan juga memiliki kapasitas yang berbeda, setiap alat muat memiliki beban maksimal dan volume maksimal muatan. Oleh karena setiap produk memiliki karakteristik yang berbeda, maka perlu dilakukan kombinasi muatan produk yang tepat agar menghasilkan volume muatan semaksimal mungkin tanpa melebihi beban maksimal alat muat.

Pada penelitian ini produk dapat dimuat dengan syarat tidak melebihi kapasitas alat muat yang dimiliki alat muat. Hal ini dikarenakan pada *multi-dimensional knapsack problem* memiliki beberapa batasan (*constraint)* yaitu berat dan volume. *Knapsack* yang dimaksud adalah alat muat yang digunakan dalam proses distribusi. Produk yang dapat dipilih untuk didistribusikan terdiri dari 10 jenis produk. *Multi-dimensional knapsack problem* yang dibahas pada penelitian ini adalah *integer knapsack* (0/1) dimana dimensi setiap barang yang akan dimasukkan ke dalam *kanpsack* tidak bisa dipisah menjadi beberapa bagian, oleh karena itu setiap satu barang hanya memiliki dua kemungkinan yaitu dimuat atau tidak dimuat sama sekali. Proses *droping* oleh UD.TOSA dilakukan 2 sampai 3 kali dalam seminggu menggunakan 9 alat muat yang tersedia. Dalam praktek keseharian UD.TOSA terkadang hanya beberapa alat muat saja yang digunakan dari 9 alat muat yang tersedia.

4.1.2 Data Produk

Data produk diperlukan untuk dapat mengetahui karakteristik masing-masing produk. Karakteristik produk yang perlu diketahui adalah nama produk, berat, dan volume. Produk yang didistribusikan oleh UD.TOSA adalah n objek yang akan

direpresentasikan menjadi kromosom. Terdapat 10 jenis produk yang memiliki karakteristik masing-masing. Karakteristik yang dimiliki produk akan digunakan dalam proses perhitungan *fitness* kromosom. Detail produk yang akan didistribusikan dapat dilihat pada 4.1.

Tabel 4.1 Data Karakteristik Produk

No	Nama	Berat (Kg)	Volume m^3
1	Cleo 115 mL	4,6	0,008602
2	Cleo 250 mL	12	0,015345
3	Cleo 550 mL	13,2	0,022815
4	Cleo 1500 mL	18	0,027712125
5	Teh Gelas	6	0,007986
6	Ake Gelas 120 mL	5,4	0,013376
7	Ake Gelas 220 mL	8,8	0,01102
8	Ale-ale	6	0,027712125
9	Teh Rio	6	0,027712125
10	Jusica	5,2	0,013376

4.1.3 Data Alat Muat

Data alat muat diperlukan untuk dapat mengetahui kapasitas muat setiap alat muat yang digunakan dalam proses *droping*. Terdapat banyak alat muat yang dimiliki oleh UD.TOSA namun hanya alat muat tertentu saja yang digunakan dalam proses *droping*. Data alat muat yang perlu diketahui adalah merek , jenis, beban maksimal, ruang maksimal alat , dan jumlah alat muat yang dimiliki. Dalam penelitian ini alat muat yang digunakan diibaratkan sebagai knapsack dalam kasus 0/1 *multi-dimensional knapsack problem*. Setiap alat muat memiliki kapasitas muat yang berbeda. Alat muat yang digunakan untuk proses *droping* dapat dilihat pada Tabel 4.2.

Tabel 4.2 Data Karakteristik Alat Muat

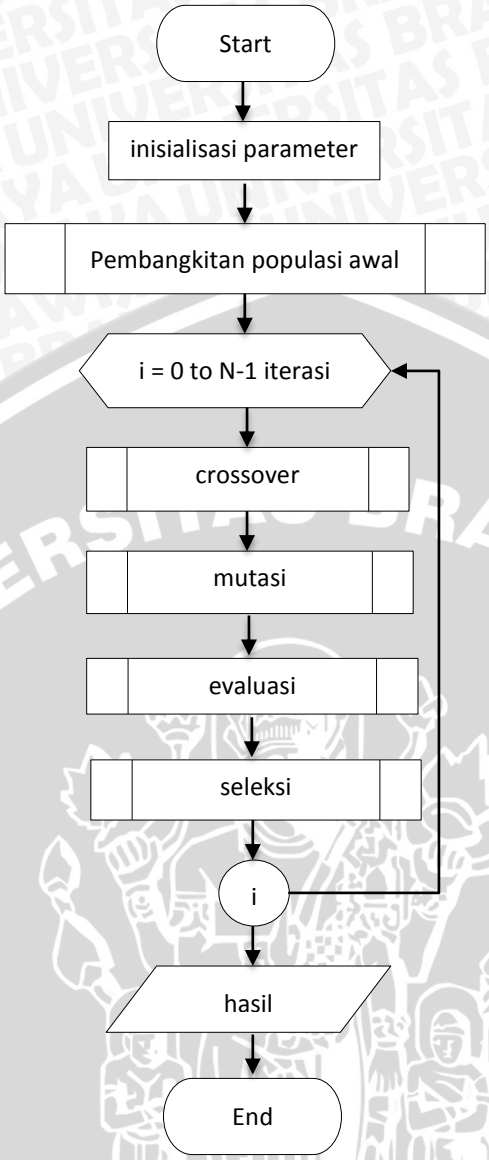
No.	Merek	Jenis	Kapasitas Beban Max (Kg)	Kapasitas Ruang Max (m^3)	Jumlah Armada (unit)
1	HINO	WU342R-HKMRHD3/110HD	10000	25,269	2
2	TOYOTA	DYNA 110 FT	8000	25,375	6
3	MITSUBISHI	COLT DIESEL FE74S (4X2) M/T	7500	25,184	1

4.2 Siklus Algoritma Genetika

Langkah-langkah penyalesaian masalah optimasi 0/1 *multi-dimensional knapsack problem* menggunakan algoritma genetika adalah sebagai berikut:

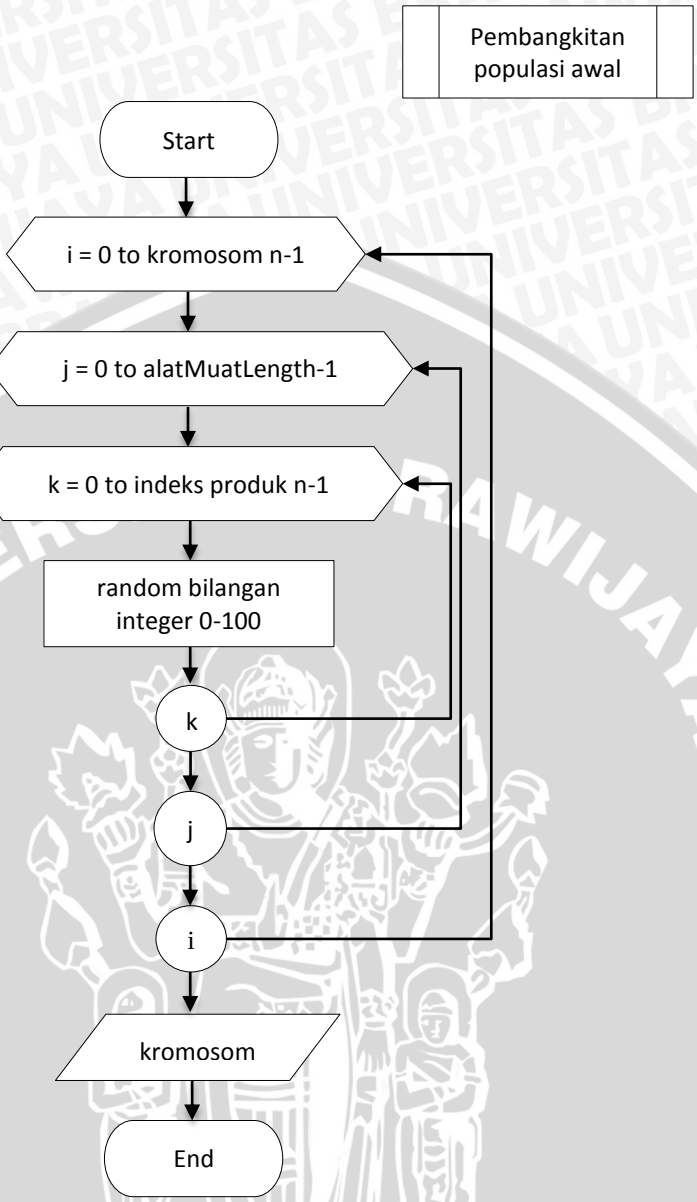
1. Inisialisasi parameter yang akan digunakan dalam algoritma genetika yaitu jumlah populasi, nilai cr, dan nilai mr. Jumlah populasi menentukan *popSize* yang dibutuhkan untuk mencapai solusi yang diinginkan. Nilai mr dan cr akan digunakan dalam proses reproduksi pada algoritma genetika.
2. Pembangkitan populasi awal yaitu proses pembangkitan kromosom secara acak. Kromosom dibangkitkan berbentuk matrik sejumlah dengan *popSize* yang sudah di inisialisasi sebelumnya.
3. Raproduksi dengan *crossover* yang melibatkan dua *parent* untuk menghasilkan *offspring*. Pada contoh perhitungan *parent* yang terpilih adalah P1 dan P2. Reproduksi dengan mutasi melibatkan satu *parent* untuk mengasilkan *offspring*. Pada contoh perhitungan parent yang terpilih adalah P1.
4. Evaluasi adalah menghitung nilai *fitness* setiap individu yang tebentuk, pada penelitian ini *fitness* dipengaruhi oleh maksimal volume yang terisi tanpa meliebihi beban dan target produk.
5. Seleksi yaitu pemilihan individu dalam populasi , seleksi yang digunakan pada penelitian ini yaitu *elitism selection.*
6. Kondisi berhenti yaitu menentukan pada generasi ke-berapa *iterasi* harus dihentikan. Kondisi berhenti telah diinisialisasi terlebih dahulu.

Siklus penyelesaian optimasi 0/1 *multi-dimensional knapsack problem* menggunakan algoritma genetika digambarkan pada Gambar 4.1.



Gambar 4.1 *Flowcart* Penyelesaian Masalah Menggunakan Algoritma Genetika

Pada Gambar 4.2 ditunjukan langkah-langkah pembangkitan populasi awal.

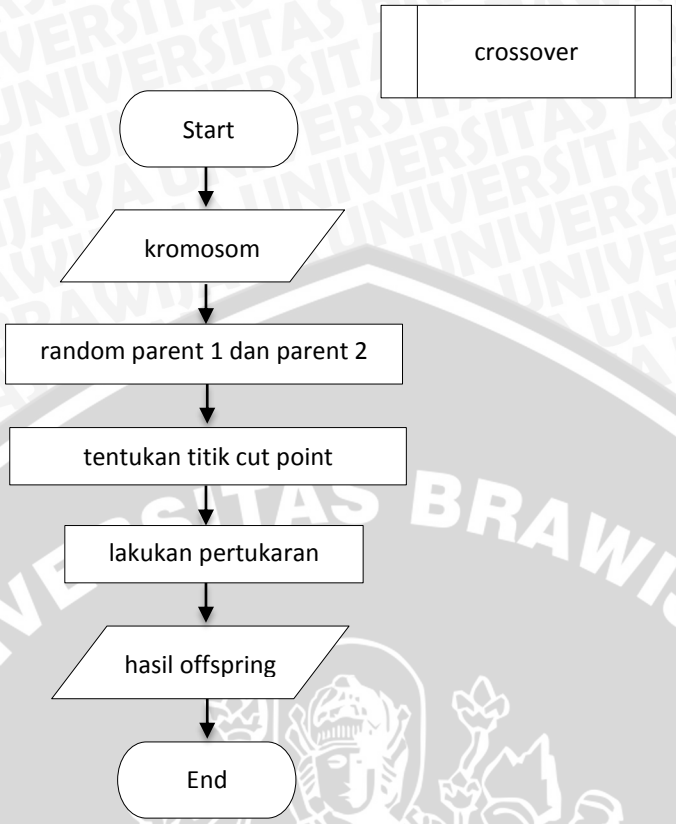


Gambar 4.2 *Flowcart* Pembangkitan populasi awal

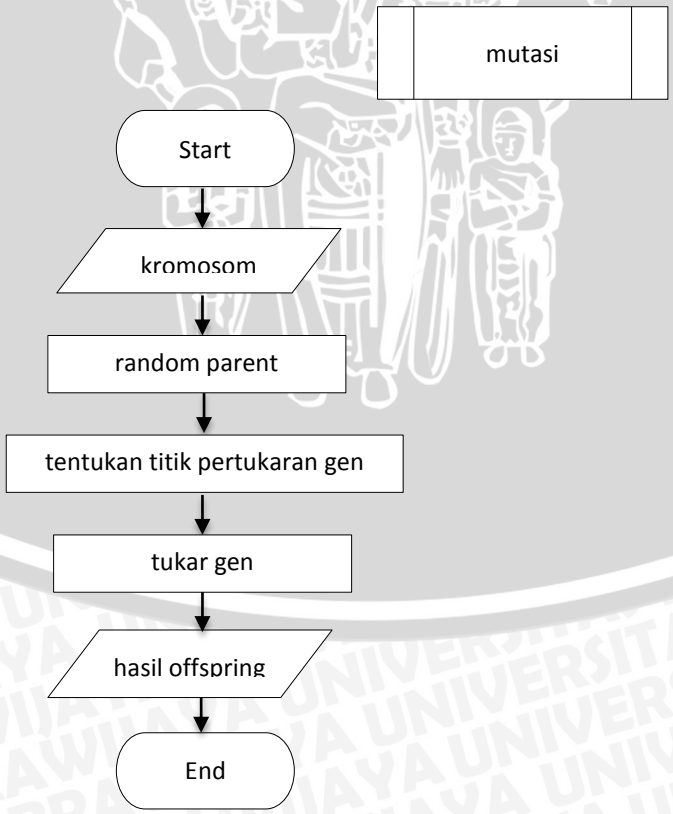
Keterangan :

1. Melakukan random antara 0-100 untuk digunakan dalam perhitungan proporsi produk yang akan dimuat sepanjang gen jenis produk pada satu alat muat.
2. Proses random diulang sejumlah i produk dan j alat muat hingga membentuk satu kromosom berupa matrik.
3. Proses random dihentikan apabila sudah terbentuk beberapa kromosom sesuai dengan *popSize* yang dinginkan yang telah diinisialisasi di awal.

Pada Gambar 4.3 dan Gambar 4.4 ditunjukan alur reproduksi menggunakan teknik *crossover* dan mutasi pada algoritma genetika.

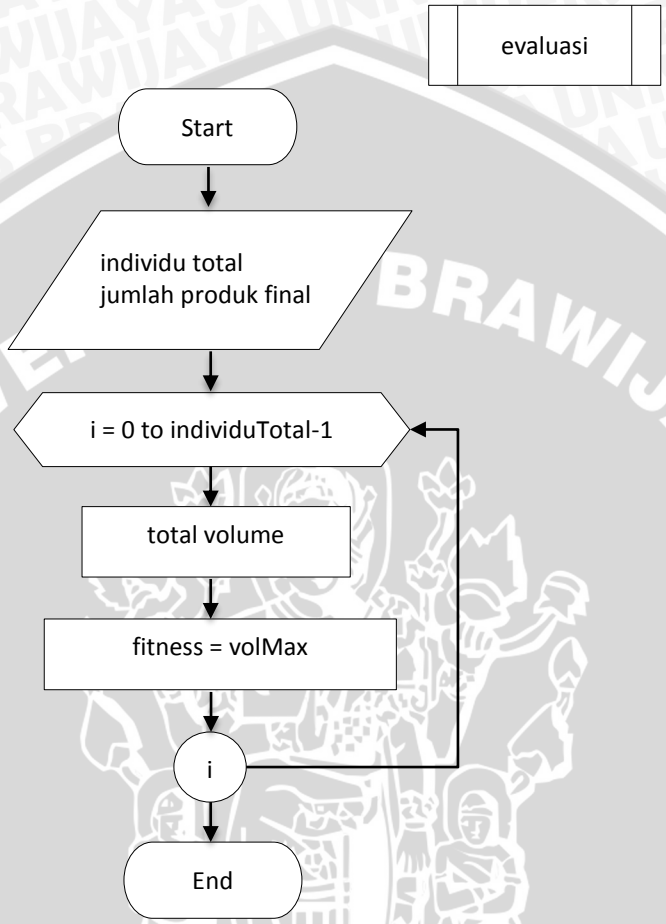


Gambar 4.3 *Flowcart Crossover*



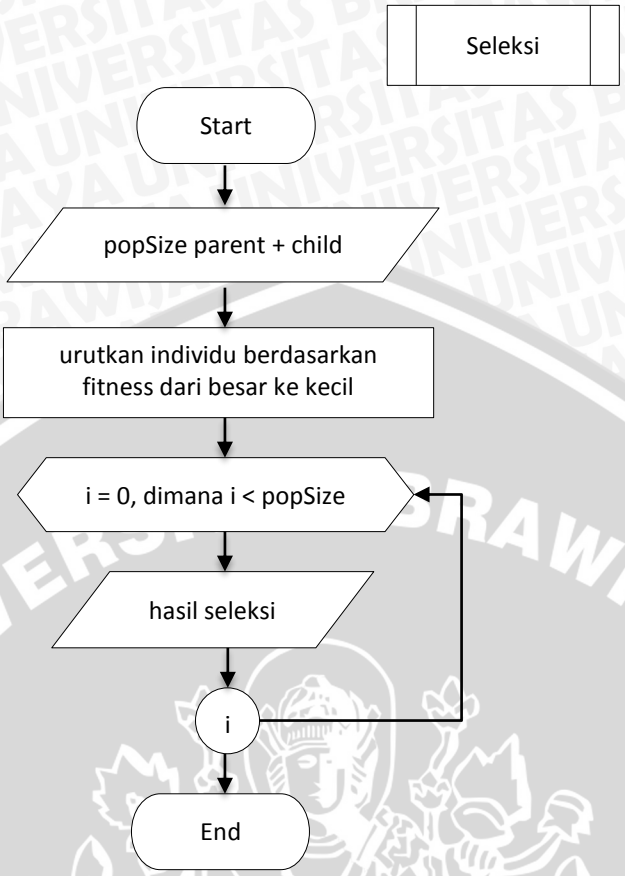
Gambar 4.4 *Flowcart Mutasi*

Pada Gambar 4.5 ditunjukan alur evaluasi menggunakan yang menghasilkan nilai *fitness* individu pada algoritma genetika.



Gambar 4.5 *Flowcart* Evaluasi

Proses terakir pada algoritma genetika adalah seleksi. Berikut pada Gambar 4.6 merupakan *flowcart* dari proses seleksi.



Gambar 4.6 *Flowcart* Seleksi *Elitism*

4.3 Penyelesaian Masalah menggunakan Algoritma Genetika

Penyelesaian masalah menggunakan algoritma genetika pada penelitian ini digunakan untuk melakukan kombinasi produk yang akan dimuat agar alat muat mampu mengangkut produk sebanyak-banyaknya tanpa melebihi kapasitas alat muat. Kromosom yang dihasilkan merepresentasikan produk-produk yang dimuat dalam alat muat (*knapsack*).

4.3.1 Representasi Kromosom dan Perhitungan Fitness

Representasi kromosom yang digunakan pada kasus optimasi *multi-dimensional knapsack problem* berupa matrik yang berisi *string integer*. Kromosom berbentuk matrik karena jenis produk dinyatakan dalam kolom dan jenis alat muat dinyatakan dalam baris. *String integer* (gen) pada kromosom bukan menyatakan jumlah (kuantitas) produk yang dimuat, melainkan nilai acak yang akan digunakan untuk menghitung proporsi volume produk agar tidak melebihi kapasitas ruang maksimal. Baris pada kromosom menunjukan kendaraan atau alat muat yang digunakan dan kolom pada kromosom menunjukan produk yang dimuat. Matrik kromosom dapat berubah-ubah sesuai dengan jumlah alat muat yang digunakan. Setiap gen pada kromosom memiliki indeks yang menyatakan jenis alat muat dan jenis produk. Setiap indeks memiliki karakteristik yang berbeda-beda sesuai dengan alat muat dan produk yang

diwakili. Misalkan baris 1 menunjukan bahwa alat muat yang digunakan adalah adalah alat muat 1 yaitu HINO WU342R-HKMRHD3/110HD dengan kapasitas berat maksimum 10.000 Kg dan ruang maksimum 10.000 cm^3. Pada kolom 1 menunjukan bahwa produk yang dimuat adalah produk 1 yaitu cleo 115 mL dengan berat 9 Kg dan volume 0,025725 cm^3. Pada Gambar 4.6 menunjukan representasi kromosom berupa matrik yang berisi string integer. Contoh representasi kromosom yang ditampilkan adalah matrik 9x10 yaitu menggunakan 9 alat muat yang tersedia dan 10 produk yang didistribusikan. Solusi yang dihasilkan adalah jumlah produk final yang didapatkan dari perhitungan proporsi dan rasio melalui kromosom yang terbentuk.

Kromosom 1	1	2	3	4	5	6	7	8	9	10
1	22	99	74	2	97	17	59	20	11	86
2	82	94	12	9	57	96	90	51	23	30
3	76	80	29	52	74	6	27	96	75	83
4	58	99	44	28	23	51	10	14	36	45
5	6	18	43	49	48	23	36	59	93	99
6	43	58	55	78	98	5	56	93	100	78
7	23	52	4	78	65	71	14	45	40	98
8	94	17	58	91	46	0	1	94	61	19
9	27	36	0	30	98	58	18	78	93	41

Gambar 4.7 Representasi Kromosom

Untuk melakukan perhitungan *fitness*, langkah awal yang dilakukan adalah menghitung proporsi volume produk yang dimuat berdasarkan volume maksimal yang dimiliki setiap alat muat. Nilai yang digunakan untuk menghitung proporsi adalah gen kromosom yang telah terbentuk sebelumnya. Rumus perhitungan proporsi produk berdasarkan volume maksimal alat muat dapat dilihat pada Persamaan 4.1.

$$Vol_{ij} = \frac{Gen_j}{\sum Gen\ j} \times VMax_i \quad (4.1)$$

Keterangan :

Vol_{ij} = Proporsi volume produk

i = jenis alat muat 1, 2, 3, ……, 9

j = jenis produk 1, 2, 3, ……, 10

Hasil perhitungan proporsi produk yang dimuat berdasarkan volume maksimal alat muat dari kromosom Gambar 4.6 dapat dilihat pada Tabel 4.3.

Tabel 4.3 Proporsi Produk Berdasarkan Volume Maksimal Alat Muat

vol ij	1	2	3	4	5	6	7	8	9	10
1	1141.515	5136.819	3839.643	103.7741	5033.045	882.0801	3061.337	1037.741	570.7577	4462.287
2	3808.93	4366.335	557.4044	418.0533	2647.671	4459.235	4180.533	2368.969	1068.358	1393.511
3	3224.916	3394.649	1230.56	2206.522	3140.05	254.5987	1145.694	4073.579	3182.483	3521.948
4	3607.23	6157.169	2736.52	1741.422	1430.453	3171.875	621.9363	870.7108	2238.971	2798.713
5	321.2025	963.6076	2301.951	2623.154	2569.62	1231.276	1927.215	3158.492	4978.639	5299.842
6	1643.261	2216.491	2101.845	2980.798	3745.105	191.0768	2140.06	3554.029	3821.536	2980.798
7	1191.071	2692.857	207.1429	4039.286	3366.071	3676.786	725	2330.357	2071.429	5075
8	4958.94	896.8295	3059.771	4800.676	2426.715	0	52.75468	4958.94	3218.035	1002.339
9	1419.557	1892.743	0	1577.286	5152.468	3049.42	946.3716	4100.944	4889.587	2155.624

Setelah dihasilkan proporsi produk berdasarkan volume maksimal alat muat, langkah selanjutnya adalah menghitung jumlah produk yang dimuat. Jumlah produk didapatkan melalui pembagian antara nilai Vol_{ij} dengan volume produk. Jumlah produk yang dihasilkan dapat dilihat pada Tabel 4.4.

Tabel 4.4 Jumlah Produk

Jml produk	1	2	3	4	5	6	7	8	9	10
1	44	135	235	1	630	196	680	129	71	450
2	148	114	34	5	331	990	929	296	133	140
3	125	89	75	28	393	56	254	510	398	355
4	140	162	168	22	179	704	138	109	280	282
5	12	25	141	33	321	273	428	395	623	535
6	63	58	129	38	468	42	475	445	478	301
7	46	70	12	52	421	817	161	291	259	512
8	192	23	188	62	303	0	11	620	402	101
9	55	49	0	20	645	677	210	513	612	217

Setelah didapatkan jumlah produk yang dimuat berdasarkan volume maksimal alat muat, langkah selanjutnya adalah menhitung berat produk pada setiap alat muat. Perhitungan berat produk dilakukan dengan cara mengalikan $Jml\ produk_{ij}$ dengan $Berat\ produk_j$ kemudian dihitung total berat pada setiap

alat muat dengan menjumlahkan berat produk yang dihasilkan. Apabila berat produk pada setiap alat muat melebihi kapasitas beban maksimal alat muat maka harus dilakukan perhitungan rasio berat yang berfungsi untuk mengurangi jumlah produk agar beratnya tidak melebihi beban maksimal alat muat. Cara menghitung rasio berat dapat dilihat pada Persamaan 4.2.

$$Rasio\ berat = \frac{DT_i}{(\sum Jmlproduk_{ij})\ B_j} \quad (4.2)$$

Keterangan :

Rasio berat = nilai selalu kurang dari 1

DT = Beban maksimal

Jmlproduk = jumlah produk yang dihasilkan dari proporsi volume produk

B = kapasitas beban maksimal

i = Alat muat 1, 2, 3, ……, 9

j = Jenis produk 1, 2, 3, ……, 10

Setelah dilakukan perhitungan rasio maka akan didapatkan jumlah produk final yang dipastikan tidak melebihi kapasitas angkut alat muat, baik volume ataupun beban maksimal alat muat. Jumlah produk final dapat dihitung dengan Persamaan 4.3. Jumlah produk final yang dihasilkan merupakan solusi permasalahan pada penelitian ini. Jumlah produk final dapat dilihat pada Tabel 4.5

$$Produk\ final = Rasio\ berat_i \times Jmlproduk_{ij} \quad (4.3)$$

Keterangan :

Produk final = jumlah produk yang dimuat tanpa melebihi kapasitas alat muat

Rasio berat = nilai rasio

Jmlproduk = jumlah produk yang dihasilkan dari proporsi volume maksimal

i = Alat muat 1, 2, 3, ……, 9

j = Jenis produk 1, 2, 3, ……, 10

Dari Tabel 4.5 dapat diketahui jumlah produk yang dimuat, misalnya pada kolom 1 baris 1 berarti produk 1 yang dimuat pada alat muat 1 sebanyak 28 buah dan begitu sesterusnya.

Tabel 4.5 Jumlah Produk Final

Final Produk	1	2	3	4	5	6	7	8	9	10
1	28	88	153	0	411	127	443	84	46	293
2	78	60	18	2	175	525	492	157	70	74
3	76	54	46	17	241	34	156	313	244	218
4	80	93	97	12	103	406	79	62	161	162
5	6	13	73	17	167	142	223	206	326	279
6	35	32	72	21	264	23	268	251	270	170
7	24	37	6	28	227	441	86	157	139	276
8	133	15	130	42	210	0	7	429	278	70
9	25	22	0	9	295	310	96	235	280	99

Setelah diketahui jumlah produk final yang tidak melebihi berat maka dapat dihitung volume produk yang termuat. Volume produk yang termuat akan digunakan dalam perhitungan nilai *fitness*. Perhitungan nilai *fitness* dapat dilakukan menggunakan Persamaan 4.4.

$$Fitness = VolMax \quad (4.4)$$

Keterangan :

Fitness = nilai fitness setiap kromosom

VolMax = total volume produk final

Nilai fitness yang dihasilkan melasui persamaan diatas adalah sebesar 86738.783. Hasil perhitungan volume maksimal yang dimuat dapat dilihat pada Tabel 4.6.

Tabel 4.6 Volume Produk Final

Vol Final	1	2	3	4	5	6	7	8	9	10	total
1	720.3	3341.8	2490.075	0	3282.246	571.5	1993.5	670.824	367.356	2900.7	16338.3
2	2006.55	2278.5	292.95	154.35	1397.55	2362.5	2214	1253.802	559.02	732.6	13251.82
3	1955.1	2050.65	748.65	1311.975	1924.626	153	702	2499.618	1948.584	2158.2	15452.4
4	2058	3531.675	1578.675	926.1	822.558	1827	355.5	495.132	1285.746	1603.8	14484.19
5	154.35	493.675	1188.075	1311.975	1333.662	639	1003.5	1645.116	2603.436	2762.1	13134.89

Tabel 4.6 (lanjutan) Volume Produk Final

Vol Final	1	2	3	4	5	6	7	8	9	10	total
6	900.375	1215.2	1171.8	1620.675	2108.304	103.5	1206	2004.486	2156.22	1683	14169.56
7	617.4	1405.075	97.65	2160.9	1812.822	1984.5	387	1253.802	1110.054	2732.4	13561.6
8	3421.425	569.625	2115.75	3241.35	1677.06	0	31.5	3425.994	2220.108	693	17395.81
9	643.125	835.45	0	694.575	2355.87	1395	432	1876.71	2236.08	980.1	11448.91
Volume Total											129237.5

4.3.2 Inisialisasi Populasi Awal

Proses inisialisasi populasi awal adalah tahapan pembentukan kromosom secara *random*. Nilai yang disimpan dalam kromosom adalah nilai *random*. Pada tahap ini dilakukan inisialisasi parameter yang digunakan dalam siklus pemanfaatan algoritma genetika. Parameter tersebut adalah jumlah *populasi* (popsize), *crossover rate* (cr), dan *mutation rate* (mr).

Contoh inisialisasi populasi awal adalah sebagai berikut :

PopSize = 4

Crossover rate = 0,4

Mutation rate = 0,2

Berdasarkan alur yang ditunjukan pada Gambar 4.2 maka dapat dibentuk beberapa individu sesuai dengan populasi yang ditentukan. Pada Tabel 4.7 merupakan contoh inisialisasi awal populasi yang terdiri dari kromosom berdasarkan formulasi permasalahan.

Tabel 4.7 Inisialisasi Populasi Awal

Individu	Kromosom									
P1	22	99	74	2	97	17	59	20	11	86
	82	94	12	9	57	96	90	51	23	30
	76	80	29	52	74	6	27	96	75	83
	58	99	44	28	23	51	10	14	36	45
	6	18	43	49	48	23	36	59	93	99
	43	58	55	78	98	5	56	93	100	78
	23	52	4	78	65	71	14	45	40	98
	94	17	58	91	46	0	1	94	61	19
	27	36	0	30	98	58	18	78	93	41

Tabel 4.7 (lanjutan) Inisialisasi Populasi Awal

Individu	Kromosom									
P2	33	10	27	55	62	51	66	54	84	17
	60	41	17	25	74	48	52	16	89	83
	6	48	2	91	62	69	100	92	15	18
	44	0	46	50	34	99	63	33	36	20
	28	38	27	97	70	24	54	89	77	86
	69	5	72	69	96	97	89	61	34	8
	33	89	0	59	55	95	29	21	21	87
	44	92	72	50	43	3	44	58	27	22
	40	74	28	42	16	48	46	88	0	3
P3	54	78	41	57	25	48	16	81	74	29
	16	68	77	74	20	65	41	67	86	4
	70	6	24	41	75	3	10	29	94	76
	23	44	81	4	61	97	6	84	56	45
	48	27	82	6	85	82	52	61	5	24
	35	89	58	14	1	47	89	22	63	71
	93	52	2	74	29	92	89	39	46	13
	40	76	79	16	75	42	24	94	52	9
	54	77	66	78	50	100	40	90	52	51
P4	59	85	92	88	12	36	6	99	21	89
	23	58	77	82	97	20	95	50	14	53
	50	17	60	98	50	40	30	75	21	95
	74	56	44	73	92	7	6	49	10	87
	98	98	8	30	83	38	3	49	11	21
	92	80	83	29	69	90	45	52	16	27
	57	59	53	47	23	16	46	7	83	64
	28	75	95	61	83	54	22	2	47	77
	59	58	38	54	61	10	62	89	2	95

4.3.3 Reproduksi

Proses setelah terbentuknya populasi awal adalah melakukan proses reproduksi. Proses reproduksi dalam penelitian ini dilakukan dengan menggunakan dua metode reproduksi, yaitu metode *crossover* dan mutasi.

4.3.3.1 Crossover

Reproduksi menggunakan metode *crossover* dilakukan dengan cara memilih 2 *parent* secara acak yang nantinya akan dikawinkan sehingga menghasilkan individu baru setelah dilakukan modifikasi pada gennya. Dalam penelitian ini teknik *crossover* yang digunakan adalah teknik *one cut point crossover*. Proses crossover sesuai dengan alur Gambar 4.3 yang pada sub bab 4.2.

Setelah proses *crossover* dilakukan akan dihasilkan *offspring* yang jumlahnya tergantung dari nilai *crossover rate* (cr). Jumlah *offspring* dapat dihitung dengan cara mengalikan nilai *popsize* yang telah diinisialisasi dengan nila cr. Dalam inisialisasi populasi awal dicontohkan bahwa nilai cr adalah 0,4 dengan popsize 4. *Offspring* yang akan terbentuk dari proses *crossover* adalah sebanyak 0,4 x 4 =1 *offspring*. Batas warna biru pada pada P1 dan warna merah jambu pada P2 merupakan titik potong yang dipilih pada teknik *one cut point crossover.* Offspring dibentuk dengan menggabungkan gen pada P1 dan P2 sesuai dengan titik potong yang dipilih. Tabel 4.8 menunjukan hasil proses crossover pada induk P2 dan P3.

Tabel 4.8 *Offspring* Proses *Crossover*

Individu	Kromosom									
P1	22	99	74	2	97	17	59	20	11	86
	82	94	12	9	57	96	90	51	23	30
	76	80	29	52	74	6	27	96	75	83
	58	99	44	28	23	51	10	14	36	45
	6	18	43	49	48	23	36	59	93	99
	43	58	55	78	98	5	56	93	100	78
	23	52	4	78	65	71	14	45	40	98
	94	17	58	91	46	0	1	94	61	19
	27	36	0	30	98	58	18	78	93	41
P2	33	10	27	55	62	51	66	54	84	17
	60	41	17	25	74	48	52	16	89	83
	6	48	2	91	62	69	100	92	15	18
	44	0	46	50	34	99	63	33	36	20
	28	38	27	97	70	24	54	89	77	86
	69	5	72	69	96	97	89	61	34	8
	33	89	0	59	55	95	29	21	21	87
	44	92	72	50	43	3	44	58	27	22
	40	74	28	42	16	48	46	88	0	3
C1	69	22	67	37	61	95	65	26	51	6
	94	67	53	100	28	65	84	29	32	90
	96	64	34	33	0	32	80	75	56	29
	88	67	16	30	33	71	3	27	13	0
	25	25	37	35	94	74	2	58	81	13
	96	35	43	10	9	40	64	93	95	30
	34	59	67	59	13	72	87	42	49	17
	64	55	40	13	24	70	80	19	93	32
	0	78	75	56	91	99	97	88	59	91

Keterangan :
1. P1 dan P2 adalah induk yang dipilih secara acak
2. C1 adalah *offspring* yang dihasilkan dalam sekali proses *crossover*

4.3.3.2 Mutasi

Sama seperti pada proses *crossover*, *offspring* yang dihasilkan pada proses mutasi juga dipengaruhi oleh nilai yang telah diinisialisasi di awal yaitu nilai mr. Teknik yang digunakan dalam proses mutasi pada penelitian ini adalah teknik *reciprocal exchange mutation*. Teknik ini melakukan pemilihan gen induk yang akan ditukar.

Offspring yang dihasilkan dapat dihitung dengan cara mengalikan *popSize* dengan mr yang telah di diinisialisasi. *Offspring* yang akan dihasilkan adalah sebanyak 0,2 x 4 = 0,8 atau bisa dibulatkan menjadi 1. Pemilihan *parent* dilakukan dengan cara *random* antara P1, P2, P3 dan P4. Warna merah dan biru pada kromosom menandakan dua gen yang terpilih yang akan ditukar pada teknik *reciprocal exchange mutation*. Warna hijau menunjukan gen baru pada offspring hasil pertukaran gen pada induk P1. Tabel 4.9 menunjukan hasil proses mutasi pada induk P1.

Tabel 4.9 *Offspring* Proses Mutasi

Individu	Kromosom									
P1	22	99	74	2	97	17	59	20	11	86
	82	94	12	9	57	96	90	51	23	30
	76	80	29	52	74	6	27	96	75	83
	58	99	44	28	23	51	10	14	36	45
	6	18	43	49	48	23	36	59	93	99
	43	58	55	78	98	5	56	93	100	78
	23	52	4	78	65	71	14	45	40	98
	94	17	58	91	46	0	1	94	61	19
	27	36	0	30	98	58	18	78	93	41
C2	22	99	74	2	97	17	59	20	11	18
	82	94	12	9	57	96	90	51	23	30
	76	80	29	52	74	6	27	96	75	83
	58	99	44	28	23	51	10	14	36	45
	6	86	43	49	48	23	36	59	93	99
	43	58	55	78	98	5	56	93	100	78
	23	52	4	78	65	71	14	45	40	98
	94	17	58	91	46	0	1	94	61	19
	27	36	0	30	98	58	18	78	93	41

Keterangan :
1. P1 adalah induk yang terpilih secara acak yang
2. C2 adalah *offspring* yang dihasilkan pada proses mutasi

4.3.4 Evaluasi

Setelah proses reproduksi selesai hingga menghasilkan *offspring*, prpses selanjutnya adalah melakukan evaluasi. Pada Gambar 4.5 sub bab 4.2 merupakan

alur proses evaluasi yang dilakukan dalam penelitian ini. Dalam proses evaluasi terdapat proses menghitung *fitness*. Individu yang akan dihitung *fitness* adalah seluruh individu yang terdapat dalam populasi baik *parent* maupun *offspring* yang dihasilkan selama proses produksi. *Fitness* yang dihasilkan menunjukan seberapa baik suatu kromosom untuk dapat dipertahankan pada generasi berikutnya. Semakin baik nilai *fitness* yang dimiliki maka semakin baik pula kromosom tersebut untuk menjadi calon solusi. Pada Gambar 4.5 sub bab 4.2 merupakan alur proses evaluasi yang dilakukan dalam penelitian ini. Populasi baru dari penggabungan individu awal dan *offspring* dapat dilihat pada Tabel 4.10.

Tabel 4.10 Individu dalam Populasi

Individu	Kromosom									
P1	22	99	74	2	97	17	59	20	11	86
	82	94	12	9	57	96	90	51	23	30
	76	80	29	52	74	6	27	96	75	83
	58	99	44	28	23	51	10	14	36	45
	6	18	43	49	48	23	36	59	93	99
	43	58	55	78	98	5	56	93	100	78
	23	52	4	78	65	71	14	45	40	98
	94	17	58	91	46	0	1	94	61	19
	27	36	0	30	98	58	18	78	93	41
P2	33	10	27	55	62	51	66	54	84	17
	60	41	17	25	74	48	52	16	89	83
	6	48	2	91	62	69	100	92	15	18
	44	0	46	50	34	99	63	33	36	20
	28	38	27	97	70	24	54	89	77	86
	69	5	72	69	96	97	89	61	34	8
	33	89	0	59	55	95	29	21	21	87
	44	92	72	50	43	3	44	58	27	22
	40	74	28	42	16	48	46	88	0	3
P3	54	78	41	57	25	48	16	81	74	29
	16	68	77	74	20	65	41	67	86	4
	70	6	24	41	75	3	10	29	94	76
	23	44	81	4	61	97	6	84	56	45
	48	27	82	6	85	82	52	61	5	24
	35	89	58	14	1	47	89	22	63	71
	93	52	2	74	29	92	89	39	46	13
	40	76	79	16	75	42	24	94	52	9
	54	77	66	78	50	100	40	90	52	51

Tabel 4.10 (lanjutan) Individu dalam Populasi

Individu	Kromosom									
P4	59	85	92	88	12	36	6	99	21	89
	23	58	77	82	97	20	95	50	14	53
	50	17	60	98	50	40	30	75	21	95
	74	56	44	73	92	7	6	49	10	87
	98	98	8	30	83	38	3	49	11	21
	92	80	83	29	69	90	45	52	16	27
	57	59	53	47	23	16	46	7	83	64
	28	75	95	61	83	54	22	2	47	77
	59	58	38	54	61	10	62	89	2	95
C3	69	22	67	37	61	95	65	26	51	6
	94	67	53	100	28	65	84	29	32	90
	96	64	34	33	0	32	80	75	56	29
	88	67	16	30	33	71	3	27	13	0
	25	25	37	35	94	74	2	58	81	13
	96	35	43	10	9	40	64	93	95	30
	34	59	67	59	13	72	87	42	49	17
	64	55	40	13	24	70	80	19	93	32
	0	78	75	56	91	99	97	88	59	91
C2	22	99	74	2	97	17	59	20	11	18
	82	94	12	9	57	96	90	51	23	30
	76	80	29	52	74	6	27	96	75	83
	58	99	44	28	23	51	10	14	36	45
	6	86	43	49	48	23	36	59	93	99
	43	58	55	78	98	5	56	93	100	78
	23	52	4	78	65	71	14	45	40	98
	94	17	58	91	46	0	1	94	61	19
	27	36	0	30	98	58	18	78	93	41

Setelah individu baru terbentuk maka perhitungan *fitness* dilakukan terhadap seluruh individu dalam populasi. Perhirungan nilai *fitness* dilakukan sesuai dengan persamaan 4.5 yang telah dijabarkan dalam sub bab 4.3.1. Pada Tabel 4.11 merupakan nilai *fitness* yang dimiliki masing-masing individu.

Tabel 4.11 Evaluasi Individu dalam Populasi

Individu	Fitness
P1	143.8336
P2	139.6666
P3	146.7637
P4	136.0314
C1	135.8778
C2	141.5642

Dari Tabel 4.11 dapat dilihat bahwa nilai *fitness* tertinggi yaitu sebesar 146.7637 dimiliki oleh individu P3.

4.3.5 Seleksi

Proses seleksi dilakukan untuk memilih individu-individu yang pantas dipertahankan untuk generasi berikutnya berdasarkan nilai *fitness* yang dimiliki. Jumlah individu yang dipilih sesuai dengan jumlah *popSize* yang telah diinisialisasi. Teknik yang digunakan untuk melakukan seleksi pada penelitian ini adalah *elitism selection*. Pada seleksi *elitism* dilakukan pengurutan individu mulai dari nilai *fitness* terbesar sampai terkecil. Individu yang memiliki *fitness* tinggi pada urutan atas akan dipilih sesuai dengan jumlah *popSize* yang dibutuhkan. Pada Gambar 4.6 sub bab 4.2 merupakan alur dalam proses seleksi yang dilakukan dalam penelitian ini.

Hasil seleksi yang dihasilkan dapat dilihat pada Tabel 4.12 dan 4.13. Proses seleksi dari 6 individu dalam populasi menghasilkan 4 individu yang layak diprtahankan pada generasi berikutnya.

Tabel 4.12 Urutan Berdasarkan *Fitness* Tertinggi

Urutan	Individu	Fitness
1	P3	146.7637
2	P1	143.8336
3	C2	141.5642
4	P2	139.6666
5	P4	136.0314
6	C1	135.8778

Tabel 4.13 Populasi Baru pada Generasi Selanjutnya

Populasi Baru	Individu asal	Fitness
P1	P3	146.7637
P2	P1	143.8336
P3	C2	141.5642
P4	P2	139.6666

Dari Tabel 4.13 dapat disimpulkan bahwa hasil terbaik terdapat pada indvidu P3 dengan nilai *fitness* sebesar 146.7637. P1 pada generasi baru merupakan individu P4 pada generasi sebelumnya.

4.4 Perancangan Pengujian

Pengujian yang dilakukan pada penelitian ini adalah pengujian algoritma yang digunakan dalam penelitian dan pengujian perbandingan optimasi hasil. Pengujian algoritma berfungsi untuk mengetahui keberhasilan algoritma

genetika pada sistem optimasi *0/1 multi-dimensional knapsack problem.* Pengujian algoritma yang akan dilakukan adalah pengujian terhadap ukuran populasi, jumlah generasi, nilai *cr* dan *mr*.

4.4.1 Pengujian Ukuran Populasi

Pengujian banyaknya populasi dilakukan untuk mengetahui seberapa banyak jumlah poupulasi yang mendekati optimal dalam pemanfaatan algoritma genetika untuk kasus optimasi 0/1 *multi-dimensional knapsack problem* dalam pendistribusian produk. Proses pengujian ukuran populasi akan dilakukan sebanyak 10 kali percobaan, dari 20 sampai 200 populasi dengan kelipatan 20. Nilai cr yang digunakan adalah 0,6 dan nilai mr yang digunakan adalah 0,3. Perancangan pengujian ukuran populasi dapat dilihat pada Tabel 4.14.

Tabel 4.14 Perancangan Pengujian Ukuran Populasi

Populasi	Nilai fitness										Rata-rata
	Percobaan ke-										
	1	2	3	4	5	6	7	8	9	10	
20											
40											
60											
80											
100											
120											
140											
160											
180											
200											

4.4.2 Pengujian Banyaknya Generasi

Pengujian banyaknya generasi dilakukan untuk mengetahui seberapa banyak jumlah generasi yang mendekati optimal dalam pemanfaatan algoritma genetika untuk kasus optimasi *multi-dimensional knapsack problem* dalam pendistribusian produk. Proses pengujian banyaknya generasi akan dilakukan sebanyak 10 kali percobaan, dari 10 sampai 100 generasi dengan kelipatan 10. Perancangan pengujian banyaknya generasi dapat dilihat pada Tabel 4.15.

Tabel 4.15 Perancangan Pengujian Banyaknya Generasi

Generasi	Nilai fitness										Rata-rata
	Percobaan ke-										
	1	2	3	4	5	6	7	8	9	10	
10											
20											
30											
40											
50											
60											
70											
80											
90											
100											

4.4.3 Pengujian Kombinasi Crossover Rate (*Cr*) dan Mutation Rate (*Mr*)

Pengujian kombinasi *cr* dan *mr* dilakukan untuk mengetahui nilai *mr* dan *cr* yang mendekati optimal dalam pemanfaatan algoritma genetika untuk kasus optimasi multi-dimensional knapsack problem dalam pendistribusian produk. Proses pengujian *cr* dan *mr* yang akan dilakukan adalah kombinasi yang jika dijumlahkan nilai *cr* dan *mr* harus bernilai 1. Nilai *cr* dan *mr* yang digunakan adalah kelipatan 0,1 antara 0 sampai 1. Perancangan pengujian kombinasi *cr* dan *mr* dapat dilihat pada Tabel 4.16.

Tabel 4.16 Perancangan Pengujian Kombinasi *Cr* dan *Mr*

Kombinasi		Nilai fitness										Rata-rata
		Percobaan ke-										
Cr	Mr	1	2	3	4	5	6	7	8	9	10	
1	0,1											
0,9	0,2											
0,8	0,3											
0,7	0,4											
0,6	0,5											
0,5	0,6											

0,4	0,7											
0,3	0,8											
0,2	0,9											
0,1	1											

4.4.4 Pengujian Perbandingan Hasil

Pengujian perbandingan hasil merupakan pengujian yang membandingkan hasil daya muat asli UD.TOSA dengan hasil daya muat sistem. Pada pengujian ini akan dilakukan perhitungan selisih beban muatan dengan beban maksimal alat muat. Perhitungan tersebut dilakukan pada hasil muatan asli dan hasil muatan sistem. Setelah ditemukan selisih beban maka akan dihitung prosentase kelebihan bebannya kemuadian dibandingkan. Untuk melakukan perhitungan selisih beban dapat digunakan Persamaan 4.6.

$$Selisih\ Beban = Bmax_i - Btotal_{ij} \quad (4.6)$$

Keterangan :

Selisih Beban = selisih antara beban maksimal alat muat dengan beban muatan

Bmax = Beban maksimal Alat muat

Btotal = Berat muatan

i = Alat muat 1, 2, 3, ……, 9

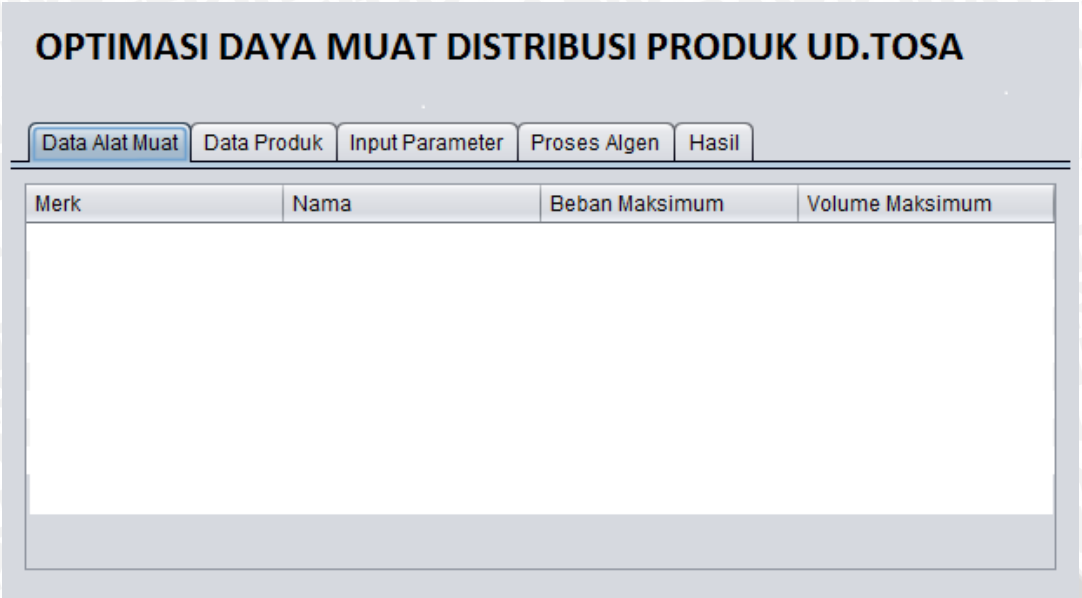
j = Jenis produk 1, 2, 3, ……, 10

4.5 Perancangan Antarmuka

Perancangan antarmuka pada penelitian ini terdapat 5 tab halaman, yaitu:

1. Tab Data Alat muat

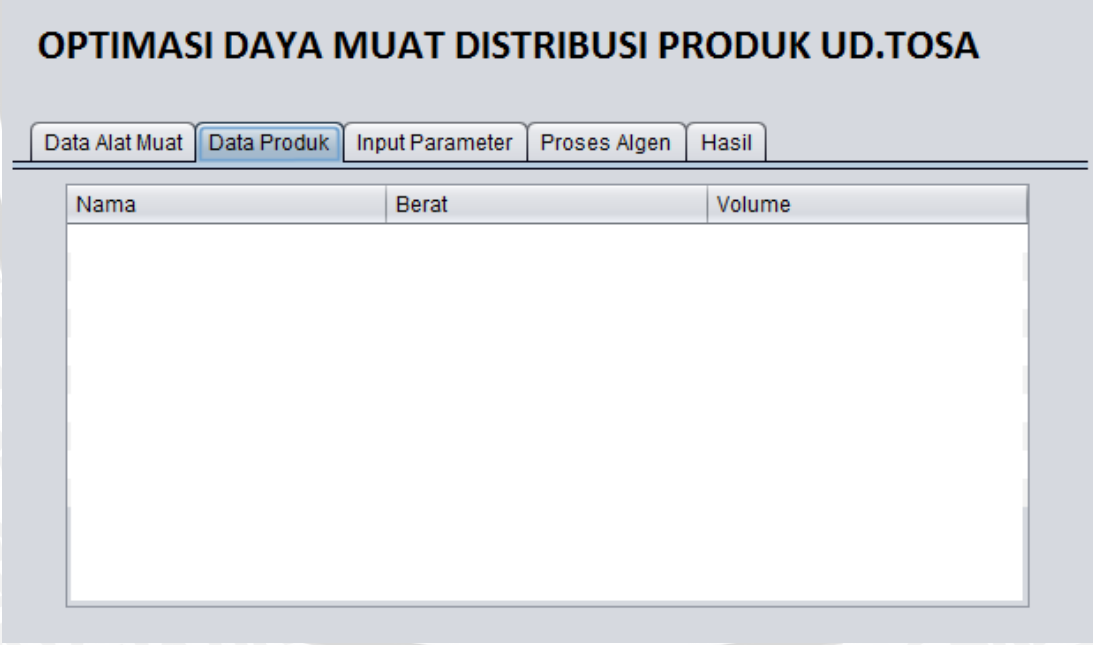
   Pada tab data alat muat menunjukan alat muat yang digunakan untuk proses *droping* oleh UD.TOSA beserta dengan karakteristik alat muat. Perancangan antarmuka dapa dilihat pada Gambar 4.8.



Gambar 4.8 Perancangan Antarmuka Tab Data Alat Muat

2. Tab Data Produk

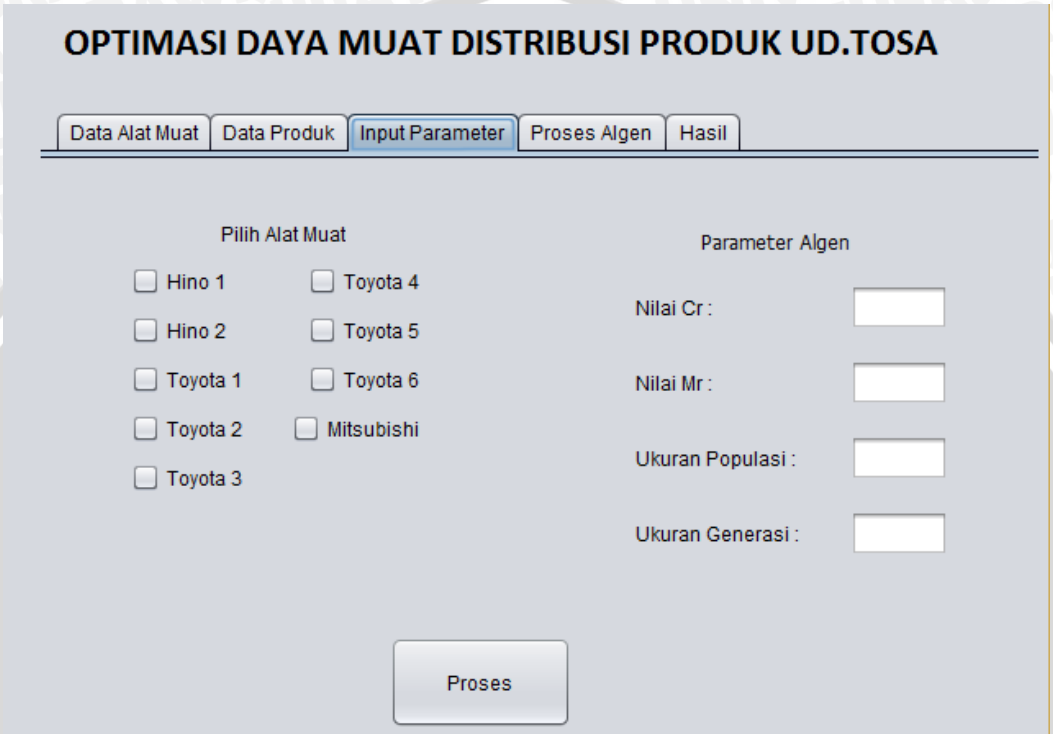
Pada tab data produk menunjukan produk-produk yang akan dimuat saat proses *droping* oleh UD.TOSA beserta dengan karakteristik masing-masing produk. Perancangan antarmuka dapa dilihat pada Gambar 4.9.



Gambar 4.9 Perancangan Antarmuka Tab Data Produk

3. Tab Input Parameter

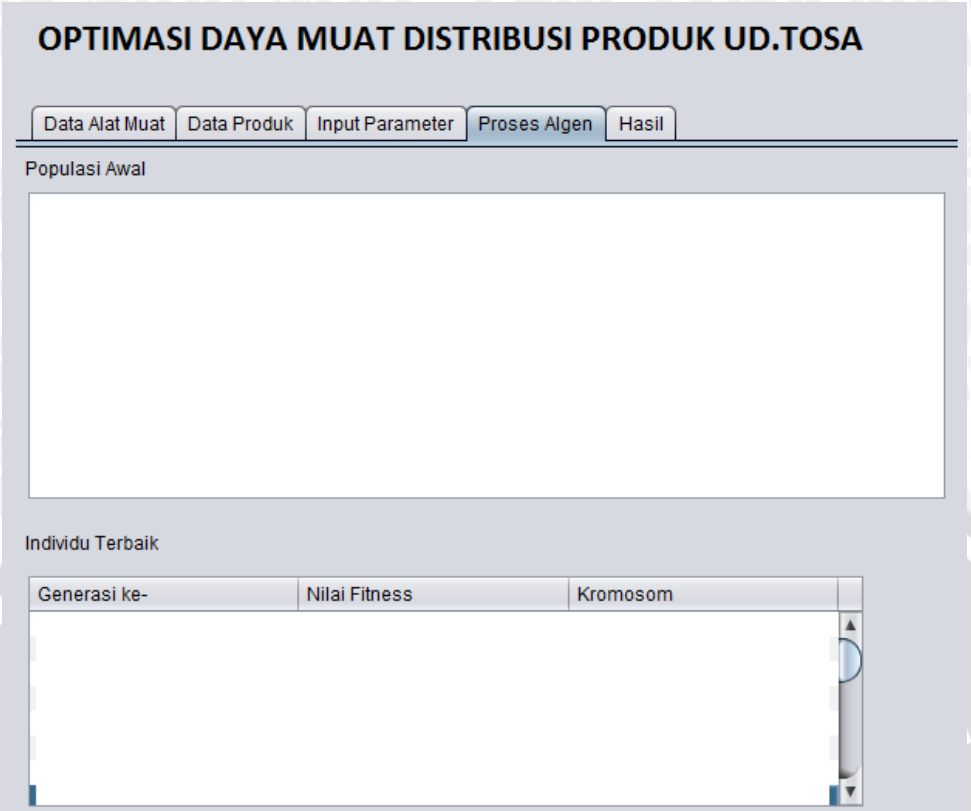
Pada halaman input parameter disediakan *field* yang harus diisi untuk melakukan inisialisasi parameter-parameter yang akan digunakan dalam proses perhitungan algoritma genetika. Perancangan antarmuka halaman input dapat dilihat pada Gambar 4.10.



Gambar 4.10 Perancangan Antarmuka Tab Input Parameter

4. Tab Proses Algen

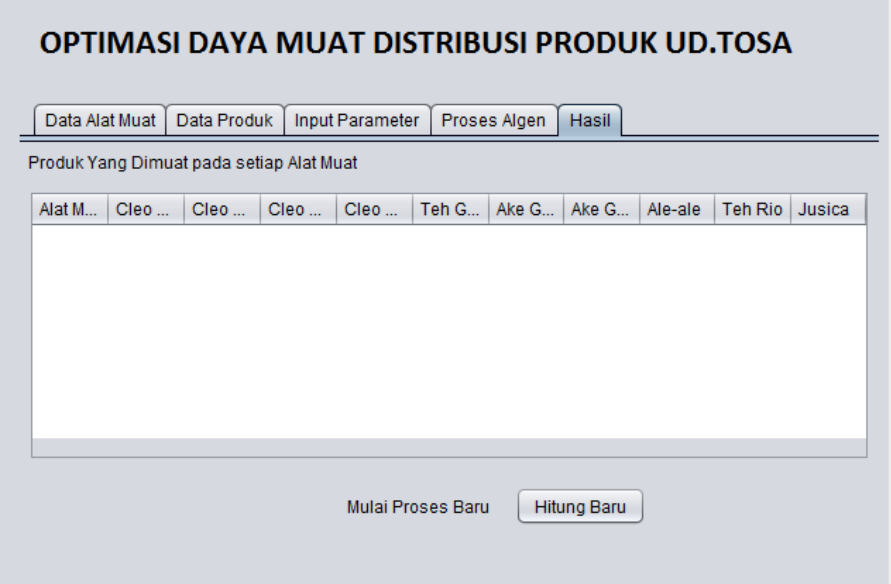
Pada tab proses algoritma menampilkan contoh proses algoritma genetika yaitu populasi awal yang dibangkitkan dan nilai fitness terbaik pada setiap generasi. Perancangan antarmuka dapat dilihat pada Gambar 4.11.



Gambar 4.11 Perancangan Antarmuka Tab Proses Algoritma Genetika

5. Tab Hasil

Pada tab hasil menunjukan solusi yang dihasilkan setelah proses optimasi. Perancangan antarmuka halaman hasil dapat dilihat pada Gambar 4.12.



Gambar 4.12 Perancangan Antarmuka Tab Hasil

BAB 5 IMPLEMENTASI

Dalam bab ini akan dijelaskan mengenai implementasi program menggunakan bahasa java yang menerapkan konsep algoritma genetika.

5.1 Implementasi Struktur Class Program

Implementasi yang dilakukan dalam penelitian ini menggunakan *software* NetBeans IDE 8.0.2 dengan bahasa pemrograman java. Data yang digunakan berasal dari *file excel*, teknik *file handling* ini menggunakan *library* Jxl.jar. Berikut merupakan penjelasan mengenai *class* yang terdapat dalam program :

1. Package ALGEN

Dalam package ini terdapat 5 *class* yang berfungsi untuk melakukan perhitungan sesuai dengan alur algoritma genetika. *Class-class* tersebut diantaranya adalah sebagai berikut :

a. *Class* AG

Class AG memiliki beberapa *method* yang berfungsi menerapkan perhitungan sesuai dengan alur dalam algoritma genetika. *Method-method* yang terdapat dalam *class* ini diantaranya berfungsi untuk melakukan proses pembagkitan populasi, proses reproduksi, dan proses seleksi.

b. *Class* Kromosom

Class Kromosom memiliki beberapa *method* yang berungsi untuk menggambarkan kromosom sampai dengan nilai *fitness* yang dimiliki. Dalam *class* ini kromosom yang diciptakan berbentuk matrik dimana baris merepresentasikan alat muat yang digunakan dan kolom merepresentasikan jenis produk yang dimuat.

c. *Class* Populasi

Class populasi memiliki beberapa *method* yang berfungsi untuk menampung individu-individu yang dapat bertahan pada generasi-generasi selanjutnya setelah melalui proses seleksi pada *class* Kromosom.

d. *Class* Optimasi

Class optimasi berisi main *method* yang berfungsi untuk melakukan perubahan jumlah populasi dan jumlah generasi. *Class* ini juga berfungsi untuk menampilkan soslusi terbaik hasil perhitungan algoritma genetika.

e. *Class* Data

Class data merupakan *class* yang berfungsi untuk melakukan deklarasi pembacaan data pada *file excel* yang akan digunakan untuk perhitungan dalam algoritma genetika.

5.2 Kode Program

Kode program *(Source code)* yang akan ditampilkan pada bab ini merupakan potongan-potongan program yang telah diimplementasikan berdasarkan alur kerja algoritma genetika.

5.2.1 Kode Program Pembentukan Kromosom

Dalam algoritma genetika langkah awal yang dilakukan adalah proses representasi kromosom. Kromosom tersebut merupakan solusi yang akan didapatkan dalam pemanfaatan algoritma genetika untuk optimasi *multi-dimensional knapsack problem*. Potongan *source code* yang akan dijelaskan pada sub bab ini diantaranya adalah pembangkitan individu dan perhitungan nilai *fitness* setiap individu.

5.2.1.1 Kode Program Pembangkitan Kromosom (Individu)

Pembangkitan kromosom pada penelitian ini akan dibentuk menjadi sebuah matrik sesuai dengan jumlah alat muat dan jenis produk. Baris pada matrik kromosom mewakili jenis alat muat dan kolom mewakili jenis produk. Kromosom yang terbentuk merupakan angka acak yang akan digunakan dalam perhitungan proporsi volume produk produk. Proses pembangkitan kromosom dapat dilihat pada *Source Code* 5.1.

```
1   public Kromosom() throws BiffException, IOException {
2           bacaData();
3           krom = new int[9][10];
4           for (int i = 0; i < krom.length; i++) {
5               for (int j = 0; j < krom[i].length; j++) {
6                   krom[i][j] = (int) (Math.random() * 100);

7               }
8           }
9   }
```

***Source Code* 5.1 Pembangkitan Kromosom**

Keterangan *source code* :

1. Baris 2 memanggil *method* bacaData() yang berupa *file excel.*
2. Inisialisasi array krom yang berupa matrik 9x10.
3. Baris 4 – 6 mengisi matrik dengan angka random Antara 0 – 100.

5.2.1.2 Kode Program Perhitungan Proporsi Volume menjadi Jumlah Produk

Solusi yang dihasilkan pada penelitian ini adalah berupa jumlah produk pada setiap alat muat. Jumlah produk ini didapatkan melalui perhitungan matrik kromosom yang terbentuk dengan menggunakan perbandingan nilai maksimal

Keterangan *source code* :

1. *Method* prppVolMax() berfungsi untuk menghitung proporsi volume produk yang dimuat pada setiap alat muat berdasarkan volume maksimal tiap alat muat.
2. Baris 4-17 merupakan perulangan untuk menghitung proporsi volume produk.
3. *Method* jmlProdukVol() berfungsi untuk menghitung jumlah produk yang berdasarkan volume maksimal alat muat.
4. Baris 27-33 merupakan perulangan umtuk menghitung jumlah setiap produk dalan setiap alat muat.
5. *Method* probBeratMax() berfungsi untung menghitung total berat produk pada setiap setiap alat muat.
6. Baris 44-50 merupakan perulangan untuk menghitung berat seluruh produk yang dimuat pada setiap alat muat.
7. Baris 52-57 merupakan perulangan untuk menghitung total berat produk pada setiap alat muat.
8. *Method* ratioBerat() berfungsi untuk menghitung rasio berat jika berat total produk yang dimuat melebihi kapasitas maksimal beban alat muat.
9. *Method* jmlProdukFinal() berfungsi untuk menghitung jumlah produk yang sesungguhnya tanpa melebihi beban maksimal kendaraan.
10. *Method* cekBerat() berfungsi untuk melakukan pengecekan apakah berat produk yang dimuat melebihi kapasitas beban maksimal, jika yam aka akan dilakukan perhitungan rasio berat untuk menanggulanginya.

5.2.1.3 Kode Program Perhitungan Nilai *Fitness*

Perhitungan nilai *fitness* pada penelitian ini adalah mencari nilai maksimum, yaitu nilai volume maksimum yang terisi pada seluruh alat muat yang digunakan. *Source code* 5.3 merupakan perhitungan volume maksimum yang terisi untuk digunakan sebagai nilai *fitness* setiap individu.

```
1   public double volMax() {
2           double[][] volMax = new
    double[krom.length][krom[0].length];
3           double[] totVolMaxAM = new double[krom.length];
4           totVolMax = 0;
5           for (int i = 0; i < krom.length; i++) {
6               for (int j = 0; j < krom[i].length; j++) {
7                   volMax[i][j] = ((double)
    jml_ProdukFinal[i][j]) * Prod[j][1];
8                   // System.out.print(volMax[i][j] + "||");
9               }
                //System.out.println();
10          }
```

```
11          for (int i = 0; i < krom.length; i++) {
12              for (int j = 0; j < krom[i].length; j++) {
13                  totVolMaxAM[i] += (double) volMax[i][j];
14              }
15              //System.out.println("Volume Max Am = " +
    totVolMaxAM[i]);
16              totVolMax += totVolMaxAM[i];
17          }
18          // System.out.println("Volume Maksimal = " +
    totVolMax);
19          return totVolMax;
20      }
21
    public double fitness() {
22          this.panggilSequenceMethod();
23          double fitness = totVolMax - totalPenalty;
24  //          System.out.print("Fitness : "+fitness);
25  //          System.out.println("");
26          return fitness;


27      }
```

Source Code* 5.2 Perhitungan *Fitness

Keterangan *source code* :

1. *Method* volMax() berfungsi untuk menghitung volume produk final yang dimuat.
2. Baris 5-9 berfungsi untuk melakukan perhitungan volume produk pada setiap alat muat.
3. Baris 11-17 berfungsi untuk menghitung total volume produk pada seluruh alat muat.
4. *Method* fitness() berfungsi untuk menghitung nilai *fitnees*.

5.2.2 Kode Program Pembangkitan Populasi Awal

Setelah melakukan proses pembangkitan individu, proses selanjutnya pada algoritma genetika adalah proses pembangkitan populasi awal. Pembangkitan populasi awal menentukan berapa banyak individu yang harus dibangkitkan sesuai ukuran populasi yang dideklarasikan. Proses pembangkitan populasi awal dapat dilihat pada *Source code* 5.4.

```
1   public Populasi(int popSize, boolean init) throws
    BiffException, IOException {
```

```
2           this.popSize = popSize;
3           kromosom = new Kromosom[popSize];
4           if (init) {//init menentukak apakah populasi itu awal
    atau populasi baru yang terbentuk
5               for (int i = 0; i < popSize; i++) {
6                   Kromosom kromBaru = new Kromosom();
7                   //menyimpan kromosom
8                   simpanKromosom(i, kromBaru);
9               }
10          }
11      }
```

***Source Code* 5.3 Pembangkitan Populasi**

Keterangan *source code* :

1. Baris 1 menunjukan parameter yang digunakan dalam konstruktor Populasi, yaitu *popSize* dengan tipe data int dan *boolean* init.
2. Baris 4 berfungsi untuk menunjukan kondisi populasi, apakah populasi tersebut merupakan populasi awal atau populasi baru yang terbentuk.
3. Baris 5-8 berfungsi untuk membangkitkan individu sesuai dengan jumlah *popSize* yang telah ditentukan.

5.2.3 Kode Program Proses Reproduksi

Dalam algoritma genetika, setelah dilakukan pembangkitan populasi maka proses selanjutnya adalah melakukan reproduksi untuk menghasilkan generasi berikutnya. Proses reproduksi dilakukan menggunakan 2 teknik yaitu teknik *crossover* dan mutasi. Proses reproduksi dapat dilihat pada *Source Code* 5.5.

```
1   private static Kromosom[] reproduksi(Populasi pop) throws
    BiffException, IOException {
2           Kromosom[] anak = new Kromosom[((int)
    (Math.round(mutationRate * pop.popSize))) + ((int)
    (Math.round(crossoverRate * pop.popSize)))];
            //offspring crosover sebanyak cr*popSize
3           int indeks = 0;
4           for (int i = 0; i < Math.round(crossoverRate *
    pop.popSize); i++) {
                //pemilihan parent
5               Kromosom parent1;
6               Kromosom parent2;
//              do {
```

```
7               parent1 = pilihParent(pop);
8               parent2 = pilihParent(pop);
9   ////                System.out.println("Crossover parent");
10  ////                            System.out.println("p1  =  ");
    parent1.printKrom();
11  ////                            System.out.println("p2  =  ");
    parent2.printKrom();
12  ////                            System.out.println("p2  =  " +
    parent2.toString());
    //              } while (parent1 == parent2);


13              //crossover
14              Kromosom offspring = crossover(parent1, parent2);
15              //child di simpan di populasi baru
16  //              System.out.println("Hasil Crossover ke-" + i +
    " = " + );
17              anak[indeks] = offspring;
18              indeks++;
19          }
20          //mutasi
21          for  (int  i  =  0;  i  <  Math.round(mutationRate  *
    pop.popSize); i++) {
22              //pemilihan parent
23              Kromosom parent3 = pilihParent(pop);
24              //mutasi
25              Kromosom offspring = mutasi(parent3);
26              anak[indeks] = offspring;
27              indeks++;
28          }
29          return anak;
30      }
31      private static Kromosom pilihParent(Populasi pop) {
32          int    randomParent    =    (int)    (Math.random()    *
33  pop.popSize);
            Kromosom dipilih = pop.getKromosom(randomParent);
34          return dipilih;
35      }
```

***Source Code* 5.4 Reproduksi**

Keterangan *source code* :

1. Baris 2 berfungsi untuk menentukan jumlah anak keseluruhan dari hasil *crossover* dan mutasi.
2. Baris 4-19 berfungsi untuk menentukan jumlah anak *crossover* dan pemilihan *parent crossover.*
3. Baris 21 berfungsi untuk menentukan jumlah anak dan pemilihan *parent* mutasi.
4. Baris 22-35 *method* pilihParent() berfungsi untuk memilih parent yang akan di-*crossover* dan dimutasi secara acak.

5.2.3.2 Kode Program Proses *Crossover*

Proses *crossover* pada penelitian ini menggunakan teknik *one cut point*. Setiap melakukan *crossover* akan menghasilkan 1 anak. Proses *crossover* akan terus dilakukan hingga anak yang dihasilkan mencukupi. Proses *crossover* dapat dilihat pada *Source code* 5.6.

```
1   private static Kromosom crossover(Kromosom parent1,
    Kromosom parent2) throws BiffException, IOException {
2           parent1 = new Kromosom();//objek untuk anak
    pertama
3   //         System.out.println(" Masuk crossover");
4         // System.out.println("Parent 1 :");
5           //   parent1.printKrom();
6           // System.out.println("Parent 2 :");
7           //  parent2.printKrom();
8           //milih point
9           int indekB = (int) (parent1.kromosomBaris() *
    Math.random());
10          int indekK = (int) (parent1.kromosomKolom() *
    Math.random());
11  //         System.out.println("indeks B = " + indekB);
12  //         System.out.println("indeks K = " + indekK);

13          //mengcrossoverkan
14          for (int i = indekB; i < parent1.kromosomBaris();
    i++) {
15              if (i == indekB) {
16                  for (int j = indekK; j <
    parent1.kromosomKolom(); j++) {
17                      parent1.krom[i][j] =
18  parent2.krom[i][j];
19                  }
                } else {
```

```
20                  for (int j = 0; j <
    parent1.kromosomKolom(); j++) {
21                      parent1.krom[i][j] =
22  parent2.krom[i][j];
23                  }
24              }
25          }
26          Kromosom k = parent1;
27  //          System.out.println("Child 1 :");
28  //          parent1.printKrom();
29          return k;
        }
```

Source Code* 5.5 Proses *Crossover

Keterangan *source code* :

1. Baris 9-10 berfungsi untuk memilih titik dimana akan dilakukan *crossover.* Baris 9 berfungsi memilih barisnya dan baris 10 memilih kolomnya.
2. Baris 14-24 berfungsi untuk melakukan penukaran sesuai dengan titik yang telah dipilih.
3. Baris 25-28 berfungsi untuk menyimpan anak yang dihasilkan melalui proses *crossover*.

5.2.3.3 Kode Program Proses Mutasi

Mutasi yang dilakukan pada penelitian ini menggunakan teknik *rechiprocal exchange mutation*. Setiap melakukan proses mutasi anak yang dihasilkan adalah 1. Proses mutasi akan terus diulang hingga jumlah anak mencukupi sejumlah *popSize*. Proses mutasi dapat dilihat pada *Source code* 5.7.

```
1   private static Kromosom mutasi(Kromosom parent3) throws
    BiffException, IOException {
2           parent3 = new Kromosom();//objek untuk anak hasil
    mutasi
3   //          System.out.println(" Masuk Mutasi");
4           //milih point 1 pada baris kolom
5   //          System.out.println("Parent 3 :");
6   //          parent3.printKrom();
7           int indekB1 = (int) (parent3.kromosomBaris() *
    Math.random());
8           int indekK1 = (int) (parent3.kromosomKolom() *
    Math.random());
9   //          System.out.println("indeks B1 = " + indekB1);
```

```
10  //          System.out.println("indeks K1 = " + indekK1);
11              int temp = parent3.krom[indekB1][indekK1];
12  //          System.out.println("temp : " + temp);
13              //milih point 2 pada baris kolom
14              int indekB2 = (int) (parent3.kromosomBaris() *
    Math.random());
15              int indekK2 = (int) (parent3.kromosomKolom() *
    Math.random());
16  //          System.out.println("indeks B1 = " + indekB1);
17  //          System.out.println("indeks K1 = " + indekK1);
18  //          System.out.println("indeks B2 = " + indekB2);
19  //          System.out.println("indeks K2 = " + indekK2);
20  //          System.out.println("satunya " +
    parent3.krom[indekB2][indekK2]);


21              parent3.krom[indekB1][indekK1] =
    parent3.krom[indekB2][indekK2];
22              parent3.krom[indekB2][indekK2] = temp;
23              Kromosom k = parent3;
24  //            System.out.println("Child 2 :");
25  //            parent3.printKrom();


26              return k;
27          }
```

***Source Code* 5.6 Proses Mutasi**

Keterangan *source code* :

1. Baris 7-8 berfungsi untuk memilih titik pertama dimana akan dilakukan pertukaran gen. Baris 7 berfungsi memilih barisnya dan baris 8 memilih kolomnya.
2. Baris 11 berfungsi menyimpan indeks titik pertama yang terpilih
3. Baris 14-15 berfungsi untuk memilih titik kedua dimana akan dilakukan pertukaran gen. Baris 14 berfungsi memilih barisnya dan baris 15 memilih kolomnya.
4. Baris 21 berfungsi untuk memasukan gen titik yang ke dua ke dalam titik yang ke pertama
5. Baris 22 berfungsi untuk memasukkan nilai temp yang berisi indeks titik pertama ke dalam titik ke dua

5.2.4 Kode Program Proses Seleksi

Seleksi yang dilakukan pada penelitian ini adalah seleksi *elitism*. Seleksi ini memastikan individu dengan nilai *fitness* tinggi yang dapat bertahan pada

generasi berikutnya. Dalam seleksi *elitism* terjadi pengurutan individu berdasarkan nilai *fitness*. Pengurutan yang dilakukan disini adalah menggunakan teknik *merge sort*. Proses seleksi dapat dilihat pada *Source code* 5.8.

```
1  public static Populasi elitism(Populasi pop, Kromosom[] child)
   throws BiffException, IOException {
2          Populasi populasiBaru = new Populasi(pop.popSize(),
   false);
3          //menggabungkan seluruh individu baru yaitu anak dan
   parent
4          Kromosom popParentChild[] = new Kromosom[pop.popSize()
   + child.length];
5          //menyimpan nilai indeks dan fitnessnya
6          double nilaiPopParentChild[][] = new
   double[pop.popSize() + child.length][2];
7          //perulangan memasukan individu ke populasi yg sudah
   ada
8          // System.out.println("Indisidu yg terbentuk : ");
9          for (int i = 0; i < pop.popSize(); i++) {
10             popParentChild[i] = pop.getKromosom(i);
11             nilaiPopParentChild[i][0] = i;
12             nilaiPopParentChild[i][1]
   popParentChild[i].fitness();
13             // System.out.println("Kromosom -" + i + " = " +
   nilaiPopParentChild[i][1]);
14         }
15         int indeks = pop.popSize();
16         for (int i = 0; i < child.length; i++) {
17             popParentChild[indeks] = child[i];
18             nilaiPopParentChild[indeks][0] = indeks;
19             nilaiPopParentChild[indeks][1]
   popParentChild[indeks].fitness();
20             // System.out.println("Kromosom -" + indeks + "
   " + nilaiPopParentChild[indeks][1]);
21             indeks++;
22         }
23         sort(nilaiPopParentChild);
24         //seleksi
25         // System.out.println("Hasil Seleksi : ");
26         for (int i = 0; i < populasiBaru.popSize(); i++) {
27             populasiBaru.simpanKromosom(i, (Kromosom)
   popParentChild[(int) nilaiPopParentChild[i][0]]);
```

```
            //  System.out.println("Kromosom -" + i + " = " +
28  nilaiPopParentChild[i][1]);
29          }
30          return populasiBaru;
31      }
```

Source Code* 5.7 Seleksi *Elitism

Keterangan *source code* :

1. Baris 4 berfungsi untuk menggabungkan seluruh individu yaitu anak dan *parent*.
2. Baris 6 berfungsi menyimpan indeks dan nilai *fitness* individu.
3. Baris 23 berfungsi untuk melakukan pengurutan individu berdasarkan nilai *fitness*
4. Baris 25-31 berfungsi untuk memasukkan kromosom terpilih ke dalam populasi baru

5.2.5 Kode Program Proses Pembacaan Data

Class Data merupakan *class* yang berfungsi untuk melakukan pembacaan data yang dibutuhkan dalam perhitungan algoritma genetika. Pada *class* Data hanya melakukan deklarasi pengambilan data pada *cell* excel. Setelah itu pembacaan data deklarasikan pada *class* optimasi dan *class* kromosom. Proses pembacaan data dapat dilihat pada *Source Code* 5.9.

```
1   private void bacaData() {
2   //        System.out.println("ini lagi baca data");
3           File Baca = new File("F:\\Data\\Book1.xls");
4           Workbook workbook = null;
5           try {
6               workbook = Workbook.getWorkbook(Baca);
7               Sheet alatMuat = workbook.getSheet(0);
8               Sheet Produk = workbook.getSheet(1);
9               int i = 0;
10  //
    System.out.println("alatMuat.getColumns()"+alatMuat.getColumns
    ());
11  //Cell merk = alatMuat.getCell(3, 1);
12  //            System.out.println("cek : "+merk.getContents());
```

```
13              for (int j = 1; j <= 9; j++) {
14                  Cell merk = alatMuat.getCell(0, j);
15                  Cell nama = alatMuat.getCell(1, j);
16                  Cell beban = alatMuat.getCell(2, j);
17                  Cell vol = alatMuat.getCell(3, j);
18                  Data.addDataAm(i, 0, merk.getContents());
19                  Data.addDataAm(i, 1, nama.getContents());
20  //
    System.out.println("..."+beban.getContents());
21                  Data.addAm(i, 0,
    Double.valueOf(beban.getContents()));
22                  Data.addAm(i, 1,
    Double.valueOf(vol.getContents()));
23                  i++;
24              }
25              i = 0;
26              for (int j = 1; j <= 10; j++) {
27                  Cell nama = Produk.getCell(0, j);
28                  Cell berat = Produk.getCell(1, j);
29                  Cell vol = Produk.getCell(2, j);
30  //                  Cell target = Produk.getCell(3, j);
31                  Data.addDataProd(i, 0, nama.getContents());
32  //
    Data.addTarProd(i,Integer.valueOf(target.getContents()) );

33  //System.out.println("..."+target.getContents());
34                  Data.addProd(i, 0,
    Double.valueOf(berat.getContents()));
35                  Data.addProd(i, 1,
    Double.valueOf(vol.getContents()));
36                  i++;
37              }
```

```
38          } catch (IOException e) {
39              e.printStackTrace();
40          } catch (BiffException e) {
41              e.printStackTrace();
42          } finally {
43              if (workbook != null) {
44                  workbook.close();
45              }
46          }
```

***Source Code* 5.8 Baca Data**

Keterangan *source code* :

1. Baris 3 berfungsi untuk mendeklarasikan lokasi *file* yang akan dibaca.
2. Baris 5-9 berfungsi deklarasi pembacaan *workbook* pada *sheet* pertama dan *sheet* kedua.
3. Baris 13-24 berfungsi untuk melakukan pembacaan data alat muat.
4. Baris 25-37 berfungsi untuk melakukan pembacaan data produk.

5.2.6 Kode Program Class Optimasi

Class Optimasi merupakan *class* yang berfungsi untuk melakukan running program yang berisi penentuan nilai parameter yaitu *cr*, *mr*, ukuran populasi, dan jumlah generasi. Kode program pada *class* Optimasi dapat dilihat pada *Source Code* 5.10.

```
1   public class Optimasi {
2       public static void main(String[] args) throws
    BiffException, IOException {
3           Optimasi Knapsack = new Optimasi();
4           Knapsack.optimasi_MKDP(0.4,0.6, 120, 70);
5       }
6       public void optimasi_MKDP(double cr, double mr, int
    popSize, int generasi) throws BiffException, IOException {
7           bacaData();
8           Populasi p = new Populasi(popSize, true);

9   System.out.println("====================================");
10          System.out.println("individu terbaik pada generasi 1
    dengan fitnes =" + p.getFitnessTerbaik().fitness());
11          /*simpan data popawal*/
```

```
12          File file = new File("F:\\Data\\PopAwal.txt");
13          if (file == null) {
14              try {
15                  new
File("F:\\Data\\PopAwal.txt").createNewFile();
16              } catch (IOException e) {
17              }
18              file = new File("F:\\Data\\PopAwal.txt");
19          }
20          try {
21              try (BufferedWriter f = new BufferedWriter(new
FileWriter(file, false))) {
22                  for (int i = 0; i < p.popSize(); i++) {
23                      f.write(p.getKromosom(i).krom());
24                      f.newLine();
                    }
                }
25          } catch (IOException e) {
26          }
27          /*proses perulangan generasi*/
28          file = new File("F:\\Data\\hasilTerbaik.txt");
29          if (file == null) {
30              try {
31                  new
File("F:\\Data\\hasilTerbaik.txt").createNewFile();
32              } catch (IOException e) {
33              }
34              file = new File("F:\\Data\\hasilTerbaik.txt");
            }
35          try {
36              try (BufferedWriter f = new BufferedWriter(new
FileWriter(file, false))) {
37                  for (int i = 0; i < generasi; i++) {
38                      System.out.println("GENERASI KE-" + (i +
1));
39                      if (i % 10 == 0) {
40                          Kromosom k = new Kromosom();
41                          p.simpanKromosom(((int)
Math.random() * popSize), k);
42                          Kromosom k2 = new Kromosom();
                            p.simpanKromosom(((int)
```

```
43  Math.random() * popSize), k2);
                        p = AG.pembangkitanPopulasi(p, cr,
44  mr);
                    } else {
45                      p = AG.pembangkitanPopulasi(p, cr,
46  mr);
                    }
                    System.out.println("Terbaik Sementara =
47  " + p.getFitnessTerbaik().fitness());
                    System.out.println("dengan kromosom =
48  ");
49                  p.getFitnessTerbaik().printKrom();
50                  f.write((i + 1) + ";" +
    p.getFitnessTerbaik().fitness() + ";" +
    p.getFitnessTerbaik().krom());
51                  f.newLine();
                }
            }
52      } catch (IOException e) {
53          System.out.println("ERROR");
54      }
55      System.out.println("Fitness Solution:");
56      System.out.println(p.getFitnessTerbaik().fitness());
57      System.out.println("Dengan kromosom = ");
58      p.getFitnessTerbaik().printKrom();
59      file = new File("F:\\Data\\hasilAkhir.txt");
60      if (file == null) {
61          try {
62              new
    File("F:\\Data\\hasilAkhir.txt").createNewFile();
63          } catch (Exception e) {
64          }
65          file = new File("F:\\Data\\hasilAkhir.txt");
66      }
67      try {
68          BufferedWriter hA = new BufferedWriter(new
    FileWriter(file, false));
68          hA.write(p.getFitnessTerbaik().Solusi());
69          hA.newLine();
70          hA.close();
71      } catch (Exception e) {
```

```
72          }
73      }
74          private void bacaData() {
75          //open file Am
            int line = 0;
76          File file = new File("F://Data/AM.txt");
77          try {
78              BufferedReader b = new BufferedReader(new
FileReader(file));
79              String arg = b.readLine();
80              while (arg != null) {
81                  Data.addJmlAm(line, Integer.valueOf(arg));
82                  arg = b.readLine();
83                  line++;
84              }
85          } catch (Exception e) {
86          }
87          Data.setJumlahAm((line));
88          System.out.println("ALAT MUAT ==
"+Data.getJumlahAm());
            /*batas*/
88          File Baca = new File("F:\\Data\\Book11.xls");
89          Workbook workbook = null;
90          try {
91              workbook = Workbook.getWorkbook(Baca);
92              Sheet alatMuat = workbook.getSheet(0);
93              Sheet Produk = workbook.getSheet(1);
94              int i = 0;
95              for (int j = 0; j < Data.getJumlahAm(); j++) {
95                  Cell merk = alatMuat.getCell(0,
Data.getJmlAm(j));
96                  Cell nama = alatMuat.getCell(1,
Data.getJmlAm(j));
97                  Cell beban = alatMuat.getCell(2,
Data.getJmlAm(j));
98                  Cell vol = alatMuat.getCell(3,
Data.getJmlAm(j));
99                  Data.addDataAm(i, 0, merk.getContents());
100                 Data.addDataAm(i, 1, nama.getContents());
101                 System.out.println("Alat Muatnya =
"+nama.getContents());
```

```
                    Data.addAm(i, 0,
Double.valueOf(beban.getContents()));
102                 Data.addAm(i, 1,
Double.valueOf(vol.getContents()));
                    i++;
103             }
                i = 0;
104             for (int j = 1; j <= 10; j++) {
105                 Cell nama = Produk.getCell(0, j);
                    Cell berat = Produk.getCell(1, j);
106                 Cell vol = Produk.getCell(2, j);
107
108                 Data.addDataProd(i, 0, nama.getContents());
109                 Data.addProd(i, 0,
Double.valueOf(berat.getContents()));
110                 Data.addProd(i, 1,
Double.valueOf(vol.getContents()));
111                 i++;
112             }
        } catch (IOException e) {
113             e.printStackTrace();
114     } catch (BiffException e) {
115             e.printStackTrace();
116     } finally {
117             if (workbook != null) {
118                 workbook.close();
119             }
120     }
121
122
```

Source Code 5.9 *Class* Optimasi

Keterangan *source code* :

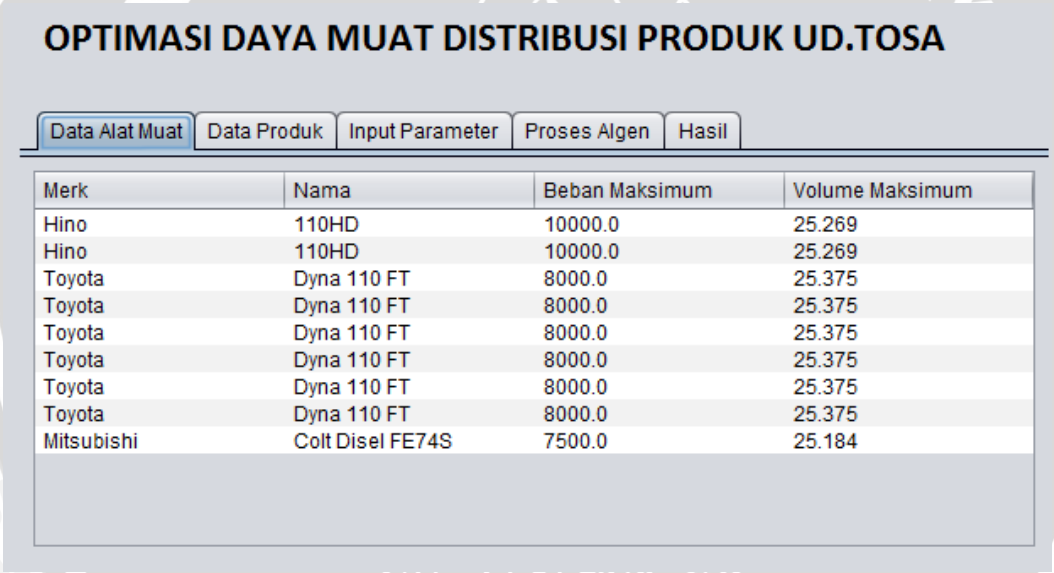
1. Baris 1-5 berfungsi untuk mendeklarasikan parameter algoritma genetika.
2. Baris 6-10 berfungsi melakukan pembangkitan populasi awal
3. Baris 11-26 berfungsi untuk menyimpan data populasi
4. Baris 27-55 berfungsi melakukan perulangan generasi. Pada baris 39-43 merupakan percabangan bahwa pada setiap generasi ke-I mod 10 maka akan dilakukan pengacakan ulang gen pada 2 kromosom.
5. Baris 55-59 berfungsi menampilkan *fitness* solusi beserta kromosomnya.
6. Baris 60-74 berfungsi menyimpan solusi kedalam file hasilAkhir.txt
7. Baris 75-104 berfungsi membaca data alat muat yang dipilih
8. Baris 105-122 berfungsi membaca data produk

5.3 Implementasi Antarmuka Pengguna (User Interface)

Antar muka pengguna dari program optimasi daya muat dalam pendistribusian produk UD.TOSA terdiri dari beberapa tab menu yang memiliki fungsi berbeda-beda. Tab menu yang tersebut diantaranya adalah tab data alat muat, tab data produk, tab input parameter, tab proses algoritma genetika, dan tab hasil. Implementasi antar muka program akan dijelaskan lebih detail pada sub bab berikut :

5.3.1 Antarmuka Tab Data Alat Muat

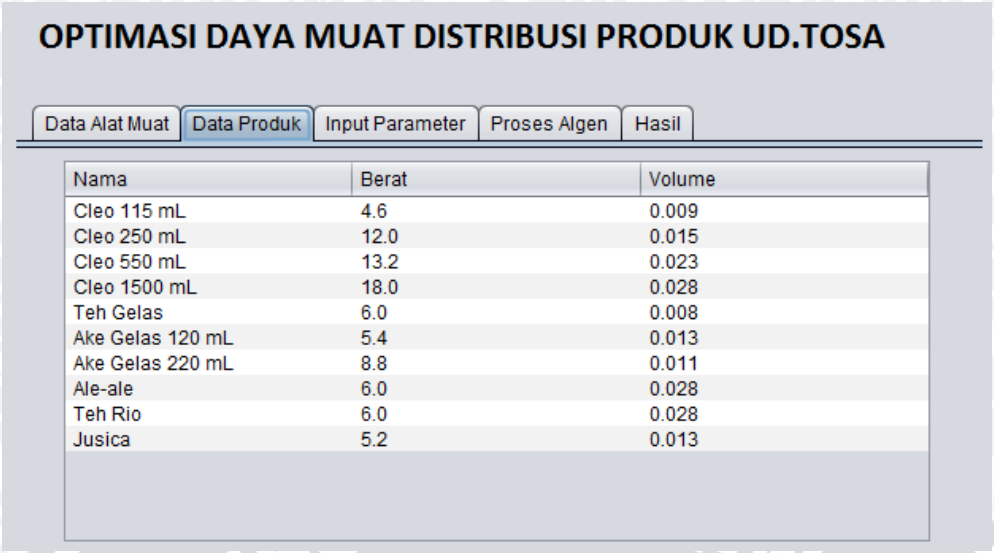
Data alat muat merupakan data utama yang digunakan dalam perhitungan optimasi *knapsack problem*. Alat muat merupakan *knapsack* (wadah) yang digunakan untuk menampuk objek berupa produk. Terdapat 9 alat muat yang digunakan dalam proses droping oleh UD.TOSA. Data alat muat yang dimiliki menampilkan merk alat muat, nama alat muat, kapasitas beban maksimum dan volume maksimum alat muat. Implementasi antarmuka halaman data alat muat dapat dilihat pada Gambar 5.1.



Gambar 5.1 Antarmuka Halaman Data Alat Muat

5.3.2 Antarmuka Tab Data Produk

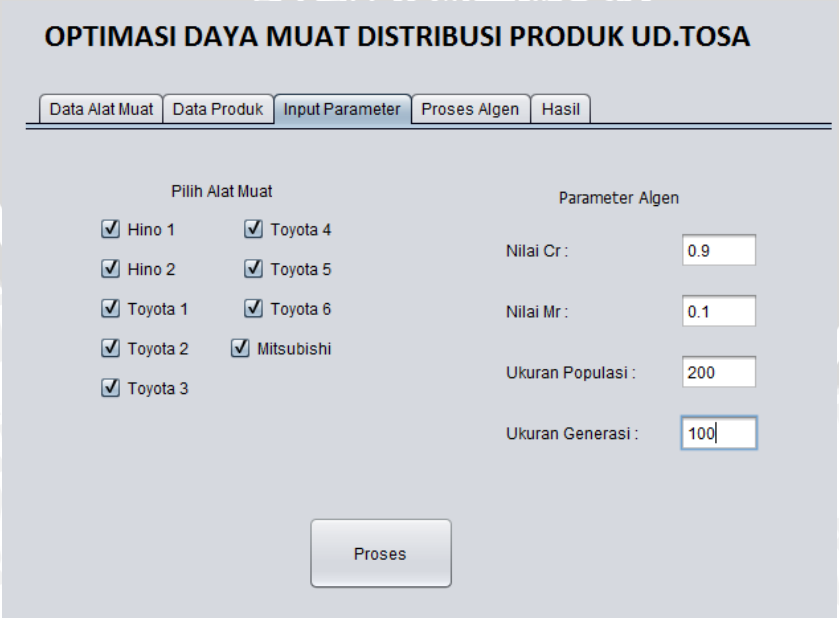
Data produk merupakan data utama yang berkaitan dengan muatan yang akan masukkan kedalam *knapsack*. Terdapat 10 alat muat yang selalu didistribusikan oleh UD.TOSA. Data produk yang dimiliki menampilkan nama produk, berat produk, dan volume produk. Implementasi antarmuka halaman data produk dapat dilihat pada Gambar 5.2.



Gambar 5.2 Antarmuka Halaman Daftar Produk

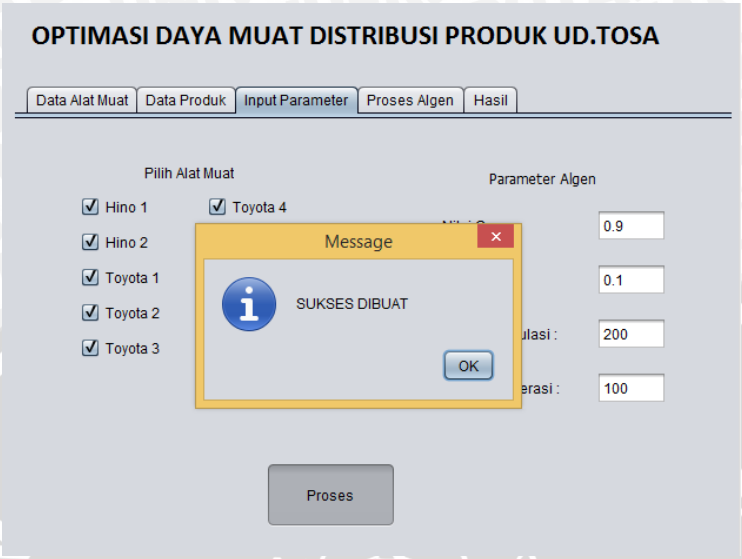
5.3.3 Antarmuka Tab Input Parameter

Halaman input parameter adalah tab yang berfungsi untuk memilih alat muat dan mengisi parameter-parameter yang digunakan dalam perhitungan optimasi menggunakan algoritma genetika. Pada halaman ini terdapat *chek box* untuk memilih alat muat dan *field* parameter yang harus diisi terlebih dahulu sebelum melakukan proses perhitungan. sebelum melakukan proses perhitungan. Parameter-parameter tersebut adalah nilai cr, nilai mr, ukuran populasi, dan banyaknya generasi. Terdapat sebuah proses untuk melakukan perhitungan menggunakan algoritma genetika sesuai dengan nilai parameter yang telah dimasukkan. Antarmuka halaman input parameter dapat dilihat pada Gambar 5.3.



Gambar 5.3 Antarmuka Halaman Input Parameter

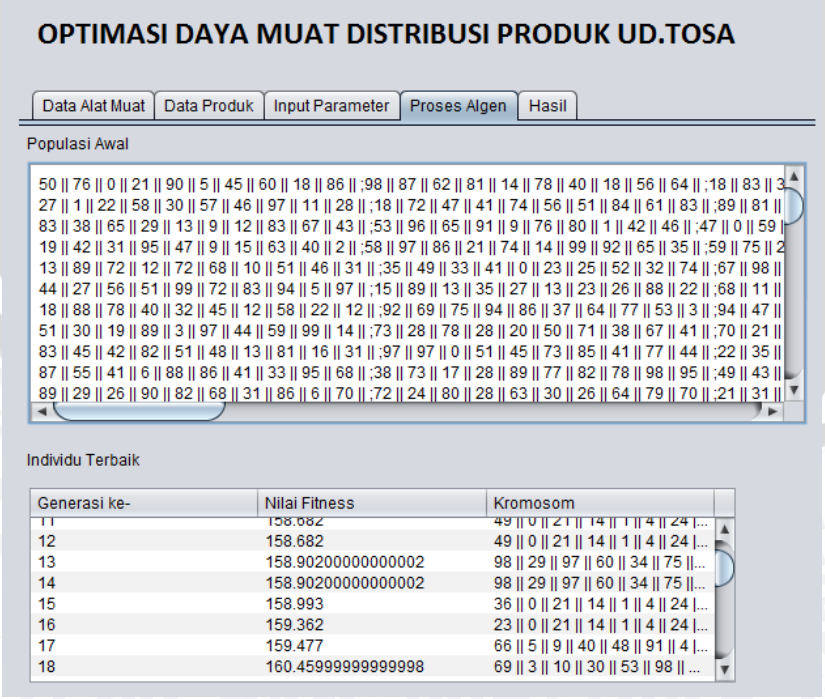
Setelah menekan tombol proses maka akan muncul pemberitahuan bahwa optimasi sesuai dengan parameter telah berhasil dibuat. Pemberitahuan yang muncul dapat dilihat pada Gambar 5.4.



Gambar 5.4 Antarmuka Pemberitahuan Perhitungan Berhasil

5.3.4 Antarmuka Tab Proses Algen

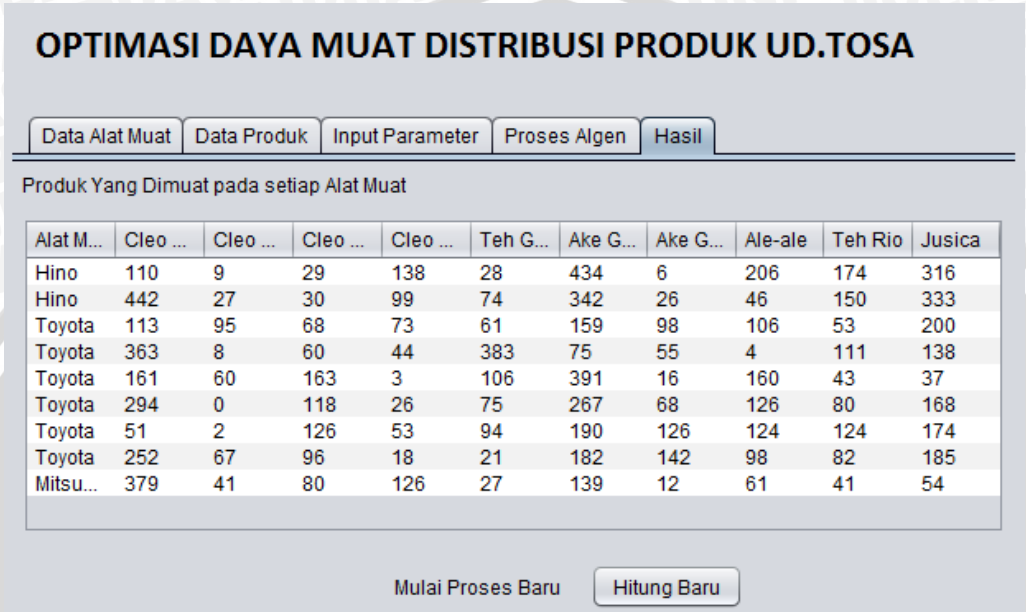
Setelah pemberitahuan bahwa perhitungan berhasil dilakukan maka proses algoritma genetika dapat dilihat pada tab proses algen. Pada halaman proses algen menampilkan contoh individu yang terbentuk pada pembangkitan awal proses algoritma genetika dan hasil nilai *fitness* terbaik pada setiap generasi. Antarmuka halaman proses algen dapat dapat dilihat pada Gambar 5.5.



Gambar 5.5 Antarmuka Halaman Proses Algen

5.3.5 Antarmuka Tab Hasil

Halaman hasil merupakan halaman yang menampilkan solusi permasalahan pada penelitian ini. Pada halaman hasil ditampilkan jumlah setiap jenis produk yang dimuat pada setiap alat muat. Pada halaman hasil *button* hitung baru yang berguna untuk melakukan perhitungan ulang tanpa harus memasukkan parameter lagi. Antarmuka halaman hasil dapat dilihat pada Gambar 5.6.



Gambar 5.6 Antarmuka Halaman Hasil

BAB 6 PENGUJIAN DAN ANALISIS

Pada bab ini dijelaskan mengenai pengujian-pengujian yang dilakukan dalam penelitian ini. Pengujian yang dilakukan adalah pengujian terhadap hasil implementasi program optimasi 0/1 *multi-dimensional knapsack problem* menggunakan algoritma genetika dalam pendistribusian produk di UD.TOSA.

6.1 Skenario Pengujian

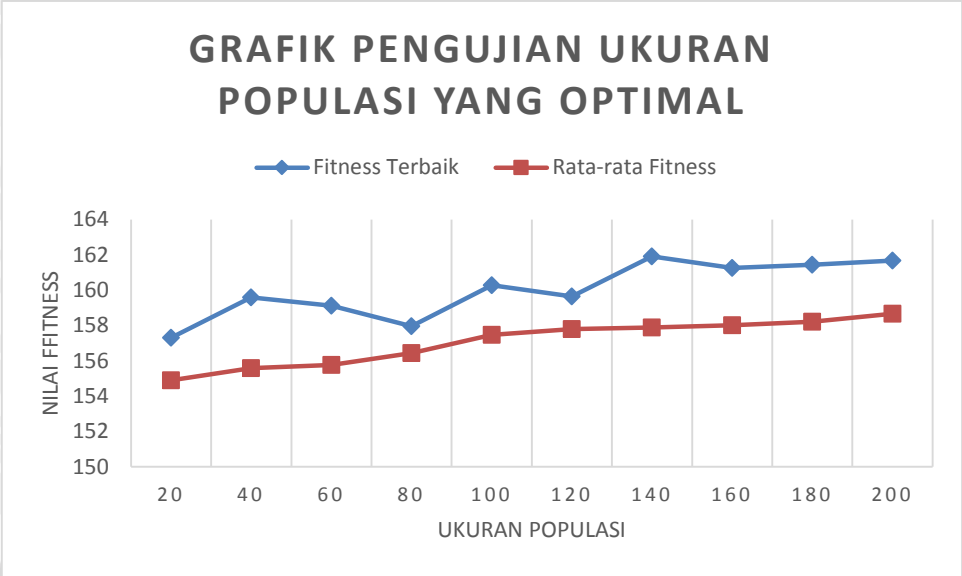
Pengujian yang dilakukan pada penelitian ini adalah pengujian terhadap parameter-parameter yang digunakan pada algoritma genetika dan pengujian perbandingan optimasi hasil. Terdapat 3 jenis pengujian yang dilakukan, yaitu pengujian terhadap ukuran populasi, banyaknya generasi, dan kombinasi nilai *cr* dan *mr*. Pengujian pertama yang dilakukan adalah menguji ukuran populasi untuk mendapatkan ukuran populasi yang optimal. Setelah diketahui ukuran populasi yang optimal maka pengujian selanjutnya adalah pengujian banyaknya generasi. Pengujian banyaknya generasi bertujuan untuk menentukan berapa banyak generasi yang optimal dengan menggunakan ukuran populasi yang paling optimal yang dihasilkan saat pengujian ukuran populasi. Setelah diketahui ukuran populasi dan banyaknya generasi yang optimal, maka pengujian selanjutnya adalah menentukan kombinasi nilai *cr* dan *mr*. Pengujian ini dilakukan untuk mengetahui kombinasi *cr* dan *mr* yang paling optimal sehingga didapatkan nilai *fitness* yang baik. Pada pengujian kombinasi nilai *cr* dan *mr* yang optimal digunakan ukuran populasi dan banyaknya generasi yang paling optimal dari pengujian yang dilakukan sebelumnya. Pengujian yang terakir adalah pengujian perbandingan hasil. Pada pengujian ini dilakukan perbandingan *overtonase* antara hasil *droping* asli UD.TOSA dan hasil *droping* sistem.

6.2 Hasil dan Analisis Pengujian Ukuran Populasi

Pengujian ukuran populsai merupakan pengujian pertama yang dilakukan dalam penelitian ini. Pengujian ukuran populasi bertujuan untuk mengetahui ukuran populasi yang paling optimal pada algoritma genetika. Hal ini dilakukan agar menghasilkan nilai *fitness* yang optimal. Pengujian ini dilakukan pada ukuran populasi dengan kelipatan dua puluh, dimulai dari 20 ukuran populasi sampai 200 ukuran populasi. Pengujian pada setiap ukuran populasi dilakukan sebanyak sepuluh kali. Banyaknya generasi yang digunakan dalam pengujian ini adalah sebanyak 10 generasi dan kombinasi nilai *cr* dan *mr* yang digunakan adalah 0,6 dan 0,3. Hasil pengujian ukuran populasi dapat dilihat pada Tabel 6.1.

Tabel 6.1 Hasil Pengujian Ukuran Populasi Optimal

Jumlah Populasi	Fitness										Fitness terbaik	Fitness Everage
	Percobaan ke-											
	1	2	3	4	5	6	7	8	9	10		
20	151.663	155.651	157.304	155.988	153.746	155.719	153.556	154.495	153.948	156.976	157.304	154.9046
40	154.052	159.587	156.762	152.339	156.13	155.128	155.236	153.555	159.392	153.756	159.587	155.5937
60	158.242	153.185	154.298	159.125	155.448	156.848	156.217	154.327	154.262	155.69	159.125	155.7642
80	154.52	157.953	157.756	156.773	156.97	157.849	155.369	155.488	155.032	156.701	157.953	156.4411
100	156.973	155.019	156.212	160.275	159.173	156.309	158.835	158.078	157.033	156.72	160.275	157.4627
120	158.96	157.068	159.648	157.262	155.081	155.81	159.567	155.958	159.604	158.957	159.648	157.7915
140	158.168	158.38	155.877	157.17	155.864	156.24	157.418	157.307	161.909	160.458	161.909	157.8791
160	157.256	158.896	157.154	157.39	156.174	157.794	157.638	158.803	161.256	157.752	161.256	158.0113
180	158.158	156.557	157.248	156.474	159.258	161.445	157.373	158.613	157.019	160.041	161.445	158.2186
200	160.623	158.272	158.123	158.54	161.674	157.465	157.4	157.586	159.46	157.51	161.674	158.6653



Gambar 6.1 Grafik Pengujian Ukuran Populasi

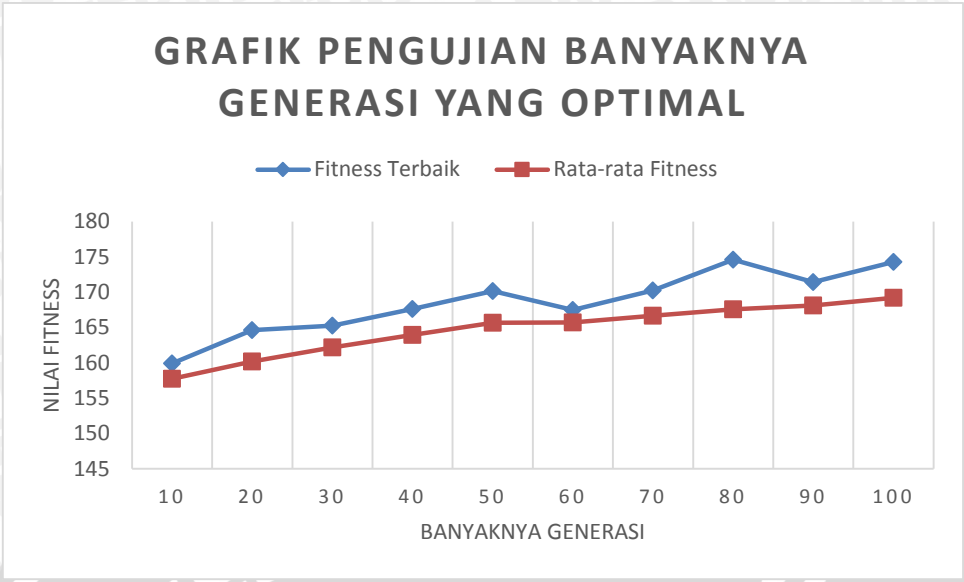
Gambar 6.1 merupakan grafik hasil pengujian ukuran populasi yang diperoleh dari Tabel 6.1. Pada grafik tersebut, sisi horisontal menunjukan jumlah populasi yang diuji dan sisi vertikal merupakan nilai *fitness* yang dihasilkan. Dilihat pada Gambar 6.1 dapat disimpulkan bahwa semakin banyak ukuran populasi maka semakin baik nilai *fitness* yang dihasilkan. Pada ukuran populasi nilai fitness yang dihasilkan sudah hamper konvergen, nilai rata-rata *fitness* terbaik dimiliki oleh populasi 200 yaitu sebesar 158,6653. Oleh karena rata-rata *fitness* tertinggi dihasilkan dengan populasi 200 maka pada pengujian selanjutnya ukuran populasi yang akan digunakan adalah sebanyak 200 populasi.

6.3 Hasil dan Analisis Pengujian Banyaknya Generasi

Pengujian banyaknya generasi merupakan pengujian kedua setelah dilakukan pengujian ukuran populasi. Pengujian banyaknya generasi bertujuan untuk mengetahui banyaknya generasi yang paling optimal pada algoritma genetika. Banyaknya generasi yang optimal dapat dilihat dari rata-rata nilai *fitness* tertinggi yang dihasilkan. Pengujian ini dilakukan pada banyak generasi dengan kelipatan sepuluh, dimulai dari 10 generasi sampai 100 generasi. Pengujian pada masing-masing generasi dilakukan sebanyak sepuluh kali. Ukuran populasi yang digunakan dalam pengujian ini adalah 200 populasi. Kombinasi nilai *cr* dan *mr* yang digunakan adalah 0,6 dan 0,3. Hasil pengujian banyaknya generasi dapat dilihat pada Tabel 6.2.

Tabel 6.2 Hasil Pengujian Banyaknya Generasi Optimal

Jumlah Generasi	Fitness										Fitness terbaik	Fitness Everage
	Percobaan ke-											
	1	2	3	4	5	6	7	8	9	10		
10	156.26	158.149	159.358	156.258	159.483	156.452	156.4	159.897	159.688	155.668	159.897	157.7613
20	159.744	159.746	164.645	157.974	160.278	159.169	159.877	161.998	158.493	160.015	164.645	160.1939
30	164.191	162.651	159.135	161.135	162.734	159.279	161.537	162.623	163.446	165.263	165.263	162.1994
40	163.032	165.728	162.199	163.17	165.411	161.3	167.638	162.085	165.517	163.167	167.638	163.9247
50	164.992	165.203	168.531	170.151	161.395	165.978	168.287	163.671	165.533	163.105	170.151	165.6846
60	163.974	167.329	164.076	167.474	166.959	162.452	166.448	165.623	167.02	165.683	167.474	165.7038
70	166.877	165.288	165.169	168.503	165.216	170.237	167.177	166.098	163.506	168.715	170.237	166.6786
80	166.889	163.353	167.394	166.717	167.87	174.605	166.133	167.233	168.212	167.319	174.605	167.5725
90	166.377	165.818	169.247	169.689	169.389	167.098	169.041	168.614	171.412	164.388	171.412	168.1073
100	174.291	167.893	173.511	166.745	168.029	168.775	167.516	169.277	166.482	169.606	174.291	169.2125



Gambar 6.2 Grafik Hasil Pengujian Banyaknya Generasi

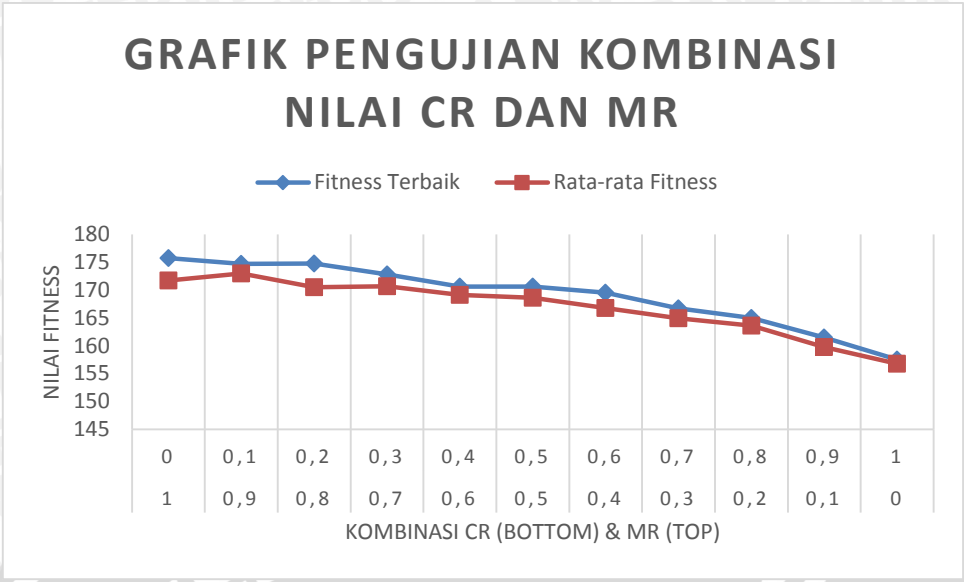
Gambar 6.2 merupakan grafik hasil pengujian banyaknya generasi yang diperoleh dari Tabel 6.2. Pada grafik tersebut, sisi horisontal merupakan banyaknya generasi yang diuji dan sisi vertikal merupakan nilai *fitness* yang dihasilkan. Dilihat dari Gambar 6.2 dapat disimpulkan bahwa semakin banyak generasi yang digunakan maka akan semakin baik nilai *fitness* yang dihasilkan. Nilai rata-rata *fitness* tertinggi diperoleh pada generasi ke-100 dengan nilai sebesar 169,2125. Oleh karena nilai rata-rata *fitness* tertinggi diperoleh pada generasi ke-100 maka jumlah generasi yang digunakan dalam pengujian selanjutnya adalah sebanyak 100 generasi.

6.4 Hasil dan Analisis Pengujian Kombinasi Nilai Cr dan Mr

Pengujian kombinasi nilai *cr* dan *mr* adalah pengujian terakhir yang dilakukan setelah diketahui ukran populasi dan banyaknya generasi yang optimal. Pengujian ini bertujuan untuk mengetahui kombinasi nilai *cr* dan *mr* yang optimal sehingga menghasilkan *fitness* yang optimal. Kombinasi nilai *cr* dan *mr* jika dijumlahkan harus menghasilkan nilai 1 sehingga populasi yang dihasilkan tetap stabil. Kombinasi nilai *cr* dan *mr* yang diuji adalah kelipatan 0,1 mulai dari 0 sampai 1. Percobaan dilakukan sebanyak 10 kali pada setiap kombinasi. Ukuran populasi yang digunakan pada pengujian ini adalah sebanyak 200 populasi dan banyaknya generasi yang digunakan adalah sebanyak 100 generasi. Hasil pengujian kombinasi *cr* dan *mr* dapat dilihat pada Tabel 6.3.

Tabel 6.3 Hasil Pengujian Kombinasi Nilai *cr* dan *mr*

Cr dan Mr		Nilai Fitness										Fitness terbaik	Fitness Everage
		Percobaan ke-											
Cr	Mr	1	2	3	4	5	6	7	8	9	10		
1	0	172.685	174.639	175.242	177.397	173.606	172.454	175.767	173.716	166.878	169.775	175.767	171.718
0,9	0,1	176.415	170.626	171.385	172.589	168.767	172.78	172.165	170.703	174.71	174.71	174.71	173.0136
0,8	0,2	170.976	173.53	172.817	169.67	170.783	170.413	168.154	171.001	168.412	174.777	174.777	170.5514
0,7	0,3	170.326	173.15	168.791	170.544	169.836	172.496	167.187	168.364	172.81	172.642	172.81	170.6998
0,6	0,4	168.066	169.07	172.268	169.794	168.854	169.007	168.127	170.627	167.848	170.054	170.627	169.1326
0,5	0,5	169.448	165.588	167.494	167.981	169.078	168.231	165.406	170.627	169.349	169.504	170.627	168.6234
0,4	0,6	162.389	164.508	166.449	165.269	167.954	169.22	165.352	166.207	163.531	169.553	169.553	166.7726
0,3	0,7	164.822	163.737	163.302	164.661	162.497	164.044	165.368	164.946	163.599	166.716	166.716	164.9346
0,2	0,8	162.701	164.369	163.502	165.56	161.251	162.179	162.065	164.464	165.019	164.321	165.019	163.6096
0,1	0,9	160.128	158.7	159.321	162.8	159.871	159.182	160.213	161.489	160.2	157.777	161.489	159.7722
0	1	157.077	155.781	155.38	154.851	156.355	157.507	156.213	155.852	156.784	157.497	157.507	156.7706



Gambar 6.3 Grafik Hasil Pengujian Kombinasi Nilai *cr* dan *mr*

Gambar 6.3 merupakan grafik hasil pengujian kombinasi nilai *cr* dan *mr* yang diperoleh dari Tabel 6.3. Berdasarkan Gambar 6.3 dapat disimpulkan bahwa semakin besar nilai *cr* maka akan semakin tinggi rata-rata *fitness* yang dihasilkan dan semakin tinggi nilai *mr* maka semakin kecil rata-rata *fitness* yang dihasilkan. Nilai rata-rata *fitness* tertinggi yang dihasilkan adalah 173,0136 diperoleh dari kombinasi *cr* dan *mr* sebesar 0,9 dan 0,1. Dari Dari hasil ini maka nilai *cr* dan *mr* yang digunakan seterusnya pada sistem adalah sebesar 0,9 dan 0,1.

6.5 Hasil dan Analisis Pengujian Perbandingan Hasil

Pengujian perbandingan optimasi hasil merupakan pengujian terakhir yang dilakukan dalam penelitian ini. Pengujian ini membandingkan hasil daya muat *droping* UD.TOSA dan *droping* yang dihasilkan sistem. Hasil *droping* UD.TOSA dapat dilihat pada Tabel 6.6 dan hasil *droping* yang diperoleh dari sistem dapat dilihat pada Tabel 6.8. Langkah pertama yang harus dilakukan adalah menghitung berat total muatan pada seluruh alat muat. Setelah diketahui berat total pada setiap alat muat, langkah selanjutnya adalah menghitung selisih berat. Untuk menghitung selisih berat digunakan persamaan 4.6 yang telah dijelaskan pada sub bab 4.4.4. Dalam perhitungan kelebihan beban muatan dibutuhkan nilai karakteristik alat muat dan produk. Karakteristik alat muat dapat dilihat pada Tabel 6.4 dan karakteristik produk dapat dilihat pada tabel 6.5.

Tabel 6.4 Karakteristik Alat Muat

Merk	Nama	Beban Max (Kg)	Volume Max (m^3)
Hino	110HD	10000	25,269
Hino	110HD	10000	25,269
Toyota	Dyna 110 FT	8000	25,375
Toyota	Dyna 110 FT	8000	25,375
Toyota	Dyna 110 FT	8000	25,375
Toyota	Dyna 110 FT	8000	25,375
Toyota	Dyna 110 FT	8000	25,375
Toyota	Dyna 110 FT	8000	25,375
Mitsubishi	Colt Disel FE74S	7500	25,184

Tabel 6.5 Karakteristik Produk

Nama	Berat (Kg)	Volume (m^3)
Cleo 115 mL	4,6	4,6
Cleo 250 mL	12	12
Cleo 550 mL	13,2	13,2
Cleo 1500 mL	18	18
Teh Gelas	6	6
Ake Gelas 120 mL	5,4	5,4
Ake Gelas 220 mL	8,8	8,8
Ale-ale	6	6
Teh Rio	6	6
Jusica	5,2	5,2

Berikut merupakan proses perhitungan yang dilakukan untuk melakukan pengujian hasil:

1. Prosentase Selisih Beban Hasil *Droping* UD.TOSA

   Berdasarkan proses droping yang dilakukan oleh UD.TOSA, didapatkan hasil muatan produk seperti pada Tabel 6.6. Dalam praktek keseharian proses *droping* UD.TOSA sudah memanfaatkan ruang alat muat dengan baik, produk yang dimuat tidak pernah melebihi kapasitas ruang yang dimiliki. Pada pengujian ini akan dilakukan pengujian terhadap kelebihan beban muatan pada setiap alat muat.

Tabel 6.6 Hasil *Droping* UD.TOSA

Muatan (kardus)	Cleo 115 mL	Cleo 250 mL	Cleo 550 mL	Cleo 1500 mL	Teh Gelas	Ake Gelas 120 mL	Ake Gelas 220 mL	Ale-ale	Teh Rio	Jusica
WU342R-HKMRHD3/110HD	308	248	83	96	150	173	65	139	127	101
WU342R-HKMRHD3/110HD	90	216	164	130	122	237	181	117	151	14
DYNA 110 FT	475	16	55	80	549	0	46	55	194	329
DYNA 110 FT	128	138	172	0	379	360	20	146	95	163
DYNA 110 FT	290	85	184	0	563	320	244	109	0	87
DYNA 110 FT	202	291	122	17	0	173	410	32	108	266
DYNA 110 FT	509	153	0	119	165	320	378	58	70	37
DYNA 110 FT	223	238	164	19	461	148	100	160	84	24
COLT DIESEL FE74S (4X2) M/T	231	183	101	98	230	277	129	115	62	136

Tabel 6.7 Prosentase Kelebihan Beban

Muatan (kardus)	Cleo 115 mL	Cleo 250 mL	Cleo 550 mL	Cleo 1500 mL	Teh Gelas	Ake Gelas 120 mL	Ake Gelas 220 mL	Ale-ale	Teh Rio	Jusica	total B	Selisih Beban	Sisa Beban Am (%)	Kelebihan Beban (%)	Rata-rata sisa beban (%)	Rata-rata kelebihan beban (%)
Hino 110 HD	2926	2480	747	1008	750	1038	390	695	635	454.5	11123.5	-1123.5	0	11.235		
Hino 110 HD	855	2160	1476	1365	610	1422	1086	585	755	63	10377	-377	0	3.77		
Toyota Dyana 110 FT	4512.5	160	495	840	2745	0	276	275	970	1480.5	11754	-3754	0	46.925		
Toyota Dyana 110 FT	1216	1380	1548	0	1895	2160	120	730	475	733.5	10257.5	-2257.5	0	28.21875		
Toyota Dyana 110 FT	2755	850	1656	0	2815	1920	1464	545	0	391.5	12396.5	-4396.5	0	54.95625	0	38.36148148
Toyota Dyana 110 FT	1919	2910	1098	178.5	0	1038	2460	160	540	1197	11500.5	-3500.5	0	43.75625		
Toyota Dyana 110 FT	4835.5	1530	0	1249.5	825	1920	2268	290	350	166.5	13434.5	-5434.5	0	67.93125		
Toyota Dyana 110 FT	2118.5	2380	1476	199.5	2305	888	600	800	420	108	11295	-3295	0	41.1875		
Mitsubishi Colt Disel	2194.5	1830	909	1029	1150	1662	774	575	310	612	11045.5	-3545.5	0	47.273333		
TOTAL													0	345.25333		

Pada Tabel 4.7 merupakan langkah perhitungan yang dilakukan hingga menghasilkan prosentase kelebihan beban pada hasil droping UD.TOSA. Berat muatan dan berat total didapatkan dari perkalian jumlah produk yang dimuat dengan berat produk yang terdapat pada tabel 6.5. Kolom selisih beban didapatkan menggunakan persamaan 4.6. Pada kolom selisih beban nilai yang dihasilkan adalah min (-), hal ini dapat diarktikan bahwa berat muatan melebihi kapasitas beban maksimal alat muat. Pada kolom rata-rata sisa beban, prosentase yang dihasilkan sebesar 0% karena tidak ada sisa beban yang dimiliki alat muat. Sedangkan pada kolom rata-rata kelebihan beban prosentasenya sebesar 38,36%. Hal ini dapat diartikan bahwa rata-rata kelebihan beban pada hasil droping UD.TOSA hampir mencapai separuh dari kapasitas beban maksimal alat muat.

2. Prosentase Selisih Beban Hasil *Droping* Sistem

Berdasarkan solusi droping yang dihasilkan oleh sistem, didapatkan hasil muatan produk seperti pada Tabel 6.8. Sistem yang dibuat memastikan untuk memaksimalkan kapasitas ruang tanpa melebihi beban maksimal alat muat. Pada pengujian ini akan dilakukan pengujian terhadap kelebihan beban muatan pada setiap alat muat untuk mengetahui apakah solusi yang dihasilkan oleh sistem lebih baik atau tidak.

Pada Tabel 4.9 merupakan langkah perhitungan yang dilakukan hingga menghasilkan prosentase kelebihan beban pada hasil droping sistem. Berat muatan dan berat total didapatkan dari perkalian jumlah produk yang dimuat dengan berat produk yang terdapat pada tabel 6.5. Kolom selisih beban didapatkan menggunakan persamaan 4.6. Pada kolom selisih beban nilai yang dihasilkan adalah plus (+), hal ini dapat diarktikan bahwa berat muatan tidak melebihi kapasitas beban maksimal alat muat. Pada kolom rata-rata sisa beban, prosentase yang dihasilkan berarti setiap alat muat masih mampu menambung berat sebanyak 0,41% lagi. Sedangkan pada kolom rata-rata kelebihan beban prosentasenya sebesar 0%. Hal ini membuktikan bahwa solusi yang dihasilkan sistem tidak melebihi kapasitas beban maksimal alat muat.

Tabel 6.8 Hasil *Droping* Sistem

Muatan (kardus)	Cleo 115 mL	Cleo 250 mL	Cleo 550 mL	Cleo 1500 mL	Teh Gelas	Ake Gelas 120 mL	Ake Gelas 220 mL	Ale-ale	Teh Rio	Jusica
WU342R-HKMRHD3/110HD	578	33	71	29	27	172	67	123	225	327
WU342R-HKMRHD3/110HD	49	166	13	7	13	349	240	340	111	134
DYNA 110 FT	169	11	63	0	313	26	131	105	146	296
DYNA 110 FT	495	26	70	28	24	329	0	126	52	181
DYNA 110 FT	341	28	11	37	96	58	95	125	135	376
DYNA 110 FT	324	105	29	0	198	107	60	103	92	264
DYNA 110 FT	278	159	20	52	14	161	160	39	103	70
DYNA 110 FT	435	132	8	10	77	244	21	4	107	283
COLT DIESEL FE74S (4X2) M/T	327	21	69	12	119	76	65	116	120	281

Tabel 6.9 Prosentase Kelebihan Beban

Muatan (kardus)	Cleo 115 mL	Cleo 250 mL	Cleo 550 mL	Cleo 1500 mL	Teh Gelas	Ake Gelas 120 mL	Ake Gelas 220 mL	Ale-ale	Teh Rio	Jusica	total B	Selisih Beban	Sisa Beban Am (%)	Kelebihan Beban (%)	Rata-rata sisa beban (%)	Rata-rata kelebihan beban (%)
Hino 110 HD	2658.8	396	937.2	522	162	928.8	589.6	738	1350	1700.4	9982.8	17.2	0.172	0	0.415462963	0
Hino 110 HD	225.4	1992	171.6	126	78	1884.6	2112	2040	666	696.8	9992.4	7.6	0.076	0		
Toyota Dyana 110 FT	777.4	132	831.6	0	1878	140.4	1152.8	630	876	1539.2	7957.4	42.6	0.5325	0		
Toyota Dyana 110 FT	2277	312	924	504	144	1776.6	0	756	312	941.2	7946.8	53.2	0.665	0		
Toyota Dyana 110 FT	1568.6	336	145.2	666	576	313.2	836	750	810	1955.2	7956.2	43.8	0.5475	0		
Toyota Dyana 110 FT	1490.4	1260	382.8	0	1188	577.8	528	618	552	1372.8	7969.8	30.2	0.3775	0		
Toyota Dyana 110 FT	1278.8	1908	264	936	84	869.4	1408	234	618	364	7964.2	35.8	0.4475	0		
Toyota Dyana 110 FT	2001	1584	105.6	180	462	1317.6	184.8	24	642	1471.6	7972.6	27.4	0.3425	0		
Mitsubishi Colt Disel	1504.2	252	910.8	216	714	410.4	572	696	720	1461.2	7456.6	43.4	0.578666667	0		
TOTAL													3.739166667	0		

Tabel 6.10 Perbandingan Prosentase Beban

Droping	Rata-rata sisa beban (%)	Rata-rata kelebihan beban (%)
UD.TOSA	0	38.36
Sistem	0.41	0

Berdasarkan Tabel 6.10 yang menampilkan prosentase rata-rata kelebihan beban yang dihasilkan dari hasil *droping* UD.TOSA dan sistem, maka dapat disimpulkan bahwa solusi *droping* yang dihasilkan oleh sistem lebih baik jika dibandingkan dengan hasil *droping* UD.TOSA.

BAB 7 PENUTUP

7.1 Kesimpulan

Kesimpulan yang didapatkan pada penelitian skripsi yang berjudul "Pemanfaatan Algoritma Genetika untuk Optimasi 0/1 *Multi-Dimensional Knapsack Problem* dalam Pendistribusian Produk (Studi Kasus UD.TOSA)" adalah sebagai berikut:

1. Pemanfaatan algoritma genetika dalam penelitian ini menggunakan representasi kromosom *integer*, representasi kromosom berbentuk matrik dimana kolom merepresentasikan jenis produk dan baris merepresentasikan alat muat. Perhitungan nilai *fitness* pada penelitian ini adalah menghitung nilai maksimum daya muat. Teknik reproduksi yang digunakan adalah *one cut point crossover* dan *reciprocal exchange mutation.* Proses seleksi individu dalam populasi dilakukan dengan menggunakan teknik *elitism selection*.
2. Ukuran parameter algoritma genetika yang digunakan dalam penelitian ini sangat berpengaruh terhadap nilai *fitness* yang dihasikan. Hal tersebut berarti bahwa ukran parameter algoritma genetika dapat mempengaruhi kualitas solusi yang ihasilkan oleh sistem. Dilihat dari hasil pengujian ukuran populasi dan banyaknya generasi yang digunakan, semakin banyak populasi dan generasi yang digunakan maka nilai fitness yang dihasilkan akan semakin tinggi. Dalam hal ini berarti bahwa semakin banyak populasi dan generasi yang digunakan maka pencrian solusi algoritma genetika akan semakin luas, namun semakin banyak populasi dan generasi yang digunakan juga dapat mengakibatkan lamanya waktu komputasi dan menyebabkan terjadinya konvergensi. Ukuran populasi dan banyaknya generasi yang optimal berdasarkan pengujian yang dilakukan adalah 200 populasi dengan nilai fitness sebesar 158,6653 dan 100 generasi dengan nilai fitness sebesar 169,2125. Berdasarkan pengujian kombinasi nilai cr dan mr yang dilakukan dapat disimpulkan bahwa semakin besar nilai cr maka *fitness* yang dihasilkan akan semakin besar dan sebaliknya semakin besar nilai mr maka *fitness* yang dihasilkan akan semakin kecil. Hasil pengujian kombinasi nilai cr dan mr yang optimal pada penelitian ini adalah 0,9 dan 0,1.
3. Nilai *fitness* yang dihasilkan pada sistem dapat digunakan untuk mengukur kualitas solusi yang dihasilkan. Semakin besar nilai fitness solusi maka semakin baik pula kualitas solusi tersebut. Berdasarkan pengujian perbandingan hasil yang telah dilakukan, dihasilkan prosentase kelebihan beban *droping* UD.TOSA sebesar 38,36% dan prosentase kelebihan beban *droping* sistem 0%. Dari hasil tersebut dapat disimpulkan bahwa solusi yang dihasilkan oleh sistem lebih baik jika dibandingkan dengan hasil *droping* asli UD.TOSA. Solusi yang dihasilkan oleh sistem dapat dipastikan tidak melebihi kapasitas alat muat, hal ini dapat mengurangi resiko kerusakan alat muat sehingga frekuensi *maintenance* tidak terlalu sering.

7.2 Saran

1. Penelitian ini dapat ditambahkan konstrain baru yaitu peletakan produk, pada penelitian ini tidak memperhatikan posisi peletakan produk sehingga apabila ditambahkan konstrain berupa peletakan produk diharapkan solusi yang dihasilkan akan semakin baik.
2. Penelitian ini dapat ditambahkan konstrain berupa target, pada penelitian ini belum memperhatikan target karena perhitungan optimasi daya muat menggunakan proporsi dan rasio.
3. Pada penelitian ini dapat dilakukan proses *crossover*, mutasi, dan seleksi menggunakan teknik yang lain dan memumngkinkan solusi yang dihasilkan bisa lebih baik.
4. Optimasi 0/1 *multi-dimensional knapsack problem* pada kasus ini dapat dikembangkan kedalam kasus lain seperti pengiriman produk ke pelanggan.

DAFTAR PUSTAKA

Alfinda, L. A., Soebroto, A. A., & Regasari, R. (2014). Sistem Pendukung Keputusan Pemilihan Penerima JAMKESMAS Menggunakan Metode Weighted Product. *DORO: Repository Jurnal Mahasiswa PTIIK Universitas Brawijaya*, vol. 3, no. 6.

Ardiansyah, M. A. (2014). Penerapan Model Transportasi dan Distribusi Vogel's Approximation Method (VAM) dan Modified Distribution (MODI) pada UD. Tani Berdikari. *Repository Universitas Hasanudin*.

Ariastuti, P. R. (2015). Penerapan Algoritma Genetika dan Algoritma Harmony Search pada Permasalahan KNAPSACK 0-1. *Digital Repository Universitas Jember*.

Aristoteles, W. d. (2015). Evaluasi Kinerja Genetic Algorithm (GA) dengan Strategi Perbaikan Kromosom Studi Kasus: Knapsack Problem. *Jurnal Komputasi*, 162.

Dimyanti, D. A. (2004). *Operation Research : Model-model Pengambilan Keputusan.* Bandung: Sinar Baru Agesindo.

Gen, M., & Cheng, R. (2000). *Genetic Algorithm and Engineering Optimization.* New York: John Willey and Son.

Kato, K. (2003). Genetic Algorithms With Decomposition Procedures for Multidimensional 0-1 Knapsack Problems With Block Angular Structures. *IEEE TRANSACTIONS ON SYSTEMS, MAN, AND CYBERNETICS—PART B: CYBERNETICS*, Vol.33, No.3.

Mahmudy, W. F. (2015). *Dasar-dasar Algoritma Evolusi.* Malang: Universitas Brawijaya.

Mian, Z. (2012). Meta-heuristics for Multidimensional Knapsack Problems. *4th International Conference on Computer Research and Development*.

Nugraha, D. C., & Mahmudy, W. F. (2015). Optimasi Vehicle Routing Problem with Time Windows pada Distribusi Katering Menggunakan Algoritma Genetika. *Institut Teknologi Sepuluh Nopember (ITS), Seminar Nasional Sistem Informasi Indonesia (SESINDO)*, Surabaya, 2-3 November, pp. 275-282.

Panharesi, Y. G., & Mahmudy, W. F. (2015). Optimasi Distribusi Barang dengan Algoritma Genetika. *DORO: Repository Jurnal Mahasiswa PTIIK Universitas Brawijaya*, vol. 5, no. 11.

Prihastuti, M. D., Soebroto, A. A., & Regasari, R. (2014). Aplikasi Sistem Pakar Untuk Pendeteksi dan Penanganan Dini Pada Penyakat Sapi Dengan Metode Dempster-Shafer Berbasis Web. *DORO: Repository Jurnal Mahasiswa PTIIK Universitas Brawijaya*, vol. 3, no. 6.

Putri, F. B., Mahmudy, W. F., & Ratnawati, D. E. (2015). Penerapan Algoritma Genetika Untuk Vehicle Routing Problem with Time Window (VRPTW)

Pada Kasus Optimasi Distribusi Beras Bersubsidi. *DORO: Repository Jurnal Mahasiswa PTIIK Universitas Brawijaya*, vol. 5, no. 1.

Sulistyorini, R., & Mahmudy, W. F. (2015). Penerapan Algoritma Genetika untuk Permasalahan Optimasi Distribusi Barang Dua Tahap. *DORO: Repository Jurnal Mahasiswa PTIIK Universitas Brawijaya*, vol. 5, no. 12.

Suyanto. (2007). *Artificial Intelligence Searching, Reasoning, Planning dan Learning.* Bandung: Informatika.

Triatmojo, W. G. (2017, Maret 10). *Overtonase Dan Apa Dampak Dari Muatan Berlebih Pada Truck Dan Jalan*. Diambil kembali dari Indotrucker: http://indotrucker.blogspot.com/2015/06/overtonase-dan-apa-dampak-dari-muatan.html

Tyas, R. A., Rahman, M. A., & Dewi, C. (2013). Implementasi Algoritma Genetika Untuk Optimasi 0/1 Multi-Dimensional Knapsack Problem Dalam Penentuan Menu Makanan Sehat. *DORO: Repository Jurnal Mahasiswa PTIIK Universitas Brawijaya*, vol. 1, no. 4.

Unal, A. N. (2013). A Genetic Algorithm for the Multiple Knapsack Problem in Dynamic Environment. *Proceedings of the World Congress on Engineering and Computer Science 2013 Vol II*.

Wahid, N., & Mahmudy, W. F. (2015). Optimasi Komposisi Makanan untuk Penderita Kolesterol Menggunakan Algoritma Genetika. *DORO: Repository Jurnal Mahasiswa PTIIK Universitas Brawijaya*, vol. 5, no. 15.

LAMPIRAN 1. MANUALISASI PROGRAM

Data Alat muat

Merk	Nama	Beban Max (Kg)	Volume Max (m^3)
Hino	110HD	10000	25.269
Hino	110HD	10000	25.269
Toyota	Dyna 110 FT	8000	25.375
Toyota	Dyna 110 FT	8000	25.375
Toyota	Dyna 110 FT	8000	25.375
Toyota	Dyna 110 FT	8000	25.375
Toyota	Dyna 110 FT	8000	25.375
Toyota	Dyna 110 FT	8000	25.375
Mitsubishi	Colt Disel FE74S	7500	25.184

Data Produk

Produk	1	2	3	4	5	6	7	8	9	10
Berat (kg)	4.6	12	13.2	18	6	5.4	8.8	6	6	5.2
Volume(m^3)	0.008602	0.015345	0.022815	0.027712	0.007986	0.013376	0.01102	0.0277 12125	0.027712	0.013376

***STEP* 1**

Melakukan *random* bilangan *integer* antara 0 – 100 untuk dijadikan gen dalam kromosom. Gen dalam kromosom bukan jumlah produk yang dimuat melainkan bilangan acak yang akan digunakan sebagai bilagan dalam penentuan proporsi volume produk.

Kromosom 1	1	2	3	4	5	6	7	8	9	10
1	22	99	74	2	97	17	59	20	11	86
2	82	94	12	9	57	96	90	51	23	30
3	76	80	29	52	74	6	27	96	75	83
4	58	99	44	28	23	51	10	14	36	45
5	6	18	43	49	48	23	36	59	93	99
6	43	58	55	78	98	5	56	93	100	78
7	23	52	4	78	65	71	14	45	40	98
8	94	17	58	91	46	0	1	94	61	19
9	27	36	0	30	98	58	18	78	93	41

***STEP* 2**

Menjumlah gen kolom 1-10 pada kromosom tiap-tiap barisnya.

kromosom 1	jumlah
1	423
2	401
3	434
4	400
5	325
6	639
7	390
8	492
9	530

***STEP* 3**

Melakukan perhitungan proporsi volume produk dengan menggunakan persamaan $Vol_{ij} = \frac{Gen_j}{\sum Gen\, j} \times VMax_i$. Proporsi volume produk yang dihasilkan berupa satuan meter kubik.

vol ij	1	2	3	4	5	6	7	8	9	10
1	1.141 515	5.136 819	3.839 643	0.103 774	5.033 045	0.882 08	3.061 337	1.037 741	0.570 758	4.462 287
2	3.808 93	4.366 335	0.557 404	0.418 053	2.647 671	4.459 235	4.180 533	2.368 969	1.068 358	1.393 511
3	3.224 916	3.394 649	1.230 56	2.206 522	3.140 05	0.254 599	1.145 694	4.073 579	3.182 483	3.521 948
4	3.607 23	6.157 169	2.736 52	1.741 422	1.430 453	3.171 875	0.621 936	0.870 711	2.238 971	2.798 713
5	0.321 203	0.963 608	2.301 951	2.623 154	2.569 62	1.231 276	1.927 215	3.158 492	4.978 639	5.299 842
6	1.643 261	2.216 491	2.101 845	2.980 798	3.745 105	0.191 077	2.140 06	3.554 029	3.821 536	2.980 798
7	1.191 071	2.692 857	0.207 143	4.039 286	3.366 071	3.676 786	0.725	2.330 357	2.071 429	5.075
8	4.958 94	0.896 83	3.059 771	4.800 676	2.426 715	0	0.052 755	4.958 94	3.218 035	1.002 339
9	1.419 557	1.892 743	0	1.577 286	5.152 468	3.049 42	0.946 372	4.100 944	4.889 587	2.155 624

***STEP* 4**

Menghitung jumlah produk yang dimuat sesuai dengan hasil proporsi volume menggunakan persamaan Vol_{ij} / $VolProduk_j$. Satuan produk yang dihasilkan yaitu kardus.

jml produk	1	2	3	4	5	6	7	8	9	10
1	132.7035	334.7552	168.2947	3.744719	630.2336	65.94498	277.7983	37.44719	20.59596	333.604
2	442.7959	284.5444	24.43149	15.08557	331.5391	333.3758	379.3587	85.48492	38.55202	104.18
3	374.9031	221.2218	53.93645	79.62297	393.1944	19.03399	103.965	146.9963	114.8408	263.3035
4	419.3479	401.2492	119.9439	62.8397	179.1201	237.1318	56.43705	31.41985	80.7939	209.2339
5	37.34045	62.79619	100.8964	94.65727	321.7656	92.05116	174.8834	113.9751	179.6556	396.2202
6	191.0324	144.4439	92.12557	107.563	468.9589	14.28505	194.1978	128.2481	137.9012	222.8468
7	138.4645	175.4876	9.07924	145.7588	421.4965	274.8793	65.78947	84.09161	74.7481	379.4109
8	576.4868	58.44441	134.1123	173.2338	303.8712	0	4.787176	178.9448	116.1237	74.93562
9	165.0264	123.3459	0	56.91682	645.1875	227.9769	85.87764	147.9837	176.4421	161.1561

***STEP* 5**

Melakukan pembulatan kebawah terhadap jumlah produk yang dihasilkan.

jml produk	1	2	3	4	5	6	7	8	9	10
1	132	334	168	3	630	65	277	37	20	333
2	442	284	24	15	331	333	379	85	38	104
3	374	221	53	79	393	19	103	146	114	263
4	419	401	119	62	179	237	56	31	80	209
5	37	62	100	94	321	92	174	113	179	396
6	191	144	92	107	468	14	194	128	137	222
7	138	175	9	145	421	274	65	84	74	379
8	576	58	134	173	303	0	4	178	116	74
9	165	123	0	56	645	227	85	147	176	161

***STEP* 6**

Menghitung berat produk untuk mengecek apakan jumlah muatan yang dihasilkan dari proporsi volume produk melebihi kapasitas beban alat muat. Perhitungan berat produk dilakukan dengan persamaan $Jml\ produk_{ij}$ X $Berat\ produk_j$ kemudian dihitung total berat pada setiap alat muat dengan menjumlahkan berat produk yang dihasilkan. Satuan berat produk yang dihasilkan adalah Kilogram (Kg).

berat produk	1	2	3	4	5	6	7	8	9	10	total
1	607.2	4008	2217.6	54	3780	351	2437.6	222	120	1731.6	15529
2	2033.2	3408	316.8	270	1986	1798.2	3335.2	510	228	540.8	14426.2
3	1720.4	2652	699.6	1422	2358	102.6	906.4	876	684	1367.6	12788.6
4	1927.4	4812	1570.8	1116	1074	1279.8	492.8	186	480	1086.8	14025.6
5	170.2	744	1320	1692	1926	496.8	1531.2	678	1074	2059.2	11691.4
6	878.6	1728	1214.4	1926	2808	75.6	1707.2	768	822	1154.4	13082.2
7	634.8	2100	118.8	2610	2526	1479.6	572	504	444	1970.8	12960
8	2649.6	696	1768.8	3114	1818	0	35.2	1068	696	384.8	12230.4
9	759	1476	0	1008	3870	1225.8	748	882	1056	837.2	11862

STEP 7

Lakukan perhitungan rasio jika total berat produk melebihi kapasitas alat muat menggunakan persamaan $Rasio\ berat = \frac{DT_i}{(\sum Jmlproduk_{ij})\, B_j}$

ratio berat	
1	0.65255
2	0.530546
3	0.614959
4	0.577451
5	0.523286
6	0.565371
7	0.539884
8	0.693091
9	0.458127

***STEP* 8**

Menghitung jumlah produk final yang tidak melebihi kapasitas ruang dan beban maksimal alat muat dengan menggunakan persamaan $Produk\ final = Rasio\ berat_i \times Jmlproduk_{ij}$. Satuan jumlah produk final yang dimuat adalah kardus.

jml produk	1	2	3	4	5	6	7	8	9	10
1	85.00225	215.0815	108.1847	1.931869	405.6926	41.85717	178.3759	23.82639	12.87913	214.4375
2	306.387	196.864	16.6364	10.39775	229.4437	230.83	262.7164	58.92058	26.34096	72.09106
3	233.9584	138.2481	33.15453	49.41901	245.844	11.88559	64.43239	91.33134	71.31351	164.5215
4	238.9916	228.7246	67.87588	35.36391	102.099	135.1814	31.94159	17.68195	45.63085	119.2106
5	25.31775	42.42435	68.42636	64.32078	219.6486	62.95226	119.0619	77.32179	122.4832	270.9684
6	116.7999	88.05858	56.25965	65.43242	286.1904	8.561251	118.6345	78.2743	83.77796	135.757
7	85.18519	108.0247	5.555556	89.50617	259.8765	169.1358	40.12346	51.85185	45.67901	233.9506
8	376.7661	37.93825	87.65044	113.1606	198.1947	0	2.616431	116.4312	75.8765	48.40398
9	104.3247	77.76935	0	35.40718	407.8149	143.5255	53.74305	92.94385	111.2797	101.7956

***STEP* 9**

Lakukan pembulatan kebawah terhadap produk final yang dihasilkan.

Final Produk	1	2	3	4	5	6	7	8	9	10
1	85	215	108	1	405	41	178	23	12	214
2	306	196	16	10	229	230	262	58	26	72
3	233	138	33	49	245	11	64	91	71	164
4	238	228	67	35	102	135	31	17	45	119
5	25	42	68	64	219	62	119	77	122	270
6	116	88	56	65	286	8	118	78	83	135
7	85	108	5	89	259	169	40	51	45	233
8	376	37	87	113	198	0	2	116	75	48
9	104	77	0	35	407	143	53	92	111	101

STEP 10

Hitung volume maksimal hasil muatan pada seluruh alat muat yang digunakan untuk dijadikan nilai *fitness*. Perhitungan volume produk menggunakan persamaan $JmlProdukFinal_{ij}$ X $volProduk_j$. Satuan volume yang dihasilkan adalah meter kubik.

Vol Final	1	2	3	4	5	6	7	8	9	10	total
1	0.73117	3.299175	2.46402	0.027712	3.23433	0.548416	1.96156	0.637379	0.332546	2.862464	16.09877
2	2.632212	3.00762	0.36504	0.277121	1.828794	3.07648	2.88724	1.607303	0.720515	0.963072	17.3654
3	2.004266	2.11761	0.752895	1.357894	1.95657	0.147136	0.70528	2.521803	1.967561	2.193664	15.72468
4	2.047276	3.49866	1.528605	0.969924	0.814572	1.80576	0.34162	0.471106	1.247046	1.591744	14.31631
5	0.21505	0.64449	1.55142	1.773576	1.748934	0.829312	1.31138	2.133834	3.380879	3.61152	17.20039
6	0.997832	1.35036	1.27764	1.801288	2.283996	0.107008	1.30036	2.161546	2.300106	1.80576	15.3859
7	0.73117	1.65726	0.114075	2.466379	2.068374	2.260544	0.4408	1.413318	1.247046	3.116608	15.51557

8	3.234352	0.567765	1.984905	3.13147	1.581228	0	0.02204	3.214607	2.078409	0.642048	16.45682
9	0.894608	1.181565	0	0.969924	3.250302	1.912768	0.58406	2.549516	3.076046	1.350976	15.76976
Volume Total											143.8336

LAMPIRAN 2. DROPING ASLI UD.TOSA

Muatan	Cleo 115 mL	Cleo 250 mL	Cleo 550 mL	Cleo 1500 mL	Teh Gelas	Ake Gelas 120 mL	Ake Gelas 220 mL	Ale-ale	Teh Rio	Jusica
WU342R-HKMRHD3/110HD	308	248	83	96	150	173	65	139	127	101
WU342R-HKMRHD3/110HD	90	216	164	130	122	237	181	117	151	14
DYNA 110 FT	475	16	55	80	549	0	46	55	194	329
DYNA 110 FT	128	138	172	0	379	360	20	146	95	163
DYNA 110 FT	290	85	184	0	563	320	244	109	0	87
DYNA 110 FT	202	291	122	17	0	173	410	32	108	266
DYNA 110 FT	509	153	0	119	165	320	378	58	70	37
DYNA 110 FT	223	238	164	19	461	148	100	160	84	24
COLT DIESEL FE74S (4X2) M/T	231	183	101	98	230	277	129	115	62	136

LAMPIRAN 3. DROPING HASIL SISTEM

Muatan	Cleo 115 mL	Cleo 250 mL	Cleo 550 mL	Cleo 1500 mL	Teh Gelas	Ake Gelas 120 mL	Ake Gelas 220 mL	Ale-ale	Teh Rio	Jusica
WU342R-HKMRHD3/110HD	578	33	71	29	27	172	67	123	225	327
WU342R-HKMRHD3/110HD	49	166	13	7	13	349	240	340	111	134
DYNA 110 FT	169	11	63	0	313	26	131	105	146	296
DYNA 110 FT	495	26	70	28	24	329	0	126	52	181
DYNA 110 FT	341	28	11	37	96	58	95	125	135	376
DYNA 110 FT	324	105	29	0	198	107	60	103	92	264
DYNA 110 FT	278	159	20	52	14	161	160	39	103	70
DYNA 110 FT	435	132	8	10	77	244	21	4	107	283
COLT DIESEL FE74S (4X2) M/T	327	21	69	12	119	76	65	116	120	281

LAMPIRAN 4. ASET KENDARAAN UD.TOSA

No.	Nopol	Merk	Type	Tahun Pembuatan	No. BPKB
1	N 9212 RC	MITSUBISHI	L300 PU FB-R	2011	I-06453903
2	N 8779 UC	HINO	WU342R-HKMRHD3/110HD	2013	J-06383926
3	N 8776 UC	HINO	WU342R-HKMRHD3/110HD	2013	J-06383925
4	N 8157 UC	MITSUBISHI	COLT DIESEL FE74S (4X2) BUT	2011	I-04039296
5	N 8358 UA	TOYOTA	DYNA 110 FT	2013	K-06750364
6	N 8357 UA	TOYOTA	DYNA 110 FT	2013	K-06750365
7	N 9432 UC	TOYOTA	DYNA 110 FT	2014	L-07737909
8	N 9431 UC	TOYOTA	DYNA 110 FT	2014	L-07737908
9	N 8342 AU	TOYOTA	DYNA 110 ST	2014	L-04896356
10	N 8341 AU	TOYOTA	DYNA 110 ST	2014	L-04896355
11	N 8783 AU	SUZUKI	ST150 PICK UP	2014	L-11740918
12	N 8911 G	MITSUBISHI	L300	2006	D-9503195
13	N 4660 CB	YAMAHA	407 VEGA-R	2007	E-5405434
14	N 716 EB	HONDA, ACCORD	SMAGM	1993	A-1275560
15	N 2650-BT	TOYOTA	INNOVA-G	2005	D-5125367
16	N 2685 02	YAMAHA	4D7 (VEGA-R)	2008	F-0885579
17	N 7863 H	MITSUBISHI	FE 304	2004	D-1288949
18	N 9398 CH	SUZUKI	ST150	2009	F-9112485
19	N 9445 K	MITSUBISHI	COLT	2007	E-7426177
20	N 9438 CI	MITSUBISHI	FE71 4X2 MT	2010	G-3725932
21	N 7949 ED	MITSUBISHI	L300	2005	D-3650400
22	N 8421 CA	DAIHATSU	S401RP PMREJJ HA	2012	K-02997186
23	N 8159 UC	MITSUBISHI	COLT DIESEL FE74S (4X2) M/T	2011	I-04039297
24	N 1949 BC	TOYOTA	AVANZA 1300 G	2011	I-03525013
25	N 8366 BD	SUZUKI	ST150	2016	0-5671856
26	N 8367 BD	SUZUKI	ST150	2016	0-5671857
27	N 8368 BD	SUZUKI	ST150	2016	0-5671858
28	N 8369 BD	SUZUKI	ST150	2016	0-5671859
29	N 8371 BD	SUZUKI	ST150	2016	0-5671860
30	N 8372 BD	SUZUKI	ST150	2016	0-5671861
31	N 8401 BD	SUZUKI	ST150	2016	0-8543872
32	N 8402 BD	SUZUKI	ST150	2016	0-8543873
33	N 8403 BD	SUZUKI	ST150	2016	0-8543874
34	N 8404 BD	SUZUKI	ST150	2016	0-8543875
35	N 8405 BD	SUZUKI	ST150	2016	0-8543876
36	N 8407 BD	SUZUKI	ST150	2016	0-8543877