TRAVELLING SALESMAN PROBLEM DENGAN KENDALA *TIME WINDOW* PADA PERUSAHAAN JASA PENGANTAR BARANG MENGGUNAKAN ALGORITMA GENETIKA

SKRIPSI

Sebagai salah satu syarat untuk memperoleh gelar
Sarjana dalam bidang Ilmu Komputer

Oleh:
RIO WAHYU SAPUTRO
0310963031-96

PROGRAM STUDI ILMU KOMPUTER
JURUSAN MATEMATIKA
FAKULTAS MIPA
UNIVERSITAS BRAWIJAYA
MALANG

2008


UNIVERSITAS BRAWIJAYA

LEMBAR PENGESAHAN SKRIPSI

***TRAVELLING SALESMAN PROBLEM* DENGAN KENDALA *TIME WINDOW* PADA PERUSAHAAN JASA PENGANTAR BARANG MENGGUNAKAN ALGORITMA GENETIKA**

Oleh:
RIO WAHYU SAPUTRO
0310963031-96

Setelah dipertahankan di depan Majelis Penguji pada 28 Juli 2008
dan dinyatakan memenuhi syarat untuk memperoleh gelar sarjana
dalam bidang Ilmu Komputer

Pembimbing I,	Pembimbing II,
Wayan Firdaus M., SSi, MT	**Edy Santoso, SSi, M.Kom**
NIP. 132 158 724	**NIP. 132 304 307**

Mengetahui,
Ketua Jurusan Matematika
Fakultas MIPA Universitas Brawijaya

Dr. Agus Suryanto, MSc
NIP. 132 126 049

UNIVERSITAS BRAWIJAYA

LEMBAR PERNYATAAN

Saya yang bertanda tangan di bawah ini :

Nama	: Rio Wahyu Saputro
NIM	: 0310963031-96
Jurusan	: Matematika
Penulis skripsi berjudul	: *Travelling Salesman Problem* dengan Kendala *Time Window* Pada perusahaan Jasa Pengantar Barang Menggunakan Algoritma Genetika

Dengan ini menyatakan bahwa :

1. Isi dari skripsi yang saya buat adalah benar-benar karya sendiri dan tidak menjiplak karya orang lain, selain nama-nama yang termaktub di isi dan tertulis di daftar pustaka dalam skripsi ini.
2. Apabila dikemudian hari ternyata skripsi yang saya tulis terbukti hasil jiplakan, maka saya akan bersedia menanggung segala resiko yang akan saya terima.

Demikian pernyataan ini dibuat dengan segala kesadaran.

Malang, 28 Juli 2008
Yang menyatakan,

(Rio Wahyu Saputro)
NIM. 0310963031

TRAVELLING SALESMAN PROBLEM DENGAN KENDALA *TIME WINDOW* PADA PERUSAHAAN JASA PENGANTAR BARANG MENGGUNAKAN ALGORITMA GENETIKA

ABSTRAK

Penelitian ini didasari atas sulitnya perusahaan jasa pengantar barang menentukan rute perjalanan dimana setiap pelanggan memiliki waktu kunjungan masing-masing. Pelanggan hanya bisa dikunjungi pada waktu yang sudah ditetukan sebelumya oleh pelanggan dan waktu tersebut dinamakan "*time window*". Metode yang digunakan untuk menyelesaikan masalah pencarian rute terpendek dengan kendala *time window* ini adalah algoritma genetika. Pada metode algoritma genetika proses untuk inisialisasi kromosom menggunakan pengkodean permutasi, *crossover* dengan *one-cut point*, mutasi dengan *shift mutation*, serta proses seleksi yang digunakan adalah metode *roullete wheel* dan *best rank.* Pada Penelitian TSPTW kali ini kedatangan sebelum *time window* dikenakan waktu tunggu dan kedatangan sesudah *time window* dikenakan penalti. Setelah dilakukan uji coba didapatkan hasil dengan menggunkan metode seleksi *crossover roullete wheel* dan mutasi *roullete wheel* diperoleh hasil yang lebih baik dibandingkan dengan menggunakan metode yang lain. Nilai *fitness* terbaik diperoleh pada *crossover rate* 0.2 dan *mutation rate* 0.6 dan nilai *fitness* mencapai konvergen antara iterasi ke-759 sampai ke-926 dari 1000 kali iterasi yang dilakukan.

TRAVELLING SALESMAN PROBLEM WITH DEADLINE TIME WINDOW IN DELIVERY SERVICE COMPANY USING GENETIC ALGORITHM

ABSTRACT

This research came from the fast of delivery service company's problem for delivery the route to be through where every costumer has own visiting time. Customer is only able to be visited at a certain time with is set by customer in previous, that such time is known as time window. The method used to solve this TSPTW problem is "Genethic algorithm". In genetic algorithm, the process of chromosom initialization is using permutation encoding, one-cut point crossover and shift mutation. The selection process used here is roullete wheel and best rank. In this TSPTW research, the arrival before time window is followed with time wait while the arrival after time window is fallowed with penalty. After testing process, it's obtained that the result using selection method roullete wheel for crossover and selection method roullete wheel for mutation is better than the other method selection.The best fitness is obtained at crossover rate of 0.2 and mutation rate 0.6, and fitness value is convergent between the iteration at 759 and 926 of 1000 iterations.

KATA PENGANTAR

Puji syukur Penulis panjatkan kepada Allah SWT atas segala limpahan rahmat dan hidayah-Nya, karena atas segala rahmat dan limpahan hidayahnya, skripsi yang berjudul "*TRAVELLING SALESMAN PROBLEM* DENGAN KENDALA *TIME WINDOW* PADA PERUSAHAAN JASA PENGANTAR BARANG MENGGUNAKAN ALGORITMA GENETIKA" ini dapat diselesaikan. skripsi ini disusun dan diajukan sebagai syarat untuk memperoleh gelar sarjana pada program studi Ilmu Komputer, jurusan Matematika, fakultas MIPA, universitas Brawijaya.

Dalam penyelesaian skripsi ini, penulis telah mendapat begitu banyak bantuan baik moral maupun materiil dari banyak pihak. Atas bantuan yang telah diberikan, penulis ingin menyampaikan penghargaan dan ucapan terima kasih yang sedalam-dalamnya kepada:

1. Bapak Wayan Firdaus Mahmudy, SSi., MT, selaku pembibing utama penulisan skripsi sekaligus Ketua Program Studi Ilmu Komputer, Jurusan Matematika, Fakultas MIPA Universitas Brawijaya.
2. Bapak Edy Santoso, M.Kom, selaku pembimbing pendamping dalam penulisan skripsi.
3. Bapak Dr. Agus Suryanto, MSc, selaku Ketua Jurusan Matematika, Fakultas MIPA Universitas Brawijaya.
4. Bapak Drs. Achmad Ridok, M.Kom, selaku penasehat akademik.
5. Kedua orang tua tercinta (Bapak Prayogo Pujianto dan Ibu Eny Wihayati) atas do'a, semangat dan materi yang sudah diberikan kepada ananda untuk menyelesaikan kuliah.
6. Adikku Ria Wahyu Luckyta PS, atas do'a dan semangat yang diberikan.
7. Rekan-rekan seperjuangan ilmu komputer 2003, Boim, Tanaya, Elly, Ratnanie, Kresno, Toni dan teman-teman lain yang tidak bisa disebutkan satu-persatu, terimaksih atas do'a dan semngat yang sudah diberikan.
8. Teman-teman yang ada di mabes semanggi barat 23 B, terimakasih atas do'a dan semangat yang sudah diberikan.
9. Segenap bapak dan ibu dosen yang telah mendidik dan mengajarkan ilmunya kepada penulis selama menempuh

pendidikan di Program Studi Ilmu Komputer Jurusan Matematika, Fakultas MIPA Universitas Brawijaya.

10. Dan semua pihak yang telah membantu dalam penyusunan skripsi ini yang tidak dapat Penulis sebutkan satu per satu.

Penulis menyadari bahwa masih banyak kekurangan dalam laporan ini, oleh karena itu Penulis sangat menghargai saran dan kritik yang sifatnya membangun demi perbaikan penulisan dan mutu isi skripsi ini untuk kelanjutan penelitian serupa di masa mendatang. Semoga laporan skripsi ini dapat bermanfaat. Amin.

Malang, 28 Juli 2008

Penulis

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

DAFTAR SAOURCECODE


UNIVERSITAS BRAWIJAYA

DAFTAR LAMPIRAN

BAB I
PENDAHULUAN

1.1 Latar Belakang Masalah

Meningkatnya mobilitas manusia dalam melakukan aktivitas di luar rumah menyebabkan kesulitan dialami oleh banyak biro perjalanan dalam hal penjemputan penumpang, disamping itu biro perjalanan juga menemui berbagai kesulitan karena persoalan kemacetan yang terjadi di banyak ruas jalan. Beberapa penjemputan penumpang seringkali mengalami keterlambatan dalam melakukan penjemputan. Hal ini dikarenakan prediksi waktu tempuh yang kurang tepat. Melesetnya jadwal penjemputan penumpang ini dapat membuat penumpang menjadi kurang nyaman.

Selain dari masalah kemacetan lalulintas biro perjalanan juga mengalami kesulitan mengenai pejemputan penumpang dimana sebagian penumpang bisa dijemput pada tenggat waktu yang sudah ditentukan oleh penumpang. Biro perjalanan harus memperhatikan hal ini dalam melakukan penjemputan agar penumpang mer

Semua informasi itu tidak hanya berguna bagi mereka yang akan melakukan perjalanan saja tetapi juga sangat diperlukan bagi perusahaaan-perusahaan yang bergerak dalam bidang *travelling, tours*, ekspedisi dan perusahaan lain yang sejenis. Di dunia usaha banyak sekali persaingan, setiap perusahaan selalu berlomba-lomba dalam menarik pelanggan untuk selalu memakai jasanya atau produk mereka. Hal ini akan membingungkan pelanggan dalam menentukan pilihan atau mempercayai suatu perusahaan tertentu. Agar sebuah perusahaan tetap menjadi pilihan pelanggan maka harus memiliki trik-trik dalam menarik minat pelanggan, seperti adanya promosi yang menarik (memasang iklan, menyebarkan brosur-brosur yang bertuliskan kata-kata yang tersusun sederhana tetapi dapat menarik minat dan mengenai sasaran) serta meminimalkan harga dengan tetap menjaga kualitas dengan baik dan tidak kalah pentingnya pelayanan juga harus ditingkatkan.

Tugas akhir ini diharapkan dapat membantu biro perjalanan yang bergerak dalam bidang *travelling* bisa lebih baik dalam memberikan pelayanan kepada pelanggan khususnya dalam hal penjemputan.

Untuk menjemput pelanggan, biro perjalanan menyediakan suatu kendaraan khusus (mobil dengan tipe tertentu) yang bisa menampung pelanggan. Setiap pelanggan akan dijemput oleh satu kendaraan yang telah disediakan. Biro perjalanan sering mengalami kesulitan dalam memberikan informasi pada pelanggan mengenai waktu tiba ditempat pelanggan (tempat penjemputan) dengan kisaran waktu yang sekecil mungkin agar pelanggan tidak harus rela menunggu mobil datang dengan membuang waktu sia-sia hanya untuk menunggu jemputan serta total lama perjalanan yang diperlukan untuk sampai ke tempat tujuan (tempat penjemputan). Kasus mengenai penjemputan pelanggan pada biro perjalanan ini termasuk permasalahan dalam bidang transportasi.

Telah banyak penelitian mengenai persoalan transportasi / pencarian rute dengan berbagai macam metode / algoritma. Terdapat banyak metode yang digunakan dalam pencarian rute, seperti *genetic algorithms* (Alvarenga dkk, 2003), metode *Djikstra's* (Kartika Gunadi, Yulia, Jeffry Tanuhardja, 2002), algoritma A*, algoritma G*reedy* (Irvan, dkk., 2005), (Taufiq, dkk., 2006), Sirkuit *Hamilton* (Denny, 2007) dan algoritma yang lainnya.

Penulis menggunakan algoritma *transitive closure* untuk memecahkan masalah di atas. Algoritma ini dipilih karena algoritma ini mempunyai kemampuan untuk memberikan alternatif jalur / rute yang lain dan dapat memprediksi kemacetan yang ada pada waktu-waktu tertentu sehingga dengan menggunakan algoritma ini diharapkan dapat meningkatkan pelayanan kepada pelanggan biro perjalanan.

Berdasarkan latar belakang yang telah dipaparkan, tugas akhir ini ditujukan untuk mencari rute tercepat pada biro perjalanan dengan menggunakan algoritma *transitive closure.* Maka penulis mengambil judul pada tugas akhir ini dengan **"Pencarian Rute Tercepat pada Biro Perjalanan menggunakan Algoritma *Transitive Closure*".**

1.2 Rumusan Masalah

Berdasarkan uraian pada latar belakang masalah, maka dalam tugas akhir ini dapat dirumuskan permasalahan sebagai berikut:

1. Bagaimanakah menentukan rute tercepat pada biro perjalanan dengan menggunakan algoritma *transitive closure* ?

2. Bagaimanakah menentukan waktu kedatangan mobil di masing-masing tempat pelanggan(tempat penjemputan) serta waktu yang dibutuhkan untuk sampai ke tempat tujuan?

1.3 Batasan Masalah

Dari permasalahan di atas, berikut ini diberikan batasan masalah untuk menghindari melebarnya masalah yang akan diselesaikan:

1. Masukannya(*input*an) adalah tempat / titik awal (*base*), tempat / titik tujuan (*destination*), waktu keberangkatan (*starting time*). Sedangkan keluaranya akan memberikan informasi alur perjalanan dan prediksi waktu di masing-masing tempat pelanggan (tempat penjemputan) serta total waktu yang dibutuhkan untuk sampai ke tujuan.
2. Penggambaran pada lokasi meliputi penggambaran titik yang akan menunjukkan sebagai tempat awal atau tempat tujuan dan penggambaran garis menunjukkan hubungan antara satu tempat dengan tempat yang lain.
3. Suatu titik dikatakan terkoneksi dengan titik lain apabila *user* memasukkan lama perjalanannya pada waktu-waktu tertentu.
4. Pemasukkan jalur / arah tidak disesuaikan dengan keadaan yang sebenarnya pada jalan.
5. Semua jalan dianggap satu arah saja.

1.4 Tujuan

Tujuan dari tugas akhir ini adalah:

1. Membantu biro perjalanan dalam hal mengambil keputusan mengenai rute mana yang sebaiknya dipilih agar dapat sampai ke tujuan dengan cepat.
2. Memprediksi waktu kedatangan di tempat tujuan pelanggan dengan kisaran waktu yang seminimal mungkin.

1.5 Manfaat

Manfaat dari tugas akhir ini adalah dapat dipakai membantu dalam hal mengambil keputusan mengenai rute mana yang sebaiknya dipilih agar dapat sampai ke tujuan dengan cepat, memprediksi waktu kedatangan di tempat tujuan pelanggan (tempat penjemputan)

dengan kisaran waktu yang seminimal mungkin agar pelayanan pada biro perjalanan tersebut memuaskan pelanggan dan dapat menarik pelanggan sebanyak mungkin serta algoritma *transitive closure* ini nantinya akan dapat diterapkan pada biro perjalanan.

1.6 Sistematika Penulisan

Tugas akhir ini disusun berdasarkan sistematika penulisan sebagai berikut:

1. BAB I PENDAHULUAN
   Berisi latar belakang masalah, perumusan masalah, batasan masalah, tujuan, manfaat dan sistematika penulisan.
2. BAB II TINJAUAN PUSTAKA
   Menguraikan teori-teori yang berhubungan dengan pencarian rute tercepat, graf, matrik *Incidence*, matrik *Adjacency*, ERD, DBMS, SQL, *Flowchart*, algoritma *transitive closure* dan *Borland Delphi 7*.
3. BAB III METODELOGI DAN PERANCANGAN
   Pada bab ini akan dijelaskan mengenai metodelogi yang digunakan dalam menyelesaikan masalah pencarian rute tercepat pada biro perjalanan serta desain sistemnya.
4. BAB IV IMPLEMENTASI DAN UJI COBA
   Pada bab ini akan dilakukan implementasi sistem dan uji coba perangkat lunak yang dibangun, yaitu apakah hasil dari pencarian rute tercepat dengan menggunakan algoritma *transitive closure* ini sudah baik dan efektif .
5. BAB V KESIMPULAN DAN SARAN
   Berisi kesimpulan dari seluruh rangkaian tugas akhir serta saran kemungkinan pengembangannya.

BAB I
PENDAHULUAN

1.2 Latar Belakang Masalah

Meningkatnya mobilitas manusia dalam melakukan aktivitas di luar rumah menyebabkan kesulitan dialami oleh banyak perusahaan khususnya bagi persusahaan yang bergerak dalam jasa pengantar barang. Masing-masing pelanggan memiliki jadwal yang berbeda-beda yang menunjukkan jam berapakah pelanggan ada di rumah dan dapat menerima pengiriman barang pada waktu yang sudah ditentukan sebelumnya oleh pelanggan. Pihak perusahaan tentunya harus memperhatikan hal ini dalam melakukan pengiriman barang.

Perusahaan mengirim barang pada saat pelanggan tersebut sedang di rumah dan tidak sibuk. Jadi apabila ada dua atau lebih pelanggan yang ingin barangnya diantar pada waktu tertentu pihak perusahaan harus bisa memperkirakan lama waktu yang dibutuhkan untuk pengiriman barang, sehingga nantinya pelanggan dapat menerima barang pada waktu yang sudah ditentukan. Barang yang dimaksud adalah merupakan paket khusus dimana paket hanya bisa diterima oleh pelanggan yang berhak. Dengan diterimanya paket oleh pelanggan diharapakan akan merasa puas dengan layanan tersebut, karena mereka dapat menerima barang pada waktu yang sudah ditentukan. Bagi pihak perusahaan juga akan memberikan keuntungan karena dengan berkurangnya kendaraan yang digunakan untuk mengirim barang, maka otomatis akan mengurangi biaya bagi perusahaan.

Permasalahan penentuan rute pengantaran barang tersebut memungkinkan suatu solusi yang paling efektif untuk diterapkan dalam persoalan yang akan dihadapi oleh perusahaan, yaitu kemacetan yang sering terjadi selama perjalanan. Kemacetan tersebut mengganggu pengantaran barang. Dibutuhkan suatu cara untuk menanggulangi gangguan tersebut, yaitu dengan cara mencari lintasan terpendek dari perusahaan ke tempat pelanggan dengan mempertimbangkan waktu kemacetan sering terjadi, waktu masing-masing pelanggan atau *time window* serta bagaimana urutan pengantaran barang kepada pelanggan.

Time window adalah tenggang waktu yang sudah ditentukan dimana memiliki waktu awal dan akhir. Dengan adanya *time window*

ini akan memberi permasalahan baru bagi perusahaan dalam hal pencarian rute terpendek untuk mengantarkan barang kepada pelanggan dari kota kekota. Sehingga muncul suatu pemecahan masalah yang disebut *Travelling Salesman Problem* dengan kendala *time Window* (TSPTW).

Travelling Salesaman problem dengan kendala *time window* (TSPTW) merupakan bentuk permasalahan baru dalam pencarian rute terpendek, untuk menentukan rute terpendek selain adanya biaya perjalanan juga terdapat batas waktu kunjungan yang sudah ditentukan atau *time window*. Masing-masing kota memiliki waktu kunjungan yang sudah ditetapkan, apabila kedatangan sebelum *time window* terbuka maka mengakibatkan waktu tunggu, sebaliknya apabila kedatangan setelah *time window* tertutup tidak diperbolehkan.

Telah terdapat banyak penelitian mengenai pencarian rute terpendek dengan kendala *time window* salah satunya seperti *On approximating a geometric prize-collecting traveling salesman problem with time windows* (Reuven Bar-Yehuda, dkk).

Penulis menggunakan algoritma genetika untuk memecahkan masalah di atas. Algoritma ini dipilih karena Algoritma genetika dapat digunakan untuk menyelesaikan masalah optimasi yang kompleks seperti mencari rute paling optimum dengan memperhatikan kondisi jalan misalnya kepadatan lalulintas, jalan dua arah dan lain-lain. Sehingga dengan menggunakan algoritma ini diharapkan dapat meningkatkan pelayanan kepada pelanggan.

Dari adanya permasalahan di atas, maka dalam tugas akhir ini akan dibangun suatu alat bantu sistem informasi pencarian rute terpendek dengan kendala *time window* dan urutan pengantaran barang kepada pelanggan. Berdasarkan latar belakang yang telah dipaparkan, maka penulis mengambil judul pada tugas akhir ini dengan **"*Travelling Salesman Problem* dengan Kendala *Time Window* Pada Perusahaan Jasa Pengantar Barang Menggunakan Algoritma genetika".**

1.2 Rumusan Masalah

Berdasarkan uraian pada latar belakang masalah, maka dalam tugas akhir ini dapat dirumuskan permasalahan sebagai berikut:

1. Bagaimanakah menentukan rute tercepat untuk mengantarkan barang kepada setiap pelanggan dengan kendala *time window* pada perusahaan jasa pengantar barang menggunakan algoritma genetika?
2. Bagaimana menerapkan algoritma genetika untuk pencarian rute optimal mengantarkan barang kepada setiap pelanggan dengan kendala *time window*?
3. Bagaimana nilai *fitness* yang dihasilkan dari seleksi untuk proses crossover dan seleksi untuk proses mutasi yang berbeda?

1.3 Batasan Masalah

Dari permasalahan di atas, berikut ini diberikan batasan masalah untuk menghindari melebarnya masalah yang akan diselesaikan:

1. Memiliki titik keberangkatan dan akhir yang sama
2. Barang yang dikirim merupakan barang paket khusus, dimana hanya bisa diterima oleh pemiliknya atau sudah diwakilkan sebelumnya kepada seseorang untuk menerimanya.
3. Jasa pengantaran barang hanya dilakukan pada satu waktu, pada penelitian ini menggunakan *one day service.*
4. Setiap kota dikunjungi pada tiap *time window* masing-masing.
5. Kedatangan sebelum *time window* dikenakan waktu tunggu.
6. Kedatangan setelah *time window* dikenakan penalti.
7. Masukkan biaya perjalanan dan *time window* tidak disesuaikan dengan keadaan yang sebenarnya.

1.4 Tujuan Penelitian

Tujuan dari tugas akhir ini adalah:

1. Membantu pihak perusahaan jasa pengantar barang dalam hal mengambil keputusan untuk mengantarkan barang kepada setiap pelanggan.
2. Untuk menerapkan algoritma genetika dalam masalah pencarian rute optimal dengan kendala *time window* pada perusahaan jasa pengantar barang.

3. Membandingkan nilai *fitness* dari proses seleksi untuk proses *crossover* dan seleksi untuk proses mutasi dengan metode seleksi yang berbeda.

1.5 Manfaat Penelitian

Manfaat dari penelitian ini adalah dapat dipakai perusahaan jasa pengantar barang dalam hal mengambil keputusan mengenai rute mana yang sebaiknya dipilih agar dapat mengantar barang kepada pelanggan pada waktu yang sudah ditentukan sebelumnya oleh pelanggan.

1.6 Metodologi Pemecahan Masalah

Untuk mencapai tujuan yang dirumuskan sebelumnya, maka metodologi yang digunakan dalam penulisan tugas akhir ini adalah:

1. Studi Literatur
   Mempelajari teori-teori yang berhubungan dengan konsep pencarian rute tependek dari berbagai referensi.
2. Pendefinisian dan analisis masalah
   Mendefinisikan dan menganalisis masalah untuk mencari solusi yang tepat.
3. Perancangan dan implementasi sistem
   Membuat perancangan perangkat lunak dengan analisis terstruktur dan mengimplementasikan hasil rancangan tersebut yaitu membuat alat bantu sistem pencarian rute terpendek dengan kendala *time window.*
4. Uji coba dan analisa hasil implementasi
   Menguji perangkat lunak, dan menganalisa hasil dari implementasi tersebut apakah sudah sesuai dengan tujuan yang dirumuskan sebelumnya, untuk kemudian dievaluasi dan disempurnakan.

1.7 Sistematika Penulisan

Tugas akhir ini disusun berdasarkan sistematika penulisan sebagai berikut:

1. BAB I PENDAHULUAN
   Berisi latar belakang masalah, perumusan masalah, batasan masalah, tujuan penelitian, manfaat penelitian, metodologi pemecahan masalah dan sistematika penulisan.
2. BAB II TINJAUAN PUSTAKA
   Menguraikan teori-teori yang berhubungan dengan *travelling salesman problem* dengan *time window* (TSPTW), algoritma genetika dan *Borland Delphi 7*.
3. BAB III METODOLOGI DAN PERANCANGAN
   Pada bab ini akan dijelaskan mengenai motodologi yang digunakan dalam menyelesaikan masalah *travelling salesman problem* dengan *time window* (TSPTW) pada perusahaan pengantar barang serta desain sistemnya.
4. BAB IV IMPLEMENTASI DAN UJI COBA
   Pada bab ini akan dilakukan implementasi sistem dan uji coba perangkat lunak yang dibangun, yaitu apakah hasil dari pencarian rute tercepat dengan *time window* ini sudah baik dan efektif .
5. BAB V KESIMPULAN DAN SARAN
   Berisi kesimpulan dari seluruh rangkaian skripsi serta saran kemungkinan pengembangannya.

BAB II
TINJAUAN PUSTAKA

2.1 *Travelling salesman Problem* dengan Kendala *Time window*

Travelling salesman problem dengan *time window* (TSPTW) adalah bentuk permasalahan baru dalam pencarian rute terpendek dimana terdapat kendala *time window*.

Time window adalah waktu yang ditetapkan masing-masing kota untuk dapat dikunjungi, apabila *time window* kota tersebut masih tertutup maka mengakibatkan waktu tunggu. Pesant (1998) mendefinisikan *travelling salesman problem* dengan *time windows* (TSPTW) sebagai permasalahan untuk mencari biaya *tour* minimal dari sekumpulan kota, dimana tiap kota hanya dikunjungi satu kali saja. Agar *feasible*, maka *tour* tersebut harus berawal dan berakhir disuatu *depot* tertentu, dalam batas *time window* tertentu, dan tiap kota harus dikunjungi pada batas *time window* mereka masing–masing. Biaya TSPTW biasanya berhubungan dengan total jarak *travel* atau total waktunya (waktu *travel* ditambah waktu tunggu ditambah waktu pelayanan). Jadi model TSPTW merupakan pengembangan TSP, yaitu dengan tambahan kendala *time windows* untuk masing–masing kota. *Time windows* [a_i, b_i] menunjukkan batas waktu pelayanan di kota *i*, dengan batas awal a_i dan batas akhir b_i. Kedatangan sebelum a_i diperbolehkan tetapi mengakibatkan adanya waktu tunggu sampai batas *time windows*, tetapi tidak diperbolehkan adanya kedatangan sesudah b_i (Sutapa dkk, 2004).

2.2 Sistem Pengantaran Barang

2.2.1 Sistem Pengantaran

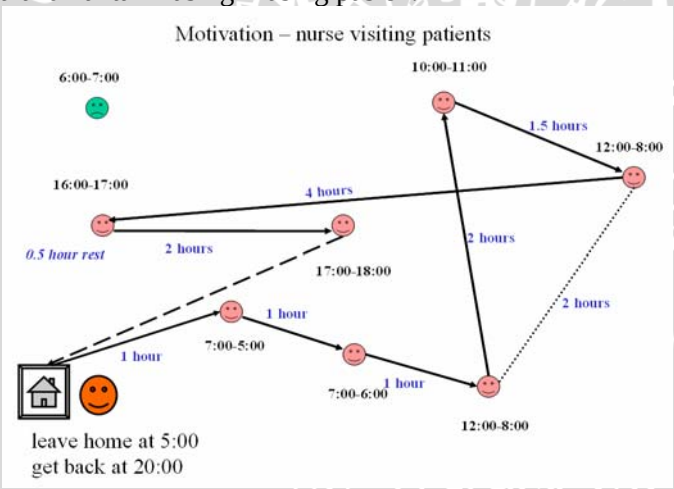
Sistem pengantaran barang meliputi:

a. Kegiatan mengantarkan barang ke masing-masing kota dengan memperhitungkan lama perjalanan, waktu tunggu, waktu pelayanan dan penalti.
b. Adanya waktu tunggu apabila sesampainya dikota tujuan *time window* kota tersebut belum terbuka.
c. Nilai penalti diberikan apabila sesampainya di kota tujuan *time window* sudah tertutup.

2.2.2 Pemilihan Rute

Ada beberapa hal yang berpengaruh terhadap pencarian rute terpendek beberapa diantaranya adalah waktu tempuh, jarak dan biaya. Salah satu pendekatan yang paling sering digunakan adalah pertimbangan dua faktor utama dalam pemilihan rute, yaitu biaya pergerakan dan nilai waktu pergerakan yang dianggap proporsional dengan jarak tempuh. Dalam beberapa model pemilihan rute dimungkinkan penggunaan bobot yang berbeda bagi faktor waktu tempuh dan faktor jarak tempuh untuk menggambarkan persepsi pengendara dalam kedua faktor tersebut.

Pada kasus pencarian rute terpendek dengan kendala *time window* ini ada beberapa hal yang berpengaruh selain waktu tempuh yaitu adanya kendala *time window* pada masing-masing kota, seperti pada Gambar 2.1, perawat mengunjungi pasienya pada waktu yang sudah ditentukan dan terlihat urutan perjalananya serta biaya perjalanan antar masing-masing pasien.



Gambar 2.1 Skema Operasi Sistem Kunjungan Perawat Ke masing-masing Pasien (Reuven Bar-Yehuda dkk)

2.3 Algoritma genetika

Algoritma genetika merupakan evolusi/perkembangan dunia komputer dalam bidang kecerdasan buatan (*artificial intellegenci*).

Sebenarnya algoritma genetika ini terinspirasi oleh teori evolusi Darwin (walaupun pada kenyataannya teori tersebut terbukti keliru).

Algoritma genetika adalah algoritma pencarian yang berdasarkan pada mekanisme sistem natural yakni genetika dan seleksi alam. Dalam aplikasi algoritma genetika, variabel solusi dikodekan kedalam struktur string yang merepresentasikan barisan gen, yang merupakan karakteristik dari solusi problem.

Berbeda dengan teknik pencarian konvensional, algoritma genetika berangkat dari himpunan solusi yang dihasilkan secara acak. Himpunan ini disebut populasi. Sedangkan setiap individu dalam populasi disebut kromosom yang merupakan representasi dari solusi. Kromosom-kromosom berevolusi dalam suatu proses iterasi yang berkelanjutan yang disebut generasi. Pada setiap generasi, kromosom dievaluasi berdasarkan suatu fungsi evaluasi (Syamsudin, 2004). Setelah beberapa generasi maka algoritma genetika akan konvergen pada kromosom terbaik, yang diharapkan merupakan solusi optimal (Goldberg,1989).

Algoritma genetika ini banyak dipakai pada aplikasi bisnis, teknik maupun pada bidang keilmuan. Algoritma ini dapat dipakai untuk mendapatkan solusi yang tepat untuk masalah optimal dari satu variabel atau multi variabel. Sebelum algoritma ini dijalankan, masalah apa yang ingin dioptimalkan itu harus dinyatakan dalam fungsi tujuan, yang dikenal dengan fungsi *fitness*. Jika nilai *fitness* semakin besar, maka sistem yang dihasilkan semakin baik. Walaupun pada awalnya semua nilai *fitness* kemungkinan sangat kecil (karena algoritma ini menghasilkannya secara random), sebagian akan lebih tinggi dari yang lain. Kromosom dengan nilai *fitness* yang tinggi ini akan memberikan probabilitas yang tinggi untuk bereproduksi pada generasi selanjutnya. Sehingga untuk setiap generasi pada proses evolusi, fungsi *fitness* yang mensimulasikan seleksi alam, akan menekan populasi kearah *fitness* yang meningkat.

Algoritma genetika sangat tepat digunakan untuk penyelesaian masalah optimasi yang kompleks dan sukar diselesaikan dengan menggunakan metode yang konvensional. Sebagaimana halnya proses evolusi di alam, suatu algoritma genetika yang sederhana umumnya terdiri dari tiga operator yaitu: operator *reproduksi* (seleksi), operator *crossover* (persilangan) dan operator mutasi.

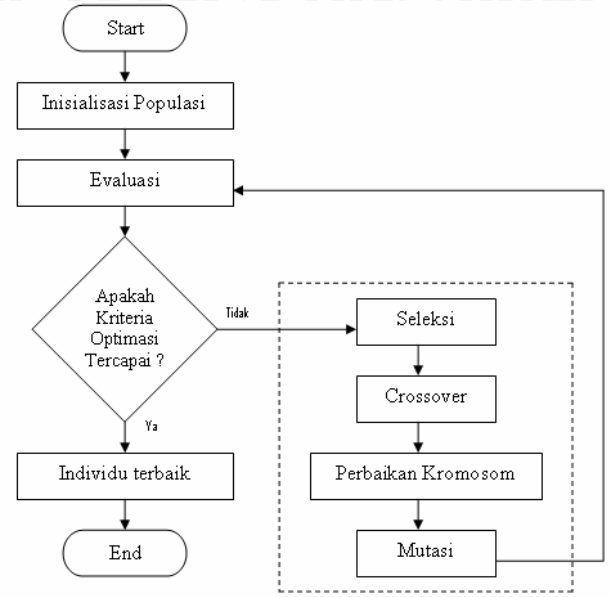
Secara garis besar, algoritma genetika dapat dijelaskan dengan algoritma berikut:

1. [**Mulai**] Membangun populasi secara random sebanyak n kromosom (sesuai dengan masalahnya).
2. **[Fitness**] Evaluasi setiap *fitness* f(x) dari setiap kromosom x pada populasi.
3. [**Populasi baru**] Membuat populasi baru dengan mengulang langkah-langkah berikut sampai populasi baru lengkap.
    1. [**Seleksi**] Pilih dua kromosom induk dari populasi berdasarkan *fitness*nya (semakin besar *fitness*nya semakin besar kemungkinannya untuk terpilih)
    2. [***Crossover***] Sesuai dengan besarnya kemungkinan c*rossover,* induk terpilih disilangkan untuk membentuk anak. Jika tidak ada *crossover*, maka anak merupakan salinan dari induknya.
    3. [**Mutasi**] Sesuai dengan besarnya kemungkinan mutasi, anak dimutasi pada setiap lokus (posisi pada kromosom).
    4. [**Penerimaan**] tempatkan anak baru pada populasi baru.
4. [**Ganti**] Gunakan populasi yang baru dibentuk untuk proses algoritma selanjutnya.
5. [**Tes**] jika kondisi akhir terpenuhi, berhenti, dan hasilnya adalah solusi terbaik dari populasi saat itu.
6. [**Ulangi**] Ke nomer 2. (Obitko,1998)

Apabila *P*(t) dan *C*(t) merupakan *parent* dan *offspring* pada generasi *t*, maka *pseudo code* dari algoritma genetika dapat dituliskan (Subekti, 2002):

```
1   procedure Algoritma Genetika
2     begin
3     t  ← 0;
4     initialize P(t);
5     evaluate P(t);
6     while (not termination condition) do
7       recombine P(t) to yield C(t);
8       evaluate P(t);
9
10    select P(t+1) from P(t) and C(t);
11    t ← t+1;
12    end
13  End
```

Gambar 2.2 *Pseudo Code* Genetic Algorithm



Gambar 2.3 Diagram Alir Algoritma Genetika

2.3.1 Representasi Kromosom

Langkah pertama pada algoritma genetika adalah menerjemahkan atau merepresentasikan masalah riil menjadi terminologi biologi. Cara untuk merepresentasikan masalah ke dalam bentuk kromosom disebut pengkodean. Terdapat beberapa cara pengkodean, dan pemilihannya berdasarkan masalah yang dihadapi, yaitu :

a. Pengkodean Biner
   Digunakan untuk masalah nilai *maximize* pada sebuah fungsi matematika.
b. Pengkodean Permutasi
   Digunakan untuk masalah pengurutan data.
c. Pengkodean Nilai
   Digunakan untuk masalah yang sangat kompleks, dimana nilai yang dikodekan merupakan representasi langsung dari masalah.
d. Pengkodean Pohon
   Digunakan untuk masalah menyusun program atau ekspresi, setiap kromosom fungsi atau perintah pada bahasa pemrograman.

Pengkodean permutasi dapat digunakan pada masalah pengurutan data (*ordering problems*), seperti wiraniaga (*Travelling Salesman Problem*) atau masalah pengurutan tugas (*Task Ordering Problem*). Pada pengkodean permutasi, setiap kromosom terdiri dari barisan angka, yang merepresentasikan angka pada urutan.

Tabel 2.1 Contoh Kromosom Pada Pengkodean Permutasi.

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

2.3.2 Seleksi

2.3.2.1 Seleksi *Roullete Wheell*

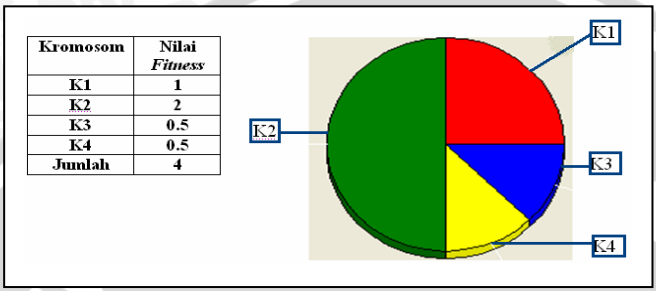
Pemilihan dua buah kromosom sebagai orang tua (*parent*) yang akan dipindah silangkan biasanya dilakukan secara proporsional sesuai dengan nilai *fitness*nya. Suatu metode seleksi yang umum dilakukan adalah *roulette wheel* (roda *roulette*). Sesuai dengan namanya, metode ini menirukan permainan *roulette wheel* dimana masing-masing kromosom menempati potongan lingkaran pada roda *roulette* secara proporsional sesuai dengan nilai *fitness*nya seperti yang ada pada Gambar 2.2. Kromosom yang memiliki nilai *fitness* besar menempati potongan lingkaran yang lebih besar dibandingkan dengan kromosom yang memiliki nilai *fitness* lebih kecil.

Untuk selengkapnya langkah-langkahnya adalah :

1. Hitung nilai *fitness* untuk setiap kromosom
2. Hitung total nilai *fitness* pada populasi
3. Hitung probabilitas seleksi pada setiap kromosom
4. Hitung probabilitas kumulatif untuk setiap kromosom

Turnament Selection adalah alternatif lain untuk pemilihan *parent*. Pada seleksi alam yang terjadi di dunia nyata, beberapa individu (biasanya individu jantan) berkompetisi dalam sebuah kelompok kecil sampai tersisa hanya satu individu pemenang. Individu pemenang inilah yang memungkinkan melakukan perkawinan. Dalam bentuk paling sederhana metode ini mengambil dua kromosom secara acak dan kemudian menyeleksi salah satu yang bernilai *fitness* paling tinggi untuk menjadi *parent* pertama. Cara

yang sama dilakukan untuk mendapatkan *parent* kedua (Suyanto, 2005), ilustrasi seleksi berdasarkan *roulette wheell* pada Gambar 2.3.



Gambar 2.4 Ilustrasi *roulette wheel*

2.3.2.2 Seleksi *Best Rank*

Pada metode seleksi rank, proses dimulai dengan merangking atau mengurutkan kromosom di dalam populasi berdasarkan *fitness*nya. Kemudian memberi nilai *fitness* baru berdasarkan urutannya. Kromosom dengan nilai terbaik akan memiliki *fitness* baru nilai 1, terbaik kedua bernilai 2 dan begitu seterusnya, sehingga kromosom yang memiliki *fitness* terburuk akan memiliki nilai *fitness* N, dimana N adalah jumlah kromosom di dalam populasi. Seperti dapat dilihat pada Tabel 2.2.

Tabel 2.2 Keadaan Seleksi *Best Rank* sebelum dirangking

Kromosom	*Fitness*
A	5
B	15
C	10
D	7
E	5

Tabel 2.3 Keadaan Seleksi *Best Rank* setelah dirangking

Kromosom	*Fitness*	*Fitness* baru
B	15	1
C	10	2
D	7	3
A	5	4
E	5	5

2.3.3 Operator Genetika

Operator genetika diigunakan untuk mengkombinasi (modifikasi) individu dalam aliran populasi guna mencetak individu pada generasi berikutnya. Ada dua operator genetika yaitu *crossover* dan mutasi.

2.3.3.1 *Crossover*

Proses *crossover* berfungsi untuk menghasilkan keturunan dari dua buah kromosom induk yang terpilih. Kromosom anak yang dihasilkan merupakan kombinasi gen-gen yang dimiliki oleh kromosom induk.

Secara umum, mekanisme *crossover* adalah sebagi berikut:

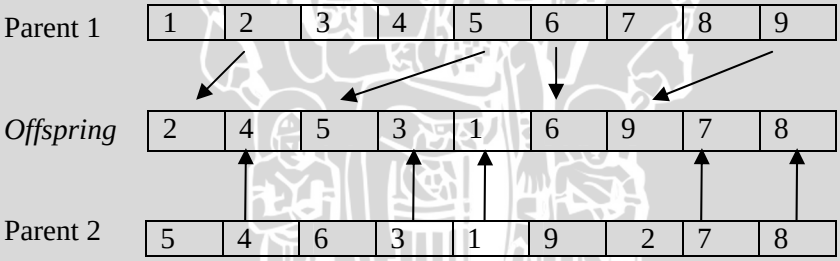
1. memilih dua buah kromosom sebagai induk.
2. memilih secara acak posisi dalam kromosom, biasa disebut titik *crossover*, sehingga masing-masing kromosom induk terpecah menjadi dua segmen.
3. lakukan pertukaran antar segmen kromosom induk untuk menghasilkan kromosom anak.

Apabila v_k merupakan individu dalam populasi k maka p*seudo code* dari proses *crossover* dapat dilihat pada Gambar 2.3. (Gen dan Cheng, 1997)

```
procedure Crossover
begin
  k  ← 0;
  while (k <= jumlah populasi) do
    r_k ← bangkitkan bilangan acak antara [0,1];
    if ( r_k  < probabilitas crossover) then
     pilih  v_k  sebagai parent untuk crosover
    end
    k ← k+1;
  end
end
```

Gambar 2.5 *Pseudo Code* Proses *Crossover*

Operator *crossover* yang akan dijelaskan disini adalah order *based crossover* dan *one-cut-point crossover*. Ilustrasi *Order Based Crossover* dapat dilihat pada Gambar 2.5.

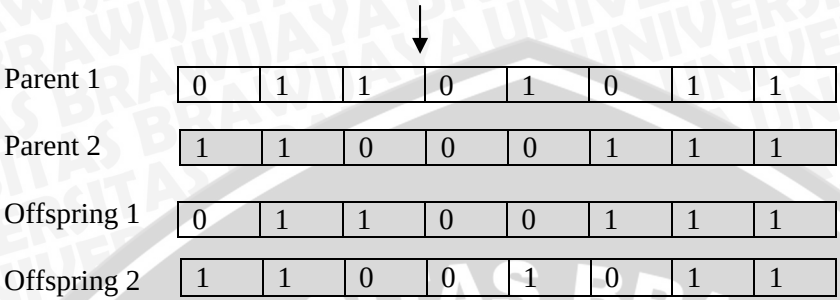


Gambar 2.6 Ilustrasi *Order Based Crossover*

Metode o*ne-cut-point crossover* analog dengan implementasi *binary*. Ilustrasi o*ne-cut-point crossover* dapat dilihat pada Gambar 2.7.

Algoritmanya adalah:

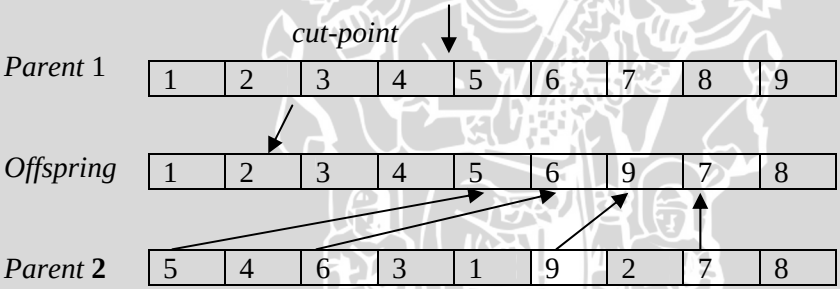
1. Memilih *site* secara *random* dari *parent* pertama.
2. Isi disebelah kanan *site* pada *parent* pertama ditukar dengan *parent* kedua untuk menghasilkan *offspring* (Gen dan Cheng, 1997 dan Syamsudin, 2004).



Gambar 2.7 Ilustrasi *one-cut-point crossover*

One-cut-point crossover dikembangkan oleh Reeves, berikut merupakan langkah-langkahnya:

1. Memilih satu *cut-point* secara *random* / acak dari *parent* pertama
2. Isi disebelah kanan *site* disesuaikan dengan urutan dari *parent* kedua untuk menghasilkan *offspring* (Gen dan Cheng, 1997 dan Syamsudin, 2004).



Gambar 2.8 Ilustrasi *One-Cut-Point Crossover Reeves*

2.3.3.2 Mutasi

Setelah mengalami proses *crossover*, pada *offspring* dapat dilakukan mutasi. Mutasi dilakukan dengan cara melakukan perubahan pada satu gen atau lebih dari sebuah individu. Peluang mutasi mengendalikan banyaknya gen baru yang akan dimunculkan untuk dievaluasi. Jika peluang mutasi terlalu kecil, banyak gen yang mungkin berguna tidak pernah dievaluasi. Tetapi bila peluang mutasi terlalu besar, maka akan terlalu banyak gangguan acak, sehingga anak akan kehilangan kemiripan dari induknya.

Mutasi berfungsi untuk menggantikan gen yang hilang dari populasi selama proses seleksi serta menyediakan gen yang tidak ada dalam populasi awal. Sehingga mutasi akan meningkatkan variasi populasi. Mutasi akan sangat berperan jika pada populasi awal hanya ada sedikit solusi yang mungkin terpilih. Sehingga operasi itu sangat berguna dalam mempertahankan keanekaragaman individu dalam populasi meskipun dengan mutasi tidak dapat diketahui apa yang terjadi pada individu baru.

Shift mutasi dapat dilakukan dengan cara:

1. Menentukan dua *site* secara *random*
2. *Site* pertama ditempatkan ke *site* kedua, untuk selanjutnya digeser ke kiri seperti terlihat pada gambar berikut (Gen dan Cheng, 1997 dan Syamsudin, 2004).

1	2	3	4	5	6	7	8	9

1	7	3	4	5	6	2	8	9

Gambar 2.9 Ilustrasi *Shift Mutation*

Pseudo Code dari proses mutasi dapat dilihat pada Gambar 2.9 (Gen dan Cheng, 1997)

```
procedure Mutasi
begin
  i  ← 0;
  while (i <= jumlah populasi) do
    pilih satu kromosom secara acak;
    ambil dua gen secara acak;
    tukar posisi kedua gen;
    i ← i+1;
  end
end
```

Gambar 2.10 *Pseudo Code* Proses Mutasi

2.4 Manfaat Algoritma Genetika

Algoritma genetika berguna dan efisien jika:

1. Ruang pencarian besar, kompleks atau tidak dipahami dengan baik.

2. Domain pengetahuan jarang atau ahli pengetahuan susah untuk mengkodekan untuk mempersempit ruang pencarian.
3. Tidak ada analisa matematika yang tersedia.
4. Metode pencarian tradisional yang gagal.

2.5 *Borland Delphi 7*

Borland Delphi 7 merupakan program aplikasi *database* yang berbasis Objek Pascal (Martina, 2004). Selain itu, Delphi 7 juga mendukung adanya *object-oriented programming* (OOP). Pemrograman berorientasi objek atau disingkat OOP (*Object-oriented programming)* adalah pemrograman yang berorientasikan pada objek jadi segala sesuatu yang ingin dimanipulasi dalam program adalah objek. Sebagai contoh, penulis adalah objek, pembaca adalah objek, mobil adalah objek dan tugas akhir ini adalah objek dan sebagainya. Bahasa OOP yang dipakai Delphi adalah perluasan dari bahasa Pascal yang dikembangkan dengan nama Turbo Pascal dan untuk membuat sebuah aplikasi Delphi, Pemrogram tidak perlu mengetahui OOP secara detail karena Delphi sudah memberikan kerangkanya.

BAB III
METODOLOGI DAN PERANCANGAN

3.1 Deskripsi Masalah

Sistem yang akan dibuat merupakan pencarian rute tercepat dengan kendala *time window*. Setiap pelanggan sudah menentukan kapan mereka akan menerima kiriman barang dari perusahaan jasa pengiriman barang pada saat mereka berada dirumah atau pada alamat yang sudah tertera pada paket yang akan dikirimkan.

Sistem ini dibuat untuk memudahkan perusahaan dalam hal pengambilan keputusan untuk mengantar barang dimana barang yang diantarkan merupakan paket khusus, dimana paket khusus hanya bisa diterima oleh orang yang bersangkutan atau sudah diwakilkan sebelumnya. Dengan adanya sistem ini diharapkan dapat membantu perusahaan dalam memecahkan masalah optimasi baik terhadap biaya perjalanan dan waktu, sehingga segala sesuatu yang akan dilakukan menjadi lebih efektif dan efisien. Dengan adanya suatu *interface/*antarmuka yang dapat mempermudah *user* karena ditampilkan pada layar komputer dan *user* juga dapat memprediksi lama perjalanan ke masing-masing pelanggan sehingga dapat sampai ketempat pelanggan pada waktu yang sudah ditentukan. Dalam hal ini *user* adalah seorang *administrator* dari perusahaan pengantar barang yang nantinya bertugas menjalankan sistem ini serta menjelaskan hasil yang diperoleh dari sistem ini kepada karyawan yang bertugas mengantar barang.

Algoritma yang digunakan pada sistem ini adalah algoritma genetika dimana algoritma genetika dapat digunakan untuk menyelesaikan masalah optimasi yang kompleks seperti mencari rute paling optimum dengan memperhatikan kondisi jalan misalnya kepadatan lalulintas, jalan satu arah dan lain-lain. Dalam tugas ini akan dijelaskan tentang penerapan algoritma genetika untuk mencari rute yang paling optimum dari *kota* asal ke *kota* tujuan. Sistem algoritma genetika yang telah didesain menggunakan representasi kromosom dalam bentuk bit string. Karena itu jenis mutasi yang digunakan adalah mutasi bit (Annis Hanawati, Thiang Dan Eleazar, 2003).

3.2 Pengumpulan Data

Pengumpulan data merupakan usaha untuk memperoleh data atau dokumen yang dibutuhkan dalam tugas akhir ini dan untuk selanjutnya data tersebut akan diproses sesuai dengan kebutuhan.

3.2.1 Studi literatur

Studi literatur merupakan cara pengumpulan data yang diperoleh dengan mengumpulkan berbagai sumber pustaka, baik berupa buku-buku, jurnal, laporan penelitian, dan lain sebagainya untuk ditelaah lebih lanjut sebagai bahan pendukung tugas akhir ini.

3.2.2 Dialog, diskusi, dan konsultasi

Dialog, diskusi dan konsultasi dengan pembimbing tugas akhir dan rekan-rekan mahasiswa dengan tujuan untuk memperoleh solusi pengambilan metode pengujian yang efektif.

3.3 Faktor-Faktor yang Mempengaruhi Pencarian Rute dan Waktu Optimal dalam Pengantaran Barang dengan Kendala *Time Window*

Beberapa komponen yang mempengaruhi pencarian rute dan waktu pengantaran secara optimal terdapat antara lain:

a. Banyaknya kunjungan
   Banyaknya pelanggan yang harus dikunjungi pada satu waktu tertentu.
b. Waktu tunggu
   Ketika *time window* belum terbuka maka akan mengakibatkan waktu tunggu.
c. Penalti
   Apabila *time window* pada kota yang akan dikunjungi sudah tertutup maka akan dikenakan penalti.
d. Waktu Pelayanan
   Waktu yang dibutuhkan pegawai pengantaran barang untuk menurunkan dan menyerahkan barang kepada pelanggan.
e. Waktu perjalanan
   Lama perjalanan yang dibutuhkan untuk mengunjungi masing-masing kota.

3.4 Waktu Tunggu

Waktu tunggu adalah waktu yang deberikan apabila kedatangan terjadi lebih awal dari *start time window*. Cara perhitungan waktu tunggu seperti pada rumus :

$$\text{Waktu tunggu} = \textit{start time window} - \text{waktu tiba}$$

Dengan adanya rumus tersebut maka dapat diketahui lama waktu tunggu, yaitu selisih antara *start time window* dan waktu tiba.

3.5 Penalti

Penalti diberikan apabila kedatangan melebihi batas *finish time window*. Cara perhitungan penalti adalah sebagai berikut :

$$\text{Penalti} = \frac{\text{waktu tiba} - \textit{finish time window}}{10} * a$$

Keterangan :
a : adalah masukan nilai penalti yang diberikan pada aplikasi

Dengan adanya rumus tersebut maka dapat diketahui penalti yang terjadi, yaitu selisih antara waktu tiba dan *finish time window*. Dengan adanya waktu tunggu dan penalti tentunya waktu yang dibutuhkan untuk mengantarkan barang semakin lama sehingga berpengaruh terhadap nilai *fitness* dimana semakin lama waktu perjalanan maka nilai *fitness* semakin kecil.

3.6 Model Genetika

Pada dasarnya algoritma genetika adalah algoritma pencarian solusi terbaik dari begitu banyak solusi yang ada. Pertama algoritma genetika bekerja dengan membuat beberapa solusi secara acak. Solusi tersebut akan mengalami proses evolusi secara terus-menerus dan akan menghasilkan suatu solusi yang lebih baik. Setiap solusi yang terbentuk mewakili satu kromosom dan satu individu terdiri dari satu kromosom. Kumpulan dari individu-individu inilah yang akan membentuk suatu populasi. Kemudian dari populasi tersebut

akan lahir individu-individu baru sampai dengan sejumlah generasi yang ditentukan.

Penyelesaian masalah dalam penelitian ini dibagi menjadi tiga tahap, yaitu

1. Membangkitkan urutan perjalanan secara acak sehingga akan terbentuknya susunan kromosom yang merepresentasikan urutan perjlanan.
2. Pembagian *time window* pada masing-masing kota.
3. Mengolah data-data yang ada dengan menggunakan Algoritma genetika untuk mendapatkan rute optimal.

3.6.1 Pengkodean Kromosom

Pencarian rute terpendek pada algoritma genetika dilakukan sekaligus atas sejumlah solusi yang mungkin terjadi yang dikenal sebagai kromosom. Individu-individu yang terdapat dalam suatu populasi disebut kromosom.

Inisialisasi kromosom bergantung pada masalah yang akan diselesaikan. Pengkodean yang digunakan untuk menentukan rute ini yaitu pengkodean permutasi. Masing-masing kromosom berisi *gen-gen* yang merepresentasikan rute pengantaran barang.

1→2→5→4→3→1

Pada urutan perjalanan diatas dapat direpresentasikan dalam bentuk kromosom sehingga menjadi seperti pada Gambar 3.1.

1	2	5	4	3	1

Gambar 3.1 Inisialisai Kromosom

Pada Gambar 3.1 Jumlah *gen* yang ada dalam satu kromosom sama dengan jumlah kunjungan yang dilakukan dan dapat diketahui urutan perjalanan dan titik mulai serta titik akhir dari perjalanan tersebut.

3.6.2 Fungsi Evaluasi

Setelah individu-individu dalam populasi telah terbentuk, maka langkah selanjutnya adalah menghitung nilai *fitness* setiap individu. Fungsi *fitness* sendiri bertujuan untuk mengetahui baik tidaknya solusi yang ada pada suatu individu, setiap individu pada populasi

harus memiliki nilai pembandingnya. Selanjutnya dari nilai *fitness* didapatkan solusi terbaik dengan cara pengurutan nilai *fitness* dari individu-individu.

Setelah kromosom terbentuk, maka selanjutnya dilakukan proses perhitungan fitness. Aturan penghitungan fungsi *fitness* yang digunakan adalah sebagai berikut :

$$Fitness = \frac{1000}{Cost + \text{Penalti}}$$

Nilai penalti diberikan apabila pada saat kunjungan tiba pada suatu kota yang *time window* sudah tertutup dan nilai penalti mempunyai pengaruh yang lebih besar terhadap nilai *fitness. Cost* adalah waktu yang dibutuhkan untuk sampai ketempat tujuan masing-masing kota. Nilai penalti dan *cost* dalam satuan menit. Semakin kecil nilai penalt*i* dan *cost*, maka nilai *fitness* akan semakin besar.

3.6.3 Seleksi

Seleksi yang digunakan adalah *roulette wheell*, dimana individu yamg memiliki nilai *fitness* besar mempunyai kemungkinan lebih besar untuk menjadi induk. Sedangkan untuk nilai *fitness* terburuk akan digantikan oleh individu baru dengan nilai *fitness* yang lebih baik.

3.6.4 *Crossover*

Apabila proses seleksi telah dilakukan dan sudah terpilih individu yang akan dijadikan induk maka operator berikutnya adalah *crossover*. *Crossover* adalah cara mengkombinasikan gen-gen induk untuk menghasilkan keturunan baru. *Crossover* yang digunakan adalah *one cut point crossover*. Pada *crossover* ini dilakukan dengan cara menukar nilai gen pada posisi gen yang sama dari kedua induk

Penentuan titik potong pada proses *crossover* ini adalah dilakukan secara random dengan ketentuan, *random*(n-2)+2, dengan n adalah jumlah node. Pada proses *crossover* ini terdapat *crossover rate* dimana diharapkan terdapat individu sebanyak n yang akan mengalami *crossover*.

Gambar 3.2 *Crossover one-cut pint*

Langkah 1 :
Menyalin gen dari induk 1 ke induk anak.

1	2	4	...	…	…	...

Langkah 2 :
Kemudian menyalin gen dari induk 2 ke anak.

3	2	1

Langkah 3 :
Menggabungkan gen-gen dari induk-induk tersebut, untuk melengkapi kromosom anak.

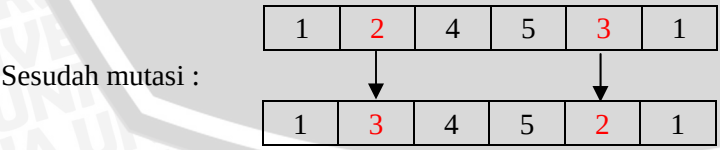
1	2	4	3	2	1

Langkah 4 :
(Seleksi)
Jika setelah proses *crossover* menghasilkan kromosom ilegal, maka akan dilakukan *repair* sesuai ketentuan yang ditetapkan.

1	2	4	3	5	1

3.6.5 Mutasi

Mutasi dilakukan untuk mencegah terjadinya konvergensi dini. Pada mutasi ada satu parameter yang sangat penting yaitu *mutation rate*. Dengan adanya *mutation rate* diharapkan terdapat sebanyak n individu yang akan mengalami mutasi. Mutasi dilakukan dengan cara *shift* atau penukaran. Mutasi dilakukan dengan cara menukar nilai dari gen dari 2 titik. Pemilihan gen yang akan ditukar dilakukan secara random. Berikut ilustrasi mutasi secara *shift.*
Sebelum mutasi:



Gambar 3.3 Ilustrasi mutasi cara *shift* satu penukaran

Pada Gambar 3.4 adalah ilustrasi mutasi penukaran gen sebanyak N.

1	3	6	5	2	8	4	9	7	1

Sesudah mutasi :

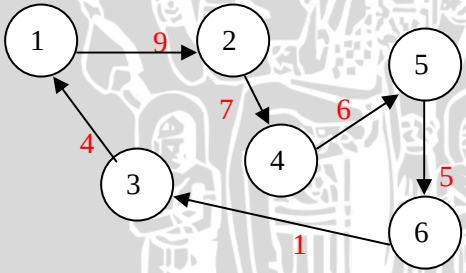
1	3	4	9	2	8	6	5	7	1

Gambar 3.4 Ilustrasi mutasi shift penukaran gen sebanyak N

Tujan dilakukan mutasi sebanyak N adalah untuk mendapatkan variasi individu yang lebih banyak.

3.7 Contoh Perhitungan Manual

Misal terdapat 5 kota yang harus dikunjungi permasalahan algoritma genetika untuk mencari rute terpendek dengan kendala *time window*. Dalam penyelesaian perhitungan manual ini digunakan populasi berukuran 4 individu dan dilakukan 2 generasi.



Gambar 3.5 Penggambaran Graf Berbobot

Keterangan : 1. Node 1 merupakan depot dimana *start* dan *finish* terdapat pada node 1.
2. Satuan ukuran biaya perjalanan kelipatan 10 menit

Tabel 3.1 *Time window*

Kota	*Time window*
1	08.00 – 16.00

2	13.20 – 14.50
3	09.10 – 09.40
4	12.10 – 13.10
5	15.30 – 16.00
6	11.00 – 11.30

Dari Tabel 3.1 terdapat waktu kunjungan masing-masing kota, dimana setiap kota harus dikunjungi pada waktu yang sudah ditentukan.

Tabel 3.2 Matrik *Cost*

Kota	**1**	**2**	**3**	**4**	**5**	**6**
1	0	9	4	3	2	8
2	9	0	7	7	4	4
3	4	7	0	5	8	1
4	3	7	5	0	6	4
5	2	4	8	6	0	5
6	8	4	1	4	5	0

Dari data pada Gambar 3.5 diubah menjadi bentuk matrik, matrik *cost* merepresentasikan *cost* yang dibutuhkan untuk melakukan perjalanan antar kota.

3.7.1 Inisialisasi Kromosom

Pada proses inisialisai kromosom pembentukan susunan kromosom dilakukan secera acak.

Kromosom 1	1	5	4	3	2	6	1
Kromosom 2	1	4	5	6	2	3	1
30	1	5	4	3	6	2	1

Kromosom 3

Kromosom 4

1	2	5	3	4	6	1

3.7.2 Evaluasi

Setelah melakukan proses inisialisasi kromosom, selanjutnya adalah melakukan pengecekan, untuk mengetahui apakah kromosom-kromosom tersebut telah memenuhi syarat atau tidak. syarat yang dimaksud adalah sampainya pegawai mengantarkan barang pada masing-masing pelanggan dengan waktu yang sudah ditentukan oleh pelanggan, apabila pegawai dalam mengantarkan barang terlalu awal meka mengakibatkan waktu tunggu sebaliknya apabila terlamabat maka akan dikenakan penalti.

Keterangan :

W*b* **:** waktu berangkat
Cost : Biaya perjalanan
Wt : waktu tunggu
Wp : waktu pelayanan = 10 menit
Total *Cost* : *cost* + Wt + Wp

Tabel 3.3 Perhitungan *cost* Kromosom 1

Induk 1

1	5	4	3	2	6	1

Rute	Wb	cost		Wt		Wp		Total cost	penalti
1 – 5	08.00	08.20	20	15.30	430	15.40	10	460	0
5 – 4	15.40	16.40	60	16.40	0	16.50	10	70	210
4 – 3	16.50	17.40	50	17.40	0	17.50	10	60	480
3 – 2	17.50	19.00	70	19.00	0	19.10	10	80	250
2 – 6	19.10	19.50	40	19.50	0	20.00	10	50	500
6 - 1	20.00	21.20	80	21.20	0	21.20	0	80	320
	Total							800	1760

Fitness : 1000/2560 = 0.390625

Pada Tabel 3.3 dapat diketahui perhitungan manual dimana keberangkatan dimulai dari kota 1 yang merupakan depot, awal dan

akhir keberangkatan terdapat pada kota 1. Keberangkatan dimulai pada pukul 08.00 menuju ke kota 5 dengan *cost* perjalanan selama 20 menit, sesampainya dikota 5 *start time window* pada 15.30 maka mengakibatkan waktu tunggu selama 430 dan waktu pelayanan selama 10 menit. Selanjutnya meninggalkan kota 5 pada pukul 15.40 menuju kota 4 *cost* perjalanan 60 menit, sesampainya dikota 4 mengalami keterlambatan maka dikenakan penalti, pada perhitungan manual ini nilai penalti yang diberikan sebesar 10 menit sehingga penaltinya menjadi 210 menit. Perhitungan selanjutnya sama seperi pada langkah-langkah diatas tetapi untuk kembali ke depot tidak dikenakan waktu tunggu dan waktu pelayanan tapi terdapat penalti.

Tabel 3.4 Perhitungan *cost* Kromosom 2

Induk 2 :

1	4	5	6	2	3	1

Rute	Wb	Cost		Wt		Wp		Total cost	penalti
1 - 4	08.00	08.30	30	12.10	220	12.20	10	260	0
4 - 5	12.20	13.20	60	15.30	130	15.00	10	200	0
5 - 6	15.00	16.30	50	16.30	0	16.40	10	60	300
6 - 2	16.40	17.20	40	17.20	0	17.30	10	50	150
23	17.30	18.40	70	18.40	0	18.50	10	80	540
3 - 1	18.50	19.30	40	19.30	0	19.30	0	40	210
	Total							690	1200

Fitness : 1000/1890 = 0.529100529100529

Tabel 3.5 Perhitungan *cost* Kromosom 3

Induk 3 :

1	5	4	3	6	2	1

Rute	Wb	cost		Wt		Wp		Total cost	penalti
1 – 5	08.00	08.20	20	15.30	430	15.40	10	160	0
5 – 4	15.40	16.40	60	16.40	0	16.50	10	70	210
4 – 3	16.50	17.40	50	17.40	0	17.50	10	60	480
3 – 6	17.50	18.00	10	18.00	0	18.10	10	20	390
6 – 2	18.10	18.50	40	18.50	0	19.00	10	50	240

2 – 1	19.00	20.30	90	20.30	0	20.30	10	100	270
	Total							750	1590

Fitness : 1000/2340 = 0.427350427350427

Tabel 3.6 Perhitungan *cost* Kromosom 4

Induk 4 :

1	2	5	3	4	6	1

Rute	Wb	cost		Wt		Wp		Total cost	penalti
1 – 2	08.00	09.30	90	13.20	230	13.30	10	330	0
2 – 5	13.30	14.10	40	15.30	80	15.40	10	130	0
5 – 3	15.40	17.00	80	17.00	0	17.10	10	90	440
3 – 4	17.10	18.00	50	18.00	0	18.10	10	60	290
4 – 6	18.10	18.50	40	18.50	0	19.00	10	50	440
6 – 1	19.00	20.20	80	20.20	0	20.20	0	80	260
	Total							740	1430

Fitness : 1000/2170= 0.460829493087558

3.7.3 Seleksi

Seleksi yang digunakan adalah seleksi *roulette wheel* dan *best rank* pada perhitungan manual ini hanya digunakan seleksi *roullete whell*, untuk lebih jelas mengetahui nilai *fitness* dari masing-masing individu pada generasi 0 dapat dilihat pada Table 3.9.

Tabel 3.7 Nilai *Fitness* kromosom generasi ke-0

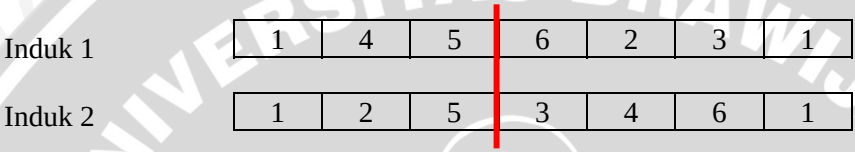
Index	***Cost***	**Penalti**	***Fitness***
Induk 1	800	1760	0.390625
Induk 2	690	1200	0.529100529100529
Induk 3	750	1590	0.427350427350427
Induk 4	740	1430	0.460829493087558

Setelah dilakukan proses evaluasi dan seleksi, proses selanjutnya adalah *crossover* dan mutasi. Tujuan dilakukanya proses tersebut adalah untuk mencari kemungkinan diperolehnya individu yang lebih baik. Pada proses pembentukan generasi kedua akan

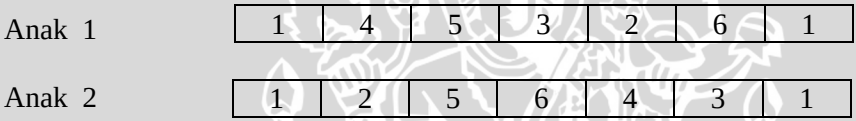
diperoleh dua anak hasil dari proses *crossover* dan dua dari hasil mutasi

3.7.4 *Crossover*

Pada proses *crossover* akan dipilih dua induk dari proses seleksi dengan metode *roullete wheel* sehingga nantinya menghasilkan 2 anak. Kromosom yang terpilih pada proses *crossover* ini adalah kromosom 2 dan 4.



Setelah dilakukan crossover dari kedua kromosom tadi maka diperoleh 2 anak seperti berikut :

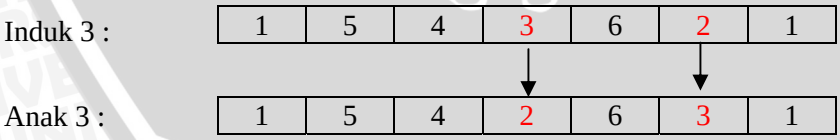


Setelah diperoleh dua anak dari proses *crossover* maka berikutnya adalah membentuk dua anak lagi dari proses mutasi.

3.7.5 Mutasi

Proses mutasi dilakukan untuk mencari kemungkinan terbentuknya kromosom yang lebih baik, dimana pada proses ini dipilih dua kromosom yaitu individu ke-3 dan ke-2, untuk lebih jelasnya proses mutasi seperti berikut ini :

Individu kedua :



Individu keempat :

Indiuk 4 :

1	4	5	6	2	3	1

Anak 4 :

1	2	5	6	4	3	1

Setelah terbentuknya kromsom baru dari proses *crossover* dan mutasi maka dilakukan evaluasi terhadap kromosom-kromosom tersebut dengan tujuan untuk memilih individu mana yang akan dgunakan untuk proses generasi selanjutnya.

Tabel 3.8 Perhitungan *cost* Kromosom anak ke-1

Anak 1

1	4	5	3	2	6	1

Rute	Wb	*cost*		Wt		Wp		Total cost	penalti
1 – 4	08.00	08.30	30	12.10	220	12.20	10	160	0
4 – 5	12.20	13.20	60	15.30	130	15.40	10	200	0
5 – 3	15.40	17.00	80	17.00	0	17.10	10	90	440
3 – 2	17.10	18.20	70	18.20	0	18.30	10	80	210
2 – 6	18.30	19.10	40	19.10	0	19.20	10	50	160
6 – 1	19.20	20.40	80	20.40	0	20.40	0	80	280
	Total							760	1390

Fitness : 1000/2150 : 0.465116279069767

Tabel 3.9 Perhitungan *cost* Kromosom anak ke-2

Anak 2

1	2	5	6	4	3	1

Rute	Wb	cost		Wt		Wp		Total cost	penalti
1 – 2	08.00	09.30	90	13.20	230	13.30	10	320	0
2 – 5	13.30	14.10	40	15.30	80	15.40	10	130	0
5 – 6	15.40	16.30	50	16.30	0	16.40	10	60	300
6 – 4	16.40	17.20	40	17.20	0	17.30	10	50	250
4 – 3	17.30	18.20	50	18.20	0	18.30	10	60	520
3 – 1	18.30	19.10	40	19.10	0	19.10	0	40	190

	Total	670	1260

Fitness : 1000/1930 : 0.518134715025907

Tabel 3.10 Tabel Perhitungan *cost* Kromosom anak ke-3

Anak 3 :

1	5	4	2	6	3	1

Rute	Wb	cost		Wt		Wp		Total cost	penalti
1 – 5	08.00	08.20	20	15.30	430	15.40	10	460	0
5 – 4	15.40	16.40	60	16.40	0	16.50	10	70	210
4 – 2	16.50	18.00	70	18.00	0	18.10	10	80	190
2 – 6	18.10	18.50	40	18.50	0	19.00	10	50	440
6 – 3	19.00	19.10	10	19.10	0	19.20	10	20	570
3 – 1	19.20	20.00	40	20.00	0	20.00	0	40	240
	Total							720	1650

Fitness : 1000/2370 : 0.421940928270042

Tabel 3.11 Tabel Perhitungan *cost* Kromosom anak ke-4

Anak 4 :

1	2	5	6	4	3	1

Rute	Wb	cost		Wt		Wp		Total cost	penalti
1 – 2	08.00	09.30	90	11.20	230	11.30	10	330	0
2 – 5	11.30	14.10	40	15.30	80	15.40	10	130	0
5 – 6	15.40	16.30	50	16.30	0	16.40	10	60	300
6 – 4	16.40	17.20	40	17.20	0	17.30	10	50	250
4 – 3	17.30	18.20	50	18.20	0	18.30	10	60	520
3 – 1	18.30	19.10	40	19.10	0	19.10	0	40	190
	Total							670	1260

Fitness : 1000/1930 : 0.518134715025907

Pada Tabel 3.12 Individu Hasil dari Generasi Pertama.

Tabel 3.12 Generasi ke-1

Index	*Cost*	Penalti	*Fitness*
Induk 2	690	1200	0.529100529100529
Anak 4	670	1260	0.518134715025907
Anak 1	760	1390	0.465116279069767
Induk 4	740	1430	0.460829493087558

Setelah diperoleh individu generasi pertama, maka proses selanjutnya adalah proses generasi ke-2. Pada proses generasi ke-2 individu generasi pertama yang digunakan sebagai induk.

Individu 1	1	4	5	6	2	3	1
Individu 2	1	2	5	6	4	3	1
Individu 3	1	4	5	3	2	6	1
Individu 4	1	2	5	3	4	6	1

Setelah ditentukan individu untuk proses generasi berikutnya maka akan dilakukan evaluasi. Evaluasi dilakukan untuk menentukan individu mana yang akan diproses *crossover* dan mutasi.

Tabel 3.13 Perhitungan *cost* kromosom 1 generasi ke-1

Kromosom 1	1	4	5	6	2	3	1

Rute	Wb	cost		Wt		Wp		Total cost	penalti
1 - 4	08.00	08.30	30	12.10	220	12.20	10	260	0
4 – 5	12.20	13.20	60	15.30	130	15.00	10	200	0
5 – 6	15.00	16.30	50	16.30	0	16.40	10	60	300
6 – 2	16.40	17.20	40	17.20	0	17.30	10	50	150
2 – 3	17.30	18.40	70	18.40	0	18.50	10	80	540
3 - 1	18.50	19.30	40	19.30	0	19.30	0	40	210
	Total							690	1200

Fitness : 1000/1890 = 0.529100529100529

Tabel 3.14 Perhitungan *cost* Kromosom 2 generasi ke-1

Kromosom 2

1	2	5	6	4	3	1

Rute	Wb	cost		Wt		Wp		Total cost	penalti
1 – 2	08.00	09.30	90	11.20	230	11.30	10	330	0
2 – 5	11.30	14.10	40	15.30	80	15.40	10	130	0
5 – 6	15.40	16.30	50	16.30	0	16.40	10	60	300
6 – 4	16.40	17.20	40	17.20	0	17.30	10	50	250
4 – 3	17.30	18.20	50	18.20	0	18.30	10	60	520
3 – 1	18.30	19.10	40	19.10	0	19.10	0	40	190
	Total							670	1260

Fitness : 1000/1930 : 0.518134715025907

Tabel 3.15 Perhitungan *cost* Kromosom 3 generasi ke-1

Kromosom 3

1	4	5	3	2	6	1

Rute	Wb	*cost*		Wt		Wp		Total cost	penalti
1 – 4	08.00	08.30	30	12.10	220	12.20	10	160	0
4 – 5	12.20	13.20	60	15.30	130	15.40	10	200	0
5 – 3	15.40	17.00	80	17.00	0	17.10	10	90	440
3 – 2	17.10	18.20	70	18.20	0	18.30	10	80	210
2 – 6	18.30	19.10	40	19.10	0	19.20	10	50	160
6 – 1	19.20	20.40	80	20.40	0	20.40	0	80	280
	Total							760	1390

Fitness : 1000/2150 : 0.465116279069767

Tabel 3.16 Perhitungan *cost* Kromosom 4 generasi ke-1

1	2	5	3	4	6	1

Kromosom 4

Rute	Wb	cost		Wt		Wp		Total cost	penalti
1 – 2	08.00	09.30	90	13.20	230	13.30	10	330	0
2 – 5	13.30	14.10	40	15.30	80	15.40	10	130	0
5 – 3	15.40	17.00	80	17.00	0	17.10	10	90	440
3 – 4	17.10	18.00	50	18.00	0	18.10	10	60	290
4 – 6	18.10	18.50	40	18.50	0	19.00	10	50	440
6 – 1	19.00	20.20	80	20.20	0	20.20	0	80	260
	Total							740	1430

Fitness : 1000/2170= 0.460829493087558

Pada proses generasi ke-2 cara yang digunakan sama yaitu dilakukan dua proses *crossover* dan dua proses mutasi sehingga diperoleh hasil dari *crossover* Individu ke-3 dan ke-4

Individu ke-3

1	4	5	3	2	6	1

Individu ke-4

1	2	5	3	4	6	1

Setelah dilakukan crossover dari kedua kromosom tadi maka diperoleh 2 anak seperti berikut :

Anak 1

1	4	5	3	2	6	1

Anak 2

1	2	5	3	4	6	1

Proses pembentukan dua individu baru berikutnya dilakukan dengan cara mutasi, Individu yang dimutasi adalah individu ke-1 sebanyak dua kali mutasi.

Individu ke-1

1	4	5	6	2	3	1

Anak 3

1	5	4	6	2	3	1

Individu 1 :

1	4	5	6	2	3	1

Anak 4 :

1	4	5	6	3	2	1

Setelah terbentuk individu baru dari proses *crossover* dan mutasi, maka dilakukan evaluasi.

Tabel 3.17 Perhitungan *Cost* Anak ke-1

Anak 1

1	4	5	3	2	6	1

Rute	Wb	cost		Wt		Wp		Total cost	penalti
1 – 4	08.00	08.30	30	12.10	220	12.20	10	260	0
4 – 5	11.20	13.20	60	15.30	130	15.40	10	200	0
5 – 3	15.40	17.00	80	17.00	0	17.10	10	90	440
3 - 2	17.10	18.20	70	18.20	0	18.30	10	80	210
2 – 6	18.30	19.10	40	19.10	0	19.20	10	50	460
6 – 1	19.20	20.40	80	20.40	0	20.40	0	80	280
	Total							760	1390

Fitness : 1000/2150 = 0.465116279069767

Tabel 3.18 Perhitungan *Cost* Anak ke-2

Anak

1	2	5	3	4	6	1

Rute	Wb	cost		Wt		Wp		Total cost	penalti
1 – 2	08.00	09.30	90	13.20	230	13.30	10	330	0
2 – 5	13.30	14.10	40	15.30	80	15.40	10	130	0
5 – 3	15.40	17.00	80	17.00	0	17.10	10	90	440
3 – 4	17.10	18.00	50	18.00	0	18.10	10	60	290
4 – 6	18.10	18.50	40	18.50	0	19.00	10	50	440
6 – 1	19.00	20.20	80	20.20	0	20.30	0	80	260
	Total							740	1430

*Fitnes*s : 1000/2170 = 0.460829493087558

Tabel 3.19 Tabel Perhitungan *Cost* Anak ke-3

Anak

1	5	4	6	2	3	1

3

Rute	Wb	cost		Wt		Wp		Total cost	penalti
1 – 5	08.00	08.20	20	15.30	430	15.40	10	460	0
5 – 4	15.40	16.40	60	16.40	0	16.50	10	70	210
4 – 6	16.50	17.30	40	17.30	0	17.40	10	50	360
6 – 2	17.40	18.20	40	18.20	0	18.30	10	50	210
2 – 3	18.30	19.40	70	19.40	0	19.50	10	80	600
3 – 1	19.50	20.30	40	20.30	0	20.30	0	40	270
	Total							750	1650

Fitnsess : 1000/2400 = 0.416666666666667

Tabel 3.20 Perhitungan *Cost* Anak ke-4

Anak 4

1	4	5	6	3	2	1

Rute	Wb	cost		Wt		Wp		Total cost	penalti
1 – 4	08.00	08.30	30	12.10	220	12.20	10	260	0
4 – 5	12.20	13.20	60	15.30	130	15.40	10	200	0
5 – 6	15.40	16.30	50	16.30	0	16.40	10	60	300
6 – 3	16.40	16.50	10	16.50	0	17.00	10	20	430
3 – 2	17.00	18.10	70	17.10	0	18.20	10	80	200
2 – 1	18.20	19.50	90	19.50	0	19.50	0	90	230
	Total							710	1160

Fitnsess : 1000/1870 = 0.53475935828877

Setelah dilakukan proses evaluasi pada generasi pertama maka diperoleh hasil generasi ke-2. Pada Tabel 3.21 merupakan hasil dari generasi ke-2.

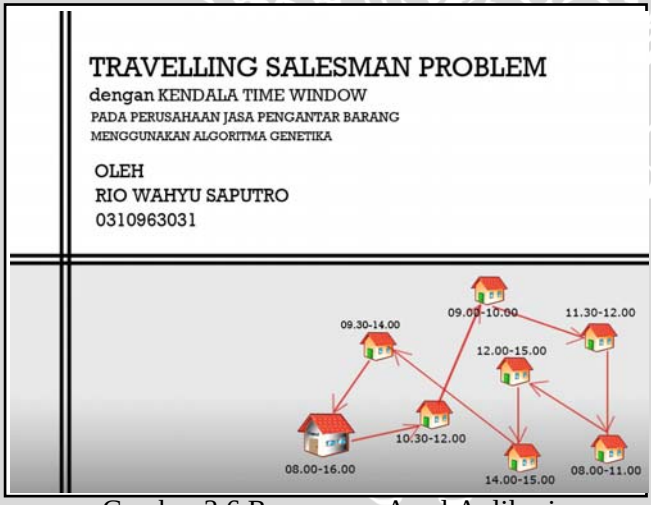
Tabel 3.21 Generasi ke-2

Index	*Cost*	Penalti	*Fitness*
Anak ke-4	710	1160	0.53475935828877
Kromosom 1	690	1200	0.529100529100529
Kromosom 2	670	1260	0.518134715025907
Anak ke-1	760	1390	0.465116279069767

Iterasi akan dihentikan apabila setelah dilakukan beberapa uji coba dan hasil dari nilai *fitness* sama atau disebut konvergen

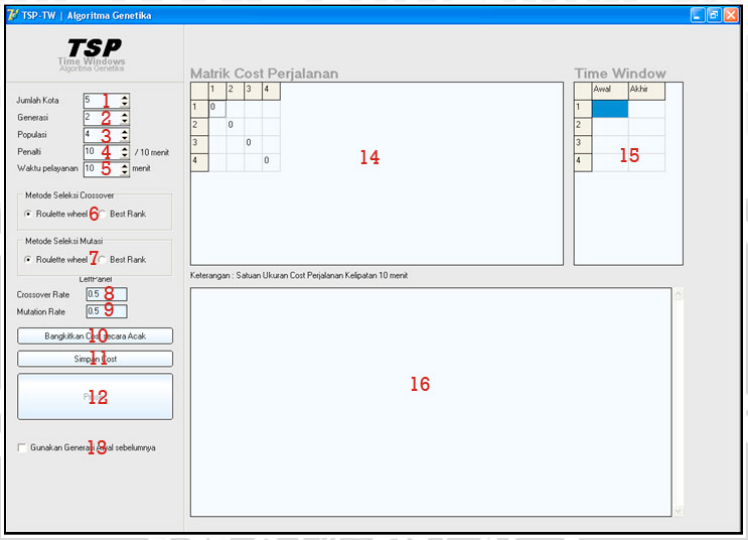
3.8 Perancangan Aplikasi

Pada perancangan antarmuka ini akan dijelaskan bentuk dari rancangan antarmuka yang akan dirancang dalam perangkat lunak ini. Dalam hal ini antarmuka dirancang harus memudahkan *user* dalam memakainya. Pada Gambar 3.6 adalah rancangan awal dari aplikasi.



Gambar 3.6 Rancangan Awal Aplikasi

Setelah rancangan awal, selanjutnya terdapat rancangan main aplikasi. Pada main aplikasi ini user dapat memasukan nilai pada parameter-parameter yang tersedia, untuk lebih jelasnya ditunjukan pada Gambar 3.7.



Gambar 3.7 Rancangan Main Aplikasi

Keterangan :

1. Input jumlah kota
2. Iterasi/Generasi
3. Besar Individu dalam satu populasi
4. Waktu Penalti
5. Waktu pelayanan
6. Metode seleksi *crossover*
7. Metode seleksi mutasi
8. *Crossover rate*
9. *Mutation rate*
10. Bangkitkan *cost* secara acak
11. Simpan *cost*
12. Proses aplikasi
13. Gunakan generasi awal sebelumnya
14. Matrik *cost* perjalanan
15. Tabel *time window*
16. Tampilan hasil

3.9 Langkah-langkah Menjalankan Aplikasi

Untuk menjalankan aplikasi pada Gambar 3.6 akan dijelaskan langkah-langkahnya sebagai berikut :

1. User memasukkan jumlah kota yang akan dikunjungi pada input jumlah kota.
2. Selanjutnya memberikan masukan generasi/iterasi yang berfungsi untuk mencari generasi terbaik.
3. Memberikan besar ukuran individu dalam satu populasi.
4. Memberikan nilai penalti.
5. Memberikan nilai waktu pelayanan.
6. Setelah parameter diisi kemudian menentukan metode *crossover*.
7. Selanjutnya menentukan metode mutasi.
8. Selanjutnya memberikan masukan *crossover rate* berupa bilangan desimal.
9. Proses selanjutnya denagn memberikan masukan *mutation rate* dengan bilangan desimal.
10. Setelah semua parameter diisi kemudian adalah membangkitkan *cost* secara acak.
11. Apabila tidak ingin mengganti individu awal, maka *check box* ditandai.
12. Setelah semua parameter diisi untuk hasil dari pembangkitan *cost* secara acak ditampilkan dalam bentuk matrik, nilai pada matrik bisa diubah sesuai keinginan.
13. Pada tabel *time window* akan ditampilkan hasil dari proses pembangkitan secara acak, nilai pada tabel *time window* juga bisa diubah sesuai keinginan tetapi pada kota 1 ridak diperbolehkan diubah karena kota 1 merupakan depot.
14. Hasil dari proses tersebut ditampilkan pada memo, dimana akan ditampilkan urutan perjalanan dan proses algoritma genetika beserta nilai *fitness* dari masing-masing individu.

BAB IV
IMPLEMENTASI DAN PEMBAHASAN

Pada bab ini akan dibahas mengenai implementasi dari representasi rancangan ke bahasa pemrograman yang dimengerti oleh komputer. Selanjutnya hasil implementasi yang dihasilkan oleh perangkat lunak, akan dievaluasi sebagai bahan pengambil kesimpulan.

4.1 Lingkungan Implementasi

Lingkungan implementasi yang akan dijelaskan dalam subbab ini adalah lingkungan implementasi perangkat keras dan perangkat lunak.

4.1.1 Lingkungan perangkat keras

Perangkat keras yang digunakan dalam pengembangan sistem pencarian rute terpendek dengan kendala *time window* ini adalah sebagai berikut :

1. Prosesor AMD Barthon 2.80 Mhz
2. Memori 512 MB
3. Harddisk dengan kapasitas 80 GB
4. Monitor 15 ”
5. Keyboard
6. Mouse

4.1.2 Lingkungan perangkat lunak

Perangkat lunak yang digunakan dalam pengembangan sistem pencarian rute terpendek dengan kendala *time window* ini adalah :

1. Sistem Operasi Windows XP SP 1
2. Borland Delphi 7

4.2 Deskripsi Program

4.2.1 Struktur Data

Pada implementasi program ini, bahasa pemrograman yang digunakan adalah *Borland Delphi 7*. Dalam perangkat lunak ini struktur data yang dipakai sebagai berikut:

```
type
  TKromosom   = array [1..MAXKOTA+1] of integer;
  TMatrixCost = array [1..MAXKOTA,1..MAXKOTA]  of integer;

TJam = record
    Jam   : integer;
    menit : integer;
  end;

  TTimeWindow = record
    start, finish : TJam;
  end;

TIndividu   = record
    Kromosom      : TKromosom;
    Cost, Penalti : integer;
    Fitness       : real;
    TimeWindow    : TTimeWindow;
    BatasAtasRoulette : real;
  end;
  TPopulasi = array [1..MAXPOPULASI] of TIndividu

var
  MatrixCost : TMatrixCost;
  TimeWindow : array [1..MAXKOTA] of TTimeWindow;
  N, PEN, SERVICE TIME : integer;
```

Gambar 4.1 Struktur Data

Tabel 4.1 Keterangan Struktur Data Algoritma genetika

Struktur Data	Keterangan
`TKromosom`	Tipe data bentukan berupa *array* dari integer untuk menyimpan kromosom
`TMatrixCost`	Tipe data bentukan berupan *array* dari integer untuk menyimpan biaya perjalanan antar kota
`TJam`	Tipe data bentukan berupa *record* untuk menyimpan waktu
`Jam`	Variabel bertipe integer untuk menyimpan waktu dalam satuan jam
`Menit`	Variabel bertipe integer untuk menyimpan waktu dalam satuan menit
`TTimeWindow`	Tipe data bentukan berupa *record* untuk menyimpan time window tiap-tiap kota
`Start`	Variabel bertipe `Tjam` untuk menyimpan awal *time window*

finish	Variabel bertipe `Tjam` untuk menyimpan akhir *time window*
TIndividu	Tipe data bentukan untuk menyimpan data individu, terdiri dari `Kromosom`, `Cost`, `Penalti`, `Fitness`, `TimeWindow`, `BatasAtasRoulette`
Kromosom	Variabel bertipe `Tkromosom` untuk menyimpan kromosom
Cost	Variabel bertipe integer untuk menyimpan *cost* perjalanan antar kota
penalti	Variabel bertipe integer untuk menyimpan penalti tiap kota
Fitness	Variabel bertipe real untuk menyimpan nilai *fitness*
TimeWindow	Variabel bertipe `TTimewindow` untuk menyimpan *time window* tiap kota
BatasAtasRoulette	Variabel bertipe real untuk membantu pada saat *roulette wheel*
TPopulasi	Tipe data bentukan berupa array dari `TIndividu` untuk menyimpan data populasi
MatrixCost	Variabel bertipe `TMatrixcost`
TimeWindow	Variabel bertipe *array* dengan masing-masing record bertipe `TtimeWindow` untuk menyimpan time window semua kota
N	Variabel bertipe integer untuk menyimpan banyaknya kota sesuai input oleh user
PEN	Variabel bertipe integer untuk menyimpan besaran penalti sesuai input oleh user
SERVICE_TIME	Variabel bertipe integer untuk menyimpan besaran waktu pelayanan sesuai input oleh user

4.2.2 Pembentukan Awal Individu

Tahap awal pembentukan individu dilakukan dengan menyusun stock kota pada `CityStock` berbentuk *array* yang akan dimasukkan dalam kromosom, kemudian satu persatu *stock* kota tersebut dipilih secara acak dan kemudian dimasukkan kedalam kromosom berurutan dari index terkecil hingga seluruh *stock* kota masuk kedalam

kromosom. *Listing* program untuk melakukan pembentukan individu awal ada pada *Sourcecode* 4.1.

```
function InitRandomIndividu(N:integer): TIndividu;
var
  ind : TIndividu;
  i,j, rnd   : integer;
  a, b, temp : integer;
  CityStock : array [1..1000] of integer;
  StockLength : integer;

begin
Randomize;
for i:=1 to N-1 do
    CityStock[i] := i+1;
  StockLength := N-1;

begin
  Randomize;

  for i:=1 to N-1 do
    CityStock[i] := i+1;
  StockLength := N-1;

  Ind.Kromosom[1] := 1;
  for i:=2 to N do
  begin
    rnd := random(StockLength)+1;
    Ind.Kromosom[i] := CityStock[rnd];
    for j:=rnd to StockLength do
      CityStock[j] := CityStock[j+1];
    dec(StockLength);
  end;

  Ind.Kromosom[N+1] := 1;

  Ind := GetCost(Ind);

  result := Ind;
end;
```

Sourcecode 4.1 Proses pembentukan awal individu

Pada S*ourcecode* 4.1 jumlah `CityStock` ini sesuai dengan jumlah inputan kota. Sehingga pada *string grid* akan diperoleh baris dan kolom yang sama. Dari *string grid* tersebut dapat diketahui panjang dari kromosom

4.2.3 *Print* Kromosom

Pada fungsi print kromosom ini akan menghasilkan *output* berupa *string* yang berisi susunan kromosom beserta *fitness* dari masing-masing individu. *Listing* program untuk print kromosom ada pada *Sourcecode* 4.2.

```
function PrintKromosom(Individu :TIndividu):string;
var
  i : integer;
  Str : String;
begin
  Str := '';
  for i:=1 to N+1 do
  begin
    Str := Str + IntToStr(Individu.Kromosom[i]);
    if (i<>N+1) then Str := Str + ' - ';
  end;
  Str := Str + '  :  ' + FloatToStr(Individu.Fitness);

  result := str;
end;
```

Sourcecode 4.2 Proses print kromosom

4.2.4 Seleksi *Roullete wheel*

```
function SetRoulette(Populasi: TPopulasi;
nPop:integer):TPopulasi;
var
  i : integer;
  totalfitness, temp : real;
begin
    totalfitness:=0;
    for i:=1 to nPop do
    begin
      totalFitness := totalFitness + Populasi[i].Fitness;
    end;
    temp :=0;
    for i:=1 to nPop do
    begin
   Populasi[i].BatasAtasRoulette := temp +
(Populasi[i].Fitness / totalFitness * 100);
      temp := temp + (Populasi[i].Fitness / totalFitness *
100);
    end;
  result := Populasi;
end;
```

Sourcecode 4.3 Seleksi *roullete wheel*

Proses pada seleksi dengan menggunakan roullete wheel pertama adalah menjumlahkan semua nilai *fitness* dari masing-masing individu dari satu populasi yang ada `totalFitness := totalFitness + Populasi[i].Fitness`. Setelah menjumlahkan nilai *fitness* langkah berikutnya menentukan batas atas *roulette* dimana dalam menentukan batas atas fitness adalah `temp + (Populasi[i].Fitness / totalFitness * 100)`. Setelah terdapat batas atas maka proses seleksi bias dilakukan dan individu yang memilki nilai *fitness* besar memiliki ruang yang besar pada *roulette* sehingga kemungkinan terpilih secara *random* juga besar.

4.2.5 Seleksi *Best Rank*

Pada seleksi *best rank,* individu diurutkan dari individu yang memiliki nilai *fitness* terbesar hingga terkecil. Individu yang memiliki nilai *fitness* besar berada pada urutan atas sehingga akan terpilih menjadi induk untuk proses generasi selanjutnya.

```
function SortIndv(Populasi:TPopulasi;
NPop:integer):TPopulasi;
var
  i, j :integer;
  CurInd : TIndividu;
begin
  //sorting by fitness from populasi
  for i:=1 to NPop-1 do
    for j:=1 to NPop+-i do
      if Populasi[j].Fitness<Populasi[j+1].Fitness then
      begin
        CurInd := Populasi[j+1];
        Populasi[j+1] := Populasi[j];
        Populasi[j] := CurInd;
      end;

  Result := Populasi;
end;
```

Sourcecode 4.4 Seleksi *best rank*

4.2.6 *Crossover*

Fungsi ini melakukan *crossover* dua individu (`Indv1` dan `Indv2`) yang diperoleh dari `TIndividu`. Untuk menentukan titik potongnya menggunakan *one cut point*, dimana penukaran *gen* dilakukan pada dua kromosom pada tempat yang sama.

```
function Crossover(Indv1:TIndividu; Indv2:TIndividu;
COPoint:integer):TPopulasi;
var
  Child : TPopulasi;
  i,j   : integer;
begin
  for i:=1 to COPoint do
  begin
    Child[1].Kromosom[i] := Indv1.Kromosom[i];
    Child[2].Kromosom[i] := Indv2.Kromosom[i];
  end;

  for i:=COPoint+1 to N do
  begin
    Child[1].Kromosom[i] := TryPossible(Child[1],Indv2, i);
    Child[2].Kromosom[i] := TryPossible(Child[2],Indv1, i);
  end;
  Child[1].Kromosom[N+1] := 1;
  Child[2].Kromosom[N+1] := 1;

  //hitung cost
  for i:=1 to 2 do
    Child[i] := GetCost(Child[i]);

  result := Child;
end;
```

SourceCode 4.5 *Crossover*

4.2.7 Mutasi

Pada proses mutasi ini *gen* yang ditukar diambil secara *random* dengan ketentuan `Random(N-1)+2` dan banyaknya mutasi juga tergantung pada panjang kromosom `(NC-1) div 4)`, tujuan dilakukanya mutasi sebanyak N adalah untuk mendapatkan variasi yang lebih banyak, seperti pada *sourcecode* 4.6.

```
function Mutasi(Indv:TIndividu; NC: Integer):TIndividu;
var
  i,j,k,l,temp   : integer;
begin
  Randomize;
  for i:=1 to ((NC-1) div 4) do
  begin
    k := Random(N-1)+2;
    repeat
      l := Random(N-1)+2;
    until l<>k;
```

```
    //Showmessage('Mutasi ' +
IntToStr(k)+':'+IntToStr(l));
    Temp := Indv.Kromosom[k];
    Indv.Kromosom[k] := Indv.Kromosom[l];
    Indv.Kromosom[l] := Temp;
  end;
  Indv := GetCost(Indv);

  result := Indv;
end;
```

Sourcecode 4.6 Mutasi

4.2.8 Fungsi *TryPosible*

Pada fungsi ini bertujuan untuk melakukan perbaikan pada kromosom hasil dari *crossover* dimana apabila dalam penukaran *gen* terdapat *ilegal* kromosom atau *gen* yang akan dipindah sudah terdapat pada urutan sebelumnya.

```
function TryPossible(Child:TIndividu;Indv:TIndividu;
Point:integer):Integer;
var
  i,j : integer;
  exist, bisa : boolean;
    begin
  exist := FALSE;
  for i:=2 to Point-1 do
    if Child.Kromosom[i]=Indv.Kromosom[Point] then
begin
      exist := TRUE;
      //showmessage(IntToStr(Indv.Kromosom[Point])+' sudah ada
         pada child ke-'+IntToStr(i));
      break;
    end;

  if not exist then
  begin
    result := Indv.Kromosom[Point];
    //showmessage(IntToStr(Indv.Kromosom[Point])+' masuk');
  end
  else
  begin
    j :=1;
    repeat
      bisa := TRUE;
      inc(j);
      for i:=2 to Point-1 do
      begin
```

```
if Child.Kromosom[i]=Indv.Kromosom[j] then
        begin
          //showmessage('tidak bisa cz sama');
          bisa := FALSE; break;
        end;
      end;
    until bisa;
    //showmessage('ahirnya ditempati indv ' +
IntToStr(Indv.Kromosom[j]));
    result := Indv.Kromosom[j];
  end;
end;
```

Sourcecode 4.7 Fungsi *TryPosible*

Pada proses *crossover* penukaran *gen* dilakukan dan fungsi ini dijalankan apakah sudah terdapat gen yang sama pada posisi sebelumnya, fungsi ini akan mencari satu-persatu *gen* dari urutan awal.

4.2.9 *Exist* Individu

Fungsi *exist* individu digunakan untuk memeriksa apakah suatu individu sudah ada dalam satu populasi. Pemeriksaan kesamaan individu ini dilakukan dengan membandingkan apakah *fitness* individu yang diperiksa sama dengan *fitness* individu-individu pada satu populasi. Pada proses mencari individu yang sama hanya dilakukan pada nilai *fitness*, maka memungkinkan adanya dua individu yang berbeda susunan kromosomnya tetapi memiliki nilai *fitness* yang sama akan diabaikan.

```
    function existIndividu(NTempPop : integer;
TempPop:TPopulasi; TempIndv:TIndividu):boolean;
var
  i,j : integer;
  sama : boolean;
begin
  sama := FALSE;
  for i:=1 to NTempPop do
  begin
sama := TRUE;
    for j:=2 to N do
  begin
    if TempPop[i].Kromosom[j]<>TempIndv.Kromosom[j] then
 begin
```

```
        sama:= FALSE;
        break;
      end;
    end;
    if sama then
      break;
  end;
  result := sama;
end;
```

Sourcecode 4.8 *Exist* Individu

4.2.10 Fungsi *Sort* Individu

Pada fungsi *sort* individu ini dilakukan penggabungan individu-individu sekaligus diurutkan dari nilai fitness yang paling besar berada pada urutan atas. Individu yang diurutkan berasal dari populasi awal dan dari anak hasil dari proses *crossover* dan mutasi.

```
var
  i, j :integer;
  CurInd : TIndividu;
begin
  for i:=1 to NChild do
    Populasi[NPop+i] := Child[i];

  //sorting by fitness from populasi and child
  for i:=1 to NPop+NChild-1 do
    for j:=1 to NPop+NChild-i do
   begin
      if Populasi[j].Fitness<Populasi[j+1].Fitness then
      begin
        CurInd := Populasi[j+1];
        Populasi[j+1] := Populasi[j];
Populasi[j] := CurInd;
      end;
    end;

  Result := Populasi;
```

Sourcecode 4.9 Fungsi Sort Individu

4.2.11 Proses Perhitungan *Cost*

Proses perhitungan *cost* dimulai dari kota keberangkatan yaitu kost 1 yang merupakan depot. Kota 1 merupakan kota dimana *start* dan *finish*, jam awal keberangkatan ditentukan pada pukul 08.00 pagi dan berakhir pada pukul 16.00 sore. Pada fungsi `GetCost` terdapat

beberapa variable diantaranya `cost, waiting, penalty, selisih, totalcost.`

```
function GetCost(Ind: TIndividu): TIndividu;
var
  a, b, cost, waiting, penalti, i, selisih, totalcost : integer;
  Now : TJam;
begin
  Now.jam   := 8;   //jam berangkat
  Now.menit := 0;

  cost    := 0;
  penalti := 0;

 //hitung cost perjalanan & waiting-time (waktu tunggu)
  for i:=2 to N do
  begin
    a := ind.kromosom[i-1];
    b := ind.kromosom[i];

    //hitung cost perjalanan
    cost := cost + (MatrixCost[b,a]*10);
    Now := TambahJam(Now, MatrixCost[b,a]*10);

//hitung waktu tunggu
    if (Now.Jam < TimeWindow[ind.kromosom[i]].start.Jam) or
       ((Now.Jam = TimeWindow[ind.kromosom[i]].start.Jam) AND
       (Now.menit < TimeWindow[ind.kromosom[i]].start.menit))
then
    begin
      selisih :=
selisihJam(Now,TimeWindow[ind.kromosom[i]].start);
      cost := cost + selisih;
    end;
  //hitung menit yang kena penalti
    if (Now.Jam > TimeWindow[ind.kromosom[i]].finish.Jam) or
       ((Now.Jam = TimeWindow[ind.kromosom[i]].finish.Jam) AND
       (Now.menit > TimeWindow[ind.kromosom[i]].finish.menit))
then
    begin
      selisih := selisihJam(TimeWindow[ind.kromosom[i]].finish,
                 Now);
      penalti := penalti + selisih;
    end;

    //hitung waktu pelayanan
    cost := cost + SERVICE_TIME;
    Now := TambahJam(Now, SERVICE_TIME); //tambahkan waktu
layanan
  end;
```

```
  //kembali ke depot
  a := ind.kromosom[i-1];
  Now := TambahJam(Now, MatrixCost[a,1]*10);
  cost := cost + (MatrixCost[a,1]*10);

//hitung menit yang kena penalti
  if (Now.Jam > TimeWindow[1].finish.Jam) or
     ((Now.Jam = TimeWindow[1].finish.Jam) AND (Now.menit >
     TimeWindow[1].finish.menit)) then
  begin
    selisih := selisihJam(TimeWindow[1].finish, Now);
    penalti := penalti + selisih;
  end;

  totalcost   := cost;
  ind.Cost    := totalcost;
  ind.Penalti := (penalti div 10) * Pen;
  ind.Fitness := 1000 / (ind.Cost + ind.Penalti);

  result := ind;
end;
```

Sourcecode 4.10 Proses Hitung *Cost*

Untuk menghitung *cost* perjalanan adalah :

1. menambahkan biaya perjalanan dari kota ke-1 yang merupakan depot sampai kota ke N, dimana kota ke-1 dimulai pada pukul 08.00 pagi. *Cost* pejalanan diperoleh dari *string grid* dan nilai dari *string grid* dikalikan 10 menit (`cost := cost + (MatrixCost[b,a]*10)`)
2. Menghitung waktu tunggu, waktu tunggu berlaku apabila sesampainya di kota tujuan dan *time window* belum terbuka. Maka *cost* waktu tunggu diperoleh dari selisih sampainya dikota N dan batas awal *time window* (`cost := cost + selisih`).
3. Menghitung penalti, *cost* penalti diperoleh dari selisih sampainya dikota tujuan dan melebihi batas akhir *time window* (`penalti := penalti + selisih`).
4. *Cost* waktu pelayanan tergantung pada inputan awal.
5. *Cost* kembali kedepot tidak ditambah dengan waktu pelayanan tetapi terdapat *cost* penalti.
6. *Cost* keseluruhan diperoleh dari (waktu keberangkatan + *cost* perjalanan + *cost* waktu tunggu + *cost* penalti)
7. Nilai *fitness* diperoleh dari `1000 / (ind.Cost + ind.Penalti);`

4.2.12 Tambah Jam

Pada fungsi tambah jam ini digunakan untuk penambahan waktu pada fungsi tambah jam ini terdapat beberapa variabel yaitu `hh, mm` dan `JamAkhir`. Keluaran fungsi berupa `JamAkhir` bertipe `Tjam` yang merupakan penjumlahan waktu JamAwal ditambahkan menit tambahan. Pada `JamAwal` dan `JamAkhir` terdiri dari jam dan menit, sedangkan `menitTambahan` adalah waktu yang ditambahkan dalam satuan menit.

```
function TambahJam(JamAwal: TJam; menitTambahan: integer) :
TJam;
var
  hh, mm : integer;
  JamAkhir : TJam;
begin
  hh := menitTambahan div 60;
  mm := menitTambahan mod 60;

  JamAkhir.Jam := JamAwal.Jam + hh;
  JamAkhir.menit := JamAwal.menit + mm;

  if JamAkhir.menit > 59 then
    begin
     JamAkhir.Jam := JamAkhir.Jam + JamAkhir.menit div 60;
     JamAkhir.menit := JamAkhir.menit mod 60;
    end;
if JamAkhir.jam > 23 then
  begin
    JamAkhir.Jam := 0 + JamAkhir.menit div 24;
  end;
result := JamAkhir;
end;
```

Sourcecode 4.11 Proses Tambah Jam

Proses untuk ganti menit ke waktu selanjutnya diperoleh dari `JamAkhir.Jam := JamAwal.Jam + hh` jam akan berganti apabila menit sudah melewati `JamAkhir.menit > 59`, sedangkan untuk ganti ke jam ke waktu selanjutnya `JamAkhir.jam > 23`.

4.2.13 Selisih Jam

Pada proses selisih jam terdapat beberapa variabel yaitu `sjam, smenit, selisih,` dimana pada `smenit` masuknya menit untuk mengubah waktu dari menit ke jam dibatasi `mod 60` dan `sjam` dimana `sjam := jamMasuk.Jam - jamAwal.Jam + (smenit div 60).`

```
function selisihJam(jamAwal: TJam; jamMasuk: TJam):integer;
var
  sjam, smenit, selisih : integer;
begin
  smenit := (jamMasuk.menit - jamAwal.menit) mod 60;
  sjam := jamMasuk.Jam - jamAwal.Jam + (smenit div 60);
  selisih := (60*sjam) + smenit;

  result := selisih;
end;
```

Sourcecode 4.12 Proses Selisih Jam

4.2.14 *Generate* Data *Random Cost* perjalanan

Tahap awal dari pencarian rute terpendek dengan kendala *time window* adalah *generate* data secara *random* dan data-data tersebut meliputi biaya perjalanan dan *time window*. *Listing* program untuk melakukan *generate* biaya perjalanan seperti pada *sourcecode* 4.13.

```
// generating random cost
for i:=1 to N do
    for j:=i+1 to N do
    begin
      sgCost.Cells[i,j] := IntToStr (Random(9)+1);
    end;
for i:=1 to N do
  begin
    sgCost.Cells[i,i] := IntToStr (0);
    MatrixCost[i,i] := 0;
    for j:=i+1 to N do
   begin
      MatrixCost[i,j] := StrToInt(sgCost.Cells[i,j]);
      MatrixCost[j,i] := MatrixCost[i,j];
      sgCost.Cells[j,i] := sgCost.Cells[i,j];
    end;
  end;
```

Sourcecode 4.13 Proses *generate cost* perjalanan

Pada *Sourcecode* 4.13, akan dihasilkan akan dihasilkan *string grid* dengan jumlah kolom dan baris yang sesuai dengan jumlah kota yang dimasukkan dimana setiap kolom mengindikasikan biaya perjalanan setiap kota dengan kota lainya. Struktur data pada *generate* biaya perjalanan dalam bentuk *array* `TMatrixCost = array [1..MAXKOTA,1..MAXKOTA]`,

4.2.15 *Generate* Tabel *Time Window*

Generate tabel *time window.* Proses *generate time window* terdapat pada *sourcecode* 4.14.

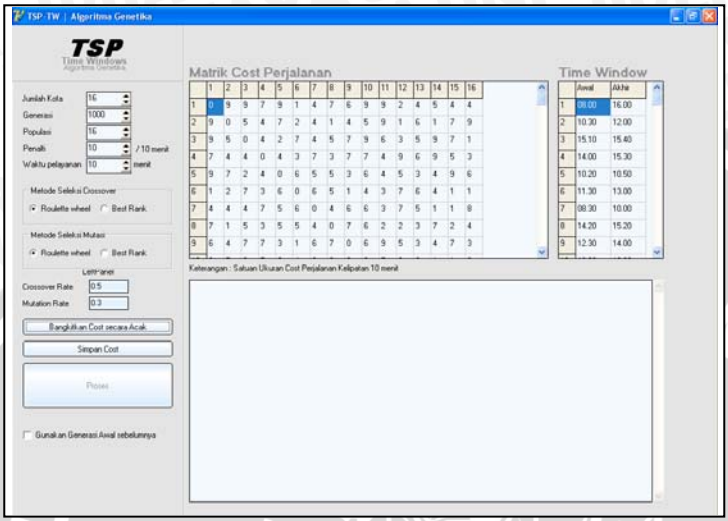
```
for i:=2 to N do
  begin
    j := Random(8)+8;
    k := (Random(3)+1)*30;

    TimeWindow[i].start.Jam := j;
    TimeWindow[i].start.menit := Random(5)*10;

    TimeWindow[i].finish := TambahJam (TimeWindow[i].start, k);

    if (TimeWindow[i].finish.Jam >= 16) then
    begin
      TimeWindow[i].finish.Jam := 16;
      TimeWindow[i].finish.menit := 0;
    end;
```

Sourcecode 4.14 Proses generate time window

Pada *sourcecode* 4.12 adalah proses *generate* tabel *time window* secara *random.* Dimana pada proses ini tipe datanya berbentuk `TTimeWindow = record` serta tipe data time window terdiri dari `start, finish : TJam;` dan `Tjam` merupakan tipe data jam yang berbentuk *record* dalam tipe data `Tjam` terdiri dari dua bagian yaitu jam dan menit. Semua data tabel time window hasil *generate* akan disimpan dalam `TTimeWindow = record`.

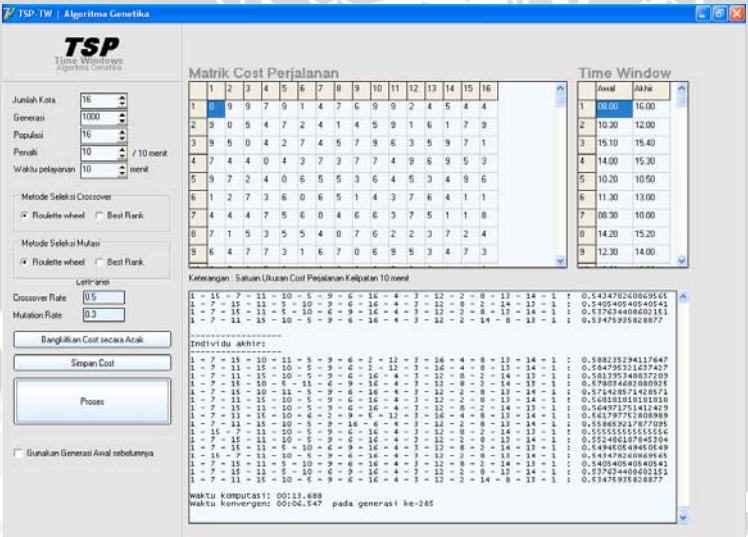
4.3 Penerapan Aplikasi

Aplikasi diterapkan dengan memasukkan data sesuai dengan keinginan *user*. Dalam kasus ini data yang dimasukkan adalah jumlah kota sebanyak 16, iterasi/generasi dilakukan sebanyak 1000 kali, waktu penalti sebesar 10 menit dan waktu pelayanan sebesar 10 menit dengan seleksi *crossover roullete wheell* dan mutasi *roullete wheell* serta *crossover rate* sebesar 0,5 diharapkan ada 8 individu mengalami mutasi dan *mutation rate* sebesar 0,3 diharapkan ada 3 individu mengalami mutasi. Untuk lebih jelasnya seperti pada Gambar 4.2.



Gambar 4.2 Tampilan Aplikasi Sebelum Proses Genetika

Apabila nilai parameter sudah disimpan dan program dijalankan, maka proses selanjutnya adalah proses algoritma genetika. Setelah melalui proses genetika maka akan diperoleh urutan perjalanan dan nilai *fitness* masing-masing kromosom, seperti pada Gambar 4.3.



Gambar 4.3 Tampilan Aplikasi Sesudah Proses Genetika

Untuk hasil lebih jelasnya, terlampir pada lampiran hasil uji coba aplikasi

4.4 Analisa Hasil

Perubahan nilai *fitness* dari inisialisasi awal hingga ditemukan *fitness* terbaik dapat dikarenakan model seleksi, *crossover rate*, *mutation rate*, banyaknya iterasi atau generasi yang digunakan.

Pada penelitian ini mencoba melakukan analisa terhadap nilai *fitness* yang dihasilkan dengan menggunakan model seleksi yang berbeda yaitu model seleksi *roullete wheel* dan *best rank*. Model seleksi *roullete wheel* dimana individu yang memiliki nilai *fitness* besar memiliki peluang lebih besar dalam pemilihan secara acak dan model seleksi *best rank* dimana individu diurutkan dari yang memiliki *fitness* besar ke kecil.

Uji coba dilakukan untuk mengetahui adanya perubahan nilai *fitness* dan iterasi mencapai konvergen. Berikut hasil dari uji coba nilai *fitness* dengan memberikan besar ukuran individu yang berbeda dalam satu populasi:

Tabel 4.2 Hasil Uji Coba Bedasarkan Nilai *Fitness* dengan Metode Seleksi Berbeda.

Ukuran individu	Uji Coba 1	Uji Coba 2	Uji Coba 3	Uji Coba 4
8	0.562138596	0.525301886	0.542829148	0.489226976
10	0.659090317	0.644898851	0.562254878	0.403954085
12	0.630412422	0.601070863	0.674968397	0.581766116
14	0.642239068	0.448052813	0.677754351	0.406265175
16	0.707239773	0.663221962	0.601910503	0.702301401

Keterangan :

Uji coba 1 : Dengan menggunkan seleksi *roullete wheel* untuk proses *crossover* dan seleksi *roullete wheel* untuk proses mutasi.

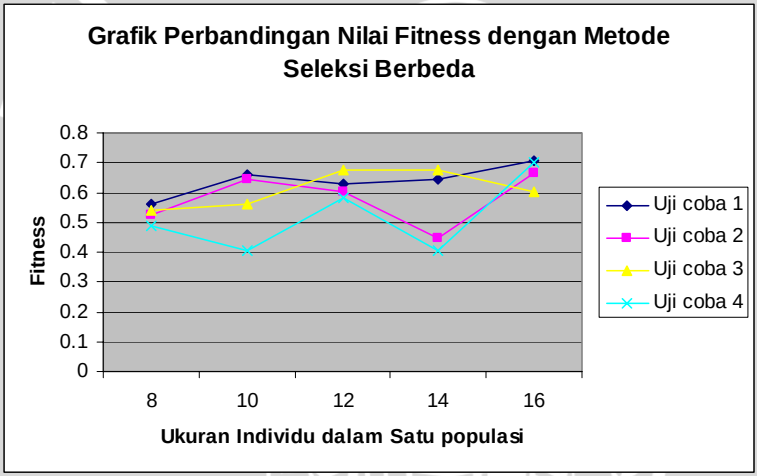
Uji coba 2 : Dengan menggunkan seleksi *roullete wheel* untuk proses *crossover* dan seleksi *best rank* untuk proses mutasi.

Uji coba 3 : Dengan menggunkan seleksi *best rank* untuk proses *crossover* dan seleksi *roullete wheel* untuk proses mutasi.

Uji coba 4 : Dengan menggunkan seleksi *best rank* untuk proses *crossover* dan seleksi seleksi *best rank* untuk proses mutasi.

Untuk lebih jelasnya hasil dari perbandingan nilai *fitness* dengan menggunakan metode seleksi yang berbeda terlampir pada lampiran perbandingan nilai *fitness*.

Pada Tabel 4.2 hasil uji dapat diketahui perbandingan nilai *fitness* dengan memberikan ukuran besar individu dalam satu populasi yang berbeda berdasarkan metode seleksi yang digunakan, untuk lebih jelasnya dapat dilihat pada Gambar 4.4.



Gambar 4.4 Grafik Perbandingan Nilai *Fitness* Berdasarkan Metode Seleksi Untuk Proses *Crossover* dan Mutasi

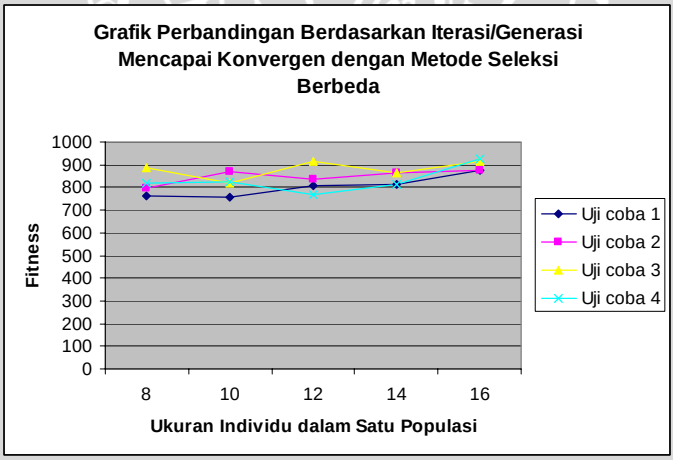
Dari uji coba dengan menggunakan metode seleksi yang berbeda untuk proses *crossover* dan mutasi, nilai *fitness* yang diperoleh menunjukkan tidak adanya kestabilan nilai. Nilai yang dihasilkan dari uji coba terlihat berbeda jauh, nilai *fitness* paling besar terdapat pada individu ukuran 16 dengan seleksi *roullete wheel* pada proses *crossover* dan seleksi *roullete wheel* untuk proses mutasi. Uji coba selanjutnya adalah dengan membandingkan metode seleksi tersebut terhadap iterasi mencapai konvergen seperti pada Tabel 4.3.

Tabel 4.3 Hasil Uji Coba Berdasarkan Iterasi Mencapai Konvergen dengan Metode Seleksi Berbeda.

Ukuran individu	Uji Coba 1	Uji Coba 2	Uji Coba 3	Uji Coba 4
8	763.6	795.8	886.6	817.4
10	759.4	870.8	820.6	825
12	806.4	835.2	917.8	765.8
14	815	862.2	863.2	815
16	873.8	874.6	916.8	926.6

Setelah dilakukan uji coba untuk mengetahui rata-rata iterasi mencapai konvergen hasil yang diperoleh individu mencapai konvergen dengan metode seleksi yang berbeda terdapat pada iterasi ke-759 sampai iterasi ke-926 dari 1000 kali iterasi yang dilakukan. Untuk lebih jelasnya terlampir pada lampiran iterasi mencapai konvergen berdasarkan.

Pada Tabel 4.3 hasil dari uji coba dapat diketahui perbandingan iterasi mencapai konvergen dengan menggunakan metode seleksi yang berbeda untuk proses *crossover* dan mutasi, untuk lebih jelasnya seperti pada Gambar 4.5.



Gambar 4.5 Grafik Perbandingan Iterasi mencapai Konvergen Berdasarkan Metode Seleksi Untuk Proses *Crossover* dan Mutasi

Dari uji coba dengan menggunakan metode seleksi yang berbeda untuk proses *crossover* dan mutasi, iterasi mencapai konvergen

antara iterasi ke-759 sampai ke-926 dari 1000 kali iterasi yang dilakukan.

Selanjutnya dilakukan uji coba terhadap *crossover rate* dan *mutation rate,* tujuan dilakukan uji coba ini adalah untuk mengetahui pada *crossover rate* dan *mutation rate* keberapa diperoleh nilai *fitness* yang paling baik. Pada uji coba kali ini dilakukan dengan 1000 kali iterasi dan ukuran individu sebesar 10, nilai penalti 10 menit dan waktu pelayanan sebesar 10 menit.

Uji Coba pertama pada *crossover rate* 0.2 dan *mutation rate* 0.2-0.6, seperti pada Tabel 4.4.

Tabel 4.4 Uji Coba *Crossover rate* 0.2 dan *mutation rate* 0.2 s/d 0.6

Metode\Mutation rate	0.2	0.3	0.4	0.5	0.6
Uji Coba 1	0.60484	0.55545	0.68534	0.71028	0.739246
Uji Coba 2	0.55652	0.63259	0.60704	0.63231	0.699145
Uji Coba 3	0.53479	0.58642	0.61015	0.59712	0.598302
Uji Coba 4	0.56362	0.58912	0.58815	0.60702	0.670217

Keterangan :

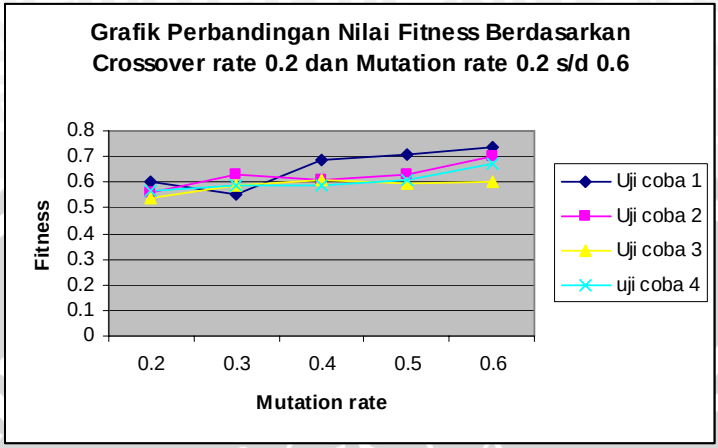
Uji coba 1 : Dengan menggunkan seleksi *roullete wheel* untuk proses *crossover* dan seleksi *roullete wheel* untuk proses mutasi.

Uji coba 2 : Dengan menggunkan seleksi *roullete wheel* untuk proses *crossover* dan seleksi *best rank* untuk proses mutasi.

Uji coba 3 : Dengan menggunkan seleksi *best rank* untuk proses *crossover* dan seleksi *roullete wheel* untuk proses mutasi.

Uji coba 4 : Dengan menggunkan seleksi *best rank* untuk proses *crossover* dan seleksi seleksi *best rank* untuk proses mutasi.

Dari Tabel 4.4 dapat diketahui pada *crossover rate* 0.2 dan *mutation rate* 0.6 diperoleh nilai *fitness* yang paling besar, untuk lebih jelasnya pada seperti pada Gambar 4.6.



Gambar 4.6 Grafik Perbandingan Nilai *Fitness* Berdasarkan *Crossover rate* 0.2 dan *mutation rate* 0.2 s/d 0.6

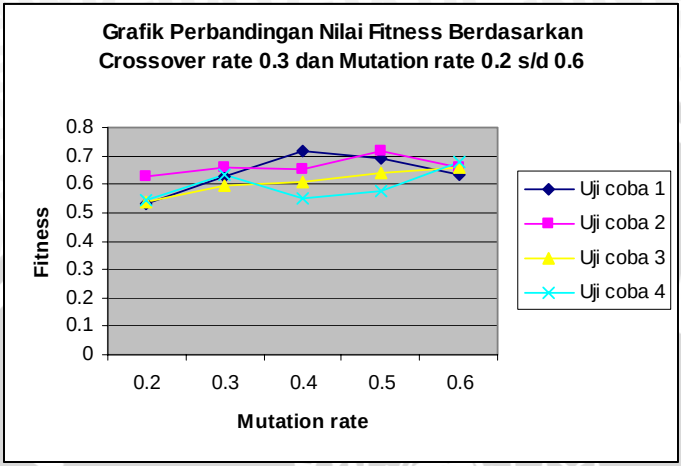
Uji Coba ke-2 pada *crossover rate* 0.3 dan *mutation rate* 0.2-0.6, seperti pada Tabel 4.5.

Tabel 4.5 Uji Coba *Crossover rate* 0.3 dan *mutation rate* 0.2 s/d 0.6

Metode\Mutation rate	0.2	0.3	0.4	0.5	0.6
Uji Coba 1	0.53422	0.62514	0.71592	0.69423	0.6365
Uji Coba 2	0.62633	0.65985	0.65562	0.71425	0.66216
Uji Coba 3	0.53921	0.59277	0.60938	0.63851	0.66043
Uji Coba 4	0.54607	0.63322	0.55114	0.57869	0.67625

Pada Gambar 4.7 dapat dilihat perbandingan nilai *fitness* pada *crosssover rate* 0.2 dan *mutation rate* 0.2 s/d 0.6.

Dari Tabel 4.5 dapat dikethui nilai *fitness* terbesar terdapat pada metode seleksi *roullete wheel* dan mutasi *roullete wheel* dengan *crossover rate* 0.3 dan *mutation rate* 0.4, untuk lebih jelasnya perbandingan nilai *fitness* dapat dilihat pada Gambar 4.7.



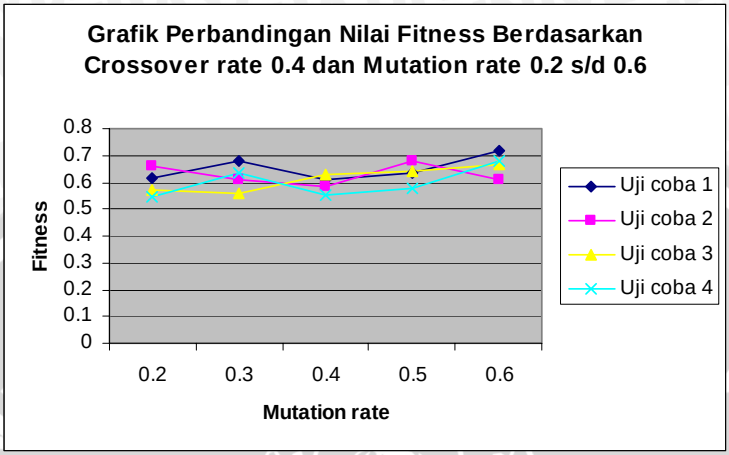
Gambar 4.7 Grafik Perbandingan Nilai *Fitness* Berdasarkan *Crossover rate* 0.3 dan *mutation rate* 0.2 s/d 0.6

Uji Coba ke-3 pada *crossover rate* 0.4 dan *mutation rate* 0.2-0.6, seperti pada Tabel 4.6

Tabel 4.6 Uji Coba *Crossover rate* 0.4 dan *mutation rate* 0.2 s/d 0.6

Metode\Mutation rate	0.2	0.3	0.4	0.5	0.6
Uji Coba 1	0.61512	0.67647	0.60664	0.63695	0.720578
Uji Coba 2	0.66079	0.60968	0.58695	0.68039	0.611959
Uji Coba 3	0.56919	0.5583	0.63	0.6402	0.665758
Uji Coba 4	0.54607	0.63322	0.55114	0.57869	0.676251

Pada Uji coba ke-3 pada Tabel 4.6 dapat dilihat bahwa pada *crossover rate* 0.4 dan *mutation rate* 0.6 dengan menggunakan metode seleksi *crossover roullete wheel* dan mutasi *roullete wheel* diperoleh nilai *fitness* yang paling besar. Pada Gambar 4.8 dapat dilihat perbandingan nilai *fitness* pada *crosssover rate* 0.4 dan *mutation rate* 0.2 s/d 0.6.



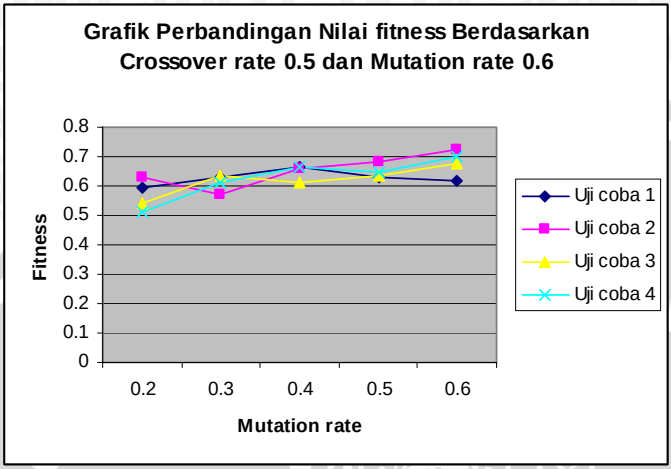
Gambar 4.8 Grafik Perbandingan Nilai *Fitness* Berdasarkan *Crossover rate* 0.4 dan *mutation rate* 0.2 s/d 0.6

Uji Coba ke-4 pada *crossover rate* 0.3 dan *mutation rate* 0.2-0.6, seperti pada Tabel 4.7.

Tabel 4.7 Uji Coba *Crossover rate* 0.5 dan *mutation rate* 0.2 s/d 0.6

Metode\Mutation rate	0.2	0.3	0.4	0.5	0.6
Uji Coba 1	0.5914	0.62668	0.66182	0.62807	0.61497
Uji Coba 2	0.62869	0.57268	0.66118	0.68361	0.72294
Uji Coba 3	0.54254	0.63697	0.60897	0.63791	0.67876
Uji Coba 4	0.51354	0.61421	0.66739	0.64436	0.70263

Pada Uji coba ke-4 pada Tabel 4.7 dapat dilihat bahwa pada *crossover rate* 0.5 dan *mutation rate* 0.6 dengan menggunakan metode seleksi *crossover roullete wheel* dan mutasi *best rank* diperoleh nilai *fitness* yang paling besar. Pada Gambar 4.9 dapat dilihat perbandingan nilai *fitness* pada *crosssover rate* 0.5 dan *mutation rate* 0.2 s/d 0.6.



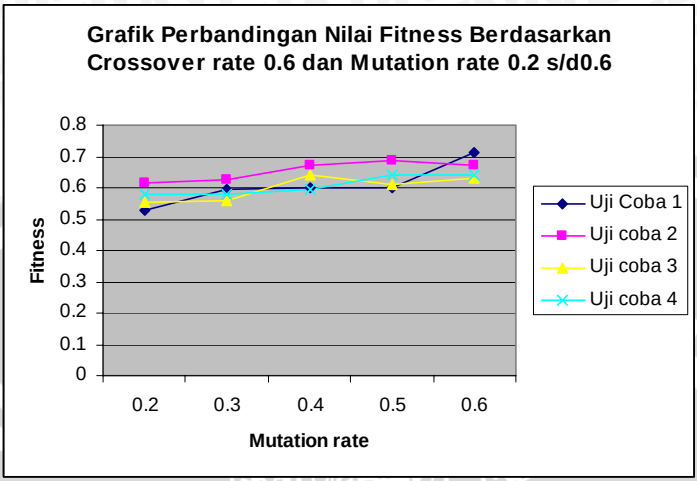
Gambar 4.9 Grafik Perbandingan Nilai *Fitness* Berdasarkan *Crossover rate* 0.5 dan *mutation rate* 0.2 s/d 0.6

Uji Coba ke-5 pada *crossover rate* 0.6 dan *mutation rate* 0.2-0.6, seperti pada Tabel 4.8.

Tabel 4.8 Uji Coba *Crossover rate* 0.6 dan *mutation rate* 0.2 s/d 0.6

Metode\Mutation rate	0.2	0.3	0.4	0.5	0.6
Uji Coba 1	0.52854	0.59315	0.60165	0.59983	0.715296
Uji Coba 2	0.61577	0.62578	0.67164	0.6878	0.670761
Uji Coba 3	0.55286	0.56114	0.63968	0.6103	0.629853
Uji Coba 4	0.57859	0.57705	0.59712	0.64231	0.643201

Pada Uji coba ke-5 pada Tabel 4.7 dapat dilihat bahwa pada *crossover rate* 0.5 dan *mutation rate* 0.6 dengan menggunakan metode seleksi *crossover roullete wheel* dan mutasi *roullete wheel* diperoleh nilai *fitness* yang paling besar. Pada Gambar 4.10 dapat dilihat perbandingan nilai *fitness* pada *crosssover rate* 0.5 dan *mutation rate* 0.2 s/d 0.6.



Gambar 4.10 Grafik Perbandingan Nilai *Fitness* Berdasarkan *Crossover rate* 0.6 dan *mutation rate* 0.2 s/d 0.6

Berdasarkan uji coba yang dilakukan beberapa kali dengan metode seleksi yang berbeda dan menggunakan *crossover rate* dan *mutation rate* yang berbeda menunjukkan bahwa dengan menggunakan metode seleksi *crossover roullete wheel* dan mutasi *roullete wheel* diperoleh nilai *fitness* yang lebih baik dibandingkan dengan metode seleksi yang lain dan nilai *fitness* yang paling tinggi diperoleh pada *crossover rate* 0.2 dan *mutation rate* 0.6. Nilai *fitness* mencapai konvergen antara iterasi ke-759 sampai ke-926 dari 1000 kali iterasi yang dilakukan.

BAB V
KESIMPULAN DAN SARAN

5.1 Kesimpulan

Dari hasil uji coba yang telah dilakukan, maka dapat diambil kesimpulan antara lain :

1. Algoritma genetika dapat digunakan sebagai alternatif solusi untuk masalah pencarian rute optimal dengan kendala *time window*.
2. Setelah melakukan pengujian dapat disimpulkan bahwa dengan menggunakan seleksi *roullete wheel* untuk proses *crossover* dan seleksi *roullete wheel* untuk proses mutasi diperoleh nilai *fitness* lebih baik dari metode seleksi yang lain.
3. Nilai *fitness* mencapai konvergen antara iterasi ke-759 sampai ke-926 dari 1000 kali iterasi yang dilakukan.

5.2 Saran

Aplikasi yang dibangun masih belum sempurna. Hal yang dapat bermanfaat untuk mengembangkan aplikasi ini adalah :

1. Aplikasi pengantaran barang dengan kendala *time window* ini dapat dikembangkan menjadi sebuah sistem informasi yang memiliki kemampuan menentukan rute perjalanan pengantaran barang pada sebuah perusahaan jasa pengantar barang dengan menggunakan data-data yang sebenarnya.
2. Setelah dilakukan uji coba terdapat beberapa pergerakan data yang tidak seragam, hal ini mungkin bisa diperbaiki dengan pemilihan operator *crossover* dan mutasi yang berbeda.

DAFTAR PUSTAKA

Bar-Yehuda R, Even Guy, shahar S. *On approximating a geometric prize-collecting traveling salesman problem with time windows.*

Gen, M. Dan Cheng, R. 1997. *Genetic Algoritm and Enginering Design*. Ashikaga Institute of Technology Ashikaga, Japan. A wiley-Interscience Publication. John Wiley & Sons, Inc. New York.

Goldberg, D. E. 1989. *Genetic Algorithms in Search, Optimization, and Machine Learning.* Addison Wesley Publishing Company. USA.

Hanawati.A, Thiang dan Eleazar, 2003 *Pencarian Rute Optimum Menggunakan Algoritma Genetik*. http://puslit.petra.ac.id/~puslit/journals/articles.php?PublishedID=ELK02020205. Tanggal akses 2 November 2007

Kusumadewi, Sri. 2003. *Artificial Intelligence (Teknik dan Aplikasinya*. Graha Ilmu. Yogyakarta.

Obitko, M. 1998. *Introduction to Genetic Algorithm*. http://cs.felk.cvut.cz/~xobitko/ga/. Tanggal akses : 8 Februari 2007

Rusdiansyah, A 2000. Perencanaan Rute dan Jadwal Kendaraan untuk Transportasi bagi Penyandang Cacat. http://ml.ryu.titech.ac.jp/~indonesia/tokodai/zoa/pdf/zoaarusdian.pdf. Tanggal akses 5 Desember 2007

Subekti, Irfan. 2002. Sistem Berbasis Pengetahuan. ITS. Surabaya.

Sutapa. I, N, Widyadana. I, G, A dan Christine, 2004. *Studi tentang Traveling Salesman Problem dan Vehicle Routing Problem dengan Time Windows*, http://www.petra.ac.id/~puslit/journals/request.php?PublishedID=IND03050202. Tanggal akses : 2 November 2007

Suyanto. 2005. *Algoritma Genetik dalam Matlab*, Andi Offset, Yogyakarta .

Syamsudin, A. 2004. *Pengenalan Algoritma Genetik*. http://www.ilmu computer.com. tanggal akses : 2 November 2007.

Lampiran 1
Hasil Uji Coba Aplikasi

Kota : 16
Iterasi : 1000
Populasi : 16 Individu
Penalti : 10 menit
Waktu pelayanan : 10 Menit
Crossover rate : 0,5 diharapkan ada 8 individu terlibat
Mutation rate : 0,3 diharapkan ada 3 individu terlibat

Individu awal:
1 - 12 - 15 - 7 - 10 - 4 - 14 - 13 - 8 - 11 - 3 - 16 - 2 - 6 - 5 - 9 - 1 : 0.221729490022173
1 - 5 - 7 - 8 - 14 - 2 - 16 - 4 - 15 - 9 - 11 - 3 - 12 - 10 - 6 - 13 - 1 : 0.211416490486258
1 - 2 - 7 - 13 - 8 - 9 - 15 - 6 - 14 - 12 - 16 - 5 - 10 - 4 - 11 - 3 - 1 : 0.193423597678917
1 - 6 - 2 - 12 - 5 - 14 - 9 - 15 - 8 - 10 - 7 - 16 - 13 - 4 - 3 - 11 - 1 : 0.179533213644524
1 - 2 - 11 - 12 - 14 - 9 - 3 - 5 - 16 - 10 - 7 - 6 - 4 - 13 - 15 - 8 - 1 : 0.177619893428064
1 - 5 - 9 - 15 - 3 - 12 - 7 - 2 - 14 - 8 - 11 - 4 - 16 - 6 - 13 - 10 - 1 : 0.175131348511384
1 - 9 - 3 - 13 - 11 - 8 - 6 - 2 - 16 - 10 - 5 - 4 - 12 - 15 - 7 - 14 - 1 : 0.171526586620926
1 - 12 - 16 - 4 - 15 - 7 - 13 - 14 - 9 - 8 - 5 - 2 - 10 - 11 - 6 - 3 - 1 : 0.171232876712329
1 - 2 - 3 - 7 - 11 - 10 - 16 - 5 - 14 - 15 - 13 - 8 - 6 - 12 - 4 - 9 - 1 : 0.161550888529887
1 - 2 - 3 - 11 - 14 - 7 - 10 - 8 - 16 - 15 - 9 - 6 - 4 - 5 - 13 - 12 - 1 : 0.159489633173844
1 - 8 - 14 - 13 - 16 - 2 - 11 - 5 - 12 - 9 - 6 - 10 - 4 - 7 - 3 - 15 - 1 : 0.157977883096367
1 - 4 - 3 - 5 - 15 - 6 - 10 - 11 - 12 - 2 - 9 - 8 - 16 - 7 - 14 - 13 - 1 : 0.148809523809524
1 - 4 - 15 - 8 - 6 - 7 - 2 - 11 - 12 - 16 - 14 - 10 - 3 - 5 - 13 - 9 - 1 : 0.145348837209302
1 - 4 - 10 - 15 - 12 - 5 - 2 - 11 - 9 - 6 - 14 - 13 - 8 - 16 - 7 - 3 - 1 : 0.141242937853107
1 - 13 - 15 - 12 - 6 - 11 - 5 - 9 - 10 - 7 - 2 - 4 - 8 - 16 - 3 - 14 - 1 : 0.140845070422535
1 - 16 - 12 - 4 - 2 - 11 - 9 - 10 - 8 - 6 - 3 - 15 - 7 - 13 - 14 - 5 - 1 : 0.136054421768707

Individu akhir:
---
1 - 7 - 15 - 10 - 11 - 5 - 9 - 6 - 2 - 12 - 3 - 16 - 4 - 8 - 13 - 14 - 1 : 0.588235294117647
1 - 7 - 11 - 15 - 10 - 5 - 9 - 6 - 2 - 12 - 3 - 16 - 4 - 8 - 13 - 14 - 1 : 0.584795321637427
1 - 7 - 11 - 15 - 10 - 5 - 9 - 6 - 16 - 4 - 3 - 12 - 8 - 2 - 14 - 13 - 1 : 0.581395348837209
1 - 7 - 15 - 10 - 5 - 11 - 6 - 9 - 16 - 4 - 3 - 12 - 8 - 2 - 14 - 13 - 1 : 0.578034682080925
1 - 7 - 15 - 10 - 11 - 5 - 9 - 6 - 16 - 4 - 3 - 12 - 2 - 8 - 13 - 14 - 1 : 0.571428571428571
1 - 7 - 11 - 15 - 10 - 5 - 9 - 6 - 16 - 4 - 3 - 12 - 2 - 8 - 13 - 14 - 1 : 0.568181818181818
1 - 7 - 15 - 11 - 10 - 5 - 9 - 6 - 16 - 4 - 3 - 12 - 8 - 2 - 14 - 13 - 1 : 0.564971751412429
1 - 7 - 11 - 15 - 10 - 6 - 2 - 9 - 5 - 12 - 3 - 16 - 4 - 8 - 13 - 14 - 1 : 0.561797752808989
1 - 7 - 11 - 15 - 10 - 5 - 9 - 16 - 6 - 4 - 3 - 12 - 2 - 8 - 13 - 14 - 1 : 0.558659217877095
1 - 15 - 7 - 11 - 10 - 5 - 9 - 6 - 16 - 4 - 3 - 12 - 8 - 2 - 14 - 13 - 1 : 0.555555555555556
1 - 7 - 15 - 11 - 10 - 5 - 9 - 6 - 16 - 4 - 3 - 12 - 2 - 8 - 13 - 14 - 1 : 0.552486187845304
1 - 7 - 15 - 11 - 5 - 10 - 6 - 9 - 16 - 4 - 3 - 12 - 8 - 2 - 14 - 13 - 1 : 0.549450549450549
1 - 15 - 7 - 11 - 10 - 5 - 9 - 6 - 16 - 4 - 3 - 12 - 2 - 8 - 13 - 14 - 1 : 0.543478260869565
1 - 7 - 15 - 11 - 5 - 10 - 9 - 6 - 16 - 4 - 3 - 12 - 8 - 2 - 14 - 13 - 1 : 0.540540540540541
1 - 7 - 15 - 11 - 5 - 10 - 6 - 9 - 16 - 4 - 3 - 12 - 2 - 8 - 13 - 14 - 1 : 0.537634408602151
1 - 7 - 11 - 15 - 10 - 5 - 9 - 6 - 16 - 4 - 3 - 12 - 2 - 14 - 8 - 13 - 1 : 0.53475935828877
Waktu komputasi: 00:13.688
Waktu konvergen: 00:06.547 pada generasi ke-285

Lampiran 2

Perbandinagn Nilai *Fitness* dengan Metode Seleksi Berbeda

Individu Ukuran 8

Uji Coba	Uji Coba 1	Uji Coba 2	Uji Coba 3	Uji Coba 4
1	0.892857143	0.595238095	0.490196078	0.462962963
2	0.490196078	0.543478261	0.621118012	0.262467192
3	0.505050505	0.666666667	0.448430493	0.662251656
4	0.24691358	0.558659218	0.649350649	0.653594771
5	0.675675676	0.262467192	0.505050505	0.4048583
Rata-rata	0.562138596	0.525301886	0.542829148	0.489226976

Individu Ukuran 10

Uji Coba	Uji Coba 1	Uji Coba 2	Uji Coba 3	Uji Coba 4
1	0.8	0.641025641	0.632911392	0.248756219
2	0.8	0.64516129	0.689655172	0.265957447
3	0.46728972	0.465116279	0.257069409	0.584795322
4	0.552486188	0.884955752	0.666666667	0.653594771
5	0.675675676	0.588235294	0.564971751	0.266666667
Rata-rata	0.659090317	0.644898851	0.562254878	0.403954085

Individu Ukuran 12

Uji Coba	Uji Coba 1	Uji Coba 2	Uji Coba 3	Uji Coba 4
1	0.649350649	0.529100529	0.649350649	0.552486188
2	1	0.684931507	0.598802395	0.793650794
3	0.268096515	0.641025641	0.680272109	0.617283951
4	0.613496933	0.540540541	0.854700855	0.689655172
5	0.621118012	0.609756098	0.591715976	0.255754476
Rata-rata	0.630412422	0.601070863	0.674968397	0.581766116

Individu Ukuran 14

Uji Coba	Uji Coba 1	Uji Coba 2	Uji Coba 3	Uji Coba 4
1	0.595238095	0.497512438	0.735294118	0.625
2	0.595238095	0.602409639	0.609756098	0.265251989
3	0.689655172	0.609756098	0.819672131	0.267379679
4	0.584795322	0.279329609	0.561797753	0.256410256

5	0.746268657	0.251256281	0.662251656	0.617283951
Rata-rata	0.642239068	0.448052813	0.677754351	0.406265175

Individu Ukuran 16

Uji Coba	Uji Coba 1	Uji Coba 2	Uji Coba 3	Uji Coba 4
1	0.657894737	0.571428571	0.78125	0.735294118
2	0.769230769	0.862068966	0.5	0.666666667
3	0.675675676	0.78125	0.689655172	0.641025641
4	0.540540541	0.819672131	0.483091787	0.762068966
5	0.892857143	0.281690141	0.555555556	0.706451613
Rata-rata	0.707239773	0.663221962	0.601910503	0.702301401

Keterangan :

Uji coba 1 : Dengan menggunkan seleksi *roullete wheel* untuk proses *crossover* dan seleksi *roullete wheel* untuk proses mutasi.

Uji coba 2 : Dengan menggunkan seleksi *roullete wheel* untuk proses *crossover* dan seleksi *best rank* untuk proses mutasi.

Uji coba 3 : Dengan menggunkan seleksi *best rank* untuk proses *crossover* dan seleksi *roullete wheel* untuk proses mutasi.

Uji coba 4 : Dengan menggunkan seleksi *best rank* untuk proses *crossover* dan seleksi seleksi *best rank* untuk proses mutasi.

Lampiran 3
Perbandingan Iterasi Mencapai Konvergen dengan Metode Seleksi Berbeda

Individu Ukuran 8

Uji Coba	Uji Coba 1	Uji Coba 2	Uji Coba 3	Uji Coba 4
1	753	596	931	982
2	726	794	875	848
3	928	990	829	905
4	467	632	962	514
5	944	967	836	838
Rata-rata	763.6	795.8	886.6	817.4

Individu Ukuran 10

Uji Coba	Uji Coba 1	Uji Coba 2	Uji Coba 3	Uji Coba 4
1	957	778	921	698
2	986	754	397	931
3	544	987	924	526
4	970	979	895	986
5	340	856	966	984
Rata-rata	759.4	870.8	820.6	825

Individu Ukuran 12

Uji Coba	Uji Coba 1	Uji Coba 2	Uji Coba 3	Uji Coba 4
1	862	917	760	909
2	912	660	901	702
3	785	935	997	807
4	474	873	971	741
5	999	791	960	670
Rata-rata	806.4	835.2	917.8	765.8

Individu Ukuran 14

Uji Coba	Uji Coba 1	Uji Coba 2	Uji Coba 3	Uji Coba 4
1	892	949	964	795
2	578	900	957	770
3	961	811	918	991

4	995	996	873	825
5	649	655	469	694
Rata-rata	815	862.2	836.2	815

Individu Ukuran 16

Uji Coba	Uji Coba 1	Uji Coba 2	Uji Coba 3	Uji Coba 4
1	550	945	940	997
2	958	743	951	860
3	981	969	924	986
4	881	992	825	819
5	999	724	944	971
Rata-rata	873.8	874.6	916.8	926.6

Keterangan :

Uji coba 1 : Dengan menggunkan seleksi *roullete wheel* untuk proses *crossover* dan seleksi *roullete wheel* untuk proses mutasi.

Uji coba 2 : Dengan menggunkan seleksi *roullete wheel* untuk proses *crossover* dan seleksi *best rank* untuk proses mutasi.

Uji coba 3 : Dengan menggunkan seleksi *best rank* untuk proses *crossover* dan seleksi *roullete wheel* untuk proses mutasi.

Uji coba 4 : Dengan menggunkan seleksi *best rank* untuk proses *crossover* dan seleksi seleksi *best rank* untuk proses mutasi.