Optimasi *Vehicle Routing Problem With Time Window (Vrptw)* Pada Distribusi Produk Pangan Menggunakan Algoritma Genetika

SKRIPSI

Untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Komputer

Disusun oleh:
Rayandra Yala Pratama
NIM: 125150200111124

PROGRAM STUDI TEKNIK INFORMATIKA
JURUSAN TEKNIK INFORMATIKA
FAKULTAS ILMU KOMPUTER
UNIVERSITAS BRAWIJAYA
MALANG
2015

PENGESAHAN

OPTIMASI *VEHICLE ROUTING PROBLEM WITH TIME WINDOW (VRPTW)* PADA DISTRIBUSI PRODUK PANGAN MENGGUNAKAN ALGORITMA GENETIKA

SKRIPSI

Diajukan untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Komputer

Disusun Oleh :
Rayandra Yala Pratama
NIM: 125150200111124

Skripsi ini telah diuji dan dinyatakan lulus pada
12 Agustus 2016
Telah diperiksa dan disetujui oleh:

Dosen Pembimbing

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

Mengetahui
Ketua Jurusan Teknik Informatika

Tri Astoto Kurniawan, S.T, M.T, Ph.D
NIP: 19710518 200312 1 001

PERNYATAAN ORISINALITAS

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 disitasi dalam naskah ini dan disebutkan dalam 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, 18 Agustus 2016

Rayandra Yala Pratama

NIM: 125150200111124

KATA PENGANTAR

Dengan menyebut nama Allah SWT Yang Maha Pengasih dan Maha Penyayang. Puji dan syukur atas kehadirat Allah SWT karena dengan limpahan rahmat-Nya penulis dapat menyelesaikan skripsi dengan judul "Optimasi *Vehicle Routing Problem With Time Window (VRPTW)* Pada Distribusi Produk Pangan Menggunakan Algoritma Genetika". Shalawat serta salam senantiasa tercurahkan kepada junjungan kita Nabi besar Muhammad SAW beserta keluarga dan para sahabatnya dan semua doa yang mengalir untuk kebaikan-kebaikan dalam muka bumi ini hingga akhir zaman. Skripsi ini disusun untuk memenuhi salah satu syarat memperoleh gelar Sarjana Komputer di Fakultas Ilmu Komputer Universitas Brawijaya Malang (FILKOM UB).

Dalam kesempatan ini penulis juga ingin menyampaikan terima kasih kepada semua pihak yang mendukung dalam proses penyelesaian skripsi ini, yakni :

1. Wayan Firdaus Mahmudy, S.Si, M.T, Ph.D, selaku Dosen Pembimbing tunggal yang telah membimbing dan memberikan banyak saran kepada penulis sehingga dapat menyelesaikan skripsi ini.
2. Tri Astoto Kurniawan, S.T, M.T, Ph.D, selaku Ketua Jurusan Teknik Informatika Fakultas Ilmu Komputer Universitas Brawijaya.
3. Seluruh Dosen FILKOM UB yang telah memberikan ilmunya kepada penulis selama masa perkuliahan berlangsung.
4. Seluruh Civitas Akademika FILKOM UB yang telah memberikan dukungan dan bantuan selama menempuh studi di Fakultas Ilmu Komputer, Jurusan Teknik Informatika Universitas Brawijaya dan selama penyelesaian skripsi ini.
5. Kedua Orang Tua Penulis serta keluarga besar atas segala do'a, nasihat, dukungan baik moril maupun materiil dalam melancarkan skripsi ini.
6. Sahabat serta teman-teman penulis, Imaratul Mufida, 11 Pria Tampan, serta teman-teman angkatan 2012-2014 Unitas Marching Band, bantuan, dukungan, motivasi dan berbagi informasi demi kelancaran skripsi.

Dengan segala kerendahan hati, penulis menyadari bahwa skripsi ini masih memiliki banyak kekurangan. Oleh karena itu kritik dan saran yang bersifat konstruktif sangat dibutuhkan sebagai pedoman untuk menyempurnakan skripsi ini agar lebih baik. Penulis berharap semoga skripsi ini dapat bermanfaat bagi diri sendiri maupun bagi semua pihak.

Malang, 18 Agustus 2016

Penulis

Rayandra.pratama@gmail.com

ABSTRAK

Dalam industri produk pangan, proses pengiriman atau proses distribusi merupakan proses yang penting karena produk pangan tidak dapat bertahan lama saat melakukan pengiriman sehingga membutuhkan waktu dan rute tercepat. Jauh tidaknya rute yang diambil menentukan besar kecilnya pengeluaran untuk proses distribusi karena semakin jauh rute yang ditempuh maka semakin besar biaya yang harus dikeluarkan. Sebaliknya, jika rute yang diambil adalah rute terpendek maka proses pengiriman akan menjadi efisien dan dapat menekan biaya pengiriman. Menentukan rute distribusi menjadi semakin sulit jika terdapat banyak pelanggan yang harus dikunjungi dan setiap pelanggan mempunyai kebijakan waktu tertentu dalam menerima pengiriman. Permasalahan ini dikenal dengan *Vehicle Routing Problem with Time Windows* (VRPTW). Permasalahan VRPTW dapat diselesaikan menggunakan algoritma genetika karena algoritma genetika menghasilkan beberapa solusi. Dalam memecahkan solusi, algoritma genetika membuat kromosom yang terdiri dari nomor-nomor yang merepresentasikan pelanggan yang harus dikunjungi. Kromosom ini yang selanjutnya digunakan dalam proses perhitungan bersama dengan operator genetika lainnya seperti ukuran populasi, banyaknya generasi, *crossover* dan *mutation rate*. Setelah mendapatkan hasil maka langkah selanjutnya adalah melakukan pengujian. Pengujian ini berfungsi untuk mencari parameter yang menghasilkan nilai *fitness* terbaik. Hasil dari pengujian didapatkan bahwa ukuran populasi terbaik sebesar 300 dengan generasi sebanyak 3000 serta kombinasi *crossover* dan *mutation rate* masing-masing 0.4 dan 0.6. Pengujian ini juga didapatkan seleksi terbaik yaitu seleksi elitis. Setelah mendapatkan parameter-parameter terbaik, didapatkan hasil nilai *fitness* dari parameter-parameter terbaik sebesar 0.000788.

Kata Kunci: Algoritma Genetika, VRPTW, distribusi, *Fitness*, *Time Window*, produk pangan

ABSTRACT

In the food products industry, the distribution process is very important because the food product can expired during distribution so fastest route is needed. Distribution expenses are depend on the route it takes since the further distance the greater the costs. On the other hand, if the route taken is the shortest one, the delivery process will be efficient and can reduce the cost of distribution. Determining the distribution becomes more difficult if there are many customers must visited and every customer has a time window for serving the delivery. This problem is known as the Vehicle Routing Problem with Time Windows (VRPTW). VRPTW problems can be solved using genetic algorithms because genetic algorithms generate multiple solutions at once. Genetic algorithms make chromosomes from serial numbers that represent the customer to visit. These chromosomes are used in the calculation process together with other genetic operators such as population size, number of generations, crossover and mutation rate. After getting the results, the next step is to test the program. This test is used to find the parameters that produce the best fitness value. The results show that the best population size is 300, 3,000 generations and the combination of crossover and mutation rate is 0.4 and 0.6. The result also showing that the best selection method is elitist selection. The fitness value from the best parameters is 0.000788.

Keywords: Genetic Algorithm, VRPTW, distribution, Fitness, Time Window, food products

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

DAFTAR SOURCE CODE

BAB 1 PENDAHULUAN

1.1 Latar belakang

Distribusi merupakan kegiatan yang dilakukan untuk memenuhi permintaan pelanggan. Pelanggan bisa berupa toko milik perseorangan maupun cabang dari pabrik yang melakukan proses distribusi. Jika jumlah pelanggan yang harus dipenuhi sedikit, maka proses distribusi akan menjadi mudah karena tempat yang harus dikunjungi sedikit dan mudah untuk menentukan rute tercepat. Namun, akan berbeda jika jumlah pelanggan yang harus dikunjungi jauh lebih banyak. Pencarian rute tercepat akan menjadi lama dan sulit jika ditambahkan waktu untuk menerima barang pada masing-masing pelanggan. Permasalahan ini dapat disebut juga *vehicle routing problem with time windows* (VRPTW). Permasalahan VRPTW ini berbeda dengan *Traveling Salesman* Product yang hanya mencari solusi berdasarkan rute saja tanpa ada batasan waktu atau *time window* (Hlaing dan Khine, 2011).

Terdapat banyak metode yang dapat digunakan untuk menyelesaikan permasalahan VRPTW. *Improved Simulated Annealing* digunakan untuk mengatasi masalah VRPTW pada *dataset* permasalahan *Solomon Benchmark Problems*. Penelitian menyimpulkan bahwa *Improved Simulated Annealing* dapat digunakan untuk menyelesaikan permasalahan VRPTW dan menghasilkan solusi dalam waktu rata-rata 82.29 detik (Mahmudy, 2014). *Distributed evolutionary algorithms* digunakan untuk menyelesaikan *vehicle routing problem* yang memberikan hasil bahwa dengan menambahkan algoritma distribusi kepada *evolutionary algorithms* mempunyai hasil yang lebih baik meskipun dengan jumlah perhitungan yang sama (Puljic dan Manger, 2012). Pada permasalahan *Vehicle Routing Problem with Soft Time Windows* juga dapat diselesaikan dengan *Improved Genetic Algorithms* yang menunjukkan hasil yang lebih baik daripada metode pola *Group-based,* CW dan tipe persilangan O-X (Li dkk, 2015).

Optimized Crossover Genetic Algorithm juga dapat digunakan untuk menyelesaikan permasalahan *vehicle routing problem with time windows* dengan hasil kualitas yang cukup baik pada total jarak terjauh (Nazif dan Lee, 2010). Permasalahan yang sama juga ditemukan pada penelitian sebelumnya dengan objek distribusi minuman bersoda. Penelitian sebelumnya mengungkapkan bahwa masalah VRPTW dapat diselesaikan dengan menggunakan *evolution strategies* yang disimpulkan dari tidak adanya waktu *tardy* dan semua pelanggan terlayani (Harun, Mahmudy dan Yudistira, 2014).

Penelitian yang sama juga dilakukan pada permasalahan pencarian rute optimal untuk distribusi air minum dapat menerapkan algoritma genetika sebagai solusi dengan kendala *time window*. Penelitian tersebut mengungkapkan semakin banyak generasi belum tentu akan mendapatkan hasil yang optimal (Sundarningsih, Mahmudy dan Sutrisno, 2015). Algoritma genetika juga dapat menyelesaikan VRPTW untuk distribusi barang dengan objek mie instan. Pada penelitian tersebut disimpulkan bahwa ukuran generasi paling optimal yaitu

berada pada generasi 2000 dan populasi optimal berada pada 140 populasi (Saputri, Mahmudy dan Ratnawati, 2015).

VRPTW merupakan permasalahan yang sepele jika jumlah pelanggan yang harus dilayani sedikit, akan tetapi menjadi permasalahan yang kompleks jika jumlah pelanggan yang harus dilayani jauh lebih banyak. Sedangkan, produk pangan merupakan barang yang dapat kadaluarsa atau membusuk jika proses pengiriman/distribusi membutuhkan waktu yang sangat lama. Pemilihan rute yang tercepat akan sangat membantu dalam proses distribusi karena akan menghemat waktu dan juga pengeluaran. Dengan pertimbangan tersebut, maka permasalahan akan diselesaikan menggunakan algoritma genetika karena algoritma genetika merupakan algoritma yang fleksibel. Algoritma ini membutuhkan input berupa fungsi tujuan dan menghasilkan lebih dari satu solusi.

1.2 Rumusan masalah

1. Bagaimana menerapkan algoritma genetika dengan *Vehicle Routing Problem With Time Windows* (VRPTW) pada pencarian rute terpendek?
2. Bagaimana nilai *fitness* yang dihasilkan dari seleksi Elitis dan *Binary Tournament?*
3. Bagaimana pengaruh parameter genetika terhadap nilai *fitness* yang didapatkan?
4. Bagaimana pengaruh *crossover rate* dan *mutation rate* terhadap nilai *fitness*?
5. Bagaimana hasil nilai *fitness* jika menggunakan parameter terbaik?

1.3 Tujuan

Tujuan umum:

Menyelesaikan permasalahan distribusi yang ada dalam *dataset Solomon's VRPTW Benchmark*.

Tujuan khusus:

1. Menyelesaikan permasalahan VRPTW menggunakan algoritma genetika.
2. Mengetahui hasil fitness untuk setiap tahapan pengujian.
3. Mengetahui hasil fitness saat diuji menggunakan seleksi Elitis dan *Binary Tournament*.

1.4 Manfaat

1.4.1 Bagi Penyusun

- Dapat mengetahui penerapan algoritma genetika untuk pencarian rute distribusi dalam permasalahan *vehicle routing problem with time windows*.

1.4.2 Bagi Mahasiswa

- Sebagai referensi pembelajaran bagi mahasiswa dengan ilmu yang sesuai dengan bidangnya
- Sebagai acuan untuk melaksanakan penelitian tugas akhir dengan tema yang sama

1.5 Batasan Masalah

Batasan masalah dalam penelitian ini adalah sebagai berikut :

1. *Dataset* diambil dari website VRPTW Benchmark Problems. http://web.cba.neu.edu/~msolomon/problems.htm.
2. *Dataset* yang dipakai dalam penelitian ini adalah tipe R1 dan nomor R101.
3. Jumlah kendaraan yang digunakan sejumlah 25 dan mempunyai kapasitas masing-masing 200.
4. Kolom *dataset* yang digunakan adalah Cust No., XCoord, YCoord, Demand, Ready Time dan Service Time.
5. Kendaraan diberangkatkan secara urut dan bergantian dengan selang waktu 5 satuan.
6. Kecepatan kendaraan konstan yaitu 2 satuan.
7. Kendaraan yang tiba sebelum waktu buka (Service Time) dikenakan pinalti waktu.
8. Kendaraan yang tidak memiliki cukup suplai saat melayani pelanggan dikenakan pinalti kapasitas.

1.6 Sistematika Pembahasan

Sistematika penyusunan laporan dibentuk sebagai uraian atau gambaran dari laporan penelitian secara umum yang meliputi beberapa bab diantaranya yaitu :

BAB I PENDAHULUAN

Menguraikan latar belakang, rumusan masalah, tujuan, manfaat, batasan masalah dari Optimasi VRPTW pada Distribusi Minuman menggunakan Algoritma Genetika.

BAB II KAJIAN PUSTAKA DAN DASAR TEORI

Menguraikan studi literatur atau kajian pustaka yang terkait dengan penelitian tentang Optimasi VRPTW pada Distribusi Minuman menggunakan Algoritma Genetika.

BAB II METODOLOGI PENELITIAN

Menguraikan tentang metode yang akan dilakukan pada penelitian tentang dari Optimasi VRPTW pada Distribusi Minuman menggunakan Algoritma Genetika.

BAB IV PERANCANGAN

Menguraikan gambaran dari perancangan sistem yang sedang diteliti sebagai acuan implementasi pada bab selanjutnya.

BAB V IMPLEMENTASI

Melakukan implementasi dari perancangan yang ada dalam bab sebelumnya dan selanjutnya membuat laporan.

BAB VI PENGUJIAN DAN ANALISIS

Menguraikan hasil pengujian berdasarkan perancangan dan implementasi dari bab sebelumnya dan kemudian hasil pengujian tersebut di analisis apakah sudah tepat sasaran atau sudah sesuai dengan rumusan dan lingkup masalah.

BAB VII PENUTUP

Menguraikan kesimpulan dan saran berdasarkan metode dan hasil penelitian yang telah dilakukan dari bab-bab sebelumnya.

BAB 2 KAJIAN PUSTAKA DAN DASAR TEORI

Bab ini membahas tentang kajian pustaka dan dasar teori terkait dengan penelitian optimasi *Vehicle Routing Problem With Time Windows* (VRPTW) menggunakan Algoritma Genetika.

2.1 Kajian Pustaka

Pada tahun 2015, Saputri, Mahmudy dan Ratnawati melakukan penelitian tentang optimasi *vehicle routing problem with time windows* (VRPTW) menggunakan algoritma genetika untuk distribusi barang. Penelitian ini dilakukan dengan tujuan untuk menerapkan VRPTW menggunakan Algoritma Genetika pada pencarian rute distribusi Mie Instan sehingga didapatkan waktu tempuh minimum. Permasalahan ini menggunakan representasi kromosom berjenis permutasi dengan panjang kromosom 30 yang masing-masing mewakili nomor toko. Penelitian tersebut menggunakan metode seleksi *Elitism* dan *Roulette Wheel*, sedangkan untuk crossover menggunakan crossover PMX. Penelitian ini menunjukkan bahwa ukuran generasi paling optimal adalah 2000 dengan nilai fitness 0.076501 dan jumlah populasi optimal adalah 140 dengan nilai fitness 0.079029. Sedangkan, kombinasi terbaik antara crossover rate dan mutation rate adalah 0.3 dan 0.7 dengan rata-rata nilai fitness sebesar 0.0767.

Pada tahun 2014, Harun, Mahmudy dan Yudhistira melakukan penelitian untuk menyelesaikan *vehicle routing problem with time windows* (VRPTW) pada distribusi minuman soda XYZ dengan menggunakan *evolution strategies*. Tujuan dari penelitian ini adalah untuk mengimplementasikan *evolution strategis* pada permasalahan VRPTW pada distribusi minuman soda XYZ sehingga pengeluaran distribusi menjadi minimum. Penelitian ini menemukan bahwa waktu komputasi dapat diminimalisir sampai 30 detik dengan menggunakan teknik mutasi *exchange mutation*, jumlah generasi 2000 dan parameter $\mu = 80$, $\lambda = 2\mu$ dan $\alpha = 0.99$. Peneliti menarik kesimpulan bahwa permasalahan VRPTW pada distribusi minuman soda XYZ dapat diselesaikan menggunakan *evolution strategies* dengan nilai fitness yang dihasilkan teknik *exchange mutation* lebih baik jika dibandingkan dengan teknik mutasi lainnya.

Sundarningsih, Mahmudy dan Sutrisno (2015) melakukan penelitian yang menerapkan algoritma genetika untuk memecahkan permasalahan VRPTW pada studi kasus air minum kemasan. Penulis mengemukakan bahwa algoritma genetika dapat menyelesaikan permasalahan optimasi yang kompleks seperti mencari rute paling optimum dengan memperhatikan waktu tempuh, jarak tempuh, waktu toko beroperasi dan lain-lain. Pada proses seleksi, peneliti ini menggunakan seleksi Elitis dan *Roulette Wheel* dengan nilai *fitness* masing-masing 0.08294 dan 0.0583 yang menjadikan seleksi Elitis lebih baik daripada seleksi *Roulette Wheel*. Peneliti berpendapat bahwa ukuran populasi optimal adalah sebanyak 100 populasi dengan nilai *fitness* 0.10134215. Pada ukuran populasi 20 sampai 100 terdapat perubahan nilai *fitness* yang cukup signifikan

sedangkan ukuran populasi 120 sampai 200, nilai *fitness* tidak menunjukkan perubahan yang signifikan dan cenderung berupa garis lurus.

Perbedaan dan persamaan antara penelitian sebelumnya dengan penelitian yang diajukan berdasarkan kajian pustaka terdapat dalam Tabel 2.1

Tabel 2.1 Perbedaan dan Persamaan Penelitian Sebelumnya dengan Penelitian yang diajukan

Judul	Objek	Metode	Keluaran
	Masukan dan Parameter	**Proses**	**Hasil Penelitian**
Optimasi Vehicle Routing Problem With Time Windows (VRPTW) Menggunakan Algoritma Genetika Pada Distribusi Barang (Paper 1)	Data toko yang harus dikunjungi	Algoritma Genetika	- Nilai *fitness*
	- Nomor toko yang akan digunakan dalam membentuk kromosom - Kecepatan kendaraan 20 km/jam		- Nilai *fitness* dengan ukuran generasi optimal (2000) adalah 0.076501 - Nilai *fitness* pada populasi optimal (140) adalah 0.079029
Implementasi Evolution Strategies untuk Penyelesaian Vehicle Routing Problem With Time Windows pada Distribusi Minuman Soda XYZ (Paper 2)	Data dummy yang disesuaikan dengan data sebenarnya	*Evolution Strategies* dan teknik *Exchange Mutation*	- Parameter, teknik mutasi dan jumlah generasi yang optimal - Waktu komputasi
	- Jumlah pelanggan - Jumlah permintaan - Waktu pelayanan - Jarak antar pelanggan		waktu komputasi dapat diminimalisir sampai 30 detik dengan menggunakan teknik mutasi *exchange mutation*, jumlah generasi 2000 dan parameter $\mu = 80$, $\lambda = 2\mu$ dan $\alpha = 0.99$
Penerapan Algoritma Genetika untuk Optimasi Vehicle Routing Problem	Data sampel toko untuk permintaan dalam satu hari	Algoritma Genetika	- Nilai *fitness*
	- Data jarak tempuh antar pelanggan air minum yang	-	-

Judul	Objek	Metode	Keluaran
	Masukan dan Parameter	Proses	Hasil Penelitian
with Time Window (VRPTW) : Studi Kasus Air Minum Kemasan (Paper 3)	ditentukan secara random - Data pelanggan berupa waktu buka, waktu tutup dan jumlah permintaan yang diambil berdasarkan sampel pada kasus *real* - Data waktu layanan berdasarkan *survey* ke beberapa toko - Jumlah kapasitas angkutan air minum		
Optimasi Vehicle Routing Problem With Time Windows (VRPTW) Menggunakan Algoritma Genetika (Usulan)	- *Dataset* yang diperoleh dari web	- Algoritma Genetika	- Nilai *fitness*
	- Koordinat (X dan Y) - Jumlah permintaan - Waktu buka - Waktu layanan		- Didapatkan nilai *fitness* yang akan dijadikan solusi

2.2 Vehicle Routing Problem With Time Windows

Kegiatan distribusi merupakan salah satu faktor penting dalam perusahaan karena terdapat banyak toko yang harus dikunjungi dalam mendistribusikan barangnya. Banyak sekali toko yang harus dikunjungi membuat perusahaan harus memilih rute distribusi yang paling tepat sehingga biaya yang dikeluarkan dalam kegiatan distribusi semakin minimum. Jika perusahaan salah memilih rute yang akan ditempuh, maka perusahaan tersebut harus merugi karena biaya yang dikeluarkan semakin besar. Permasalahan ini dikenal sebagai *Vehicle Routing Problem* (VRP). VRP merupakan permasalahan yang didesain

untuk mencari rute kendaraan yang memiliki biaya paling kecil dengan depot sebagai tempat awal mulai dan selesai melakukan distribusi. Kendaraan pengangkut barang hanya membawa barang sesuai dengan kapasitas kendaraan dan hanya mengunjungi toko sekali selama proses distribusi. Permasalahan VRP yang dipengaruhi oleh jam buka toko atau wakt-waktu tertentu disebut *Vehicle Routing Problem With Time Windows* (VRPTW). VRPTW membuat kendaraan pengangkut barang mengunjungi toko pada jam yang telah ditentukan oleh toko tersebut atau dengan kata lain kendaraan harus berkunjung pada waktu toko tersebut buka.

2.3 Algoritma Genetika

Algoritma genetika merupakan tipe algoritma evolusi yang banyak diaplikasikan dalam berbagai bidang seperti biologi, fisika, sosial dan lain-lain. Hal ini terjadi karena kemampuan dari algoritma genetika yang dapat menyelesaikan berbagai permasalahan yang kompleks dan berkembang dengan pesat seiring perkembangan teknologi (Mahmudy, 2013).

Tahap awal dari algoritma genetika yaitu membangkitkan populasi awal secara acak. Populasi awal ini merupakan calon solusi yang terdiri dari kromosom-kromosom yang akan diregenerasi tiap iterasi. Pada setiap iterasi, kromosom tersbut dihitung nilai *fitness*nya. Setelah iterasi, dihasilkan kromosom baru yang disebut *offspring*. *Offspring* dihasilkan dari penggabungan dua kromosom yang telah mengalami mutasi sebelumnya. Penggabungan dua kromosom ini menggunakan operator *crossover* atau penyilangan. Proses iterasi akan dilakukan sampai ditemukan solusi yang paling optimal atau sampai batas iterasi yang ditentukan sistem.

Proses dalam algoritma genetika adalah sebagai berikut :

1. [Inisialisasi] Membuat individu-individu acak yang terdiri dari kromosom tertentu. Kromosom merupakan solusi dari permasalahan
2. [Iterasi/*looping*] Melakukan proses tertentu selama belum menemui kondisi berhenti
    a. [Reproduksi] Proses untuk menghasilkan keturunan dari individu-individu dalam populasi sebelumnya
    b. [Evaluasi] Menghitung nilai *fitness* pada setiap kromosom. Kromosom dengan nilai *fitness* yang tinggi akan dipilih untuk menjadi calon solusi
    c. [Seleksi] memilih individu dari kumpulan populasi dan *offspring*. Semakin tinggi nilai *fitness* dari suatu kromosom, semakin besar peluang kromosom tersebut untuk dipilih.

2.3.1 Representasi *Chromosome*

Dalam algoritma genetika, masalah riil harus diterjemahkan kedalam terminology biologi atau yang disebut juga representasi *chromosome*. Pengkodean (*encoding*) merupakan cara yang dapat digunakan untuk

merepresentasikan *chromosome*. Jenis pengkodean bergantung dengan permasalahan yang terjadi.

Menurut sturktur dari pengkodean dibagi menjadi dua bagian yaitu pengkodean 1 dimensi dan 2 dimensi. Pengkodean 1 dimensi terdiri dari pengkodean biner, oktal, heksadesimal, permutasi dan nilai sedangkan pengkodean 2 dimensi adalah pengkodean *tree* (Kumar, 2013).

2.3.1.1 Pengkodean Biner

Pengkodean biner merupakan bentuk yang paling umum yang setiap kromosom direpresentasikan dengan bilangan biner 0 dan 1. Pengkodean ini dapat digunakan untuk menyelesaikan permasalahan *knapsack* dengan 0 menandakan barang tersebut tidak ada dan sebaliknya. Contoh pengkodean biner dapat dilihat dalam Tabel 2.2 berikut :

Tabel 2.2 Pengkodean Biner

Kromosom1	001101011101
Kromosom2	111001011001

2.3.1.2 Pengkodean Oktal

Pengkodean oktal merupakan pengkodean yang merepresentasikan kromosom dengan bilangan oktal 0-7. Contoh pengkodean oktal terdapat dalam Tabel 2.3 berikut :

Tabel 2.3 Pengkodean Oktal

Kromosom1	726141046512
Kromosom2	047112630421

2.3.1.3 Pengkodean Heksadesimal

Pengkodean heksadesimal menggunakan bilangan heksadesimal (0-9, A-F) sebagai representasi kromosomnya. Contoh pengkodean ini terdapat dalam Tabel 2.4 :

Tabel 2.4 Pengkodean Heksadesimal

Kromosom1	3B2104C
Kromosom2	AA523B9

2.3.1.4 Pengkodean Permutasi

Pengkodean permutasi dapat digunakan dalam menyelesaikan permasalahan dalam hal pengurutan seperti *travelling salesman problem* (TSP). Dalam pengkodean permutasi, kromosom direpresentasikan dengan nomor urut dari tujuan-tujuan yang akan dilalui. Tabel 2.5 merupakan contoh pengkodean permutasi.

Tabel 2.5 Pengkodean Permutasi

Kromosom1	5 7 2 1 4 9 3 6 10 8
Kromosom2	10 5 9 2 8 7 1 6 3 4

Terdapat beberapa operator *crossover* yang dapat digunakan pada pengkodean permutasi antara lain *partially mapped crossover* (PMX), *cycle crossover* (OCX) dan *order crossover* (OX).

2.3.1.5 Pengkodean Nilai (*Value Encoding*)

Dalam pengkodean ini, masing-masing kromosom direpresentasikan dengan beberapa tipe nilai seperti integer, bilangan riil, karakter abjad, maupun berupa objek. Jika menggunakan bilangan tipe integer, maka dapat mengaplikasikan *crossover* yang sama dengan pengkodean biner. Pengkodean ini dapat digunakan pada permasalahan jaringan neural yang menggunakan pembobotan untuk input. Contoh pengkodean nilai dapat dilihat dalam Tabel 2.6.

Tabel 2.6 Pengkodean Nilai (*Value Encoding*)

Kromosom1	AHDGBCDCHDFKG
Kromosom2	1.34, 2.12, 0.11, 4.21

2.3.1.6 Pengkodean *Tree*

Pengkodean *tree* digunakan dalam pemrograman yang berhunbungan dengan evolusi atau genetika. Kromosom direpresentasikan dengan gambar pohon.

2.3.2 Mutasi

Mutasi merupakan operator genetika setelah *crossover*. Mutasi mengubah gen dari satu kromosom secara acak. Cara sederhana untuk melakukan mutasi adalah mengganti satu gen dari suatu kromosom. Mutasi berguna untuk menjaga keberagaman suatu populasi sehingga menghasilkan solusi yang lebih baik.

Terdapat beberapa tipe mutasi yang ada dalam algoritma genetika. Dalam permasalahan distribusi, representasi bilangan *integer* merupakan representasi terbaik untuk diterapkan (Hasan dan Saleh, 2011). Representasi *biner* tidak dapat digunakan dalam permasalahan distribusi karena terdapat bilangan 0 yang berarti salah atau tidak dikunjungi. Berikut merupakan tipe mutasi berdasarkan representasi bilangan *integer*:

2.3.2.1 *Reciprocal*

Tipe mutasi ini bekerja dengan cara menukarkan dua gen yang telah dipilih secara acak.

2.3.2.2 *Inversion*

Mutasi ini menentukan dua titik potong secara acak, lalu membalik urutan gen yang berada di antara dua titik potong tersebut.

2.3.2.3 *Insertion*

Mutasi ini memilih gen secara acak lalu menempatkan gen tersebut di tempat yang dipilih secara acak.

2.3.2.4 *Boundary*

Tipe ini memilih gen secara acak lalu menggantinya dengan batas atas atau batas bawah. Batas atas dan batas bawah dipilih secara acak.

2.3.2.5 *Displacement*

Mutasi ini menempatkan sekumpulan gen yang berurutan ke tempat yang ditentukan secara acak.

2.3.2.6 *Uniform*

Mutasi ini menentukan gen untuk digantikan dengan nilai acak yang berada di antara batas yang telah ditentukan *user*. Proses menentukan gen dilakukan dengan mencari gen yang sama dengan nilai acak tersebut.

2.3.3 Seleksi

Proses seleksi merupakan proses yang penting dalam algoritma genetika dimana proses ini menentukan kromosom yang akan digunakan dalam generasi selanjutnya. Terdapat beberapa metode seleksi yang dapat digunakan dalam algoritma genetika, antara lain:

2.3.3.1 Seleksi *Binary Tournament*

Seleksi ini bekerja seperti sistem turnamen yang mencari satu pemenang dari dua individu. Pada algoritma genetika, seleksi ini mencari dua individu secara acak dari populasi. Proses seleksi sebagai berikut:

1. Menentukan nilai n sejumlah populasi awal.
2. Untuk setiap n, melakukan:
   a. Memilih dua individu secara acak
   b. Memilih individu dengan nilai *fitness* yang paling tinggi diantara dua individu yang dipilih.

2.3.3.2 Seleksi Elitis

Seleksi elitis merupakan metode untuk mengambil individu dengan nilai *fitness* tertinggi yang selanjutnya akan digunakan menjadi generasi baru. Metode ini yaitu merangkai kromosom dengan urutan dari nilai *fitness* tertinggi hingga terendah lalu memilih masing-masing n-kromosom teratas dari rangkaian tersebut, dimana n merupakan ukuran populasi awal.

Metode ini berbeda dengan metode *binary tournament* yang memilih individu secara acak. Metode ini mempertahankan individu dengan nilai *fitness* terbaik untuk lanjut ke generasi berikutnya.

2.3.3.3 Seleksi *Roulette Wheel*

Metode seleksi *roulette wheel* merupakan metode seleksi yang mengadopsi permainan *roulette wheel* dengan menempatkan kromosom pada bagian-bagian dari roda sesuai dengan nilai probabilitas. Nilai probabilitas didapatkan dari perhitungan nilai *fitness*. Semakin besar nilai probabilitas suatu kromosom maka bagian yang didapatkan dalam roda tersebut akan semakin besar. Proses seleksi *roulette wheel* adalah sebagai berikut:

1. Menghitung nilai *fitness* (fv) masing-masing individu.
2. Menghitung total *fitness* (Sf) dari seluruh individu.
3. Menghitung rata-rata *fitness* (Af) dari semua individu.
4. Menghitung *expected fitness* (Ef).
5. Menghitung probabilitas masing-masing individu berdasarkan nilai dari (Ef).
6. Menghitung nilai probabilitas kumulatif (*Probcum*).
7. Membangkitkan angka random (G) 0-*Probcum*.
8. Menyeleksi individu dengan jumlah nilai Ef individu sebelumnya dan juga individu tersebut (*probcum*) lebih besar dari nilai (G).
9. Kembali ke langkah ke-6 untuk melakukan perulangan sebanyak *popsize*.

2.3.3.4 Seleksi Rangking / *Rank Selection*

Metode seleksi ini bekerja berdasarkan perangkingan nilai *fitness*. Setelah didapatkan rangking setiap kromosom, maka melakukan perhitungan nilai *fitness* menggungakan rumus baru. Setelah itu, menyeleksi kromosom menggunakan metode seleksi *roulette wheel*. Berikut merupakan proses *rank selection*:

1. Mengurutkan kromosom berdasarkan nilai *fitness* dari tertinggi hinggan terendah.
2. Memberi rangking pada masing-masing kromosom.
3. Menghitung nilai *fitness* baru menggunakan rumus dibawah ini:

$$F = \max - (max - min) * \frac{rank-1}{Npop-1} \quad (1)$$

Dimana 1<max<=2 & min = 2-max

2.3.4 *Fitness*

Baik tidaknya suatu individu dapat diukur melalui fungsi *fitness*. Semakin besar nilai yang dihasilkan dari fungsi *fitness* maka semakin baik individu tersebut. Nilai *fitness* digunakan dalam menentukan seberapa baik individu tersebut untuk digunakan dalam generasi selanjutnya.

Algoritma genetika merupakan algoritma yang menentukan solusi permasalahan berdasarkan nilai *fitness* yang dihasilkan. Solusi yang optimal adalah solusi dengan nilai *fitness* tertinggi. Dalam kasus distribusi, terdapat pinalti yang dikenakan jika kendaraan yang mengangkut barang tersebut mengalami kendala baik keterlambatan waktu maupun kurangnya ketersediaan barang yang sampai. Nilai *fitness* berbanding terbalik dengan total pinalti, semakin sedikit pinalti yang diterima, maka akan semakin baik nilai *fitness* yang dihasilkan. Nilai *fitness* dapat dihitung dengan persamaan (2):

$$fitness = \frac{1}{f_x} \quad (2)$$

Dimana :

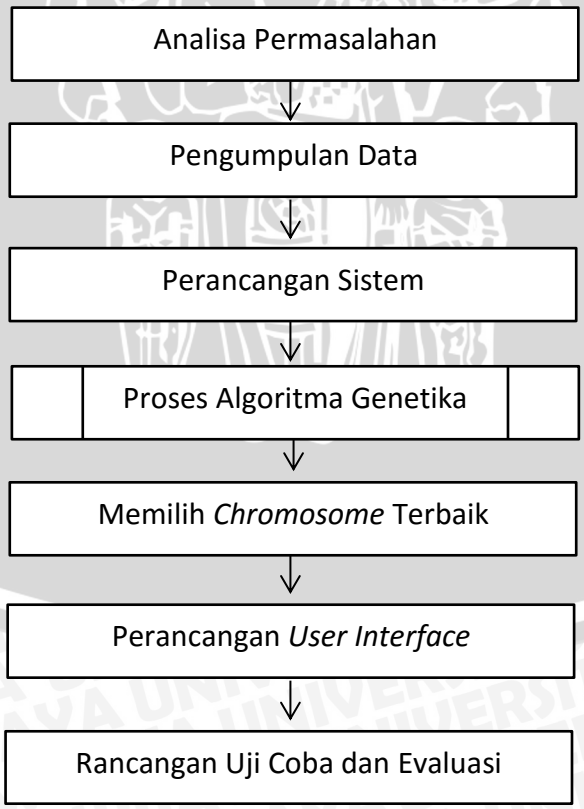
$$f_x = \sum jarak + \sum pinalti$$

BAB 3 METODOLOGI PENELITIAN

3.1 Tahapan Penelitian

Penelitian ini bersifat implementatif, yang berarti algoritma genetika diterapkan untuk memecahkan permasalahan *vehicle routing problem with time windows* (VRPTW) yang telah disediakan dalam *website* VRPTW Benchmark Problems. Penelitian ini menggunakan pendekatan perancangan (*design*) yang mempunyai tahapan penelitian seperti yang digambarkan pada Gambar 3.1 berikut:

1. Menganalisa permasalahan yang akan dibahas pada penelitian ini .
2. Memilih *dataset* dari *website* VRPTW Benchmark Problems.
3. Merancang sistem berdasarkan masalah yang telah dideskripsikan sebelumnya.
4. Melakukan perhitungan menggunakan Agoritma Genetika dengan data yang diperoleh dan berdasarkan deskripsi permasalahan.
5. Menentukan kromosom terbaik yang akan digunakan sebagai solusi dari permasalahan yang ada.
6. Merancang tampilan program.
7. Melakukan pengujian terhadap sistem untuk melihat nilai parameter yang paling optimal lalu mengevaluasi hasil dari uji coba sistem.



Gambar 3.1 Diagram Alir Metode Penelitian

3.2 Teknik Pengumpulan Data

Data yang digunakan dalam penelitian ini merupakan data yang berasal dari *website* VRPTW Benchmark Problems (http://web.cba.neu.edu/~msolomon/problems.htm). Dalam *website* tersebut terdapat 6 macam data yang berbeda. Data berdasarkan geografis yang acak terdapat dalam data R1 dan R2, data yang terkelompok terdapat dalam data C1 dan C2 sedangkan RC1 dan RC2 merupakan data campuran dari kedua *dataset*. R1, C1 dan RC1 adalah data yang menyediakan kapasitas kendaraan 200 sehingga hanya dapat melayani 5-10 pelanggan dalam sekali distribusi. Sedangkan R2, C2 dan RC2 menyediakan kapasitas lebih dari 600 sehingga dapat melayani lebih dari 30 pelanggan. Data yang digunakan dalam penelitian ini adalah data dengan nomor R101. Dalam data R101, kolom *due date* tidak digunakan dan menggunakan kolom selain *due date* dalam perhitungan.

3.3 Algoritma yang Digunakan

Penelitian ini menggunakan algoritma genetika yang menghasilkan hasil akhir berupa kromosom dengan nilai *fitness* terbaik. Kromosom dengan nilai *fitness* terbaik merupakan representasi dari urutan pelanggan yang dikunjungi dalam proses distribusi.

3.4 Kebutuhan Sistem

Kebutuhan sistem dalam bab ini digunakan untuk mengetahui kebutuhan yang digunakan peneliti dalam menyelesaikan pengembangan sistem. Kebutuhan ini meliputi *software* dan *hardware*. Untuk menunjang penelitian maka spesifikasi yang dibutuhkan adalah sebagai berikut:

1. Spesifikasi Kebutuhan Perangkat Keras (*Hardware*)
    - Komputer/laptop dengan spesifikasi :
        - Processor Intel Core i5 (2.5 GHz)
        - RAM 4 GB DDR 3
        - VGA Nvidia GeForce GT 525M 512 MB
    - Mouse
2. Spesifikasi Kebutuhan Perangkat Lunak (*Software*)
    - Microsoft Word 2013, untuk penyusunan skripsi
    - Microsoft Excel 2013, untuk melakukan perhitungan manual
    - Netbean, untuk pemrograman berbasis java

BAB 4 PERANCANGAN

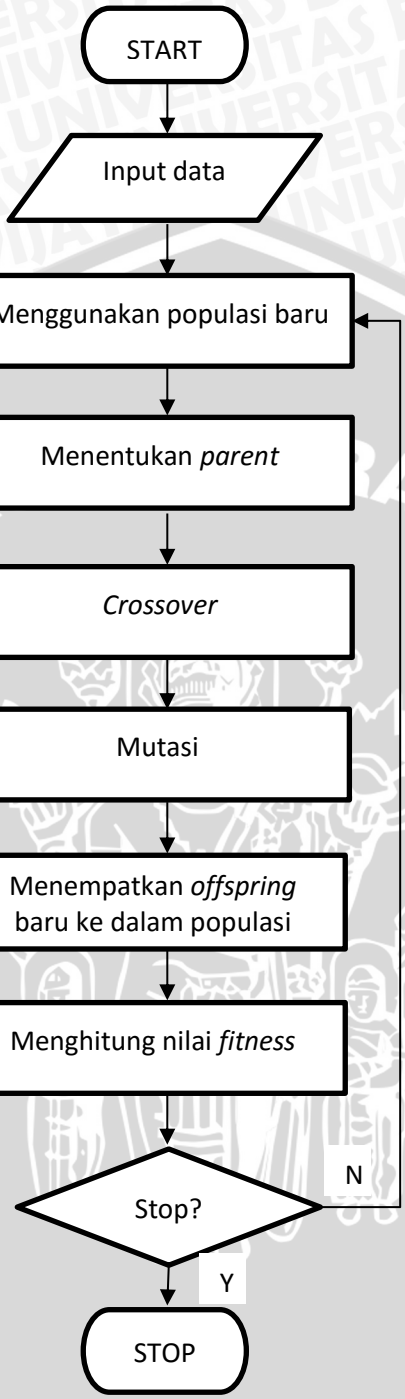
4.1 Formulasi Permasalahan

Vehicle routing problem with time window merupakan permasalahan yang sering dihadapi dalam kasus distribusi seperti yang telah disediakan oleh *website* VRPTW Benchmark Problems. Dalam kasus tersebut terdapat beberapa komponen yang dapat digunakan dalam mencari rute tercepat. *Cust No.* merupakan urutan pelanggan yang telah diberi nomor. *XCoord* dan *YCoord* merupakan koordinat lokasi pelanggan tersebut dan dapat digunakan untuk menentukan jarak dari pelanggan satu ke pelanggan lainnya mengunakan *Euclidean Distance*. *Demand* merupakan jumlah pesanan tiap pelanggan yang harus dilayani. *Ready Time* adalah waktu buka pelanggan tersebut untuk dapat dilayani dan *Service Time* adalah waktu layanan di tiap pelanggan.

Pada kasus VRPTW, terdapat 100 pelanggan yang harus dilayani. Pelanggan-pelanggan ini nantinya akan digunakan sebagai kromosom yang akan dihitung nilai fitnessnya.

4.2 Siklus Algoritma Genetika

Algoritma genetika bekerja berdasarkan teori Darwin dimana individu yang bertahan adalah individu dengan kriteria yang paling cocok. Dalam hal ini, individu yang bertahan adalah individu dengan nilai *fitness* paling tinggi. Algoritma ini dimulai dengan membuat kromosom yang didapatkan dari nilai acak, lalu kromosom ini digunakan untuk membentuk generasi baru dengan harapan bahwa generasi baru dapat lebih baik dari generasi sebelumnya. Setelah itu didapatkan solusi baru yang akan dipilih berdasarkan nilai *fitness*nya yang disebut *offspring*. Proses tersebut akan diulang-ulang sampai memenuhi suatu kondisi tertentu. Berikut ini diagram alur dari proses algoritma genetika yang digambarkan pada Gambar 4.1.



Gambar 4.1 Diagram Alir Algoritma Genetika

4.3 Siklus Penyelesaian Masalah *Vehicle Routing Problem with Time Window* (VRPTW) Menggunakan Algoritma Genetika

Pada penelitian ini terdapat 100 pelanggan dan diambil 11 sampel pelanggan teratas yang harus dilayani. 11 data teratas terdiri dari 10 pelanggan

dan 1 pabrik (*Cust No.* 0). Contoh data yang digunakan ditampilkan pada Tabel 4.1.

Tabel 4.1 Contoh Data Toko

Cust No.	Koor. X	Koor. Y	Demand	Ready Time	Service Time
0	35	35	0	0	0
1	41	49	10	161	10
2	35	17	7	50	10
3	55	45	13	116	10
4	55	20	19	149	10
5	15	30	26	34	10
6	25	30	3	99	10
7	20	50	5	81	10
8	10	43	9	95	10
9	55	60	16	97	10
10	30	60	16	124	10

Pelanggan-pelanggan tersebut direpresentasikan dengan 10 titik dengan jarak yang berbeda. Jarak antar pelanggan dapat dicari menggunakan rumus *Euclidean Distance*. Jarak pada setiap pelanggan yang dibulatkan terdapat dalam Tabel 4.2.

Tabel 4.2 Tabel Jarak Setiap Toko

Tujuan/Km	Pabrik	1	2	3	4	5	6	7	8	9	10
Pabrik	-	15	18	22	25	21	11	21	26	32	25
1	15	-	33	15	32	32	25	21	32	18	16
2	18	33	-	34	20	24	16	36	36	47	43
3	22	15	34	-	25	43	34	35	45	15	29
4	25	32	20	25	-	41	32	46	51	40	47
5	21	32	24	43	41	-	10	21	14	50	34
6	11	25	16	34	32	10	-	21	20	42	30
7	21	21	36	35	46	21	21	-	12	36	14
8	26	32	36	45	51	14	20	12	-	48	26
9	32	18	47	15	40	50	42	36	48	-	25
10	25	16	43	29	47	34	30	14	26	25	-

Dalam data terdapat 25 kendaraan yang akan digunakan dalam penelitian ini, tetapi untuk contoh perhitungan akan dinunakan 3 kendaraan. Daftar kendaraan yang digunakan disebutkan dalam Tabel 4.3.

Tabel 4.3 Daftar Kendaraan

Kendaraan ke-i	Waktu Berangkat	Kapasitas kg)
1	20	40
2	30	40
3	40	40

4.3.1 Parameter Algoritma Genetika

Tahap pertama adalah inisialisasi parameter awal yang menentukan seberapa banyak *chromosome* dalam satu populasi dan menentukan jumlah gen dalam satu *chromosome*. Banyaknya pelanggan yang dituju diinisialisasikan sebagai gen yang ada dalam *chromosome*. Pada saat pengiriman, kecepatan kendaraan pengirim adalah konstan yaitu 2 satuan/menit. Dengan c*rossover rate* adalah 0,2 dan *mutation rate* adalah 0.2 makan akan menghasilkan *offspring* berjumlah 2 untuk *crossover* dan mutasi. Terdapat 5 *popsize* yang akan digunakan dalam penelitian ini.

4.3.2 Representasi Kromosom dan Perhitungan Fitness

Permasalahan *vehicle routing problem with time window* merupakan masalah kombinatorial, maka kromosom akan direpresentasikan menggunakan pengkodean permutasi. Pengkodean permutasi adalah pengkodean yang menggunakan angka untuk merepresentasikan suatu masalah dan digunakan sebagai urutan (Sundarningsih, Mahmudy dan Sutrisno, 2015).

Dalam permasalahan distribusi, kromosom tersusun atas representasi nomor toko yang harus dikunjungi. Jika terdapat 10 toko yang harus dikunjungi, maka kromosom yang terbentuk adalah [1 2 3 4 5 6 7 8 9 0]. Tetapi, kromosom dibangkitkan secara acak sehingga kromosom yang terbentuk tidak selalu berurutan.

Untuk menghitung nilai *fitness*, dibutuhkan jarak total dan penalti pada setiap kromosom. PInalti adalah nilai yang diberikan jika kendaraan tersebut tidak mencapai kondisi tertentu. Dalam hal ini, pinalti diberikan jika kendaraan tersebut datang sebelum pelanggan tersebut buka atau tidak dapat memenuhi permintaan permintaan. Jika diberikan kromosom X1 [4 7 1 8 10 2 5 6 9 3] dengan jam buka pelanggan yang berbeda, kecepatan kendaraan konstan 2 satuan/menit dan kendaraan pertama berangkat pada waktu 20, maka untuk perhitungan pinalti ditunjukkan dalam Tabel 4.4 berikut:

Tabel 4.4 Perhitungan Pinalti

Truk1 (20, 40 kg)

Node	Permintaan	Suplai	jarak (km)	waktu (menit)	sampai	*Ready time*	tunggu	mulai	layanan	selesai	pinalty (menit)	pinalti (kg)
4	19	21	25	50	70	149	79	149	10	159	79	0
7	5	16	46	92	251	81	0	251	10	261	0	0
1	10	6	21	42	303	161	0	303	10	313	0	0
8	9	-3	32	63	376	95	0	376	10	386	0	3
Total			124	247							79	3

Truk 2 (30, 40 kg)

Node	Permintaan	Suplai	jarak (km)	waktu (menit)	sampai	*Ready time*	tunggu	mulai	layanan	selesai	pinalty (menit)	pinalti (kg)
10	16	24	25	51	81	124	43	124	10	134	43	0
2	7	17	43	87	221	50	0	221	10	231	0	0
5	26	-9	24	48	278	34	0	278	10	288	0	9
6	3	-12	10	20	308	99	0	308	10	318	0	12
Total			103	205							43	21

Truk 3 (40, 40 kg)

Node	Permintaan	Suplai	jarak (km)	waktu (menit)	sampai	*Ready time*	tunggu	mulai	layanan	selesai	pinalty (menit)	pinalti (kg)
9	16	24	32	64	104	97	0	104	10	114	0	0
3	13	11	15	30	144	116	0	144	10	154	0	0
Total			47	94							0	0

Setelah mendapatkan penalti dan total jarak dari toko ke toko, selanjutnya menghitung nilai *fitness* menggunakan persamaan (3):

$$fitness = \frac{1}{C+Pw+Ps} \quad (3)$$

C merupakan total jarak yang ditempuh oleh semua kendaraaan. Pw merupakan total pinalti waktu yang didapatkan dari persamaan (4). Pinalti waktu berlaku jika waktu sampai kurang dari waktu ready. Ps adalah total pinalti kapasitas.

$$Pw = |\, w_{sampai} - w_{ready}| \quad (4)$$

Setelah menghitung menggunakan rumus di atas, berikut merupakan contoh hasil nilai *fitness* yang ditunjukkan dalam Tabel 4.5.

Tabel 4.5 Perhitungan Nilai *Fitness*

Kromosom	Total Pinalti waktu (Pw)	Total Pinalti Kapasitas (Ps)	Total Jarak (C)	*fitness*
[4 7 1 8 10 2 5 6 9 3]	122	24	273	0.002384582

4.3.3 Inisialisasi Populasi Awal

Populasi awal dibuat berdasarkan jumlah yang ditentukan sebelumnya. Di dalam satu populasi, terdapat 5 *popsize* yang masing-masing memiliki 15 *chromosome* yang dibangkitkan secara random. Jumlah *chromosome* ini dibangkitkan pada rentang angka 1-10 dikarenakan jumlah toko yang harus dikunjungi sebanyak 10 pelanggan. Daftar *popsize* beserta *chromosome* terdapat dalam Tabel 4.6 berikut:

Tabel 4.6 Data Populasi Awal

No.	Chromosome									
	1	2	3	4	5	6	7	8	9	10
1	4	7	1	8	10	2	5	6	9	3
2	2	4	1	5	7	9	3	8	10	6
3	8	5	7	3	6	4	2	9	1	10
4	10	2	6	1	4	8	5	7	3	9
5	1	8	4	7	9	3	6	10	2	5

4.3.4 Reproduksi

Pada proses reproduksi, terdapat dua subproses yaitu *crossover* dan mutasi. Metode *crossover* yang digunakan yaitu metode *order crossover* (OX) sedangkan metode mutase yang digunakan adalah *reciprocal exchange mutation*.

4.3.4.1 *Crossover*

Proses *crossover* adalah proses yang menghasilkan kromosom baru yang disebut *offspring* dengan cara menyilangkan dua kromosom induk atau *parent*. *Parent* didapatkan dari pengambilan 2 individu secara acak dari populasi. Metode yang digunakan adalah *order crossover* (OX). *Crossover rate* (Cr) yang ditentukan sebelumnya adalah 0,4, dengan *popsize* berjumlah 5 akan menghasilkan 2 *offspring*. Penelitian ini hanya menggunakan *offspring* 1 untuk hasil *crossover*. Jumlah *offspring* ini dihasilkan bergantung dari hasil perkalian *crossover rate* dan *popsize*.

Misalkan dengan populasi pada Tabel 4.7, dipilih p3 dan p4 sebagai *parent* 1 dan *parent* 2. Setelah memilih *parent*, maka menentukan 2 bilangan acak untuk digunakan sebagai acuan agar mendapatkan calon *offspring*. 2

bilangan acak yang digunakan adalah 4 dan 8. Tabel 4.7 menunjukkan proses *order crossover*.

Tabel 4.7 Proses *Order Crossover*

Parent

P3	8	5	7	3	6	4	2	9	1	10
P4	10	2	6	1	4	8	5	7	3	9

Calon *offspring*

O1	x	x	x	3	6	4	2	9	x	x
O2	x	x	x	1	4	8	5	7	x	x

Urutan P3 dan P4 baru

P3 baru	1	10	8	5	7	3	6	4	2	9
P4 baru	3	9	10	2	6	1	4	8	5	7

Sisa P3 dan P4

P3	10	3	6	2	9
P4	10	1	8	5	7

Ofspring

O1	8	5	7	3	6	4	2	9	10	1
O2	6	2	9	1	4	8	5	7	10	3

1. Menentukan dua bilangan acak misalkan X dan Y sebagai acuan.
2. Menyalin segmen diantara X dan Y yang telah ditentukan dan diletakkan dalam calon offspring dengan urutan yang sama.
3. Menentukan parent baru dengan urutan [bilangan setelah Y][bilangan sebelum X][segmen] dan tetap memperhatikan urutan bilangan.
4. Menghilangkan bilangan yang sama pada P3 dengan segmen dari P4, dan sebaliknya.
5. Meletakkan sisa dari P3 dengan mengisi bilangan setelah segmen terlebih dahulu, setelah penuh selanjutnya mengisi bilangan sebelum segmen ([sisa 2][segmen][sisa 1])

4.3.4.2 Mutasi

Pada penelitian ini, metode mutasi yang digunakan adalah *reciprocal exchange mutation*. Metode ini menghasilkan *child* dengan cara memilih dua gen atau posisi (*exchange point* / XP) pada kromosom parent secara acak lalu menukarkan nilai pada kedua gen tersebut. *Parent* merupakan kromosom yang dipilih secara acak dari populasi. Misalnya *mutation rate* (Mr) adalah 0,4 dan *popsize* 5, maka didapatkan 2 *offspring* dengan 2 kali proses. Tabel 4.8 menunjukkan proses mutasi dengan *parent* P1 dan P5 dan menghasilkan *child* yang ditunjukkan dalam Tabel 4.9.

Tabel 4.8 Proses Mutasi *Parent* 1 dan *Parent* 2

P1	4	7	1	8	10	2	5	6	9	3
C1	4	2	1	8	10	7	5	6	9	3

P5	1	8	4	7	9	3	6	10	2	5
C5	1	3	4	7	9	8	6	10	2	5

Tabel 4.9 *Child* 1 dan *Child* 5

C1	4	2	1	8	10	7	5	6	9	3
C5	1	3	4	7	9	8	6	10	2	5

4.3.5 Nilai *Fitness*

Nilai *fitness* dapat diperoleh dari perhitungan total jarak pelanggan, waktu pelayanan dan waktu penalti. Untuk mendapatkan nilai fitness, dilakukan perhitungan manual pada setiap toko. Contoh perhitungan untuk *chromosome* ke-1 atau P1 ditunjukkan pada Tabel 4.10.

Tabel 4.10 Perhitungan *fitness*

P1	4	7	1	8	10	2	5	6	9	3

Truk1 (20, 40 kg)

Node	Permintaan	Suplai	jarak (km)	waktu (menit)	sampai	buka	tunggu	mulai	layanan	selesai	pinalty (menit)	pinalti (kg)
4	19	21	25	50	70	149	79	149	10	159	79	0
7	5	16	46	92	251	81	0	251	10	261	0	0
1	10	6	21	42	303	161	0	303	10	313	0	0
8	9	-3	32	63	376	95	0	376	10	386	0	3
Total			124	247							79	3

Truk 2 (30, 40 kg)

Node	Permintaan	Suplai	jarak (km)	waktu (menit)	sampai	buka	tunggu	mulai	layanan	selesai	pinalty (menit)	pinalti (kg)
10	16	24	25	51	81	124	43	124	10	134	43	0
2	7	17	43	87	221	50	0	221	10	231	0	0
5	26	-9	24	48	278	34	0	278	10	288	0	9
6	3	-12	10	20	308	99	0	308	10	318	0	12
Total			103	205							43	21

Truk 3 (40, 40 kg)

Node	Permintaan	Suplai	jarak (km)	waktu (menit)	sampai	buka	tunggu	mulai	layanan	selesai	pinalty (menit)	pinalti (kg)
9	16	24	32	64	104	97	0	104	10	114	0	0
3	13	11	15	30	144	116	0	144	10	154	0	0
Total			47	94							0	0

Perhitungan manual akan dihasilkan total jarak dari semua pelanggan, waktu perjalanan, waktu pelayanan, pinalti waktu, pinalti kapasitas, dan total waktu dari awal kendaraan berangkat sampai layanan terakhir dilakukan. Apabila kendaraan tiba lebih awal dari jam buka toko, maka akan ditambahkan jam tunggu yang menghasilkan pinalti waktu waktu mulai dimulai dari waktu buka pelanggan tersebut. Jika kendaraan tidak dapat memenuhi permintaan pelanggan makan dikenakan pinalti kapasitas. Pinalti total dihitung dari jumlah pinalti waktu dan pinalti kapasitas.

Setelah didapatkan hasil perhitungan manual, maka dilakukan perhitungan nilai *fitness* menggunakan persamaan (5):

$$fitness = \frac{1}{D+\sum P} \quad (5)$$

Dimana :

- D adalah jarak tempuh antar toko
- $\sum P$ adalah total pinalti

Dari persamaan 4.1 didapatkan nilai *fitness* untuk P1. Perhitungan manual dilanjutkan sampai individu C4. Hasil dari perhitungan manual pada semua individu dapat dilihat pada Tabel 4.11 berikut :

Tabel 4.11 Nilai *Fitness*

Individu	Jarak Toko	Penalti	Fitness
P1	273.3509864	146.0098049	0.002384582
P2	276.1958028	109.1359012	0.002595167
P3	237.3864481	177.6786088	0.002409261
P4	257.4672346	186.2884453	0.002253492
P5	296.2658648	124.5056652	0.002376587
C1	247.6499994	120.1515061	0.002718858
C2	257.5405645	191.6491251	0.002226231
C3	226.6250789	144.0098049	0.002698073
C4	273.1261301	126.5056652	0.002502303

4.3.6 Seleksi

Proses seleksi digunakan untuk memilih individu dari hasil proses algoritma genetika. Terdapat dua metode yang digunakan untuk melakukan proses seleksi yaitu seleksi *roulette wheel* dan seleksi elitis.

4.3.6.1 Seleksi *Binary Tournament*

Seleksi *Binary* Torunament merupakan seleksi yang mencari individu terbaik diantara dua individu acak berdasarkan nilai *fitness*. Proses seleksi sebagai berikut :

1. Menentukan nilai n. Misalkan n=3 dan daftar nilai *fitness* ditunjukkan dalam Tabel 4.12.

Tabel 4.12 Daftar *Fitness*

Individu	*Chromosome*	Fitness
P1	4 7 1 8 10 2 5 6 9 3	0.002384582
P2	2 4 1 5 7 9 3 8 10 6	0.002595167
P3	8 5 7 3 6 4 2 9 1 10	0.002409261
P4	10 2 6 1 4 8 5 7 3 9	0.002253492
P5	1 8 4 7 9 3 6 10 2 5	0.002376587
C1	8 5 7 3 6 4 2 9 10 1	0.002718858
C2	6 2 9 1 4 8 5 7 10 3	0.002226231
C3	4 2 1 8 10 7 5 6 9 3	0.002698073
C4	1 3 4 7 9 8 6 10 2 5	0.002502303

2. Untuk setiap n, melakukan :
   a. Memilih dua individu secara acak
      Perulangan n=1 sampai n=3 ditunjukkan dalam Tabel 4.13.

Tabel 4.13 Tabel Perulangan Sejumlah n

Perulangan n= 1

P3	8 5 7 3 6 4 2 9 1 10	0.002409261
P5	1 8 4 7 9 3 6 10 2 5	0.002376587

Perulangan n= 2

C3	4 2 1 8 10 7 5 6 9 3	0.002698073
P1	4 7 1 8 10 2 5 6 9 3	0.002384582

Perulangan n= 3

C2	6 2 9 1 4 8 5 7 10 3	0.002226231
C4	1 3 4 7 9 8 6 10 2 5	0.002502303

b. Memilih individu dengan nilai *fitness* yang paling tinggi diantara dua individu yang dipilih. Dari Tabel 4.13, untuk n=1 didapatkan P3, n=2 didapatkan C3 dan n=3 didapatkan C4.

Hasil proses seleksi *Binary Tournament* ditunjukkan dalam Tabel 4.14.

Tabel 4.14 Hasil Seleksi *Binary Tournament*

P3	8 5 7 3 6 4 2 9 1 10
C3	4 2 1 8 10 7 5 6 9 3
C4	1 3 4 7 9 8 6 10 2 5

4.3.6.2 Seleksi Elitis

Metode seleksi elitis merupakan metode seleksi yang memanfaatkan nilai *fitness* dari masing-masing individu yang diurutkan dari nilai tertinggi ke nilai paling rendah. Langkah-langkah yang dilakukan adalah sebagai berikut:

1. Mengurutkan setiap individu dari nilai *fitness* tinggi ke rendah.
2. Mengambil individu teratas sebanyak jumlah populasi yang telah ditentukan sebelumnya.

Contoh hasil seleksi elitis dapat dilihat pada Tabel 4.15.

Tabel 4.15 Hasil Seleksi Elitis

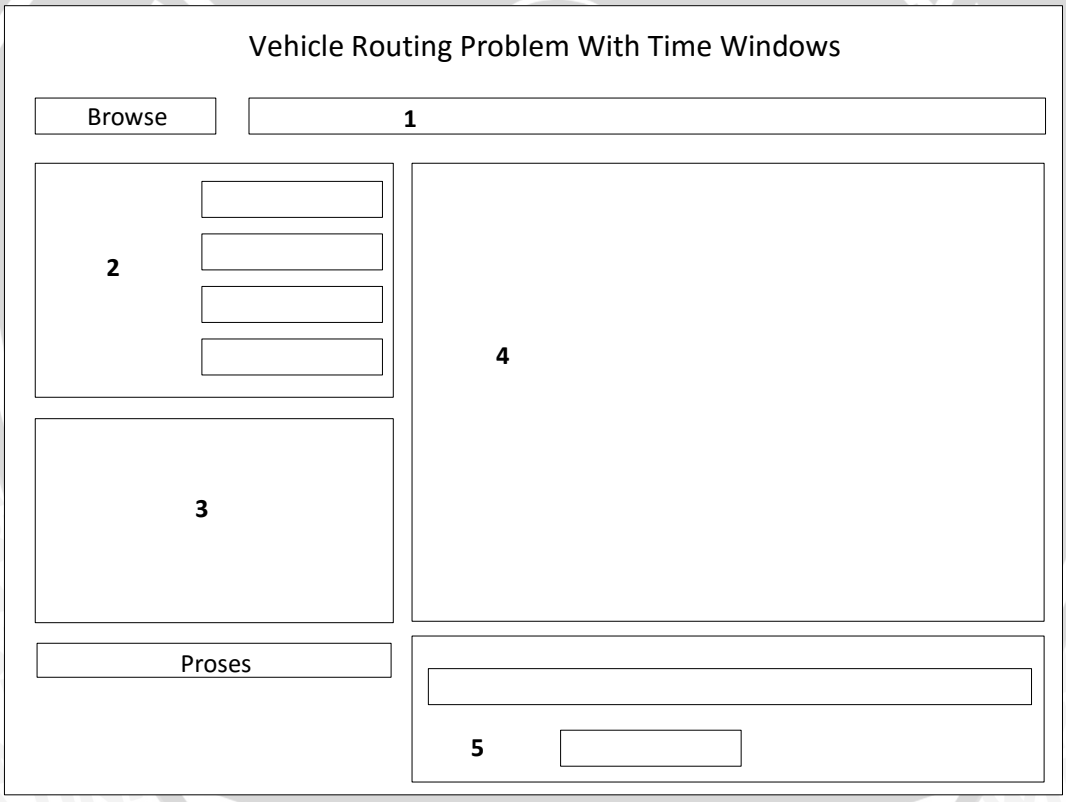
Individu	*Chromosome*	Fitness
C3	4 2 1 8 10 7 5 6 9 3	0.00855
C4	1 3 4 7 9 8 6 10 2 5	0.00847
P2	2 4 1 5 7 9 3 8 10 6	0.00758
P5	1 8 4 7 9 3 6 10 2 5	0.00746
C2	6 2 9 1 4 8 5 7 10 3	0.00685
P1	4 7 1 8 10 2 5 6 9 3	0.00641

Individu	*Chromosome*	Fitness
C1	8 5 7 3 6 4 2 9 10 1	0.00625
P3	8 5 7 3 6 4 2 9 1 10	0.00621
P4	10 2 6 1 4 8 5 7 3 9	0.00515

4.4 Perancangan *User Interface*

Pada penelitian Optimasi *vehicle routing problem with time windows* (VRPTW) menggunakan algoritma genetika, peneliti merancang *user inteface* yang terdiri dari dua bagian yaitu bagian *input* dan *output* (hasil). Pengguna dapat memasukkan inputan di bagian pertama dan akan mendapatkan hasil di bagian kedua. Pada tampilan awal, terdapat tombol *browse* untuk memilih *dataset* yang berupa file *excel*. Saat tombol *browse* ditekan, akan muncul jendela baru yang membuat pengguna dapat memilih *dataset* yang diinginkan. Berikut merupakan rancangan *user interface* yang dapat dilihat dalam gambar 4.1.

Gambar 4.1 Tampilan *User Interface*



Keterangan:

1. Setelah pengguna memilih *dataset*, *filepath* akan muncul pada kotak 1.
2. Kotak 2 berguna untuk memasukkan nilai parameter genetika seperti ukuran populasi maksimal, banyaknya generasi, *crossover rate* dan *mutation rate*.
3. Kotak 3 berguna untuk memilih metode seleksi yang akan digunakan. Kotak 3 akan berisi pilihan metode seleksi elitis dan metode seleksi *binary tournament*.

4. Kotak 4 berisi 2 kolom yaitu kolom *dataset* dan kolom hasil. Kolom *dataset* berisi data yang akan digunakan. Sedangkan kolom hasil berisi hasil dari proses algoritma genetika berupa kromosom dan nilai *fitness* beserta generasi.
5. Kotak 5 berisi kromosom terbaik beserta nilai *fitness* kromosom tersebut.

4.5 Perancangan Pengujian

Pada perancangan pengujian terdapat beberapa mekanisme pengujian untuk membandingkan beberapa kondisi yang menghasilkan nilai *fitness* yang terbesar. Beberapa mekanisme pengujian adalah sebagai berikut:

1. Pengujian untuk membandingkan nilai *fitness* menggunakan seleksi elitis dan seleksi *binary tournament*.
2. Pengujian untuk mengetahui ukuran populasi yang optimal.
3. Pengujian untuk mengetahui banyaknya generasi yang optimal.
4. Pengujian untuk mengetahui kombinasi *crossover* dan *mutation rate* yang optimal.
5. Pengujian untuk mengetahui besar nilai *fitness* jika menggunakan parameter terbaik.

4.5.2 Pengujian Perbandingan Metode Seleksi

Pengujian dilakukan dengan membandingkan hasil dari proses algoritma genetika VRPTW menggunakan dua metode seleksi yang berbeda yaitu seleksi elitis dan *binary tournament*. Program akan diberi inputan yang sama untuk setiap metode seleksi yaitu populasi sebanyak 50, generasi sejumlah 500, *crossover rate* 0.5 dan *mutation rate* 0.5. Pengujian akan dilakukan sebanyak 10 kali dikarenakan hasil untuk setiap generasi berbeda. Tabel 4.16 Menunjukkan scenario pengujian perbandingan metode seleksi.

Tabel 4.16 Pengujian perbandingan metode seleksi

Fitness dalam percobaan ke-	Metode Seleksi	
	Elitis	*Binary Tournament*
1		
2		
3		
4		
5		
6		
7		
8		
9		

Fitness dalam percobaan ke-	Metode Seleksi	
	Elitis	*Binary Tournament*
10		

4.5.3 Pengujian Jumlah Populasi Optimal

Pengujian dilakukan dengan membandingkan hasil dari proses algoritma genetika VRPTW menggunakan jumlah populasi yang berbeda-beda. Proses pengujian dilakukan sebanyak 10 kali dengan jumlah generasi sebanyak 100, *crossover rate* 0.5 dan *mutation rate* 0.5 dengan menggunakan seleksi elitis. Jumlah populasi yang digunakan adalah 50-350 dengan kelipatan 50. Tabel 4.17 menunjukkan mekanisme pengujian jumlah populasi optimal.

Tabel 4.17 Pengujian jumlah populasi maksimal

Fitness percobaan ke-	Jumlah Populasi						
	50	100	150	200	250	300	350
1							
2							
3							
4							
5							
6							
7							
8							
9							
10							

4.5.4 Pengujian Jumlah Generasi Optimal

Pengujian dilakukan dengan membandingkan hasil dari proses algoritma genetika VRPTW menggunakan jumlah generasi yang berbeda-beda. Proses pengujian dilakukan sebanyak 10 kali dengan jumlah populasi sebanyak 100, *crossover rate* 0.5 dan *mutation rate* 0.5. Jumlah generasi yang digunakan adalah 500-2500 dengan kelipatan 500. Tabel 4.18 menunjukkan mekanisme pengujian jumlah generasi optimal.

Tabel 4.18 Pengujian jumlah generasi optimal

Fitness percobaan ke-	Jumlah Generasi						
	500	1000	1500	2000	2500	3000	3500
1							
2							

Fitness percobaan ke-	Jumlah Generasi						
	500	1000	1500	2000	2500	3000	3500
3							
4							
5							
6							
7							
8							
9							
10							

4.5.5 Pengujian Kombinasi Crossover rate dan Mutation Rate

Pengujian dilakukan dengan membandingkan hasil dari proses algoritma genetika VRPTW menggunakan jumlah kombinasi dari *crossover* dan *mutation rate* yang berbeda-beda. Proses pengujian dilakukan sebanyak 10 kali dengan jumlah populasi sebanyak 50 dan jumlah generasi sebanyak 500. Kombinasi pada *crossover* dan *mutation rate* menggunakan selisih 0.2. Tabel 4.19 menunjukkan mekanisme pengujian jumlah populasi optimal.

Tabel 4.19 Pengujian kombinasi *crossover* dan *mutation rate*

Fitness percobaan ke-	Kombinasi Cr dan Mr					
	Cr = 1	Cr = 0.8	Cr = 0.6	Cr = 0.4	Cr = 0.2	Cr = 0
	Mr = 0	Mr = 0.2	Mr = 0.4	Mr = 0.2	Mr = 0.8	Mr = 1
1						
2						
3						
4						
5						
6						
7						
8						
9						
10						

4.5.6 Pengujian Menggunakan Parameter Terbaik

Pengujian ini dilakukan untuk mengetahui rute yang paling optimal dan nilai *fitness* terbaik. Pengujian ini menggunakan nilai terbaik untuk setiap parameter yang didapatkan dari pengujian-pengujian sebelumnya. Ukuran

populasi yang digunakan sebesar 300 dan banyaknya generasi sebesar 3000. *Crossover rate* dan *mutation rate* menggunakan nilai 0.4 dan 0.6 serta metode seleksi menggunakan seleksi elitis. Pengujian ini dilakukan sebanyak 10 kali percobaan dan diambil hasil dengan nilai *fitness* terbaik.

Fitness pada percobaan ke-									
1	2	3	4	5	6	7	8	9	10

BAB 5 MPLEMENTASI

5.1 Struktur *Class*

Permasalahan *Vehicle Routing Problem With Time Windows* dapat diimplementasikan menggunakan program yang menerapkan algoritma genetika. Dalam program yang dibuat, terdapat beberapa kelas untuk menunjang program diantaranya kelas *Utils*, *Genetic*, *ReadData*, dan kelas GUI. Seluruh proses algoritma genetika terdapat dalam kelas *Genetic*, sedangkan kelas *Utils* digunakan sebagai penunjang proses algoritma genetika. Kelas *ReadData* berfungsi sebagai tempat untuk mendapatkan data awal. Kelas GUI berguna sebagai aktifitas utama.

5.1.1 Populasi Awal

Populasi awal yang akan digunakan dalam program dibuat dalam kelas *Genetic* pada *method initPop()*. Mula mula, program membuat kromosom dengan bilangan random yang berbeda sejumlah banyaknya toko. Setelah itu mengumpulkan kromosom sejumlah bilangan yang diinputkan oleh user. Proses membuat populasi awal dapat dilihat dalam *Source Code* 5.1

Source Code 5.1 Populasi awal

```
1   private void initPop () {
2       for (int i=0; i<this.popSize; i++) {
3          ArrayList <Integer> chromosome = new <Integer> ArrayList ();
4          int max = 100;
5          int min = 1;
6
7          for (int j=min; j<=max; j++) {
8             chromosome.add(j);//membuat kromosom dari 1-100
9          }
10
11         Collections.shuffle(chromosome);//mengacak isi kromosom
12         this.pop.add(chromosome);//menambahkan kedalam populasi
13      }
14    }
```

5.1.2 *Crossover*

Tipe *crossover* yang digunakan adalah *order crossover* (*OX Crossover*). Method untuk melakukan proses ini berada dalam kelas *Genetic*. Mula mula menentukan *parent1* dan *parent2* yang diambil secara random dari populasi. Setelah itu menentukan titik potong k1 dan k2 seara random. Lalu melakukan proses *crossover* yang menghasilkan *offspring* baru. Proses *OX Crossover* ditunjukkan dalam Source Code 5.2.

Source Code 5.2 *OX Crossover*

```
ArrayList <ArrayList <Integer>> offspring = new <ArrayList <Integer>>
ArrayList ();
      ArrayList <Integer> p1 = new <Integer> ArrayList ();
      ArrayList <Integer> p2 = new <Integer> ArrayList ();
      ArrayList <Integer> o1 = new <Integer> ArrayList ();
      ArrayList <Integer> o2 = new <Integer> ArrayList ();
      ArrayList <Integer> tp1 = new <Integer> ArrayList ();
      ArrayList <Integer> tp2 = new <Integer> ArrayList ();
      ArrayList <Integer> to1 = new <Integer> ArrayList ();
      ArrayList <Integer> to2 = new <Integer> ArrayList ();
      //memilih 2 bilangan secara acak dari popsize
      int k1=this.getRandomNum(this.pop.size());
      int k2=this.getRandomNum(this.pop.size());

      while (k1==k2){
         k2 = this.getRandomNum(this.pop.size());
      }
      //memilih kromosom 1&2 untuk proses crossover
      p1.addAll(this.pop.get(k1));
      p2.addAll(this.pop.get(k2));
      //memilih titik potong 1&2
      int x1 = this.getRandomNum(p1.size()-1);
      int x2 = this.getRandomNum(p2.size()-1);
      int x;
      int y;

      if(x1<x2){
         x = x1;
         y = x2;
      }
      else {
         x = x2;
         y = x1;
      }
      //memindah gen ke calon offspring
      for (int i=x; i<=y; i++) {
         to1.add(p1.get(i));
      }

      for (int i=x; i<=y; i++) {
         to2.add(p2.get(i));
      }
```

```
43          //memindah gen ke parent sementara
44      for (int i=(y+1); i<p1.size(); i++) {
45         tp1.add(p1.get(i));
46      }
47
48      for (int i=(y+1); i<p2.size(); i++) {
49         tp2.add(p2.get(i));
50      }
51
52      for (int i=0; i<=y; i++) {
53         tp1.add(p1.get(i));
54      }
55
56      for (int i=0; i<=y; i++) {
57         tp2.add(p2.get(i));
58      }
59
60          //menghapus gen yang sama
61      for (int i=0; i<tp1.size(); i++) {
62         for (int j=0; j<to2.size(); j++) {
63            if (tp1.get(i)==to2.get(j)) {
64               tp1.remove(i);
65               if (i>0) {
66                  i--;
67               }
68               else {
69                  i=0;
70               }
71            }
72         }
73      }
74
75      for (int i=0; i<tp2.size(); i++) {
76         for (int j=0; j<to1.size(); j++) {
77            if (tp2.get(i)==to1.get(j)) {
78               tp2.remove(i);
79               if (i>0) {
80                  i--;
81               }
82               else {
83                  i=0;
84               }
```

```
85              }
86            }
87        }
88        //set pointer
89        int pointer = p1.size()-y-1;
90
91            //menaruh gen ke offspring
92        for (int i=pointer; i<tp2.size(); i++) {
93          o1.add(tp2.get(i));
94        }
95
96        o1.addAll(to1);
97
98        for (int i=0; i<pointer; i++) {
99          o1.add(tp2.get(i));
100       }
101
102           //remove duplikat
103       offspring.add(removeDuplicate(o1));
104       System.out.println(offspring);
105       return offspring;
106   }
```

5.1.3 Mutasi

Terdapat beberapa metode mutasi dalam algoritma genetika, tetapi dalam penelitian ini menggunakan metode mutasi *reciprocal exchange mutation*. Pertama, membuat dua buah bilangan random sebagai *exchange point* atau xp. Setelah itu menukar masing-masing gen dalam kromosom pada index ke-xp. Source code 5.3 menunjukkan proses mutasi.

Source Code 5.3 Mutasi

```
1   public ArrayList <Integer> REMutation (ArrayList <Integer>
2   chromosome){
3       //memilih 2 bilangan acak dari total popsize
4       int xp1 = this.getRandomNum(chromosome.size());
5       int xp2 = this.getRandomNum(chromosome.size());
6
7       while (xp1==xp2){
8         xp2 = this.getRandomNum(chromosome.size());
9       }
10
```

```
11        int gen1 = chromosome.get(xp1);//memilih gen untuk ditukar
12        int gen2 = chromosome.get(xp2); //memilih gen untuk ditukar
13
14        chromosome.set(xp1, gen2);//  menukar gen 1
15        chromosome.set(xp2, gen1);//  dengan gen 2
16
17        System.out.println(chromosome);
18        return chromosome;
19    }
```

5.1.4 *Fitness*

Method ini berfungsi untuk menghitung fitness tiap kromosom dalam populasi. Hasil dari *method* ini berupa nilai *fitness.* Proses menghitung fitness ditunjukkan dalam Source Code 5.4.

Source Code 5.4 Perhitungan *Fitness*

```
1   public double getFitness(ArrayList <Integer> chromosome){
2         int jmlKendaraan = 25;
3         int waktuBerangkat=0;
4         int kecKendaraan = 2;
5         int kapasitas = 200;
6         int pinaltiKapasitas = 0;
7         double pinaltiWaktu = 0;
8         double waktuSampai=0;
9         double waktuMulai=0;
10        double waktuSelesai=0;
11        double fitness = 0;
12        int ik=1;//index kendaraan
13        double totalJarak=0;
14        double jarak=0;
15        int counter=0;
16        int prevIndex = 0;
17
18        for(int i2=0;i2<jmlKendaraan;i2++){
19           for(int i4=0;i4<4;i4++){
20
21              int index = chromosome.get(counter);
22              if (counter > 0){
23                 prevIndex = chromosome.get(counter-1);
24              }
25
26              //Menghitung Kapasitas
27              if(i4==0) kapasitas=200;
28
```

```
29
30              kapasitas = kapasitas - data.getPermintaan(index);
31              if (kapasitas < 0){pinaltiKapasitas += Math.abs(kapasitas);}
32  //menghitung pinalti Kapasitas
33
34              //menghitung Jarak
35              if(i4==0){jarak = data.getJarak(0, index);}
36              else {jarak = data.getJarak(index, prevIndex);}
37              totalJarak = totalJarak + jarak;
38
39              //Menghitung waktu
40              double waktuKendaraan = jarak*kecKendaraan;
41              if(i4==0){waktuSampai = waktuBerangkat + waktuKendaraan
42  ;}
43              else {waktuSampai = waktuKendaraan + waktuSelesai;}
44
45              if (waktuSampai<data.getWaktuBuka(index)) {
46                 waktuMulai=data.getWaktuBuka(index);
47                 pinaltiWaktu = data.getWaktuBuka(index)-waktuSampai;
48              }
49              else waktuMulai=waktuSampai;
50              waktuSelesai=waktuMulai + data.getWaktuPelayanan(index);
51              pinaltiWaktu+=pinaltiWaktu;
52
53              counter++;
54            }
55            waktuBerangkat+=5;
56         }
57         fitness = 1/(totalJarak+(pinaltiKapasitas+pinaltiWaktu));
58         System.out.println("fitness "+fitness);
59         return fitness ;
60      }
```

5.1.5 Seleksi Elitis

Seleksi elitis terdapat dalam *method Elitism* dalam kelas *Genetic*. *Method* ini berfungsi untuk menyeleksi kromosom-kromosom berdasarkan nilai fitness kromosom tersebut. *Method* ini menghasilkan *offspring* yang akan digunakan dalam generasi selanjutnya. Proses seleksi *Elitis* ditunjukkan dalam Source Code 5.5.

Source Code 5.5 Seleksi Elitis

```
1  public ArrayList <ArrayList <Integer>> Elitism (int n, ArrayList <Double>
2  fitness, Map <Double,Integer> fitlist){
3
4       ArrayList <ArrayList <Integer>> offspring = new <ArrayList
5  <Integer>> ArrayList();
6       ArrayList <Integer> gentemp = new ArrayList<Integer>();
7
8       Collections.sort(fitness);
9       Collections.reverse(fitness);
10
11      for (int i=0; i<fitness.size(); i++){
12         gentemp.add(fitlist.get(fitness.get(i)));
13      }
14      System.out.println("n "+n);
15      for (int i=0; i<n;i++){
16         offspring.add((ArrayList) this.pop.get(gentemp.get(i)));
17      }
18
19      return offspring;
20    }
```

5.1.6 Seleksi *Binary Tournament*

Seleksi *Binary Tournament* terdapat dalam *method BinaryTournament* dalam kelas *Genetic*. *Method* ini berfungsi untuk menyeleksi kromosom-kromosom dengan membandingkan *fitness* dari dua kromosom yang dipilih secara random. *Method* ini menghasilkan *offspring* yang akan digunakan dalam generasi selanjutnya. Proses seleksi *Binary Tournament* ditunjukkan dalam Source Code 5.6.

Source Code 5.6 Seleksi *Binary Tournament*

```
1  public ArrayList <ArrayList <Integer>> BinaryTournament(int n,
2  ArrayList <Double> f, Map <Double,Integer> fitlist){
3       //f adalah arraylist fitness; fitlist adalah hashmap dari fitness
4       ArrayList <ArrayList <Integer>> offspring = new ArrayList <ArrayList
5  <Integer>>();
6       ArrayList <Double> fitness = new ArrayList <Double>();
7       ArrayList <Integer> ranNum = new <Integer> ArrayList ();
8
9       for (int i=0; i<n; i++) {
10         ranNum.add(i);
11      }
12
13      Collections.shuffle(ranNum);
```

```
14        for (int i=1; i<n; i++){
15
16           double f1 = f.get(ranNum.get(i-1));//memilih fitness pada index
17           double f2 = f.get(ranNum.get(i)); //memilih fitness pada index
18
19           if (f1>f2) {
20              offspring.add(this.pop.get(fitlist.get(f1)));//memilih
21    kromosom dari index yang didapat dari hashmap fitlist
22           }
23           else {
24              offspring.add(this.pop.get(fitlist.get(f1)));
25           }
26        }
27
28        return offspring;
29     }
```

5.1.7 Proses Algoritma Genetika

Proses algoritma genetika terdapat dalam method *algorithm*. Dalam *method* ini, seluruh *method-method* sebelumnya digunakan untuk melakukan proses algoritma genetika. Proses algoritma genetikan ditunjukkan dalam Source Code 5.7.

Source Code 5.7 Algoritma Genetika

```
1     public void algorithm(String seleksi){
2        Map<Double,Integer> fitList = new HashMap<Double,Integer>();
3        ArrayList <Double> fitness = new ArrayList<Double>();
4        ArrayList hasilguitemp = new ArrayList();
5
6        //membuat populasi
7        this.initPop();
8        System.out.println("Populasi Awal : ");
9
10    System.out.println("=========================================
11    ==============");
12       for(int i=0;i<this.pop.size();i++){
13          System.out.println("chromosome index ke-"+i+" :
14    "+this.pop.get(i));
15       }
16       System.out.println("Pop size "+this.popSize);
17       System.out.println("Gen size "+this.genSize);
18       System.out.println("mut rate "+this.mr);
19       System.out.println("ccrossover rate "+this.cr);
20       System.out.println("pop size "+this.pop.size());
```

```
21      int mutNum=(int)(this.cr*this.pop.size());
22      int cNum=(int)(mr*this.pop.size());
23
24      for (int g=1;g<=this.genSize;g++){
25
26         fitness.clear();//menghilangkan isi arraylist fitness
27         fitList.clear();//menghilangkan isi hashmap fitlist
28
29         System.out.println("GENERASI "+g);
30
31 System.out.println("=========================================
32 ==============");
33         System.out.println("Crossover : "+mutNum);
34
35         //menambah hasil Order Crossover ke populasi
36         for(int i=0;i<mutNum;i++){
37            resultCross.addAll(this.OXCrossover());
38         }
39         this.pop.addAll(resultCross);
40
41
42 System.out.println("=========================================
43 ==============");
44         System.out.println("Mutasi : "+cNum);
45
46         // menambah hasil Reciprocal Mutation ke populasi
47         for(int i=0;i<cNum;i++){
48
49 this.pop.add(this.REMutation(this.pop.get(getRandomNum(this.pop.siz
50 e()))));
51         }
52
53         //menghitung nilai fitness masing masing kromosom dan
54 ditambah ke hashmap
55
56 System.out.println("=========================================
57 ==============");
58         System.out.println("Fitness : ");
59
60         for (int i = 0; i<this.pop.size();i++){
61
62            double fit=getFitness(this.pop.get(i));
63            fitness.add(fit);
64            fitList.put(fit, i);
```

```
65              //masukkan hasil untuk tampilan hasil di GUI
66              this.hasilGenerasi.add(g);
67              this.hasilpop.addAll(pop);
68              this.hasilFitness.add(fit);
69          }
70
71          System.out.println(fitList);
72          System.out.println("Fitness "+fitness);
73
74  System.out.println("=========================================
75  ==============");
76          System.out.println("Seleksi : "+seleksi);
77          if (seleksi=="elitis"){
78              //System.out.println(pop.getPop().size());
79              this.setNewPop(Elitism(this.popSize, fitness, fitList ));
80          }
81          else{
82              this.setNewPop(BinaryTournament(this.popSize, fitness, fitList
83  ));
84          }
85
86  System.out.println("=========================================
87  ==============");
88          System.out.println("Pop Baru : ");
89
90          for(int i=0;i<this.pop.size();i++){
91              System.out.println("chromosome index ke-"+i+" :
92  "+this.pop.get(i));//+" | Fitness : "+getFitness(this.pop.get(i)));
93          }
94      }
95      setBest(pop.get(0), getFitness(this.pop.get(0)));
96    }
```

BAB 6 PENGUJIAN DAN PEMBAHASAN

6.1 Hasil dan Pembahasan Pengujian Perbandingan Metode Seleksi

Pengujian yang dilakukan pertama kali adalah menguji metode seleksi yang lebih baik sehingga akan digunakan untuk pengujian selanjutnya. Pengujian ini dilakukan untuk membandingkan metode seleksi elitis dan *binary tournament* dengan menggunakan ukuran populasi sebesar 50, banyaknya generasi sebesar 500, crossover dan *mutation rate* masing-masing sebesar 0.5. Sebanyak 100 data pelanggan digunakan dalam proses pengujian. Proses pengujian dilakukan sebanyak 10 kali untuk masing-masing metode seleksi. Tabel 6.1 dan Gambar 6.1 menunjukkan hasil pengujian untuk membandingkan metode seleksi.

Tabel 6.1 Hasil Pengujian Perbandingan Metode Seleksi

Fitness dalam percobaan ke-	Metode Seleksi	
	Elitis	*Binary Tournament*
1	0.00061419590	0.00027004902
2	0.00059479839	0.00031383530
3	0.00058473316	0.00025335286
4	0.00059210899	0.00031554781
5	0.00059306453	0.00030458235
6	0.00060398788	0.00019507762
7	0.00060972595	0.00023951350
8	0.00057694477	0.00031717405
9	0.00057422781	0.00027935956
10	0.00058875458	0.00031012828
Rata-rata	0.00059325420	0.00027986203

Gambar 6.1 Grafik Hasil Pengujian Perbandingan Metode Seleksi

Gambar 6.1 merupakan grafik dari hasil pengujian perbandingan metode seleksi yang masing-masing dilakukan sebanyak 10 kali percobaan. Nilai *fitness* yang dihasilkan melalui seleksi elitis lebih stabil jika dibandingkan dengan nilai *fitness* yang dihasilkan melalui seleksi *binary tournament*. Dalam seleksi elitis, individu-individu dipilih berdasarkan nilai *fitness* tertinggi yang sebelumnya telah dilakukan *sorting*. Hal ini membuat seleksi elitis mempertahankan individu-individu dengan nilai *fitness* tertinggi untuk digunakan dalam generasi selanjutnya. Berbeda dengan seleksi elitis, metode seleksi *binary tournament* bekerja dengan cara membandingkan nilai *fitness* dari dua individu yang dipilih secara acak sehingga individu dengan nilai *fitness* yang rendah dapat lolos untuk generasi selanjutnya. Rata-rata *fitness* dari seleksi elitis juga lebih tinggi dibanding dengan rata-rata *fitness* yang dihasilkan oleh seleksi *binary tournament* yaitu 0.00059325420 untuk seleksi elitis dan 0.00027986203 untuk seleksi *binary tournament*. Dalam pengujian untuk membandingkan metode seleksi, diketahui bahwa metode seleksi elitis lebih baik daripada metode seleksi *binary tournament* sehingga pada pengujian selanjutnya akan menggunakan metode seleksi elitis.

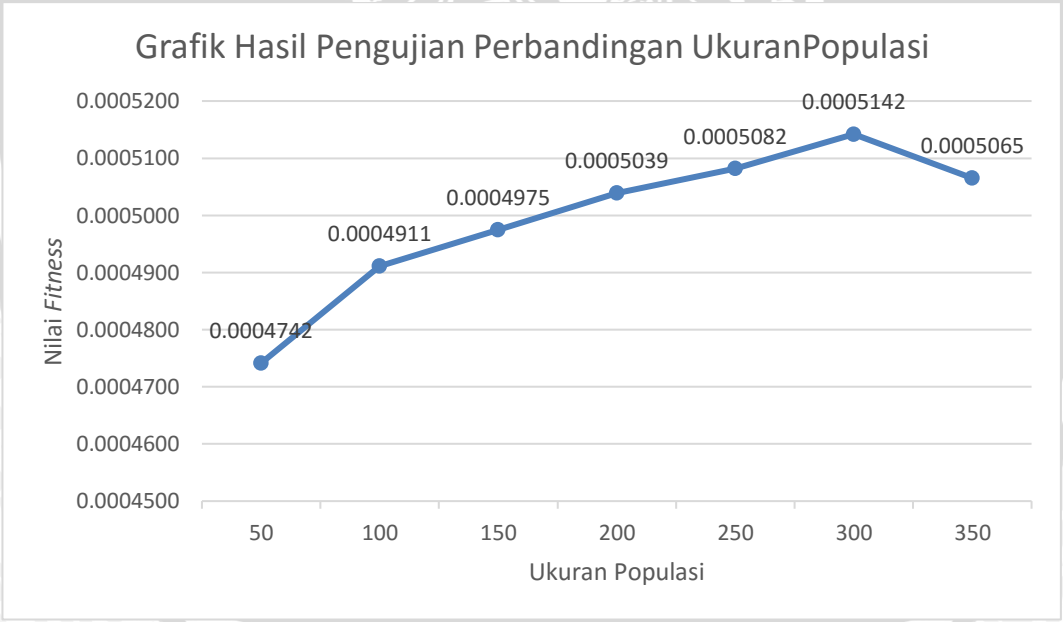
6.2 Hasil dan Pembahasan Pengujian Ukuran Populasi Optimal

Pengujian selanjutnya adalah pengujian untuk mengetahui ukuran populasi optimal yang dilakukan pada 100 pelanggan. Pengujian ini menggunakan jumlah generasi sebanyak 100 dengan crossover dan *mutation rate* masing-masing sebesar 0.5. Ukuran populasi yang digunakan adalah 50-350 dengan kelipatan 50. Percobaan ini dilakukan sebanyak 10 kali untuk setiap ukuran populasi. Tabel 6.2 dan Gambar 6.2 menunjukkan hasil pengujian untuk membandingkan ukuran populasi.

Tabel 6.2 Hasil Pengujian Perbandingan Ukuran Populasi

Fitness percobaan ke-	Ukuran Populasi						
	50	100	150	200	250	300	350
1	0.0004751	0.0004985	0.0004736	0.0005099	0.0004903	0.0005099	0.0005029
2	0.0004524	0.0005056	0.0004746	0.0004719	0.0005007	0.0005238	0.0005021
3	0.0004824	0.0004888	0.0004861	0.0005058	0.0004893	0.0005312	0.0005079
4	0.0004787	0.0004747	0.0005153	0.0005256	0.0005325	0.0005122	0.0005098
5	0.0004693	0.0004891	0.0005167	0.0004967	0.0005282	0.0005317	0.0005008
6	0.0004586	0.0004928	0.0004864	0.0004878	0.0005211	0.0004970	0.0004995
7	0.0004617	0.0004894	0.0005294	0.0004914	0.0005080	0.0005200	0.0005252
8	0.0004885	0.0004970	0.0004881	0.0005056	0.0005081	0.0004970	0.0005111
9	0.0004948	0.0004869	0.0004855	0.0005312	0.0005133	0.0005043	0.0004933
10	0.0004801	0.0004883	0.0005189	0.0005128	0.0004907	0.0005152	0.0005127
Rata-rata	0.0004742	0.0004911	0.0004975	0.0005039	0.0005082	0.0005142	0.0005065

Gambar 6.2 Grafik Hasil Pengujian Perbandingan Ukuran Populasi



Gambar 6.2 merupakan hasil nilai *fitness* dari ukuran populasi 50 sampai populasi 350. Dapat dilihat bahwa nilai *fitness* naik secara signifikan dimulai dari populasi berjumlah 50 dan mencapai puncak pada populasi berjumlah 300 dengan nilai *fitness* 0.0005142. Tetapi nilai *fitness* mengalami penurunan pada populasi berjumlah 350 dengan nilai *fitness* 0.0005065. Dapat ditarik kesimpulan bahwa ukuran populasi 300 merupakan ukuran populasi yang paling optimal untuk permasalahan VRPTW.

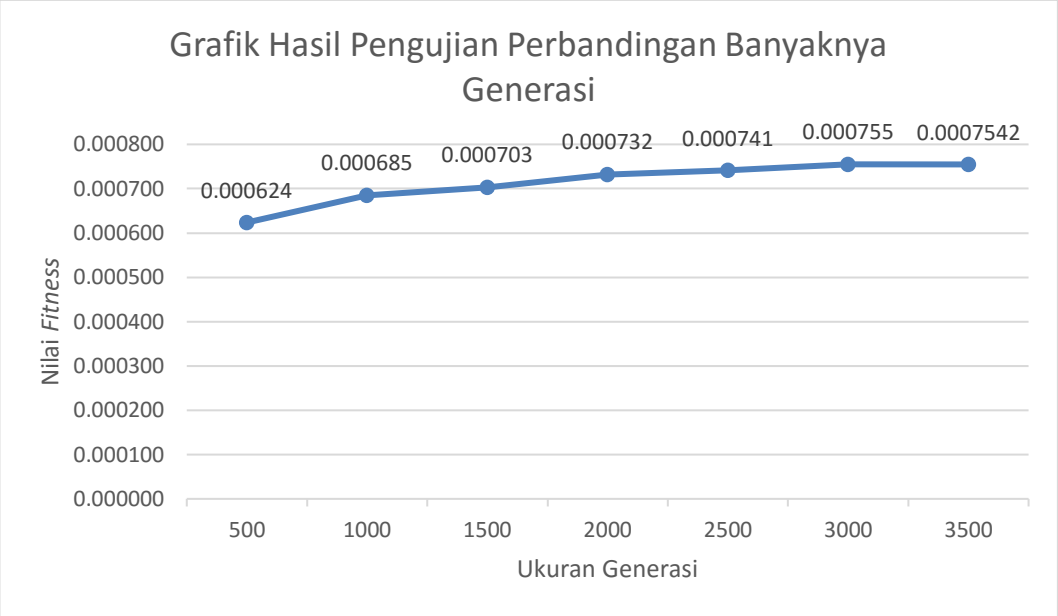
6.3 Hasil dan Pembahasan Pengujian Banyaknya Generasi Optimal

Pengujian ini dilakukan untuk mengetahui banyaknya generasi yang paling optimal dan menggunakan 100 data pelanggan. Pengujian ini menggunakan populasi sebesar 100 dengan *crossover* dan *mutation rate* masing-masing sebesar 0.5. Banyaknya generasi yang akan diuji adalah 500-3500 dengan kelipatan 500. Pengujian dilakukan sebanyak 10 kali pada masing-masing jumlah generasi. Tabel 6.3 dan Gambar 6.3 menunjukkan hasil pengujian untuk membandingkan banyaknya generasi.

Tabel 6.3 Hasil Pengujian Perbandingan Banyaknya Generasi

Fitness percobaan ke-	Banyaknya Generasi						
	500	1000	1500	2000	2500	3000	3500
1	0.0006238	0.0006840	0.0006822	0.0007173	0.0007532	0.0007604	0.0007531
2	0.0006513	0.0006569	0.0007102	0.0007305	0.0007561	0.0007451	0.0007330
3	0.0006108	0.0006998	0.0007180	0.0007517	0.0007223	0.0007426	0.0007610
4	0.0006207	0.0006713	0.0007349	0.0007233	0.0007462	0.0007485	0.0007673
5	0.0006351	0.0006751	0.0006981	0.0007232	0.0007453	0.0007593	0.0007500
6	0.0006428	0.0006959	0.0007081	0.0007427	0.0007376	0.0007578	0.0007602
7	0.0006085	0.0006886	0.0007077	0.0007325	0.0007279	0.0007700	0.0007662
8	0.0006100	0.0006984	0.0006733	0.0007184	0.0007329	0.0007654	0.0007718
9	0.0006210	0.0006889	0.0006898	0.0007372	0.0007412	0.0007507	0.0007335
10	0.0006128	0.0006937	0.0007108	0.0007472	0.0007472	0.0007506	0.0007463
Rata-rata	0.0006237	0.0006853	0.0007033	0.0007324	0.0007410	0.0007550	0.0007542

Gambar 6.3 Grafik Hasil Pengujian Perbandingan Banyaknya Generasi



Dalam Gambar 6.3 nampak nilai *fitness* pada setiap banyaknya generasi. Nilai *fitness* mengalami kenaikan yang cukup signifikan dari jumlah generasi 500 ke 1000 dan terus mengalami kenaikan sampai jumlah generasi 2000. Akan tetapi, nilai *fitness* mulai stabil pada jumlah generasi 2500 dan mencapai puncak pada generasi 3000 dengan nilai *fitness* sebesar 0.0007550. Setelah generasi 3000, nilai *fitness* mengalami penurunan nilai. Sehingga, banyaknya generasi yang paling optimal dalam permasalahan VRPTW adalah 3000.

6.4 Hasil dan Pembahasan Pengujian Kombinasi Crossover dan Mutation Rate

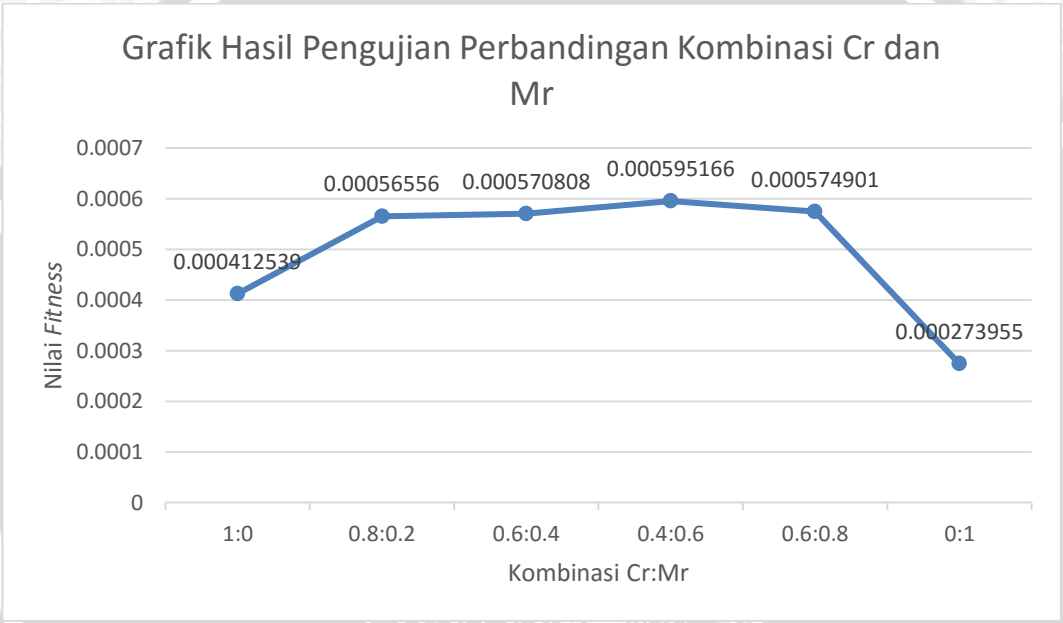
Pengujian ini dilakukan untuk mengetahui kombinasi *crossover rate* dan *mutation rate* untuk menghasilkan nilai *fitness* yang terbesar. Dalam pengujian ini, terdapat 100 data pelanggan yang akan dilakukan dalam proses pengujian. Jumlah populasi yang digunakan sebesar 50 dan jumlah generasi sebesar 500. Pengujian dilakukan 10 kali untuk setiap kombinasi. Tabel 6.4 dan Gambar 6.4 menunjukkan hasil pengujian untuk membandingkan kombinasi Cr dan Mr.

Tabel 6.4 Hasil Pengujian Perbandingan Kombinasi Cr dan Mr

Fitness percobaan ke-		Kombinasi Cr dan Mr				
	Cr = 1	Cr = 0.8	Cr = 0.6	Cr = 0.4	Cr = 0.2	Cr = 0
	Mr = 0	Mr = 0.2	Mr = 0.4	Mr = 0.2	Mr = 0.8	Mr = 1
1	0.00039125437	0.00053917380	0.00058949478	0.00058898853	0.00055888698	0.00025161242
2	0.00039125437	0.00057704613	0.00057504521	0.00060993090	0.00057266753	0.00032662097
3	0.00039506579	0.00053996018	0.00057705361	0.00056745523	0.00055832586	0.00033929874
4	0.00039937672	0.00056978520	0.00056326021	0.00061442908	0.00058564398	0.00029595411

5	0.00041686393	0.00055178851	0.00053912517	0.00059890812	0.00060264474	0.00017226282
6	0.00043643440	0.00057916726	0.00054862883	0.00059707588	0.00057743299	0.00022321380
7	0.00040801395	0.00056531639	0.00057865809	0.00062178400	0.00058313346	0.00025621964
8	0.00039814642	0.00059694543	0.00057821973	0.00057293657	0.00055860138	0.00030832950
9	0.00043885681	0.00057660720	0.00058418328	0.00058440359	0.00057267857	0.00026865775
10	0.00045012004	0.00055981176	0.00057441309	0.00059574312	0.00057899251	0.00029737535
Rata-rata	**0.00041253868**	**0.00056556019**	**0.00057080820**	**0.00059516550**	**0.00057490080**	**0.00027395451**

Gambar 6.4 Grafik Hasil Pengujian Perbandingan Kombinasi Cr dan Mr



Gambar 6.4 menunjukkan grafik nilai *fitness* untuk setiap kombinasi Cr dan Mr. Nilai *fitness* mengalami kenaikan yang signifikan dari kombinasi Cr:Mr 1:0 ke 0.8:0.2 tetapi setelah kombinasi kedua tidak menunjukkan kenaikan yang signifikan dan memuncak pada kombinasi 0.4:0.6. Nilai *fitness* justru turun pada kombinasi 0.6:0.4 dan mengalami penurunan yang signifikan pada kombinasi Cr:Mr 0:1. Hal ini menunjukkan bahwa algoritma genetika tidak dapat menghasilkan *fitness* yang optimal jika salah satu *crossover rate* atau *mutation rate* diberikan nilai 0. Kombinasi Cr:Mr yang paling optimal terletak pada kombinasi 0.4:0.6 dengan nilai *fitness* sebesar 0.00059516550.

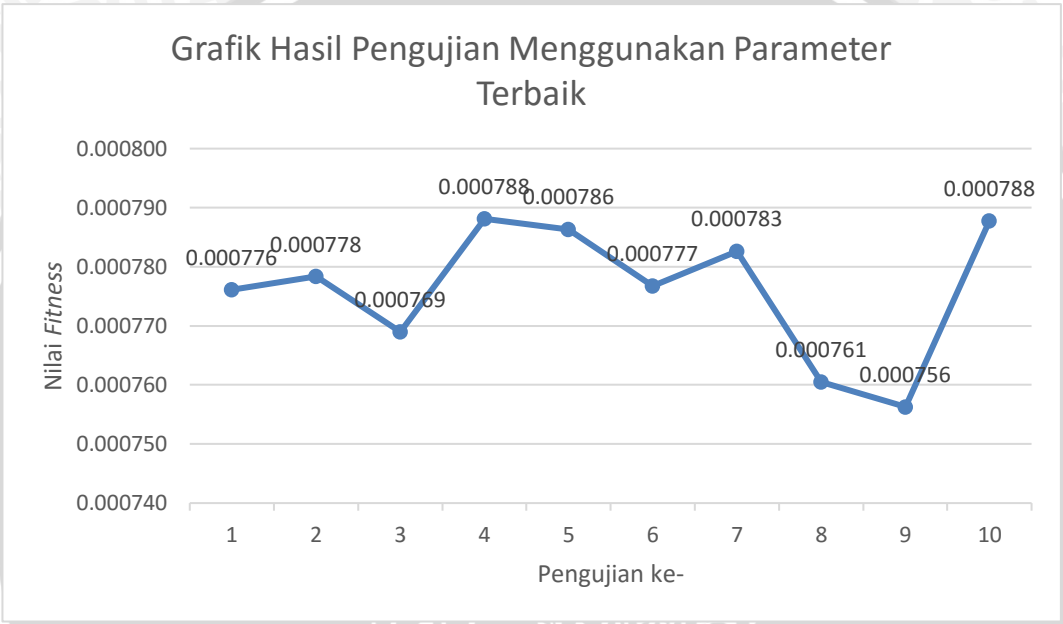
6.5 Hasil dan Pembahasan Pengujian Menggunakan Parameter Terbaik

Pengujian ini menggunakan nilai terbaik dari parameter-parameter yang telah diuji sebelumnya. Pada pengujian ini, ukuran populasi yang digunakan sebesar 300 dengan generasi sebanyak 3000 generasi. Kombinasi *crossover rate* dan *mutation rate* sebesar 0.4:0.6 dan metode seleksi yang digunakan adalah metode seleksi elitis. Pengujian dilakukan sebanyak 10 kali dan diambil satu hasil

dengan nilai *fitness* terbaik. Hasil pengujian ditunjukkan dalam Tabel 6.5 dan Gambar 6.5.

Tabel 6.5 Hasil Pengujian Menggunakan Parameter Terbaik

Fitness pada percobaan ke-									
1	2	3	4	5	6	7	8	9	10
0.000776	0.000778	0.000769	0.000788	0.000786	0.000777	0.000783	0.000761	0.000756	0.000788



Gambar 6.5 Grafik Hasil Pengujian Menggunakan Parameter Terbaik

Gambar 6.5 merupakan hasil dari pengujian menggunakan parameter terbaik yang digambarkan melalui grafik. Nilai *fitness* yang didapatkan merupakan nilai *fitness* dari masing-masing percobaan saat melakukan pengujian. Dari 10 kali percobaan, nilai fitness mengalami kenaikan dan penurunan. Nilai *fitness* terbesar dihasilkan dari pengujian ke-4 dan ke-10 dengan nilai *fitness* sebesar 0.000788. Setelah pengujian ke4, nilai *fitness* mengalami penurunan dan kenaikan sampai pengujian ke-10. Dalam pengujian ini, *fitness* mengalami naik turun tetapi tidak melebihi 0.000788 sehingga dapat disimpulkan bahwa nilai *fitness* terbaik berada pada nilai 0.000788. Jalur yang dihasilkan pada percobaan ke-4 adalah 23, 81, 29, 35, 48, 43, 75, 49, 69, 27, 53, 26, 19, 42, 73, 67, 10, 8, 14, 16, 70, 28, 55, 71, 58, 52, 13, 47, 90, 89, 6, 56, 32, 4, 51, 68, 22, 12, 77, 79, 2, 18, 98, 17, 7, 37, 87, 72, 15, 83, 59, 99, 41, 39, 66, 3, 57, 21, 76, 24, 40, 54, 9, 33, 20, 1, 82, 91, 88, 96, 92, 46, 11, 93, 100, 36, 62, 5, 84, 44, 31, 60, 85, 45, 74, 50, 80, 78, 30, 25, 65, 34, 63, 94, 97, 95, 64, 61, 86, 38 dan jalur distribusi yang dihasilkan pada percobaan ke-10 adalah 87, 13, 21, 68, 64, 42, 95, 36, 58, 67, 65, 34, 19, 100, 85, 61, 82, 59, 99, 98, 74, 76, 80, 78, 88, 83, 17,

38, 43, 96, 92, 8, 41, 4, 81, 20, 70, 26, 55, 71, 2, 53, 27, 49, 90, 23, 51, 24, 57, 40, 28, 25, 30, 12, 29, 35, 62, 5, 91, 46, 10, 89, 72, 50, 63, 94, 6, 56, 31, 73, 77, 33, 48, 44, 16, 86, 15, 18, 37, 93, 11, 84, 14, 45, 22, 1, 52, 47, 75, 3, 79, 9, 32, 7, 97, 60, 69, 39, 66, 54.

BAB 7 KESIMPULAN DAN SARAN

7.1 Kesimpulan

Kesimpulan yang dapat diambil dari skripsi ini antara lain:

1. Permasalahan VRPTW dapat diselesaikan dengan menerapkan algoritma genetika untuk mencari rute distribusi yang optimal. Algoritma genetika membangkitkan individu-individu yang terdiri dari *node-node* pelanggan secara acak sehingga didapatkan rute-rute distribusi. Setelah membangkitkan individu-individu secara acak, maka algoritma genetika menghitung nilai *fitness* setiap individu dan menyeleksi individu-individu tersebut sehingga didapatkan rute distribusi yang paling optimal berdasarkan nilai *fitness*.
2. Berdasarkan pengujian terhadap metode seleksi, metode seleksi elitis lebih unggul dengan rata-rata *fitness* sebesar 0.00059325420 jika dibanding dengan metode seleksi *binary tournament* yang hanya menghasilkan rata-rata *fitness* sebesar 0.00027986203. Metode seleksi elitis lebih unggul dikarenakan metode ini mempertahankan individu-individu yang mempunyai nilai *fitness* tertinggi untuk lolos ke generasi selanjutnya. Sedangkan, metode seleksi *binary tournament* membandingkan nilai *fitness* dua individu yang dipilih secara acak sehingga memungkinkan untuk individu dengan nilai *fitness* rendah untuk lolos ke generasi selanjutnya atau individu dengan nilai *fitness* tinggi tetapi tidak terpilih untuk mengikuti seleksi.
3. Nilai *fitness* yang dihasilkan dalam proses algoritma genetika bergantung pada parameter genetika yang diinputkan oleh pengguna. Pengujian dilakukan untuk mengetahui pengaruh tiap parameter terhadap nilai *fitness* yang dihasilkan. Pada pengujian ukuran populasi didapatkan bahwa nilai *fitness* mengalami kenaikan dari ukuran populasi 50 dan berhenti pada puncaknya di 300 lalu menurun di ukuran populasi 350. Nilai *fitness* yang didapatkan pada ukuran populasi 300 sebesar 0.0005065. Pengujian juga dilakukan untuk mencari banyaknya generasi yang menghasilkan nilai *fitness* terbesar. Nilai *fitness* mengalami kenaikan dari generasi 500 sampai generasi 3000 dan mengalami penurunan pada generasi 3500 sehingga banyaknya generasi yang optimal adalah 3000 dengan nilai *fitness* sebesar 0.0007550.
4. Kombinasi parameter genetika *crossover rate* dan *mutation rate* mempengaruhi hasil nilai *fitness*. Pada pengujian yang dilakukan dengan mengkombinasikan nilai Cr dan Mr didapatkan bahwa kombinasi terbaik adalah *crossover rate* sebesar 0.4 dan *mutation rate* sebesar 0.6 yang menghasilkan nilai *fitness* sebesar 0.0005951655.
5. Pada pengujian menggunakan parameter terbaik menghasilkan nilai *fitness* 0.000788. Nilai *fitness* yang dihasilkan merupakan nilai yang tertinggi jika dibandingkan dengan pengujian-pengujian sebelumnya, hal ini terjadi

karena pengujian ini menggunakan parameter-parameter terbaik yang didapatkan melalui pengujian sebelumnya.

7.2 Saran

Aplikasi ini dapat dikembangkan lebih lanjut dengan menambahkan jumlah pelanggan maupun memperlebar *time window* dan dapat dikembangkan dengan membandingkan lebih banyak metode seleksi sehingga hasil yang didapatkan dapat lebih bervariasi.

DAFTAR PUSTAKA

Harun, I.A, Mahmudy, W.F, dan Yudistira, N., Implementasi *Evolution Strategies* untuk Penyelesaian *Vehicle Routing Problem With Time Windows* pada Distribusi Minuman Soda XYZ, 2014. Jurnal Skripsi Mahasiswa PTIIK Universitas Brawijaya.

Sundarningsih, D., Mahmudy, W.F, & Sutrisno, Penerapan Algoritma Genetika Untuk Optimasi *Veehicle Routing Problem With Time Window* (VRPTW) : Studi Kasus Air Minum Kemasan, 2015. Jurnal Skripsi Mahasiswa PTIIK Universitas Brawijaya.

Saputri, M.W, Mahmudy, W.F & Ratnawati, D.E, Optimasi *Vehicle Routing Problem with Time Window* (VRPTW) Menggunakan Algoritma Genetika Pada Distribusi Barang, 2015. Jurnal Skripsi Mahasiswa PTIIK Universitas Brawijaya.

Mahmudy, W.F, Modul Algoritma Evolusi, 2013. Modul Kulia Semester Ganjil 2013-2014 PTIIK Universitas Brawijaya.

Gandhi, S., Khan, D., & Solanki, V.S, *A Comparative Analysis of Selection Scheme*, 2012. International Journal od Soft Computing and Engineering (IJSCE).

Li, P. dkk, *Vehicle Routing Problem with Soft Time Windows Based on Improved Genetic Algorithm for Fruits and Vegetables Distribution,* 2015. Hindawi Publishing Corporation.

Nazif, H., Lee, L.S, *Optimized Crossover Genetic Algorithm for Vehicle Routing Problem with Time Windows*, 2010. American Journal of Applied Sciences.

Puljic, K., Manger, R., *A Distributed Evolutionary Algorithm With A Superlinear Speedup For Solving The Vehicle Routing Problem,* 2012. Computing and informatics.

Hlaing, Z.C.S.S, Khine, M.A., *Solving Traveling Salesman Problem by Using Improved And Colony Optimization Algorithm*, 2011. International Journal of Information and Education Technology.

Kumar, A., *Encoding Schemes in Genetic Algorithm*, 2013. International Journal of Advanced Research in IT and Engineering.

Lukas, S., Anwar. T., Yuliani, W., Penerapan Algoritma Genetika Untuk *Traveling Salesman Problem* Dengan Menggunakan Metode *Order Crossover* dan *Insertion Mutation*, 2005. Seminar Nasional Aplikasi Teknologi Informasi 2005 (SNATI 2005).

Alabsi, F., Naoum, R., *Comparison of Selection Methods and Crossover Operations using Steady State Genetic Based Intrusion Detection System*, 2012. Journal of Emerging Trends in Computing and Information Sciences.

Mahmudy, Wayan Firdaus, *Improved Simulated Annealing For Optimization Of Vehicle Routing Problem With Time Windows (VRPTW),* 2014. *Kursor Journal,* vol. 7, no. 3, page 109-116.

Hasan, B. H. F., Saleh, M. S. M., *Evaluating the Effectiveness of Mutation Operators on the Behavior of Genetic Algorithms Applied to Non-deterministic Polynomial Problems*, 2011. *Informatica* 35 (2011) 513-518.

Soni, N., Kumar, Dr. T., *Study of Various Mutation Operators in Genetic Algorithms*, 2014. (IJCSIT) *International Journal of Computer Science and Information Technologies*, Vol. 5 (3) , 2014, 4519-4521.