PEMBANGUNAN PERANGKAT LUNAK PEMBANGKIT JADWAL UNTUK LIGA INDONESIA

TUGAS AKHIR

Disusun Oleh :

ANINDHITA DEWABHARATA
NRP. 5199 100 052

JURUSAN TEKNIK INFORMATIKA
FAKULTAS TEKNOLOGI INFORMASI
INSTITUT TEKNOLOGI SEPULUH NOPEMBER
SURABAYA
2005

PEMBANGUNAN PERANGKAT LUNAK PEMBANGKIT JADWAL UNTUK LIGA INDONESIA

TUGAS AKHIR

Diajukan Guna Memenuhi Sebagian Persyaratan
Untuk Memperoleh Gelar Sarjana Komputer
Pada
Jurusan Teknik Informatika
Fakultas Teknologi Informasi
Institut Teknologi Sepuluh Nopember
Surabaya

Mengetahui / Menyetujui,

Dosen Pembimbing I	Dosen Pembimbing II
Ir. Siti Rochimah, M.T.	Faizal Johan Atletiko, S.Kom
NIP. 132 103 631	NIP. 132 300 414

SURABAYA
JANUARI, 2005

ABSTRAK

Dalam Tugas Akhir ini akan dilakukan sebuah penerapan algoritma genetika untuk melakukan pengaturan penjadwalan kompetisi sepak bola Indonesia. Pembangkitkan jadwal pertandingan akan dilakukan antar peserta selama satu musim kompetisi. Jadwal tersebut akan dibangkitkan berdasarkan kendala – kendala yang ditentukan prioritasnya, sehingga dapat menghasilkan jadwal kompetisi bagi klub peserta Liga Indonesia.

Algoritma genetika merupakan suatu proses iteratif yang disetiap iterasinya melakukan evaluasi solusi dengan masalah yang dipecahkan. Dalam evaluasi antara solusi dengan masalah yang dipecahkan, digunakan suatu fungsi obyektif yaitu fungsi evaluasi (evaluation function). Solusi dengan nilai fungsi evaluasi yang tinggi mempunyai kesempatan yang besar untuk dipilih kembali untuk penurunan solusi (generasi) pada iterasi berikutnya.

Berdasarkan uji coba dan evaluasi, pemodelan dan penjadwalan dapat dilakukan dengan menggunakan metode algoritma genetika dengan operator genetika menggunakan cycle crossover sebagai metode pindah silang dan reciprocal exchange mutation sebagai metode mutasinya, Hanya saja masih ditemukan pelanggaran – pelanggaran terhadap batasan, ini dapat dilihat dari nilai fitness yang diperoleh. Jumlah generasi, populasi dan parameter operator genetika berpengaruh terhadap kromosom yang dihasilkan baik dari nilai fitness dan kebenaran yang diperoleh, tapi tentu saja mempengaruhi kecepatan eksekusi jumlah data peserta yang semakin besar, akan mempengaruhi nilai fitness, kecepatan dan kebenaran jadwal yang dibuat. Semakin besar jumlah data peserta, semakin lama waktu yang diperlukan untuk melakukan eksekusi dan semakin menurun nilai fitness yang berdampak pada penurunan kebenaran jadwal.

***Kata Kunci** : Algoritma Genetika, Mutasi, Crossover, Seleksi, Jadwal, Offspring, Parent dan Kromosom.*

KATA PENGANTAR

Puja dan puji syukur terucap kehadirat ALLAH SWT atas segala rahmat dan hidayah-Nya yang telah dikaruniakan kepada penulis sehingga bisa menyelesaikan Tugas Akhir ini dengan judul :

"PEMBANGUNAN PERANGKAT LUNAK PEMBANGKIT JADWAL UNTUK LIGA INDONESIA"

Tugas Akhir ini disusun sebagai salah satu syarat akademis yang harus penulis lalui untuk menyelesaikan pendidikan Sarjana Strata Satu (S-1) di jurusan Teknik Informatika, Fakultas Teknologi Informasi, Institut Teknologi Sepuluh Nopember Surabaya

Dalam penyusunan Tugas Akhir ini penulis berusaha menerapkan ilmu – ilmu yang telah didapatkan selama menempuh masa perkuliahan. Dengan selesainya penyusunan Tugas Akhir ini, besar harapan penulis dapat memberikan manfaat bagi orang lain.

Tiada gading yang tak retak, begitu pula tidak ada manusia yang sempurna. Meski telah berusaha, dengan segala kerendahan hati penulis menerima saran dan kritik bila Tugas Akhir ini masih terdapat banyak kekurangan. Dan tak lupa atas bantuan dan masukan yang telah diberikan, penulis mengucapkan terima kasih yang sebesar – besarnya kepada :

1. Orang tuaku tercinta, Papa Wayan T. Soeasthya, Mama Titiek Ediningsih. Terima kasih atas do'a, pengorbanan, kesabaran, tanggung jawab, perhatian,

kasih sayang sepanjang masa dan dorongan semangat selama ini. Hanya berkat doa dan bimbinganlah penulis dapat menyelesaikan Tugas Akhir ini.

2. Ibu Siti Rochimah dan Bapak Faizal Johan Atletiko selaku dosen pembimbing Tugas Akhir. Penulis mengucapkan terima kasih dan penghormatan yang setinggi – tingginya atas bantuan, bimbingan saran, dan masukan selama pengerjaan Tugas Akhir ini.
3. Bapak Yudhi Purwanato selaku Ketua Jurusan Teknik Informatika, dan Kepala Laboratorium Pemrograman.
4. Bapak Ahmad Holil Noor Ali, Bapak Hari Ginardi, Bapak Waskitho Wibisono, Bapak Faizal Johan Atletiko, dan Ibu Nanik Suciati selaku dosen wali. Terima kasih atas bimbingan dan kesabaran yang diberikan.
5. Ibu Nanik Suciati, Bapak Rully Soelaiman, dan Ibu Diana Purwitasari sebagai dosen penguji dalam Tugas Akhir ini.
6. Bapak F.X Arunanto selaku Kasie Laboratorium Komputing, tempat penulis menyelesaikan sebagian Tugas Akhir.
7. Dosen – dosen di laboratorium pemrograman, tempat penulis menghabiskan waktu selama kuliah. Bapak Royyana Muslim Ijtihadie, Bapak Faizal Johan Atletiko, Bapak Tohari Ahmad, Bapak Darlis Heru Mukti dan Bapak Imam Kuswardayan. Terima kasih atas ilmu, saran, bimbingan, sumbangan semangat, konsumsi dan persahabatan selama ini.
8. Segenap dosen pengajar Jurusan Teknik Informatika, atas segala ilmu dan bimbingannya selama kuliah. Semoga ilmu yang diajarkan akan dapat bermanfaat bagi penulis sendiri maupun masyarakat luas. Segenap karyawan

TU, RBTC dan Laboratorium Jurusan Teknik Informatika, yang telah membantu proses kelancaran perkuliahan dan banyak membantu penulis. Pak Narno, Pak Cucuk, Bu Tutik, Mas Yudi, Mas Sugeng, Pak Kodir, Pak Soleh, Pak Pri, Pak Is, Pak Mu'in, Mas Pur, Mbak Eva, Mas Hermono, Mas Bambang, Gayuh, dan Wawan. Satpam kampus TC Pak Moko, Pak Karmono, Pak Bagio dan Pak Jarwo. Terima kasih telah memberikan keamanan selama kuliah

9. Saudara dan kerabat dekat, Mas Adithya Wisnuberata, Om R. Hartono Soedjoko, dan semua kerabat yang telah mendahului penulis menghadap ALLAH SWT, atas segala dukungan, perhatian, kesabaran, nasehat dan pelajaran yang diberikan.
10. Sahabat – sahabat baik selama di Teknik Informatika ITS, semoga selamanya hingga akhir jaman dan di akhirat nanti. I Gusti Ngurah Rai Dharma Widhura, Samiyah, Hindrawan Aris, Agung Wahyu Wibowo, Firman Fathoni, Liga Awami, dan Rully Agus Hendrawan. *Thanks to all of you for your magic hand and mind.*
11. Teman – teman C-0F, Ervan, Rifqi, Kemal, Dhion, Nahid, Maman, Ade Reza, Anton, Hendra, Rono, Roy, Wahib, Shidiq, Nafis, Fajar, Eva, Mediana, Anna, AdeTA, Ningrum, Anny, AAK, Nina, Nuning, Rahma, Fajar, Eli, Ridha, Armand, Firstiar dan semua teman – teman C-0F yang belum sempat ditulis dan disebutkan. *Thanks to all of you, for your help, kindness and friendship.*
12. Teman – teman seperjuangan TA, Mas Nugie, Mas Gershom, Rai, AW, Kecu, Surya HP, Nina, dan Yoga.

13. Administrator laboratorium pemrograman, Mas Rizki, Mas Kendy, Ade, Gunna, Joko, I'in, Jakka, JP, Rozi, Kholimi, Huda dan Hendra, Kris, Zainal, Hendry, Iwan, Eva, Ratna dan Wawan. Terima kasih telah memberikan tempat dan memperlancar proses perkuliahan di TC. Administrator laboratorium komputing, Mas Gershom, Robby, Eddy, Albert, Dhewy dan lain - lain, Administrator laboratoium AJK, RPL dan IBS di TC. Dan Koordinator serta Asisten Praktikum di TC. Terima kasih atas sumbangsih pengabdian dan kerja kerasnya untuk memperlancar proses perkuliahan di TC.
14. Penghuni laboratorium dan HIMA, Mas Ajun, Hatfi, Widya, Lufi, Melinda, Edwin, Connie, Yogi, Karpo, Prevan, Surya, Hendra, Hoirul, Warna Agung, Farid Pak Dhe, Taqi, Luqman, Zhain, Hanif, Haris, Victor Lunggu, dan Dekom.
15. FC Informatika, Mas Gershom, Pak Roy, Pak Jo, Bang Darwan, Mas Widhi, Mas Ajie, Mas Gosan, Mas Henry, Mas Chempez, Bang Bornok, Mas Zola, Mas Andi KI, Surya HP, Rai, Nahid, AW, Victor, Pak Dhe, Huda, Hendra, Dwi Arie, Fadelis, Yance, A'an, Kiki, Dede', Seno, Fajri, Sigit Bersaudara, Mardi, Andika, Arysco dan lainnya.
16. Keluarga besar Ngagel Tirto III / 9A, (Alm) Mbah Sumini dan keluarga, Pak Guru Sukatno, dan Sugeng.
17. BlackHole (10.126.11.212), ns1 (202.155.84.178), Gajah (10.126.11.252), Meteor (10.126.12.35), Antariksa (10.126.12.5), Nebula (10.126.11.11), i32 XNU, PIXMA, dan HP3535, serta L 5855 UB. Situs *Search Engine* terbaik www.google.com. Terima kasih atas kerja keras kalian selama ini.

18. Dua orang hawa yang telah menemani, memberi semangat, dukungan, curahan kasih sayang, perhatian dan bantuan, baik materi maupun spiritual, dikala senang maupun sedih sampai saat ini, Ilmayani dan *my little sister* Nina "Ndung-dung" Anggraini. *Thanks for everything.*
19. Setetes embun, semoga selalu memancarkan dan memberikan hawa kesejukannya setiap saat.
20. Dan terima kasih sebesar – besarnya kepada diri penulis sendiri yang telah berhasil untuk mengalahkan rasa malas yang ada.

Surabaya, Januari 2005

Penulis

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

BAB I
PENDAHULUAN

1.1. Latar Belakang

Liga Indonesia merupakan kompetisi resmi sepak bola profesional yang telah berlangsung sebanyak sembilan musim kompetisi. Kompetisi ini dibagi menjadi tiga divisi, yaitu Divisi Utama, Divisi I dan Divisi II. Divisi Utama merupakan fokus utama dari Liga Indonesia. Peserta kompetisi diikuti oleh klub - klub sepak bola dari seluruh Indonesia.

Mulai Liga Indonesia IX, sistem kompetisi yang digunakan adalah sistem kompetisi penuh, hal ini berarti setiap peserta harus bertanding sebanyak dua kali, yaitu kandang dan tandang, dengan masing – masing peserta yang lain.

Di dalam peraturan yang dikeluarkan PSSI disebutkan bahwa jika klub bertanding di kandang maksimal tiga kali berturut – turut begitu juga untuk pertanding tandang. Hal ini untuk menjaga faktor mental pemain, dengan asumsi jika pertandingan kandang dapat memetik poin maksimal, sedangkan pada pertandingan tandang bisa jadi kehilangan poin.

Dengan luas wilayah Indonesia, jarak merupakan permasalahan dalam pengaturan jadwal kompetisi. Dengan sistem kompetisi penuh, setiap klub diharuskan bertanding melawan klub – klub lain yang memiliki jarak cukup jauh. Contoh : Persipura Jayapura harus bertanding dengan PSPS Pekanbaru di kandang PSPS Pekanbaru, di Riau, begitu juga sebaliknya.

Jika terjadi pengaturan jadwal yang tidak menguntungkan secara pengeluaran biaya, contohnya apabila klub Persipura Jayapura bertanding ke

Pekanbaru, lalu harus kembali ke Jayapura untuk bertanding dengan klub lain sebagai tuan rumah, kemudian juga harus bertanding lagi ke Padang untuk melawan klub Semen Padang, maka akan mengakibatkan suatu klub harus mengeluarkan biaya yang cukup besar.

Disamping mengakibatkan pengeluaran biaya yang cukup besar, pengaturan jadwal yang buruk dapat menyebabkan kerugian internal klub seperti kelelahan yang diderita pemain, dikarenakan kekurangan waktu untuk pengembalian kondisi, sehingga pemain mudah cedera.

Dalam Tugas Akhir ini akan dibuat sebuah perangkat lunak yang akan melakukan pengaturan penjadwalan kompetisi sepak bola Indonesia. Perangkat lunak ini akan membangkitkan jadwal pertandingan antarpeserta selama satu musim kompetisi. Jadwal tersebut akan dibangkitkan berdasarkan kendala – kendala yang ditentukan prioritasnya. Disini kendala jarak akan mendapat prioritas, tetapi tetap harus memperhatikan kendala yang lain, sehingga dapat menghasilkan jadwal kompetisi yang meminimalkan *over-head* bagi klub peserta Liga Indonesia.

Perangkat lunak yang akan dibangun dalam Tugas Akhir ini diberi nama Penjadwalan Sepak Bola.

1.2. Tujuan dan Manfaat

Tujuan penelitian Tugas Akhir ini adalah :

1. Mempelajari dan memperdalam teori :
   - Penjadwalan
   - Algoritma Genetik, dan

- Metode *Linear Sequensial Model.*

2. Menerapkan algoritma genetik pada perangkat lunak penjadwalan kompetisi sepak bola Liga Indonesia.

1.3. Permasalahan

Ruang lingkup masalah dalam Tugas Akhir ini adalah :

1. Memodelkan permasalahan penjadwalan dengan mendefinisikan kendala dari permasalahan penjadwalan kompetisi, yang kemudian diikuti pemberian prioritas agar bisa diimplementasikan ke dalam perangkat lunak.
2. Menerapkan metode algoritma genetik untuk membuat perangkat lunak penjadwalan kompetisi Liga Indonesia yang menghasilkan jadwal tetap.

1.4. Batasan Permasalahan

Batasan masalah dalam Tugas Akhir ini adalah:

1. Penjadwalan Sepak Bola ini mengambil studi kasus kompetisi Liga Indonesia X tahun 2004.
2. Penjadwalan Sepak Bola ini hanya berlaku bagi Divisi Utama Liga Indonesia.
3. Peserta dalam penjadwalan ini berjumlah genap.
4. Dalam setiap harinya *(matchday)* jumlah tim yang bertanding adalah jumlah ideal.

1.5. Metodologi

Pembuatan tugas akhir ini terbagi menjadi beberapa tahapan sebagai berikut :

1. Studi literatur yang berhubungan dengan permasalahan penjadwalan dan algoritma genetika yang dipelajari dari paper dan artikel di internet, serta berkonsultasi dengan pembimbing.
2. Perancangan sistem yang mencakup :
   - Pemodelan masalah: dilakukan dengan merepresentasikan permasalah penjadwalan ke dalam model yang sesuai dengan metode GA.
   - Perancangan antarmuka: dilakukan agar antarmuka dari perangkat lunak dapat bersifat mudah digunakan.
3. Implementasi dan pengujian sistem berdasarkan batasan yang telah ditetapkan.
4. Evaluasi sistem dan melakukan perbaikan sistem untuk mendapatkan hasil yang lebih baik.
5. Penyusunan laporan tugas akhir.

1.6. Sistematika Laporan

Sebagai gambaran umum dari laporan Tugas Akhir yang disusun, berikut ini akan dijelaskan tentang sistematika penyusunan laporan Tugas Akhir ini secara garis besar.

Bab I Pendahuluan

Menjelaskan latar belakang, permasalahan yang diangkat beserta batasan-batasannya, tujuan dan manfaat yang ingin dicapai dari pembuatan Tugas Akhir ini serta metodologi yang dipakai.

Bab II Dasar Teori

Memaparkan hasil studi literatur berupa rangkuman artikel-artikel yang membahas Algoritma Genetika, penjadwalan liga Indonesia serta pendekatan Algoritma Genetika dalam penjadwalan.

Bab III Perancangan Perangkat Lunak

Membahas rancangan aplikasi yang akan dibangun dimulai dari deskripsi umum sistem, batasan sistem, proses-proses yang terjadi, representasi kromosom serta desain *database* sebagai penyimpanan data fisik.

Bab IV Pembuatan Perangkat Lunak

Menunjukkan implementasi dari tiap rancangan yang telah dibuat yaitu lingkungan aplikasi sistem, implementasi proses dan antarmukanya serta implementasi kromosom dan basis data.

Bab V Ujicoba dan Evaluasi Perangkat Lunak

Membahas pengujian aplikasi, dari lingkungan pengujian, data yang diuji serta macam-macam skenario pengujian. Hasil dari tiap uji coba akan dievaluasi dan dianalisa dari segi hasil dan waktu.

Bab VI Kesimpulan dan Saran

Menyatakan kesimpulan–kesimpulan yang didapat dari proses pembuatan Tugas Akhir, beserta saran – saran untuk pengembangan selanjutnya.

BAB II

DASAR TEORI

Pada bab ini akan dibahas tentang teori yang akan diimplementasikan pada aplikasi, teori tersebut adalah Algoritma Genetik. Setelah itu juga akan dibahas tentang sistem penjadwalan yang digunakan pada kompetisi Liga Indonesia.

2.1. Algoritma Genetika

Pada penjelasan tentang konsep dasar algoritma gentika berikut ini akan diperkenalkan beberapa istilah yang digunakan dalam algoritma genetika sebagai terminologi untuk memahami kinerja algoritma genetik, penjelasan secara garis besar tentang algoritma genetik dalam bentuk struktur umum algoritma genetik, dan karakteristik algoritma genetik dalam hal memanipulasi solusi, serta kemiripannya dengan prinsip algoritma pencarian (*searching*) konvensional.

2.1.1. Terminologi

Algoritma genetika merupakan suatu proses iteratif yang disetiap iterasinya melakukan evaluasi solusi dengan masalah yang dipecahkan. Dalam evaluasi antara solusi dengan masalah yang dipecahkan, digunakan suatu fungsi obyektif yaitu fungsi evaluasi (*evaluation function*) [Mic95]. Solusi dengan nilai fungsi evaluasi yang tinggi mempunyai kesempatan yang besar untuk dipilih kembali untuk penurunan solusi (generasi) pada iterasi berikutnya.

Solusi dalam algoritma genetika direpresentasikan sebagai kromosom. Solusi tersebut memiliki beberapa keterangan atau variabel yang berupa aturan, kalimat, kombinasi bilangan dan sebagainya. Keterangan tersebut pada kromosom dipandang sebagai gen. Masing – masing gen mempunyai nilai atau harga yang disebut alela. Secara simbolik untaian tersebut dapat ditulis seperti,

S: s1s2s3...sN

S merupakan kromosom, sedangkan s1, s2, hingga sN merupakan gen. Nilai s1, s2, hingga sN adalah alela dari masing-masing gen [Wal92].

Algoritma genetika melakukan pencarian solusi pada sekumpulan populasi kromosom. Dengan demikian proses pencarian yang dilakukan bersifat serentak (paralel) [Gol89]. Hal – hal yang mempengaruhi variasi, optimalitas, dan kecepatan proses adalah jumlah kromosom yang tepat untuk suatu populasi awal dan jumlah kromosom yang lolos dari seleksi setiap generasi.

Suatu solusi baru dihasilkan melalui proses replikasi atau reproduksi dari solusi yang lama. Dalam proses replikasi ini dapat terjadi proses pindah – silang gen (*cross-over*), atau mutasi gen, atau lompatan gen (*reordering* kromosom). Mekanisme ini menghasilkan kromosom (solusi) baru yang bervariasi, dan mungkin lebih baik dari solusi sebelumnya. Kromosom terbaik inilah yang diharapkan merupakan solusi untuk permasalahan yang dihadapi.

2.1.1.1. Struktur Umum

Algoritma genetika berawal dari inisialisasi himpunan solusi yang didapat secara acak. Himpunan solusi awal ini disebut dengan istilah

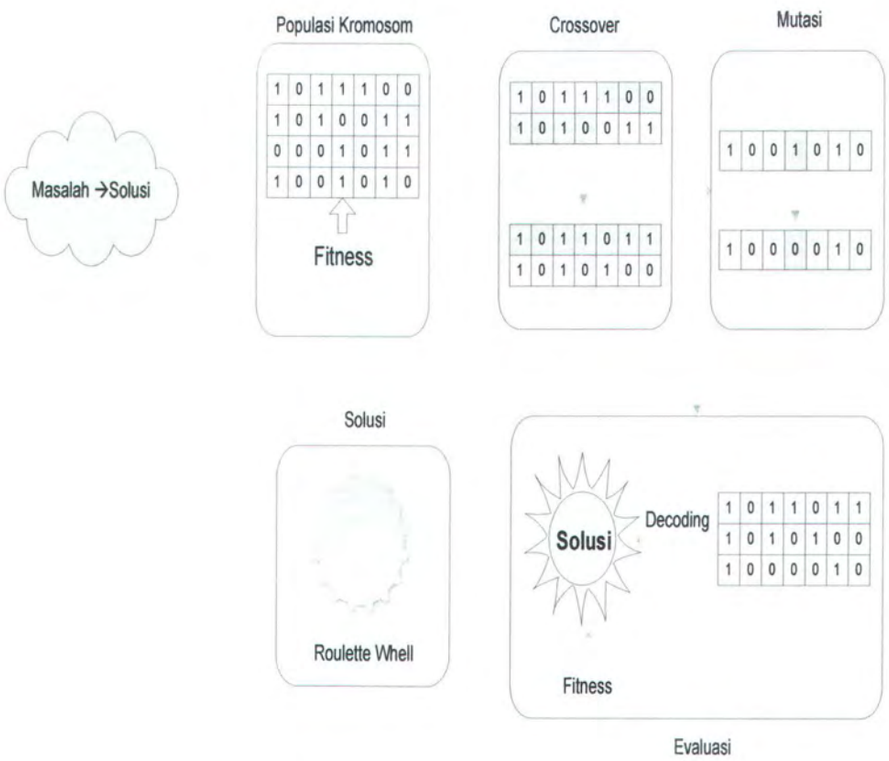
populasi awal. Kromosom – kromosom yang terdapat pada populasi awal akan berevolusi menurut proses iterasi yang berkelanjutan yang disebut generasi. Generasi dapat diartikan sebagai populasi baru hasil iterasi. Pada setiap generasi, kromosom – kromosom dievaluasi berdasarkan suatu takaran tertentu yang disebut *fitnes.* Sedangkan pada proses iterasinya, diberlakukan dua jenis operasi untuk menghasilkan generasi – generasi berikutnya, kedua operasi tersebut adalah :

- Operasi Genetika (*genetic operations*), yang terdiri atas tukar silang (*crossover*) dan mutasi (*mutation*), dan
- Operasi evolusi (evolution operation), yaitu seleksi (*selection*).

Pembentukan generasi berikutnya diawali dengan membentuk kromosom – kromosom baru yang disebut *offspring. Offspring* dibentuk dengan melalui dua operasi genetika, yaitu tukar silang dan mutasi. Setelah melalui kedua operasi tersebut, *offspring* yang dihasilkan bersama dengan beberapa kromosom lama (*parents*) memasuki tahap evaluasi dan dilanjutkan dengan seleksi. Evaluasi adalah menetapkan nilai kromosom yang disebut nilai *fitness* untuk kemudian dilakukan seleksi. Seleksi adalah pemilihan *offspring* dan/atau *parents* berdasarkan suatu nilai *fitness*-nya. Kromosom yang memilki *fitness value* lebih tinggi mempunyai kemungkinan lebih tinggi untuk dipilih sebagai anggota populasi baru. Populasi baru ini yang digunakan pada proses regenerasi berikutnya. Setelah melalui beberapa generasi, algoritma akan konvergen pada kromosom terbaik pada generasi tertentu [Cheng dan Gen, 1996].

Kromosom terbaik inilah yang diharapkan merupakan solusi untuk permasalahan yang dihadapi.

Untuk lebih jelasnya, diberikan ilustrasi sebagai berikut :



Gambar 2.1 Struktur Umum Algoritma Genetika

Jika diketahui P(t) adalah *parents* dan C(t) adalah *offspring* pada suatu generasi t, maka struktur umum algoritma genetik dapat dituangkan kedalam suatu prosedur sebagai berikut :

procedure genetic algoritms;

begin

t = 0;

```
inisialisasi P(t);
evaluasi P(t);
while (bukan kondisi terminasi) do
begin
rekombinasi P(t) menjadi C(t); {operasi tukar silang dan
mutasi}
    evaluasi C(t);
    dapatkan P( t+1); {hasil seleksi P(t) dan seleksi C(t)}
    t = t + 1;
end;
end;
```

2.1.1.2. Eksploitasi dan Eksplorasi

Teknik pemecahan masalah yang secara universal dan sering digunakan adalah pencarian (*searching*). Terdapat beragam macam teknik pencarian, tetapi secara umum dapat dibedakan menjadi dua macam strategi berdasarkan karakteristiknya. Strategi pertama disebut *blind search*, yaitu strategi pencarian yang tidak memerlukan tambahan informasi untuk mengarahkan proses pencariannya kedalam daerah perncarian terbaik, sehingga lebih terarah untuk menemukan solusi terbaik dan optimal.

Beragam teknik pencarian yang telah dikenal tersebut, pada dasarnya memiliki dua karakteristik utama di dalam proses pencarian atau

iterasi yang dilakukan, yaitu mengeksploitasi solusi terbaik dan mengeksplorasi daerah pencarian [Booker, 1987].

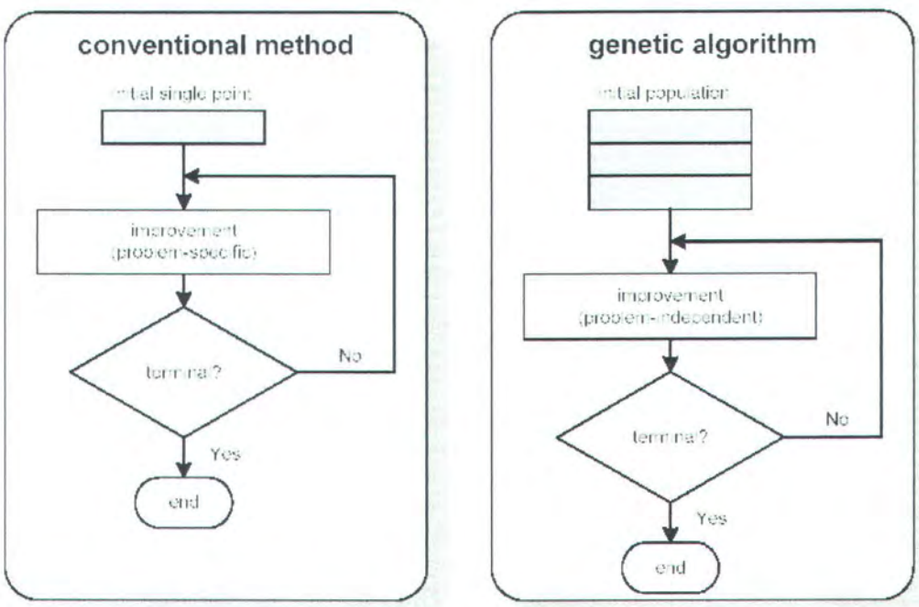
Teknik *hill climbing* adalah salah satu contoh teknik pencarian yang mengeksploitasi solusi terbaik dengan terus menerus melakukan iterasi sejauh iterasi itu masih mungkin dilakukan. Namun teknik ini mengabaikan eksplorasi daerah pencarian. Sedangkan contoh teknik pencarian yang mengeksplorasi daerah pencarian adalah *random search.* Namun teknik ini mengabaikan eksploitasi solusi tebaik dalam iterasinya. Algoritma genetika merupakan metode yang menggabungkan keduanya dengan menyeimbangkan eksplorasi daerah pencarian dan eksploitasi solusi terbaik [Michalewics, 1996].

Berawal dari daerah solusi yang sangat luas yang dihasilkan secara random dan disebut populasi. Kemudian melalui proses tukar silang, seluruh daerah solusi dieksplorasi. Selain itu proses tukar silang juga melakukan iterasi yang mengeksploitasi solusi terbaik dengan mempertahankan solusi yang memiliki nilai *fitness* yang lebih tinggi untuk dipilih sebagai *parents* pada iterasi berikutnya.

2.1.1.3. Pencarian Berbasis Populasi

Pada umunya algoritma pemecahan masalah optimasi merupakan urutan langkah menuju solusi optimal. Berawal dari satu nilai yang kemudian dieksploitasi menurut iterasi tertentu. Sedangkan algoritma genetika berawal dari beberapa nilai yang disebut populasi awal. Kemudian melalui proses iterasi tertentu bergerak dari populasi yang satu

kepada populasi yang diturunkan berikutnya. Untuk lebih jelasnya dapat dilihat pada diagram alir di bawah ini :



Gambar 2.2 Diagram Alir Perbandingan Algoritma Konvensional dengan Algoritma Genetika

2.1.2. Cara Kerja Algoritma Genetik

Yang dimaksud dengan cara kerja algoritma genetik pada pembahasan berikut ini adalah urut – urutan proses penyelesaian masalah dengan menggunakan algoritma genetik seperti yang dijelaskan menurut struktur umum algoritma genetik dan operasi evolusi, serta fungsi untuk menilai kromosom yang masih dapat dipertahankan pada setiap generasinya

2.1.2.1. Representasi dan Notasi Skema

Hal yang paling mendasar pada pemecahan masalah dengan menggunakan algoritma genetik adalah merepresentasikan atau mengkodekan masalah ke dalam bentuk kromosom – kromosom solusi.

Sebuah kromosom merupakan representasi dari sebuah solusi. Kromosom merupakan rangkaian gen yang memuat nilai terkodekan, sehingga kromosom dapat disebut juga dengan *string*. Setiap gen memuat nilai tertentu yang terkodekan menurut cara representasi tertentu. Pada umumnya representasi yang sering digunakan oleh para peneliti adalah dengan *bit string*.

Dasar teori yang dapat digunakan untuk menjelaskan cara representasi solusi adalah dengan bit string dikenal dengan notasi skema [Holland, 1975]. Di dalam notasi skema ini digunakan asumsi bahwa setiap bit string dapat dibentuk dengan rangkaian tiga digit/simbol, yaitu '0','1', dan '*'. Simbol '*' mempunyai makna *don't care*.

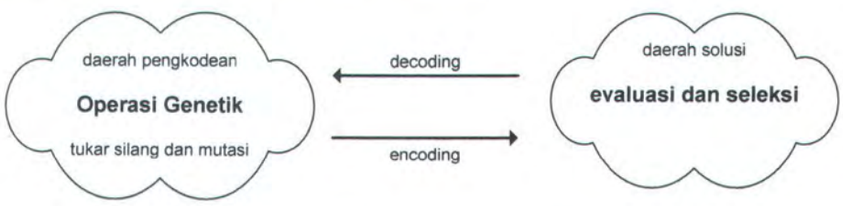
Misalnya terdapat notasi skema (*111100100). Menurut notasi tersebut akan terdapat kemungkinan dibentuk dua bit string, yaitu (0111100100) dan (11111100100). Sehingga dapat disimpulkan bahwa bila terdapat notasi skema dengan n simbol * akan dapat dibentuk 2^n bit string. Dan sebaliknya bila terdapat bit string dengan panjang m akan dapat dinotasikan sebanyak 2^m.

Jika skema dinyatakan dengan S dan orde skema (o(S)) adalah jumlah '0' dan '1' yang terdapat pada skema S, maka bila terdapat skema S_1=(***000*110) dan S_2=(***000*0*), dapat dinyatakan $\delta(S_1)=6$ dan $\delta(S_2)=4$. Dan jika panjang skema S ($\delta(S)$) adalah jarak antara *fixed bit* ('0' atau '1') yang pertama dan terakhir, maka $\delta(S_1)=6$ dan $\delta(S_2)=4$. Sedangkan

jumlah string dalam suatu populasi yang cocok dengan skema S pada waktu t dinyatakan dengan ε(S,t) [Michalewicz,1996].

2.1.2.2. Daerah Pengkodean dan Daerah Solusi

Algoritma genetik bekerja pada daerah pengkodean dan daerah solusi. Operasi genetik yang terdiri atas tukar silang dan mutasi bekerja pada daerah pengkodean. Sedangkan fungsi evaluasi dan seleksi bekerja pada daerah solusi [Cheng dan Gen, 1996].



Gambar 2.3 Daerah Solusi dan Daerah Pengkodean

Dalam hubungannya dengan kedua daerah tersebut, terdapat beberapa fenomena yang harus diperhatikan, yaitu :

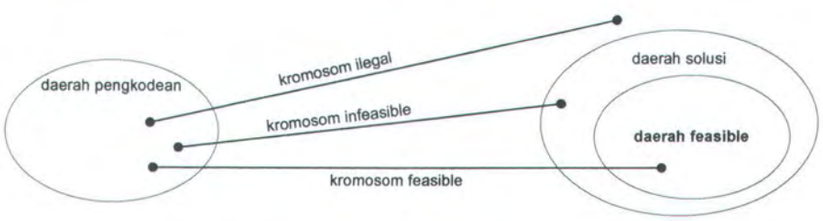
1. Kelayakan kromosom
2. Legalitas kromosom
3. Pemetaan yang unik

Kelayakan kromosom menunjukkan apakah sebuah solusi yang dikodekan sudah tepat berada dalam daerah yang layak di antara domain permasalahan. Sedangkan legalitas kromosom menunjukkan pada kenyataan bahwa sebuah kromosom benar – benar dalam merepresentasikan sebuah solusi di dalam domain. Kemudian dengan merujuk bahwa setiap solusi selalu dikodekan pada setiap kromosom, maka antara solusi dan kromosom terdapat suatu hubungan pemetaan.

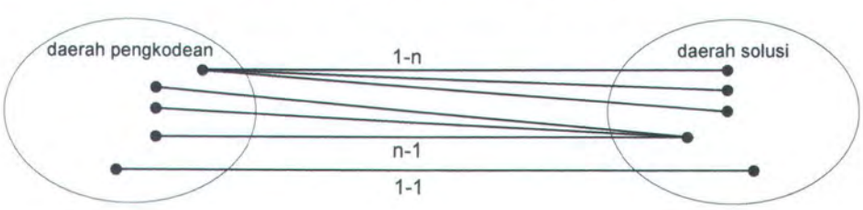
Tiga jenis pemetaan yang mungkin antara kromosom dan solusi, yaitu :

1. Pemetaan 1-1
2. Pemetaan n-1
3. Pemetaan 1-n

Pemetaan antara solusi dan kromosom yang terbaik adalah bila di antara keduanya memiliki relasi pemetaan 1-1.



Gambar 2.4 Feasibilitas danLegalitas



Gambar 2.5 Pemetaan Kromosom-Solusi

2.1.2.3. Fungsi Evaluasi

Yang dimaksud dengan fungsi evaluasi disini adalah suatu fungsi yang memberikan penilaian berupa nilai *fitness* kepada kromosom. Bila pada awal iterasi nilai *fitness* semua kromosom ini mempunyai rentang yang lebar, seiring dengan bertambahnya generasi, beberapa kromosom mempunyai rentang nilai *fitness* semakin kecil dan mendominasi populasi. Hal ini menyebabkan fungsi evaluasi menjadi masalah yang krusial dalam algoritma genetik karena dominasi kromosom dalam populasi ini dapat

menyebabkan iterasi yang hanya terfokus pada kromosom – kromosom tersebut sebagai akibat tingginya nilai *fitness* yang dimiliki. Jika hal ini terjadi, algoritma genetik akan konvergen terlalu dini, padahal nilai yang dihasilkan belum tentu optimal secara keseluruhan. Dan inilah yang disebut dengan terjebak pada lokal optimal. Hal ini dapat diatasi dengan menghindarkan kromosom super yang mendominasi populasi.

Fungsi evaluasi dilakukan dengan tiga tahap :

Tahap 1 Konversikan genotip menjadi fenotip. Bila kromosom berbentuk bit string, maka konversi dilakukan dengan mengkonversikan *binary* ke nilai *real* relatif

$x_k = (x_1^k, x_2^k)$, k = 1,2,...,pop_size pop_size : ukuran populasi

Tahap 2 Evaluasi fungsi tujuan $f(x^k)$

Tahap 3 Konversikan nilai fungsi tujuan menjadi nilai *fitness*

Eval $(v_k) = f(x^k)$, k= 1,2,...,pop_size

Sebagai contoh dalam penjadwalan, Fungsi evaluasi dapat didefinisikan sebagi sebuah rumusan fungsi seperti berikut ini : $\frac{1}{1+(p(x))}, \frac{1}{1+(p(x))^2}, \frac{1}{1+\sqrt{p(x)}}, \frac{1}{1+\ln(p(x))}$ dimana $p(x)$ adalah total nilai penalti dari nilai bobot penalti.

2.1.2.4. Tukar silang (*crossover*)

Selama dekade terakhir ini, beberapa operator *crossover* telah diusulkan atau digunakan, semisal *partial-mapped crossover* (PMX),

order crossover (OX), *cycle crossover* (CX), *postiton-based crossover*, *order-based crossover*, *heuristic crossover*, dan sebagainya. Secara kasar, operator ini dapat diklasifikasikan dalam dua kelas :

- Pendekatan *Canonica.l*
- Pendekatan *Heuristic.*

Pendekatan *Canonical* dapat dipandang sebagai suatu perluasan dua titik atau banyak titik *crosssover* dari *binery string* ke penyajian permutasi. Biasanya, penyajian permutasi akan menghasilkan keturunan tidak sah oleh dua titik atau multipoint *crossover,* di dalam pengertian yang menyangkut itu beberapa objek besar kemungkinan luput atau hilangdan juga mungkin terjadi duplikasi pada keturunan itu. Perbaikan prosedur ditambahkan di dalam pendekatan ini untuk memecahkan sifat melanggar hukum keturunan.

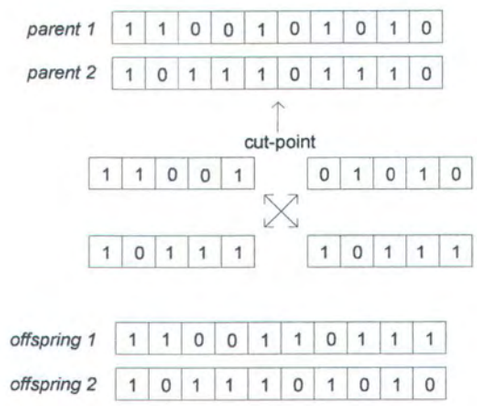
Inti dari pendekatan *canonical* adalah yang mekanisme random buta (*blind random*). Tidak ada jaminan suatu keturunan yang diproduksi oleh *crossover* menjadi lebih baik dibanding *parents* mereka. Sedangkan Aplikasi heuristik pada *crossover* berusaha untuk menghasilkan suatu keturunan yang lebih baik.

Berikut beberapa contoh operator *crossover* :

- Pindah Silang Satu Titik

  Prosedur dari pindah silang satu titik adalah menghasilkan sebuah nomer secara acak (lebih kecil atau sama dengan panjang kromosom) sebagai

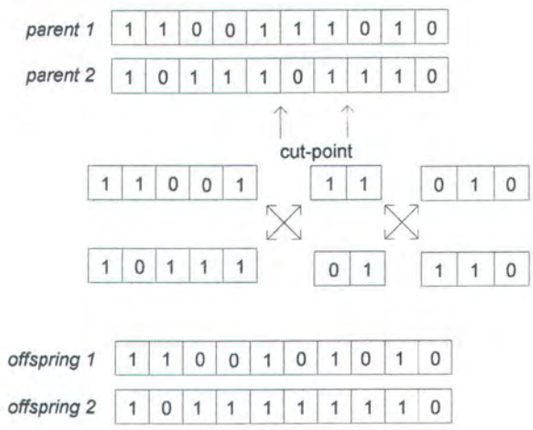
posisi (*pivot*) pindah silang. Bit-bit sebelum posisi tersebut tetap dipertahankan sedang bit-bit setelah posisi tersebut dipersilangkan antara kedua orangtua.



Gambar 2.6 Operasi Tukar Silang One Cut-Point

- Pindah Silang Banyak Titik (*Multipoint*)

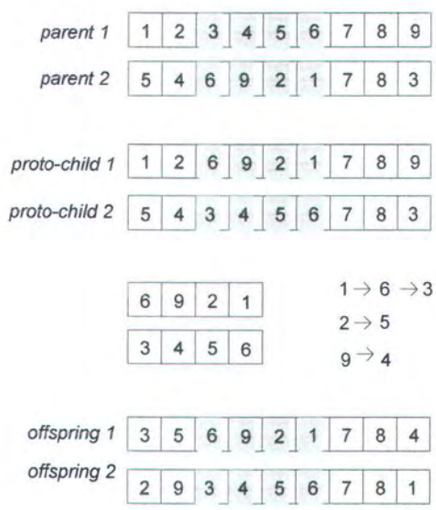
  Prosedur dari pindah silang banyak titik adalah menghasilkan beberapa (lebih dari satu) nomer secara acak (lebih kecil atau sama dengan panjang kromosom) sebagai posisi (*pivot*) pindah silang. Bit-bit sebelum posisi tersebut tetap dipertahankan sedang bit-bit setelah posisi tersebut dipersilangkan antara kedua orangtua.



Gambar 2.7 Operasi Tukar Silang Uniform

- *Partial Mapped Crossover (PMX)*

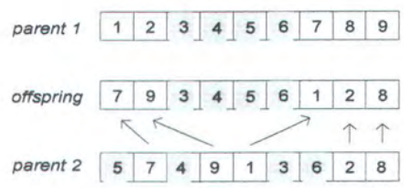
  PMX diusulkan oleh Goldberg dan Lingle [Gen97]. PMX dipandang sebagai suatu perluasan dua titik *crosssover* dari *binary string* ke penyajian permutasi. PMX menggunakan suatu prosedur perbaikan khusus untuk mengatasi pelanggaran yang disebabkan oleh *two-point crossover* sederhana. Inti dari PMX adalah *two-point crossover* ditambah dengan prosedur perbaikan.



Gambar 2.8 Operasi PMX

- *Order Crossover (OX)*

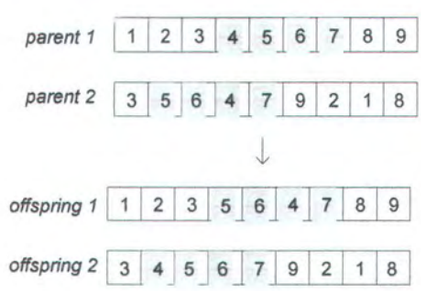
  OX diusulkan oleh Davis [Gen97]. OX dapat dilihat sebagai salah satu jenis variasi dari PMX dengan prosedure perbaikan yang berbeda.



Gambar 2.9 Operasi OX

- *Heuristic Crossover*

  Pertama kali dipresentasikan oleh Grefenstette [Gen97]. Pada *heuristic konvensional* ada dua dasar pendekatan, yaitu : *nearest neighbor* and *best insertion heuristics*. *Crossover Grefenstette* telah diimplementasikan dengan menggunakan mekanisme *nearest neighbor heuristic.* Cheng and Gen mendesain *crossover* dengan mekanisme *best insertion* untuk *vehicle routing* dan *scheduling*.

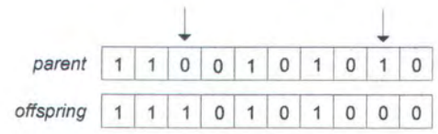


Gambar 2.10 Operasi Heuristic

2.1.2.5. Mutasi

Mutasi adalah operasi membentuk *offspring* dengan mengubah susunan gen *parent*. Mutasi dapat dilakukan dengan metode inversi, yaitu terhadap suatu nilai gen dilakukan inversi dari '0' ke '1' atau sebaliknya. Metode yang lain adalah dengan menukar gen – gen yang terdapat pada kromosom yang disebut *exchange mutation*. Metode tersebut dapat dilihat pada gambar berikut :

Gambar 2.11 Operasi Mutasi Inversi



Gambar 2.12 Operasi Mutasi Exchange

Selain itu terdapat juga beberapa operator mutasi lainnya seperti *Insertion Mutation, Dsplacement Mutation, Heuristic Mutation.*

2.1.2.6. Seleksi

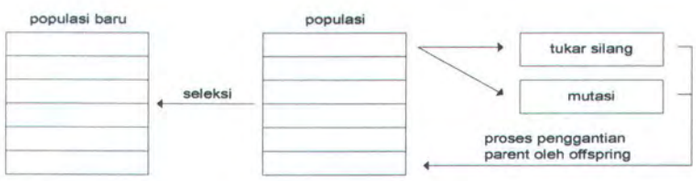
Ada tiga permasalahan penting yang sangat mempengaruhi seleksi yang harus diperhatikan, yaitu :

1. Daerah sampling
2. Mekanisme seleksi
3. Probabilitas seleksi.

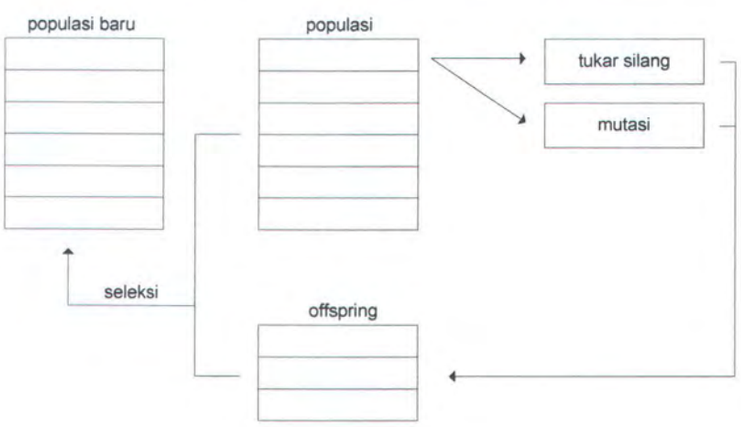
2.1.2.6.1. Daerah Sampling

Seleksi menghasilkan generasi berikutnya. Kromosom – kromosom yang berada di dalamnya dapat berasal dari *parent* atau *offspring* atau gabungan keduanya. Asal kromosom inilah yang disebut dengan daerah sampling.

Daerah sampling dikatakan sebagai daerah sampling tetap *(regular sampling space)* bila *offspring* yang dihasilkan langsung menggantikan *parent* kemudian dikenakan seleksi. Namun bila *offspring* dan *parent* memiliki kemungkinan yang sama untuk bertahan hidup dan mengikuti seleksi untuk membentuk generasi baru, maka proses seleksi ini dikarenakan pada daerah sampling yang diperluas (*enlarged sampling space*). Berikut ini ilustrasi daerah sampling :



Gambar 2.13Seleksi dengan Daerah Sampling yang Tetap



Gambar 2.14 Seleksi dengan Daerah Sampling Yang Diperluas

2.1.2.6.2. Mekanisme Sampling

Mekanisme sampling berguna untuk memilih kromosom daerah sampling yang dipertahankan hidup pada proses seleksi. Ada tiga jenis mekanisme sampling, yaitu :

- Sampling stokastik

Tiap kromosom dipilih berdasarkan kemungkinan bertahan hidup dengan memberikan probabilitas seleksi untuk setiap kromosom berdasarkan nilai fungsinya. Untuk kromosom k dengan nilai *fitness* f_k, probabilitas seleksi p_k adalah :

$$p_k = f_k \Big/ \sum_{j=1}^{pop\_size} f_j$$

Persamaan 2.1

Contoh sampling stokastik adalah Roullete Wheel

- Sampling deterministik

Pemilihan kromosom berdasarkan kromosom terbaik dari daerah sampling. Metode elitis adalah contoh sampling deterministik, yaitu menentukan kromosom elit untuk diwariskan pada generasi berikutnya.

- Sampling campuran

Menggabungkan sampling stokastik dan sampling deterministik. Contohnya adalah seleksi turnamen. Dalam seleksi

turnamen, sekelompok kromosom dipilih secara random untuk reproduksi.

2.1.2.6.3. Probabilitas Seleksi

Probabilitas seleksi diberlakukan pada mekanisme sampling stokastik. Kromosom dengan nilai *fitness* tinggi akan mendominasi populasi. Untuk menghindari hal ini dilakukan penyekalaan nilai *fitness* (*fitness scale)*. Nilai *fitness* yang disesuaikan dinyatakan dengan f_k'. Hubungan f_k' dengan nilai *fitness* sebenarnya f_k adalah $f_k' = g(f_k)$.

2.2. Sistem Penjadwalan Pada Kompetisi Liga Indonesia

Semenjak Liga Indonesia IX sistem yang digunakan yang digunakan telah mengalami perubahan, yang pada awalnya membagi peserta ke dalam dua wilayah berdasarkan letak geografis dari *home-base* peserta, yaitu wilayah barat dan wilayah timur, sistem ini mirip dengan sistem *round-robin*. Sedangkan mulai Liga Indonesia X sudah menggunakan sistem kompetisi penuh.

2.2.1. Definisi Kompetisi Penuh

Kompetisi penuh merupakan sistem pertandingan yang mempertemukan keseluruhan peserta tim dalam satu musim kompetisi. Setiap peserta akan bertanding sebanyak dua kali dengan peserta lainnya, yaitu bertanding di kandang sendiri dan kandang lawan. Total pertandingan yang dimainkan oleh setiap peserta adalah :

$$n_p = (n-1)*2$$

Persamaan 2.2

Dimana n adalah jumlah peserta liga dan np adalah jumlah seluruh pertandingan yang dimainkan oleh masing – masing peserta.. Sedangkan jumlah keseluruhan pertandingan yang ada dalam satu musim kompetisi adalah :

$$n_q = \left(n/2 * n_p\right) \text{ atau bisa } n_q = (n-1) * n$$

Persamaan 2.3

Dimana n adalah jumlah peserta liga dan np adalah jumlah pertandingan yang dimainkan setiap peserta selama satu musim kompetisi. Sedangkan nq adalah jumlah keseluruhan pertandingan yang ada dalam satu musim kompetisi.

Sistem kompetisi ini telah digunakan di negara – negara yang maju liga sepak bolanya, seperti di negara – negara eropa. Begitu juga di negara – negara Asia seperti Jepang, Korea Selatan, Arab Saudi sudah menerapkan sistem ini. Sedangkan untuk negara di kawasan ASEAN, baru beberapa negara saja yang menggunakannya, Indoenesia adalah negara ketiga yang menggunakan sistem ini setalah Thailand dan Malaysia.

Sistem ini digunakan karena dianggap paling adil bagi seluruh peserta liga, seluruh peserta bakal merasakan kekuatan dari peserta lainnya, tak ada satu pun yang tidak bertemu, berbeda dengan sistem pembagian wilayah seperti pada kompetisi liga Indonesia terdahulu, tidak setiap tim dapat merasakan kekuatan tim yang lain, tim wilayah barat tidak bertemu dengan tim wilayah timur, kecuali yang bertemu di ajang empat besar.

2.2.2. Kondisi yang Harus Diperhatikan dalam Pembuatan Jadwal

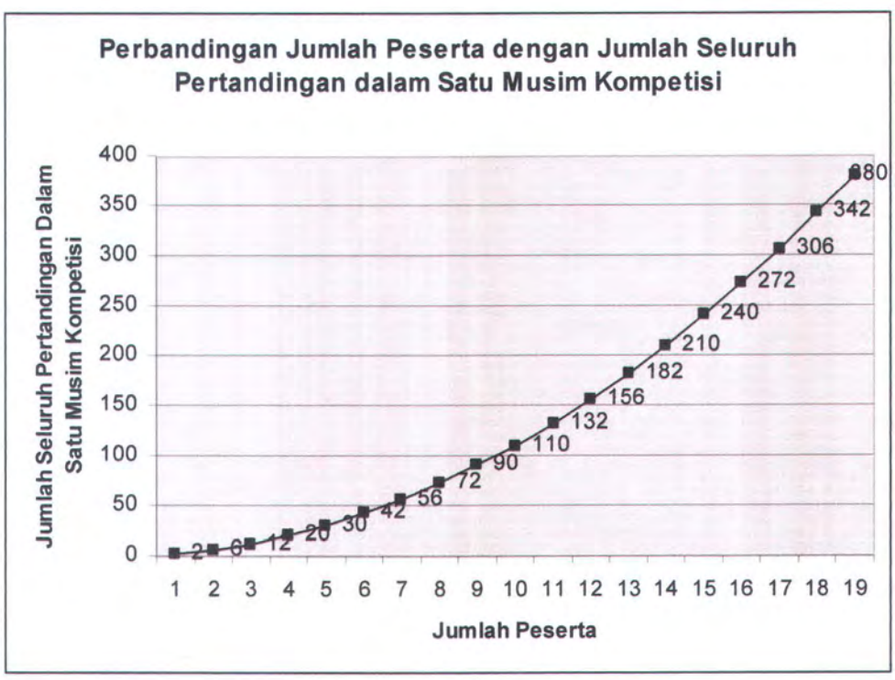
Agar dapat mencapai hasil yang maksimal dalam pelaksanaan kegiatannya, dalam pembuatan jadwalnya perlu diperhatikan beberapa kondisi dan aturan. Kondisi dan aturan ini tentunya sangat mempengaruhi jalannya kompetisi, apabila kondisi ini diabaikan bukan tidak mungkin jalannya kompetisi dapat terganggu, berkurang mutunya, bahkan bukan tidak mungkin kompetisi akan terhenti. Beberapa kondisi dan aturan yang harus diperhatikan antara lain adalah : jumlah peserta dan lama pelaksanaan, jumlah pertandingan dalam satu pekan, format rute *home* dan *away* dan letak geografis dari *home base* para peserta, serta kondisi pemain (masa *recovery*) dan tim.

2.2.2.1. Jumlah Peserta dan Lama Pelaksanaan

Di beberapa negara Eropa rata – rata jumlah peserta untuk sebuah liga adalah 18 atau 20 klub dan berlangsung selama kira – kira 9 sampai dengan 10 bulan, dari bulan September hingga bulan Mei, sedangkan untuk negara Asia, jumlah pesertanya adalah rata – rata dari 12 hingga 18 klub.

Jumlah peserta menentukan jumlah pertandingan yang akan dimainkan, yang tentunya akan berdampak pada lamanya kompetisi. Sebagai contoh, dalam pelaksanaan Liga Indoensia IX jumlah peserta adalah 20 klub, maka jumlah pertandingan yang dimainkan oleh setiap klub adalah sebanyak 38 kali pertandingan, dan jumlah seluruh pertandingan yang ada dalam satu musim kompetisi adalah 380 pertandingan. Sedangkan pada Liga Indoensia X jumlah pesertanya

dikurangi menjadi 18 klub, maka jumlah pertandingan yang dimainkan oleh setiap pesertanya berkurang menjadi 34 pertandingan, dan jumlah pertandingan yang ada dalam satu musim menjadi 306 pertandingan. Jika dilihat secara langsung jumlah peserta yang ikut sangat berpengaruh terhadap jumlah seluruh pertandingan yang ada dalam satu musim kompetisi. Perbandingan dari hal tersebut dapat dilihat dari gambar 2.14 berikut ini:



Gambar 2.15 Perbandingan Jumlah Peserta dengan Jumlah Seluruh Pertandingan dalam Satu Musim Kompetisi

Selain jumlah peserta yang ikut serta, ada hal lain yang dapat mempengaruhi lamanya pelaksanaan, yaitu jadwal dari turnamen regional dan tim nasional. Jadwal regional seperti European Championship Cup atau UEFA Cup di Eropa, Asian Championship Cup di Asia. Sedangkan

untuk jadwal tim nasional tergantung dari ada atau tidaknya turnamen yang diselenggarakan. Kedua hal tersebut bisa menggangu pelaksanaan kompetisi karena harus diikuti oleh beberapa klub --klub peringkat pertama dan kedua hasil kompetisi liga sebelumnya-- dan beberapa pemain dari beberapa klub untuk bergabung dalam tim nasional.

Untuk negara – negara Eropa maupun Amerika Latin, jadwal turnamen baik turnamen regional maupun tim nasional yang ada, biasanya sudah terjadwal dari tahun ke tahun, dan biasanya waktunya tidak berubah – ubah, sedangkan untuk negara – negara di Asia, masih terjadi turnamen dan jadwal yang berubah – ubah dari tahun ke tahun.

2.2.2.2. Jumlah Pertandingan dalam Satu Pekan

Seperti yang telah dijelaskan pada subbab sebelumnya bahwa jumlah peserta bisa menentukan jumlah pertandingan yang dimainkan dan berpengaruh dalam lama pelaksanaan. Lama pelaksanaan mempunyai sebuah rentang waktu, dari suatu tanggal mulai hingga tanggal berakhir.

Rentang waktu pelaksanaan bisa dipendekkan ataupun dipanjangkan dengan mengatur jumlah pertandingan dalam satu pekannya. Jadi ada kemungkinan walaupun jumlah peserta lebih banyak tapi lama pelaksanaan bisa sama. Lama pelaksanaan kompetisi yang berjumlah 18 peserta dengan jumlah 20 peserta berakhir sama, padahal jika dilihat dari jumlah pertandingan yang harus dimainkan adalah 34 hari pertandingan (*matchday*) untuk 18 peserta dan 38 *matchday* untuk 20 peserta hampir berbeda satu bulan. Kejadian ini bisa dilihat di Liga Italia dengan Liga

Spanyol, walaupun jumlah peserta mereka berbeda tapi untuk musim kompetisi 2003 – 2004 lama pelaksanaannya sama.

Di Liga Indonesia X jumlah pertandingan setiap pekannya adalah dua kali pertandingan untuk setiap pesertanya. Hari yang digunakan adalah hari Selasa atau Rabu dan Sabtu atau Minggu. Jadi sebuah tim peserta akan bermain pada hari Selasa atau Rabu, kemudian akan bermain lagi pada hari Sabtu atau Minggu, idealnya pemain yang bermain pada hari Selasa, akan bermain lagi pada hari Sabtu, kemudian tim yang bermain pada hari Rabu akan bermain lagi pada hari Minggu, tapi hal ini tidak memungkinkan, masih mungkin terjadi yang bermain pada hari Rabu, akan bermain lagi pada hari Sabtu.

2.2.2.3. Format Rute Home, Away dan Letak Geografis dari Home Base para Peserta

Format rute *home* dan *away* ini merupakan urutan perjalanan pertandingan yang dilakukan oleh sebuah tim. Untuk pelaksanaan di Indonesia, kondisi ini perlu diperhatikan, karena untuk melakukan sebuah perjalanan *away* diperlukan biaya yang besar karena jarak antara *home base* peserta ada yang jauh, contoh : jarak *home base* antara Pesija di Jakarta dengan Persipura di Jayapura.

Kombinasi dalam melakukan urutan perjalanan *home* dan *away* ialah berurutan *away* atau *home* maksimal sama sebanyak dua kali, jadi dihindari keadaan sebuah tim untuk melakukan pertadingan di *home* terus menerus baru kemudian *away* terus menerus, hal ini bisa merugikan

karena biasanya pertandingan *home* sangat mungkin untuk meraih hasil maksimal.

Akan semakin penting lagi, apabila dalam satu pekannya jumlah pertandingan yang dilakukan sebanyak dua kali pertandingan. Sebuah tim apabila bertanding *away* dengan jarak yang jauh, lalu harus bertanding di *home base* atau harus bertanding *away* ditempat yang relatif jauh dari pertandingan pertama, akan mengakibatkan kehabisan waktu dan tenaga untuk melakukan perjalanan tersebut. Untuk itu apabila format rute pertandingan *home* dan *away* berurutan maksimal dua kali, tentunya dengan pertimbangan ketika *away*, tim yang dihadapai adalah tim yang jarak *home base*-nya berdekatan, contoh apabila Persipura melakukan pertandingan *away* ke Jawa, maka tim yang akan dihadapinya adalah Persebaya Surabaya kemudian Persik Kediri, atau ke PSS Sleman lalu ke Persijatim Solo FC, setelah itu kembali lagi bertanding di *home base*.

Pada pelaksanaan Liga Indonesia X masih terdapat tim yang bertanding di *home base* sebayak lebih dari dua kali, seperti Persebaya Surabaya, Persija Jakarata, Persita Tangerang, tapi untuk pertadingan *away* ketiga tim tersebut hanya bertanding berurutan sebayak dua kali. Untuk pertandingan berurutan baik *home* maupun *away* lebih dari dua kali dialami oleh Persik Kediri. Persik mengalami jadwal *away* sebanyak empat kali secara berurutan yaitu ketika melawan Persipura, PSM, Persela dan Persebaya dan mengalami tiga kali pertandingan *home* secara berurutan, yaitu ketika menghadapai Deltras, PSPS, dan PSMS. Walaupun

ketika melakukan pertandingan *away* secara berurutan tersebut jarak hari antara pertandingan melawan PSM dan Persela adalah sebelas hari, tapi secara mental sangat tidak menguntungkan.

2.2.2.4. Kondisi Pemain dan Tim

Kondisi berikut ini sangat erat hubungannya dengan tiga kondisi sebelumnya, apabila ketiga kondisi sebelumnya belum terpenuhi maka akan berpengaruh pada kondisi pemain dan tim, jumlah pertandingan yang banyak dengan jarak hari yang pendek – pendek dan jarak tempuh *home base* antar peserta berjauhan maka akan mengakibatkan buruknya kondisi pemain. Dari buruknya kondisi pemain akan mengakibatkan jeleknya kondisi tim, sehingga mutu sebuah pertandingan akan dipertanyakan. Dengan kondisi fisik dan metal yang kurang akan mengakibatkan permainan yang kasar, sehingga tidak enak ditonton dan dapat merugikan kondisi pemain sendiri. Dari mutu pertandingan yang jelek akan menghasilkan output yang jelek, padahal kompetisi Liga Indonesia merupakan inputan untuk menghasilkan tim nasional yang baik sehingga prestasi sepak bola kita bisa diperhitungkan baik di Asia maupun di dunia. Dengan dana yang dikeluarkan begitu besar maka diharapkan hasil yang diperolch juga baik.

BAB III

ANALISA DAN PERANCANGAN PERANGKAT LUNAK

Bab ini berisi penjelasan proses perancangan perangkat lunak, meliputi deskripsi kebutuhan perangkat lunak, pemodelan data, pemodelan fungsional, serta perancangan antarmuka.

3.1. Deskripsi Kebutuhan Perangkat Lunak

Dalam deskripsi kebutuhan perangkat lunak ini akan dijelaskan tentang masukan dan keluaran dari sistem yang ada dan pembagian hak pengguna pada sistem termasuk akses-akses yang bisa dilakukan oleh masing-masing pengguna. Untuk selanjutnya perangkat lunak pembangkitan jadwal ini diberi nama Kick Off.

Data masukan yang diterima berasal dari administrator. Data yang diinputkan berupa data tim peserta liga Indonesia, stadion – stadion yang digunakan, tanggal dan waktu pelaksanaan setiap pertandingan. Administrator juga menetapkan parameter Algoritma Genetika dan kriteria jadwal. Barulah sistem membangkitkan jadwal dengan menerapkan Algoritma Genetika. Hasil pembuatan jadwal akan disimpan, yang kemudian dapat dilihat oleh administrator dan user.

Pengguna Kick Off dapat dibagi menjadi dua jenis yaitu administrator dan user. Penjelasan dari setiap pengguna adalah sebagai berikut :

1. Administrator

   Administrator adalah pengguna yang mempunyai hak penuh dalam mengakses Kick Off. Hal-hal yang dapat dilakukan oleh seorang administrator adalah:

   - Mengelola data klub, data stadion, data tanggal pertandingan, dan data wilayah.
   - Membangkitkan jadwal.
   - Melihat jadwal.

2. User Umum

   User umum adalah pengguna yang akan melihat hasil dari jadwal yang telah dibangkitkan. User umum bisa terdiri dari klub peserta, maupun masyarakat umum.

Selanjutnya Kick Off ini akan terdiri dari dua aplikasi, yaitu aplikasi untuk mengolah data masukan, dan menampilkan jadwal yang telah dibangkitankan dan aplikasi untuk pembangkit jadwal.

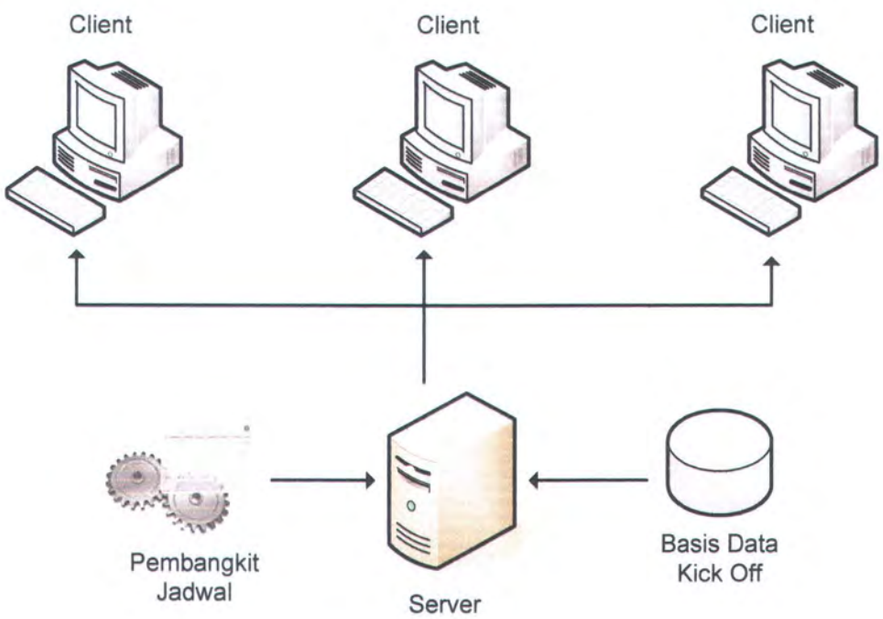
Aplikasi untuk mengolah data masukan dan menampilkan jadwal akan diimplementasikan menjadi sebuah web dengan menggunakan PHP, sedangkan datanya akan disimpan dalam sebuah basis data dengan menggunakan Oracle 9i. Untuk itu diperlukan sebuah server, sebagai server web dalam hal ini server PHP dan server Oracle 9i.

Untuk aplikasi yang kedua, yaitu aplikasi pembangkit jadwal, akan menggunakan bahasa pemrograman Java, dengan *compiler*-nya JDK 1.4. Aplikasi berbentuk *desktop application* dan akan mengambil data inputan dari basis data

yang telah ada, yaitu data klub, dan inputan langsung dari user sebagai parameter algoritma genetika.

Konfigurasi pada komputer *client* untuk user tidak memerlukan spesifikasi yang khusus. Hanya diperlukan sebuah *browser* internet seperti Internet Explorer untuk bisa menjalankan Kick Off pada *client*.

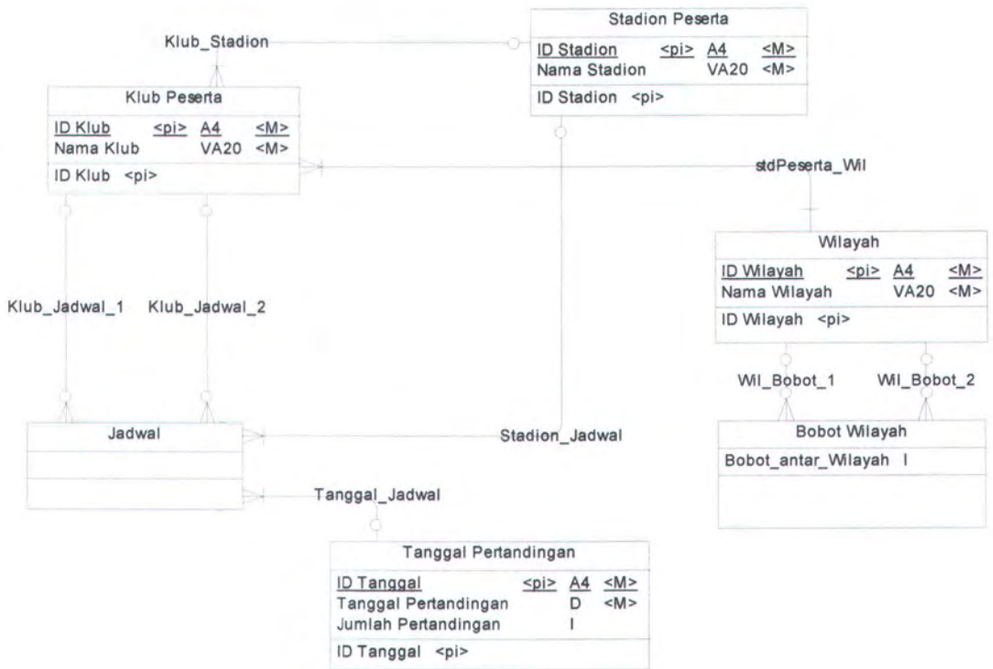
Pada Gambar 3.1 berikut ini akan digambarkan deskripsi umum dari Kick Off.



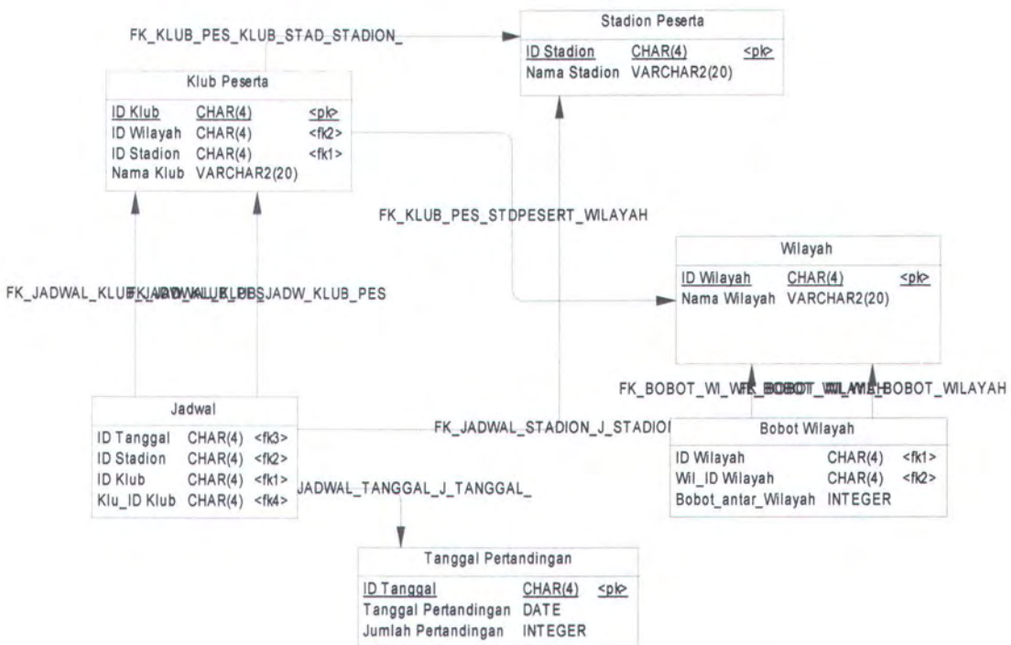
Gambar 3.1 Deskripsi Umum Kick Off

3.2. Pemodelan Data

Dalam bagian ini akan dijelaskan mengenai pemodelan basisdata yang akan dibuat dengan menggunakan *Power Designer*. Gambar 3.2 menggambarkan rancangan model data konseptual (CDM) yang dipakai pada Kick Off dan pada gambar 3.3 akan menggambarkan tentang model data fisikal.



Gambar 3.2 Model Data Konseptual Basis Data Kick Off



Gambar 3.3 Model Data Fisikal Basis Data Kick Off

Berikutnya akan dijelaskan tentang tabel – tabel yang ada dalam basis data Kick Off.

Tabel 3.1 Tabel Klub Peserta

Atribut	Tipe Data	Keterangan	Null
ID Klub	char	• *Primary key.* • Akan diambil sebagai input aplikasi pembangkit jadwal, dan dibangkitkan menjadi kromosom	Tidak
ID Wilayah	char	• Foreign Key dari tabel Wilayah • Menunjukkan wilayah dari sebuah klub	Tidak
ID Stadion	char	• Foreign Key dari tabel Stadion • Menunjukkan stadion yang digunakan sebagai *homebase*	Tidak
Nama Klub	varchar	Nama Klub Peserta Liga	Tidak

Tabel 3.1 merupakan tabel Klub Peserta, yaitu tabel yang digunakan untuk menyimpan data tentang klub peserta.

Tabel 3.2 Tabel Stadion

Atribut	Tipe Data	Keterangan	Null
ID Stadion	char	• *Primary Key* • Akan digunakan sebagai Foreign key pada tabel Klub Peserta	Tidak
Nama Stadion	varchar	Nama stadion yang digunakan.	Tidak

Tabel kedua yang digunakan adalah tabel Stadion, spesifikasi dari tabel ini dapat dilihat pada tabel 3.2. Tabel ketiga adalah tabel Wilayah, yang akan menunjukkan letak wilayah dari *homebase* klub peserta. Dan tabel keempat

merupakan tabel Tanggal Pertandingan. Tabel ketiga dan keempat dapat dilihat pada tabel 3.3 dan 3.4.

Tabel 3.3 Tabel Wilayah

Atribut	Tipe Data	Keterangan	Null
ID Wilayah	char	• *Primary Key* • Akan digunakan sebagai Foreign key pada tabel Klub Peserta	Tidak
Nama Wilayah	varchar	Nama wilayah tempat dari *homebase*	Tidak

Tabel 3.4 Tabel Tanggal Pertandingan

Atribut	Tipe Data	Keterangan	Null
ID Tanggal Pertandingan	char	• *Primary Key* • Akan digunakan sebagai Foreign key pada tabel Jadwal	Tidak
Tanggal Tanding	date	Menunjukan tanggal pertandingan	Tidak

Tabel berikutnya adalah tabel Bobot Wilayah, tabel ini digunakan untuk mendapatkan nilai antar wilayah yang ada. Tabel ini dapat dilihat pada tabel 3.5. Tabel terakhir pada basis data Kick Off adalah tabel Jadwal. Tabel ini mendapatkan input dari aplikasi pembangkit jadwal. Tabel ini juga akan sebagai input pada view yang akan ditampilkan ke user.

Tabel 3.5 Tabel Bobot Wilayah

Atribut	Tipe Data	Keterangan	Null
ID Wilayah	char	*Foreign Key* dari tabel wilayah	Tidak
Wilayah_ID Wilayah	char	*Foreign Key* dari tabel wilayah	Tidak
Bobot Nilai	Integer	Nilai dari jarak antar wilayah.	Tidak

Tabel 3.6 Tabel Jadwal

Atribut	Tipe Data	Keterangan	Null
ID Tanggal	char	*Foreign Key* dari tabel Tanggal	Tidak
ID Stadion	char	*Foreign Key* dari tabel Stadion	Tidak
ID Klub	char	Klub bertindak sebagai kandang	Tidak
Klu_ID Klub	char	Klub bertindak sebagai tandang	Tidak

3.3. Perancangan Algoritma Genetika pada Pembangkitan Jadwal

Pada subbab ini akan dijelaskan tentang perancangan pembangkit jadwal dengan menggunakan algoritma genetika.

3.3.1. Batasan yang Digunakan

Batasan yang harus dipenuhi sebuah jadwal liga Indonesia adalah:

1. Dalam satu hari sebuah tim hanya bertanding satu kali. Berarti bahwa sebuah tim tidak mungkin berada pada dua atau lebih stadion yang berbeda.
2. Stadion yang digunakan hanya memainkan satu pertandingan setiap harinya. Stadion yang digunakan oleh lebih dari satu tim sebagai *home base*, juga hanya boleh digunakan satu kali dalam satu hari.

Sedangkan pertimbangan-pertimbangan yang digunakan adalah:

1. Kombinasi dalam melakukan urutan perjalanan *home* dan *away* ialah berurutan *away* atau *home* maksimal sama sebanyak dua kali, jadi dihindari keadaan sebuah tim untuk melakukan pertadingan di *home* terus menerus baru kemudian *away* terus menerus.
2. Ketika *away*, tim yang dihadapi adalah tim yang jarak *home base*-nya berdekatan, contoh apabila Persipura melakukan pertandingan *away* ke Jawa, maka tim yang akan dihadapinya adalah Persebaya Surabaya kemudian Persik Kediri, atau ke PSS Sleman lalu ke Persijatim Solo FC, setelah itu kembali lagi bertanding di *home*.

3.3.2. Representasi Tabel Jadwal

Pertama dilihat struktur dari tabel penjadwalan. Tabel ini berbentuk matrik dua dimensi. Isi dari matrik tersebut --yang kemudian akan disebut "set"-- berupa data tim yang bertanding. Ada kemungkinan sebuah set tidak ada isinya apabila memang tidak ada yang bertanding. Untuk lebih jelasnya dapat dilihat gambar berikut ini

Hari 1	Hari 2	...	Hari k
Set_1	Set_{N+1}	...	$Set_{(k-1)N+1}$
Set_2	Set_{N+2}	...	$Set_{(k-1)N+1}$
...	...	...	...
Set_N	Set_{2N}	...	Set_{kN}

(a)

Hari 1	Hari 2	Hari 3	...	Hari 12
ab	db	ad	...	bc
cd	ac	cb	...	da

(b)

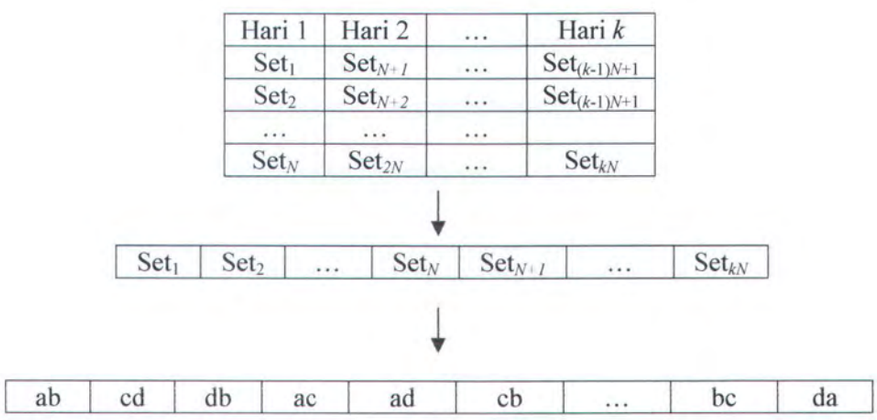
Gambar 3.4 Tabel Jadwal Berbetuk Matrik

Berdasarkan gambar diatas N merupakan jumlah pertandingan yang dimainkan dalam satu hari, sedangkan k merupakan banyaknya hari yang digunakan dalam satu musim kompetisi. Isi set merupakan data tim yang bertanding, satu set berisi 2 tim yang akan saling berhadapan, dari data tim tersebut dapat diketahui tempat yang digunakan untuk bertanding.

Untuk lebih jelasnya dapat dilihat contoh berikut : Tim yang bertanding { a, b, c, d }, maka isi sebuah set adalah permutasi dua – dua dari tim yang bertanding tersebut, dapat dilihat bentuk setnya pada gambar 3.4 (b)

Seperti yang telah dijelaskan diatas, dari sebuah set dapat diketahui tempat pertandingan dilangsungkan, dengan catatan kita harus membuat ketetapan bahwa tempat pertandingan adalah kandang dari tim yang disebut duluan dalam sebuah set, jadi jika isi sebuah set adalah "ab" maka tempat pertandingan dilangsungkan adalah di tempat atau di stadion dari tim "a".

Kemudian dilakukan konversi untuk direpresentasikan ke dalam bentuk kromosom, yaitu dengan membuat struktur linear. Struktur linear yang digunakan adalah *vertical linear* [San]. Pada *vertical linear,* struktur linear didapat dengan cara mengambil set pertama dari hari pertama, kemudian diurutkan sampai dengan set ke-N dari hari pertama tersebut, selanjutnya ditambahkan dengan set pertama dari hari kedua hingga set terakhir dari hari kedua, begitu seterusnya. Hasilnya dapat dilihat pada gambar 3.5.



Gambar 3.5 Konversi Matrik ke Vertical Linear

Hasil dari linearisasi vertikal tersebut merupakan representasi tabel jadwal ke dalam bentuk kromosom pada algoritma genetika.

3.3.3. Fungsi *Fitnes*, Nilai Penalti dan Seleksi

Fungsi *fitnes* dapat didefinisikan sebagai berikut:

$$f(x) = \frac{1}{1+(p(x))}$$

Persamaan 3.1

dimana p(x) adalah total nilai penalti dari nilai bobot penalti [San]

Kelangsungan hidup dari sebuah individu (kromosom) dapat dihitung dari nilai penaltinya. Pemetaan akan dilakukan oleh fungsi *fitness*.

Pada awalnya, ketika nilai penalti tinggi, perbedaan antara nilai *fitness* tidak cukup penting untuk membuat perbedaan antara individu yang bersaing, ketika nilai *fitness* tersebut menjadi dekat dengan nilai optimal, maka nilai penalti menjadi lebih rendah. Dengan fungsi *fitness* maka perbedaan akan semakin terlihat, sehingga akan menampilkan individu yang lebih baik.

Untuk nilai penalti dapat didefinisikan dengan fungsi sebagai berikut :

$$p(x) = \sum_{i=1}^{k} (wi.ni(c) + vi.mi(x))$$

Persamaan 3.2

dan w_i adalah bobot penalti, $n_i(c)$ adalah banyaknya pelanggaran terhadap batasan yang harus dipenuhi (*hard constraint*). Sedangkan v_i juga merupakan bobot penalti tapi untuk banyaknya jumlah pelanggaran $m_i(c)$ terhadap pertimbangan *(soft constarint)*.

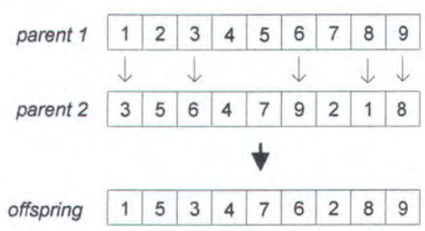
Selanjutnya seleksi dilakukan terhadap individu yang dipilih dari generasi sebelumnya menurut nilai dari kemungkinan kelangsungan hidup mereka. Seleksi ini akan menggunakan daerah sampling tetap *(regular sampling space)* dengan mekanisme sampling deterministik.

3.3.4. Operator Genetika

3.3.4.1. Pindah Silang dan Mutasi

Sudah banyak ahli yang memperkenalkan metode – metode pindah silang, seperti *one-point*, *order crossover*, *partial mapped crossover*, dan sebagainya. Untuk menyelesaikan permasalahan kali ini akan dicoba dengan menggunakan *cycle crossover.*

Cycle crossover tidak menggunakan titik *crossover*. Pertama set dari *parent* pertama akan disalinkan ke *offspring*, kemudian akan dicari set petama dari *parent* kedua, lalu dicari isi set yang sama di *parent* pertama. Lalu akan disalinkan ke *offspring*. Setelah itu akan dicari lagi pada posisi yang sama di parent kedua dan seterusnya. Itu dilakukan hingga tidak ada yang dapat disalinkan lagi. Maka set – set *offspring* yang masih kosong akan diisi dengan set dari *parent* kedua. Untuk lebih jelasnya dapat dilihat gambar 3.6

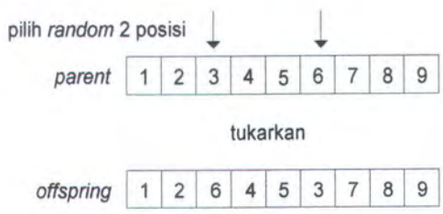


Gambar 3.6 Proses Crossover

Peranan dari mutasi dalam Algoritma Genetika adalah adanya jaminan dari pencarian heuristik. Mutasi mencoba mendapatkan penemuan individual dari bagian yang tidak terpecahkan dari ruang masalah sampai pada waktu yang sekarang.

Situasi dasar dari konsistensi mutasi *preserver* adalah sebagai berikut: pilihlah dua titik dan suatu panjang tertentu, lalu lakukan pertukaran pada perpotongan menggunakan jarak panjang secara acak dari titik pertama dan perpotongan pada titik kedua.

Solusi yang lebih efisien adalah bukan dengan perpotongan pada panjang tertentu tapi dengan sebuah set ditukarkan lalu diulangai berkali kali [San]. Lihat gambar 3.7



Gambar 3.7 Proses Mutasi

3.3.1.1. Elitisme

Dengan menggunakan populasi *non-overlapping*, populasi yang lama akan digantikan semua oleh yang baru. Dengan Elitisme, semua populasi akan digantikan oleh populasi yang baru kecuali individu yang layak. Individu dari populasi lama yang layak tersebut, kemudian disebut dengan Individu elit akan berada pada populasi yang baru.

3.3.1.2. Hubungan antara Ukuran Populasi dan Jumlah Generasi

Waktu eksekusi secara kasar tergantung ukuran populasi dan jumlah generasi, dan pemusatan kecepatan bergantung pada perbandingan kedua hal tersebut. Populasi yang kecil dapat memproses banyak generasi, sedangkan populasi yang besar hanya dapat melakukan beberapa generasi

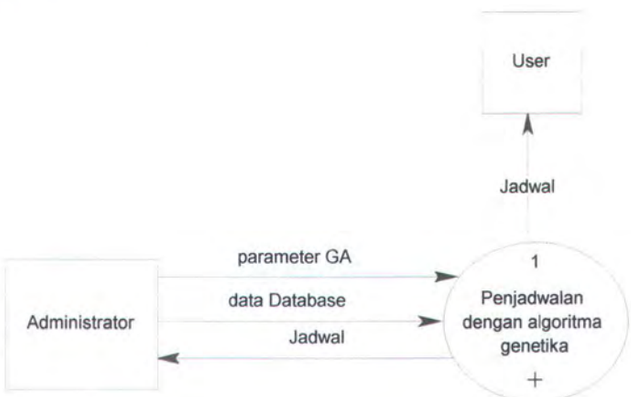
saja. Pada populasi yang besar juga akan mengakibatkan jumlah kombinasi pada proses *crossover* maupun mutasi.

3.4. Perancangan Proses

Sistem yang dirancang dapat dimodelkan dengan menggunakan pemodelan *Data Alur Diagram* (DAD) menggunakan notasi Gane dan Sarson, pada Power Designer 6.1. Berikut penjelasan tiap proses menurut levelnya dari level nol sampai dengan level dua.

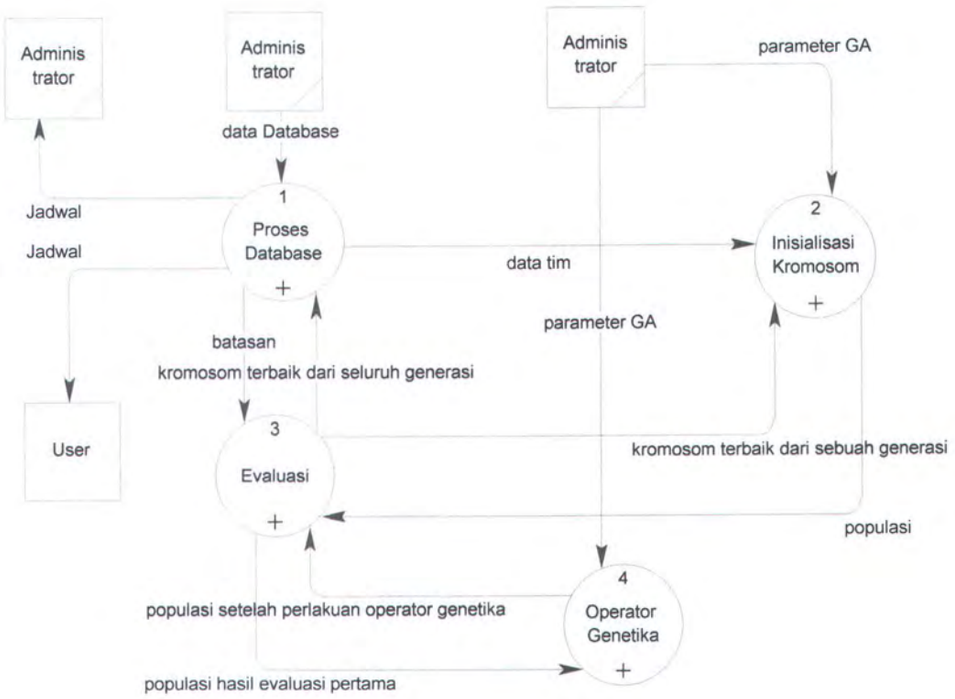
3.4.1. DAD Level 0

Pada data alur diagram level nol ini terdapat sebuah entitas user. User melakukan proses masukan data, dan eksekusi prosesnya. Kemudian user dapat melihat atau mendapatkan hasil dari proses pembangkitan jadwal. Pada proses pemasukan, user memasukkan data tim peserta, data stadion dan data tanggal serta waktu pelaksanaan setiap pertandingan. DAD level 0 dapat dilihat pada gambar 3.8.



Gambar 3.8 DAD Level 0

3.4.2. DAD Level 1



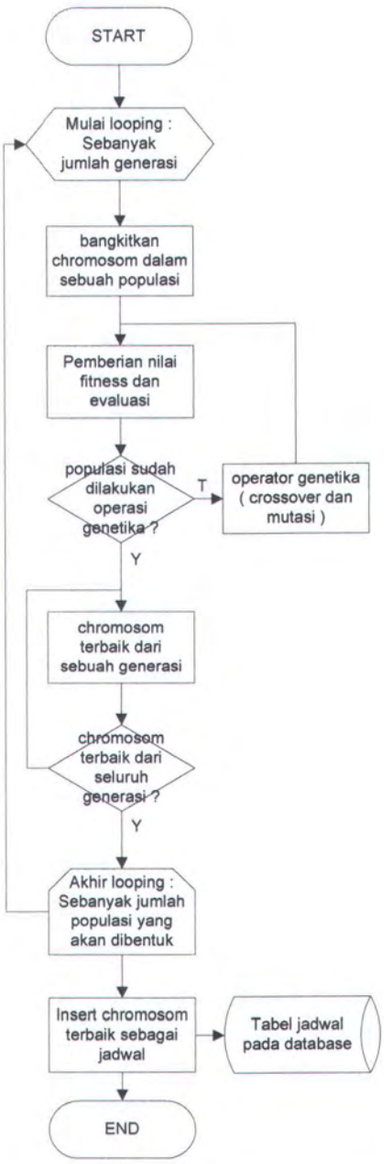
Gambar 3.9 DAD Level 1

Dari DAD level satu, proses pembangkitan jadwal dapat diturunkan menjadi empat proses pada DAD level satu, yaitu : Proses Basis Data, Inisialisasi Kromosom, Evaluasi, dan Operator Genetika. Keempat proses tersebut dapat dilihat pada gambar 3.9.

Proses basis data melakukan proses pengambilan data dari basis data yang akan digunakan oleh proses lainnya. Proses yang akan menggunakan hasil pengolahan dari proses basisdata adalah proses inisial kromosom dan proses evaluasi. Pada proses basis data juga melakukan proses pemasukan data hasil pembangkitan jadwal ke basis data, yang selajutnya akan ditampilkan ke user sebagai sebuah bentuk jadwal.

Proses kedua pada DAD level satu ini adalah proses inisialisasi kromosom. Dalam proses ini dilakukan pembentukan kromosom dari inputan data yang diperoleh proses basis data. Apabila telah dilakukan pembangkitan sebuah generasi, maka kromosom atau individu terbaik dari generasi tersebut akan diikutkan untuk pembentukan populasi generasi berikutnya. Selanjutnya adalah proses evaluasi. Proses evaluasi akan melakukan pencarian kromosom terbaik dengan perhitungan nilai *fitness*. Dan proses yang terakhir adalah proses operator genetika, didalam proses ini dilakukan operasi mutasi dan operasi pindah silang pada kromosom – kromosom pada sebuah populasi.

Untuk diagram alur dari level satu dapat dilihat pada gambar 3.10. Proses penentuan kromosom terbaik akan dicari dari beberapa generasi, setiap generasi tersebut akan terdiri dari sebuah populasi. Setiap generasi akan memperoleh sebuah kromosom terbaik, dari populasinya tersebut, setiap kromosom dalam poluasi akan melalui proses pemberian nilai fitness dan evaluasi, operasi genetika dan seleksi.



Gambar 3.10 Diagram Alur Level 1

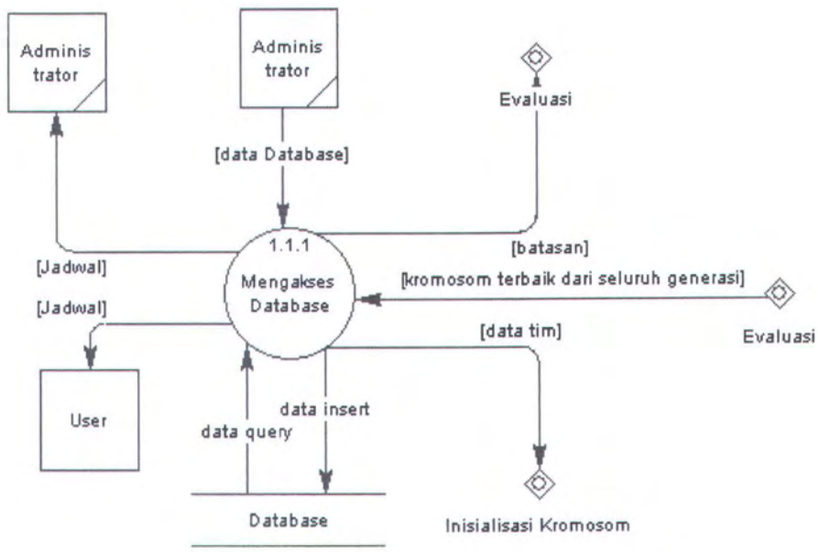
Pertama akan ditentukan jumlah generasi yang akan diproses, dalam setiap generasi tersebut akan dibentuk individu kromosom dengan jumlah populasi tertentu. Kemudian setiap individu tersebut akan diberikan nilai fitness, setelah diberikan nilai fitness maka akan dilakukan proses operasi genetika yang meliputi *crossover* dan mutasi. Populasi kromosom yang telah

dilakukan rekayasa tersebut kemudian akan dihitung lagi nilai fitness-nya, dan dicari kromosom dengan nilai fitness terbaik. Kromosom dengan nilai fitness terbaik akan dibandingkan dengan kromosom terbaik dari generasi lainnya, kromosom terbaik dari seluruh generasi yang ada akan dipakai sebagai jadwal.

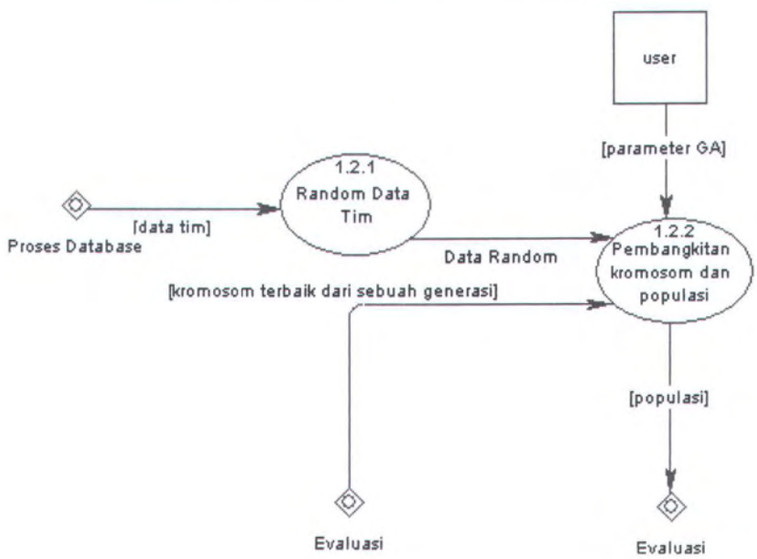
3.4.3. DAD Level 2

Untuk DAD Level dua terdapat 8 buah proses, proses pertama yaitu proses Mengakses Database merupakan turunan dari proses basis data, proses ini seperti terlihat pada gambar 3.11. Kemudian dua proses berikutnya yaitu Random Data Tim, serta Pembangkitan Kromosom dan Populasi, merupakan penurunan dari proses Inisialisasi Kromosom, kedua proses ini dapat dilihat pada gambar 3.12. Dan tiga proses selanjutnya merupakan turunan dari proses Evaluasi yaitu Nilai Pinalti, Fungsi *Fitness*, dan Seleksi, seperti yang terlihat pada gambar 3.13. Dua proses terakhir merupakan turunan dari proses Operator Genetika yaitu proses Operasi Pindah Silang dan Proses Mutasi, seperti yang terlihat pada gambar 3.14.

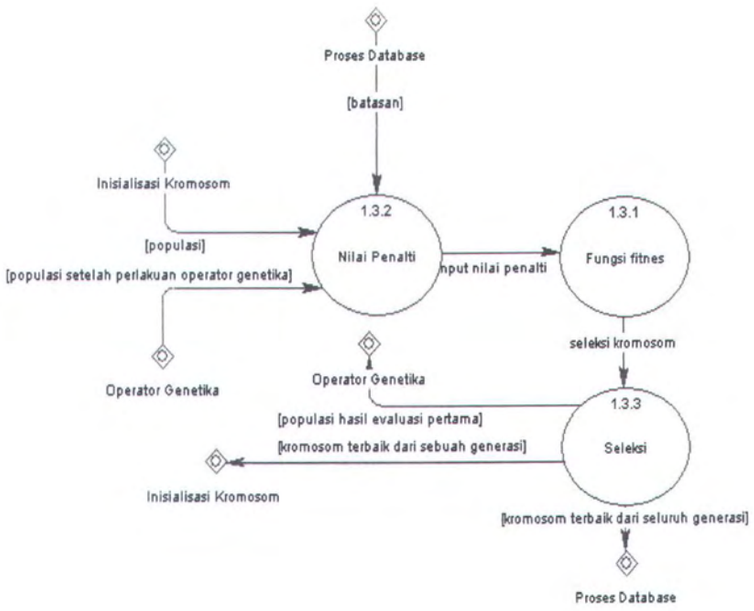
Proses Mengakses Database melakukan operasi pada langsung pada data storage yang ada. Proses input data yang dilakukan oleh user akan diinputkan kedalam basis data yang telah disediakan. Selanjutnya dari proses ini seperti telah dijelaskan diatas akan diambil data yang akan dibentuk menjadi kromosom pada proses selanjutnya. Data tersebut akan diambil dari basis data, query akan mengambil id dari klub peserta.



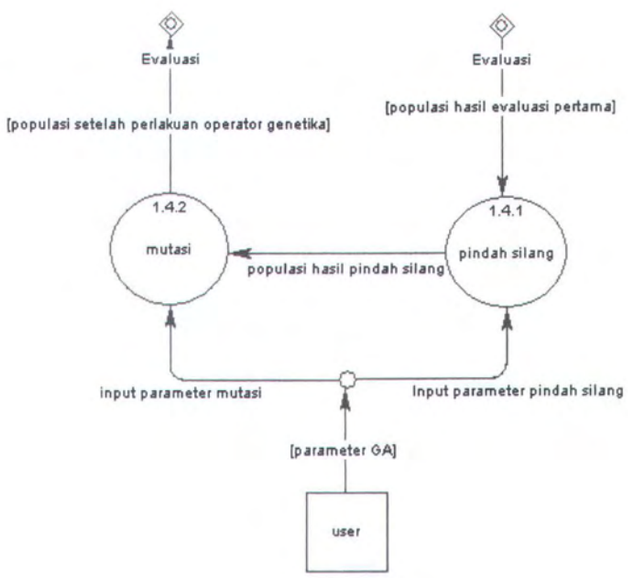
Gambar 3.11 DAD Level 2 Proses DDL dan DML



Gambar 3.12 DAD Level 2 Proses Inisialisasi Kromosom



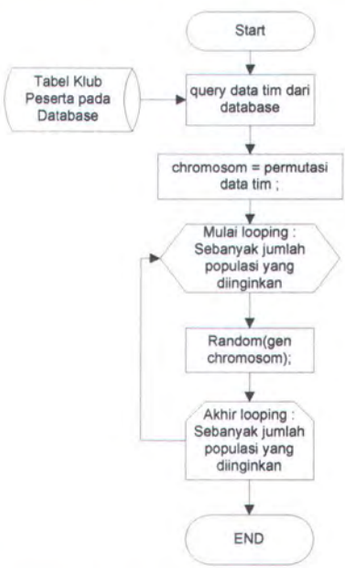
Gambar 3.13 DAD Level 2 Proses Evaluasi



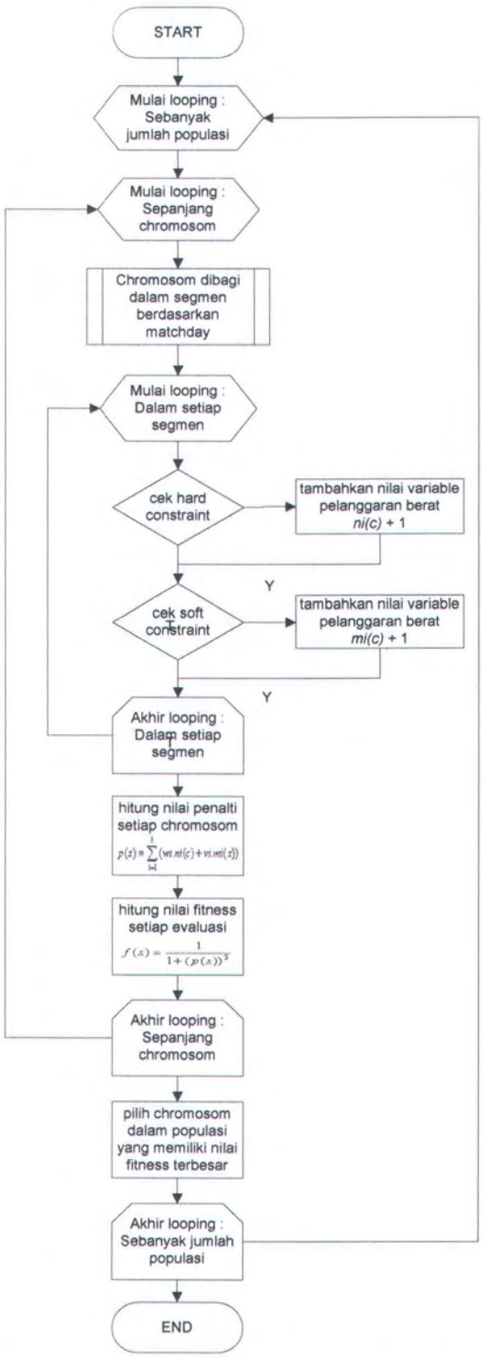
Gambar 3.14 DAD Level 2 Operator Genetika

Seperti yang terlihat pada gambar 3.12, proses random data tim akan menerima inputan dari basis data yang akan digunakan untuk mengacak tim, pengacakan akan menggunakan fungsi permutasi untuk menghasilkan tim – tim yang akan bertanding dalam sebuah generasi. Berikutnya data tim bertanding yang telah teracak akan dibangkitakan menjadi kromosom, lalu akan menjadi sebuah populasi. Besar kecilnya populasi akan ditentukan oleh parameter yang dikirimkan oleh user. Ukuran populasi akan tetap dalam sebuah eksekusi pembangkitan aplikasi keseluruhan.

Pada gambar 3.15, diagram alur level 2 dari proses Inisialisasi Kromosom memperlihatkan bahwa kromosom pertama diambil dari data peserta yang ada di basis data, kemudian kromosom, dibentuk dari permutasi antar tim, sebanyak jumlah populasi yang diinginkan.



Gambar 3.15 Diagram Alur Level 2 Inisialisasi Kromosom



Gambar 3.16 Diagram Alur Level 2 Fungsi Fitness dan Evaluasi

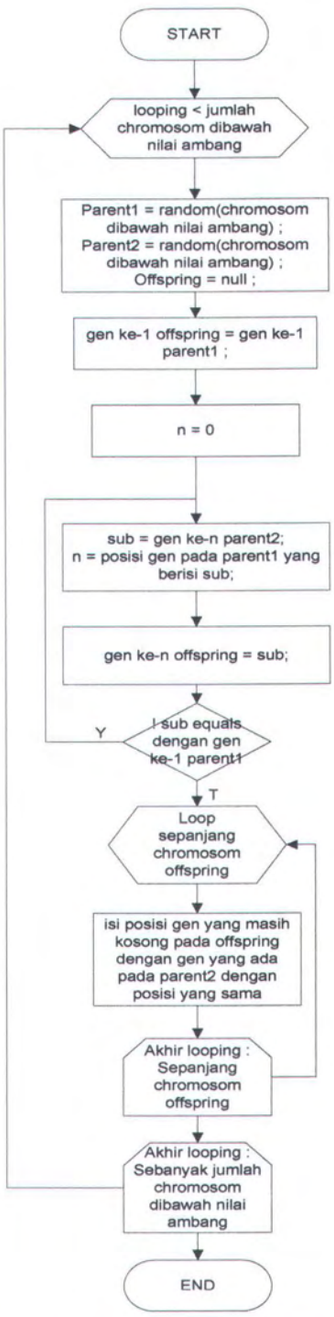
Sebuah generasi akan menghasilkan individu terbaik yang akan diikutkan lagi pada populasi generasi berikutnya. Setelah dibentuk populasi, akan dihitung nilai pinalti dari setiap individu, nilai pinalti tersebut nantinya akan dimasukkan kedalam fungsi *fitness*, setelah fungsi *fitness* diperoleh, maka akan diseleksi untuk mendapatkan individu dengan nilai terbaik. Dalam setiap eksekusi aplikasi, proses yang merupakan turunan dari proses evaluasi ini akan dilakukan sebanyak dua kali yaitu sebelum melalui proses operator genetika dan setelah melalui proses operator genetika. Diagram alur dari proses evaluasi dapat dilihat pada gambar 3.16.

Setiap kromosom dalam sebuah populasi akan dibagi menjadi beberapa segmen sesuai dengan jumlah hari pertandingan *(matchDay)*. Dalam setiap segmen akan dihitung jumlah pelanggaran yang dialami baik itu pelanggaran berat maupun pelanggaran ringan. Setelah jumlah pelanggaran berat dan ringannya diketahui maka akan dimasukkan kedalam fungsi penalti dan fungsi fitness. Setiap kromosom dalam sebuah populasi akan memiliki nilai fitness, kromosom dengan nilai fitness terbaik akan dipilih sebagai kromosom terbaik, dan akan dibandingkan dengan kromosom terbaik dari generasi yang lain, kromosom dengan nilai fitness terbaik dari seluruh generasi akan menjadi jadwal.

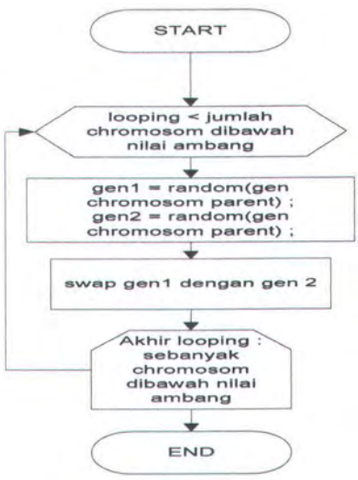
Proses pindah silang dan mutasi akan memberi perlakuan pada populasi sehingga dihasilkan populasi baru. Diagram alur dari mutasi dan crossover dapat dilihat pada gambar 3.17 dan gambar 3.18. Proses crossover menggunakan algoritma *cycle crossover.* Awalnya memilih dua buah

kromosom yang akan bertindak sebagai parent, parent didapatkan dari kromosom yang memiliki nilai fitness dibawah nilai ambang. Setelah ditentukan kedua parentnya, maka selanjutnya gen ke-1 dari parent1 akan di-copy ke gen ke- 1 dari offspring. Gen pertama pada parent2 akan dicari pada parent1 lalu gen yang ditemukan pada parent1 tersebut akan di-copy-kan sesuai dengan posisi gen tersebut pada offspring. Selanjutnya lihat gen parent2 pada posis yang sama dengan gen parent1 tersebut, cari posisi dan gen tersebut di parent1, copy-kan ke offspring, begitu seterusnya hingga gen pada parent2 sama dengan gen pertama dari parent1.

Proses mutasi dilakukan pada kromosom yang memiliki nilai fitness dibawah nilai ambang, nilai ambang ini ditentukan oleh user. Setiap kromosom akan dicari dua buah titik gen yang akan dimutasi. Gen pertama dan kedua akan dipilih secara acak, tentunya kedua gen tersebut harus berbeda. Setelah dipilih lalu kedua gen tersebut ditukar, dan akan membentuk offspring. Jumlah offspring adalah sebanyak kromosom yang berada dibawah nilai ambang, dan offspring ini yang akan diikutkan kedalam proses berikutnya.



Gambar 3.17 Diagram Alur Level 2 Crossover



Gambar 3.18 Diagram Alur Level 2 Mutasi

3.5. Perancangan Antar Muka

Bagian ini akan menjelaskan mengenai perancangan antar muka dari perangkat lunak Kick Off

3.5.1. Perancangan Antar Muka Pengolahan Data dan Penampilan Jadwal

. Antar muka dari aplikasi pengolahan data dan penampilan jadwal akan menampilkan jadwal pertandingan. Jadwal pertandingan ini dapat diakses oleh siapa saja, baik itu administrator maupun user umum. Jadwal pertandingan akan ditampilkan dengan bentuk tabel, yang berisi data dari tanggal pertandingan, stadion yang digunakan, dan klub yang bertanding sebagai *home* dan sebagai *away*. Selain itu diberikan juga fasilitas pencarian dalam jadwal pertandingan tersebut, yaitu pencarian dari suatu tanggal jadwal pertandingan, suatu stadion yang digunakan ataupun klub sebagai *home* maupun *away*. Untuk tampilan administrator selain dapat melihat jadwal

pertandingan, juga dapat menambah, mengubah, dan menghapus data dari klub peserta, tanggal pertandingan, stadion yang digunakan, wilyah dari letak homebase klub peserta. Untuk dapat melakukan proses pengolahan data tersebut harus melalui proses *login* terlebih dahulu.

3.5.2. Perancangan Antar Muka Pembangkit Jadwal

Untuk antar muka pembangkit jadwal, yang dirancang adalah antar muka masukan parameter, dan keluaran grafik serta jadwal. Struktur rancangan dari antar muka masukan terdiri dari parameter jumlah generasi, jumlah populasi, probabilitas mutasi dan probabilitas *crossover.*

BAB IV

IMPLEMETASI PERANGKAT LUNAK

4.1. Lingkungan Sistem

Lingkungan sistem dimaksudkan sebagai lingkungan penggunaan perangkat pendukung yang membentuk *prototipe*. Perangkat pendukung yang dimaksud terdiri atas perangkat keras, sebagaimana dideskripsikan Tabel 4.1, dan perangkat lunak, yang dideskripsikan Tabel 4.2.

Tabel 4.1 Perangkat Keras Lingkungan Sistem Prototipe

No	Jenis Lingkungan Sistem	Perangkat Keras
1.	Prosesor	Intel Pentium IV 2,4 GHz
2.	Memori	512 MB
3.	Hard Disk	80 GB
4.	Memori Video	64 MB

Tabel 4.2 Perangkat Lunak Lingkungan Sistem Prototipe

No	Jenis Lingkungan Sistem	Perangkat Lunak
1.	Sistem Operasi	Microsoft XP Professional
2.	DBMS *Server*	Oracle 9i Release 2
3.	Compiler	Java SDK 4.1.2
4.	Web *Scripting*	PHP v4.2.1
5.	Kompiler	GCC
6.	Web *Browser*	Internet Explorer 6.0.

Prototipe dibangun dalam lingkungan sistem dengan perangkat keras dengan spesifikasi sederhana dengan harapan, jika dapat berjalan dengan benar maka penerapan dalam lingkungan sistem dengan perangkat keras dengan spesifikasi yang lebih modern akan tidak mengalami hambatan yang berarti.

4.2. Implementasi Sistem

Bagian ini akan menjelaskan implementasi dari sistem yang telah diuraikan pada tahap perancangan. Implementasi tersebut meliputi: implementasi basis data, implementasi algoritma genetika pada pembangkit jadwal yang akan menjadi inti dari aplikasi pembangkit jadwal, dan implementasi antar muka, baik itu antar muka untuk aplikasi pembangkit jadwal, maupun aplikasi untuk menampilkan jadwal dan pengolahan data.

4.2.1. Implementasi Basis Data

Basis data pada sistem ini menggunakan Oracle 9i Release 2. Pembahasan implementasi pada subbab menjelaskan mengenai pembuatan tabel beserta atribut – atributnya, pengaturan nilai dari atribut – atribut pada table – table tersebut dan pengisian tabelnya.

Tabel 4.3. merupakan daftar kode SQL Oracle untuk pembuatan table – table yang diperlukan.

Tabel 4.3 Daftar Sript SQL Oracle untuk Tabel

Tabel	Kode SQL

Klub Peserta	`create table KLUB_PESERTA (`
	`ID_KLUB CHAR(4) not null,`
	`ID_WILAYAH CHAR(4) not null,`
	`ID_STADION CHAR(4),`
	`NAMA_KLUB VARCHAR2(20) not null,`
	`constraint PK_KLUB_PESERTA primary key (ID_KLUB));`

Stadion Peserta	`create table STADION_PESERTA (`
	`ID_STADION CHAR(4) not null,`
	`NAMA_STADION VARCHAR2(20) not null,`
	`constraint PK_STADION_PESERTA primary key (ID_STADION));`

Tanggal Pertandingan	`create table TANGGAL_PERTANDINGAN (`
	`ID_TANGGAL CHAR(4) not null,`
	`TANGGAL_PERTANDINGAN DATE not null,`
	`JUMLAH_PERTANDINGAN INTEGER,`
	`constraint PK_TANGGAL_PERTANDINGAN primary key`

	(ID_TANGGAL));
Wilayah	create table WILAYAH (ID_WILAYAH CHAR(4) not null, NAMA_WILAYAH VARCHAR2(20) not null, constraint PK_WILAYAH primary key (ID_WILAYAH));
Bobot Wilayah	create table BOBOT_WILAYAH (ID_WILAYAH_HOME CHAR(4), ID_WILAYAH_AWAY CHAR(4), BOBOT_ANTAR_WILAYAH INTEGER);
Jadwal	create table JADWAL (ID_TANGGAL CHAR(4), ID_STADION CHAR(4), ID_KLUB_HOME CHAR(4), ID_KLUB_AWAY CHAR(4));

Tabel 4.4. merupakan daftar kode SQL Oracle untuk pembuatan index pada beberapa tabel

Tabel 4.4 Daftar Sript SQL Oracle untuk Index

Tabel	Kode SQL
Bobot Wilayah	create index WIL_BOBOT_1_FK on BOBOT_WILAYAH (ID_WILAYAH_HOME ASC);
	create index WIL_BOBOT_2_FK on BOBOT_WILAYAH (ID_WILAYAH_AWAY ASC);
Jadwal	create index KLUB_JADWAL_1_FK on JADWAL (ID_KLUB_HOME ASC);
	create index STADION_JADWAL_FK on JADWAL (ID_STADION ASC);
	create index TANGGAL_JADWAL_FK on JADWAL (ID_TANGGAL ASC);
	create index KLUB_JADWAL_2_FK on JADWAL (ID_KLUB_AWAY ASC);
Klub Peserta	create index KLUB_STADION_FK on KLUB_PESERTA (ID_STADION ASC);

Dan tabel 4.5 merupakan kode SQL Oracle untuk penambahan *relational constraint* yang diperlukan pada beberapa tabel.

Tabel 4.5 Daftar Sript SQL Oracle untuk Penambahan Constraint

Index	Kode SQL
Bobot Wilayah	alter table BOBOT_WILAYAH add constraint FK_BOBOT_WI_WIL_BBT_WILAYAH_1 foreign key (ID_WILAYAH_HOME) references WILAYAH (ID_WILAYAH);
	alter table BOBOT_WILAYAH add constraint FK_BOBOT_WI_WIL_BBT_WILAYAH_2 foreign key (ID_WILAYAH_AWAY) references WILAYAH (ID_WILAYAH);
Jadwal	alter table JADWAL add constraint FK_JADWAL_KLUB_JADW_KLUB_PES_1 foreign key (ID_KLUB_HOME) references KLUB_PESERTA (ID_KLUB);
	alter table JADWAL add constraint FK_JADWAL_KLUB_JADW_KLUB_PES_2 foreign key (ID_KLUB_AWAY) references KLUB_PESERTA (ID_KLUB);
	alter table JADWAL add constraint FK_JADWAL_STADION_J_STADION_ foreign key (ID_STADION) references STADION_PESERTA (ID_STADION);
	alter table JADWAL add constraint FK_JADWAL_TANGGAL_J_TANGGAL_
Klub Peserta	alter table KLUB_PESERTA add constraint FK_KLUB_PES_KLUB_STAD_STADION_ foreign key (ID_STADION) references STADION_PESERTA (ID_STADION);
	alter table KLUB_PESERTA add constraint FK_KLUB_PES_STDPESERT_WILAYAH foreign key (ID_WILAYAH) references WILAYAH (ID_WILAYAH);

Selain proses untuk membuat tabel juga ada beberapa proses tambahan untuk membantu proses utama yang ada di dalam basis data tersebut, yaitu proses pembuatan *view*, dan proses penambahan otomatis pada tabel jadwal.

Pembuatan *view* berguna untuk melakukan pembuatan jadwal yang akan dilihat oleh pengguna. *View* yang akan dibuat terdiri dari dua buah view, view yang pertama yaitu view `KLUB_PESERTA_VIEW` yang akan membantu proses view berikutnya yaitu *view* `JADWAL_VIEW`. `JADWAL_VIEW` ini

merupakan jadwal pertandingan yang akan dilihat oleh pengguna. Untuk script dari kedua view tersebut dapat dilihat pada tabel 4.6

Tabel 4.6 Daftar Sript SQL Oracle untuk Pembuatan View

View	Kode SQL
KLUB_PESERTA_VIEW	CREATE OR REPLACE FORCE VIEW INDIE.KLUB_PESERTA_VIEW(ID_KLUB, ID_WILAYAH, ID_STADION, NAMA_KLUB) AS SELECT "ID_KLUB","ID_WILAYAH","ID_STADION","NAMA KLUB" FROM klub_peserta;
JADWAL_VIEW	CREATE OR REPLACE FORCE VIEW INDIE.JADWAL_VIEW (TANGGAL_TANDING, NAMA_STADION, HOME, AWAY) AS SELECT tgl.TANGGAL_TANDING,std.NAMA_STADION,klb1 .NAMA_KLUB as HOME,klb2.NAMA_KLUB AS AWAY FROM jadwal jdw,tanggal_pertandingan tgl, stadion_peserta std, klub_peserta klb1, klub_peserta_view klb2 WHERE jdw.ID_TANGGAL = tgl.ID_TANGGAL AND jdw.ID_STADION = std.ID_STADION AND jdw.ID_KLUB_HOME = klb1.ID_KLUB AND jdw.ID_KLUB_AWAY = klb2.ID_KLUB;

Dan untuk proses penambahan data otomatis di tabel jadwal, yaitu penambahan pada kolom stadion, berdasarkan data yag ditambahkan pada kolom id_klub_home. Proses penambahan otomatis ini akan menggunakan *trigger*. *Trigger* yang dipakai adalah *trigger after insert*, yang bernama Trg_after_jadwal. Script dari trigger ini dapat dilihat pada tabel 4.7

Tabel 4.7 Daftar Sript SQL Oracle untuk Pembuatan Trigger

View	Kode SQL
Trg_After_Jadwal	CREATE OR REPLACE TRIGGER Trg_After_Jadwal BEFORE INSERT OR UPDATE OF id_klub_home ON jadwal FOR EACH ROW DECLARE -- *local variables here* BEGIN SELECT id_stadion into :new.id_stadion FROM klub_peserta WHERE id_klub = :new.id_klub_home; END Trg_After_Jadwal;

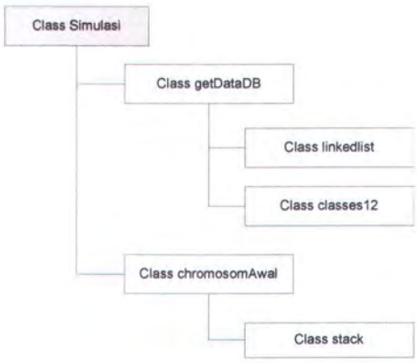
4.2.2. Implementasi Algoritma Genetika pada Pembangkitan Jadwal

Subbab ini merupakan pembahasan implementasi dari perancangan algoritma genetika pada pembangkitan jadwal, yang ada di Bab III. Untuk melakukan proses implementasi ini digunakan bahasa pemrograman Java. Dan aplikasi yang dibuat adalah *desktop application* Gambar 4.1 merupakan diagram dari class yang digunakan.

4.2.2.1. Pengambilan Data dari Basis Data

Untuk melakukan proses yang ada di algoritma genetika, diperlukan data yang berguna sebagai unsur pembentuk kromosom, data tersebut akan menjadi gen – gen pada kromosom. Data yang diambil berasal dari basis data. *Class* yang akan menangani proses ini adalah *class* getDataDB.

Untuk melakukan koneksi ke basis data, diperlukan *class* tambahan yang tidak ada pada standar dari JDK, yaitu classes12.jar. Selanjutnya dilakukan penambahan *connection string* yang mengacu pada basis data yang kita miliki, script dari *connection string* tersebut dapat dilihat pada Gambar 4.2.



Gambar 4.1 Digram Class

```
Class.forName("oracle.jdbc.driver.OracleDriver");

Connection conn =
DriverManager.getConnection("jdbc:oracle:thin:"+
dbUser+"/"+dbPass+"@"+dbIpAddress+":"+dbPort+":"+dbName);
```

Gambar 4.2 Connection String

Data yang diambil dari basis data untuk gen kromosom adalah ID dari Tabel Klub Peserta yang berada dalam Basis Data. Gambar 4.3 merupakan proses pengambilan ID Klub Peserta dari tabel klub peserta. Data tersebut akan disimpan dalam sebuah *array*, yang kemudian *array* tersebut akan di-*parsing* menuju *class* yang akan melakukan proses inisialisasi kromosom, *array* yang digunakan untuk menyimpan data tersebut ialah *MyArray.* Sebelum data disimpan dalam *array* diperlukan sebuah *linked list* untuk membantu proses perpindahan data. Keseluruhan proses ini berada pada fungsi getDataB() pada *class* getDataDB

```
public String[] getDataB(){
    try{
          ...
          st ← conn.createStatement()
          rs ← st.executeQuery("select id_klub from
klub_peserta")
          rsmd ← rs.getMetaData()
          //inisialisasi header dari tabel
hdrNum ← rsmd.getColumnCount()
header ← new String[hdrNum]

for(i=0;i<hdrNum;i++)
          {
               Simpan columnname ke header[i];
  }

//init the table data
          while(rs.next())
          {
       String[] s ← new String[hdrNum];
       for(i=0;i<hdrNum;i++)
       {
                  simpan data ke s[i]
       }
       data s simpan ke linked list
  }
       inisialisasi dataLength dengan nilai panjang linked list
inisialisasi data sebagai String[dataLength][hdrNum];
inisialisasi MyArray sebagai new String[dataLength];
for(i=0;i<dataLength;i++)
{
       simpan isi linked list ke MyData
       for(j=0;j<hdrNum;j++)
       {
              data[i][j] ← MyData[j];
              MyArray[i] ← MyData[j];
       }
}}
catch(Exception e)
{
              System.out.println("e "+e);
}
    return MyArray;  }
```

Gambar 4.3 Pseudo Code Ambil Data Untuk Inisialisai Kromosom

Setelah data diperoleh dan disimpan di *MyArray*, kemudian akan diparsing menuju *class* lainnya untuk dilakukan inisialisai kromosom dan pembentukan populasi. *Class* yang akan melakukan proses pembangkitan kromosom dan populasi itu adalah *class* chromosomAwal.

4.2.2.2. Pembangkitan Kromosom dan Populasi

Seperti yang telah dijelaskan diatas, *class* ini membentuk individu kromosom dan populasi dari setiap generasi. *Class* ini akan membentuk populasi kromosom dalam sebuah vektor, sehingga untuk membuatnya diperlukan *class* java.util.Vektor yang telah ada pada standar JDK.

Data yang didapat dari *class* getDataDB akan dipermutasikan antar data tersebut, sehingga diperoleh gen – gen untuk sebuah individu kromosom. Misal data yang ada { a, b, c, d } maka hasil permutasi dua – dua yang diperoleh adalah {ab, ac, ad, ba, bc, bd, ca, cb, cd, da, db, dc}. Hasil ini merupakan awal dari sebuah bentuk kromosom, setiap gen yang terbentuk tersebut di-*push* kedalam *stack*. Gambar 4.4 menunjukkan proses tersebut. Lalu di pop satu persatu ke dalam sebuah array bernama Gen_temp. Isi dari Gen_temp akan diacak posisinya, hal ini dapat dilihat pada gambar 4.5. Sedangkan gambar 4.6 akan memperlihatkan hasil dari pengacakan tersebut adalah sebuah kromosom dan disimpan dalam vektor bernama temp. Proses pengacakan ini akan dilakukan sebanyak jumlah individu kromosom yang diperlukan dalam sebuah kromosom. Proses permutasi, pengacakan data, penyimpanan ke vektor tersebut dilakukan dalam sebuah fungsi yang bernama initPopulasi().

```
...
protected String output;
inisialisasi s sebagi stack(1000);
inisialisasi temp sebagai Vector();

public String initPopulasi(String array_temp[], int
matchAll_temp)
...
    {
        //permutation data
        for (int i=0;i<array_temp.length;i++)
        {
            for (int j=0;j<array_temp.length;j++)
            {
                if (array_temp[i] tidak sama dengan array_temp[j])
                {
                  a ← no++;
                  output diisi dengan array_temp[i]+array_temp[j];
                  push isi output ke stack
                }
            }
        }
```

Gambar 4.4 Pseudo Code Permutasi Data untuk Membentuk Gen

```
        for (int i=0;i<matchAll_temp;i++)
        {
              pop isi stack ke Gen_temp[i];
        }
        int srclen ← Gen_temp.length;
        for(int i = 0;i<srclen;i++)
        {
              int x <- random_range dari 0 sampai srclen-1
              String tmp ← Gen_temp[i];
              Gen_temp[i] ← Gen_temp[x];
              Gen_temp[x] ← tmp;
        }
```

Gambar 4.5 Pseudo Code Acak Gen Kromosom

```
        for(int i=0;i<srclen;i++)
        {
              isi vector temp dengan Gen_temp[i];
        }
        return temp.toString();
    }
```

Gambar 4.6 Pseudo Code Pembentukan Populasi dari Kromosom

Hasil dari pembangkitan kromosom dan populasi dapat dilihat pada gambar 4.7. Pada gambar terlihat populasi terdiri dari sepuluh buah kromosom dan setiap kromosom terdiri dari dua belas buah gen.

```
[
[db, bc, ca, ab, ac, dc, da, ad, ba, cd, cb, bd],
[da, ad, bd, cb, db, dc, bc, ca, ac, ba, ab, cd],
[bc, db, dc, cb, ba, bd, ad, cd, da, ac, ca, ab],
[db, ac, cd, bc, ca, dc, ad, bd, cb, ba, ab, da],
[da, ac, db, ca, cd, bd, cb, bc, ad, ab, dc, ba],
[bd, db, bc, cd, ad, ca, ab, dc, da, ac, ba, cb],
[da, dc, cd, bd, ca, ad, ac, bc, ba, db, cb, ab],
[db, cb, cd, ca, dc, ad, bd, bc, ab, ba, ac, da],
[ac, bd, bc, db, ab, ca, dc, da, cd, ba, cb, ad],
[cd, ad, cb, bd, ab, da, bc, ca, ba, dc, ac, db]]
```

Gambar 4.7 Populasi Kromosom

4.2.2.3. Operator Genetika

Pada subbab ini akan dibahas pengimplementasian proses operator genetika yang terdiri dari proses *crossover* dan mutasi. Proses *crossover* dilakukan pada kromosom yang memiliki nilai *fitness* dibawah nilai ambang, nilai ambang ini diinputkan oleh user. Jumlah *offspring* yang dibentuk sesuai dengan jumlah kromosom *parent*, jadi apabila ada tiga buah *parent* maka *offspring* yang terbentuk berjumlah tiga buah. *Parent* yang digunakan dalam setiap pembentukan *offspring*, dipilih secara random. Kedua buah *parent* akan saling mempertukarkan gen – gen yang dimilikinya, seperti yang telah dijelaskan pada Bab III, metode yang digunakan dalam proses *crossover* adalah *cycle crossover*.

```
public void crossItOver()
{
   for(int h = 0; h < population.size(); h++)
   {
       inisialisasi parent1 sebagai Vector
       dengan nilai =getChromosomParent();
       inisialisasi parent2 sebagai Vector
       dengan nilai =getChromosomParent();

       cetak parent1;
       cetak parent2;

       inisialisasi index sebagi int dengan nilai = 0;
       inisialisasi sub sebagai String;
       inisialisasi offs sebagai Vector;
       for(int i = 0; i < parent1.length; i++)
       {
          offs.add(new String(""));
       }
       inisialisasi start sebagai string
       dengan nilai = parent1.get(index).toString();
       tambahkan kedalam offs nilai dari (index,start);
       do
       {
          sub ← parent2.get(index).toString();
          index ← parent1.indexOf(sub);
          tambahkan kedalam offs nilai dari(index,sub);
          hapus (index+1) dari offs;
       }while(sub tidak equals dengan start);
       for(int i = 0; i < parent1.length; i++)
       {
       // pengisisan gen kosong pada offs
       if(kromosom dari kromosom ke-i pada offs equals
       dengan(""))
          {
              tambahkan (i,kromosom ke-i pada parent2) ke offs
              hapus (i+1) dari offs;
          }
       }
       hapus kromosom ke-(offs.size()-1) dari offs;
       cetak("offspring ke-" + (h+1) + ": " + offs.toString() +
       " " + offs.size());
       tambahkan (offs) ke offspring;
       }
}
```

Gambar 4.8 Pseudo Code Crossover

Proses *crossover* ditangani oleh sebuah fungsi yang bernama crossItOver() seperti yang terlihat pada gambar 4.8, dan dibantu oleh sebuah fungsi untuk mendapatkan kedua *parent* yang dibutuhkan, fungsi itu yaitu getChromosomParent(). Untuk fungsi getChromosomParent()

pseudo code-nya dapat dilihat pada gambar 4.9. Sedangkan untuk hasil dari proses crossover dapat dilihat pada gambar 4.10.

```
public Vector getChromosomParent()
{
  int max ← population.size()-1;
  inisialisasi chr sebagai chromosomAwal dengan nilai =
    (chromosomAwal)population.get(Random dari 0 sampai max));
return(chr.getChromosomVector());
}
```

Gambar 4.9 Pseudo Code Fungsi getChromosomParent()

Seperti terlihat pada gambar 4.7, awalnya kedua buah *parent*, parent1 dan parent2 diambil dengan cara acak, memanggil fungsi getChromosomParent(). Dilakukan inisialisai posisi gen pada index dengan memberikan nilai nol, ini untuk mendapatkan posisi gen ke-0 dari parent1 dan parent2. Selanjutnya disiapkan vektor untuk menyimpan kromosom *offspring* sementara dengan nama variabel offs, awalnya offs diisi dengan spasi, sebanyak jumlah panjang kromosom. Kemudian dilakukanlah proses *crossover*, posisi gen ke-1 dari parent1 ditambahkan kedalam offs, lalu dicari gen pada posisi ke-1 dari parent2, gen tersebut lalu dicari di parent1 seperti pada kode berikut ini

```
sub ← parent2.get(index).toString();
index ← parent1.indexOf(sub);,
```

Kemudian ditambahkan pada offs sesuai dengan gen tersebut dan posisinya pada parent1, setelah itu dilihat lagi gen pada parent2 dengan posisi sesuai dengan posisi pada parent1 tadi, ditambahkan lagi gen tadi

dan sesuai dengan posisi yang ditemukan pada parent1 pada offs, begitu seterusnya sampai ditemukan gen pada parent2 yang sama dengan gen posisi ke-1 pada parent1.

Kemudian sisa gen yang masih kosong pada offs akan diisi dengan gen pada parent2 sesuai dengan posisi gen yang kosong pada offs. Proses ini dapat dilihat pada Gambar 4.7 pada bagian pengisisan gen kosong pada offs.

```
[bc, db, dc, cb, ba, bd, ad, cd, da, ac, ca, ab]
[db, ac, cd, bc, ca, dc, ad, bd, cb, ba, ab, da]
offspring ke-1: [bc, db, cd, cb, ba, dc, ad, bd, da, ac, ca, ab]
[cd, ad, cb, bd, ab, da, bc, ca, ba, dc, ac, db]
[db, ac, cd, bc, ca, dc, ad, bd, cb, ba, ab, da]
offspring ke-2: [cd, ac, cb, bc, ca, da, ad, bd, ba, dc, ab, db]
[ac, bd, bc, db, ab, ca, dc, da, cd, ba, cb, ad]
[bc, db, dc, cb, ba, bd, ad, cd, da, ac, ca, ab]
offspring ke-3: [ac, db, bc, cb, ab, bd, dc, cd, da, ba, ca, ad]
```

Gambar 4.10 Proses Crossover

Setelah melalui proses *crossover*, *offspring* akan melalui proses mutasi, proses mutasi ini ditangani oleh sebuah fungsi yaitu fungsi swapIt. *Pseudo code* dari fungsi swapIt dapat dilihat pada gambar 4.11. Proses mutasi ini dilakukan pada seluruh kromosom pada offspring.

Pertama – tama ditentukan posisi dari gen yang akan ditukar, posisi ini diambil secara acak, kemudian dilakukan pertukaran. Seperti pada gambar 4.10 point1 dan point2 merupakan posisi gen yang akan diambil dengan cara random, tentunya harus dicek dahulu, bahwa posisi gen yang dipertukarkan tidak boleh posisi yang sama, kerena jika sama tentunya tidak terjadi mutasi. Setalah posisi diperoleh, maka kemudian diambil gen pada posisi tersebut, gen tersebut disimpan dalam variabel section1 dan

section2, setelah itu dilakukan proses pertukaran, yaitu dengan cara menambahkan ke dalam kromosom gen section2 dengan posisi point1, dan gen section1 dengan posisi point1. Proses ini dilakukan berulang – ulang pada setiap kromosom. Proses pertukaran gen ini akan dilakukan secara berulang – ulang sesuai dengan nilai *probabilitas* mutasi. Nilai probabilitas mutasi ini akan disimpan pada *variabel* repeat

Pemilihan gen dilakukan pada gen – gen keseluruhan populasi, jadi parentnya adalah populasi dari kromosom – kromosom tersebut.

Akan diperoleh jumlah offspring yang sama dengan jumlah *parent*-nya, jadi apabila ada sepuluh parent maka jumlah *offspring*-nya juga sepuluh. *Offspring* ini kemudian akan diproses pada proses selanjutnya.

```
public void swapIt(Vector chromosom, int repeat)
{
   inisialisasi maxrandom sebagai int dengan nilai
chromosom.size()-1;
   for(int i = 0; i < repeat; i++)
   {
      inisialisasi point1 dengan nilai = getRandom(0,maxrandom);
      inisialisasi point2 sbg int dengan nilai = -1;
       do
       {
         point2 ← getRandom(0,maxrandom);
       }while(point1 == point2);
   String section1 ← chromosom.get(point1).toString();
   String section2 ← chromosom.get(point2).toString();
   tambahkan pada chromosom ←(point1,section2);
   hapus pada chromosom ← (point1+1);
   tambahkan pada chromosom ← (point2,section1);
   hapus pada chromosom ← (point2+1);
   }
}
```

Gambar 4.11 Pseudo Code Mutasi

4.2.2.4. Nilai Penalti, Fungsi Fitness dan Seleksi

Nilai pinalti diberikan untuk mengetahui jumlah pelanggaran yang ada dalam setiap kromosom. Jumlah pelanggaran tersebut diakumulasikan

kemudian dikalikan dengan konstanta pelanggaran yang telah diberikan. Pelanggaran dibedakan menjadi dua yaitu pelanggaran pada batasan yang harus dipenuhi dan pelanggaran pada batasan pertimbangan. Setiap batasan tersebut mempunyai nilai konstanta masing – masing, untuk batasan yang harus dipenuhi mempunyai sebuah konstanta sedangkan untuk batasan pertimbangan mempunyai juga harus memiliki nilai konstanta dan sesuai dengan persamaan 3.1 yang ada pada bab 3, maka persamaan nilai pinalti menjadi :

$$p(x) = \sum_{i=1}^{k} (1000.ni(c) + 500.mi(x))$$

Persamaan 4.1

```
public void evaluateFitness()
 {
Inisialisasi fitness sebagai float dengan nilai = 1;
Inisialisasi tempe sebagai string dengan nilai = "";
int counter ← (int)Math.floor((double)teamNum/2);
for(int i = 0; i < teamNum * (teamNum - 1); i += counter)
       {
           int tempCounter ← counter - 1;
           boolean isFalse ← false;
           tempe += "{";
           for(int j = i; j < i + counter; j++)
           {
               String first ← temp.get(j).toString();
       tempe += first + ",";
       for(int k = j+1; k <= j+tempCounter; k++)
       {       String second ← temp.get(k).toString();
               while(ditemukan kesalahan pada hard constraint)
               {
               fitness += 1000;
               isFalse = true;
               }do
               while(ditemukan kesalahan pada soft constraint)
               {
               fitness += 500;
               isFalse = true;
               }do}
           tempCounter--;}
tempe ← tempe.substring(0,tempe.length()-1) + "}";}}
```

Gambar 4.12 Pseudo Code Perhitungan Nilai Pinalti

dengan 1000 dan 500 merupakan nilai konstanta, sedangkan nilai *ni* akan bertambah sesuai dengan bertambahnya jumlah pelanggaran pada batasan yang harus dipenuhi, dan nilai *mi* akan bertambah sesuai dengan bertambahnya nilai pelanggaran pada batasan pertimbangan.Gambar 4.12 menunjukkan fungsi dari perhitungan nilai pinalti.

```
public static int cekHard(String first, String second)
{
      return (second.indexOf(first.charAt(0)) > -1 ||
      second.indexOf(first.charAt(1)) > -1 ? 1 : 0);
}
```

Gambar 4.13 Pseudo Code Pencarian Batasan

Untuk menghitung nilai fitness diperlukan pencarian batasan yang harus dipenuhi dan batasan yang dipertimbangkan, salah satu pencarian batasan dapat dilihat pada Gambar 4.13. Untuk perhitungan nilai *fitness*-nya, dapat dilihat pada Gambar 4.14. Pseudo code pada Gambar 4.14 tersebut merupakan implementasi dari persamaan 3.2.

```
public float getFitness()
{
      return (1/(1 + (fitness * fitness)));
}
```

Gambar 4.14. Pseudo Code Perhitungan Nilai Fitness

Sedangkan untuk hasil dari kromosom yang telah diberi nilai fitness dapat dilihat pada Gambar 4.15.

```
result:
[bd, df, de, be, fe, ba, fd, cb, ab, ef, ca, ce, ec, dc, ea, fc,
bc, ac, ad, af,ed, da, bf, fa, cf, cd, eb, db, ae, fb] =>
0.011037284
Terbaik gen. ke-10:
[fc, ce, be, bd, ec, fd, ea, de, fa, cd, ab, ef, fe, bc, ae, fb,
db, da, ed, ac,cb, cf, ba, ad, bf, dc, df, ca, eb, af] =>
0.011037284
Terbaik keseluruhan(4) :
[eb, fd, de, fc, da, fe, be, cf, bd, cd, ab, df, ec, fa, ce, ca,
bf, cb, dc, ed,ac, ea, af, fb, ad, ae, db, bc, ef, ba] =>
0.098019995
```

Gambar 4.15 Hasil Kromosom Setelah Dihitung Fitness-nya

Proses seleksi akan memilih kromosom terbaik dari setiap generasi. Selain itu proses seleksi juga dilakukan untuk memilih kromosom yang akan digunakan sebagai parent pada proses crossover. Seperti telah dijelaskan pada bab ketiga, akan menggunakan daerah sampling tetap (regular sampling space) dengan mekanisme sampling deterministik. Ini berarti populasi yang akan diproses pada generasi berikutnya dan yang dikenakan pada proses seleksi adalah populasi baru (offspring) untuk menggantikan populasi parent. Untuk mekanisme seleksinya akan dipilih kromosom terbaik dari sampling yang ada.

Pada gambar 4.16 dan 4.17, akan menjelaskan pseudo code dari pemilihan kromosom terbaik dari setiap generasi dan keseluruhan generasi. Setiap generasi akan menghasilkan sebuah kromosom terbaik, kromosom tersebut akan disimpan dan dibandingan dengan kromosom terbaik dari generasi berikutnya, kromosom terbaik dari keseluruhan generasi akan diproses untuk diubah menjadi jadwal yang akan dimasukkan kedalam basis data yaitu tabel jadwal. Untuk lebih jelasnya

tentang proses pengubahan bentuk kromosm menjadi bentuk jadwal dapat dilihat pada gambar 4.18.

```
int best = 0;
for(int i = 1; i < population.size(); i++)
{
double first = ((chromosomAwal)population.get(best)).getFitness();
double second = ((chromosomAwal)population.get(i)).getFitness();
if(first < second)best = i;
}
System.out.println("Terbaik gen. ke-" + (h+1) + ": " +
((chromosomAwal)population.get(best)).toString());
```

Gambar 4.16 Pemilihan Kromosom Terbaik Setiap Generasi

```
theBest ← kromosom terbaik;
if(theBest == null)
{
  theBest ← new chromosomAwal(
       ((chromosomAwal)population.get(best)).getChromosomVector(),M
       yArray.length);
  genbest ← h+1;
}
else
{ if(((chromosomAwal)population.get(best)).getFitness() >
  theBest.getFitness())
{ theBest ← new chromosomAwal(
       ((chromosomAwal)population.get(best)).getChromosomVect
       or(),MyArray.length);
       genbest ← h+1;
}}
Cetak theBest; //kromosom terbaik seluruh generasi
```

Gambar 4.17 Pemilihan Kromosom Terbaik Seluruh Generasi

```
dDB = new getDataDB();
String inputSQLDel = " DELETE FROM JADWAL ";
this.dDB.executeSQL(inputSQLDel);
for(int t=0;t<hasilBest.size();t++)
{ int p = t/9; //inisialisasi posisi dari home dan away
  p=p+1; //inisialisasi posisi dari home dan away
  Inisialisasi str ← new String[2];
  str[0] ← hasilBest.get(t).toString().substring(0,1);
  str[1] ← hasilBest.get(t).toString().substring(1);
  String inputSQL <- "INSERT INTO JADWAL
  (ID_TANGGAL,ID_KLUB_HOME,ID_KLUB_AWAY)
  VALUES('T"+p+"','"+
  str[0].toString()+"','"+str[1].toString()+"') ";
  String inputSQL2 ← "commit";
  eksekusi (inputSQL);
  eksekusi (inputSQL2);}
```

Gambar 4.18 Mengolah Kromosom untuk Dimasukkan ke Tabel Jadwal

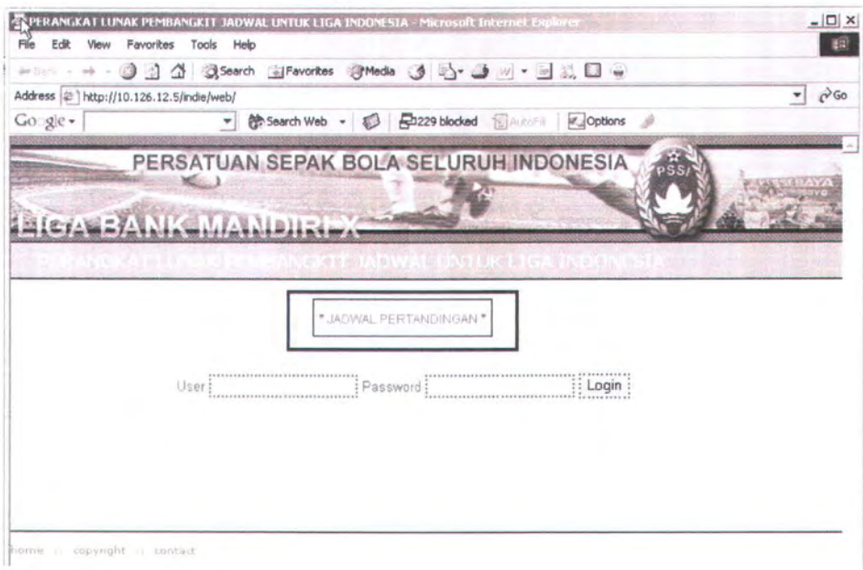
4.2.3. Implementasi Antar Muka

Untuk mempermudah dalam penggunaan aplikasi ini diperlukan suatu tampilan muka yang membantu user.

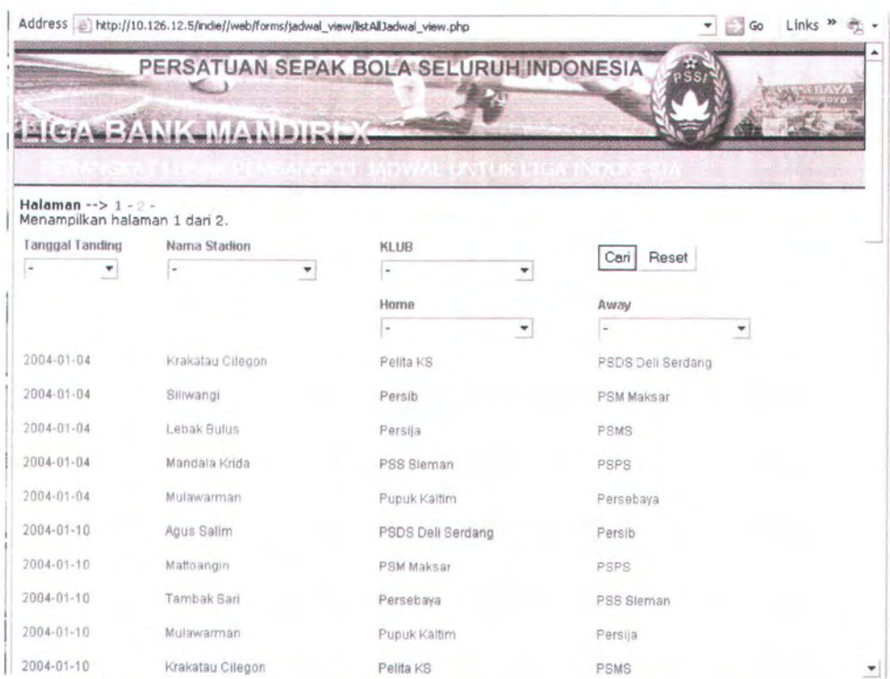
4.2.3.1. Antar Muka Pengolahan Data dan Penampilan Jadwal

Untuk aplikasi penampilan jadwal dan pengolahan basis data akan berbasis web. Antar muka akan dibagi menjadi dua, yaitu antar muka untuk user umum dan antar muka untuk administrator.

Tampilan utama akan menampilkan *link* menuuju ke halaman jadwal pertandingan dan *form login* bagi administrator. *Link* jadwal pertandingan dapat diakses oleh siapa saja. Tampilan utama dapat dilihat pada gambar 4.19 , sedangkan tampilan dari halaman jadwal pertandingan dapat dilihat pada gambar 4.20 .Untuk kode dari halaman utama dapat dilihat pada gambar 4.21 ,



Gambar 4.19 Halaman Utama Pengolahan Data dan Penampilan Jadwal



Gambar 4.20 Halaman Jadwal Pengolahan Data dan Penampilan Jadwal

```
<TR BGCOLOR="<? echo $rowBgColor; ?>" VALIGN="TOP">
      <TD>
           <? echo $thisJadwal_viewInfo->getTanggal_tanding(); ?>
            
      </TD>
      <TD>
           <? echo $thisJadwal_viewInfo->getNama_stadion(); ?>
            
      </TD>
      <TD>
           <? echo $thisJadwal_viewInfo->getHome(); ?>
            
      </TD>
      <TD>
           <? echo $thisJadwal_viewInfo->getAway(); ?>
            
      </TD>
</TR>
```

Gambar 4.21 Kode untuk Halaman Jadwal

Untuk menampilkan jadwal pertandingan, akan dipanggil view yang ada di basis data, kemudian ditampilkan dengan menggunakan tabel pada halaman jadwal tersebut. Beberapa fungsi dari proses untuk menampilkan jadwal dapat dilihat pada gambar 4.22 , 4.23 , 4.24 ,

```
define("TABLE_JADWAL_VIEW","jadwal_view");
// Primary Key untuk Table jadwal_view
define("FIELD_JADWAL_VIEW_PK","tanggal_tanding");
// Field Name Mapping untuk Table jadwal_view
define("FIELD_JADWAL_VIEW_TANGGAL_TANDING","tanggal_tanding");
define("FIELD_JADWAL_VIEW_NAMA_STADION","nama_stadion");
define("FIELD_JADWAL_VIEW_HOME","home");
define("FIELD_JADWAL_VIEW_AWAY","away");
// Display Labels dari Field untuk Table jadwal_view
define("LABEL_JADWAL_VIEW_TANGGAL_TANDING","Tanggal Tanding");
define("LABEL_JADWAL_VIEW_NAMA_STADION","Nama Stadion");
define("LABEL_JADWAL_VIEW_HOME","Home");
define("LABEL_JADWAL_VIEW_AWAY","Away");
```

Gambar 4.22 Definisi dari View yang Ditampilkan

```
function getJadwal_viewByPK($pkValue)
{  $searchItems[] = new
searchItem(FIELD_JADWAL_VIEW_TANGGAL_TANDING,$pkValue);
   $dbResultsInfo =  $this->searchJadwal_view($searchItems);
   $resultsArray =  $dbResultsInfo->getResultsArray();
 if (count($resultsArray)==0) {return false;}
 else if (count($resultsArray)>1)
 {  $thisError = new errorHandler();
    $thisError->setQuitProgram(true);
    $thisError->setDisplayError(true);
    $thisError->setEmailAdmin(true);
    $thisError->setUserErrorMessage("An error  has occured while
this application was trying to do some operation on the database.
");
    $thisError->setProgramErrorMessage("Should get only one row of
data. Program retreived more than row--> ".count($resultsArray)."
Error Occured when trying to get Info by Id on Table -->Jadwal_view
ID --> : ".$pkValue);
    $thisError->setErrorPage($_SERVER['PHP_SELF']);
    $thisError->handleError();
    return false;     }
else if (count($resultsArray==1)) { return $resultsArray[0]; } }
```

Gambar 4.23 Fungsi untuk Mengambil Isi Jadwal

```
function populateJadwal_viewInfoFromResultSet($result)
{ $thisJadwal_viewInfo = new Jadwal_viewInfo();
  $thisJadwal_viewInfo->
       setTanggal_tanding($result->fields
              [FIELD_JADWAL_VIEW_TANGGAL_TANDING]);
  $thisJadwal_viewInfo->
       setNama_stadion($result->fields
              [FIELD_JADWAL_VIEW_NAMA_STADION]);
  $thisJadwal_viewInfo->
       setHome($result->fields[FIELD_JADWAL_VIEW_HOME]);
  $thisJadwal_viewInfo->
       setAway($result->fields[FIELD_JADWAL_VIEW_AWAY]);
  return $thisJadwal_viewInfo;}
```

Gambar 4.24 Fungsi untuk Menampilkan Jadwal pada Halaman Web

Seperti terlihat pada gambar 4.20 pada halaman jadwal terdapat fasilitas pencarian, yaitu pencarian dari isi jadwal tersebut. Pencarian dilakukan berdasarkan tanggal pertandingan, stadion yang digunakan, klub, klub sebagai *home*, klub sebagai *away*. Pada gambar 4.25 akan diperlihatkan fungsi dari proses pencarian

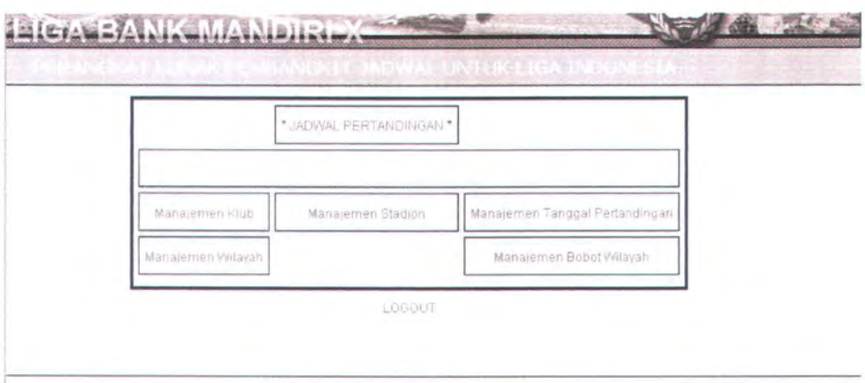
```
Function searchJadwal_view($searchItems="",$start="",
$limit="",$fieldsToReturn="",$sortByField="",$sortOrder="")
{ $thisSearchUtils = new searchUtils();
  $thisSearchUtils->setSearchItems($searchItems);
  $thisSearchUtils->setStart($start);
  $thisSearchUtils->setLimit($limit);
  $thisSearchUtils->setFieldsToReturn($fieldsToReturn);
  $thisSearchUtils->setOrderByField($sortByField);
  $thisSearchUtils->setOrderDirection($sortOrder);
  $thisSearchUtils->setTable(TABLE_JADWAL_VIEW);
  $sql = $thisSearchUtils->getSearchSQL();
  // Eksekusi Query
  $thisDatabaseQuery = new databaseQuery();
  $thisDatabaseQuery->setSqlQuery($sql);
  $thisDatabaseQuery->setStart($start);
  $thisDatabaseQuery->setLimit($limit);
  $result = $thisDatabaseQuery->executeQuery();
  $thisDatabaseResultsInfo = new databaseResultsInfo();
  if ($result==false){ return $thisDatabaseResultsInfo;} else
  {  // Mendapatkan hasil dari query
     $resultSet = $thisDatabaseQuery->getResultSet();
     $thisJadwal_viewInfoArray = array();
     $thisJadwal_viewInfo = new Jadwal_viewInfo();
  while (!$resultSet->EOF)
  { $thisJadwal_viewInfo = $this->
       populateJadwal_viewInfoFromResultSet($resultSet);
       $thisJadwal_viewInfoArray[] = $thisJadwal_viewInfo;
      $resultSet->MoveNext();
  }
$thisDatabaseResultsInfo->setResultsArray(
$thisJadwal_viewInfoArray);
$thisDatabaseResultsInfo->setTotalNumberOfRows($thisDatabaseQuery->
      getTotalRows());
$thisDatabaseResultsInfo->setStart($thisDatabaseQuery->getStart());
$thisDatabaseResultsInfo->setLimit($thisDatabaseQuery->getLimit());
$thisDatabaseResultsInfo->setSortedBy($thisSearchUtils->
      getOrderByField());
$thisDatabaseResultsInfo->setSortOrder($thisSearchUtils->
      getOrderDirection());
return $thisDatabaseResultsInfo; } }
```

Gambar 4.25 Fungsi untuk Melakukan Pencarian

Untuk tampilan bagi administrator diperlukan proses login terlebih dahulu, proses login ini terdapat di halaman utama, seperti terlihat pada gambar 4.19 . Setelah melakukan proses login, administrator akan ditampilkan menu pengolahan data, berisi tentang tabel – tabel pada basis data yang dapat diolah, dapat dilihat pada gambar 4.27 .Untuk kode dari proses login dapat dilihat pada gambar 4.26 ,

```
session_start();
if ($_POST["action"] == "login")
{
if (($_POST["txtuser"] == "admin") && ($_POST["txtpass"] ==
"admin"))
{
      $_SESSION["login"] = "admin";
}
else $msg = "<table align=center><tr><td>Autentifikasi
Gagal!</td></tr></table><br/>";
}
if ($_REQUEST["action"] == "logout") {
$_SESSION["login"] = "";
unset($_SESSION["login"]);
}
```

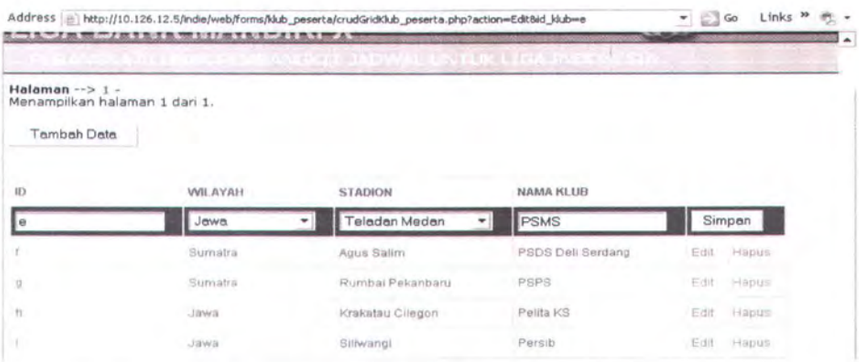
Gambar 4.26 Fungsi untuk Melakukan Pencarian



Gambar 4.27 Fungsi untuk Melakukan Pencarian

Administrator akan melakukan proses penambahan, pengubahan dan penghapusan data pada menu – menu yang ada, proses tersebut akan berhubungan langsung dengan basis data yang ada. Gambar 4.28 , akan menunjukkan halaman web dari pengolahan data tersebut, sedangkan gambar 4.29 , 4.30 , 4.31 , dan 4.32 ,akan memperlihatkan kode dari proses penambahan, pengubahan, penghapusan data pada menu klub

peserta dan potongan kode dari tampilan halaman pengolahan data tersebut.



Gambar 4.28 Fungsi untuk Melakukan Pencarian

```
$sql .= "UPDATE ".TABLE_KLUB_PESERTA;
$sql .= " SET ";
$sql .=  FIELD_KLUB_PESERTA_ID_KLUB." = '".$thisKlub_pesertaInfo->
      getId_klub()."' , ";
$sql .= FIELD_KLUB_PESERTA_ID_WILAYAH." ='".$thisKlub_pesertaInfo->
      getId_wilayah()."' , ";
$sql .= FIELD_KLUB_PESERTA_ID_STADION." ='".$thisKlub_pesertaInfo->
      getId_stadion()."' , ";
$sql .= FIELD_KLUB_PESERTA_NAMA_KLUB." = '".$thisKlub_pesertaInfo->
      getNama_klub()."'";
$sql .="WHERE".FIELD_KLUB_PESERTA_PK."  ='".$thisKlub_pesertaInfo->
      getId_klub()."'";
$thisDatabaseQuery = new databaseQuery();
$thisDatabaseQuery->setSqlQuery($sql);
$result = $thisDatabaseQuery->executeQuery();
```

Gambar 4.29 Kode untuk Proses Pengubahan Data

```
$sql = "";
   $sql .= "DELETE FROM ".TABLE_KLUB_PESERTA;
   $sql .= " WHERE ";
   $sql .= FIELD_KLUB_PESERTA_ID_KLUB." = '".$thisKlub_pesertaInfo-
>getId_klub()."' ";
$thisDatabaseQuery = new databaseQuery();
$thisDatabaseQuery->setSqlQuery($sql);
$result = $thisDatabaseQuery->executeQuery();
```

Gambar 4.30 Kode untuk Proses Penghapusan Data

```
$sql .= "INSERT INTO ".TABLE_KLUB_PESERTA;
$sql .= "(".FIELD_KLUB_PESERTA_ID_KLUB.",
       ".FIELD_KLUB_PESERTA_ID_WILAYAH." , ";
$sql .= "".FIELD_KLUB_PESERTA_ID_STADION." ,
       ".FIELD_KLUB_PESERTA_NAMA_KLUB."";
$sql .= ")";
$sql .= " VALUES ";
$sql .= "('".$thisKlub_pesertaInfo->getId_klub()."' ,
       '".$thisKlub_pesertaInfo->getId_wilayah()."' ,
       '".$thisKlub_pesertaInfo->getId_stadion()."' , ";
$sql .= "'".$thisKlub_pesertaInfo->getNama_klub()."')";
$thisDatabaseQuery = new databaseQuery();
$thisDatabaseQuery->setSqlQuery($sql);
$result = $thisDatabaseQuery->executeQuery();
```

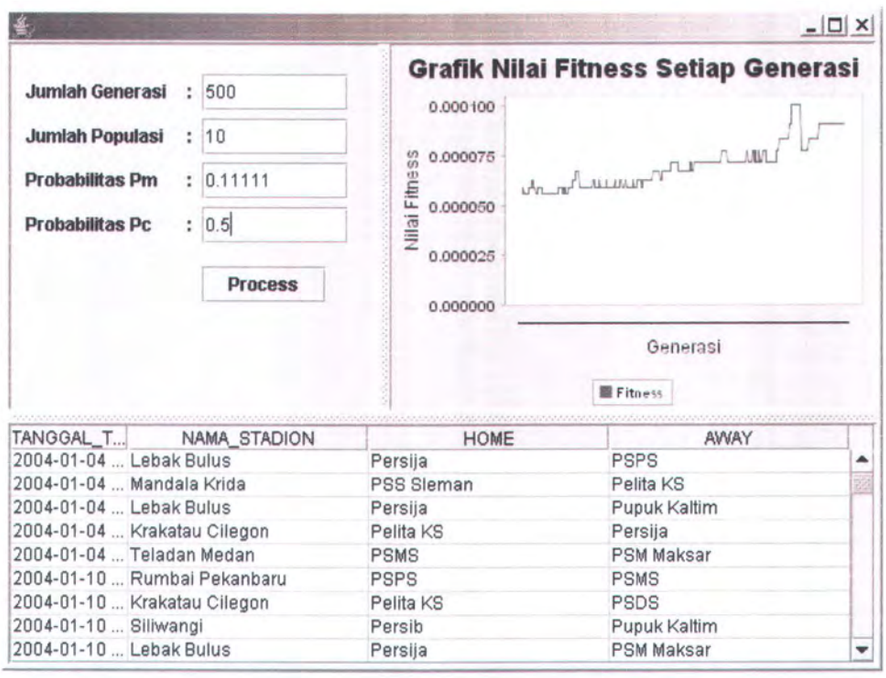
Gambar 4.31 Kode untuk Proses Penambahan Data

```
<br><? } ?>
<TABLE CELLPADDING="4" CELLSPACING="0" BORDER="1"
ID="tableKlub_pesertaId" WIDTH="100%">
<TR><TD><b><? echo LABEL_KLUB_PESERTA_ID_KLUB; ?></b></TD><TD>
        <b><? echo LABEL_KLUB_PESERTA_ID_WILAYAH; ?></b></TD><TD>
        <b><? echo LABEL_KLUB_PESERTA_ID_STADION; ?></b></TD><TD>
        <b><? echo LABEL_KLUB_PESERTA_NAMA_KLUB; ?></b></TD>
        <TD colspan="2"> </TD></TR><?php
for ($a=0;$a<count($thisKlub_pesertaInfoArray);$a++){
$activeEdit = false;
$thisKlub_pesertaInfo = $thisKlub_pesertaInfoArray[$a];
if (($a%2)==0) { $rowBgColor = ROW_COLOR1; } else { $rowBgColor =
ROW_COLOR2; }
if ($activeEditId == $thisKlub_pesertaInfo->getId_klub()) {
$rowBgColor = "blue"; $activeEdit=true; }?>
<? if ($activeEdit) { ?>
<form name="editGrid" method="post" action="<? echo
$_SERVER['PHP_SELF']; ?>"><? } ?>
<TR BGCOLOR="<? echo $rowBgColor; ?>" VALIGN="TOP">
<? if (!$activeEdit) {
?><TD> <? echo $thisKlub_pesertaInfo->getId_klub(); ?> </TD>
<? } else { ?> <TD>
<INPUT TYPE="text" NAME="thisId_klubField" VALUE="<? echo
$thisKlub_pesertaInfo->getId_klub(); ?>"> </TD><? } ?>
<? if (!$activeEdit) { ?><TD>
```

Gambar 4.32 Potongan Kode dariTampilan Halaman Pengolahan Data

4.2.3.2. Antar Muka Pembangkit Jadwal

Dalam aplikasi pembangkit jadwal, pemberian masukan parameter dan melihat hasil dari pembangkitan jadwal, yaitu grafik dan tabel jadwal, dapat dilihat pada gambar 4.33



Gambar 4.33 Tampilan Antar Muka Pembangkit Jadwal

Pada tampilan antar muka terdapat empat buah *textbox* yang berguna untuk memberikan masukan pada parameter genetika, yaitu jumlah generasi, jumlah populasi, probabilitas mutasi, dan probabilitas *crossover*. Untuk tampilan grafik menggunakan komponen JFreeChart, yang akan menampilkan grafik hubungan antara jumlah generasi dengan nilai fitness dari kromosom terbaik setiap generasi. Dan tampilan jadwal

menggunakan komponen JTable. Tabel ini akan menampilkan tanggal pertandingan, stadion yang digunakan, dan tim yang bertanding.

BAB V

UJI COBA DAN EVALUASI

5.1. Uji Coba

Uji coba akan dilakukan untuk melihat keberhasilan dan kecepatan pada lingkungan sistem yang telah disebutkan dalam subbab 4.1, Lingkungan Sistem. Uji coba pertama perangkat lunak yang dibuat, akan dilakukan dengan cara merubah, jumlah populasi, probabilitas mutasi dan probabilitas *crossover*.

Tabel 5.1 akan menjelaskan tentang ujicoba yang dilakukan dengan jumlah data sepuluh:

Tabel 5.1 Uji Coba Pertama Dengan Jumlah Data 10 Jumlah Generasi Tetap

Generasi	Populasi	Pm	Pc	Coba	Fitness	Generasi ke-	Waktu
100	10	0.1	0.5	1	6.50E-05	65	8.312
				2	7.14E-05	89	8.953
100	10	0.2	0.5	1	7.14E-05	47	13.578
				2	6.25E-05	87	14.219
100	10	0.4	0.5	1	5.88E-05	5	26.765
				2	7.14E-05	77	26.156
100	10	0.8	0.5	1	7.14E-05	49	48.438
				2	6.25E-05	6	52.422
100	20	0.1	0.5	1	7.69E-05	75	16.047
				2	7.14E-05	92	14.718
100	20	0.2	0.5	1	9.09E-05	85	25.953
				2	7.14E-05	73	26.156
100	20	0.4	0.5	1	8.33E-05	70	49.329
				2	1.00E-04	83	46.125
100	20	0.8	0.5	1	8.33E-05	97	96.61
				2	1.00E-04	99	93.937
1000	10	0.111	0.25	1	6,50E-05	227	119,281 s
				2	6,25E-05	589	110,641 s
1000	10	0.222	0.25	1	7.14E-05	826	191,844 s
				2	6.25E-05	754	198,36 s
1000	10	0.444	0.25	1	6,2E-05	79	199,65 s

				2	6,25E-05	77	206,156 s
1000	10	0.888	0.25	1	7.14E-05	498	684,38 s
				2	6.25E-05	643	524,22 s
1000	20	0.111	0.25	1	7.69E-05	752	126,047 s
				2	7.14E-05	92	127,18 s
1000	20	0.222	0.25	1	9.09E-05	711	159,53 s
				2	7.14E-05	734	161,56 s
1000	20	0.444	0.25	1	8.33E-05	701	199,56 s
				2	1.00E-04	836	261,25 s
1000	20	0.888	0.25	1	8.33E-05	973	536.61 s
				2	1.00E-04	990	393,7 s
1000	10	0.111	0.5	1	5.00E-04	712	159,828 s
				2	1.67E-04	1000	124,69 s
1000	10	0.222	0.5	1	5.00E-04	742	100,359 s
				2	4.90E-04	793	82,094 s
1000	10	0.444	0.5	1	4.95E-04	609	128.61 s
				2	5.00E-04	627	130.25 s
1000	10	0.888	0.5	1	4.99E-04	391	151,172 s
				2	5.00E-04	773	390.844
1000	20	0.111	0.5	1	5.00E-04	250	140,563 s
				2	5.00E-04	419	141,88 s
1000	20	0.222	0.5	1	5.00E-04	215	173,241 s
				2	5.00E-04	258	175,719 s
1000	20	0.444	0.5	1	5.00E-04	255	110,188 s
				2	5.00E-04	258	106,187 s
1000	20	0.888	0.5	1	4.95E-04	473	703,515 s
				2	5.00E-04	218	181,219 s

Ujicoba yang dilakukan, seperti yang diperlihatkan oleh tabel 5.1, menggunakan jumlah data tim peserta yang tetap. Tapi dengan kombinasi dari jumlah generasi, jumlah populasi, probabilitas mutasi dan probabilitas crossover. Nilai fitness terbaik ditemukan pada jumlah generasi 1000, jumlah populasi 10 dan 20, dan nilai probabilitas mutasi dan crossover tidak memegang peranan.

Tabel 5.2 akan menjelaskan tentang ujicoba yang dilakukan dengan jumlah data besar, sesuai dengan jumlah data peserta liga indonesia:

Tabel 5.1 Uji Coba Pertama Dengan Jumlah Data Besar Jumlah Generasi Tetap

Generasi	Populasi	Pc	Pm	Coba	Fitnes	Generasi ke-	Waktu
1000	10	0,25	0,1	1	5,00E-05	388	280.188 s
				2	4,76E-05	20	279.422 s
				3	4,76E-05	107	278.063 s
				4	4,76E-05	29	277.859 s
				5	4,76E-05	103	279.937 s
1000	10	0,25	0,2	1	4,76E-05	65	518.61 s
				2	4,76E-05	21	515.359 s
				3	4,76E-05	39	522.234 s
				4	5,00E-05	311	522.485 s
				5	4,76E-05	44	521.812 s
1000	10	0,25	0,3	1	4,76E-05	80	751.235 s
				2	4,76E-05	84	730.407 s
				3	4,76E-05	7	742.937 s
				4	4,76E-05	10	743.031 s
				5	4,76E-05	19	737.703 s
1000	10	0,25	0,4	1	4,76E-05	61	975.047 s
				2	4,76E-05	8	974.594 s
				3	5,00E-05	494	974.969 s
				4	5,00E-05	849	969.687 s
				5	4,76E-05	63	975.344 s
1000	10	0,25	0,5	1	4,76E-05	14	1222.016 s
				2	4,76E-05	76	1214.266 s
				3	4,76E-05	26	1203.109 s
				4	5,00E-05	939	1207.016 s
				5	5,00E-05	146	1198.64 s
1000	10	0,25	0,6	1	4,76E-05	34	1391.671 s
				2	4,76E-05	32	1391.516 s
				3	5,00E-05	812	1391.25 s
				4	4,76E-05	72	1440.203 s
				5	4,76E-05	13	1440.125 s
1000	10	0,25	0,7	1	4,76E-05	40	1699.829 s
				2	4,76E-05	11	1685.359 s
				3	4,76E-05	31	1659.469 s
				4	4,76E-05	13	1652.297 s
				5	5,00E-05	253	1660.641 s
1000	10	0,25	0,8	1	4,76E-05	1	1894.734 s
				2	5,00E-05	606	1886.203 s
				3	4,76E-05	1	1896.406 s
				4	4,76E-05	17	1893.141 s
				5	4,76E-05	40	1896.484 s
1000	20	0,25	0,1	1	5,88E-05	690	516.031 s
				2	4,76E-05	8	541.297 s
				3	4,76E-05	13	540.875 s

				4	5,26E-05	527	532.766 s
				5	5,26E-05	24	540.672 s
1000	20	0,25	0,2	1	4,76E-05	4	1018.313 s
				2	5,00E-05	240	1016.453 s
				3	5,00E-05	14	1022.156 s
				4	5,55E-05	789	1018.328 s
				5	5,55E-05	34	1018.512 s
1000	20	0,25	0,3	1	5,26E-05	26	1445.469 s
				2	5,00E-05	366	1451.172 s
				3	5,00E-05	316	1463.265 s
				4	5,00E-05	53	1458.875 s
				5	5,26E-05	110	1466.188 s
1000	20	0,25	0,4	1	5,00E-05	608	1958.141 s
				2	5,00E-05	698	1952.125 s
				3	5,00E-05	83	1981.016 s
				4	5,26E-05	898	1976.234 s
				5	4,76E-05	6	1978.734 s
1000	20	0,25	0,5	1	5,00E-05	365	2460.687 s
				2	5,00E-05	84	2493.922 s
				3	5,26E-05	561	2540.188 s
				4	5,00E-05	120	1924.094 s
				5	5,00E-05	255	2588.547 s
1000	20	0,25	0,6	1	5,00E-05	391	2273,563 s
				2	5,26E-05	247	2913.594 s
				3	5,00E-05	400	2850.0 s
				4	5,00E-05	118	2448.937 s
				5	5,00E-05	669	2918.719 s
1000	20	0,25	0,7	1	5,26E-05	139	2642,546 s
				2	5,00E-05	52	3338.875 s
				3	5,00E-05	156	2681.391s
				4	5,00E-05	266	3385.735 s
				5	5,26E-05	198	2157.991s
1000	20	0,25	0,8	1	5,26E-05	868	3778.125 s
				2	5,00E-05	36	2708.265 s
				3	5,00E-05	44	3819.171 s
				4	5,00E-05	63	3778.047 s
				5	5,00E-05	27	2732.063 s
1000	10	0,5	0,1	1	5,55E-05	993	223.516 s
				2	5,26E-05	895	231.625 s
				3	5,88E-05	688	222.562 s
				4	5,26E-05	436	224.828 s
				5	5,88E-05	699	221.531s
1000	10	0,5	0,2	1	5,55E-05	910	423.937 s
				2	5,88E-05	868	417.328 s
				3	5,00E-05	498	430.797 s
				4	5,88E-05	768	405.469 s

				5	5,26E-05	315	491.578 s
1000	10	0,5	0,3	1	5,26E-05	466	605.266 s
				2	7,69E-05	911	550.531 s
				3	5,55E-05	768	608.937 s
				4	5,26E-05	152	719,594 s
				5	5,26E-05	914	730.828 s
1000	10	0,5	0,4	1	7,14E-05	761	875.735 s
				2	5,26E-05	652	813.438 s
				3	5,26E-05	538	792.656 s
				4	7,14E-05	845	736.468 s
				5	5,26E-05	581	805.703 s
1000	10	0,5	0,5	1	5,00E-05	609	1015.828 s
				2	5,26E-05	991	979.907 s
				3	5,55E-05	883	969.688 s
				4	5,00E-05	47	994.625 s
				5	5,00E-05	300	983.5 s
1000	10	0,5	0,6	1	5,00E-05	80	1171.75 s
				2	5,55E-05	726	1164.75 s
				3	5,88E-05	955	1270.453 s
				4	6,25E-05	447	1190.766 s
				5	6,25E-05	912	1277.266 s
1000	10	0,5	0,7	1	5,26E-05	856	1345.844 s
				2	5,55E-05	981	1380.515 s
				3	5,00E-05	813	1352.297 s
				4	5,26E-05	441	1378.828 s
				5	5,55E-05	729	1668.031 s
1000	10	0,5	0,8	1	4,76E-05	48	1942.64 s
				2	4,76E-05	19	1911.109 s
				3	6,25E-05	861	1709.344 s
				4	5,55E-05	890	1774.172 s
				5	6,25E-05	996	1743.297 s
1000	20	0,5	0,1	1	7,69E-05	817	489.438 s
				2	6,67E-05	939	485.157 s
				3	7,69E-05	848	481.484 s
				4	8,33E-05	972	470.765 s
				5	6,67E-05	744	502.187 s
1000	20	0,5	0,2	1	8,33E-05	958	880.953 s
				2	1,00E-04	905	895.593 s
				3	1,11E-04	950	886.828 s
				4	6,67E-05	832	930.86 s
				5	1,25E-04	932	682.391 s
1000	20	0,5	0,3	1	8,33E-05	977	1097.5 s
				2	9,09E-05	925	966.734 s
				3	9,09E-05	940	1041.094 s
				4	9,09E-05	964	1015.922 s
				5	3,33E-04	969	929.828 s

1000	20	0,5	0,4	1	8,33E-05	996	1332.469 s
				2	9,09E-05	883	1350.359 s
				3	9,09E-05	924	1343.687 s
				4	8,33E-05	836	1445.953 s
				5	1,00E-04	804	1300.063 s
1000	20	0,5	0,5	1	9,09E-05	861	1749.359 s
				2	8,33E-05	991	1666.703 s
				3	1,11E-04	770	1599.016 s
				4	9,09E-05	858	1683.203 s
				5	1,00E-04	856	1593.25 s
1000	20	0,5	0,6	1	7,69E-05	948	2007.125 s
				2	1,11E-04	839	1865.031 s
				3	9,09E-05	956	1867.734 s
				4	1,67E-04	958	1817.422 s
				5	7,69E-05	979	2076.5 s
1000	20	0,5	0,7	1	7,69E-05	645	2289.859 s
				2	9,09E-05	871	2215.359 s
				3	7,69E-05	969	2427.047 s
				4	1,00E-04	923	2306.578 s
				5	1,25E-04	859	2110.438 s
1000	20	0,5	0,8	1	9,09E-05	939	2595.922 s
				2	2,50E-04	979	2366.266 s
				3	1,00E-04	925	2535.5 s
				4	7,69E-05	995	2764.078 s
				5	9,09E-05	907	2599.844 s

Seperti juga pada tabel 5.1, pada tabel 5.2 dilakukan dengan jumlah data tetap, dan dilakukan kombinasi pada jumlah populasi, nilai probabilitas crossover dan nilai probabilitas mutasi. Fitness terbaik ditemukan pada jumlah generasi 1000, jumlah populasi 20, nilai probabilitas crossover 0,5 dan nilai probabilitas permutasi 0,3, mulai generasi ke 969.

Uji coba kedua akan dilakukan terhadap jumlah data yang berbeda – beda, tetapi dengan nilai parameter genetika tetap.

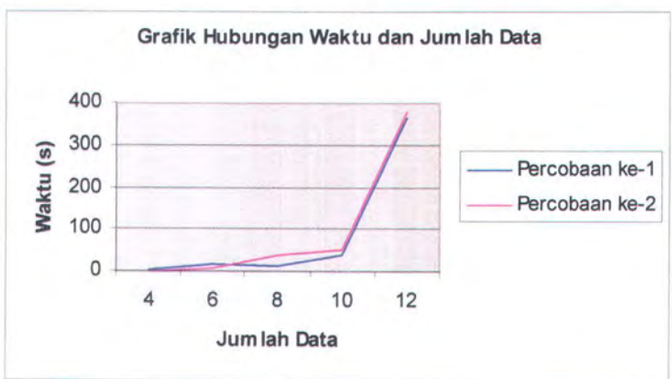
- Jumlah generasi yang dilakukan sebanyak 1000 kali
- Jumlah populasi sebanyak 20 buah.

- Probabilitas mutasi yang digunakan sebesar 0,111, dan probabilitas crossover sebesar 0,5.

Hasil dari pengujian dapat dilihat pada tabel 5.2.

Tabel 5.2 Uji Coba Kedua

Data Peserta	Per-cobaan	Fitness	Gen ke-	Waktu
4	1	0.5	2	4.047
	2	0.5	2	2
6	1	5.00E-04	6	15.782
	2	0.5	9	7.688
8	1	5.00E-04	32	14
	2	4.95E-04	41	41
10	1	5.00E-04	250	40.563
	2	5.00E-04	419	54.188
12	1	9.09E-05	960	3667
	2	9.09E-05	875	176.674



Gambar 5.1 Grafik Hubungan Waktu dan Jumlah Data

Dan perbandingan jadwal antara data berjumlah empat dan data berjumlah dua belas, dapat dilihat pada gambar 5.2, 5.3, 5.4 untuk data dengan jumlah besar dan 5.5 untuk data dengan jumlah kecil.

Halaman --> 1 - 2 -
Menampilkan halaman 1 dari 2.

Tanggal Tanding	Nama Stadion	KLUB	Cari / Reset
-	-	-	
		Home	Away
		-	-
2004-01-04	Krakatau Cilegon	Pelita KS	PSDS Deli Serdang
2004-01-04	Siliwangi	Persib	PSM Maksar
2004-01-04	Lebak Bulus	Persija	PSMS
2004-01-04	Mandala Krida	PSS Sleman	PSPS
2004-01-04	Mulawarman	Pupuk Kaltim	Persebaya
2004-01-10	Agus Salim	PSDS Deli Serdang	Persib
2004-01-10	Mattoangin	PSM Maksar	PSPS
2004-01-10	Tambak Sari	Persebaya	PSS Sleman
2004-01-10	Mulawarman	Pupuk Kaltim	Persija
2004-01-10	Krakatau Cilegon	Pelita KS	PSMS
2004-01-14	Krakatau Cilegon	Pelita KS	Pupuk Kaltim
2004-01-14	Mandala Krida	PSS Sleman	PSDS Deli Serdang
2004-01-14	Mattoangin	PSM Maksar	PSMS
2004-01-14	Lebak Bulus	Persija	Persebaya
2004-01-14	Siliwangi	Persib	PSPS
2004-01-18	Teladan Medan	PSMS	PSS Sleman
2004-01-18	Agus Salim	PSDS Deli Serdang	PSM Maksar
2004-01-18	Rumbai Pekanbaru	PSPS	Pupuk Kaltim
2004-01-18	Tambak Sari	Persebaya	Persija
2004-01-18	Siliwangi	Persib	Pelita KS
2004-01-24	Teladan Medan	PSMS	Persija
2004-01-24	Krakatau Cilegon	Pelita KS	Persib
2004-01-24	Mandala Krida	PSS Sleman	PSM Maksar

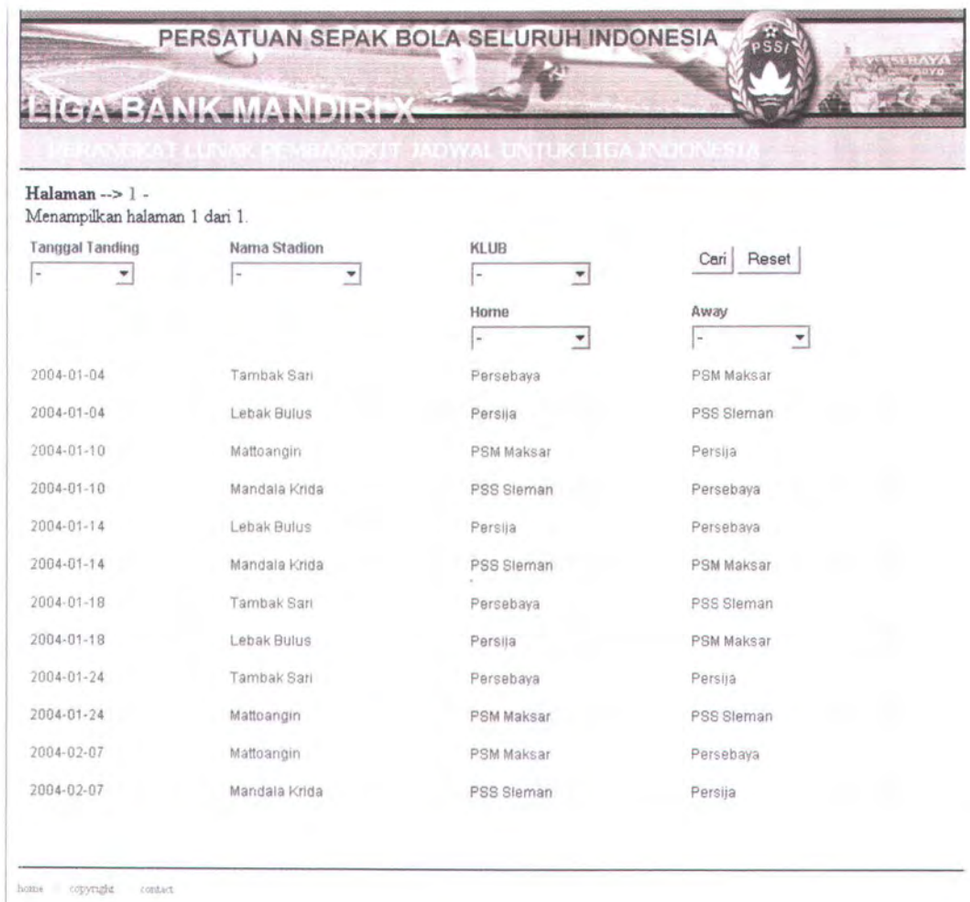
Gambar 5.2 Jadwal dengan Data Berjumlah Besar (1)

2004-01-24	Tambak Sari	Persebaya	Pupuk Kaltim
2004-01-24	Rumbai Pekanbaru	PSPS	PSDS Deli Serdang
2004-02-07	Agus Salim	PSDS Deli Serdang	PSPS
2004-02-07	Mandala Krida	PSS Sleman	Pupuk Kaltim
2004-02-07	Mattoangin	PSM Maksar	Pelita KS
2004-02-07	Tambak Sari	Persebaya	PSMS
2004-02-07	Siliwangi	Persib	Persija
2004-02-11	Teladan Medan	PSMS	Persebaya
2004-02-11	Siliwangi	Persib	PSS Sleman
2004-02-11	Krakatau Cilegon	Pelita KS	PSM Maksar
2004-02-11	Agus Salim	PSDS Deli Serdang	Pupuk Kaltim
2004-02-11	Rumbai Pekanbaru	PSPS	Persija
2004-02-15	Teladan Medan	PSMS	Pupuk Kaltim
2004-02-15	Mandala Krida	PSS Sleman	Pelita KS
2004-02-15	Siliwangi	Persib	PSDS Deli Serdang
2004-02-15	Mattoangin	PSM Maksar	Persija
2004-02-15	Tambak Sari	Persebaya	PSPS
2004-02-19	Agus Salim	PSDS Deli Serdang	Pelita KS
2004-02-19	Mandala Krida	PSS Sleman	PSMS
2004-02-19	Mattoangin	PSM Maksar	Persebaya
2004-02-19	Lebak Bulus	Persija	Pupuk Kaltim
2004-02-19	Rumbai Pekanbaru	PSPS	Persib
2004-02-29	Teladan Medan	PSMS	Persib
2004-02-29	Mandala Krida	PSS Sleman	Persija
2004-02-29	Krakatau Cilegon	Pelita KS	Persebaya
2004-02-29	Mulawarman	Pupuk Kaltim	PSDS Deli Serdang
2004-02-29	Rumbai Pekanbaru	PSPS	PSM Maksar
2004-03-06	Agus Salim	PSDS Deli Serdang	PSS Sleman
2004-03-06	Mulawarman	Pupuk Kaltim	PSMS
2004-03-06	Tambak Sari	Persebaya	Pelita KS
2004-03-06	Mattoangin	PSM Maksar	Persib
2004-03-06	Lebak Bulus	Persija	PSPS
2004-03-10	Rumbai Pekanbaru	PSPS	PSMS
2004-03-10	Krakatau Cilegon	Pelita KS	PSS Sleman
2004-03-10	Tambak Sari	Persebaya	Persib
2004-03-10	Mulawarman	Pupuk Kaltim	PSM Maksar
2004-03-10	Lebak Bulus	Persija	PSDS Deli Serdang
2004-03-14	Agus Salim	PSDS Deli Serdang	PSMS
2004-03-14	Siliwangi	Persib	Pupuk Kaltim

Gambar 5.3 Jadwal dengan Data Berjumlah Besar (2)

2004-03-14	Krakatau Cilegon	Pelita KS	Persija
2004-03-14	Tambak Sari	Persebaya	PSM Maksar
2004-03-14	Rumbai Pekanbaru	PSPS	PSS Sleman
2004-03-18	Teladan Medan	PSMS	Pelita KS
2004-03-18	Mattoangin	PSM Maksar	PSDS Deli Serdang
2004-03-18	Lebak Bulus	Persija	PSS Sleman
2004-03-18	Mulawarman	Pupuk Kaltim	PSPS
2004-03-18	Siliwangi	Persib	Persebaya
2004-03-27	Agus Salim	PSDS Deli Serdang	Persebaya
2004-03-27	Lebak Bulus	Persija	PSM Maksar
2004-03-27	Mulawarman	Pupuk Kaltim	PSS Sleman
2004-03-27	Siliwangi	Persib	PSMS
2004-03-27	Krakatau Cilegon	Pelita KS	PSPS
2004-04-11	Teladan Medan	PSMS	PSM Maksar
2004-04-11	Agus Salim	PSDS Deli Serdang	Persija
2004-04-11	Mandala Krida	PSS Sleman	Persib
2004-04-11	Mulawarman	Pupuk Kaltim	Pelita KS
2004-04-11	Rumbai Pekanbaru	PSPS	Persebaya
2004-04-18	Teladan Medan	PSMS	PSDS Deli Serdang
2004-04-18	Rumbai Pekanbaru	PSPS	Pelita KS
2004-04-18	Lebak Bulus	Persija	Persib
2004-04-18	Mandala Krida	PSS Sleman	Persebaya
2004-04-18	Mattoangin	PSM Maksar	Pupuk Kaltim
2004-06-26	Teladan Medan	PSMS	PSPS
2004-06-26	Tambak Sari	Persebaya	PSDS Deli Serdang
2004-06-26	Mattoangin	PSM Maksar	PSS Sleman
2004-06-26	Lebak Bulus	Persija	Pelita KS
2004-06-26	Mulawarman	Pupuk Kaltim	Persib

Gambar 5.4 Jadwal dengan Data Berjumlah Besar (3)



Gambar 5.5 Jadwal dengan Data Berjumlah Kecil

5.2. Evaluasi

Berdasarkan kedua uji coba diatas, penjadwalan dapat dilakukan dengan menggunakan metode algoritma genetika. Hanya saja masih ditemukan pelanggaran – pelanggaran terhadap batasan, ini dapat dilihat dari nilai fitness yang diperoleh.

Jumlah generasi, populasi dan parameter genetika tidak selalu berpengaruh terhadap kromosom yang dihasilkan baik dari nilai fitness dan kebenaran yang

diperoleh, tapi tentu saja mempengaruhi kecepatan eksekusi, hal ini dapat dilihat dari tabel 5.1 dan 5.2.

Pada uji coba kedua memperlihatkan bahwa dengan jumlah data peserta yang semakin besar, akan mempengaruhi nilai fitness, kecepatan dan kebenaran jadwal yang dibuat. Proses pengacakan juga memegang peranan dalam hal kecepatan, sesekali dengan adanya proses pengacakan akan mengakibatkan lamanya proses eksekusi.

Letak ditemukannya kromosom terbaik dari seluruh generasi pada suatu generasi tidak ditentukan oleh besar kecilnya jumlah generasi, jumlah populasi dan parameter genetika, hal ini dikarenakan adanya proses pengacakan.

BAB VI
KESIMPULAN DAN SARAN

6.1. Kesimpulan

Berdasarkan pada perancangan dan pembuatan sistem terhadap permasalahan yang diangkat, maka dapat diambil kesimpulan sebagai berikut.

1. Pemodelan terhadap penjadwalan Liga Indonesia dapat dilakukan dengan suatu metode, dalam hal ini algoritma genetika.
2. Pembuatan perangkat lunak pembangkit jadwal bisa dilakukan dengan menggunakan metode algoritma genetika.
3. Masih ditemukannya kromosom tanpa pelanggaran pada batasan, menyebabkan masih adanya peranan user dalam menentukan jadwal yang ada.

6.2. Saran

Saran – saran yang diberikan dalam pengembangan pembuatan program ini :

1. Penggunaan metode yang lain dan masalah penulisan program yang berbeda atau dengan teknik yang berbeda, sehingga dapat mempercepat proses pembangkitan jadwal pada data – data besar dan kebenaran yang diperoleh.
2. Dalam pelaksanaan kedepan Liga Indonesia akan menggunakan format dua wilayah lagi, oleh sebab itu diharapkan dapat dikembangkan perangkat lunak pembangkit jadwal untuk penjadwalan kedepan.

DAFTAR PUSTAKA

1. [Gol89] Goldberg, David E. (1989). *Genetic Algorithm, in Search, Optimization, and Machine Learning*. Addison Wesley Publishing Co., Reading, MA.
2. [Mic95] Michalewicz, Z. (1995). *Genetic Algorithms + Data Structure = Evolution Programs* (3rd ed). University of North Carolina : Springer.
3. [Wal92] Walbridge, Charles T. (1992). Genetic Algorithm, What Computers Can Learn From Darwin, CD-ROM.
4. [Gen97] Gen, Mitsui. Runwei Cheng: *Genetic Algorithms & Enginering Design*, A wiley-Interscience Publication, John Wiley & Sons, Inc. Canada 1997.
5. [Kel01] Kelley, B: *Genetic Programming and a Genetic Algorithm for Approximate Optimal Timetable Design*, CSc540 - Theory of Computing June 11, 2001.
6. [Mur] Murata, Tadahiko. Mitsuo Gen. *Performance Evaluation of Solution-Based GA and Rule-Based GA for Scheduling Problems.* Department of Industrial and Information Systems Engineering, Ashikaga Institute of Technology
7. [San] Gyori, Sandor. Zoltan Petres, Annamaria R. Varkonyi-Koczy. *Genetic Algorithm in Timetabling. A New Approach.* Budapest University of Technology and Economics Department of Measurement and Information Systems M"uegyetem rkp. 9., Budapest, Hungary, H-1521.