BAB II

LANDASAN TEORI

2.1 Algoritma Genetika

Evolutionary Algorithm merupakan terminologi umum yang menjadi payung bagi empat istilah : algoritma genetika (*genetic algorithm*), pemrograman genetika (*genetic programming*), strategi evolusi (*evolution strategies*), dan pemrograman evolusi (*evolutionary programming*). Tetapi, jenis *evolutionary algorithm* yang paling populer dan banyak digunakan adalah algoritma genetika (*genetic algorithm*).

Algoritma genetika merupakan evolusi/ perkembangan dunia komputer dalam bidang kecerdasan buatan (*artificial intelligence*). Sebenarnya kemunculan algoritma genetika ini terinspirasi oleh teori evolusi Darwin (walaupun pada kenyatanya teori tersebut terbukti keliru) dan teori-teori dalam ilmu biologi, sehingga banyak istilah dan konsep biologi yang digunakan. Karena itu sesuai dengan namanya, proses-proses yang terjadi dalam algoritma genetika sama dengan apa yang terjadi pada evolusi biologi.

Algoritma genetika merupakan teknik pencarian nilai optimum secara *stochastic* berdasarkan mekanisme seleksi alam. Algoritma genetika berbeda dengan teknik konvergensi konvensional yang lebih bersifat deterministik [7]. Metodenya sangat berbeda dengan kebanyakan algoritma optimasi lainnya, yaitu mempunyai ciri-cirinya sebagai berikut :

a. Menggunakan hasil pengkodean dari parameter, bukan parameter itu sendiri.
b. Bekerja pada populasi bukan pada sesuatu yang unik.
c. Menggunakan nilai satu-satunya pada fungsi dalam prosesnya. Tidak mengunakan fungsi luar atau pengetahuan luar lainnya.
d. Menggunakan fungsi transisi probabilitas, bukan sesuatu yang pasti.

2.1.1 Sejarah Singkat Algoritma Genetika

Awal sejarah perkembangan dari algoritma genetika (*genetic algorithm*) dimulai pada tahun 1960. Pada waktu itu, I. Rochenberg dalam bukunya yang berjudul "*Evolution Strategies*" mengemukakan tentang evolusi komputer (*computer evolutionary*) yang kemudian dikembangkan oleh peneliti lain. Algoritma genetika sendiri pertama kali dikembangkan oleh John Holland pada tahun 1970-an di New York Amerika Serikat yang dikembangkan bersama mahasiswa dan rekan-rekannya. Hal tersebut bisa dibuktikan dengan adanya buku yang dibuat oleh Holland dengan judul "*Adaptation in Natural and Artificial System*" yang diterbitkan pada tahun 1975.

Lalu tujuh belas tahun kemudian, John Koza melakukan penelitian suatu program yang berkembang dengan menggunakan algoritma genetika. Program yang dikenal dengan sebutan metode "*Genetic Programming*" tersebut dibuat menggunakan LISP (bahasa pemrogramannya dapat dinyatakan dalam bentuk *parse tree* yaitu objek kerjanya pada algoritma genetika). Sampai sekarang algoritma genetika ini terus digunakan untuk memecahkan permasalahan yang sulit dipecahkan dengan menggunakan algoritma konvensional.

2.1.2 Aplikasi Algoritma Genetika

Sejak pertama kali dirintis oleh John Holland pada tahun 1960-an, algoritma genetika telah dipelajari, diteliti, dan diaplikasikan secara luas pada berbagai bidang. Algoritma genetika banyak digunakan pada masalah praktis yang berfokus pada pencarian parameter-parameter optimal.

Hal ini membuat banyak orang mengira bahwa algoritma genetika hanya bisa digunakan untuk masalah optimasi. Pada kenyataannya, algoritma juga memiliki performansi yang bagus untuk masalah-masalah selain optimasi [7].

Keuntungan penggunaan algoritma genetika sangat jelas terlihat dari kemudahan implementasi dan kemampuannya untuk menemukan solusi yang bagus (bisa diterima) secara cepat untuk masalah-masalah berdimensi tinggi. Algoritma genetika sangat berguna dan efisien untuk masalah-masalah dengan karakteristik sebagai berikut :

a. Ruang masalah sangat besar, kompleks, dan sulit dipahami.
b. Kurang atau bahkan tidak ada pengetahuan yang memadai untuk merepresentasikan masalah ke dalam ruang pencarian yang lebih sempit.
c. Tidak tersedianya analisis matematika yang memadai.
d. Ketika metode-metode konvensional sudah tidak mampu meyelesaikan masalah yang dihadapi.
e. Solusi yang diharapkan tidak harus paling optimal, tetapi cukup bagus atau bisa diterima.
f. Terdapat batasan waktu, misalnya *real time system* atau sistem waktu nyata.

Algoritma genetika sudah banyak diaplikasikan untuk penyelesaian masalah dan pemodelan dalam bidang teknologi, bisnis, dan *entertainment*, seperti :

1) Optimasi

Algoritma genetika untuk optimasi numerik dan optimasi kombinatorial seperti *Traveling Salesman Problem* (TSP), perancangan *Integrated Circuit* atau IC [4], *Lob Shop Scheduling* [2], optimasi video dan suara.

2) Pemrograman Otomatis

Algoritma genetika telah digunakan untuk melakukan proses evolusi terhadap program komputer untuk merancang struktur komputasional, seperti *cellular automata* dan *sorting network.*

3) *Machine Learning*

Algoritma genetika telah berhasil diaplikasikan untuk memprediksi struktur protein, dan berhasil diaplikasikan dalam perancangan *neural networks* (jaringan syaraf tiruan) untuk melakukan proses evolusi terhadap aturan-aturan pada *learning classifier system* atau *symbolic production system,* juga digunakan untuk mengontrol robot.

4) Model Ekonomi

Algoritma genetika telah digunakan untuk memodelkan proses-proses inovasi dan pembangunan *bidding strategies*.

5) Model Sistem Imunisasi

Algoritma genetika telah berhasil digunakan untuk memodelkan berbagai aspek pada sistem imunisasi alamiah, termasuk *somatic mutation* selama kehidupan individu dan menemukan keluarga dengan gen ganda (*multi-gene families*) sepanjang waktu evolusi.

6) Model Ekologis

Algoritma genetika berhasil digunakan untuk memodelkan fenomena ekologis seperti *host-parasite co-evolutions*, simbiosis, dan aliran sumber daya dalam ekologi.

2.1.3 Proses Pada Algoritma Genetika

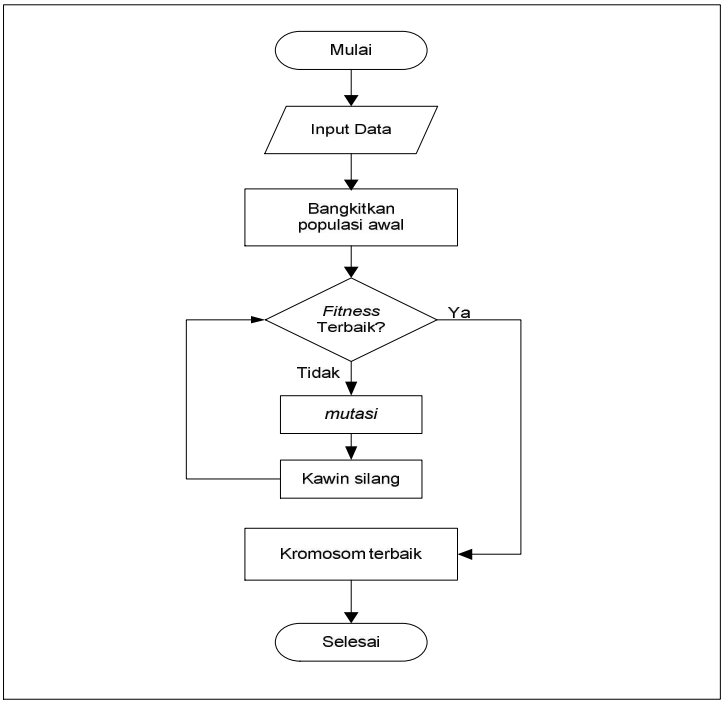
Algoritma genetika adalah algoritma pencarian yang berdasarkan pada mekanisme sistem natural yakni genetik dan seleksi alam. Dalam aplikasi algoritma genetik, variabel solusi dikodekan ke dalam struktur *string* yang merepresentasikan barisan gen, yang merupakan karakteristik dari solusi problem. Berbeda dengan teknik pencarian konvensional, algoritma genetik berangkat dari himpunan solusi yang dihasilkan secara acak. Himpunan ini disebut populasi. Sedangkan setiap individu dalam populasi disebut kromosom yang merupakan representasi dari solusi. Kromosom-kromosom berevolusi dalam suatu proses iterasi yang berkelanjutan yang disebut generasi. Pada setiap generasi, kromosom dievaluasi berdasarkan suatu fungsi evaluasi. Setelah beberapa generasi maka algoritma genetik akan konvergen pada kromosom terbaik, yang diharapkan merupakan solusi optimal.

Algoritma genetika adalah algoritma pencarian hasil yang terbaik, yang didasarkan pada perkawinan dan seleksi gen secara alami. Kombinasi perkawinan ini dilakukan dengan proses acak (*random*). Dimana struktur gen hasil proses perkawinan ini, akan menghasilkan gen inovatif untuk diseleksi.

Dalam setiap generasi, ciptaan buatan yang baru (hasil perkawinan), diperoleh dari bit-bit dan bagian-bagian gen induk yang terbaik. Diharapkan dengan mengambil dari gen induk yang terbaik ini diperoleh gen akan yang lebih baik lagi. Meskipun pada kenyataannya tidak selalu tercipta gen anak yang lebih baik dari induknya. Ada kemungkinan lebih baik, sama baiknya, atau bahkan mungkin lebih buruk.

Tujuan dari algoritma genetika ini adalah menghasilkan populasi yang terbaik dari populasi awal. Sedangkan keuntungan dari algoritma genetika adalah sifat metoda pencariannya yang lebih optimal, tanpa terlalu memperbesar ruang pencarian.

Dalam menyusun suatu algoritma genetika menjadi program, maka diperlukan beberapa tahapan proses, yaitu proses pembuatan generasi awal, proses genetika, proses seleksi, dan pengulangan proses sebelumnya.



Gambar 2.1 *Flowchart* Proses Algoritma Genetika

Pada algoritma genetika ini terdapat beberapa definisi penting yang harus dipahami sebelumnya, yaitu :

a. Gen

   Gen merupakan nilai yang menyatakan satuan dasar yang membentuk suatu arti tertentu dalam satu kesatuan gen yang dinamakan kromosom.

b. Kromosom / Individu

   Kromosom merupakan gabungan dari gen-gen yang membentuk nilai tertentu dan menyatakan solusi yang mungkin dari suatu permasalahan.

c. Populasi

   Populasi merupakan sekumpulan individu yang akan diproses bersama dalam satu satuan siklus evolusi.

d. *Fitness*

   Fitness menyatakan seberapa baik nilai dari suatu individu yang didapatkan.

e. Seleksi

   Seleksi merupakan proses untuk mendapatkan calon induk yang baik.

f. *Crossover*

   Crossover merupakan proses pertukaran atau kawin silang gen-gen dari dua induk tertentu.

g. Mutasi

   Mutasi merupakan proses pergantian salah satu gen yang terpilih dengan nilai tertentu.

h. Generasi

   Generasi merupakan urutan iterasi dimana beberapa kromosom bergabung.

i. *Offspring*

   Offspring merupakan kromosom baru yang dihasilkan.

Untuk menyelesaikan suatu permasalahan menggunakan Algoritma Genetika, perlu diketahui beberapa macam *encoding* guna menentukan operator *crossover* dan mutasi yang akan digunakan. *Encoding* tersebut tergantung pada permasalahan apa yang diangkat. Beberapa macam *encoding* akan dijelaskan di bawah ini :

1) *Binary Encoding*

Encoding jenis ini sering digunakan. Kromosom dari *binary encoding* ini berupa kumpulan dari nilai *biner* 0 dan 1.

Contohnya:

Chromosome1 1101100100110110

Chromosome2 1101011000011110

Dalam *Binary Encoding* memungkinkan didapatkan kromosom dengan nilai *allele* yang kecil, tetapi kekurangannya tidak dapat digunakan untuk beberapa permasalahan dan terkadang diperlukan adanya koreksi setelah proses *crossover* dan mutasi. Salah satu permasalahan yang menggunakan *encoding* adalah menghitung nilai maksimal dari suatu fungsi.

2) *Permutation Encoding*

Kromosom dari *permutation encoding* ini berupa kumpulan dari nilai integer yang mewakili suatu posisi dalam sebuah urutan. Biasanya digunakan pada permasalahan TSP (*Travelling Salesman Problem*).

Contohnya:

Chromosome 1 1 4 7 9 6 3 5 0 2 8

Chromosome 2 9 3 2 5 8 1 6 0 4 7

3) *Value Encoding*

Kromosom dari *value encoding* berupa kumpulan dari suatu nilai, yang bisa berupa macam-macam nilai sesuai dengan permasalahan yang dihadapi, seperti bilangan *real*, *char* atau objek yang lain. *Encoding* ini merupakan pilihan yang bagus untuk beberapa permasalahan khusus, biasanya diperlukan metode khusus untuk memproses *crossover* dan mutasinya sesuai dengan permasalahan yang dihadapi.

Contohnya:

Chromosome 1 A B E D B C A E D D

Chromosome 2 N W W N E S S W N N

4) *Tree Encoding*

Tree Encoding biasanya digunakan pada *genetic programming*. Kromosom yang digunakan berupa sebuah *tree* dari beberapa objek, seperti fungsi atau *command* pada *genetic programming*.

2.1.3.1 Membuat Generasi Awal

Pendefinisian individu merupakan proses pertama yang harus dilakukan dalam Algoritma Genetika yang menyatakan salah satu solusi yang mungkin dari suatu permasalahan yang diangkat. Pendefinisian individu atau yang biasa disebut juga merepresentasikan kromosom yang akan diproses nanti, dilakukan dengan mendefinisikan jumlah dan tipe dari gen yang digunakan dan tentunya dapat mewakili solusi permasalahan yang diangkat.

Proses pembuatan generasi awal dilakukan dengan membangkitkan populasi secara acak, dimana populasi tersebut berisi beberapa kromosom yang telah didefinisikan sebelumnya. Dalam proses ini perlu diperhatikan syarat-syarat yang harus dipenuhi untuk menunjukkan suatu solusi dari permasalahan dan jumlah kromosom yang digunakan dalam satu populasi. Jika kromosom yang digunakan terlalu sedikit, maka individu yang dapat digunakan untuk proses *crossover* dan mutasi akan sangat terbatas, sehingga menyia-nyiakan proses yang ada. Tetapi jika jumlah kromosom yang digunakan terlalu banyak, akan memperlambat proses algoritma genetika yang dilakukan. Jumlah kromosom yang dianjurkan lebih besar dari jumlah gen yang ada dalam satu kromosom, tetapi juga harus disesuaikan dengan permasalahan, apabila jumlah gennya terlalu banyak, tidak juga dianjurkan seperti itu.

Pertama, dibentuk sebuah populasi untuk sejumlah gen. Susunan gen ini terbentuk dari kromosom yang disusun membentuk suatu *string*. Nilai *string* dibentuk secara *random* dengan memilih setiap kromosom dengan kode tertentu secara *random* pada *string*. Setiap *string* didekodekan menjadi satu set parameter yang dapat mewakilinya. Parameter ini merupakan model numerik ruang permasalahan. Model numerik ini memberikan pemecahan berdasarkan masukan parameter.

Sebagai dasar kualitas pemecahan, setiap *string* diberi nilai *fitness*. Dengan nilai *fitness* tersebut, operasi genetika (kawin silang, dan mutasi) dipergunakan untuk menciptakan generasi baru *string*. Set *string* baru tersebut didekodekan dan dievaluasi lagi, sampai generasi *string* yang baru terbentuk kembali. Proses ini diulang-ulang sampai sejumlah inputan generasi dari *user*.

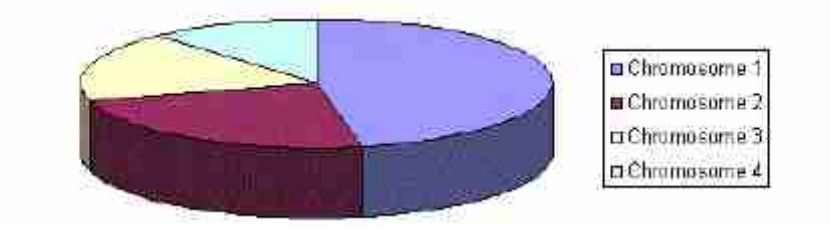
2.1.3.2 Proses Seleksi

Operasi seleksi dilakukan dengan memperhatikan *fitness* dari tiap individu, manakah yang dapat dipergunakan untuk generasi selanjutnya. Seleksi ini digunakan untuk mendapatkan calon induk yang baik, semakin tinggi nilai fitnessnya maka semakin besar juga kemungkinan individu tersebut terpilih. Terdapat beberapa macam cara seleksi untuk mendapatkan calon induk yang baik, diantaranya adalah seleksi *roulette wheel*, *steady state*, *tournament* dan *rank*. Proses seleksi yang biasa digunakan adalah mesin roulette (*roulette wheel*).

Beberapa penjelasan tentang keempat metode seleksi di atas adalah sebagai berikut :

1) *Roulette Wheel*

Calon induk yang akan dipilih berdasarkan nilai *fitness* yang dimilikinya, semakin baik individu tersebut yang ditunjukkan dengan semakin besar nilai fitnessnya akan mendapatkan kemungkinan yang lebih besar untuk terpilih sebagai induk. Misalkan saja *roulette wheel* merupakan tempat untuk menampung seluruh kromosom dari tiap populasi, maka besarnya tempat dari *roulette wheel* tersebut menunjukkan seberapa besar nilai *fitness* yang dimiliki oleh suatu kromosom, semakin besar nilai *fitness* tersebut, maka semakin besar pula tempat yang tersedia. Ilustrasinya dapat digambarkan sebagai berikut :



Gambar 2.2 Ilustrasi seleksi dengan *Roulette Wheel*

2) *Steady State*

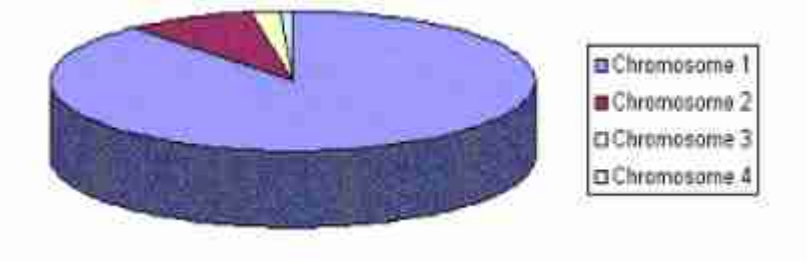
Metode ini tidak banyak digunakan dalam proses seleksi karena dilakukan dengan mempertahankan individu yang terbaik. Pada setiap generasi, akan dipilih beberapa kromosom dengan nilai fitnessnya yang terbaik sebagai induk, sedangkan kromosom-kromosom yang memiliki nilai *fitness* terburuk akan digantikan dengan *offspring* yang baru. Sehingga pada generasi selanjutnya akan terdapat beberapa populasi yang bertahan.

3) *Tournament*

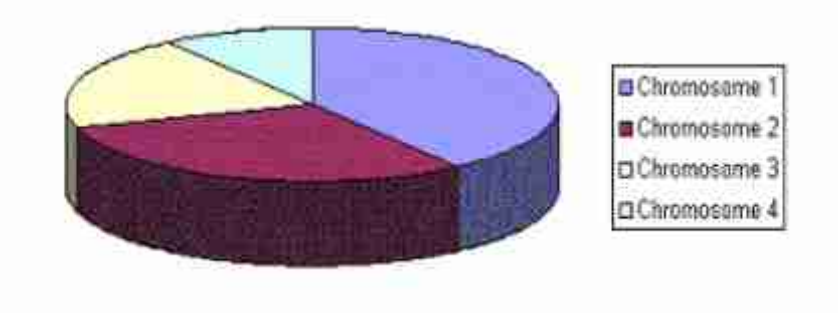
Dalam metode seleksi *tournament* sejumlah n individu dipilih secara acak dan kemudian menentukan fitnessnya. Kebanyakan metode seleksi ini digunakan pada *binary*, dimana hanya dua individu yang dipilih.

4) *Rank*

Seleksi ini memperbaiki proses seleksi yang sebelumnya yaitu *roulette wheel* karena pada seleksi tersebut kemungkinan salah satu kromosom mempunyai nilai *fitness* yang mendominasi hingga 90% bisa terjadi, sehingga nilai *fitness* yang lain akan mempunyai kemungkinan yang sangat kecil sekali untuk terpilih. Sehingga dalam seleksi *rank*, dilakukan perumpamaan sesuai dengan nilai fitnessnya, nilai *fitness* terkecil diberi nilai 1, yang terkecil kedua diberi nilai 2, dan begitu seterusnya sampai yang terbagus diberi nilai N (jumlah kromosom dalam populasi). Nilai tersebut yang akan diambil sebagai presentasi tepat yang tersedia. Ilustrasinya dapat dilihat seperti pada gambar berikut :



Gambar 2.3 Ilustrasi Metode *Rank Selection* (situasi sebelum ranking)



Gambar 2.4 Ilustrasi Metode *Rank Selection* (situasi sudah ranking)

2.1.3.3 Proses Regenerasi

Dalam proses regenerasi ini dilakukan tiga buah proses utama yang dipilih secara acak untuk setiap generasi. Namun pemilihan secara acak ini berdasarkan persentase tertentu. Ketiga proses tersebut adalah mutasi, kawin silang, atau reproduksi. Dari ketiga proses ini prosentase kemungkinan proses tersebut dijalankan terhadap suatu generasi adalah sama. Karena masing-masing proses mempunyai kemungkinan menghasilkan gen terbaik. Sekalipun dalam proses regenerasi tidak dibawa sifat gen induknya, namun ada kemungkinan menghasilkan gen terbaik. Kemudian dilakukan proses seleksi dan pengulangan proses regenerasi sejumlah generasi.

2.1.3.4 Proses Mutasi

Mutasi juga merupakan salah satu operator penting dalam algoritma genetika selain *crossover*. Metode dan tipe mutasi yang dilakukan juga tergantung pada *encoding* dan permasalahan yang diangkat. Berdasarkan encodingnya terdapat beberapa macam, diantaranya adalah sebagai berikut :

1) *Binary Encoding*

Melakukan inversi pada bit yang terpilih, 0 menjadi 1 dan sebaliknya, 1 menjadi 0.

Contoh : 1**1**001001 => 1**0**001001

2) *Permutation Encoding*

Memilih dua nilai dari gen dan menukarnya.

Contoh : (1 **2** 3 4 5 **8** 9 7) => (1 **8** 3 4 5 6 **2** 9 7)

Beberapa operator mutasi telah diciptakan untuk representasi permutasi, seperti metode *inversion*, *insertion*, *displacement*, dan *reciprocal exchange mutation*.

a. *Inversion Mutation*

Inversion mutation memilih dua posisi dalam sebuah kromosom dengan cara acak dan kemudian menginversikan substring di antara dua posisi tersebut.

b. *Insertion Mutation*

Insertion Mutation memilih sebuah gen dengan cara acak dan memasukkan ke dalam kromosom dengan cara acak pula.

c. *Displacement Mutation*

Displacement Mutation memilih sebuah sub/sekelompok gen dengan cara acak kemudian memasukkan ke dalam kromosom dengan cara acak.

d. *Reciprocal Exchange Mutation* (REM)

Reciprocal Exchange Mutation memilih dua posisi secara acak, kemudian menukar dua gen dalam posisi tersebut.

3) *Value Encoding*

Menentukan sebuah nilai kecil yang akan ditambahkan atau dikurangkan pada salah satu gen dalam kromosom.

Contoh : (1.29 5.68 **2.86 4.11** 5.55) => (1.29 5.68 **2.73 4.22** 5.55)

4) *Tree Encoding*

Node yang terpilih akan diubah.

Karena proses mutasi juga merupakan salah satu operator dasar dalam algoritma genetika, sehingga sama dengan *crossover*, mutasi juga memerlukan probabilitas dengan proses yang sama seperti pada probabilitas *crossover*. Individu dengan nilai probabilitas yang lebih kecil dari probabilitas yang telah ditentukan yang akan melewati proses mutasi. Nilai probabilitas mutasi ini menunjukkan seberapa sering gen tertentu dari kromosom yang telah diproses dengan *crossover* akan melewati mutasi. Jika tidak ada proses mutasi, maka *offspring* yang dihasilkan akan sama dengan hasil individu setelah proses *crossover*, tanpa ada perubahan sedikitpun. Proses mutasi ini biasanya dilakukan untuk mencegah terjadinya lokal optimum, proses mutasi ini sebaiknya tidak terlalu sering dilakukan karena proses algoritma genetika akan cepat berubah menjadi *random search*. Pada probabilitas mutasi, jika terlalu rendah akan mengakibatkan banyak gen yang berguna tidak sempat untuk dimanfaatkan dan jika terlalu besar akan menyebabkan *offspring* kehilangan sifat dari induknya dan tidak akan dapat memanfaatkan lagi proses evolusi alamiah.

2.1.3.5 Proses Kawin Silang

Proses kawin silang (*crossover*) adalah salah satu operator penting dalam algoritma genetika, metode dan tipe *crossover* yang dilakukan tergantung dari *encoding* dan permasalahan yang diangkat. Ada beberapa cara yang bisa digunakan untuk melakukan *crossover* sesuai dengan encodingnya yang dijelaskan sebagai berikut:

1) *Binary Encoding*

a. *Crossover* Satu Titik

Memilih satu titik tertentu, selanjutnya nilai biner sampai titik crossovernya dari induk pertama digunakan dan sisanya dilanjutkan dengan nilai biner dari induk kedua.

Contoh :

11001011 + 11011**111** = **11001111**

b. *Crossover* Dua Titik

Memilih dua titik tertentu, lalu nilai biner sampai titik *crossover* pertama pada induk pertama digunakan, dilanjutkan dengan nilai biner dari titik pertama sampai titik kedua dari induk kedua, kemudian sisanya melanjutkan nilai biner dari titik kedua pada induk pertama lagi.

Contoh :

11001**11** + 11**011**11 = **11011111**

c. *Crossover Uniform*

Nilai biner yang digunakan dipilih secara *random* dari kedua induk.

Contoh :

1**10**01**11** + **1**10**111**01 = **11011111**

d. *Crossover* Aritmatik

Suatu operasi aritmetika digunakan untuk menghasilkan *offspring* yang baru.

Contoh :

11001011 + 11011111 = **11001001** (AND)

2) *Permutation Encoding*

Memilih satu titik tertentu, nilai permutasi sampai titik *crossover* pada induk pertama digunakan lalu sisanya dilakukan *scan* terlebih dahulu, jika nilai permutasi pada induk kedua belum ada pada *offspring* nilai tersebut ditambahkan.

Contoh :

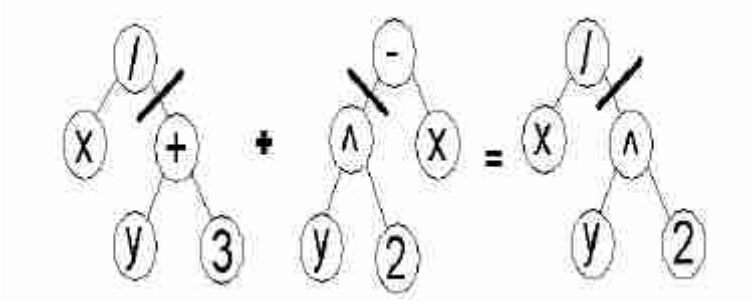
(**12345**6789) + (453**689**721) = **12345689**

3) *Value Encoding*

Semua metode *crossover* pada *binary crossover* bisa digunakan.

4) *Tree Encoding*

Memilih satu titik tertentu dari tiap induk, dan menggabungkan *tree* dibawah titik pada induk pertama dan *tree* di bawah titik pada induk kedua.



Gambar 2.5 Contoh *Tree Encoding* pada *Crossover*

2.1.3.6 Kondisi Berhenti

Offspring merupakan kromosom baru yang dihasilkan setelah melalui proses-proses di atas. Kemudian pada *offspring* tersebut dihitung fitnessnya apakah sudah optimal atau belum, jika sudah optimal berarti *offspring* tersebut merupakan solusi optimal, tetapi jika belum optimal maka akan diseleksi kembali, begitu seterusnya sampai terpenuhi kriteria berhenti.

2.2 Perkembangan Metode Penjadwalan

Sekarang ini banyak ditemukan metode dan algoritma-algoritma yang dibuat untuk tujuan memecahkan persoalan-persoalan yang ada. Kemudian pada perkembangannya metode atau algoritma tersebut mulai diterapkan untuk memecahkan persoalan penjadwalan, antara lain algoritma semut atau *Ant Colony Optimization* (ACO) dengan pendekatan *Max Min Ant System* (MMAS), *Taboo Search*, dan teknik pewarnaan graf (*Coloring Graph*).

2.2.1 *Ant Colony Optimization*

Ant Colony Optimization (ACO) terinspirasi oleh koloni-koloni semut dalam mencari makan. Semut-semut tersebut meninggalkan zat (*pheromone*) di jalan yang mereka lalui. Algoritma ACO ini merupakan algoritma pencarian berdasarkan probabilistik berbobot, sehingga butir pencarian dengan bobot yang lebih besar akan berakibat memiliki kemungkinan terpilih yang lebih besar pula [6].

2.2.2 *Tabu Search*

Tabu Search adalah salah satu metode metaheuristik yang dipergunakan untuk memecahkan permasalahan-permasalahan optimasi global. *Tabu Search* merupakan suatu teknik optimasi yang menggunakan *short-term memory* untuk menjaga agar proses pencarian tidak terjebak pada nilai optimum lokal. Metode ini menggunakan *Tabu List* untuk menyimpan sekumpulan solusi yang baru saja dievaluasi. Selama proses optimasi, pada setiap iterasi solusi yang akan dievaluasi akan dicocokkan terlebih dahulu dengan isi *Tabu List* untuk melihat apakah solusi tersebut sudah ada pada *Tabu List*. Apabila solusi tersebut sudah ada, maka akan dievaluasi lagi pada iterasi berikutnya. Kemudian bila sudah tidak ada lagi solusi yang menjadi anggota *Tabu List*, maka nilai terbaik yang baru saja diperoleh merupakan solusi yang sebenarnya [1].

2.2.3 *Coloring Graph*

Teknik pewarnaan graf merupakan salah satu subjek yang menarik dan terkenal dalam bidang graf. Teori-teori mengenainya telah banyak dikembangkan dan berbagai algoritma dengan kelebihan dan kelemahan masing-masing telah dibuat untuk menyelesaikannya. Aplikasi dari teknik ini juga telah banyak diterapkan di berbagai bidang, salah satunya adalah membuat jadwal. Perencanaan jadwal disini khususnya diterapkan pada pekerjaan-pekerjaan atau hal-hal yang saling terkait, misalnya hal-hal yang berlangsung pada waktu yang sama, atau pekerjaan yang menggunakan sumber daya yang sama, dan sebagainya. Teknik pewarnaan graf akan membuat jadwal kerja yang dapat menghasilkan hasil yang maksimum dengan cara yang paling efisien [3].

2.3 Basis Data

Pada bagian ini akan dibahas mengenai definisi dari basis data, operasi dasar basis data, model data yang dipergunakan dalam basis data, serta model keterhubungan antar entitas-entitas yang ada.

2.3.1 Definisi Basis Data

Basis data terdiri dari dua kata, basis dapat diartikan sebagai markas/ gudang, tempat bersarang/ berkumpul, sedangkan data adalah representasi dari fakta "dunia nyata" yang memiliki suatu objek, seperti manusia (pegawai, siswa, pembeli, pelanggan, dll), barang, hewan, peristiwa, konsep, keadaan, dan sebagainya yang direkam atau disimpan dalam bentuk angka, huruf, symbol, teks, gambar, bunyi, atau kombinasinya.

Basis data dapat didefinisikan dalam sejumlah sudut pandang, seperti :

a. Himpunan kelompok data (arsip) yang saling berhubungan dan diorganisasikan sedemikian rupa agar nantinya dapat dimanfaatkan dengan cepat dan mudah.
b. Kumpulan data yang saling berhubungan dan disimpan secara bersama sedemikian rupa dan tanpa perulangan (*redundancy*) yang tidak perlu untuk memenuhi berbagai kebutuhan.
c. Kumpulan file/table/arsip yang saling berhubungan dan disimpan dalam media penyimpanan elektronik.

2.3.2 Operasi Dasar Basis Data

Di dalam sebuah *disk*, basis data dapat dibuat dan dapat pula ditiadakan. Dalam sebuah basis data, kita dapat menempatkan satu atau lebih file atau tabel. Pada tabel inilah sesungguhnya data disimpan atau ditempatkan. Setiap basis data pada umumnya dibuat untuk mewakili sebuah semesta data yang spesifik, misalkan basis data kepegawaian, basis data akademik, basis data *inventory* (pergudangan), dan sebagainya.

Operasi-operasi yang dapat dilakukan berkenaan dengan basis data meliputi :

a. Pembuatan basis data baru (*create database*)
b. Penghapusan basis data (*drop database*)
c. Pembuatan file atau tabel (*create table*)
d. Penghapusan file atau tabel (*drop table*)
e. Penambahan atau pengisian data baru ke dalam tabel suatu basis data (*insert*)
f. Menampilkan data dari sebuah file atau tabel (*select*)
g. Pengambilan data dari sebuah file atau tabel (*retrieve/ search*)
h. Pengubahan data dari sebuah file atau tabel (*update*)
i. Penghapusan data sebuah file atau tabel (*delete*)

2.3.3 Model Data

Model data dapat didefinisikan sebagai kumpulan perangkat konseptual untuk menggambarkan data, hubungan data, *semantic* (makna) data, dan batasan data. Karena yang ingin ditunjukan adalah makna dari data dan keterhubungan dengan data lain, maka model data ini lebih tepat jika disebut "*model data logic*".

Ada sejumlah cara dalam merepresentasikan model data dalam perancangan basis data, yang secara umum dibagi ke dalam dua kelompok.

1) Model data berdasar objek
    a. Model keterhubungan entitas (*entity-relational ship model*)
    b. Model berorientasi objek (*object oriented model*)
    c. Model data semantik (*semantic data model*)
    d. Model data fungsional (*functional data model*)
2) Model data berdasar *record*
    a. Model relasional (*relational model*)
    b. Model hirarki (*hierarchical model*)
    c. Model jaringan (*network model*)

2.3.4 Model Keterhubungan Entitas

Pada model ER, semesta data yang ada di “dunia nyata” diterjemahkan/ ditransformasikan dengan memanfaatkan sejumlah perangkat konseptual menjadi sebuah diagram data yang umum disebut dengan ER. Terdapat dua komponen utama pembentuk model ER, yaitu entitas (*entity*) dan relasi (*relation*), kedua komponen ini akan dideskripsikan lebih lanjut melalui sejumlah atribut/ properti.

Entitas merupakan sebuah individu yang mewakili sesuatu yang nyata eksistensinya dan dapat dibedakan dari sesuatu yang lain. Sekelompok entitas yang sejenis dan berada dalam lingkup yang sama membentuk sebuah himpunan entitas (*entity set*), akan tetapi pada beberapa literatur entitas disebutkan sebagai entitas saja.

Sederhananya entitas menunjuk pada individu suatu objek, sedangkan himpunan entitas menunjuk pada famili dari individu tersebut. Perbedaan antara entitas dan himpunan entitas dapat terlihat sebagai contoh yang menjadi himpunan entitas misalnya adalah semua pelanggan atau pelanggan saja, sedangkan entitasnya adalah Ihsan, Rissa, dan Ujang.

Tiap entitas memiliki atribut yang mendeskripsikan karakter untuk properti dari entitas tersebut. Penetapan atribut-atribut yang relevan bagi seluruh entitas merupakan hal penting dalam pembentukan model data. Penetapan atribut bagi sebuah entitas umumnya memang didasarkan pada fakta yang ada, tetapi tidak selalu seperti itu. Adapun jenis-jenis atribut antara lain :

1) *Key* dan atribut deskriptif

   Key adalah satu atau gabungan dari beberapa atribut yang dapat membedakan semua baris data (*row*) dalam tabel secara unik. Artinya jika suatu atribut dijadikan *key*, tidak boleh ada dua atau lebih baris data dengan nilai yang sama untuk atribut tersebut. Sedangkan atribut yang tidak termasuk sebagai *key* disebut atribut deskriptif.

2) Atribut sederhana dan atribut komposit

   Atribut sederhana adalah atribut yang tidak dapat dipilih lagi. Sedangkan atribut komposit adalah atribut yang masih dapat diuraikan lagi menjadi sub-sub atribut yang masing-masing atribut memiliki makna.

3) Atribut bernilai tunggal dan atribut bernilai banyak

   Atribut bernilai tunggal ditunjukan pada atribut yang memiliki paling banyak satu nilai untuk setiap baris data. Sedangkan atribut bernilai banyak ditunjukan pada atribut yang dapat kita isi dengan lebih dari satu nilai, tetapi jenisnya sama.

4) Atribut harus bernilai dan atribut bernilai *null*

   Ada sejumlah atribut pada suatu tabel yang ditetapkan harus berisi data jadi nilainya tidak boleh kosong, atribut-atribut semacam itu disebut *mandatory attribute*. Sebaliknya terdapat pula atribut yang boleh dikosongkan (*non mandatory attribute*). Nilai konstanta *null* digunakan untuk mengisi atribut-atribut yang nilainya belum ada atau tidak ada.

5) Atribut turunan

   Atribut turunan adalah atribut yang nilainya diperoleh dari pengolahan atau dapat diturunkan dari atribut atau tabel lain yang berhubungan. Atribut demikian sebenarnya dapat dihilangkan dari sebuah tabel karena nilai-nilainya bergantung pada nilai yang ada di atribut lain.

Relasi menunjukan adanya hubungan diantara sejumlah entitas yang berasal dari himpunan entitas yang berbeda. Misalnya, entitas seorang mahasiswa dengan nim=’10103413’ dan nama_mahasiswa=’Ihsan Sani Abdullah’ (yang ada pada himpunan entitas mahasiswa) mempunyai relasi dengan entitas sebuah mata kuliah dengan kode_mata_kuliah=’IF33216’ dan nama_kuliah=’Logika Matematika’. Relasi diantara kedua entitas tadi mengandung arti bahwa mahasiswa Ihsan Sani Abdullah dengan NIM 10103413 mengambil mata kuliah Logika Matematika dengan kode mata kuliah IF33216.

2.3.5 Nilai Kardinalitas

Dalam model ER dikenal nilai kardinalitas atau derajat relasi, yaitu nilai yang menujukan jumlah maksimum entitas yang dapat berelasi dengan entitas pada himpunan entitas yang lain. Derajat relasi yang diantara dua himpunan entitas (misalnya A dan B) dapat berupa :

1) Satu ke satu (*one to one*)

   Yaitu satu *entity* dalam A dihubungkan dengan maksimum satu *entity* dalam B.

2) Satu ke banyak (*one to many*)

   Yaitu satu *entity* dalam A dihubungkan dengan sejumlah *entity* dalam B. Satu *entity* dalam B dihubungkan dengan maksimum satu *entity* dalam A.

3) Banyak ke satu (*many to one*)

   Yaitu satu *entity* dalam A dihubungkan dengan maksimum satu *entity* dalam B. Satu *entity* dalam B dapat dihubungkan dengan sejumlah *entity* dalam A.

4) Banyak ke banyak (*many to many*)

   Yaitu sejumlah *entity* dalam A dihubungkan dengan sejumlah *entity* dalam B.