OPTIMASI DISTRIBUSI MULTI-PRODUCT MENGGUNAKAN ALGORITMA GENETIKA (STUDI KASUS : PENGIRIMAN MAKANAN BEKU)

SKRIPSI

Laboratorium Komputasi Cerdas dan Visualisasi

Untuk memenuhi sebagian persyaratan
untuk meraih gelar Sarjana Komputer

Disusun Oleh
Sumayyah Ula Syahidah
NIM. 115060807111118

KEMENTRIAN RISET TEKNOLOGI DAN PENDIDIKAN TINGGI
PROGRAM STUDI INFORMATIKA
PROGRAM TEKNOLOGI INFORMASI DAN ILMU KOMPUTER
UNIVERSITAS BRAWIJAYA
MALANG
2015

LEMBAR PERSETUJUAN

OPTIMASI DISTRIBUSI MULTI-PRODUCT MENGGUNAKAN ALGORITMA GENETIKA

(STUDI KASUS : PENGIRIMAN MAKANAN BEKU)

SKRIPSI

LABORATORIUM KOMPUTASI CERDAS DAN VISUALISASI

Untuk memenuhi sebagian persyaratan mencapai gelar Sarjana Komputer

Disusun oleh :
SUMAYYAH ULA SYAHIDAH
NIM. 115060807111118

Skripsi ini telah diperiksa dan disetujui oleh dosen pembimbing

Dosen Pembimbing I

Wayan Firdaus Mahmudy, S.Si., M.T., Ph.D.
NIP. 19720919 199702 1 001

LEMBAR PENGESAHAN

OPTIMASI RUTE DISTRIBUSI MULTI-PRODUCT MENGGUNAKAN ALGORITMA GENETIKA (STUDI KASUS : PENGIRIMAN MAKANAN BEKU)

SKRIPSI

Diajukan untuk memenuhi persyaratan memperoleh gelar Sarjana Komputer

Disusun Oleh :

SUMAYYAH ULA SYAHIDAH

NIM. 115060807111118

Skripsi ini telah diuji dan dinyatakan lulus pada tanggal 6 Juli 2015

Dosen Penguji I **DosenPenguji II**

Agus Wahyu Widodo, S.T.M.Cs **Hurriyatul Fitriyah, S.T, M.Sc**
NIP. 19740805 200112 1 001 **NIK. 2013048510012001**

Dosen Penguji III

Mochammad Hannats Hanafi l., S.ST, M.T
NIK. 201405 881229 1 1 001

Mengetahui,

Ketua Program Studi Informatika / Ilmu Komputer

Drs. Mardji, M.T.
NIP. 19670801 199203 1 001

PERNYATAAN ORISINALITAS SKRIPSI

Saya menyatakan dengan sebenar-benarnya bahwa sepanjang pengetahuan saya, di dalam naskah SKRIPSI ini tidak terdapat karya ilmiah yang pernah diajukan oleh orang lain untuk memperoleh gelar akademik di suatu perguruan tinggi dan tidak terdapat karya atau pendapat yang pernah ditulis atau diterbitkan oleh orang lain, kecuali yang secara tertulis dikutip dalam naskah ini dan disebutkan dalam sumber kutipan dan daftar pustaka.

Apabila ternyata didalam naskah SKRIPSI ini dapat dibuktikan terdapat unsur-unsur PLAGIASI, saya bersedia SKRIPSI ini digugurkan dan gelar akademik yang telah saya peroleh (SARJANA) dibatalkan, serta diproses sesuai dengan peraturan perundang-undangan yang berlaku. (UU No. 20 Tahun 2003, Pasal 25 ayat 2 dan Pasal 70).

Malang,25 Juli 2015

Mahasiswa

Sumayyah Ula Syahidah

NIM. 115060800111111

Abstrak

Sumayyah Ula Syahidah. 2015 : Optimasi Distribusi Multi-Product Menggunkana Algoritma Genetika (Studi Kasus : Pengiriman Makanan Beku). Program Teknologi Informasi dan Ilmu Komputer, Universitas Brawijaya, Malang.

Dosen Pembimbing : Wayan Firdaus Mahmudy, S.Si., M.T., Ph.D

Sebuah perusahaan yang bergerak di bidang industri tidak lepas dari masalah pendistribusian barang. Distribusi barang menjadi tonggak dalam pentingnya suatu perusahaan industri dalam kelancaran operasionalnya. Proses distribusi juga merupakan proses yang penting yang membutuhkan biaya yang tidak seditkit demi kelancarannya, perusahaan perlu menganalisa mengenai kebutuhan dan pengeluaran yang akan digunakan jika menginginkan biaya distribusi yang minimal. Agar memperoleh biaya yang minimal, maka dalam penelitian ini menawarkan solusi menggunakan algoritma genetika dengan cara menentukan rute yang ditempuh oleh mobil pengantar dengan meminimalkan isi muatan truk bergantung pada jenis produk yang akan diantar. Penelitian ini menggunakan representasi kromosom dengan permutasi dengan satu segmen. Segmen ini adalah permutasi untuk agen. Panjang kromosom tergantung banyaknya jumlah agen dalam sebuah perusahaan. Dan mobil pengantar yang digunakan berjumlah dua buah. Untuk mengukur solusi tersebut maka diperlukan perhitungan fitness yang didapatkan dari total biaya dan sisa muatan mobil pengantar. Dari hasil uji coba, dengan ukuran populasi 80, banyaknya generasi 1500, nilai cr 0.2 dan nilai mr 0.8 maka akan didapatkan rata – rata fitness terbaik. Hasil akhir merupakan kromosom terbaik yang menjadi keberhasilan distribusi barang dengan total biaya terkecil dan dengan sisa muatan terkecil.

Kata Kunci: Algoritma genetika, distribusi multi-product

Abstract

Sumayyah Ula Syahidah. 2015 : ***Optimization Distribution multy- product with Genetic Algorithms*. (C*ase Study: Frozen Food delivery).Information Technology and Computer Science Program, Brawijaya University, Malang.***
***Advisor* : Wayan Firdaus Mahmudy, S.Si., M.T., Ph.D**

A company that involve in the industry field, can not be separated from distribution problems. Distribution is a milestone in the importance of an industrial company for the smooth operation. Distribution process is one of the most important processes in a company that did not spend a little cost for this process. A company needs to analyse the distribution process such as needs and expenses that need to be spend for the smoothness of distribution process. In order to obtain a minimal cost, this research offers a computing solution using genetics Algorithm by determining which route should be taken by the carrier car. This research uses permutation representation of chromosome 1 segment. The segment is a permutation of the agent. This chromosome lenght depends on the number of agents in a company. For the carrier car number, is from the company too. To measure these solution, the calculation of fitness which obtained from the total cost and the rest of cargo truck. From the test, we got the best average fitness that contains population size 80, number of generations 1500, and the value of cr 0.2 and 0.8 for mutation. The best result is the best chromosome for distribution multy-product with the smallest total cost and the smalles load.

Keywords: Genetics Algorithm, optimization of distribution multy-product

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

KATA PENGANTAR

Puji syukur penulis panjatkan kehadirat Allah SWT, yang atas rahmat-Nya maka penulis dapat menyelesaikan penyusunan proposal skripsi yang berjudul "Optimasi Distribusi Makanan Beku Menggunakan Algoritma Genetika". Proposal skripsi ini diajukan sebagai salah satu syarat untuk mendapatkan gelar sarjana S-1 di Program Teknologi Informasi dan Ilmu Komputer Universitas Brawijaya.

Keberadaan skripsi ini tidak terlepas dari bimbingan dan bantuan dari berbagai pihak, untuk itu pada kesempatan ini penulis menyampaikan rasa terimakasih dan perhargaan sebesar- besarnya kepada:

1. Bapak Wayan Firdaus Mahmudy, S.Si., M.T.,Ph.D.sebagai pembimbing I yang dengan sabar memberikan saran dan masukan perbaikan skripsi ini;
2. Bapak Ir. Sutrisno, M.T, Bapak Ir. Heru Nurwasito, M.Kom, Bapak Himawat Aryadita, S.T, M.Sc, dan Bapak Edy Santoso, S.Si, M.Kom selaku Ketua, Wakil Ketua 1, Wakil Ketua 2, dan Wakil Ketua 3 Fakultas Ilmu Komputer Universitas Brawijaya;
3. Bapak Drs. Mardji, MT dan Bapak Issa Arwani, S.Kom., M.Sc selaku Ketua dan Sekretaris Program Studi Informatika/Ilmu Komputer serta segenap Bapak / Ibu Dosen, Staff Administrasi dan Perpustakaan Fakultas Ilmu Komputer Universitas Brawijaya;
4. Kedua orang tua penulis, Ibu Andi dan Bapak Taufik, dan keluarga yang tidak pernah bosan dan lupa untuk memberikan doa dan dorongan semangat hingga terselesainya proposal skripsi ini;
5. Ibu Fatimah, selaku pemilik "*frozen food home industry"* yang bersedia menjadi narasumber dalam penelitian ini;
6. Seluruh mahasiswa fakultas ilmu komputer universitas brawijaya khususnya angkatan 2011
7. Seluruh dosen Program Studi Informatika Universitas Brawijaya, atas dukungan dan kerjasamanya;
8. Semua pihak yang tidak dapat disebutkan satu – persatu.

Semoga materi ini dapat bermanfaat dan menjadi sumbangan pemikiran bagi pihak yang membutuhkan, khususnya bagi penulis sehingga tujuan yang diharapkan dapat tercapai.

Malang, 25 Juli 2015

Penulis

BAB I
PENDAHULUAN

1.1 Latar Belakang

Perusahaan yang bergerak di bidang industri tidak lepas dari masalah pendistribusian barang. Distribusi barang menjadi tonggak dalam pentingnya suatu perusahaan industri dalam kelancaran operasionalnya. Dengan distribusi barang yang tepat, dapat mempermudah penyampaian barang dan jasa dari produsen kepada konsumen, sehingga penggunaannya sesuai dengan yang diperlukan.Dalam salah satu penelitian, disebutkan bahwa biaya dan proses distribusi berbanding lurus dengan penjualannya. Dalam proses penjualan, biaya distribusi berperan penting, 96% volume penjualan disebabkan oleh biaya distribusi sedangkan 4% disebabkan oleh merupakan pengaruh variabel lain dan faktor lainnya (Munawar & Marpaung, 2008).

Menurut Angipora (2002) (dalam Fadillah, 2013), jalur distribusi adalah suatu jalur yang dilalui oleh arus barang-barang dari produsen ke perantara dan akhirnya sampai pada pemakai. distribusi adalah kegiatan-kegiatan pengiriman barang yang dilakukan oleh suatu organisasi dengan tujuan agar produk yang dihasilkannya dapat diterima oleh konsumen baik melalui perantara ataupun tidak.

Frozen Food merupakan suatu makanan yang sangat diminati pada sekarang ini. Untuk itu *Frozen-snack* salah satu *home industry* penghasil makanan bekuyang banyak diminati. Dalam *Frozen-Snack* beragam makanan beku di produksi. Dan beragam makanan beku itu terdiri dari dua produk yang berbeda yaitu produk 1 untuk makanan beku berupa roti dan berbagai macam donat dan produk 2 untuk makanan seperti kebab, samosa dan sejenisnya. *Frozen-snack* ini membagi makanan bekunya dalam dua kategori agar agen mudah dalam melakukan pemesanan.

Untuk itu dalam penjualannya *home industry* ini mendistribusikan barangnya melalui beberapa agen yang telah tersebar di berbagai daerah di jawa timur ini seperti surabaya, sidoarjo, gresik. Untuk itu dibutuhkan sistem pendistribusian yang baik untuk mengirim masing-masing pesanan agen yang

akan diantarkan untuk pelanggan. Jika pendistribusian terhadap agen perusahaan baik dan menghemat biaya maka biaya yang dihemat dalam pendistribusian bisa digunakan untuk produksi dan lainnya.

Banyak sekali yang dapat metode yang dapat digunakan dalam optimasi. Contohnya dalam salah satu penelitian adalah Optimasi Penjadwalan Pengiriman Produk Jadi Menggunakan Pendekatan Binary Integer Programming (Studi Kasus Di PT Tiga Pilar Sejahtera Surakarta)(Susanti,E & Sriyanto, 2006).Dalam penelitian ini didapati kesimpulan bahwa dengan dilakukannya penelitian tersebut didapatkan manfaat diantaranya proses penjadwalan untuk perusahaan terkait lebih terstruktur dan lebih mudah dipahami dan dapat diperoleh solusi optimal untuk penjadwalannya. Untuk total biaya yang didapatkan dari hasil penelitian ini lebih menghemat biaya dari yang dibuat perusahaan sebelumnya (Susanti,E & Sriyanto, 2006).

Berdasarkan permasalahan diatas, solusi yang dapat ditawarkan adalah dengan mengoptimasi penjadwalan distribusi barang untuk menghemat waktu dan biaya dalam melakukan pendistribusian barang jadi. Dalam objek ini yang diteliti adalah mengingat frozen food merupakan hasil industri yang memiliki waktu *expire* atau memiliki jangka waktu tertentu sehingga sangat di perlukan efisiensi waktu yang tepat dalam pendistribusiannya.

Salah satu metode yang dapat digunakan dalam optimasi distribusi ini adalah dengan menggunakan algoritma genetika. Dengan menggunakan metode algoritma genetika yang tepat kita akan mendapatkan representasi kromosom yang baik, parameter algoritma yang tepat dan kualitas solusi yang maksimal dalam pendistribusian ini.

Untuk contoh yang menggunakan algoritma genetika adalah Penerapan Algoritma Genetika untuk Optimasi *Vehicle Routing Problem with Time Window* (VRPTW) Studi Kasus Air Minum Kemasan (Saputri, MW, Mahmudy, WF & Ratnawati, 2015). Dalam penelitian ini dapat disimpulkan bahwa algoritma genetika dapat diterapkan dalam pencarian rute optimal air minum dengan

kendala *time window* dengan dilakukannya pengujian terhadap data yang telah didapat (Putri, 2015).

Dan dalam penelitian lainnya mengenai algoritma genetika (Putri, 2015) yang berjudul Penerapan Algoritma Genetik *Untuk Vehicle Routing Problem with Time Window* (VRPTW) Pada Kasus Optimasi Distribusi Beras Bersubsidi. Dalam penelitian ini algoritma genetika dapat diterapkan dalam pencarian rute optimal distribusi beras bersubsidi menggunakan *time window* dengan melakukan pengujian data yang diuji juga, dengan melakukan perubahan parameter *crossover*, *mutation* dan jumlah populasi (Putri,2015).

Dengan beberapa contoh penelitian algoritma diatas lainnya telah dibuktikan bahwa, algoritma genetika memiliki kemampuan dalam menyelesaikan berbagai masalah kompleks dalam menghadapi masalah optimasi (Mahmudy, 2014). Diharapkan algoritma genetika ini mampu menyelesaikan masalah dalam optimasi pendistribusian barang dalam produksi *frozen food* ini.

1.2 Rumusan Masalah

Berdasarkan paparan latar belakang tersebut, maka rumusan masalah adalah sebagai berikut:

1. Bagaimana cara mengoptimalkan pendistribusian barang dalam produksi *frozen food* dengan algoritma genetika?
2. Bagaimana menentukan representasi kromosom yang baik dalam distribusi barang?
3. Bagaimana menentukan parameter algoritma genetika yang tepat?
4. Bagaimana mengukur kualitas solusi yang dihasilkan oleh algoritma genetika?

1.3 Batasan Masalah

Batasan masalah dalam penelitian ini adalah:

1. Data yang digunakan pada penelitian ini adalah data yang diambil dari salah satu home industri penghasil makanan beku di sidoarjo.
2. Bobot makanan beku tidak menjadi pertimbangan dalam pendistribusian

3. Volume kendaraan pengantar yang digunakan untuk parameter pendistribusian barang.
4. Biaya yang dikeluarkan mobil satu dan mobil dua untuk pengangkutan barang berbeda.
5. Biaya transportasi dihitung dari jarak tiap pengangkutan mobil dikalikan dengan biaya angkut mobil sesuai muatan mobil.
6. Bahasa pemrograman yang digunakan adalah bahasa Java.
7. Sistem ini akan dirancang menggunakan metode penelitian algoritma genetika.

1.4 Tujuan Penelitian

Tujuan dari penelitian ini adalah sebagai berikut :

1. Mengoptimalkan pendistribusian *frozen food* sesuai kebutuhan, dan menghemat biaya yang ditentukan
2. Untuk mengetahui bagaimana cara menentukan representasi kromosom yang efisien untuk mengoptimalkan distribusi barang
3. Untuk mengetahui bagaimana menentukan parameter algoritma genetika yang tepat.

1.5 Manfaat Penelitian

Manfaat dari penelitian ini adalah :

1. Memberikan informasi bagi perusahaan sebagai bahan pertimbangan dalam proses distribusi barang untuk peningkatan produktivitas perusahaan
2. Meminimumkan total biaya dalam pendistribusian barang serta memaksimalkan produksi

1.6 Sistematika Penulisan

Untuk mencapai tujuan yang diharapkan maka sistematika penulisan yang disusun dalam tugas akhir ini adalah sebagai berikut :

BAB I PENDAHULUAN

Bab ini berisi tentang latar belakang, rumusan masalah, batasan masalah, tujuan penelitian, manfaat penelitian, sistematika penulisan dan waktu pengerjaan.

BAB II TINJAUAN PUSTAKA

Bab ini terbagi menjadi dua sub bab. Yang pertama kajian pustaka yang membahas tentang penelitian yang sebelumnya pernah dilakukan. Yang kedua adalah dasar teori yang membahas teori-teori pendukung yang digunakan untuk perancangan sistem optimasi distribusi makanan beku menggunakan algoritma genetika.

BAB III METODELOGI

Membahas tentang metode penulisan yang digunakan seperti studi literatur, perancangan perangkat lunak, implementasi, pengujian, dan analisis serta perhitungan manual dan representasi kromosom yang akan digunakan.

BAB IV PERANCANGAN

Membahas tentang perancangan interface beserta pengujian program.

BAB V IMPLEMENTASI DAN PEMBAHASAN

Membahas tentang hasil perancangan dari analisis kebutuhan dan implementasi sistem optimasi Optimusi distribusi makanan beku menggunakan algoritma genetika.

BAB VI PENGUJIAN

Memuat tentang hasil pengujian dan analisis terhadap sistem yang telah direalisasikan.

BAB VII KESIMPULAN

Memuat kesimpulan yang diperoleh dari pembuatan dan pengujian perangkatlunak yang dikembangkan dalam skripsi ini serta saran – saran untukpengembangan lebih lanjut.

BAB II

TINJAUAN PUSTAKA

Pada bab dua ini terdiri dari tinjauan pustaka dan dasar teori. Kajian pustaka membahas penelitian yang telah ada dan diusulkan. Dasar teori membahas teori yang diperlukan untuk menyusun penelitian yang diusulkan. Kajian pustaka pada penelitian ini adalah membandingkan penelitian sebelumnya yang berjudul 'Optimasi Penjadwalan Pengiriman Produk Jadi Menggunakan Pendekatan Binary Integer Programming', 'Penerapan Algoritma Genetik untuk *Vehicle Routing Problem with Time Windows* (VRPTW) pada Kasus Optimasi Distribusi Beras Bersubsidi' dan 'Penerapan Algoritma Genetika untuk *Optimasi Vehicle Routing Problem with Time Window* (VRPTW) Studi Kasus Air Minum Kemasan'. Pada penelitian ini konsep teori yang diperlukan berdasarkan latar belakang dan rumusan masalah adalah: Pengertian dasar Algoritma Genetika, sedikit penjelasan tentang garis besar distribusi barang, dan lainnya.

2.1 Kajian Pustaka

Kajian pustaka berisi penelitian-penelitian sebelumnya yang terkait atau memiliki kesesuaian dengan penelitian penulis. Kajian pustaka bermanfaat sebagai pembanding penulis dalam melakukan penelitian ini. Perbandingan tersebut ditampilkan pada Tabel 2.1.

Tabel 2.1 Tabel Kajian Pustaka

No.	Judul	Penulis	Perbandingan	
			Kajian pustaka	Skripi penulis
1.	Optimasi Penjadwalan Pengiriman Produk Jadi Menggunakan	Susanti,E dan Sriyanto	Penelitian ini menggunakan metode *linear programming* Penelitian ini bertujuan untuk meminimalkan biaya transportasi dengan	Pada skripsi penulis menggunakan algoritma genetika. Penelitian ini bertujuan untuk meminimalkan biaya distribusi dengan meminimalkan sisa

	Pendekatan Binary Integer Programming (Studi Kasus Di PT Tiga Pilar Sejahtera Surakarta)		memaksimalkan pengalokasikan barang yang ada pada sumber agar kebutuhan beberapa distributor terpenuhi.	muatan.
2.	Penerapan Algoritma Genetik untuk Vehicle Routing Problem with Time Windows (VRPTW) pada Kasus Optimasi Distribusi Beras Bersubsidi	Farah Bahtera Putri	Pada penelitian ini menggunakan *Vehicle Routing Problem with Time Windows (VRPTW)* yaitu distributor hanya dapat dilayani selama selang waktu yang telah ditentukan yaitu jam buka dan jam tutupnya pelayanan. Pada penelitian ini bertujuan agar pengirim dapat tiba sesuai dengan jam pelayanan distributor tersebut sehingga meminimalkan distributor yang tidak dilayani. Agar pengirim tiba pada waktunya, maka pengirim harus mempertimbangkan jarak tempuhnya. Penelitian ini menggunakan pengujian seleksi antara elitism selection dengan binary tournament selection. Berdasarkan hasil pengujian seleksi dalam penelitian ini, elitism selection mempunyai hasil yang lebih baik dari pada	Pada skripsi penulis bertujuan untuk meminimalkan biaya distribusi dengan meminimalkan sisa muatan . Penelitian ini menggunakan elitism selection berdasarkan hasil pengujian penelitian sebelumnya yang membandingkan performa elitism selection dengan binary tournament selection.

			binary tournament selection.	
3.	Penerapan Algoritma Genetika untuk Optimasi Vehicle Routing Problem with Time Window (VRPTW) Studi Kasus Air Minum Kemasan	Dita Sundarnigsih1, Wayan Firdaus Mahmudy, Sutrisno	Pada penelitian ini menggunakan *Vehicle Routing Problem with Time Windows (VRPTW)* yaitu distributor hanya dapat dilayani selama selang waktu yang telah ditentukan yaitu jam buka dan jam tutupnya pelayanan. Pada penelitian ini bertujuan agar pengirim dapat tiba sesuai dengan jam pelayanan distributor tersebut sehingga meminimalkan distributor yang tidak dilayani.	Pada skripsi penulis, untuk meminimalkan biaya distribusi dengan mencari rute terpendek dan yang memiliki sisa muatan yang paling sedikit. Dalam skripsi penulis ini tidak tergantung oleh waktu buka dan tutup pelayanan.

2.2 Distribusi Produk

Menurut swastha (2000) (dalam Firdaus, 2012)distribusi merupakan saluran yang digunakan oleh produsen untuk menyalurkan barang dari produsen ke konsumen ataupun pemakai industri. Distribusi atau penyaluran barang merupakan kegiatan yang penting dalam suatu kegiatan pemasaran, karena apabila produk yang dihasilkan memiliki kualitas yang baik namun kegiatan pendistribusiannya tidak berjalan dengan lancar atau macet, maka perusahaan penyedia barang akan mendapatkan kerugian baik materi maupun citra yang tidak baik di mata para konsumen. Kegiatan distribusi yang gagal biasanya dikarenakan kegiatan distribusi yang kurang tepat sehingga barang di pasaran kurang terjual dan menjadikan konsumen tidak puas. Dengan demikian distribusi memiliki peranan penting untuk pertumbuhan perusahaan (Mukodim, 2007).

2.3 Algoritma Genetika

Algoritma Genetika tipe GA yang paling sering dipergunakan karena kemampuannya untuk menyelesaikan berbagai masalah kompleks yang berhadapan dengan masalah optimasi model matematikanya bahkan yang sulit untuk dibangun modelnya (Mahmudy, 2013). Algoritma genetika sendiri sudah sangat popular digunakan untuk menyelesaikan masalah-masalah optimasi yang bersifat kompleks di bidang fisika, biologi, ekonomi, sosiologi dan lain lain (Mahmudy, 2014). Salah satu penerapannya yaitu optimasi penjadwalan produksi dalam bidang industri manufaktur menggunakan algoritma genetika (Mahmudy, Marian & Luong dalam Mahmudy, 2014). Algoritma genetika diilhami oleh ilmu genetika, karena itu istilah yang digunakan dalam algoritma genetika banyak diadopsi dari ilmu tersebut. Apabila dibandingkan dengan prosedur pencarian dan optimasi biasa, algoritma genetika berbeda dalam beberapa hal sebagai berikut (Michalewicz 1996, dalam Mahmudy, 2014):

1. Manipulasi data dilakukan melalui suatu parameter yang disebut dengan chromosome.
2. Proses pencarian dilakukan dari beberapa titik dalam satu populasi, tidak dari satu titik saja.
3. Proses pencarian menggunakan informasi dari fungsi tujuan.
4. Pencariannya menggunakan stochastic operators yang bersifat probabilistik, tidak menggunakan aturan deterministik.

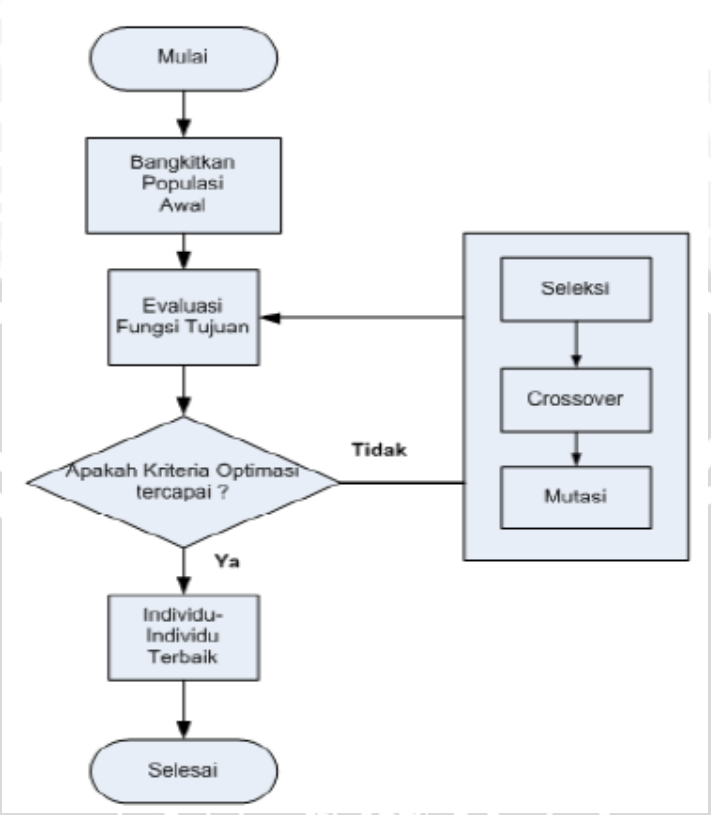
Sedangkan kelebihan AG sebagai metode adalah sebagai berikut (Mahmudy, 2014):

1. GA adalah algoritma yang berbasis populasi yang biasanya digunakan pada optimasi maslaah dengan ruang perncarian yang luas dan kompleks. Karena GA mencakup populasi yang luas maka memunkinkan untuk GA melompat keluar dari daerah optimumnya sendiri (Gen & Cheng 1997).

2. AG menghasilkan himpunan solusi yang optimal yang membantu untuk menyelesaikan masalah dalam banyak objek sekalipun (Mahmudy & Rahman 2011).
3. Individu yang ada pada populasi bisa diletakkan pada beberapa sub-populasi yang diproses pada sejumlah komputer secara paralel. Hal ini bisa mengurangi waktu komputasi pada masalah yang sangat kompleks (Defersha & Chen 2010; Qi, Burns & Harrison 2000). Penggunaan sub-populasi juga bisa dilakukan pada hanya satu komputer untuk menjaga keragaman populasi dan meningkatkan kualitas hasil pencarian (Mahmudy 2009).
4. GA dapat digunakan untuk mencari solusi dengan banyak variabel. Variabel dalam GA tersebut bisa berupa diskrit ataupun kontinyu (Haupt & Haupt 2004).
5. untuk pengkodean solusi, GA menggunakan *chromosome* sehingga dapat melakukan pencarian tanpa memperhatikan informasi spesifik dari masalah yang akan diselesaikan.
6. Beberapa penelitian menyebutkan bahwa hasil hybrid GA atay *Hybrid Gas(HGA)* cukup fleksibel untuk di hybridisasikan dengan algoritma lainnya.
7. Ga bersifat *erodic* yaitu setiap solusi dapat diperoleh dari solusi yang lain hanya dengan beberapa langkah saja.

2.4 Struktur Algoritma Genetika

Gambar 2.1 merupakan *flowchart* atau alur algoritma genetika



Gambar 2.1 Alur Algoritma Genetika

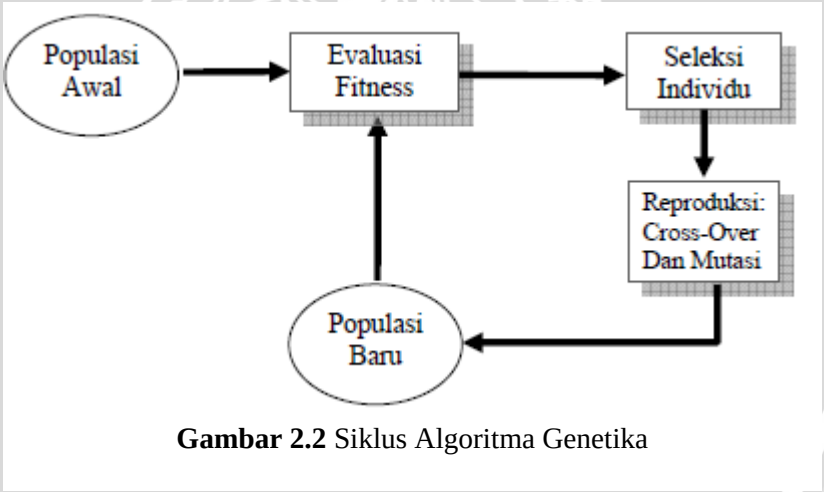
Solusi dari sebuah masalah harus dipetakan terlebih dahulu menjadi string chromosome. String chromosome tersusun dari sejumlah gen yang merupakan variabel keputusan yang akan digunakan pada solusi. Yang nantinya representasi dari string chromosome tersebut beserta *fitness* yang akan menilai seberapa layak sebuah chromosome tersebut untuk dimasukkan kedalam algoritma genetika. Dalam banyak kasus, bagaimana merepresentasikan sebuah solusi menjadi chromosome sangat menentukan kualitas dari solusi yang dihasilkan (Mahmudy, Marian & Luong 2012b, dalam Mahmudy, 2013).

Proses dalam algoritma genetika diawali dengan inisialisasi, yaitu menciptakan individu-individu secara acak yang memiliki susunan gen (*chromosome*) tertentu. *Chromosome* ini mewakili solusi dari permasalahan yang

akan dipecahkan. Reproduksi dilakukan untuk menghasilkan keturunan (*offspring*) dari individu-individu yang ada di populasi. Evaluasi digunakan untuk menghitung kebugaran (*fitness*) setiap *chromosome*. Semakin besar *fitness* maka semakin baik *chromosome* tersebut untuk dijadikan calon solusi. Seleksi dilakukan untuk memilih individu dari himpunan populasi dan *offspring* yang dipertahankan hidup pada generasi berikutnya. Fungsi probabilistis digunakan untuk memilih individu yang dipertahankan hidup. Individu yang lebih baik (mempunyai nilai kebugaran/*fitness* lebih besar) mempunyai peluang lebih besar untuk terpilih (Gen & Cheng 1997, dalam Mahmudy, 2013).

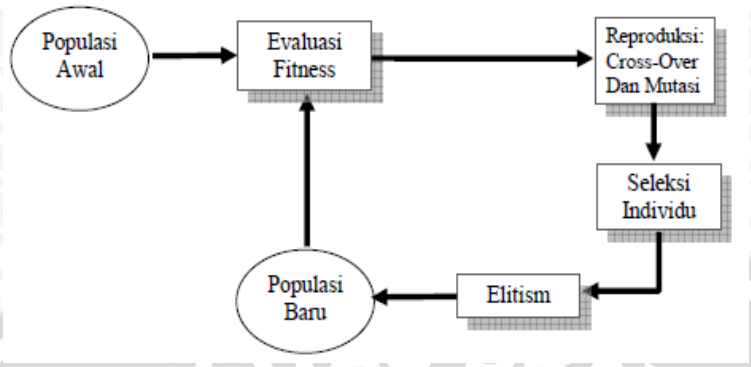
2.5 Siklus Algoritma Genetika

Siklus algoritma genetika pertama kali dikenalkan oleh David Goldberg. Dimana gambar dari siklus tersebut terdapat pada gambar 2.2



Gambar 2.2 Siklus Algoritma Genetika

Siklus ini kemudian dikembangkan oleh *zbigniw michalewiz* dengan menambahkan operator elitism dan membalik proses seleksi setelah proses reproduksi.



Gambar 2.3 Siklus Algoritma Genetika 2

2.6 Representasi Kromosom

Representasi kromosom merupakan proses awal sebelum masuk proses regenerasi. Cara merepresentasikan permasalahan dalam kromosom merupakan suatu hal yang penting dalam algoritma genetika. Model representasi kromosom yang dapat dipergunakan untuk menyelesaikan suatu masalah, antara lain adalah (Suwirmayanti, 2014):

1. Kromosom biner

Kromosom biner adalah kromosom yang disusun dari gen-gen yang bernilai 0 dan 1. Kromosom ini merupakan model standar dalam algoritma genetika. Kromosom biner merupakan model yang sederhana dengan tingkat keberhasilan yang tinggi. Gambar 2.4 memperlihatkan contoh kromosom biner

Kromosom A	1	0	1	1	0
Kromosom B	1	1	0	0	0

Gambar 2.4 Contoh Kromosom Biner

2. Kromosom float

Kromosom float merupakan model yang memiliki jumlah paramete yang banyak.Tingkat keberhasilan dari bentuk kromosom ini rendah dalam kecepatan

(jumlah generasi). Model crossover dan mutasi pada kromosom float ini sangat berbeda dengan model crossover dan mutasi pada kromosom biner. Sehingga perlu strategi khusus didalam menentukan model crossover dan mutasi. Gambar2.5 menunjukkan contoh kromosom *float.*

Kromosom A	1.2	3.4	1.6	2.2	4.1
Kromosom B	2.2	3.1	1.2	4.2	1.6

Gambar 2.6.5 Contoh Kromosom Float

3. Kromosom permutasi

Kromosom kombinatorial atau biasa disebut dengan kromosom permutasi adalah kromosom yang disusun dari gen–gen yang dinilai berdasarkan urututannya. Kromosom ini biasanya digunakan untuk menyelesaikan permasalahan pengurutan seperti Ordering task dan TSP. Gambar 2.6 dibawah ini merupakan contoh gambar kromosom permutasi.

Kromosom A	8 5 4 9 1 2 3 6 7
Kromosom B	9 1 2 4 3 8 5 7 6

Gambar 2.6 Contoh Kromosom Permutasi

4. Kromosom Nilai

Masalah yang akan dipecahkan kromosom nilai biasanya sulit dipecahkan jika menggunakan kromosom biner. Kromosom niali terdiri dari gen-gen bernilai string dari suatu nilai atau simbol. Contoh representasi kromosom pada Gambar 2.7

Kromosom A	1,2 3,3 2,5 1,6 4,4 35
Kromosom B	AHDKHSTFNSFKSL
Kromosom C	(left) (right) (back)

Gambar 2.7 Contoh Kromosom Nilai

2.7 Crossover

Proses rekombinasi pindah silang atau crossover adalah menyilangkan dua kromosom sehingga membentuk kromosom baru yang harapannya lebih baik daripada kromosom sebelumnya. Tidak semua kromosom pada suatu himpunan akan mengalami proses rekombinasi. Kemungkinan suatu kromosom akan mengalami proses pindah silang didasarkan pada probabilitas pindah silang yang telah ditemukan terlebih dahulu. Probablitas crossover dapat menyatakan banyaknya kromosom yang akan dilakukan crossover(Hardianti, 2013).

Macam-macam pindah silang (Crossover)antara lain (Suwirmayanti, 2014) :

1. Pindah silang satu titik*(One-Cut-Point)*

   Pada pindah silang satu titik (*one-point crossover*), satu titik sepanjang kromosom dipilih secara *random*. Segmen dari induk dari titik point ke kiri atau kekanan ditukar untuk menghasilkan individu baru
2. Pindah silang dua titik (*Two-cut-point*)

   Pada pindah silang dua titik, dua titik sepanjang kromosom dipilih secara *random*. Segmen induk diantara kedua titik potong dipertukarkan untuk menghasilkan individu baru.
3. Pindah silang banyak titik (*N-cut-point*)

Pada pindah silang banyak titik, sejumlah N titik sepanjang kromosom dipilih secara *random*, dimana titik potong tidak boleh sama.

4. Pindah Silang seragam

   Pada uniform crossover, sejumlah N titik ditentukan secara random baik itu jumlah maupun letaknya. Segmen yang dipertukarkan sesuai dengan jumlah titik potong yang didapatkan secara random tersebut.

2.8 Mutasi

Mutasi adalah penggantian satu atau lebih gen dalam suatu kromosom yang terjadi tanpa melibatkan kromosom yang lain. Pada kromosom biner, mutasi dilakukan dengan mengubah gen biner 0 menjadi 1 dan 1 menjadi 0. Pada kromosom float ada dua macam mutasi yang banyak dilakukan yaitu random mutation dan shift mutation.

1. Random mutation adalah mengganti gen yang telah dimutasi secara acak.
2. Shift mutation adalah menggeser nilai gen termutasi sebesar ε, di mana ε adalah bilangan kecil yang ditentukan.

Probabilitas mutasi yang baik berada pada 0 sampai dengan 0.3. Probabilitas yang terlalu besar menyebabkan konvergensi yang sulit didapatkan dan probabilitas yang terlalu kecil menyebabkan terjadinya penjebakan dalam optimum lokal (Adhy, 2010).

2.9 Pemrograman Java

Java menurut definisi adalah nama untuk sekumpulan teknologi untuk membuat dan menjalankan perangkat lunak pada komputer standalone ataupun pada lingkungan jaringan. *Java* dirancang sebagai bahasa pemrograman yang handal dan aman. Aplikasi-aplikasi yang dibangun dengan bahasa *Java* sangat handal dengan manajemen memori yang bagus.

Java merupakan bahasa pemrograman berorientasi objek. Java memiliki keseimbangan, menyediakan mekanisme peng-class-an sederhana, dengan model antar muka dinamik yang intuitif hanya jika diperlukan. Java memiliki beberapa kemampuan yang memungkinkan program melakukan beberapa hal pada saat

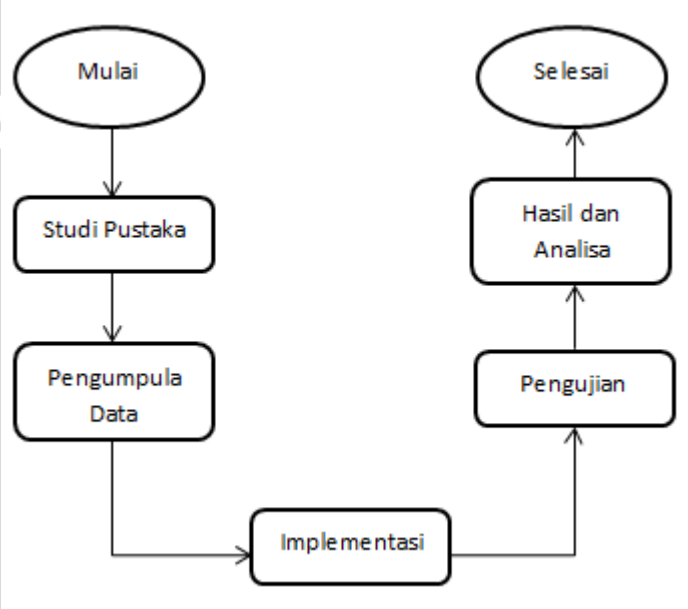
bersamaan, tanpa harus kesulitan menangani proses yang akan terjadi selanjutnya (Tasmawati, 2008).

BAB III

METODELOGI PENELITIAN

3.1 Tahapan Penelitian

Tahapan penelitian atau yang disebut dengan metodologi penelitian akan menjelaskan mengenai langkah – langkah yang akan digunakan dalam pembuatan aplikasi optimasi distribusi barang dengan algoritma genetika. Contoh alur tahapan penelitian dalam Gambar 3.1



Gambar 3.1 Alur Tahapan Penelitian

Beberapa tahapan penelitian yang digunakan dalam pembuatan skripsi ini adalah :

1. Studi Pustaka

   Studi pustaka merupakan pengumpulan referensi yang nantinya akan dijadikan pedoman sesuai dengan penelitian yang diangkat. Manfaat dari studi pustaka atau studi litelatur ini adalah dapat memberikan gambaran menyeluruh mengenai sejauh mana penelitian-penelitian yang terkait dengan penelitian yang diambil.dalam hal ini referensi diambil dari jurnal, laporan, maupun buku yang terkait dengan judul penelitian.

2. Pengumpulan Data

Pengumpulan data yang diambil untuk penelitian ini adalah dengan menggunakan data primer. Data primer adalah data yang diperoleh oleh peneliti secara langsung ke sumber datanya. Data primer ini biasa juga disebut data asli atau data baru yang *up to date*. Untuk mendapatkan data primer ini, peneliti harus mengumpulkannya secara langung ke sumbernya. Teknik pengumpulan data primer ini bisa dilakukan dengan cara wawancara, observasi, dan juga melalui penyebaran kuisioner. Karena dalam hal ini membutuhkan jarak, maka mencari jarak antar agen dan perusahaan Makanan beku serta jarak antar agen menggunakan *googlemaps*. *Googlemaps* adalah sebuah peta untuk memperkirakan jarak lokasi dalam skala tertentu.

3. Implementasi

melakukan implementasi sistem berdasarkan rancangan dan analisa data yang telah dibuat sebelumnya dalam bahasa pemrograman dan hasil akhir berupa sebuah aplikasi.

4. Pengujian

Melakukan uji coba terhadap sistem yang telah dibuat.

5. Hasil dan Analisa

Pada tahap ini dilakukan analisa dari uji coba, serta menguji apakah metode yang digunakan telah memberikan hasil optimal. Hasil *fitness* dari uji coba metode algoritma genetika akan dilakukan evaluasi, untuk mendapatkan hasil yang mendekati optimal.

3.2 Kebutuhan Sistem

Kebutuhan sistem yang dibahas meliputi kebutuhan perangkat keras dan perangkat lunak yang digunakan untuk implementasi. Tujuan dari penetapan spesifikasi kebutuhan adalah untuk mengurangi kesalahan saat implementasi perangkat lunak sehingga meminimalisir kesalahan dalam membangun perangkat lunak.

3.2.1 Spesifikasi Perangkat Keras

Spesifikasi perangkat keras pada penelitian ini dijelaskan pada Tabel 3.1

Table 3.1 Spesifikasi Perangkat Keras

Komponen Komputer	**Spesifikasi**
Processor	Intel core i5
RAM	2GB
Memory	Free 20GB

3.2.2 Spesifikasi Perangkat Lunak

Spesifikasi perangkat lunak untuk pembuatan optimasi distribusi makanan beku menggunakan algoritma genetika dijelaskan pada tabel 3.2

Table 3.2 Spesifikasi Perangkat Keras

Tools	**Nama Software**
Program	Java
Dokumentasi	Microsoft Word 2010
Sistem Operasi	Windows 8 64 bit

3.3 Formulasi Masalah

Data yang di dapat dari optimasi distribusi makanan beku ini adalah hasil wawancara langsung pada pemilik home industri makanan beku di surabaya. Data yang di dapat berupa :

1. Agen di home industri makanan beku
2. Banyaknya pesanan setiap agen di home industri makanan beku.
3. Alamat setiap agen yang memesan di home industri makanan beku.
4. Banyak kendaraan yang digunakan untuk mengirim makanan beku pada setiap agen. Banyaknya kendaraan untuk mengirim ada 2 kendaraan berupa mobil.
5. Jarak antar alamat antar agen
6. Biaya distribusi yang dikeluarkan tiap mobil dalam proses pengiriman barang ke agen.

Biaya distribusi yang akan dioptimasi dalam penilitian ini adalah biaya yang digunakan oleh distributor saat proses distribusi barang kepada pelanggan. Biaya

distribusi dihitung mulai dari pemasok ke pelanggan (biaya pergi) sekaligus saat distributor kembali ke pemasok (biaya pulang). Biaya yang dikeluarkan pada proses distribusi ini berdasarkan beberapa aspek, diantaranya jarak tempuh (*j*) jarak tempuh yang digunakan adalah km, biaya angkut saat kendaraan kosong (b_0), biaya angkut saat kendaraan terisi (b_1), isi yang diangkut oleh kendaraan (w) dan isi maksimal kendaraan (w_{max}). Biaya pergi dapat dihitung dengan menggunakan rumus berikut.

$$biaya\ pergi = j * (b_0 + (b_1 - b_0) * \frac{m}{m_{\max}}) \quad (3\text{-}1)$$

Sedangkan untuk biaya pulang, yaitu biaya kembali ke pemasok dapat dihitung dengan menggunakan rumus berikut

$$biaya\ kembali = j * b_0 \quad (3\text{-}2)$$

Dari data diatas, algoritma genetika untuk menentukan biaya yang paling minimum. Dalam kasus ini dicontohkan permintaan setiap agen dalam seminggu beserta alamat tinggal agen. Contoh nama produk pada Tabel 3.3 dan tabel pesanan agen pada Tabel 3.4

Tabel 3.3 Tabel Produk

No	Produk	Nama Produk
1	P1	Donat Keju
2	P2	Donat Kacang
3	P3	Donat Coklat
4	P4	Donat Bulat
5	P5	Risol Mayo
6	P6	Sosis Solo
7	P7	Tahu Bakso
8	P8	Samosa
9	P9	Martabak
10	P10	Pastel
11	P11	Kebab

Table 3.4 Tabel Permintaan Agen

Agen	Produk 1						Produk 2					ALAMAT
	P1	P2	P3	P4	P5	P6	P7	P8	P9	P10	P11	
1	10	10	30	50	30	20	-	-	20	20	50	SEPANJANG
2	10	10	50	50	20	40	-	-	40	20	-	SURABAYA MARGOREJO
3	10	5	20	50	20	20	-	20	30	30	-	SURABAYA MONOKROM
4	5	5	10	50	20	20	-	10	20	-	20	TENGGILIS SBY
5	10	10	50	50	20	40	-	-	40	-	-	SURABAYA TEGALSARI
6	20	20	20	50	20	5	-	-	-	20	40	SIDOARJO TROPODO
7	-	-	-	50	-	-	-	-	-	-	10	SIDOARJO TAMAN
8	20	-	20	50	20	10	-	-	-	-	-	GRESIK
9	-	-	-	50	-	-	-	-	-	-	-	SURABAYA SAWAHAN
10	20	20	30	40	40	-	-	-	-	-	-	SURABAYA MULYOREJO
11	5	-	20	70	10	10	-	-	-	-	-	SURABAYA SUKOLILO
12	-	-	-	10	10	-	-	-	-	-	50	SIDOARJO SUKOREJO
13	5	5	5	50	20	20	-	5	10	-	40	CANDI SIDOARJO
14	10	-	30	80	30	-	20	-	-	-	-	GRESIK
15	10	10	10	10			10		10			WARU SIDOARJO

Dan di pada tabel 3.5 merupakan tabel jarak antar pelanggan.

Tabel 3.5 Tabel Data Jarak

AGEN	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	0	6,6	6,1	9,8	12,7	10,9	5,6	28,6	13,6	15,9	13,6	13,4	25,3	28,6	10,4
2	6,6	0	2,1	4,7	6,7	9,9	12,9	25	6,8	10	12,5	15,7	23,8	25	9,2
3	6,1	2,1	0	7	6,7	10,7	12,3	26,5	7,1	10,3	13	15,1	23,2	26,5	11,3
4	9,8	4,7	7	0	11,8	8,5	13,1	35,8	11,9	8,6	9,9	15,9	24	35,8	5,7
5	12,7	6,7	6,7	11,8	0	14,6	16,2	20,3	2,5	7,5	11,6	19	27,1	20,3	15,2
6	10,9	9,9	10,7	8,5	14,6	0	13,1	35,6	15,2	12,4	14,2	13,6	21,8	35,6	2,1
7	5,6	12,9	12,3	13,1	16,2	13,1	0	34,1	17,7	20,2	22,2	11,6	20,9	34,1	12,5
8	28,6	25	26,5	35,8	20,3	35,6	34,1	0	19	25,7	30,9	30,9	48,6	0	35,4
9	13,6	6,8	7,1	11,9	2,5	15,2	17,7	19	0	9,9	14	20,5	35,2	19	22
10	15,9	10	10,3	8,6	7,5	12,4	20,2	25,7	9,9	0	6,1	22,7	30,9	25,7	10,6
11	13,6	12,5	13	9,9	11,6	14,2	22,2	30,9	14	6,1	0	25,3	33,4	30,9	12,1
12	13,4	15,7	15,1	15,9	19	13,6	11,6	30,9	20,5	22,7	25,3	0	10,3	30,9	14
13	25,3	23,8	23,2	24	27,1	21,8	20,9	48,6	35,2	30,9	33,4	10,3	0	48,6	21,7
14	28,6	25	26,5	35,8	20,3	35,6	34,1	0	19	25,7	30,9	30,9	48,6	0	35,4
15	10,4	9,2	11,3	5,7	15,2	2,1	12,5	35,4	22	10,6	12,1	14	21,7	35,4	0

Dari tabel diatas, diperlihatkan jarak antar agen dari satu tempat agen ke tempat agen yang lainnya. Contohnya jarak dari agen A ke agen B sebanyak 6.6 km. Pada tabel 3.6merupakan tabel data biaya kendaraan dari tempat awal.

Tabel 3.6 Tabel Data Biaya Kendaraan

Kendaraan	Jumlah/unit	Biaya muatan kosong	Biaya muatan penuh	Jumlah barang (pack)
M1	1	4000	3500	300 pack
M2	1	5000	4500	300 pack

Dari Tabel 3.6, diketahui terdapat kendaraan M1 dan M2 dengan jumlah masing-masing perunit 1 buah dengan biaya keberangkatan penuh kendaraan M1 adalah 3500 dan kendaraan M2 adalah 4500 dengan jumlah barang yang diangkut per keberangkatan adalah 300 pack.

3.4 Siklus Penyelesaian Masalah Menggunakan Algoritma Genetika

Dari data tersebut nantinya akan digunakan oleh algoritma genetika untuk menentukan kombinasi distribusi makanan beku yang paling optimum yang nantinya akan menghasilkan distribusi yang memiliki biaya yang paling minimum.

3.4.1 Representasi Kromosom

Pada proses perhitungan optimasi distribusi ini, kromosom akan dikodekan dalam angka permutasi, memiliki panjang string 15 sesuai dengan banyaknya agen. Contoh gambar representasi kromosom ada pada Gambar 3.2 dibawah ini :

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

Gambar 3.2 Representasi Kromosom

Berdasarkan representasi kromosom diatas, dijelaskan bahwa mobil 1 dan mobil 2 masing-masing akan mengangkut pesanan agen hingga pesanan tersebut diantarkan ke keseluruhan lima belas agen tersebut. Contohnya mobil 1 akan mengangkut pesanan agen 2 dengan jumlah produk 1 sekian dan jumlah produk 2 sekian, jika belum mencapai maksimal pengangkutan maka akan dilanjutkan dengan produk milik agen selanjutnya. Penjelasan dalam Tabel 3.7

Tabel 3.7 Tabel Penjelasan

Mobil	Agen	Produk	Sisa Muatan
M1	3	70 Produk 1 50 Produk 2	180
	4	40 Produk dan 40 Produk 2	100
	5	50 Produk 1 dan 0 produk 2	50
	11	0 Produk 1 dan 90 Produk 2	40
	12	40 Produk 1 dan 0 Produk 2	260
	7	25 Produk 1 dan 75 Produk 2	160
	6	40 Produk 1 dan 10 Produk 2	10
	8	100 Produk 1 dan 50 Produk 2	150
M2	9	40 Produk 1 dan 40 Produk 2	220
	2	70 Produk 1 dan 80 Produk 2	70
	1	50 Produk 1 dan 40 Produk 2	20
	13	100 Produk 1 dan 80 Produk 2	0
	15	30 Produk 1 dan 50 Produk 2	220
	14	0 Produk 1 dan 50 Produk 2	170
Total Sisa Muatan			170

K1 yang melakukan distribusi kepada pelanggan 1 (U1) dengan muatan maksimal sebesar 150 pcs. Tetapi K1 mendapatkan *penalty* karena muatan pada pengiriman tidak maksimal. Muatan pengiriman pada P1 hanya sebesar 80 pcs sehingga terdapat sisa 70 pcs pada K1. Hal inilah yang disebut sebagai *penalty* pada penelitian ini

Pada tabel 3.7 dijelaskan jika kromosom pada individu tertentu adalah [3 4 5 1112 7 6 8 9 2 1 13 15 14]. M1 melakukan distribusi pada agen 3 dengan

muatan maksimal sebanyak 300 pack akan tetapi M1 mendapatka *penalty* karena muatan pengiriman tidak maksimal. Muatan pengiriman pada 3 hanya sebesar 70 dari kategori produk 1 dan 50 dari kategori produk 2 sehingga terdapat sisa muatan sebanyak 180. Hal inilah yang disebut dengan *penalty* pada penelitian ini.

Perhitungan biaya dilakukan sesuai dengan biaya masing-masing kendaraan pada Tabel 3.6 dan total jarak yang ditempuh pada Tabel 3.5. Pada M1 dilakukan distribusi kepada 8 agen yaitu agen [3 4 5 11 12 7 6 8] diketahui jarak dari produsen ke Agen 3 adalah 10 km jarak dari agen 3 – 4 adalah 7 km dan jarak dari agen 4 – 5 adalah 11, 8 km begitu seterusnya hingga jarak paling terakhir dari agen 6 – 8 35,6 km. Sedangkan biaya muatan M1 saat penuh adalah 3.500/ Km dan saat kosong adalah 4000/Km. Total biaya distribusi yang dilakukan oleh M1 adalah pengiriman pada agen 3 ke agen 4 kemudian agen 5 dan seterusnya kemudian ditambah dengan biaya kembali agen 8 ke produsen. Biaya yang dikeluarkan M1 untuk distribusi adalah biaya yang dikeluarkan pada distribusi ke agen 3 ditambah ke agen 4 dan seterusnya hingga agen 8. Pada permasalahan ini biaya kembali diakumulasikan dari agen yang terakhir dikunjungi yaitu agen 8. Sehingga total biaya distribusi pada M1 yang dilakukan untuk individu 1 adalah total keseluruhan yang didapatkan dari biaya pergi dan biaya kembali dalam sekali M1 melakukan pendistribusian.

Proses ini sama halnya dengan M2 yang melalui 6 agen dalam individu pertama tadi yaitu [9 2 1 13 15 14]. Sehingga rute yang dilalui oleh M2 adalah produsen-9-2-1-13-15-14-produsen. Perhitungan distribusi yang dilalui diambil dalam satu contoh yaitu agen 9 adalah 20,5 x ((4000-3500) x 80/300) = 273500. Sedang biaya untuk kembali saat muatan kosong adalah 28.5 x 5000 = 142500. Total biaya keseluruhan diperoleh dari hasil total pendistribusian M1 + biaya distribusi M2 + biaya muatan akhir.

Setelah total *penalty* dan biaya dari permasalahan tersebut didapatkan, maka nilai *fitness* dapat dihitung menggunakan persamaan *fitness* sebelumnya. Nilai *fitness* yang dihasilkan adalah

Berdasarkan ketentuan yang dijelaskan maka fungsi *fitness* didapat dari

$$f = \frac{1}{total\ biaya + (penalty\ x\ total\ sisa\ muatan)}\ x\ C$$

Fungsi *fitness* yang digunakan pada permasalahan ini yaitu fungsi minimasi. *Fitness* didapatkan dari nilai penalty dan total jarak. Penalti diberikan jika sebuah mobil memiliki sisa muatan. Jumlah dari seluruh penalti dan total jarak akan menjadi tolak ukur untuk mengetahui nilai *fitness*. Berdasarkan fungsi minimasi, semakin besar nilai penalti dan total jarak, maka *fitness* yang didapatkan semakin kecil begitu juga sebaliknya. Pada penelitian ini, nilai *fitness* yang dihasilkan akan dikalikan dengan C atau konstanta agar proses perhitungan *fitness* mudah dilakukan dan mencegah ketika nilai *fitness* yang dihasilkan bernilai kecil

3.4.2 Inisialisasi populasi

Inisialisasi populasi dibangkitkan dengan merandom urutan agen 1-15. Contoh *popsize = 5,* maka akan dihasilkan populasi sesuai dengan Tabel 3.8 berikut.

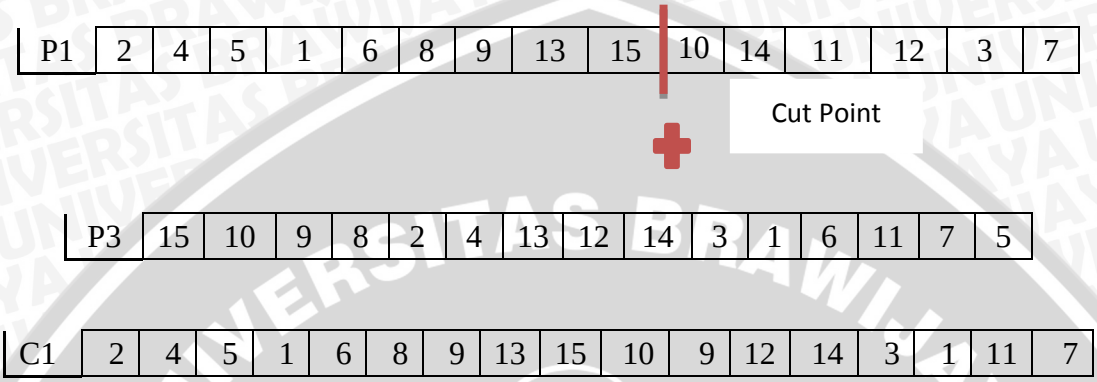
Tabel 3.8 Tabel Populasi Inisial

	Kromosom														
	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9	x_{10}	x_{11}	x_{12}	x_{13}	x_{14}	x_{15}
P1	2	4	5	1	6	8	9	13	15	10	14	11	12	3	7
P2	1	3	6	7	9	11	13	14	15	10	12	8	2	4	5
P3	15	10	9	8	2	4	13	12	14	3	1	6	11	7	5
P4	3	9	13	1	5	2	4	8	6	10	11	12	14	15	7
P5	5	7	9	10	15	4	14	13	12	11	3	2	1	6	8

3.4.3 Reproduksi

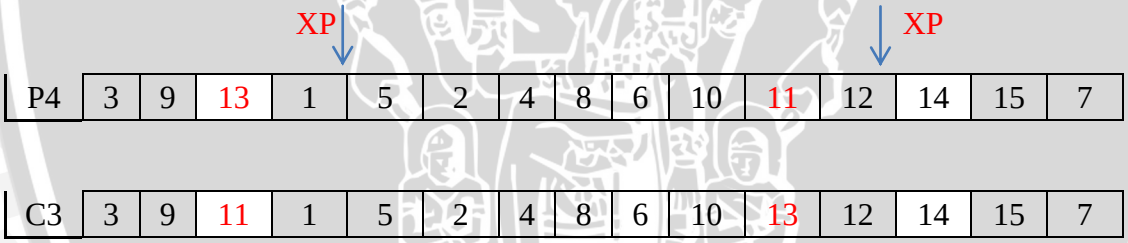
Pada proses reproduksi, digunakan proses *crossover* dan *mutasi* untuk menghasilkan keturunan *(offspring)* dari induk yang telah dibangkitkan sebelumnya. Contoh jika sebuah c*rossover rate* yang digunakan adalah 0.4, dan *mutation rate* yang digunakan adalah 0.6, maka nantinya akan didapatkan 0,4 x 5 = 2 anak dari proses *crossover* dan 0,6 x 5 = 3 anak dari proses mutasi. Angka 5 didapatkan sesuai jumlah masukan *popsize.*, Langkah pertama yang dilakukan

adalah memilih dua induk yang nantinya akan dilakukan proses crossover pada kedua induk tersebut. Proses *crossover* yang pertama adalah menetukan *cut point*, *crossover* menghasilkan satu *child* setiap prosesnya. Anak dari hasil *crossover* memiliki kromosom dari induk pertama sepanjang *cut point*, dan mendapatkan sisanya dari induk kedua. Contoh proses *crossover* pada Gambar 3.1.



Gambar 3.3 Proses Crossover

Sedangkan untuk metode mutasi yang digunakan adalah *reciprocal exchange mutation.* Dalam hal ini, memilih *exchange point*/XP dalam satu induk kemudian di tukar. Contoh proses mutasi pada Gambar 3.2.



Gambar 3.4 Proses Mutasi

3.4.4 Evaluasi

Setelah mendapatkan jumlah anak (*offspring*) dari sejumlah individu, langkah selanjutnya adalah evaluasi. Evaluasi ditujukan untuk mengetahui nilai masing-masing individu dalam suatu populasi. Semakin besar nilai fitness yang dimiliki oleh suatu individu, maka kemampuan untuk bertahan hidup juga semakin besar. Penelitian ini bertujuan untuk menentukan rute mobil yang terpendek pada permasalahan optimasi distribusi barang.

3.4.5 Seleksi

Seleksi yang digunakan pada metode ini adalah *elitism selection* yaitu seleksi yang melibatkan individu terbaik dalam populasi yang ada. Dalam hal ini,

metode yang digunakan untuk seleksi adalah memilih fitness terbaik dari kumpulan individu di populasi (parent) dan offspring (Mahmudy, 2013). Dalam hal ini, mengurutkan nilai fitness terbaik dari yang jumlah nya besar ke fitness yang jumlahnya lebih kecil. Jadi, dalam seleksi ini hanya individu-individu terbaik yang akan menjadi individu generasi berikutnya.

Proses pengurutan pada permasalahan ini menggunakan *buble sort*. Buble sort merupakan algoritma sorting dengan melakukan penukaran data tepat disebelahnya secara terus menerus sampai tidak ada perubahan lagi atau urutannya sudah benar. Untuk lebih jelasnya bisa dilihat pada ilustrasi berikut :

Misalnya terdapat sebuah array dengan elemen – elemen “4, 2, 5, 3, 9”. Array ini akan diurutkan dari nilai yang tertinggi hingga ke rendah maka prosesnya adalah sebagai berikut :

Iterasi pertama

1. (4 2 5 3 9) melakukan pengecekan apakah 4 lebih besar dari 2? Jika iya maka tetap menjadi (4 2 5 3 9)
2. (4 2 5 3 9) melakukan pengecekan apakah 2 lebih besar dari 5? Jika tidak maka posisi 2 dan 5 ditukar menjadi (4 5 2 3 9)
3. (4 5 2 3 9) melakukan pengecekan apakah 2 lebih besar dari 3? Jika tidak maka posisi 2 dan 3 ditukar menjadi (4 5 3 2 9)
4. (4 5 3 2 9) melakukan pengecekan apakah 2 lebih besar dari 9? Jika tidak maka posisi 2 dan 9 ditukar menjadi (4 5 3 9 2)

Proses ini akan lanjut ke iterasi selanjutnya sampai urutan benar benar tidak berubah atau urutan sudah benar.

Pada tahap seleksi, individu terpilih sebagai generasi selanjutnya dipilih dengan menggunakan metode *elitism*. Jumlah individu dipilih sebanyak jumlah *pop_size* dimana pada contoh kasus nilai *pop_size* = 5. Individu dipilih dengan meranking seluruh individu berdasarkan nilai fitness terbaik. Contoh individu hasil *crossover* dan *mutation* pada tabel 3.9 dan contoh individu dengan hasil fitness terbaik pada tabel 3.10

Tabel 3.9 Individu Baru Hasil Seleksi

	Kromosom															Fitness
	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9	x_{10}	x_{11}	x_{12}	x_{13}	x_{14}	x_{15}	
P1	1	3	6	7	9	11	13	14	15	2	12	10	8	4	5	0,9435
P2	3	9	13	1	5	2	4	8	6	10	11	12	14	15	7	0,62739
P3	15	10	9	8	2	6	13	12	14	1	4	2	11	7	5	0,8132
P4	1	3	6	7	9	11	13	14	15	10	12	8	2	4	5	0,67995
P5	14	1	8	5	7	15	10	12	9	1	4	2	11	7	5	0,8143
P6	2	4	5	1	6	8	9	13	15	10	14	11	12	3	7	0,77074
P7	12	4	1	5	14	9	8	13	15	10	6	7	2	3	11	0,9027
P8	3	9	5	1	6	8	4	13	15	10	14	11	12	2	7	0,8232
P9	15	10	9	8	2	4	13	12	14	3	1	6	11	7	5	0,7888
P10	5	7	9	10	15	4	14	13	12	11	3	2	1	6	8	0,80522

Tabel 3.10 Individu dengan fitness terbaik

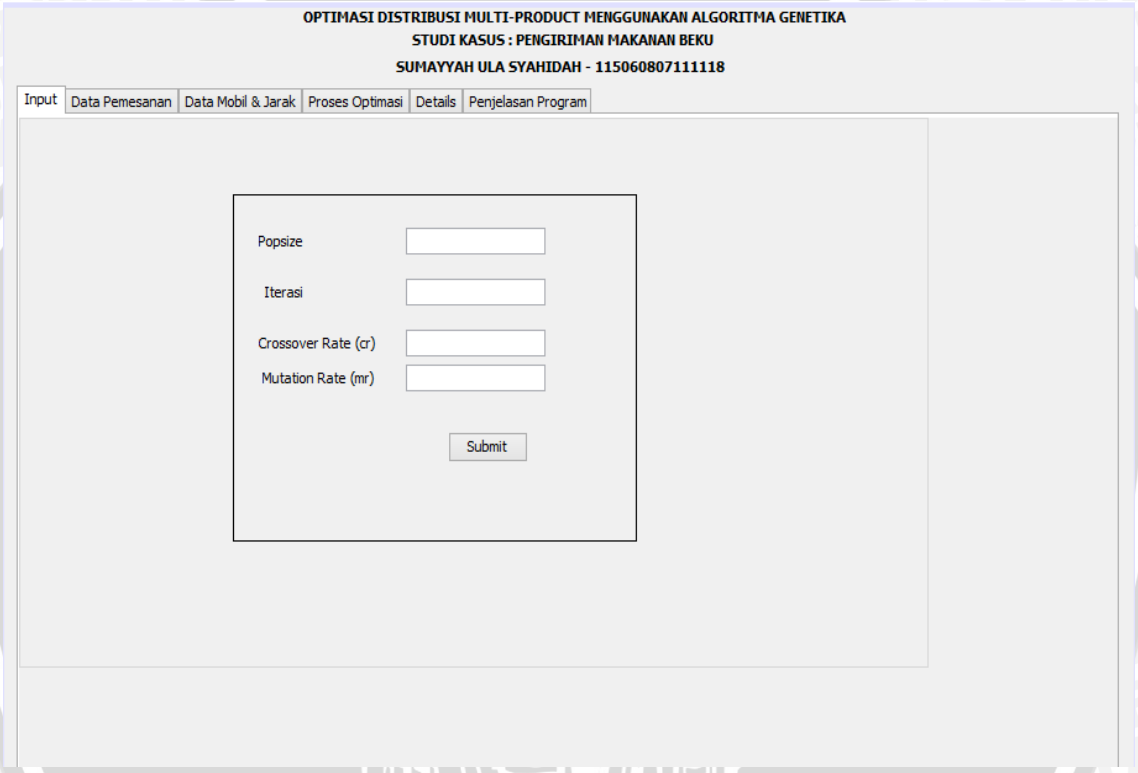
	Kromosom															Fitness
	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9	x_{10}	x_{11}	x_{12}	x_{13}	x_{14}	x_{15}	
P1	3	9	13	1	5	2	4	8	6	10	11	12	14	15	7	0,627392
P2	1	3	6	7	9	11	13	14	15	10	12	8	2	4	5	0,679948
P3	2	4	5	1	6	8	9	13	15	10	14	11	12	3	7	0,770743
P4	15	10	9	8	2	4	13	12	14	3	1	6	11	7	5	0,788799
P5	5	7	9	10	15	4	14	13	12	11	3	2	1	6	8	0,805218

Pada tabel 3.10 individu yang diambil sebanyak 5 berdasarkan dengan nilai popsize atau populasi yang dimasukkan *user*. Individu diurutkan dari individu yang memiliki nilai fitness yang paling kecil hingga nilai fitness yang tinggi.

BAB IV
PERANCANGAN

4.1 Perancangan Interface

Perancangan interface atau biasa disebut dengan *interface* ini terdiri dari satu halaman program yang berisi tabel representasi kromosom dan tabel hasil.*Interface* pada program ini dibuat dengan sedemikian rupa agar dapat diakses oleh user peneliti saja.Perancangan *Interface*Pada gambar 4.1



Gambar 4.1 *Interface*

Keterangan Gambar :

1. Tab paned pertama berisi input, untuk menginputkan populasi, generasi, *crossoverrate* dan *mutation rate*.
2. Tab paned kedua berisi data pemesanan oleh agen.
3. Tab paned ketiga berisi data mobil dan data jarak per agen.
4. Tab paned keempat merupakan hasil akhir individu dari proses optimasi.
5. Tab paned ke lima merupakan detail dari proses optimasi .

6. Tab paned ke enam berisi penjelasan dari individu terbaik.

4.2 Perancangan Pengujian

Dalam Algoritma Genetika, tidak ada metode yang pasti, karena itu algoritma genetika mencari hasil yang hampir optimal. Oleh sebab itu, dilakukan uji coba untuk menguji keoptimalan program yaitu sebagai berikut :

1. Uji Coba populasi yang optimal.
2. Uji coba untuk menentukan iterasi yang optimal.
3. Uji Coba untuk menentukan *crossoverrate* dan *mutationrate* yang terbaik.

4.2.1 Uji coba Populasi

Uji coba banyak populasi ini dilakukan agar mendapat banyak populasi yang tepat untuk menghasilkan solusi yang terbaik. Karena ukuran populasi mempengaruhi ketepatan algoritma genetika dalam menghasilkan solusi yang optimum. Contoh banyak populasi pada tabel 4.1 dibawah ini adalah keipatan 20 yaitu 20, 40, 60, 80, 100, 120 dan 140.

Tabel 4.1 Rancang Uji Coba Populasi

Populasi	Niali Fitness										Rata-rata fitnes
	Pengujian ke-										
	1	2	3	4	5	6	7	8	9	10	
20											
40											
60											
80											
100											
120											
140											

4.2.2 Uji Coba iterasi

Pada tahap ini dilakukan uji coba iterasi pada program untuk menghasilkan hasil yang optimal dalam memecahkan masalah optimasi ini. Pengujian ini bertujuan untuk mengetahui pengaruh perubahan jumlah iterasi terhadap besar nilai fitness maksimum yang dihasilkan. Uji coba dilakukan pada iterasi kelipatan 500 dari 500 hingga 3500. Dengan nilai cr dan mr hasil uji coba kombinasi cr dan mr terbaik dan ukuran populasi yang diuji cobakan juga dari hasil uji coba populasi terbaik. Contoh uji coba iterasi pada tabel 4.2 dibawah ini.

Tabel 4.2 Rancang Uji Coba Generasi

Banyak Generasi	Nilai *Fitness*										Rata-rata *fitness*
	Percobaan generasi ke-										
	1	2	3	4	5	6	7	8	9	10	
500											
1000											
1500											
2000											
2500											
3000											
3500											

4.2.3 Uji Coba Crossover dan mutation rate terbaik

Uji coba ini dilakukan untuk mendapatkan kombinasi crossoverrate dan mutationrate yang optimal dalam perhitungan algoritma genetika ini. Tujuan dilakukan uji coba ini untuk menentukan seberapa besarkah pengaruh *crossoverrate* dan *mutation* rate pada nilai *fitness* nantinya. Nilai cr dan mr yang digunakan dalam uji coba ini adalah antara nilai 0 sampai dengan 1. Setiap pengujian dilakukan 10 kali percobaan kemudian diambil nilai *fitnessnya*. Tabel 4.3 merupakan contoh uji coba nilai kombinasi cr dan mt.

Tabel 4.3 Rancang Uji Coba Kombinasi Cr Mr

cr	mr	Nilai Fitness										Rata-rata fitness
		Pengujian Ke-										
		1	2	3	4	5	6	7	8	9	10	
1	0.0											
0.9	0.1											
0.8	0.2											
0.7	0.3											
0.6	0.4											
0.5	0.5											
0.4	0.6											
0.3	0.7											
0.2	0.8											
0.1	0.9											
0.0	1											

BAB V

IMPLEMENTASI

5.1 Implementasi Program

5.1.1 Proses pengecekan array

Pada proses ini digunakan untuk mengacak array data dari agen dan truk sebagai inisialisai awal. Gambar 5.1 merupakan source code dari proses mengacak array

```
1   public static void shuffle(Object[] array){
2       int noOfElements = array.length;
3       Object temp;
4       int s;
5       for(int i=0; i<noOfElements; i++){
6           s = i + (int)(Math.random()*(noOfElements-i));
7           temp = array[s];
8           array[s] = array[i];
9           array[i] = temp;
10      }
11  }
12
```

Gambar 5.1 proses pengecekan array

Penjelasan dari source code diatas adalah :

1. Baris 2-4 merupakan proses inisialisasi variabel
2. Baris 5 - 11 merupakan proses untuk mengacak array. Pada baris 6 digunakan untuk mengacak indek array. Pada baris 8 – 10 merupakan proses pertukaran nilai dari indeks yang diacak tersebut dengan indeks yang ditunjuk oleh perulangan

5.1.2 Implementasi Proses perhitungan fitness

Pada proses perhitungan fitness memerlukan 3 parameter yaitu data agen yang disimpan dalam array, data mobil yang disimpan dalam array dan jenis nilai yang dikembalikan. Gambar 5.2 merupakan source code dari proses perhitungan fitness

```
1   public double hitungFitness(int kroAgen[], int kroMobil[], int
2   nilai_kembali){
```

```
3   double hasil;
4         int pesan[] = new int[jumlah_agen];
5         int tempPesan1[] = new int[jumlah_agen];
6         int tempPesan2[]= new int[jumlah_agen];
7         int tempPesan3[] = new int[jumlah_agen];
8         int tempPesan4[]= new int[jumlah_agen];
9         long muatan[] = new long[mobil];
10        long tempMuatan[] = new long[mobil];
11        long muatan1[] = new long[mobil];
12        long tempMuatan1[] = new long[mobil];
13        long biayaKosong[] = new long[mobil];
14        long biayaPenuh[] = new long[mobil];
15        int jumTruk[] = new int[mobil];
16        int kend[] = new int[mobil];
17        int jarak[][] = new int[jumlah_agen+1][jumlah_agen+1];
18
19        for(int i=0; i<dPes.idPesan; i++){
20           pesan[i] = dPes.jumPesan[i];
21           tempPesan1[i] = dPes.pesan1[i];//bayar
22           tempPesan2[i] = dPes.pesan2[i];//muatan
23           tempPesan3[i]= dPes.pesan1[i];//muatan
24           tempPesan4[i]= dPes.pesan2[i];//bayar
25
26        }
27
28        for(int i=0; i<dK.n; i++){
29           muatan[i] = dK.kapasitas[i];
30           tempMuatan[i] = dK.kapasitas[i];
31           muatan1[i] = dK.kapasitas[i];
32           tempMuatan1[i] = dK.kapasitas[i];
33        }
34        System.arraycopy(dK.angKosong, 0, biayaKosong, 0, dK.n);
35        System.arraycopy(dK.angBayar, 0, biayaPenuh, 0, dK.n);
36        System.arraycopy(dK.jumKend, 0, jumTruk, 0, dK.n);
37        System.arraycopy(dK.idKend, 0, kend, 0, dK.n);
38        for( int i=0; i<dJ.baris; i++){
39           System.arraycopy(dJ.jar[i], 0, jarak[i], 0, dJ.jumlah);
40        }
41        int sisaMuatan = 0;
42        int sisaMuatan1 = 0 , sisaMuatanTot = 0;
43        int i = 0, x = 0;
44        double biaya =0, totalBiaya1 =0, totalBiaya=0, totalBiaya2 = 0;
45        double  biaya1 = 0;
46        int sisa = 0;
47        int sisa1 = 0;
48        int simpan, bayar, pjgJalur;
49        int simpan1, bayar1, pjgJalur1;
```

```
50          int jalur[] = new int[kroMobil.length];
51          for(int j=0; j<kroMobil.length; j++){
52             System.out.println("");
53             kend [kroMobil[0]-1] = kroMobil [0];
54             kend [kroMobil [1]-1] = kroMobil [1];
55             System.out.println("mobil 0[ : "+kroMobil [0]);
56             System.out.println("mobil [1]: "+kroMobil [1]);
57             jalur[j] = 0;
58             bayar = 1;
59             if(x<kroAgen.length){
60                do{
61                   if((tempPesan2[kroAgen[x]-1] >=
62          tempMuatan[kend[kroMobil[0]-1]-1])&& (tempPesan3[kroAgen[x]-1]
63          >= tempMuatan[kend[kroMobil[1]-1]-1])){
64                      sisa = 0;
65                      tempPesan2[kroAgen[x]-1] -=
66          tempMuatan[kend[kroMobil[0]-1]-1];
67                      tempMuatan[kend[kroMobil[0]]-1] =
68          muatan[kend[kroMobil[0]-1]-1];
69                      System.out.println("muatan:
70          "+muatan[kend[kroMobil[0]-1]-1]);
71                      if (tempPesan3[kroAgen[x]-1] >=
72          tempMuatan1[kend[kroMobil[1]-1]-1]){
73                        sisa1=0;
74                        tempPesan3[kroAgen[x]-1] -=
75          tempMuatan1[kend[kroMobil[1]-1]-1];
76                          tempMuatan1[kend[kroMobil[1]-1]-1] =
77          muatan1[kend[kroMobil[1]-1]-1];
78
79          System.out.println("muatan1:"+muatan1[kend[kroMobil[1]-1]-1]);
80                                    }
81                   }
82                   else{
83                      sisa = (int) (tempMuatan[kend[kroMobil[0]-1]-1] -
84          tempPesan2[kroAgen[x]-1]);
85                      tempPesan2[kroAgen[x]-1] = 0;
86                      tempMuatan[kend[kroMobil[0]-1]-1] = sisa;
87                      sisa1 = (int) (tempMuatan1[kend[kroMobil[1]-1]-1] -
88          tempPesan3[kroAgen[x]-1]);
89                      tempPesan3[kroAgen[x]-1] = 0;
90                      tempMuatan1[kend[kroMobil[1]-1]-1] = sisa1;
91                      System.out.println("sisa: "+sisa);
92                      System.out.println("sisa1"+sisa1);
93                      x++;
94                   }
95                   jalur[j]++;
96                }while((sisa>0 && sisa1>0 && x<kroAgen.length));
```

```
97              System.out.println("jalur "+jalur[j]);
98              tempMuatan[kend[kroMobil[0]-1]-1] =
99      muatan[kend[kroMobil[0]-1]-1];
100             System.out.println("kend 1: "+kend[kroMobil[0]-1]);
101             tempMuatan1[kend[kroMobil[1]-1]-1] =
102     muatan1[kend[kroMobil[1]-1]-1];
103             System.out.println("kend 2: "+kend[kroMobil[1]-1]);
104             do{
105                System.out.println("");
106                System.out.print(" - Agen "+kroAgen[i]);
107                if((tempPesan1[kroAgen[i]-1] >=
108     tempMuatan[kend[kroMobil[0]-1]-1]&& tempPesan4[kroAgen[i]-1]
109     >= tempMuatan1[kend[kroMobil[1]-1]-1])){
110                     sisaMuatan = 0;
111                     simpan = (int)(tempMuatan[kend[kroMobil[0]-1]-1]);
112                     tempPesan1[kroAgen[i]-1] -=
113     tempMuatan[kend[kroMobil[0]-1]-1];
114                     tempMuatan[kend[kroMobil[0]-1]-1] =
115     muatan[kend[kroMobil[0]-1]-1];
116                       if(tempPesan4[kroAgen[i]-1] >=
117     tempMuatan1[kend[kroMobil[1]-1]-1]){
118                       sisaMuatan1 = 0;
119                       simpan1 =
120     (int)(tempMuatan1[kend[kroMobil[1]-1]-1]);
121                       tempPesan4[kroAgen[i]-1] -=
122     tempMuatan1[kend[kroMobil[1]-1]-1];
123                       tempMuatan1[kend[kroMobil[1]-1]-1] =
124     muatan1[kend[kroMobil[1]-1]-1];
125                       if(bayar == 1){
126                     biaya1 =
127     jarak[0][kroMobil[i]]*(biayaKosong[kend[kroMobil[0]-
128     1]]+(biayaPenuh[kend[kroMobil[0]-1]]-
129     biayaKosong[kend[kroMobil[0]-1]])*(muatan[kend[kroMobil[0]-1]]-
130     sisa)/(muatan[kend[kroMobil[0]-1]]));
131                     if(jalur[j]==1){
132                       biaya1 +=
133     jarak[kroAgen[i]][0]*biayaKosong[kend[kroMobil[0]-1]];
134                       //System.out.println("biaya kososngmobil1:
135     "+biayaKosong[kend[kroMobil[0]-1]]);
136                     }
137                   }
138                   else{
139                     biaya1 = jarak[kroAgen[i-
140     1]][kroAgen[i]]*(biayaKosong[kend[kroMobil[1]-1]-
141     1]+(biayaPenuh[kend[kroMobil[1]-1]-1]-
142     biayaKosong[kend[kroMobil[1]-1]-
143     1])*(simpan+sisaMuatan)/muatan[kend[kroMobil[1]-1]-1]);
```

```
144                         if(jalur[j]==1){
145                             biaya1 +=
146 jarak[kroAgen[i]][0]*biayaKosong[kend[kroMobil[1]-1]-1];
147                             //System.out.println("biayakosongmobil 2:
148 "+biayaKosong[kend[kroMobil[1]-1]-1]);
149 
150                         }
151                     }
152                     totalBiaya1 += biaya1;
153                     }
154                         if(bayar == 1){
155                         biaya =
156 jarak[0][kroMobil[i]]*(biayaKosong[kend[kroMobil[0]-
157 1]]+(biayaPenuh[kend[kroMobil[0]-1]]-
158 biayaKosong[kend[kroMobil[0]-1]])*(muatan[kend[kroMobil[0]-1]]-
159 sisa)/(muatan[kend[kroMobil[0]-1]]));
160                         if(jalur[j]==1){
161                             biaya +=
162 jarak[kroAgen[i]][0]*biayaKosong[kend[kroMobil[0]-1]];
163                         }
164                     }
165                     else{
166                         biaya = jarak[kroAgen[i-
167 1]][kroAgen[i]]*(biayaKosong[kend[kroMobil[1]-1]-
168 1]+(biayaPenuh[kend[kroMobil[1]-1]-1]-
169 biayaKosong[kend[kroMobil[1]-1]-
170 1])*(simpan+sisaMuatan)/muatan[kend[kroMobil[1]-1]-1]);
171                         if(jalur[j]==1){
172                             biaya +=
173 jarak[kroAgen[i]][0]*biayaKosong[kend[kroMobil[1]-1]-1];
174 
175                         }
176                     }
177                     totalBiaya += biaya;
178                 }
179                 else{
180                     sisaMuatan = (int)tempMuatan[kend[kroMobil[0]-1]-1]
181 - tempPesan1[kroAgen[i]-1];
182                     tempMuatan[kend[kroMobil[0]-1]-1] = sisaMuatan;
183 
184                     sisaMuatan1 = (int)tempMuatan1[kend[kroMobil[1]-1]-
185 1] - tempPesan4[kroAgen[i]-1];
186                     tempMuatan1[kend[kroMobil[1]-1]-1] = sisaMuatan1;
187 
188                     if(bayar == 1){
189                         biaya =
190 jarak[0][kroAgen[i]]*(biayaKosong[kend[kroMobil[0]-1]-
```

```
1]+(biayaPenuh[kend[kroMobil[0]-1]-1]-
biayaKosong[kend[kroMobil[0]-1]-1])*(muatan[kend[kroMobil[0]-1]-
1]-sisa)/muatan[kend[kroMobil[0]-1]-1]);
                        if(jalur[j]==1){
                            biaya +=
jarak[kroAgen[i]][0]*biayaKosong[kend[kroMobil[0]-1]-1];

                        }
                    }
                    else{
                        biaya = jarak[kroAgen[i-
1]][kroAgen[i]]*(biayaKosong[kend[kroMobil[1]-1]-
1]+(biayaPenuh[kend[kroMobil[1]-1]-1]-
biayaKosong[kend[kroMobil[1]-1]-1])*(tempPesan1[kroAgen[i]-
1]+sisaMuatan-sisa)/muatan[kend[kroMobil[1]-1]-1]);
                        if(jalur[j]==1){
                            biaya +=
jarak[kroAgen[i]][0]*biayaKosong[kend[kroMobil[1]-1]-1];
                        }
                    }
                    totalBiaya += biaya;
                    i++;
                }

                jalur[j]--;
                bayar++;

            }while(sisaMuatan>0 && i<kroAgen.length);
        }
    }
    if(nilai_kembali == 1){
        //hasil = (double) (100000 / (totalBiaya + 100 * sisaMuatan));
        totalBiaya = totalBiaya + totalBiaya1;
        sisaMuatanTot = sisaMuatan1 +sisaMuatan;
        hasil = (double)1000000/(totalBiaya+sisaMuatanTot);
        String fitness = new DecimalFormat("##.###").format(hasil);
        System.out.println("Cek Fitness: "+fitness+" Biaya:
"+totalBiaya+" Sisa: "+sisaMuatan);
        return hasil;
    }
    else if(nilai_kembali == 2){
        hasil = (int) sisaMuatan+sisaMuatan1;
        System.out.println("");
        System.out.println("Cek Sisa Muatan: "+hasil);
        return hasil;
    }
    else{
```

```
238             hasil = (int) totalBiaya;
239             System.out.println("");
240             System.out.println("Cek Total Biaya: "+hasil);
241             return hasil;
242         }
243 }
```

Gambar 5.2 proses perhitungan *fitness*

Penjelasan dari *source code* diatas adalah :

1. Baris 3 – 54 merupakan proses inisialisasi variabel.
2. Baris 59 – 104 merupakan proses menghitung sisa muatan tiap mobil yang digunakan untuk mengangkut barang ke agen.
3. Baris 105 – 110 merupakan proses penyimpanan kembali nilai muatan mobil ke dalam variabel temp_muatan.
4. Baris 129 – 234 merupakan proses menghitung total biaya dari proses distribusi barang.
5. Baris 236 – 259 merupakan proses untuk menentukan nilai mana yang akan dikembalikan. Jika nilai_kembali sama dengan 1 maka nilai yang dikembalikan adalah nilai fitness namun jika nilai_kembali tidak sama dengan 1 maka nilai yang dikembalikan adalah total biaya.

5.1.3 Implementasi Proses *crossover*

Proses crossover yang digunakan menggunakan *one cut point.* Dengan parameter yang digunakan adalah titik potong 1 dan dua parent. Gambar 5.3 merupakan *sourcecode* dari proses *crossover :*

```
1   public String childCr(int titikS1,  String p1, String p2){
2     //try{
3     String pisah1[] = p1.split(" ");
4     String pisah2[] = p2.split(" ");
5     String anak = "";
6     int segmen1_p1[] = new int[jumlah_agen];
7     int segmen1_p2[] = new int[jumlah_agen];
8     int anak_s1[] = new int[jumlah_agen];
9     int j = 0;
10    int simpan = 0;
11    for(int i=0; i<pisah1.length; i++){
12      if(i<segmen1_p1.length){
13        segmen1_p1[i] = Integer.parseInt(pisah1[i]);
14        segmen1_p2[i] = Integer.parseInt(pisah2[i]);
```

```
15
16              }
17          }
18          //crossover segmen1
19          for(int i=0; i<titikS1; i++){
20             anak_s1[i] = segmen1_p1[i];
21          }
22          int x = 0;
23          for(int i=0; i<segmen1_p1.length; i++){
24             for(int k=0; k<titikS1; k++){
25                if(anak_s1[k]!=segmen1_p2[i]) simpan = segmen1_p2[i];
26                else{
27                   simpan = 0; break;
28                }
29             }
30             if(simpan!=0){
31                anak_s1[titikS1+x] = simpan;
32                x++;
33             }
34             anak += anak_s1[i];
35             anak += " ";
36          }
37          return anak;
        }
```

Gambar 5.3 implementasi proses *crossover*

1. Baris 4 dan 5 merupakan pemisahan kromosom tipe data string menjadi sebuah array. Pemisahan kromosom ini menggunakan spasi. Parent 1 akan disimpan ke dalam variabel pisah1 sedangkan parent 2 akan disimpan ke dalam variabel pisah2
2. Baris 6-10 proses inisialisasi variabel
3. Baris 11-16 merupakan proses untuk menyimpan hasil pemisahan kromosoom.
4. Baris 18 –34 merupakan proses crossover. Baris 19 – 20 adalah proses crossover dimana gen pertama akan diambil dari parent pertama sesuai dengan cut point.

5.1.4 Implementasi Proses Mutasi

Proses mutasi yang digunakan adalah reciprocal exchange. Proses mutasi ini memerlukan 2 parameter yaitu titik 1 untuk dan titidk 2 untuk satu kromosom. Gambar 5.4 merupakan source code dari proses mutasi

```
1   public String childMr(int titik1S1, int titik2S1,  String p, int s){
```

```
2       //try{
3       String pisah[] = p.split(" ");
4       String anak = "";
5       int segmen1_p[] = new int[jumlah_agen];
6       //int segmen2_p[] = new int[mobil];
7       int cpS1[] = new int[jumlah_agen];
8       //int cpS2[] = new int[mobil];
9       int j = 0;
10      for(int i=0; i<pisah.length; i++){
11         if(i<segmen1_p.length){
12            segmen1_p[i] = Integer.parseInt(pisah[i]);
13            cpS1[i] = Integer.parseInt(pisah[i]);
14         }
15      }
16      if(s==1){
17
18         segmen1_p[titik1S1] = cpS1[titik2S1];
19         segmen1_p[titik2S1] = cpS1[titik1S1];
20
21      }
22
23      else{
24         segmen1_p[titik1S1] = cpS1[titik2S1];
25         segmen1_p[titik2S1] = cpS1[titik1S1];
26
27      }
28      j = 0;
29      for(int i=0; i<pisah.length; i++){
30         if(i<segmen1_p.length){
31            anak += segmen1_p[i];
32            anak += " ";
33         }
34
35      }
36      return anak;
37
    }
```

Gambar 5.4 implementasi proses mutasi

Penjelasan sourcecode :

1. Baris 3 merupakan kromosom yang dipisah di taruh didalam sebuah array
2. Baris 4-9 merupakan inisialisasi variabel
3. Baris 10-16 menyimpal hasil pemisahan kromosom
4. Baris 17-37 proses mutasi

5.1.5 Implementasi Proses Seleksi

Proses seleksi yang digunakan adalah *elitism*. Proses *elitism* disini menggunakan *buble sort*. Gambar 5.5 merupakan *source code* dari proses seleksi :

```
1       //seleks
2       System.out.println("Hasil Seleksi");
3               for(int coba=0; coba<popsize+childCr+childMr;
4       coba++){
5                       cpIndividu[counter][coba] =
6       individu[counter][coba];
7                       cpFitness[counter][coba] = fitness[counter][coba];
8               }
9       //bubble sort
10       String temp1;
11      double temp;
12     for(int i=1; i<=popsize+childCr+childMr; i++){
13       for(int j=0; j<popsize+childCr+childMr-i; j++){
14
15      if(cpFitness[counter][j]<cpFitness[counter][j+1]){
16                      temp = cpFitness[counter][j];
17                      temp1 = cpIndividu[counter][j];
18                      cpFitness[counter][j] =
19      cpFitness[counter][j+1];
20                      cpIndividu[counter][j] =
21      cpIndividu[counter][j+1];
22                      cpFitness[counter][j+1] = temp;
23                      cpIndividu[counter][j+1] = temp1;
24                  }
25                }
26              }
27              //hasil seleksi
28              for(int ulang = 0; ulang<popsize; ulang++){
29                individu[counter+1][ulang] =
30      cpIndividu[counter][ulang];
31                fitness[counter+1][ulang] =
32      cpFitness[counter][ulang];
33                System.out.print(ulang+"
34      "+individu[counter][ulang]+" - ");
35                System.out.print(fitness[counter+1][ulang]);
36                System.out.println("");
37              }
38
39
40
```

Gambar 5.5 proses implementasi seleksi

Penjelasan dari *source code* diatas adalah :

1. Baris 2 – 8 merupakan proses untuk menyimpan variabel chromosome dan fitness ke variabel sementara yaitu dengan nama temp_chromosome dan temp_fitness.
2. Baris 10 – 11 merupakan proses inisialisasi variabel.
3. Baris 12 – 28 merupakan proses *sorting* dengan *buble sort*.
4. Baris 30 – 39 merupakan hasil seleksi untuk generasi selanjutnya. Individu yang diambil adalah individu dengan nilai fitness terbaik sesuai dengan jumlah *popsize*.

5.1.6 Implementasi Proses untuk Algoritma Genetika

Pada proses ini merupakan proses untuk menjalan algoritma genetika. Proses ini merupakan proses utama yang digunakan untuk memanggil proses – proses diatas, Gambar 5.6 merupakan source code dari proses algoritma genetika :

```
1   // TODO add your handling code here:
2         Object[] row = {"Iterasi", "Chomosome", "Biaya", "Sisa Muatan",
3   "Fitness"};
4         model = new DefaultTableModel(null, row);
5         tabel_proses.setModel(model);
6         model.getDataVector().removeAllElements();
7         model.fireTableDataChanged();
8         Object[] a = new Object[5];
9
10        //inisialisasi variabel
11        int popsize = Integer.parseInt(txtPop1.getText());
12        int iterasi = Integer.parseInt(txtItr1.getText());
13        double Cr = Double.parseDouble(txtCr1.getText());
14        double Mr = Double.parseDouble(txtMr1.getText());
15        int childCr = (int)(popsize*Cr);
16        int childMr = (int)(popsize*Mr);
17        int child = childCr+childMr;
18
19        //inisialisasi populasi
20        String [][] individu = new
21  String[iterasi+2][popsize+childCr+childMr];
22            double fitness[][] = new double
23  [iterasi+2][popsize+childCr+childMr];
24            String [][] cpIndividu = new
25  String[iterasi+2][popsize+childCr+childMr];
26            double cpFitness[][] = new double
```

```
[iterasi+2][popsize+childCr+childMr];
        long biaya = 0;
        int sisaMuatan = 0;

        String [] dataAgen = new String[jumlah_agen];
        for(int i=0; i<dataAgen.length; i++){
           dataAgen[i] = Integer.toString(i+1);
           System.out.print(dataAgen[i]+" ");
        }
        String [] dataMobil = new String[mobil];
        for(int i=0; i<dataMobil.length; i++){
           dataMobil[i] = Integer.toString(i+1);
           System.out.print(dataMobil[i]+" ");
        }
        //kromosom
        int kroAgen[] = new int[jumlah_agen];
        int kroMobil[] = new int[mobil];
        for(int coba=0; coba<popsize; coba++){
           individu[0][coba]="";
           GUI.shuffle(dataAgen);
           GUI.shuffle(dataMobil);
           for(int i=0; i<dataAgen.length; i++){
              individu[0][coba] += dataAgen[i];
              individu[0][coba] += " ";
              kroAgen[i] = Integer.parseInt(dataAgen[i]);
           }
           for(int i=0; i<dataMobil.length; i++){
              individu[0][coba] += dataMobil[i];
              individu[0][coba] += " ";
              kroMobil[i] = Integer.parseInt(dataMobil[i]);
           }
           fitness[0][coba] = (double) hitungFitness(kroAgen,
kroMobil, 1);
           System.out.println(individu[0][coba]+"-"+fitness[0][coba]);
        }
        //PERHITUNGAN FITNESS

        int counter;
        for(counter=0; counter<=iterasi; counter++){
           System.out.println("iterasi "+counter);
           for(int coba=0; coba<popsize; coba++){
              System.out.print(coba+" "+individu[counter][coba]+" - ");
              System.out.println(fitness[counter][coba]);
              System.out.println("");
           }
           //crossover
           String p1, p2;
```

```
74              int t1, t2;
75              t1 = (int)(double)(Math.random()*jumlah_agen);
76              //t2 = (int)(double)(Math.random()*mobil);
77              if(t1==0) t1=jumlah_agen;
78              //if(t2==0) t2=mobil;
79              System.out.println("titik crossover: "+t1);
80              for(int coba=popsize; coba<popsize+childCr; coba++){
81                 p1 =
82  individu[counter][(int)(double)(Math.random()*popsize)];
83                 p2 =
84  individu[counter][(int)(double)(Math.random()*popsize)];
85                 individu[counter][coba] = childCr(t1, p1, p2);
86                     //String pisah[] =
87  individu[hitung][coba].split("?<=\\G.{1}");
88                 String pisah[] = individu[counter][coba].split(" ");
89                 int j = 0;
90                 for(int i=0; i<pisah.length; i++){
91                    if(i<kroAgen.length){
92                       kroAgen[i] = Integer.parseInt(pisah[i]);
93                    }
94                    else{
95                       kroMobil[j] = Integer.parseInt(pisah[i]);
96                       j++;
97                    }
98                 }
99                 //hitung fitness
100                fitness[counter][coba]= (double) hitungFitness(kroAgen,
101 kroMobil,1);
102                System.out.print(coba+" "+individu[counter][coba]+" - ");
103                System.out.print(fitness[counter][coba]+"parent: "+p1);
104                System.out.println("");
105             }
106             //mutasi
107             String p;
108             int segmen, k=0;
109             int t1S1, t2S1;
110             t1S1 = (int)(double)(Math.random()*jumlah_agen);
111             do{
112                t2S1 = (int)(double)(Math.random()*jumlah_agen);
113             }while(t2S1==t1S1);
114             for(int coba=popsize+childCr;
115 coba<popsize+childCr+childMr; coba++){
116                p =
117 individu[counter][(int)(double)(Math.random()*popsize)];
118                if(k<(30*childMr/100)) segmen = 1;
119                else segmen = 2;
120                individu[counter][coba] = childMr(t1S1, t2S1,  p,
```

```
segmen);
                //String pisah[] =
individu[hitung][coba].split("(_?<=\\G.{1})");
            String pisah[] = individu[counter][coba].split(" ");
            int j = 0;
            for(int i = 0; i<pisah.length; i++){
              if(i<kroAgen.length){
                kroAgen[i] = Integer.parseInt(pisah[i]);
              }
            }
            //pehitungan fitness
            fitness[counter][coba] = (double) hitungFitness(kroAgen,
kroMobil, 1);
            System.out.print(coba+" "+individu[counter][coba]+" ");
            System.out.print(fitness[counter][coba]+" "+" parent "+p);
            System.out.println("");
            k++;
          }
          //seleksi
          System.out.println("Hasil Seleksi");
          for(int coba=0; coba<popsize+childCr+childMr; coba++){
            cpIndividu[counter][coba] = individu[counter][coba];
            cpFitness[counter][coba] = fitness[counter][coba];
          }
          //bubble sort
          String temp1;
          double temp;
          for(int i=1; i<=popsize+childCr+childMr; i++){
            for(int j=0; j<popsize+childCr+childMr-i; j++){
              if(cpFitness[counter][j]<cpFitness[counter][j+1]){
                temp = cpFitness[counter][j];
                temp1 = cpIndividu[counter][j];
                cpFitness[counter][j] = cpFitness[counter][j+1];
                cpIndividu[counter][j] = cpIndividu[counter][j+1];
                cpFitness[counter][j+1] = temp;
                cpIndividu[counter][j+1] = temp1;
              }
            }
          }
          //hasil seleksi
          for(int ulang = 0; ulang<popsize; ulang++){
            individu[counter+1][ulang] = cpIndividu[counter][ulang];
            fitness[counter+1][ulang] = cpFitness[counter][ulang];
            System.out.print(ulang+" "+individu[counter][ulang]+" -
");
            System.out.print(fitness[counter+1][ulang]);
            System.out.println("");
```

```
168             }
169             //hitung biaya
170                String pisah[] = individu[counter+1][0].split(" ");
171             int j = 0;
172             for(int i=0; i<pisah.length; i++){
173                if(i<kroAgen.length){
174                   kroAgen[i] = Integer.parseInt(pisah[i]);
175                }
176                else{
177                   kroMobil[j] = Integer.parseInt(pisah[i]);
178                   j++;
179                }
180             }
181             biaya = (int) hitungFitness(kroAgen, kroMobil, 3);
182             sisaMuatan = (int) hitungFitness(kroAgen, kroMobil, 2);
183
184             a[0] = counter;
185             a[1] = individu[counter+1][0];
186             a[2] = biaya;
187             a[3] = sisaMuatan;
188             a[4] = fitness[counter+1][0];
189             model.addRow(a);
190
191          }
192          //tampilan
193          txtRutePendek.setText(individu[counter][0]);
194          txtFitness.setText(Double.toString(fitness[counter][0]));
195          txtRtPendek.setText(Long.toString(biaya));
196          txtFitness1.setText(Integer.toString(sisaMuatan));
197          jTextArea1.setText(penjelasan(kroAgen, kroMobil));
198
```

Gambar 5.6 proses algoritma genetika

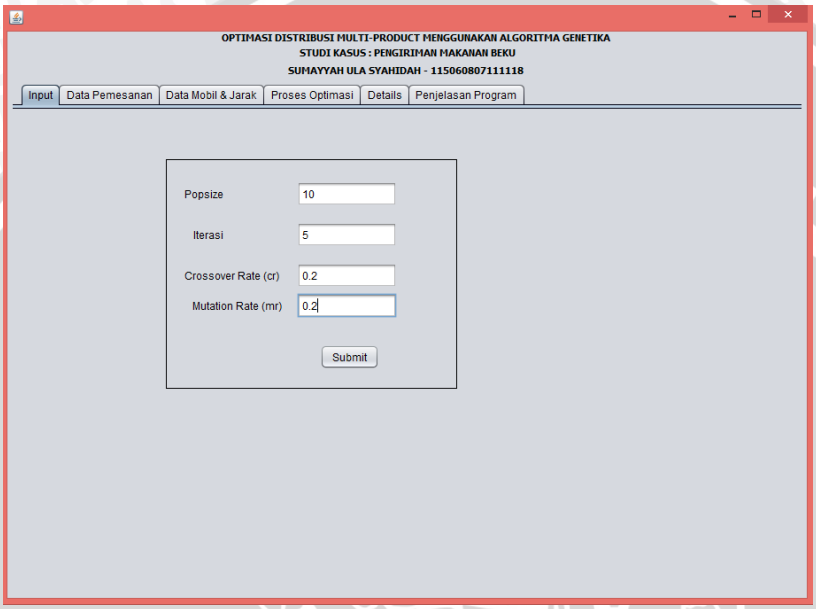
1. Baris 7 merupakan awal perhitungan waktu yang digunakan untuk menjalan program.
2. Baris 18 – 27 merupakan proses inisialisasi variabel.
3. Baris 20 – 79 merupakan proses inisialisasi awal pada algoritma genetika beserta perhitungan fitnessnya.
4. Baris 80 – 120 merupakan proses untuk menjalan proses crossover beserta perhitungan fitnessnya.
5. Baris 121 – 157 merupakan proses untuk menjalankan proses mutasi beserta perhitungan fitnessnya.
6. Baris 158 – 196 merupakan proses seleksi.

7. Baris 197– 210 merupakan proses untuk pengambilan nilai total biaya yang disimpan dalam variabel biaya.

5.2 Implementasi *User Interface*

1. Halaman Input

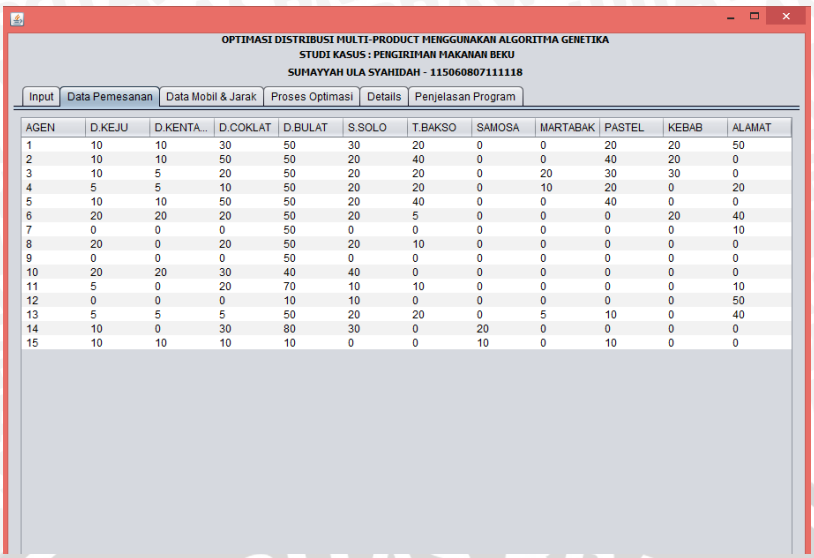
Pada halaman input ini, user akan menginputkan nilai popsize, iterasi dan crossover dan mutasi Gambar 5.7 Menunjukkan halaman input



Gambar 5.7 Halaman Input

2. Halaman data pemesanan

Halaman data pemesanan berisi pesanan agen per produk. Gambar 5.8 HalamanData Pemesanan.

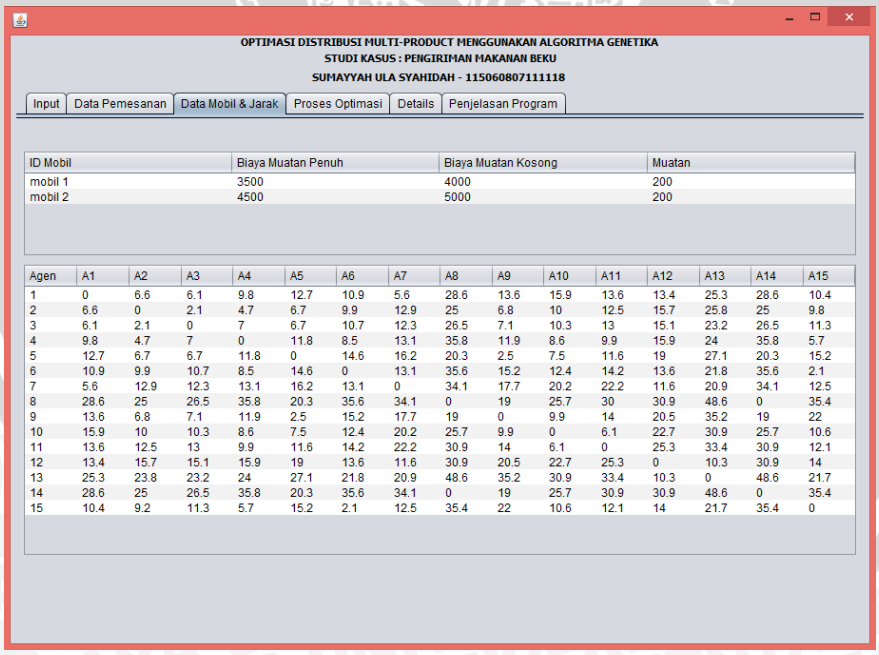


AGEN	D.KEJU	D.KENTA...	D.COKLAT	D.BULAT	S.SOLO	T.BAKSO	SAMOSA	MARTABAK	PASTEL	KEBAB	ALAMAT
1	10	10	30	50	30	20	0	0	20	20	50
2	10	10	50	50	20	40	0	0	40	20	0
3	10	5	20	50	20	20	0	20	30	30	0
4	5	5	10	50	20	20	0	10	20	0	20
5	10	10	50	50	20	40	0	0	40	0	0
6	20	20	20	50	20	5	0	0	0	20	40
7	0	0	0	50	0	0	0	0	0	0	10
8	20	0	20	50	20	10	0	0	0	0	0
9	0	0	0	50	0	0	0	0	0	0	0
10	20	20	30	40	40	0	0	0	0	0	0
11	5	0	20	70	10	10	0	0	0	0	10
12	0	0	0	10	10	0	0	0	0	0	50
13	5	5	5	50	20	20	0	5	10	0	40
14	10	0	30	80	30	0	20	0	0	0	0
15	10	10	10	10	0	0	10	0	10	0	0

Gambar 5.8 halaman data pemesanan

3. Halaman Data Jarak & Mobil

Halaman data jarak dan mobil berisi data kendaraan pengangkut seperti, biaya penuh dan kosong kendaraan serta kapasitas kendaraan. Dan berisi data jarak antar satu agen dengan agen lainnya. Pada Gambar 5.9 Halaman Data Jarak & Mobil



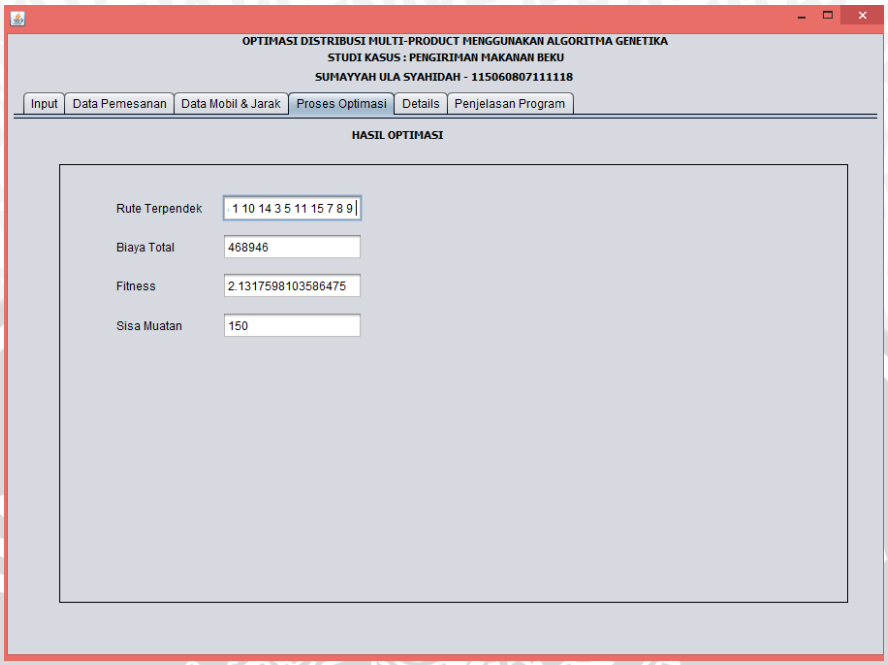
ID Mobil	Biaya Muatan Penuh	Biaya Muatan Kosong	Muatan
mobil 1	3500	4000	200
mobil 2	4500	5000	200

Agen	A1	A2	A3	A4	A5	A6	A7	A8	A9	A10	A11	A12	A13	A14	A15
1	0	6.6	6.1	9.8	12.7	10.9	5.6	28.6	13.6	15.9	13.6	13.4	25.3	28.6	10.4
2	6.6	0	2.1	4.7	6.7	9.9	12.9	25	6.8	10	12.5	15.7	25.8	25	9.8
3	6.1	2.1	0	7	6.7	10.7	12.3	26.5	7.1	10.3	13	15.1	23.2	26.5	11.3
4	9.8	4.7	7	0	11.8	8.5	13.1	35.8	11.9	8.6	9.9	15.9	24	35.8	5.7
5	12.7	6.7	6.7	11.8	0	14.6	16.2	20.3	2.5	7.5	11.6	19	27.1	20.3	15.2
6	10.9	9.9	10.7	8.5	14.6	0	13.1	35.6	15.2	12.4	14.2	13.6	21.8	35.6	2.1
7	5.6	12.9	12.3	13.1	16.2	13.1	0	34.1	17.7	20.2	22.2	11.6	20.9	34.1	12.5
8	28.6	25	26.5	35.8	20.3	35.6	34.1	0	19	25.7	30	30.9	48.6	0	35.4
9	13.6	6.8	7.1	11.9	2.5	15.2	17.7	19	0	9.9	14	20.5	35.2	19	22
10	15.9	10	10.3	8.6	7.5	12.4	20.2	25.7	9.9	0	6.1	22.7	30.9	25.7	10.6
11	13.6	12.5	13	9.9	11.6	14.2	22.2	30.9	14	6.1	0	25.3	33.4	30.9	12.1
12	13.4	15.7	15.1	15.9	19	13.6	11.6	30.9	20.5	22.7	25.3	0	10.3	30.9	14
13	25.3	23.8	23.2	24	27.1	21.8	20.9	48.6	35.2	30.9	33.4	10.3	0	48.6	21.7
14	28.6	25	26.5	35.8	20.3	35.6	34.1	0	19	25.7	30.9	30.9	48.6	0	35.4
15	10.4	9.2	11.3	5.7	15.2	2.1	12.5	35.4	22	10.6	12.1	14	21.7	35.4	0

Gambar 5.9 halaman data jarak dan mobil

4. Halaman Proses Optimasi

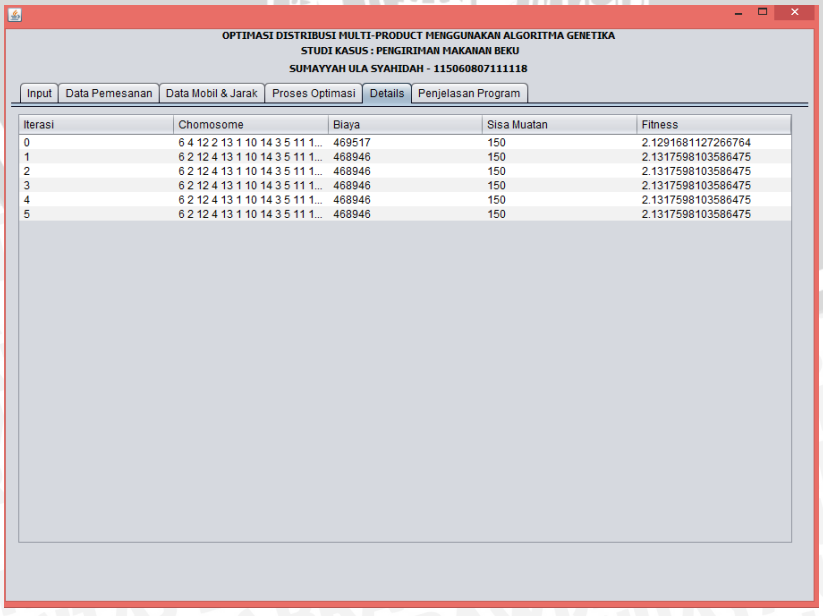
Halaman proses optimasi berisi hasil optimasi algoritma genetika. Gambar 5.10 Halaman Proses Optimasi



Gambar 5.10 Halaman Proses Optimasi

5. Halaman Detail Optimasi

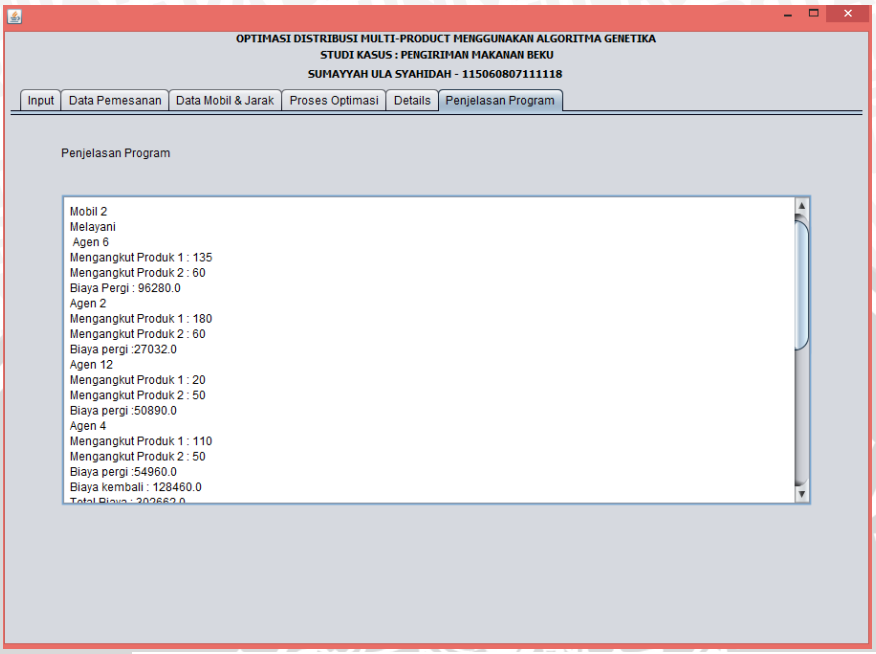
Halaman detail optimasi barisi detail optimasi kromosom dengan nilai fitness tertinggi tiap generasi. Gambar 5.11 Halaman Detail Optimasi



Gambar 5.11 Halaman Detail Optimasi

6. Halaman Penjelasan Program

Berisi penjelasan dengan generasi terbaik. Gambar 5.12 Halaman penjelasan Program



Gambar 5.12 Halaman Penjelasan Program

BAB VI

PENGUJIAN DAN ANALISIS

Bab ini akan membahas mengenai analisis dan pengujian terhadap sistem yang telah diimplementasikan pada tahap sebelumnya. Proses pengujian ini dilakukan dalam 3 kali pengujian yaitu pengujian terhadap ukuran populasi, pengujian terhadap banyaknya generasi dan pengujian terhadap *crossover rate* dan *mutation rate*.

6.1 Pengujian Banyaknya Generasi

Pengujian banyaknya generasi digunakan untuk menentukan banyaknya generasi yang terbaik untuk menghasilkan solusi terbaik dalam kasus ini. Banyaknya generasi yang diuji adalah kelipatan 500 yaitu dimulai dari 500, 1000, 1500, 2000, 2500, 3000 dan 3500. Untuk lebih detailnya mengenai parameter yang digunakan pada uji coba banyaknya generasi adalah sebagai berikut :

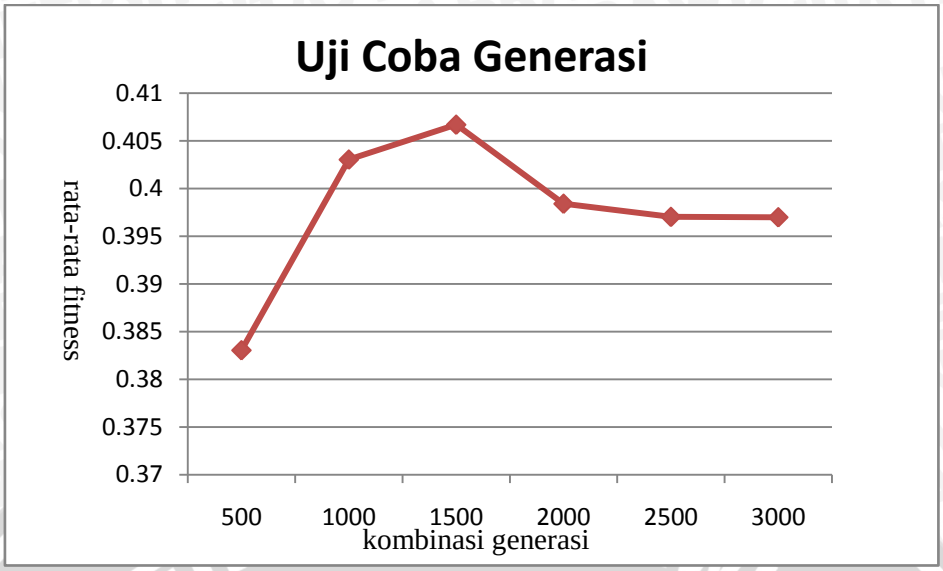
a. Ukuran populasi = 80
b. Banyaknya generasi = 500 - 3500
c. *Crossover Rate* = 0.5
d. *Mutation Rate* = 0.5

Pengujian generasi ini dilakukan sebanyak 10 kali tiap generasinya. Hasil uji generasi dapat dilihat pada Tabel 6.1

Tabel 6.1 Hasil Uji Banyak Generasi

Generasi	Nilai Fitness										Rata-rata fitnes
	Pengujian Ke-										
	1	2	3	4	5	6	7	8	9	10	
500	0.4109	0.2952	0.4028	0.3953	0.3817	0.4043	0.4099	0.3115	0.3323	0.4865	0.33048
1000	0.3953	0.3205	0.3487	0.3485	0.3233	0.4028	0.4865	0.3330	0.4099	0.4390	0.38075
1500	0.3903	0.3718	0.3903	0.4570	0.3860	0.3165	0.3948	0.3948	0.3085	0.3953	0.4067
2000	0.4115	0.4099	0.4049	0.3385	0.3718	0.4109	0.3953	0.3205	0.4099	0.41092	0.39841
2500	0.3570	0.4099	0.3079	0.4115	0.4570	0.3570	0.3330	0.4570	0.3085	0.4115	0.39703
3000	0.4099	0.4108	0.4109	0.4099	0.4865	0.3485	0.4109	0.3085	0.2950	0.3790	0.39699
3500	0.4042	0.3948	0.3948	0.4028	0.4028	0.3948	0.3338	0.4049	0.4109	0.4865	0.40303

Berdasarkan grafik hasil uji coba pada Gambar 6.2, generasi sebanyak 1500 mencapai rata – rata *fitness* terbaik. Pada generasi ke 500 – 1000 terjadi kenaikan rata – rata fitness. Namun pada generasi ke 1000 – 3500 kenaikan rata-rata fitnes sudah mulai menunjukkan angka stabil, atau perubahan untuk rata-rata fitness sudah tidak begitu besar. Semakin banyak generasi maka semakin besar waktu komputasinya dan kecil kemungkinannya bisa menghasilkan solusi yang lebih baik (Mahmudy, 2014).



Gambar 6.1 Grafik Uji Coba Generasi

6.2 Pengujian Ukuran Populasi

Pengujian ukuran populasi digunakan untuk menentukan ukuran populasi yang terbaik untuk menghasilkan solusi terbaik dalam kasus ini. Ukuran populasi yang akan diuji adalah kelipatan 20 yaitu dimulai dari 20, 40, 60, 80, 100, 120, dan 140. Untuk lebih detailnya mengenai parameter yang digunakan pada uji coba populasi adalah sebagai berikut :

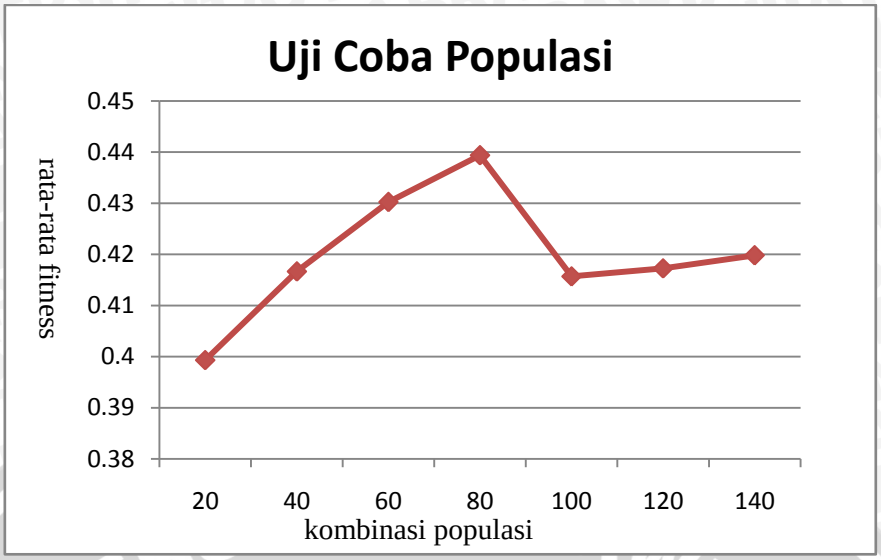
a. Ukuran populasi = 20-140
b. Banyaknya generasi = 1000
c. Crossover Rate = 0.5
d. Mutation Rate = 0.5

Pengujian terhadap populasi dilakukan sebanyak 10 kali setiap populasi. Hasil pengujian dapat dilihat dalam Tabel 6.2

Tabel 6.2 Hasil Uji Coba Populasi

Populasi	Niali Fitness										rata ftnes
	Pengujian ke-										
	1	2	3	4	5	6	7	8	9	10	
20	0.4109	0.3085	0.3953	0.3095	0.3385	0.4570	0.4028	0.3385	0.4042	0.3023	0.39931
40	0.3256	0.2783	0.2561	0.4109	0.3948	0.4865	0.4570	0.4109	0.3256	0.4109	0.41665
60	0.3025	0.4115	0.4099	0.4109	0.3570	0.4028	0.4109	0.4570	0.3718	0.4570	0.43022
80	0.3463	0.3948	0.4099	0.4865	0.4570	0.4028	0.3233	0.4109	0.3937	0.4570	0.43937
100	0.4028	0.3104	0.4028	0.3903	0.3937	0.4570	0.3953	0.4049	0.4099	0.3207	0.41569
120	0.4109	0.4470	0.4099	0.4109	0.3948	0.3903	0.4028	0.4865	0.4043	0.4115	0.41725
140	0.4099	0.4109	0.3937	0.4109	0.3251	0.3817	0.3937	0.4099	0.3953	0.3276	0.4198

Berdasarkan grafik hasil uji pada Gambar 6.2, semakin besar ukuran populasi atau popsize maka rata – rata fitness yang dihasilkan cenderung makin besar. Dari grafik tersebut bisa dilihat jika pada ukuran populasi 80, rata – rata nilai fitness mencapai nilai terbaik atau tertinggi. Pada ukuran populasi mulai dari 40 ke 140 sudah memperlihatkan jika rata – rata fitness tidak mengalami perubahan yang begitu besar. Perubahan yang tidak begitu besar ini terjadi karena anak yang dihasilkan pada saat proses reproduksi mirip dengan induknya (Mahmudy, 2014).



Gambar 6.2 Grafik Uji Coba Populasi

6.3 Pengujian Crossover dan Mutation Rate

Uji coba berdasarkan *crossover rate* (*cr*) dan *mutation rate* (*mr*) dilakukan untuk mengetahi kombinasi *cr* dan *mc* yang paling baik untuk menghasilkan fitness terbaik. Nilai *cr* dan *mr* yang digunakan dalam uji coba ini antara 0 dan 1. Pengujian pada *cr* dan *mr* ini juga menggunakan hasil uji coba yang telah dilakukan sebelumnya yaitu hasil uji coba ukuran populasi dan uji coba banyaknya generasi. Untuk lebih detailnya mengenai parameter yang digunakan dalam uji coba kombinasi *cr* dan *mr* adalah sebagai berikut :

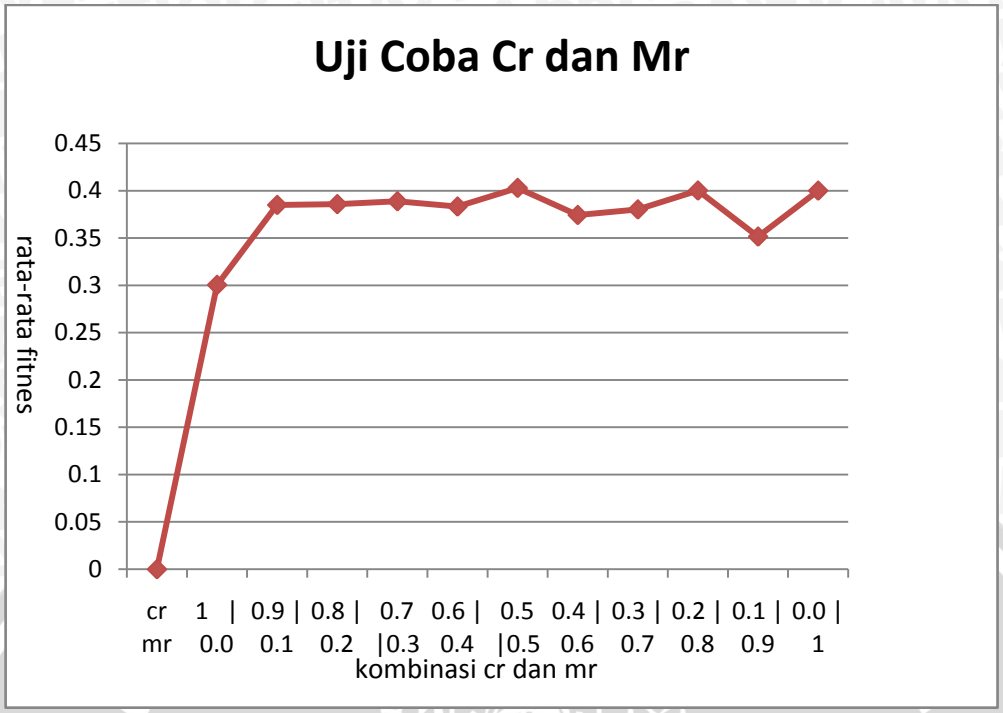
a. Ukuran populasi = 80
b. Banyaknya generasi = 1500

Pengujian ukuran populasi ini dilakukan sebanyak 10 kali. Hasil pengujian dapat dilihat pada Tabel 6.3 berikut ini :

Tabel 6.3 Hasil Uji Crossover dan Mutation

cr	mr	Nilai Fitness										Rata-rata fitness
		Pengujian Ke-										
		1	2	3	4	5	6	7	8	9	10	
1	0.0	0.3261	0.2735	0.2451	0.2525	0.3143	0.2870	0.2715	0.3818	0.3783	0.2747	0.30048
0.9	0.1	0.3085	0.4570	0.2955	0.4049	0.4109	0.4099	0.4109	0.2932	0.4470	0.4115	0.38493
0.8	0.2	0.4115	0.3937	0.3948	0.3937	0.4865	0.3075	0.3026	0.3953	0.3860	0.3860	0.38576
0.7	0.3	0.3517	0.3953	0.3903	0.4570	0.3817	0.3718	0.4109	0.4099	0.3085	0.4109	0.3888
0.6	0.4	0.3948	0.3937	0.3948	0.3948	0.3817	0.3938	0.4109	0.3463	0.4109	0.3111	0.38323
0.5	0.5	0.3860	0.4109	0.3948	0.3485	0.4049	0.3953	0.4109	0.4570	0.4109	0.4099	0.40291
0.4	0.6	0.3205	0.4099	0.3256	0.3953	0.4099	0.4109	0.3860	0.3330	0.4109	0.3323	0.3743
0.3	0.7	0.4570	0.3550	0.4028	0.3251	0.3085	0.3043	0.4109	0.4570	0.3860	0.3953	0.38019
0.2	0.8	0.3207	0.4865	0.4109	0.4109	0.4099	0.4028	0.4028	0.3718	0.4099	0.4534	0.40796
0.1	0.9	0.3207	0.3207	0.4088	0.3493	0.3233	0.2866	0.3669	0.3485	0.4115	0.3790	0.35153
0.0	1	0.4865	0.3207	0.4865	0.4109	0.4115	0.4109	0.3948	0.4049	0.4028	0.4109	0.40004

Berdasarkan grafik hasil uji coba dibawah ini, rata - rata fitness yang diperoleh sangat beragam karena tidak ada suatu ketetapan nilai cr dan mr yang digunakan untuk memperoleh solusi optimal. Permasalahan yang ingin diselesaikan mempengaruhi dalam memperoleh nilai kombinasi yang tepat (Mahmudy, 2014). Nilai cr yang terlalu rendah mengakibatkan sebuah permasalahan sulit mendapatkan solusi terbaik karena algoritma genetika kurang efektif dalam melakukan pembelajaran dari generasi sebelumnya (Mahmudy, 2014). Pada uji coba ini didapatkan kombinasi cr = 0.2 dan mr 0.8 menghasilkan rata – rata fitness terbaik.



Gambar 6.3 Grafik Uji Coba Kombinasi Cr Mr

6.4 Pengujian parameter terbaik.

Dari hasil pengujian parameter pada beberapa sub-bab diatas, didapatkan hasil terbaik dari tiap-tiap parameter yang diujikan. Dalam h

al ini dijelaskan bahwa nilai parameter mempengaruhi rata-rata hasil *fitness*. Pada data dengan ukuran besar (generasi dan populasi), algoritma genetik memperoleh solusi yang mendekati optimal akan tetapi waktu yang di butuhkan untuk mendapatkan hasil tersebut cenderung lama. Solusi yang mendekati optimal yang didapatkan setelah terjadinya konvergensi yaitu pada populasi : 80 dengan rata-rata fitness = 0.43937, generasi : 1500 dengan rata-rata fitness = 0.4067, Pc : 0.2 dan Pm : 0.8 dengan rata-rata fitness = 0.40796.

Dengan menjalankan Algoritma Genetika menggunakan parameter terbaik diatas, didapatkan nilai kromosom beserta sisa muatan dan biaya. Gambar 6.4 merupakan kromosom yang didapatkan dari menjalankan algoritma genetika menggunakan parameter terbaik

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

Gambar 6.4 Hasil Kromosom Parameter Terbaik

Gambar kromosom diatas merupakan alur distribusi terbaik yang dihasilkan saat menjalankan algoritma genetika dengan parameter terbaik. Diketahui bahwa mobil 1 akan membawa produk kategori 1 dan produk kategori 2 pada agen 5 terlebih dahulu dilanjutkan dengan agen 9 dan seterusnya. Begitu juga dengan mobil 2, akan mengantarkan produk kategori 1 dan produk kategori 2 pada agen 4 terlebih dahulu dilanjutkan dengan agen 2 dan seterusnya. Dengan alur distribusi seperti itu, didapatkan sisa muatan yang sedikit yaitu sebanyak 95 dan fitness yang dihasilkan adalah 0.36695 untuk algoritma genetika dengan parameter terbaik.

BAB VII

PENUTUP

7.1 Kesimpulan

Berdasarkan hasil uji coba parameter algoritma genetika pada permasalahan optimasi distribusi multi-produk, terdapat beberapa kesimpulan yaitu :

1. Algoritma genetika dapat menyelesaikan permasalahan ini menggunakan representasi permutasi dengan kromosom satu segmen dan panjang kromosom yang digunakan berjumlah 15.
2. Parameter terbaik dengan rata – rata nilai *fitness* tertinggi yang didapatkan dari hasil pengujian berupa jumlah populasi sebanyak 80, dengan banyak generasi sebanyak 1500, dan crossover rate dan mutation rate terbaik masing-masing sebanyak 0.2 dan 0.8. parameter terbaik tersebut menghasilkan hasil terbaik untuk optimasi distribusi multi-produk ini.
3. Untuk mengukur solusi dari permasalahan optimasi distribusi ini menggunakan perhitungan nilai *fitness* yang diperoleh dari total biaya yang dikeluarkan dalam proses distribusi dan total sisa muatan mobil pengantar satu dan mobil pengantar dua.

7.2 Saran

Penelitian ini dapat dikembangkan untuk menyelesaikan masalah optimasi distribusi barang dengan menggunakan metode crossover, mutasi dan seleksi yang berbeda sehingga mempengaruhi nilai fitness suatu individu dan hasil dari algoritma genetika. Pada penelitian selanjutnya agar menambahkan data agen, data mobil pengantar, data produk serta data jarak yang lebih banyak dan lebih bervariasi.

Daftar Pustaka

Susanti, E & Sriyanto, ES. 2006. *Optimasi Penjadwalan Pengiriman Produk Jadi Menggunakan Pendekatan Binary Integer Programming (Studi Kasus Di Pt Tiga Pilar Sejahtera Surakarta)* Program Studi Teknik Industri, Universitas Diponegoro, Semarang.

Sundarnigsih,D, Mahmudy WF & Sutrisno. 2015. *Penerapan Algoritma Genetika untuk Optimasi Vehicle Routing Problem with Time Window (VRPTW) Studi Kasus Air Minum Kemasan.* Program Teknologi Informasi Dan Ilmu Komputer Universitas Brawijaya.

Putri FB, Mahmudy WF & Ratnawati DF. 2015. *Penerapan Algoritma Genetika untuk Optimasi Vehicle Routing Problem with Time Window (VRPTW) Pada Kasus Beras Bersubsidi*. Program Teknologi Informasi Dan Ilmu Komputer Universitas Brawijaya.

Fadilah, AP . 2013. *Analisis Pengaruh Produk, Harga, Promosi Dan Saluran Distribusi Terhadap Keputusan Pembelian Konsumen (Studi Pada Outlet The Body Shop Java Mall Semarang)*. FEB-UnDip

Munawar, A & Marpaung.2008. *Pengaruh Biaya Saluran Distribusi Terhadap Tingkat Volume Penjualan Pada Pt. Winner Garments. Dosen Akademi Manajemen Kesatuan dan STIE Kesatuan*.

Mahmudy, WF. 2014. *Algoritma Evolusi*. Program Teknologi Informasi dan Ilmu Komputer, Universitas Brawijaya, Malang.

Tasmawati, Mila. 2008. *Aplikasi Konversi Regular Grammar Menjadi Ekspresi Regular Dengan Menggunakan Bahasa Pemrograman Java. Informatika*, Universitas Gunadarma.

Firdaus, Marianus. 2012. *Pengaruh Biaya Distribusi Terhadap Hasil Penjualan Produk Pt Sesional Suplies Indonesia.* Fakultas Ekonomi, Universitas Gunadarma, Depok.

Mukodim, didin. 2007. *Pengaruh Biaya Distribusi Terhadap Hasil Penjualan Produk Pt Sesional Suplies Indonesia.* Fakultas Ekonomi, Universitas Gunadarma, Depok.

Hardianti, Y & Purwanto. 2013. *Penerapan Algoritma Genetika Dalam Penyelesaian Travelling Salesman Problem With Precedence Constraints (TSPPC).* FMIPA, Universitas Negri Malang, Malang

Suwirmayanti, N. 2014. *Optimasi Pusat Cluster K-Prototype Dengan Algoritma Genetika*. Program Pascasarjana, Universitas Udayana, Denpasar.

Adhy, S & Kushartantya.2010. *Penyelesaian Masalah Job Shop Menggunakan Algoritma Genetika*. Teknik Informatika, Universitas Diponegoro, Semarang.