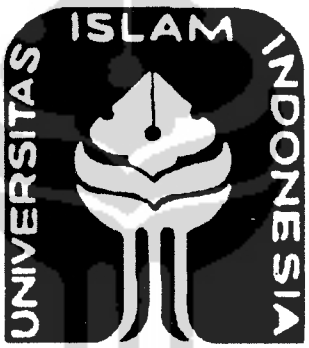
PEMBUATAN PERANGKAT LUNAK SHORGA UNTUK MENENTUKAN JALUR TERPENDEK ANTAR KOTA MENGGUNAKAN ALGORITMA GENETIKA

TUGAS AKHIR

Diajukan sebagai Salah Satu Syarat
Untuk Memperoleh Gelar Sarjana
Jurusan Teknik Informatika



Disusun Oleh :

Nama : Fajar Saptono
NIM : 03 523 218

JURUSAN TEKNIK INFORMATIKA
FAKULTAS TEKNOLOGI INDUSTRI
UNIVERSITAS ISLAM INDONESIA
YOGYAKARTA

2007

LEMBAR PENGESAHAN PEMBIMBING

PEMBUATAN PERANGKAT LUNAK SHORGA UNTUK MENENTUKAN JALUR TERPENDEK ANTAR KOTA MENGGUNAKAN ALGORITMA GENETIKA

TUGAS AKHIR

Disusun Oleh :

Nama : Fajar Saptono

NIM : 03 523 218

Yogyakarta, 24 Agustus 2007

Pembimbing,

Taufiq Hidayat, ST, MCS.

LEMBAR PERNYATAAN KEASLIAN
HASIL TUGAS AKHIR

Saya yang bertandatangan di bawah ini,

Nama : Fajar Saptono
NIM : 03 523 218

Menyatakan bahwa seluruh komponen dan isi dalam Laporan Tugas Akhir ini adalah hasil karya saya sendiri. Apabila di kemudian hari terbukti bahwa ada beberapa bagian dari karya ini adalah bukan hasil karya saya sendiri, maka saya akan siap menanggung resiko dan konsekuensi apapun.

Demikian pernyataan ini saya buat, semoga dapat dipergunakan sebagaimana mestinya.

Yogyakarta, 24 Agustus 2007

Fajar Saptono

ISLAM
UNIVERSITAS
INDONESIA

KATA PENGANTAR

بسم الله الرحمن الرحيم

Assalamu'alaikum Wr.Wb

Alhamdulillah, segala puji bagi Allah SWT atas segala rahmat, hidayah dan inayah-Nya, sehingga penulisan laporan tugas akhir yang berjudul **Pembuatan Perangkat Lunak SHORGA untuk Menentukan Jalur Terpendek Antar Kota Menggunakan Algoritma Genetika** dapat penulis selesaikan dengan baik. Sholawat serta salam juga penulis persembahkan kepada Nabi Muhammad SAW beserta para sahabat.

Laporan tugas akhir ini disusun sebagai salah satu syarat guna memperoleh gelar Sarjana Teknik Informatika pada Universitas Islam Indonesia. Dan juga sebagai sarana untuk mempraktekkan secara langsung ilmu dan teori yang telah diperoleh selama menjalani masa studi di Jurusan Teknik Informatika Fakultas Teknologi Industri Universitas Islam Indonesia.

Penyusunan laporan tugas akhir ini tidak lepas dari bimbingan, dukungan dan bantuan baik materiil maupun spirituil dari berbagai pihak. Oleh karena itu dalam kesempatan ini dengan segala kerendahan hati, penulis ingin menyampaikan ucapan terima kasih yang sebesar-besarnya kepada:

a. Bapak dan Mamak atas segala do'a, pengorbanan, kasih sayang, serta dorongan baik spirituil maupun materiil sehingga penulis bisa melakukan yang terbaik.

b. Bapak Edy Suandi Hamid, selaku Rektor Universitas Islam Indonesia dan seluruh jajaran Rektorat Universitas Islam Indonesia.

c. Bapak Fathul Wahid, ST., M.Sc, selaku Dekan Fakultas Teknologi Industri Universitas Islam Indonesia. Terima kasih atas masukan dan motivasi selama ini.

d. Bapak Yudi Prayudi, S.Si., M.Kom, selaku Ketua Jurusan Teknik Informatika. Terima kasih atas kemudahan dan dukungan yang telah diberikan.

e. Bapak Taufiq Hidayat, ST, MCS selaku dosen pembimbing dan Kepala Lab. PIT yang telah memberikan pengarahan, bimbingan, motivasi serta masukan selama pelaksanaan tugas akhir dan penulisan laporan.

f. Dosen-dosen Jurusan Teknik Informatika. Terima kasih atas semua ilmu pengetahuan dan motivasi serta bantuannya.

g. Teman-teman PIT'ers (Ienx, Nyu2n, Ice-tea, Endro, Boo), lanjutin terus penelitiannya. Teman-teman Lab Informatika Terpadu & Mas Andan, dengan kebersamaannya.

h. Seluruh keluarga besarku: Mas Adi & Mbak Yanti (Adam, Ivan), Mas Wawan & Mbak Vetty (Wibi, Sandy, Windy), Mbak Ayi & Mas Taufik (Sakhi), Mas Heru & Mbak Vira, Mbak Yani, yang telah memberikan semangat, dorongan dan dukungan penuh.

i. Sahabat ter'hebat'ku, Ika Kurnianti Ayuningtias (iyut ☺), terimakasih atas semuanya: senyuman, perhatian, nasihat dan pelajaran yang terindah untuk hidupku.

j. Semua sahabatku yang belum tersebut di manapun mereka berada. Terima kasih atas do'a dan dukungannya.

k. Semua pihak yang telah memberikan bantuan dan dorongan.

Semoga Allah SWT melimpahkan rahmat dan hidayahnya kepada semua pihak yang telah membantu terselesaikannya penulisan laporan tugas akhir ini.

Penulis menyadari bahwa dalam penyusunan laporan tugas akhir ini masih jauh dari sempurna, maka dengan segala keterbukaan penulis mengharapkan segala kritik dan saran yang membantu proses penyempurnaan di masa mendatang.

Akhir kata semoga laporan ini dapat bermanfaat bagi penulis dan semua pembaca.

Wassalamu'alaikum Wr.Wb.

Yogyakarta, 24 Agustus 2007

Fajar Saptono

SARI

Sebuah perjalanan terkadang membutuhkan jalur/rute yang terpendek. Biasanya jalur terpendek tersebut didapatkan dengan cara menghitung waktu yang ditempuh, ataupun berdasarkan jarak dari kota asal ke kota tujuan. Semakin banyak alternatif jalur ke kota tujuan, semakin rumit cara untuk menghitung jalur terpendek. Untuk itu diperlukan sebuah mekanisme yang handal untuk dapat menentukan jalur terpendek dari kota sumber ke kota tujuan. Penerapan metode AI dalam perhitungan jalur terpendek merupakan salah satu solusi untuk dapat menyelesaikan masalah jalur terpendek.

Algoritma genetika merupakan salah satu metode heuristik yang cukup dikenal dalam pemecahan masalah optimasi. Diharapkan pemecahan masalah pencarian jalur terpendek menggunakan algoritma genetika dapat menghasilkan nilai optimum yang akurat dan cepat. Pemodelan perangkat lunak *Shortest Path Problem using Genetic Algorithms* (SHORGA) dilakukan menggunakan notasi UML (*Unified Modelling Language*) yang merupakan standar dalam dunia industri perangkat lunak untuk melakukan visualisasi, perancangan dan pendokumentasian suatu perangkat lunak yang berorientasi objek. Dengan menggunakan algoritma genetika dalam pengolahan data graf, diperoleh output berupa peta hasil pencarian jalur terpendek.

Hasil dan pembahasan menunjukkan bahwa sistem ini dapat digunakan sebagai referensi bagi nilai keakuratan metode algoritma genetika dalam penentuan jalur terpendek. Kelebihan dari sistem ini antara lain dibangun berbasis J2SE dengan tampilan yang *user friendly*, dan kekurangannya adalah nilai yang kurang akurat untuk data dengan jumlah banyak.

Kata-kunci: Algoritma Genetika, Pencarian Jalur Terpendek

DAFTAR ISI

LEMBAR PENGESAHAN PEMBIMBING ii
LEMBAR PENGESAHAN KEASLIAN TUGAS AKHIR iii
LEMBAR PENGESAHAN PENGUJI iv
PERSEMBAHAN v
MOTTO vi
KATA PENGANTAR vii
SARI x
DAFTAR ISI xi
DAFTAR GAMBAR xiii
DAFTAR TABEL xv
BAB I PENDAHULUAN 1
1.1 Latar Belakang 1
1.2 Rumusan Masalah 2
1.3 Batasan Masalah 2
1.4 Tujuan Penelitian 3
1.5 Manfaat Penelitian 3
1.6 Metode Penelitian 3
1.6.1 Studi Pendahuluan 4
1.6.2 Perancangan Model 4
1.6.3 Pengumpulan Data 4
1.6.4 Metode Pembuatan Perangkat Lunak 5
1.7 Sistematika Penulisan 5
BAB II DASAR TEORI 8
2.1 Teori Graf 8
2.1.1 Definisi Graf 8
2.1.2 Macam-Macam Graf 9
2.1.3 Representasi Graf 11
2.2 Permasalahan Optimasi 15
2.2.1 Penyelesaian Masalah Optimasi 15
2.2.2 Permasalaahan Jalur Terpendek 16
2.3 Algoritma Genetika 17
2.3.1 Definisi Algoritma Genetika 17
2.3.2 Proses Algoritma Genetika 20
BAB III METODOLOGI 32
3.1 Analisis Kebutuhan Perangkat Lunak 32
3.1.1 Metode Analisis 32
3.1.2 Hasil Analisis 32
3.1.3 Kebutuhan Antar Muka 34
3.1.4 Kebutuhan Perangkat Lunak 35
3.1.5 Kebutuhan Perangkat Keras 35
3.2 Perancangan Perangkat Lunak 36

3.2.1 Metode Perancangan 36
3.2.2 Hasil Perancangan 36
3.3 Implementasi Perangkat Lunak 63
3.3.1 Batasan Implementasi 63
3.3.2 Implementasi Antar Muka 64
3.3.3 Implenentasi Prosedural 70
BAB IV PEMBAHASAN 75
4.1 Pengujian Sistem 75
4.1.1 Pengujian Normal 75
4.1.2 Pengujian Tidak Normal 82
BAB V SIMPULAN DAN SARAN 83
5.1 Simpulan 83
5.2 Saran 83
DAFTAR PUSTAKA xvi

DAFTAR GAMBAR

Gambar 2.1 Contoh graf dengan simpul dan sisi 8
Gambar 2.2 Graf berarah dan berbobot 9
Gambar 2.3 Graf tidak berarah dan berbobot 10
Gambar 2.4 Graf berarah dan tidak berbobot 10
Gambar 2.5 Graf tidak berarah dan tidak berbobot 10
Gambar 2.6 Senarai kedekatan graf tidak berbobot ABCDEFG 13
Gambar 2.7 Senarai kedekatan graf berbobot ABCDEFG 14
Gambar 2.8 Graf ABCDEFG 16
Gambar 2.9 Representasi string bit dan pohon 21
Gambar 2.10 Seleksi roda roulette 22
Gambar 2.11 Seleksi *Stocastic universal sampling* 23
Gambar 2.12 Lingkungan linear (jarak=2):full dan half ring 24
Gambar 2.13 Lingkungan dimensi-2 (jarak=1):full dan half cross 25
Gambar 2.14 Lingkungan dimensi-2 (jarak=1):full dan half star 25
Gambar 2.15 Rekombinasi diskret 27
Gambar 2.16 Area induk dan anak pada rekombinasi menengah 28
Gambar 2.17 Posisi anak yang mungkin pada rekombinasi menengah 28
Gambar 2.18 Rekombinasi satu titik 29
Gambar 2.19 Rekombinasi banyak titik 29
Gambar 2.20 Diagram alir GA sederhana 31
Gambar 3.1 *Use Case Diagram* SHOR-GA 38
Gambar 3.2 *Class Diagram* modul *gui* dan *genetics* SHOR-GA 39
Gambar 3.3 *Class Diagram* modul gui SHOR-GA 40
Gambar 3.4 *Class Diagram* modul *genetics* SHOR-GA 41
Gambar 3.5 *Sequence diagram* gambar peta 43
Gambar 3.6 *Sequence diagram* cari jalur terpendek 44
Gambar 3.7 *Activity diagram* pencarian jalur terpendek 46
Gambar 3.8 Peta Provinsi Riau 47
Gambar 3.9 Representasi senarai untuk graf pada peta Provinsi Riau 48
Gambar 3.10 Rancangan antar muka halaman utama SHOR-GA 59
Gambar 3.11 Rancangan antarmuka tab *Map* halaman *Shortest Path* perangkat lunak SHOR-GA 60
Gambar 3.12 Rancangan antarmuka tab *Genetic Algorithms* halaman *Shortest Path* perangkat lunak SHOR-GA 61
Gambar 3.13 Rancangan antarmuka halaman *about* SHOR-GA 62
Gambar 3.14 Rancangan antarmuka halaman *help* SHOR-GA 63
Gambar 3.15 Implementasi Antar Muka Halaman Utama SHOR-GA 64
Gambar 3.16 Implementasi antarmuka tab *Map* halaman *Shortest Path* perangkat lunak SHOR-GA 65
Gambar 3.17 Implementasi antarmuka tab *Map* halaman *Shortest Path* perangkat lunak SHOR-GA setelah data digambarkan 66

Gambar 3.18 Antarmuka tab *Algorithms* halaman *Shortest Path* perangkat lunak SHOR-GA setelah data kota dimasukkan 67
Gambar 3.19 Antarmuka tab *Algorithms* halaman *Shortest Path* perangkat lunak SHOR-GA setelah pemrosesan selesai 68
Gambar 3.20 Implementasi Antarmuka halaman *about* SHOR-GA 69
Gambar 3.21 Implementasi antarmuka halaman *help* SHOR-GA 69
Gambar 4.1 Tampilan Menu Utama 76
Gambar 4.2 Tampilan Menu *Shortest Path* 76
Gambar 4.3 Tampilan Masukan *Select Province* 77
Gambar 4.4 Tampilan masukan parameter GA, kota asal dan tujuan 78
Gambar 4.5 Tampilan hasil pemrosesan jalur terpendek 79
Gambar 4.6 Graf untuk pengujian 80
Gambar 4.7 Perbandingan keakuratan algoritma genetika 81
Gambar 4.8 Peringatan jika *field* diisi selain angka 82

DAFTAR TABEL

Tabel 2.1 Matriks kedekatan graf berarah 11
Tabel 2.2 Matriks kedekatan graf tidak berarah 12
Tabel 3.1 Populasi Awal 50
Tabel 3.2 Probabilitas *Fitness* generasi pertama 51
Tabel 3.3 Bilangan acak untuk seleksi pada generasi pertama 52
Tabel 3.4 Hasil seleksi generasi pertama 53
Tabel 3.5 Bilangan acak untuk rekombinasi generasi pertama 54
Tabel 3.6 Induk untuk rekombinasi generasi pertama 55
Tabel 3.7 Hasil rekombinasi pada generasi pertama 55
Tabel 3.8 Bilangan acak untuk mutasi generasi pertama 56
Tabel 3.9 Kromosom terkena mutasi generasi pertama 57
Tabel 4.1 Perhitungan Bobot Menggunakan Algoritma Dijkstra 81

BAB I

PENDAHULUAN

1.1 Latar Belakang

Perkembangan teknologi informasi yang semakin pesat diimbangi dengan semakin banyaknya kebutuhan manusia akan informasi. Informasi merupakan salah satu kebutuhan yang sangat diperlukan dalam pengambilan keputusan. Informasi bisa didapatkan dengan cara manual ataupun menggunakan komputer. Komputer sebagai perangkat teknologi canggih akhirnya terpilih sebagai salah satu alternatif yang paling mungkin dalam membantu menyelesaikan pekerjaan dan menangani arus informasi dalam jumlah yang besar serta membantu dalam pengambilan keputusan yang cepat dan akurat. Hasil kerja sistem komputer ini diakui lebih cepat, teliti dan akurat dibandingkan dengan manusia. Hal inilah yang mendorong lahirnya ilmu Kecerdasan Buatan (*Artificial Intellegence,* AI).

Sebuah perjalanan terkadang membutuhkan jalur/rute yang terpendek. Biasanya jalur terpendek tersebut didapatkan dengan cara menghitung waktu yang ditempuh, ataupun berdasarkan jarak dari kota asal ke kota tujuan. Semakin banyak alternatif jalur ke kota tujuan, semakin rumit cara untuk menghitung jalur terpendek. Untuk itu diperlukan sebuah mekanisme yang handal untuk dapat menentukan jalur terpendek dari kota sumber ke kota tujuan. Penerapan metode AI dalam perhitungan jalur terpendek merupakan salah satu solusi untuk dapat menyelesaikan masalah dengan jalur yang banyak dan rumit. Pemanfaatan

komputer juga dapat mempercepat perhitungan dan memperoleh hasil yang lebih cepat dibandingkan cara manual.

Algoritma Genetika (*Genetic Algorithm*, GA) merupakan salah satu cabang dari AI. Penemu GA, Holland mengatakan bahwa setiap masalah yang berbentuk adaptasi (alami maupun buatan) dapat diformulasikan dalam terminologi genetika [KUS05]. GA juga sering digunakan pada penyelesaian masalah optimasi, seperti pada kasus *Travelling Salesman Problem* (TSP), *Minimum Spanning Tree* (MST), dan Masalah Jalur Terpendek (*Shortest Path Problem*). Penggunaan GA pada kasus MST dan TSP telah diuji pada banyak penelitian, akan tetapi kasus *Shortest Path* lebih banyak dilakukan menggunakan algoritma konvensional dengan perhitungan matematis biasa [MUT07]. Diharapkan pembuatan perangkat lunak SHORGA (*Shortest Path Problem using Genetic Algorithms*) untuk menentukan jalur terpendek menghasilkan suatu penelitian yang dapat bermanfaat dan menjadi referensi bagi peneliti selanjutnya.

1.2 Rumusan Masalah

Berdasarkan latar belakang di atas dapat dirumuskan permasalahan yang akan diselesaikan yaitu membangun perangkat lunak SHORGA untuk menyelesaikan masalah jalur terpendek dengan menggunakan GA.

1.3 Batasan Masalah

Dalam melaksanakan suatu penelitian diperlukan adanya batasan agar tidak menyimpang dari yang telah direncanakan sehingga tujuan yang sebenarnya dapat tercapai. Batasan masalah yang diperlukan yaitu:

1. Perangkat lunak SHORGA dibangun untuk menentukan jalur terpendek antar kota menggunakan GA.
2. Masukan yang diperlukan antara lain nama provinsi, posisi koordinat kota, serta parameter GA.
3. Keluaran yang dihasilkan antara lain jalur terpendek dan bobot (jarak), serta hasil dan pelaporan proses GA.
4. Perangkat lunak dibangun menggunakan bahasa pemrograman Java.

1.4 Tujuan Penelitian

Tujuan yang diharapkan dari penulisan tugas akhir ini adalah

1. Membuat suatu perangkat lunak yang dapat menentukan jalur terpendek antar kota di suatu provinsi.
2. Sebagai referensi bagi peneliti lain, mengenai tingkat keakuratan penentuan jalur terpendek menggunakan GA.

1.5 Manfaat Penelitian

Manfaat yang diharapkan dari penelitian ini adalah sebagai informasi bagi para peneliti bahwa GA dapat diterapkan pada kasus optimasi, khususnya pencarian jalur terpendek.

1.6.4 Metode Pembuatan Perangkat Lunak

Setelah pengumpulan data, diperlukan metode untuk perancangan dan pembuatan perangkat lunak. Metode pembuatan perangkat lunak yang digunakan pada tugas akhir ini adalah:

a. Analisis Data

Analisis data dilakukan untuk mengolah data yang sudah didapat dan mengelompokkan data sesuai dengan kebutuhan perancangan.

b. Desain

Tahap ini merupakan tahap penerjemahan dari keperluan data yang telah dianalisis ke dalam bentuk antar muka yang mudah dimengerti oleh pengguna.

c. Implementasi

Implementasi pada perangkat lunak (*software*) menggunakan teknologi Java.

d. Pengujian

Pengujian terhadap perangkat lunak yang telah dibangun, dengan pengujian secara normal dan tidak normal.

1.7 Sistematika Penulisan

Dalam penyusunan tugas akhir ini, sistematika penulisan dibagi menjadi beberapa bab sebagai berikut :

BAB I PENDAHULUAN

Bab ini berisi pembahasan masalah umum yang meliputi latar belakang masalah, rumusan masalah, batasan masalah, tujuan

penelitian, manfaat penelitian, metodologi penelitian dan sistematika penulisan.

BAB II LANDASAN TEORI

Bagian ini memuat landasan teori yang berfungsi sebagai sumber atau alat dalam memahami permasalahan yang berkaitan dengan teori graf, teori jalur terpendek, dan teori mengenai CA.

BAB III METODOLOGI

Bagian ini memuat uraian tentang metode analisis kebutuhan perangkat lunak yang dipakai, serta hasil analisis kebutuhan perangkat lunak berupa analisis kebutuhan proses, analisis kebutuhan masukan, analisis kebutuhan keluaran, kebutuhan perangkat lunak, kebutuhan perangkat keras dan kebutuhan antar muka, perancangan diagram UML (*Unified Modelling Language*), perancangan GA dan implementasi perangkat lunak yang dibuat dan memuat dokumentasi atau tampilan form-form yang telah dibangun.

BAB IV HASIL DAN PEMBAHASAN

Bab ini membahas tentang analisis kinerja dari perangkat lunak. Pada bagian ini mengulas analisis hasil pengujian terhadap sistem yang dibandingkan dengan kebenaran dan kesesuaiannya dengan kebutuhan perangkat lunak yang telah dituliskan pada bagian sebelumnya.

BAB V SIMPULAN DAN SARAN

Memuat kesimpulan-kesimpulan yang merupakan rangkuman dari hasil dan pembahasan perangkat lunak pada bagian sebelumnya dan saran yang perlu diperhatikan berdasarkan keterbatasan yang ditemukan dan asumsi-asumsi yang dibuat selama pembuatan perangkat lunak.

BAB II

LANDASAN TEORI

2.1 Teori Graf

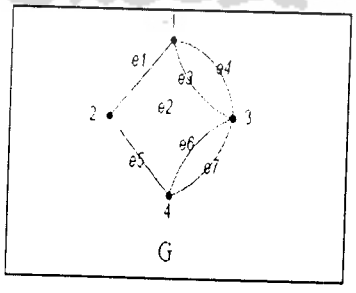
2.1.1 Definisi Graf

Graf adalah kumpulan simpul (*nodes*) yang dihubungkan satu sama lain melalui sisi/busur (*edges*) [ZAK06]. Suatu graf G terdiri dari dua himpunan yang berhingga, yaitu titik-titik tidak kosong (simbol V(G)) dan himpunan garis-garis (simbol E(G)) [SIA02].

Vertex (simpul): V = himpunan simpul yang terbatas dan tidak kosong.

Edge (sisi/busur): E = himpunan sisi yang menghubungkan sepasang simpul.

Simpul-simpul pada graf dapat merupakan obyek sembarang seperti kota, atom-atom suatu zat, nama anak, jenis buah, komponen alat elektronik dan sebagainya. Sisi dapat menunjukkan hubungan (relasi) sembarang seperti rute penerbangan, jalan raya, sambungan telepon, ikatan kimia, dan lain-lain. Notasi graf: G (V,E) artinya graf G memiliki simpul V dan sisi E.



Gambar 2.1 Contoh graf dengan simpul dan sisi

Pada gambar 2.1 diatas, ditunjukkan bahwa G adalah graf dengan:

$V = \{1,2,3,4\}$

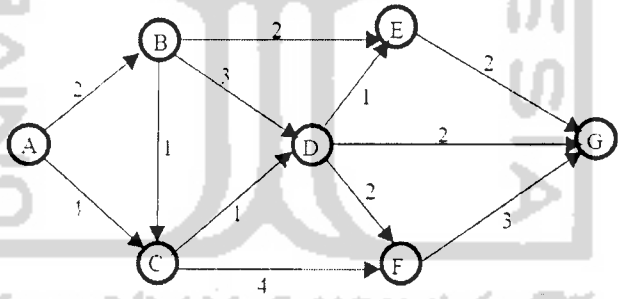
$E = \{(1,2),(2,3),(1,3),(1,3),(2,4),(3,4),(3,4)\}$

$= \{e1, e2, e3, e4, e5, e6, e7\}$

2.1.2 Macam-macam Graf

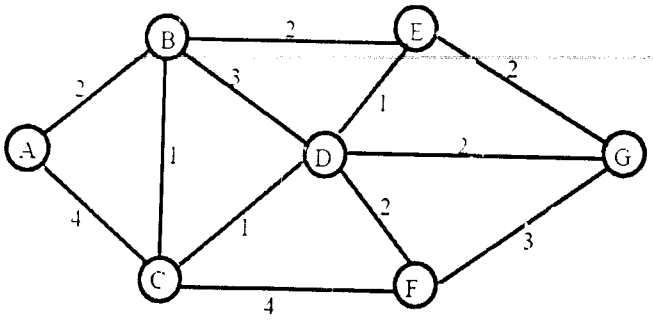
Menurut arah dan bobotnya, graf dibagi menjadi empat bagian, yaitu [SIA02]:

1. Graf berarah dan berbobot: tiap busur mempunyai anak panah dan bobot. Contoh graf berarah dan berbobot dapat dilihat pada gambar 2.2.



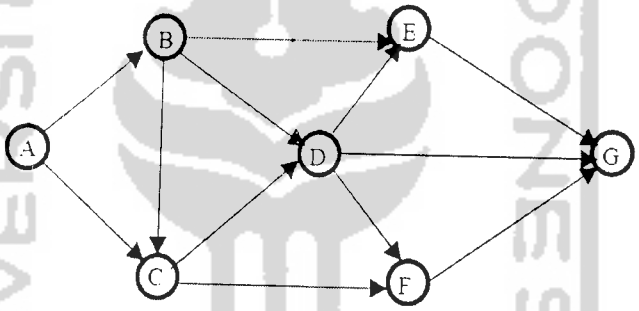
Gambar 2.2 Graf berarah dan berbobot

2. Graf tidak berarah dan berbobot: tiap busur tidak mempunyai anak panah tetapi mempunyai bobot. Contoh graf tidak berarah dan berbobot dapat dilihat pada gambar 2.3.



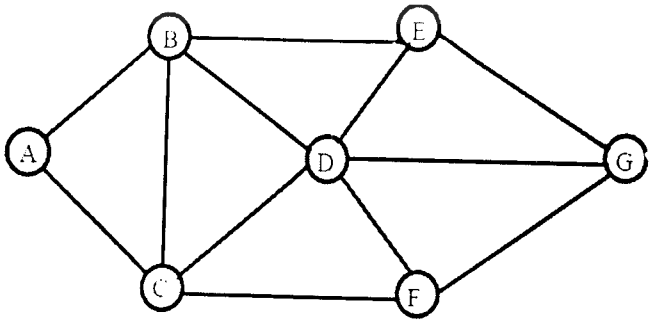
Gambar 2.3 Graf tidak berarah dan berbobot

3. Graf berarah dan tidak berbobot: tiap busur mempunyai anak panah yang tidak berbobot. Contoh graf berarah dan tidak berbobot dapat dilihat pada gambar 2.4.



Gambar 2.4 Graf berarah dan tidak berbobot

4. Graf tidak berarah dan tidak berbobot: tiap busur tidak mempunyai anak panah dan tidak berbobot. Contoh graf tidak berarah dan tidak berbobot dapat dilihat pada gambar 2.5.



Gambar 2.5 Graf tidak berarah dan tidak berbobot

2.1.3 Representasi Graf

Suatu graf dapat direpresentasikan ke beberapa bentuk. Representasi graf dapat digunakan untuk mengimplementasikan graf tersebut ke dalam bentuk tertentu, sehingga dapat digunakan pada berbagai kasus yang berbeda. Representasi graf yang sering digunakan diantaranya [ZAK06]:

1. Matriks Kedekatan (*Adjacency Matrix*)

   Untuk suatu graf dengan jumlah simpul sebanyak n, maka matriks kedekatan mempunyai ukuran n x n (n baris dan n kolom). Jika antara dua buah simpul terhubung maka elemen matriks bernilai 1, dan sebaliknya bernilai 0 jika tidak terhubung. Representasi matriks kedekatan biasanya digunakan untuk graf berarah dan graf tidak berarah. Tabel matriks kedekatan untuk graf berarah dapat dilihat pada Tabel 2.1.

Tabel 2.1 Matriks kedekatan graf berarah

	A	B	C	D	E	F	G
A	0	1	1	0	0	0	0
B	1	0	1	1	1	0	0
C	1	1	0	1	0	1	0
D	0	1	1	0	1	1	1
E	0	1	0	1	0	0	1
F	0	0	1	1	0	0	1
G	0	0	0	1	1	1	0

Pada tabel 2.1 di atas, elemen matriks kedekatan bernilai 0 untuk diagonal dan elemen yang tidak terhubung dengan simpul lain (elemen matriks bernilai

0 jika simpul tidak terhubung dengan simpul lainnya). Sedangkan elemen lainnya yang terhubung bernilai 1.

Tabel matriks kedekatan untuk graf tidak berarah dapat dilihat pada Tabel 2.2.

Tabel 2.2 Matriks kedekatan graf tidak berarah

	A	B	C	D	E	F	G
A	0	2	4	0	0	0	0
B	2	0	1	3	2	0	0
C	4	1	0	1	0	4	0
D	0	3	1	0	2	2	1
E	0	2	0	1	0	0	2
F	0	0	4	2	0	0	3
G	0	0	0	2	2	3	0

Pada tabel 2.2 di atas, elemen matriks kedekatan bernilai 0 untuk diagonal dan elemen yang tidak terhubung dengan simpul lain (elemen matriks bernilai 0 jika simpul tidak terhubung dengan simpul lainnya). Sedangkan elemen lainnya yang terhubung berisi bobot dari graf tersebut.

Ruang (memori) yang diperlukan untuk matriks kedekatan adalah:

$$nxn = (N)^2 \tag{2.1}$$

Untuk graf tidak berarah:

a. Ruang yang diperlukan = $\frac{1}{2}N^2 - \frac{1}{2}N$ (2.2)

b. Derajat simpul i = $\sum_{j=1}^{n} A[i, j]$ (2.3)

Untuk graf berarah :

a. Derajat luar (*out degree*) = jumlah dalam 1 baris (matriks) atau bayaknya busur dengan simpul V sebagai kepala.

b. Derajat dalam (*in degree*) = jumlah dalam 1 kolom (matriks) atau banyaknya busur dengan simpul V sebagai ekor.

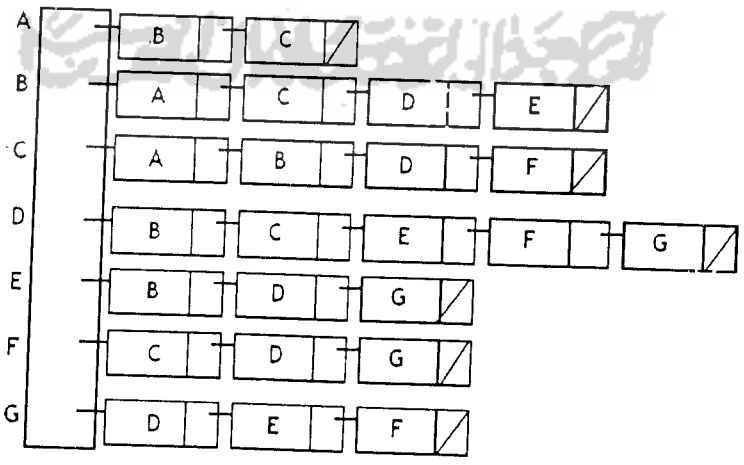
2. Senarai Kedekatan (*Adjacency List*)

Pada simpul x dapat dianggap sebagai suatu senarai yang terdiri dari simpul pada graf yang berdekatan dengan x. Representasi senarai kedekatan mempunyai kesamaan fleksibilitas dengan matriks kedekatan, akan tetapi representasi ini lebih tersusun rapi.

Ruang (memori) yang diperlukan untuk n simpul dan e sisi pada graf tidak berarah:

$$n \text{ head node} + 2e \text{ node list} \qquad (2.4)$$

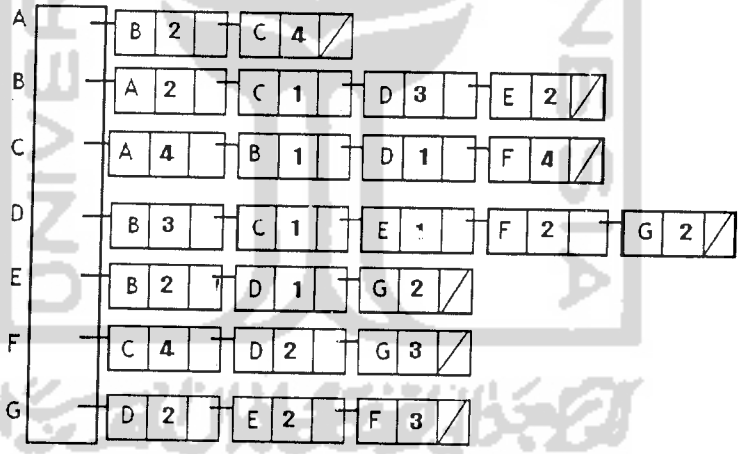
Senarai kedekatan untuk graf tidak berbobot ABCDEFG dapat dilihat pada gambar 2.6 [SAP07].



Gambar 2.6 Senarai kedekatan graf tidak berbobot ABCDEFG

Pada gambar 2.6 di atas, senarai kedekatan graf tidak berbobot ABCDEFG dapat dilihat pada keterkaitan yang ada pada tiap simpul graf. Simpul A dapat berkait dengan simpul B dan C, simpul B berkait dengan simpul A, C, D dan E, simpul C berkait dengan simpul A, B, D dan F, simpul D berkait dengan simpul B, C, E, F dan G, simpul E berkait dengan simpul B, D dan G, simpul F berkait dengan simpul C, D dan G, dan simpul G berkait dengan simpul D, E dan F.

Senarai kedekatan untuk graf berbobot ABCDEFG dapat dilihat pada gambar 2.7:



Gambar 2.7 Senarai kedekatan graf berbobot ABCDEFG

2.2 Permasalahan Optimasi

2.2.1 Penyelesaian Masalah Optimasi

Masalah optimasi adalah suatu permasalahan yang bertujuan untuk menentukan hasil terbaik dari suatu kasus. Masalah optimasi dapat menyelesaikan hasil terbaik dengan keluaran yang minimum ataupun maksimum.

Secara umum, penyelesaian masalah optimasi dapat dilakukan dengan menggunakan dua metode, yaitu metode konvensional dan metode heuristik. Metode konvensional diterapkan dengan perhitungan matematis biasa, sedangkan metode heuristik diterapkan dengan perhitungan kecerdasan buatan [MUT07].

1. Metode Konvensional

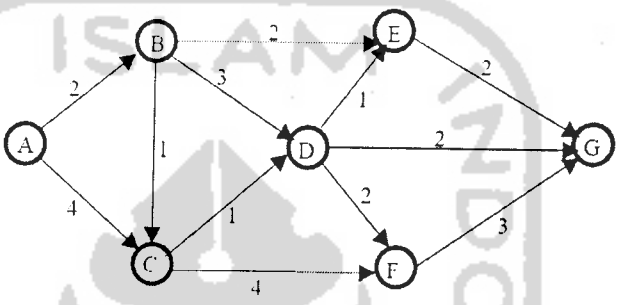
   Metode konvensional adalah metode yang menggunakan perhitungan matematis biasa. Ada beberapa algoritma konvensional yang biasa digunakan untuk melakukan permasalahan optimasi, diantaranya: algoritma Dijkstra, algoritma Floyd-Warshall, algoritma Bellman-Ford, algoritma Warshall, algoritma Kruskal, dan algoritma Prim.

2. Metode Heuristik

   Metode Heuristik adalah sub bidang dari AI yang digunakan untuk melakukan pencarian dan optimasi. Ada beberapa algoritma pada metode heuristik yang biasa digunakan dalam permasalahan optimasi, diantaranya GA, algoritma semut, logika fuzzy, jaringan syaraf tiruan, pencarian tabu, *simulated annealing*, *hill* climbing, *generate and test* dan lain-lain.

2.2.2 Permasalahan Jalur Terpendek (*Shortest Path Problem*)

Jalur terpendek adalah suatu jaringan pengarahan perjalanan dimana sesorang pengarah jalan ingin menentukan jalur terpendek antara dua kota, berdasarkan beberapa jalur alternatif yang tersedia, dimana titik tujuan hanya satu. Gambar 2.8 menunjukkan suatu graf ABCDEFG yang berarah dan berbobot [SAP07].



Gambar 2.8 Graf ABCDEFG

Pada gambar 2.8 diatas, misalkan kota A adalah kota asal dan kota G adalah kota tujuan. Untuk menuju kota G, dapat dipilih beberapa jalur yang tersedia dengan jumlah bobot masing-masing:

A → B → C → D → E → G	= 2 + 1 + 1 + 1 + 2	= 7
A → B → C → D → F → G	= 2 + 1 + 1 + 2 + 3	= 9
A → B → C → D → G	= 2 + 1 + 2 + 3	= 8
A → B → C → F → G	= 2 + 1 + 4 + 3	= 10
A → B → D → E → G	= 2 + 3 + 1 + 2	= 8
A → B → D → F → G	= 2 + 1 + 4 +3	= 10
A → B → D → G	= 2 + 3 +2	= 7

A → B → E → G	= 2 + 2 +2	= 6
A → C → D → E → G	= 4 + 1 + 1 + 2	= 8
A → C → D → F → G	= 4 + 1 + 2 + 3	= 10
A → C → D → G	= 4 + 1 + 2	= 7
A → C → F → G	= 4 + 4 + 3	= 11

Berdasarkan data diatas, didapatkan jalur terpendek adalah jalur A→B→E→G, dengan bobot terkecil yaitu 6. Pada permasalahan jalur terpendek, bobot direpresentasikan sebagai jarak antar kota dengan nilai tertentu. Apabila pada suatu kasus hanya terdapat koordinat kota dan jarak antar kota belum diketahui, maka jarak antar kota dapat dihitung dengan menggunakan variabel-variabel berupa koordinat kota tersebut.

2.3 Algoritma Genetika

2.3.1 Definisi Algoritma Genetika

Algoritma genetika (*Genetic Algorithm*, GA) adalah algoritma pencarian yang didasarkan atas mekanisme seleksi alami dan evolusi biologis. GA mengkombinasikan antara deretan struktur dengan pertukaran informasi acak ke bentuk algoritma pencarian dengan beberapa perubahan bakat pada manusia. Pada setiap generasi, himpunan baru dari deretan individu dibuat berdasarkan kecocokan pada generasi sebelumnya [GOL89].

Pada dasarnya ada 4 kondisi yang sangat mempengaruhi proses evaluasi, yakni sebagai berikut [KUS05]:

a. Kemampuan organisme untuk melakukan reproduksi.

b. Keberadaan populasi organisme yang bisa melakukan reproduksi.

c. Keberagaman organisme dalam suatu populasi.

d. Perbedaan kemampuan untuk bertahan hidup.

GA pertama kali dikembangkan oleh Holland dari Universitas Michigan (1975). Holland mengatakan bahwa setiap masalah yang berbentuk adaptasi (alami maupun buatan) dapat diformulasikan dalam terminologi genetika [KUS05].

GA memiliki sifat kokoh, memiliki keseimbangan antara efisiensi dan efektifitas dalam menyelesaikan kasus-kasus dari wilayah masalah yang berbeda. Hal ini akan berakibat pada penurunan biaya perancangan ulang sistem. Biaya perancangan ulang yang lebih rendah menunjukkan bahwa sistem tersebut memiliki daya adaptasi yang tinggi sehingga sistem ini dapat bertahan lebih lama dibandingkan dengan sistem-sistem lainnya.

GA menggunakan mekanisme seleksi alam dan ilmu genetik sehingga istilah-istilah pada GA akan bersesuaian dengan istilah-istilah pada seleksi alam dan ilmu genetik. Pada ilmu genetik, kromosom terdiri dari susunan gen-gen. Tiap gen mengandung nilai atau sifat tertentu yang disebut *allele*, sedangkan posisi gen dalam kromosom disebut *locus*. Selanjutnya, satu atau beberapa kromosom bergabung membentuk paket genetik yang disebut genotif. Interaksi genotif dengan lingkungannya disebut fenotif.

Pada GA, kromosom bersesuaian dengan string yang dibentuk dari beberapa karakter. Setiap karakter mempunyai posisi (*locus*) dan mengandung nilai tertentu (*allele*). Satu atau beberapa string akan bergabung membentuk

struktur (genotif). Bila struktur tersebut dikodekan, akan diperoleh satu titik yang merupakan salah satu alternatif solusi (fenotif).

Satu siklus iterasi GA (sering disebut sebagai generasi) terdapat dua proses, yakni proses seleksi dan rekombinasi. Proses seleksi adalah proses evaluasi kualitas setiap string didalam populasi untuk memperoleh peringkat calon solusi. Berdasarkan hasil evaluasi, dipilih string-string yang akan mengalami proses rekombinasi. Proses pemilihan biasanya dilakukan secara acak, string dengan kualitas yang lebih baik akan memiliki peluang lebih besar untuk terpilih sebagai calon-calon string generasi berikutnya.

Proses rekombinasi meliputi proses genetika untuk memperoleh string baru dari pertukaran karakter dari calon-calon string yang didapat pada tahap seleksi. String-string pada generasi baru dihasilkan dengan menggunakan operasi genetik secara acak pada calon string yang terpilih pada tahap seleksi. Proses rekombinasi akan menghasilkan string-string baru yang berbeda dibandingkan induknya dan dengan demikian diperoleh domain pencarian yang baru [SAP04].

Cara kerja GA sangat sederhana, hanya mencakup proses penduplikasian string-string dan pertukaran bagian-bagian dari string. Meskipun cukup sederhana, tetapi mempunyai kemampuan untuk menyelesaikan persoalan optimasi. Kemampuan ini didukung oleh tiga operator genetik yaitu reproduksi, rekombinasi dan mutasi. Pada reproduksi terjadi proses penduplikasian string berdasarkan nilai fungsi objektifnya. Nilai objektif ini dapat dilihat sebagai suatu keuntungan yang ingin dicapai atau dimaksimalkan. Sementara proses pertukaran bagian-bagian string dilakukan oleh operator rekombinasi dan mutasi

Di samping ketiga operator dasar (reproduksi, rekombinasi, dan mutasi), parameter-parameter genetik (jumlah populasi, maksimum generasi, probabilitas rekombinasi, probabilitas mutasi, dan lain-lain), serta asumsi-asumsi yang digunakan dalam pemodelannya juga mempunyai peran penting.

2.3.2 Proses Algoritma Genetika

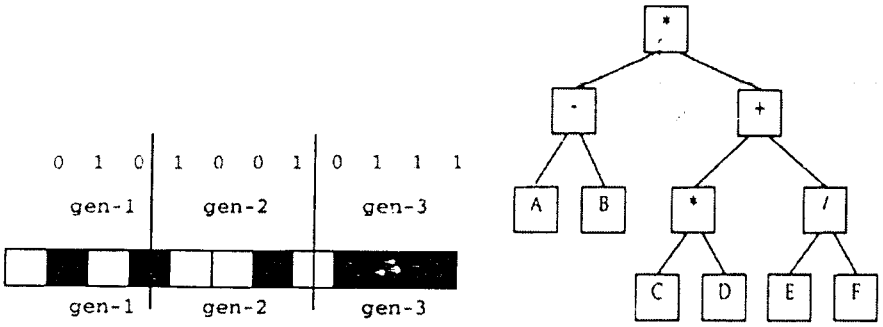
Pada GA terdapat beberapa proses yaitu :

1. Proses Pengkodean (*Encoding*)

   Pengkodean adalah salah suatu proses yang sulit dalam GA. Hal ini disebabkan karena proses pengkodean untuk setiap permasalahan berbeda-beda karena tidak semua teknik pengkodean cocok untuk setiap permasalahan. Proses pengkodean ini menghasilkan suatu deretan yang kemudian disebut kromosom. Kromosom terdiri dari sekumpulan bit yang dikenal sebagai gen.

   Ada beberapa macam teknik pengkodean yang dapat dilakukan dalam GA, diantaranya pengkodean biner (*binary encoding*), pengkodean permutasi (*permutation encoding*), pengkodean nilai (*value encoding*) dan pengkodean pohon (*tree encoding*) [LUK05].

   Pada proses pengkodean, gen dapat direpresentasikan dalam bentuk string bit, pohon, array bilangan real, daftar aturan, elemen permutasi, elemen program, atau representasi lainnya yang dapat diimplementasikan untuk operator genetika [KUS05]. Gambar 2.9 menunjukkan representasi string bit dan pohon.



Gambar 2.9 Representasi string bit dan pohon

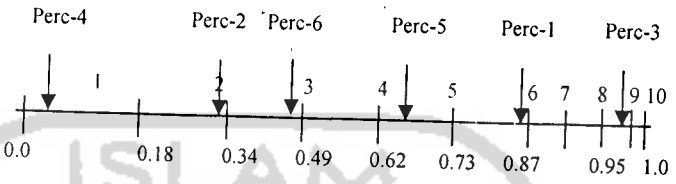
2. Proses Seleksi

Seleksi adalah proses untuk menentukan individu mana saja yang akan dipilih untuk dilakukan rekombinasi dan bagaimana keturunan terbentuk dari individu-individu terpilih tersebut. Langkah pertama yang dilakukan dalam seleksi adalah pencarian nilai *fitness*. Masing-masing individu dalam suatu wadah seleksi akan menerima probabilitas reproduksi yang tergantung pada nilai obyektif dirinya sendiri terhadap nilai obyektif dari semua individu dalam wadah seleksi tersebut. Nilai *fitness* kemudian akan digunakan pada tahap seleksi berikutnya.

Ada beberapa macam proses seleksi yang ada pada GA, diantaranya [KUS05]:

a. Seleksi dengan Roda Roulette (*Roulette Wheel Selection*), dengan memetakan individu-individu dalam suatu segmen garis secara berurutan sedemikian hingga tiap-tiap segmen individu memiliki ukuran yang sama dengan ukuran *fitness*-nya. Sebuah bilangan acak dibangkitkan dan individu yang memiliki segmen dalam kawasan bilangan acak tersebut

akan terseleksi. Proses ini akan diulang hingga diperoleh sejumlah individu yang diharapkan. Gambar 2.10 menunjukkan seleksi roda roulette.



Gambar 2.10 Seleksi roda roulette

Algoritma seleksi roda roulette:

i. Hitung total *fitness* (F):

$$TotFitness = \sum F_k; k = 1,2,..,popsize \quad (2.5)$$

ii. Hitung *fitness* relatif tiap individu

$$p_k = F_k / TotFitness \quad (2.6)$$

iii. Hitung *fitness* komulatif

$$q_1 = p_1 \quad (2.7)$$

$$q_k = q_{k-1} + p_k; k = 2,3,..,popsize \quad (2.8)$$

iv. Pilih induk yang akan menjadi kandidat untuk disilangkan dengan cara:

Bangkitkan bilangan acak r.

Jika $q_k \in r$ dan $q_{k+1} > r$, maka pilih kromosom ke $(k+1)$ sebagai kandidat induk.

b. Seleksi berdasarkan Ranking Fitness (*Rank-based Fitness*), yaitu dengan cara mengurutkan populasi menurut nilai objektifnya. Misalkan N adalah jumlah individu dalam suatu populasi, Pos adalah posisi individu dalam populasi tersebut (posisi terendah suatu individu adalah $Pos = 1$, dan posisi tertingginya $Pos = N$). Sedangkan SP adalah *selective pressure*. Nilai *fitness* dari suatu individu dapat dihitung sebagai berikut:

i. Linear Ranking

$$Fitness(Pos) = 2\text{-}SP + 2(SP\text{-}1)(Pos\text{-}1)/(N\text{-}1) \quad (2.9)$$

Nilai $SP \in [1,2]$ (2.10)

ii. Non-Linear Ranking

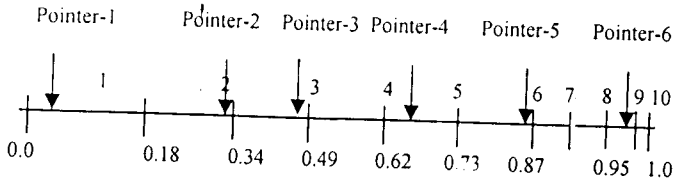
$$Fitness(Pos) = Nind \times X^{(Pos-1)} / \sum (x^{(i-1)}); i = 1..N \quad (2.11)$$

Sedangkan X dihitung sebagai akar polinomial:

$$(SP-1) \times X^{(N-1)} + SP \times X^{(N-2)} + .. + SP \times X + SP = 0 \quad (2.12)$$

Nilai $SP \in [1, N-2]$ (2.13)

c. Seleksi *Stocastic Universal Sampling*, dengan memetakan individu-individu seperti halnya roda roulette, kemudian memberikan sejumlah *pointer* sebanyak individu yang ingin diseleksi pada garis tersebut. Gambar 2.11 menunjukkan contoh seleksi *stocastic universal sampling*.

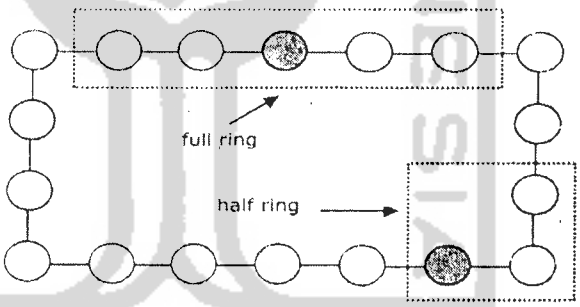


Gambar 2.11 Seleksi *Stocastic universal sampling*

d. Seleksi Lokal (*Local Selection*), seleksi yang dilakukan hanya pada konstrain tertentu yang disebut dengan nama lingkungan lokal. Interaksi individu hanya dilakukan di dalam wilayah tersebut. Lingkungan tersebut diterapkan sebagai struktur dimana populasi tersebut terdistribusi. Lingkungan tersebut juga dapat dipandang sebagai kelompok pasangan-pasangan yang potensial. Struktur lingkungan pada seleksi lokal dapat berbentuk:

i. Linear: *Full* dan *Half Ring*

Gambar 2.12 menunjukkan seleksi lokal, dalam lingkungan linear (*full & half ring*).

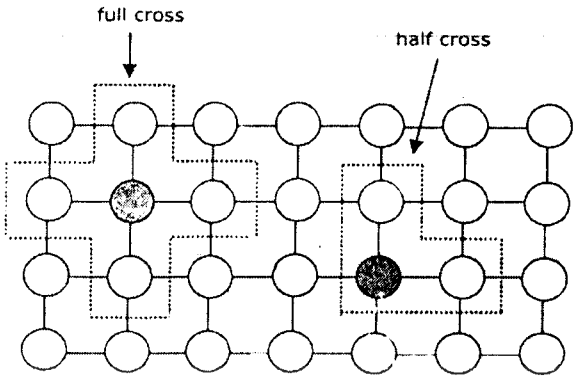


Gambar 2.12 Lingkungan linear (jarak=2):full dan half ring

ii. Dimensi-2, terdiri dari:

1. *Full cross* dan *half cross*

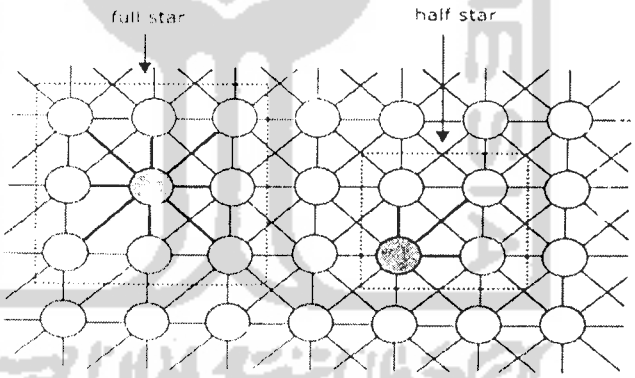
Gambar 2.13 menunjukkan seleksi lokal, dalam lingkungan dimensi-2 (*full & half cross*).



Gambar 2.13 Lingkungan dimensi-2 (jarak=1):full dan half cross

2. *Full star* dan *half star*

Gambar 2.14 menunjukkan seleksi lokal, dalam lingkungan dimensi-2 (*full & half star*).



Gambar 2.14 Lingkungan dimensi-2 (jarak=1):full dan half star

e. Seleksi dengan Pemotongan (*Truncation Selection*), seleksi buatan yang biasanya digunakan oleh polulasi yang jumlahnya sangat besar. Pada metode ini, individu-individu diurutkan berdasarkan nilai fitnessnya. Hanya individu-individu terbaik saja yang akan diseleksi sebagai induk.

Parameter yang digunakan dalam metode ini adalah suatu nilai ambang *trunc* yang mengindikasikan ukuran populasi yang akan diseleksi sebagai induk yang berkisar antara 5%-10%. Individu-individu yang ada di bawah nilai ambang ini tidak akan menghasilkan keturunan.

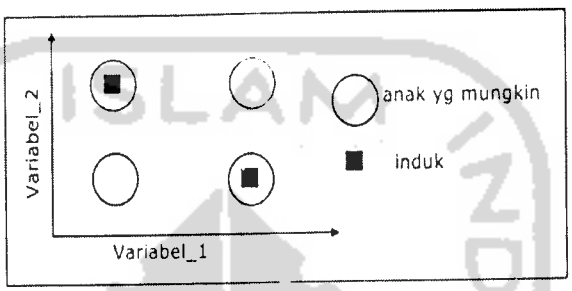
f. Seleksi dengan Turnamen (*Tournament Selection*), menetapkan suatu nilai turnamen untuk individu-individu yang dipilih secara acak dari suatu populasi. Individu-individu yang terbaik dalam kelompok ini akan diseleksi sebagai induk. Parameter yang digunakan dalam metode ini adalah ukuran *tour* yang bernilai antara 2 sampai N (jumlah individu dalam suatu populasi).

3. Proses Rekombinasi

Rekombinasi adalah proses untuk menyilangkan dua kromosom sehingga membentuk kromosom baru yang harapannya lebih baik dari pada induknya. Rekombinasi dikenal juga dengan nama *crossover*. Tidak semua kromosom pada suatu populasi akan mengalami proses rekombinasi. Kemungkinan suatu kromosom mengalami proses rekombinasi didasarkan pada probabilitas *crossover* yang telah ditentukan terlebih dahulu. Probabilitas *crossover* menyatakan peluang suatu kromosom akan mengalami *crossover*.

Ada beberapa macam proses rekombinasi yang ada pada GA, diantaranya [KUS05]:

i.Rekombinasi diskret, dengan menukar nilai variabel antar kromosom induk. Rekombinasi ini dapat digunakan untuk sembarang variabel (biner, real atau simbol). Gambar 2.15 menunjukkan posisi anak yang mungkin setelah terjadinya rekombinasi diskret.

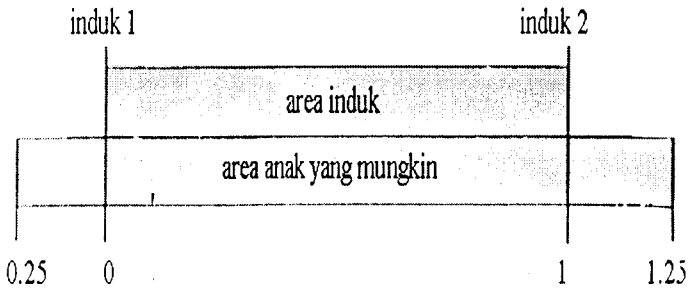


Gambar 2.15 Rekombinasi diskret

ii.Rekombinasi menengah, merupakan metode rekombinasi yang hanya digunakan untuk variabel *real* dan variabel yang bukan biner. Nilai variabel anak dipilih di sekitar dan antara nilai-nilai variabel induk. Anak dihasilkan menurut aturan sebagai berikut:

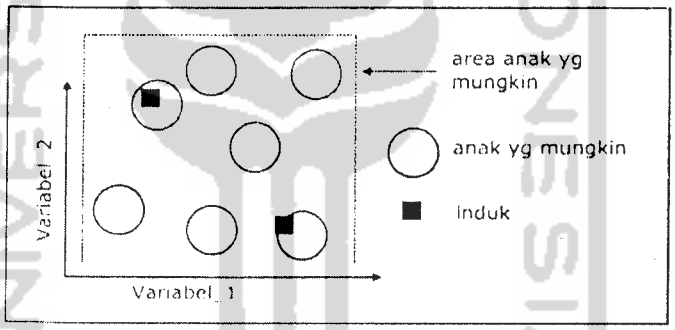
$$anak = induk\ 1 + alpha(induk\ 2 - induk1) \qquad (2.14)$$

dengan alpha adalah faktor skala yang dipilih secara acak pada interval $[-d,\ 1+d]$, biasanya d=0.25. Tiap-tiap variabel pada anak merupakan hasil kombinasi variabel-variabel menurut aturan diatas dengan nilai alpha dipilih ulang untuk tiap variabel. Gambar 2.16 menunjukkan area induk dan anak pada rekombinasi menengah.



Gambar 2.16 Area induk dan anak pada rekombinasi menengah

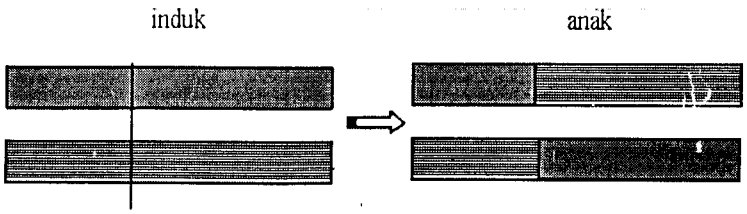
Sedangkan gambar 2.17 menunjukkan posisi anak yang mungkin pada rekombinasi menengah.



Gambar 2.17 Posisi anak yang mungkin pada rekombinasi menengah

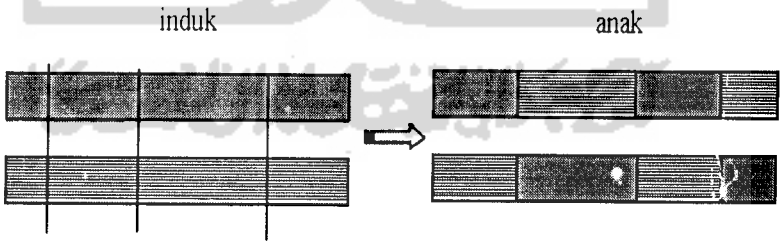
iii. Rekombinasi garis, memiliki prinsip yang sama dengan rekombinasi menengah, dengan nilai alpha sama untuk semua variabel.

iv. Rekombinasi satu titik, dengan menukar variabel-variabel antar kromosom pada satu titik untuk menghasilkan anak. Gambar 2.18 menunjukkan penukaran variabel antar kromosom pada titik untuk menghasilkan anak.



Gambar 2.18 Rekombinasi satu titik

v.Rekombinasi banyak titik, dengan menukar variabel-variabel antar kromosom pada banyak titik untuk menghasilkan anak. Pada rekombinasi ini, m posisi penyilangan $k_i(k = 1,2,..,N-1; i = 1,2,..,m)$ dengan N = panjang kromosom diseleksi secara acak dan tidak diperbolehkan ada posisi yang sama, serta diurutkan naik. Gambar 2.19 menunjukkan variabel–variabel ditukar antar kromosom pada titik tersebut untuk menghasilkan anak.



Gambar 2.19 Rekombinasi banyak titik

vi.Rekombinasi seragam, dengan membuat sebuah *mask* penyilangan sepanjang panjang kromosom secara acak yang menunjukkan bit-bit dalam

mask yang mana induk akan mensupply anak dengan bit-bit yang ada. Induk mana yang akan menyumbangkan bit ke anak dipilih secara acak dengan probabilitas sama.

vii. Penyilangan dengan permutasi, dengan cara memilih sub-barisan suatu turnamen dari satu induk dengan tetap menjaga urutan dan posisi sejumlah kota yang mungkin terhadap induk lainnya.

4. Proses Mutasi

Mutasi adalah proses penambahan nilai acak yang sangat kecil dengan probabilitas rendah pada variabel keturunan. Peluang mutasi didefinisikan sebagai persentasi dari jumlah total gen pada populasi yang mengalami mutasi. Peluang mutasi mengendalikan banyaknya gen baru yang akan dimunculkan untuk dievaluasi. Jika peluang mutasi terlalu kecil, banyak gen yang mungkin berguna tidak dievaluasi, tetapi bila peluang mutasi ini terlalu besar maka akan terlalu banyak gangguan acak, sehingga anak akan kehilangan kemiripan dari induknya dan algoritma juga akan kehilangan kemampuan untuk belajar dari *history* pencarian [KUS05].

Ada beberapa macam proses mutasi yang ada pada GA, diantaranya:

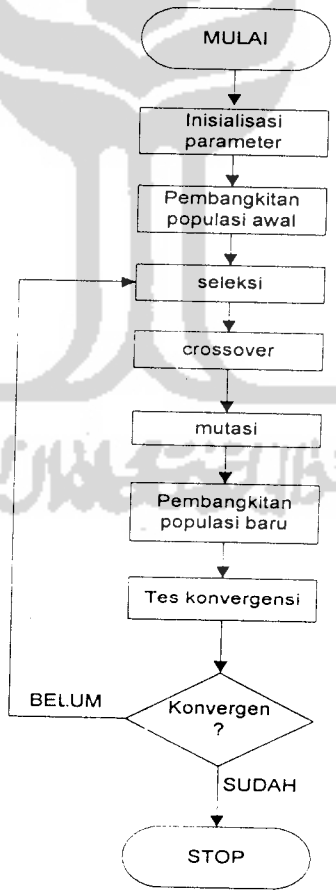
a. Mutasi bilangan real, dengan mendefinisikan ukuran langkah mutasi, kecil atau besar.

b. Mutasi biner, dengan mengganti satu atau beberapa nilai gen dari kromosom.

Berikut adalah *pseudocode* GA [EIB03] :

```
BEGIN
        GENERASI = 0
        INISIALISASI parameter
        BANGKITKAN populasi
        EVALUASI nilai fitness
        DO
                GENERASI = GENERASI+1
                SELEKSI induk
                REKOMBINASI pasangan induk
                MUTASI hasil keturunan
                EVALUASI nilai fitness
                BANGKITKAN populasi baru
        WHILE (maksimum generasi)
END
```

Gambar 2.20 menunjukkan diagram alir untuk GA sederhana:



Gambar 2.20 Diagram alir GA sederhana

BAB III

METODOLOGI

3.1 Analisis Kebutuhan Perangkat Lunak

3.1.1 Metode Analisis

Metode analisis yang digunakan dalam penelitian ini adalah analisis berorientasi objek dengan menggunakan alat dan teknik yang disesuaikan dengan sistem. Arsitektur sistem berusaha didefinisikan sebaik mungkin agar pengembangan perangkat lunak dapat berjalan dengan terstruktur dan jelas meskipun mengalami peningkatan kompleksitas yang signifikan.

Pada tahap analisis ini, digunakan suatu alat untuk melakukan pemodelan agar pengembangan perangkat lunak dapat memenuhi semua kebutuhan pengguna dengan lengkap dan tepat. Pemodelan dilakukan menggunakan notasi UML (*Unified Modelling Languag*e) yang merupakan standar dalam dunia industri perangkat lunak untuk melakukan visualisasi, perancangan dan pendokumentasian suatu perangkat lunak yang berorientasi objek.

3.1.2 Hasil Analisis

Berdasarkan data yang diperoleh melalui obeservasi selama penelitian maka didapatkan hasil analisis yang terdiri dari kebutuhan proses, kebutuhan masukan dan kebutuhan keluaran.

1. Kebutuhan Proses

Kebutuhan proses dalam perangkat lunak *Shortest Path Application using Genetic Algorithm* (SHOR-GA) antara lain:

a. Proses penggambaran graf ke bidang.

Pada proses penggambaran graf ke bidang, termasuk juga pemilihan data provinsi yang diambil dari data XML, kemudian diproses untuk bisa digambarkan ke bidang yang ada.

b. Proses penentuan jalur terpendek dan bobotnya.

Proses penentuan jalur terpendek merupakan proses inti dari aplikasi SHOR-GA. Proses ini mempunyai input berupa parameter GA, kota sumber dan kota tujuan.

2. Analisis Kebutuhan Masukan

Perangkat lunak SHOR-GA hanya memiliki satu level akses, yaitu pengguna. Kebutuhan masukan dari user dibutuhkan untuk beberapa menu pada tab yang digunakan untuk keperluan yang berbeda, yaitu:

A. Tab Map

a. *Select Province* : Nama provinsi yang akan dicari.

b. Kota Sumber.

c. Kota Tujuan.

B. Tab Genetic Algorithms

a. *Max Edges Init Sols* : Maksimum inisialisasi sisi yang diproses.

b. *Init Pop Size* : Ukuran Populasi Awal.

c. *Size of Population*: Ukuran Populasi.

d. *Min Difference* : Selisih minimum.

e. *Number of Generation* : Banyak generasi.

f. *Crossover Probability* : Probabilitas *crossover*.

3. Analisis Kebutuhan Keluaran

Sedangkan kebutuhan keluaran yang dihasilkan oleh perangkat lunak SHOR-GA yaitu:

a. Jarak antar kota: jarak antar kota yang dihasilkan dari posisi koordinat kota.

b. Jalur terpendek yang dihasilkan beserta grafiknya.

3.1.3 Kebutuhan Antar Muka

Perancangan antar muka dilakukan dengan menggunakan komponen Java (Swing dan AWT) dan kakas Netbeans 5.0. Penggunaan komponen Java Swing, AWT dan kakas Netbeans 5.0 merupakan pilihan yang tepat untuk mengimplementasikan perangkat lunak, dengan tampilan yang indah dan memudahkan pengguna untuk menggunakan sistem, dan sifat orientasi obyek dan modularisasi dari Java juga membuat *programmer* lebih mudah untuk memisahkan tampilan dari kode, sehingga memudahkan pengembangan perangkat lunak.

3.1.4 Analisis Kebutuhan Perangkat Lunak

1. Sun Microsystem Java 2 Standard Edition SDK (J2SDK 1.6), dengan tambahan library JDOM (Java Document Object Model) untuk parser XML.
2. Kakas pemrograman menggunakan Netbeans 5.0.
3. Editor XML menggunakan Notepad++.
4. Perancangan UML menggunakan Rational Rose 2000, dan perancangan *flowchart* menggunakan Microsoft Visio 2003.
5. Desain menggunakan Corel Draw X3, Adobe Photoshop CS2, ACDSee 8.0 dan Microsoft Paint.

3.1.5 Analisis Kebutuhan Perangkat Keras

Penggunaan sistem komputer sebagai alat bantu dalam menyelesaikan tugas-tugas atau pekerjaan sudah bukan menjadi hal yang aneh, tapi merupakan suatu keharusan karena banyak kemudahan-kemudahan yang bisa diperoleh.

Komputer terdiri perangkat keras dan perangkat lunak. Perangkat lunak memberikan instruksi-instruksi kepada perangkat keras untuk melakukan suatu tugas tertentu.

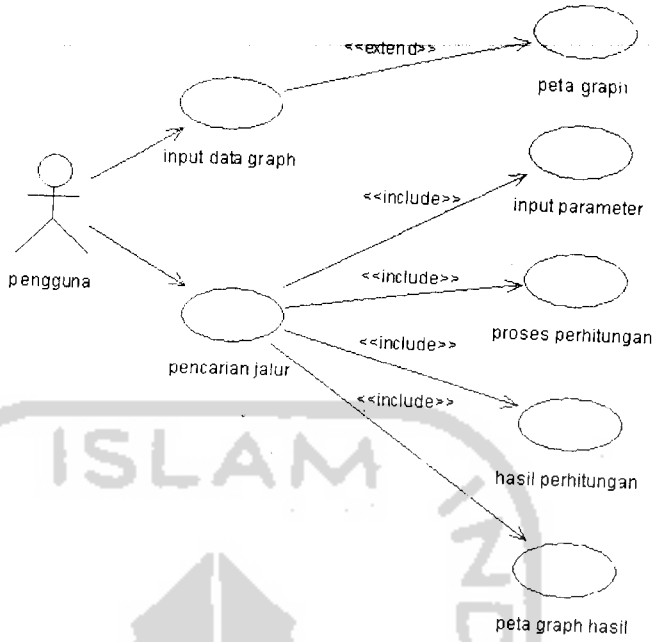
Penggunaan komputer sebagai alat bantu suatu kejadian yang benar-benar terjadi di kehidupan nyata yang sering digunakan, oleh karena itu penulis berusaha untuk membuat salah satu peristiwa atau kejadian yang terjadi di dunia nyata yaitu pencarian jalur terpendek dengan menggunakan komputer.

industri yang dibuat di bawah pengawasan *Object Management Group* (OMG). Untuk lebih menjelaskan perancangan aplikasi yang dibangun, digunakan 4 (empat) model diagram, yaitu: *use case diagram*, *class diagram*, *sequence diagram* dan *activity diagram*.

a. *Use Case Diagram*

Merupakan diagram yang bekerja dengan cara mendeskripsikan tipikal interaksi antara *user* (pengguna) sebuah sistem dengan suatu sistem tersendiri melalui sebuah cerita bagaimana sebuah sistem dipakai. *Use case diagram* terdiri dari sebuah aktor dan interaksi yang dilakukannya, aktor tersebut dapat berupa manusia, perangkat keras, sistem lain, ataupun yang berinteraksi dengan sistem.

Pada perangkat lunak SHOR-GA, *use case* menjelaskan tentang hubungan antara sistem dengan aktor. Hubungan ini dapat berupa *input* aktor ke sistem ataupun *output* ke aktor. *Use-case* merupakan dokumen naratif yang mendeskripsikan kasus-kasus atau kejadian-kejadian daripada aktor dalam menggunakan *system* untuk menyelesaikan sebuah proses. Gambar 3.1 menjelaskan perangkat lunak SHOR-GA dalam model *use-case diagram*:



Gambar 3.1 *Use Case Diagram* SHOR-GA

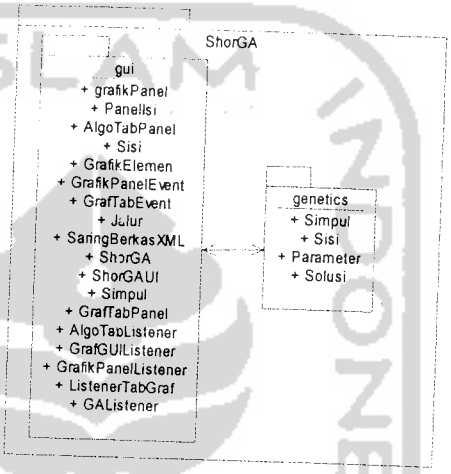
Pada *Use case diagram* di atas, *user* (pengguna perangkat lunak SHOR-GA) dapat melakukan aktivitas yang ada pada *use case*. Aktivitas yang terdapat pada *use case diagram* perangkat lunak SHOR-GA antara lain: input data graph (peta), pencarian jalur (input parameter, proses perhitungan, hasil perhitungan dan peta graf hasil).

b. *Class Diagram* (Diagram Kelas)

Class diagram digunakan untuk melakukan visualisasi struktur kelas-kelas dari suatu sistem dan merupakan tipe diagram yang paling banyak digunakan. *Class diagram* juga dapat memperlihatkan hubungan antar kelas dan penjelasan detail tiap-tiap kelas di dalam model desain (*logical view*) dari suatu sistem.

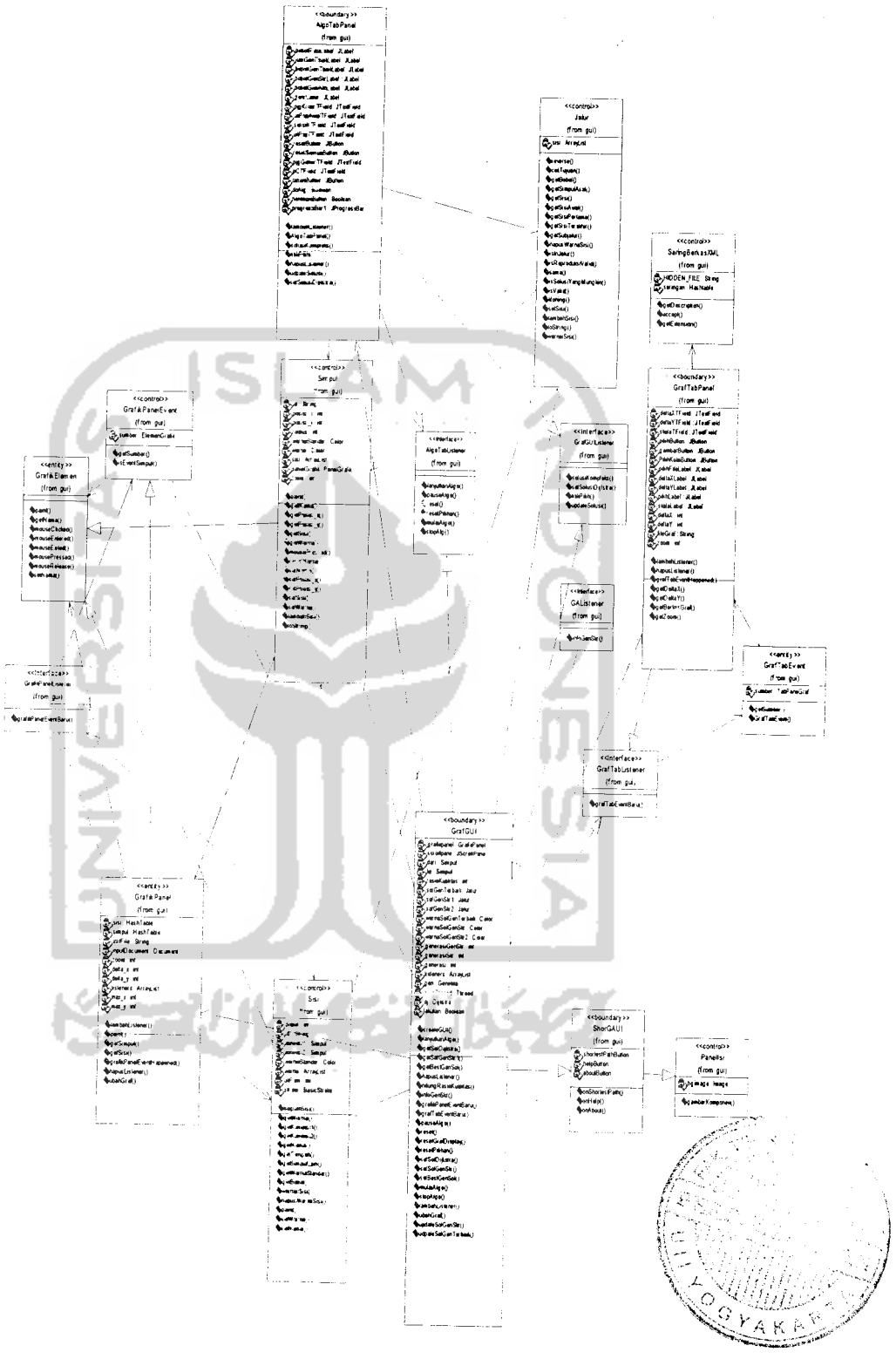
Selam proses desain, *class diagram* berperan dalam menangkap struktur dari semua kelas yang membentuk arsitektur sistem yang dibuat.

Berikut ini adalah akan dijelaskan *class diagram* yang digunakan untuk melakukan visualisasi struktur kelas-kelas yang terdapat dalam perangkat lunak SHOR-GA yang dibagi ke dalam 2 buah modul utama, yaitu modul *gui* dan modul *genetics*.



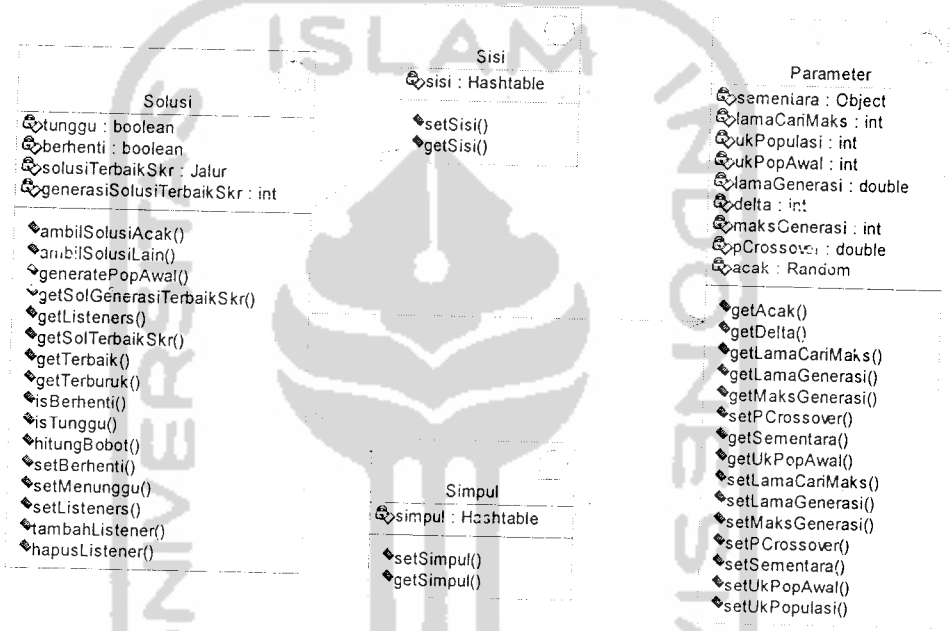
Gambar 3.2 *Class Diagram* modul *gui* dan *genetics* SHOR-GA

Gambar 3.2 di atas menunjukkan bahwa perangkat lunak SHOR-GA dirancang dengan menggunakan dua modul, yaitu modul *gui* dan modul *genetics*. Modul *gui* berfungsi sebagai pembangun banyak kelas graf, sedangkan modul *genetics* berfungsi untuk kelas-kelas parameter GA dan solusinya. Gambar 3.3 menunjukkan *Class Diagram modul gui* pada perangkat lunak SHOR-GA.



Gambar 3.3 *Class Diagram* modul gui SHOR-GA

Gambar 3.3 di atas menunjukkan modul *gui* dari perancangan *Class Diagram* perangkat lunak SHOR-GA. Pada gambar 3.3 ditunjukkan bahwa modul *gui* berisi kelas-kelas untuk perancangan graf, hingga perancangan panel dengan GUI (*Graphical User Interface*) untuk menu dan perhitungan. Gambar 3.4 menunjukkan *Class Diagram modul genetics* pada perangkat lunak SHOR-GA.



Gambar 3.4 *Class Diagram* modul *genetics* SHOR-GA

Gambar 3.4 di atas menunjukkan modul *genetics* dari perancangan *Class Diagram* perangkat lunak SHOR-GA. Pada gambar 3.4 ditunjukkan bahwa modul *genetics* berisi kelas-kelas untuk parameter GA, dan perhitungan solusi jalur terpendek.

c. *Sequence Diagram*

Sequence diagram digunakan untuk menjelaskan interaksi objek yang disusun dalam suatu urutan waktu. Diagram ini secara khusus berasosiasi dengan *use case*. *Sequence diagram* juga dapat memperlihatkan tahap demi tahap proses yang seharusnya terjadi untuk menghasilkan sesuatu di dalam *use case*.

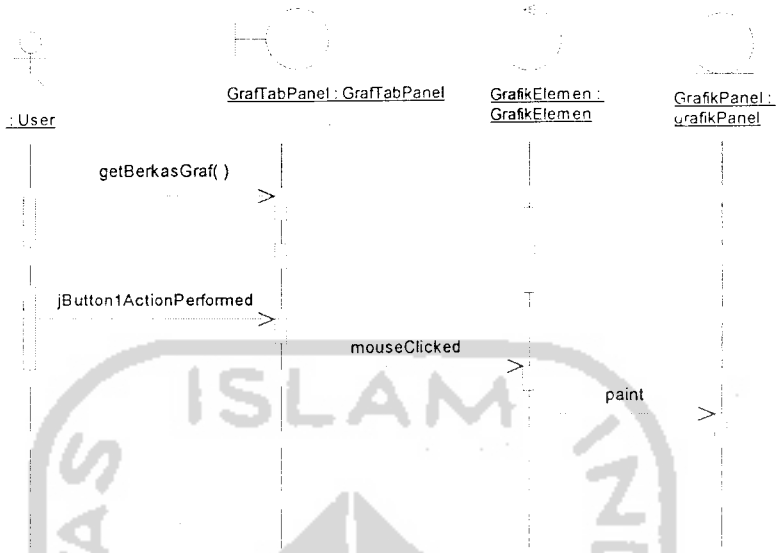
Pada bagian ini akan dijelaskan *sequence diagram* dari perangkat lunak SHOR-GA. Dalam *sequence diagram* ini menggambarkan interaksi antar objek pada sistem secara berurutan. Penjelasan *sequence diagram* akan dijelaskan berdasarkan atas modul-modul yang dibuat yaitu modul *genetics* dan modul *gui*.

i. *Sequence Diagram* gambar peta

Sequence Diagram gambar peta menunjukkan interaksi *user* dengan perangkat lunak untuk melakukan penggambaran peta. Objek yang berkaitan dengan *sequence* ini adalah sebagai berikut.

Aktor	:	*User*
ClassBoundary	:	GrafTabPanel
Clas Control	:	GrafikElemen
Clas Entity	:	GrafikPanel
Keterangan	:	1. *User* mengawali *sequence* ini dengan memanggil *method* jButton1ActionPerformed(). 2. GraphTabPanel melakukan instansiasi ke GrafikElemen dan memanggil *method* mouseClicked() untuk memproses *input* dari *user*. 3. GrafikElemen melakukan instansiasi ke GrafikPanel dan memanggil *method* paint() untuk menggambarkan peta.

Sequence diagram gambar peta dapat dilihat pada gambar 3.5.



Gambar 3.5 *Sequence diagram* gambar peta

ii. *Sequence Diagram* cari jalur terpendek

Sequence Diagram cari jalur terpendek menunjukkan interaksi *user* dengan perangkat lunak untuk melakukan pencarian jalur terpendek. Objek yang berkaitan dengan *sequence* ini adalah sebagai berikut.

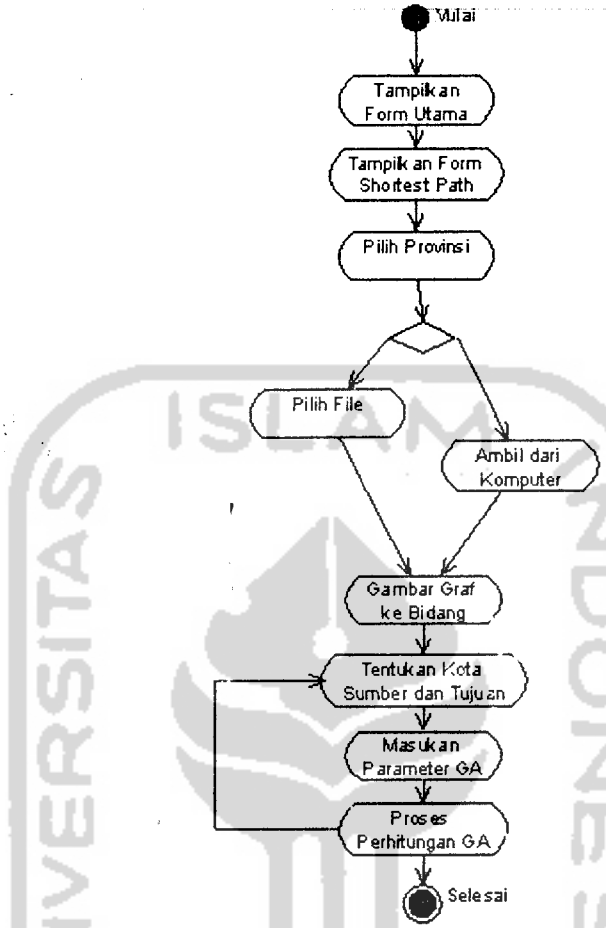
Aktor	:	*User*
Class Boundary	:	AlgoTabPanel
Class Control	:	Solusi
Class Entity	:	GrafGUI
Keterangan	:	1. *User* mengawali *sequence* ini dengan memanggil *method* startAlgo().
		2. AlgoTabPanel menginstansiasi Solusi untuk mendapatkan solGenTerbaikSkr
		3. Solusi menginstansiasi GrafGUI untuk mendapatkan solGenTerbaik

Sequence diagram `cari jalur terpendek` dapat dilihat pada gambar 3.6.

kegiatan mulai dari menggambarkan peta ke bidang, sampai pencarian jalur terpendek. Untuk urutan aktivitas dijelaskan sebagai berikut :

1. *User* memilih menu *Shortest Path* pada menu utama.
2. Sistem menampilkan form *Shortest Path.*
3. Selanjutnya *user* memilih provinsi, jika provinsi tidak ada di *combobox*, *user* dapat melakukan pencarian data xml di komputer.
4. User menekan tombol *Draw Map* untuk menggambarkan graf/peta ke bidang, kemudian mengisikan kota sumber dan kota tujuan.
5. User berpindah tab ke tab GA, kemudian mengisikan parameter GA, dan memulai algoritma dengan menekan tombol *Start Algorithm.*
6. Sistem menghasilkan keluaran berupa proses-proses GA.
7. User dapat mengulangi lagi dari langkah ke-5 untuk melakukan pencarian dengan data sama.
8. Kemudian sistem mengakhiri kegiatan ini.

Activity diagram pencarian jalur terpendek dapat dilihat pada gambar 3.7.



Gambar 3.7 *Activity diagram* pencarian jalur terpendek

2. Perancangan Algoritma Genetika

Perancangan GA pada perangkat lunak SHOR-GA terdiri dari perancangan parameter GA, dan perancangan alur kerja GA. Berikut adalah perancangan GA pada perangkat lunak SHOR-GA [SAP07a]:

a. Perancangan Kromosom dan Struktur Data

Langkah awal dalam perancangan GA untuk menentukan jalur terpendek adalah representasi kromosom. Sebagai contoh pada provinsi 24 kota, perancangan kromosomnya terdapat pada tabel 3.1 berikut:

Tabel 3.1 Perancangan Kromosom

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
2	4	1	3	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
3	1	4	5	14	2	6	7	8	9	10	11	12	13	15	16	17	18	19	20	21	22	23	24
4	2	3	1	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
5	3	6	12	22	1	2	4	7	8	9	10	11	13	14	15	16	17	18	19	20	21	23	24
6	5	7	1	2	3	4	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
7	6	8	1	2	3	4	5	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
8	7	9	11	22	1	2	3	4	5	6	10	12	13	14	15	16	17	18	19	20	21	23	24
9	8	10	22	1	2	3	4	5	6	7	11	12	13	14	15	16	17	18	19	20	21	23	24
10	9	11	23	1	2	3	4	5	6	7	8	12	13	14	15	16	17	18	19	20	21	22	24
11	8	10	1	2	3	4	5	6	7	9	12	13	14	15	16	17	18	19	20	21	22	23	24
12	5	13	24	22	1	2	3	4	6	7	8	9	10	11	14	15	16	17	18	19	20	21	23
13	12	14	15	19	1	2	3	4	5	6	7	8	9	10	11	16	17	18	20	21	22	23	24
14	9	13	15	1	2	4	5	6	7	8	9	10	11	12	16	17	18	19	20	21	22	23	24
15	16	14	22	1	2	3	4	5	6	7	8	9	10	11	12	13	17	18	19	20	21	23	24
16	15	17	18	20	1	2	3	4	5	6	7	8	9	10	11	12	13	14	19	21	22	23	24
17	16	21	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	18	19	20	22	23	24
18	16	19	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	17	20	21	22	23	24
19	13	15	18	24	1	2	3	4	5	6	7	8	9	10	11	12	14	16	17	20	21	22	23
20	16	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	17	18	19	21	22	23	24
21	17	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	18	19	20	22	23	24
22	5	8	9	12	1	2	3	4	6	7	10	11	13	14	15	16	17	18	19	20	21	23	24
23	10	1	2	3	4	5	6	7	8	9	11	12	13	14	15	16	17	18	19	20	21	22	24
24	12	18	1	2	3	4	5	6	7	8	9	10	11	13	14	15	16	17	19	20	21	22	23

Pada tabel 3.1 di atas terdapat 24 kemungkinan kromosom yang dapat terjadi pada perancangan algoritma genetika untuk provinsi Riau. Kolom berwarna putih menunjukkan jalur kromosom yang dilewati dari titik sumber ke titik tujuan, sedangkan yang berwarna gelap adalah sisa kromosom yang tidak dilewati. Untuk kasus jalur terpendek, kromosom yang diperhitungkan hanya sampai titik tujuan. Jika titik tujuan telah terpenuhi, maka selanjutnya diisi dengan kota yang belum dilewati.

Tabel 3.1 Populasi Awal

	Kromosom	Panjang Jalur	Fitness
1	20-16-15-13	380	0.00263
2	20-16-15-14-13	510	0.00196
3	20-16-18-19-13	515	0.00194
4	20-16-15-13	380	0.00263
5	20-16-15-13	380	0.00263
6	20-16-15-19-13	520	0.00192
7	20-16-18-19-15-13	675	0.00148
8	20-16-15-13	380	0.00263
9	20-16-15-13	380	0.00263
10	20-16-15-19-13	520	0.00192
11	20-16-15-19-13	520	0.00192
12	20-16-15-13	380	0.00263
13	20-16-18-19-24-12-13	665	0.00150
14	20-16-18-19-15-13	675	0.00148
15	20-16-18-19-15-13	675	0.00148
16	20-16-15-13	380	0.00263
17	20-16-18-19-24-12-13	665	0.00150
18	20-16-15-14-13	510	0.00196
19	20-16-15-13	380	0.00263
20	20-16-15-13	380	0.00263

Setelah populasi awal terbentuk, maka langkah selanjutnya untuk melakukan seleksi adalah menentukan nilai probabilitas *fitness* sebanyak ukuran populasi. Nilai probabilitas *fitness* dapat dicari sebagai berikut [KUS05]:

$$TotalFitness = \sum_{k=1}^{PopSize} F_k, k = (1,2,..Popsize) \tag{3.1}$$

$$p_k = F_k / TotalFitness \tag{3.2}$$

Pada tabel 3.1 di atas, dapat dihitung nilai Total Fitness adalah :

$$TotalFitness = (F_1 + F_2 + F_3 + F_4 + F_5 + F_6 + F_7 + F_8 + F_9 + F_{10} + F_{11} + F_{12} + F_{13} + F_{14} + F_{15} + F_{16} + F_{17} + F_{18} + F_{19} + F_{20}) \quad (3.4)$$

$$= \mathbf{0.04273}$$

Nilai probabilitas *fitness* untuk generasi pertama untuk peta provinsi Riau dapat dilihat pada tabel 3.2.

Tabel 3.2 Probabilitas *Fitness* generasi pertama

	p_k	q_k
1	0.06154	0.06154
2	0.04586	0.10740
3	0.04540	0.15280
4	0.06154	0.21434
5	0.06154	0.27588
6	0.04493	0.32081
7	0.03463	0.35544
8	0.06154	0.41698
9	0.06154	0.47852
10	0.04493	0.52345
11	0.04493	0.56838
12	0.06154	0.62992
13	0.03510	0.66502
14	0.03463	0.69965
15	0.03463	0.73428
16	0.06154	0.79582
17	0.03510	0.83092
18	0.04586	0.87678
19	0.06154	0.93832
20	0.06154	1.0000

Langkah selanjutnya untuk melakukan seleksi adalah menentukan bilangan acak sebanyak ukuran populasi. Bilangan acak pertama dapat dilihat pada tabel 3.3.

Tabel 3.3 Bilangan acak untuk seleksi pada generasi pertama

	Bilangan
1	0,053039
2	0,293448
3	0,191108
4	0,977403
5	0,285298
6	0,291933
7	0,049475
8	0,223642
9	0,135376
10	0,293143
11	0,517823
12	0,651604
13	0,014671
14	0,815414
15	0,867205
16	0,649939
17	0,367285
18	0,658849
19	0,988222
20	0,385257

Setelah ditentukan bilangan acak sebanyak ukuran populasi, maka bilangan tersebut di seleksi. Hasil seleksi pada generasi pertama dapat dilihat pada tabel 3.4.

Tabel 3.4 Hasil seleksi generasi pertama

	Kromosom	Fitness	Ke-
1	20-16-15-13	0.00263	1
2	20-16-15-19-13	0.00192	6
3	20-16-15-13	0.00263	4
4	20-16-15-13	0.00263	20
5	20-16-15-19-13	0.00192	6
6	20-16-15-19-13	0.00192	6
7	20-16-15-13	0.00263	1
8	20-16-15-13	0.00263	5
9	20-16-18-19-13	0.00194	3
10	20-16-15-19-13	0.00192	6
11	20-16-15-19-13	0.00192	10
12	20-16-18-19-24-12-13	0.00150	13
13	20-16-15-13	0.00263	1
14	20-16-18-19-24-12-13	0.00150	17
15	20-16-15-14-13	0.00196	18
16	20-16-18-19-24-12-13	0.00150	13
17	20-16-15-13	0.00263	8
18	20-16-18-19-24-12-13	0.00150	13
19	20-16-15-13	0.00263	20
20	20-16-15-13	0.00263	8

d. Proses Rekombinasi

Rekombinasi adalah proses untuk menyilangkan dua kromosom sehingga membentuk kromosom baru yang harapannya lebih baik dari pada induknya. Untuk melakukan rekombinasi, langkah awal yang dilakukan adalah menyiapkan bilangan acak sejumlah ukuran populasi. Bilangan acak untuk rekombinasi pada generasi pertama dapat dilihat pada tabel 3.5.

Tabel 3.5 Bilangan acak untuk rekombinasi generasi pertama

	Bilangan
1	0,691593
2	0,186397
3	0,105565
4	0,306714
5	0,733123
6	0,646295
7	0,47203
8	0,598315
9	0,702275
10	0,813434
11	0,386613
12	0,215199
13	0,508527
14	0,49977
15	0,313639
16	0,265695
17	0,541782
18	0,511524
19	0,422662
20	0,183394

Setelah dicari bilangan acak untuk rekombinasi, maka dapat dicari induk untuk rekombinasi. Induk untuk rekombinasi pada generasi pertama didapatkan dengan mencari nilai bilangan acak yang lebih kecil dari nilai probabilitas crossover. Daftar induk untuk rekombinasi generasi pertama dapat dilihat pada tabel 3.6.

Tabel 3.6 Induk untuk rekombinasi generasi pertama

	Kromosom	Fitness	v
1	20-16-15-19-13	0.00192	v_2'
2	20-16-15-13	0.00263	v_3'
3	20-16-15-13	0.00263	v_4'
4	20-16-15-13	0.00263	v_7'
5	20-16-15-19-13	0.00192	v_{11}'
6	20-16-18-19-24-12-13	0.00150	v_{12}'
7	20-16-18-19-24-12-13	0.00150	v_{14}'
8	20-16-15-14-13	0.00196	v_{15}'
9	20-16-18-19-24-12-13	0.00150	v_{16}'
10	20-16-15-13	0.00263	v_{19}'
11	20-16-15-13	0.00263	v_{20}'

Silangkan (v_2' dengan v_3'), (v_4' dengan v_7'), (v_{11}' dengan v_{12}'), (v_{14}' dengan v_{15}') dan (v_{19}' dengan v_{20}'). Hasil persilangan dapat dilihat pada tabel 3.7.

Tabel 3.7 Hasil rekombinasi pada generasi pertama

v	Kromosom	v	Kromosom
v_1'	20-16-15-13	v_{11}"	20-16-18-19-24-12-13
v_2"	20-16-15-13	v_{12}"	20-16-15-19-13
v_3"	20-16-15-19-13	v_{13}'	20-16-15-13
v_4"	20-16-15-13	v_{14}"	20-16-15-14-13
v_5'	20-16-15-19-13	v_{15}"	20-16-18-19-24-12-13
v_6'	20-16-15-19-13	v_{16}'	20-16-18-19-24-12-13
v_7"	20-16-15-13	v_{17}'	20-16-15-13
v_8'	20-16-15-13	v_{18}'	20-16-18-19-24-12-13
v_9'	20-16-18-19-13	v_{19}"	20-16-15-13
v_{10}'	20-16-15-19-13	v_{20}"	20-16-15-13

e. Mutasi

Mutasi adalah proses penambahan nilai acak yang sangat kecil dengan probabilitas rendah pada variabel keturunan. Untuk melakukan mutasi, langkah awal yang dilakukan adalah menyiapkan bilangan acak sejumlah ukuran populasi. Bilangan acak untuk mutasi pada generasi pertama dapat dilihat pada tabel 3.8.

Tabel 3.8 Bilangan acak untuk mutasi generasi pertama

	Bilangan
1	0,836547
2	0,491103
3	0,352292
4	0,452879
5	0,104273
6	0,895753
7	0,904721
8	0,985736
9	0,884342
10	0,733687
11	0,234109
12	0,849063
13	0,693136
14	0,112484
15	0,180653
16	0,687485
17	0,989665
18	0,274317
19	0,817097
20	0,039304

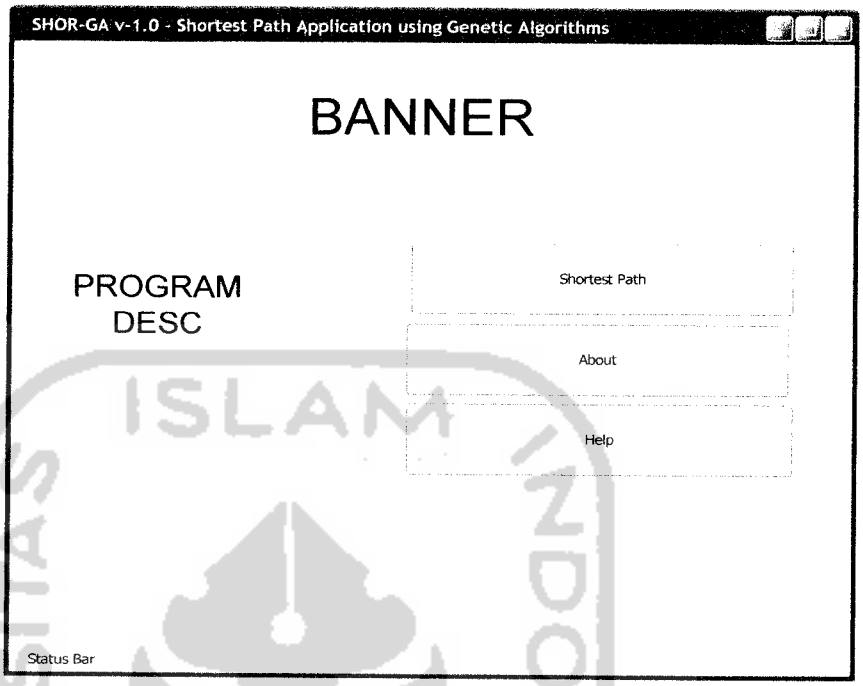
sehingga didapatkan jalur terpendek antara kota Teluk Merbau (kota ke-20) dan Tebing Tinggi (kota ke-13) adalah jalur Teluk Merbau (20) – Sedinginan (16) – Duri (15) – Tebing Tinggi (13) dengan bobot 380 km.

3. Perancangan Antar Muka

Rancangan antar muka dari perangkat lunak SHOR-GA terdiri dari antar muka halaman utama, halaman *Shortest Path* tab *Map*, halaman *Shortest Path* tab *Genetic Algorithms*, halaman *about* dan halaman *help*.

a. Halaman utama

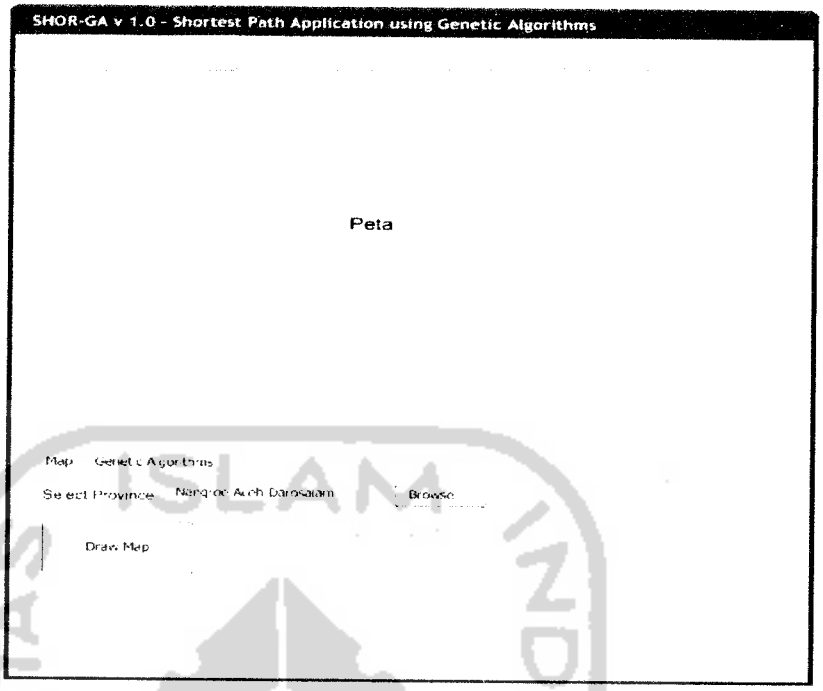
Halaman utama merupakan halaman yang muncul pertama kali ketika *user* menggunakan aplikasi. Pada perangkat lunak SHOR-GA, halaman utama terdiri atas 3 menu, yaitu menu *Shortest Path* (untuk menentukan jalur terpendek), menu *about* (tentang program dan *programmer*) dan menu *help* (manual dan dokumentasi program). Gambar 3.10 menunjukkan rancangan antar muka halaman utama SHOR-GA.



Gambar 3.10 Rancangan antar muka halaman utama SHOR-GA

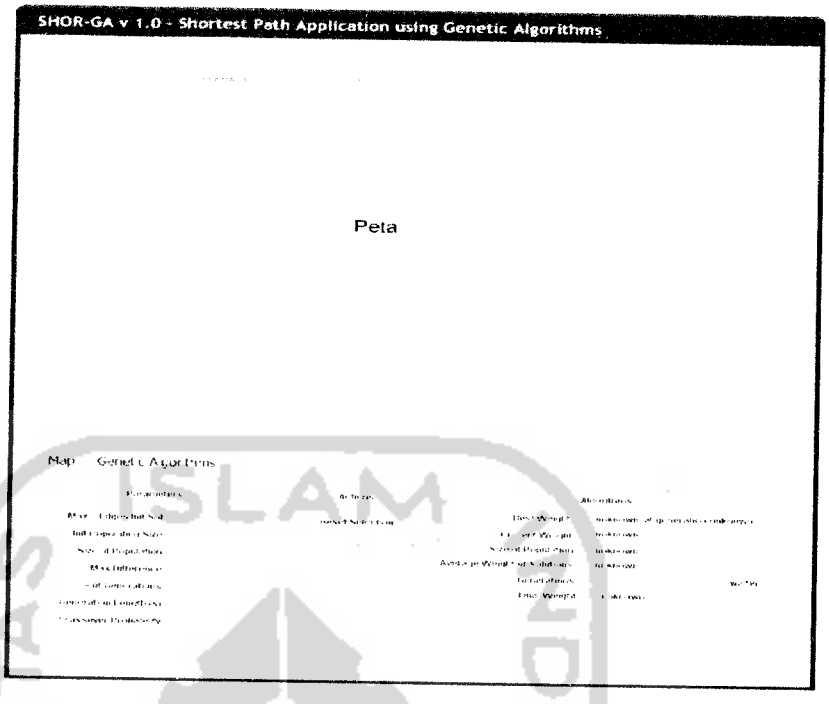
b. Halaman *Shortest Path*

Halaman *Shortest Path* merupakan halaman inti dari perangkat lunak SHOR-GA. Halaman ini terdiri dari 2 tab, yaitu tab *Map* untuk mendefinisikan graf (peta) dan tab *Genetic Algorithms* untuk mendefinisikan algoritma genetika. Gambar 3.11 menunjukkan rancangan antar muka tab *Map* halaman *Shortest Path* SHOR-GA.



Gambar 3.11 Rancangan antarmuka tab *Map* halaman *Shortest Path* perangkat lunak SHOR-GA

Pada rancangan antarmuka tab *Map*, terdapat tombol untuk pemilihan provinsi dan penggambaran peta ke bidang. Fungsi tab *map* adalah untuk menggambarkan peta dari file XML yang tersedia. Gambar 3.12 menunjukkan rancangan antarmuka tab *Genetic Algorithms* halaman *Shortest Path* SHOR-GA.

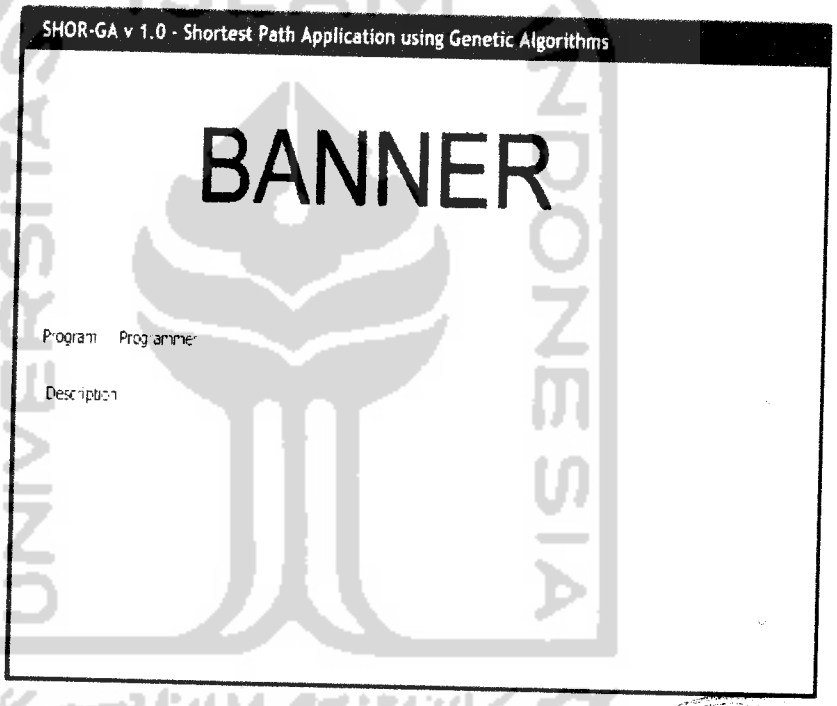


Gambar 3.12 Rancangan antarmuka tab *Genetic Algorithms* halaman *Shortest Path* perangkat lunak SHOR-GA

Pada rancangan antarmuka tab *Genetic Algorithms*, terdapat beberapa untuk pengisian parameter algoritma genetika, tombol untuk menjalankan proses serta hasil akhir proses. Fungsi tab *Genetic Algorithms* adalah untuk mencari jalur terpendek menggunakan algoritma genetika.

c. **Halaman *about***

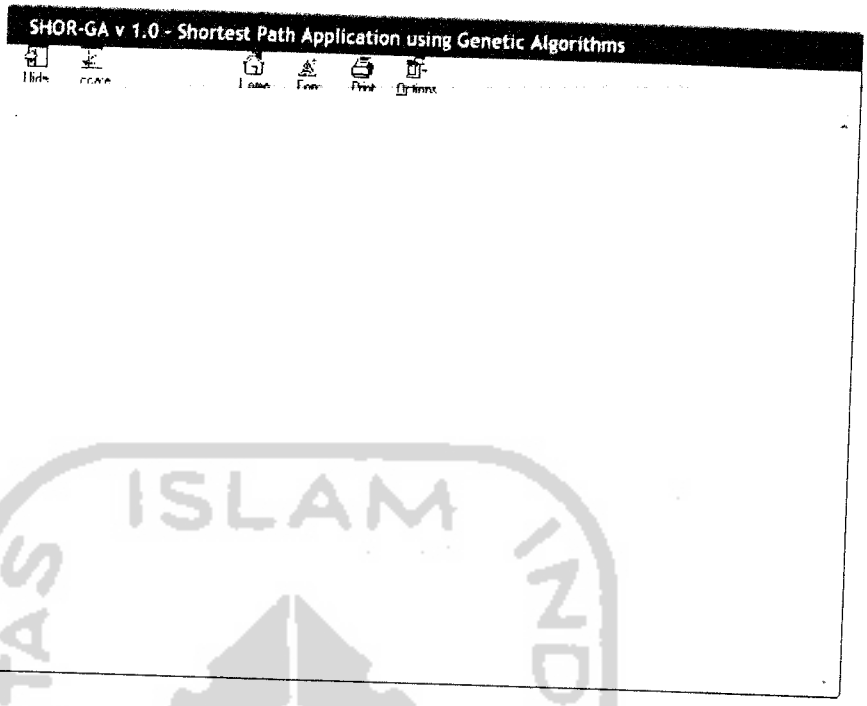
Halaman *about* adalah halaman yang berisikan data tentang SHOR-GA dan programmer. *User* dapat menghubungi programmer jika terdapat kesulitan dalam menjalankan atau mengembangkan program. Gambar 3.13 menunjukkan rancangan antarmuka halaman *about.*



Gambar 3.13 Rancangan antarmuka halaman *about* SHOR-GA

d. **Halaman *help***

Halaman *help* adalah halaman yang berisikan manual dan cara penggunaan perangkat lunak SHOR-GA. Gambar 3.14 menunjukkan rancangan antarmuka halaman *help* SHOR-GA.



Gambar 3.14 Rancangan antarmuka halaman *help* SHOR-GA

3.3 Implementasi Perangkat Lunak

Pada tahap implementasi, sistem dioperasikan dalam keadaan sesungguhnya. Tujuan implementasi ini adalah untuk mengetahui apakah sistem yang dibuat benar dan sesuai dengan perancangan yang disiapkan.

3.3.1 Batasan Implementasi

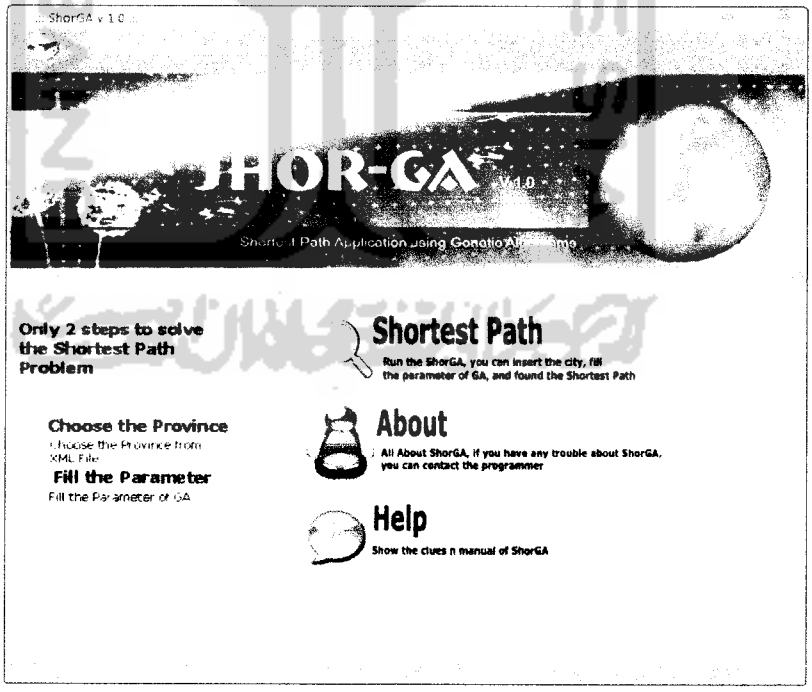
Pada implementasi perangkat lunak SHOR-GA akan dijelaskan bagaimana sistem ini bekerja, dengan memberikan tampilan form-form dan tampilan output yang dibuat. Implementasi terdiri dari implementasi antarmuka dan implementasi prosedural.

3.3.2 Implementasi antarmuka

Implementasi antarmuka merupakan tampilan antarmuka sistem sesuai dengan perancangannya. Implementasi antar muka dari perangkat lunak SHOR-GA terdiri dari antar muka halaman utama, halaman *Shortest Path* tab *Map*, halaman *Shortest Path* tab *Genetic Algorithms*, halaman *about* dan halaman *help*.

a. Halaman utama

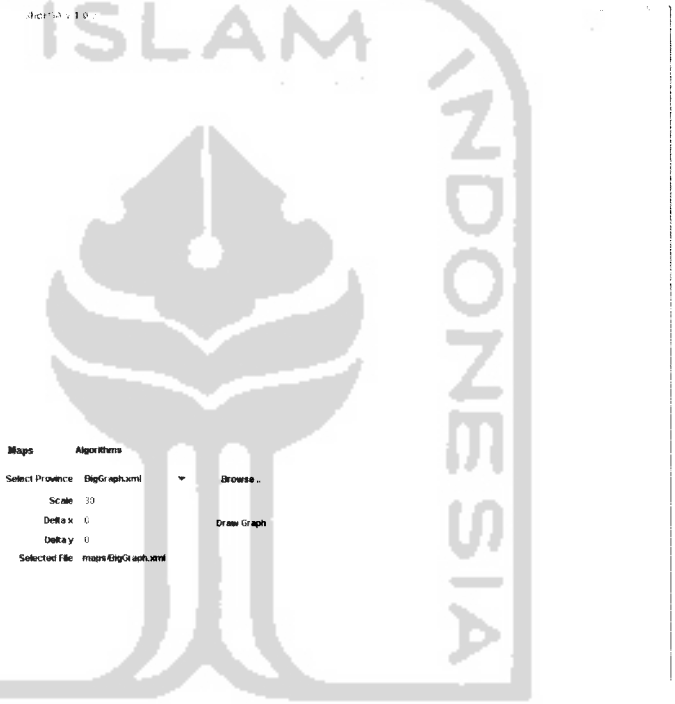
Halaman utama merupakan halaman yang muncul pertama kali ketika *user* menggunakan aplikasi. Pada perangkat lunak SHOR-GA, halaman utama terdiri atas 3 menu, yaitu menu *Shortest Path* (untuk menentukan jalur terpendek), menu *about* (tentang program dan *programmer*) dan menu *help* (manual dan dokumentasi program). Gambar 3.15 menunjukkan implementasi antar muka halaman utama SHOR-GA.



Gambar 3.15 Implementasi Antar Muka Halaman Utama SHOR-GA

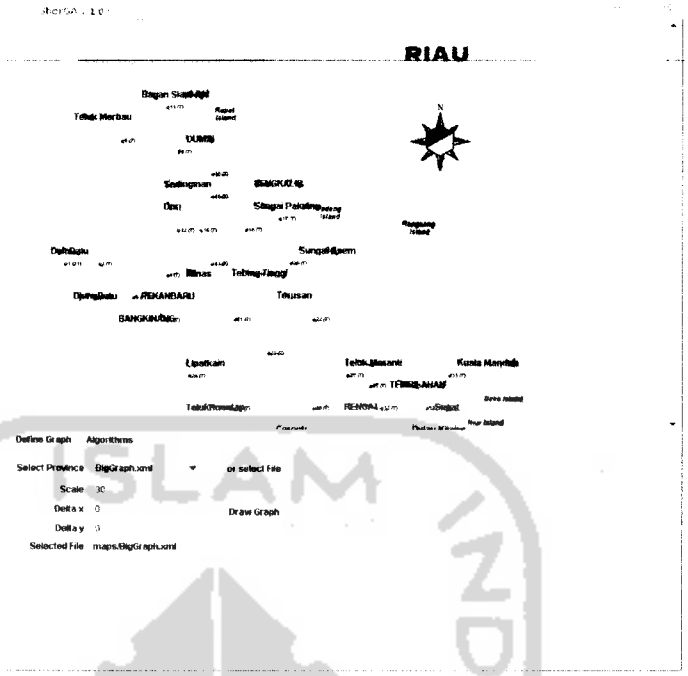
b. Halaman *Shortest Path*

Halaman *Shortest Path* merupakan halaman inti dari perangkat lunak SHOR-GA. Halaman ini terdiri dari 2 tab, yaitu tab *Map* untuk mendefinisikan graf (peta) dan tab *Algorithms* untuk mendefinisikan algoritma genetika. Gambar 3.16 menunjukkan implementasi antar muka tab *Map* halaman *Shortest Path* SHOR-GA.



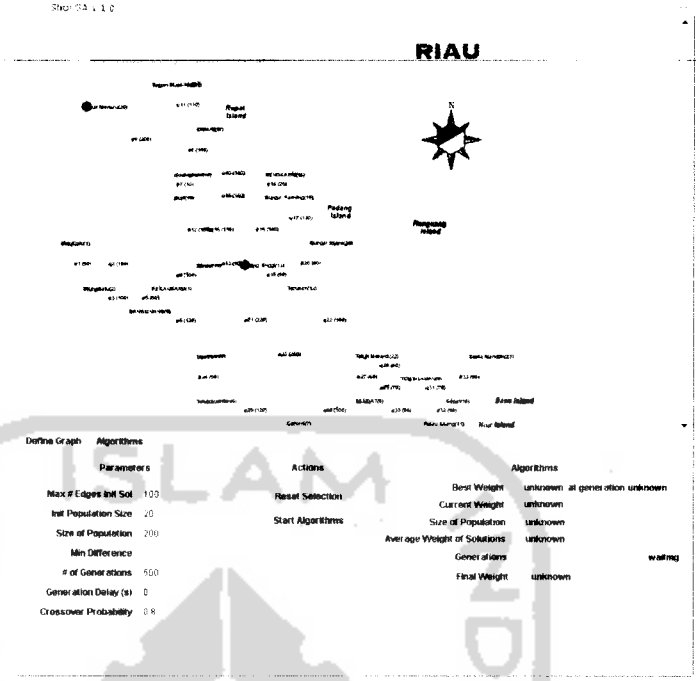
Gambar 3.16 Implementasi antarmuka tab *Map* halaman *Shortest Path* perangkat lunak SHOR-GA

Pada implementasi antarmuka tab *Map*, terdapat tombol untuk pemilihan provinsi dan penggambaran peta ke bidang. Fungsi tab *Map* adalah untuk menggambarkan peta dari file XML yang tersedia. Gambar 3.17 menunjukkan implementasi antarmuka tab *Map* halaman *Shortest Path* SHOR-GA setelah data digambarkan.



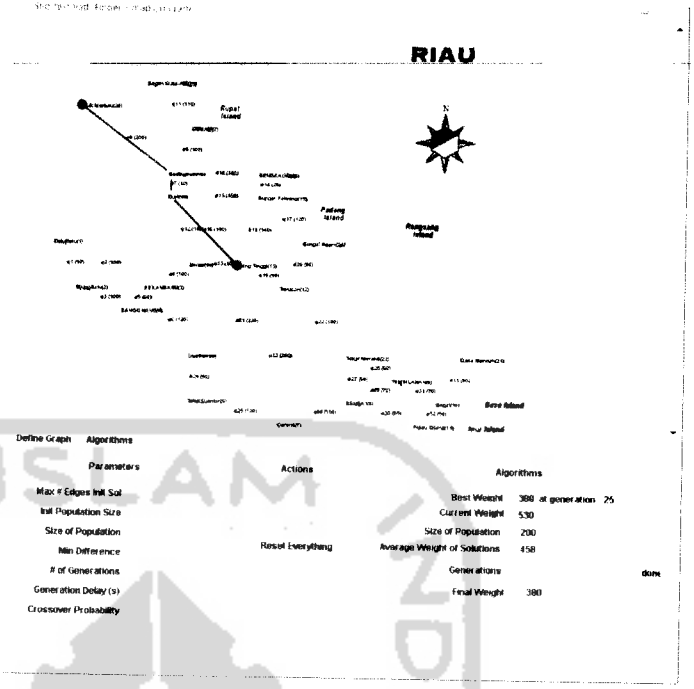
Gambar 3.17 Implementasi antarmuka tab *Map* halaman *Shortest Path* perangkat lunak SHOR-GA setelah data digambarkan

Sedangkan gambar 3.18 menunjukkan implementasi antarmuka tab *Algorithms* halaman *Shortest Path* SHOR-GA setelah data kota asal dan kota tujuan dimasukkan.



Gambar 3.18 Antarmuka tab *Algorithms* halaman *Shortest Path* perangkat lunak SHOR-GA setelah data kota dimasukkan

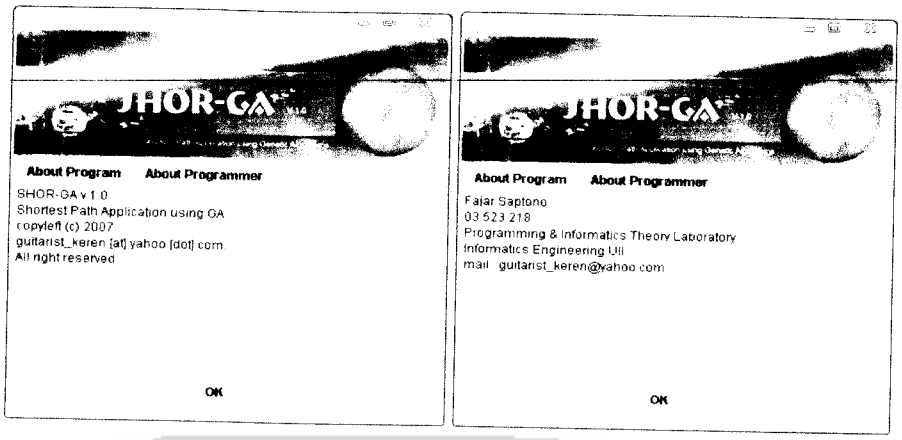
Pada implementasi antarmuka tab *Algorithms*, terdapat beberapa untuk pengisian parameter algoritma genetika, tombol untuk menjalankan proses serta hasil akhir proses. Fungsi tab *Algorithms* adalah untuk mencari jalur terpendek menggunakan algoritma genetika. gambar 3.19 menunjukkan implementasi antarmuka tab *Algorithms* halaman *Shortest Path* SHOR-GA setelah pemrosesan selesai.



Gambar 3.19 Antarmuka tab *Algorithms* halaman *Shortest Path* perangkat lunak SHOR-GA setelah pemrosesan selesai.

c. **Halaman *about***

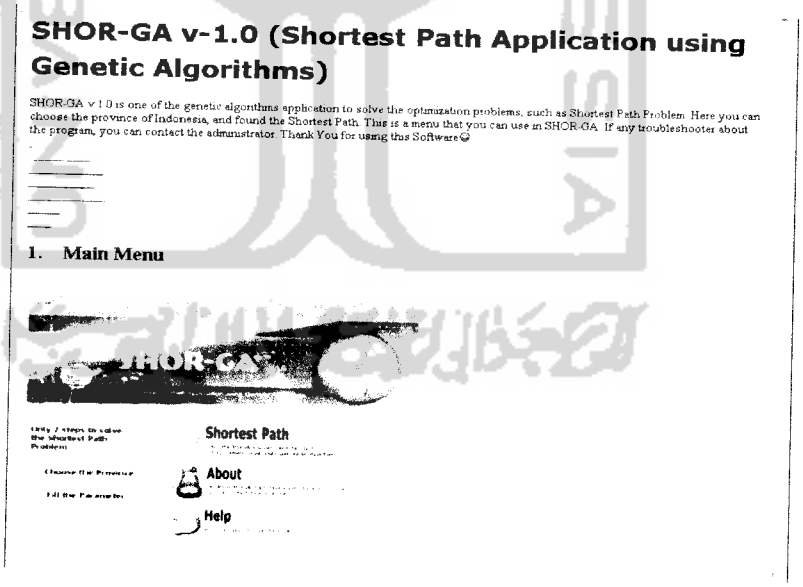
Halaman *about* adalah halaman yang berisikan data tentang SHOR-GA dan programmer. *User* dapat menghubungi programmer jika terdapat kesulitan dalam menjalankan atau mengembangkan program. Gambar 3.13 menunjukkan implementasi antarmuka halaman *about*.



Gambar 3.20 Implementasi Antarmuka halaman *about* SHOR-GA

d. **Halaman *help***

Halaman *help* adalah halaman yang berisikan manual dan cara penggunaan perangkat lunak SHOR-GA. Gambar 3.21 menunjukkan implementasi antarmuka halaman *help* SHOR-GA.



Gambar 3.21 Implementasi antarmuka halaman *help* SHOR-GA

3.3.3 Implementasi Prosedural

Implementasi prosedural merupakan implementasi pada pemrograman sistem. Namun yang dijelaskan di sini adalah pemrograman GA, yang merupakan proses inti dalam perangkat lunak SHOR-GA.

1. *Method* generatePopAwal

Method ini berfungsi untuk meng-*generate* populasi awal dari atribut *ukPopAwalTField.* Populasi dibangun dengan membangun acak jalur dimulai dari simpul sumber. Berikut adalah implementasi *method* generatePopAwal.

```
public ArrayList<Jalur> generatePopAwal(Simpul dari,Simpul ke){
  ArrayList<Jalur> popAwal = new ArrayList<Jalur>
    (this.getUkPopulasi());
  while (popAwal.size()<=this.getUkPopAwal()){
    Jalur solAwal=new Jalur();
    Simpul posisiSkr=dari;
    while(solAwal.getSisi().size()<=this.getLamaCariMaks()){
      ArrayList<Sisi> sisiAwal=posisiSkr.getSisi();
      Sisi sisiBerikut=null;
      int sisiPilihan=Genetics.getAcak().nextInt
        (sisiAwal.size());
      sisiBerikut=sisiAwal.get(sisiPilihan);
      solAwal.tambahSisi(sisiBerikut);
      posisiSkr=sisiBerikut.getSimpulLain(posisiSkr);
      if(sisiBerikut.isiSimpul(ke)){
        popAwal.add(solAwal);
        break;
      }else if(solAwal.getSisi().size()==this.getLamaCariMaks()){
              break;
            }
      if(this.isTunggu()){
        synchronized(this.getSementara()){
          try{
            this.getSementara().wait();
          }catch(InterruptedException e){
             e.printStackTrace();
           }
         }
       }
     if(this.isBerhenti()){
       break;
     }
   }
   if (this.isBerhenti()) {
     break;
   }
}
```

```
    return popAwal;
  }
```

2. *Method* reproduksi

Method ini berfungsi untuk mengeksekusi langkah reproduksi dari GA. Berikut adalah implementasi *method* reproduksi.

```
public Jalur reproduksi(Jalur indukReproduksi,Simpul dari,Simpul
ke){
  Jalur induk=indukReproduksi.kloning();
  int ukInduk=induk.getSisi().size();
  int temp1=-1;
  int temp2=-1;
  while(temp1==-1||temp2==-1||temp1==temp2){
    temp1 = Genetics.acak.nextInt(ukInduk);
    temp2 = Genetics.acak.nextInt(ukInduk);
  }
  if(temp1<temp2){
    Sisi sisi2=induk.getSisi().remove(temp2);
    Sisi sisi1=induk.getSisi().remove(temp1);
    induk.getSisi().add(temp1,sisi2);
    induk.getSisi().add(temp2,sisi1);
  }else{
    int gantikan=temp2;
    temp2=temp1;
    temp1=gantikan;
    Sisi sisi2=induk.getSisi().remove(temp2);
    Sisi sisi1=induk.getSisi().remove(temp1);
    induk.getSisi().add(temp1,sisi2);
    induk.getSisi().add(temp2,sisi1);
  }
  if(induk.isReproduksiValid(dari,ke)){
    return induk;
  }else{
    return indukReproduksi.kloning();
  }
}
```

3. *Method* rekombinasi

Method ini berfungsi untuk mengeksekusi langkah rekombinasi dari algoritma genetika. Berikut adalah implementasi *method* rekombinasi.

```
public ArrayList<Jalur> rekombinasi(Simpul dari,Simpul ke,
  ArrayList<Jalur> solusi){
  Jalur ayah=this.ambilSolusiAcak(solusi);
  int percobaanAyah = 0;
  while(ayah.getSisi().size()<2&&percobaanAyah++<
```

```
    this.getMaksReproduksi()){
     ayah=this.ambilSolusiAcak(solusi);
    }
    if(ayah.getSisi().size()<2){
      ArrayList<Jalur> hasil=new ArrayList<Jalur>();
      Jalur anak=this.reproduksi(ayah,dari,ke);
      hasil.add(anak);
      return hasil;
    }
    Jalur genAwal=null;
    Jalur ibu=null;
    int percGenBaru=0;
    while(ibu==null&&percGenBaru++<this.getMaksReproduksi()){
      genAwal=this.jalurPotong(ayah);
      Simpul simpulAkhirGen=genAwal.getSimpulAsal(dari);
      int percGen=0;
      while(ibu==null&&percGen++<this.getMaksReproduksi()){
        Jalur probIbu=this.ambilSolusiLain(solusi,ayah);
        Sisi sisiAkhirGenAwal=genAwal.getSisiAwal();
        if(probIbu.getSisi().size()<3){
          continue;
        }
        Jalur selisihProbIbu=probIbu.kloning();
        selisihProbIbu.getSisi().remove(0);
        selisihProbIbu.getSisi().remove(
        selisihProbIbu.getSisi().size()-1);
        if(selisihProbIbu.getSisi().contains(sisiAkhirGenAwal)){
          if(probIbu.cekTujuan(dari,simpulAkhirGen,
            genAwal.getSisiAwal())){
            ibu=probIbu;
            }
        }
      }
    }
    if(ibu==null){
      return null;
    }
    if(Math.abs(ayah.getBobot()-ibu.getBobot())<=this.getDelta()){
      ArrayList<Jalur> hasil=new ArrayList<Jalur>();
      if(ayah.getBobot()-ibu.getBobot()>0){
        Jalur anak=this.reproduksi(ayah,dari,ke);
        hasil.add(anak);
      }else{
         Jalur anak=this.reproduksi(ibu,dari,ke);
         hasil.add(anak);
       }
     return hasil;
    }else{
      Jalur anakPertama=this.pasangan(genAwal,ibu);
      Jalur genKedua=ibu.getSubjalur(genAwal.getSisiAwal());
      Jalur anakKedua=this.pasangan(genKedua,ayah);
      ArrayList<Jalur> anaknya=new ArrayList<Jalur>(2);
      anaknya.add(anakPertama);
      anaknya.add(anakKedua);
      return anaknya;
  }
```

4. *Method* hitungBobot

Method ini berfungsi untuk menghitung bobot dari jalur yang ada. Berikut adalah implementasi *method* hitungBobot.

```
public Hashtable<Jalur,Integer> hitungBobot(ArrayList<Jalur>
  jalurnya){
  Hashtable<Jalur,Integer> bobotnya=new Hashtable<Jalur,Integer>
    (jalurnya.size());
  for (Jalur p:jalurnya){
    bobotnya.put(p,p.getBobot());
  }
  return bobotnya;
}
```

5. *Method* getTerbaik

Method ini berfungsi untuk mendapatkan jalur terbaik dari suatu graf. Berikut adalah implementasi *method* getTerbaik.

```
public Jalur getTerbaik(Hashtable<Jalur,Integer>bobotnya){
  int bobotTerbaik=Integer.MAX_VALUE;
  Jalur jalurTerbaik=null;
  Set<Jalur> kuncinya=bobotnya.keySet();
  for(Jalur p:kuncinya){
    int bobotIni=bobotnya.get(p);
    if(bobotIni<bobotTerbaik){
      bobotTerbaik=bobotIni;
      jalurTerbaik=p;
      }
    }
  return jalurTerbaik;
}
```

6. *Method* ambilSolusiAcak

Method ini berfungsi untuk mengambil solusi secara acak. Berikut adalah implementasi *method* ambilSolusiAcak.

```
public Jalur ambilSolusiAcak(ArrayList<Jalur> solusi){
  int batas=solusi.size();
  int indeksSol=Genetics.acak.nextInt(batas);
  return solusi.get(indeksSol);
}
```

7. *Method* hitungReratapopulasi

Method ini berfungsi untuk menghitung nilai rata-rata dari seluruh populasi. Berikut adalah implementasi *method* hitungReratapopulasi.

```
int hitungNilaiReratapopulasi(Hashtable<Jalur,Integer> bobotnya){
  int totalWeight=0;
  for(Integer bobotIni:bobotnya.values()){
    totalWeight+=bobotIni;
  }
  return totalWeight/bobotnya.size();
}
```

BAB IV
HASIL dan PEMBAHASAN

4.1 Pengujian Sistem

Sebelum sistem diterapkan pada lingkungan sebenarnya, maka diperlukan evaluasi/pengujian terhadap berbagai aspek. Pengujian ini dilakukan agar kemungkinan terjadinya kesalahan/*error* pada sistem dapat diidentifikasi sejak awal. Pada tahap pengujian dan analisis perangkat lunak SHOR-GA, dilakukan pembandingan antara kebenaran serta kesesuaian program dengan kebutuhan sistem.

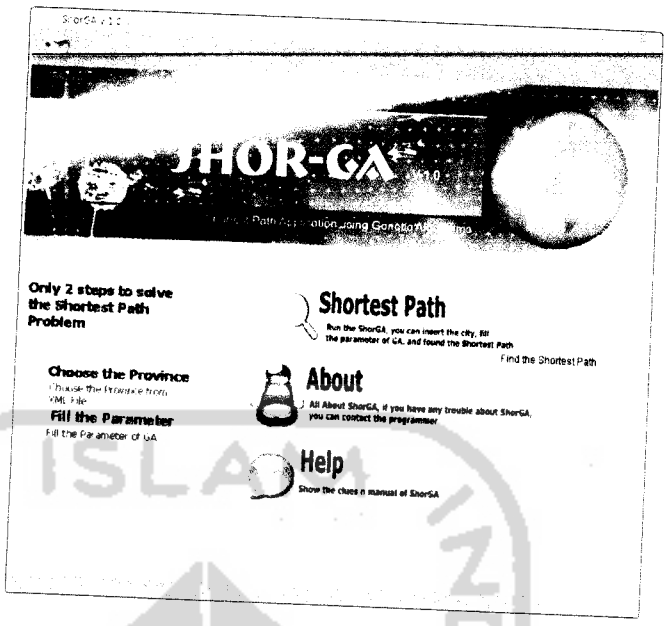
Pengujian dilakukan dengan mengisikan input ke dalam form-form yang sudah dijelaskan pada BAB III. Metode pengujian dengan melakukan pengujian normal dan pengujian tidak normal.

4.11 Pengujian Normal

Dilakukan dengan memberikan masukan yang menurut spesifikasi awal dan pengetahuan yang diijinkan. Setelah diberikan masukan yang sesuai, dilakukan analisis pembandingan antara kebenaran masukan serta kesesuaian program dengan kebutuhan sistem.

1. Menu Utama

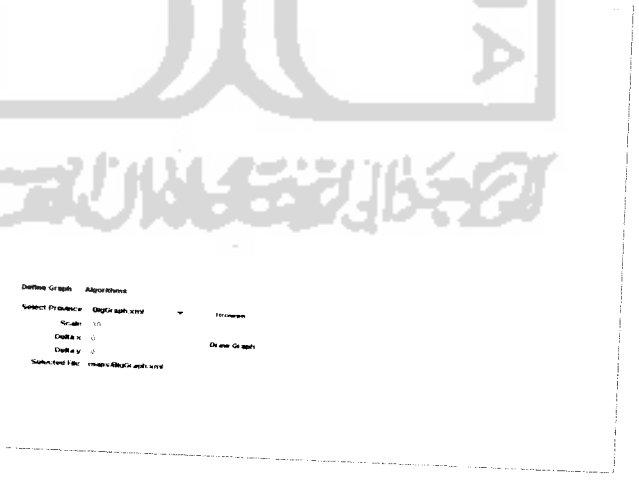
   Menu ini adalah menu untuk tampilan utama. Menu terdiri dari tombol pilihan *Shortest Path, about* dan *help*. Tampilan pada sistem dapat dilihat pada gambar 4.1.



Gambar 4.1 Tampilan Menu Utama

2. Menu *Shortest Path*

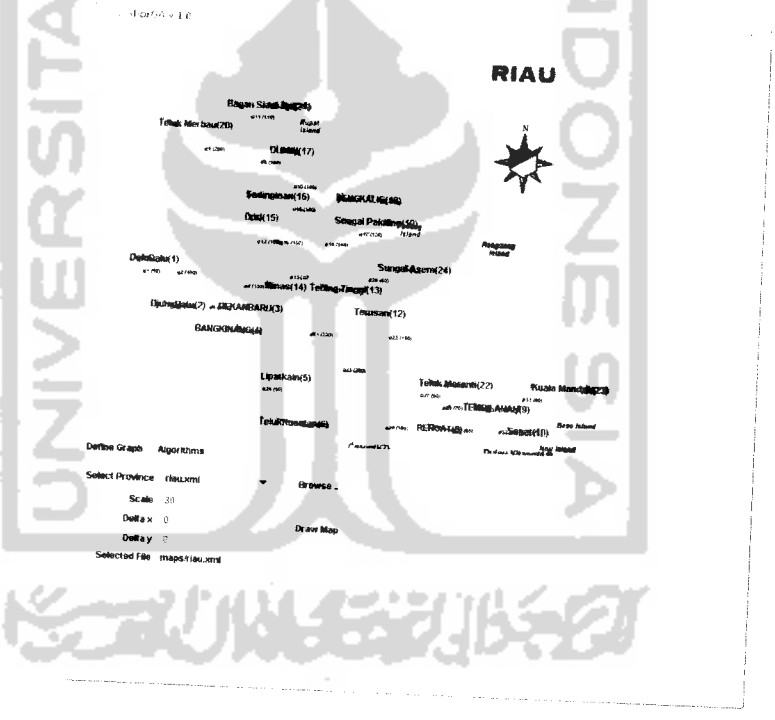
Menu ini adalah menu untuk melakukan pencarian jalur terpendek. Tampilan pada sistem dapat dilihat pada gambar 4.2.



Gambar 4.2 Tampilan Menu *Shortest Path*

3. Masukan *Select Province*

Untuk melakukan pencarian jalur terpendek, langkah pertama yang harus dilakukan oleh pengguna adalah melakukan pemilihan provinsi. Setelah pemilihan, pengguna dapat menekan tombol *Draw Map*, sehingga akan tampak peta daerah yang dipilih. Misalkan pengguna memilih file Riau.xml, maka Tampilan pada sistem dapat dilihat pada gambar 4.3.



Gambar 4.3 Tampilan Masukan *Select Province*

4. Masukan Parameter Algoritma Genetika, Kota Asal dan Kota Tujuan

Setelah pemilihan provinsi, pengguna dapat memasukkan parameter algoritma genetika dengan memindah *tab* ke tab *Genetic Algorithms*. Setelah itu pengguna dapat memilih kota asal dan kota tujuan untuk

mulai menghitung jalur terpendek. Misalkan dimasukkan parameter GA, kota asal dan kota tujuan sebagai berikut.

Ukuran populasi: 200

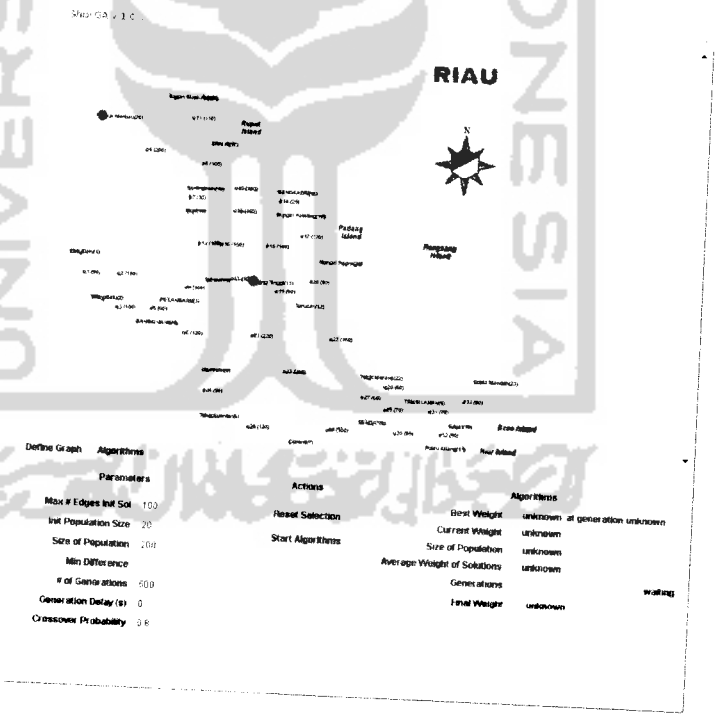
Banyak generasi: 500

Probabilitas Crossover: 0.8

Kota Asal: Teluk Merbau

Kota Tujuan: Tebing Tinggi

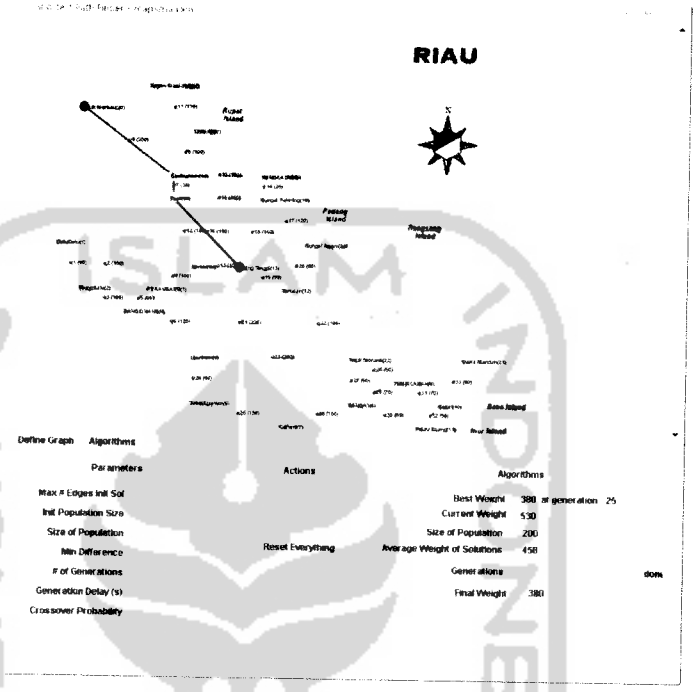
Maka tampilan pada sistem dapat dilihat pada gambar 4.4.



Gambar 4.4 Tampilan masukan parameter GA, kota asal dan tujuan

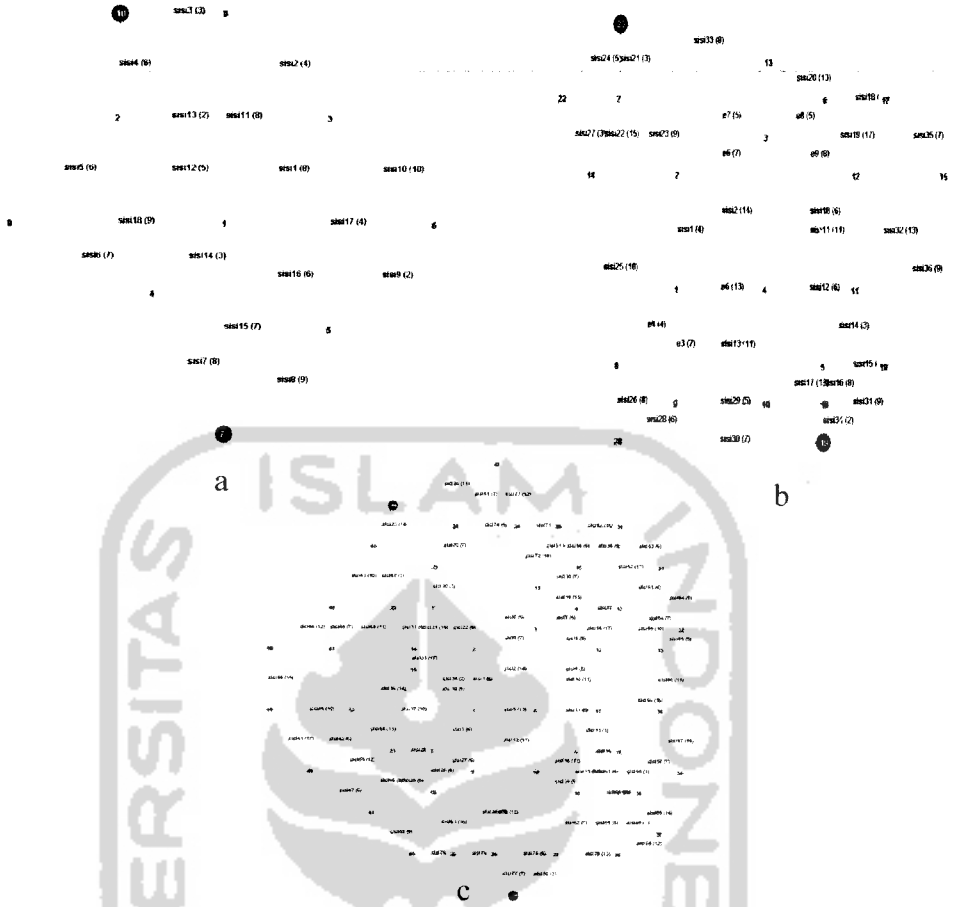
Setelah pengguna memasukkan parameter GA, posisi kota asal dan posisi kota tujuan, kemudian pengguna dapat menekan tombol *Start*

Algorithm untuk memulai GA melakukan proses. Setelah proses selesai, maka peta akan menampilkan tampilan seperti gambar 4.5



Gambar 4.5 Tampilan hasil pemrosesan jalur terpendek

Pengujian diatas merupakan langkah-langkah pengujian normal. Untuk mengetahui tingkat keakuratan algoritma genetika dilakukan pengujian untuk data dengan 3 data graf, yaitu data provinsi (graf) dengan kota (simpul) kecil (gambar 4.6a), simpul menengah (gambar 4.6b) dan simpul besar (gambar 4.6c) [SAP07b].



Gambar 4.6 Graf untuk pengujian

Pada gambar 4.6 di atas, terdapat 3 data graf, yaitu graf dengan simpul kecil, menengah dan besar. Data tersebut digunakan untuk melakukan pengujian guna mengukur tingkat keakuratan algoritma genetika. Sebagai data, tabel 4.1 menunjukkan perhitungan bobot untuk ketiga graf di atas menggunakan algoritma konvensional (algoritma Dijkstra). Sebagai catatan, algoritma konvensional lebih akurat dalam perhitungan.

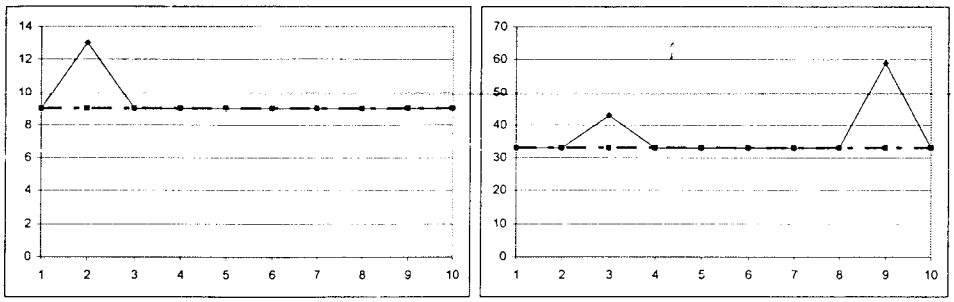
Tabel 4.1 Perhitungan Bobot Menggunakan Algoritma Dijkstra

Jenis Kota	Jumlah Simpul	Bobot Optimum
Kecil	10	9
Menengah	23	33
Besar	50	47

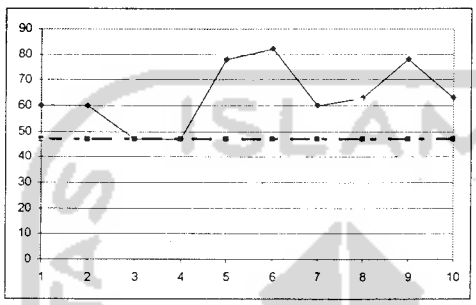
Tabel 4.2 Perhitungan Bobot Menggunakan Algoritma Genetika

	10 simpul	23 simpul	50 simpul
1	9	33	60
2	13	33	60
3	9	43	47
4	9	33	47
5	9	33	78
6	9	33	82
7	9	33	60
8	9	33	63
9	9	59	78
10	9	33	63

Pada tabel 4.1 di atas, telah ditunjukkan bobot optimum yang dihitung menggunakan algoritma dijkstra, sedangkan perhitungan menggunakan GA ditunjukkan pada tabel 4.2. Gambar 4.7 berikut menunjukkan perbandingan hasil keakuratan antara metode konvensional dengan metode algoritma genetika untuk kota (simpul) kecil (4.7a), simpul menengah (4.7b) dan simpul besar (4.7c). Garis lurus menunjukkan metode konvensional, sedangkan garis putus-putus menunjukkan metode algoritma genetika.



a b



c

Keterangan:
a. Kota Kecil
b. Kota Menengah
c. Kota Besar
------- Algoritma Dijkstra

_____ Algoritma Genetika

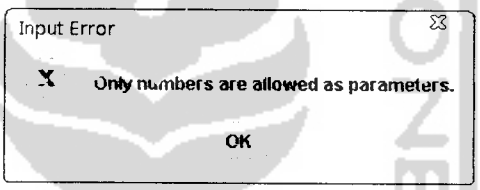
Gambar 4.7 Perbandingan keakuratan algoritma genetika dengan algoritma dijkstra

Berdasarkan grafik pada gambar 4.7 di atas dapat disimpulkan bahwa untuk parameter genetika yang sama pada tiap kota dan tiap pengujian, semakin banyak jumlah simpul, semakin berkurang tingkat keakuratan algoritma genetika, karena itu tingkat keakuratan algoritma genetika untuk menentukan jalur terpendek akan sangat berpengaruh pada jumlah simpul yang akan diperhitungkan dan parameter algoritma genetika yang dimasukkan.

1.1.2 Pengujian Tidak Normal

Dilakukan dengan memberikan masukan dengan spesifikasi yang tidak diijinkan sehingga sistem akan memberikan reaksi lain. Reaksi sistem berupa berupa peringatan *(alert)* atau penanganan kesalahan *(error handling)*.

Penanganan kesalahan ini dilakukan untuk menangkap error yang terjadi ketika salah satu field pada form inputan kosong atau ketidaksesuaian tipe data. Contoh penanganan kesalahan input terdapat pada pemasukan *field* yang membutuhkan data berupa angka, contohnya pada pengisian parameter GA. Jika pada *field* tidak dimasukkan angka, maka akan muncul peringatan seperti pada gambar 4.6.



Gambar 4.7 Peringatan jika *field* diisi selain angka

BAB V
SIMPULAN dan SARAN

1.1 Simpulan

Dari hasil penelitian dan pembahasan yang telah dilakukan, dapat diambil kesimpulan bahwa perangkat lunak SHOR-GA:

1. Mampu menentukan jalur terpendek antar kota dengan tampilan *user friendly*.
2. Dengan bantuan GA mempercepat waktu pemrosesan dibandingkan algoritma konvensional, akan tetapi hasilnya belum tentu akurat.
3. Tingkat keakuratan GA bergantung pada banyak simpul dan penentuan parameter GA.

1.2 Saran

Saran-saran untuk pengembangan *software* ke depannya berdasarkan kesimpulan yang diperoleh antara lain :

1. Untuk pengembangan sistem selanjutnya, cakupan bahasan bisa diperluas tidak hanya antar kota, namun bisa mencakup antar wilayah di kota-kota besar.
2. Penyajian data akan lebih baik jika digabung dengan SIG (Sistem Informasi Geografis), dan dibuat perbandingannya dengan algoritma konvensional sehingga informasi yang disajikan akan lebih lengkap.

3. Diharapkan dalam pengembangan sistem selanjutnya dapat melakukan perbandingan *Shortest Path Problem* dengan metode heuristik lainnya, sehingga dapat ditemukan metode mana yang lebih akurat.