BAB II

LANDASAN TEORI

2.1. Sistem

Sistem adalah kumpulan dari elemen-elemen yang berinteraksi untuk mencapai tujuan tertentu. Komponen tersebut tidak lepas sendiri-sendiri. Subsistem tersebut saling berinteraksi dan saling berhubungan membentuk satu kesatuan sehingga tujuan sistem dapat tercapai. Suatu sistem mempunyai karakteristik tertentu, yaitu mempunyai komponen, batasan sistem, lingkungan luar sistem, penghubung, masukan, keluaran, pengolah dan tujuan. (Moh Masduki Putra, 2010).

2.2. Online

Online adalah segala aktivitas yang menggunakan internet dan dapat menghubungkan banyak orang, dimana Anda bisa berkomunikasi/berhubungan/terkoneksi dengan banyak orang melalui dunia maya. *Online* mempunyai jangkauan yang sangat luas, baik dalam negeri ataupun luar negeri (Sugesti, 2012).

2.3. Database

Database adalah sekumpulan tabel-tabel yang berisi data dan merupakan kumpulan dari field atau kolom. Struktur file yang mmenyusun sebuah database adalah Data Record dan Field. Data alalah suatu satuan informasi yang akan diolah. Sebelum diolah, data dikumpulkan didalam suatu file database.

Record adalah data yang isinya merupakan satu kesatuan seperti NamaUser dan password. Setiap keterangan yang mencakup NamaUser dan Password dinamakan satu record. Sertiap record diberi nomor urut yang disebut nomor record(Record number).

Field adalah sub bagian dari record (Anhar, 2010).

2.4. Sistem Pakar

Secara umum, sistem pakar (*expert system*) adalah sistem yang berusaha mengadopsi pengetahuan manusia ke komputer, agar komputer dapat menyelesaikan masalah seperti yang biasa dilakukan oleh para ahli. Ada beberapa definisi tentang sistem pakar, diantaranya :

a. Menurut Durkin : Sistem pakar adalah suatu program komputer yang dirancang untuk memodelkan kemampuan penyelesaian masalah yang dilakukan seorang pakar.

b. Menurut Ignizio : Sistem pakar adalah suatu model dan prosedur yang berkaitan, dalam suatu domain tertentu, yang mana tingkat keahliannya dapat dibandingkan dengan keahlian seorang pakar.

c. Menurut Giarratano : Sistem pakar adalah suatu sistem komputer yang bisa menyamai atau meniru kemampuan seorang pakar.

d. Menurut Turban : Sistem pakar (*expert system*) adalah paket perangkat lunak Pengambilan keputusan atau pemecahan masalah yang dapat mencapai tingkat performayang setara atau bahkan lebih dengan pakar manusia di beberapa bidang khusus dan biasanya mempersempit area masalah.

Ide dasar dari sistem pakar, teknologi kecerdasan buatan terapan adalah sederhana. Keahlian ditransfer dari pakar ke suatu komputer. *Knowledge* ini kemudian disimpan didalam komputer, dan pengguna menjalankan komputer untuk nasihat spesifik yang diperlukan. Sistem pakar menanyakan fakta-fakta dan dapat membuat inferensi hingga sampai pada kesimpulan khusus. Kemudian layaknya konsultan manusia, sistem pakar akan memberi nasihat kepada *non expert* dan menjelaskan, jika perlu logika dibalik nasihat yang diberikan. *Knowledge* dalam sistem pakar mungkin saja seorang ahli, atau *knowledge* yang umumnya terdapat dalam buku, jurnal, website dan orang yang mempunyai pengetahuan tentang suatu bidang. Sistem pakar yang baik dirancang agar dapat menyelesaikan suatu permasalahan tertentu dengan meniru kerja dari para ahli (Fadli, 2010).

2.4.1. Konsep Dasar Sistem Pakar

Konsep dasar sistem pakar mengandung : keahlian, ahli, pengalihan keahlian, inferensi, aturan dan kemampuan menjelaskan. Keahlian adalah suatu kelebihan penguasaan pengetahuan di bidang tertentu yang diperoleh dari pelatihan, membaca atau pengalaman. Contoh bentuk pengetahuan yang termasuk keahlian adalah :

a. Fakta-fakta pada lingkup permasalahan tertentu.

b. Teori-teori pada lingkup permasalahan tertentu.

c. Prosedur-prosedur dan aturan-aturan berkenaan dengan lingkup permasalahan tertentu.

d. Strategi-strategi global untuk menyelesaikan masalah.

e. Meta-knowledge (pengetahuan tentang pengetahuan).

Bentuk-bentuk ini memungkinkan para ahli untuk dapat mengambil keputusa lebih cepat dan lebih baik daripada seseorang yang bukan ahli. Seorang ahli adalah seseorang yang mampu menjelaskan suatu tanggapan, mempelajari hal-hal baru seputar topik permasalahan (domain), menyusun kembali pengetahuan jika dipandang perlu, memecah aturan-aturan jika dibutuhkan, dan menentukan relevan tidaknya keahlian mereka. Pengalihan keahlian dari para ahli ke komputer untuk kemudian dialihkan lagi ke orang lain yang bukan ahli, merupakan tujuan utama dari sistem pakar. Proses ini membutuhkan 4 aktivitas yaitu :

a. Tambahan pengetahuan

b. Representasi pengetahuan.

c. Inferensi pengetahuan.

d. Pengalihan pengetahuan ke user.

Pengetahuan yang disimpan di komputer disebut dengan nama basis pengetahuan. Ada 2 tipe pengetahuan, yaitu : fakta dan prosedur (biasanya berupa aturan). Salah satu fitur yang harus dimiliki oleh sistem pakar adalah kemampuan untuk menalar. Jika keahlian-keahlian sudah tersimpan sebagai basis pengetahuan dan sudah tersedia program yang mampu mengakses basisdata, maka komputer harus dapat diprogram untuk membuat inferensi. Proses inferensi ini dikemas dalam bentuk motor inferensi (inference engine). Sebagian besar sistem pakar komersial dibuat

dalam bentuk rule-based systems, yang mana pengetahuannya disimpan dalam bentuk aturan-aturan. Aturan tersebut biasanya berbentuk IF-THEN. Fitur lainnya dari sistem pakar adalah kemampuan untuk merekomendasi. Kemampuan inilah yang membedakan sistem pakar dengan sistem konvensional (Fadli, 2010).

2.5. Algoritma

Algoritma adalah Jantung dari ilmu komputer. Banyak sekali cabang ilmu komputer yang masuk dalam *terminology algoritma*. Dalam kehidupan sehari-hari banyak terdapat proses yang dapat dinyatakan dalam suatu algoritma. Contohnya adalah langkah-langkah dalam membuat kopi, tahapan-tahapan yang dilakukan bila ingin berpergian dengan kendaraan tertentu, dapat juga disebut sebagai algoritma. Pada saat membuat kopi selalu ada urutan langkah-langkah dalam pembuatannya. Secara umum, pihak (benda) yang mengerjakan proses disebut pemroses (*processor*). Pemroses tersebut dapat berupa manusia, *computer*, robot, alat-alat elektronik, atau benda-benda lainnya. Pemroses melakukan suatu proses dengan melaksanakan atau mengeksekusi algoritma yang menjabarkan proses tersebut (Budianto, 2010).

2.5.1. Sejarah Algoritma

Asal-usul kata Algoritma mempunyai sejarah yang unik. Para ahli sejarah matematika menemukan bahwa rupanya kata Algoritma berasal dari nama seorang ahli matematika dari Uzbekistan yang hidup di masa tahun 770-840 masehi yaitu Abu Ja'far Muhammad Ibnu Musa Al-Khuwarizmi. Kata "Al-

Khuwarizmi" diara oleh orang barat menjadi *Algorism.* Al-Khuwarizmi menulis buku yang berjudul Kitab Al Jabar Wal Muwabala yang memiliki arti "Buku Pemugaran dan Pengurangan" . Dari judul buku ini juga diperoleh asal usul kata "Aljabar" *(Algebra).* Perubahan dari kata *Algorism* menjadi *Algorithm* muncul karena kata *Algorism* sering dikelirukan dengan *Arithmetic,* sehingga akhiran *–sm* berubah menjadi *–thm.* Karena perhitungan dengan menggunakan angka Arab sudah menjadi hal yang biasa, maka kata *Algorithm* berangsur-angsur dipakai sebagai metode perhitungan (komputasi) secara umum, sehingga kehilangan makna kata aslinya. Dalam Bahasa Indonesia, kata *Algorithm* deserap menjadi Algoritma (Budianto, 2010).

2.5.2. Definisi Algoritma

Definisi dari Algoritma adalah "Urutan langkah-langkah logis penyelesaian masalah dalam bentuk kalimat dengan jumlah kata terbatas, tetapi disusun secara sistematis dan logis". (Budianto, 2010).

2.5.3. Ciri Algoritma

Algoritma memiliki beberapa cirri, antara lain :

a. Mempunyai awal dan akhir

   Setiap Algoritma memiliki tahap-tahap di mana algoritman akan mulai dan tahap algoritma akan berakhir.

b. Setiap langkah didefinisikan dengan tepat

Setiap langkah dalam suatu metode algoritma harus didefinisikan dengan tepat agar tidak terjadi kerancuan dan kesalahan saat algoritma dijalankan.

c. Memiliki masukan (*input*)

Algoritma dalam menyelesaikan suatu masalah harus memiliki *variable-variable* tertentu yang dijadikan masukan untuk menyelesaikan suatu masalah.

d. Mempunyai keluaran (*output*)

Algoritma jika sudah selesai dijalankan harus menghasilkan suatu hasil yang merupakan solusi dalam permasalahan yang diselesaikan dalam algoritma tersebut.

e. Harus efektif (bisa menyelesaikan persoalan)

Suatu algoritmna harus dapat menyelesaikan persoalan yang ada, untuk tujuan tersebutlah suatu algoritma dibangun.

2.5.4. Sifat Algoritma

a. *Input*

Sidat ini berarti, suatu algoritma memiliki kondisi awal sebelum dilaksanakan.

b. *Output*

Sifat ini berarti, suatu algoritma menghasilkan keluaran setelah dilaksanakan.

c. *Definitif*

Sifat ini berarti, langkah-langkah dari suatu algoritma terdefinisi dengan jelas.

d. *Finit*

Sifat ini berarti, suatu algoritma melakukan langkah yang terbatas jumlahnya dalam mengolah *input* menjadi *output*.

e. *Efektif*

Sifat ini berarti, suatu algoritma dapat member solusi sesuai harapan.

f. *General*

Sifat ini berarti, suatu algoritma berlaku untuk setiap himpunan *input*.

2.6. Algoritma Genetika

Algoritma Genetika adalah algoritma *heuristic adaptive* yang memiliki dasar pemikiran atau gagasan *evolusioner* untuk proses seleksi alami dan Genetika. Konsep dasar dari algoritma Genetika dirancang untuk menirukan proses di dalam sistem alami yang penting bagi evolusi makhluk hidup untuk dapat terus bertahan hidup, yang secara rinci teori ini dicetuskan oleh Charles Darwin (1859) yaitu "Survival of the Fittest".

Pelopor pertama penggunaan metode algoritma Genetika adalah John Holland pada tahun 60-an. Algoritma Genetika menggunakan analogi seleksi alam yang bekerja dari suatu populasi yang terdiri dari berbagai individu (gen), yang masing-masing individu mempresentasikan suatu solusi yang mungkin muncul dari persoalan yang dihadapi. Dalam hal ini, individu yang terpilih dilambangkan dengan sebuah nilai *fitness* yang digunakan untuk mencari solusi terbaik dari persoalan yang ada.

Kemampuan individu yang tinggi memiliki kesempatan untuk dapat melakukan reproduksi melalui kawin silang (*crossover*) dengan individu yang lain pada populasi yang sama. Individu baru yang terbentuk atau dihasilkan membawa sifat-sifat dari induknya. Sedangkan individu didalam populasi yang telah melalui proses seleksi namun tidak terpilih akan mati dengan sendirinya. Beberapa generasi baru yang terbentuk dan memiliki ketahanan hidup yang kuat akan bermunculan didalam populasi tersebut, yang kemudian dari proses seleksi, kawin silang, dan mutasi yang berkelanjutan menghasilkan generasi dengan kemampuan bertahan hidup yang bagus (Afandi, 2010). Untuk menggunakan algoritma genetik, solusi permasalahan direpresentasikan sebagai khromosom. Tiga aspek yang penting untuk penggunaan algoritma genetik :

1. Defenisi fungsi *fitness*
2. Defenisi dan implementasi representasi genetik
3. Defenisi dan implementasi operasi genetik

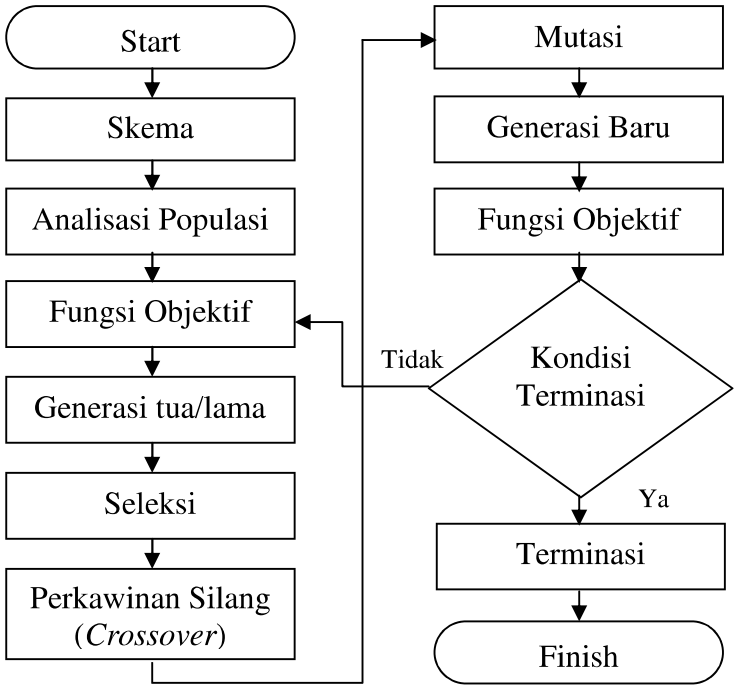
Jika ketiga aspek di atas telah didefinisikan, algoritma genetik akan bekerja dengan baik. Tentu saja, algoritma genetik bukanlah solusi terbaik untuk memecahkan segala masalah. Sebagai contoh, metode tradisional telah diatur untuk untuk mencari penyelesaian dari fungsi analitis *convex* yang "berperilaku baik" yang variabelnya sedikit. Pada kasus-kasus ini, metode berbasis kalkulus lebih unggul dari algoritma genetik karena metode ini dengan cepat menemukan solusi minimum ketika algoritma genetik masih menganalisa bobot dari populasi awal.

Problem-problem ini pengguna harus mengakui fakta dari pengalaman ini dan memakai metode tradisional yang lebih cepat tersebut. Akan tetapi,

banyak persoalan realistis yang berada di luar golongan ini. Selain itu, untuk persoalan yang tidak terlalu rumit, banyak cara yang lebih cepat dari algoritma genetik. Jumlah besar dari populasi solusi, yang merupakan keunggulan dari algoritma genetik, juga harus mengakui kekurangannya dalam dalam kecepatan pada sekumpulan komputer yang dipasang secara seri-*fitness function* dari tiap solusi harus dievaluasi. Namun, bila tersedia komputer-komputer yang paralel, tiap prosesor dapat mengevaluasi fungsi yang terpisah pada saat yang bersamaan. Karena itulah, algoritma genetik sangat cocok untuk perhitungan yang paralel.

2.6.1. Struktur Umum Algoritma Genetika

Algoritma genetik memberikan suatu pilihan bagi penentuan nilai parameter dengan meniru cara reproduksi genetik, pembentukan kromosom baru serta seleksi alami seperti yang terjadi pada makhluk hidup. Algoritma Genetik secara umum dapat diilustrasikan dalam diagram alir berikut ini :



Gambar 2. 1 Struktur Umum Algoritma Genetika

Algoritma genetik mempunyai karakteristik-karakteristik yang perlu diketahui sehingga dapat terbedakan dari prosedur pencarian atau optimasi yang lain, yaitu:

1. Algoritma genetik dengan pengkodean dari himpunan solusi permasalahan berdasarkan parameter yang telah ditetapkan dean bukan parameter itu sendiri.
2. Algoritma genetik pencarian pada sebuah solusi dari sejumlah individu-individu yang merupakan solusi permasalahan bukan hanya dari sebuah individu.
3. Algoritma genetik informasi fungsi objektif (*fitness*), sebagai cara untuk mengevaluasi individu yang mempunyai solusi terbaik, bukan turunan dari suatu fungsi.
4. Algoritma genetik menggunakan aturan-aturan transisi peluang, bukan aturan-aturan deterministik.

Variabel dan parameter yang digunakan pada algoritma genetik adalah:

1. Fungsi *fitness* (fungsi tujuan) yang dimiliki oleh masing-masing individu untuk menentukan tingkat kesesuaian individu tersebut dengan kriteria yang ingin dicapai.
2. Populasi jumlah individu yang dilibatkan pada setiap generasi.
3. Probabilitas terjadinya persilangan (*crossover*) pada suatu generasi.
4. Probabilitas terjadinya mutasi pada setiap individu.

5. Jumlah generasi yang akan dibentuk yang menentukan lama penerapan algoritma genetik.

Secara umum struktur dari suatu algoritma genetik dapat mendefenisikan dengan langkah-langkah sebagai berikut:

1. Membangkitkan populasi awal

   Populasi awal ini dibangkitkan secara random sehingga didapatkan solusi awal. Populasi itu sendiri terdiri atas sejumlah kromosom yang merepresentasikan solusi yang diinginkan.

2. Membentuk generasi baru

   Untuk membentuk generasi baru, digunakan operator reproduksi/ seleksi, *crossover* dan mutasi. Proses ini dilakukan berulang-ulang sehingga didapatkan jumlah kromosom yang cukup untuk membentuk generasi baru dimana generasi baru ini merupakan representasi dari solusi baru. Generasi baru ini dikenal denga istilah anak (*offspring*).

3. Evaluasi solusi

   Pada tiap generasi, kromosom akan melalui proses evaluasi dengan menggunakan alat ukur yang dinamakan *fitness*. Nilai *fitness* suatu kromosom menggambarkan kualitas kromosom dalam populasi tersebut. Proses ini akan mengevaluasi setiap populasi dengan menghitung nilai *fitness* setiap kromosom dan mengevaluasinya sampai

terpenuhi kriteria berhenti. Bila kriteria berhenti belum terpenuhi maka akan dibentuk lagi generasi baru dengan mengulangi langkah 2. Beberapa kriteria berhenti sering digunakan antara lain: berhenti pada generasi tertentu, berhenti setelah dalam beberapa generasi berturut-turut didapatkan nilai *fitness* tertinggi tidak berubah, berhenti dalam *n* generasi tidak didapatkan nilai *fitness* yang lebih tinggi.

Fungsi *fitness* tersebut sebagai berikut :

$$Fitness = \frac{1}{1 + Penalti} \qquad \ldots\ldots(2.1)$$

Dimana :

$$Penalti = \sum Bp \ \sum Np \qquad \ldots\ldots(2.2)$$

Dari persamaan diatas nilai fitness ditentukan oleh nilai *penalty*. *Penalty* tersebut menunjukkan jumlah pelanggaran kendala pada suatu kromosom. Semakin tinggi nilai *fitness* akan semakin besar kemungkinan kromosom tersebut terpilih ke generasi berikutnya. Jadi nilai *penalty* berbanding terbalik dengan nilai *fitness*, semakin kecil nilai *penalty* (jumlah pelanggaran) semakin besar nilai *fitness*nya.

Jadi fungsi *fitness* :

$$Fitness = \frac{1}{1 + \sum Bp \sum Np} \qquad \ldots\ldots(2.3)$$

Keterangan :

Bp : Bobot Pelanggaran

Np : Indikator Pelanggaran

2.6.2. Pengkodean

Pengkodean adalah suatu teknik untuk menyatakan populasi awal sebagai calon solusi suatu masalah ke dalam suatu kromosom sebagai suatu kunci pokok persoalan ketika menggunakan algoritma genetik. Berdasarkan jenis symbol yang digunakan sebagai nilai suatu gen, metode pengkodean dapat diklasifikasikan sebagai berikut :

1. Pengkodean biner merupakan cara pengkodean yang paling umum digunakan karena adalah yang pertama kali digunakan dalam algoritma genetik oleh Holland. Keuntungan pengkodean ini adalah sederhana untuk diciptakan dan mudah dimanipulasi. Pengkodean biner memberikan banyak kemungkinan untuk kromosom walaupun dengan jumlah nilai-nilai yang mungkin terjadi pada suatu gen yang sedikit (0 dan 1). Di pihak lain, pengkodean biner sering tidak sesuai untuk banyak masalah dan kadang pengoreksian harus dilakukan setelah operasi *crossover* dan mutasi.
2. Pengkodean bilangan riil adalah suatu pengkodean bilangan dalam bentuk riil. Masalah optimalisasi fungsi dan optimalisasi kendala lebih tepat jika diselesaikan dengan pengkodean bilangan riil karena struktur topologi ruang genotif untuk pengkodean bilangan riil identik dengan ruang fenotifnya, sehingga mudah membentuk operator

genetik yang efektif dengan cara memakai teknik yang dapat digunakan yang berasal dari metode konvensional.

3. Pengkodean bilangan bulat adalah metode yang mengkodekan bilangan dalam bentuk bilangan bulat. Pengkodean ini baik digunakan untuk masalah optimisasi kombinatorial.
4. Pengkodean struktur data adalah model pengkodean yang menggunakan struktur data. Pengkodean ini digunakan untuk masalah kehidupan yang lebih kompleks seperti perencanaan jalur robot, dan masalah pewarnaan *Graph*.

2.6.3. Operator Genetika

Algoritma genetik merupakan proses pencarian yang heuristik dan acak sehingga penekanan pemilihan operator yang digunakan sangat menentukan keberhasilan algoritma genetik dalam menemukan solusi optimum suatu masalah yang diberikan. Hal yang harus diperhatikan adalah menghindari terjadinya konvergensi *premature*, yaitu mencapai solusi optimum yang belum waktunya, dalam arti bahwa solusi yang diperoleh adalah hasil optimum lokal.

Operator genetik yang digunakan setelah proses evaluasi tahap pertama membentuk populasi baru dari generasi sekarang. Operator-operator tersebut adalah operator seleksi, *crossover* dan mutasi.

a. Seleksi

Seleksi bertujuan memberikan kesempatan reproduksi yang lebih besar bagi anggota populasi yang paling fit. Langkah pertama dalam seleksi ini adalah pencarian nilai *fitness*. Masing-masing individu dalam suatu wadah seleksi akan menerima probabilitas reproduksi yang tergantung pada nilai objektif dirinya sendiri terhadap nilai objektif dari semua individu dalam wadah seleksi tersebut. Nilai *fitness* inilah yang nantinya akan digunakan pada tahap seleksi berikutnya.

Kemampuan algoritma genetik untuk memproduksi kromosom yang lebih baik secara *progresif* tergantung pada penekanan selektif (*selective pressure*) yang diterapkan ke populasi. Penekanan selektif dapat diterapkan dalam dua cara. Cara pertama adalah membuat lebih banyak kromosom anak yang dipelihara dalam populasi dan memilih hanya kromosom-kromosom terbaik bagi generasi berikut. Walaupun orang tua dipilih secara acak, metode ini akan terus menghasilkan kromosom yang lebih baik berhubungan dengan penekanan selektif yang diterapkan pada individu anak tersebut.

Cara lain menerapkan penekanan selektif adalah memilih orang tua yang lebih baik ketika membuat keturunan baru. Dengan metode ini, hanya kromosom sebanyak yang dipelihara dalam populasi yang perlu dibuat bagi generasi berikutnya. Walaupun penekanan selektif tidak diterapkan ke level keturunan, metode ini

akan terus menghasilkan kromosom yang lebih baik, karena adanya penekanan selektif yang diterapkan ke orangtua.

Ada beberapa metode untuk memilih kromosom yang sering digunakan antara lain adalah seleksi roda rolet (*roulette wheel selection*), seleksi ranking (*rank selection*) dam seleksi turnamen (*tournament selection*).

Dalam penelitian ini, metode yang digunakan adalah seleksi roda rolet (*roulette wheel selection*). Pada seleksi ini, orang tua dipilih berdasarkan *fitness* mereka. Lebih baik kualitas suatu kromosom, lebih besar peluangnya untuk terpilih. Probabilitas suatu individu terpilih untuk *crossover* sebanding dengan *fitness*nya. Cara penyeleksian ini merupakan peniruan dari permainan roda rolet.

b. Crossover

Crossover (perkawinan silang) bertujuan menambah keanekaragaman string dalam populasi dengan penyilangan antar-string yang diperoleh dari sebelumnya. Beberapa jenis *crossover* tersebut adalah :

1. *Crossover* 1-titik

   Pada *crossover* dilakukan dengan memisahkan suatu string menjadi dua bagian dan selanjutnya salah satu bagian dipertukarkan dengan salah satu bagian dari string yang lain yang telah dipisahkan dengan cara yang sama. Proses yang

demikian dinamakan operator *crossover* satu titik seperti diperlihatkan pada gambar berikut :

Tabel 2. 1 Contoh *Crossover* 1-titik

Kromosom Orangtua 1	**11001**011
Kromosom Orangtua 2	11011**111**
Keturunan	**11001111**

2. Crossover 2-titik

Proses crossover ini dilakukan dengan memilih dua titik crossover. Kromosom keturunan kemudian dibentuk dengan barisan bit dari awal kromosom sampai titik crossover pertama disalin dari orangtua pertama, bagian dari titik crossover pertama dan kedua disalin dari orangtua kedua, kemudian selebihnya disalin dari orangtua pertama lagi.

Tabel 2. 2 Contoh *Crossover* 2-titik

Kromosom Orangtua 1	**11**001**11**
Kromosom Orangtua 2	11**011**11
Keturunan	**11011111**

3. *Crossover* seragam

Crossover seragam manghasilkan kromosom keturunan dengan menyalin bit-bit secara acak dari kedua orangtuanya.

Tabel 2. 3 Contoh *Crossover* seragam

Kromosom Orangtua 1	**110**01**11**
Kromosom Orangtua 2	110**111**11
Keturunan	**11011111**

c. Mutasi

Mutasi merupakan proses mengubah nilai dari satu atau beberapa gen dalam suatu kromosom. Operasi *crossover* yang dilakukan pada kromosom dengan tujuan untuk memperoleh kromosom-kromosom baru sebagai kandidat solusi pada generasi mendatang denga *fitness* yang lebih baik, dan lama-kelamaan menuju solusi optimum yang diinginkan. Akan tetapi, untuk mencapai hal ini, penekanan selektif juga memegang peranan yang penting. Jika dalam proses pemilihan kromosom-kromosom cenderung pada kromosom yang memiliki *fitness* yang tinggi saja, *konvergensi premature*, yaitu mencapai solusi yang optimal lokal sangat mudah terjadi. Untuk menghindari *konvergensi premature* tersebut dan tetap menjaga perbedaan (*diversity*) kromosom-kromosom dalam populasi, selain melakukan penekanan selektif yang lebih efisien, operator mutasi juga dapat digunakan. Proses mutasi dalam system biologi berlangsung dengan mengubah isi *allele* gen pada suatu *locus* dengan *allele* yang lain. Proses mutasi ini bersifat acak sehingga tidak selalu menjamin bahwa setelah proses mutasi akan diperoleh kromosom dengan *fitness* yang lebih baik.

Operator mutasi merupakan operasi yang menyangkut satu kromosom tertentu. Beberapa cara operasi mutasi diterapkan dalam algoritma genetik menurut jenis pengkodean terhadap *phenotype*, antara lain :

1. Mutasi dalam Pengkodean Biner

   Mutasi pada pengkodean biner merupakan operasi yang sangat sederhana. Proses yang dilakukan adalah menginversi nilai bit pada posisi tertentu yang terpilih secara acak (atau menggunakan skema tertentu) pada kromosom, yang disebut *inverse* bit.

   Tabel 2. 4 Contoh Mutasi pada pengkodean biner

Kromosom sebelum mutasi	1 0 0 1 0 **1** 1 1
Kromosom setelah mutasi	1 0 0 1 0 **0** 1 1

2. Mutasi dalam Pengkodean Permutasi

   Proses mutasi yang dilakukan dalam pengkodean biner dengan mengubah langsung bit-bit pada kromosom tidak dapat dilakukan pada pengkodean permutasi karena konsistensi urutan permutasi harus diperhatikan. Salah satu cara yang dapat dilakukan adalah dengan memilih dua posisi (*locus*) dari kromosom dan kemudian nilainya saling dipertukarkan.

   Tabel 2. 5 Contoh Mutasi pada pengkodean permutasi

Kromosom sebelum mutasi	1 2 **3** 4 5 6 **7** 8 9
Kromosom setelah mutasi	1 2 **7** 4 5 6 **3** 8 9

3. Mutasi dalam Pengkodean Nilai

   Mutasi pada pengkodean nilai hampir sama dengan yang dilakukan pada pengkodean biner, tetapi yang dilakukan bukan menginversi nilai bit. Penerapannya bergantung pada jenis nilai yang digunakan. Sebagai contoh untuk nilai riil, proses mutasi dapat dilakukan seperti yang dilakukan pada

pengkodean permutasi, dengan saling mempertukarkan nilai dua gen pada kromosom.

4. Mutasi dalam Pengkodean Pohon

   Mutasi dalam pengkodean pohon dapat dilakukan antara lain dengan cara mengubah operator (+, -, *, /) atau nilai yang terkandung pada suatu verteks pohon yang dipilih. Atau, dapat juga dilakukan dengan memilih dua verteks dari pohon dan saling mempertukarkan operator atau nilainya.

Tidak setiap gen selalu dimutasi tetapi mutasi dikontrol dengan probabilitas tertentu yang disebut dengan *mutation rate* (probabilitas mutasi) dengan notasi *Pm*. Jenis operator mutasi antara lain:

1. Mutasi Terarah

   Mutasi terarah tergantung dari informasi gen. Informasi gen tersebut berupa nilai pelanggaran gen (*violation gen*). Ini berarti bahwa setiap gen mempunyai peluang yang berbeda untuk terjadi mutasi. Gen yang mempunyai nilai pelanggaran yang lebih besar maka gen tersebut mempunyai peluang untuk terjadi mutasi. Mutasi ini menghubungkan nilai pelanggaran relatif (nilai pelanggaran suatu gen dibagi dengan nilai pelanggaran total suatu kromosom) dengan probabilitas terjadinya mutasi dari suatu gen pada kromosom. Hubungan tersebut dinyatakan secara matematis sebagai berikut :

$$nr(\mathrm{i}) = \frac{n(i)}{1+n_{total}} \qquad \ldots\ldots(2.4)$$

$$Pm(\mathrm{i}) = (1 + (nr(\mathrm{i}))^2\, pm \qquad \ldots\ldots(2.5)$$

Keterangan :

$nr(i)$: nilai pelanggaran relative gen ke-i

$ntotal$: nilai pelanggaran total kromosom

$pm(i)$: probabilitas mutasi gen ke-i

pm : probabilitas mutasi

2. Mutasi Biasa

   Mutasi biasa tidak tergantung dari informasi gen. Setiap gen mempunyai peluang yang sama untu terjadi mutasi.

d. Parameter Genetika

Seperti halnya dengan algoritma lainnya, Algoritma Genetik juga mempunyai parameter-parameter dalam menjalankan fungsinya. Parameter-parameter ini akan sangat berpengaruh terhadap kinerja Algoritma Genetik dalam menyelesaikan masalah yang dihadapi. Parameter-parameter tersebut antara lain :

1. Probabilitas Persilangan (*Crossover Probability*)

   Menunjukkan kemungkinan *crossover* terjadi antara 2 kromosom. Jika tidak terjadi *crossover* maka keturunannya akan sama persis dengan kromosom orangtua, tetapi tidak berarti generasi yang baru akan sama persis dengan generasi yang lama. Jika probabilitas *crossover* 100% maka semua keturunannya dihasilkan dari *crossover*. *Crossover* dilakukan dengan harapan bahwa kromosom yang baru akan lebih baik.

2. Probabilitas Mutasi (*Mutation Probability*)

   Menunjukkan kemungkinan mutasi terjadi pada gen-gen yag menyusun sebuah kromosom. Jika tidak terjadi mutasi maka keturunan yang dihasilkan setelah *crossover* tidak berubah. Jika terjadi mutasi bagian kromosom akan berubah. Jika probabilitas 100%, semua kromosom dimutasi. Jika probabilitasnya 0%, tidak ada yang mengalami mutasi.

3. Jumlah Individu

   Menunjukkan jumlah kromosom yang terdapat dalam populasi (dalam satu generasi). Jika hanya sedikit kromosom dalam populasi maka algoritma genetik akan mempunyai sedikit variasi kemungkinan untuk melakukan *crossover* antara orangtua karena hanya sebagian kecil dari *search space* yang dipakai. Sebaliknya jika terlalu banyak maka algoritma genetik akan berjalan lambat.

4. Jumlah Populasi

   Menetukan jumlah populasi atau banyaknya generasi yang dihasilkan, digunakan sebagai batas akhir proses seleksi, persilangan dan mutasi.