BAB II

LANDASAN TEORI

2.1 Konsep Dasar Pohon Rentang Minimum (*Minimum Spanning Tree*)

Suatu masalah rentang-minimum (*minimum spanning tree*) menyangkut suatu himpunan nodes dan suatu himpunan cabang yang diusulkan (*proposed*), yang satupun tidak ada yang terorientasi. Tiap-tiap cabang yang diusulkan memiliki suatu biaya tak-negatif yang berkaitan dengannya. Tujuannya adalah menyusun suatu jaringan tersambung yang mengandung semua nodes dan sedemikian rupa sehingga jumlah biaya yang berkaitan dengan cabang-cabang yang benar digunakan adalah minimum.

Minimum spanning tree merupakan variasi dari persoalan rute terpendek yang perbedaannya terletak pada lintasan yang dicari. Pada rute terpendek dicari lintasan/rute dari sumber ke tujuan yang memberi total jarak minimum, sedangkan pada *minimum spanning tree* ini yang dipersoalkan adalah menentukan busur-busur yang menghubungkan *nodes* yang ada pada pada jaringan, sehingga diperoleh panjang busur total minimum [EFE03].

Sedangkan persamaan pada kedua kasus ini adalah suatu jaringan tak terarah, dengan informasi yang mencakup nodes-nodes yang beserta jarak berupa biaya, waktu dan besaran lainnya antar nodes.

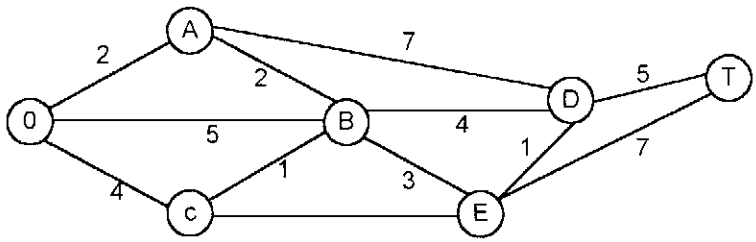
Contoh-contoh praktis dari persoalan spanning tree antara lain :

1. Perencanaan jaringan tranportasi. Dalam hal ini *nodes*nya merupakan terminal, sedangkan busur-busurnya dapat berupa jalan raya. Persoalan ialah menentukan pola transportasi yang dapat melayani seluruh terminal dengan jarak minimum.
2. Perencanaan jaringan komunikasi berskala besar.
3. Perencanaan jaringan distribusi.

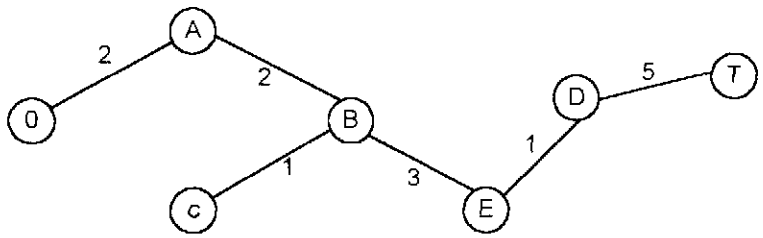
Persoalan *minimum spanning tree* ini dapat diselesaikan dengan cara mudah sebagai berikut :

1. Memilih sembarang salah satu *nodes*, kemudian menghubungkan *nodes* tersebut dengan *nodes* lain yang terdekat.
2. Menentukan *nodes* lain yang belum terhubung, jarak yang paling dekat dengan *nodes* yang sudah terhubung pada langkah sebelumnya, kemudian menghubungkan *nodes* ini.
3. Mengulangi langkah ini sehingga seluruh *nodes* dapat terhubungi.

Persoalan *minimum spanning tree* akan lebih mudah jika menggunakan gambar. Gambaran sederhana dari pengertian *minimum spanning tree* adalah sebagai berikut:



Gambar 2.1 Contoh persoalan *minimum spanning tree*



Gambar 2.2 Alternatif *tree* dari jaringan pada gambar 2.1

Total sisi-sisi yang membentuk *spanning tree*, dapat dirumuskan sebagai berikut :

$$F = \sum_{e_{ij} \in E} w_{ij} \qquad (2.1)$$

Keterangan :

F = total sisi i = indek node dari e = sisi

W = jarak j = indek node tujuan E = graf

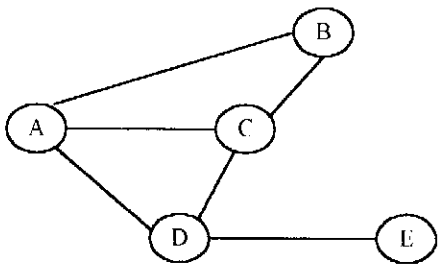
2.2 Konsep Jaringan

Model jaringan sering ditampilkan sebagai kerangka kerja dan struktur perhitungan untuk memperbaiki pendekatan tradisional menjadi analisis sistem. Dengan analisis jaringan dapat diselesaikan persoalan-persoalan untuk mendapatkan lintas terpendek atau terpanjang yang menghubungkan nodes awal dengan melalui beberapa nodes alternatif sampai nodes akhir.

Untuk memahami apa yang dimaksud dengan jaringan, berikut ini dijelaskan definisi, notasi, dan simbol yang digunakan. Sebuah jaringan (*network*) adalah suatu susunan garis edar (*path*) yang menghubungkan berbagai titik, dimana satu barang atau lebih bergerak dari satu titik ke titik lain.

Demikian juga pengertian jaringan menurut Hamdy A Taha adalah sebuah jaringan terdiri dari sekelompok node yang dihubungkan oleh busur atau cabang [TAH87]. Setiap orang akrab dengan berbagai jaringan seperti sistem jalan tol, jaringan telepon, jaringan rel kereta api, dan jaringan televisi.

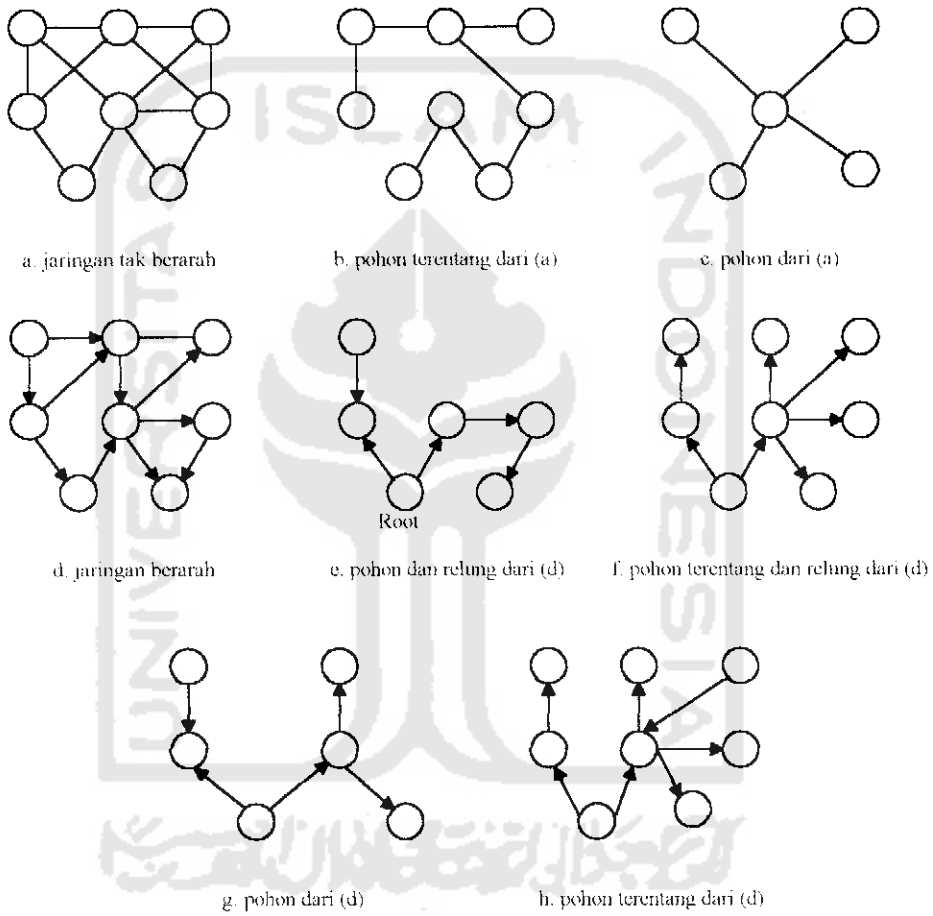
Jaringan diilustrasikan sebagai diagram yang terdiri dari dua komponen penting nodes dan cabang. Nodes melambangkan titik-titik persimpangan, sebagai contoh persimpangan beberapa jalan. Cabang menghubungkan nodes-nodes tersebut mencerminkan arus dari satu titik ke titik lain dalam jaringan tersebut. Nodes-nodes dalam diagram jaringan dilambangkan dengan lingkaran, dan cabang dilambangkan dengan garis yang menghubungkan nodes-nodes tersebut. Umumnya nodes-nodes itu melambangkan lokasi, seperti kota, persimpangan, atau stasiun udara atau kereta api, sedangkan cabang merupakan garis edar yang menghubungkan nodes-nodes tersebut.



Gambar 2.3 Jaringan terdiri dari 5 buah node

Tujuan dari jaringan adalah untuk menentukan jarak terpendek, waktu tersingkat, atau biaya terendah diantara titik-titik dalam jaringan.

Pengertian-pengertian lain yang diperoleh dalam suatu jaringan dapat dilihat pada gambar 2.4 berikut :



Gambar 2.4 Pohon dan Relung

Gambar ini memberikan ilustrasi tentang jaringan tak berarah, jaringan berarah, pohon, dan relung. Suatu pohon (*tree*) dari jaringan adalah suatu rangkaian terhubung yang tidak mengandung gelang. Dalam memandang suatu pohon, maka arah tidak mempengaruhi. Jadi, suatu rangkaian adalah suatu pohon jika dan hanya jika setiap nodes dihubungkan dengan nodes yang lain oleh satu

alur saja. Suatu pohon terentang (*spanning tree*) dari suatu rangkaian adalah suatu pohon yang merupakan rangkaian bagian dari suatu rangkaian.

Suatu pohon dapat diperoleh dengan cara menghilangkan beberapa kaitan tertentu. Bila dalam suatu pohon satu kaitan diputuskan maka akan terjadi rangkaian yang tak terhubungkan. Jadi suatu pohon terentang merupakan suatu rangkaian dengan jumlah hubungan yang minimum.

Pohon terentang sangat berguna dalam menentukan alur terpendek dalam suatu jaringan transportasi. Alur terpendek dari nodes asal ke semua nodes yang lain membentuk suatu pohon terentang. Nodes asal dari pohon dengan alur terpendek kadang-kadang disebut akar dari pohon tersebut. Bila arah dari kaitan-kaitan perlu diperhatikan, maka pohon yang terdiri dari rantai dari suatu nodes ke semua nodes yang lain disebut relung.

Notasi standar untuk menggambarkan sebuah jaringan G adalah G= (N,A), dimana N adalah himpunan node dan A adalah himpunan busur. Sebagai contoh dalam Gambar 2.5 jaringan yang terdiri dari lima node dan delapan busur yang dijabarkan dengan x.

N = {1, 2, 3, 4, 5}

A = {(1,3), (1,2), (2,3), (2,4), (2,5), (3,4), (3,5), (4,5)}

Gambar 2.5 Jaringan 5 node dengan 8 busur terhubung

Suatu jenis arus tertentu berkaitan dengan setiap jaringan (misalnya, arus produk minyak dalam sebuah jaringan pipa dan arus lalu lintas dalam jaringan transportasi). Pada umumnya, arus dalam sebuah busur dibatasi oleh kapasitasnya, yang dapat terbatas atau tidak terbatas [EFE03].

2.3 Algoritma Genetik

2.3.1 Deskripsi Algoritma Genetik

Algoritma Genetik merupakan salah satu algoritma pencarian terstruktur yang didasarkan pada anologi mekanisme seleksi dan informasi genetik alami atau merupakan suatu model komputasi yang diilhami oleh evolusi seleksi alamiah yang didasarkan pada teori genetik. Algoritma Genetik adalah suatu metode pembelajaran mesin, berdasarkan atas prinsip evolusi genetik dimana spesies yang mampu beradaptasi dengan lingkungannya akan dapat terus hidup, sedangkan yang tidak mampu akan tersingkir. Dalam penggunaannya, Algoritma Genetik meniru beberapa proses yang ditemukan pada proses evolusi ilmiah atau konsep Algoritma Genetik itu sendiri berhubungan erat dengan konsep genetik dan teori evolusi yang dikemukakan oleh Darwin yang memberikan kerangka kerja secara teoritis tentang adaptasi biologis yang merupakan perilaku yang diturunkan dari beberapa mekanisme evolusi alam seperti seleksi, rekombinasi, mutasi dan reproduksi. Aplikasi Algoritma Genetik sendiri telah meluas ke berbagai bidang seperti optimasi dan klasifikasi, pencarian strategi investasi atau pemrograman otomatis [MAN00]. Di Jerman ada Rechenberg dan Schewel yang tekniknya dikenal dengan nama strategi evolusi (*evolution strategies*). Conrad dan

koleganya mengembangkan apa yang dinamakan sebagai dinamika evolusioner (*evolutionary dynamics*). Dan masih banyak lagi yang mengembangkan teknik-teknik seperti diatas.

Algoritma Genetik pertama kali dikemukakan oleh John Holland dari Universitas Michigan yang memulai penelitiannya pada awal tahun 1960. Penelitian pertamanya yang dipublikasikan adalah *"adaption in natural and artificial systems"* pada tahun 1975. Menurut Holland *"Apabila evolusi dapat bekerja dengan sangat baik untuk organisme, mengapa tidak dapat digunakan untuk program komputer?"*. Penelitian Holland menyimpulkan dua hal yaitu untuk menjelaskan dan mempelajari proses adaptasi sistem alami, dan untuk mendesain/merancang sistem cerdas yang mempunyai persamaan atau mengandung mekanisme dengan sistem yang alami. Penerapan Algoritma Genetik terutama dikaitkan dengan metode adaptif untuk memecahkan masalah pencarian dan optimasi. Teori dasarnya ialah genetik bawaan dari populasi yang ada secara potensial memiliki solusi, atau solusi yang lebih baik, terhadap masalah yang akan dihadapi. Solusi ini belum aktif karena kombinasi genetik yang dialami terpecah dalam beberapa subjek [GEN97].

Evolusi terjadi pada kromosom, alat organik yang mengkodekan struktur makhluk hidup. Makhluk hidup sebagian terbentuk melalui proses pengkodean kromosom. Individu-individu yang ada pada saat tertentu dalam suatu populasi merupakan individu-individu yang kuat yang berhasil bertahan hidup, sedangkan individu-individu yang lemah akan punah. Individu-individu yang bertahan hidup

akan membentuk individu-individu baru sehingga terjadi proses regenerasi. Beberapa teori dasar evolusi yang diadopsi oleh Algoritma Genetik adalah :

1. Evolusi adalah proses yang bekerja pada kromosom bukan pada makhluk hidup yang dikodekannya.
2. Seleksi alamiah adalah hubungan antara kromosom dan kinerja dari struktur yang dikodekannya. Proses seleksi alamiah menyebabkan kromosom yang mengkodekan struktur yang sesuai lebih banyak direproduksi dibandingkan yang tidak sesuai.
3. Proses reproduksi adalah titik saat terjadi evolusi. Mutasi dapat menyebabkan kromosom anak berbeda dengan induknya dan rekombinasi akan menciptakan kromosom yang jauh berbeda dengan mengkombinasikan bagian kromosom dari kedua induknya
4. Evolusi biologis tidak mempunyai memori. Apapun yang berhubungan dengan fungsi individu dalam lingkungannya terdapat dalam gen (sekumpulan kromosom yang dibawa individu) dan struktur yang dikodekannya.

Teori dasar evolusi diatas, apabila secara sempurna digabungkan ke dalam sebuah pemrograman komputer, mampu menghasilkan teknik pencarian untuk memecahkan masalah yang sulit dengan cara yang sama dengan yang dilakukan oleh alam melalui evolusi.

Istilah istilah yang diadopsi oleh Algoritma Genetik dari istilah istilah yang terdapat dalam sistem biologi ialah :

1. Kromosom dengan *string*

   Kromosom merupakan tempat penyimpanan informasi genetik. Informasi yang disimpan merupakan representasi dari masalah yang dihadapi. Di dalam Algoritma Genetik, *string* dianalogikan sebagai kromosom

2. *Genotype* dengan struktur

   Kombinasi satu atau beberapa kromosom membentuk fungsi kerja suatu organisme. Interaksi sekumpulan disebut *Genotype*. Di dalam Algoritma Genetik, struktur dianalogikan dengan *Genotype*

3. *Phenotype* dengan set parameter (alternatif solusi)

   Interaksi di dalam struktur terjadi karena adanya proses transformasi kode - kode genetik. Proses ini disebut *Phenotype*. Jadi *Phenotype* merupakan representasi set parameter masalah yang dihadapi. Representasi kode dapat berupa numerik atau non numerik.

4. Gen dengan *feature* (karakter)

   Suatu kromosom dibentuk oleh beberapa gen. Di dalam Algoritma Genetik *string* dibentuk oleh beberapa *feature*.

5. *Alleles* denga *feature value*

   Suatu *feature* memiliki nilai tertentu. Di dalam sistem biologi nilai *feature* tersebut disebut *alleles*.

6. *Locus* dengan *positions*

   Letak gen di dalam kromosom disebut *locus*. Setiap *feature* memiliki urutan posisi di dalam *string*.

7. *Fitness Function* (fungsi sesuaian) dengan fungsi tujuan

   Setiap kromosom memiliki fungsi sesuaian berdasarkan nilai dan letak gen. Fungsi sesuaian merupakan fungsi tujuan yang digunakan dalam proses evaluasi permasalahan yang dihadapi.

Proses Algoritma Genetik dalam mencari suatu nilai optimasi (maksimum/minimum) adalah sebagai berikut :

1. Penentuan model dari sistem buatan dengan mendefinisikan spesies – spesies dengan struktur gen dan kromosom yang ditentukan berdasarkan sifat-sifatnya.
2. Perhitungan nilai *fitness* (nilai kebugaran) dari setiap spesies berdasarkan struktur gennya. Nilai ini digunakan sebagai patokan optimal tidaknya suatu nilai.
3. Pemilihan induk yaitu individu-individu dengan nilai *fitness* terbaik untuk dijadikan induk dalam menghasilkan individu-individu baru.
4. Proses reproduksi terdiri dari *CrossOver* (perkawinan silang) dan mutasi. Dari reproduksi ini akan dihasilkan individu-individu terbaru.

Dengan melakukan proses diatas secara berulang-ulang, diharapkan induk yang baik akan diperoleh generasi anak dengan spesies lebih baik.

Dengan demikian, algoritma genetik melakukan simulasi evolusi pada suatu populasi kromosom. Seperti evolusi, algoritma genetik menyelesaikan masalah pencarian kromosom yang baik, dengan memanipulasi bagian yang terdapat dalam kromosom secara acak (*random*). Secara abstraksi, di dalam genetik setiap organisme terdiri dari sel-sel. Setiap sel terdiri dari sejumlah

kromosom yang merupakan sederetan *DNA* sebagai pembentuk organisme secara keseluruhan. Setiap kromosom terdiri dari gen-gen yang merupakan blok *DNA* dan berperan menyandikan suatu karakteristik tertentu dari suatu makhluk hidup. Sebuah gen merupakan sekumpulan kemungkinan *alleles* (nilai yang mungkin ditempati suatu gen) yang masing-masing mengkodekan sebagai variasi dari karakteristik tertentu. Tiap gen mempunyai posisinya sendiri yang dinamakan *locus*. Algoritma Genetik memperlakukan informasi dari suatu model atau solusi sebagai serangkaian kode-kode yang berentetan. Pengkodean ini mirip dengan informasi genetik yang dikodekan oleh gen-gen pada suatu kromosom.

Kumpulan gen-gen tertentu dalam kromosom (*genome*) dinamakan *Genotype*. Hasil perkembangan lanjut dari *Genotype* merupakan *Phenotype* yang merupakan mesin ketahanan hidup dari evolusi. Evolusi alam merupakan proses seleksi terhadap entitas-entitas populasi di muka bumi berdasarkan tingkat ketahan hidup. Menurut Darwin, melalui seleksi alam ini, akan terdapat entitas yang bertahan hidup dan juga entitas yang mati berdasarkan tingkat ketahanan hidup masing-masing entitas. Jadi evolusi alam dapat dibayangkan sebagai suatu proses pencarian dalam cakupan sangat besar dari *Genotype* untuk menghasilkan struktur yang efisien dalam melaksanakan fungsi yang diperlukan untuk bertahan hidup.

Bila evolusi alam dipandang sebagai evolusi buatan maka dapat dikatakan sebagai masalah optimasi. Fungsi obyektif dari evolusi adalah untuk menemukan suatu kumpulan sifat yang memaksimalkan kesempatan bertahan hidup dan bereproduksi. Anggota populasi merepresentasikan kandidat solusi, sedangkan

masalahnya sendiri merupakan lingkungan. Tiap solusi diterapkan dalam masalah dan diberikan suatu nilai *fitness* (nilai kebugaran) yang menunjukkan kinerjanya dalam suatu masalah. *Fitness* dapat ditentukan dengan suatu fungsi tujuan atau dengan suatu penilaian subjektif. Operator-operator dalam algoritma genetik meliputi operasi seleksi, operasi *CrossOver* (rekombinasi), operasi mutasi, serta pembaharuan generasi. Masing-masing operator dapat diterapkan dengan berbagai cara yang berbeda, sehingga algoritma genetik yang akan dihasilkan juga bervariasi.

2.3.2 Penentuan Model Sistem

Untuk menyelesaikan suatu masalah dengan Algoritma Genetik, maka masalah tersebut harus dimodelkan terlebih dahulu melalui pengkodean yang dapat dipecahkan oleh metode komputasi. Jenis pengkodean tergantung pada permasalahannya, dapat berupa barisan bit, permutasi, himpunan, dsb. Pemodelan solusi merupakan proses konversi dari *Phenotype* menjadi *Genotype*. Suatu *fitness* yang sering berupa fungsi objektif masalah itu sendiri, digunakan sebagai penentu nilai *fitness* kandidat solusi (kromosom) yang sedang diproses, sehingga menentukan kelayakan bertahannya *genome* tersebut.

Sebelum melakukan perhitungan optimasi, ditentukan terlebih dahulu model sistem yang akan dibuat. Sistem buatan terdiri dari variabel-variabel yang sesuai dengan model alamiah, yaitu setiap satuan nilai dianggap sebagai satu individu spesies yang memiliki gen, kromosom dan nilai *fitness*. Individu-individu dalam suatu populasi dianalogikan dengan set-set solusi yang mungkin dari suatu

permasalahan optimasi. Gen adalah satu satuan unsur terkecil dari suatu nilai yang dapat berupa bit, simbol atau karakter tergantung dasar yang digunakan untuk merepresentasikan sebuah set solusi dalam permasalahan yang dihadapi. Kromosom adalah suatu struktur gen-gen yang membentuk suatu kesatuan yang mewakili suatu nilai atau karakteristik. Nilai *fitness* adalah suatu nilai dari spesies berdasarkan kromosomnya. Model dari algoritma genetik seperti halnya permasalahan optimasi di bagi menjadi dua bagian yaitu : fungsi tujuan (*objective function*) dan batasan-batasan (*constraints*).

2.3.3 Pembangkitan Generasi Awal

Pembangkitan generasi awal dari suatu populasi adalah dengan melakukan penyusunan gen secara acak sehingga membentuk beberapa kromosom dan membentuk satu individu.

1	2	3	4	6	5	7	8
0	1	2	1	4	4	4	7

Gambar 2.6 Kromosom awal dari suatu populasi

2.3.4 Perhitungan Nilai *Fitness*

Deretan kromosom dapat dievaluasi dengan suatu nilai *fitness* yaitu alat seleksi. Pehitungan nilai *fitness* untuk setiap individu mempunyai tujuan nutuk mengetahui nilai optimasi dari sebuah generasi. *Fitness* dapat ditentukan dengan suatu fungsi tujuan (nilai maksimal atau minimal suatu fungsi kedaaan) dan batasan-batasan nilai *fitness* yang diberikan kepada suatu model. Semakin kecil jumlah kesalahan yang terjadi maka semakin besar nilai *fitness*nya.

Persamaan *Nilai Fitness* dapat dirumuskan sebagai berikut [HID02] :

$$f(x) = \frac{1}{g(x) + (N * h(x) * 10^5)} \tag{2.2}$$

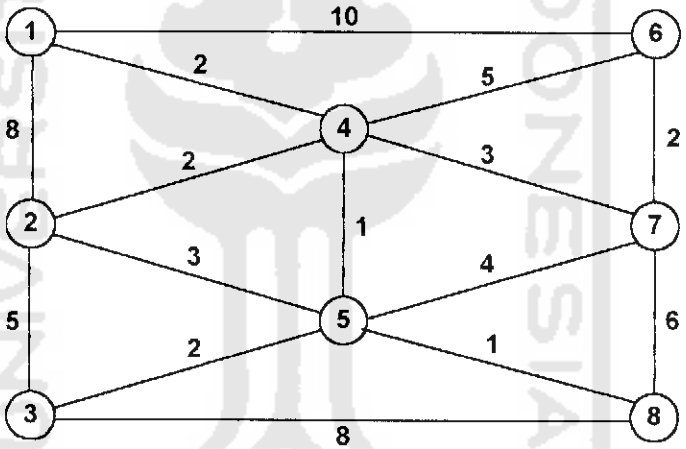
Dengan:

g(x) = fungsi untuk menghitung jarak

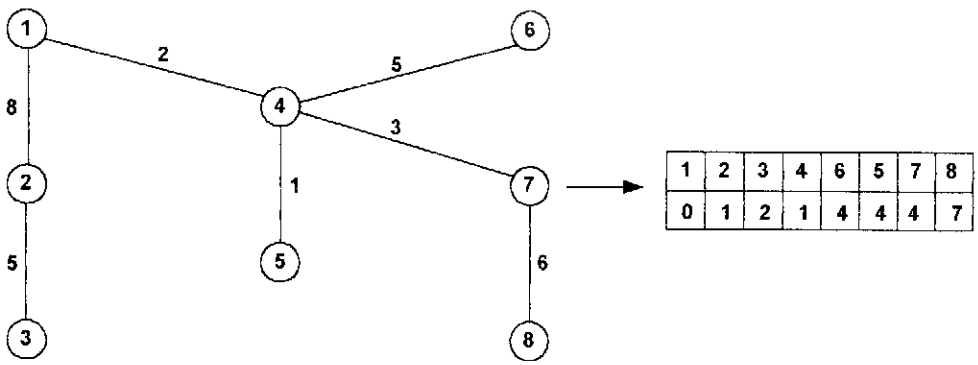
h(x) = fungsi untuk pencarian banyak busur kesalahan

N = banyaknya simpul (panjang kromosom)

Contoh 2.2 :



Gambar 2.7 Persoalan *minimum spanning tree*



Gambar 2.8 Kromosom dan *tree* tanpa busur kesalahan

Penyelesaian :

$g(x) = 8+5+2+5+1+3+6 = 30$

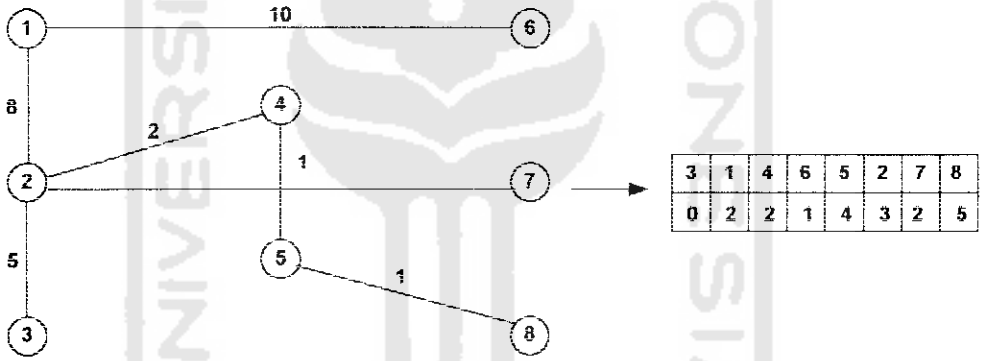
$N = 8$

$h(x) = 0$

$$f(x) = \frac{1}{g(x) + (N * h(x) * 10^5)}$$

$$= \frac{1}{30 + (8 * (0) * 100.000)}$$

$$= 0.03$$



Gambar 2.9 Kromosom dan *tree* dengan busur kesalahan 1

Penyelesaian :

$g(x) = 8+2+10+1+5+0+1 = 27$

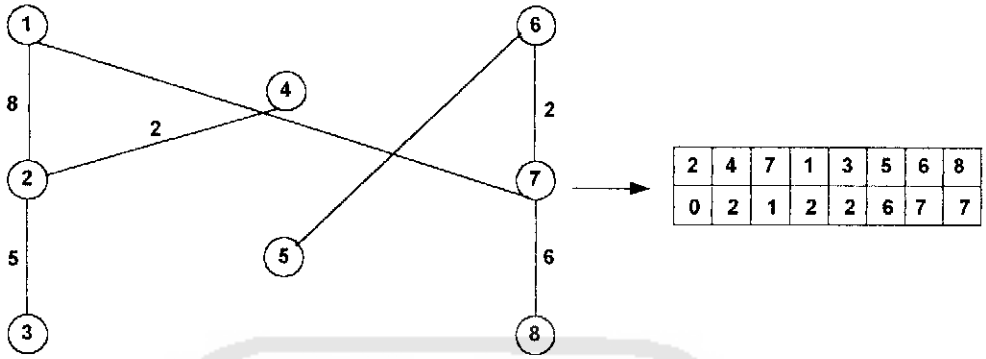
$N = 8$

$h(x) = 1$; busur ke 7 ke 2

$$f(x) = \frac{1}{g(x) + (N * h(x) * 10^5)}$$

$$= \frac{1}{27 + (8 * (1) * 10^5)}$$

$$= 1{,}25.10^{-6}$$



Gambar 2.10 Kromosom dan *tree* dengan busur kesalahan 2

Penyelesaian:

$g(x) = 2+0+8+5+0+2+6 = 23$

$N = 8$

$h(x) = 2$; busur ke 7 ke 1 & busur ke 5 ke 6

$$f(x) = \frac{1}{g(x)+(N*h(x)*10^5)}$$

$$= \frac{1}{23+(8*(2)*10^5)}$$

$$= 6{,}25.10^{-7}$$

2.3.5 Pemilihan Induk

Operasi seleksi digunakan untuk memilih kromosom dalam proses *CrossOver* nantinya. Jenis operasi seleksi yang biasa digunakan antara lain seleksi secara acak, seleksi roda roulette, seleksi ranking, dan seleksi turnamen. Masing masing seleksi menerapkan penekanan selektif, yaitu memilih kromosom berdasarkan nilai *fitness*nya, kecuali pada seleksi acak. Seleksi berguna untuk mengekploitasi ruang pencarian kandidat solusi. Seleksi disini adalah pemilihan-pemilihan individu dengan nilai kebugaran terbaik untuk dijadikan induk dalam menghasilkan individu-individu baru.

Pemilihan kromosom-kromosom terbaik itu dapat dilakukan dengan menggunakan mekanisme mesin *roulette* dimana semua kromosom ditempatkan dalam suatu lingkaran. Prosentase luas dari setiap kromosom pada lingkaran *roulette* didasarkan pada nilai *fitness*nya. Semakin besar nilai *fitness* yang dimiliki oleh sebuah kromosom, maka semakin besar pula peluang kromosom tersebut terpilih menjadi kromosom induk untuk generasi selanjutnya.

Gambar 2.11 Mekanisme Seleksi Roda Rolet (*Roulette Wheels)*

Kemudian mesin *roulette* diputar dimana jarumnya akan menunjuk bagian kromosom yang terpilih. Dalam gambar 2.11 terlihat bahwa jarumnya menunjuk pada kromosom 2, maka kromosom 2 terpilih menjadi kromosom induk dalam proses *CrossOver* dan mutasi.

2.3.6 *CrossOver* (Perkawinan Silang)

CrossOver adalah suatu proses penukaraan gen dari dua individu induk untuk memperoleh dua individu anak. Ini dilakukan dengan harapan agar diperoleh individu-individu anak dengan struktur gen yang lebih baik. Operasi *CrossOver* (perkawinan silang) mengkombinasi ulang gen-gen dari dua kromosom yang diperoleh melalui proses seleksi untuk memperoleh kromosom yang lebih baik.

Pada operasi *CrossOver* dibutuhkan dua induk untuk menghasilkan keturunan (*binary operator*). Untuk itu setiap *CrossOver* diperlukan dua induk dari hasil seleksi, jika hasil seleksi merupakan bilangan ganjil maka dibuang satu kromosomnya sehingga menjadi bilangan genap. Jenis *CrossOver* antara lain ialah *Partial Mapped CrossOver, Order CrossOver, Position Based CrossOver, Order Based CrossOver, Cycle CrossOver*. Dengan masing-masing penjelasan sebagai berikut [GEN97]:

1. *Partial Mapped CrossOver*

   Partial Mapped CrossOver (PMX) diperkenalkan oleh Goldberg dan Lingle. PMX menggunakan *procedure* khusus untuk mengatasi terjadinya *offspring* yang tidak legal dari persilangan dua titik. Inti dari PMX adalah

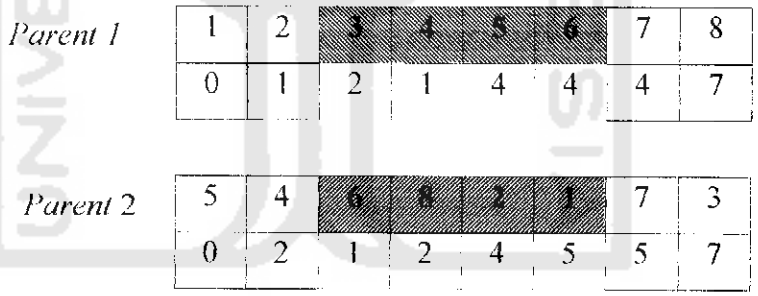
persilangan dua titik ditambah dengan prosedur perbaikan. Cara kerja PMX adalah sebagai berikut :

Langkah 1: Pilih dua posisi sepanjang *string* secara acak bebas. *Substring* yang didapat dari dua posisi yang telah dipilih disebut *mapping sections*.

Langkah 2 : Tukarkan posisi dari *substring* antara kedua *parent* untuk menghasilkan *proto-child*.

Langkah 3 : Analisis *mapping relationship* diantara dua bagian *mapping* tersebut.

Langkah 4 : Buat *offspring* menjadi legal jika dari langkah 2 ditemukan *offspring* yang tidak memenuhi kriteria legal (*illegitimate*)



Gambar 2.12 a Posisi awal kromosom sebelum operasi *CrossOver* dengan *PMX*

Silangkan posisi substring antara induk 1 dan 2

Proto-child	1	2	6	8	2	1	7	8
	0	1	2	1	4	4	4	7
Proto-child	5	4	3	4	5	6	7	3
	0	2	1	2	4	5	5	7

Gambar 2.12 b Proses penyilangan *substring* antara parent 1dan 2

Tentukan relasi *mapping*

$1 \leftrightarrow 6 \leftrightarrow 3$
$2 \leftrightarrow 5$
$8 \leftrightarrow 4$

Gambar 2.12 c Penentuan relasi *mapping*

Legal *offspring* dari relasi *mapping*

Offspring 1	3	5	6	8	2	1	7	4
	0	1	2	1	4	4	4	7
Offspring 2	2	8	3	4	5	6	7	1
	0	2	1	2	4	5	5	7

Gambar 2.12 d *Legal offspring*

2. *Order CrossOver*

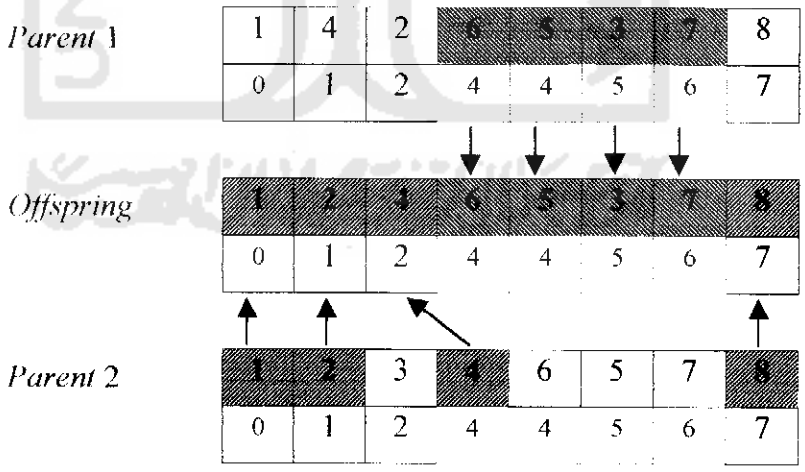
Order CrossOver dikenalkan oleh Davis, merupakan variasi dari *PMX* dengan perbedaan pada prosedur pembenahan *CrossOver*. Cara kerja *OX* adalah sebagai berikut :

Langkah 1 : Pilih *substring* secara bebas dari *parent*.

Langkah 2 : Copy *substring* ke *offspring* yang akan digenerasi dengan posisi yang sama.

Langkah 3 : Abaikan gen dengan nilai yang sama dengan yang sudah ada di langkah 2. Hasilkan urutan yang memiliki *substring* yang dibutuhkan oleh *protochild*.

Langkah 4 : Tempatkan sisa *substring parent* lainnya dengan posisi acak ke *protochild* dengan posisi dari kiri ke kanan menurut aturan dalam menggenerasi *offspring*.



Gambar 2.13 *CrossOver* dengan *Order CrossOver*

3. *Position Based CrossOver*

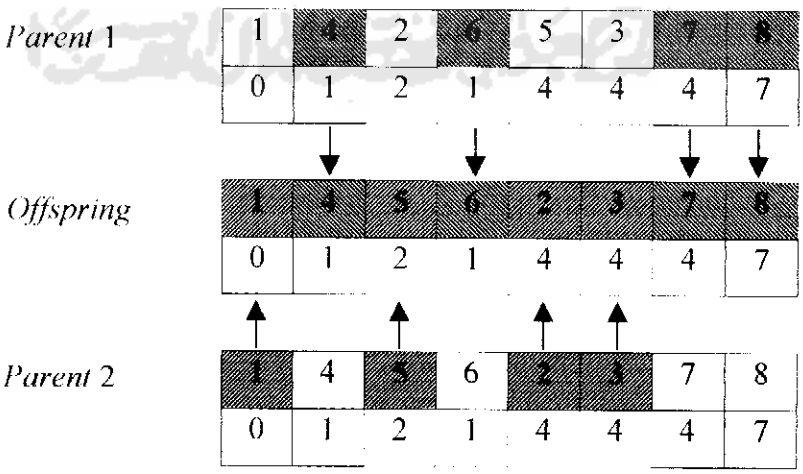
Position Based CrossOver dikenalkan oleh Syswerda, pada intinya sejenis representasi *CrossOver* pada umumnya dengan prosedur pembenahan. *Position Based* disebut juga variasi dari *OX* dengan pemilihan *substring* secara tidak urut. Cara kerja *Position Based CrossOver* adalah sebagai berikut:

Langkah 1 : Pilih satu set posisi *parent* secara acak.

Langkah 2 : Copy *substring* sesuai dengan posisi awalnya ke *protochild* sesuai dengan posisi awal *parent* asalnya.

Langkah 3 : Abaikan *offspring* dengan nilai yang sama dengan yang sudah ada di langkah 2. Hasilkan urutan yang memiliki *substring* yang dibutuhkan oleh *protochild*.

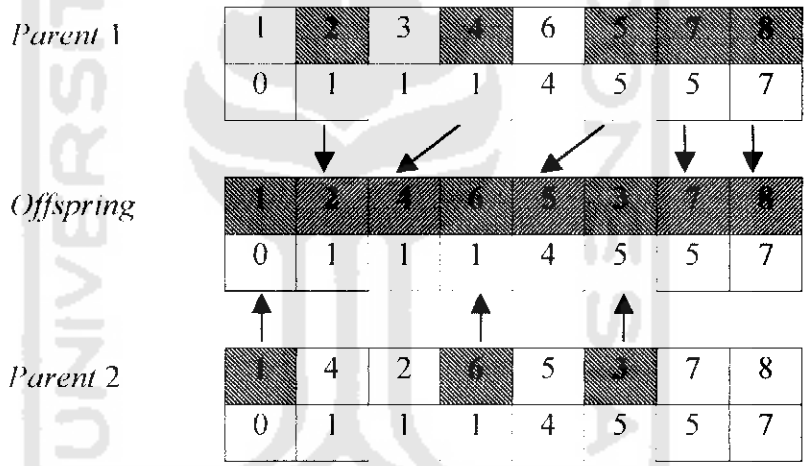
Langkah 4 : Tempatkan *substring* dengan posisi acak ke *protochild* dengan posisi dari kiri ke kanan menurut aturan dalam menggenerasi *offspring*.



Gambar 2.14 *CrossOver* dengan *Position Based CrossOver*

4. *Order Based CrossOver*

Order Based CrossOver dikenalkan oleh Syswerda, merupakan variasi dari *position based CrossOver*, perbedaanya terletak pada penempatan *substring* dari *parent* yang pertama yang ditempatkan sesuai order atau *substring* yang belum ada di *protochild* dari posisi yang telah ditempati *substring* dari *parent* kedua. Cara kerja *Order Based CrossOver* adalah sebagai berikut :



Gambar 2.15 *CrossOver* dengan *Order Based CrossOver*

5. *Cycle CrossOver*

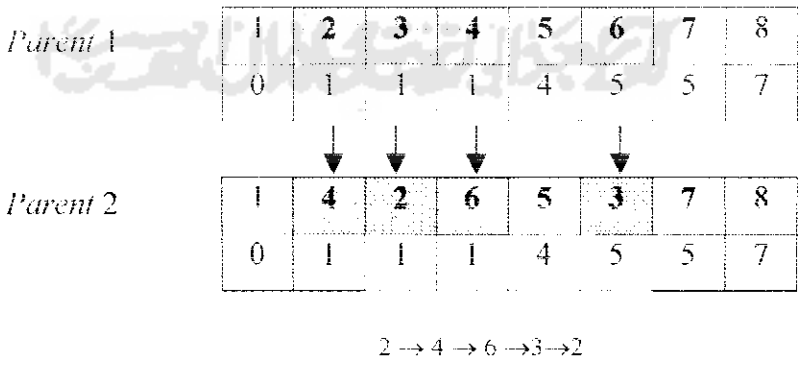
Cycle CrossOver (CX) dikenalkan oleh Oliver, Smith dan Holland. Seperti *position based CrossOver*, CX akan mengambil *substring* dari salah satu *Parent* dan memilih sisanya dari *Parent* lainnya. Perbedaannya adalah *substring* dari *Parent* pertama tidak dipilih secara acak dan urutan node pada *substring* yang dipilih mempunyai hubungan berantai diantara dua *Parent* tersebut. Cara kerja *Cycle CrossOver* adalah sebagai berikut :

Langkah 1 : Temukan hubungan antara posisi *substring* diantara kedua *Parent*.

Langkah 2 : Copy urutan gen yang sudah terdefinisi (saling terhubung) dalam langkah 1 ke dalam sebuah *child*.

Langkah 3 : Tentukan urutan *substring* sisa dari *substring* yang sudah ada di dalam rantai CX.

Langkah 4 : isi posisi *child* dengan sisa *substring*.



Gambar 2.16 a Proses pencarian mata rantai hubungan

Copy *substring* dari hubungan mata rantai ke *child*

Proto-child

	2	3	4		6		
0	1	1	1	4	5	5	7

Gambar 2.16 b Proses pembentukan awal *protochild*

Tentukan sisa urutan *substring* untuk child

Parent 2

1	4	2	6	5	3	7	8
0	1	1	1	4	5	5	7

Urutan *substring* yang tersisa

1	5	7	8

Gambar 2.16 c Proses pencarian sisa *protochild*

Offsspring

1	2	3	4	5	6	7	8
0	1	1	1	4	5	5	7

Gambar 2.16 d Hasil final operasi CX

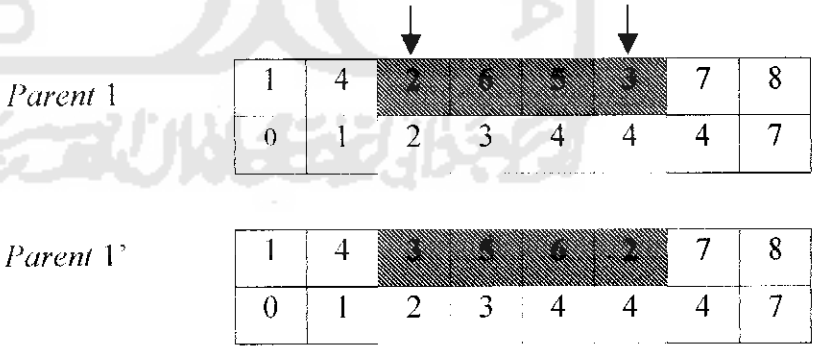
2.3.7 Mutasi (*Mutation*)

Operasi mutasi melakukan perubahan nilai gen-gen pada kromosom dengan cara menyisipkan kode informasi ke dalam suatu individu. Mutasi bertujuan untuk menjamin bahwa algoritma genetik melakukan ekplorasi pada ruang pencarian sehingga tidak terperangkap ke dalam suatu bagian tertentu. Cara melakukan mutasi tergantung pada jenis pengkodean kromosom.

Pada Mutasi ini diperlukan satu induk untuk menghasilkan satu anak (*unary operator*). Jenis mutasi antara lain *Inversion Mutation, Insertion Mutation, Displacement Mutation, Reciprocal Exchange Mutation.* Penjelasan masing-masing metode sebagai berikut :

1. *Inversion Mutation*

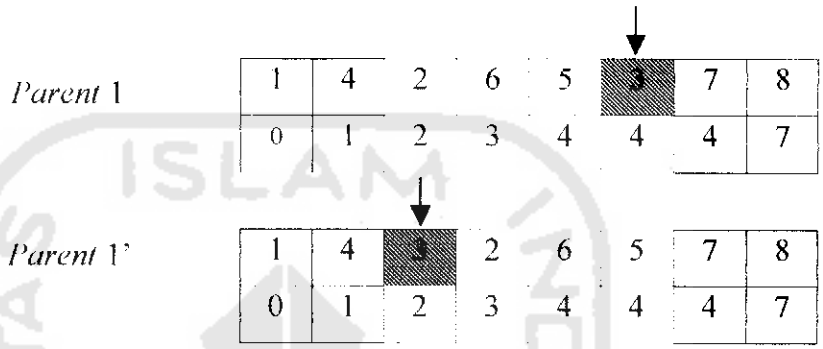
   Inversion Mutation memilih dua posisi gen secara acak dari kromosom secara acak dan membalikkan urutan diantara posisi yang telah terpilih sebelumnya.



Gambar 2.17 Operasi mutasi dengan *Inversion mutation*

2. *Insertion Mutation*

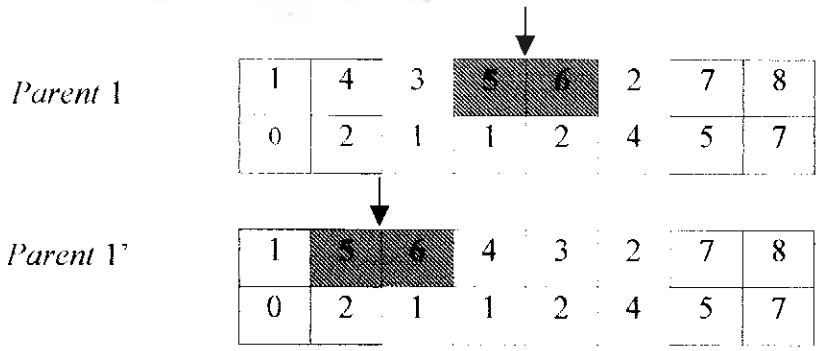
Insertion Mutation memilih nomor gen secara acak dan melakukan operasi insert secara acak pula ke kromosom tersebut.



Gambar 2.18 Operasi mutasi dengan *Insertion mutation*

3. *Displacement Mutation*

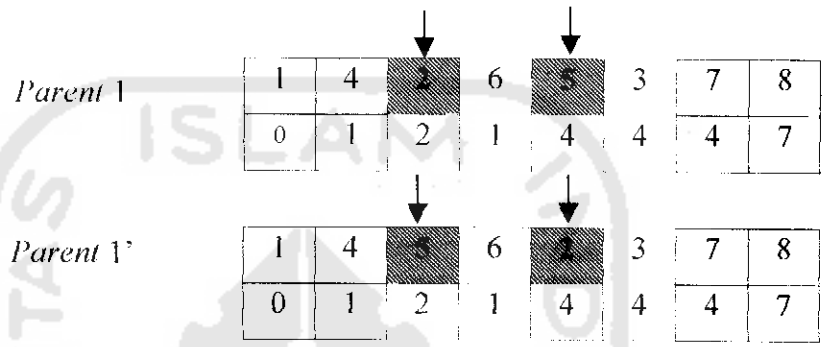
Displacement Mutation akan memilih *substring* secara acak dan melakukan *swapping* (menukar posisi) seperti dalam gambar berkut ini. *Insertion* dapat dikatakan sebagai kondisi tertentu di mana *displacement* hanya mempunyai satu gen.



Gambar 2.19 Operasi mutasi dengan *Displacement mutation*

4. *Reciprocal Mutation*

Reciprocal Mutation akan memilih dua posisi secara acak dan melakukan swapping (menukar posisi) seperti dalam gambar berikut ini.



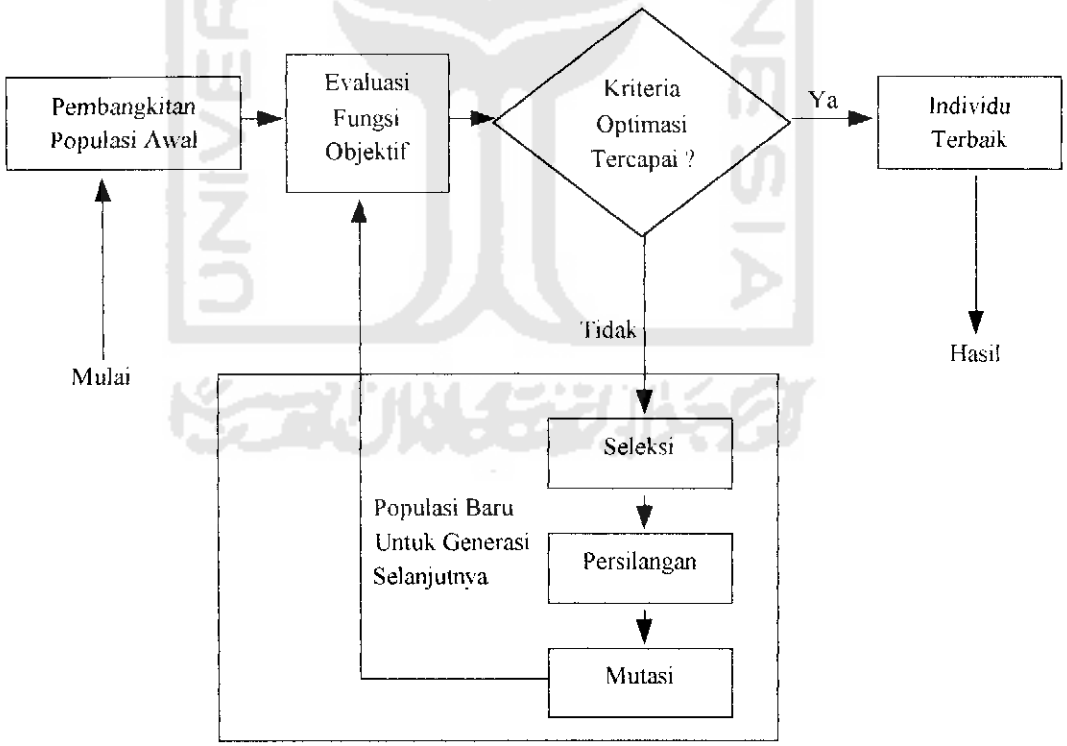
Gambar 2.20 Operasi mutasi dengan *Reciprocal mutation*

2.3.8 Pembaharuan Generasi

Proses pembaharuan generasi dimulai dengan operasi seleksi, *CrossOver* (perkawinan silang) dan mutasi (*mutation*), yang dilakukan untuk mengganti populasi lama dengan populasi baru, dengan harapan bahwa populasi baru tersebut akan mempunyai nilai *fitness* yang lebih baik daripada populasi lama. Pembaharuan tersebut dapat dilakukan dengan menggantikan secara keseluruhan populasi lama dengan populasi baru yang disebut *generational update*, atau menggantikan sebagian kromosom dalam populasi lama dengan kromosom-kromosom baru yang disebut *continous update*. Jika pendekatan selektif dilakukan cenderung mempertahankan kromosom yang baik dari generasi lama ke generasi baru dan hanya menggantikan kromosom yang nilai *fitness*nya kurang baik, maka jenis pembaharuan ini disebut *steady state update*.

2.3.9 Bagan Alir untuk Prosedur Algoritma Genetik

Pembangkitan generasi awal di mulai dengan pemodelan dan pengkodean. Setiap model atau solusi dikodekan sebagai suatu deretan node dengan panjang tertentu. Individu-individu kemudian dievaluasi oleh suatu fungsi tujuan yang akan menghitung nilai *fitness* dari setiap individu. Evaluasi di sini adalah untuk menentukan apakah kriteria optimasi telah dipenuhi, jika tidak maka dilakukan pembangkitan generasi baru melalui proses seleksi, *CrossOver*, dan mutasi. Generasi anak yang dihasilkan kemudian dievaluasi kembali berdasarkan kriteria optimasi. Langkah ini dilakukan berulang-ulang sampai diperoleh individu-individu terbaik atau generasi maximum telah tercapai.



Gambar 2.21 Struktur Algoritma Genetik