ANALISA PERBANDINGAN METODE *ROULETTE WHEEL SELECTION, RANK SELECTION* DAN *TOURNAMENT SELECTION* PADA AGLORITMA GENETIKA (STUDI KASUS : *TRAVELLING SALESMAN PROBLEM* (TSP))

TUGAS AKHIR

Diajukan Sebagai Salah Satu Syarat
Untuk Memperoleh Gelar Sarjana Teknik Pada
Jurusan Teknik Informatika

oleh :

INAYATI
10351022921

JURUSAN TEKNIK INFORMATIKA
FAKULTAS SAINS DAN TEKNOLOGI
UNIVERSITAS ISLAM NEGERI SULTAN SYARIF KASIM RIAU
PEKANBARU
2010

ANALISA PERBANDINGAN METODE *ROULETTE WHEEL SELECTION, RANK SELECTION* DAN *TOURNAMENT SELECTION* PADA ALGORITMA GENETIKA (STUDI KASUS : *TRAVELLING SALESMAN PROBLEM* (TSP))

INAYATI
10351022921

Tanggal Sidang : 28 Juni 2010
Periode Wisuda : Juli 2010

Jurusan Teknik Informatika
Fakultas Sains dan Teknologi
Universitas Islam Negeri Sultan Syarif Kasim Riau

ABSTRAK

Algoritma genetika dapat digunakan untuk menyelesaikan permasalahan *travelling salesman problem* (TSP). Metode seleksi yang digunakan diantaranya *roulette wheel selection* (RWS) metode seleksi yang memilih orang tua berdasarkan nilai kecocokannya, *rank selection* (RS) metode seleksi yang melakukan perankingan untuk populasi terlebih dahulu, dan *tournament selection* (TS) dimana sebuah grup dipilih secara acak dan memilih kromosom yang terbaik dari grup tersebut.

Penelitian dilakukan untuk mencari metode seleksi mana yang paling efektif dalam menyelesaikan permasalahan TSP yang grafnya lengkap dan berbobot dengan melihat 4 aspek perbandingan, yaitu *time complexity* tiap-tiap metode, *space complexity* tiap-tiap metode, apakah tiap-tiap metode menemukan solusi (*completeness*), dan solusi dengan metode tersebut merupakan *optimality* atau tidak.

Hasil dari penelitian ini baik secara manual maupun sistem menunjukkan bahwa metode TS memiliki *time complexity* dan *space complexity* yang lebih baik dibandingkan metode RWS dan RS. Secara sistem untuk 100 kali percobaan hingga jumlah kota 200, persentase kecepatan TS dibandingkan RWS dan RS adalah 22.46% dan 9.59%. Persentase memori TS dibandingkan RWS dan RS adalah 12.20% dan 6.97%. Ketiga metode ini dapat menemukan solusi (*completeness*) dan sama-sama *optimality* dari proses pencarian solusi permasalahan TSP.

Kata Kunci : Algoritma genetika, *rank selection*, *roulette wheel selection*, *tournament selection*, *travelling salesman problem*

COMPARATIVE ANALYSIS OF METHODS ROULETTE WHEEL SELECTION, RANK SELECTION AND TOURNAMENT SELECTION ON GENETIC ALGORITHM (CASE STUDY : TRAVELLING SALESMAN PROBLEM (TSP))

INAYATI
10351022921

Session Date: June 28, 2010
Graduation Period: July 2010

Department of Informatics
Faculty of Science and Technology
State Islamic University of Sultan Sharif Kasim Riau

ABSTRACT

Genetic algorithms can be used to solve the problems traveling salesman problem (TSP). The method of selection that is used among the roulette wheel selection (RWS) selection method selecting parents based on the value of suitability, rank selection (RS) methods which make the selection for the population ranking first, and tournament selection (TS) where a randomly selected group and choosing the best chromosomes from the group.

The study was conducted to find the selection method will be most effective in solving the TSP problem is a complete and weighted graph by looking at four aspects of the comparison, the time complexity of each method, the space complexity of each method, whether each method of finding solutions (completeness) , and solutions with an optimality method or not.

Results from this study either manually or the system shows that the TS method has time complexity and space complexity better than the RWS and RS methods. In the system for up to 100 times the number of 200 trials, the percentage rate TS compared to RWS and RS was 22:46% and 9:59%. The percentage of memory TS compared to RWS and RS was 12:20% and 6.97%. Third, this method can find a solution (completeness) and the same both optimality of the process of finding solutions for TSP problems.

Keywords: genetic algorithm, rank selection, roulette wheel selection, tournament selection, traveling salesman problem

DAFTAR ISI

Halaman

BAB I

PENDAHULUAN

1.1 Latar Belakang

Algoritma adalah salah satu konsep matematika dan analisis untuk membantu seseorang dalam menyelesaikan suatu masalah atau persoalan. Algoritma disusun dari sekumpulan langkah berhingga, masing-masing langkah mungkin memerlukan satu operasi atau lebih. Algoritma umumnya dirancang untuk menyelesaikan suatu masalah spesifik dan dengan usaha yang paling minimum.

Algoritma genetika (*genetic algorithm*) adalah suatu algoritma pencarian yang berbasis pada mekanisme seleksi alam dan genetika. Algoritma genetika merupakan salah satu algoritma yang sangat tepat digunakan dalam menyelesaikan masalah optimasi kompleks, yang sulit dilakukan oleh metode konvensional.

Algoritma genetika merupakan proses pencarian yang *heuristic* dan acak sehingga penekanan pemilihan operator yang digunakan sangat menentukan solusi optimum suatu masalah yang diberikan. Operator-operator yang digunakan pada algoritma genetika ada tiga yaitu operator seleksi, operator *crossover* (penyilangan) dan operator mutasi.

Permasalahan *travelling salesman problem* atau yang lebih dikenal dengan TSP merupakan permasalahan yang dapat diselesaikan dengan mengunakan algoritma genetika. Deskripsi permasalahannya yaitu “Diberikan sejumlah kota dan

jarak antar kota. Kita diminta untuk menentukan sirkuit terpendek yang dilalui oleh seorang pedagang. Pedagang tersebut berangkat dari sebuah kota asal, menyinggahi setiap kota tepat satu kali dan kembali lagi ke kota asal keberangkatan. Kota dapat dinyatakan sebagai simpul graf, sedangkan sisi menyatakan jalan yang menghubungkan antar dua buah kota. Bobot pada sisi menyatakan jarak antar dua buah kota".

Pada kasus TSP, ada beberapa metode dari operator seleksi pada algoritma genetika yang dapat digunakan, diantaranya *roulette wheel selection, rank selection,* dan *tournament selection* (Boukreev, 2001). *Roulette wheel selection* adalah metode seleksi yang memilih orang tua berdasarkan nilai kecocokannya (*fitness*). Kromosom yang lebih baik memiliki persentasi dipilih yang lebih besar. *Rank selection* adalah metode seleksi yang melakukan perankingan untuk populasi terlebih dahulu, dan setiap kromosom akan mendapat nilai kecocokan berdasarkan rankingnya pada populasi. *Tournament selection* merupakan suatu mekanisme pemilihan kromosom dalam suatu populasi, dimana sebuah grup (biasanya terdiri dari 2 sampai 7 kromosom) dipilih secara acak dan yang terbaik (biasanya cuma satu, namun mungkin lebih) dipilih dari salah satu golongan elit yang merupakan kromosom terbaik yang ditemukan sejauh ini.

Boukreev (2001) membadingkan ketiga metode seleksi *roulette wheel selection, rank selection* dan *tournament selection* berdasarkan pada aplikasi yang telah dibuatnya. Selain itu, dia hanya memfokuskan pada hasil waktu yang tercepat di antara ketiga metode untuk mendapatkan jarak yang terpendek.

Untuk melihat secara keseluruhan keefektifan metode *roulette wheel selection, rank selection* dan *tournament selection* dirumuskan empat aspek perbandingan yaitu *time complexity, space complexity, completeness* dan *optimality. Time complexity* adalah jumlah waktu yang diperlukan untuk menyelesaikan permasalahan. *Space complexity* adalah jumlah memori yang didapat pada saat solusi ditemukan pada suatu permasalahan. *Completeness* dapat diartikan apakah solusi untuk menyelesaikan permasalahan itu memang ada atau tidak. *Optimality* dapat diartikan bahwa solusi yang ditemukan merupakan solusi yang optimal.

Aspek pembanding antara metode *roulette wheel selection, rank selection* dan *tournament selection* yang hanya dilakukan dengan analisa secara tertulis yaitu dengan menggunakan notasi Big-O. Notasi Big-O adalah suatu nilai dari penyelesaian masalah dengan merujuk proses kerja dari penyelesaian masalah tersebut.

Berdasarkan penjabaran-penjabaran di atas, maka penulis bermaksud untuk membandingkan 3 buah metode dari operator seleksi dalam algoritma genetika yaitu metode *roulette wheel selection, rank selection* dan *tournament selection*. Perbandingan dilakukan untuk menentukan metode mana yang lebih efisien diterapkan, sehingga dalam pengimplementasian pada persoalan TSP, karakteristik yang cepat, tepat dan handal dapat tercipta.

1.2 Rumusan Masalah

Dari latar belakang yang sudah dikemukakan, dapat diambil suatu rumusan masalah yaitu "Bagaimana menganalisa perbandingan metode *roulette wheel selection, rank selection* dan *tournament selection* dalam algoritma genetika dengan studi kasus *travelling salesman problem* (TSP)".

1.3 Batasan Masalah

Batasan masalah dari penelitian ini adalah :

1. Graf TSP yang akan diselesaikan adalah graf lengkap dan berbobot.
2. Masalah yang akan diselesaikan adalah *symetric travelling salesman problem*, yaitu jarak dari i ke j sama dengan jarak dari j ke i.
3. Proses *encoding* yang digunakan adalah permutasi *encoding*, proses *crossover* dilakukan dengan *single point crossover* dan mutasi yang dilakukan adalah *random only improving*.

1.4 Tujuan

Adapun tujuan dari penelitian ini adalah :

1. Menganalisa secara manual dengan notasi Big-O untuk melakukan perbandingan terhadap metode *roulette wheel selection*, *rank selection* dan *tournament selection* dalam mengatasi permasalahan TSP menggunakan parameter COST.

2. Membuat suatu aplikasi yang dapat memperlihatkan perbandingan antara metode *roulette wheel selection*, *rank selection* dan *tournament selection* terhadap parameter COST dalam mengatasi permasalahan TSP.

1.5 Sistematika Penulisan

Sistematika penulisan tugas akhir ini dibagi menjadi 6 (enam) bab yang masing-masing bab telah dirancang dengan suatu tujuan tertentu. Berikut penjelasan tentang masing-masing bab :

BAB I PENDAHULUAN

Berisikan tentang deskripsi umum dari tugas akhir ini, yang meliputi latar belakang, rumusan masalah, batasan masalah, tujuan, serta sistematika penulisan.

BAB II LANDASAN TEORI

Berisi penjelasan tentang teori dasar algoritma, teknik pencarian, algoritma genetika, graf, *travelling salesman problem*, dan parameter pengukur keefektifan algoritma pencarian.

BAB III METODOLOGI PENELITIAN

Berisi pembahasan mengenai metodologi atau cara penelitian yang dilakukan dalam penyelesaian tugas akhir, yaitu pengumpulan data, analisa, perancangan, implementasi, pengujian dan membuat kesimpulan.

BAB IV ANALISA DAN PERANCANGAN

Berisi pembahasan mengenai analisis algoritma genetika menggunakan metode seleksi yang berbeda-beda terhadap kasus TSP dalam parameter COST, rangkuman hasil analisa keefektifan algoritma genetika menggunakan metode *roulette wheel selection*, *rank selection* dan *tournament selection* dalam parameter COST, model persoalan penyelesaian TSP dengan algoritma genetika menggunakan metode *roulette wheel selection, rank selection* dan *tournament selection*, dan perancangan simulasi yang mencakup lingkungan perancangan dan perancangan antarmuka.

BAB V IMPLEMENTASI DAN PENGUJIAN

Pada bab ini akan dibahas mengenai implementasi simulasi TSP dengan algoritma genetika, alasan pemilihan perangkat lunak, batasan implementasi, lingkungan implementasi, pengujian simulasi, hasil pengujian dan kesimpulan pengujian.

BAB VI PENUTUP

Dalam bab ini akan dijelaskan mengenai beberapa kesimpulan dan saran sebagai hasil akhir dari penelitian yang telah dilakukan.

BAB II

LANDASAN TEORI

Algoritma adalah metode yang dapat digunakan komputer untuk menyelesaikan masalah. Algoritma disusun dari sekumpulan langkah berhingga, masing-masing langkah mungkin memerlukan satu operasi atau lebih. Algoritma umumnya dirancang untuk menyelesaikan suatu masalah spesifik dan dengan usaha yang paling minimal (Hariyanto, 2003). Pada dasarnya algoritma pencarian dibagi dalam dua bagian, yaitu pencarian *heuristic* dan pencarian *blind search* atau *brute force*.

2.1 Teknik Pencarian

Teknik pencarian adalah suatu cara untuk melakukan pencarian dengan mempergunakan algoritma dan ketentuan matematika sehingga dapat ditentukan solusi dari permasalahan yang sedang diteliti. Teknik pencarian dan pelacakan dibedakan menjadi dua bagian yaitu (Kusumadewi, 2003) :

1. Teknik *blind search* (*uninformed search*/pencarian buta)
2. Teknik *heuristic search* (*informed search*/pencarian terbimbing)

2.1.1 *Blind Search* atau *Brute Force*

Blind search atau *brute force* adalah salah satu teknik pencarian yang tidak memiliki pengetahuan yang spesifik terhadap suatu masalah dalam mengembangkan

satu *node* ke *node* yang lainnya (Russel, 2003) atau teknik pencarian yang tidak mempunyai pengetahuan yang dapat digunakan untuk pencarian secara langsung (Suyoto, 2004).

Beberapa contoh algoritma pencarian dengan teknik *blind search* adalah *depth first search* (DFS), algoritma *tree search*, dan *breadth first search* (BFS).

2.1.2 *Heuristic Search*

Heuristic adalah prinsip atau informasi atau *knowledge* (bersifat *problem-specific*) yang dapat digunakan sebagai panduan dalam penelusuran untuk mencapai *goal states* dengan cara yang efektif. *Heuristic* juga dapat dikatakan sebagai estimasi seberapa dekat *current state* dengan *goal state* dan ia yang membedakan penelusuran yang bersifat "*intelligence*" dengan yang tidak. *Heuristic* tidak unik dan merupakan gabungan dari beberapa prinsip atau informasi namun tidak menjamin secara penuh dicapainya *goal states*. *Heuristic* adalah salah satu metode dari *informed search* dan dapat dianggap sebagai *pruning* (memotong pohon) dengan mempertimbangkan *node* yang *promising* (menjanjikan atau lebih pasti menuju *goal states*). Oleh karena itu, seyogyanya fungsi *heuristic* tidak terlalu rumit (sederhana dan mudah untuk dihitung) karena akan diaplikasikan ke setiap *node* (Russel, 2003).

Beberapa contoh algoritma pencarian *heuristic* adalah :

1. *Best first search* (BsFS)

   Merupakan metode yang membangkitkan suksesor dengan mempertimbangkan harga (didapat dari fungsi *heuristic* tertentu) dari setiap

node, bukan dari aturan baku seperti DFS maupun BFS. Untuk mengimplementasikan algoritma pencarian ini, diperlukan dua buah senarai, yaitu OPEN untuk mengelola *node-node* yang pernah dibangkitkan tetapi belum dievaluasi dan CLOSE untuk mengelola *node-node* yang pernah dibangkitkan dan sudah dievaluasi.

2. *Hill climbing*

   Strategi *hill climbing* mengembangkan *node* yang ada dalam pencarian dan mengevalusi anak-anaknya. Anak yang terbaik dipilih untuk ekspansi selanjutnya. Teknik *hill climbing* adalah teknik yang dipakai untuk mencapai tujuan dengan memilih *node-node* ke tujuan yang paling dekat pada proses pencarian di dalam suatu *graph* (Kusumadewi, 2004).

3. Algoritma A*

   Algoritma A* adalah sebuah algoritma yang telah diperkaya dengan menerapkan suatu *heuristic*, algoritma ini membuang langkah-langkah yang tidak perlu dengan pertimbangan bahwa langkah-langkah yang dibuang sudah pasti merupakan langkah yang tidak akan mencapai solusi yang diinginkan.

4. Algoritma genetika

   Algoritma genetika adalah algoritma pencarian data dan optimasi yang didasari pada proses mekanika alamiah, dimana sifat-sifat suatu spesies sangat bergantung pada gen-gen dan susunannya. Keberagaman pada evolusi biologis adalah variasi dari kromosom antar individu organisme. Variasi

kromosom ini akan mempengaruhi laju reproduksi dan tingkat kemampuan organisme untuk tetap hidup (Kusumadewi, 2003).

Proses dasar pada algoritma genetika adalah sebagai inisialisasi populasi awal, evaluasi setiap kromosom, seleksi, proses *crossover* dan mutasi.

2.2 Algoritma Genetika

Algoritma genetika adalah bagian dari perkembangan dalam ilmu komputer, yakni termasuk dalam bidang kecerdasan buatan (Obitko, 1998). Kecerdasan buatan (*artificial intelligence*) adalah studi untuk membuat komputer melakukan sesuatu dimana pada saat ini masih lebih baik bila dilakukan oleh manusia. Ide dari evolusi komputer diperkenalkan pada tahun 1960 oleh I. Rechenberg dalam kerjanya "*evolution strategies*". Algoritma genetika pertama kali dikembangkan oleh John Hollan dari Universitas Michigan, 1975 (Obitko, 1998). John Hollan mengatakan bahwa setiap masalah yang berbentuk adaptasi (alami maupun buatan) dapat diformulasikan dalam terminologi genetika (Kusumadewi, 2003).

Algoritma genetika terinspirasi dari teori Darwin tentang proses evolusi yang terjadi pada makhluk hidup, yaitu solusi untuk suatu masalah akan dipecahkan dengan cara berevolusi (Obitko, 1998). Proses evolusi ini bertujuan untuk menghasilkan keturunan yang lebih baik. Metode algoritma genetika bekerja dengan suatu fungsi biaya (*cost function*) sebagai fungsi yang menguji kualitas solusi yang dalam hal ini dilambangkan sebagai suatu individu dalam satu generasi. Suatu solusi akan dikodekan dengan kode string dan dapat dianggap sebagai DNA, kemudian akan

dikawinkan dengan solusi lainnya. Individu yang baru terlahir dianggap sebagai calon solusi baru.

Algoritma genetika adalah algoritma pencarian data dan optimasi yang didasari pada proses mekanika alamiah, dimana sifat-sifat suatu spesies sangat bergantung pada gen-gen dan susunannya. Keberagaman pada evolusi biologis adalah variasi dari kromosom antar individu organisme. Variasi kromosom ini akan mempengaruhi laju reproduksi dan tingkat kemampuan organisme untuk tetap hidup (Kusumadewi, 2003). Kunci dari algoritma genetika adalah pembangkitan secara acak turunan (kemungkinan) pola pemasangan untuk kemudian dicari mana yang paling optimal. Dari yang paling optimal, dibangkitkan lagi secara acak pola pemasangan yang baru dan kemudian dicari lagi optimasinya. Akan terjadi perputaran (*loop*) beberapa kali untuk mendapatkan hasil yang optimal.

Secara umum algoritma genetika ini banyak dipakai pada aplikasi bisnis, teknik maupun pada bidang keilmuan, namun secara khusus pemakaian algoritma genetika digunakan pada pencarian nilai tertentu dari suatu fungsi atau sistem dan pada pencarian nilai optimal dari suatu fungsi atau sistem (Thiang, 2001). Algoritma genetika ini juga dapat dipakai untuk mendapatkan solusi yang tepat untuk masalah optimal dari satu variabel atau multi variabel. Nilai yang akan dicari tidak harus selalu berupa bilangan, tetapi dapat juga berupa informasi tertentu. Begitu juga dengan fungsi atau sistem tidak selalu berarti fungsi matematis yang dinyatakan dengan $f(x,y)$ tetapi dapat juga berarti operasional yang dapat menghasilkan nilai.

Algoritma genetika digunakan pada sistem yang sulit untuk dimodelkan dengan model matematik, fungsi-fungsinya mempunyai variabel bebas yang sulit diprediksi.

Teknik pencarian pada algoritma genetika dilakukan sekaligus berdasarkan jumlah solusi yang mungkin yang dikenal dengan populasi. Individu yang terdapat dalam satu populasi disebut dengan istilah kromosom. Populasi awal dibangun secara acak, sedangkan populasi berikutnya merupakan hasil evaluasi kromosom-kromosom melalui iterasi yang disebut generasi. Pada setiap generasi, kromosom akan melalui proses evaluasi dengan menggunakan alat ukur yang disebut fungsi *fitness*. Nilai *fitness* dari suatu kromosom akan menunjukkan kualitas kromosom dalam populasi tersebut, generasi berikutnya dikenal dengan istilah anak (*offspring*), terbentuk dari gabungan 2 kromosom induk dengan menggunakan operator penyilangan (*crossover*). Selain operator penyilangan, suatu kromosom dapat juga dimodifikasi dengan menggunakan operator mutasi. Populasi generasi yang baru dibentuk dengan cara menyeleksi nilai *fitness* dari kromosom induk dan kromosom anak, serta menolak kromosom-kromosom lainnya sehingga ukuran populasi konstan (Kusumadewi, 2003).

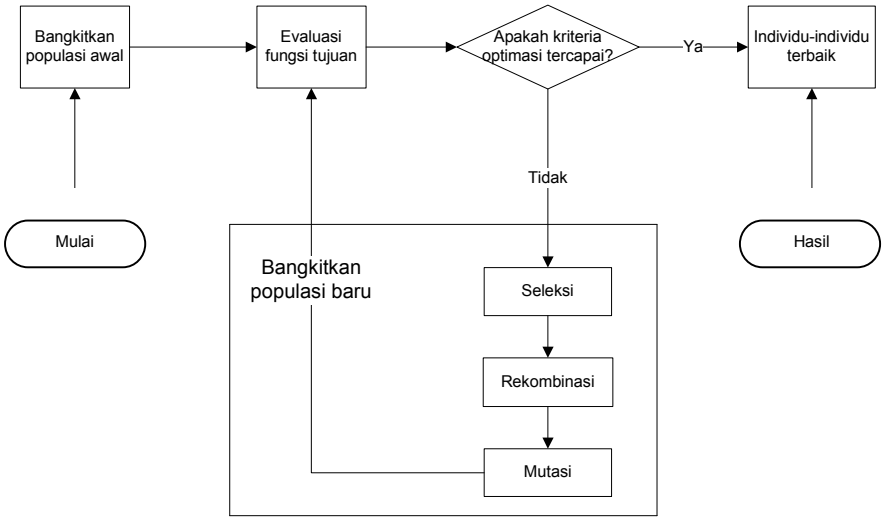
```
Line  function GA
 1    G ← Generasi;
 2    P ← Populasi;
 3    {Inisialisasi}
 4    for i=1 to P do
 5       Generate kromosom (i);
```

```
 6   end for
 7   while (current_gen<G) do
 8      Begin
 9        {evaluasi fungsi}
10        for k=1 to P do
11          Hitung nilai fungsi objektif dari kromosom (k);
12        end for
13        {Operator Seleksi}
14        while (populasi baru<=P) do
15          Pilih kromosom bertahan menggunakan metode seleksi;
16        end while
18        crossover_loop = (P*%c)/2;
19        {Operator crossover}
20        for l=1 to crossover_loop do
21          Acak 2 kromosom sebagai orang tua;
22          Generate anak kromosom menggunakan operasi crossover;
23        end for
24        mutasi_loop=P*%m;
25        {mutasi}
26        for m=1 to mutasi_loop do
27         Acak 1 kromosom sebagai orang tua;
28         Generate anak kromosom menggunakan operasi mutasi;
29        end for
30     end begin
31   end while
```

Algoritma 2.1 *Pseudocode* Algoritma Genetika (Pongcharoen, 2007)

Proses dasar pada algoritma genetika adalah sebagai berikut :

1. Inisialisasi populasi awal dengan kromosom secara acak,
2. Evaluasi setiap kromosom dalam populasi,
3. Lakukan seleksi,
4. Bangkitkan kromosom baru dengan melakukan proses *crossover* dan mutasi,
5. Hapus kromosom pada generasi sebelumnya untuk memberi ruang pada kromosom baru,
6. Evaluasi kromosom baru dan masukkan ke dalam populasi,
7. Berhenti sampai terpenuhi kriteria, jika tidak kembali ke langkah 3.



Gambar 2.1 Proses Dasar Algoritma Genetika (Kusumadewi, 2003)

Terdapat 6 komponen utama pada algoritma genetika, yaitu (Kusumadewi, 2003) :

1. Teknik penyandian
2. Prosedur inisialisasi
3. Fungsi evaluasi
4. Seleksi
5. Operator genetika
6. Penentuan parameter

2.2.1 Teknik Penyandian

Teknik penyandian yang dilakukan terhadap gen dari kromosom. Gen merupakan bagian dari kromosom, satu gen biasanya mewakili satu variabel. Gen dapat direpresentasikan dalam bentuk (Obitko, 1998) :

1. String bit : 10011, 01101, 11101
2. Permutasi : 1, 2, 3, 4, 5 atau a, b, c, d
3. Bilangan real : 65.25, 25.12, -45.36
4. Elemen permutasi : 1, 3, 5, 6
5. Daftar aturan : R1, R4, R8
6. Value : [black], [red], [white], atau
   1.2324, 5.3243, 0.4556, 2.3293, 2.4545
7. Elemen program : pemrograman genetika
8. *Tree encoding* : berupa pohon pencarian

2.2.2 Prosedur Inisialisasi

Ukuran populasi tergantung pada masalah yang akan dipecahkan dan jenis operator genetika yang akan diimplementasikan, kemudian harus dilakukan inisialisasi terhadap kromosom yang terdapat pada populasi tersebut. Inisialisasi kromosom pada populasi awal dilakukan secara acak dengan tetap memperhatikan domain solusi dan kendala permasalahan yang ada.

```
Line  procedure PopInitializer(GAPopulation & p)
 1    {membuat kromosom sebanyak jumlah populasi}
 2       for i=0 to i<p.size() do
 3          p.individual(i).initialize();
 4       end for
```

Algoritma 2.2 *Pseudocode* Proses Inisialisasi (Wall,1996)

Dalam membuat genom untuk TSP, tidak bisa digunakan model representasi standar. Karena setiap kota haruslah unik dalam gen dan tidak bisa diduplikasi. Oleh karena itu, digunakan model representasi sekuensial pada genom dimana setiap kota di list pada urutan yang keberapa kota itu dikunjungi (Widhiyasa, 2007).

2.2.3 Fungsi Evaluasi

Ada dua hal yang harus dilakukan dalam melakukan evaluasi kromosom, yaitu evaluasi fungsi objektif (fungsi tujuan) dan konversikan fungsi objektif ke dalam fungsi *fitness*.

```
Line  procedure PopEvaluator(GAPopulation & p)
 1    {mengevaluasi kromosom sebanyak jumlah populasi}
 2       for i=0 to i<p.size() do
 3          p.individual(i).evaluate();
 4       end for
```

Algoritma 2.3 *Pseudocode* Proses Evaluasi (Wall,1996)

2.2.4 Seleksi

Seleksi ini bertujuan untuk memberikan kesempatan reproduksi yang lebih besar bagi anggota populasi yang paling baik. Proses seleksi akan menentukan individu-individu mana saja yang akan dipilih untuk dilakukan rekombinasi. Langkah pertama yang harus dilakukan adalah pencarian nilai *fitness*. Ada beberapa metoda seleksi yang terdapat dalam algoritma genetika, antara lain :

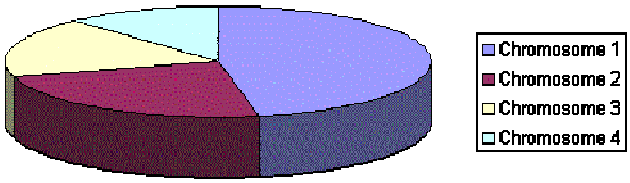
a. Pemilihan acak (*random selection*)

   Metode ini akan mengembalikan nilai *fitness*, metode ini akan melakukan pencarian buta kecuali jika digunakan dalam kombinasi dengan metode yang lain.

b. Pemilihan roda roulette (*roulette wheel selection*)

   Metode seleksi *roulette wheel selection* merupakan metode yang paling sederhana, dan sering juga dikenal dengan nama *stochastic sampling with replacement.* Metode ini memilih satu individu dengan kemungkinan proporsi secara langsung untuk nilai *fitness*-nya, yaitu memilih kromosom terbaik dengan cara menghitung setiap nilai kromosom dan membandingkannya

dengan nilai kromosom lainnya. *Roulette wheel selection* memilih orang tua berdasarkan nilai kecocokannya (*fitness*). Kromosom yang lebih baik memiliki presentasi dipilih yang lebih besar. Pada metode ini, individu-individu dipetakan dalam suatu segmen garis secara berurutan sedemikian hingga tiap-tiap segmen individu memiliki ukuran yang sama dengan ukuran *fitness*-nya.



Gambar 2.2 Diagram Pie Metode *Roulette Wheel Selection*

Gambar 2.2 adalah contoh diagram pie yang merepresentasikan kromosom-kromosom hasil dari *roulette wheel selection*. Warna dominan adalah kromosom dengan probabilitas paling banyak untuk dipilih.

Nilai fitness pada seleksi ini dihitung dengan rumus :

$$Fitness_i = 1/Nilai\_Objektif_i \quad (2.1)$$

Probabilitas tiap kromosom dihitung dengan menggunakan rumus :

$$Prob_i = f_i/(f_1+f_2+\ldots\ldots+f_N) \quad (2.2)$$

```
Line   procedure RWS
  1    {Melakukan proses seleksi untuk mendapatkan
  2     Kromosom-kromosom yang bertahan}
```

```
3    Hitung Nilai fitness dan total fitness;
4    Hitung Probabilitas P;
5    Acak R antara 0-1;
6    Sum=0;
7    for i=1 to Populasi do
8       Sum=sum+P_i;
9       if (Sum>=R)
10         return i;
11      end if
12   end for
```

Algoritma 2.4 *Pseudocode* Proses Seleksi *Roulette Wheel Selection* (www.iba.k.u-tokyo.ac.jp ,2007)

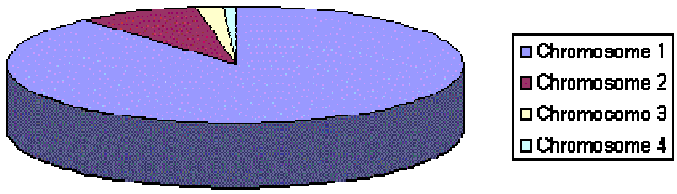
Rawlins (1991) dalam bukunya menyatakan bahwa *time complexity* untuk metode *roulette wheel selection* yaitu :

$$T(n) = O(n^2) \qquad (2.3)$$

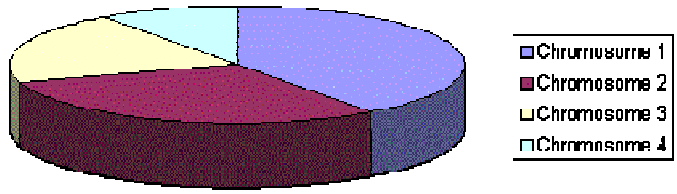
c. Pemilihan dengan rangking (*rank selection*)

Metode *roulette wheel selection* memiliki masalah ketika nilai kecocokannya berbeda sangat jauh. Misalkan saja kromosom terbaik memiliki nilai kecocokan (*fitness*) sebesar 90%. Maka kemungkinan kromosom lain dipilih akan menjadi sangat kecil. Dengan metode *rank selection*, pertama-tama akan dilakukan perankingan untuk populasi dan setiap kromosom akan mendapat nilai *fitness* berdasarkan rankingnya pada populasi. Kromosom terburuk akan mendapat nilai 1, kedua terburuk akan mendapatkan nilai 2, dan seterusnya,

yang terbaik akan mendapatkan nilai *n*, dimana *n* adalah jumlah total kromosom pada populasi.



Gambar 2.3 Diagram Pie sebelum Ranking (Diagram berdasarkan *Fitness)*



Gambar 2.4 Diagram Pie sesudah Ranking (Diagram berdasarkan Urutan)

Nilai *fitness* pada seleksi ini dihitung dengan rumus :

$$\text{Fitness}_i = \text{Populasi } size\text{-}(i\text{-}1) \qquad (2.4)$$

```
Line   procedure RS
 1     {Melakukan proses seleksi untuk mendapatkan
 2      Kromosom-kromosom yang bertahan}
 3     Buat Ranking dan beri nilai fitness;
 4     Hitung Probabilitas P;
 5     Acak R antara 0-1;
 6     sum=0;
 7     for i=1 to Populasi do
 8          sum=sum+Pi;
```

```
 9          if (sum>=r)
10               return i;
11          end if
12     end for
```

Algoritma 2.5 *Pseudocode* Proses Seleksi *Rank Selection*
(www.iba.k.u-tokyo.ac.jp ,2007)

Rawlins (1991) dalam bukunya menyatakan bahwa *time complexity* untuk metode *rank selection* yaitu :

$$T(n) = O(n \log n) \quad (2.5)$$

d. *Steady-state selection*

Ide utama dari metoda ini adalah terdapat bagian besar dari kromosom yang harus bertahan untuk generasi selanjutnya. Algoritma genetika akan bekerja sebagai berikut :

i. Dalam setiap generasi akan dipilih sedikit kromosom untuk membentuk keturunan baru.
ii. Kemudian beberapa kromosom dipindahkan dan keturunan yang baru akan menempati tempat tersebut.
iii. Kriteria berhenti dari populasi bertahan untuk generasi baru.

e. *Tournament selection*

Pada seleksi alami yang terjadi di dunia nyata, beberapa individu (biasanya individu jantan) berkompetisi dalam sebuah kelompok kecil samapi tersisa hanya satu individu pemenang. Individu pemenang inilah yang bisa kawin

(pindah silang). Metode *roulette wheel selection* tidak mengakomodasi masalah ini. *Tournament selection* mencoba mengadopsi karakteristik alami ini.

Dalam bentuk paling sederhana, metode ini mengambil dua kromosom secara *random* dan kemudian menyeleksi salah satu yang bernilai *fitnees* paling tinggi untuk menjadi orang tua pertama. Cara yang sama dilakukan lagi untuk mendapatkan orang tua kedua. Selain itu, pemilihan kromosom dapat dilakukan dengan cara sebuah grup (biasanya terdiri dari 2 sampai 7 kromosom) dipilih secara acak dan yang terbaik (biasanya cuma satu, namun mungkin lebih) dipilh dari sala satu golongan elit yang merupakan kromosom terbaik yang ditemukan sejauh ini. Metode *tournament selection* yang lebih rumit adalah dengan mengambil *m* kromosom secara *random*, kemudian kromosom bernilai *fitness* tertinggi dipilih sebagai orang tua pertama jika bilangan *random* yang dibangkitkan kurang dari suatu nilai batasan yang ditentukan *p* dalam integral [0,1]. Pemilihan orang tua akan dilakukan secara *random* dari *m*-1 kromosom yang ada jika bilangan *random* yang dibangkitkan lebih dari atau sama dengan *p*. pada *tournament selection,* variabel *m* adalah *tournament size* dan *p* adalah *tournament probability.* Biasanya *m* diset sebagai suatu nilai yang sangat kecil misal 4 atau 5, sedangkan *p* biasanya diiset sekitar 0,75.

```
Line  procedure TS
 1    {Melakukan proses seleksi untuk mendapatkan
 2     Kromosom-kromosom yang bertahan}
 3    Repeat
 4      Ambil k random kromosom dalam populasi;
 5      Dari kromosom k, ambil kromosom dengan nilai
           terbaik;
 6      if kromosom terbaik sama, ambil salah satu
 7    Until (2 orang tua terpilih)
```

Algoritma 2.6 *Pseudocode* Proses Seleksi *Tournament Selection* (Jaakkola ,2008)

Rawlins (1991) dalam bukunya menyatakan bahwa *time complexity* untuk metode *tournament selection* yaitu :

$$T(n) = O(n) \qquad (2.6)$$

f. *Elitism*

Ketika membuat populasi baru dengan cara *crossover* dan mutasi, terdapat kesempatan yang akan menyebabkan hilangnya kromosom terbaik. Metode ini pertama-tama akan menyalin kromosom terbaik untuk populasi baru. Metode ini dapat sangat cepat dalam mengurangi *perform* dari algoritma genetika, karena mencegah ditemukannya solusi terbaik.

2.2.5 Operator Genetika

Ada 2 operator dalam algoritma genetika, yaitu (Kusumadewi, 2003) :

1. Operator untuk melakukan rekombinasi, yang terdiri dari :
   a. Rekombinasi bernilai real
      i. Rekombinasi diskret
      ii. Rekombinasi menengah
      iii. Rekombinasi garis
   b. Rekombinasi garis yang diperluas
   c. Rekombinasi bernilai biner (*crossover*)

      Crossover adalah memilih gen dari kromosom orang tua dan menghasilkan keturunan baru. Cara yang sederhana untuk melakukan langkah ini adalah memilih secara acak beberapa nilai *crossover* dan setiap sebelumnya nilai tersebut disalin dari orang tua pertama kemudian setiap sesudahnya nilai tersebut disalin dari orang tua kedua. Operator ini bertanggung jawab untuk menghasilkan kromosom anak. Metoda yang digunakan untuk proses *crossover* adalah :

      i. Satu nilai (*one point*), bagian dari kromosom orang tua pertama disalin dan disaat lain diambil dalam order yang sama dengan yang berada dalam orang tua kedua. Orang tua ditukarkan secara acak pada awal *crossover*.

Contoh :

*11001*100 + 11011111 = *11001*111

(123456789) + (45321*6897*) = (12345*6897*)

One point crossover yang dikembangkan oleh revees adalah (Gen, 1997) :

1. Memilih satu *cut-point* secara *random*/acak dari *parent* pertama,
2. Isi di sebelah kanan site disesuaikan dengan urutan dari *parent* kedua untuk menghasilkan *offspring*.

```
Line   procedure SinglePoint Crossover
 1     (const GAGenome& p1, const GAGenome& p2,
 2     GAGenome* c1, GAGenome* c2)
 3     {Melalukan persilangan antara 2 orang tua}
 4     GA1DBinaryStringGenome
 5     &mom=(GA1DBinaryStringGenome &)p1;
 6     GA1DBinaryStringGenome
 7     &dad=(GA1DBinaryStringGenome &)p2;
 8     n=0;
 9     site = GARandomInt(0, mom.length());
10     len = mom.length() - site;
11     if(c1) {
12     GA1DBinaryStringGenome
13     &sis=(GA1DBinaryStringGenome &)*c1;
14          Sis.copy(mom, 0, 0, site);
15          Sis.copy(dad, site, site, len);
```

```
16    n++;
17    }
18    if(c2) {
19    GA1DBinaryStringGenome
20    &bro=(GA1DBinaryStringGenome &)*c2;
21          Bro.copy(dad, 0, 0, site);
22          Bro.copy(mom, site, site, len);
23    n++;
24    }
25    return n;
```

Algoritma 2.7 *Pseudocode* Proses *Crossover* (Wall ,1996)

ii. Dua nilai (*two point*), dua bagian dari orang tua pertama disalin dan saat berhenti diantaranya akan diambil dalam order yang sama dengan orang tua kedua. Orang tua ditukarkan secara acak pada substring.

Contoh :

11001011 + 11011110 = 11011111

iii. Penyilangan seragam (*uniform crossover*), setiap bit anak dalam satu kromosom dipilih secara acak dari orang tua pertama atau kedua.

Contoh : 11001011 + 11011101 = 11011111

iv. Penyilangan aritmatika (*arithmetic crossover*), beberapa operasi aritmatika dilakukan untuk membuat keturunan baru, seperti AND, OR, XOR.

Contoh : 11001011 + 11011111 = 11001001 (AND)

Banyaknya proses *crossover* yang dilakukan dalam populasi ditentukan dengan rumus :

crossover_loop = Populasi *size**%Pc/2 (2.7)

2. Mutasi

Proses ini dilakukan untuk menempatkan semua solusi dalam populasi ke dalam sebuah lokal optimum dari pemecahan masalah. Mutasi adalah menukar keturunan baru secara acak. Metode yang bisa digunakan adalah :

a. Mutasi secara normal acak (*normal random*), beberapa kota dipilih dan ditukarkan.
b. *Random only improving*, beberapa kota dipilih secara acak dan akan ditukarkan jika memiliki solusi yang lebih baik (nilai *fitness* bertambah).

```
Line  procedure Mutasi
 1    {Mengambil kromosom yang akan dimutasi}
 2    Mutate = 0;
 3    if(Math.random() < Mutation_probability) {
 4       iswap1 = acak gen;
 5       iswap2 = acak gen;
 6       temp = off[iswap1];
 7       Off[iswap1] = off1[iswap2];
 8       Off[iswap2] = temp;
 9    mutate++;
10    }
```

```
11    return mutate;
```

Algoritma 2.7 *Pseudocode* Proses Mutasi (JJB ,1999)

c. *Sistematic only improving*, kota secara teratur dipilih dan ditukarkan jika memiliki solusi yang lebih baik (nilai *fitness* bertambah).

d. *Random improving*, sama dengan metoda *Random only improving*, namun sebelumnya dilakukan mutasi *random* biasa.

e. *Sistematic improving*, sama dengan metoda *Sistematic only improving*, namun sebelumnya dilakukan mutasi *random* biasa.

Banyaknya proses mutasi yang dilakukan dalam populasi ditentukan dengan rumus :

mutasi_*loop* = Populasi *size**%Pm (2.8)

2.2.6 Penentuan Parameter

Yang disebut parameter disini adalah parameter kontrol Algoritma Genetika, yaitu ukuran populasi (*popsize*), peluang *crossover* (Pc), dan peluang mutasi (Pm). Ada beberapa rekomendasi yang bisa digunakan (Kusumadewi, 2003) :

1. Untuk permasalahan yang memiliki kawasan solusi cukup besar, nilai parameter kontrolnya :

   (*popsize*; Pc; Pm) = (50; 0.6; 0.001)

2. Bila rata-rata *fitness* setiap generasi digunakan sebagai indikator, maka nilai parameter kontrolnya :

(*popsize*; Pc; Pm) = (30; 0.95; 0.01)

3. Bila *fitness* dari individu terbaik dipantau pada setiap generasi, maka usulannya adalah :

   (*popsize*; Pc; Pm) = (80; 0.45; 0.01)

4. Ukuran populasi sebaiknya tidak lebih kecil dari 30.

2.2.7 Termination

Proses algoritma genetika akan terus dilakukan sampai suatu kondisi terminasi/berhenti ditemukan. Kondisi terminasi/berhenti yang umum dipergunakan yaitu (Widhiyasa, 2007) :

a. Suatu solusi ditemukan yang memenuhi kriteria minimum.
b. Generasi telah mencapai suatu tingkat tertentu.
c. *Budget* yang dialokasikan (misalnya waktu komputasi) telah dicapai.
d. Solusi dengan nilai kecocokan tertinggi akan mencapai atau telah mencapai suatu batas dimana proses selanjutnya yang akan dilakukan tidak akan menghasilkan hasil yang lebih baik.
e. Inspeksi secara manual dan berkala.
f. Kombinasi dari berbagai macam cara.

2.3 Graf

2.3.1 Definisi Graf

Graf G didefenisikan sebaga pasangan himpunan (V,E), yang dalam hal ini (Munir, 2001) :

V = himpunan tidak kosong dari simpul-simpul

= $\{v_1, v_2, v_3, ..., v_n\}$

E = himpunan sisi (*edges* atau *arcs*) yang menghubungkan sepasang simpul

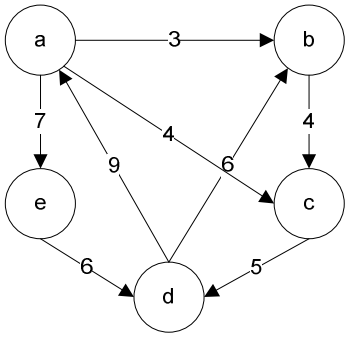
= $\{e_1, e_2, e_3, ..., e_n\}$

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

2.3.2 Jenis-jenis Graf

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

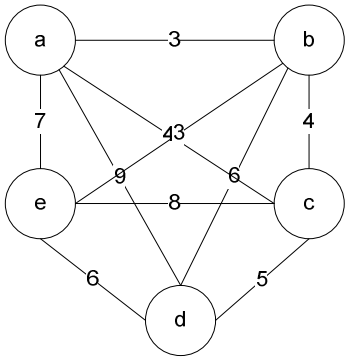
1. Graf berarah dan berbobot, yaitu tiap busur mempunyai anak panah dan bobot



Gambar 2.5 Graf Berarah dan Berbobot

Gambar 2.5 menunjukkan graf berarah dan berbobot yang terdiri dari lima titik yaitu a, b, c, d dan e. Titik a menunjukkan arah ke titik b dan e, titik b menunjukkan arah ke titik c, dan seterusnya. Bobot antara titik a ke titik b dapat diketahui yaitu 3.

2. Graf tidak berarah dan berbobot, yaitu tiap busur tidak mempunyai anak panah tetapi mempunyai bobot.

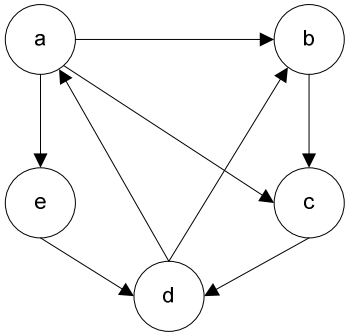


Gambar 2.6 Graf Tidak Berarah dan Berbobot

Gambar 2.6 menunjukkan graf berarah dan berbobot yang terdiri dari lima titik yaitu a, b, c, d dan e. Titik a tidak menunjukkan arah ke titik b, c, d dan e,

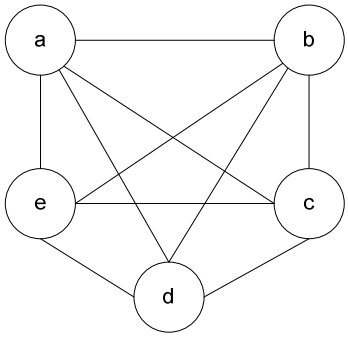
titik b tidak menunjukkan arah ke titik c, d dan e, dan seterusnya. Bobot antara titik a ke titik b dapat diketahui yaitu 3.

3. Graf berarah dan tidak berbobot, yaitu tiap busur mempunyai anak panah dan tidak berbobot.



Gambar 2.7 Graf Berarah dan Tidak Berbobot

4. Graf tidak berarah dan tidak berbobot, yaitu tiap busur tidak mempunyai anak panah dan tidak mempunyai bobot.



Gambar 2.8 Graf Tidak Berarah dan Tidak Berbobot

2.4 *Travelling Salesman Problem*

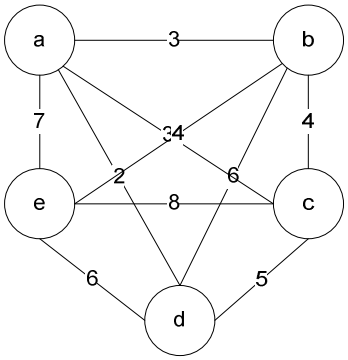
Travelling salesman problem (TSP) termasuk dalam persoalan yang sangat terkenal dalam teori graf. Nama persoalan ini diilhami oleh seorang pedagang yang mengunjungi sejumlah kota. Uraian persoalan ini adalah sebagai berikut :

1. Diberikan sejumlah kota dan jarak antar kota.
2. Tentukan sirkuit terpendek yang harus dilalui oleh seorang pedagang bila pedagang itu bergerak dari sebuah kota asal dan menyinggahi setiap kota tepat satu kali dan kembali ke kota asal keberangkatan.

Untuk persoalan di atas kota dapat dinyatakan sebagai simpul graf, sedangkan sisi menyatakan jalan yang menghubungkan antar dua buah kota. Bobot pada sisi menyatakan jarak antara dua buah kota.

Banyaknya perjalanan keliling TSP yang mungkin pada K*n* adalah (*n*-1)!/2. perhitungan ini sederhana karena suatu perjalanan keliling adalah suatu permutasi dari *n* puncak. Bagaimanapun, banyaknya permutasi berbeda, *n*! dikurangi oleh suatu faktor 1/2*n* sebab arah i ke j sama dengan j ke i dan puncak pertama bebas.

Sebagai ilustrasi atas pengertian kita terhadap TSP, diberi suatu contoh persoalan. Pertimbangkan graf G=(V,E) pada Gambar 2.9, dimana v={a,b,c,d,e} sejumlah kota yang harus dikunjungi dan E={(a,b),(a,c),(a,d),(a,e),(b,c),(b,d),(b,e),(c,d),(c,e),(d,e)} merupakan busur penghubung antar dua kota dimana ongkos dari kota i ke kota j sama dengan ongkos dari kota j ke kota i.



Gambar 2.9 Graf TSP Simetris

Misalkan seorang pedagang berangkat dari kota a dan ia harus mengelilingi setiap kota satu kali dan kembali ke kota awal keberangkatan, jalur manakah yang harus dilalui oleh pedagang supaya memberikan total ongkos paling minimum? Di sini terdapat beberapa solusi yang mungkin antaranya T_1=(a,d,c,b,e) dan T_2=(a,d,e,b,c). ongkos perjalanan lebih kecil yang dimiliki T_2 yakni 19 dibandingkan dengan ongkos perjalanan T_1 yakni 21. dengan jelas ditunjukkan bahwa T_2 solusi yang mungkin untuk contoh permasalahan ini sebagai perjalanan keliling TSP optimal.

2.4.1 Sejarah *Travelling Salesman Problem*

Awal munculnya TSP dikemukakan oleh Karl Menger seorang matematikawan pada tahun 1930-an. Menger mendefinisikan persoalan sebagai tugas penemuan pada sejumlah titik yang masing-masing jaraknya diketahui. Aturannya adalah berangkat dari titik awal ke titik terdekat, dan seterusnya hingga diperoleh rute terpendek (Munir, 2001).

Hingga bertahun-tahun persoalan ini menjadi sebuah persoalan sulit dalam optimasi kombinatorial yaitu menguji rute perjalanan satu persatu. Pada tahun 1949 Julia Robinson memberikan solusi integer dengan mempublikasikan tulisannya yang berjudul "*On the Hamiltonian Game*". Dalam tulisannya tersebut didapatkan sebuah metode untuk memecahkan persoalan yang berhubungan dengan TSP. di tahun 1950-an banyak ilmuwan yang mencoba memecahkan persoalan TSP ini, seperti B. Dantzig, Fulkerson, and Johnson, yang memberikan rumusan persoalan TSP (Cummings, 2002).

Perumusan TSP yang asli sebagai suatu program bilangan bulat dihubungkan dengan Dantzig et al. (1954) dimana masalah dirumuskan sebagai program linier nol-satu dengan menghubungkan masing-masing tepi e suatu variabel biner X(e). satu-satunya sasaran mungkin berfungsi dalam perumusan ini akan memperkecil suatu penjumlahan variabel tepi. Jika berat mewakili ongkos keliling sepanjang suatu tepi, sasaran mempunyai arti fisik pengecilan total panjang perjalanan keliling (Cummings, 2002).

2.4.2 Aplikasi *Travelling Salesman Problem*

Persoalan TSP adalah untuk menentukan sirkuit Hamilton yang memiliki bobot minimum pada sebuah graf terhubung. Pada perkembangan selanjutnya TSP tidak hanya digunakan pada persoalan pedagang keliling saja, namun digunakan juga dalam permasalahan distribusi untuk mencari urutan rute dari suatu lokasi awal

melewati seluruh pelanggan dan terakhir kembali lagi ke lokasi awal tersebut dengan biaya perjalan semurah-murahnya.

TSP ini sangat banyak digunakan untuk masalah sehari-hari, sebagai contoh (Munir, 2001) :

1. Sebuah mobil pos ditugaskan mengambil surat dari kotak pos yang tersebar pada *n* buah lokasi di berbagai sudut kota. Graf dengan *n*+1 simpul dapat digunakan untuk menyajikan persoalan, satu simpul menyatakan kantor pos tempat mobil pos mulai berangkat. Sisi (i,j) diberi bobot yang sama dengan jarak dari kotak pos i ke kotak pos j. Rute yang dilalui mobil pos adalah sebuah perjalanan (*tour*) yang mengunjungi setiap kotak pos hanya satu kali dan kembali lagi ke kantor pos asal. Kita harus menentukan rute perjalanan yang mempunyai total jarak terpendek.
2. Misalkan kita ingin menggunakan lengan robot untuk mengencangkan mur pada beberapa buah peralatan mesin dalam sebuah jalur perakitan. Lengan robot mulai berada dari posisi awalnya (yaitu di atas mur pertama, kemudian mengencangkannya), lalu berturut-turut pindah ke mur berikutnya dan kembali lagi ke posisi awal. Siklus yang dibentuk jelaslah perjalanan mengunjungi simpul-simpul sebuah graf. Tiap simpul menyatakan mur, sisi menyatakan perpindahan dan bobot setiap sisi menyatakan waktu yang diperlukan lengan robot untuk berpindah di antara dua simpul. Perjalanan dengan biaya minimum berarti meminimumkan waktu yang dibutuhkan robot untuk menyelesaikan tugasnya yaitu total waktu perpindahan dari sebuah mur

ke mur lainnya, sedangkan waktu pengencangan mur tidak dimasukkan dalam perhitungan.

3. Dalam lingkungan produksi terdapat beberapa komoditi yang dihasilkan oleh sekumpulan mesin. Proses fabrikasi merupakan siklus. Tiap-tiap siklus produksi menghasilkan *n* komoditi berbeda. Bila pekerjaan mesin diubah dari produksi komoditas i ke komoditas j, perubahan tersebut mendatangkan biaya sebesar C*ij*. Diinginkan untuk menemukan runtunan produksi yang menghasilkan *n* komoditas ini. Runtunan tersebut harus meminimumkan total biaya akibat perubahan urutan produksi (biaya lainnya tidak tergantung pada runtunan). Karena komoditas diproses secara siklus, adalah perlu untuk memasukkan biaya untuk memulai siklus berikutnya. Ini adalah biaya akibat perubahan produksi komoditi terakhir ke komoditi pertama. Masalah ini dapat dianggap sebagai MPP pada graf *n* simpul dengan sisi yang bobotnya C*ij*.

Pada persoalan TSP ini, jika setiap simpul mempunyai sisi ke simpul lainnya, maka graf yang akan merepresentasikanya adalah graf lengkap berbobot. Pada sembarang graf lengkap dengan *n* buah simpul ($n>2$), jumlah sirkuit Hamilton yang berbeda adalah $(n-1)!/2$. rumus ini dihasilkan dari kenyataan bahwa dimulai dari sembarang simpul kita mempunyai $n-1$ buah sisi untuk dipilih dari simpul pertama, $n-2$ sisi dari simpul kedua, $n-3$ sisi dari simpul ketiga, dan seterusnya. Ini adalah pilihan yang independen, sehingga kita memperoleh $(n-1)!$ pilihan. Jumlah itu harus dibagi dengan 2, karena tiap sirkuit Hamilton terhitung dua kali, sehingga semuanya adalah $(n-1)!/2$ buah sirkuit Hamilton.

Persoalan TSP adalah persoalan yang sulit (*hard problem*) dipandang dari sudut komputasinya. Artinya secara teoritis TSP dapat dipecahkan dengan mengenumerasikan (*n*-1)!/2 buah sirkuit Hamilton, menghitung panjang rute masing-masing sirkuit, dan kemudian memilih sirkuit yang memiliki panjang rute terpendek. Untuk *n* yang besar, jumlah sirkuit Hamilton yang harus diperiksa akan semakin banyak. Misalnya untuk 20 simpul (*n*=20) maka akan terdapat 19!/2 sirkuit Hamilton atau sekitar 6*1000000 penyelesaian.

2.4.3 Variasi dalam *Travelling Salesman Problem*

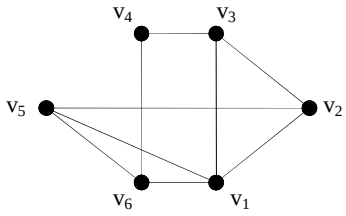
Macam-macam *travelling salesman problem* (Reinelt, 2001) :

1. *Symmetric travelling salesman problem* (TSP)

   Diberikan sekumpulan *n node* dan jarak dari tiap pasangan *node*, TSP menemukan jalur perjalanan dari total jarak minimal yang digunakan untuk mengunjungi setiap *node* satu kali. Jarak dari *node* i ke *node* j sama dengan jarak dari j ke i. untuk TSP *symmetric*, jumlah jalur yang mungkin yakni diberikan sebanyak (*n*-1)!/2.

2. *Asymmetric travelling salesman problem* (ATSP)

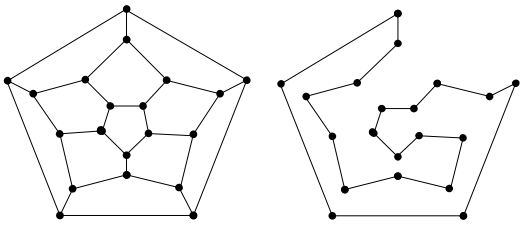
   Diberikan sekumpulan *n node* dan jarak dari tiap pasangan *node*, ATSP menemukan jalur perjalanan dari total jarak minimal yang digunakan untuk mengunjungi setiap *node* satu kali. Dalam kasus ini, Jarak dari *node* i ke *node* j dan jarak dari *node* j ke *node* i akan berbeda (Vicky, 2001).



Gambar 2.10 Simpul Graf

3. *Hamiltonian cycle problem* (HCP)

   Diberikan sebuah grafik, lakukan pengujian apakah graf berisi hamiltonian *cycle* atau tidak. Hamiltonian *cycle* adalah sebuah jalur yang melalui sebuah graf dengan dimulai dan diakhiri pada *vertex* yang sama dan setiap *vertex* hanya dikunjungi satu kali. Metode di atas ditemukan oleh W.R. Hamilton. HCP adalah masalah spesial untuk TSP dengan jarak tiap kota menjadi 1. deskripsi input untuk hcp + a graf g = (v,e).



Gambar 2.11 Input dan Output Hamiltonian *Cycle Problem*

4. *Sequential ordering problem* (SOP)

   Problem ini sama dengan *symmetric travelling salesman problem* dengan penambahan konstrain. Diberikan sekumpulan set dari *n node* dan jarak untuk setiap *node*, temukan jalur hamiltonian dari *node* 1 untuk *n node* dalam jarak

minimal dengan penganbilan konstrain *precedence* dalam penjumlahan. SOP dengan *precedence* konstrain berisi bobot minimum jalur hamiltonian dalam graf langsung dengan bobot dalam *node*. SOP dapat diformulasikan sebagai kasus umum dari ATSP.

5. *Capacitated vehicle routing problem* (CVRP)

   Diberikan *n*-1 *node*, satu depot atau stasiun dan jarak dari *node* ke depot. Semua *node* memiliki kebutuhan yang dapat dipenuhi oleh gudang. Untuk mengantar kepada tiap *node*, sejumlah truk telah tersedia dengan identifikasi kapasitas. Masalahnya adalah menemukan perjalanan yang memiliki total panjang minimal untuk truk yang memenuhi permintaan *node* tanpa mengganggu kapasitas truk. Jumlah dari truk tidak spesifik. Setiap perjalanan akan mengunjungi subset dari *node* serta memulai dan mengakhiri pada stasiun. CVRP adalah VRP dengan penambahan konstrain. Tujuan dari CVRP adalah untuk meminimalkan kendaraan dan menjumlahkan waktu perjalanan, juga total permintaan dari komoditi untuk setiap rute yang tidak melebihi kapasitas dari kendaraan. Solusi untuk CVRP sama dengan VRP, tapi dengan penambahan batasan dengan total permintaan dari pelanggan dalam rute (Reinelt, 2001).

6. *The euclidean travelling salesman selection problem*

   Diberikan *n* buah kota ditempatkan dalam pesawat, tujuan dari metoda ini menemukan rute terpendek dari semua kota yang dikunjungi, dengan syarat setiap kota hanya dikunjungi satu kali dan kembali ke kota tempat memulai

perjalanan. Masalah yang timbul pada metoda ini adalah jika salah satu kota yang akan dikunjungi dipindahkan. Perpindahan tersebut akan mengakibatkan perbedaan rute yang dijalani.

7. *On-line travelling salesman problem*

   Dalam *online* TSP permintaan untuk mengunjungi kota muncul secara langsung pada saat salesman melakukan perjalanan. Tujuannya adalah menemukan rute untuk pedagang dengan hasil akhir yang paling mudah. Dua versi dari masalah sudah didefenisikan terlebih dahulu. Perbedaannnya adalah apakah pedagang kembali ke tempat *node* memulai atau tidak.

2.5 Parameter Pengukur Keefektifan Algoritma Pencarian

Untuk melihat keefektifan suatu *searching algorithm*, (Russel, 2003) merumuskan 4 (empat) parameter sebagai berikut :

a. *Time complexity*, yang menyatakan waktu yang diperlukan untuk mencapai sasaran. Ini sangat berkaitan erat dengan cpu *time* dan *branching factor*. *Time complexity* dapat dihitung dengan menggunakan notasi Big-O.
b. *Space complexity*, yang mengukur jumlah memori yang dibutuhkan untuk implementasi *search* dan diukur dalam bentuk besar *byte*. *Space complexity* dapat dihitung dengan menggunakan notasi Big-O.
c. *Completeness*, yang mengukur jaminan bahwa *goal state* dicapai oleh *search* berdasarkan pada *searching algorithm* yang dipakai.

d. *Optimality*, yang memberikan ukuran jaminan bahwa *solution path* adalah paling minimum.

2.5.1 Pengertian Notasi Big-O

Notasi Big-O adalah suatu nilai dari penyelesaian masalah dengan merujuk proses kerja dari penyelesaian masalah tersebut. Sebuah algoritma tidak saja harus benar, tetapi juga harus efisien. Keefisienan algoritma diukur dari beberapa jumlah waktu dan ruang (*space*) memori yang dibutuhkan untuk menjalankannya. Kebutuhan waktu dan ruang suatu algoritma bergantung pada ukuran masukan (*n*), yang menyatakan jumlah data yang diproses. Dengan menggunakan besaran kompleksitas waktu/ruang algoritma, dapat menentukan laju peningkatan waktu/ruang yang diperlukan algoritma dengan meningkatnya ukuran masukan *n*.

Adapun teorema Big-O (Munir, 2003) :

Misalkan T1(n) = O (f(n)) dan T2(n) = O(g(n)), maka

(a) $T1(n) + T2(n) = O(f(n)) + O(g(n)) = O(\max f(n), g(n))$ (2.9)

(b) $T1(n)T2(n) = O(f(n))O(g(n)) = O(f(n)g(n))$ (2.10)

(c) $O(cf(n)) = O(f(n))$, c adalah konstanta (2.11)

(d) $f(n) = O(f(n))$ (2.12)

2.5.2 Kompleksitas Komputasi (Waktu)

Pengukuran kinerja kualitatif algoritma biasanya dilakukan dengan menyatakan kinerja sebagai satu persamaan sederhana yang menunjukkan hubungan

> < >

antara ukuran masukan dan kinerja. Cara tradisional untuk menyatakan kinerja adalah dengan menggunakan Notasi Big-O, yaitu waktu komputasi algoritma berbanding terhadap suatu fungsi tertentu : f(*n*)=O(g(*n*)) jika dan hanya jika terdapat konstanta c >= 0 dan konstanta n_0 >= 0 sehingga |f(*n*)| <= |g(*n*)| untuk semua *n* >= n_0, dimana *n* merupakan karakteristik dari masukan yang diberikan pada algoritma, yang umumnya menunjukkan jumlah data yang ada. Notasi Big-O menghilangkan semua konstanta dan faktor kecuali yang dominan.

a. Pengisian nilai (*assignment*), perbandingan, operasi aritmetik, *read*, *write* membutuhkan waktu O(1).
b. Pengaksessan elemen larik atau memilih *field* tertentu dari sebuah *record* membutuhkan waktu O(1).
c. if C then S1 else S2 membutuhkan waktu TC + max(TS1, TS2) yang dalam hal ini TC, TS1 dan TS2 adalah kompleksitas waktu C, S1 dan S2.
d. Kalang for. Kompleksitas waktu kalang for adalah jumlah pengulangan dikali dengan kompleksitas waktu badan (*body*) kalang.
e. While C do S; dan repeat S until C; untuk kedua buah kalang, kompleksitas waktunya adalah jumlah pengulangan dikali dengan kompleksitas waktu badan C dan S. Masalah yang muncul adalah bila jumlah pengulangan tidak dapat ditentukan karena pengulangan dilakukan bergantung pada kondisi yang harus dipenuhi.
f. Prosedur dan fungsi. Waktu yang dibutuhkan untuk memindahkan kendali ke rutin yang dipanggil adalah O(1).

g. Untuk fungsi/prosedur rekursif, digunakan teknik perhitungan kompleksitas dengan *relasi rekurens*.

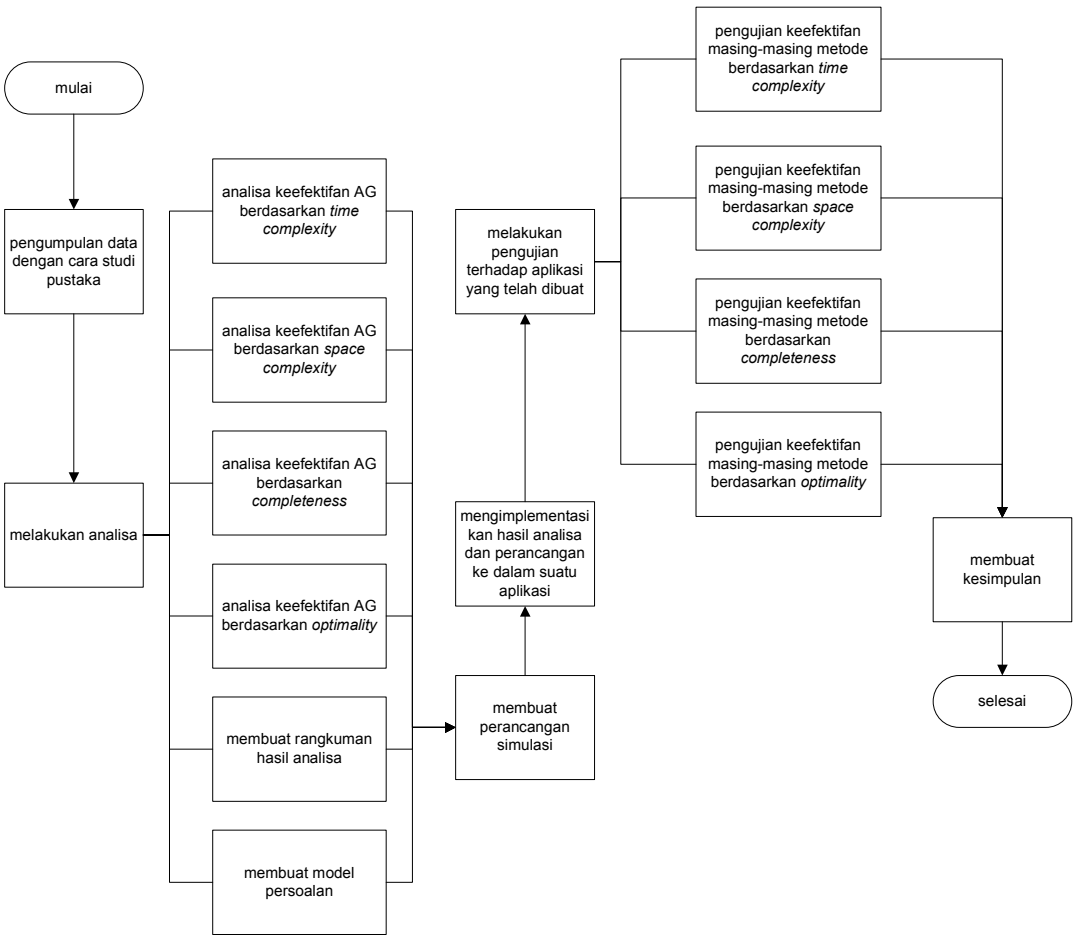
2.5.3 Kompleksitas Ruang

Analisa kebutuhan ruang suatu algoritma umumnya lebih mudah dibandingkan analisis kompleksitas komputasi, analisis kebutuhan ruang juga dilakukan dengan teknik yang sama dengan yang digunakan kompleksitas waktu. Analisis kebutuhan ruang biasanya hanya menghubung ruang untuk menyimpan data, tidak memasukkan ruang yang diperlukan untuk menyimpan algoritma. Kebutuhan ruang biasanya juga dinyatakan dengan notasi Big-O.

BAB III

METODOLOGI PENELITIAN

Metodologi atau cara penelitian yang digunakan dalam pembuatan tugas akhir ini adalah pengumpulan data, analisa, perancangan, implementasi dan pengujian.



Gambar 3.1 Flowchart Tahap-tahap Penelitian

3.1 Tahap Pengumpulan Data

Pengumpulkan data dilakukan dengan cara studi pustaka, yaitu membaca dan mencari informasi baik itu di literatur maupun di internet. Adapun guna pengumpulan data ini adalah untuk :

a. Mempelajari dan memahami kinerja dari algoritma genetika
b. Mempelajari dan memahami kinerja dari metode *roulette wheel selection*
c. Mempelajari dan memahami kinerja dari metode *rank selection*
d. Mempelajari dan memahami kinerja dari metode *tournament selection*
e. Mempelajari dan memahami permasalahan *travelling salesman problem* terutama *symmetric travelling salesman problem*
f. Mempelajari dan memahami penghitungan kompleksitas waktu dan kompleksitas ruang dengan menggunakan notasi Big-O

3.2 Tahap Analisa

Melakukan analisa keefektifan algoritma genetika dengan menggunakan tiga metode seleksi yaitu metode *roulette wheel selection*, *rank selection* dan *tournament selection* pada studi kasus *travelling salesman problem* (TSP) dengan melihat 4 kriteria, yaitu *time complexity, space complexity, completeness* dan *optimality*.

Langkah-langkah yang dilakukan pada tahap analisa ini adalah :

1. Melakukan analisa keefektifan algoritma genetika berdasarkan *time complexity* terhadap permasalah TSP

a. Menghitung *time complexity* proses inisialisasi
b. Menghitung *time complexity* proses evaluasi
c. Menghitung *time complexity* proses seleksi *roulette wheel selection*
d. Menghitung *time complexity* proses seleksi *rank selection*
e. Menghitung *time complexity* proses seleksi *tournament selection*
f. Menghitung *time complexity* proses c*rossover*
g. Menghitung *time complexity* proses mutasi
h. Menghitung *time complexity* algoritma genetika dengan menggunakan metode seleksi *roulette wheel selection* terhadap permasalahan TSP
i. Menghitung *time complexity* algoritma genetika dengan menggunakan metode seleksi *rank selection* terhadap permasalahan TSP
j. Menghitung *time complexity* algoritma genetika dengan menggunakan metode seleksi *tournament selection* terhadap permasalahan TSP

2. Melakukan analisa keefektifan algoritma genetika berdasarkan *space complexity* terhadap permasalahan TSP
   a. Menghitung *space complexity* algoritma genetika dengan menggunakan metode seleksi *roulette wheel selection* terhadap permasalahan TSP
   b. Menghitung *space complexity* algoritma genetika dengan menggunakan metode seleksi *rank selection* terhadap permasalahan TSP
   c. Menghitung *space complexity* algoritma genetika dengan menggunakan metode seleksi *tournament selection* terhadap permasalahan TSP

3. Melakukan analisa keefektifan algoritma genetika berdasarkan *completeness* terhadap permasalahan TSP
    a. Analisa algoritma genetika dengan menggunakan metode seleksi *roulette wheel selection* berdasarkan *completeness* terhadap permasalahan TSP
    b. Analisa algoritma genetika dengan menggunakan metode seleksi *rank selection* berdasarkan *completeness* terhadap permasalahan TSP
    c. Analisa algoritma genetika dengan menggunakan metode seleksi *tournament selection* berdasarkan *completeness* terhadap permasalahan TSP
4. Melakukan analisa keefektifan algoritma genetika berdasarkan *optimality* terhadap permasalahan TSP
    a. Analisa algoritma genetika dengan menggunakan metode seleksi *roulette wheel selection* berdasarkan *optimality* terhadap permasalahan TSP
    b. Analisa algoritma genetika dengan menggunakan metode seleksi *rank selection* berdasarkan *optimality* terhadap permasalahan TSP
    c. Analisa algoritma genetika dengan menggunakan metode seleksi *tournament selection* berdasarkan *optimality* terhadap permasalahan TSP
5. Membuat rangkuman analisis algoritma genetika dalam penyelesaian permasalahan TSP
    a. Membuat rangkuman analisis algoritma genetika dengan metode *roulette wheel selection* dalam penyelesaian permasalahan TSP

b. Membuat rangkuman analisis algoritma genetika dengan metode *rank selection* dalam penyelesaian permasalahan TSP
c. Membuat rangkuman analisis algoritma genetika dengan metode *tournament selection* dalam penyelesaian permasalahan TSP

6. Membuat model persoalan TSP menggunakan algoritma genetika

3.3 Tahap Perancangan

Membuat rancangan model simulasi persoalan TSP menggunakan algoritma genetika yang akan memperlihatkan kinerja ketiga metode seleksi pada algoritma genetika, yaitu metode *roulette wheel selection*, metode *rank selection* dan metode *tournament selection.*

3.4 Tahap Implementasi

Implementasi merupakan tahap dimana sistem siap dioperasikan pada keadaan yang sebenarnya, termasuk kegiatan penulisan kode program yang digunakan, sehingga akan diketahui apakah sistem yang dibuat benar-benar dapat menghasilkan solusi atau target yang diinginkan. Pada tahap implementasi akan dikemukakan tentang :

1. Alasan pemilihan perangkat lunak
2. Batasan implementasi
3. Lingkungan implementasi
4. Hasil implementasi

3.5 Tahap Pengujian

Pengujian simulasi adalah tahap dimana dilakukan perbandingan antara masing-masing metode seleksi yang ada pada algoritma genetika dan dapat dilihat metode mana yang paling efektif untuk menyelesaikan kasus TSP.

1. Pengujian terhadap parameter *time complexity*

   Membuat tabel dan grafik perbandingan berdasarkan *time complexity* antara algoritma genetika dengan *roulette wheel selection*, algoritma genetika dengan *rank selection*, dan algoritma genetika dengan *tournament selection* baik secara manual maupun secara sistem.

2. Pengujian terhadap parameter *space complexity*

   Membuat tabel dan grafik perbandingan berdasarkan *space complexity* antara algoritma genetika dengan *roulette wheel selection*, algoritma genetika dengan *rank selection*, dan algoritma genetika dengan *tournament selection* baik secara manual maupun secara sistem.

3. Pengujian terhadap parameter *completeness*
4. Pengujian terhadap parameter *optimality*
5. Kesimpulan pengujian

BAB IV

ANALISA DAN PERANCANGAN

Analisa perangkat lunak merupakan langkah pemahaman persoalan sehingga didapatkan kesimpulan dari langkah-langkah pemahaman persoalan tersebut. Sedangkan tahap perancangan merupakan tindakan kedua dari analisa sehingga bentuk rancangan yang dibuat sesuai dengan hasil apa yang diharapkan dari persoalan yang dibahas dan dapat dipahami oleh pengguna atau *user*.

Pada algoritma genetika ada 6 komponen yang penting, yaitu teknik penyandian, inisialisasi, fungsi evaluasi, seleksi, operator genetika dan penentuan parameter.

1. Teknik Penyandian

   Teknik penyandian untuk tugas akhir ini dilakukan secara *permutation encoding*. Teknik ini dipilih karena merepresentasikan kromosom berupa urutan angka atau huruf yang menggambarkan urutan suatu kejadian, sehingga sangat cocok untuk representasi kromosom pada TSP yaitu berupa urutan kota.

2. Inisialisasi

   Inisialisasi kromosom pada populasi awal dilakukan secara acak dengan tetap memperhatikan domain solusi dan kendala permasalahan yang ada. Inisialisasi

dilakukan dengan model representasi sekuensial dimana setiap kota di *list* pada urutan yang keberapa kota itu dikunjungi.

3. Fungsi evaluasi

   Fungsi evaluasi adalah melakukan penilaian dari setiap kromosom yang ada dalam populasi. Pada kasus TSP, fungsi evaluasi nya adalah total *cost* dari perjalanan.

4. Seleksi

   Seleksi yang digunakan ada 3, yaitu *roulette wheel selection, rank selection* dan *tournament selection. Roulette wheel selection* dipilih karena metode seleksi ini sering digunakan dalam penyelesaian masalah menggunakan algoritma genetika. *Rank selection* dipilih karena metode ini mengantisipasi tidak terseleksinya kromosom dengan *fitness* yang baik karena nilai probabilitas kromosom tersebut rendah. *Tournament selection* dipilih karena pemilihan induk untuk proses *crossover* dan mutasi hanya berdasarkan nilai *fitness* terbaik dalam suatu grup pada populasi.

5. Operator genetika *crossover*

   Crossover dilakukan dengan *single point crossover*. Metode ini dipilih karena pemilihan titik perpotongan dilakukan sekali terhadap orang tua pertama, gen selanjutnya yang dipilih dari orang tua kedua adalah yang belum pernah terkunjungi. Kromosom yang dihasilkan cenderung menuruni sifat dari orang tua nya.

6. Operator genetika mutasi

   Mutasi dilakukan dengan *random only improving*. Metode ini dipilih karena diharapkan kromosom anak tidak kehilangan kemiripan dengan ornag tuanya.

7. Penentuan parameter

   Parameter yang digunakan *popsize* 30, 50 dan 80. Peluang *crossover* 45% dan peluang mutasi 10%.

Analisa dilakukan untuk melihat 3 metode seleksi yang lebih baik digunakan untuk kasus TSP, yaitu metode *roulette wheel selection, rank selection* dan *tournament selection.*

4.1 Analisa Keefektifan Algoritma Genetika untuk Kasus TSP

Parameter-parameter yang digunakan untuk melihat keefektifan algoritma genetika pada kasus TSP adalah sebagai berikut :

a. Efisiensi *time complexity* algoritma genetika dalam penyalesaian kasus TSP. *Time complexity* dihitung dengan menggunakan notasi Big-O.
b. *Space complexity*, jumlah memori yang dibutuhkan untuk penyelesaian kasus TSP. *Space complexity* dihitung dengan menggunakan notasi Big-O.
c. *Completeness*, apakah algoritma genetika dapat menemukan solusi untuk kasus TSP jika terdapat beberapa sulusi.

d. *Optimality*, apakah solusi pada kasus TSP merupakan sulusi yang terbaik jika ditemukan beberapa solusi.

4.1.1 Analisa Keefektifan Algoritma Genetika berdasarkan *Time Complexity*

Untuk menentukan apakah algoritma genetika pada kasus TSP lebih efisien, digunakan perhitungan kompleksitas waktu dengan notasi Big-O.

4.1.1.1 Kompleksitas Waktu Proses Inisialisasi

Merujuk pada Algoritma 2.2 dilakukan analisa kompleksitas waktu Big-O sebagai berikut :

Observasi : Algoritma inisialisasi bekerja untuk mendapatkan beberapa kromosom atau solusi yang mungkin dari sebuah persoalan.

Langkah 1 : Perintah *loop for* (2-4) membutuhkan waktu eksekusi sebanyak populasi. Misalkan populasi dilambangkan dengan n maka waktu yang dibutuhkan adalah O(n).

Langkah 2 : Waktu yang dibutuhkan untuk operasi yang berada dalam kalang adalah sebanyak jumlah kota. Misalkan jumlah kota dilambangkan dengan m maka waktu yang dibutuhkan adalah O(m).

Langkah 3 : Berdasarkan Persamaan 2.10 maka total waktu yang diperlukan untuk semua operasi inisialisasi adalah :

$$T(n) = O(n) \cdot O(m)$$
$$= O(nm)$$

4.1.1.2 Kompleksitas Waktu Proses Evaluasi

Merujuk pada Algoritma 2.3 dilakukan analisa kompleksitas waktu Big-O sebagai berikut :

Observasi : Algoritma evaluasi bekerja untuk mendapatkan nilai fungsi objektif dari tip-tiap kromosom hasil inisialisasi.

Langkah 1 : Perintah *loop for* (2-4) membutuhkan waktu eksekusi sebanyak populasi. Misalkan populasi dilambangkan dengan n maka waktu yang dibutuhkan adalah O(n).

Langkah 2 : Waktu yang dibutuhkan untuk operasi yang berada dalam kalang adalah sebanyak jumlah kota. Misalkan jumlah kota dilambangkan dengan m maka waktu yang dibutuhkan adalah O(m).

Langkah 3 : Berdasarkan Persamaan 2.10 maka total waktu yang diperlukan untuk semua operasi evaluasi adalah :

$$T(n) = O(n) \cdot O(m)$$
$$= O(nm)$$

4.1.1.3 Kompleksitas Waktu Proses *Roulette Wheel Selection*

Merujuk pada Persamaan 2.3 maka kompleksitas waktu proses *roulette wheel selection* yaitu :

$$T(n) = O(n^2)$$

4.1.1.4 Kompleksitas Waktu Proses *Rank Selection*

Merujuk pada Persamaan 2.5 maka kompleksitas waktu proses *rank selection* yaitu :

$$T(n) = O(n \log n)$$

4.1.1.5 Kompleksitas Waktu Proses *Tournament Selection*

Merujuk pada Persamaan 2.6 maka kompleksitas waktu proses *tournament selection* yaitu :

$$T(n) = O(n)$$

4.1.1.6 Kompleksitas Waktu Proses *Crossover*

Merujuk pada Algoritma 2.7 dilakukan analisa kompleksitas waktu Big-O sebagai berikut :

Observasi : Algoritma *crossover* bekerja dari sejumlah kromosom hasil seleksi untuk mendapatkan anak (*offspring*).

Langkah 1 : Pemindahan kromosom induk pertama ke kromosom anak (11 dan 17) membutuhkan waktu O(m). Pemindahan kromosom induk kedua ke kromosom anak (12 dan 18) membutuhkan waktu $O(m^2)$.

Langkah 2 : Kromosom induk dipilih sebanyak *crossover_loop*. Pemilihan kromosom induk membutuhkan waktu eksekusi sebanyak O(n).

Langkah 3 : Berdasarkan Persamaan 2.9 dan persamaan 2.10 maka total waktu yang diperlukan untuk semua operasi *crossover* adalah :

$T(n) = O(n) . (O(m) + O(m^2))$

$= O(n) . O(max(m,m^2))$

$= O(nm^2)$

4.1.1.7 Kompleksitas Waktu Proses Mutasi

Merujuk pada Algoritma 2.8 dilakukan analisa kompleksitas waktu Big-O sebagai berikut :

Observasi : Algoritma mutasi bekerja dari sejumlah kromosom hasil seleksi untuk mendapatkan anak (*offspring*).

Langkah 1 : Pemindahan kromosom induk ke kromosom anak (6-9) membutuhkan waktu O(m).

Langkah 2 : Kromosom induk dipilih sebanyak mutasi_*loop*. Pemilihan kromosom induk membutuhkan waktu eksekusi sebanyak O(n).

Langkah 3 : Berdasarkan Persamaan 2.10 maka total waktu eksekusi yang diperlukan untuk semua operasi mutasi adalah :

$$T(n) = O(n) \cdot O(m)$$
$$= O(nm)$$

4.1.1.8 Kompleksitas Waktu Algoritma Genetika

Setelah menghitung kompleksitas waktu masing-masing proses yang ada pada algoritma genetika, maka akan dihitung kompleksitas waktu keseluruhan yang diperlukan algoritma genetika.

1. Merujuk pada Algoritma 2.1, kompleksitas waktu algoritma genetika dengan metode *roulette wheel selection* dalam notasi Big-O adalah sebagai berikut :
   a. Proses inisialisasi (4-6) membutuhkan waktu eksekusi O(nm)
   b. Proses Evaluasi (10-12) membutuhkan waktu eksekusi sebanyak O(nm)
   c. Proses seleksi dengan *roulette wheel selection* (14-16) membutuhkan waktu eksekusi $O(n^2)$
   d. Proses *crossover* (20-23) membutuhkan waktu eksekusi $O(nm^2)$
   e. Proses mutasi (26-28) membutuhkan waktu eksekusi O(nm)
   f. Untuk kalang while (7-31), waktu eksekusi yang diperlukan adalah jumlah pengulangan dikalikan dengan total kompleksitas waktu proses b, c, d dan e. jumlah pengulangan yang dilakukan adalah jika dipenuhinya kriteria terminasi. Berdasarkan Persamaan 2.9 dan Persamaan 2.10, misalkan kriteria terminasi dilambangkan dengan d maka waktu eksekusi yang diperlukan adalah :

$T(n) = O(d) . ((O(nm) + O(n^2)) + (O(nm^2) + O(nm))$

$= O(d) . (O(max(nm,n^2) + O(max(,nm^2,nm)))$

$= O(d) . (O(n^2) + O(nm^2))$

$= O(d) . O(max(n^2,nm^2))$

$= O(d) . O(nm^2)$

$= O(dnm^2)$

g. Total waktu eksekusi yang diperlukan algoritma genetika dengan *roulette wheel selection* dalam penyelesaian kasus TSP adalah total kompleksitas waktu proses a dan f, yaitu :

$T(n) = O(nm) + O(dnm^2)$

$= O(max(nm,dnm^2))$

$= O(dnm^2)$

dimana :

d = kriteria terminasi telah tercapai,

n = Ukuran populasi, dan

m = Jumlah kota

2. Merujuk pada Algoritma 2.1, kompleksitas waktu algoritma genetika dengan metode *rank selection* dalam notasi Big-O adalah sebagai berikut :

a. Proses inisialisasi (4-6) membutuhkan waktu eksekusi O(nm)

b. Proses Evaluasi (10-12) membutuhkan waktu eksekusi sebanyak O(nm)

c. Proses seleksi dengan *rank selection* (14-16) membutuhkan waktu eksekusi O(n log n)

d. Proses *crossover* (21-24) membutuhkan waktu eksekusi $O(nm^2)$

e. Proses mutasi (26-28) membutuhkan waktu eksekusi O(nm)

f. Untuk kalang while (7-31), waktu eksekusi yang diperlukan adalah jumlah pengulangan dikalikan dengan total kompleksitas waktu proses b, c, d dan e. jumlah pengulangan yang dilakukan adalah jika dipenuhinya kriteria terminasi. Berdasarkan Persamaan 2.9 dan Persamaan 2.10, misalkan kriteria terminasi dilambangkan dengan d maka waktu eksekusi yang diperlukan adalah :

$T(n) = O(d) . ((O(nm) + O(n \log n)) + (O(nm^2) + O(nm))$

$= O(d) . (O(\max(nm,n \log n) + O(\max(,nm^2,nm)))$

$= O(d) . (O(nm) + O(nm^2))$

$= O(d) . O(\max(nm,nm^2))$

$= O(d) . O(nm^2)$

$= O(dnm^2)$

g. Total waktu eksekusi yang diperlukan algoritma genetika dengan *rank selection* dalam penyelesaian kasus TSP adalah total kompleksitas waktu proses a dan f, yaitu :

$T(n) = O(nm) + O(dnm^2)$

$= O(\max(nm,dnm^2))$

$= O(dnm^2)$

dimana :

d = kriteria terminasi telah dicapai,

n = Ukuran populasi, dan

m = Jumlah kota

3. Merujuk pada Algoritma 2.1, kompleksitas waktu algoritma genetika dengan metode *tournament selection* dalam notasi Big-O adalah sebagai berikut :
   a. Proses inisialisasi (4-6) membutuhkan waktu eksekusi O(nm)
   b. Proses Evaluasi (10-12) membutuhkan waktu eksekusi sebanyak O(nm)
   c. Proses seleksi dengan *tournament selection* (14-16) membutuhkan waktu eksekusi O(n)
   d. Proses *crossover* (21-24) membutuhkan waktu eksekusi $O(nm^2)$
   e. Proses mutasi (26-28) membutuhkan waktu eksekusi O(nm)
   f. Untuk kalang while (7-31), waktu eksekusi yang diperlukan adalah jumlah pengulangan dikalikan dengan total kompleksitas waktu proses b, c, d dan e. jumlah pengulangan yang dilakukan adalah jika dipenuhinya kriteria terminasi. Berdasarkan Persamaan 2.9 dan Persamaan 2.10, misalkan kriteria terminasi dilambangkan dengan d maka waktu eksekusi yang diperlukan adalah :

$T(n) = O(d) . ((O(nm) + O(n^2)) + (O(nm^2) + O(nm))$

$= O(d) . (O(max(nm,n) + O(max(,nm^2,nm)))$

$= O(d) . (O(nm) + O(nm^2))$

$= O(d) . O(max(nm,nm^2))$

$= O(d) . O(nm^2)$

$= O(dnm^2)$

g. Total waktu eksekusi yang diperlukan algoritma genetika dengan *tournament selection* dalam penyelesaian kasus TSP adalah total kompleksitas waktu proses a dan f, yaitu :

$T(n) = O(nm) + O(dnm^2)$

$= O(max(nm,dnm^2))$

$= O(dnm^2)$

dimana :

d = kriteria terminasi telah dicapai,

n = Ukuran populasi, dan

m = Jumlah kota

4.1.2 Analisa Keefektifan Algoritma Genetika berdasarkan *Space Complexity*

Keefektifan algoritma genetika dengan parameter *space complexity* ditujukan untuk mengetahui berapa banyak memori yang digunakan dalam penyelesaian *travelling salesman problem*. Dalam pencarian suatu solusi yang dilakukan dengan computer memerlukan jumlah *space complexity* yang didasarkan kepada kesulitan

dalam proses pencarian. Kesulitan tersebut akan memaksa jumlah *space complexity* yang dihasilkan akan semakin besar.

Pada algoritma genetika, *space complexity* berbanding lurus dengan *time complexity*. Secara kompleksitas ruang (Big-O), algoritma genetika dengan metode *roulette wheel selection*, algoritma genetika dengan metode *rank selection* atau algoritma genetika dengan metode *tournament selection* mempunyai rumus yang sama yaitu :

$$T(n) = O(dnm^2)$$

dimana :

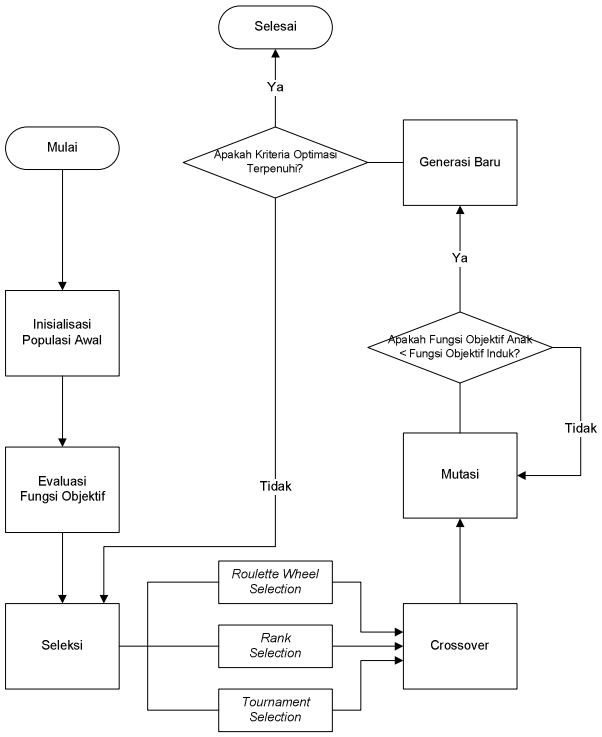
d = kriteria terminasi telah dicapai,

n = Ukuran populasi, dan

m = Jumlah kota

4.1.3 Analisa Keefektifan Algoritma Genetika berdasarkan *Completeness*

Algoritma genetika adalah algoritma pencarian data dan optimasi yang didasari pada proses mekanika alamiah, dimana sifat-sifat suatu spesies sangat bergantung pada gen-gen dan susunannya. Algoritma genetika dapat menemukan solusi dari permasalahan *travelling salesman problem*.



Gambar 4.1 Flowchart Proses Algoritma Genetika

4.1.3.1 *Completeness* Algoritma Genetika menggunakan Metode *Roulette Wheel Selection*

Adapun langkah-langkah yang ditempuh algoritma genetika menggunakan metode *roulette wheel selection* untuk mendapatkan solusi adalah sebagai berikut :

1. Inisialisasi populasi awal dengan kromosom secara acak. Kromosom yang dimaksud dalam kasus TSP adalah sirkuit Hamilton yang dapat dibentuk dari graf lengkap berbobot dan berarah.
2. Evaluasi setiap kromosom dalam populasi dengan cara menghitung panjang rute dari sirkuit Hamilton yang telah dibentuk sebelumnya.

3. Lakukan seleksi dengan metode *roulette wheel selection*. Langkah-langkah yang dilakukan pada metode ini adalah sebagai berikut :
    a. Hitung nilai fitness tiap-tiap kromosom dengan menggunakan rumus pada Persamaan 2.1 dan totalkan seluruh nilai fitness.
    b. Sampai dengan ukuran populasi, lakukan :
        i. Hitung probabilitas kromosom dengan menggunakan rumus pada Persamaan 2.2
        ii. Hitung nilai komulatif dari probabilitas
        iii. Acak R dengan range 0-1
        iv. Jika nilai komulatif lebih besar dan sama dengan R maka kromosom diambil sebagai populasi yang bertahan
4. Bangkitkan kromosom baru dengan melakukan proses *one point crossover* dan proses *random only improving*.
5. Hapus kromosom pada generasi sebelumnya untuk memberi ruang pada kromosom baru.
6. Evaluasi kromosom baru, dan masukkan ke dalam populasi.
7. Berhenti sampai terpenuhi kriteria, jika tidak kembali ke langkah 3.

4.1.3.2 *Completeness* Algoritma Genetika menggunakan Metode *Rank Selection*

Adapun langkah-langkah yang ditempuh algoritma genetika menggunakan metode *rank selection* untuk mendapatkan solusi adalah sebagai berikut :

1. Inisialisasi populasi awal dengan kromosom secara acak. Kromosom yang dimaksud dalam kasus TSP adalah sirkuit Hamilton yang dapat dibentuk dari graf lengkap berbobot dan berarah.
2. Evaluasi setiap kromosom dalam populasi dengan cara menghitung panjang rute dari sirkuit Hamilton yang telah dibentuk sebelumnya.
3. Lakukan seleksi dengan metode *rank selection*. Langkah-langkah yang dilakukan pada metode ini adalah sebagai berikut :
    a. Berdasarkan nilai objektifnya, maka urutkan nilai objektif yang terkecil ke yang terbesar (ubah susunan kromosom)
    b. Hitung nilai fitness tiap-tiap kromosom dengan menggunakan rumus pada Persamaan 2.4
    c. Sampai dengan ukuran populasi, lakukan :
        i. Hitung probabilitas kromosom dengan menggunakan rumus pada Persamaan 2.2
        ii. Hitung nilai komulatif dari probabilitas
        iii. Acak R dengan range 0-1
        iv. Jika nilai komulatif lebih besar dan sama dengan R maka kromosom diambil sebagai populasi yang bertahan

4. Bangkitkan kromosom baru dengan melakukan proses *one point crossover* dan proses *random only improving*.
5. Hapus kromosom pada generasi sebelumnya untuk memberi ruang pada kromosom baru.
6. Evaluasi kromosom baru, dan masukkan ke dalam populasi.
7. Berhenti sampai terpenuhi kriteria, jika tidak kembali ke langkah 3.

4.1.3.3 *Completeness* Algoritma Genetika menggunakan Metode *Tournament Selection*

Adapun langkah-langkah yang ditempuh algoritma genetika menggunakan metode *tournament selection* untuk mendapatkan solusi adalah sebagai berikut :

1. Inisialisasi populasi awal dengan kromosom secara acak. Kromosom yang dimaksud dalam kasus TSP adalah sirkuit Hamilton yang dapat dibentuk dari graf lengkap berbobot dan berarah.
2. Evaluasi setiap kromosom dalam populasi dengan cara menghitung panjang rute dari sirkuit Hamilton yang telah dibentuk sebelumnya.
3. Lakukan seleksi dengan metode *tournament selection*. Langkah-langkah yang dilakukan pada metode ini adalah sebagai berikut :
    a. Ambil k (ukuran *tournament*) dari populasi
    b. Cari kromosom dengan nilai objektif yang paling kecil
    c. Ambil kromosom sebagai kromosom yang bertahan
    d. Jika ada kromosom dengan permutasi yang sama, buang salah satunya

e. Lakukan proses sampai didapat dua kromosom induk

4. Bangkitkan kromosom baru dengan melakukan proses *one point crossover* dan proses *random only improving*.
5. Hapus kromosom pada generasi sebelumnya untuk memberi ruang pada kromosom baru.
6. Evaluasi kromosom baru, dan masukkan ke dalam populasi.
7. Berhenti sampai terpenuhi kriteria, jika tidak kembali ke langkah 3.

4.1.4 Analisa Keefektifan Algoritma Genetika berdasarkan *Optimality*

4.1.4.1 *Optimality* Algoritma Genetika menggunakan Metode *Roulette Wheel Selection*

Berdasarkan c*ompleteness* algoritma genetika menggunakan metode *roulette wheel selection,* solusi yang ditemukan merupakan solusi yang optimal yang dilihat dari terminasi yang telah dicapai selama proses pencarian solusi *travelling salesman problem*.

4.1.4.2 *Optimality* Algoritma Genetika menggunakan Metode *Rank Selection*

Berdasarkan c*ompleteness* algoritma genetika menggunakan metode *rank selection*, solusi yang ditemukan merupakan solusi yang optimal yang dilihat dari terminasi yang telah dicapai selama proses pencarian solusi *travelling salesman problem*.

4.1.4.3 *Optimality* Algoritma Genetika menggunakan Metode *Tournament Selection*

Berdasarkan c*ompleteness* algoritma genetika menggunakan metode *tournament selection*, solusi yang ditemukan merupakan solusi yang optimal yang dilihat dari terminasi yang telah dicapai selama proses pencarian solusi *travelling salesman problem*.

4.2 Rangkuman Analisa Algoritma Genetika dalam Parameter COST

Dari hasil analisa yang telah dibuat, maka dapat ditulis rangkuman analisa Algoritma Genetika menggunakan metode seleksi *roulette wheel selection*, *rank selection* dan *tournament selection*.

Tabel 4.1 Rangkuman Analisa Algoritma Genetika Menggunakan Metode *Roulette Wheel Selection*

Parameter	Kesimpulan
Completeness	Melakukan pencarian solusi pada kasus TSP dengan cara mengambil beberapa kromosom (sirkuit Hamilton) dan kemudian melakukan seleksi terhadap kromosom tersebut berdasarkan nilai komulatif masing-masing kromosom dalam populasi untuk mendapatkan kromosom yang bertahan. Solusi dapat ditemukan.
Optimality	Hasil dari *optimality* adalah jumlah generasi yang dicapai atau terminasi telah dicapai pada generasi yang keberapa saat nilai optimal ditemukan. Solusi yang ditemukan merupakan solusi yang optimal.

Time complexity	Dari hasil *completeness* maka dapat ditentukan berapa lama waktu yang dibutuhkan selama proses pencarian selesai.
Space complexity	*Space complexity* yang digunakan berbanding lurus dengan *time complexity*

Tabel 4.2 Rangkuman Analisa Algoritma Genetika Menggunakan Metode *Rank Selection*

Parameter	Kesimpulan
Completeness	Melakukan pencarian solusi pada kasus TSP dengan cara mengambil beberapa kromosom (sirkuit Hamilton) dan kemudian melakukan seleksi terhadap kromosom tersebut berdasarkan ranking tiap-tiap kromosom dalam populasi untuk mendapatkan kromosom yang bertahan. . Solusi dapat ditemukan.
Optimality	Hasil dari *optimality* adalah jumlah generasi yang dicapai atau terminasi telah dicapai pada generasi yang keberapa saat nilai optimal ditemukan. Solusi yang ditemukan merupakan solusi yang optimal.
Time complexity	Dari hasil *completeness* maka dapat ditentukan berapa lama waktu yang dibutuhkan selama proses pencarian selesai.
Space complexity	*Space complexity* yang digunakan berbanding lurus dengan *time complexity*

Tabel 4.3 Rangkuman Analisa Algoritma Genetika Menggunakan Metode *Tournament Selection*

Parameter	Kesimpulan
Completeness	Melakukan pencarian solusi pada kasus TSP dengan cara mengambil beberapa kromosom (sirkuit Hamilton) dan kemudian melakukan pencarian dua kromosom induk berdasarkan nilai objektif yang paling kecil dalam populasi untuk selanjutnya dilakukan proses *crossover* dan mutasi. Solusi dapat ditemukan.
Optimality	Hasil dari *optimality* adalah jumlah generasi yang dicapai atau terminasi telah dicapai pada generasi yang keberapa saat nilai optimal ditemukan. Solusi yang ditemukan merupakan solusi yang optimal.
Time complexity	Dari hasil *completeness* maka dapat ditentukan berapa lama waktu yang dibutuhkan selama proses pencarian selesai.
Space complexity	*Space complexity* yang digunakan berbanding lurus dengan *time complexity*

4.3 Model Persoalan

Berdasarkan analisa yang telah dilakukan, maka akan dijelaskan lebih lanjut dalam contoh kasus *travelling salesman problem*.

4.3.1 Analisa Algoritma Genetika terhadap *Completeness*

∞	4	11	8	7	8	7
4	∞	5	2	6	3	9
11	5	∞	15	9	12	8
8	2	15	∞	10	10	15
7	6	9	10	∞	11	12
8	3	12	10	11	∞	9
7	9	8	15	12	9	∞

Gambar 4.2 Matrik Berukuran 7 x 7

Persoalan TSP ini digambarkan dalam bentuk matrik dengan baris (i) dan kolom (j) merepresentasikan sebuah simpul. Untuk baris dan kolom yang sama diberi nilai ∞. Pada *symmetric travelling salesman problem*, nilai (i,j) sama dengan nilai (j,i).

Ukuran yang digunakan dalam analisa ini adalah sebagai berikut :

a. Jumlah kota = 7 kota
b. Ukuran populasi awal = 15 populasi
c. Probabilitas *crossover* = 25%
d. Probabilitas mutasi = 10%

4.3.1.1 Inisialisasi Populasi Awal

Populasi awal dipilih secara acak, misalkan diperoleh hasil seperti pada tabel berikut :

Tabel 4.4 Populasi Awal

Kromosom ke	Bentuk Permutasi
1	7 2 5 4 3 1 6 7
2	5 3 4 2 1 6 7 5
3	5 4 2 1 3 7 6 5
4	6 3 1 2 5 7 4 6
5	6 7 1 4 3 2 5 6
6	7 3 2 1 5 6 4 7
7	2 4 5 1 6 7 3 2
8	3 6 5 7 1 4 2 3
9	4 5 2 6 1 7 3 4
10	3 5 1 6 4 2 7 3
11	1 5 3 7 6 2 4 1
12	7 4 2 5 1 3 6 7
13	4 2 5 7 6 3 1 4
14	3 2 4 1 5 7 6 3
15	1 3 6 5 7 2 4 1

4.3.1.2 Evaluasi Fungsi

Hitung panjang rute perjalanan dari kromosom untuk mendapatkan fungsi objektif kromosom.

Tabel 4.5 Evaluasi Fungsi

Kromosom ke	Bentuk Permutasi	Fungsi Objektif
1	7 2 5 4 3 1 6 7	71
2	5 3 4 2 1 6 7 5	58
3	5 4 2 1 3 7 6 5	52
4	6 3 1 2 5 7 4 6	72
5	6 7 1 4 3 2 5 6	67
6	7 3 2 1 5 6 4 7	66

7	2 4 5 1 6 7 3 2	56
8	3 6 5 7 1 4 2 3	49
9	4 5 2 6 1 7 3 4	59
10	3 5 1 6 4 2 7 3	48
11	1 5 3 7 6 2 4 1	52
12	7 4 2 5 1 3 6 7	55
13	4 2 5 7 6 3 1 4	60
14	3 2 4 1 5 7 6 3	59
15	1 3 6 5 7 2 4 1	67

4.3.1.3 Seleksi

Proses seleksi dilakukan dengan cara membuat kromosom yang mempunyai *fitness* kecil mempunyai kemungkinan terpilih lebih besar atau mempunyai nilai probabilitas yang tinggi.

Metode seleksi yang digunakan ada tiga, yaitu *roulette wheel selection*, *rank selection* dan *tournament selection*. Proses *crossover* dan mutasi dilakukan berdasarkan metode seleksi yang digunakan.

a. *Roulette wheel selection*

Sebelum dilakukan proses seleksi dengan metode *roulette-wheel*, terlebih dahulu dilakukan perhitungan nilai *fitness*, probabilitas dan komulatif dari kromosom.

i. Nilai Fitness

Fitness [i] = 1/Fungsi Objektif[i]

Fitness [1] = 1/71 = 0.0140

Fitness [2] = 1/58 = 0.0172

Fitness [3] = 1/52 = 0.0192

Fitness [4] = 1/72 = 0.0138

Fitness [5] = 1/67 = 0.0149

Fitness [6] = 1/66 = 0.0151

Fitness [7] = 1/56 = 0.0178

Fitness [8] = 1/49 = 0.0204

Fitness [9] = 1/59 = 0.0169

Fitness [10] = 1/48 = 0.0208

Fitness [11] = 1/52 = 0.0192

Fitness [12] = 1/55 = 0.0181

Fitness [13] = 1/60 = 0.0167

Fitness [14] = 1/59 = 0.0169

Fitness [15] = 1/57 = 0.0175

Total Fitness = 0.0140 + 0.0172 + 0.0192 + 0.0138 + 0.0149 + 0.0151 + 0.0178 + 0.0204 + 0.0169 + 0.0208 + 0.0192 + 0.0181 + 0.0167 + 0.0169 + 0.0175

= 0.2585

ii. Probabilitas

P[i] = Fitness[i]/Total Fitness

P[1] = 0.0140/0.2585 = 0.0542

P[2] = 0.0172/0.2585 = 0.0665

P[3] = 0.0192/0.2585 = 0.0743

P[4] = 0.0138/0.2585 = 0.0534

P[5] = 0.0149/0.2585 = 0.0576

P[6] = 0.0151/0.2585 = 0.0584

P[7] = 0.0178/0.2585 = 0.0689

P[8] = 0.0204/0.2585 = 0.0789

P[9] = 0.0169/0.2585 = 0.0654

P[10] = 0.0208/0.2585 = 0.0805

P[11] = 0.0192/0.2585 = 0.0743

P[12] = 0.0181/0.2585 = 0.0700

P[13] = 0.0167/0.2585 = 0.0646

P[14] = 0.0169/0.2585 = 0.0654

P[15] = 0.0175/0.2585 = 0.0677

iii. Komulatif

Komulatif[i] = Komulatif[i] + P[i]

Komulatif[1] = 0 + 0.0541 = 0.0541

Komulatif[2] = 0.0541 + 0.0665 = 0.1206

Komulatif[3] = 0.1206 + 0.0743 = 0.1949

Komulatif[4] = 0.1949 + 0.0534 = 0.2483

Komulatif[5] = 0.2483 + 0.0576 = 0.3059

Komulatif[6] = 0.3059 + 0.0584 = 0.3643

Komulatif[7] = 0.3643 + 0.0689 = 0.4332

Komulatif[8] = 0.4332 + 0.0789 = 0.5121

Komulatif[9] = 0.5121 + 0.0654 = 0.5775

Komulatif[10] = 0.5775 + 0.0805 = 0.6580

Komulatif[11] = 0.6580 + 0.0743 = 0.7323

Komulatif[12] = 0.7323 + 0.0700 = 0.8023

Komulatif[13] = 0.8023 + 0.0646 = 0.8669

Komulatif[14] = 0.8669 + 0.0654 = 0.9323

Komulatif[15] = 0.9323 + 0.0677 = 1

Setelah mendapatkan komulatif dari tiap-tiap kromosom maka setiap kromosom akan dibandingkan dengan nilai yang diacak R sebanyak populasi dari *range* 0-1. Jika Komulatif[i] > R[i] maka kromosom[i] akan diambil sebagai kromosom yang bertahan.

Misalkan didapat kromosom yang bertahan adalah kromosom[2], kromosom[3], kromosom[5], kromosom[6], kromosom[8], kromosom[10], kromosom[13], kromosom[14] dan kromosom[15].

Tabel 4.6 Kromosom Hasil Seleksi *Roulette Wheel Selection*

Kromosom ke	Bentuk Permutasi	Fungsi Objektif
1	5 3 4 2 1 6 7 5	58
2	5 4 2 1 3 7 6 5	52
3	6 7 1 4 3 2 5 6	67
4	7 3 2 1 5 6 4 7	66
5	3 6 5 7 1 4 2 3	49
6	3 5 1 6 4 2 7 3	48
7	4 2 5 7 6 3 1 4	60

8	3 2 4 1 5 7 6 3	59
9	1 3 6 5 7 2 4 1	57

b. *Rank selection*

Pada seleksi dengan metode ini pertama-tama dilakukan pengurutan terhadap nilai objektif kromosom. Nilai objektif yang terkecil menempati urutan pertama sedangkan nilai objektif yang terbesar menempati urutan ke n. Untuk nilai *fitness* kromosom, kromosom ranking pertama mempunyai nilai *fitness* n, kromosom ranking kedua menpunyai nilai *fitness* n-1, sampai dengan kromosom ranking terakhir mempunyai nilai *fitness* 1.

Tabel 4.7 Kromosom Berdasarkan Rank

Kromosom ke	**Bentuk Permutasi**	**Fungsi Objektif**	**Nilai *Fitness***
1	3 5 1 6 4 2 7 3	48	15
2	3 6 5 7 1 4 2 3	49	14
3	5 4 2 1 3 7 6 5	52	13
4	1 5 3 7 6 2 4 1	52	12
5	7 4 2 5 1 3 6 7	55	11
6	2 4 5 1 6 7 3 2	56	10
7	1 3 6 5 7 2 4 1	57	9
8	5 3 4 2 1 6 7 5	58	8
9	4 5 2 6 1 7 3 4	59	7
10	3 2 4 1 5 7 6 3	59	6
11	4 2 5 7 6 3 1 4	60	5
12	7 3 2 1 5 6 4 7	66	4
13	6 7 1 4 3 2 5 6	67	3
14	7 2 5 4 3 1 6 7	71	2
15	6 3 1 2 5 7 4 6	72	1

Sebelum dilakukan proses yang sama dengan metode *roulette wheel selection*, terlebih dahulu dilakukan perhitungan probabilitas dan komulatif dari kromosom.

i. Probabilitas

P[i] = Fitness[i]/Total Fitness

P[1] = 15/120 = 0.125

P[2] = 14/120 = 0.117

P[3] = 13/120 = 0.108

P[4] = 12/120 = 0.100

P[5] = 11/120 = 0.092

P[6] = 10/120 = 0.083

P[7] = 9/120 = 0.075

P[8] = 8/120 = 0.067

P[9] = 7/120 = 0.058

P[10] = 6/120 = 0.050

P[11] = 5/120 = 0.042

P[12] = 4/120 = 0.033

P[13] = 3/120 = 0.025

P[14] = 2/120 = 0.017

P[15] = 1/120 = 0.008

ii. Komulatif

Komulatif[i] = Komulatif[i] + P[i]

Komulatif[1] = 0 + 0.125 = 0.125

Komulatif[2] = 0.125 + 0.117 = 0.242

Komulatif[3] = 0.242 + 0.108 = 0.350

Komulatif[4] = 0.350 + 0.100 = 0.450

Komulatif[5] = 0.450 + 0.092 = 0.542

Komulatif[6] = 0.542 + 0.083 = 0.625

Komulatif[7] = 0.625 + 0.075 = 0.700

Komulatif[8] = 0.700 + 0.067 = 0.767

Komulatif[9] = 0.767 + 0.058 = 0.825

Komulatif[10] = 0.825 + 0.050 = 0.875

Komulatif[11] = 0.875 + 0.042 = 0.917

Komulatif[12] = 0.917 + 0.033 = 0.950

Komulatif[13] = 0.950 + 0.025 = 0.975

Komulatif[14] = 0.975 + 0.017 = 0.992

Komulatif[15] = 0.992 + 0.008 = 1

Setelah mendapatkan komulatif dari tiap-tiap kromosom maka setiap kromosom akan dibandingkan dengan nilai yang diacak sebanyak populasi dari *range* 0-1. Jika Komulatif[i] > R[i] maka kromosom[i] akan diambil sebagai kromosom yang bertahan.

Misalkan didapat kromosom yang bertahan adalah kromosom[1], kromosom[2], kromosom[4], kromosom[5], kromosom[8], kromosom[9], kromosom[11], kromosom[13] dan kromosom[15]

Tabel 4.8 Kromosom Hasil Seleksi *Rank Selection*

Kromosom ke	Bentuk Permutasi	Fungsi Objektif
1	3 5 1 6 4 2 7 3	48
2	3 6 5 7 1 4 2 3	49
3	1 5 3 7 6 2 4 1	52
4	7 4 2 5 1 3 6 7	55
5	5 3 4 2 1 6 7 5	58
6	4 5 2 6 1 7 3 4	59
7	4 2 5 7 6 3 1 4	60
8	6 7 1 4 3 2 5 6	67
9	6 3 1 2 5 7 4 6	72

c. *Tournament selection*

Pada seleksi dengan metode ini dilakukan pengambilan beberapa kromosom dalam populasi untuk dicari kromosom yang mempunyai nilai objektif terkecil. Proses ini akan terus berulang sampai ditemukan dua kromosom induk.

Misalkan pada pencarian kromosom induk pertama, kromosom yang dipilih adalah kromosom[2], kromosom[5], kromosom[6], kromosom[8], kromosom[10] dan kromosom[15]. Kromosom-kromosom ini kemudian dibandingkan dan didapat kromosom yang mempunyai nilai objektif yang terkecil adalah kromosom[10] dengan nilai 48.

Misalkan pada pencarian kromosom induk kedua, kromosom yang dipilih adalah kromosom[1], kromosom[4], kromosom[6], kromosom[9], kromosom[10], kromosom[13] dan kromosom[14]. Kromosom yang mempunyai nilai objektif terkecil yang didapat adalah kromosom[10] dengan nilai 48.

Karena kromosom induk pertama sama dengan kromosom induk kedua, maka kromosom induk kedua akan dipilih kembali. Misalkan pada pencarian selanjutnya kromosom yang dipilih adalah kromosom[1], kromosom[4], kromosom[7], kromosom[11], kromosom [14] damn kromosom[15]. Kromosom yang mempunyai nilai terkecil yang didapat adalah kromosom[11] dengan nilai 52.

Kromosom induk yang akan melakukan proses *crossover* adalah kromosom[10] dan kromosom[11].

Tabel 4.9 Kromosom Hasil Seleksi *Tournament Selection*

Kromosom ke	Bentuk Permutasi	Fungsi Objektif
1	3 5 1 6 4 2 7 3	48
2	1 5 3 7 6 2 4 1	52

4.3.1.4 *Crossover*

Proses *crossover* dilakukan dengan cara mengambil satu titik perpotongan pada gen orang tua untuk saling ditukarkan. Berdasarkan metode seleksi yang digunakan maka proses *crossover* dibedakan menjadi tiga bagian.

a. *Crossover* hasil seleksi *roulette wheel selection*

Sebelum dilakukan proses *crossover*, terlebih dahulu hitung banyaknya proses yang akan dilakukan, yaitu :

crossover_loop = (Populasi *size**%Pc)/2

$$= \frac{\frac{9*25}{100}}{2} = \frac{9*25}{100} * 2$$

= 450/100 = 4,5

= 5 kali persilangan

Untuk setiap persilangan, ambil dua kromosom orang tua secara acak dan ambil titik perpotongan secara acak pada kromosom orang tua.

i. Misalkan kromosom yang diambil pada proses persilangan yang pertama adalah kromosom[3] dan kromosom[7] dengan titik perpotongan dilakukan pada gen ke 4.

Offspring[1] = | 6 | 7 | 1 | 4 | 3 | 2 | 5 | 6 | >< | 4 | 2 | 5 | 7 | 6 | 3 | 1 | 4 |

= | 6 | 7 | 1 | 4 | 2 | 5 | 3 | 6 |

Offspring[2] = | 4 | 2 | 5 | 7 | 6 | 3 | 1 | 4 | >< | 6 | 7 | 1 | 4 | 3 | 2 | 5 | 6 |

= | 4 | 2 | 5 | 7 | 6 | 1 | 3 | 4 |

ii. Misalkan kromosom yang diambil pada proses persilangan yang kedua adalah kromosom[1] dan kromosom[5] dengan titik perpotongan dilakukan pada gen ke 3.

Offspring[1] = | 5 | 3 | 4 | 2 | 1 | 6 | 7 | 5 | >< | 3 | 6 | 5 | 7 | 1 | 4 | 2 | 3 |

= | 5 | 3 | 4 | 6 | 7 | 1 | 2 | 5 |

Offspring[2] = | 3 | 6 | 5 | 7 | 1 | 4 | 2 | 3 | >< | 5 | 3 | 4 | 2 | 1 | 6 | 7 | 5 |

= | 3 | 6 | 5 | 4 | 2 | 1 | 7 | 3 |

iii. Misalkan kromosom yang diambil pada proses persilangan yang ketiga adalah kromosom[6] dan kromosom[9] dengan titik perpotongan dilakukan pada gen ke 5.

Offspring[1] = | 3 | 5 | 1 | 6 | 4 | 2 | 7 | 3 | >< | 1 | 3 | 6 | 5 | 7 | 2 | 4 | 1 |

= | 3 | 5 | 1 | 6 | 4 | 7 | 2 | 3 |

Offspring[2] = | 1 | 3 | 6 | 5 | 7 | 2 | 4 | 1 | >< | 3 | 5 | 1 | 6 | 4 | 2 | 7 | 3 |

= | 1 | 3 | 6 | 5 | 7 | 4 | 2 | 1 |

iv. Misalkan kromosom yang diambil pada proses persilangan yang keempat adalah kromosom[2] dan kromosom[4] dengan titik perpotongan dilakukan pada gen ke 4.

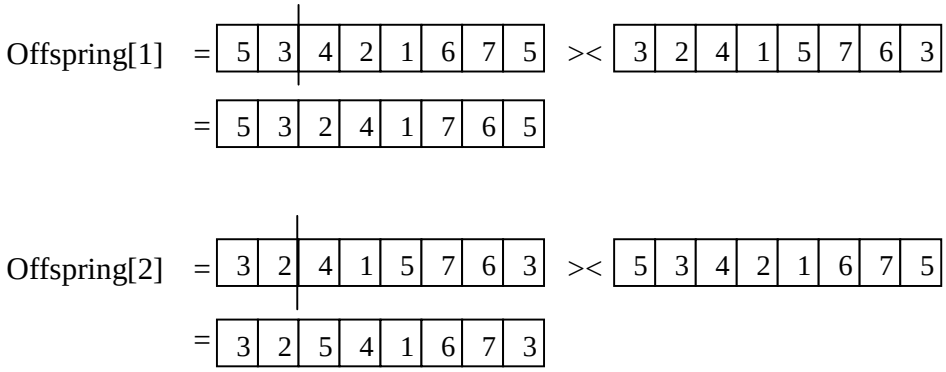
Offspring[1] = | 5 | 4 | 2 | 1 | 3 | 7 | 6 | 5 | >< | 7 | 3 | 2 | 1 | 5 | 6 | 4 | 7 |

= | 5 | 4 | 2 | 1 | 7 | 3 | 6 | 5 |

Offspring[2] = | 7 | 3 | 2 | 1 | 5 | 6 | 4 | 7 | >< | 5 | 4 | 2 | 1 | 3 | 7 | 6 | 5 |

= | 7 | 3 | 2 | 1 | 5 | 4 | 6 | 7 |

v. Misalkan kromosom yang diambil pada proses persilangan yang kelima adalah kromosom[1] dan kromosom[8] dengan titik perpotongan dilakukan pada gen ke 2.



Tabel 4.10 Kromosom Hasil *Crossover* Berdasarkan Seleksi *Roulette Wheel Selection*

Kromosom ke	**Bentuk Permutasi**	**Fungsi Objektif**
1	6 7 1 4 2 5 3 6	49
2	4 2 5 7 6 1 3 4	58
3	5 3 4 2 1 6 7 5	58
4	3 6 5 4 2 1 7 3	47
5	3 5 1 6 4 7 2 3	59
6	3 5 1 6 4 2 7 3	48
7	1 3 6 5 7 4 2 1	56
8	5 4 2 1 7 3 6 5	47
9	7 3 2 1 5 4 6 7	55
10	5 3 2 4 1 7 6 5	53
11	3 2 5 4 1 6 7 3	57

b. *Crossover* hasil seleksi *rank selection*

Sebelum dilakukan proses *crossover*, terlebih dahulu hitung banyaknya proses yang akan dilakukan, yaitu :

crossover_loop = (Populasi size*%Pc)/2

$$= \frac{\frac{9*25}{100}}{2} = \frac{9*25}{100} * 2$$

= 450/100 = 4,5

= 5 kali persilangan

Untuk setiap persilangan, ambil dua kromosom orang tua secara acak dan ambil titik perpotongan secara acak pada kromosom orang tua.

i. Misalkan kromosom yang diambil pada proses persilangan yang pertama adalah kromosom[3] dan kromosom[4] dengan titik perpotongan dilakukan pada gen ke 3.

Offspring[1] = | 1 | 5 | 3 | 7 | 6 | 2 | 4 | 1 | >< | 7 | 4 | 2 | 5 | 1 | 3 | 6 | 7 |

= | 1 | 5 | 3 | 7 | 4 | 2 | 6 | 1 |

Offspring[2] = | 7 | 4 | 2 | 5 | 1 | 3 | 6 | 7 | >< | 1 | 5 | 3 | 7 | 6 | 2 | 4 | 1 |

= | 7 | 4 | 2 | 1 | 5 | 3 | 6 | 7 |

ii. Misalkan kromosom yang diambil pada proses persilangan yang kedua adalah kromosom[5] dan kromosom[9] dengan titik perpotongan dilakukan pada gen ke 4.

Offspring[1] = | 5 | 3 | 4 | 2 | 1 | 6 | 7 | 5 | >< | 6 | 3 | 1 | 2 | 5 | 7 | 4 | 6 |

= | 5 | 3 | 4 | 2 | 6 | 1 | 7 | 5 |

Offspring[2] = | 6 | 3 | 1 | 2 | 5 | 7 | 4 | 6 | >< | 5 | 3 | 4 | 2 | 1 | 6 | 7 | 5 |

= | 6 | 3 | 1 | 2 | 5 | 4 | 7 | 6 |

iii. Misalkan kromosom yang diambil pada proses persilangan yang ketiga adalah kromosom[2] dan kromosom[8] dengan titik perpotongan dilakukan pada gen ke 4.

Offspring[1] = | 3 | 6 | 5 | 7 | 1 | 4 | 2 | 3 | >< | 6 | 7 | 1 | 4 | 3 | 2 | 5 | 6 |

= | 3 | 6 | 5 | 7 | 1 | 4 | 2 | 3 |

Offspring[2] = | 6 | 7 | 1 | 4 | 3 | 2 | 5 | 6 | >< | 3 | 6 | 5 | 7 | 1 | 4 | 2 | 3 |

= | 6 | 7 | 1 | 4 | 3 | 5 | 2 | 6 |

iv. Misalkan kromosom yang diambil pada proses persilangan yang keempat adalah kromosom[1] dan kromosom[6] dengan titik perpotongan dilakukan pada gen ke 2.

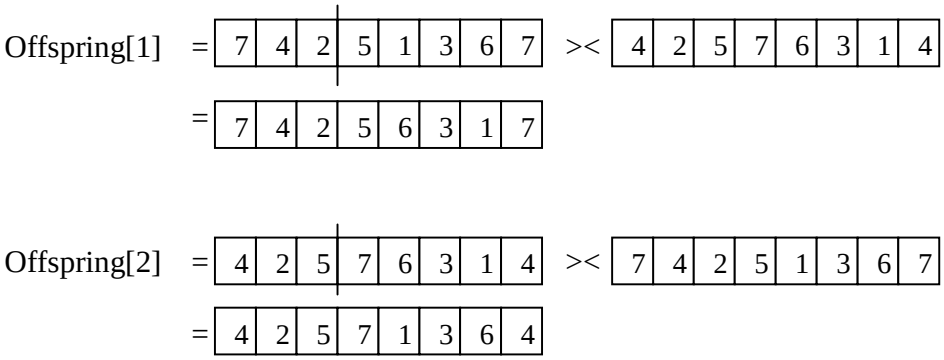
Offspring[1] = | 3 | 5 | 1 | 6 | 4 | 2 | 7 | 3 | >< | 4 | 5 | 2 | 6 | 1 | 7 | 3 | 4 |

= | 3 | 5 | 4 | 2 | 6 | 1 | 7 | 3 |

Offspring[2] = | 4 | 5 | 2 | 6 | 1 | 7 | 3 | 4 | >< | 3 | 5 | 1 | 6 | 4 | 2 | 7 | 3 |

= | 4 | 5 | 3 | 1 | 6 | 2 | 7 | 4 |

v. Misalkan kromosom yang diambil pada proses persilangan yang kelima adalah kromosom[4] dan kromosom[7] dengan titik perpotongan dilakukan pada gen ke 3.

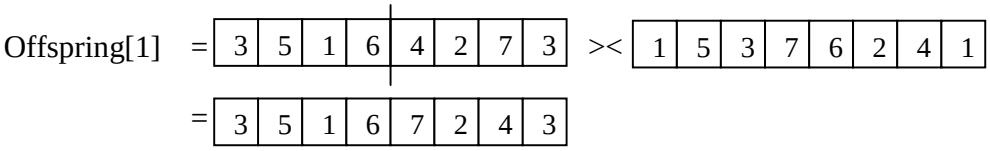


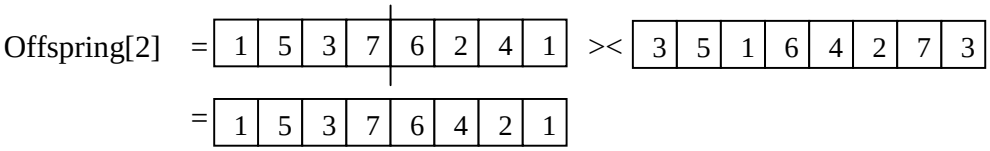
Tabel 4.11 Kromosom Hasil *Crossover* Berdasarkan Seleksi *Rank Selection*

Kromosom ke	Bentuk Permutasi	Fungsi Objektif
1	3 5 1 6 4 2 7 3	48
2	1 5 3 7 6 2 4 1	52
3	7 4 2 1 5 3 6 7	55
4	5 3 4 2 6 1 7 5	55
5	6 3 1 2 5 4 7 6	71
6	3 6 5 7 1 4 2 3	49
7	6 7 1 4 3 5 2 6	55
8	3 5 4 2 6 1 7 3	44
9	4 5 3 1 6 2 7 4	67
10	7 4 2 5 6 3 1 7	69
11	4 2 5 7 1 3 6 4	51

c. *Crossover* hasil seleksi *tournament selection*

Dari proses seleksi didapat kromosom induk yang akan melakukan proses *crossover* adalah kromosom[10] dan kromosom[11]. Misalkan perpotongan dilakukan pada gen ke 4.





Tabel 4.12 Kromosom Hasil *Crossover* Berdasarkan Seleksi *Tournament Selection*

Kromosom ke	Bentuk Permutasi	Fungsi Objektif
1	3 5 1 6 7 2 4 3	59
2	1 5 3 7 6 4 2 1	52

4.3.1.5 Mutasi

Mutasi yang dilakukan adalah dengan cara menukarkan antara dua gen yang dipilih secara acak dan melihat nilai objektif yang dihasilkannya. Proses ini dilakukan pada kromosom hasil *crossover*.

a. Mutasi hasil *crossover* seleksi *roulette wheel selection*

Sebelum dilakukan proses mutasi, terlebih dahulu hitung banyaknya proses yang akan dilakukan, yaitu :

mutasi_*loop* = Populasi *size**%Pm

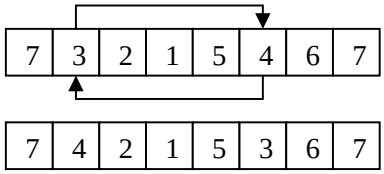
$$= \frac{11*10}{100}$$

= 110/100 = 1,1

= 1 kali mutasi

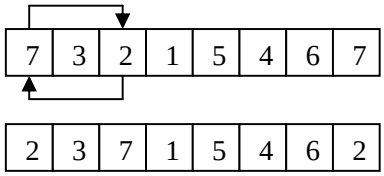
Jadi hanya ada satu kali mutasi yang akan dilakukan pada kromosom hasil *crossover*. Misalkan kromosom yang terpilih adalah kromosom[9] dengan gen yang saling ditukarkan adalah gen[2] dan gen[6], maka :

Offspring =



Karena nilai objektif kromosom induk sama besar dengan kromosom *offspring* (55=55) maka gen diacak kembali untuk saling ditukarkan. Misal yang terpilih adalah gen[1] dan gen[3].

Offspring =



Nilai Objektif kromosom *offspring* lebih kecil dibandingkan nilai objektif kromosom induk (52<55) maka kromosom *offspring* diambil sebagai generasi baru.

Tabel 4.13 Kromosom Hasil Mutasi berdasarkan Seleksi *Roulette Wheel Selection*

Kromosom ke	Bentuk Permutasi	Fungsi Objektif
1	6 7 1 4 2 5 3 6	49
2	4 2 5 7 6 1 3 4	58
3	5 3 4 2 1 6 7 5	58
4	3 6 5 4 2 1 7 3	47
5	3 5 1 6 4 7 2 3	59
6	3 5 1 6 4 2 7 3	48
7	1 3 6 5 7 4 2 1	56
8	5 4 2 1 7 3 6 5	47
9	2 3 7 1 5 4 6 2	52
10	5 3 2 4 1 7 6 5	53
11	3 2 5 4 1 6 7 3	57

b. Mutasi hasil *crossover* seleksi *rank selection*

Sebelum dilakukan proses mutasi, terlebih dahulu hitung banyaknya proses yang akan dilakukan, yaitu :

mutasi_*loop* = Populasi *size**%Pm

$$= \frac{11*10}{100}$$

= 110/100 = 1,1

= 1 kali mutasi

Jadi hanya ada satu kali mutasi yang akan dilakukan pada kromosom hasil *crossover*. Misalkan kromosom yang terpilih adalah kromosom[5] dengan gen yang saling ditukarkan adalah gen[4] dan gen[5], maka :

Offspring =

6	3	1	2	5	4	7	6

=

6	3	1	5	2	4	7	6

Nilai Objektif kromosom *offspring* lebih kecil dibandingkan nilai objektif kromosom induk (67<71) maka kromosom *offspring* diambil sebagai generasi baru.

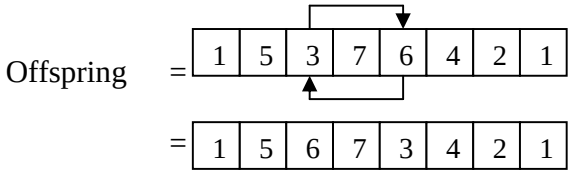
Tabel 4.14 Kromosom Hasil Mutasi berdasarkan Seleksi *Rank Selection*

Kromosom ke	Bentuk Permutasi	Fungsi Objektif
1	3 5 1 6 4 2 7 3	48
2	1 5 3 7 6 2 4 1	52
3	7 4 2 1 5 3 6 7	55
4	5 3 4 2 6 1 7 5	55
5	6 3 1 5 2 4 7 6	67

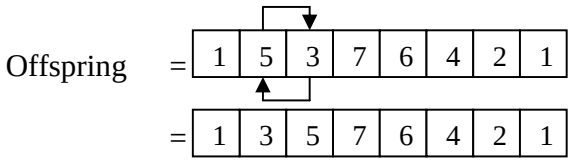
6	3 6 5 7 1 4 2 3	49
7	6 7 1 4 3 5 2 6	55
8	3 5 4 2 6 1 7 3	44
9	4 5 3 1 6 2 7 4	67
10	7 4 2 5 6 3 1 7	69
11	4 2 5 7 1 3 6 4	51

c. Mutasi hasil *crossover* seleksi *tournament selection*

Dari kromosom hasil *crossover*, maka akan dilakukan mutasi pada salah satu kromosom. Misalkan mutasi dilakukan pada kromosom[2] dengan gen yang akan dilakukan mutasi adalah gen[3] dan gen[5].



Karena nilai objektif kromosom induk lebih kecil daripada kromosom *offspring* (52<61) maka gen diacak kembali untuk saling ditukarkan. Misal yang terpilih adalah gen[2] dan gen[3].



Nilai Objektif kromosom *offspring* lebih kecil dibandingkan nilai objektif kromosom induk (48<52) maka kromosom *offspring* diambil sebagai generasi baru.

Tabel 4.15 Kromosom Hasil Mutasi berdasarkan Seleksi *Tournament Selection*

Kromosom ke	Bentuk Permutasi	Fungsi Objektif
1	3 5 1 6 7 2 4 3	59
2	1 3 5 7 6 4 2 1	48

4.3.1.6 Nilai Optimal

Setelah proses mutasi, maka satu generasi telah terselesaikan. Nilai optimal dapat dilihat pada populasi baru yang terbentuk.

a. Nilai Optimal hasil seleksi *roulette wheel selection*

Tabel 4.16 Generasi Kedua Algoritma Genetika menggunakan Metode *Roulette Wheel Selection*

Kromosom ke	Bentuk Permutasi	Fungsi Objektif
1	6 7 1 4 2 5 3 6	49
2	4 2 5 7 6 1 3 4	58
3	5 3 4 2 1 6 7 5	58
4	3 6 5 4 2 1 7 3	47
5	3 5 1 6 4 7 2 3	59
6	3 5 1 6 4 2 7 3	48
7	1 3 6 5 7 4 2 1	56
8	5 4 2 1 7 3 6 5	47
9	2 3 7 1 5 4 6 2	52
10	5 3 2 4 1 7 6 5	53
11	3 2 5 4 1 6 7 3	57

Pada Tabel 4.16 dapat dilihat bahwa seleksi *roulette wheel selection* menghasilkan kromosom yang paling optimal adalah 3→6→5→4→2→1→7→3 dan atau 5→4→2→1→7→3→6→5 dengan nilai 47.

Hingga generasi yang keenam (dapat dilihat pada lampiran A Langkah-langkah *Roulette Wheel Selection* Detail) kromosom yang paling optimal dengan metode ini adalah 1→3→5→4→2→6→7→1 dengan nilai 46.

b. Nilai optimal hasil seleksi *rank selection*

Tabel 4.17 Generasi Kedua Algoritma Genetika menggunakan Metode *Rank Selection*

Kromosom ke	Bentuk Permutasi	Fungsi Objektif
1	3 5 1 6 4 2 7 3	48
2	1 5 3 7 6 2 4 1	52
3	7 4 2 1 5 3 6 7	55
4	5 3 4 2 6 1 7 5	55
5	6 3 1 5 2 4 7 6	67
6	3 6 5 7 1 4 2 3	49
7	6 7 1 4 3 5 2 6	55
8	3 5 4 2 6 1 7 3	44
9	4 5 3 1 6 2 7 4	67
10	7 4 2 5 6 3 1 7	69
11	4 2 5 7 1 3 6 4	51

Pada Tabel 4.17 dapat dilihat bahwa seleksi *rank selection* menghasilkan kromosom yang paling optimal adalah 3→5→4→2→6→1→7→3 dengan nilai 44.

Hingga generasi yang keempat (dapat dilihat pada lampiran B Langkah-langkah *Rank Selection* Detail) kromosom yang paling optimal dengan metode ini adalah 3→5→4→2→6→1→7→3 dengan nilai 44.

c. Nilai optimal hasil seleksi *tournament selection*

Tabel 4.18 Generasi Kedua Algoritma Genetika menggunakan Metode *Tournament Selection*

Kromosom ke	Bentuk Permutasi	Fungsi Objektif
1	7 2 5 4 3 1 6 7	71
2	5 3 4 2 1 6 7 5	58
3	5 4 2 1 3 7 6 5	52
4	6 3 1 2 5 7 4 6	72
5	6 7 1 4 3 2 5 6	67
6	7 3 2 1 5 6 4 7	66
7	2 4 5 1 6 7 3 2	56
8	3 6 5 7 1 4 2 3	49
9	4 5 2 6 1 7 3 4	59
10	3 5 1 6 7 2 4 3	59
11	1 3 5 7 6 4 2 1	48
12	7 4 2 5 1 3 6 7	55
13	4 2 5 7 6 3 1 4	60
14	3 2 4 1 5 7 6 3	59
15	1 3 6 5 7 2 4 1	57

Pada Tabel 4.18 dapat dilihat bahwa seleksi *tournament selection* menghasilkan kromosom yang paling optimal adalah 1→3→5→7→6→4→2→1 dengan nilai 48.

Hingga generasi yang kelima (dapat dilihat pada lampiran C Langkah-langkah *Tournament Selection* Detail) kromosom yang paling optimal dengan metode ini adalah 1→3→5→4→2→6→7→1 dengan nilai 46.

4.3.2 Analisa Algoritma Genetika terhadap *Optimality*

1. *Optimality* metode seleksi *roulette wheel selection*

   Berdasarkan hasil dari *completeness*, solusi yang ditemukan merupakan solusi yang optimal dilihat dari generasi yang dilakukan selama pencarian solusi *travelling salesman problem.*

2. *Optimality* metode seleksi *rank selection*

   Berdasarkan hasil dari *completeness*, solusi yang ditemukan merupakan solusi yang optimal dilihat dari generasi yang dilakukan selama pencarian solusi *travelling salesman problem.*

3. *Optimality* metode seleksi *tournament selection*

   Berdasarkan hasil dari *completeness*, solusi yang ditemukan merupakan solusi yang optimal dilihat dari generasi yang dilakukan selama pencarian solusi *travelling salesman problem.*

4.3.3 Analisa Algoritma Genetika Terhadap *Time Complexity*

1. *Time complexity* algoritma genetika menggunakan metode *roulette wheel selection*

   Waktu yang diperlukan algoritma genetika menggunakan metode *roulette wheel selection* pada penyelesaian kasus TSP ini adalah :

   $T(n) = dnm^2$

   $= 6.15.7^2$

   = 4.410 mikrodetik

2. *Time complexity* algoritma genetika menggunakan metode *rank selection*

   Waktu yang diperlukan algoritma genetika menggunakan metode *rank selection* pada penyelesaian kasus TSP ini adalah :

   $T(n) = dnm^2$

   $= 3.15.7^2$

   = 2.205 mikrodetik

3. *Time complexity* algoritma genetika menggunakan metode *tournament selection*

   Waktu yang diperlukan algoritma genetika menggunakan metode *tournament selection* pada penyelesaian kasus TSP ini adalah :

   $T(n) = dnm^2$

   $= 5.15.7^2$

   = 3.675 mikrodetik

4.3.4 Analisa Algoritma Genetika Terhadap *Space Complexity*

2. *Space complexity* algoritma genetika menggunakan metode *roulette wheel selection*

   Memori yang diperlukan algoritma genetika menggunakan metode *roulette wheel selection* pada penyelesaian kasus TSP ini adalah berbanding lurus dengan waktu yang diperlukan, yaitu:

T(n) = dnm^2

= $6.15.7^2$

= 4.410 byte

3. *Space complexity* algoritma genetika menggunakan metode *rank selection*

   Memori yang diperlukan algoritma genetika menggunakan metode *rank selection* pada penyelesaian kasus TSP ini adalah berbanding lurus dengan waktu yang diperlukan, yaitu :

   T(n) = dnm^2

   = $3.15.7^2$

   = 2.205 byte

4. *Space Complexity* Algoritma Genetika Menggunakan Metode *tournament selection*

   Memori yang diperlukan Algoritma Genetika menggunakan metode *tournament selection* pada penyelesaian kasus TSP ini adalah berbanding lurus dengan waktu yang diperlukan, yaitu :

   T(n) = dnm^2

   = $5.15.7^2$

   = 3.675 byte

4.4 Perancangan Simulasi

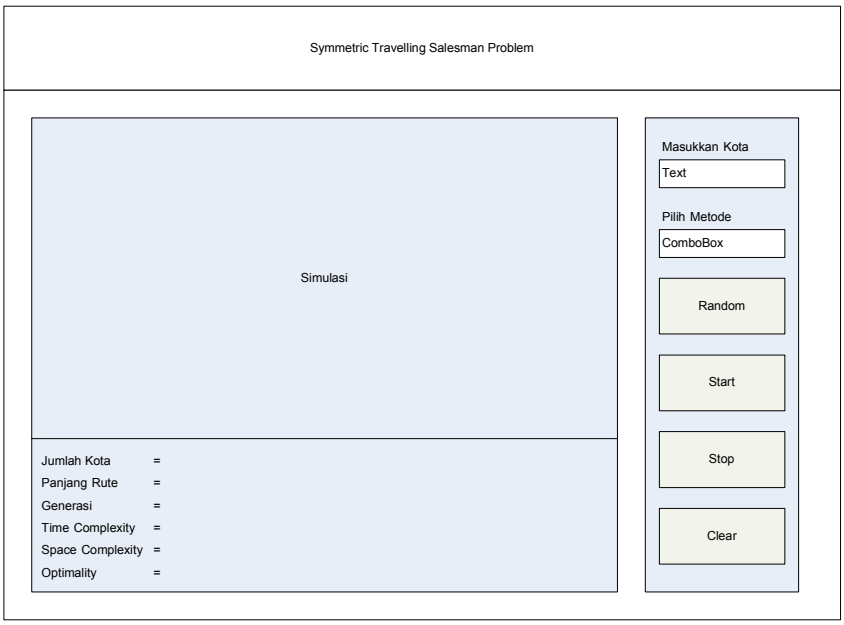
Perancangan simulasi algoritma genetika untuk kasus *travelling salesman problem* terdiri dari lingkungan perancangan dan perancangan antarmuka.

4.4.1 Lingkungan Perancangan

Lingkungan perancangan yang digunakan adalah sebagai berikut :

1. Perangkat keras
   a. Processor : Intel® Pentium® 4 2.00 GHz
   b. Memori : 224 MB of RAM
   c. Harddisk : 20 GB
2. Perangkat lunak
   a. Sistem operasi : Windows XP Profesional Vers. 2002 Service Pack 2
   b. Bahasa pemrograman : Microsoft Visual C++ 6.0 Service Pack 5

4.4.2 Perancangan Antarmuka



Gambar 4.3 Rancangan Menu *Travelling Salesman Problem*

Perancangan antarmuka seperti pada Gambar 4.3 adalah :

1. Pada Text masukkan jumlah kota yang diinginkan.
2. Pada ComboBox pilih metode seleksi yaitu *Roultte Wheel*, *Rank*, atau *Tournament*
3. Tombol Random Node akan menghasilkan graf pada Layar simulasi sesuai dengan jumlah kota
4. Tombol Start akan membuat layar simulasi bekerja untuk mendapatkan penyelesaian kasus TSP
5. Tombol Stop akan membuat layar simulasi berhenti sehingga hasil akhir merupakan penyelesaian kasus TSP

6. Tombol Clear akan menghapus hasil akhir dari proses dan kembali ke graf yang dibentuk
7. Hasil akhir akan ditampilkan pada Output dibawah layar simulasi

BAB V

IMPLEMENTASI DAN PENGUJIAN

5.1 Implementasi

Implementasi merupakan tahap dimana sistem siap dioperasikan pada keadaan yang sebenarnya, termasuk kegiatan penulisan kode program yang digunakan, sehingga akan diketahui apakah sistem yang dibuat benar-benar dapat menghasilkan solusi atau target yang diinginkan.

5.1.1 Alasan Pemilihan Perangkat Lunak

Perangkat lunak yang digunakan yaitu Microsoft Visual C++ 6.0. Pemilihan perangkat lunak ini didasarkan pertimbangan bahwa Microsoft Visual C++ 6.0 merupakan bahasa pemrograman yang berbasis *object oriented programming* (OOP) yang memungkinkan *programmer* untuk menggunakan *coding-coding* yang pernah dipakai untuk pembuatan program baru dan juga memudahkan pengembangan program yang sudah ada.

5.1.2 Batasan Implementasi

Dalam mengimplementasikan tugas akhir ini diberikan batasan-batasan sebagai berikut :

1. Mengimplementasikan algoritma genetika dengan target perhitungan *symmetric travelling salesman problem* optimal yang variabelnya berupa panjang lintasan.
2. *Vertex* dan *edge* dimunculkan secara *random*.
3. Jumlah kota maksimal 200 kota.

5.1.3 Lingkungan Implementasi

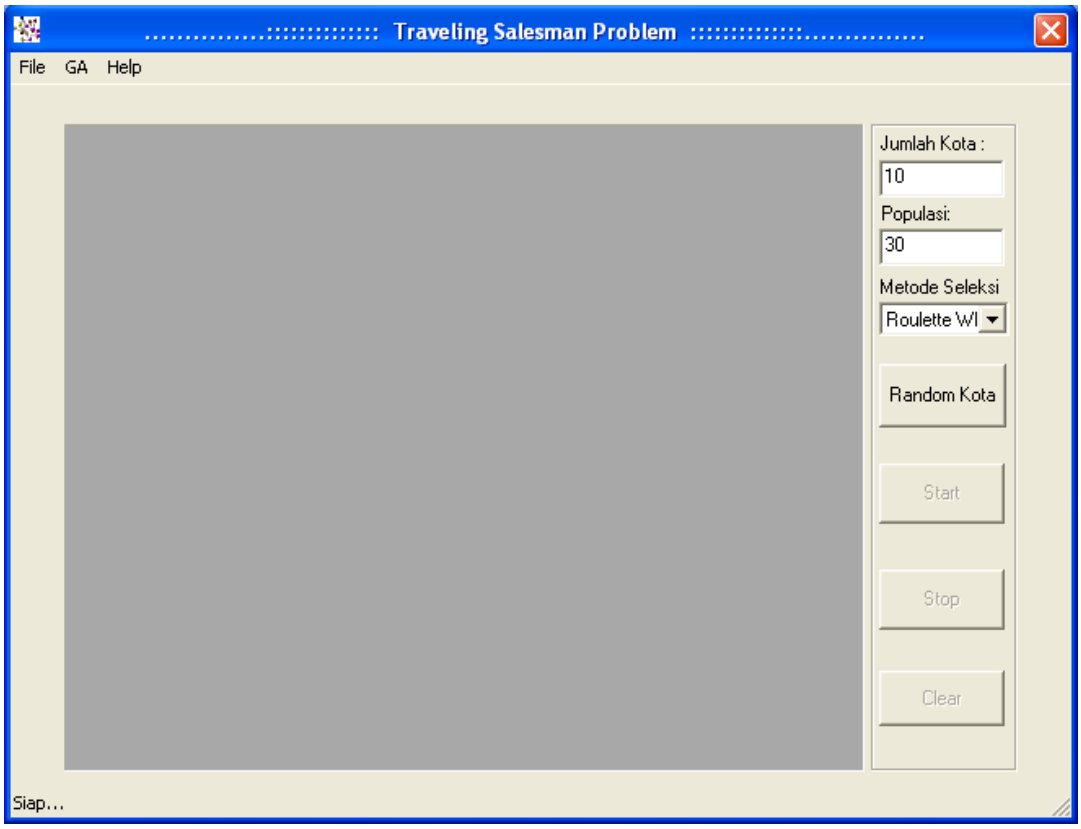
Lingkungan implementasi simulasi adalah sebagai berikut :

1. Perangkat keras
   a. Processor : Intel® Pentium® 4 2.00 GHz
   b. Memori : 224 MB of RAM
   c. Harddisk : 20 GB
2. Perangkat lunak
   a. Sistem operasi : Windows XP Profesional Vers. 2002 Service Pack 2
   b. Bahasa pemrograman : Microsoft Visual C++ 6.0 Service Pack 5

5.1.4 Hasil Implementasi

Hasil implementasi berupa mensimulasikan pencarian solusi pada permasalah *travelling salesman problem* dengan menggunakan teknik pencarian *heuristic search*, dimana algoritma yang digunakan adalah algoritma genetika.

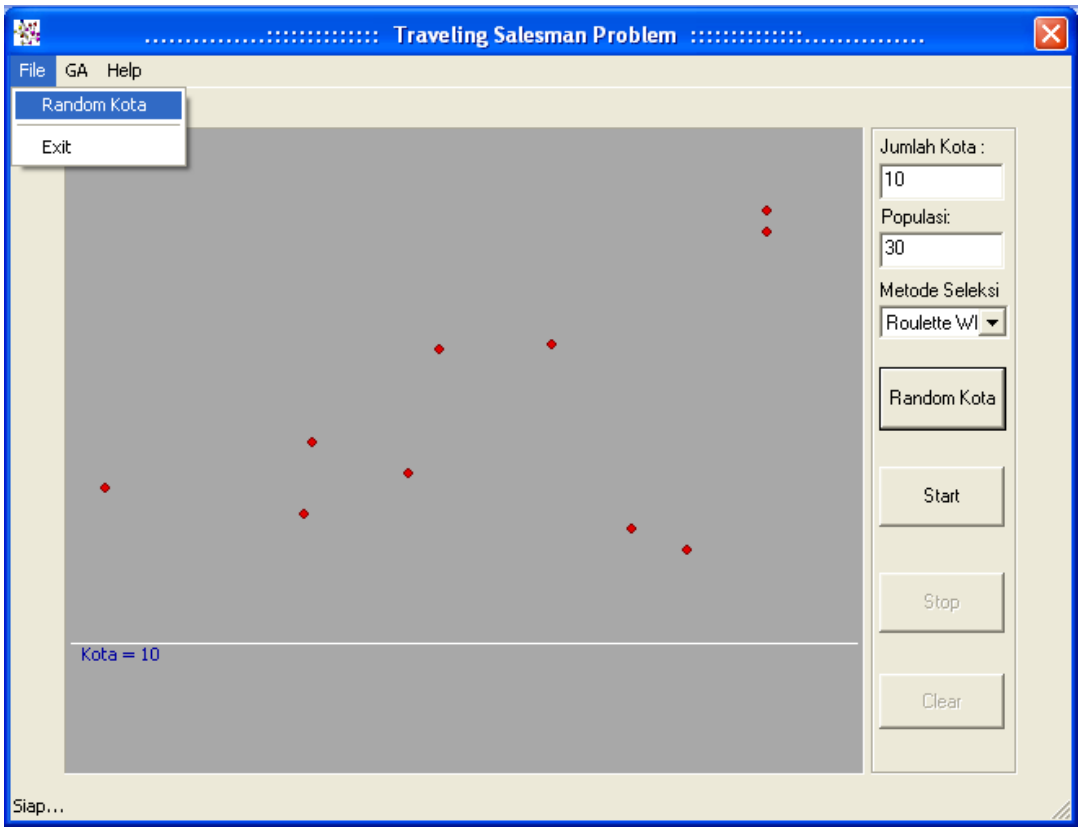
5.1.4.1 Antarmuka Menu Utama Aplikasi



Gambar 5.1 Antarmuka Menu Utama Aplikasi

Ketika masuk ke dalam menu utama, jumlah kota sudah terisi 10 kota, populasi 25 dan metode seleksi di *setting* Roulette Wheel. Ubahlah jumlah kota, populasi dan metode seleksi sesuai dengan yang diinginkan.

5.1.4.2 Antarmuka Random Kota



Gambar 5.2 Antarmuka Random Kota

Hasil eksekusi *click* Random Kota yaitu menghasilkan *vertex* sebanyak jumlah kota yang diinginkan. *Edge* pada antarmuka ini tidak ditampilkan, karena apabila ditampilkan, *interface* untuk pengguna dirasa kurang bersahabat.

5.2 Pengujian Simulasi

Pengujian simulasi merupakan salah satu tahap di dalam menemukan kesalahan-kesalahan program yang mungkin terjadi. Sebelum program diaplikasikan,

terlebih dahulu dilakukan pengujian yang bertujuan untuk membandingkan hasil analisa dan kenyataan yang dihadapi dan membuktikan prilaku algoritma secara nyata dengan perhitungan waktu proses dalam format detik.

Penulis akan mengujikan pencarian solusi *symmetric travelling salesman problem* dengan jumlah kota dan populasi yang bervariasi, serta metode seleksi yang digunakan yaitu *roulette wheel selection, rank selection,* dan *tournament selection*.

5.2.1 Lingkungan Pengujian

Lingkungan pengujian yang akan digunakan adalah sebagai berikut :

1. Perangkat keras
   a. Processor : Intel® Pentium® 4 2.00 GHz
   b. Memori : 224 MB, 512 MB dan 1 GB of RAM
   c. Harddisk : 20 GB
2. Perangkat lunak
   a. Sistem operasi : Windows XP Profesional Vers. 2002 Service Pack 2
   b. Bahasa pemrograman : Microsoft Visual C++ 6.0 Service Pack 5

5.2.2 Pengujian pada Metode *Roulette Wheel Selection*

Pengujian dilakukan dengan cara memasukkan jumlah kota pada Textbox dan tekan tombol Random Kota untuk menghasilkan titik-titik pada layar simulasi. Kemudian memilih metode *roulette wheel* pada combo Metode Seleksi dan tekan

tombol Start untuk menjalankan sistem. Sistem akan berhenti jika tombol Stop sudah ditekan.

Uji coba pada metode ini dilakukan sebanyak 100 kali dengan nilai kota dan ukuran populasi yang bervariasi, di antaranya jumlah kota 10 dengan populasi 30, jumlah kota 10 dengan populasi 50, dan jumlah kota 10 dengan populasi 80.

5.2.3 Pengujian pada Metode *Rank Selection*

Pengujian dilakukan dengan cara memilih metode *rank selection* pada combo Metode Seleksi. Kemudian menghapus *edge-edge* yang ada pada layar simulasi hasil dari proses seleksi *roulette wheel* tanpa menghapus titik yang telah terbentuk dengan menekan tombol Clear. Tekan tombol Start untuk menjalankan sistem. Tekan tombol Stop untuk menghentikan sistem.

Uji coba pada metode seleksi ini juga dilakukan sebanyak 100 kali dengan nilai kota, kota yang dibentuk dan ukuran populasi yang sama dengan pengujian pada metode *roulette wheel*.

5.2.4 Pengujian pada Metode *Tournament Selection*

Pengujian dengan metode ini dilakukan dengan cara memilih metode *tournament selection* pada combo Metode Seleksi. Kemudian sama halnya dengan metode *rank*, menghapus *edge-edge* yang ada pada layar simulasi hasil dari proses seleksi *rank* tanpa menghapus titik yang telah terbentuk dengan menekan tombol

Clear. Tekan tombol Start untuk menjalankan sistem. Tekan tombol Stop untuk menghentikan sistem.

Uji coba pada metode seleksi ini juga dilakukan sebanyak 100 kali dengan nilai kota, kota yang dibentuk dan ukuran populasi yang sama dengan pengujian pada metode *roulette wheel* dan metode *rank*.

5.2.5 Hasil Pengujian

Hasil pengujian yang akan ditunjukkan yaitu hasil pengujian terhadap parameter *time complexity*, hasil pengujian terhadap parameter *space complexity*, hasil pengujian terhadap parameter *completeness*, dan hasil pengujian terhadap parameter *optimality*.

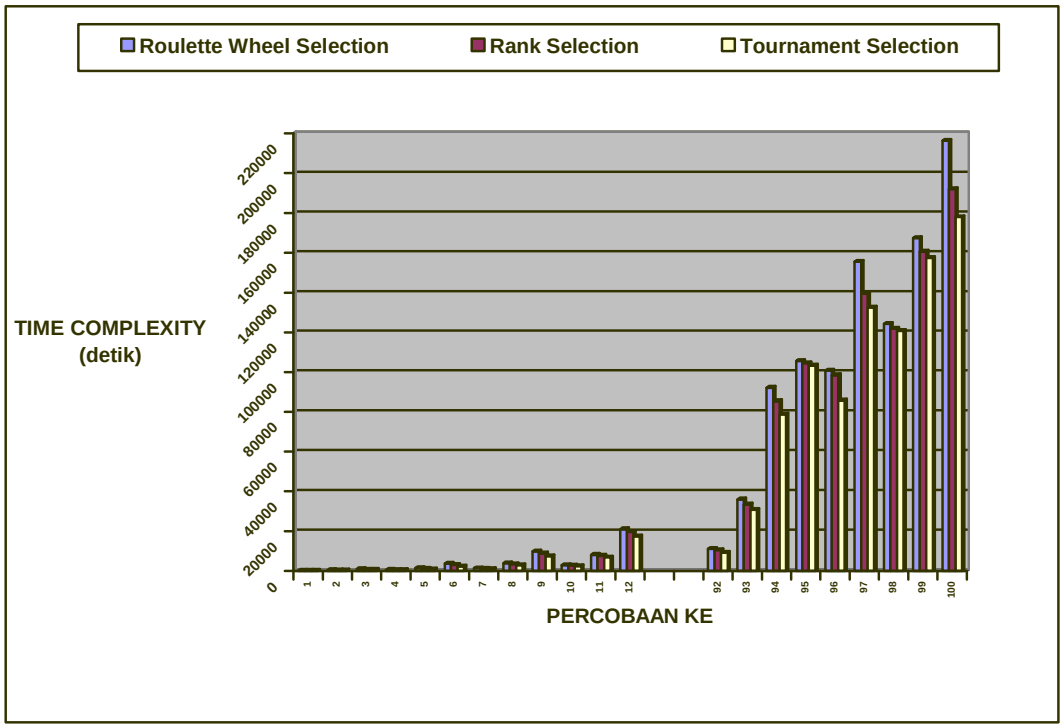
5.2.5.1 Hasil Pengujian terhadap Parameter *Time Complexity*

Untuk metode *tournament selection*, *time complexity* yang dibutuhkan tidak terlalu lama karena pencarian orang tua pertama dan kedua didasarkan pada nilai cost yang paling optimal yang ada pada populasi untuk dilakukan proses selanjutnya. Metode *rank selection* mempunyai waktu yang lebih lama dibandingkan metode *roulette wheel selection* karena selain melihat *cost* yang optimal di dalam populasi, juga dilihat probabilitas kromosom tersebut dalam populasi. Metode *roulette wheel selection* mempunyai waktu yang lebih lama dibandingkan kedua metode lainnya karena pemilihan orang tua tidak dilakukan berdasarkan nilai *cost* yang optimal tetapi

hanya berdasarkan nilai probabilitas yang ada pada populasi, jadi kemungkinan kromosom yang mempunyai nilai *cost* yang optimal akan terbuang.

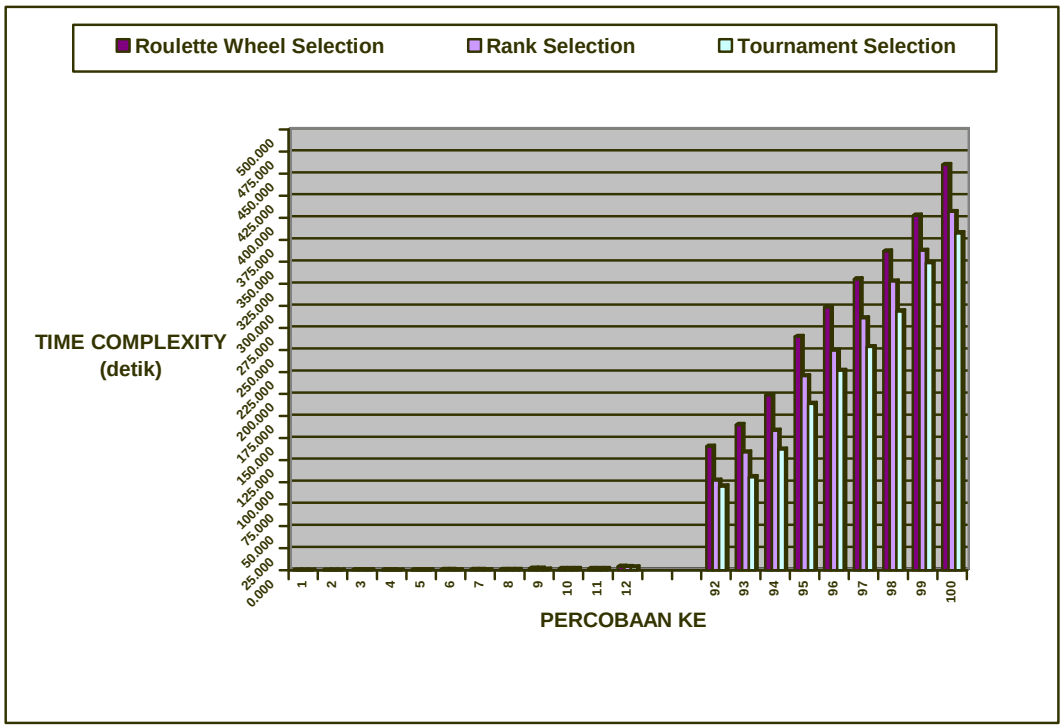
Penjelasan mengenai *symmetric travelling salesman problem* dengan 100 kali percobaan dapat dilihat di Lampiran D Pengujian Detail.

Grafik *time complexity* proses untuk permasalahan *symmetric travelling salesman problem* secara manual adalah sebagai berikut :



Gambar 5.3 Grafik Perbandingan Metode *Roulette Wheel Selection, Rank Selection* dan *Tournament Selection* berdasarkan *Time Complexity* Secara Manual

Grafik *time complexity* proses untuk permasalahan *symmetric travelling salesman problem* secara sistem adalah sebagai berikut :



Gambar 5.4 Grafik Perbandingan Metode *Roulette Wheel Selection, Rank Selection* dan *Tournament Selection* berdasarkan *Time Complexity* Secara Sistem

Berdasarkan pengujian *time complexity* untuk beberapa susunan jumlah kota dan banyak populasi pada TSP, didapat suatu analisa terhadap ketiga metode *roulette wheel selection, rank selection* dan *tournament selection* yaitu metode *tournament selection* membutuhkan waktu yang lebih sedikit dibandingkan metode *rank selection* dan metode *roulette wheel selection.*

Hal ini dapat dilihat dari Tabel 5.1, Gambar 5.3 dan Gambar 5.4 yang menunjukkan bahwa untuk 100 kali percobaan hingga jumlah kota 200,secara manual persentase perbandingan *time complexity* metode *tournament selection* dibandingkan

roulette wheel selection dalam hitungan detik adalah 11.72%, dan secara sistem persentase perbandingan *time complexity* metode *tournament selection* dibandingkan *roulette wheel selection* dalam hitungan detik adalah 22.46%.

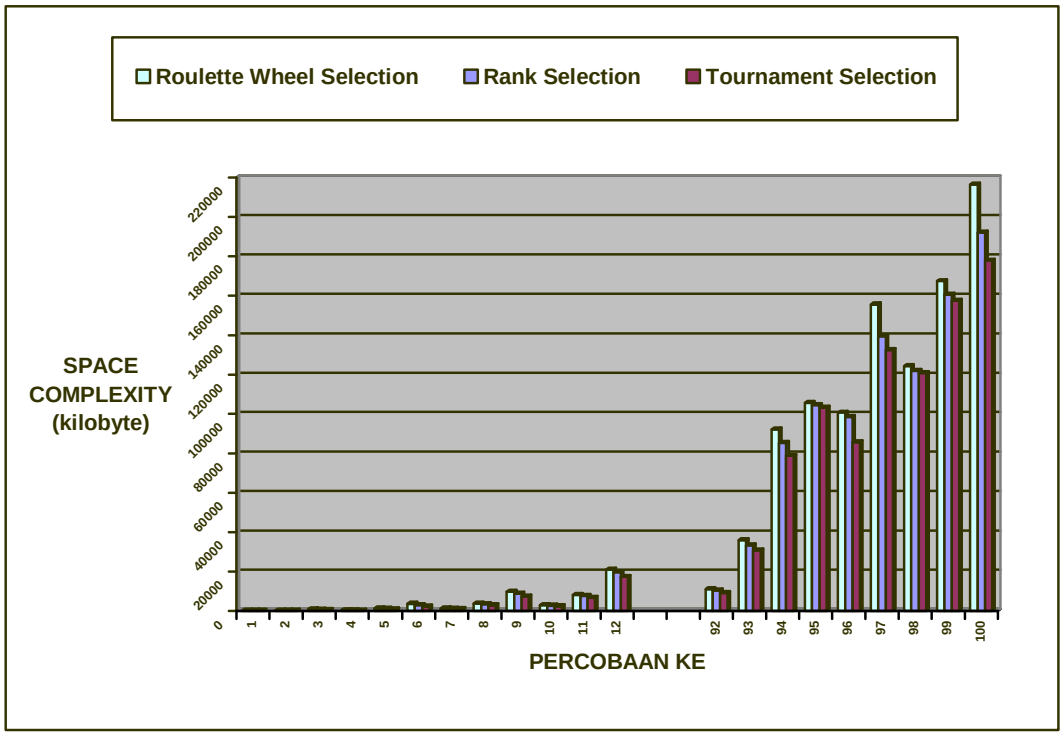
Untuk persentase perbandingan *time complexity* metode *tournament selection* dibandingkan *rank selection* secara manual dalam hitungan detik adalah 5.68%, secara sistem persentase perbandingan *time complexity* metode *tournament selection* dibandingkan *rank selection* dalam hitungan detik adalah 9.59%.

5.2.5.2 Hasil Pengujian terhadap Parameter *Space Complexity*

Eksekusi ketiga metode seleksi pada program menunjukkan bahwa *space complexity* memiliki tingkat perbedaan yang cukup signifikan. Pada metode *tournament selection*, memori yang digunakan lebih sedikit daripada metode *roulette wheel selection* dan metode *rank selection*.

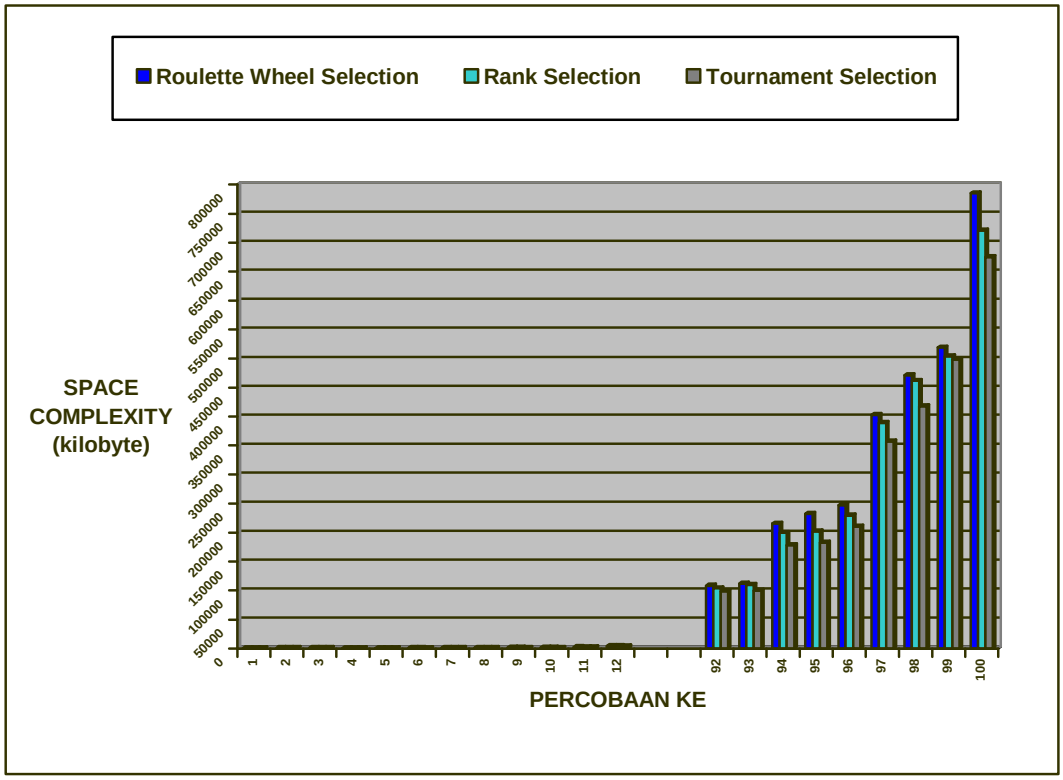
Penjelasan mengenai *symmetric travelling salesman problem* dengan 100 kali percobaan dapat dilihat di Lampiran D Pengujian Detail.

Grafik *space complexity* proses untuk permasalahan *symmetric travelling salesman problem* secara manual adalah sebagai berikut :



Gambar 5. 5 Grafik perbandingan *Roulette Wheel Selection, Rank Selection* dan *Tournament Selection* berdasarkan *Space Complexity* Secara Manual

Grafik *space complexity* proses untuk permasalahan *symmetric travelling salesman problem* secara sistem adalah sebagai berikut :



Gambar 5. 6 Grafik perbandingan *Roulette Wheel Selection, Rank Selection* dan *Tournament Selection* berdasarkan *Space Complexity* Secara Sistem

Berdasarkan pengujian *space complexity* untuk beberapa susunan jumlah kota dan banyak populasi pada TSP, didapat suatu analisa terhadap ketiga metode *roulette wheel selection, rank selection* dan *tournament selection* yaitu metode *tournament selection* membutuhkan waktu yang lebih sedikit dibandingkan metode *rank selection* dan metode *roulette wheel selection.*

Hal ini dapat dilihat dari Tabel 5.2, Gambar 5.2 dan Gambar 5.6 yang menunjukkan bahwa untuk 100 kali percobaan hingga jumlah kota 200, secara

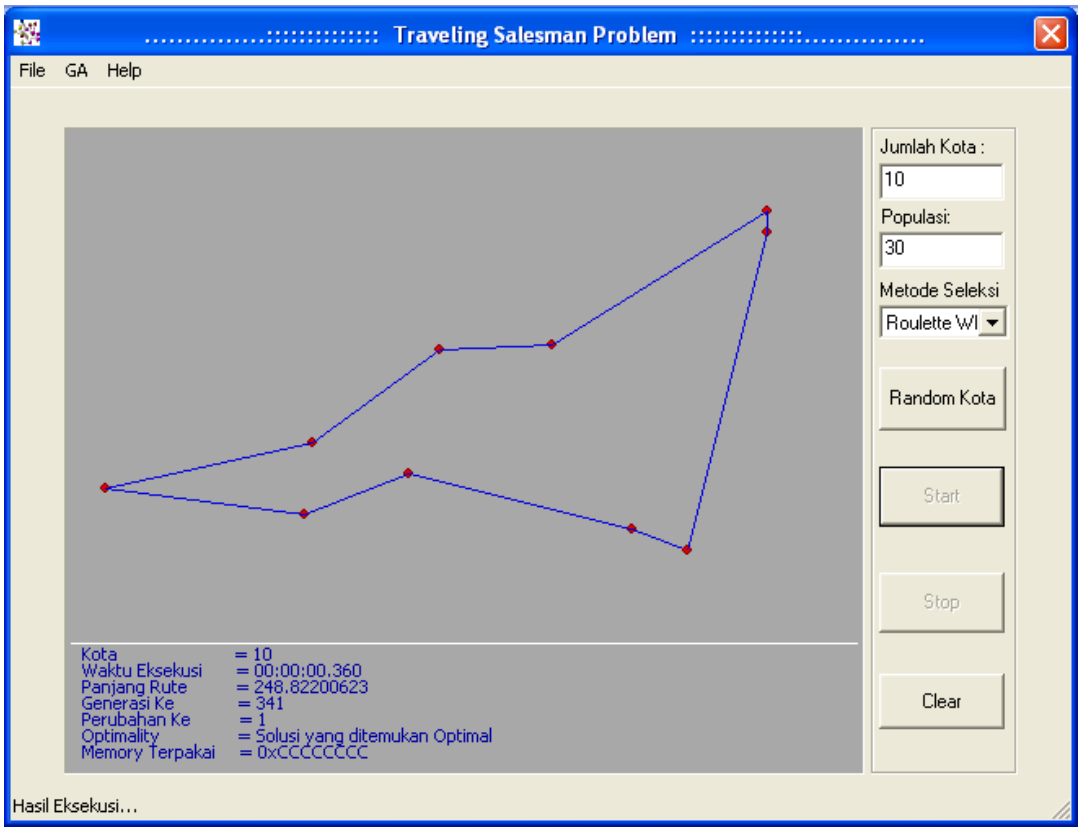
manual persentase perbandingan *space complexity* metode *tournament selection* dibandingkan *roulette wheel selection* dalam hitungan kilobyte adalah 11.72%, dan secara sistem persentase perbandingan *space complexity* metode *tournament selection* dibandingkan *roulette wheel selection* dalam hitungan kilobyte adalah 12.20%.

Untuk persentase perbandingan *space complexity* metode *tournament selection* dibandingkan *rank selection* secara manual dalam hitungan kilobyte adalah 5.68%, secara sistem persentase perbandingan *space complexity* metode *tournament selection* dibandingkan *rank selection* dalam hitungan kilobyte adalah 6.97%.

5.2.5.3 Hasil Pengujian terhadap Parameter *Completeness*

Pengujian yang dilakukan dengan parameter *completeness* ditujukan untuk melihat apakah solusi *symmetric travelling salesman problem* merupakan solusi yang paling optimal. Pada Algoritma Genetika, permasalahan TSP selalu menemukan solusi yang mendekati optimal. Jadi untuk ketiga metode *roulette wheel selection*, *rank selection* dan *tournament selection* merupakan metode yang *completeness* untuk menyelesaikan permasalahan TSP.

Di bawah ini merupakan penggunaan metode *roulette wheel selection* dalam menemukan solusi dari *Symmetric travelling salesman problem* dengan jumlah kota 10 dan populasi 30.



Gambar 5.7 Penggunaan Metode *Roulette Wheel Selection*

5.2.5.4 Hasil Pengujian terhadap Parameter *Optimality*

Pengujian terhadap perbandingan ketiga metode dengan parameter *time complexity*, *space complexity* dan *completeness* menunjukkan bahwa metode *tournament selection* lebih optimal dibandingkan dengan kedua metode lainnya.

5.2.6 Kesimpulan Pengujian

Dari 100 kali pengujian yang dilakukan terhadap metode *roulette wheel selection*, *rank selection* dan *tournament selection* dapat disimpulkan bahwa :

1. Pada pengujian *time complexity*, untuk jumlah kota lebih dari 30 metode *roulette wheel selection* bekerja lebih lambat dibandingkan dengan kedua metode lainnya. Metode yang paling cepat untuk kasus TSP ini adalah metode *tournament selection*. Metode *rank selection* berada di antara kedua metode lainnya. Lebih cepat dibandingkan metode *roulette wheel selection* dan lebih lambat daripada metode *tournament selection*. Penghitungan *time complexity* ketiga metode pada program *symmetric travelling salesman problem* didapat dengan cara menghitung waktu proses yang dilakukan untuk mendapatkan solusi yang paling optimal. Apabila dalam waktu yang berbeda didapatkan hasil panjang rute yang terus sama, maka dianggap bahwa solusi itu adalah solusi yang paling optimal.
2. Pada *space complexity*, eksekusi metode *tournament selection* membutuhkan memori yang lebih sedikit dibandingkan kedua metode lainnya. Dengan perhitungan secara manual, pemakaian *space* pada metode *roulette wheel selection, rank selection* dan *tournament selection* berbanding lurus dengan *time complexity*. Untuk implementasi pada kasus TSP, *space complexity* didapatkan dengan menghitung banyak memori yang digunakan untuk menjalankan program. Makin lama waktu yang dikerjakan, makin besar pula memori yang dibutuhkan.
3. Pada pengujian *completeness*, ketiga metode seleksi ini mampu menemukan solusi dari permasalah *symmetric travelling salesman problem* dengan cepat meskipun kemungkinan bahwa solusi itu yang paling optimal sangat sulit dipastikan. Solusi yang dihasilkan merupakan solusi yang paling mendekati

optimal. Jumlah kota yang banyak akan membuat pencarian solusi semakin lambat. Oleh sebab itu, banyaknya kota dibatasi hanya sampai 200 kota.

4. Pada *optimality*, ketiga metode sama-sama optimal meskipun dari panjang rute yang dihasilkan, waktu yang dibutuhkan dan ruang yang terpakai, metode *tournament selection* lebih *optimality* dibandingkan kedua metode seleksi lainnya.

BAB VI

PENUTUP

6.1 Kesimpulan

Berdasarkan pengujian yang dilakukan pada ketiga metode seleksi *roulette wheel selection*, *rank selection* dan *tournament selection* dapat disimpulkan sebagai berikut :

1. Pada parameter *time complexity*, metode *tournament selection* lebih cepat dibandingkan dengan metode *roulette wheel selection* dan metode *rank selection.* Pengujian dari sisi sistem, menunjukkan persentase kecepatan *tournament selection* dibandingkan *roulette wheel selection* untuk 100 kali percobaan hingga jumlah kota 200 adalah 22.46% dan persentase kecepatan *tournament selection* dibandingkan *rank selection* untuk untuk 100 kali percobaan hingga jumlah kota 200 adalah 9.59%.
2. Pada parameter *space complexity*, metode *tournament selection* membutuhkan memori yang lebih sedikit daripada metode *roulette wheel selection* dan metode *rank selection*. Pengujian dari sisi sistem, memperlihatkan persentase memori *tournament selection* dibandingkan *roulette wheel selection* untuk untuk 100 kali percobaan hingga jumlah kota 200 adalah 12.2% dan persentase memori *tournament selection* dibandingkan

rank selection untuk 100 kali percobaan hingga jumlah kota 200 adalah 6.97%.

3. Pada parameter *completeness*, metode *roulette wheel selection*, *rank selection* dan *tournament selection*, menemukan solusi yang diinginkan. *Completeness* berhubungan dengan generasi yang dihasilkan. Makin banyak generasi yang dilakukan, makin menentukan bahwa solusi optimal telah dicapai.
4. Pada parameter o*ptimality*, solusi yang ditemukan merupakan solusi optimal berdasarkan banyak proses yang telah dilakukan. Panjang rute pada proses pencarian TSP menentukan bahwa solusi optimal yang diperoleh. Dalam waktu yang relatif cepat metode *tournament selection* menemukan panjang rute yang lebih optimal dibandingkan metode *roulette wheel selection* dan metode *rank selection*.
5. Jika jumlah kota dimasukkan lebih dari 200 maka proses akan membutuhkan waktu yang cukup lama dan proses pada aplikasi akan berkerja lambat.

6.2 Saran

Beberapa hal yang dapat diungkapkan sebagai saran untuk perbaikan di masa yang akan datang adalah sebagai berikut :

1. Untuk pengembangan selanjutnya, dapat dilakukan perbandingan metode-metode yang ada pada proses *cossover* dan perbandingan metode-metode yang ada pada proses mutasi.

2. Dapat dilakukan dengan menggunakan berbagai notasi selain notasi Big-O, seperti notasi Θ, notasi-Ω, notasi-ω dan lain-lain.
3. Selain graf lengkap dan berbobot, dapat dikembangkan pada graf yang tidak lengkap, berarah dan berbobot atau pada graf yang lengkap, berarah dan berbobot.
4. Selain *symetric travelling salesman problem*, dapat dikembangkan pada variasi TSP yang lain, seperti *assymetric travelling salesman problem, hamiltonian cycle problem, the euclidean traveling salesman selection problem, online travelling salesman problem* dan lain-lain.

DAFTAR PUSTAKA

Boukreev, Konstantin "*Genetic Algorithm and Travelling Salesman Problem*" [Online] Available http://www.generation5.org/content/2001/tspapp.asp, diakses 28 Mei 2009

Cummings, Nigel, "*The OR Sosiety, A BRIEF HISTORY of TSP*" [Online] Available http://www.orsoc.org.uk/about/topic/news/tspjune.htm, diakses 9 Maret 2009

Desiani, Anita, dan Muhammad Arhami, "*Konsep Kecerdasan Buatan*", ANDI, Yogyakarta, 2006

Gen, M., dan Cheng R., "*Genetic Algorithm and Enginering Design*", John Wiley & Sons Inc., Japan, 1997

Hariyanto, Bambang, "*Struktur Data Memuat Dasar Pengembangan Orientasi Objek*", edisi kedua, INFORMATIKA, Bandung, 2003

http://www.iba.k.u-tokyo.ac.jp/english/Research.htm, diakses 18 Desember 2008

Jaakkola, Toni "*Solving the Travelling Salesman Problem Using a Genetic Algorithm*" [Online] Available http://www.www2.lut.fi/~tjaakkol/GA-TSP.ppt, diakses 15 Maret 2009

JJB, "*Genetic Algorithm to solve the Travelling Salesman problem*" [Online] Available http://www.cbunn.cithep.caltech.edu/Java/Genetic.java, diakses 20 Mei 2009

Kusumadewi, Sri, "*Artificial Intelligence : Teknik dan Aplikasinya*", Graha Ilmu, Yogyakarta, 2003

Munir, Rinaldi, "*Teknik Penyelesaian Persoalan*", Institut Teknologi Bandung, Bandung, 2001

Obitko, Marek "*Genetic Algorithm*" [Online] Available http://cs.felk.cvut.cz/, diakses 20 Mei 2009

Pongcharoen, Pupong, Warattapop Chainate, dan Peeraya Thapatsuwan, "*Exploration of Genetic Parameters and Operators through Travelling Salesman Problem*", *ScienceAsia 33*, Thailand, 2007

Rawlins, J.E. Gregory, "*Foundation Of Genetic Algorithm*", Morgan Kaufmann Publisher Inc., California, 1991

Reinelt, Genhard "*TSPLIB*" [Online] Available http://www.iwr.uni-heidelberg.de, diakses 28 Mei 2009

Russell, Stuart, dan Norvig, Peter, "*Artificial Inteligient A Modrn Approach*", Prentice – Hall International, Inc, New Jersey, 2003

Suyoto, "*Intelegensi Buatan : Teori dan Pemrograman*", Penerbit Gava Media, Yogyakarta, 2004

Thiang, Ronald Kurniawan, dan Hany Fernando, "*Implementasi Algoritma Genetika pada Micro Controller*", Universitas Kristen Petra, 2001

Vicky, H. Mack, "*On The Asymmetric Travelling Salesman Problem with Replenishment Arcs*", The University of Melbourne , Department of Mathematics and Statistics, 2001

Wall, Matthew, "*Galib*" [Online] Available http://lancet.mit.edu/ga/, diakses 28 Mei 2009

Widhiyasa, Arief, "*Kajian Genetic Algorithm Dalam Penyelesaian TSP*" [Online] Available http://www.informatika.org/~rinaldi/Matdis/2006-2007/Makalah/Makalah0607-119.pdf, diakses 20 Mei 2009