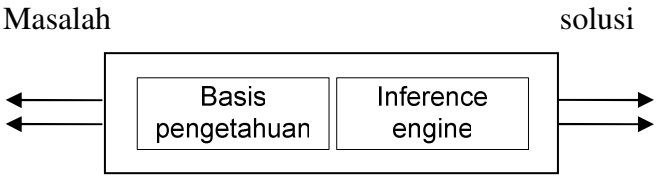
BAB II

LANDASAN TEORI

Seperti yang telah kita ketahui bahwa pada sistem yang menggunakan kecerdasan (*artificial intelligence*) akan mencoba memberikan output berupa solusi dari suatu masalah berdasarkan kumpulan pengetahuan yang telah ada.



Gambar 2.1 sistem yang menggambarkan kecerdasan buatan

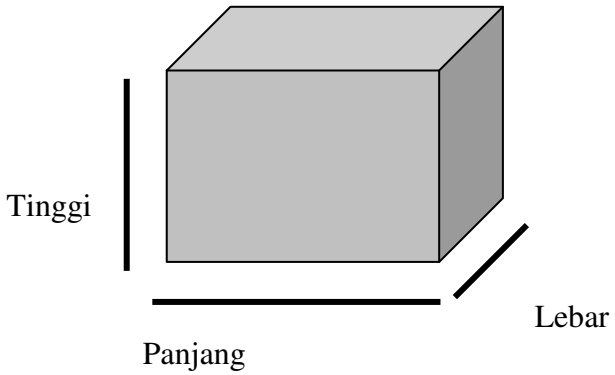
Pada gambar 2.1 input yang diberikan pada sistem yang menggunakan kecerdasan buatan berupa masalah. Pada sistem harus dilengkapi dengan sekumpulan pengetahuan yang ada pada basis pengetahuan. Sistem harus memiliki *inference engine* agar mampu mengambil kesimpulan berdasarkan fakta/pengetahuan. Output yang diberikan berupa solusi dari masalah sebagai *inference*.

Teori-teori yang menjadi landasan pada penulisan tugas akhir ini adalah teori algoritma genetika, teori pengaturan box, dan teori pengembangan perangkat lunak. Beberapa konsep-konsep dari teori-teori ini yang berhubungan dengan tugas akhir ini akan dijelaskan pada subbab-subab selanjutnya.

2.1 Pengaturan Box

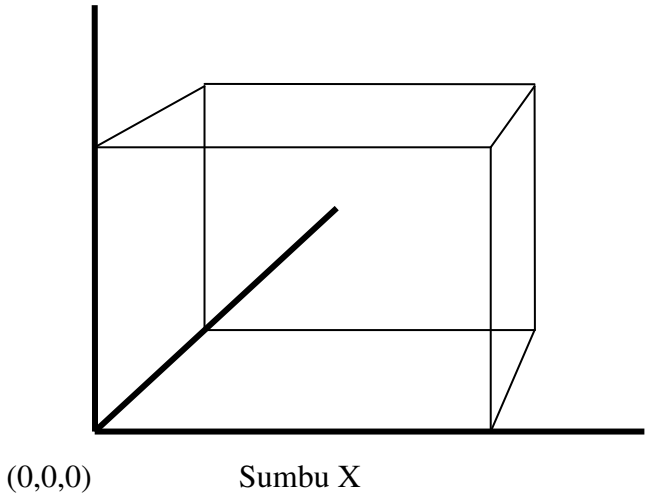
Pengaturan box adalah masalah umum yang sering dihadapi dalam kehidupan sehari-hari. Dalam permasalahan ini box selalu berada dalam suatu

kontainer (tidak terletak diruang bebas). Box yang berbentuk kotak memiliki panjang, lebar dan tinggi yang dapat diilustrasikan pada gambar 2.2. Dalam penulisan untuk box pada kontainer, panjang akan selalu berasosiasi dengan sumbu X, lebar akan selalu berasosiasi dengan sumbu Y dan tinggi akan selalu berasosiasi dengan sumbu Z.



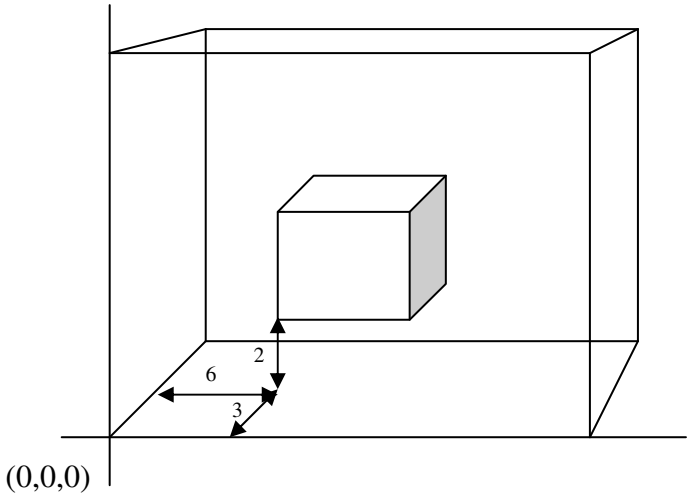
Gambar 2.2 Box berbentuk kotak

Kontainer akan berbentuk kotak yang salah satu sudut dari alasnya berimpit dengan titik (0, 0, 0), hal ini diilustrasikan pada gambar 2.3.



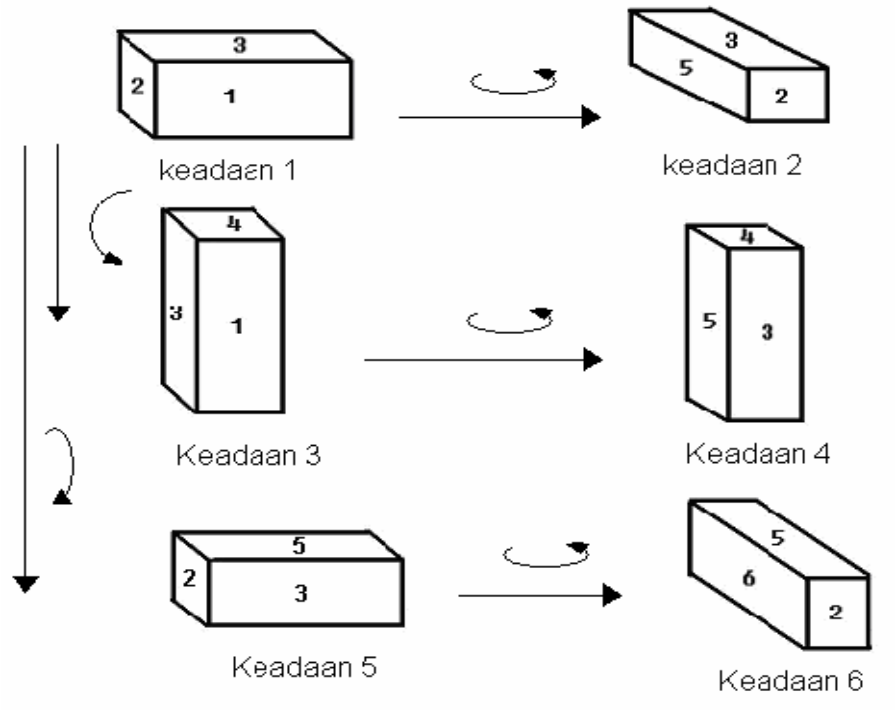
Gambar 2.3 Kontainer berbentuk kotak

Hal ini membuat suatu kotak yang dikatakan berada pada posisi (6, 3, 2) dalam suatu kontainer akan berada pada posisi yang diilustrasikan pada gambar 2.4.



Gambar 2.4 box pada posisi (6,3,2) dalam suatu kontainer

Karena dalam tugas akhir ini diasumsikan bahwa box bisa dibolak-balik maka box tersebut karena berbentuk kotak bisa memiliki enam kemungkinan keadaan dalam peletakannnya yaitu seperti yang diilustrasikan oleh gambar 2.5



Gambar 2.5 Kemungkinan keadaan Box

2.1.1 Tujuan Pengaturan Box

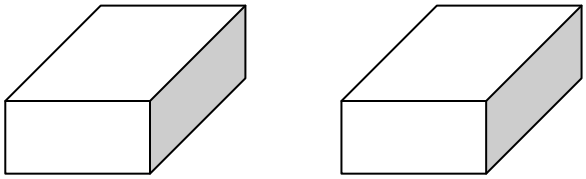
Beberapa tujuan dalam melakukan pengaturan box adalah:

- Menghasilkan susunan box yang sesuai dengan kategori tertentu. Misalnya box yang berat diletakkan di bawah box yang lebih ringan, box yang sering dipakai diletakkan lebih tatas pada suatu tumpukan.
- Menghasilkan susunan box yang padat yaitu sususan box yang lebih mementingkan pada optimasi ruang yang digunakan oleh susunan box tersebut.

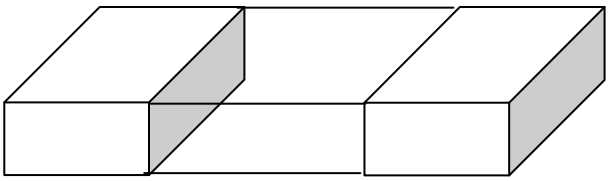
Tugas akhir ini akan menggunakan tujuan yang kedua, yaitu untuk menghasilkan susunan box yang menggunakan sedikit mungkin ruang dengan menggunakan perangkat lunak bantu.

2.1.2 Penilaian Tingkat Kepadatan Susunan Box

Cara untuk mengukur tingkat kepadatan susunan box adalah dengan menghitung volume dari “kotak imajiner”. “kotak imajiner” yang dimaksud adalah sebuah kotak terkecil yang mungkin dibuat, yang dapat menampung seluruh box. Misalkan suatu susunan box seperti pada gambar 2.6 (a), maka suatu kotak terkecil yang mungkin dibuat, yang dapat menampung seluruh box adalah dapat diilustrasikan pada gambar 2.6(b).



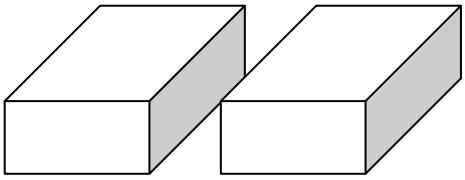
(a) susunan box



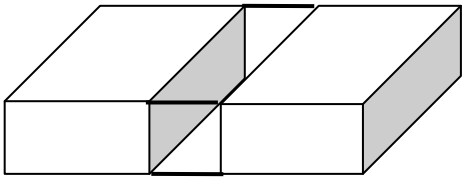
(b) Susunan box (dengan kotak imajiner)

Gambar 2.6 Contoh kotak imajiner

Suatu susunan yang berbeda untuk kedua box tersebut diperlihatkan pada gambar 2.7. Dari kedua gambar ini terlihat bahwa kotak imajiner yang dibentuk oleh susunan box gambar 2.7 mempunyai volume yang lebih kecil dari kotak imajiner yang dibentuk oleh susunan box gambar 2.6, dan dari pengetahuan umum telah terbukti bahwa solusi gambar 2.7 memang lebih baik dari solusi gambar 2.6, sehingga dapat diambil kesimpulan bahwa volume kotak imajiner berkorelasi dengan tingkat kepadatan dari susunan box, yaitu semakin kecil kotak imajiner maka semakin padat susunan suatu box.



(a) susunan box



(c) Susunan box (dengan kotak imajiner)

Gambar 2.7 Contoh kotak imajiner

2.2 Algoritma Genetika

Algoritma genetika (biasa disingkat GA) adalah suatu metode optimasi dan simulasi yang perkembangannya tidak lepas dari paradigma evolusi yang ditunjukkan pertama kali oleh Charles Darwin (1809-1882) yang diteruskan dalam perkembangan teori genetika. Evolusi Darwinian yang berbasis pada konsep "*survival of the fittest"* menyatakan bahwa evolusi jenis-jenis spesies mahluk hidup dan ekosistemnya terjadi karen seleksi alam. Algoritma genetika juga merupakan algoritma pencarian heuristik yang didasarkan atas mekanisme evolusi biologis. Keberagaman pada evolusi biologis adalah variasi dari kromosom antar individu organisme. Variasi kromosom ini akan mempengaruhi laju reproduksi dan tingkat kemampuan organisme untuk tetap hidup. Pada dasarnya ada empat kondisi yang sangat mempengaruhi proses evolusi, yaitu:

a. Kemampuan organisme untuk melakukan reproduksi.

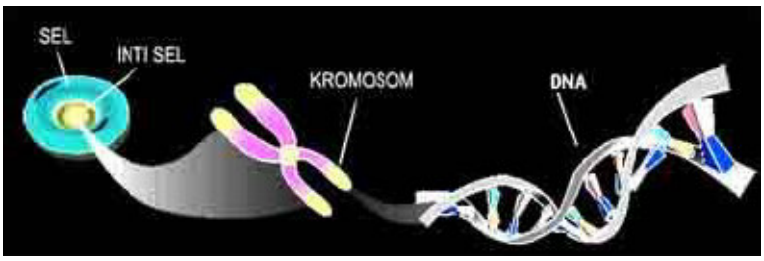
b. Keberadaan populasi organisme yang bisa melakukan reproduksi.

c. Keberagaman organisme dalam suatu populasi.

d. Perbedaan kemampuan untuk *survive.*

Individu yang lebih kuat (fit) akan memiliki tingkat kemampuan survival dan tingkat reproduksi yang lebih tinggi jika dibandingkan dengan individu yang kurang fit. Pada kurun waktu tertentu (sering dikenal dengan istilah generasi), populasi secara keseluruhan akan lebih banyak memuat organisme yang fit.

Algoritma genetika pertama kali dikembangkan oleh John Holland dari Universitas Michigan (1975). John Hollan mengatakan bahwa setiap masalah yang berbentuk adaptasi (alami maupun buatan) dapat diformulasikan dalam teminologi genetika. Algoritma Genetika adalah simulasi dari proses evolusi Darwin dan operasi genetika atas kromosom.



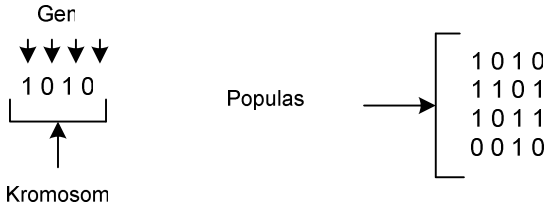
Gambar 2.8 Struktur Genetika

Teori genetika memodelkan sifat-sifat bawaan mahluk hidup dari gen-gen yang diwariskan dari suatu mahluk hidup kepada keturunannya. Setiap mahluk hidup mempunyai kumpulan gen yang jumlahnya untuk setiap spesies adalah sama. Sekumpulan gen atau kromosom mengandung cetak biru dan sifat suatu mahluk hidup secara terdistribusi (Gambar 2.8). Perubahan variasi dari kumpulan

gen dalam kromosom, baik melalui perkawinan atau mutasi adalah basis dari perubahan sifat dasar spesies.

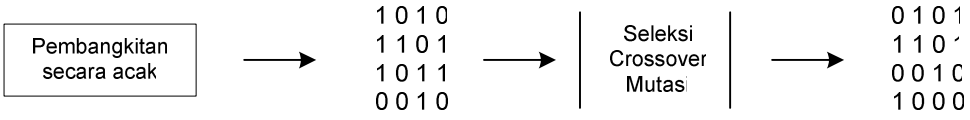
2.2.1 Struktur Umum Algoritma Genetika

Pada algoritma ini, teknik pencarian dilakukan sekaligus atau sejumlah solusi, sekumpulan solusi ini dikenal dengan istilah **populasi**. Individu yang terdapat dalam suatu populasi disebut dengan istilah **kromosom**. Hubungan antara gen, popualsi dan kromosom dapat dilihat pada gambar 2.9.



Gambar 2.9 Gen, Kromosom dan Populasi

Kromosom ini merupakan suatu solusi yang masih berbentuk simbol. Populasi awal dibangun secara acak, sedangkan populasi berikutnya merupakan hasil evolusi kromosom-kromosom melalui iterasi yang disebut dengan istilah **Generasi**. Generasi dapat diilustrasikan pada gambar 2.10.



Gambar 2.10 Generasi

Pada setiap generasi, kromosom akan melalui proses evaluasi dengan menggunkan alat ukur yang disebut dengan **fungsi *fitness*** atau **fungsi evaluasi**.

Nilai *fitness* dari suatu kromosom akan menunjukkan kualitas kromosom dalam populasi tersebut. Generasi berikutnya dikenal dengan istilah anak (***offspring***) terbentuk dari gabungan dua kromosom generasi sekarang yang bertindak sebagai induk (***parent)*** dengan menggunakan operator penyilangan (***crossover).*** Selain operator penyilangan, suatu kromosom dapat juga dimodifikasi dengan menggunakan operator **mutasi.** Populasi generasi yang baru dibentuk dengan cara menyeleksi nilai fitnness dari kromosom induk (*parent*) dan nilai fitness dari kromosom anak (*offspring*), serta menolak kromosom-kromosom yang lainnnya sehingga ukuran populasi (jumlah kromosom dalam suatu populasi) konstan. Setelah melalui beberapa generasi, maka algoritma ini akan konvergen ke kromosom terbaik.

2.2.2 Komponen-Komponen Utama Algoritma Genetika

Ada 6 komponen utam dalam algoritma genetika, yaitu:

a. Teknik Penyandian

Teknik penyandian disini meliputi penyandian gen dan kromosom. Gen merupakan bagian dari kromosom. Satu gen biasanya akan mewakili satu variabel.

Gen dapat direpresentasikan dalam bentuk *string* bit, pohon, *array* bilangan real, daftar aturan, elemen permutasi, elemen program, atau representasi lainnya yang dapat diimplementasikan untuk operator genetika.

Bentuk-bentuk representasi kromosom:

- *String* bit : 10011, 01101, 11101, dst.
- Bilangan real : 65.65, -67.98, 562.88, dst.

- Elemen permutasi : E2, E10, E5, dst.
- Daftar Aturan : R1, R2, R3, dst.
- Elemen Program : pemrograman genetika
- Struktur lainnya

b. Prosedur Inisialisasi

Proises inisialisasi adalah proses yang bertujuan untuk membangkitkan generasi awal yang akan menjadi dasar untuk membangkitkan generasi-generasi selanjutnya. Ukuran populasi tergantung pada masalah yang akan dipecahkan dan jenis operator genetika yang akan diimplementasikan. Setelah ukuran populasi ditentukan, kemudian harus dilakukan inisialisasi terhadap kromosom yang terdapat pada populasi tersebut. Inisialisasi kromosom dilakukan secara acak, namun demikian harus tetap memperhatikan domain solusi dan kendala permasalahan yang ada.

c. Fungsi Evaluasi

Ada dua hal yang harus dilakukan dalam mengevaluasi kromosom, yaitu evalusi objektif (fungsi tujuan) dan konversi fungsi objektif kedalam fungsi *fitness*. Secara umum, fungsi *fitness* diturunkan dari fungsi objektif dengan nilai yang negatif. Apabila ternyata fungsi objektif memiliki nilai negatif, maka perlu ditambahkan suatu konstanta C agar nilai *fitness* yang tebentuk menjadi tidak negatif.

d. Seleksi

Seleksi ini bertujuan untuk memberikan kesempatan reproduksi yang lebih besar bagi anggota populasi yang paling fit.

Ada beberapa metode seleksi dari induk, antara lain:

1. *Rank-based fitness assigment*
2. *Roulette wheel selection*
3. *Stochastic universal sampling*
4. *Local Selecyion*
5. *Truncation selection*
6. *Toutnament selection*

e. Operator Genetika

Ada dua operator genetika, yaitu:

1. Operator untuk melakukan rekombinasi, yang terdiri atas:
    - Rekombinasi bernilai real.
        - Rekombinasi diskret
        - Rekombinasi *intermediate* (menengah)
        - Rekombinasi garis
        - Rekombinasi garis yang diperluas
    - Rekombinasi bernilai biner (*crossover*)
        - *Crossover* satu titik
        - *Crossover* banyak titik
        - *Crossover* seragam
        - *Crossover* dengan permutasi
2. Mutasi :
    - Mutasi berniali real
    - Mutasi bernilai biner

f. Penentuan Parameter GA

Yang disebut dengan parameter disini adalah parameter kontrol algoritma genetika, yaitu ukuran populasi (*popsize*), peluang *crossover* (P_C) dan peluang mutasi (P_m). Nilai parameter ini ditentukan juga berdasarkan permasalahan yang akan dipecahkan. Ada beberapa rekomendasi yang akan digunakan, antara lain:

- Untuk permasalahan yang memiliki kawasan solusi cukup besar, De Jong merekomendasikan untuk nilai parameter control:

  (*popsize*; $P_{C;}$ P_m) = (50; 0,6; 0,001)

- Bila rata-rata *fitness* setiap generasi digunakan sebagai indikator, maka Grefenstette merekomendasikan:

  (*popsize*; $P_{C;}$ P_m) = (30; 0,95; 0,01)

- Bila *fitness* dari individu terbaik dipantau pada setiap generasi, maka usulannnya adalah:

  (*popsize*; $P_{C;}$ P_m) = (80; 0,45; 0,01)

- Ukuran populasi sebaiknya tidak lebih kecil dari 30, untuk sembarang jenis permasalahan.

2.2.3 Seleksi

Seleksi untuk menentukan individu-individu mana saja yang akan dipilih untuk dilakukan rekombinasi dan bagaimana *offspring* terbentuk dari individu-individu terpilih tersebut. Langkah pertama yang akan dilakukan 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-tahap seleksi berikutnya.

Metode-metode yang digunakan untuk melakukan seleksi antara lain adalah sebagai berikut:

a. Rank-Based *fitness*

Pada *rank-based* fitness, populasi diturunkan menurut nilai objektifnya. Nilai *fitness* dari tiap-tiap individu hanya tergantung pada posisi individu tersebut dalam urutan dan tidak dipengaruhi oleh nilai objektifnya.

b. Seleksi Roda *Roulette (roulette Wheel Selection)*

Metode seleksi roda *roulette* ini merupakan metode yang paling sederhana, dan sering juga dikenal dengan nama *stochastic sampling with replacement*. Pada metode ini, individu-individu dipetakan dalam suatu segmen garis secara berurutan sedemikian hingga tiap-tiap segmen individu memiliki ukuran sama dengan ukuran *fitness*-nya. Sebuah bilangan acak dibangkitkan dan individu yang memiliki segmen dalam kawasan bilangan acak tersebut akan terseleksi. Proses ini diulang hingga diperoleh sejumlah individu yang diharapkan.

Pada Tabel 2.1 menunjukan probabilitas seleksi dari 11 individu. Individu pertama memiliki *fitness* terbesar, dengan demikian dia juga memiliki interval terbesar. Sedangkan individu ke-10 memiliki *fitness* terkecil kedua. Individu ke-11 memiliki *fitness* terkecil sehingga tidak memiliki kesempatan untuk melakukan reproduksi.

Tabel 2.1 Probabilitas seleksi & nilai *fitness*

Individu ke-	1	2	3	4	5	6	7	8	9	10	11
Nilai *fitness*	2.0	1.8	1.6	1.4	1.2	1.0	0.8	0.6	0.4	0.2	0.0
Probabilitas seleksi	0.18	0.16	0.15	0.13	0.11	0.09	0.07	0.06	0.03	0.02	0.0

c. *Stochastic Universal Sampling*

Stochastic Universal Sampling memiliki nilai bisa nol dan penyebaran yang minimum. Pada metode ini, individu-individu dipetakan dalam suatu segmen garis secara berurutan sedemikian sehingga tiap-tiap segmen individu memiliki ukuran yang sama dengan ukuran *fitness*-nya seperti halnya pada seleksi roda *roulette*. Kemudian diberikan sejumlah pointer sebanyak individu yang ingin diseleksi pada garis tersebut. Misalkan N adalah jumlah individu yang akan diseleksi, maka jarak antar pointer adalah 1/N dan posisi pointer pertama diberikan secara acak pada *range* [1. 1/N].

d. Seleksi Lokal (*Local Selection*)

Pada seleksi lokal, setiap individu yang berada di dalam konstrain tertentu disebut dengan nama lingkungan lokal. Interaksi antarindividu hanya dilakukan diwilayah tersebut. Lingkungan tersebut ditetapkan sebagai struktur dimana populasi tersebut terdistribusi. Lingkungan tersebut juga dapat dipandang sebagai kelompok pasangan-pasangan yang potensial.

Langkah pertama adalah menyeleksi separo pertama dari populasi yang berpasangan secara acak (atau menggunakan salah satu metode yang sudah dijelaskan sebelumnya, seperti : ***Stochastic Universal Sampling***). Kemudian

lingkungan baru tersebut diberikan pada setiap individu yang terseleksi. Pada lingkungan yang baru tersebut, bisa diseleksi pasangan-pasangan yang cocok (pasangan yang terbaik, pasangan yang memiliki *fitness* proporsional, atau pasangan yang seragam). Struktur lungkungan pada seleksi lokal ini dapat berbentuk:

- Linear : *full ring* dan *half ring*
- Dimensi-dua:
    - *Full cross* dan *half cross*
    - *Full star* dan *half star*
- Dimensi-tiga dan struktur yang lebih kompleks yang merupakan kombinasi dari kedua struktur di atas.

Jarak antara individu dengan struktur tersebut akan sangat menentukan ukuran lingkungan (jumlah tetangga dari suatu individu). Hal ini dapat dilihat pada tabel 2.2. jarak antara satu individu dengan individu yang lainnnya tentu saja akan terbatas oleh ukuran lingkungan suatu individu. Individu yang terdapat dalam lingkungan dengan ukuran yang lebih kecil, tentu saja akan lebih terisolasi jika dibandingkan dengan individu yang terletak pada lingkungan dengan ukuran yang lebih besar.

Tabel 2.2 Jumlah tetangga (ukuran lingkungan) untuk seleksi lokal

Struktur	Jarak	
	1	**2**
Full ring	2	4
Half ring	1	2
Full cross	4	8(12)
Half cross	2	4(5)
Full star	8	24
Half star	3	8

e. Seleksi dengan Pemotongan (*Truncation Selection*)

Pada metode-metode seleksi yang telah dijelaskan terdahulu, seleksi dilakukan secara alami. Pada seleksi dengan pemotongan ini, lebih berkesan sebagai seleksi buatan. Seleksi ini biasanya digunakan oleh populasi yang jumlahnya sangat besar. Pada metode ini, individu-individu diurutkan berdasarkan nilai *fitness*-nya. Hanya individu-individu terbaik saja yang akan diseleksi sebagai induk. Parameter yang digunakan dalam metode ini adalah suatu nilai ambang *trunc* yang mengindikasikan ukuran populasi yang akan diseleksi sebagai induk yang berkisar antara 50%-10%. Individu-individu yang ada dibawah nilai ambang ini tidak akan menghasilkan keturunan.

f. Seleksi dengan Turnamen (*Tournament Selection*)

Pada metode seleksi dengan turnamen ini, akan ditetapkan suatu nilai *tour* untuk individu-individu yang dipilih secara acak dari suatu populasi. Individu-individu yang terbaik dalam kelompok ini akan diseleksi sebagai induk. Parameter

yang digunakan pada metode ini adalah ukuran tour yang bernilai antara dua sampai N (jumlah individu dalam suatu populasi).

2.2.4 Rekombinasi

Dalam melakukan rekombinasi dilakukan pembuatan individu-individu baru yang merupakan hasil kombinasi dari kromosom *parent*. Beberapa metode yang bisa digunakan adalah:

a. Rekombinasi Diskret

Rekombinasi diskret akan menukar nilai variabel antarkromosom induk. Misalkan ada dua individu dengan tiga variabel.

Induk 1: 12 25 5

Induk 2: 123 4 34

Untuk tiap-tiap variabel yang menyumbangkan variabelnya ke anak dipilih secara acak dengan probabilitas yang sama.

Sampel 1: 2 2 1

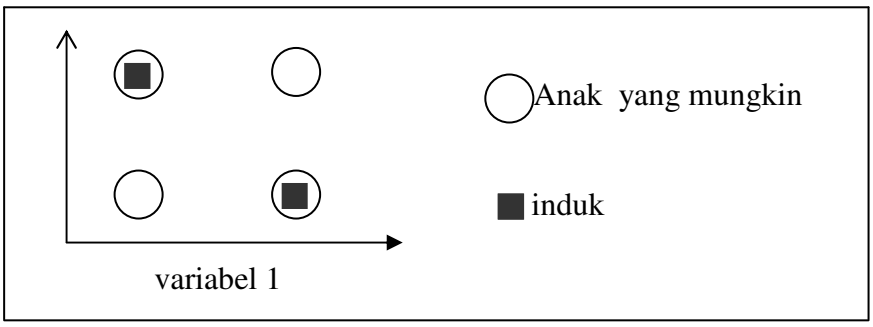
Sampel 2: 1 2 1

Setelah rekombinasi, kromosom-kromosom baru yang terbentuk:

Anak 1: 123 4 5

Anak 2: 12 4 5

Gambar 2.11 menunjukkan posisi yang mungkin dari anak setelah rekombinasi diskret.



Gambar 2.11 Rekombinasi Diskret

Rekombinasi ini dapat digunakan untuk sembarang variabel (biner, real, atau simbol).

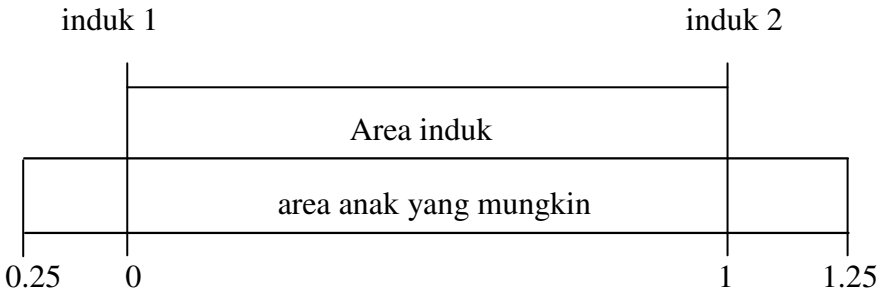
b. Rekombinasi Menengah

Rekombinasi menengah merupakan metode rekombinasi yang hanya dapat digunakan untuk variabel real (dan variabel yang bukan biner). Nilai variabel anak dipilih di sekitar dan antara nilai-nilai variabel induk.

Anak dihasilkan menurut aturan sebagai berikut:

Anak = induk 1 + alpha (induk 2 – induk 1)

Dengan alpha adalah faktor skala yang dipilih secara acak pada interval [-d, 1+d], biasanya d=0,25. tiap-tiap variabel pada anak merupakan hasil kombinasi variabel-variabel menurut aturan di atas dengan nilai alpha dipilih ulang untuk variabel.



Gambar 2.12 Area induk dan anak pada rekombinasi menengah

Gambar 2.12 menunjukkan area induk dan anak yang mungkin. Misalkan ada 2 individu dengan 3 variabel, yaitu:

induk 1: 12 25 5

induk 2: 123 4 34

Misalkan nilai alpha yang terpilih adalah:

sampel 1: 0,5 1.1 -0,1

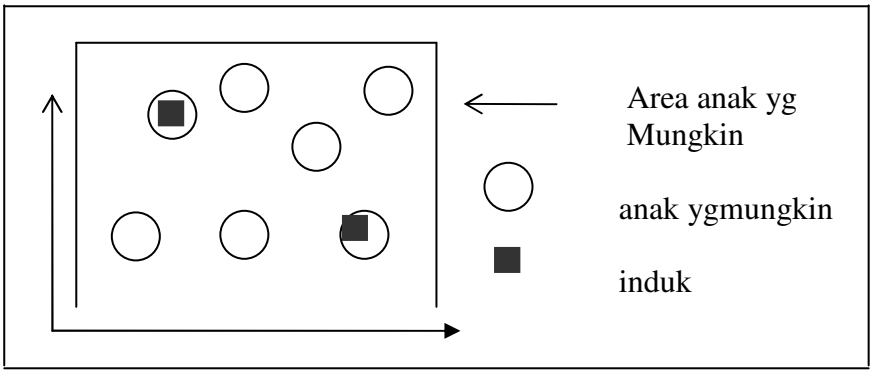
sampel 2: 0,1 0,8 0,5

Setelah rekombinasi, kromosom-kromosom baru yang terbentuk:

anak 1: 67,5 1,9 2,1

anak 2: 23,1 8,2 19,5

Gambar 2.13 menunjukkan posisi yang mungkin dari anak setelah rekomendasi menengah.



Gambar 2.13 Posisi anak yang mungkin pada rekombinasi menengah.

Rekombinasi ini dapat digunakan untuk sembarang variabel (biner, real, atau simbol).

c. Rekombinasi Garis

Pada dasarnya rekombinasi garis ini sama dengann rekombinasi menengah, hanya saja nilai alpha untuk semua variabel sama. Misalkan ada dua kromosom dengan tiga variabel, yaitu:

Induk 1: 12 25 5

Induk 2: 123 4 34

Untuk tiap-tiap variabel induk yang menyumbangkan variabelnya ke anak dipilih secara acak denga probabilitas yang sama.

Sampel 1: 0,5

Sampel 2: 0,1

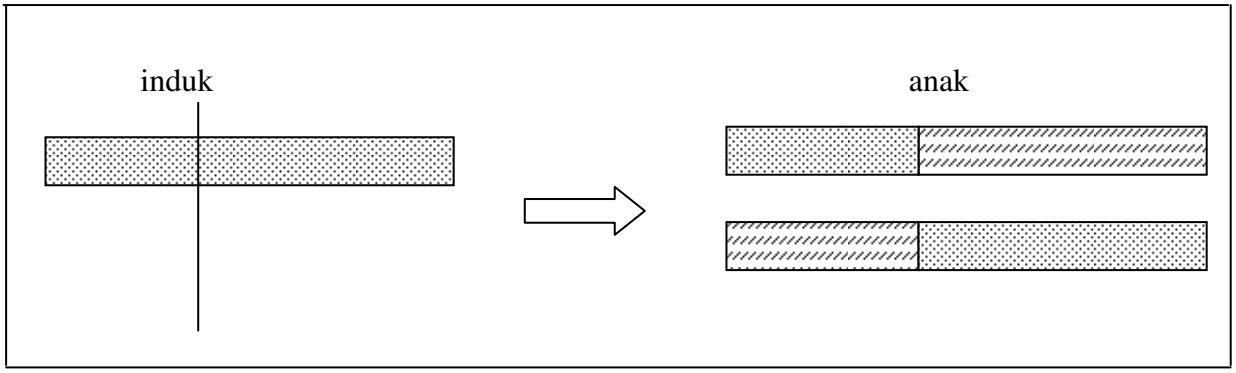
Setelah rekomendasi, kromosom-kromosom baru yang terbentuk:

Anak 1: 67,5 14,5 19,5

Anak 2: 23,1 22,9 7,9

d. Penyilangan Satu Titik (*single-point crossover*)

Pada penyilangan satu titik, posisi penyilangan k (k=1, 2, ..., N-1) dengan N=panjang kromosom diseleksi secara acak. Variabl-variabel ditukar antarkromosom pada titik tersebut untuk menghasilkan anak (Gambar 2.14).



Gambar 2.14 ***Single-point crossover***

Misalkan ada dua kromosom dengan panjang 12:

Induk 1: 0 1 1 1 0 | 0 1 0 1 1 1 0

Induk 2: 1 1 0 1 0 | 0 0 0 1 1 0 1

Posisi penyilangan yang terpilih: misalkan 5

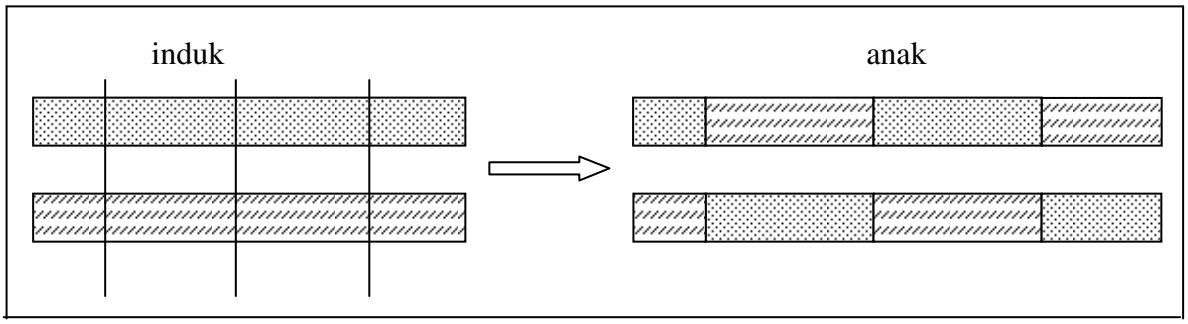
Setelah penyilangan, diperoleh kromosom-kromosom baru:

Anak 1: 0 1 1 1 0 | 0 0 0 1 1 0 1

Anak 2: 1 1 0 1 0 | 0 1 0 1 1 1 0

e. Penyilangan Banyak Titik (*Multi-point crossover*)

Pada penyilangan banyak titik, m posisi penyilangan k_i (k-1,2, ..., N-1, i=1, 2, ..., m) dengan N=panjang kromosom diseleksi secara acak dan tidak diperbolehkan ada posisi yang sama, serta diurutkan naik. Variabel-variabel ditukar antarkromosom pada titik tersebut untuk menghasilkan anak (gambar 2.15).



Gambar 2.15 ***Multi-point crossover***

Misalkan ada dua kromosom dengan panjang 12:

Induk 1: 0 1 1 1 0 0 1 0 1 1 1 0

Induk 2: 1 1 0 1 0 0 0 0 1 1 0 1

Posisi penyilangan yang terpilih :

Misalkan (m=3): 2 6 10

Setelah penyilangan, diperoleh kromosom-kromosom baru:

Anak 1: 0 1 | 0 1 0 0 | 0 0 1 1 | 0 1

Anak 2: 1 1 | 1 1 0 0 | 1 0 1 1 | 1 0

f. Penyilangan Seragam (*Uniform Crossover*)

Pada penyilangan seragam, setiap lokasi memiliki potensi sebagai tempat penyilangan. Sebuah *mask* penyilangan dibuat sepanjang kromosom secara acak yang menunjukkan bit-bit dalam *mask* yang mana induk akan mensupply anak dengan bit-bit yang ada.

Induk mana yang akan menyumbangkan bit ke anak dipilih secara acak dengan probabilitas yang sama. Disini anak -1 akan dihasilkan dari induk – 1 jika bit *mask* bernilai 1, atau sebaliknya, anak – 1 akan dihasilkan dari induk – 2 jika bit *mask* bernilai 0. sedangkan anak – 2 dihasilkan dari kebalikan *mask*.

Misalkan ada dua kromosom dengan panjang 12:

Induk 1: 0 1 1 1 0 0 1 0 1 1 1 0

Induk 2: 1 1 0 1 0 0 0 0 1 1 0 1

Mask bit:

Sampel 1: 1 0 0 1 1 1 0 0 1 1 0 1

Sampel 2: 0 1 1 0 0 0 1 1 0 0 1 0

Setelah penyilangan, diperoleh kromosom-kromosom baru:

Anak 1: 0 1 0 1 0 0 0 0 1 1 0 0

Anak 2: 1 1 1 1 0 0 1 0 1 1 1 1

g. Penyilangan dengan Permutasi (*Permutation Crossover*)

Pada penyilangan dengan permutasi ini, kromosom-kromosom anak diperoleh dengan cara memilih sub-barisan suatu tour dari satu induk dengan tetap menjaga urutan dan posisi sejumlah gen yang mungkin terhadap induk yang lainnnya.

2.2.5 Mutasi

Setelah mengalami proses rekombinasi, pada *offspring* dapat dilakukan mutasi. Variabel *offspring* dimutasi dengan menambahkan nilai acak yang sangat kecil (ukuran langkah mutasi), dengan probabilitas yang rendah. Peluang mutasi (P_m) didefinisikan sebagai presentasi dari jumlah total gen pada populasi yang mengalami mutasi. Peluang mutasi mengendalikan banyaknya gen baru yang akan dimunculkan untuk di evaluasi. Jika peluang mutasi terlalu kecil, banyak gen yang mungkin berguna tidak pernah dievaluasi. Tetapi bila peluang evaluasi ini terlalu besar , maka akan terlalu banyak gangguan acak, sehingga anak akan kehilangan kemiripan dari induknya dan juga algoritma akan kehilangan kemampuan untuk belajar dari histori pencarian. Ada beberapa pendapat mengenai laju mutasi ini. Ada yang berpendapat bahwa, laju mutasi sebesar 1/n akan memberikan hasil yang cukup baik. Ada juga yang beranggapan bahwa laju mutasi tidak tegantung pada ukuran populasinya. Kromosom hasil mutasi harus diperiksa, apakah masih masih berada pada domain solusi, dan bila perlu bisa dilakukan perbaikan.

Mutasi ini berperan untuk menggantikan gen yang hilang dari populasi akibat proses seleksi yang memungkinkan munculnya kembali gen yang tidak muncul pada inisialisasi populasi.

Beberapa metode yang sering digunakan:

a. Mutasi Bilangan Real

Pada mutasi bilangan real, ukuran langkah mutasi biasanya sangat sulit ditentukan. Ukuran yang kecil biasanya sering mengalami kesuksesan, namun kadang ukuran yang lebih besar akan berjalan lebih cepat.

Operator mutasi untuk bilangan ini dapat ditetapkan sebagai:

- Variabel yang dimutasi = variabel ± *range**delta; (+ atau – memiliki probabilitas yang sama).
- *Range* = 0,5 * domain variabel; (interval pencarian)
- Delta = $\Sigma(a_i*2^{-1})$; a_i=1 engan probabilitas 1/m, selain itu a_i =0 dengan m=20.

b. Mutasi Biner

Cara sederhana untuk mendapatkan mutasi biner adalah dengan mengganti satu atau beberapa nilai gen dari kromosom. Langkah – langkah mutasi ini adalah:

- Hitung jumlah gen pada populasi (panjang kromosom dikalikan dengan ukuran populasi)
- Pilih secara acak gen yang akan dimutasi.
- Tentukan kromosom dari gen yang terpilih untuk dimutasi.
- Ganti nilai gen (0 ke 1, atau 1 ke 0) dari kromosom yang akan dimutasi tersebut.

2.3 Alat Bantu Sistem

2.3.1 Diagram Konteks

Diagram konteks merupakan gambaran seluruh proses dan entitas yang terlibat pada suatu sistem, diagram konteks hanya memiliki satu proses atau belum terdapat data store. Dan diagram konteks dapat kita lihat bagian yang berperan dalam aktifitas sistem tersebut, dan juga keterkaitan antara tiap tiap masukan data sampai keluaran.

2.3.1.1 Data Flow Diagram (DFD)

DFD digunakan merupakan sebuah teknik merepresentasikan adanya aliran diagram informasi dan penyimpanan data yang diaplikasikan pada saat data bergerak dari proses awal input menjadi sebuah output. DFD dapat digunakan untuk menyajikan sebuah sistem atau perangkat lunak pada setiap tingkat abstraksi. DFD juga memberikan mekanisme bagi pemodelan fungsional dan pemodelan aliran informasi. Simbol-simbol yang digunakan dalam DFD adalah:

1. Kesatuan luar (*Eksternal Entity*)

   Kesatuan luar merupakan kesatuan dilingkungan luar sistem yang dapat berupa orang atau sisyem lainnya, yang akan memberikan input atau menerima input dari sistem. Kesatuan atau entitas digambarkan dengan kotak.

2. Proses (*Process)*

   Proses digambarkan dengan lingkaran. Proses ini akan mengolah input menjadi output.

3. Arus Data (*Data Flow*)

   Arus data ini menunjukkan arus dari data yang dapat berupa masukan untuk sistem atau hasil dari proses sistem. Arus data ini mengalir diantara proses, simpanan data dan kesatuan luar, dengan menggunakan simbol sebuah panah.

4. Simpanan Data (*Data Store)*

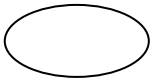
   Jika data perlu dipertahankan karena suatu sebab, maka digunakan media penyimpanan data. Dalam istilah DFD, penyinpanan data adalah suatu penampungan luar. Simapanan data dapat digambarkan sebagai set garis parallel, segiempat terbuka atau bentuk lonjong.

2.3.2 Pengertian Flowchart

Didalam tahapan perancangan sistem, digunakan alat bantu berupa diagram alir (flowchart). Adapun simbol-simbol yang digunakan dalam flowchart tersebut adalah:

1. Simbol *Terminator*

   Simbol yang digunakan untuk menandakan awal dan akhir dari kejadian (event). Gambarnya adalah sebagai berikut:



Gambar 2.16 Simbol terminator dalam flowchart

2. Simbol Proses

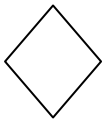
   Simbol yang digunakan untuk menandakan proses yang terjadi. Gambarnya adalah sebagai berikut:



Gambar 2.17 Simbol proses dalam flowchart

3. Simbol Keputusan

   Simbol yang digunakan untuk menandakan suatu kondisi, keputusan / pertanyaan. Gambarnya adalah sebagai berikut:



Gambar 2.18 Simbol keputusan dalam flowchart

4. Simbol Aliran Kejadian

   Simbol yang digunkan untuk menandakan aliran data informasi dari suatu bagian ke bagian lainnya.. gambarnya adalah sebagai berikut:



Gambar 2.19 Simbol Aliran kejadian dalam flowchart

2.3.3 Perangkat Lunak Penunjang

Untuk membuat sebuah program pengaturan box pada kontainer diperlukan terlebih dahulu pemahaman terhadap perangkat lunak yang digunakan yaitu Borland Delphi 7.0.

2.3.3.1 Sekilas Borland Delphi 7.0

2.3.3.1.1 Sejarah Borland Delphi

Ide munculnya Delphi sebenarnya berasal dari bahasa pemrograman yang cukup terkenal yaitu pascal. Bahasa pascal itu sendiri telah diciptakan pada tahun 1971 oleh ilmuan dari Swiss, yaitu Niklous Wirth. Nama Pascal diambil dari ahli matematika dan filsafat dari perancis, yaitu Balise Pascal (1623-1622).

Sejak saat itu, munsul beberapa versi pascal diantaranya Turbo Pascal yang dirilis olah *Borland International Incorforation* tahun 1983. Turbo Pascal ini memiliki sedikit perbedaan dengan Pascal standar, antara lain dalam hal pengolahan string, penambahan beberapa procedure dan fungsi.

Turbo Pascal muncul pertama kali hanya dapat dijalankan di sistem operasi DOS. Namun dalam perkembangan selanjutnya, *Borland International* juga merilis Turbo Pascal yang berjalan pada Windows 3.x, yaitu Turbo Pascal for Dos dan Turbo Pascal for windows menjadi satu paket bahasa pemrograman yang dikenal dengan nama Borland Pascal versi 7. Karena pemrograman windows bahasa pascal masih dirasakan cukup sulit. Sejak tahun 1993 Borland International mengembangkan bahasa Pascal yang bersifat visual. Hasil pengembangan ini adalah dirilisnya Delphi 1 pada tanggal 28 Februari 1995. perkembangan Delphi tidak berhenti sampai disitu saja, satu tahun berikutnya pada tahun 1996 *Borland Internat*ional merilis Delphi 2 yang hanya bisa dijalankan pada Windows 95 dan Windows NT. Pada tahun 1997, 1998, 1999, Borland International yang berganti menjadi *Inprise Corporation* berturut-turut

kembali merilis penyempurnaan Delphi, yaitu Delphi 3, 4, 5, 6, yang bisa dijalankan pada sistm operasi Windows 95, 98, dan Windows NT.

2.3.3.1.2 Aplikasi dan Form

Aplikasi merupakan penerapan, pengimplementasian suatu permasalahan pekerjaan kedalam suatu sarana atau media yang dapat digunakan untuk menerapkan permasalahan tersebut, sehingga menjadi satu bentuk program tanpa menghilangkan nilai-nilai dasar dari permasalahan yang ada. Aplikasi dapat dibedakan menjadi aplikasi Windows dan aplikasi konsol. Aplikasi windows adalah aplikasi yang berjalan pada windows. Aplikasi non wondows contohnya yang berjalan pada Dos yang biasa disebut aplikasi konsol.

Secara umum, sebuah aplikasi paling sedikit melibatkan form. Namun biasanya sebuah aplikasi melibatkan banyak form. Sebuah form umumnya melibatkan komponen lain. Kotak Combobox dan Tombol Radiobutton merupakan contoh komponen. Tetapi tidak semua komponen terlihat secara visual, komponen yang terlihat secara visual biasa disebut dengan control. Pada Delphi, sebuah aplikasi akan diletakkan pada sebuah proyek. Sebuah proyek dapat membawahi sejumlah form. Form merupakan inti dari setiap aplikasi pada Delphi karena dianggap sebagai dasar aplikasi windows.

2.3.3.1.3 Gambaran Singkat Bagain-Bagian IDE

IDE (*Integrated Developement Environment*) adalah sebuah lingkungan dimana tools yang diperlukan untuk mendesain, menjalankan dan mengetes

sebuah aplikasi yang terjadi dan terhubung baik sehingga memudahkan dalam pengembangan program. Melalui IDE ini pemrograman secara visual merancang tampilan untuk pemakai (antar muka pemakai).

2.3.3.2 Kelebihan –Kelebihan Delphi

Pembangunan perangkat lunak ini dilakukan dalam lingkungan pengembangan terpadu (*Integrated Development Environment*) Delphi 7.0 dari Borland. Pemilihan tools ini didasarkan pada beberapa pertimbangan berikut:

1. Delphi mendukung pemrograman berorientasi objek. Dengan demikian, hasil analisis yang telah dilakukan pada tahap sebelumnya dapat diimplementasikan dengan mudah.
2. Delphi memiliki kelas-kelas yang telah terdefinisi. Hal ini membuat pemrograman tidak perlu dilakukan dari awal. Pemrograman cukup dilakukan dengan menggunakan kembali atau menurunkan kelas-kelas baru dari kelas yang sudah ada.
3. Delphi memiliki kelas-kelas control antar muka visual yang telah terdefinisi. Hal ini memudahkan dalam pembangunan perangkat lunak dengan antarmuka grafis yang *user friendly.*
4. Delphi memiliki fasilitas *Borland Database Engine* (BDE) yang menyebarkan lingkungan basis data transparansi dan kecepatan pengaksesan yang jauh lebih baik.