UNIVERSITAS INDONESIA

APLIKASI ALGORITMA GENETIKA HIBRIDA PADA *VEHICLE ROUTING PROBLEM WITH TIME WINDOWS*

SKRIPSI

SRI ASTUTI
0806325743

FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM
PROGRAM STUDI SARJANA MATEMATIKA
DEPOK
JUNI 2012

UNIVERSITAS INDONESIA

APLIKASI ALGORITMA GENETIKA HIBRIDA PADA *VEHICLE ROUTING PROBLEM WITH TIME WINDOWS*

SKRIPSI
Diajukan sebagai salah satu syarat untuk memperoleh gelar sarjana sains

SRI ASTUTI
0806325743

FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM
PROGRAM STUDI SARJANA MATEMATIKA
DEPOK
JUNI 2012

HALAMAN PERNYATAAN ORISINALITAS

Skripsi ini adalah hasil karya sendiri,

dan semua sumber yang dikutip maupun dirujuk

telah saya nyatakan dengan benar

Nama	: Sri Astuti
NPM	: 080632573
Tanda Tangan	: 
Tanggal	: 20 Juni 2012

HALAMAN PENGESAHAN

Skripsi ini diajukan oleh :

Nama	:	Sri Astuti
NPM	:	080632573
Program Studi	:	Matematika
Judul Skripsi	:	Aplikasi Algoritma Genetika Hibrida pada *Vehicle Routing Problem with Time Windows*

Telah berhasil dipertahankan di hadapan Dewan Penguji dan diterima sebagai bagian persyaratan yang diperlukan untuk memperoleh gelar Sarjana Sains pada Program Studi S1 Matematika, Fakultas Matematika dan Ilmu Pengetahuan Alam Universitas Indonesia.

DEWAN PENGUJI

Pembimbing I	: Rahmi Rusin, S.Si, M.ScTech	( )
Pembimbing II	: Dhian Widya, S.Si, M.Kom	( )
Penguji I	: Prof. Dr. Djati Kerami	( )
Penguji II	: Dr. Yudi Satria, M.T.	( )
Penguji III	: Helen Burhan, M.Si	( )

Ditetapkan di : Depok
Tanggal : 20 Juni 2012

KATA PENGANTAR

Alhamdulillah, puji syukur kepada Allah SWT atas segala rahmat dan hidayah yang telah diberikan sehingga penulis dapat menyelesaikan tugas akhir ini. Tidak lupa juga sholawat serta salam penulis panjatkan kepada Nabi Muhammad SAW. Penulis sadar bahwa tugas akhir ini dapat terselesaikan berkat bantuan dan dukungan dari berbagai pihak. Oleh karena itu penulis ingin mengucapkan terima kasih kepada pihak-pihak yang memberikan bantuan dan dukungan selama penulis menjalani perkuliahan dan juga dalam penyelesaian tugas akhir ini. Ucapan terima kasih ditujukan kepada :

1. Rahmi Rusin, S.Si, M.ScTech dan Dhian Widya, S.Si, M.Kom selaku pembimbing I dan pembimbing II yang telah meluangkan waktu dan pikiran di tengah kesibukannya, dan selalu memberikan semangat kepada penulis untuk menyelesaikan tugas akhir ini.
2. Dr. Yudi Satria, M.T selaku Ketua Departemen, Rahmi Rusin, S.Si, M.ScTech selaku Sekretaris Departemen, dan Dr. Dian Lestari DEA selaku Koordinator Pendidikan yang telah membantu proses penyelesaian tugas akhir ini.
3. Dra. Netty Sunandi M.Si selaku pembimbing akademik yang telah meluangkan waktunya untuk memberikan saran kepada penulis selama proses perkuliahan hingga penyelesaian tugas akhir ini.
4. Seluruh Staf Pengajar di Departemen Matematika UI atas ilmu yang diberikan kepada penulis.
5. Seluruh Karyawan di Departemen Matematika UI atas bantuan yang diberikan kepada penulis.
6. Mba Santi yang selalu membantu Penulis dkk untuk menyelesaikan administrasi Seminar.
7. Keluarga tercinta (Mama, Bapak, Mba Iya dan Mas Hendrik, Mba Ani dan Mas Zaenal, Septi) atas dukungan, semangat dan kasih sayang yang diberikan kepada penulis.

8. *The little ones in my family* (Rijal) yang memberi semangat baru kepada penulis.
9. Teman - teman Matematika 2008 yang menemani penulis mengarungi kehidupan kampus, memberi warna baru pada kehidupan penulis (Risya, Ines, Numa, Mei, Mella, Nadia, Qiqi, Nita, Ade, Awe, Dhila, Hindun, Dheni, Anisah, Arkies, Maul, Uni Achi, Aya, Bang Andy, Adhi, Bowo, Ucil, Vika, Cindy, Resti, Siwi, Emy, Janu, Juni, Dhea, Oline, puput, purwo, Ega, Eka, Icha, Sita, Luthfa, Citra, Hendry, Yulial, Arief, Fani, Wulan, Zee, Ifah, Umbu, Arman, Uchid, Agnes, Danis, Dede, Dhewe, Dian, Dini, Nora, Yulian, Agy, Maimun, Masykur, Dheni)
10. Kakak-kakak yang berjuang bersama Ka Anis, Ka Nora, Ka Ayat, Ka Fauzan, Ka Putri, Ka Manda, Ka Mita, Ka Sica.
11. Numa, Ines, Risya, Cindy, Qiqi, Ade, Dhila, Sita, Hindun, Bang Andy, Adhi, Arief, Awe, Dheni atas semangat, bantuan, dan kebersamaan yang dilalui. Bismillah semoga ikatan ini sampai ke syurga, aamiin.
12. Andy Marhadi Sutanto yang selalu memberikan semangat, dukungan dan bantuan kepada penulis untuk tetap yakin dapat menyelesaikan tugas akhir ini.
13. Anak Kos Kauhan (Ines, Numa, Yiska, Vely, Upi, Prisna) + kos depan kauhan (Risya) atas gaulannya di *coffee shop* kutek a.k.a warkop.
14. Dian, Azhar, Wenang, Kiki, Dwi a.k.a fitri, terimakasih atas jalan-jalan dan canda tawa yang dilalui bersama dari SMA hingga sekarang.
15. Ria, Wina, dan Icha yang membantu penulis memahami materi dalam menyelesaikan tugas akhir ini.
16. Komunitas HosMate dan MEC, semoga apa yang akan kita lakukan nanti bermanfaat bagi diri sendiri dan sesama, aamiin.
17. Dinda dan Ai (2009) yang memberikan dorongan semangat kepada penulis.
18. Seluruh keluarga besar Matematika angkatan 2009, 2010, 2011, dan 2012
19. Semua pihak yang membantu penulis yang tidak dapat disebutkan satu persatu.

Akhir kata, penulis memohon maaf atas kesalahan dan kekurangan pada tugas akhir ini. Semoga tugas akhir ini dapat memberikan manfaat bagi penulis dan pembaca.

Penulis

2012

HALAMAN PERNYATAAN PERSETUJUAN PUBLIKASI
TUGAS AKHIR UNTUK KEPENTINGAN AKADEMIS

Sebagai sivitas akademik Universitas indonesia, saya yang bertanda tangan di bawah ini :

Nama	: Sri Astuti
NPM	: 0806325743
Program Studi	: Sarjana
Departemen	: Matematika
Fakultas	: MIPA (Matematika dan Ilmu Pengetahuan Alam)
Jenis karya	: Skripsi

demi pengembangan ilmu pengetahuan, menyetujui untuk memberikan kepada Universitas Indonesia Hak Bebas Royalti Noneksklusif (*Non-exclusive Royalty Free Right*) atas karya ilmiah saya yang berjudul :
Aplikasi Algoritma Genetika Hibrida pada *Vehicle Routing Problem with Time Windows*.

beserta perangkat yang ada (jika diperlukan). Dengan Hak Bebas Royalti Noneksklusif ini Universitas Indonesia berhak menyimpan, mengalihmedia/format-kan, mengelola dalam bentuk pengkalan data (database), merawat, dan memublikasikan tugas akhir saya selama tetap mencantumkan nama saya sebagai penulis/pencipta dan pemilik Hak Cipta.
Demikian surat ini saya buat dengan sebenarnya.

Dibuat di : Depok
Tanggal : 20 Juni 2012
Yang Menyatakan

(Sri Astuti)

ABSTRAK

Nama : Sri Astuti
Program Studi : Matematika
Judul : Aplikasi Algoritma Genetika Hibrida pada *Vehicle Routing Problem with Time Windows*

Vehicle Routing Problem with Time Windows (VRPTW) adalah masalah penentuan rute kendaraan dalam pendistribusian barang/jasa ke sejumlah pelanggan yang memiliki biaya minimum dengan tambahan kendala *time windows*, biaya direpresentasikan oleh total jarak yang ditempuh kendaraan dari depot dan kembali ke depot. Pada tugas akhir ini, digunakan algoritma genetika hibrida untuk menyelesaikan VRPTW. 50% populasi awal dibentuk dengan menggunakan metode *Push Forward Insertion Heuristic* (PFIH) dilanjutkan dengan *λ-Interchange*, dan 50% lainnya dibentuk secara acak. Tiga operator utama algoritma genetika yang digunakan adalah *ranking based selection*, *merge-heuristic crossover*, dan *sequence based mutation*. Pada tugas akhir ini juga akan diimplementasikan algoritma genetika hibrida pada VRPTW dengan perangkat lunak.

Kata Kunci : *Vehicle Routing Problem with Time Windows* (VRPTW), Algoritma Genetika Hibrida, PFIH, *λ-Interchange*, *ranking based selection, merge-heuristic crossover*, *sequence based mutation.*
xii + 64 halaman ; 14 gambar, 12 Tabel
Daftar Pustaka : 15 (1993 - 2010)

ABSTRACT

Name : Sri Astuti
Program Study : Matematika
Tittle : An Application of Hybrid Genetic Algorithm for Vehicle Routing Problem with Time Windows

Vehicle Routing Problem with Time Windows (VRPTW) is a problem of determining the route of vehicles that has minimum cost in the distribution of goods /services to a number of customers with addition of time constraint, the cost is represented by the total distance traveled by vehicles from depot and returned to depot. In this final project, a hybrid genetic algorithm used to solve VRPTW. 50% of initial population is generated by Push Forward Insertion Heuristic (PFIH) and then λ-Interchange, and the other 50% is randomly generated. Three major operator that used in this final project are ranking based selection, merge-heuristic crossover, and sequence based mutation. Hybrid genetic algorithm is implemented on Solomon's benchmark data of VRPTW.

Keywords : Vehicle Routing Problem with Time Windows (VRPTW), Hybrid Genetic Algorithm, PFIH, λ-Interchange, ranking based selection, merge-heuristic crossover, sequence based mutation
xii + 64 pages ; 14 pictures, 12 Tables
Bibliography : 15 (1993 - 2010)

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

DAFTAR LAMPIRAN

BAB 1
PENDAHULUAN

1.1 Latar Belakang

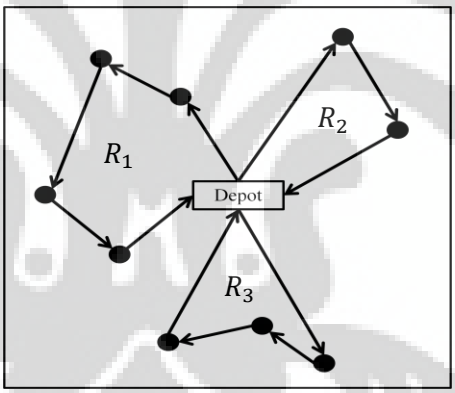
Transportasi merupakan salah satu faktor pendukung aktifitas manusia. Alat transportasi mempermudah kita untuk berpindah dari suatu lokasi ke lokasi lain, seperti pergi kerja, kuliah, sekolah, berbelanja, dan juga dalam pendistribusian barang/jasa, dengan alat transportasi pendistribusian barang/jasa menjadi lebih cepat dan mudah. Dalam bidang distribusi barang/jasa, aspek yang diperhatikan adalah bagaimana cara meminimumkan biaya pendistribusian barang/jasa tersebut dan tujuan pendistribusian barang tercapai, salah satu cara untuk meminimumkan biaya adalah menentukan rute optimal kendaraan. Masalah penentuan rute optimal kendaraan ini dalam bidang matematika selanjutnya disebut *Vehicle Routing Problem* (VRP).

VRP adalah masalah kombinatorial yang cukup dikenal dalam bidang Riset Operasi, yaitu masalah penentuan rute pendistribusian barang/jasa ke pelanggan-pelanggan dengan lokasi yang berbeda dengan permintaan yang sudah diketahui, dari satu atau lebih depot dan memenuhi beberapa kendala (Yeun dkk, 2008). VRP memiliki tujuan meminimumkan biaya yang dapat direpresentasikan oleh total jarak tempuh atau jumlah kendaraan yang digunakan atau keduanya. Masalah ini dapat digambarkan dalam graf $G(V, A)$ dengan $V = \{0, 1, 2.., n\}$ merupakan himpunan simpul yang menggambarkan pelanggan-pelanggan dengan simpul 0 merupakan depot, sedangkan A merupakan himpunan busur yang menghubungkan depot dengan pelanggan dan juga menghubungkan antar pelanggan (Laporte 1992). Aplikasi dari masalah VRP dapat dilihat di kehidupan sehari-hari, seperti pengiriman koran kepada pembeli yang berlangganan, penentuan rute bus sekolah, pengiriman surat dari kantor pos, dan lain lain.

Penelitian mengenai VRP telah dilakukan sejak akhir tahun 50-an setelah Dantzig dan Ramser memberi penjelasan bahwa masalah ini merupakan generalisasi dari *Traveling Salesman Problem* (TSP) (Yeun dkk, 2008). Beberapa

contoh masalah yang merupakan VRP antara lain *Capacitated Vehicle Routing Problem* (CVRP), *Vehicle Routing Problem with Pick-Up and Delivery* (VRPPD), *Dynamic Vehicle Routing Problem* (DVRP) dan *Vehicle Routing Problem with Time Windows* (VRPTW). VRPTW merupakan VRP yang diikuti kendala *time windows* dan kapasitas kendaraan.

Solusi layak untuk masalah VRPTW adalah himpunan rute yang dilayani oleh sejumlah kendaraan dari depot menuju pelanggan-pelanggan dengan permintaan yang sudah diketahui dan tidak melanggar kendala *time windows* dan kapasitas kendaraan. Solusi VRPTW dapat digambarkan dalam bentuk graf, dimana rute perjalanan merupakan lintasan yang berawal dari satu simpul (depot), kemudian menghubungkan simpul-simpul pelanggan dalam rute sesuai dengan urutan kunjungan hingga kembali lagi ke simpul awal. Penggambaran solusi layak dapat dilihat pada Gambar 1.1 dimana pada gambar tersebut terdapat 3 rute (R_1, R_2, R_3) yang melayani 9 pelanggan.



[Sumber : Ghoseiri, 2009]

Gambar 1.1 Contoh Solusi Layak Masalah VRPTW

VRPTW merupakan masalah kombinatorial yang termasuk ke dalam NP-*hard problem*, sehingga untuk masalah yang besar sulit diselesaikan dengan metode eksak. Sejak akhir tahun 50-an banyak penelitian telah dilakukan untuk menyelesaikan masalah VRPTW, metode yang paling sering digunakan adalah metode heuristik. Metode heuristik adalah teknik pendekatan untuk menyelesaikan suatu masalah. Metode ini menghasilkan solusi yang mendekati solusi eksak dengan waktu komputasi yang lebih singkat dari metode eksak.

Simulated Annealing (SA), *Tabu Search* (TS), *Ant Colony Optimization* (ACO), Algoritma Genetika (AG), dan *Particle Swarm Optimization* (PSO) merupakan beberapa contoh dari metode heuristik yang sering digunakan untuk menyelesaikan masalah VRPTW. Dalam tugas akhir ini akan digunakan metode AG untuk menyelesaikan masalah VRPTW. Metode AG diadaptasi dari konsep bagian terkecil dari makhluk hidup yaitu sel yang terdiri dari sekumpulan kromosom, dimana setiap kromosom terdiri dari gen-gen yang menggambarkan karakteristik makhluk hidup (bentuk wajah, warna rambut, tinggi badan, dan lain-lain) yang dapat mengalami proses persilangan dan mutasi.

Berdasarkan konsep tersebut, algoritma AG terbagi dalam empat tahap. Tahap pertama adalah pembentukan solusi awal yang dapat dilakukan dengan beberapa metode heuristik ataupun acak. Tahap kedua merupakan tahap seleksi untuk memilih orang tua berdasarkan nilai *fitness* yang ditentukan oleh sebuah fungsi. Tahap selanjutnya dilakukan persilangan terhadap orang tua yang telah terpilih untuk membangun populasi baru yang disebut teknik *crossover*. Tahap terakhir dalam metode AG adalah melakukan mutasi terhadap solusi yang didapat. Pada AG, rute perjalanan dapat direpresentasikan dalam bentuk kromosom, dimana gen-gen dalam kromosom menggambarkan pelanggan-pelanggan yang harus dilayani, sedangkan urutan gen pada kromosom menggambarkan urutan pelanggan yang harus dikunjungi lebih dulu pada suatu rute.

Pada tugas akhir ini, pembentukan solusi awal dilakukan dengan menggabungkan metode heuristik dan acak. Metode heuristik yang digunakan adalah metode *Push Forward Insertion Heuristic* (PFIH) yang dilanjutkan dengan metode λ-*Interchange*. Proses ini digunakan untuk membentuk 50% solusi layak awal, sementara 50% lainnya dibangun secara acak. Karena penggunaan metode PFIH dalam algoritma AG, maka metode ini dikenal sebagai Algoritma Genetika Hibrida.

1.2 Perumusan Masalah dan Ruang Lingkup

Perumusan masalah dalam tugas akhir ini adalah bagaimana menjelaskan aplikasi algoritma genetika hibrida pada VRPTW.

Ruang Lingkup masalah dalam penulisan tugas akhir ini adalah implementasi algoritma genetika hibrida pada data *benchmark Solomon*.

1.3 Metode Penelitian

Penelitian dilakukan dengan studi literatur dan simulasi

1.4 Tujuan Penelitian

Tujuan dari penulisan skripsi ini adalah untuk menjelaskan aplikasi algoritma genetika hibrida pada VRPTW dan mengimplementasikan algoritma tersebut pada data *benchmark Solomon* menggunakan perangkat lunak.

BAB 2
LANDASAN TEORI

2.1 *Vehicle Routing Problem* (VRP)

Vehicle Routing Problem (VRP) didefinisikan sebagai masalah penentuan rute optimal kendaraan untuk pendistribusian barang/jasa ke pelanggan-pelanggan dengan lokasi yang berbeda dengan permintaan yang sudah diketahui, dari satu atau lebih depot yang memenuhi beberapa kendala (Yeun dkk, 2008). VRP memiliki tujuan meminimumkan biaya yang direpresentasikan oleh total jarak tempuh kendaraan-kendaraan yang digunakan untuk melayani pelanggan-pelanggan.

Masalah ini merupakan generalisasi dari *m-Traveling Salesman Problem* (*m*-TSP) dimana pada TSP diberikan himpunan *N* kota dan seorang *salesman* yang ingin menemukan jalur terpendek untuk mengunjungi setiap kota tepat satu kali dan selesai di kota asal (Ho, Lim, & Oon, 2001), sementara pada *m*-TSP terdapat *m salesman* yang akan mengunjungi *N* kota tepat satu kali. Pada VRP, kota-kota di *m*-TSP merupakan pelanggan-pelanggan yang memiliki permintaan terhadap barang/jasa, *salesman* pada *m*-TSP merupakan kendaraan, dimana masing-masing kendaraan memiliki kapasitas tertentu sehingga total permintaan dari pelanggan-pelanggan pada suatu rute tidak boleh melebihi kapasitas kendaraan yang melayani rute tersebut, dan setiap pelanggan dalam VRP hanya boleh dikunjungi satu kali. Karena kendala kapasitas inilah VRP sering disebut sebagai *Capacitated Vehicle Routing Problem* (CVRP). Pada kedua masalah ini, perjalanan dimulai dan diakhiri pada kota/pelanggan yang sama. Pada VRP kota/pelanggan tempat kendaraan berangkat dan kembali disebut depot jadi setiap kendaraan harus berangkat kemudian melayani pelanggan-pelanggan sesuai rute yang ditentukan dan kembali lagi ke depot.

Selain CVRP, terdapat beberapa contoh masalah VRP, yaitu :

- *Vehicle Routing Problem with Pickup and Delivery* (VRPPD) merupakan VRP dengan permintaan yang terdiri dari jemput dan antar.

- *Dynamic Vehicle Routing Problem* (DVRP) merupakan VRP yang terdapat penambahan pelanggan baru saat kendaraan sedang melayani pelanggan.
- *Vehicle Routing Problem with Time Windows* (VRPTW) merupakan CVRP dengan penambahan kendala waktu (*time windows*) pada masing-masing pelanggan dan depot.

Pada tugas akhir ini akan dibahas mengenai masalah VRPTW yang akan dijelaskan di Subbab 2.2 berikut.

2.2 *Vehicle Routing Problem with Time Windows* (VRPTW)

VRPTW adalah masalah penentuan rute kendaraan dengan biaya minimum untuk melayani seluruh pelanggan dan memenuhi kendala kapasitas kendaraan dan *time windows* pada masing-masing pelanggan dan depot. *Time window* pada depot didefinisikan sebagai selang waktu kendaraan berangkat dan kembali lagi ke depot. *Time window* pada depot disebut juga *scheduling horizon*, yaitu setiap kendaraan tidak boleh meninggalkan depot sebelum waktu awal depot dimulai dan harus kembali ke depot sebelum waktu akhir depot selsesai. Sedangkan *time window* pada masing-masing pelanggan didefinisikan sebagai selang waktu dimana kendaraan dapat memulai pelayanan di pelanggan, dengan kata lain, kendaraan dapat memulai pelayanan setelah waktu awal pelanggan dimulai dan sebelum waktu akhir pelanggan selesai. Jika kendaraan datang sebelum waktu awal pelanggan, maka kendaraan harus menunggu sampai tiba waktu awal pelanggan dapat dilayani, dan kendaraan yang menunggu tidak dikenakan biaya tambahan. Terdapat dua tipe *time windows* pada VRPTW, yaitu *hard time windows* dan *soft time windows*. Pada *hard time windows* kendaraan harus tiba di pelanggan sebelum waktu akhir pelanggan. Kendaraan yang datang setelah waktu akhir pelanggan disebut *tardy*. Pada *soft time windows* kendaraan boleh datang setelah waktu akhir pelanggan dan dikenakan biaya tambahan. (Solomon dkk, 2005).

Selanjutnya akan dijelaskan mengenai model matematika untuk VRPTW (Amini, 2010). Dalam memodelkan VRPTW diperlukan beberapa asumsi, berikut merupakan asumsi-asumsi yang digunakan :

- Terdapat satu depot dan sejumlah kendaraan di depot. Kendaraan-kendaraan tersebut digunakan untuk melayani pelanggan-pelanggan dan memiliki kapasitas yang sama.
- Banyaknya kendaraan di depot tidak dibatasi
- Setiap pelanggan hanya dikunjungi satu kali
- Rute perjalanan setiap kendaraan harus dimulai dan kembali ke depot dengan tidak melanggar *time window* depot
- *Time windows* yang digunakan adalah *hard time windows*
- Kecepatan kendaraan konstan, dan masalah kemacetan maupun gangguan lainnya selama perjalanan (kecelakaan, ditilang, dan lain-lain) diabaikan
- Jumlah permintaan pelanggan-pelanggan pada suatu rute tidak boleh melebihi kapasitas kendaraan pada rute tersebut
- Setiap pelanggan harus dilayani dalam *time windows* yang telah ditentukan

Masalah VRPTW dapat digambarkan oleh graf $G(V, A)$, dimana $V = \{0,1,2, \ldots, n\}$ merupakan himpunan simpul yang merepresentasikan pelanggan-pelanggan yang akan dilayani dengan permintaan yang sudah diketahui dan depot berada di simpul 0, sedangkan A merupakan himpunan busur pada graf yang menghubungkan depot dengan pelanggan dan juga menghubungkan antar pelanggan (Laporte, 1992).

Misalkan terdapat n pelanggan, $V = \{0, 1, 2, \ldots, n\}$ dengan 0 merupakan depot. Misalkan $i, j \in V$, jarak antar pelanggan i ke pelanggan j dinotasikan dengan d_{ij}, sedangkan waktu tempuh kendaraan dari pelanggan i ke pelanggan j dinotasikan dengan t_{ij}. Karena kecepatan kendaraan konstan, maka d_{ij} proporsional terhadap t_{ij}. *Time window* pelanggan i dinotasikan dengan $[a_i, b_i]$ dimana a_i merupakan waktu awal pelanggan i dan b_i merupakan waktu akhir pelanggan i, sedangkan *time window* pada depot dinotasikan dengan $[a_0, b_0]$ dan $a_0 = 0$. Setiap pelanggan i memiliki waktu pelayanan s_i dan permintaan permintaan atas barang/jasa sebanyak q_i. Kendaraan-kendaraan yang digunakan untuk melayani permintaan pelanggan-pelanggan memiliki kapasitas yang sama, yaitu Q.

Misalkan K merupakan himpunan kendaraan yang tersedia di depot. Untuk setiap kendaraan k dan setiap jalur yang menghubungkan pelanggan i dan pelanggan j, dimana $i \neq j$ dan $i, j \neq 0$, didefinisikan variabel keputusan x_{ijk} sebagai berikut :

$$x_{ijk} = \begin{cases} 1 & \text{jika kendaraan } k \text{ melayani pelanggan } j \text{ setelah pelanggan } i \\ 0 & \text{lainnya} \end{cases} \quad (2.1)$$

Selain itu didefinisikan pula y_{jk}, yaitu waktu kendaraan k memulai pelayanan di pelanggan j. Karena diasumsikan $a_0 = 0$, maka $y_{0k} = 0$ untuk setiap kendaraan k.

Jika kendaraan k melayani pelanggan j setelah melayani pelanggan i, maka waktu mulai pelayanan pelanggan j bergantung pada waktu mulai pelayanan pelanggan i, lamanya waktu pelayanan pelanggan i, dan waktu tempuh dari pelanggan i ke pelanggan j. Jika kendaraan datang di pelanggan j di dalam *time window* pelanggan j, maka pelanggan j langsung dilayani, sedangkan jika kendaraan datang di pelanggan j sebelum waktu awal pelanggan j, maka kendaraan harus menunggu dan waktu mulai pelayanan akan sama dengan waktu awal pelanggan j. Jadi waktu mulai pelayanan pelanggan $j, y_{jp} = \max\{a_j, y_{ip} + s_i + t_{ij}\}$ dengan $y_{ip} + s_i + t_{ij}$ merupakan waktu kedatangan pelanggan j setelah melayani pelanggan i.

Berikut merupakan model matematika untuk VRPTW :

$$\min z = \sum_{k \in K} \sum_{i \in V} \sum_{j \in V} c_{ij} x_{ijk} \quad (2.2)$$

d.s

$$\sum_{k \in K} \sum_{j \in V} x_{ijk} = 1 \qquad \forall i \in V - \{0\} \quad (2.3)$$

$$\sum_{i \in V} q_i \sum_{j \in V} x_{ijk} \leq Q \qquad \forall k \in K \quad (2.4)$$

$$\sum_{j \in V} x_{0jk} = 1 \qquad \forall k \in K \quad (2.5)$$

$$\sum_{i \in V} x_{ihk} - \sum_{j \in V} x_{hjk} = 0 \qquad \forall h \in V, \forall k \in K \quad (2.6)$$

$$\sum_{i \in V} x_{i0k} = 1 \qquad \forall k \in K \qquad (2.7)$$

$$x_{ijk}\left(y_{ik} + s_i + t_{ij} - y_{jk}\right) \leq 0 \qquad \forall i, j \in V, \forall k \in K \qquad (2.8)$$

$$a_i \leq y_{ik} \leq b_i \qquad \forall i \in V \ , \forall k \in K \qquad (2.9)$$

$$x_{ijk} \in \{0,1\} \qquad \forall i, j \in V, \forall k \in K \qquad (2.10)$$

Model di atas merupakan model pemrograman bilangan bulat biner yang bertujuan meminimumkan total biaya perjalanan, c_{ij} menyatakan biaya yang dikeluarkan yang merupakan jarak tempuh kendaraan dari pelanggan i menuju pelanggan j. Kendala (2.3) memastikan bahwa setiap pelanggan hanya dikunjungi satu kali, dan kendala (2.4) menyatakan jumlah permintaan pelanggan-pelanggan dalam satu rute tidak melebihi kapasitas kendaraan yang melayani rute tersebut. Kendala (2.5) dan (2.7) menyatakan bahwa semua kendaraan harus memulai rute perjalanan dari depot dan kembali lagi ke depot, sementara kendala (2.6) menyatakan bahwa setelah kendaraan mengunjungi seorang pelanggan, kendaraan harus meninggalkan pelanggan tersebut menuju pelanggan lain. Kendala (2.8) menyatakan bahwa jika kendaraan k mengunjungi pelanggan j setelah pelanggan i, maka kendaraan k akan melayani pelanggan j setelah kendaraan melayani pelanggan i. Waktu kendaraan k memulai pelayanan di pelanggan j dapat dihitung dengan menjumlahkan waktu mulai pelayanan di pelanggan i, waktu pelayanan pelanggan i, dan waktu tempuh dari pelanggan i ke pelanggan j. Kendala (2.9) menyatakan waktu dimulai pelayanan untuk pelanggan i harus berada di dalam *time window* pelanggan i, dan kendala (2.10) menyatakan variabel keputusan merupakan variabel biner. Solusi layak pada VRPTW berupa sekumpulan rute yang melayani semua pelanggan dan memenuhi semua kendala yang diberikan.

Penelitian mengenai penyelesaian VRPTW telah banyak dilakukan sejak tahun 1950-an, baik menggunakan metode eksak maupun menggunakan metode heuristik atau metode pendekatan. Beberapa metode eksak yang dapat digunakan untuk menyelesaikan VRPTW antara lain *Column Generation Algorithm, Lagrangian relaxation based on Algorithm, A Branch-and-Cut Algorithm.* VRPTW merupakan NP-*Hard problem*, sehingga penyelesaian menggunakan algoritma eksak akan lebih sulit untuk jumlah pelanggan yang banyak. Oleh

karena itu digunakan metode heuristik untuk menyelesaikannya. Beberapa metode heuristik yang dapat digunakan untuk menyelesaikan VRPTW adalah *Simulated Annealing* (SA), *Tabu Search* (TS), *Ant Colony Optimization* (ACO), *Particle Swarm Optimization* (PSO) dan Algoritma Genetika (AG). Pada tugas akhir ini, VRPTW diselesaikan menggunakan AG, penjelasan mengenai AG disampaikan di Subbab 2.3 berikut.

2.3 Algoritma Genetika (AG)

Algoritma Genetika (AG) pertama kali ditemukan oleh John Holland pada tahun 1960. Bersama dengan murid dan teman-temannya, John Holland memublikasikan AG dalam buku yang berjudul *Adaption of Natural and Artificial Systems* pada tahun 1975 (Coley, 1999). AG merupakan algoritma optimisasi yang terinspirasi oleh gen dan seleksi alam. Algoritma ini mengkodekan solusi-solusi yang mungkin ke dalam struktur data dalam bentuk kromosom-kromosom dan mengaplikasikan operasi rekombinasi genetik ke struktur data tersebut (Whitley, 2002).

Kromosom terdiri dari gen-gen yang menyimpan informasi genetik dimana setiap gen mengkodekan protein khusus yang menentukan sifat genetik individu, seperti warna kulit, tipe rambut, dan lain-lain. Sifat-sifat inilah yang membedakan setiap individu dengan individu lain. Oleh karena itu, individu dapat direpresentasikan dengan kromosom yang merepresentasikan kemungkinan solusi.

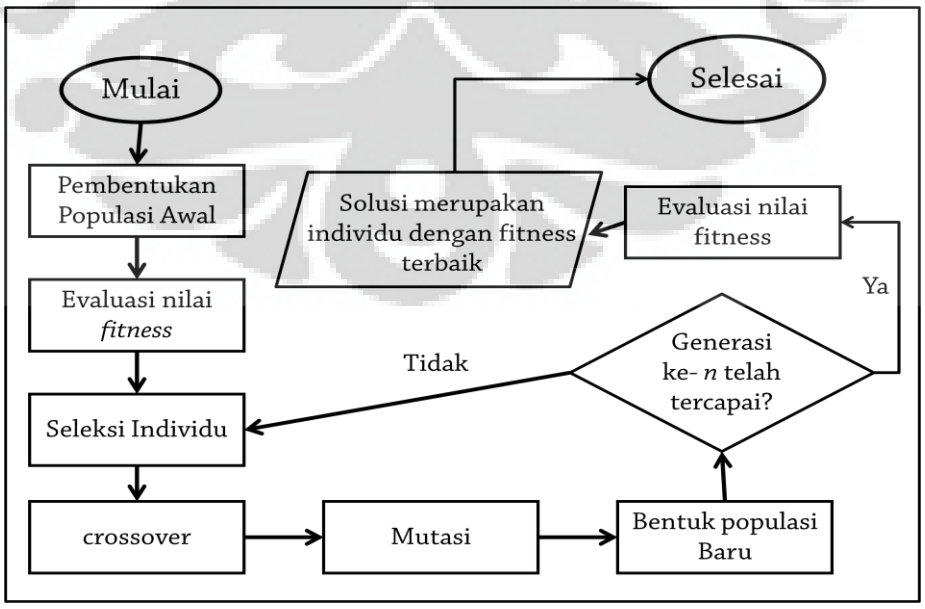
Dalam teori evolusi dikenal istilah seleksi alam, dimana individu yang baik akan bertahan sementara individu yang tidak baik akan punah. Individu dikatakan baik jika mampu bertahan hidup terhadap perubahan cuaca, dan lain-lain. Proses seleksi alam menyebabkan individu harus menyesuaikan diri dengan cara memperbaiki gen-gen melalui proses reproduksi dan menghasilkan keturunan atau anak (*offspring*) berikutnya yang membawa sifat-sifat kedua orang tuanya. AG merupakan algoritma yang mengadaptasi proses tersebut untuk memilih individu-individu yang baik. Pada algoritma ini, baik atau tidaknya suatu individu dilihat berdasarkan nilai *fitness* yang ditentukan oleh suatu fungsi. Tujuan yang

ingin dicapai pada algoritma ini adalah mendapatkan individu yang memiliki nilai *fitness* terbaik dengan cara menerapkan proses rekombinasi genetik yaitu *crossover* dan mutasi jika diperlukan. Nilai *fitness* pada umumnya dilihat dari fungsi objektif. Contohnya pada masalah VRPTW, nilai *fitness* masing-masing kromosom dihitung berdasarkan fungsi objektif, yaitu

$$z = \sum_{k \in K} \sum_{i \in V} \sum_{j \in V} c_{ij} x_{ijk}$$

yang merupakan total jarak yang ditempuh kendaraan-kendaraan untuk melayani semua pelanggan. Tiga operator utama dalam metode AG adalah seleksi, *crossover*, dan mutasi.

Algoritma ini dimulai dengan pembentukan himpunan individu yang diwakili oleh kromosom, himpunan kromosom ini disebut populasi awal. Populasi awal dapat dibentuk secara acak ataupun dengan metode heuristik (Ghoseiri & Ghanndpour, 2009). Dengan mengevaluasi nilai *fitness* masing-masing kromosom, akan dipilih sejumlah kromosom yang memiliki nilai *fitness* terbaik dan dilakukan rekombinasi genetik sehingga menghasilkan keturunan untuk pembentukan populasi baru pada generasi berikutnya. Kemudian langkah tersebut dilakukan berulang-ulang hingga mencapai solusi optimum atau setelah generasi telah sampai ke generasi ke-n. Gambar 2.1 merupakan *flowchart* AG secara umum.



Gambar 2.1 *Flowchart* Algoritma Genetika

Sebelum membentuk populasi awal, dibutuhkan representasi solusi ke dalam kromosom. Pada Subbab 2.3.1 akan dijelaskan mengenai representasi solusi ke dalam kromosom dengan pengkodean.

2.3.1. Pengkodean pada AG

Pada AG, data yang direpresentasikan oleh gen-gen dapat dikodekan dengan menggunakan beberapa metode, yaitu pengkodean bilangan biner, pengkodean permutasi, pengkodean nilai, dan pengkodean *tree*. Penggunaan metode pengkodean disesuaikan dengan masalah yang diselesaikan. Berikut akan diberikan contoh untuk masing-masing pengkodean (Obitko, 1998).

2.3.1.1 Bilangan Biner

Pada pengkodean dengan bilangan biner, setiap kromosom merupakan barisan dari bilangan biner 0 atau 1. Sebagai contoh diberikan pengkodean untuk kromosom A dan B pada Gambar 2.2.

Kromosom A	1101100101100101011100101
Kromosom B	1111111000000110000001111

[Sumber : Obitko, 1998]

Gambar 2.2 Kromosom dengan pengkodean bilangan biner

Contoh masalah yang menggunakan pengkodean dengan bilangan biner adalah *Knapsack problem*. Dalam masalah ini disajikan data barang beserta ukurannya yang akan dimasukkan ke dalam kantong (*knapsack*) dengan kapasitas yang telah ditentukan. Tujuannya adalah memaksimumkan nilai barang dalam kantong dan tidak melanggar kapasitas kantong. Bilangan biner pada kromosom merepresentasikan barang yang akan dimasukkan ke dalam kantong. Jika nilainya 1 maka barang tersebut akan dimasukkan ke dalam kantong, sementara jika nilainya 0 barang tersebut tidak dimasukkan ke dalam kantong.

2.3.1.2 Nilai

Pada pengkodean nilai, kromosom merupakan barisan dari suatu nilai, nilai dapat berupa bilangan real atau karakter. Pada Gambar 2.4 diberikan contoh untuk pengkodean nilai, kromosom A terdiri dari bilangan real, kromosom B terdiri dari barisan huruf kapital.

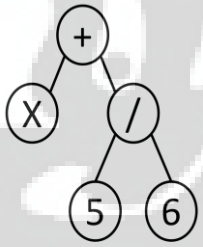
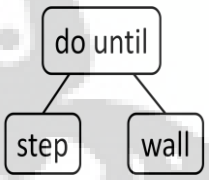
Kromosom A	1.2324 5.3243 0.4556 2.3293 2.4545
Kromosom B	ABDJEIFJDHDIERJFDLDFLFEGT

[Sumber : Obitko, 1998]

Gambar 2.3 Kromosom dengan pengkodean nilai

2.3.1.3 *Tree*

Pada pengkodean *tree*, kromosom berbentuk pohon dari suatu objek, seperti fungsi atau perintah dalam *programming*. Pada Gambar 2.5 diberikan contoh untuk pengkodean *tree*.

Kromosom A	Kromosom B
	
$(+x(/5y))$	(do_until step wall)

[Sumber : Obitko, 1998]

Gambar 2.4 Kromosom dengan pengkodean *tree*

2.3.1.4 Permutasi

Pada pengkodean permutasi, setiap kromosom merupakan barisan bilangan bulat positif, seperti yang dicontohkan pada Gambar 2.3 berikut.

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

[Sumber : Obitko, 1998]

Gambar 2.5 Kromosom dengan pengkodean permutasi

Contoh masalah yang menggunakan pengkodean ini adalah *Travelling Salesman Problem* (TSP). Pada masalah ini diberikan data berupa kota dan jarak antar kota. Tujuannya adalah menentukan rute dimana *salesman* harus mengunjungi semua kota dengan total jarak tempuh minimum. Bilangan pada kromosom merepresentasikan kota yang akan dikunjungi, sementara urutan bilangan pada kromosom merepresentasikan urutan kota yang akan dikunjungi oleh *salesman*. Karena VRP merupakan generalisasi dari *m*-TSP, sehingga pada tugas akhir ini akan digunakan pengkodean permutasi.

2.3.2. Pembentukan Populasi Awal

Populasi awal pada AG terdiri dari kromosom-kromosom yang merepresentasikan kemungkinan solusi. Pada VRPTW, populasi adalah sekumpulan kromosom yang merepresentasikan rute kendaraan-kendaraan, gen-gen pada kromosom merepresentasikan pelanggan yang akan dilayani, sedangkan urutan gen pada kromosom merepresentasikan urutan pelanggan yang akan dilayani terlebih dahulu. Banyaknya kromosom pada populasi disebut *popsize* yang disesuaikan dengan ukuran masalah yang ingin diselesaikan.

Pembentukan populasi ini dapat dilakukan dengan dua metode, yaitu acak dan heuristik. Salah satu metode heuristik yang dapat digunakan untuk membentuk populasi awal adalah metode *Push Forward Insertion Heuristic* (PFIH). Penggunaan metode heuristik pada AG dapat mempercepat waktu pencarian solusi karena populasi awal akan mendekati solusi akhir (Ghoseiri dkk, 2009). Pada skripsi ini akan digunakan metode heuristik dan acak, metode heuristik yang digunakan akan dijelaskan pada Bab 3.

Setelah terbentuk populasi, dilakukan proses seleksi dimana kromosom-kromosom orang tua dipilih untuk memasuki proses *crossover*. Proses seleksi kromosom akan dijelaskan pada Subbab 2.3.3 berikut.

2.3.3. Seleksi

Seperti layaknya proses seleksi alam, tahap seleksi pada AG memilih kromosom terbaik untuk menjadi kromosom orang tua. Kromosom orang tua yang dipilih adalah sebanyak ukuran populasi, kromosom-kromosom orang tua inilah yang akan diproses pada tahap selanjutnya untuk menghasilkan anak yang diharapkan membawa sifat-sifat baik kedua orang tuanya. Beberapa metode yang dapat digunakan untuk memilih kromosom orang tua adalah *roulette wheel selection*, *tournament selection*, *ranking-based selection*.

Pada tugas akhir ini, metode seleksi yang digunakan adalah *ranking-based selection*. Metode ini dipilih karena kedua metode lainnya memilih orang tua proporsional terhadap nilai *fitness* kromosom, artinya kromosom dengan nilai *fitness* terbaik memiliki kemungkinan terpilih lebih besar dibandingkan kromosom lainnya, hal ini dapat menyebabkan keragaman populasi berkurang sehingga solusi mungkin menuju lokal optimal.

Pemilihan orang tua pada *Ranking Based Selection* dilakukan dengan menggunakan formula sebagai berikut (Ghoseiri, 2009):

$$\text{Pilih}(K) = \left\{ K_j \in K | j = P - \left\lfloor \frac{-1 + \sqrt{1 + 4 \times rand \times (P^2 + P)}}{2} \right\rfloor \right\} \qquad (2.11)$$

K adalah himpunan dari P kromosom yang diurutkan secara naik berdasarkan nilai *fitness*, $rand$ adalah sebuah bilangan acak yang nilainya berada di interval (0,1) dan berdistribusi *uniform*, simbol $\lfloor b \rfloor$ menyatakan bilangan bulat terbesar yang lebih kecil atau sama dengan b. Formula (2.11) menghasilkan kromosom dari himpunan K yang akan dipilih lebih dahulu untuk selanjutnya diproses pada tahap rekombinasi (*crossover* dan mutasi). Pemilihan orang tua dengan formula (2.11) dilakukan berulang-ulang sebanyak ukuran populasi untuk mendapatkan urutan orang tua yang akan memasuki proses *crossover* dan mutasi.

Sebagai contoh, misalkan terdapat 50 kromosom pada populasi awal, dengan himpunan $K = \{K_1, K_2, \ldots, K_{49}, K_{50}\}$. Misalkan akan dipilih sepasang orang tua, misalkan bilangan acak pertama yang dibangkitkan adalah 0.8147, maka dengan menggunakan formula (2.11) orang tua pertama yang terpilih adalah orang tua j dengan

$$j = 50 - \left\lfloor \frac{-1 + \sqrt{1 + 4 \times (0.8147) \times (50^2 + 50)}}{2} \right\rfloor = 5$$

Misalkan pula bilangan acak kedua yang terpilih adalah 0.6324, maka dengan menggunakan formula (2.11) orang tua kedua yang terpilih adalah orang tua j dengan

$$j = 50 - \left\lfloor \frac{-1 + \sqrt{1 + 4 \times (0.6324) \times (50^2 + 50)}}{2} \right\rfloor = 11$$

Jadi, dua kromosom orang tua yang terpilih untuk tahap selanjutnya adalah kromosom K_5 dan kromosom K_{11}.

Kromosom-kromosom orang tua yang terpilih pada proses seleksi selanjutnya memasuki proses *crossover* untuk menghasilkan kromosom keturunan yang akan dijelaskan pada Subbab 2.3.4 berikut.

2.3.4. *Crossover*

Crossover adalah operator pada AG yang mengkombinasikan dua kromosom orang tua untuk menghasilkan kromosom keturunan. Proses kombinasi ini disebut juga proses persilangan. Keturunan yang diharapkan adalah keturunan yang lebih baik dari kedua orang tuanya. *Crossover* memilih gen-gen pada kromosom orang tua dan membuat keturunan dari gen-gen tersebut. Namun, tidak semua kromosom orang tua yang terpilih mengalami proses *crossover*. Peluang terjadinya proses *crossover* pada kromosom orang tua dalam proses *crossover* berdasarkan suatu probabilitas yang disebut probabilitas *crossover* (p_c). Untuk setiap pasang kromosom orang tua akan dibangun bilangan acak yang nilainya terletak di interval (0,1). Jika nilai bilangan acak tersebut lebih kecil atau sama dengan probabilitas *crossover*, maka satu pasang kromosom orang tua ini akan memasuki proses *crossover*. Jika tidak, maka gen-gen pada kromosom orang tua akan secara langsung disalin ke kromosom keturunan. Beberapa metode *crossover* yang dapat digunakan antara lain adalah 1-*point crossover*, *n-point crossover*, *heuristic crossover*, dan *merge crossover*.

2.3.4.1 1-*point crossover*

Operator 1-*point crossover* menghasilkan 2 kromosom keturunan, dengan cara memilih secara acak satu titik *crossover* pada kedua kromosom orang tua. Keturunan pertama dibentuk dengan menyalin semua gen sebelum titik *crossover* pada orang tua 1 dan menyalin semua gen setelah titik *crossover* pada orang tua 2. Sedangkan keturunan kedua dibentuk dengan menyalin semua gen sebelum titik *crossover* pada orang tua 2 dan menyalin semua gen setelah titik *crossover* pada orang tua 1.

Berikut akan diberikan contoh 1-*point crossover*. Misalkan dari tahap seleksi terpilih dua kromosom orang tua, yaitu $4-3-5-2-6-1$ dan $4-5-3-1-2-6$. Kemudian ditentukan secara acak titik *crossover* pada kedua orang tua. Misalkan terpilih titik *crossover* pada posisi sebagai berikut :

Orang tua 1 : $\mathbf{4-3-5} \mid 2-6-1$

Orang tua 2 : $4-5-3 \mid \mathbf{1-2-6}$

Maka dua kromosom keturunan yang terbentuk adalah sebagai berikut :

Keturunan 1 : $\mathbf{4-3-5-1-2-6}$

Keturunan 2 : $4-5-3-2-6-1$

2.3.4.2 *n-point crossover*

Operator *n-point crossover* memilih secara acak *n* titik *crossover* pada kedua kromosom orang tua untuk menghasilkan 2 kromosom keturunan. Keturunan pertama dibentuk dengan menyalin semua gen pada orang tua 1 sebelum titik *crossover* 1, kemudian menyalin semua gen orang tua 2 setelah titik *crossover* 1 dan sebelum titik *crossover* 2, dan menyalin semua gen pada orang tua 1 setelah titik *crossover* 2, begitu seterusnya hingga titik *crossover n*. Sedangkan keturunan kedua dibentuk dengan menyalin semua gen pada orang tua 2 sebelum titik *crossover* 1, kemudian menyalin semua gen pada orang tua 1 setelah titik *crossover* 1 dan sebelum titik *crossover* 2, dan menyalin semua gen pada orang tua 2 setelah titik *crossover* 2, begitu seterusnya hingga titik *crossover n*.

Berikut akan diberikan contoh 2-*point crossover*. Misalkan dari tahap seleksi terpilih dua kromosom orang tua, yaitu

$4-3-5-2-6-1-7-8-9$ dan $4-5-3-1-2-6-8-7-9$.

Kemudian ditentukan secara acak 2 titik *crossover* pada kedua orang tua, misalkan titik *crossover* yang terpilih sebagai berikut :

Orang tua 1 : $\mathbf{4-3}\,|\,5-2-6-1\,|\,\mathbf{7-8-9}$

Orang tua 2 : $4-5\,|\,\mathbf{3-1-2-6}\,|\,8-7-9$

Maka dua kromosom keturunan yang terbentuk sebagai berikut :

Keturunan 1 : $\mathbf{4-3-3-1-2-6-7-8-9}$

Keturunan 2 : $4-5-5-2-6-1-8-7-9$

Pada contoh di atas terlihat bahwa gen 3 muncul 2 kali pada Keturunan 1 dan gen 5 muncul 2 kali pada Keturunan 2. Jadi, penggunaan dua tipe *crossover* di atas tidak cocok untuk masalah yang memperhatikan urutan seperti TSP atau VRP, karena memungkinkan terjadi duplikasi gen dalam kromosom atau dengan kata lain satu gen dapat muncul dua kali dalam kromosom keturunan.Oleh karena itu beberapa peneliti menggunakan metode lain yang dapat diterapkan pada masalah yang memperhatikan urutan, yaitu *heuristic crossover* atau *merge crossover* yang akan dijelaskan di Subbab 2.3.4.3 dan 2.3.4.4 berikut.

2.3.4.3 *Heuristic crossover*

Heuristic crossover memilih gen untuk keturunan dengan mempertimbangkan jarak antar gen pada kromosom orang tua. Langkah yang dilakukan pertama kali adalah memilih satu titik *crossover* secara acak dari kedua orang tua. Kemudian gen pertama pada keturunan dipilih secara acak dari gen pertama setelah titik *crossover* pada kedua orang tua. Setelah gen pertama pada kromosom keturunan diperoleh, akan dipilih gen selanjutnya yang memiliki jarak terdekat dengan gen pertama pada kromosom keturunan tersebut. Proses pemilihan ini akan terus berlanjut hingga semua gen pada keturunan telah didapatkan. Metode ini hanya menghasilkan satu keturunan.

Pada setiap proses pemilihan gen, setelahnya dilakukan penukaran atau penghapusan gen yang masuk ke kromosom keturunan pada orang tua yang tidak

terpilih. Penukaran ataupun penghapusan gen setelah pemilihan gen keturunan dilakukan untuk menghindari duplikasi pada iterasi selanjutnya. Jika digunakan penukaran gen maka *heuristic crossover* disebut *heuristic crossover* 1 (HX1) dan jika digunakan penghapusan gen maka *heuristic crossover* disebut *heuristic crossover* 2 (HX2). Pada tugas akhir ini yang akan digunakan adalah operator yang melakukan penukaran gen, yaitu operator HX1.

Berikut akan diberikan contoh *heuristic crossover.* Misalkan dari tahap seleksi terpilih dua kromosom orang tua, $8-11-3-5-6-4-2-12-1-9-7-10$ dan $1-2-3-4-5-6-7-8-9-10-11-12$. Kemudian dipilih secara acak titik *crossover* pada kedua kromosom orang tua, misalkan titik *crossover* yang terpilih adalah sebagai berikut

Orang tua 1 : $\mathbf{8-11-3-5-6-4} \mid 2-12-1-9-7-10$

Orang tua 2 : $1-2-3-4-5-6 \mid \mathbf{7-8-9-10-11-12}$

Misalkan terpilih gen 2 sebagai gen pertama kromosom keturunan, kemudian gen 2 dan gen 7 pada orang tua 2 ditukar sehingga urutan pada orang tua 2 menjadi :

Orang tua 2 : $1-\mathbf{7}-3-4-5-6 \mid \mathbf{2}-8-9-10-11-12$

Setelah penukaran terjadi, gen selanjutnya pada kromosom keturunan dipilih dengan memperhatikan jarak $d(2,12)$ dan $d(2,8)$. Jika $d(2,8) < d(2,12)$ maka akan dipilih gen 8 untuk kromosom keturunan. Selanjutnya gen 8 dan gen 12 pada orang tua 1 ditukar. Kemudian proses ini dilakukan terus menerus hingga semua gen pada keturunan didapatkan.

Untuk contoh di atas, misalkan didapat kromosom keturunan sebagai berikut :

Keturunan : $2-8-1-9-7-10-12-11-3-4-5-6$

2.3.4.4 *Merge crossover*

Merge crossover menghasilkan satu keturunan dimana gen pertama pada kromosom keturunan dipilih dengan mempertimbangkan urutan masing-masing gen. Misalkan terdapat dua gen i dan j dimana gen i harus berada sebelum gen j, maka penempatan kedua gen tersebut dalam satu kromosom harus mengikuti keterurutan gen tersebut. Seperti pada *heuristic crossover*, titik *crossover* dan gen

pertama dipilih secara acak. Gen selanjutnya akan dipilih berdasarkan urutan yaitu gen dengan urutan lebih awal akan terpilih. Dalam metode ini juga dilakukan penukaran atau penghapusan gen pada setiap proses pemilihan gen kromosom keturunan, Jika digunakan penukaran gen maka *merge crossover* disebut *merge crossover* 1 (MX1) dan jika digunakan penghapusan gen maka *merge crossover* disebut *merge crossover* 2 (MX2). Pada tugas akhir ini, yang akan digunakan adalah MX1.

Berikut akan diberikan contoh penggunaan *merge crossover* pada VRPTW. Misalkan dari tahap seleksi terpilih dua kromosom orang tua $8-11-3-5-6-4-2-12-1-9-7-10$ dan $1-2-3-4-5-6-7-8-9-10-11-12$ dan urutan gen-gen tersebut dilihat dari waktu awal pelanggan yang kecil hingga yang besar adalah sebagai berikut :

Urutan gen : 1 2 3 4 5 6 7 8 9 10 11 12

Kemudian dipilih secara acak titik *crossover* pada kedua kromosom orang tua, misalkan titik *crossover* yang terpilih adalah sebagai berikut :

Orang tua 1 : $\mathbf{8-11-3}\,|\,5-6-4-2-12-1-9-7-10$

Orang tua 2 : $1-2-3\,|\,\mathbf{4-5-6-7-8-9-10-11-12}$

Misalkan terpilih gen 4 pada kromosom keturunan, kemudian gen 4 dan gen 5 pada orang tua 1 ditukar sehingga urutan pada orang tua 1 menjadi :

Orang tua 1 : $8-11-3\,|\,\mathbf{4}-6-\mathbf{5}-2-12-1-9-7-10$

Selanjutnya dilihat gen-gen setelah gen 4 pada masing-masing orang tua. Jika *time window* gen 5 lebih awal dari *time window* gen 6, maka gen 5 yang terpilih untuk dimasukkan ke kromosom keturunan. Selanjutnya gen 5 dan gen 6 pada orang tua 1 ditukar. Kemudian proses ini dilakukan terus menerus hingga semua gen pada keturunan didapatkan.

Untuk contoh di atas, misalkan didapat kromosom keturunan sebagai berikut :

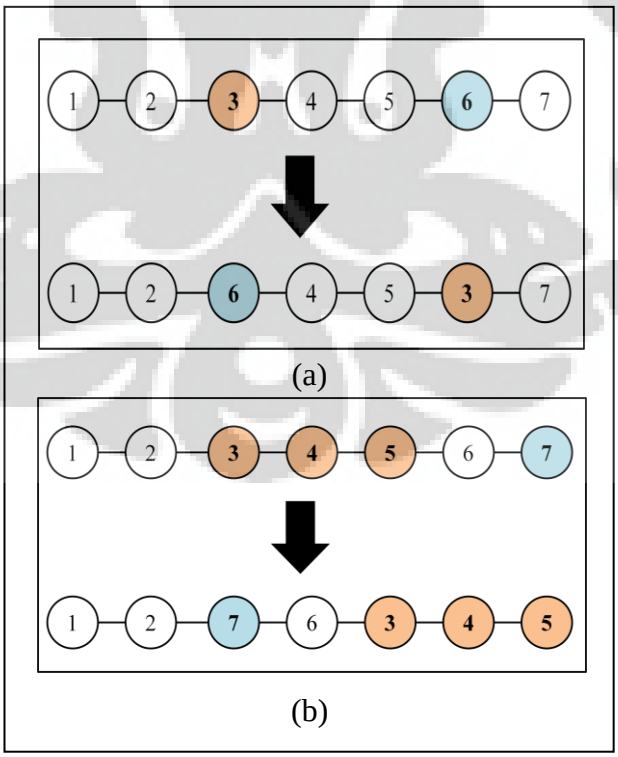
Keturunan : $4-5-6-2-8-1-9-7-10-12-11-3$

Penggunaan kedua operator ini pada masalah VRPTW memungkinkan untuk menghasilkan kromosom keturunan yang lebih baik dalam memenuhi kendala jarak dan *time windows*.

2.3.5. Mutasi

Setelah operator *crossover* diterapkan, operator selanjutnya adalah mutasi. Mutasi dilakukan untuk menjaga keragaman kromosom pada satu populasi. Tidak semua keturunan dari *crossover* mengalami mutasi. Banyaknya orang tua yang mengalami mutasi akan bergantung pada probabilitas mutasi (p_m) yang ditentukan. Orang tua yang tidak mengalami mutasi akan secara langsung disalin ke kromosom keturunan.

Beberapa operator mutasi yang dapat digunakan adalah penukaran gen pada kromosom, penukaran barisan gen pada kromosom, penggantian nilai gen dengan inversnya. Pemilihan operator yang digunakan bergantung pada cara pengkodean masalah. Operator mutasi yang dapat digunakan untuk masalah yang menggunakan pengkodean dengan bilangan biner adalah metode mengganti nilai gen dengan inversnya. Sedangkan untuk masalah yang menggunakan pengkodean permutasi, operator mutasi yang dapat digunakan adalah menukar gen dan menukar barisan gen pada kromosom. Penggambaran kedua metode tersebut dapat dilihat pada Gambar 2.6.



[Thangiah,1996]

Gambar 2.6 Operator mutasi, (a) penukaran gen, (b) penukaran barisan gen

Metode penukaran gen melakukan penukaran posisi gen di dalam satu kromosom. Pada Gambar 2.6(a), gen 3 dan gen 6 bertukar posisi. Sedangkan metode penukaran barisan gen memiliki prinsip yang sama dengan menukar gen, hanya saja gen yang ditukar dapat lebih dari satu dan berurutan. Seperti contoh yang diberikan pada Gambar 2.6(b), gen 3, 4, dan 5 ditukar dengan gen 7. Salah satu operator mutasi lain yang dapat digunakan adalah *sequence based mutation* yang berprinsip seperti menukar barisan gen. Pada tugas akhir ini akan digunakan *sequence based mutation*, pemilihan operator *sequence based mutation* didasarkan pada penyesuaian terhadap masalah. Masalah yang akan diselesaikan pada tugas akhir ini adalah VRPTW dimana solusi merupakan himpunan rute. Oleh karena itu operator ini dipilih, karena operator ini menerapkan mutasi terhadap kromosom yang terlebih dahulu dikembalikan ke solusi berupa kumpulan rute. *Sequence based mutation* akan dijelaskan pada Bab 3.

BAB 3
APLIKASI ALGORITMA GENETIKA HIBRIDA PADA *VEHICLE ROUTING PROBLEM WITH TIME WINDOWS*

Pada bab ini akan dijelaskan mengenai penggunaan algoritma genetika hibrida (AGH) dalam menyelesaikan VRPTW. Pada Subbab 3.1 akan dijelaskan mengenai pengkodean yang digunakan untuk VRPTW. Penyelesaian masalah VRPTW dimulai dengan membentuk solusi awal berupa populasi yang terdiri dari sekumpulan kromosom yang akan dijelaskan pada Subbab 3.2. Setelah mendapatkan populasi awal, proses seleksi dilakukan berdasarkan *ranking-based selection* untuk memilih kromosom orang tua dari populasi awal yang terbentuk yang akan dijelaskan pada Subbab 3.3. Selanjutnya pada Subbab 3.4 akan dijelaskan mengenai proses *crossover* untuk orang tua yang terpilih dengan teknik *heuristic crossover* dan *merge crossover*. Langkah selanjutnya adalah melakukan mutasi pada beberapa individu yang dihasilkan yang akan dijelaskan pada Subbab 3.5. Proses ini akan terus diulang hingga mencapai solusi optimal atau generasi ke-*n* telah dicapai. Pada VRPTW solusi yang diharapkan adalah sekumpulan rute yang melayani seluruh pelanggan dengan biaya minimum. Dalam hal ini biaya direpresentasikan oleh total jarak tempuh kendaraan-kendaraan untuk melayani seluruh pelanggan.

3.1 Pengkodean Solusi pada Kromosom

Dalam tugas akhir ini akan digunakan pengkodean permutasi yang telah dijelaskan di Subbab 2.3.1.4 dimana kromosom berisi barisan bilangan bulat positif. Solusi dari VRPTW merupakan kumpulan rute yang melayani seluruh pelanggan dan memenuhi kendala kapasitas serta *time windows*. Pada tugas akhir ini, semua rute pada satu solusi dikodekan dalam satu kromosom tanpa ada tanda pemisah antar rute dalam solusi tersebut. Pengembalian kromosom ke dalam bentuk rute harus memenuhi kendala kapasitas dan *time windows* untuk memisahkan rute satu dan rute lainnya. Sebagai contoh, solusi $S = \{R_1, R_2, R_3\}$

dimana $R_1 : 0-1-2-4$, $R_2 : 0-7-0$, dan $R_3 : 0-3-5-6-0$. Maka pengkodean kromosom untuk solusi tersebut adalah $1-2-4-7-3-5-6$.

3.2 Pembentukan Populasi Awal

Seperti yang telah dijelaskan pada bab sebelumnya, representasi solusi pada AG merupakan kromosom. Sekumpulan kromosom akan membentuk populasi, dimana populasi pertama yang dibentuk disebut populasi awal. Pada tugas akhir ini, 50% populasi awal akan dibentuk dengan menggunakan metode *Push Forward Insertion Heuristic* (PFIH) dilanjutkan *λ-Interchange* untuk membentuk solusi lingkungan dan 50% lainnya dibentuk dengan acak.

Tahap pertama yang dilakukan pada PFIH adalah membuat satu rute, yaitu dengan cara memilih satu pelanggan sebagai pelanggan pertama, kemudian menyisipkan pelanggan yang belum dilayani ke rute yang terbentuk hingga salah satu kendala tidak terpenuhi atau tidak ada lagi pelanggan yang dapat disisipkan. Pelanggan pertama pada setiap rute dipilih berdasarkan fungsi biaya c_i berikut (Thangiah, 1999) :

$$c_i = -\alpha d_{0i} + \beta b_i + \gamma \left(\frac{P_i}{360}\right) d_{0i} \tag{3.1}$$

dengan c_i adalah total biaya untuk pelanggan i yang merupakan penjumlahan dari jarak depot ke pelanggan i (d_{0i}), waktu akhir pelanggan i (b_i), dan sudut koordinat polar pelanggan i terhadap depot (P_i) dengan α, β, dan γ adalah konstanta. Pada artikel yang dibuat oleh Thangiah (1999), nilai masing-masing parameter yang digunakan adalah $\alpha = 0.7$, $\beta = 0.1$, dan $\gamma = 0.2$. Hal ini berarti, prioritas pemilihan pelanggan dilihat dari total jarak, sudut polar koordinat, dan waktu akhir pelanggan. Pelanggan dengan biaya terkecil akan dipilih menjadi pelanggan pertama yang harus dikunjungi.

Setelah pelanggan pertama terpilih, akan dipilih pelanggan lain dari pelanggan-pelanggan yang belum dilayani yang meminimumkan biaya penyisipan antara setiap busur pada rute dan tidak melanggar kendala *time windows* serta kapasitas kendaraan pada rute tersebut.

Misalkan $R_p : 0 - p_1 - p_2 - \cdots - p_n - 0$ adalah rute ke-p dimana 0 merupakan depot, p_1 adalah pelanggan pertama, p_2 adalah pelanggan kedua, dan

p_n adalah pelanggan ke-n yang dilayani pada rute tersebut. Misalkan pelanggan j dilayani tepat setelah pelanggan ke-i dilayani pada rute tersebut, dan misalkan akan disisipkan pelanggan k di antara pelanggan i dan j. Kendala *time windows* akan terpenuhi jika setelah disisipkan pelanggan k, waktu mulai pelayanan pelanggan k lebih kecil atau sama dengan waktu akhir pelanggan k dan perubahan waktu mulai pelayanan pelanggan j tidak membuat kendaraan sampai di pelanggan tersebut melebihi waktu akhir pelanggan j. Waktu mulai pelayanan kendaraan di pelanggan j sebelum disisipkan pelanggan k adalah y_{jp}, dan waktu mulai pelayanan kendaraan di pelanggan j setelah disisipkan pelanggan k dinotasikan dengan y'_{jp}.

Perubahan waktu mulai pelayanan di pelanggan j dapat mempengaruhi waktu mulai pelayanan di pelanggan-pelanggan setelah pelanggan j, sehingga diperlukan pengecekan kelayakan untuk semua pelanggan setelah pelanggan j. Oleh karena itu didefinisikan nilai *Push Forward* untuk semua pelanggan setelah pelanggan k. Nilai *Push Forward* akan bernilai 0 jika perubahan waktu yang diakibatkan oleh penyisipan pelanggan k tidak mempengaruhi waktu mulai pelayanan di pelanggan j. Namun, jika waktu mulai pelayanan di pelanggan j berubah, pengecekan waktu mulai pelayanan ini akan terus berlanjut untuk pelanggan-pelanggan setelah pelanggan j hingga nilai *Push Forward* salah satu pelanggan 0 atau kendaraan terlambat (kendaraan memulai pelayanan melebihi waktu akhir pelanggan) di salah satu pelanggan. Sedangkan untuk depot akan dihitung waktu kendaraan sampai di depot setelah melayani pelanggan-pelanggan pada rute tersebut.

Selanjutnya untuk setiap rute dimana terdapat penyisipan pelanggan yang layak dihitung biaya penyisipan, sebagai berikut:

$$C = D_{i,k,j,p} + \phi W_{i,k,j,p} \tag{3.2}$$

dengan $D_{i,k,j,p}$ merupakan total jarak yang ditempuh kendaraan rute p setelah disisipkan pelanggan k di antara i dan j, dan $W_{i,k,j,p}$ merupakan total waktu tempuh kendaraan rute p yang merupakan penjumlahan total jarak dan total waktu pelayanan setiap pelanggan pada rute tersebut. Pada artikel Thangiah (1996),

konstanta ϕ ditetapkan sebesar 1% dari $D_{i,k,j,p}$. Rute dengan biaya penyisipan minimum akan dipilih dan dimasukkan ke solusi sekarang.

Proses penyisipan ini berlangsung hingga tidak ada lagi pelanggan yang dapat disisipkan sehingga dibutuhkan rute baru untuk menampung pelanggan yang belum dilayani dengan asumsi banyaknya kendaraan yang digunakan selalu tersedia. Keseluruhan langkah dalam PFIH dapat dilihat pada Gambar 3.1.

Setelah diperoleh solusi dari PFIH, tahap selanjutnya adalah pembentukan solusi lingkungan dengan menggunakan metode *λ-Interchange*. Pada metode ini dilakukan perpindahan pelanggan antar rute.

Misalkan $S = \{R_1, R_2 \dots, R_p, R_q, \dots, R_k\}$ adalah solusi awal yang terbentuk dari metode PFIH dimana R_1 adalah himpunan pelanggan yang dilayani pada rute 1, R_2 adalah himpunan pelanggan yang dilayani pada rute 2, dan seterusnya. *λ-Interchange* antar rute R_p dan R_q adalah perpindahan subset $S_1 \subseteq R_p$ dengan $|S_1| \leq \lambda$ dan subset $S_2 \subseteq R_q$ dengan $|S_2| \leq \lambda$, sehingga menghasilkan dua rute baru $R'_p = (R_p - S_1) \cup S_2$ dan $R'_q = (R_q - S_2) \cup S_1$ dan satu solusi lingkungan $S' = \left(S - \{R_p, R_q\}\right) \cup \{R'_p, R'_q\}$. Solusi lingkungan $N_\lambda(S)$ dari solusi S adalah himpunan dari semua solusi lingkungan S' yang didapat dari metode *λ-Interchange* untuk suatu bilangan bulat λ. Pada tugas akhir ini akan dilakukan pertukaran untuk λ = 1 dan λ = 2. Pertukaran rute pada himpunan S mengikuti urutan berikut :

$$(R_1, R_2), (R_1, R_3) \dots (R_1, R_k), (R_2, R_3), (R_2, R_4) \dots (R_2, R_k), \dots, (R_{k-1}, R_k) \quad (3.2)$$

Tahap 1. Buat rute pertama yang berangkat dari depot. $r = 1$

Tahap 2. Jika semua pelanggan telah dilayani

lanjutkan ke tahap 7

Jika tidak

untuk setiap pelanggan *j*

Hitung biaya berdasarkan (3.1)

Tahap 3. Pilih pelanggan pertama, j^* dari pelanggan - pelanggan yang belum dilayani yang memenuhi kendala *time windows* dan kapasitas dengan biaya minimum.

Tahap 4. Masukkan j^* ke rute r dan perbaharui kapasitas rute

Tahap 5. Untuk setiap pelanggan j yang belum masuk rute :

Tentukan tempat penyisipan pelanggan yang layak dan hitung biaya penyisipan berdasarkan (3.2).

Jika tidak ditemukan tempat layak untuk penyisipan pelanggan pada rute r

Buat rute baru $r = r + 1$, kembali ke tahap 2.

Tahap 6. Pilih pelanggan j^* dengan biaya penyisipan paling minimum,

dan sisipkan pelanggan pada tempat penyisipan yang layak di rute r.

Perbaharui kapasitas rute r. Kembali ke tahap 5.

Tahap 7. Solusi awal telah terbentuk. STOP

[Thangiah, 1996]

Gambar 3.1 Algoritma PFIH

Banyaknya pertukaran rute yang mungkin adalah sebanyak kombinasi 2 dari k, yaitu $C_2^k = \frac{k(k-1)}{2}$. Pertukaran pelanggan pada setiap rute dilakukan dengan cara memindahkan pelanggan dari satu rute ke rute yang lain atau menukar pelanggan dari satu rute dengan pelanggan dari rute yang lain. Aturan penukaran pelanggan mengikuti operator (i, j), dimana besarnya i, j lebih kecil atau sama dengan besarnya λ. Karena pada tugas akhir ini digunakan $\lambda = 1$ dan $\lambda = 2$ sehingga operator yang digunakan adalah (0,1), (1,0), (1,1), (0, 2), (2,0), (2, 1), (1, 2), dan (2,2).

Operator (i, j) pada rute (R_p, R_q) berarti memindahkan sebanyak i pelanggan dari R_p ke R_q, dan memindahkan sebanyak j pelanggan dari R_q ke R_p. Sebagai contoh untuk operator (1,2) pada rute (R_p, R_q) berarti memindahkan satu pelanggan dari R_p ke R_q dan 2 pelanggan dari R_q ke R_p. Penerimaan solusi-solusi

lingkungan S' tersebut didasarkan pada fungsi biaya $C(S)$ berikut (Thangiah, 1999):

$$C(S) = \sum_{k=1}^{K} D_k + \phi W_k + \eta O_k + \kappa T_k \qquad (3.3)$$

Perhitungan biaya solusi lingkungan mempertimbangkan total jarak tempuh kendaraan rute k (D_k), total waktu tempuh kendaraan rute k (W_k), total kelebihan muatan untuk kendaraan rute k (O_k) , dan total keterlambatan kendaraan rute k (T_k) dengan masing-masing konstanta bernilai $\phi = 0.01D_k$, $\eta = 0.1D_k$, dan $\kappa = 0.01D_k$. Berikut adalah algoritma λ-*Interchange* untuk VRPTW (Thangiah, 1999):

Tahap 1. Cari solusi awal S untuk VRPTW dengan metode PFIH

Tahap 2. Pilih solusi lingkungan $S' \in N_\lambda(S)$ sesuai urutan pada (3.2)

Tahap 3. Jika $C(S') < C(S)$

terima S', kembali ke tahap 2

Jika tidak

lanjutkan ke tahap 4

Tahap 4. Jika solusi lingkungan $S' \in N_\lambda(S)$ telah selesai diperiksa

lanjutkan ke tahap 5

Jika tidak

kembali ke tahap 2

Tahap 5. STOP

Gambar 3.2 Algoritma λ-*Interchange*

Kemudian dilakukan pemeriksaan kendala untuk solusi-solusi S' yang diterima. Solusi - solusi yang memenuhi kendala kemudian diurutkan berdasarkan total jarak, dimulai dari solusi yang memiliki total jarak terkecil hingga solusi yang memiliki total jarak terbesar. Banyaknya solusi lingkungan layak yang terpilih untuk dimasukkan ke populasi awal adalah sebanyak setengah dari *popsize* dikurang 1, karena 1 solusi telah didapatkan dari PFIH.

Selanjutnya akan diberikan contoh masalah 3.1 yang merupakan masalah pengiriman barang dengan 6 pelanggan dan 1 depot. Akan dicari rute pelayanan pelanggan dengan biaya minimum dimana biaya ditentukan sebagai total jarak

yang dilalui setiap kendaraan untuk melayani semua pelanggan. Kapasitas masing-masing kendaraan (Q) adalah 50. Data pelanggan disajikan dalam Tabel 3.1 dan data jarak antar pelanggan dan jarak masing-masing pelanggan dengan depot disajikan dalam Tabel 3.2, diasumsikan jarak antar pelanggan simetris artinya $d_{ij} = d_{ji}$. Kolom pertama pada Tabel 3.1 merupakan pelanggan i, dengan $i = \{0, 1, 2, 3, 4, 5, 6\}$, 0 merupakan depot, kolom kedua merupakan permintaan pelanggan i, dan kolom-kolom selanjutnya berturut-turut adalah waktu awal, waktu akhir, waktu pelayanan, sudut koordinat polar, dan biaya pelanggan i berdasarkan formula (3.1). Pada contoh ini ukuran populasi ditentukan sebanyak ukuran masalah yaitu 6 kromosom dalam 1 populasi, dimana 3 solusi akan dibentuk dari metode PFIH dan λ-*Interchange*, 3 solusi lainnya akan dibentuk secara acak.

Tabel 3.1 Data pelanggan

i	q_i	a_i	b_i	s_i	P_i	c_i
0	0	0	800	0	-	-
1	15	534	605	90	-84.2894	45.4890
2	20	262	317	90	83.6598	2.1105
3	10	76	129	90	-51.3402	3.5704
4	20	47	124	90	-56.3099	-0.7834
5	15	203	260	90	-16.6992	-11.0254
6	9	95	105	90	13.1340	-10.8393

Sesuai dengan algoritma PFIH pada Gambar 3.1, tahap pertama adalah membuat rute pertama R_1. Untuk pelanggan pertama pada R_1 dipilih pelanggan dengan c_i yang minimum. Dari Tabel 3.1 didapatkan pelanggan 5 yang memiliki nilai c_i minimum.

Tabel 3.2 Data jarak antar pelanggan dan pelanggan dengan depot

d_{ij}	0	1	2	3	4	5	6
0	0	20.0998	45.2769	12.8062	18.0278	52.2015	30.8058
1	20.0998	0	65.0692	31.6228	37.3363	62.6817	38.8973
2	45.2769	65.0692	0	37.3363	33.5410	62.6498	45.4863
3	12.8062	31.6228	37.3363	0	5.3852	42.2966	38.1182
4	18.0278	37	33.5410	5.3852	0	40	40.7922
5	52.2015	62.6817	62.6498	42.2966	40	0	80.3990
6	30.8058	38.8973	45.4863	38.1182	40.7922	80.3990	0

Selanjutnya dilakukan pengecekan kapasitas kendaraan dan *time windows*. R_1 memenuhi kendala kapasitas karena permintaan pelanggan 5 yaitu 15 yang lebih kecil dari kapasitas kendaraan dan memenuhi *time window* pelanggan 5 karena kendaraan memulai pelayanan di pelanggan 5 sebelum waktu akhir pelanggan 5, yaitu di waktu $y_{51} = 203$. Waktu mulai pelayanan kendaraan di pelanggan 5 dapat diperoleh dengan perhitungan berikut :

$$y_{51} = \max\{a_5, y_{01} + s_0 + d_{01}\}$$
$$= \max\{203{,}0 + 0 + 52.2015\}$$
$$= 203$$

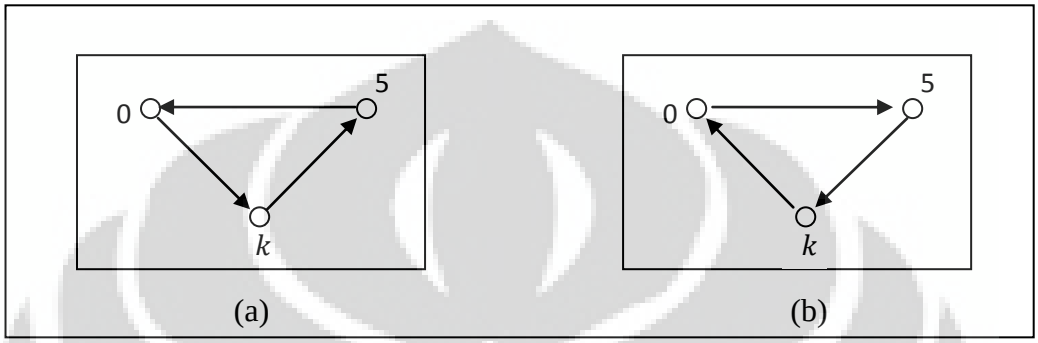
dan kendaraan sampai di depot sebelum *time window* depot berakhir yaitu $y_{01} = 345.2015$. Waktu kendaraan sampai di depot diperoleh dengan perhitungan sebagai berikut :

$$y_{01} = \max\{a_0, y_{51} + s_5 + d_{15}\}$$
$$= \max\{0{,}203 + 90 + 52.2015\}$$
$$= \max\{0{,}345.2015\}$$
$$= 345.2015$$

Selanjutnya rute dan total permintaan kendaraan diperbaharui menjadi $R_1 : 0 - 5 - 0$ dan total permintaan 15.

Tahap selanjutnya adalah pemilihan pelanggan untuk disisipkan ke dalam R_1. Untuk setiap pelanggan yang belum dilayani akan dilakukan pencarian tempat penyisipan yang layak, kemudian dilakukan perhitungan biaya penyisipan tersebut untuk setiap busur (i, j) pada R_1 sesuai dengan formula (3.2).

Misalkan akan dilakukan penyisipan pelanggan k pada R_1. Terdapat dua kemungkinan penyisipan pelanggan k pada rute R_1, kemungkinan pertama yaitu pelanggan k disisipkan di posisi setelah depot sebelum pelanggan 5 yang digambarkan pada Gambar 3.3(a), kemungkinan kedua yaitu pelanggan k disisipkan di posisi setelah pelanggan 5 sebelum depot yang digambarkan pada Gambar 3.3(b).



Gambar 3.3 Kemungkinan penyisipan pelanggan k pada R_1

Akan disisipkan pelanggan-pelanggan yang belum masuk rute, yaitu pelanggan 1, 2, 3, dan 4. Misalkan akan disisipkan pelanggan 1 seperti Gambar 3.3(a) dan Gambar 3.3(b). Untuk Gambar 3.3(a) terlebih dahulu akan dilihat kelayakan penyisipan untuk kendala *time windows*, dimulai dari pelanggan yang disisipkan, yaitu pelanggan 1. Waktu mulai pelayanan pelanggan 1 diperoleh sebagai
berikut :

$$y'_{11} = \max\{a_1, y_{01} + s_0 + t_{01}\}$$
$$= \max\{534, 0 + 0 + 20.0998\}$$
$$= 534$$

Kendaraan memulai pelayanan di pelanggan 1 tepat pada waktu awal pelanggan 1, sehingga memenuhi *time window* pelanggan 1.

Selanjutnya akan diperiksa waktu mulai pelayanan kendaraan untuk pelanggan setelah pelanggan 1, yaitu pelanggan 5, sebagai berikut :

$$y'_{51} = \max\{a_5, y_{11} + s_1 + t_{15}\}$$
$$= \max\{203, 534 + 90 + 62.6817\}$$
$$= \max\{203, 686.6817\}$$
$$= 686.6817$$

Karena y'_{51} melebihi waktu akhir pelanggan 5, yaitu $b_5 = 260$, maka penyisipan pelanggan 1 sesuai dengan Gambar 3.3(a) pada rute R_1 tidak layak, perhitungan tidak dilanjutkan untuk pelanggan selanjutnya (depot).

Selanjutnya dilakukan perhitungan waktu mulai pelayanan untuk penyisipan pelanggan 1 pada posisi sesuai dengan Gambar 3.3(b), sebagai berikut :

$$y'_{11} = \max\{a_1, y_{51} + s_5 + t_{51}\}$$
$$= \max\{534, 203 + 90 + 62.6817\}$$
$$= \max\{534, 355.6817\}$$
$$= 534$$

Kendaraan memulai pelayanan di pelanggan 1 pada waktu $y'_{11} = 534$, sehingga memenuhi *time window* pelanggan 1. Pemeriksaan berlanjut untuk pelanggan selanjutnya, yaitu depot. Waktu kedatangan pelanggan di depot diperoleh sebagai berikut :

$$y'_{01} = \max\{a_0, y'_{11} + s_1 + t_{10}\}$$
$$= \max\{0, 534 + 90 + 20.0998\}$$
$$= \max\{0, 644.0998\}$$

Kendaraan tiba di depot sebelum *time window* depot berakhir, sehingga penyisipan pelanggan 1 pada R_1 sesuai dengan Gambar 3.3(b) memenuhi kendala *time windows*. Rute ini juga memenuhi kendala kapasitas, karena total permintaan pelanggan pada rute R_1 setelah disisipkan pelanggan 1 lebih kecil dari kapasitas kendaraan, yaitu $q_5 + q_1 = 15 + 15 = 30$ lebih kecil dari $Q = 50$.

Setelah diperoleh tempat penyisipan yang layak, langkah selanjutnya adalah menghitung biaya untuk rute tersebut sesuai dengan (3.2) sebagai berikut :

$$C = D + 0.01DW$$
$$= d_{05} + d_{51} + d_{10} + 0.01(d_{05} + d_{51} + d_{10})(d_{05} + d_{51} + d_{10} + s_0 + s_5 + s_1)$$
$$= 134.9830 + 0.01(134.9830)(314.9830)$$
$$= 560.1565$$

Hal yang sama juga diterapkan untuk pelanggan-pelanggan lain yang belum masuk rute, yaitu pelanggan 3 dengan posisi $0 - 3 - 5 - 0$ dan pelanggan 4 dengan posisi $0 - 4 - 5 - 0$, biaya penyisipan masing-masing pelanggan dihitung sesuai dengan formula (3.2), sehingga diperoleh pelanggan dengan tempat penyisipan yang layak dan biaya penyisipan pelanggan tersebut sebagai berikut:

Tabel 3.3 Posisi layak dan biaya penyisipan pelanggan k di R_1

k	Posisi di R_1	C
1	$0-5-1-0$	560.1565
3	$0-3-5-0$	415.5944
4	$0-4-5-0$	430.1470

Berdasarkan algoritma PFIH pada Gambar 3.1, pelanggan yang dipilih adalah pelanggan yang memenuhi kendala dan memiliki biaya penyisipan minimum, yaitu pelanggan 3. Selanjutnya rute R_1 diperbaharui menjadi $R_1: 0-3-5-0$ dengan total permintaan 25.

Proses penyisipan ini terus dilakukan hingga tidak ada lagi pelanggan yang dapat disisipkan pada rute R_1 sehingga dibutuhkan rute lainnya untuk melayani pelanggan yang belum masuk rute. Untuk masalah di atas, dengan menggunakan PFIH didapat solusi $S = \{R_1, R_2, R_3\}$ dengan $R_1 : 0-3-5-1-0$, $R_2 : 0-6-2-0$, dan $R_3 : 0-4-0$.

Selanjutnya akan diberikan contoh pencarian solusi lingkungan dari solusi PFIH dengan menggunakan *λ-Interchange*. Sebelum membentuk solusi lingkungan akan dicari terlebih dahulu biaya yang harus dikeluarkan untuk solusi S, sesuai dengan formula (3.3) didapat biaya $C(S) = 1270$.

Pembentukan lingkungan untuk solusi tersebut didapatkan melalui pertukaran pelanggan antar rute R_1 dan R_2 untuk setiap operator $(i,j), i,j \leq 2$ dan menghasilkan rute baru, kemudian dihitung biaya untuk masing-masing solusi lingkungan dan dibandingkan dengan biaya untuk solusi yang diperoleh dari metode PFIH. Berikut merupakan hasil pertukaran R_1 dan R_2 oleh operator $(1,0)$:

- $S_1' = \{R_1', R_2', R_3\}$, dimana $R_1' : 0-5-1-0$, $R_2' : 0-3-6-2-0$, dan $R_3 : 0-4-0$
- $S_2' = \{R_1', R_2', R_3\}$, dimana $R_1' : 0-5-1-0$, $R_2' : 0-6-3-2-0$, dan $R_3 : 0-4-0$
- $S_3' = \{R_1', R_2', R_3\}$, dimana $R_1' : 0-5-1-0$, $R_2' : 0-6-2-3-0$, dan $R_3 : 0-4-0$
- $S_4' = \{R_1', R_2', R_3\}$, dimana $R_1' : 0-3-1-0$, $R_2' : 0-5-6-2-0$, dan $R_3 : 0-4-0$

- $S_5' = \{R_1',\ R_2',\ R_3\}$, dimana $R_1' : 0-3-1-0$, $R_2' : 0-6-5-2-0$, dan $R_3 : 0-4-0$
- $S_6' = \{R_1',\ R_2',\ R_3\}$, dimana $R_1' : 0-3-1-0$, $R_2' : 0-6-2-5-0$, dan $R_3 : 0-4-0$
- $S_7' = \{R_1',\ R_2',\ R_3\}$, dimana $R_1' : 0-3-5-0$, $R_2' : 0-1-6-2-0$, dan $R_3 : 0-4-0$
- $S_8' = \{R_1',\ R_2',\ R_3\}$, dimana $R_1' : 0-3-5-0$, $R_2' : 0-6-1-2-0$, dan $R_3 : 0-4-0$
- $S_9' = \{R_1',\ R_2',\ R_3\}$, dimana $R_1' : 0-3-5-0$, $R_2' : 0-6-2-1-0$, dan $R_3 : 0-4-0$

Berdasarkan algoritma *λ-Interchange* pada Gambar 3.2, langkah selanjutnya adalah melakukan pengecekan terhadap S' secara berurut, S' yang memiliki biaya lebih kecil dari solusi yang didapat dari PFIH akan dimasukkan ke dalam populasi sebagai salah satu individu (kromosom), biaya S' dihitung berdasarkan formula (3.3). Pengecekan dimulai untuk S_1', biaya untuk S_1' adalah $C(S_1') = 1539.1$. Karena $C(S_1') > C(S)$, maka S_1' tidak diterima menjadi solusi.

Pengecekan ini dilakukan untuk setiap kemungkinan solusi lingkungan dari setiap operator, sehingga untuk masalah di atas diperoleh :

- Solusi 1 terdiri dari 3 rute, $R_1 : 0-3-5-1-0$, $R_2 : 0-6-2-0$, dan $R_3 : 0-4-0$
- Solusi 2 terdiri dari 3 rute, $R_1 : 0-5-0$, $R_2 : 0-6-2-0$, dan $R_3 : 0-4-3-1-0$
- Solusi 3 terdiri dari 3 rute, $R_1 : 0-3-1-0$, $R_2 : 0-6-2-0$, dan $R_3 : 0-4-5-0$

Setelah 50% populasi awal telah diperoleh, maka proses pencarian lingkungan berhenti, dan 50% lainnya dibangun secara acak. Misalkan dengan cara acak terpilih 3 solusi yang memenuhi kendala kapasitas dan *time windows*, yaitu :

- Solusi 4 terdiri dari 4 rute, $R_1 : 0-3-1-0$, $R_2 : 0-4-2-0$, $R_3 : 0-5-0$, dan $R_4 : 0-6-0$
- Solusi 5 terdiri dari 3 rute, $R_1 : 0-6-2-1-0$, $R_2 : 0-4-5-0$, dan $R_3 : 0-3-0$

- Solusi 6 terdiri dari 5 rute, $R_1 : 0-6-1-0$, $R_2 : 0-2-0$, $R_3 : 0-5-0$, $R_4 : 0-4-0$, dan $R_5 : 0-3-0$

Dengan demikian, populasi awal pada VRPTW telah terbentuk, sebelum masing-masing solusi dikodekan ke kromosom, terlebih dahulu dihitung nilai *fitness* untuk masing-masing individu. Nilai *fitness* dihitung berdasarkan fungsi biaya pada VRPTW yang direpresentasikan oleh total jarak. Akan diberikan contoh perhitungan nilai *fitness* pada solusi pertama, yaitu :

$$z_1 = d_{03} + d_{35} + d_{51} + d_{10} + d_{06} + d_{62} + d_{20} + d_{04} + d_{40}$$
$$= 12.8062 + 42.2966 + 62.6817 + 20.0998 + 30.8058 + 45.4863 + 45.2769 + 18.0278 + 18.0278$$
$$= 295.5089$$

Hal yang sama dilakukan untuk solusi lainnya, sehingga nilai *fitness* untuk masing-masing solusi dapat dilihat pada Tabel 3.4 :

Tabel 3.4 Nilai *fitness* Solusi pada populasi awal

Keturunan	*Fitness*
Solusi 1	295.5089
Solusi 2	301.1075
Solusi 3	296.3271
Solusi 4	327.3892
Solusi 5	297.3028
Solusi 6	346.4278

Selanjutnya, masing-masing individu ini dikodekan ke dalam kromosom dengan menggunakan pengkodean permutasi seperti yang telah dijelaskan pada Subbab 2.3.1.4. Pengkodean dilakukan dengan menggabungkan setiap pelanggan sesuai dengan urutannya di masing-masing rute tanpa adanya tanda untuk memisahkan rute yang satu dengan rute yang lainnya dan tidak memasukkan depot ke dalam kromosom. Misalkan untuk solusi 1, pengkodean yang diperoleh adalah $3-5-1-6-2-4$. Hal yang sama juga dilakukan untuk solusi-solusi lain, hasil pengkodean solusi-solusi tersebut dan nilai *fitness* masing-masing solusi dapat dilihat di Tabel 3.5.

Tabel 3.5 Pengkodean solusi ke kromosom dan nilai *fitness* masing-masing kromosom populasi awal

Solusi	Kromosom	Representasi Kromosom	*fitness*
1	Kromosom 1	$3-5-1-6-2-4$	295.5089
2	Kromosom 2	$5-6-2-4-3-1$	301.1075
3	Kromosom 3	$3-1-6-2-4-5$	296.3271
4	Kromosom 4	$3-1-4-2-5-6$	327.3892
5	Kromosom 5	$6-2-1-4-5-3$	297.3028
6	Kromosom 6	$6-1-2-5-4-3$	346.4278

3.3 Seleksi

Seperti yang telah dijelaskan pada Subbab 2.3.5, operator seleksi memilih orang tua yang akan dikenakan operator *crossover*.

Langkah pertama pada proses ini adalah mengurutkan kromosom sesuai dengan nilai *fitness* masing-masing. Kromosom-kromosom diurutkan dari yang memiliki nilai *fitness* paling kecil hingga yang memiliki nilai *fitness* paling besar, dan disimpan di dalam K.Berdasarkan Tabel 3.5, daftar K yang terbentuk adalah $K = \{K_1, K_3, K_5, K_2, K_4, K_6\}$. Kemudian dari daftar K tersebut, orang tua dipilih berdasarkan formula (2.11) pada Bab 2, yaitu :

$$\text{Pilih}(K) = \left\{ K_j \in K | j = P - \left\lfloor \frac{-1 + \sqrt{1 + 4.rand.(P^2 + P)}}{2} \right\rfloor \right\}$$

dimana P menyatakan banyaknya kromosom pada populasi, dalam masalah ini adalah 6 kromosom. Berdasarkan formula di atas, misalkan kromosom-kromosom orang tua yang terpilih adalah :

Tabel 3.6 Populasi Kromosom Orang Tua

j	Orang tua	Kromosom	Representasi kromosom
4	Orang tua 1	Kromosom 2	$5-6-2-4-3-1$
2	Orang tua 2	Kromosom 3	$3-1-6-2-4-5$
4	Orang tua 3	Kromosom 2	$5-6-2-4-3-1$
3	Orang tua 4	Kromosom 5	$6-2-1-4-5-3$
2	Orang tua 5	Kromosom 3	$3-1-6-2-4-5$
1	Orang tua 6	Kromosom 1	$3-5-1-6-2-4$

Dari populasi orang tua yang terpilih, selanjutnya memasuki proses *crossover* seperti yang akan dijelaskan pada Subbab 3.4 berikut.

3.4 *Crossover*

Orang tua - orang tua yang terpilih pada proses seleksi selanjutnya memasuki proses *crossover*. Dalam proses ini tidak semua orang tua mengalami proses *crossover*. Terpilihnya orang tua untuk mengalami *crossover* bergantung pada probabilitas *crossover* (p_c). Pada tugas akhir ini, digunakan nilai $p_c = 0.8$.

Tahapan pada *crossover* dimulai dengan mengambil dua orang tua berbeda secara berurutan kemudian membangkitkan bilangan acak antara 0 dan 1. Jika bilangan acak yang dibangkitkan lebih kecil atau sama dengan p_c, maka sepasang orang tua yang terpilih akan memasuki proses *heuristic crossover* dan *merge crossover*, operator yang akan digunakan adalah HX1 dan MX1. Kromosom orang tua yang tidak mengalami proses *crossover* akan disalin ke kromosom keturunan.

Misalkan orang tua yang terpilih adalah orang tua 1 dan 2, yaitu $5-6-2-4-3-1$ dan $3-1-6-2-4-5$. Kemudian dibangkitkan bilangan acak, misalkan bilangan acak yang dibangkitkan lebih kecil dari p_c, maka kedua orang tua ini akan memasuki proses *heuristic* dan *merge crossover*. Akan dijelaskan proses pembentukan kromosom keturunan menggunakan *heuristic crossover*. Tahap pertama adalah memilih titik *crossover* untuk kedua orang tua, misalkan titik *crossover* yang terpilih adalah sebagai berikut :

Orang tua 1 : $5-6-2-4-3 \mid 1$

Orang tua 2 : $3-1-6-2-4 \mid 5$

Gen pertama pada kromosom keturunan dipilih dari gen setelah titik *crossover* pada kedua orang tua yaitu gen 1 atau gen 5. Misalkan gen yang tepilih adalah gen 5, selanjutnya gen 1 dan gen 5 pada orang tua 1 ditukar, sehingga orang tua 1 menjadi $1-6-2-4-3 \mid 5$. Kemudian gen selanjutnya diperoleh dari gen setelah 5 pada kedua orang tua yang memiliki jarak terdekat dengan 5, yaitu pelanggan 3 karena $d(5,3) < d(5,1)$ (data jarak dapat dilihat di Tabel 3.2). Sehingga kromosom keturunan yang terbentuk hingga saat ini terdiri dari gen 5 dan gen 3. selanjutnya gen 1 dan gen 3 pada orang tua 1 ditukar, sehingga orang tua 1 menjadi $3-6-2-4-1 \mid 5$. Hal tersebut dilakukan berulang-ulang

hingga mencapai gen sebelum titik *crossover*, sehingga kromosom keturunan yang terbentuk adalah :

Keturunan 1 : $5-3-1-6-4-2$

Orang tua berikutnya yang terpilih adalah orang tua 3 dan 4, kemudian orang tua 5 dan 6. Setelah melalui proses *heuristic crossover*, selanjutnya kedua orang tua akan memasuki proses *merge crossover*. Tahapan pada metode ini sama dengan *heuristic crossover*, hanya saja pemilihan gen didasarkan pada *time windows*. Berikut diberikan Tabel 3.7 yang berisi keturunan yang diperoleh dari hasil *crossover*.

Tabel 3.7 Kromosom Keturunan hasil *crossover*

Keturunan	Representasi kromosom
Keturunan 1	$5-3-1-6-4-2$
Keturunan 2	$4-3-1-2-5-6$
Keturunan 3	$4-3-1-6-2-5$
Keturunan 4	$5-3-1-2-6-4$
Keturunan 5	$4-2-3-1-6-5$
Keturunan 6	$4-3-1-5-6-2$

Setelah melalui proses *crossover*, kromosom anak selanjutnya memasuki proses mutasi. Pada tugas akhir ini digunakan operator *sequence based mutation* yang akan dijelaskan pada Subbab 3.5 berikut.

3.5 Mutasi

Metode *sequence based mutation* (SBM) melibatkan dua kromosom keturunan hasil *crossover* untuk mengalami mutasi. Kromosom-kromosom keturunan tersebut dikodekan kembali ke solusi yang merupakan himpunan rute pelanggan. Tahap pertama, metode ini memilih secara acak *break point* yaitu titik di antara dua pelanggan di salah satu rute pada masing - masing solusi. Solusi yang baru didapat dengan cara menghubungkan pelanggan yang dilayani sebelum *break point* pada solusi 1 dengan pelanggan yang dilayani setelah *break point* pada solusi 2.

Solusi yang dihasilkan pada proses ini tidak selalu memenuhi, artinya tidak semua pelanggan akan masuk rute atau terdapat pelanggan yang muncul dua kali dalam satu solusi. Oleh karena itu operator perbaikan diterapkan sebagai berikut pada proses ini :

- Jika pelanggan muncul dua kali pada satu rute, salah satu dari keduanya dihilangkan dari rute. Jika pelanggan muncul di rute hasil mutasi dan muncul di rute yang lama, pelanggan dihilangkan dari rute yang lama.
- Jika pelanggan belum masuk rute, maka pelanggan ini disisipkan di tempat yang meminimumkan biaya dan memenuhi kendala kapasitas dan *time windows*. Jika tidak ada tempat untuk dilakukan penyisipan, maka solusi ini tidak diterima.

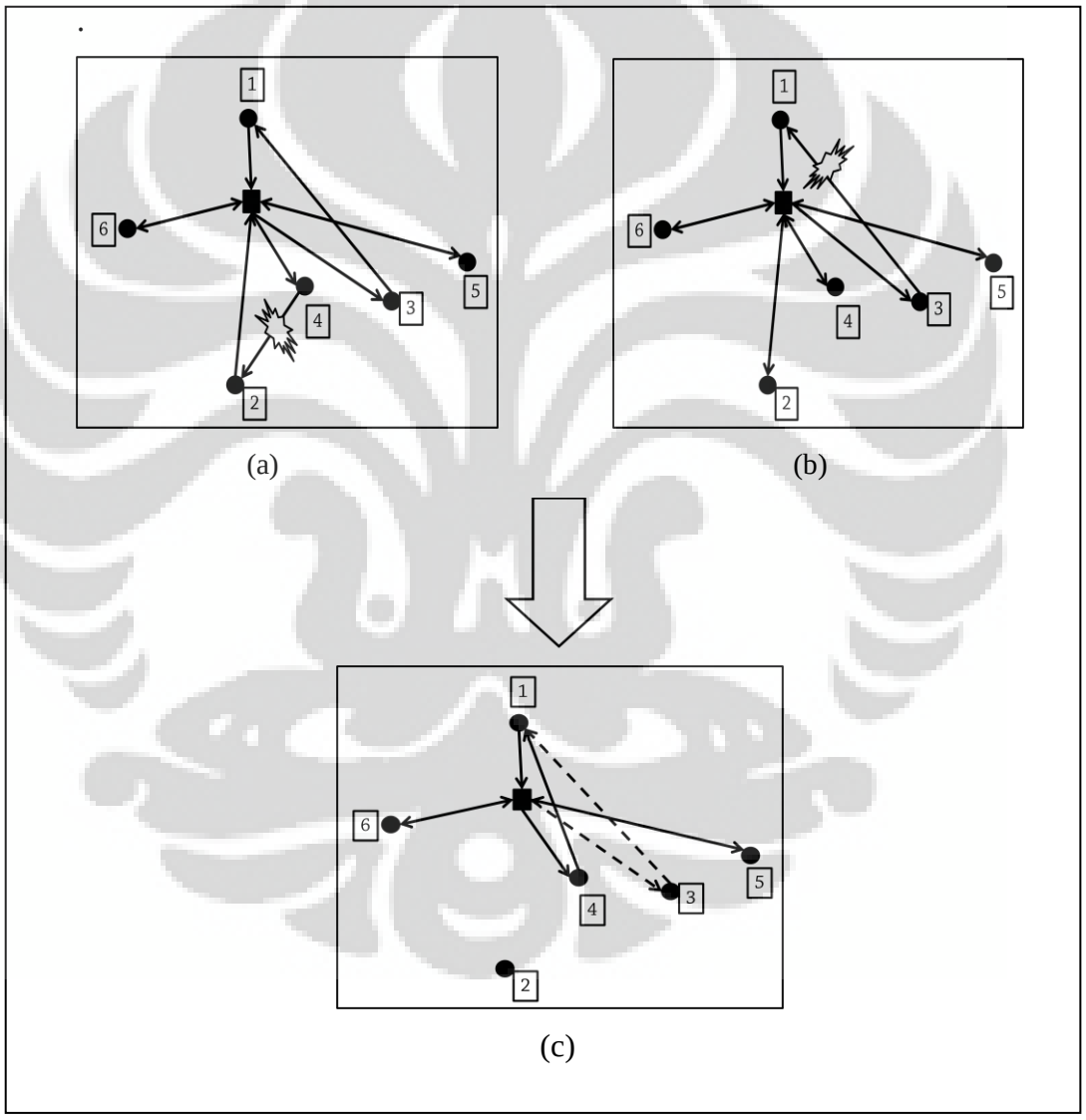
Akan diberikan contoh penerapan operator *sequence based mutation* untuk kromosom keturunan 1 dan keturunan 2 yang diperoleh dari *crossover* di Subbab 3.4, kedua kromosom ini dikembalikan ke bentuk solusi menjadi :

- Solusi 1 terdiri dari 4 rute, $R_1: 0-5-0$, $R_2: 0-3-1-0$, $R_3: 0-6-0$, dan $R_4: 0-4-2-0$
- Solusi 2 terdiri dari 5 rute, $R_1: 0-4-0$, $R_2: 0-3-1-0$, $R_3: 0-2-0$, $R_4: 0-5-0$, dan $R_5: 0-6-0$

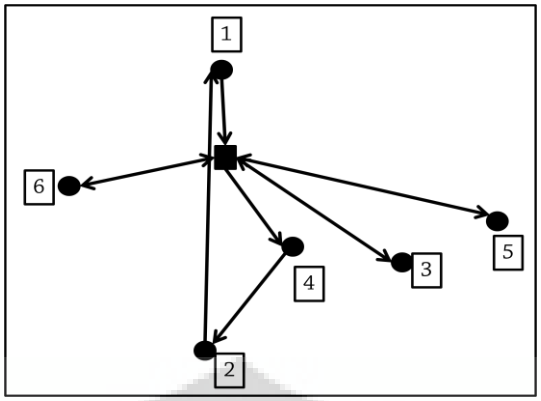
Kemudian dipilih secara acak rute pada masing-masing solusi, misalkan terpilih rute R_4 pada solusi 1 dan rute R_2 pada solusi 2, dipilih pula *break point* pada masing-masing rute tersebut, misalkan terpilih *break point* diantara pelanggan 4 dan pelanggan 2 pada rute R_4 yang digambarkan pada Gambar 3.4(a) dan *break point* diantara pelanggan 3 dan pelanggan 1 pada rute R_2 yang digambarkan pada Gambar 3.4(b). Rute pertama pada solusi baru hasil mutasi merupakan gabungan antara gen sebelum *break point* pada rute R_4 dan gen setelah *break point* pada rute R_2, yaitu $R_1': 0-4-1-0$, sedangkan rute lainnya dibentuk dengan menyalin rute R_1, R_2 dan R_3 pada solusi 1. Solusi hasil mutasi yang dihasilkan adalah $R_1: 0-4-1-0$, $R_2: 0-5-0$, $R_3: 0-3-1-0$, dan $R_4: 0-6-0$ yang dapat dilihat pada Gambar 3.4(c).

Solusi tersebut tidak memenuhi karena pelanggan 1 muncul dua kali pada R_1 dan R_3. Sehingga diperlukan operator perbaikan yaitu dengan menghapus pelanggan 1 pada R_3, dan solusi menjadi $R_1: 0-4-1-0$, $R_2: 0-5-0$,

$R_3: 0-3-0$, dan $R_4: 0-6-0$. Namun solusi tersebut masih belum memenuhi karena pelanggan 2 belum masuk rute manapun, sehingga diperlukan operator perbaikan, yaitu dengan menyisipkan pelanggan 2 di tempat yang layak dan meminimumkan biaya berupa total jarak. Solusi hasil mutasi yang diperoleh adalah $R_1: 0-4-2-1-0$, $R_2: 0-5-0$, $R_3: 0-3-0$, dan $R_4: 0-6-0$. Gambar 3.5 memperlihatkan hasil penerapan operator perbaikan pada solusi hasil SBM.



Gambar 3.4 Proses SBM, (a) Solusi 1, (b) Solusi 2, (c) solusi hasil penerapan SBM



Gambar 3.5 Hasil Solusi penerapan operator perbaikan

Solusi ini kemudian dikodekan kembali ke kromosom, menjadi $4-2-1-5-3-6$. Hal yang sama dilakukan untuk kromosom-kromosom keturunan lainnya, hasil pengkodean kromosom-kromosom keturunan dapat dilihat pada Tabel 3.8.

Tabel 3.8 Kromosom Keturunan hasil mutasi

Keturunan	Representasi kromosom
Keturunan 1	$4-2-1-5-3-6$
Keturunan 2	$5-3-1-6-4-2$
Keturunan 3	$4-3-6-2-1-5$
Keturunan 4	$5-3-1-2-6-4$
Keturunan 5	$4-1-3-2-6-5$
Keturunan 6	$4-3-2-5-6-1$

3.6 Pembentukan Populasi untuk Generasi Berikutnya

Pembentukan populasi untuk generasi selanjutnya didasarkan pada nilai *fitness*. Kromosom keturunan hasil *crossover* dan mutasi akan dimasukkan ke dalam populasi selanjutnya jika memiliki nilai *fitness* lebih kecil daripada kromosom dengan *fitness* terbesar pada populasi sekarang dimana kromosom keturunan yang akan dimasukkan ke populasi harus berbeda dengan masing-masing kromosom yang ada di populasi. Ukuran populasi selanjutnya akan selalu sama dengan *popsize* pada pembentukan populasi awal.

Akan dijelaskan proses pembentukan populasi baru untuk contoh masalah 3.1. Sebelumnya dihitung terlebih dahulu nilai *fitness* untuk masing-masing keturunan hasil *crossover* dan mutasi yang disajikan pada Tabel 3.9.

Tabel 3.9 Nilai *fitness* Keturunan hasil *crossover* dan mutasi

Keturunan	Representasi kromosom	*Fitness*
Keturunan 1	$4-2-1-5-3-6$	308.5725
Keturunan 2	$5-3-1-6-4-2$	327.3892
Keturunan 3	$4-3-6-2-1-5$	327.5321
Keturunan 4	$5-3-1-2-6-4$	357.1529
Keturunan 5	$4-1-3-2-6-5$	336.5617
Keturunan 6	$4-3-2-5-6-1$	325.6810

Nilai *fitness* masing-masing kromosom kemudian dibandingkan dengan nilai *fitness* populasi awal yang dapat dilihat pada Tabel 3.5. Kromosom yang memiliki *fitness* terbesar pada populasi awal adalah kromosom 6 dengan nilai *fitness* 346.4278. Perbandingan kromosom keturunan dimulai dari kromosom keturunan 1, karena nilai *fitness* kromosom keturunan 1 lebih kecil dari 346.4278, maka kromosom keturunan 1 masuk ke populasi baru menggantikan kromosom 6. Selanjutnya dilakukan perbandingan nilai *fitness* kromosom keturunan 2 dengan kromosom yang memiliki *fitness* terbesar berikutnya pada populasi awal, yaitu kromosom 4 dengan nilai *fitness* 327.3892. Karena nilai *fitness* kromosom keturunan 2 sama dengan 327.3892, maka kromosom ini tidak masuk ke populasi baru. Perbandingan tersebut terus dilakukan hingga kromosom keturunan terakhir. Dan kromosom-kromosom yang terpilih untuk populasi berikutnya dapat dilihat pada Tabel 3.10.

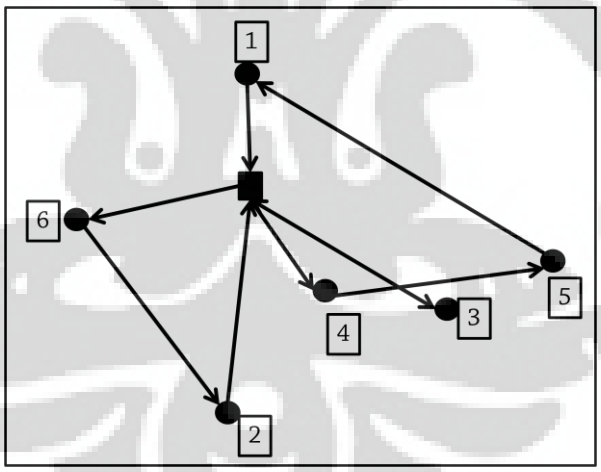
Proses dalam algoritma genetika akan terus berlanjut hingga kriteria berhenti terpenuhi. Kriteria berhenti yang digunakan pada tugas akhir ini bergantung pada generasi yang dicapai, jika generasi ke-n telah tercapai maka algoritma genetika akan berhenti, dan menerima solusi sebagai solusi optimal. Untuk contoh masalah di atas, nilai n yang digunakan adalah 60.

Tabel 3.10 Populasi pada generasi ke-2

Keturunan	Representasi kromosom	*Fitness*
Kromosom 1	$3-5-1-6-2-4$	295.5089
Kromosom 2	$5-6-2-4-3-1$	301.1075
Kromosom 3	$3-1-6-2-4-5$	296.3271
Kromosom 4	$4-3-6-2-1-5$	325.6810
Kromosom 5	$6-2-1-4-5-3$	297.3028
Kromosom 6	$4-2-1-5-3-6$	308.5725

3.7 Tahap Perbaikan

Setelah mencapai generasi ke-60, 10% dari populasi yang didapat dari algoritma genetika diambil secara acak kemudian dikodekan kembali menjadi solusi berupa kumpulan rute, lalu solusi tersebut dikenakan proses 1-*Interchange*, dan 2-*Interchange*. Kemudian dipilih kromosom dengan nilai *fitness* terbaik, sehingga didapat solusi untuk masalah 3.1 adalah 3 rute dengan $R_1: 0-\ 6-2-0$, $R_2: 0-3-0$, $R_3: 0-4-5-\ 1-0$ dan nilai *fitness* 287.9908. Solusi tersebut dapat digambarkan dengan graf seperti Gambar 3.6 berikut ini :



Gambar 3.6 Gambar Solusi VRPTW untuk contoh masalah 3.1

3.8 Algoritma Genetika Hibrida untuk VRPTW

Setelah diberikan penjelasan mengenai algoritma genetika pada Subbab 3.1 sampai dengan Subbab 3.8, berikut akan ditampilkan algoritma genetika hibrida (AGH) untuk menyelesaikan VRPTW :

Tahap 1. *Input* data

Tahap 2. Bentuk Populasi awal dengan PFIH, *λ-Interchange* dan acak

Untuk generasi = 1 *to* max_generasi *do*

Hitung nilai *fitness* untuk setiap kromosom pada populasi

Bentuk Populasi Orang tua dengan Seleksi

Lakukan *Crossover* dan mutasi jika memenuhi ketentuan

Bentuk populasi baru

Tahap 3. Jika telah mencapai generasi ke-*n*

Lakukan Tahap perbaikan, solusi yang dipilih adalah solusi dengan nilai *fitness* terkecil, lanjutkan ke tahap 4

Jika tidak

Kembali ke tahap 2

Tahap 4. STOP

Gambar 3.7 AGH untuk VRPTW

BAB 4
IMPLEMENTASI PROGRAM

Pada bab sebelumnya telah dijelaskan mengenai penyelesaian masalah *Vehicle Routing Problem with Time Windows* menggunakan algoritma genetika hibrida. Pada bab ini algoritma genetika hibrida yang telah dijelaskan pada bab sebelumnya akan diimplementasikan kedalam perangkat lunak yaitu menggunakan Matlab R2010a. Implementasi dilakukan pada komputer dengan spesifikasi sebagai berikut :

Processor : Pentium(R) E5500, Dual-Core @2.80GHz
Memory : 1.9GB
OS : Kernel Linux 2.6.35-31-generic
Ubuntu Release 10.10 (maverick)
HD free Space : 157.9 GB

Data *benchmark* Solomon yang digunakan terdiri dari 25 pelanggan dengan tipe C101 (Solomon), dan diasumsikan jarak antar pelanggan simetris artinya $d_{ij} = d_{ji}$. Berikut adalah parameter-parameter yang digunakan pada algoritma genetika hibrida :

- Kapasitas Kendaraan = 200
- Ukuran populasi = 25
- Jumlah generasi = 1000
- Probabilitas *crossover* = 0.8
- Probabilitas mutasi = 0.2
- Banyaknya kromosom yang masuk tahap perbaikan = 3

Penerapan algoritma genetika hibrida dimulai dengan membentuk populasi awal berupa himpunan kromosom sebanyak ukuran masalah yaitu 25. Pembentukan 50% dari populasi awal dilakukan dengan metode *Push Forward Insertion Heuristic* yang menghasilkan 1 solusi dan λ-*Interchange* yang

menghasilkan 12 solusi, sementara 50% sisanya dibangun secara acak yang menghasilkan 12 solusi. Kemudian untuk masing-masing solusi akan dihitung nilai *fitness*nya, pada Tabel 4.1 ditampilkan nilai *fitness* untuk masing-masing solusi pada populasi awal.

Tabel 4.1 Populasi Awal

Solusi	*fitness*	Solusi	*fitness*	Solusi	*fitness*	Solusi	*fitness*
1	335.0433	8	316.1451	15	742.2363	22	665.0012
2	284.1551	9	331.1648	16	721.3681	23	704.3831
3	284,5201	10	330.4445	17	693.9417	24	591.8480
4	318.0908	11	330.4445	18	739.8586	25	562.5504
5	307.6484	12	334.3919	19	688.4422		
6	329.9907	13	655.4159	20	664.7723		
7	316.1451	14	604.3056	21	707.4329		

Dapat dilihat bahwa berdasarkan nilai *fitness*, solusi yang dihasilkan oleh PFIH dan *λ-Interchange* lebih baik dari solusi yang dihasilkan secara acak. Solusi terbaik pada populasi awal adalah solusi ke-2 dengan nilai *fitness* 284.1551.

Solusi-solusi pada populasi awal tersebut kemudian dikodekan ke kromosom dan memasuki proses seleksi, *crossover*, dan mutasi. Proses tersebut dilakukan hingga populasi ke-1000 telah tercapai. Kemudian setiap kromosom dalam populasi tersebut dikodekan kembali ke solusi dan dihitung nilai *fitness* masing-masing solusi. Tabel 4.2 berikut menampilkan nilai *fitness* generasi ke-1000.

Tabel 4.2 Generasi ke-1000

Solusi	*fitness*	Solusi	*fitness*	Solusi	*fitness*	Solusi	*fitness*
1	215.9651	8	222.0709	15	218.8043	22	215.7365
2	211.8136	9	221.9381	16	213.9791	23	220.1988
3	219.5347	10	213.9791	17	219.3230	24	212.1772
4	217.5713	11	223.0819	18	223.0960	25	213.5710
5	219.9464	12	214.3421	19	191.8236		
6	220.4971	13	221.6369	20	218.0889		
7	215.7037	14	216.7272	21	221.8452		

Dapat dilihat dari Tabel 4.2 bahwa solusi-solusi yang memiliki nilai *fitness* lebih kecil akan menggantikan solusi-solusi yang memiliki nilai *fitness* besar pada

populasi awal, sehingga algoritma genetika terbukti efektif untuk menurunkan total jarak.

Dari Tabel 4.2, solusi dengan nilai *fitness* terbaik akan diambil menjadi solusi optimal. Dari Tabel 4.2, untuk masalah C101, diperoleh solusi dengan nilai total jarak **191.8136** dan jumlah rute **3**, berikut rute untuk solusi C101 :

- $R_1 : 0 - 5 - 3 - 7 - 8 - 10 - 11 - 9 - 6 - 4 - 2 - 1 - 0$
- $R_2 : 0 - 20 - 24 - 25 - 23 - 22 - 21 - 0$
- $R_3 : 0 - 13 - 17 - 18 - 19 - 15 - 16 - 14 - 12 - 0$

Solusi tersebut mendekati solusi optimal yang diketahui yaitu terdiri dari 3 rute dan total jarak 191.3. (Solomon).

BAB 5
KESIMPULAN DAN SARAN

5.1 Kesimpulan

Berdasarkan pembahasan yang telah di jelaskan pada Bab 1 sampai Bab 4, kesimpulan yang dapat diambil dari tugas akhir ini adalah :

- Pada tugas akhir ini algoritma genetika yang digabungkan dengan *Push Forward Insertion Heuristic* dan *λ-Interchange* untuk pembentukan solusi awal atau dikenal sebagai algoritma genetika hibrida (AGH). AGH dapat digunakan untuk menyelesaikan VRPTW dengan menggunakan tiga operator utama yang digunakan adalah *ranking based selection, merge-heuristic crossover*, dan *sequence based mutation*.
- Pada implementasi algoritma genetika untuk VRPTW dengan data *benchmark solomon*, solusi awal yang terbentuk memiliki nilai *fitness* 284.1551. setelah dilakukan seleksi, *crossover*, dan mutasi sebanyak 1000 iterasi diperoleh solusi akhir dengan nilai *fitness* 191.8136. sesuai dengan hasil yang didapat dari implementasi tersebut algoritma genetika hibrida terbukti cukup efektif mengurangi total jarak.

5.2 Saran

Saran dalam pengembangan tugas akhir ini adalah :

- Algoritma pada tugas akhir ini hanya terfokus pada menurunkan total jarak, sebaiknya untuk penelitian selanjutnya dapat menerapkan algoritma genetika untuk meminimumkan banyak kendaraan yang digunakan dan total jarak tempuh kendaraan
- Generasi pada AG ditambahkan sehingga kemungkinan solusi mendekati solusi optimal akan lebih besar

DAFTAR PUSTAKA

Amini, S., Javanshir, H., & Tavakkoli-Moghaddam, R. (2010). A Pso Approach for Solving VRPTW with Real Case Study. *IJRRAS* .

Coley, D. A. (1999). *An Introduction to Genetic Algorithms for Scientists and Engineers.* World Scientific Publishing Co. Pte. Ltd.

Ghoseiri, K., & Ghannadpour, S. F. (2010). Multi-Objective Vehicle Routing Problem with Time Windows using Goal Programming and Genetic Algorithm. *Applied Soft Computing 10* , 1096-1107.

Ghoseiri, K., & Ghanndpour, S. F. (2009). Hybrid Genetic Algorithm for Vehicle Routing and Scheduling Problem. *Journal of Applied Science 9 (1)* , 79-87.

Ho, W.-K., Lim, A., & Oon, W.-C. (2001). *Maximizing Paper Spread in Examination Timetabling Using a Vehicle Routing Method.* singapore: School of Computing, National University of singapore.

Jr., J. L., & Wainwright, R. L. (1993). Multiple Vehicle Routing with Time and Capacity Constraint Using Genetic Algorithms. *ICGA-93* , 452-459.

Laporte, G. (1992). The Vehicle Routing Problem : An Overview off exact and approximate algorithms. *European Journal of Operational Research 59* , 345-358.

Obitko, M. (1998). Dipetik April 2012, dari www.obitko.com/tutorials/genetic-algorithms.

Sarwadi, & KSW, A. (2004). Algoritma Genetika untuk Penyelesaian masalah Vehicle Routing. *Jurnal Matematika dan komputer 7* , 1-10.

Solomon, M. M. Diambil kembali dari http://w.cba.neu.edu/~msolomon.

Solomon, M. M., Larsen, J., Kallehauge, B., & Madsen, O. B. (2005). Vehicle Routing Problem with Time Windows. Dalam G. Desaulniers, J. Desrosiers, & M. S. Marius, *Column Generation.* Springer Science + Bussines Media.

Toth, P., & Vigo, D. (2002). *The Vehicle Routing Problem.* United Stated of America: Societyfor Industrial and Applied Mathematics.

Whitley, D. (1994). A Genetic Algorithm. *Statistics and Computing (4)* , 65-85.

Whitley, D. (2002). Genetic Algorithms Evolutionary Computing. *Van Nostrand's Scientific Encyclopedia* .

Yeun, L. C., Ismail, R. W., Omar, K., & Zirour, M. (2008). Vehicle Routing Problem : Model and Solution. *Journal of Quality Measurement and Analysis* , 205-218.

LAMPIRAN

Lampiran 1. *Source Code* Algoritma Genetika Hibrida

ag.m

```
clear; tic;

%% pembentukan Populasi Awal
PFIH;  lambda;  solrand;

%% algoritma genetika
for i = 1000
    solkrom;    seleksi;    silang;    kromrut;    mutasi;    load
solusi.mat;

%% Pembentukan populasi baru
    for j = 1:length(anm)
        [brk ind] = max(ttj);
        if ti(j) < brk
            masuk = 1;
            for cek = 1:length(S)
                if length(anm{j}) == length(S{cek})
                    if isequal(jari{j},jar{cek})
                        masuk = 0;
                        break;
                    end
                end
            end
            if masuk == 1
                S{ind} = anm{j};   ttj(ind) = ti(j);
                jar{ind} = jari{j};
                save solusi.mat S ttj jar;
            end
        end
    end
end

%% 10% solusi dikenakan lambda interchange
ran = randi(length(S),1,ceil(0.1*length(S)));
for j = ran
    R = S{j};
    for i=1:length(R)
        D = 0; W = 0;
        for j = 1:length(R{i})-1
            D = D + d(R{i}(j),R{i}(j+1));
            W = d(R{i}(j),R{i}(j+1)) + W + f(R{i}(j),7);
        end
        s(i) = D;    cR(i) = D + 0.01*D*W;
    end
    ttk = sum(s);    pop = 2;
    save data.mat R cR d f Q ttk pop;
    lambda
    solsem = S{2};    jarsem = jrkrute(solsem,d);
    load solusi.mat
    masuk = 1;
```

```
    for cek = 1:length(S)
        if length(solsem) == length(S{cek})
            if isequal(jar{cek},jarsem)
                masuk = 0;
                break;
            end
        end
    end
    if masuk == 1
        S{j} = solsem;    jar{j} = jarsem;
    end
end

%% menghitung nilai fitness solusi akhir
for p=1:length(S)
    jaru{p} = jrkrute(S{p},d);    ttj(p) = sum(jaru{p});
end

%% solusi dengan fitness terkecil merupakan solusi optimal
[biaya inde]= min(ttj);    solusi = S{inde};      biaya;
waktu=toc;
```

PFIH.m

```
clear;

%% data pelanggan
f = load('C101_25.txt');
[x y] = size(f);  u = zeros(1,x);  u(1)=1; Q = 200;
pop = (x-1)/2;
for i=1:x
    for j=i+1:x
        d(i,j)= sqrt((f(j,2)-f(i,2))^2 + (f(j,3)-f(i,3))^2);
        d(j,i) = d(i,j);
    end
end
p = atan((f(:,3) - f(1,3))./(f(:,2) - f(1,2)));
plr = p*unitsratio('degree','radian');
c = -0.7*d(:,1) + 0.1*f(:,6) + 0.2*(plr/360).* d(:,1);
c(1)= 0; rute = 1;
while sum(u) < numel(u)

%% mencari pelanggan pertama
    r = 1; rpos = 1; umin = u; jalan = 1;
    while jalan == 1
        cmin = min(c(umin == 0));
        for j = find(umin==0)
            if (c(j) == cmin)
                break
            end
        end
        at(j)=d(1,j);
        if (f(j,4)< Q && at(j)< f(j,6))
            rpos = rpos + 1;   r(rpos) = j;   u(j)=1;  jalan = 0;
        end
        umin(j)= 1;       u(j) = 1;
    end
    r(rpos+1) = 1;
```

```
%% menghitung waktu mulai pelayanan masing" pelanggan rute
sekarang
    ms = 0;
    for i=1:length(r)-1
        at(i+1) = ms(i) + f(r(i),7) + d(r(i),r(i+1));
        if at(i+1) < f(r(i+1),5);
            ms(i+1) = f(r(i+1),5);
        elseif at(i+1) >= f(r(i+1),5);
            ms(i+1) = at(i+1);
        end
    end

%% Penyisipan pelanggan
    sisip = 1; ye = 0;
    while sisip == 1;
        rfix = r; msfix = ms; bia = inf;
        for i = find(u==0)
            for j = 1:length(r)-1
                r1 = ins(r,i,j);    ms1 = ins(ms,0,j);

%% cek kendala Kapasitas dan Time Windows
                if sum(f(r1,4))<=Q
                    pfoke = 1;
                    for k = j+1:length(r1)
                        at(k) = ms1(k-1)+f(r1(k-1),7)
                                + d(r1(k-1),r1(k));
                        if at(k) < f(r1(k),5);
                            ms1(k) = f(r1(k),5);
                        elseif at(k) >= f(r1(k),5);
                            ms1(k) = at(k);
                        end
                        if ms1(k) > f(r1(k),6)
                            pfoke = 0;
                            break
                        elseif k ~= j+1 && ms1(k) - ms(k-1) == 0
                            break
                        end
                    end

%% menghitung biaya penyisipan
                    if pfoke ==1
                        D = 0; s = 0;
                        for k = 1:length(r1)-1
                            D = D + d(r1(k),r1(k+1));
                            s = s + f(r1(k),7);
                        end
                        W = D + s;
                        if D + 0.01*D*W < bia
                            bia = D + 0.01*D*W;   rfix = r1;
                            msfix = ms1;
                        end
                    end
                end
            end
        end
        if isequal(r, rfix)
            sisip = 0;
        else
            u(rfix)=1;   r = rfix;   ms = msfix;
```

```
        end
        bia = inf;
    end
    R{rute} = r    MS{rute} = ms;    rute = rute + 1;
end

%% menghitung biaya solusi untuk lambda interchange
T = 0; O = 0;
for i=1:length(R)
    D = 0; W = 0;
    for j = 1:length(R{i})-1
        D = D + d(R{i}(j),R{i}(j+1));
    end
    W = D+sum(f(R{i},7));    s(i) = D;    cR(i) = D + 0.01*D*W;
end
ttk = sum(s); cS = sum(cR);
save data.mat R cR MS d f Q ttk pop;
```

lambda.m

```
clear; load data.mat
n = numel(R);  com = combntns(1:n,2); [a b] = size(com);   S{1} =
R;   sol = 0;
for k = 1:a
    s = com(k,1);    m = com(k,2);
    ls = length(R{s});    lm = length(R{m});

%% Operator (0,1)
    for i = 2:lm-1
        for j = 1:ls-1
            R{s} = ins(R{s},R{m}(i),j);
            R{m} = del(R{m},i);
            uji
            load data.mat R
        end
    end
    load data.mat R

%% Operator (1,0)
        for i = 2:ls-1
            for j = 1:lm-1
                R{m} = ins(R{m},R{s}(i),j);
                R{s} = del(R{s},i);
                uji
                load data.mat R
            end
        end
        load data.mat R

%% operator (1,1)
    for i = 2:length(R{s})-1
        for j = 2:length(R{m})-1
            temp = R{s}(i);
            R{s}(i) = R{m}(j);
            R{m}(j) = temp;
            uji
            load data.mat R
        end
```

```
    end
    load data.mat R

%% Operator (0,2)
    for mi = 2:length(R{m})-1
        for mj = mi+1:length(R{m})-1
            for si = 1:length(R{s})-1
                Rs = ins(R{s},R{m}(mi),si);
                for sj = si+1:length(R{s})
                    R{s} = Rs;
                    R{s} = ins(R{s},R{m}(mj),sj);
                    R{m} = del(R{m},mj);
                    R{m} = del(R{m},mi);
                    uji
                    load data.mat R
                end
            end
        end
    end
    load data.mat R

%% Operator (2,0)
    for si = 2:length(R{s})-1
        for sj = si+1:length(R{s})-1
            for mi = 1:length(R{m})-1
                Rm = ins(R{m},R{s}(si),mi);
                for mj = mi+1:length(R{m})
                    R{m} = Rm;
                    R{m} = ins(R{m},R{s}(sj),mj);
                    R{s} = del(R{s},sj);
                    R{s} = del(R{s},si);
                    uji
                    load data.mat R
                end
            end
        end
    end
    load data.mat R

%% operator (2,1)
    if length(R{s}) > 3 && length(R{m}) > 2
        scom = nchoosek(2:length(R{s})-1,2);
        for i = 1:nchoosek(length(R{s})-2,2)
            stemp = R{s}(scom(i,:));
            for j = 2:length(R{m})-1
                mtemp = R{m}(j);
                kali = 2;
                if scom(i,2) == scom(i,1) + 1
                    kali = 1;
                end
                for k = 1:kali
                    R{m} = ins(R{m},stemp,j);
                    R{m} = del(R{m},j);
                    R{s} = del(R{s},scom(i,2));
                    R{s} = del(R{s},scom(i,1));
                    R{s} = ins(R{s},mtemp,scom(i,k)-k);
                    uji
                    load data.mat R
                end
```

```
            end
        end
    end
    load data.mat R

%% operator (1,2)
    if length(R{m}) > 3 && length(R{s}) > 2
        scom = nchoosek(2:length(R{m})-1,2);
        for i = 1:nchoosek(length(R{m})-2,2)
            stemp = R{m}(scom(i,:));
            for j = 2:length(R{s})-1
                mtemp = R{s}(j);
                kali = 2;
                if scom(i,2) == scom(i,1) + 1
                    kali = 1;
                end
                for k = 1:kali
                    R{s} = ins(R{s},stemp,j);
                    R{s} = del(R{s},j);
                    R{m} = del(R{m},scom(i,2));
                    R{m} = del(R{m},scom(i,1));
                    R{m} = ins(R{m},mtemp,scom(i,k)-k);
                    uji
                    load data.mat R
                end
            end
        end
    end
    load data.mat

%% operator (2,2)
    if length(R{s}) > 3 && length(R{m}) > 3
        scom = nchoosek(2:length(R{s})-1,2);
        mcom = nchoosek(2:length(R{m})-1,2);
        for i = 1:nchoosek(length(R{s})-2,2)
            temp = R{s}(scom(i,:));
            for j = 1:nchoosek(length(R{m})-2,2)
                R{s}(scom(i,:)) = R{m}(mcom(j,:));
                R{m}(mcom(j,:)) = temp;
                uji
                load data.mat R
            end
        end
    end
    load data.mat R
end
in = 1;
for i = 1:length(solu)
    penuh = cektw(solu{i},f,d,Q);
    if penuh == 1
        sollam{in} = solu{i};  cost(in) = bia(i);  in = in + 1;
    end
end

for i = 1:length(sollam)
    jarak(i) = sum(jrkrute(sollam{i},d));
end
```

```
[v x] = sort(jarak); sollam = sollam(x); cost = cost(x); jarak =
jarak(x);
for i = 1:pop-1
    S{i+1} = sollam{i};
end
save sol.mat S;
```

solrand.m

```
clear; load data.mat; load sol.mat
[a b] = size(f); tot = a-1; sol = floor((a-1)/2 + 1);

%% Solusi Acak
while sol <= tot
    u = [1 zeros(1,tot)];  rute = 1;
    while sum(u) < length(u)
        r = [1 1]; usol = u;  at = 0; ms=0;
        while sum(usol) < length(usol);
            ac = find(usol==0);  k = acak(ac);
            pos = numel(r)-1;   rtemp = ins(r,k,pos);
            if sum(f(rtemp,4))<= Q
                attemp = at;   mstemp = ms;
                at = ms + f(rtemp(pos),7)
                     + d(rtemp(pos),rtemp(pos+1));
                if at <= f(rtemp(pos+1),5);
                    ms = f(rtemp(pos+1),5);
                elseif at > f(rtemp(pos+1),5)
                    ms = at;
                end
                if ms <= f(rtemp(pos+1),6)
                    at_d = ms + f((pos+1),7) + d(rtemp(pos+1),1);
                    if at_d <= f(1,6)
                        r = rtemp;  usol(k) = 1;
                    else
                        usol(k) = 1; at = attemp; ms = mstemp;
                    end
                else
                    usol(k) = 1; at = attemp; ms = mstemp;
                end
            else
                usol(k) = 1;
            end
        end
        Stemp{rute} = r;  u(r) = 1;  rute = rute+1;
    end

    for j=1:numel(S)
        if isequal(Stemp,S{j})
            break;
        else
            S{sol} = Stemp;  sol = sol+1;
            break;
        end
    end
    clear Stemp;
end

%% menghitung fitness
```

```
for i = 1 :length(S)
    jar{i}=jrkrute(S{i},d);    ttj(i) = sum(jar{i});
end
save solusi.mat S ttj jar;
```

solkrom.m

```
clear; load solusi.mat; pop = length(S);
%% Pengkodean solusi ke kromosom
for k = 1:pop
    l=0;
    for i = 1:length(S{k})
        for j = 2:length(S{k}{i})-1
            l = l+1;       kr(l) = S{k}{i}(j);
        end
    end
    krom{k} = kr;
end
save krom.mat krom;
```

seleksi.m

```
clear; load solusi.mat; load krom.mat
pop = length(S);

%% Sort kromosom sesuai fitness
[a b] = sort(ttj); krom = krom(b); ttj = ttj(b);

%% memiilih Orang tua
for i = 1:pop
    pil = pop - floor((-1 + sqrt(1+(4*rand*(pop^2+pop))))/2);
    ot{i}= krom{pil};
end
save ot.mat
```

silang.m

```
clear; load ot.mat; load data.mat; load solusi.mat
pop = length(ot); u = zeros(1,pop); pc = 0.8; an = ot;
while sum(u) < numel(u)
    u0 = find(u==0);    i = u0(1);
    if length(u0)==1
       u0(2)=i;
    end
    for j = u0(2:length(u0))
        if ~isequal(ot{i},ot{j})
            u(i) = 1; u(j) = 1;
            if  rand <= pc
                %% Heuristic Crossover
                O{i} = ot{i};            O{j} = ot{j};
                 tc = acak(1:length(O{i}));
                 tc1 = mod(tc,length(O{i}))+1;
                kt(1) = acak([O{i}(tc1) O{j}(tc1)]);
                if kt(1) == O{i}(tc1)
                    l = find(O{j}==  kt(1));
                    temp = O{j}(tc1);
                    O{j}(tc1) = O{i}(tc1);
```

```
            O{j}(l) = temp;
        elseif  kt(1) == O{j}(tc1)
            l = find(O{i}==  kt(1));
            temp = O{i}(tc1);
            O{i}(tc1) = O{j}(tc1);
            O{i}(l) = temp;
        end
        m = 1;
        for s = mod(tc+1:tc+length(O{i})-1,length(O{i}))+1
            if (d(kt(m),O{i}(s)) <= d(kt(m),O{j}(s)))
                kt(m+1) = O{i}(s);
                l = find(O{j}==kt(m+1));  temp = O{j}(s);
                O{j}(s) = O{i}(s);
               O{j}(l) = temp;
            else
                kt(m+1) = O{j}(s);
                l = find(O{i}==kt(m+1));  temp = O{i}(s);
                O{i}(s) = O{j}(s);   O{i}(l) = temp;
            end
            m = m+1;
        end
        an{i} = kt;

        %% Merge Crossover
        O{i} = ot{i};    O{j} = ot{j};
        tc = acak(1:length(O{i})); kt = [];
         tc1 = mod(tc,length(O{i}))+1;
        kt(1) = acak([O{i}(tc1) O{j}(tc1)]);
        if kt(1) == O{i}(tc1)
            l = find(O{j}==kt(1));      temp = O{j}(tc1);
            O{j}(tc1) = O{i}(tc1);   O{j}(l) = temp;
        elseif  kt(1) == O{j}(tc1)
            l = find(O{i}==  kt(1));    temp = O{i}(tc1);
            O{i}(tc1) = O{j}(tc1);   O{i}(l) = temp;
        end
        m = 1;
        for s = mod(tc+1:tc+length(O{i})-1,length(O{i}))+1
            if f(kt(m),5) <= f(O{j}(s),5)
                kt(m+1) = O{i}(s);
                l = find(O{j} == kt(m+1));
               temp = O{j}(s);
                O{j}(s) = O{i}(s);  O{j}(l) = temp;
            else
                kt(m+1) = O{j}(s);
                l = find(O{i} == kt(m+1));
               temp = O{i}(s);
                O{i}(s) = O{j}(s);  O{i}(l) = temp;
            end
            m = m+1;
        end
        an{j} = kt;
    else
        an{i} = ot{i};  an{j} = ot{j};
    end
    break
else
    if j == u0(length(u0))
        an{i} = ot{i}; an{j} = ot{j};
        u(i)=1; u(j)=1;
```

```
                break
            end
        end
    end
end
save cross.mat an;
```

kromrut.m

```
clear; load cross.mat; load data.mat
po = length(an);
for i = 1:po
    u = zeros(1,length(an{i}));    rute = 1;
    while sum(u) < numel(u)
        r = [1 1];  pk = length(an{i});  at = 0; ms = 0;
        for j = find(u==0)
            pr = length(r)-1;  rtemp = ins(r,an{i}(j),pr);
            if sum(f(rtemp,4))<= Q
                attemp = at;    mstemp = ms;
                at = ms + f(rtemp(pr),7)
                   + d(rtemp(pr),rtemp(pr+1));
                if at <= f(rtemp(pr+1),5);
                    ms = f(rtemp(pr+1),5);
                elseif at > f(rtemp(pr+1),5)
                    ms = at;
                end
                if ms <= f(rtemp(pr+1),6)
                    at_d = ms + f((pr+1),7) + d(rtemp(pr+1),1);
                    if at_d <= f(1,6)
                        r = rtemp;    u(j) = 1;
                    else
                        break
                    end
                else
                    break
                end
            else
                break
            end
        end
        anr{i}{rute} = r;  rute = rute+1;
    end
    anr{i};
end
save anr.mat anr;
```

mutasi.m

```
clear; load anr.mat; load data.mat; load solusi.mat
pa = length(anr);  [jumPel kolom] = size(f);
for i = 1:2:pa
    if i == pa
        anm{i}=anr{i};
    else
        j = i+1; r = rand;
        if r <= 0.2
            for k = i:j
                anm{k} = anr{k};   rm(k) = acak(1:length(anm{k}));
                ukrm = length(anm{k}{rm(k)})-1;
```

```
                tm(k) = acak(2:ukrm);
            end
            anm{i}{rm(i)} = [anr{i}{rm(i)}(1:tm(i))
anr{j}{rm(j)}(tm(j)+1:length(anr{j}{rm(j)}))];
            anm{j}{rm(j)} = [anr{j}{rm(j)}(1:tm(j))
anr{i}{rm(i)}(tm(i)+1:length(anr{i}{rm(i)}))];

            for t = i:j

%% cek planggan dobel dlm rute mutasi
                l = 2;
                while l <= length(anm{t}{rm(t)})-2
                    for m = l+1:length(anm{t}{rm(t)})-1
                        if anm{t}{rm(t)}(l) == anm{t}{rm(t)}(m)
                            anm{t}{rm(t)} = del(anm{t}{rm(t)},m);
                        end
                    end
                    l = l+1;
                end

%% cek kendala rute mutasi
                r = anm{t}{rm(t)};  cek(t) = 1;
                if sum(f(r,4))<= Q
                    at = 0; ms = 0;
                    for s = 1:length(r)-1
                        at(s+1) = ms(s) + f(r(s),7)
                                  + d(r(s),r(s+1));
                        if at(s+1) <= f(r(s+1),5);
                            ms(s+1) = f(r(s+1),5);
                        else
                            ms(s+1) = at(s+1);
                        end
                        if ms(s+1) > f(r(s+1),6)
                            anm{t} = anr{t};   cek(t) = 0;
                            break
                        end
                    end
                else
                    anm{t} = anr{t};   cek(t) = 0;
                end
            end
            for t = i:j
                if cek(t) == 1

% hapus planggan dobel dalam solusi
                 for p = anm{t}{rm(t)}(2:length(anm{t}{rm(t)})-1)
                        if isempty(find(anr{t}{rm(t)} == p, 1))
                            for rute = [1:rm(t)-1
                                rm(t)+1:length(anm{t})]
                                pr = find(anm{t}{rute} == p);
                                if ~isempty(pr)
                                    anm{t}{rute} =
                                del(anm{t}{rute}, pr);
                                end
                            end
                        end
                    end

% menghapus rute kosong
```

```
                    for ho = length(anm{t}):-1:1
                        if numel(anm{t}{ho}) == 2
                            anm{t} = del(anm{t},ho);
                        end
                    end

%% insert planggan yang belum masuk rute
                    ada = zeros(1,jumPel);
                    for rute = 1:length(anm{t})
                        ada(anm{t}{rute}) = 1;
                    end
                    for p = find(ada == 0)
                        bia = inf;        masuk = 0;
                        for rute = 1:length(anm{t})
                            for org = 1:length(anm{t}{rute})-1
                                rut1 = ins(anm{t}{rute},p,org);
                                cebi = 1;
                                if sum(f(rut1),4) <= Q
                                    at = 0; ms = 0;
                                    for ce=2:length(anm{t}{rute})
                        at(ce) = ms(ce-1)+
                          f(rut1(ce-1),7)+d(rut1(ce-1),rut1(ce));
                                        if at(ce) <= f(rut1(ce),5)
                                            ms(ce)=f(rut1(ce),5);
                                        else
                                            ms(ce) = at(ce);
                                        end
                                        if ms(ce) > f(rut1(ce),6)
                                            cebi = 0;
                                            break;
                                        end
                                    end
                                else
                                    cebi = 0;
                                end
                                if cebi == 1
                                    D = 0;
                                    for ce = 1:length(rut1)-1
                                        D = D +
                                        d(rut1(ce),rut1(ce+1));
                                    end
                                    if D <= bia
                                        bia = D;
                                        rutfix = rut1;
                                        posfix = rute;
                                        masuk = 1;
                                    end
                                end
                            end
                        end
                        if masuk == 0
                            anm{t} = anr{t};
                        else
                            anm{t}{posfix} = rutfix;
                        end
                    end
                else
                    anm{k} = anr{k};
                end
```

```
            end
        else
            anm{i} = anr{i};    anm{j} = anr{i};
        end
    end
end
for i=1:length(anm)
    D = 0;
    for j = 1:length(anm{i})
        for k = 1:length(anm{i}{j})-1
            D = D + d(anm{i}{j}(k),anm{i}{j}(k+1));
        end
    end
    ti(i) = D;
end
for i = 1: length(anm)
    jari{i}=jrkrute(anm{i},d);    ti(i)=sum(jari{i});
end
save anm.mat anm ti jari;
```

uji.m

```
O = 0;  ms = 0; Ms = 0; Cs = 0; cs = 0; ttj = 0;
for sm = [m s]
    W = 0; D = 0; T = 0; jar = 0;
    for h=1:length(R{sm})-1
        D = D + d(R{sm}(h),R{sm}(h+1));
        at(h+1) = ms(h) + f(R{sm}(h),7) + d(R{sm}(h),R{sm}(h+1));
        if at(h+1) < f(R{sm}(h+1),5);
            ms(h+1) = f(R{sm}(h+1),5);
        elseif at(h+1) >= f(R{sm}(h+1),5);
            ms(h+1) = at(h+1);
        end
        W = d(R{sm}(h),R{sm}(h+1)) + W + f(R{sm}(h),7);
        if ms(h+1) > f(R{sm}(h+1),6)
            T = T + (ms(h+1)-f(R{sm}(h+1),6));
        end
    end
    if sum(f(R{sm},4)) > Q
        O = O + sum(f(R{sm},4)) - Q;
    end
    if sum(R{sm}) == 2
        R = del(R,sm);
    end
    Cs(sm) = D + 0.01*D*W + 0.1*D*O + 0.01*D*T;
end
cs = sum(Cs); css =cs + sum(cR)-cR(s)-cR(m);
if css < sum(cR)
    sol = sol + 1;    bia(sol) = css;    solu{sol} = R;
end
```

jrkrute.m

```
function jarak = jrkrute(S, d)
for i = 1:length(S)
    jar = 0;
    for j = 1:length(S{i})-1
```

```
        jar = jar + d(S{i}(j),S{i}(j+1));
    end
    jarak(i) = jar;
end
jarak = sort(jarak);
```

cektw.m

```
function penuh = cektw(S,f,d,Q)
penuh = 1;
for i = 1:length(S)
    if sum(f(S{i},4)) <= Q
        ms = 0; at = 0;
        for j = 2:length(S{i})-1
            at(j) = ms(j-1) + f(S{i}(j-1),7)
                    + d(S{i}(j-1),S{i}(j));
            if at(j) <= f(S{i}(j),5)
                ms(j) = f(S{i}(j),5);
            elseif at(j) > f(S{i}(j),5)
                ms(j) = at(j);
            end
            if ms(j) > f(S{i}(j),6)
                penuh = 0;
                break;
            end
        end
    else
        penuh = 0;
        break;
    end
end
```

ins.m

```
function A = ins(A,x,i)
if isempty(A)
    fprintf('\nMatrix kosong atau index melebihi batas\n');
else
    A = [A(1:i) x A(i+1:length(A))];
end
end
```

del.m

```
function A = del(A,i)
if isempty(A) || i>length(A)
    fprintf('\nMatrix kosong atau index melebihi batas\n');
elseif i == 1
    A = A(i+1:length(A));
elseif i == length(A)
    A = A(1:length(A)-1);
else
    A= [A(1:i-1) A(i+1:length(A))];
end
```