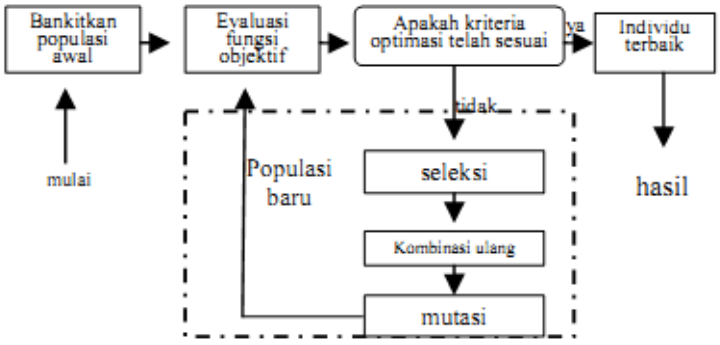
Author : Fauzi Yudhi S., Afiahayati

3.1 Algoritma Genetika

3.1.1 Pengertian Algoritma Genetika

Menurut Entin (2011) algoritma genetika yang ditemukan oleh John Holland dan dikembangkan oleh muridnya David Goldberg adalah cabang dari algoritma evolusi yang memiliki metode *adaptive* yang biasa digunakan untuk memecahkan suatu pencarian nilai dalam sebuah masalah optimasi di dunia nyata dan berbagai bidang. Algoritma ini bekerja dengan menduplikasi proses genetik yang terdapat pada makhluk hidup dimana perkembangan generasi dalam sebuah populasi yang alami secara lambat laun mengikuti prinsip seleksi alam. Algoritma ini bekerja dengan sebuah populasi yang terdiri dari individu-indiviu yang masing-masing individu mempresentasikan sebuah solusi yang mungkin bagi persoalan yang ada. Pertahanan yang tinggi dari individu memberikan kesempatan untuk melakukan reproduksi melalui perkawinan silang dengan individu yang lain dalam populasi tersebut dan menghasilkan invidu yang membawa beberapa sifat dari induknya. Sedangkan individu yang tidak dapat bertahan akan mati dengan sendirinya. Oleh karena itu, hanya individu-individu yang bagus dan kuat saja yang akan tetap tinggal dan bertahan. Siklus algoritma genetika yang pada awalnya David Goldberg membuatnya tanpa menggunakan langkah *elitism*, kemudian telah diperbarui oleh Michalewicz dengan menggunakan proses *elitism*. Siklus algoritma genetika dapat dilihat pada gambar 3.1.

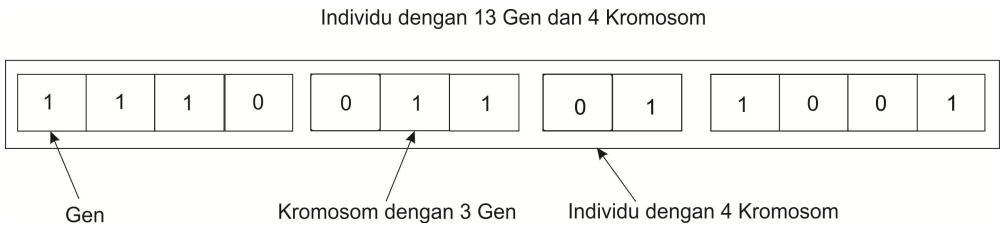


Gambar 3.1 Siklus algoritma genetika

3.1.2 Representasi Kromosom

Gen adalah sebuah nilai yang menyatakan satuan dasar yang membentuk suatu arti tertentu. Nilai dari gen tersebut dinamakan dengan *allele*. Kromosom adalah gabungan dari gen-gen yang membentuk nilai tertentu. Sedangkan individu adalah satu nilai atau keadaan yang menyatakan salah satu solusi yang mungkin dari sebuah permasalahan. Sekumpulan individu yang diproses bersama dalam satu siklus proses evolusi dinamakan dengan populasi dimana satu satuan siklus proses evolusi tersebut dinamakan dengan generasi (Basuki, 2003).

Setiap individu memiliki beberapa kromosom. Representasi bentuk dari kromosom itu dapat diwakilkan oleh sejumlah gen. Contoh representasi kromosom ditunjukkan pada gambar 3.2.



Gambar 3.2 Contoh representasi bentuk individu pada algoritma genetika

Representasi kromosom yang terdapat pada algoritma genetika ada beberapa jenis antara lain sebagai berikut :

1. Representasi Biner

   Setiap gen hanya bisa bernilai 0 atau 1. Memberikan banyak kemungkinan untuk kromosom, walaupun dengan jumlah *allele* yang sedikit. Namun banyak masalah yang tidak sesuai bila menggunakan model biner. Masalah yang cocok untuk penggunaan representasi biner adalah *knapsack problem*. Pengertian dari bentuk *knapsack problem* adalah terdapat suatu tas atau karung yang digunakan untuk memasukkan sesuatu. Tapi tidak semua barang bisa ditampung kedalam karung tersebut. Karung tersebut hanya dapat menyimpan beberapa objek sesuai dengan ukurannya. Contoh representasi secara biner ditunjukkan pada gambar 3.3.

0	1	0	0	1

Gambar 3.3 Representasi kromosom secara Biner

2. Representasi Integer

Setiap gen dapat bernilai bilangan bulat. Bilangan bulat digunakan untuk merepresentasikan nomor urut, posisi, atau kualitas objek. Bilangan bulat memiliki nilai angka dari 0,1,2 hingga 9 sehingga memiliki jumlah nilai *allele* yang lebih banyak bila dibandingkan dengan representasi biner. Contoh representasi kromosom integer ditunjukkan pada gambar 3.4.

1	0	4	7	8	4

Gambar 3.4 Representasi kromosom secara Integer

3. Representasi Real

Setiap gen pada kromosom bernilai bilangan real. Sehingga penyebaran data pada nilai *allele* lebih detail yaitu dapat mencakup nilai antara 0 hingga 1. Representasi real cocok untuk permodelan yang membutuhkan penghitungan secara detail. Contoh representasi kromosom secara real ditunjukkan pada gambar 3.5.

1,3446	0,1257	2,5413	3,8967	4,8476	2,9675

Gambar 3.5 Representasi kromosom secara Real

4. Representasi Permutasi

Representasi permutasi digunakan untuk merepresentasikan kromosom yang memperhatikan posisi/urutan gen atau nilai gen. Representasi seperti ini dapat digunakan dalam masalah pengurutan *(ordering problem)* atau untu kasus *Travelling Salesman Problem* (TSP). Contoh representasi secara permutasi ditunjukkan pada gambar 3.6.

1	4	3	2	6	5

Gambar 3.6 Representasi kromosom untuk Permutasi

3.1.3 Nilai *Fitness*

Nilai fitness adalah nilai yang menyatakan seberapa baik suatu solusi (individu). Nilai *fitness* ini dijadikan acuan dalam mencapai nilai optimal dalam algoritma genetika. Dengan *nilai fitness*, suatu individu dievaluasi berdasarkan suatu fungsi tertentu sebagai ukuran nilai kualitas. Di dalam kasus optimasi, ada dua masalah yaitu maksimasi yang bertujuan untuk mencari nilai maksimal dan minimasi bertujuan untuk mencari nilai minimal dari sesuatu (Tamajati, 2013).

Pengukuran nilai *fitness* pada setiap individu didapatkan dengan melakukan evaluasi terhadap nilai kromosom pada masing-masing individu tersebut. Pengukuran nilai *fitness* untuk setiap permasalahan bisa berbeda-beda tergantung bentuk dari permasalahan yang ada dan tatacara pengukuran yang dapat dilakukan pada permasalahan tersebut. Misalkan untuk kasus permasalahan yang representasi kromosomnya berupa rute-rute jalur minimum dalam kasus transportasi barang. Maka evaluasi nilai *fitness* dapat dihitung menggunakan jarak minimum yang didapatkan dari rute-rute yang ditunjukkan oleh representasi kromosom pada individu tersebut. Kesimpulannya adalah untuk bisa mendapatkan evaluasi nilai *fitness*, kita harus bisa memahami permasalahan yang dimodelkan oleh algoritma genetika dan mencari bentuk pengevaluasian yang paling sesuai untuk model tersebut.

3.1.4 Operator Algoritma Genetika

Coley (1999) menyebutkan bahwa algoritma genetika menggunakan tiga operator yaitu seleksi, *cross over* dan mutasi.

1. Seleksi

Seleksi bertujuan untuk mencari individu/kromosom dengan nilai fitness yang lebih baik untuk dilakukan perkawinan silang.

Pada algoritma genetika, terdapat 3 metode seleksi (Suyanto, 2008) yaitu :

a. Fitness Proportionate Selection (FPS)

1. Roulette Wheel

Pada metode *Roulette Wheel*, individu-individu dipetakan dalam suatu segmen garis secara berurutan. Setiap individu dipetakan menurut nilai *fitness*nya. Jika f_i merupakan nilai *fitness* untuk individu dalam suatu i populasi. Maka probabilitas individu itu terpilih dapat dirumuskan dalam persamaan 3.1.

$$p_i = \frac{f_i}{\sum_{j=1}^{N} f_j} \qquad \textbf{(3.1)}$$

Keterangan :

f_i = nilai *fitness* individu i

N = jumlah total individu dalam populasi tersebut

2. Baker's SUS

Pada metode ini, individu dipetakan dalam satu segmen garis secara berurutan sedemikian sehingga tiap-tiap segmen individu memiliki ukuran yang sama dengan ukuran *fitness*nya seperti halnya dengan metode Roda Roulette. Kemudian diberikan sejumlah *pointer* sebanyak individu yang ingin diseleksi pada garis tersebut. Andaikan N adalah jumlah individu yang akan diseleksi, maka jarak antar *pointer* adalah 1/N dan posisi *pointer* petama diberikan secara acak pada range [0,1/N]. Langkah proses dari seleksi *Baker's SUS* ini adalah sebagai berikut :

i. Menghitung jumlah individu yang ada dan mencari nilai 1/N.
ii. Menentukan bilangan random dari 0-1 sebanyak maksimal jumlah dari putaran.
iii. Membentuk garis bilangan dimana segmen yang terbentuk berasal dari 1/N yang ditambahkan terus menerus hingga mendekati 1.
iv. Mencari daerah mana saja yang masuk ke seleksi dan individu mana saja yang masuk ke seleksi.

b. Rank-Based Selection

1. Linear Ranking

Langkah pengerjaan dari seleksi *linear ranking* adalah semua individu yang berada dalam populasi diurutkan berdasarkan nilai fitnessnya secara *ascending*. Nilai fitness hasil perankingan dihitung menggunakan rumus pada persamaan 3.2.

$$f'(Pos) = (2 - S) + 2(S - 1)\frac{(Pos-1)}{(S-1)} \qquad \textbf{(3.2)}$$

S adalah *selective pressure* (probabilitas terpilih individu terbaik dibandingkan dengan rata-rata probabilitas terpilih semua individu). S berada dalam interval [1,2]. Sedangkan Pos adalah posisi individu dalam populasi, dimana individu terburuk yang nilai *fitness*nya paling rendah berada di posisi 1 dan individu terbaik di posisi N.

2. Non-Linear Rangking

Rumus untuk penghitungan seleksi *Non-Linear Rangking* ditunjukkan oleh persamaan 3.3.

$$f'(Pos) = \frac{NX^{Pos-1}}{\sum_{i=1}^{N} X^{i-1}} \qquad \textbf{(3.3)}$$

Dimana X adalah akar polinomial dari persamaan 3.4.

$$0 = (S-N)X^{N-1} + SX^{N-2} + \ldots + SX + S \qquad \textbf{(3.4)}$$

N = ukuran populasi

S = *Selective pressure* dalam interval [1,N-2]

c. Tournament Selection

Pada metode ini, langkah pertama pengerjaannya adalah dengan menetapkan suatu nilai tour untuk individu-individu yang dipilih secara random dari suatu populasi. Individu-individu yang terbaik dalam kelompok ini akan diseleksi sebagai induk. Parameter yang digunakan pada metode ini adalah ukuran tour untuk masing-masing kelompok yang bernilai antara 2 sampai N (jumlah individu dalam suatu populasi). Sehingga untuk setiap ukuran tersebut akan dilakukan proses seleksi berdasarkan aturan tertentu. Banyak cara yang bisa dilakukan untuk memilih satu pemenang (orangtua) dari setiap kelompok tersebut.

Ada empat hal yang bisa dijadikan acuan untuk membangun prosedur penentuan pemenang turnamen, yaitu :

i. Ada atau tidaknya proses perankingan.
ii. Penentuan ukuran sampling dari tiap-tiap kelompok.
iii. Kromosom yang sudah pernah terpilih sebagai kontestan bisa terpilih lagi atau tidak.
iv. Kontestan terbaik (dengan *fitness* tertinggi) akankah selalu menjadi pemenang (deterministik) atau bergantung pada suatu probabilistik tertentu yang telah ditentukan sebelumnya.

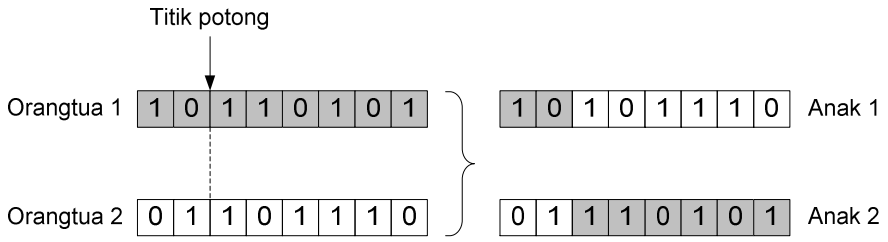
2. *Crossover*

Crossover adalah proses mengambil sebagian atau potongan barisan *allele* dari *dna genome* untuk disatukan menjadi *genome* keturunan atau *offspring*. Macam-macam dari *crossover* tergantung dengan representasi kromosom yang dilakukan.

a). *Crossover* untuk representasi Biner

2. *Crossover* satu titik (*1-point crossover*)

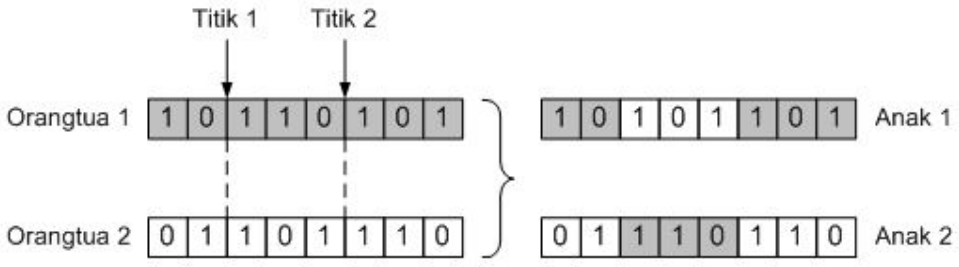
Crossover jenis ini akan menukar nilai gen dari suatu kromosom setelah titik *point* tertentu. Contoh *crossover* 1 titik ditunjukkan pada gambar 3.7.



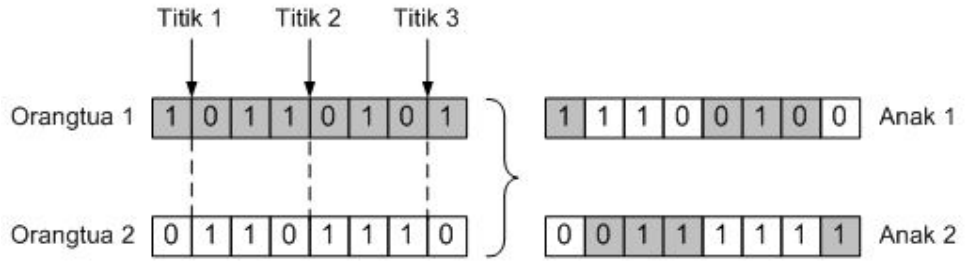
Gambar 3.7 Contoh *crossover* 1 titik

3. *Crossover* banyak titik (*Multipoint crossover*)

Crossover jenis ini akan menukar nilai gen dari suatu kromosom setelah titik *point* tertentu. Kemudian bila mendapati titik lain, bagian kromosom selanjutnya tidak akan tertukar. Bila mendapati titik lagi, maka bagian selanjutnya akan ditukar. Begitu seterusnya berselang-seling tergantung jumlah titik yang diberikan. Contoh *crossover* 2 titik ditunjukkan pada gambar 3.8 dan untuk 3 titik pada gambar 3.9.



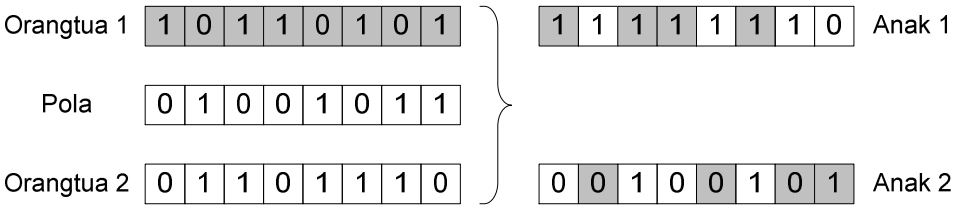
Gambar 3.8 Contoh *crossover* 2 titik



Gambar 3.9 Contoh *crossover* 3 titik

4. *Crossover* seragam (*Uniform crossover*)

Pada *crossover* seragam dilakukan penukaran sesuai dengan pola yang telah ditentukan. Jika menemui pola 1 maka gen antar kedua orang tua ditukar dan jika menemui 0 maka gen yang diambil tetap. Contoh *crossover* secara seragam ditunjukkan pada gambar 3.10.



Gambar 3.10 Contoh *crossover* seragam

b). *Crossover* untuk representasi Integer

Pada intinya *crossover* untuk representasi integer sama dengan *crossover* representasi biner. Yang berbeda hanya terletak pada nilai *allele* pada tiap-tiap gennya untuk representasi kromosom dimana untuk biner nilainya antara 0 atau 1 dan untuk integer memiliki rentang nilai antara 0 sampai 9. Sehingga pada representasi integer, juga memiliki 3 jenis *crossover* yang sama seperti *crossover* pada representasi biner yaitu *crossover* satu titik, banyak titik, dan seragam.

c). *Crossover* untuk representasi Real

Representasi untuk bilangan real setiap gen bisa memiliki nilai yang sangat bervariasi karena rentangnya sangat banyak dapat mencapai bilangan desimal antara 0 hingga 1. Sehingga secara umum untuk melakukan *crossover* terdapat 2 cara yaitu sebagai berikut :

1. *Crossover Discrete*

Setiap gen pada anak z berasal dari salah satu orangtuanya (x,y) dengan probabilitas yang sama, $z_i = x_i$ or y_i. Cara pemilihan posisi gen bisa menggunakan *crossover* banyak titik atau *crossover* seragam.

2. Crossover *Intermediate*

Memanfaatkan ide pembangunan anak yang berupa kromosom "antara" dari kedua orangtuanya. Oleh karena itu, *crossover* jenis ini disebut juga *arithmetic crossover*. Setiap gen pada anak z diperoleh berdasarkan persamaan 3.5.

$$z_i = \alpha x_i + (1 - \alpha) y_i \qquad \textbf{(3.5)}$$

Dimana $0 \leq \alpha \leq 1$. Parameter α bisa dibuat konstan *(uniform arithmetical crossover)*, variabel (misalnya, bergantung pada usia populasi), atau ditentukan secara acak pada setiap saat.

Terdapat tiga model *arithmetic crossover*, yaitu :

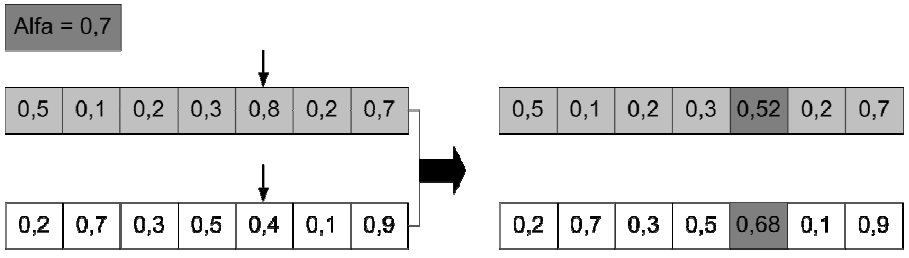
a. single arithmetic crossover

Kromosom orang tua ini dinyatakan sebagai $<x_1,...,x_n>$ dan $<y_1,....,y_n>$.. Selanjutnya memilih satu gen secara acak yang dimisalkan dengan k. Kedua anak dapat dihasilkan dengan menggunakan persamaan 3.6 dan 3.7.

$$\text{Anak 1} : \langle x_1,..., x_{k-1}, \alpha y_k + (1-\alpha) x_k,..., x_n \rangle \qquad \textbf{(3.6)}$$

$$\text{Anak 2} : \langle y_1,..., y_{k-1}, \alpha x_k + (1-\alpha) y_k,..., y_n \rangle \qquad \textbf{(3.7)}$$

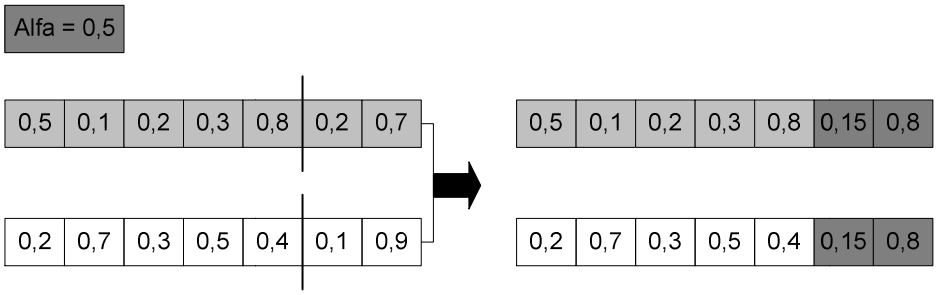
Misalkan sebagai contoh dengan nilai α = 0,7 . Maka untuk mendapatkan nilai gen pada kromosom orang tua x dan kromosom orang tua y yang baru. Maka ilustrasi penghitungannya dapat dilihat pada gambar 3.11.



Gambar 3.11 Ilustrasi kasus *single arithmetic crossover*

b. simple arithmetic crossover

Kromosom orang tua ini dinyatakan sebagai <x_1,...,x_n> dan <y_1,....,y_n>.. Selanjutnya memilih satu gen secara acak yang dimisalkan dengan k. Nilai gen yang mengalami penghitungan secara *arithmetic crossover* merupakan nilai gen yang dimulai dari titik k, hingga nilai gen terakhir secara keseluruhan. Jadi penentuan nilai k menjadi titik dimulainya *arithmetic crossover*. Untuk penghitungannya sama dengan bentuk *single arithmetic crossover*. Ilustrasi untuk kasus jenis *simple arithmetic crossover* dapar dilihat pada gambar 3.12.



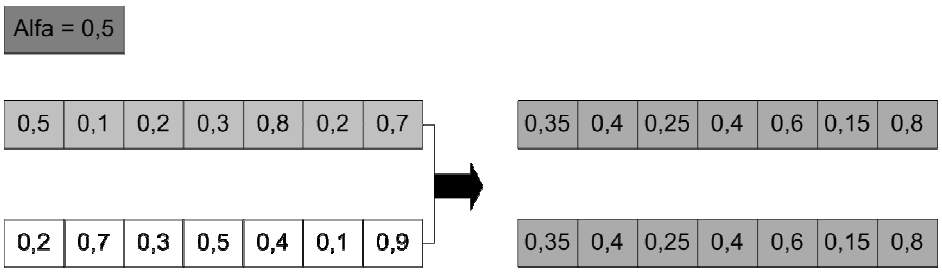
Gambar 3.12 Ilustrasi kasus *simple arithmetic crossover*

c. whole arithmetic crossover

Untuk jenis *whole arithmetic crossover* maka semua gen yang ada pada kromosom orang tua akan dilakukan proses penghitungan secara aritmatika. Gambar 3.13 menunjukkan ilustrasi untuk kasus *whole arithmetic crossover*. Untuk mendapatkan anak 1 dan anak 2 dapat menggunakan persamaan 3.8 dan 3.9.

$$Anak_1 = \sum_{i=1}^{n} \alpha . x_n + (1 - \alpha) . y_n \quad \textbf{(3.8)}$$

$$Anak_2 = \sum_{i=1}^{n} \alpha . y_n + (1 - \alpha) . x_n \quad \textbf{(3.9)}$$



Gambar 3.13 Ilustrasi kasus *whole arithmetic crossover*

d). *Crossover* untuk representasi Permutasi

Representasi permutasi digunakan untuk masalah-masalah permutasi yang mengandalkan posisi atau urutan gen sehingga cross over untuk representasi biner, integer maupun real tidak bisa digunakan untuk masalah-masalah permutasi karena mungkin saja ada dua gen yang bernilai sama dalam satu kromosom.

1. Order crossover

Langkah-langkah dalam melakukan *order crossover* adalah :

i. Memilih segmen kromosom dari kedua orangtua secara acak dengan cara membangkitkan dua titik potong, TP1 dan TP2.
ii. Menyalin bagian ini secara searah ke kedua anaknya dimana segmen kromosom orangtua 1 disalin ke anak 1 dan segmen kromosom orang tua 2 disalin ke anak 2.
iii. Menyalin gen-gen orangtua 2, yang tidak ada di anak 1, ke anak 1 dengan aturan mulai dari posisi setelah TP2, memperhatikan urutan yang ada pada orangtua 2 dan kembali ke posisi awal setelah akhir kromosom (*wrapping*).
iv. Melakukan langkah 3 dengan cara yang sama terhadap orang tua 1 untuk menghasilkan anak 2.

2. Partially mapped crossover

Langkah-langkah dalam melakukan *partially mapped crossover* adalah :

i. Memilih segmen kromosom dari kedua orangtua secara acak dengan cara membangkitkan dua titik, TP1 dan TP2.
ii. Menyalin segmen orangtua 1 ke anak 1.
iii. Mulai dari posisi TP1 melakukan pemetaan gen-gen yang ada di segmen orangtua 2 tetapi tidak ada di segmen orangtua 1.
iv. Mewariskan setiap gen tersebut ke anak 1 pada posisi hasil pemetaan.
v. Setelah semua gen di dalam segmen sudah diwariskan ke anak 1, maka posisi-posisi gen anak 1 yang masih kosong diisi dengan gen-gen orangtua 2 pada posisi-posisi yang bersesuaian.
vi. Melakukan hal sama pada untuk membangkitkan anak 2.

3. Cycle crossover

Langkah-langkah dalam melakukan *cycle crossover* adalah :

i. Mencari cycle pada orangtua 1 dengan cara :
   - Mulai dari posisi pertama orangtua 1 yang belum diwariskan.
   - Membuat panah ke posisi yang sama pada orangtua 2.
   - Membuat panah ke posisi gen yang bernilai sama pada orangtua 1.
   - Menambahkan gen ini ke cycle.
   - Mengulangi langkah i sampai panah kembali ke posisi awal.

ii. Mewariskan gen-gen orangtua yang berada pada cycle ini kepada kedua anaknya sesuai dengan posisinya dengan cara menyilang, searah, menyilang, searah, dan seterusnya.

4. Edge recombination

Langkah-langkah dalam melakukan *edge recombination* adalah :

i. Memilih elemen awal secara acak dari semua elemen yang ada. Kemudian memasukkan elemen terpilih ke hasil.

ii. Set elemen terpilih sebagai *current element*.

iii. Menghapus *current element* dari semua daftar *edge* yang ada di tabel.

iv. Memilih satu elemen dari daftar *edge* yang ada di current element dengan aturan :

   - Jika ada elemen yang merupakan *common edge* (*edge* berada pada kedua orang tua), maka memilih elemen tersebut.

   - Jika tidak ada *common edge*, memilih elemen yang memiliki daftar *edge* terpendek (jumlah *edge* yang paling sedikit).

   - Jika semua elemen memiliki jumlah *edge* yang sama, maka memilih elemen secara acak.

v. Jika daftar *edge* yang ada di current element sudah kosong, maka :

   - Mengubah pemilihan elemen terakhir untuk mencari elemen lain yang memiliki daftar edge tidak kosong.

   - Jika perubahan elemen tidak berhasil, memilih elemen baru secara acak.

e). Crossover i Path Relinking

Rekombinasi dilakukan dengan membuat banyak anak yang memiliki perbedaan secara berurutan sehingga mirip suatu jalur. Kemudian memilih sejumlah anak yang memiliki fitness tertinggi.

Membuat anak-anak yang memiliki perbedaan secara berurutan dapat dilakukan dengan langkah berikut :

i. membuat Anak 1 yang sebagian besar gen-nya (misal 90%) diambil dari Orang tua 1 dan sebagian kecil gen lainnya (10%) diambil dari Orang tua 2.

ii. membuat Anak 2 dengan porsi gen yang sedikit berbeda, misalnya 80% dari Orang tua 1 dan 20% dari Orang tua 2.

Dengan demikian, bisa dikatakan Anak 1 mirip dengan Orang tua 1. Anak 2 mirip dengan Anak 1 dan seterusnya sehingga Anak ke-n mirip dengan Orang tua 2.

f). *Crossover Multi Parent*

Perkawinan silang ini dilakukan antara lebih dari dua kromosom orang tua dalam proses cross over akan menghasilkan hasil yang lebih baik. Pendekatan yang bisa digunakan yaitu :

1. Berdasarkan frekuensi *allele*

   Pendekatan ini dilakukan pembangkitan pola yang menyerupai pada cross over seragam dengan kemungkinan nilai sebanyak jumlah orangtuanya. Jika pada *uniform crossover* dilakukan pembangkitan pola dengan dua kemungkinan nilai (berdasarkan pelemparan koin), maka pada pendekatan ini pembangkitan pola dilakukan dengan kemungkinan nilai sebanyak jumlah orangtuanya. Jika orangtuanya berjumlah tiga, maka pembangkitan pola dilakukan untuk menghasilkan tiga nilai berbeda.

2. Berdasarkan segmentasi dan rekombinasi

   Pendekatan ini dilakukan dengan mengimplementasikan cross over banyak titik yang pewarisan gennya dilakukan secara diagonal.

3. Berdasarkan operasi-operasi numerik pada *allele* bernilai real

   Pendekatan ini dilakukan dengan mengimplementasikan *arithmetic cross over.*

3. *Mutasi*

Mutasi adalah proses mengubah nilai dari satu atau beberapa gen dalam suatu genome yang bertujuan untuk menghindari konvergensi prematur yaitu pencapain suatu nilai atau hasil yang belum atau bukan maksimal (Suyanto, 2008). Terdapat 4 jenis mutasi di algoritma genetika yaitu :

a). Mutasi untuk representasi Biner

Membalik gen yang bernilai 1 dengan 0 dan sebaliknya.

b). Mutasi untuk representasi Integer

Terdiri dari 3 macam yaitu :

i. Operasi matematis pada nilai integer dimana mengubah nilai dengan menambah, mengurangi, mengalikan, atau membagi gen yang ada di kromosom.
ii. Pemilihan nilai secara acak dilakukan dengan mengganti gen pada kromosom dengan nilai random atau acak yang telah ditentukan.
iii. Mutasi *creep* (perlahan) dilakukan dengan mengubah nilai gen pada kromosom dengan perlahan sesuai grafik.

c). Mutasi untuk representasi Real

Nilai gen pada representasi real bersifat kontinyu sedangkan representasi biner maupun integer yang bersifat diskrit. Terdapat 2 macam mutasi untuk representasi real yaitu :

i. Mutasi *uniform*, nilai gen-gen pada kromosom didapatkan dari pembangkitan bilangan secara acak dengan distribusi seragam (*uniform distribution*).
ii. Mutasi *non-uniform* dengan distribusi tetap dilakukan menyerupai dengan metode *creep* pada representasi integer dengan representasi kromosomnya berupa bilangan real.

d). Mutasi untuk representasi Permutasi

Mutasi untuk representasi permutasi dilakukan dengan suatu cara tertentu yang menjamin agar kromosom yang dihasilkan akan tetap valid. Terdapat 4 macam jenis mutasi ini yaitu :

i. Mutasi pertukaran (*swap mutation*)

Dilakukan dengan memilih dua posisi gen secara acak dan mempertukarkan nilai gen pada kedua posisi itu.

ii. Mutasi penyisipan (*insert mutation*)

Dilakukan dengan memilih dua posisi gen secara acak dan menyisipkan gen kedua ke dekatnya gen pertama dan menggeser posisi gen kromosom itu.

iii. Mutasi pengacakan (*scramble mutation*)

Pemilihan sejumlah gen yang ditentukan yang disebut dengan segmen lalu nilai-nilai gen pada segmen tersebut dilakukan pengacakan.

iv. Mutasi pembalikan (*inversion mutation*)

Pemilihan sejumlah gen yang ditentukan yang disebut dengan segmen lalu nilai-nilai gen pada segmen tersebut dilakukan pembalikan dari posisi awal menjadi posisi akhir dan seterusnya.

3.1.5 Update Generasi (*Elitism*)

Setelah dilakukan proses seleksi, *crossover,* dan mutasi pada suatu populasi generasi. Maka langkah selanjutnya adalah melakukan *update* generasi untuk menentukan individu mana yang akan tetap bertahan *(survive)* pada populasi tersebut. Terdapat beberapa jenis metode untuk melakukan seleksi *survivor* diantaranya adalah sebagai berikut (Suyanto, 2008) :

1. *Update* Generasi *(Holland)*

   Genome yang dipilih/dianggap *survive* adalah *genome* hasil atau *offspring,* sedangkan *genome parents* dianggap mati atau keluar dari populasi.

2. *Continous Update*

   Memungkinkan berkumpulnya *genome offspring* dan *parents* dalam satu generasi karena dilakukan pemilihan secara acak baik itu dari gen *offspring* maupun *parents.*

3. Elitisme *(Generational Model)*

   Suatu populasi berukuran N kromosom/individu pada suatu generasi diganti dengan N individu baru pada generasi berikutnya.

4. *Steady State Update*

   Penggantian kromosom dilakukan hanya pada sejumlah kromosom tertentu dan memilih 2 *genome* sebagai *parents* dari *genome offsppring* yang digunakan untuk mengganti orangtua, genome terjelek dan tertua dari populasi.

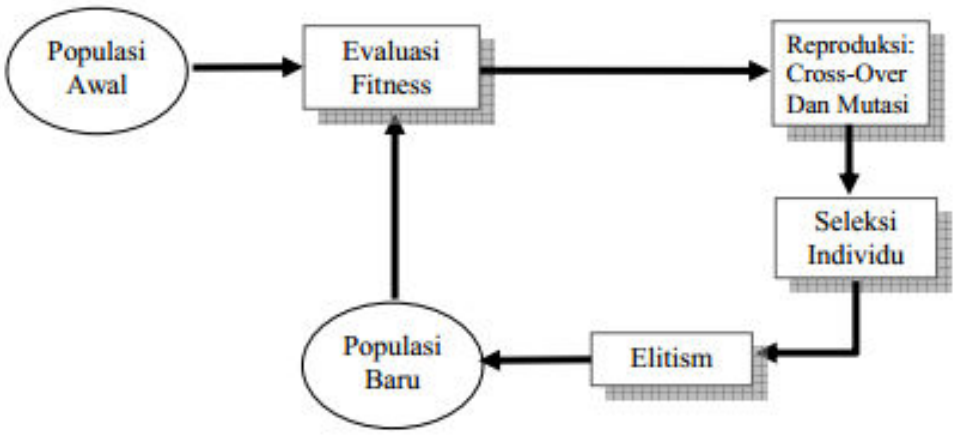
3.1.6 Parameter Algoritma Genetika

1. Probabilitas *Crossover* (Pc)

   Pc adalah kemungkinan atau probabilitas porsi *genome offspring* yang dihasilkan dari proses *crossover*. Misalkan Pc=100%, maka semua *genome offspring* dihasilkan dari proses *crossover*. Sedangkan Pc=0%, maka tidak pernah terjadi *crossover*. Pc biasanya berkisar antara 65-90% untuk menghasilkan keturunan yang baik.

2. Probabilitas Mutasi (Pm)

   Pm adalah kemungkinan atau probabilitas *genome* yang dirubah. Misalkan Pm = 100%, maka semua *genome* diubah sedangkan 0% tidak ada *genome* yang diubah. Pm biasanya diambil sebagai rata-rata mutasi dengan menghitung satu per panjang *genome* dengan *range* antara 1/NL sampai dengan 1/L dimana N adalah ukuran populasi dan L adalah panjang kromosom atau jumlah gen. Siklus algoritma genetika secara umum dapat dilihat pada gambar 3.14.



Gambar 3.14 Siklus algoritma genetika (Entin, 2011).