PENGAPLIKASIAN ALGORITMA GENETIKA DALAM MENENTUKAN JALUR JALAN OPTIMAL WILAYAH KOTA PARIAMAN DENGAN LINTASAN TERPENDEK (*SHORTEST PATH)*

Rida Fadila[1] , Eka Sabna[2] , Erdisna[3]
[1,3] Universitas Putra Indonesia "YPTK" Padang
Jl Raya Lubuk Begalung Padang Sumbar
[2]STMIK Hang Tuah Pekanbaru
Jl Mustafa Sari No 5 Pekanbaru Riau
Email : [2]es3jelita@yahoo.com

ABSTRAK

Algoritma Genetika adalah teknik pencarian dan optimasi yang terinspirasi oleh prinsip genetik dan seleksi alam (teori evolusi Darwin).Algoritma ini digunakan untuk mendapatkan solusi yang tepat untuk permasalahan optimasi dengan satu variabel atau multi variabel.

Permasalahan *Travelling Salesman Problem* merupakan salah satu persoalan optimasi kombinatorial. TSP merupakan persoalan yang sulit bila dipandang dari sudut komputasinya. Beberapa metode telah digunakan untuk memecahkan persoalan tersebut. Dan algoritma genetika merupakan solusi dalam menentukan perjalanan terpendek yang melalui kota lainnya hanya sekali dan kembali ke kota asal keberangkatan.

Pada algoritma genetika, teknik pencarian dilakukan sekaligus atas sejumlah solusi yang dikenal dengan istilah populasi. Individu yang terdapat dalam satu populasi disebut dengan istilah kromosom. Algoritma genetika ini terdiri dari beberapa prosedur utama yaitu prosedur seleksi, *crossover*, mutasi dan elitisme. Algoritma genetika dirancang menjadi suatu program dengan menggunakan Matlab 7.9 untuk penyelesaian permasalahan tersebut.

Kata Kunci : Algoritma Genetika, TSP, Optimasi,Populasi, Kromosom

ABSTRACT

Genetic algoritms are search and optimation technique inspired by the principles of genetic and natural selection (Darwin's theory of evolution). This algoritm is used to get a proper solution to problem of optimization of a single or multiple variables.

Problems Travelling Salesman Problem is one of combinatorial optimization problems. TSP is a difficult problem computation when viewed from the point of computational. Several methods have been used to solve the problem. And genetic algoritm is the solution in determining shortest journey to through other city only once and return to the original departure city.

In genetic algoritm, search techniques performed well on a number of solutions known as population. Individual contained in a population referred to as chromosomes.Genetic algorithm consists of several main procedures that the procedure of selection, crossover, mutation and elitism.Genetic algorithms are designed to be a program by using Matlab 7.9 for solving these problems.

***Keywords**: Genetic Algorithm, TSP, Optimation, Population, Cromossom.*

1. PENDAHULUAN

1.1 Latar Belakang

Masalah dalam menentukan rantaian terpendek diantara pasangan *node* (titik) tertentu dalam suatu *graph* telah banyak menarik perhatian. Persoalan dirumuskan sebagai kasus khusus dari persoalan aliran biaya minimal dan algoritma efisien yang tersedia untuk menghitung lintasan terpendek dan biaya minimum.*Shortest Path* yang diperoleh akan meminimumkan fungsi linear khusus dari *Path* seperti jarak, waktu dan biaya dihadapi selama melakukan perjalanan. Perumusan persoalan ini akan menjadi salah satu kegunaan dari lintasan dengan jarak (waktu) diminimumkan terhadap biaya yang dianggarkan.

Beberapa metode algoritma yang telah dikembangkan untuk menyelesaikan persoalan jalur terpendek diantaranya Algoritma Djikstra, Algoritma Floyd-Warshall dan Algoritma Bellman-Ford. Algoritma ini dapat diselesaikan dengan cepat jika kota-kota yang akan dikunjunginya sedikit. Seiring dengan itu muncul permasalahan bagaimana menentukan jalur terpendek jika terdapat banyak jalur alternatif ke kota tujuan dengan mempertimbangkan efisiensi dan waktu sehingga diperlukan ketepatan dalam menentukan jalur terpendek antar suatu kota. Semakin banyak alternatif jalur ke kota tujuan, semakin rumit cara untuk menghitung jalur terpendek. Untuk itu diperlukan metode/cara yang handal untuk dapat menentukan jalur terpendek dari kota asal ke kota tujuan sehingga diperoleh solusi yang terbaik.

Penggunaan metode AI *(Artificial Intelligent)* atau kecerdasan buatan dalam perhitungan jalur terpendek merupakan salah satu solusi untuk dapat menyelesaikan masalah dengan jalur yang banyak dan rumit.Metode AI merupakan bagian dari ilmu komputer yang mempelajari bagaimana membuat mesin (komputer) dapat melakukan pekerjaan seperti dan sebaik manusia bahkan bisa lebih baik daripada yang dilakukan manusia.

Pada tahun 70-an muncul sebuah algoritma baru yang dikenal dengan Algoritma Genetika *(Genetic Algoritm,GA)* yang merupakan salah satu cabang dari AI. Algoritma Genetika ini diperkenalkan oleh John Holland dari University of Michigan yang kemudian dipopulerkan oleh salah satu muridnya yaitu David Goldberg, sehingga Algoritma Genetika mulai digunakan secara luas ke berbagai bidang, termasuk untuk memecahkan permasalah-permasalahan optimasi.

Berdasarkan uraian diatas penulis bermaksud membangun sebuah sistem untuk membantu dalam mengambil suatu keputusan. Konsep rancangan aplikasi tersebut dituangkan dalam sebuah penelitian dengan judul : "**PENGAPLIKASIAN ALGORITMA GENETIKA DALAM MENENTUKAN JALUR JALAN OPTIMAL WILAYAH KOTA PARIAMAN DENGAN LINTASAN TERPENDEK *(SHORTEST PATH)*"**

Berkaitan dengan latar belakang diatas, maka dapat dirumuskan masalah yang dihadapi adalah bagaimana menentukan lintasan terpendek dengan menggunakan algoritma genetika sehingga dicapai suatu solusi yang terbaik.

Adapun tujuan dari penelitian ini adalah sebagai berikut : Mewujudkan Aplikasi Algoritma Genetik dalam menentukan Jalur Jalan Optimal Wilayah Kota Pariaman Dengan Lintasan Terpendek (Shortest Path)

Agar pembahasan penelitian tidak mengambang dan dapat selalu terarah terhadap permasalahan yang dihadapi, maka diberikan batasan-batasan khusus terhadap pembahasan penelitian tersebut. Adapun batasan-batasan khusus yang diberikan adalah :

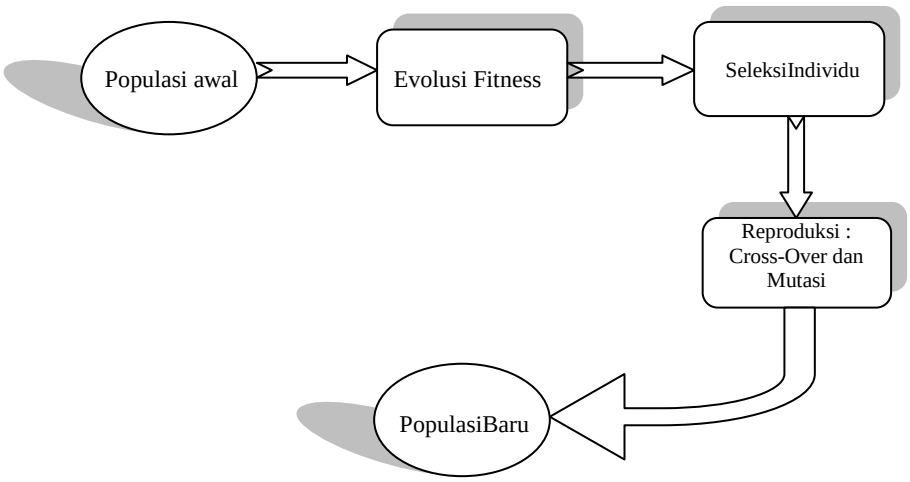
1. Kasus yang dibahas adalah pencarian lintasan terpendek (*shortest path problem*) dari permasalahan *Travelling Salesman Problem* yang diselesaikan dengan Algoritma Genetika.
2. Beberapa hal yang harus ada untuk menyelesaikan TSP ini antara lain jumlah kota yang nantinya akan direpresentasikan sebagai kromosom dan letak kota yang bisa dinyatakan dalam bentuk koordinat sehingga bisa diperoleh jarak antar kota.
3. Jumlah kota yang diteliti adalah 10 kota dengan *graph* yang saling terhubung.

Penelitian ini menawarkan penyelesaian yang lebih mudah dalam perhitungan untuk pencarian jalur terpendek.

1.2 Konsep Dasar Algoritma Genetika

a. Siklus Algoritma Genetika

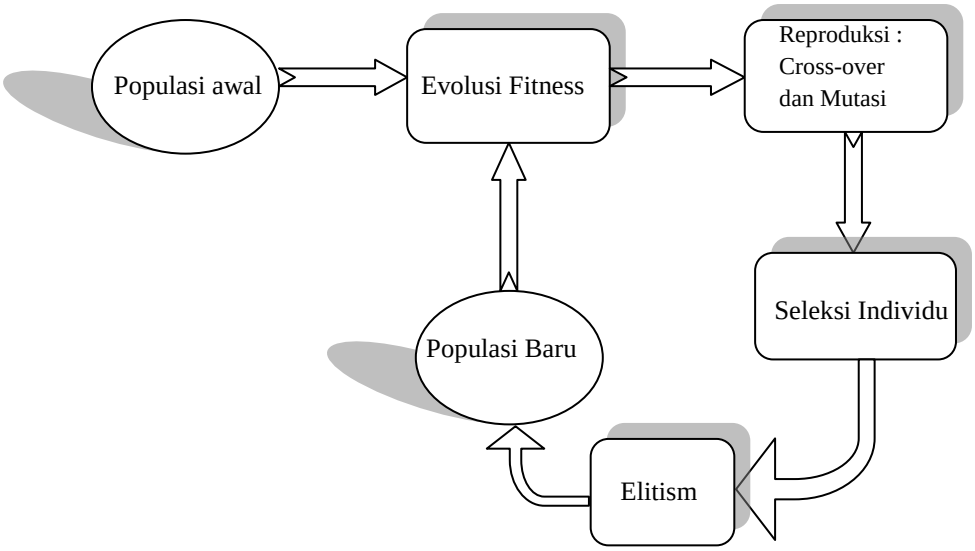
Siklus dari algoritma genetika pertama kali dikenalkan oleh David Goldberg dimana gambar siklus tersebut adalah sebagai berikut :



Gambar 1 Siklus Algoritma Genetika

Siklus ini kemudian diperbaiki oleh beberapa ilmuwan yang mengembangkan algoritma genetika, yaitu Zbigniew Michalewicz dengan menambahkan operator elitism dan membalik proses seleksi setelah proses reproduksi.

Berikut adalah gambar siklus algoritma genetika yang telah diperbaiki tersebut.



Gambar 2 Perbaikan Siklus Algoritma Genetika

Pada gambar diatas diilustrasikan diagram alir penggunaan probabilitas mutasi pada proses mutasi. Proses yang diilustrasikan tersebut adalah cara mudah melakukan mutasi. Proses mutasi dilakukan tidak harus seperti pada proses tersebut. Proses yang lain bisa dengan melakukan mutasi pada gen sebanyak probabilitas mutasi*jumlah gen, dumana posisi gen yang akan dilakukan mutasi dipilih secara acak.

b. **Sekilas Tentang Lintasan Terpendek *(Shortest Path)***

Shortest Path Problem (SPP) dikenal sebagai salah satu kelompok permasalahan jaringan yang banyak menarik perhatian peneliti sejak beberapa dekade terdahulu. SPP dideskripsikan sebagai persoalan untuk menentukan lintasan yang harus dilalui ketika mengunjungi suatu kota hingga diperoleh jarak tempuh terpendek. Perkembangan selanjutnya menunjukkan bahwa SPP telah menjadi salah satu persoalan dunia nyata.Beberapa diantaranya adalah perencanaan proyek *(project scheduling)*, manajemen keuangan *(cash flow management)*, sistem komunikasi *(communication system)*, penentu rute kendaraan *(transportation routing)*, manajemen proyek *(project management)*, karenanya pengembangan metode yang efisien untuk menyelesaikan SPP menjadi sangat penting, salah satu teknik yang dapat digunakan untuk proses penyelsaian persoalan SPP adalah Algoritma Genetika.

Proses pencarian lintasan terpendek adalah salah satu contoh persoalan SPP, misalnya pada suatu daerah terdapat jaringan transportasi darat yang menghubungkan kota-kota. Dari fakta tersebut, akan ditentukan jarak terpendek yang menghubungkan kota A ke kota B dan kembali lagi ke kota A. Untuk mencapai kota tujuan, perjalanan harus melalui kota-kota diantara kedua kota tersebut. Jika jumlah kota diantara keduanya tidak terlalu banyak, maka proses penentuan lintasan terpendek akan jadi solusi yang cukup mudah. Namun jika kota yang harus dilalui

jumlahnya ratusan bahkan ribuan, maka ini akan jadi persoalan serius. Persoalan seperti inilah yang dikelompokkan dalam persoalan SPP.

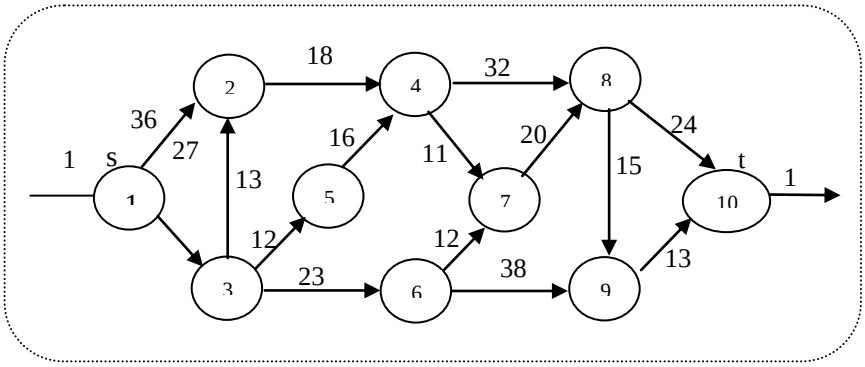
Untuk memahami persoalan SPP, berikut ini adalah uraian singkat tentang beberapa karakteristiknya,

- Graf berarah **G=(V,A)** dimana **V** adalah sekumpulan node (kota) dan **A** adalah sekumpulan arc (jalan)
- Cij adalah cost atau jarak untuk arc (i,j)
- Source node : node 1
- Destination node : node n
- Variabel :

$$Xij = \begin{cases} 1, \textit{ if link } (i,j) \textit{ is include in the path} \\ 0, \text{otherwise} \end{cases}$$

contoh data :

i	j	Cij
1	2	36
1	3	27
2	4	18
3	2	13
3	5	12
3	6	23
4	7	11
4	8	32
5	4	16
6	7	12
6	9	38
7	8	20
8	9	15
8	10	24
9	10	13



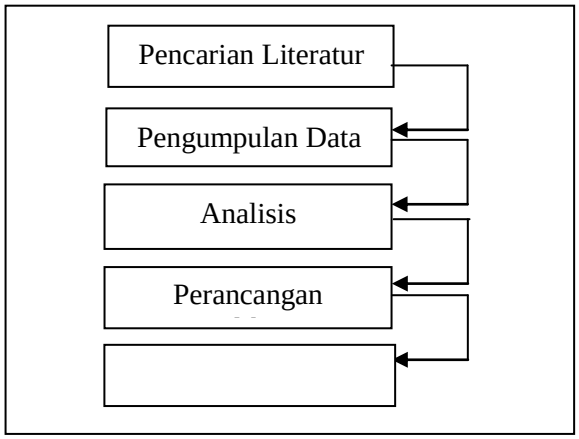
Gambar 3 Contoh Data Yang Disertakan Graf Berarah

Salah satu contoh dari *shortest path problem* yang paling sering dibahas adalah *Travelling Salesman Problem* dimana permasalahan tersebut adalah permasalahan untuk menemukan jarak terpendek untuk memutari semua *vertex* yang ada dan kemudian kembali pada titik awal pencarian rute. (Akmal Junaidi, Admi Syarif, Tristiyanto, Rico Adrian (Paper Akmal) : 2008)

2. METODE PENELITIAN

2.1 Tahapan Penelitian

Penelitian merupakan suatu siklus. Setiap tahapan akan diikuti oleh tahapan lain secara terus menerus. Tahapan-tahapan penelitian yang dilakukan oleh penulis dapat dilihat pada gambar dibawah ini :



Gambar 4 Kerangka Kerja Penelitian

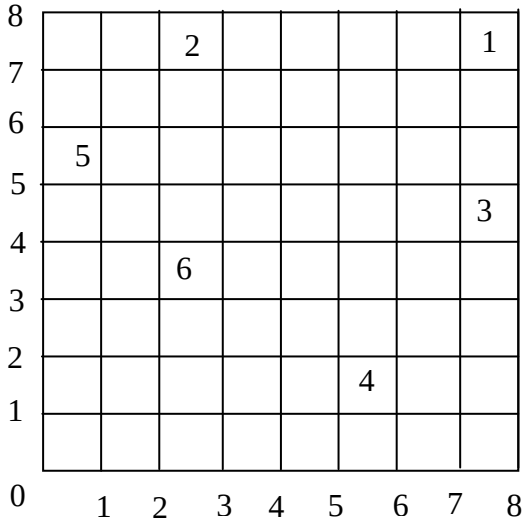
3. ANALISA DAN PERANCANGAN

3.1 Analisa Sistem

Adapun langkah-langkah dalam penyelesaian TSP dengan algoritma genetika adalah sebagai berikut :

a. **Skema pengkodean**

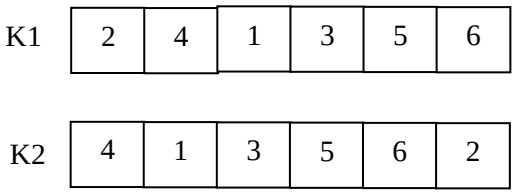
TSP dapat dirumuskan sebagai berikut : terdapat sekumpulan N node dengan posisi-posisi koordinatnya {Xi, Yi}, i = 1,2,….,N perhatikan gambar 5 dibawah ini. Terdapat 6 node yang harus dikunjungi.



Gambar 5 Peta Dua Dimensi Untuk TSP. Sejumlah Enam Kota Diberi Nomor Urut 1 Sampai 6, Dan Posisi-Posisinya Berupa Koordinat Dua Dimensi (x,y).

Suatu solusi dipresentasikan ke dalam suatu kromosom yang berisi nomor urut dari semua kota yang ada. Masing-masing nomor urut kota hanya boleh

muncul satu kali didalam kromosom sehingga satu kromosom mempresentasikan satu rute perjalanan (satu solusi) yang valid. Dimana suatu kromosom mempresentasikan suatu permutasi dari nomor urut kota 1,2,3…., *N*. Dengan demikian untuk gambar 5, suatu contoh kromosom adalah seperti pada gambar 6 dibawah ini :

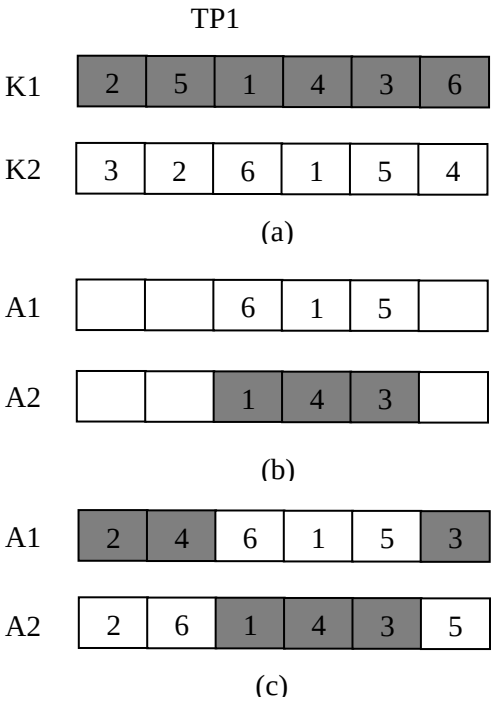


Gambar 6 Representasi Kromosom Untuk TSP

Pada gambar 6 kromosom K1 dan K2 mempresentasikan rute perjalanan yang sama. Hal ini bisa dipahami karena, secara siklus, K1 dan K2 memang memberikan rute perjalanan yang sama.

b. **Pindah Silang**

Pindah silang dapat diimplementasikan dengan skema *order crossover*. Pada skema ini satu bagian kromosom dipertukarkan dengan tetap menjaga urutan kota yang bukan bagian dari kromosom tersebut. Ilustrasi skema order crossover dapat dilihat pada gambar dibawah ini :



Gambar 7 Pindah Silang Menggunakan Skema *Order Crossover*

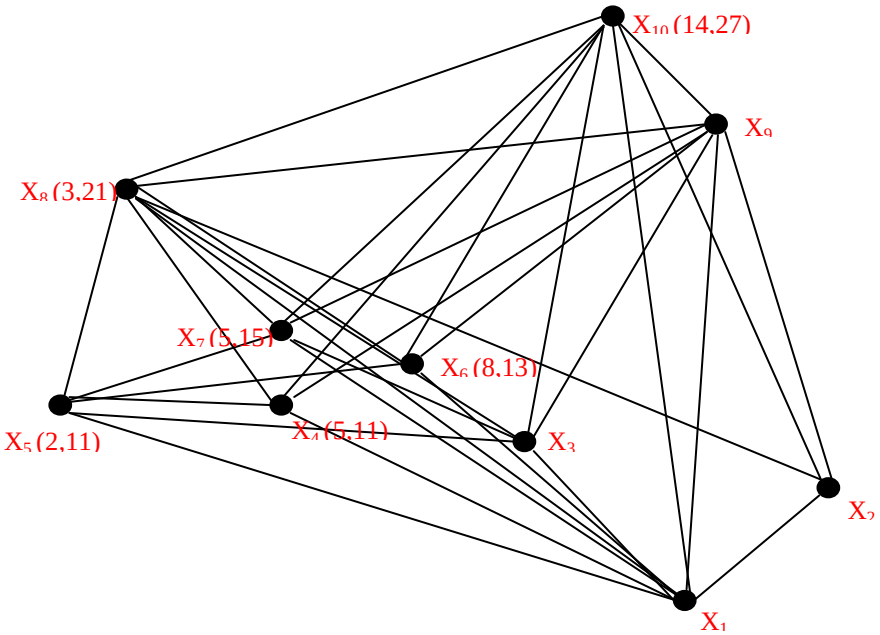
Mula-mula 2 buah titik potong, TP1 dan TP2 dibangkitkan secara random untuk memotong 2 buah kromosom orang tua, K1 dan K2 (gambar 7.a) kemudian 2 kromosom anak, A1 dan A2 mendapatkan gen-gen dari bagian kromosom K1 dan K2 secara menyilang kromosom A1 mendapatkan {6,1,5} dan A2 mendapatkan {1,4,3} (gambar 7.b). Posisi-posisi gen yang masih kosong pada kromosom A1 diisi dengan gen-gen dari K1, secara berurutan dari gen 1 sampai gen 6 yang belum ada pada A1. Hal yang sama juga dilakukan untuk kromosom A2 (gambar 7.c).

c. **Mutasi**

Operator mutasi biasanya diimplementasikan dengan menukarkan gen termutasi dengan gen lain yang dipilih secara random. Misalnya, kromosom {2,3,4,1,5} dapat termutasi menjadi kromosom {4,3,2,1,5}. Dalam hal ini gen 1 dan gen 3 saling ditukarkan. Skema mutasi ini dikenal sebagai *swapping mutation.*

d. **Perancangan Sistem**

Berikut adalah penyelesaian permasalahan *Travelling Salesman Problem* dengan menggunakan algoritma genetika. Terdapat 10 kota yang akan dikunjungi yaitu kota 1 (sunur), kota 2 (kurai taji), kota 3 (lapai), kota 4 (gelombang), kota 5 (pasar pariaman), kota 6 (jati), kota 7 (rawang), kota 8 (pauh), kota 9 (sei.pasak), dan kota 10 (koto marapak). Letak masing-masing kota dinyatakan dalam koordinat. Perjalanan dimulai dari kota pertama dan akhirnya juga akan berakhir dikota pertama. Akan ditentukan jalur terpendek atau total bobot minimum yang akan ditempuh untuk mengunjungi 10 kota tersebut. Kota tersebut dipresentasikan dalam graf G berikut ini :



Gambar 8 *Graph* dengan G vertex

Penyelesaian :

Langkah 1 : Inisialisasi

Koordinat masing-masing kota dapat dilihat pada graf G pada gambar 8. Dengan menggunakan persamaan 8 diperoleh jarak antar kota sebagaimana dipresentasikan dalam matrik bobot sisi graf G berukuran 10x10 sebagai berikut :

Tabel 1 Inisialisasi

	X_1	X_2	X_3	X_4	X_5	X_6	X_7	X_8	X_9	X_{10}
X_1	0	6.708	9.434	13.454	15.811	13.038	16.401	22.472	21.024	25.021
X_2	6.708	0	8.246	13.342	16.279	11.180	14.765	19.849	15.133	19.416
X_3	9.434	8.246	0	5.099	8.062	3.605	7.071	13.035	14.318	17.464
X_4	13.454	13.342	5.099	0	3.000	3.605	4.000	10.198	16.279	18.357
X_5	15.811	16.279	8.062	3.000	0	6.324	5.000	10.049	18.439	20.000
X_6	13.038	11.180	3.605	3.605	6.324	0	3.606	9.434	12.806	15.231
X_7	16.401	14.765	7.071	4.000	5.000	3.606	0	6.325	13.601	15.000
X_8	22.472	19.849	13.035	10.198	10.049	9.434	6.325	0	13.153	12.529
X_9	21.024	15.133	14.318	16.279	18.439	12.806	13.601	13.153	0	4.472
X_{10}	25.021	19.416	17.464	18.357	20.000	15.231	15.000	12.529	4.472	0

Langkah 2 :

1. Bentuk populasi awal dengan cara membangkitkan kromosom secara acak sebanyak ukuran populasi.
2. Hitung panjang jalur masing-masing kromosom dengan cara :

   Untuk ukuran populasi :

Kromosom [1] = 3-6-8-7-5-4-1-2-9-10

Panjang jalur [1] = jarak(3-6) + jarak(6-8) + jarak (8-7) + jarak(7-5) + jarak(5-4) + jarak(4-1) + jarak(1-2) + jarak(2-9) + jarak(9-10) + jarak(10-3)

= 3,605 + 9,434 + 6,325 + 5,000 + 3,000 + 13,454 + 6,708 + 15,133 + 4,472 + 17,464 = 84,595

Dan seterusnya sampai populasi ke-30. Hasil selengkapnya dapat dilihat pada tabel 3.4.

Langkah 3 :

Menghitung total *fitness*

TotalFitness= fitness(1) + fitness(2) + fitness(3) + … +fitness(30)

= 0,0118 + 0,0103 + 0,0095 + …. + 0,0076

= 0,3551

1. Hitung *fitness*relatif

   *Fitness*relatif dihitung dengan menggunakan persamaan

   P[i] = *Nilaifitness* [i]/*Totalfitness*

   Untuk *fitness* (1)

   P[1] = 0,0118/0,3551= 0,0332

   Dan seterusnya sampai populasi ke-30. Hasil selengkapnya dapat dilihat pada tabel 2
2. Hitung *fitness* kumulatif

   Fitness kumulatif dihitung dengan menggunakan persamaan berikut :

   Q[i] = Q[i-1] + P[i]

   Untuk *fitness* (1) dan (2)

Q[1] = 0,0332

Q[2] = 0,0332 + 0,0290 = 0,0622

Tabel 2 Fitness Relatif dan Fitness Kumulatif

Pop	P [i]	Q[i]	Pop	P [i]	Q[i]	Pop	P [i]	Q[i]
1	0,0332	0,0332	**11**	0,0329	0,3673	**21**	0,0301	0,7178
2	0,0290	0,0622	**12**	0,0335	0,4008	**22**	0,0363	0,7541
3	0,0267	0,0889	**13**	0,0327	0,4335	**23**	0,0332	0,7873
4	0,0324	0,1213	**14**	0,0377	0,4712	**24**	0,0386	0,8259
5	0,0284	0,1497	**15**	0,0403	0,5115	**25**	0,0389	0,8648
6	0,0349	0,1846	**16**	0,0349	0,5464	**26**	0,0276	0,8924
7	0,0594	0,244	**17**	0,0355	0,5819	**27**	0,0349	0,9273
8	0,0293	0,2733	**18**	0,0377	0,6196	**28**	0,0270	0,9543
9	0,0262	0,2995	**19**	0,0349	0,6545	**29**	0,0239	0,9783
10	0,0349	0,3344	**20**	0,0332	0,6877	**30**	0,0214	0,9996

3. Seleksi Roda *Roulette-Wheel*

a. Bangkitkan nilai acak r yang bernilai antara 0 dan 1 sebanyak ukuran populasi (30). Bilangan acak untuk proses seleksi dapat dilihat pada table 3 berikut :

Tabel 3 Bilangan Acak Untuk Proses Seleksi

Pop	Bil. Acak	Pop	Bil. Acak	Pop	Bil. Acak
1	0,3120	**11**	0,0321	**21**	0,6142
2	0,0153	**12**	0,7529	**22**	0,7582
3	0,4522	**13**	0,8081	**23**	0,3364
4	0,6413	**14**	0,5042	**24**	0,8453
5	0,0094	**15**	0,6666	**25**	0,0635
6	0,9310	**16**	0,8468	**26**	0,1124
7	0,0521	**17**	0,8884	**27**	0,3737
8	0,4442	**18**	0,9905	**28**	0,4046
9	0,3572	**19**	0,4321	**29**	0,8789
10	0,0987	**20**	0,9325	**30**	0,6873

Jika R[k] < C[k] maka kromosom ke-k sebagai induk, selain itu pilih kromosom ke-k sebagai induk dengan syarat C[k-1] < R[k] < C[k]. Putar *roulette-wheel* sebanyak jumlah populasi yaitu 30 kali. Untuk i= 1,2,…,30, jika R[i] ≤ Q[1] maka pilih kromosom (1) pada urutan ke-1. Untuk j= 2,3,…,29, jika Q[j] < R[i] ≤ Q[j+1] maka pilih kromosom ke [j+1] pada urutan ke-i.

Untuk populasi ke-1 :
Q[9] < R[i] ≤ Q[10] yaitu 0,2995 < 0,3120 ≤ 0,3344 maka pilih kromosom pada populasi ke-10 pada urutan ke-1. Begitu seterusnya sampai populasi ke-30.Hasil selengkapnya dapat dilihat pada tabel 4.

Tabel 4. Hasil Seleksi Roda Roulette Wheel

Pop	Kromosom	Fitness	Ke-	Pop	Kromosom	Fitness	Ke-
1	3-2-9-10-8-7-5-4-6-1	0,0124	10	**16**	10-8-7-6-5-4-3-1-2-9	0,0138	25
2	9-7-4-6-5-3-1-2-10-8	0,0103	2	**17**	4-10-8-9-7-5-6-3-1-2	0,0098	26
3	2-1-3-4-5-7-6-8-10-9	0,0134	14	**18**	8-2-10-5-6-1-7-4-3-9	0,0076	30
4	4-5-7-6-8-10-9-3-2-1	0,0124	19	**19**	6-10-9-3-2-1-4-5-7-8	0,0116	13
5	3-6-8-7-5-4-1-2-9-10	0,0118	1	**20**	2-3-1-4-7-5-6-9-8-10	0,0096	28
6	2-3-1-4-7-5-6-9-8-10	0,0096	28	**21**	4-5-7-6-8-10-9-3-2-1	0,0124	19
7	9-7-4-6-5-3-1-2-10-8	0,0103	2	**22**	2-8-10-9-7-6-4-5-3-1	0,0118	23
8	2-1-3-4-5-7-6-8-10-9	0,0134	14	**23**	9-10-8-7-4-6-3-1-2-5	0,0117	11
9	9-10-8-7-4-6-3-1-2-5	0,0117	11	**24**	10-8-7-6-5-4-3-1-2-9	0,0138	25
10	5-4-7-8-6-3-9-10-2-1	0,0115	4	**25**	4-7-8-6-3-5-1-9-10-2	0,0095	3
11	3-6-8-7-5-4-1-2-9-10	0,0118	1	**26**	5-4-7-8-6-3-9-10-2-1	0,0115	4
12	6-9-10-8-7-5-4-3-2-1	0,0129	22	**27**	7-8-5-4-6-2-1-3-9-10	0,0119	12
13	7-6-4-5-3-1-2-9-10-8	0,0137	24	**28**	6-10-9-3-2-1-4-5-7-8	0,0116	13
14	7-5-4-6-3-1-2-9-10-8	0,0143	15	**29**	4-10-8-9-7-5-6-3-1-2	0,0098	26
15	3-2-1-4-5-6-7-9-10-8	0,0118	20	**30**	3-2-1-4-5-6-7-9-10-8	0,0118	20

4. Proses *Crossover*

a. Bangkitkan bilangan acak r yang bernilai antara 0 dan 1 sebanyak ukuran populasi (30). Bilangan acak untuk proses *crossover* dapat dilihat pada tabel 5 berikut :

Tabel 5 Bilangan Acak Untuk Proses Crossover

Pop	Bil.Acak	Pop	Bil.Acak	Pop	Bil.Acak
1	0,7452	**11**	0,2829	**21**	0,4509
2	0,4273	**12**	0,9011	**22**	0,5354
3	0,8452	**13**	0,8831	**23**	0,2789
4	0,6532	**14**	0,4252	**24**	0,5730
5	0,7891	**15**	0,6768	**25**	0,9730
6	0,5241	**16**	0,9392	**26**	0,4043
7	0,7562	**17**	0,1212	**27**	0,3066
8	0,4555	**18**	0,3230	**28**	0,2796
9	0,7321	**19**	0,9394	**29**	0,5166
10	0,5452	**20**	0,8788	**30**	0,7271

b. Untuk i= 1,2,…, 30

Jika R[i] < Pc atau R[i] < 0,500 maka pilih kromosom ke-i sebagai induk.

Dari tabel 5 terlihat bahwa populasi yang bilangan yang kurang dari Pc adalah populasi ke : 2, 8, 11, 14, 17, 18, 21, 23, 26, 27, dan 28. Hal ini berarti bahwa kromosom yang berhak untuk melakukan *crossover* adalah kromosom pada populasi seperti yang telah disebutkan diatas. Tabel 6 menunjukkan kromosom-kromosom yang bilangan acaknya kurang dari Pc.

Tabel 6 Kromosom Yang Bilangan Acaknya Kurang Dari Pc

Pop	Kromosom	Fitness	Pop	Kromosom	Fitness	Pop	Kromosom	Fitness
2	9-7-4-6-5-3-1-2-10-8	0,0103	**17**	4-10-8-9-7-5-6-3-1-2	0,0098	**26**	5-4-7-8-6-3-9-10-2-1	0,0115
8	2-1-3-4-5-7-6-8-10-9	0,0134	**18**	8-2-10-5-6-1-7-4-3-9	0,0076	**27**	7-8-5-4-6-2-1-3-9-10	0,0119
11	3-6-8-7-5-4-1-2-9-10	0,0118	**21**	4-5-7-6-8-10-9-3-2-1	0,0124	**28**	6-10-9-3-2-1-4-5-7-8	0,0116
14	7-5-4-6-3-1-2-9-10-8	0,0143	**23**	9-10-8-7-4-6-3-1-2-5	0,0117			

c. Untuk k=1,2,…,11, karena mod(11,2) ≠ 0, sehingga k=k-1 = 11-1 = 10. Jadi buang salah satu kromosom, misalkan kromosom ke 28 dibuang. Tabel 7 menunjukkan kromosom-kromosom yang akan dilakukan *crossover*.

Table 7 Kromosom Yang Akan Dilakukan Crossover

Pop	Kromosom	Fitness
2	9-7-4-6-5-3-1-2-10-8	0,0103
8	2-1-3-4-5-7-6-8-10-9	0,0134
11	3-6-8-7-5-4-1-2-9-10	0,0118
14	7-5-4-6-3-1-2-9-10-8	0,0143
17	4-10-8-9-7-5-6-3-1-2	0,0098
18	8-2-10-5-6-1-7-4-3-9	0,0076
21	4-5-7-6-8-10-9-3-2-1	0,0124
23	9-10-8-7-4-6-3-1-2-5	0,0117
26	5-4-7-8-6-3-9-10-2-1	0,0115
27	7-8-5-4-6-2-1-3-9-10	0,0119

d. Lakukan *crossover* antara kromosom 2 dengan 8, 11 dengan 14, 17 dengan 18, 21 dengan 23, dan kromosom 26 dengan 27.

Crossover antara kromososm 2 dengan 8 :

Induk_1	9	7	4	6	5	3	1	2	10	8
Induk_2	2	1	3	4	5	7	6	8	10	9
Anak_1	x	X	x	x	5	3	x	x	x	X
Anak_2	x	X	x	x	5	7	x	x	x	X
Anak_1	2	1	4	7	5	3	6	8	10	9
Anak_2	9	4	6	3	5	7	1	2	10	8

Lakukan proses *crossover* dengan cara yang sama untuk kromosom 11 dengan 14, 17 dengan 18, 21 dengan 23, dan kromosom 26 dengan 27. Sehingga diperoleh hasilnya seperti terlihat pada tabel 8 sebagai berikut :

Tabel 8 Kromosom-kromosom Setelah Dilakukan *Crossover*

Pop	Kromosom	Fitness	Pop	Kromosom	Fitness
1	3-2-9-10-8-7-5-4-6-1	0,0124	**16**	10-8-7-6-5-4-3-1-2-9	0,0138
2	2-1-4-7-5-3-6-8-10-9	0,0103	**17**	8-2-10-9-7-5-6-1-4-3	0,0098
3	2-1-3-4-5-7-6-8-10-9	0,0134	**18**	4-10-8-5-6-9-7-3-1-2	0,0076
4	4-5-7-6-8-10-9-3-2-1	0,0124	**19**	6-10-9-3-2-1-4-5-7-8	0,0116
5	3-6-8-7-5-4-1-2-9-10	0,0118	**20**	2-3-1-4-7-5-6-9-8-10	0,0096
6	2-3-1-4-7-5-6-9-8-10	0,0096	**21**	9-10-7-6-8-4-3-1-2-5	0,0124
7	9-7-4-6-5-3-1-2-10-8	0,0103	**22**	2-8-10-9-7-6-4-5-3-1	0,0118
8	9-4-6-3-5-7-1-2-10-8	0,0134	**23**	4-5-8-7-6-10-9-3-2-1	0,0117
9	9-10-8-7-4-6-3-1-2-5	0,0117	**24**	10-8-7-6-5-4-3-1-2-9	0,0138
10	5-4-7-8-6-3-9-10-2-1	0,0115	**25**	4-7-8-6-3-5-1-9-10-2	0,0095
11	8-2-10-9-7-5-6-1-4-3	0,0118	**26**	7-8-5-4-6-1-3-10-2-9	0,0115
12	6-9-10-8-7-5-4-3-2-1	0,0129	**27**	5-4-7-8-6-10-2-3-9-1	0,0119
13	7-6-4-5-3-1-2-9-10-8	0,0137	**28**	6-10-9-3-2-1-4-5-7-8	0,0116
14	3-6-8-7-5-4-2-9-1-10	0,0143	**29**	4-10-8-9-7-5-6-3-1-2	0,0098
15	3-2-1-4-5-6-7-9-10-8	0,0118	**30**	3-2-1-4-5-6-7-9-10-8	0,0118

5. Proses Mutasi

populasi (30Bangkitkan bilangan acak r yang bernilai antara 0 dan 1 sebanyak ukuran). Bilangan acak untuk proses mutasi dapat dilihat pada tabel 9 berikut :

Tabel 9 Bilangan Acak Untuk Proses Mutasi

Pop	Bil.Acak	Pop	Bil.Acak	Pop	Bil.Acak
1	0,3291	**11**	0,5294	**21**	0,9872
2	0,4782	**12**	0,6593	**22**	0,9123
3	0,1345	**13**	0,7594	**23**	0,7320
4	0,7594	**14**	0,1393	**24**	0,1274
5	0,8694	**15**	0,9752	**25**	0,8493
6	0,3958	**16**	0,9138	**26**	0,1232
7	0,1648	**17**	0,9302	**27**	0,9473
8	0,7583	**18**	0,2739	**28**	0,2837
9	0,9831	**19**	0,3859	**29**	0,9237
10	0,3479	**20**	0,9347	**30**	0,6475

Untuk i=1,2,…,30 maka, jika R[i] < 0,100 maka kromososm ke-i terkena mutasi. Pada tabel 9, karena tidak ada R[i] < 0,100 maka tidak ada kromosom yang terkena mutasi. Sehingga hasil kromosom setelah dilakukan proses mutasi sama dengan hasil kromososm setelah dilakukan *crossover*. Jadi populasi baru hasil mutasi sama dengan populasi baru hasil *crossover* yang terlihat pada tabel 9.

6. Proses Elitisme

a. Bangkitkan bilangan acak r yang bernilai antara 0 dan 1 sebanyak ukuran populasi (30). Bilangan acak untuk proses elitism dapat dilihat pada tabel 10.

Tabel 10 Bilangan Acak Untuk Proses Elitisme

Pop	Bil.Acak	Pop	Bil.Acak	Pop	Bil.Acak
1	0,5845	**11**	0,8641	**21**	0,7532
2	0,6453	**12**	0,9537	**22**	0,2536
3	0,7439	**13**	0,9153	**23**	0,6281
4	0,0594	**14**	0,1254	**24**	0,5273
5	0,6397	**15**	0,9638	**25**	0,6372
6	0,6484	**16**	0,4352	**26**	0,6372
7	0,1257	**17**	0,3253	**27**	0,6248
8	0,8463	**18**	0,7543	**28**	0,4172
9	0,8165	**19**	0,3253	**29**	0,6372
10	0,1254	**20**	0,6422	**30**	0,8152

b. Untuk i=1,2,…,30 maka, jika R[i] < 0,100 maka lakukan penggantian pada kromosom ke-i. Dari tabel 10 terlihat bahwa bilangan acak yang nilainya kurang dari 0,100 adalah bilangan acak pada populasi ke-4 maka lakukan penggantian pada kromosom ke-4.

c. Ganti kromosom ke-i dengan kromosom yang memiliki nilai fitness paling tinggi pada kromosom setelah dilakukan mutasi. Tabel 11 menunjukkan penggantian kromosom

Tabel 11 Penggantian Kromosom

Pop	Kromosom	Fitness
4	4-5-7-6-8-10-9-3-2-1	0,0124
25	10-8-7-6-5-4-3-1-2-9	0,0138

Tabel 12 menunjukkan kromosom hasil elitisme (setelah dilakukan penggantian kromosom).

Tabel 12 Kromosom Setelah Dilakukan Penggantian

Pop	Kromosom	Fitness	Pop	Kromosom	Fitness
1	3-2-9-10-8-7-5-4-6-1	0,0124	**16**	10-8-7-6-5-4-3-1-2-9	0,0138
2	2-1-4-7-5-3-6-8-10-9	0,0103	**17**	8-2-10-9-7-5-6-1-4-3	0,0098
3	2-1-3-4-5-7-6-8-10-9	0,0134	**18**	4-10-8-5-6-9-7-3-1-2	0,0076
4	10-8-7-6-5-4-3-1-2-9	0,0138	**19**	6-10-9-3-2-1-4-5-7-8	0,0116
5	3-6-8-7-5-4-1-2-9-10	0,0118	**20**	2-3-1-4-7-5-6-9-8-10	0,0096
6	2-3-1-4-7-5-6-9-8-10	0,0096	**21**	9-10-7-6-8-4-3-1-2-5	0,0124
7	9-7-4-6-5-3-1-2-10-8	0,0103	**22**	2-8-10-9-7-6-4-5-3-1	0,0118
8	9-4-6-3-5-7-1-2-10-8	0,0134	**23**	4-5-8-7-6-10-9-3-2-1	0,0117
9	9-10-8-7-4-6-3-1-2-5	0,0117	**24**	10-8-7-6-5-4-3-1-2-9	0,0138
10	5-4-7-8-6-3-9-10-2-1	0,0115	**25**	4-7-8-6-3-5-1-9-10-2	0,0095

11	8-2-10-9-7-5-6-1-4-3	0,0118	**26**	7-8-5-4-6-1-3-10-2-9	0,0115
12	6-9-10-8-7-5-4-3-2-1	0,0129	**27**	5-4-7-8-6-10-2-3-9-1	0,0119
13	7-6-4-5-3-1-2-9-10-8	0,0137	**28**	6-10-9-3-2-1-4-5-7-8	0,0116
14	3-6-8-7-5-4-2-9-1-10	0,0143	**29**	4-10-8-9-7-5-6-3-1-2	0,0098
15	3-2-1-4-5-6-7-9-10-8	0,0118	**30**	3-2-1-4-5-6-7-9-10-8	0,0118

Populasi akhir untuk generasi ke-i dapat dilihat pada tabel 13

Tabel 13 Populasi Akhir Generasi ke-i

Pop	Kromosom	Panjang Jalur	Fitness	Pop	Kromosom	Panjang Jalur	Fitness
1	3-6-8-7-5-4-1-2-9-10	84,595	0,0118	16	5-6-9-10-8-7-3-2-1-4	80,935	0,0124
2	9-7-4-6-5-3-1-2-10-8	96,832	0,0103	17	5-4-3-1-2-6-7-9-10-8	79,678	0,0126
3	4-7-8-6-3-5-1-9-10-2	105,491	0,0095	18	8-7-5-4-6-3-2-1-9-10	74,514	0,0134
4	10-8-7-6-5-4-3-1-2-9	72,63	0,0138	19	4-5-7-6-8-10-9-3-2-1	80,767	0,0124
5	10-6-4-5-7-3-8-2-1-9	98,995	0,0101	20	3-2-1-4-5-6-7-9-10-8	84,975	0,0118
6	7-8-6-5-4-3-1-2-9-10	80,929	0,0124	21	10-6-7-5-4-3-2-1-9-8	93,596	0,0107
7	6-7-5-4-3-1-2-9-10-8	74,415	0,0134	22	6-9-10-8-7-5-4-3-2-1	77,223	0,0129
8	2-10-8-5-4-7-6-9-3-1	95,866	0,0104	23	2-8-10-9-7-6-4-5-3-1	84,866	0,0118
9	1-7-8-10-6-9-4-5-3-2	108,058	0,0093	24	7-6-4-5-3-1-2-9-10-8	72,874	0,0137
10	3-2-9-10-8-7-5-4-6-1	80,782	0,0124	25	10-8-7-6-5-4-3-1-2-9	72,63	0,0138
11	9-10-8-7-4-6-3-1-2-5	85,396	0,0117	26	4-10-8-9-7-5-6-3-1-2	102,053	0,0098
12	7-8-5-4-6-2-1-3-9-10	84,091	0,0119	27	9-7-5-4-3-1-2-6-8-10	80,457	0,0124
13	6-10-9-3-2-1-4-5-7-8	86,188	0,0116	28	2-3-1-4-7-5-6-9-8-10	104,362	0,0096
14	2-1-3-4-5-7-6-8-10-9	74,415	0,0134	29	9-5-2-8-7-3-6-4-1-10	118,12	0,0085
15	7-5-4-6-3-1-2-9-10-8	69,811	0,0143	30	8-2-10-5-6-1-7-4-3-9	131,598	0,0076

Jadi hasil populasi akhir pada generasi ke-i pada tabel 13 terlihat bahwa :
Fitness terbaik berada pada populasi ke-4 dengan nilai fitness 0,0138.
Fitness terburuk berada pada populasi ke-30 dengan nilai fitness 0,0076.
Populasi akhir pada generasi ke-1 ini akan dijadikan sebagai populasi awal untuk generasi ke-2 dan lakukan langkah 1 sampai 3 untuk populasi ke-2 sampai generasi maksimum (generasi ke-50).

Langkah 4 :
Setelah dilakukan pengujian dengan Matlab 7.9 diperoleh rekap hasil masing-masing generasi seperti terlihat pada tabel 14 berikut :

Tabel 14 Rekap Hasil Setelah Diuji Dengan Matlab 7.9

NO.	POLA JALUR TSP										F. TERBAIK	F. TERBURUK	RATA - RATA
1	**10**	**9**	**5**	**1**	**2**	**6**	**4**	**3**	**7**	**8**	**0.0127**	**0.0076**	**0.0092**
2	**10**	**9**	**5**	**1**	**2**	**6**	**4**	**3**	**7**	**8**	**0.0127**	**0.0078**	**0.0097**
3	**10**	**9**	**5**	**1**	**2**	**6**	**4**	**3**	**7**	**8**	**0.0127**	**0.0076**	**0.0100**
4	**1**	**2**	**6**	**3**	**7**	**5**	**4**	**8**	**9**	**10**	**0.0155**	**0.0076**	**0.0106**
5	**1**	**2**	**6**	**3**	**7**	**5**	**4**	**8**	**9**	**10**	**0.0155**	**0.0083**	**0.0109**

6	1	2	6	3	7	5	4	8	9	10	0.0155	0.0083	0.0112
7	1	2	6	3	7	5	4	8	9	10	0.0155	0.0085	0.0117
8	1	2	6	3	7	5	4	8	9	10	0.0155	0.0085	0.0118
9	1	2	6	3	7	5	4	8	9	10	0.0155	0.0085	0.0121
10	1	2	6	3	7	5	4	8	10	9	0.0157	0.0085	0.0125
11	1	2	6	3	7	5	4	8	10	9	0.0157	0.0088	0.0131
12	1	2	3	7	6	5	4	8	9	10	0.0159	0.0087	0.0138
13	1	2	3	7	6	5	4	8	9	10	0.0159	0.0085	0.0137
14	1	2	3	7	6	5	4	8	9	10	0.0159	0.0086	0.0136
15	1	2	3	7	6	5	4	8	9	10	0.0159	0.0095	0.0138
16	1	2	3	6	7	5	4	8	9	10	0.0172	0.0096	0.0138
17	1	2	3	6	7	5	4	8	9	10	0.0172	0.0092	0.0135
18	1	2	3	6	7	5	4	8	9	10	0.0172	0.0097	0.0145
19	1	2	3	6	7	5	4	8	9	10	0.0172	0.0097	0.0150
20	1	2	3	6	7	5	4	8	9	10	0.0172	0.0096	0.0142
21	1	2	3	6	7	5	4	8	9	10	0.0172	0.0096	0.0151
22	1	2	3	6	7	5	4	8	9	10	0.0172	0.0105	0.0146
23	1	2	3	6	7	5	4	8	9	10	0.0172	0.0099	0.0153
24	1	2	3	6	7	5	4	8	9	10	0.0172	0.0097	0.0147
25	1	2	3	6	5	4	7	8	9	10	0.0179	0.0086	0.0149
26	1	2	3	6	5	4	7	8	9	10	0.0179	0.0092	0.0156
27	1	2	3	6	5	4	7	8	9	10	0.0179	0.0089	0.0154
28	1	2	3	6	5	4	7	8	9	10	0.0179	0.0104	0.0163
29	1	2	3	6	5	4	7	8	9	10	0.0179	0.0105	0.0165
30	1	2	3	6	5	4	7	8	9	10	0.0179	0.0111	0.0170
31	1	2	3	6	5	4	7	8	9	10	0.0179	0.0123	0.0168
32	1	2	3	6	5	4	7	8	9	10	0.0179	0.0105	0.0163
33	1	2	3	6	5	4	7	8	9	10	0.0179	0.0106	0.0161
34	1	2	3	6	5	4	7	8	9	10	0.0179	0.0117	0.0164
35	1	2	3	6	5	4	7	8	9	10	0.0179	0.0105	0.0162
36	1	2	3	6	5	4	7	8	9	10	0.0179	0.0105	0.0162
37	1	2	3	6	5	4	7	8	9	10	0.0179	0.0123	0.0170
38	1	2	3	6	5	4	7	8	9	10	0.0179	0.0105	0.0167
39	1	2	3	6	5	4	7	8	9	10	0.0179	0.0105	0.0168
40	1	2	3	6	5	4	7	8	9	10	0.0179	0.0099	0.0161
41	1	2	3	6	5	4	7	8	9	10	0.0179	0.0105	0.0169
42	1	2	3	6	5	4	7	8	9	10	0.0179	0.0105	0.0163
43	1	2	3	6	5	4	7	8	9	10	0.0179	0.0105	0.0168
44	1	2	3	6	5	4	7	8	9	10	0.0179	0.0105	0.0160
45	1	2	3	6	5	4	7	8	9	10	0.0179	0.0105	0.0170
46	1	2	3	6	5	4	7	8	9	10	0.0179	0.0106	0.0175

47	**1**	**2**	**3**	**6**	**5**	**4**	**7**	**8**	**9**	**10**	**0.0179**	**0.0179**	**0.0179**
48	**1**	**2**	**3**	**6**	**5**	**4**	**7**	**8**	**9**	**10**	**0.0179**	**0.0179**	**0.0179**
49	**1**	**2**	**3**	**6**	**5**	**4**	**7**	**8**	**9**	**10**	**0.0179**	**0.0179**	**0.0179**
50	**1**	**2**	**3**	**6**	**5**	**4**	**7**	**8**	**9**	**10**	**0.0179**	**0.0179**	**0.0179**

Fitness rata-rata generasi ke-i diperoleh dengan :

$$\text{Fitness rata-rata (47)} = \frac{\textit{FitnessTerbaik}\ (47) + \textit{FitnessTerburuk}\ (47)}{2}$$

$$= \frac{0{,}0179 + 0{,}0179}{2}$$

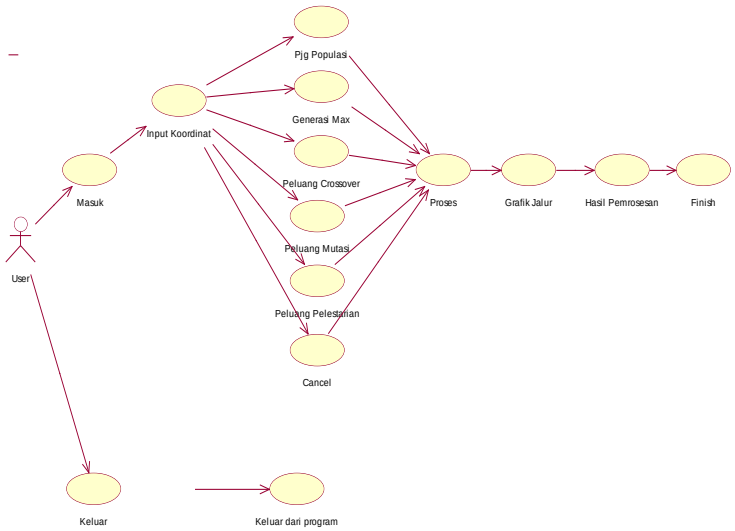
$$= \frac{0{,}0358}{2}$$

Dari tabel 14 terlihat bahwa nilai fitness paling maksimum adalah 0,0176 dengan kromosom 1-2-3-6-5-4-7-8-9-10. Ini berarti bahwa jalur terpendek setelah dilakukan pencarian oleh algoritma genetika untuk 50 generasi adalah 1-2-3-6-5-4-7-8-9-10 dengan panjang jalur 55,8342.

Pembuatan *Use Case* Diagram

Use case diagram menggambarkan bagaimana aktifitas *user* dalam penggunaan program ini untuk mencapai tujuan yang diharapkan. Dalam notasi *use case*, pengguna sistem disebut dengan *actor (role). Use case* sesungguhnya merupakan deskripsi logika dari fungsionalitas bagian sistem/perangkat lunak tertentu. *Use case* tidak secara langsung merupakan konstruksi implementasi sistem/perangkat lunak yang kita kembangkan.

Use case pada umumnya digambarkan menggunakan bentuk geometri elips dengan nama*use case* didalamnya atau bagian bawahnya. *Use case* terhubung dengan garis tegas ke actor yang terkomunikasi dengannya. Berikut diagram *use case* yang dirancang dalam aplikasi ini.



Gambar 9 U*se Case* Diagram dalam Aplikasi Algoritma Genetik

Sistem perangkat lunak penentuan jalur terpendek pada TSP diatas digunakan oleh user hendak melakukan pencarian jalur jalan optimum dengan aplikasi Matlab 7.9. User dalam kasus ini setelah masuk kedalam sistem, harus menginputkan koordinat yang menjadi parameter dari algoritma genetika yaitu input panjang populasi, input genetika maksimal, input peluang *crossover*, input peluang mutasi serta inputkan peluang pelestarian. Apabila selama proses penginputan terjadi pembatalan, maka *user* dapat menghapusnya dengan menekan tombol *cancel.*

Selanjutnya, bila *user* telah selesai menginputkan semua parameter dengan benar maka *user* harus menekan tombol proses. Setelah itu akan keluar grafik jalur optimalnya serta pemrosesannya. Jika telah selesai maka tekan *finish.*

4. PENUTUP

4.1 Kesimpulan

Walaupun solusi TSP yang dihasilkan oleh algoritma genetika belum tentu merupakan solusi paling optimal (misalnya apabila yang dilalui sangat banyak), namun algoritma genetika akan menghasilkan solusi yang lebih optimal pada setiap generasinya. Hal tersebut terlihat dari nilai *fitness* tiap generasi.

Kelebihan algoritma genetika sangat terlihat dari adaptivitasnya dalam menyelesaikan masalah. Begitu kita bisa mengkodekan masalah ke dalam kromosom dan bisa membangun fungsi *fitness*, maka kita dapat membangun algoritma genetika untuk masalah tersebut. Beberapa komponen algoritma genetika, misalnya Inisialisasi Populasi, *Linear Fitness Ranking*, *Roulette-Wheel*, Pindah Silang dan Mutasi bisa digunakan untuk beberapa masalah berbeda termasuk masalah TSP.

4.2 Saran

Dalam perancangan sistem ini dihadapi beberapa kendala atau pun kekurangan yang perlu diatasi. Saran terhadap pengembangan sistem ini kedepan adalah :

1. Diharapkan nantinya agar dapat dikembangkan sebuah pencarian TSP dengan algoritma genetika yang dapat ditampilkan dalam bentuk peta sebenarnya dari suatu daerah yang diteliti dengan cakupan jarak dan wilayah yang lebih besar dan luas.
2. Agar dapat dilakukan pengujian menggunakan bahasa pemrograman lain seperti Java, Bahasa C, Delphi, Visual Basic dll.

DAFTAR PUSTAKA

Akmal Junaidi, Admi Syarif, Tristiyanto, Rico Adrian. 2008. Paper Akmal

Kusumadewi,S., 2003, *Artificial Intelligence (Teknik dan Aplikasinya)*, Yogyakarta : Graha Ilmu

Kusumadewi,S., dan Hari, p., 2005, Penyelesaian Masalah Optimasi dengan Teknik teknik Heuristik, Yogyakarta : Graha Ilmu

F.Saptono, I.Mutakhiroh, T. Hidayat, dan A. Fauziyah (2007). Perbandingan Performansi Algoritma Genetik dan Algoritma Semut untuk Penyelesaian Shortest Path Problem. Seminar Nasional Sistem dan Informatika. Bali. 16 November 2007.

http://informatika.web.id/algoritma-genetika.htm

http://repository.usu.ac.id/bitstream/123456789/16595/4/Chapter%2011.pdf

http://stta.name/data_lp3m/yuliani.pdf