APLIKASI PENJADWALAN MATAKULIAH DENGAN MENGGUNAKAN ALGORITMA GENETIKA DAN METODE CINSTRAINT SATISFACTION PADA ACTIVE DATABASE, STUDI KASUS DI JURUSAN TEKNIK INFORMATIKA ITS

TUGAS AKHIR

Disusun Oleh :

GIRI WIRIAPRADJA
NRP. 5103 109 608

JURUSAN TEKNIK INFORMATIKA
FAKULTAS TEKNOLOGI INFORMASI
INSTITUT TEKNOLOGI SEPULUH NOPEMBER
SURABAYA
2006

APLIKASI PENJADWALAN MATAKULIAH DENGAN MENGGUNAKAN ALGORITMA GENETIKA DAN METODE CINSTRAINT SATISFACTION PADA ACTIVE DATABASE, STUDI KASUS DI JURUSAN TEKNIK INFORMATIKA ITS

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 1	Dosen Pembimbing 2
Dr. Ir. Joko Lianto B, M.Sc	Darlis Heru Murti, S.Kom
NIP. 131 996 151	NIP. 132 306 430

SURABAYA
JANUARI, 2006

ABSTRAK

Pembuatan jadwal pada Jurusan Teknik Informatika ITS harus dilakukan pada setiap pergantian semester. Padahal pembuatan jadwal ini membutuhkan waktu, tenaga dan ketelitian untuk membuatnya. Karena pembuatan sebuah jadwal diharuskan memperhatikan aturan-aturan dalam penjadwalan dan waktu tunggu agar mahasiswa tidak terlalu lama menunggu suatu kuliah ke kuliah selanjutnya. Karena itu diperlukan penjadwalan otomatis yang dapat membuat jadwal dengan cepat dan mudah, sehingga masalah pembuatan jadwal matakuliah dapat diselesaikan dengan lebih efisien.

Pada Tugas Akhir ini berdasarkan studi kasus permasalahan yang terjadi didalam Jurusan Teknik Informatika ITS telah diimplementasikan sebuah perangkat lunak dengan memanfaatkan constraint satisfaction untuk menyelesaikan masalah pembuatan jadwal dan algoritma genetika dalam menyelesaikan optimasi jadwal dalam hal waktu tunggu mahasiswa.

Dari uji coba yang dilakukan, aplikasi yang dibuat mampu melakukan pembuatan jadwal yang dapat memenuhi aturan-aturan penjadwalan. Dan selain itu juga mampu melakukan optimasi terhadap waktu tunggu mahasiswa.

***Kata Kunci**: algoritma genetika, constraint satisfaction problem, optimasi.*

KATA PENGANTAR

Alhamdulillahirabbil'aalamiin, segala puji syukur hanya kehadirat Allah SWT atas segala rahmat dan hidayahNya, sehingga penyusunan Laporan Tugas Akhir ini dapat terselesaikan.

Tujuan pelaksanaan dan penulisan Tugas Akhir yang berjudul " **Aplikasi Penjadwalan Matakuliah Dengan Menggunakan Algoritma Genetika Dan Metode Constraint Satisfaction Pada Active Database, Studi Kasus Di Jurusan Teknik Informatika"** ini adalah untuk memenuhi salah satu syarat memperoleh gelar sarjana S-1 Teknik Di Jurusan Teknik Informatika, Fakultas Teknologi Informasi (FTIf), Institut Teknologi Sepuluh Nopember (ITS) Surabaya.

Dengan keterbatasan waktu dan kemampuan penulis, mohon maaf jika ada kekurangan dalam Tugas Akhir ini. Untuk itu kritik dan saran yang membangun sangat penulis harapkan. Akhir kata semoga Tugas Akhir ini dapat bermanfaat bagi pembaca.

Surabaya, Januari 2006

Penulis

UCAPAN TERIMA KASIH

Dalam penyusunan Tugas Akhir ini tidak lepas dari bantuan, bimbingan, pengarahan maupun koreksi dan dorongan moril yang diberikan dari berbagai pihak. Untuk itu perkenankanlah penulis mengucapkan terima kasih yang sebesar-besarnya kepada :

1. Dr. Ir Joko Lianto B, M.Sc dan Darlis Heru Murti, S.Kom, sebagai dosen pembimbing yang telah banyak memberikan bantuan, bimbingan serta dorongan selama penyelesaian tugas akhir ini.
2. Yudhi Purwananto, S.Kom, M.Kom., selaku Ketua Jurusan Teknik Informatika ITS.
3. Bapak dan Ibu dosen Jurusan Teknik Informatika – ITS yang telah memberikan ilmunya selama Penulis menempuh kuliah.
4. Seluruh staf dan karyawan Jurusan Teknik Informatika – ITS, Pak Yudi, Pak Narno, Mbak Eva, dan karyawan-karyawan lain atas bantuannya dalam mengurus banyak hal.
5. Ayah dan Ibu yang dengan seringnya memberikan Doa, dukungan, dan kesabarannya. Semoga Papa dan Mama selalu mendapatkan kemuliaan, kesabaran, kebahagiaan didunia dan akhirat, serta selalu dalam lindungan Allah SWT.
6. Kakak yang telah memberi banyak masukan dan kebijaksanaan dan Adik yang selalu membantu menyemangati.
7. Sahabatku Adi Nurrudin, Jeffryanto Pasaribu, Ria Artikawati dan Peny Gamma yang selalu memarahi di kala malas, memotivasi dikala

kehilangan semangat, memberikan petuah dikala bingung. Terima kasih untuk semua *sharing* dan *caring* yang telah diberikan.

8. Teman satu kost Ciptian junior dan senior, Mas Supri, Mas Priyono, Harista junior dan senior, dan Sukron yang sering bermain game dan jalan-jalan bersama.
9. Teman-Temanku Moda, Doma, Icuk, Indra, Adi Karunia, Pandu, Bolon, Dewi, Dita, Anton, Erwin, Erma, Sari, Shinta, Yuanita, Haris, dan Rizki yang sama-sama berjuang dalam pengerjaan TA.
10. Ria, Keke, Kunti, Anis dan Dyah yang sama-sama bingung dalam pengerjaan TA. Trims untuk berbagi pengetahuan yang kalian miliki.
11. Teman-teman yang belum disebutkan, atas semua kebersamaan dan dukungannya selama ini.
12. Semua pihak yang tidak dapat penulis sebutkan satu persatu.

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

BAB I
PENDAHULUAN

Pada bab pendahuluan ini dijelaskan hal-hal yang melatarbelakangi pembuatan tugas akhir ini, tujuan, permasalahan yang dihadapi, batasan masalah dan metodologi pembuatannya.

1.1 Latar Belakang

Pembuatan jadwal pada Jurusan Teknik Informatika ITS harus dilakukan pada setiap pergantian semester. Padahal pembuatan jadwal ini membutuhkan waktu, tenaga dan ketelitian untuk membuatnya. Karena itu diperlukan penjadwalan otomatis yang dapat membuat jadwal dengan cepat dan mudah, sehingga masalah pembuatan jadwal matakuliah dapat diselesaikan dengan lebih efisien.

Dan dalam pembuatan jadwal tersebut harus memperhatikan aturan-aturan penjadwalan yang sudah ditentukan. Yang memastikan mahasiswa dapat mengambil matakuliah sesuai dengan kurikulum yang ditentukan jurusan, sehingga jika mahasiswa mengambil matakuliah sesuai kurikulum maka tidak ada matakuliah yang tidak bisa diambil.

Untuk dapat mengatasi permasalahan pembuatan jadwal matakuliah tersebut maka diharapkan tugas akhir yang mengambil studi kasus di Jurusan Teknik Informatika ITS ini dapat membantu untuk menyelesaikan masalah dengan lebih efisien.

1.2 Perumusan Masalah

Permasalahan yang diangkat dalam menyelesaikan tugas akhir ini adalah :

- Bagaimana menangani pembuatan jadwal kuliah, berdasarkan data yang tersedia.
- Bagaimana caranya mengatur jadwal matakuliah sehingga mahasiswa dapat mengikuti matakuliah sesuai kurikulum.
- Bagaimana menangani jadwal–jadwal matakuliah yang menginginkan waktu tertentu.
- Bagaimana menerapkan permasalahan ini ke dalam bentuk yang bisa diselesaikan dengan metode *Constraint Satisfaction Problem*.
- Bagaimana menerapkan algoritma genetika untuk menyusun data input agar mendapatkan hasil yang lebih optimal pada waktu tunggu mahasiswa.

1.3 Batasan Masalah

Dari permasalahan diatas, maka batasan dalam tugas akhir ini adalah:

- Pembuatan jadwal dilakukan dengan mengimplementasikan metode *Constraint Satisfaction Problem* dengan *backtracking search* menggunakan *Forward Checking*.
- Studi kasus yang dipakai dalam aplikasi ini adalah Akademik, sehingga data yang dipakai adalah data matakuliah yang ada di Jurusan Teknik Informatika ITS.

- Ruang lingkup *Constraint Satisfaction Problem* dalam Tugas Akhir ini dibatasi dengan variabel dan *constraint* yang sudah diketahui, dengan *domain* yang *finite* dan *discrete* untuk setiap variabel.
- Hasil dari proses dapat berhasil ataupun tidak.
- Metode ini hanya dipakai untuk menangani pembuatan jadwal dari aturan penjadwalan.
- Jumlah sks tidak mempengaruhi Jadwal.
- Penghitungan waktu tunggu tidak memasukan kelas pilihan.
- Pada suatu matakuliah yang memiliki banyak kelas pada suatu hari, Waktu tunggu yang dibuat mengambil kelas yang paling akhir yang diajarkan pada hari itu sebagai data waktu tunggu yang dihitung.
- Kelas pilihan dianggap tidak memiliki semester yang pasti, sehingga boleh bentrok dengan semua semester kecuali dengan semester 6, 7, 8.

1.4 Tujuan

Tujuan dari pembuatan tugas akhir ini untuk membuat sebuah aplikasi penjadwalan yang menerapkan Algoritma Genetika dan *Constraint Satisfaction* untuk melakukan otomatisasi pembuatan jadwal dan optimasi waktu tunggu mahasiswa pada proses penjadwalan kuliah, dengan memperhatikan aturan-aturan penjadwalan.

1.5 Sistematika Pembahasan Tugas Akhir

Pembuatan buku tugas akhir ini dilakukan dengan pembagian bab sebagai berikut:

1. Pendahuluan.

   Bab ini menjelaskan tentang latar belakang yang mendasari dikerjakannya tugas akhir ini, bab ini terdiri dari latar belakang, perumusan masalah, batasan masalah, tujuan dan sistematika pembahasan tugas akhir.

2. Dasar Teori.

   Pada bab ini menjelaskan secara singkat tentang teori-teori yang digunakan selama proses pengerjaan tugas akhir ini, Literatur yang digunakan meliputi buku referensi, dokumentasi internet.

3. Perancangan dan Implementasi Perangkat Lunak.

   Pada tahap ini diuraikan mengenai penerapan teori dan algoritma yang dikembangkan dalam bentuk perangkat lunak. Berikut juga dijelaskan mengenai lingkungan pembangunan Sistem penjadwalan, meliputi hardware, software, dan modul-modul yang digunakan.

4. Uji Coba dan Evaluasi.

   Pada tahap ini, asumsi implementasi sudah selesai, dilakukan uji coba kebenaran dengan beberapa data yang telah disiapkan. Kemudian, hasil pengujian ini akan dievaluasi untuk menentukan validitas sistem

yang dibuat sebagai bahan pertimbangan perlu tidaknya melakukan perbaikan pada program.

5. Penutup.

Pada tahap ini terdiri dari dua bagian, yaitu kesimpulan dan saran. Bagian kesimpulan berisikan kesimpulan yang diambil oleh penulis dan bagian saran berisikan saran-saran yang diberikan oleh penulis untuk pengembangan tugas akhir ini selanjutnya.

BAB II

DASAR TEORI

Dalam bab II ini dibahas beberapa teori dasar untuk menunjang penyelesaian tugas akhir ini, antara lain mengenai *Combinatorial search problem*, *Np-Hard*, *Tree*, *Depth first search*, *Heuristic*, Algoritma Genetika, *Constraint Satisfaction Problem*, *dan Active Database*.

2.1 Combinatorial search problems

Combinatorial search problems, adalah permasalahan untuk menanyakan secara eksplisit dan implisit untuk mencari objek kombinasi seperti : permutasi, *subset*, atau membuat kombinasi yang memenuhi *constraint* tertentu dan memiliki properti tertentu yang diinginkan [LEV03]. Secara umum masalah seperti ini merupakan masalah yang paling sulit dalam perhitungan baik dari segi teori maupun praktek [WIKI][LEV03]. Kesulitan tersebut dikarenakan fakta berikut. Pertama, jumlah kombinasi objek akan bertambah dengan cepat sesuai skala masalahnya, biasanya mencapai ukuran yang sangat besar bahkan untuk masalah yang tingkat kesulitannya biasa. Kedua, tidak adanya algoritma yang dikenal dapat menyelesaikan seluruh masalah combinatorial dalam waktu cepat. Ketiga, kebanyakan ilmuwan komputer percaya algoritma seperti itu tidak akan pernah ada [LEV03].

Beberapa *combinatorial problem* bisa diselesaikan dengan efisien oleh algoritma tertentu, tetapi biasanya hal tersebut dianggap keberuntungan dengan adanya pengecualian terhadap rule. [LEV03].

2.2 NP-Hard Problem

Dalam studi masalah kerumitan komputasi (*computational complexity*), yang pertama diperhatikan oleh peneliti komputer dan ahli komputasi adalah apakah sebuah masalah bisa diselesaikan dalam waktu *polynomial* (*Polynomial time*) oleh suatu algoritma. Definisi algoritma dapat menyelesaikan masalah dalam waktu polynomial adalah jika efisiensi waktu *worst-case* berupa O(p(n)) dimana p(n) adalah *polynomial* dari ukuran input masalah n. permasalahan yang bisa diselesaikan dalam waktu *polynomial* disebut *tractable*, masalah yang tidak dapat diselesaikan dalam waktu *polynomial* adalah *intractable* [LEV03].

Suatu permasalahan termasuk dalam P (*polynomial*) jika dapat diselesaikan dalam waktu *polynomial* oleh suatu algoritma (*deterministic*). Definisi yang formal untuk permasalahan yang termasuk dalam P, adalah hanya *decision problems*, yang berbentuk permasalahan dengan jawaban ya atau tidak.

Algoritma *nondeterministic* adalah prosedur dengan dua tahap yang mengambil input sebuah *instances* I dari *decision problem* dan melakukan :

Tahap *Nondeterministic* ("*guessing*"): *string* S sembarang dibuat untuk menjadi sebuah kandidat solusi untuk *instances* I yang diberikan (tetapi mungkin juga merupakan sebuah sampah).

Tahap *Deterministic* ("*Verification*"): sebuah algoritma *deterministic* mengambil I dan S sebagai input, dan output yes jika S mewakili solusi dari I. (jika S tidak mewakili solusi dari I, algoritma dapat membarikan jawaban no atau diijinkan untuk melanjutkan). Suatu algoritma *nondeterministic* disebut *nondeterministic polynomial* jika efisiensi waktu dari *stage* verifikasinya *polynomial*.

NP (*nondeterministic polynomial*) Problems adalah *decision problems* yang bisa diselesaikan dengan *nondeterministic polynomial algorithm*. Kemudian ada *decision problems* yang dikenal sebagai *Np-complete*, yaitu masalah lain di NP yang bisa dikurangi waktu *polynomial-nya*.

Suatu *decision problems* D1 dibilang *polynomial reducible* untuk suatu *decision problems* D2 jika ada fungsi t yang merubah *instances* dari D1 ke *instances* D2 seperti :

1. t memetakan *instances* yes dari D1 ke *instances* D2 dan semua no *instances* dari D1 ke *instances* D2.
2. t dapat dihitung dengan algoritma *polynomial-time*.

Jadi permasalahan NP-complete adalah

1. termasuk dalam kelas NP.
2. setiap masalah dalam NP adalah *polynomial reducible* terhadap D.

NP-Hard Problem adalah suatu masalah yang serumit NP-Complete, tetapi tidak ada algoritma dengan waktu *polynomial* yang dikenal dapat menyelesaikan

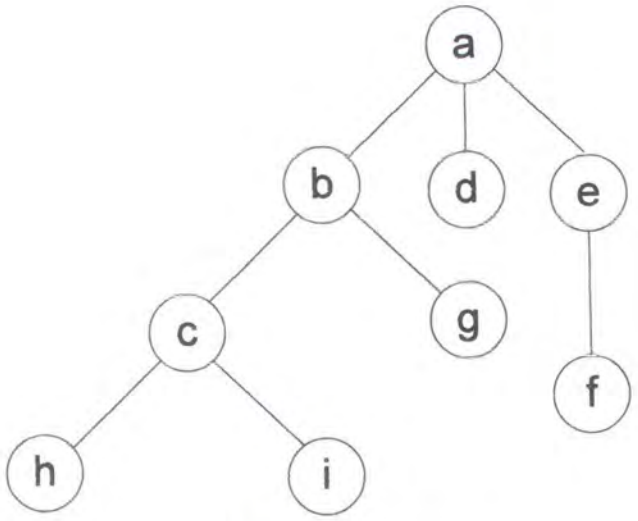
masalah ini, dan banyak anggapan bahwa algoritma ini tidak akan pernah ada[LEV03].

Jika ukuran masalah ini kecil biasanya diselesaikan dengan *exhaustive search* algorithm (Algoritma *Brute-force* yang dipakai untuk menyelesaikan *combinatorial problem*).

Pendekatan lain yang dipakai untuk menyelesaikan ini adalah dengan menyelesaikannya dengan algoritma yang cepat. Pendekatan ini biasanya dipakai untuk mencari suatu solusi yang cukup baik, tetapi tidak harus yang paling baik[LEV03].

2.3 Tree

Tree adalah sebuah kumpulan item yang terstruktur secara hirarki[AHO87]. *Tree* atau *rooted tree* merupakan kumpulan dari suatu item yang disebut "*vertices*" atau "*nodes*", yang dihubungkan satu sama lainnya dengan garis yang disebut "*edges*" atau "*arcs*". *Tree* memiliki sebuah *root* yang merupakan salah satu *vertices* dalam *tree* yang ditempatkan pada posisi paling atas (level 0 dari tree), *root* tersebut dapat memiliki *adjacent* (hubungan) berupa *edges* ke vertices dibawahnya (level 1), dan *root* berjarak dua *edges* dari *vertices* dibawahnya lagi (level 2), dan seterusnya. *Vertices* v yang memiliki hubungan ke *vertices* u diatasnya disebut *child* dari u dan kebalikannya disebut *parent*, *vertices* yang tidak memiliki *child* disebut *leaf* (daun). *Vertices* juga memiliki apa yang disebut *siblings* (saudara) yaitu *vertices* yang memiliki *parent* yang sama[LEV03]. Bentuk *tree* dapat digambarkan dalam Gambar 2.1.



Gambar 2.1 : *Tree*

Contoh :

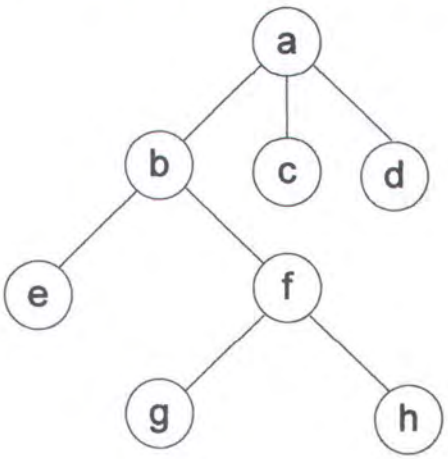
- *Vertices* adalah a, b, c, d, e, f, g, h, dan i
- *Leaf* adalah d, g, h, i, f
- *Parent* dari vertices b adalah a
- *Child* dari a adalah b, d, e
- *Siblings* dari b adalah d dan e
- *Root tree* adalah a

Depth sebuah *vertices* v adalah, jarak terpendek dari *root* ke *vertices* v. Sedangkan *height* dari tree adalah jarak terjauh dari sebuah *root* ke *leaf*[LEV03]. Dari gambar 2.1 dapat dicontohkan memiliki *depth vertices* c adalah 2, dan *height* dari *tree* adalah 3.

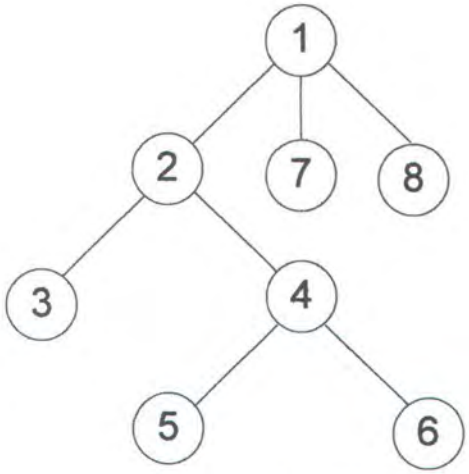
2.4 Depth first search

Ide dasar *depth first search* adalah untuk menembus sedalam mungkin pada *tree*, sebelum mencari ke *vertex* yang lain[LEV03][AHO87]. Untuk melakukan *depth first search* pada bentuk *tree* digambarkan pada gambar 2.2 (a). Mulai dari *vertex* A, dimulai dengan memilih beberapa *vertex* yang *adjacent*

dengan a (misal b) Kemudian memilih *vertex* yang *adjacent* dengan b, dan seterusnya. Pada masing-masing titik dipilih *vertex* yang belum digunakan sebelumnya. *Vertex-vertex* yang telah dipilih tersebut diberi nomor secara berurutan. Jika sudah sampai suatu nomor dan tidak ada lagi nomor berikutnya, maka kembali ke angka sebelumnya dan memilih nomor berikutnya. Dan dipilih lagi *vertex* yang *adjacent* dengan *vertex* pada nomor tersebut. Urutan langkah-langkah tersebut dapat dilihat pada gambar 2.2 (b).



(a)



(b)

Gambar 2.2 : Depth First Search. (a) menggambarkan tree yang ingin ditelusuri dengan DFS, (b) menggambarkan urutan langkah yang dijalani oleh Depth First Search.

2.5 Heuristic

Algoritma *heuristic* merupakan kumpulan aturan yang menghasilkan pemecahan masalah melalui pengalaman, tetapi tidak harus merupakan hasil yang terbaik [WIKI][AHO87]. Aturan-aturan ini mungkin kaku atau fleksibel untuk suatu masalah. *Heuristic* yang baik akan dapat menghasilkan pemecahan yang optimal dengan banyak percobaan.

Suatu masalah kombinatorial dapat mempunyai cukup banyak algoritma *heuristic* yang berbeda. Setiap algoritma digunakan untuk masalah yang khusus dan disusun sehingga memberikan keuntungan untuk penyelesaian ke masalah itu sendiri. Secara khusus, prinsip operasi hampir semua *heuristic* kombinatorial adalah menemukan salah satu optimasi. Kebanyakan algoritma *heuristic* berdasar pada prinsip optimasi lokal, dan dibatasi dengan suatu aturan tertentu sehingga hasil akhirnya akan dapat mendekati hasil optimal.

2.6 Algoritma Genetika

Algoritma Genetika (GA) ini pertama kali diciptakan oleh John Holland dari Unversitas Michigan (1975). Awalnya Holland tidak bertujuan membuat algoritma untuk menyelesaikan masalah tertentu, tetapi untuk melihat fenomena adaptasi yang terjadi di alam dan untuk membuat cara dimana mekanisme dari adaptasi alamiah bisa digambarkan dalam sistem komputer[MEL99][KUS05] [KAR98].

John Holland mengatakan bahwa setiap masalah yang berbentuk adaptasi (alami maupun buatan) dapat diformulasikan dalam terminologi genetika. Algoritma genetika adalah simulasi dari proses evolusi Darwin dan operasi

genetika atas kromosom. Dalam buku "*Adaptation in Natural and Artificial Systems*" karangan Holland di tahun 1975, mempersembahkan algoritma genetika sebagai abstraksi dari evolusi biologis dan memberikan gambaran teori pada adaptasi di algoritma genetika. GA buatan Holland adalah metode untuk memindahkan satu populasi "kromosom atau Individu" ke populasi lain dengan menggunakan "seleksi alami" bersama dengan operator genetika lainnya seperti *crossover*, dan mutasi. Setiap kromosom mengandung sekumpulan "gen". Operator seleksi memilih kromosom didalam populasi yang akan diijinkan untuk melakukan reproduksi, dan kromosom yang memiliki *fitness* yang lebih baik akan dimungkinkan menghasilkan *offspring* yang lebih banyak daripada yang kurang fit. Pada kurun waktu tertentu (sering dikenal dengan istilah generasi), populasi secara keseluruhan akan lebih banyak memuat generasi yang *fit*. *Crossover* menukar bagian dari dua kromosom, meniru rekombinasi biologis diantara dua organisme "*single-chromosome*" ("haploid"). Mutasi secara random mengubah nilai gen pada beberapa lokasi didalam kromosom.

Algoritma Genetika adalah algoritma pencarian heuristik yang didasarkan atas mekanisme evolusi biologis. Keberagaman yang terjadi pada evolusi biologis dikarenakan adanya variasi pada setiap organisme kromosom. Variasi kromosom ini akan mempengaruhi reproduksi dan tingkat kemampuan organisme untuk tetap hidup. Sebuah generasi dalam suatu populasi mengikuti prinsip seleksi alam yaitu individu yang kuat akan selamat dan yang lemah akan mati. Pada dasarnya ada 4 kondisi yang mempengaruhi proses evolusi, yaitu sebagai berikut :

1. Kemampuan organisme untuk melakukan reproduksi.

2. Keberadaan populasi organisme yang bisa melakukan reproduksi.
3. Keberagaman organisme dalam suatu populasi.
4. Perbedaan kemampuan untuk survive.

Secara garis besar, metode yang dapat disebut algoritma genetika adalah metode yang setidaknya memiliki komponen seperti : populasi dari kromosom, seleksi berdasarkan *fitness*, *crossover* untuk memproduksi *offspring* baru, dan random *mutasi* pada *offspring* [MEL99].

2.6.1 Struktur umum Algoritma Genetika

Algoritma genetika ini berbeda dengan metode pencarian konvensional, algoritma genetika memulai dari himpunan kandidat solusi yang dihasilkan secara acak. Himpunan solusi ini dikenal dengan istilah populasi. Setiap Individu yang terdapat dalam satu populasi disebut dengan istilah kromosom, yang merupakan representasi dari sebuah solusi. Sebuah kromosom dinyatakan dengan rangkaian dari simbol (*string*) atau dikenal dengan istilah gen. Pada algoritma genetika, Populasi awal dibentuk secara acak sedangkan populasi berikutnya merupakan hasil evolusi Kromosom-kromosom dalam suatu iterasi yang berkelanjutan, yang disebut dengan istilah generasi. Kromosom-kromosom yang terbentuk selanjutnya diperoleh dari operasi yang dilakukan pada kromosom induk (*parent*) yang ada didalam populasi, dikenal dengan istilah anak (*offspring*). Kromosom *offspring* dapat terbentuk dengan melakukan penyilangan (*crossover*) dari dua parent yang menghasilkan kombinasi dari dua kromosom. Selain *crossover*, kromosom baru

juga dapat terbentuk dengan melakukan modifikasi suatu kromosom yang disebut dengan istilah mutasi.

Pada setiap generasi, kromosom akan melalui proses seleksi dengan menggunakan alat ukur yang disebut dengan fungsi *fitness*. Nilai *fitness* dari suatu kromosom menunjukan kualitas kromosom dalam populasi tersebut. Generasi baru akan dibentuk dengan menyeleksi nilai *fitness* dari kromosom *parent* dan kromosom *offspring*, serta menolak kromosom-kromosom yang tidak cukup fit sehingga ukuran populasi (jumlah kromosom dalam suatu populasi) konstan.

Jika pendekatan meniru seleksi alam disederhanakan maka secara umum akan menjadi langkah :

1. Menyusun populasi awal kromosom (kandidat solusi).
2. Menghitung *fitness* f(x) dari setiap kromosom x didalam populasi.
3. Ulangi langkah dibawah sampai n *offspring* telah dibuat.
    a. Memilih pasangan kromosom *parent* dari populasi yang sekarang, kemungkinan terpilih meningkat seiring dengan meningkatnya nilai *fitness*.
    b. Dengan kemungkinan cp ("*crossover probability*" atau "*crossover rate*"), *crossover* pasangan pada titik yang dipilih secara random (dengan probabilitas *uniform*) untuk membentuk dua *offspring*.
    c. Mutasikan dua offspring tersebut pada dengan probabilitas mp (kemungkinan probabilitas atau *mutation rate*), dan tempatkan kromosom hasil di populasi baru. Jika n berjumlah ganjil, satu anggota populasi baru bisa dihapuskan secara random.

4. Ganti populasi yang sekarang dengan populasi baru.
5. Kembali ke langkah 2.

Setiap iterasi dari proses ini disebut generasi. Dan seluruh set dari generasi disebut *run*. Pada akhir *run* seringkali terdapat ada satu atau lebih kromosom yang memiliki *fitness* yang tinggi di populasi.

2.6.2 Operator – operator Algoritma Genetika

2.6.2.1 Seleksi

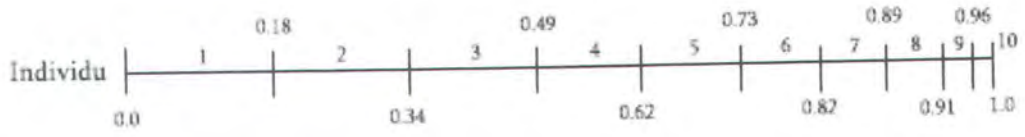
Seleksi bertujuan untuk memberikan kesempatan reproduksi yang lebih besar bagi anggota populasi yang paling *fit* [MEL99][KUS05]. Seleksi akan menentukan individu–individu mana saja yang akan dipilih untuk mendapatkan generasi baru. Langkah pertama yang dilakukan dalam seleksi ini adalah pencarian nilai *fitness*. Masing–masing individu dalam suatu wadah seleksi akan menerima probabilitas reproduksi yang tergantung pada nilai objektif dari semua individu dalam wadah seleksi tersebut. Nilai *fitness* inilah yang dipakai pada tahap-tahap seleksi berikutnya Pada hal ini perlu diperhatikan agar populasi tidak menjadi terlalu besar, karena akan memakan banyak waktu untuk suatu generasi, dan agar populasi juga tidak menjadi terlalu kecil dan mengakibatkan kromosom terlalu mirip sehingga operasi kromosom tidak akan banyak berpengaruh.

2.6.2.1.1 Seleksi roda roulette(Roulette Wheel Selection)

Metode seleksi roda *roulette* ini merupakan metode yang sederhana, dan sering disebut juga dengan *stochastic sampling with replacement*. Pada metode ini, individu-individu dipetakan dalam suatu segmen garis secara berurutan sedemikian hingga tiap-tiap segmen individu memiliki ukuran yang sama dengan ukuran *fitness*-nya. Sebuah bilangan *random* (dengan probabilitas *uniform*) dibangkitkan dan individu yang memiliki segmen dalam kawasan bilangan random tersebut akan terseleksi. Proses ini diulang hingga diperoleh sejumlah individu yang diharapkan [KUS05].

Individu ke-	1	2	3	4	5	6	7	8	9	10	11
Nilai Fitness	2,0	1,8	1,6	1,4	1,2	1,0	0,8	0,6	0,4	0,2	0,0
Probabilitas seleksi	0,18	0,16	0,15	0,13	0,11	0,09	0,07	0,05	0,04	0,02	0,00

Tabel 2.1 : Tabel *fitness*



Gambar 2.3 : Seleksi roda roulette.

Jika memiliki data seperti pada tabel 2.1, dibangkitkan enam nilai *random* (*uniform*) dan keluar nilai–nilai sebagai berikut :

0.10 0.29 0.43 0.65 0.80 0.94

Jika dilihat pada gambar 2.3 didapatkan kromosom terpilih :

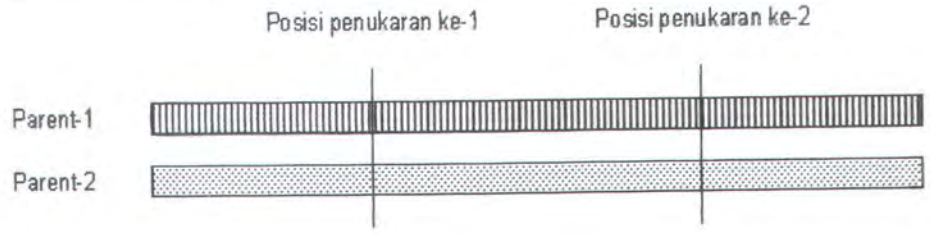
1 2 3 5 6 9

2.6.2.2 Crossover

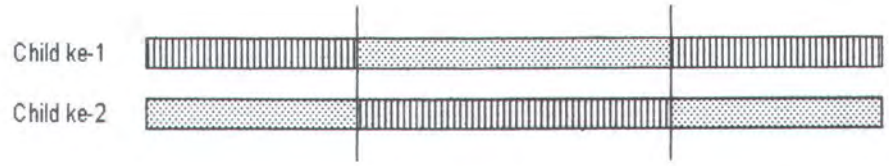
Crossover adalah proses dari pertukaran dua kromosom yang menciptakan *offspring* yang mewarisi karakter dari kedua *parent*. Dengan mewarisi karakter parent tersebut tujuan dari melakukan crossover adalah untuk mencari kombinasi gen terbaik dari *parent*. Adapun tingkat terjadinya *crossover* didasarkan pada probabilitas. *Parent* yang terpilih akan dipasangkan dengan *parent* lain yang juga terpilih dalam proses *crossover*. Dari setiap pasangan akan terbentuk dua *offspring*. Operator algoritma genetika memilih suatu titik dan menukar urutan sebelum dan sesudah titik tersebut diantara dua kromosom untuk menciptakan *offspring* baru [KUS05].

2.6.2.2.1 Penyilangan dua titik

Pada penyilangan banyak titik, posisi penyilangan diseleksi secara random dan tidak diperbolehkan ada posisi yang sama, serta diurutkan naik. Variabel-variable ditukar antar kromosom pada titik tersebut untuk menghasilkan anak.



(a)



(b)

Gambar 2.4 : *Crossover*. (a) dua parent yang akan di *crossover*. (b) : dua *child* hasil *crossover*

2.6.2.2.2 Penyilangan dengan permutasi

Pada penyilangan dengan permutasi ini, kromosom-kromosom *offspring* dipilih dan disilangkan dengan tetap menjaga urutan dan posisi sejumlah nilai yang mungkin terhadap induk yang lainnya, sisanya ditukarkan berdasarkan pemetaan yang didapatkan ketika penyilangan. Dicontohkan dibawah ini :

Misal :

Parent 1 :	1	2	3	4	5	6	7	8	9
Parent 2 :	4	5	3	1	8	7	6	9	2

Akan menghasilkan kandidat pertukaran seperti ini :

Offspring 1 :	x	x	x	1	8	7	6	x	x
Offspring 2 :	x	x	x	4	5	6	7	x	x

Dari pertukaran diatas didapatkan pemetaan :

1 – 4, 8 – 5, 7 – 6, 6 -7.

Kemudian kita mengcopykan sisa *gen* dari *parent*-1 ke *offspring*-1dengan menggunakan pemetaan :

Offspring 1 :	1-4	2	3	1	8	7	6	8-5	9	
Offspring 1 :	4	2	3	1	8	7	6	5	9	

Dan hal yang sama terjadi pada *offspring*-2 :

Offspring 2 :	4-1	5-8	3	4	5	6	7	9	2	
Offspring 2 :	1	8	3	4	5	6	7	9	2	

2.6.2.3 Mutasi

Operator Mutasi mengubah field individual (*gen*) didalam kromosom berdasarkan pada probabilitas. Mutasi biasanya dilakukan setelah semua operator genetik yang lain tetapi sebelum seleksi.

Dalam konteks algoritma genetik mendekati konvergensi pada suatu nilai, dengan adanya percampuran antara kandidat dengan kualitas tinggi, maka suatu populasi memiliki kecenderungan untuk menjadi mirip. Hal ini baik jika titik konvergensi adalah *global optimum*, tetapi jika titik konvergensi adalah *local optima* maka suatu populasi yang *homogen* tidak akan dapat menemukan solusi *optimal*. Mutasi dapat dipakai untuk menghindari kecenderungan suatu populasi untuk menjadi mirip [MEL99].

Fungsi probabilitas mutasi dipakai untuk setiap *offspring* kromosom, dimana nilai dan jumlah gen yang akan dimutasi dipilih secara acak. Nilai

probabilitas ini memiliki peranan yang cukup penting karena jika terlalu kecil suatu karakter baru akan terlalu lambat muncul di populasi, dan jika terlalu tinggi setiap generasi akan cenderung untuk selalu berbeda dengan generasi sebelumnya.

Pada gambar dibawah ini terpilih nilai pada urutan ke tiga dan ke enam untuk dilakukan mutasi :

Offspring	1	2	3	4	5	6	7	8	9

Offspring sesudah dilakukan operasi mutasi :

Offspring	1	2	6	4	5	3	7	8	9

2.6.2.4 Fitness

Salah satu pemakaian umum GA adalah fungsi optimasi, dimana tujuannya adalah untuk menemukan suatu nilai parameter untuk dimaksimalkan. *Fitness* adalah suatu nilai yang dipakai untuk tolak ukur kualitas kromosom[KUS05].

2.7 Constraint Satisfaction

Pada bagian ini akan dijelaskan bagaimana *Constraint Satisfaction Problem* (CSP) berkerja untuk menyelesaikan masalah *combinatorial search problem.*

CSP adalah suatu permasalahan dimana seseorang harus mencari nilai untuk set variabel (*finite*) yang memenuhi set *constraint* (*finite*)[MAD03][KTI] [TSA99][RUS03]. Pencarian pada CSP biasanya eksponensial karena

permasalahan berupa NP-*complete*. Dalam pengerjaannya ada banyak pendekatan yang dapat dipakai untuk mengurangi ruang pencarian dan dapat menemukan solusi dalam waktu yang cepat, tetapi dalam pengerjaan buku TA ini hanya akan dibahas mengenai *Backtracking* (BT), *Forward Checking* (FC) dan *Minimum Remaining Value* (MRV).

Secara umum CSP terdiri dari komponen :

- Variabel adalah suatu penampung yang dapat diisi berbagai nilai. Sebuah variable disebut *instantiated* ketika sudah diisikan dengan nilai dari *domain*-nya dengan suatu *assignment*, selain itu disebut *unassigned*.
- *Constraint* adalah suatu aturan yang ditentukan, yang akan mengatur nilai yang boleh diisikan ke variabel, atau kombinasi variabel. Constraint terbagi menjadi dua jenis, yaitu *hard constratint* jika suatu variabel diharuskan dipenuhi dengan suatu nilai dari domain, atau disebut *soft constraint* jika tidak harus dipenuhi dengan pemberian nilai dari domain.
- *Domain* adalah kumpulan nilai legal yang dapat diisi ke variabel, *domain* dari setiap variabel ke variabel lainnya dapat berbeda bergantung pada *constraint*.
- Solusi adalah *assignment* nilai-nilai dari *domain* ke setiap variabel dengan tidak ada *constraint* yang dilanggar. Sebuah solusi yang sudah terisi semua variabelnya dan semua *constraint* terpenuhi disebut *consistent complete assignment*, selain itu disebut *partial solution*.

Sebuah masalah dapat memiliki satu, banyak, atau tidak ada solusi. Jika sebuah masalah memiliki satu atau lebih solusi,. Jika CSP tidak memiliki solusi, masalah ini disebut *over-constrained* atau *inconsistent* dan CSP dengan lebih dari satu solusi disebut *under-constrained* atau *consistent*.

CSP dimulai dari solusi kosong (tidak ada variable yang terisi) yang berakhir dengan sebuah solusi yang memenuhi semua *constraint* pada permasalahan (*consistent*). *Backtracking* digunakan bila terjadi suatu pencarian yang gagal (ada variabel yang sudah tidak memiliki nilai legal). Algoritma dapat digunakan untuk mendapatkan solusi (jika ada), atau sebaliknya untuk membuktikan bahwa permasalahan tersebut tidak memiliki solusi.

Ada dua metode umum untuk menyelesaikan CSP :

- *Constructive methods* (*systematic search strategies*) : dimulai dengan mengisikan nilai ke variabel yang memenuhi *constraint*, dan mengembangkan *assignment* sedikit demi sedikit ke variabel lainnya selama *consistency* terjaga. Jika tidak ada nilai yang dapat diisikan ke variable maka digunakan *backtracking* untuk memilih nilai yang lain pada nilai yang terakhir diisikan.
- *Destructive methods* (*repair strategies*) : mengijinkan pengisian semua variabel pada inisialisasi. Jika ada variabel yang tidak *consistent* maka hilangkan nilai yang tidak diinginkan sehingga tercapai *consistency*. Jika hal tersebut tidak menghasilkan solusi maka gunakan *backtracking*.

CSP mengandung sebuah set variabel yang berasosiasi dengan domain yang *finite* dan sebuah set *constraint* yang membatasi nilai-nilai yang dapat dipakai oleh variabel. Dalam mencari solusi untuk CSP, sebuah nilai diisikan ke setiap variabel dari *domain* pada setiap variabel, selagi tidak melanggar *constraint*. Algoritma untuk menyelesaikan CSP biasanya menggunakan algoritma yang umum yaitu *systematic search algorithm*. Kebanyakan algoritma menyelesaikan CSP secara sistimatis dengan melihat *assignment* yang memungkinkan untuk variabel, dengan mencoba mengisikan nilai dari *domain* kepada setiap variabel satu demi satu tanpa melanggar *constraint* sampai solusi ditemukan. Jika solusi belum ditemukan dan nilai legal untuk suatu variable sudah habis maka langkah yang terakhir akan diganti dengan langkah alternatif yang belum dicoba. Jika penelusuran hanya dilakukan secara sistimatis saja, ada kemungkinan untuk mencoba semua solusi.

2.7.1 Backtracking

Algoritma yang paling umum untuk melakukan pencarian sistematik untuk menyelesaikan CSP adalah *backtracking*, Jika ada permasalahan dan urutannya penyelesaiannya tidak menjadi masalah. Permasalahannya harus mengandung sebuah set variabel dimana setiap variabel harus diberi sebuah nilai, tergantung *constraint* untuk masalahnya [RUS03][WIKI]. Algoritma *backtracking search* (penelusuran kembali) adalah suatu bentuk algoritma *depth-first-search*. *Backtracking* dapat dilihat sebagaimana *searching* dalam *tree*, dimana setiap *node* mewakili *state* dan turunan dari setiap *node* mewakili ekstensi dari *state* tersebut.

Backtracking secara meningkat berusaha mengembangkan solusi *partial* yang *consistent* untuk variabel, menuju ke solusi. Di dalam metode *backtracking*, variabel diisikan secara *sequential* selagi semua variabel relevan dengan *constraint* yang sudah diinisialisasi. Jika solusi partial melanggar *constraint*, backtracking melakukan langkah kembali ke solusi partial sebelumnya dan memilih nilai lain yang belum dicoba untuk variabel yang ingin diisikan, langkah tersebut berguna untuk menghindari eksplorasi lebih lanjut dari solusi partial yang salah. Kekurangannya adalah *backtracking* cenderung mencoba semua kombinasi untuk mendapatkan solusi. tetapi kekuatannya adalah pada banyaknya implementasi lain yang bisa dipakai bersamaan dengan *backtracking*, untuk menghidari banyak kombinasi *partial* dan mempercepat waktu penyelesaian. [WIKI]

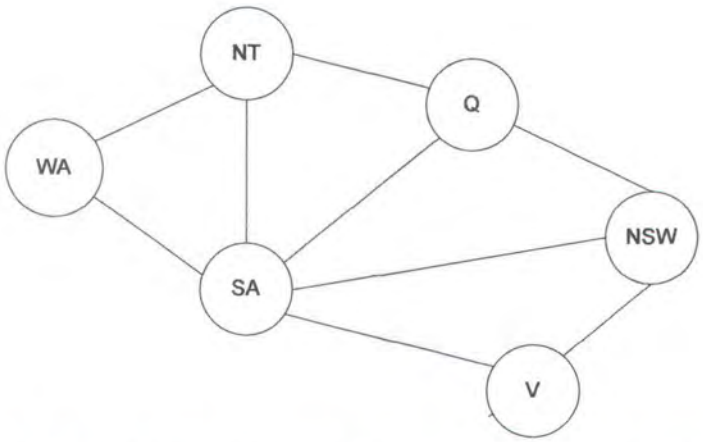
Keuntungan dari *backtracking* adalah pemeriksaan *consistency* hanya perlu dilakukan terhadap *assignment* yang terakhir dilakukan, karena pemeriksaan terhadap *assignment* yang sebelumnya sudah dilakukan sebelumnya. Jika *searching* berhasil mencapai kondisi dimana sudah tidak ada *inconsistency* dan variabel sudah terisi semua, maka sebuah *consistent complete assignment* akan didapatkan. Jika menginginkan beberapa solusi maka akan dilakukan *backtracking* dan mencari solusi selanjutnya.

Berdasarkan perbandingannya dengan input, masalah yang berkembang secara eksponensial (atau lebih cepat) dapat diselesaikan dengan *backtracking*. Konsep *backtracking* adalah membuat solusi dari komponen pada suatu waktu dan mengevaluasi kandidat solusi *partial* tersebut. Jika solusi *partial* tersebut bisa

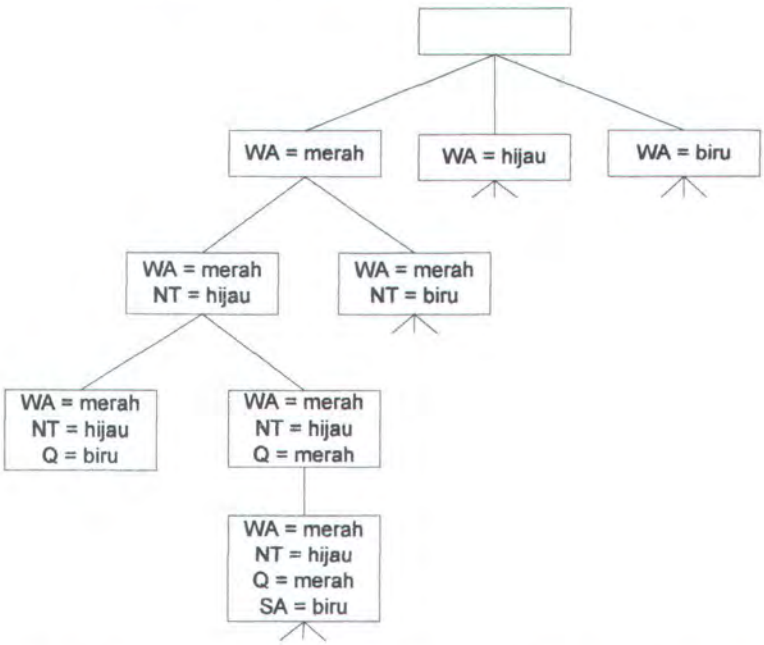
dikembangkan lebih jauh tanpa melanggar *constraint* permasalahan, maka *backtracking* akan mengambil langkah yang masih *legitimate* untuk menjadi komponen berikutnya. Jika langkah yang *legitimate* sudah tidak ada, tidak adanya komponen alternatif tersebut harus dicermati. Algoritma kembali mundur satu langkah untuk menganti komponen terakhir dari solusi *partial* dengan langkah lain yang belum dipilih. Pendekatan yang dipakai tersebut memungkinkan untuk menyelesaikan masalah kombinasi dengan tingkat kesulitan dalam skala yang cukup besar, walaupun dalam kasus terburuknya, masih harus menghadapi permasalahan dari *eksponensial-explosion.* (*exhaustive search*)

Penyelesaian tersebut dapat digambarkan sebagai membuat *tree*, disebut *state-space tree*. *Root*-nya mewakili *initial state* sebelum pencarian solusi dimulai. *Node* pada pencarian level pertama di *tree* menunjukan pilihan yang dibuat untuk komponen pertama di solusi, node pada level dua menunjukan pilihan untuk komponen kedua, dan begitu seterusnya. Sebuah *node* pada *state-space tree* disebut menjanjikan (*promising*) jika menunjukan bahwa solusi *partial* yang sedang dibuat masih menuju *solusi complete*. Sebaliknya disebut tidak menjanjikan (*nonpromising*). *Leaves* (daun) mewakili salah satu diantara *promising* atau *nonpromising* yang ditemukan oleh algoritma. Dalam kebanyakan kasus, *state-space tree* untuk algoritma *backtracking* dibuat berdasarkan *depth-first search*. Jika *node promising*, maka *child* dibuat dengan menambahkan pilihan pertama yang *legitimate* untuk menjadi solusi, dan proses berpindah ke *child*. Dan jika *node* ternyata menjadi *nonpromising*, algoritma *backtrack* ke *parent node* untuk mencari pilihan alternatif untuk komponen terakhir. Jika pilihan tersebut

juga tidak ada maka proses akan pindah lagi ke *parent* node diatasnya, dan demikian selanjutnya. Akhirnya jika ketemu solusi *complete* untuk suatu masalah, pencarian dapat berhenti (jika hanya satu solusi yang dicari) atau melakukan *backtrack* dan melanjutkan untuk mencari solusi yang lainnya.



Gambar 2.5 **:** *Map coloring*, Masalah *map-colouring problem* dimana setiap *nodes* harus diisikan dengan salah satu dari warna : merah, hijau, biru. Tetapi warna *vertices* yang bertetangga tidak boleh sama.



Gambar 2.6 : *Search tree*, sebagian dari search tree yang dibuat oleh *backtracking* untuk masalah *map-colouring problem* digambar 2.4

2.7.2 Forward Checking

Forward checking adalah salah satu cara untuk mencegah terjadinya konflik. *Forward checking* berkerja dengan melarang nilai-nilai yang belum diisikan ke variabel untuk dipakai. Ketika suatu nilai diisikan ke suatu variabel, nilai yang berada di *domain* dari variabel yang konflik dengan *assignment* tersebut akan dihilangkan dari *domain* [RUS03][WIKI]. Keuntungan dari *forward checking* adalah jika suatu *domain* dari variabel menjadi kosong, maka akan segera diketahui bahwa solusi partial tersebut adalah *inconsistent*. Maka *forward checking* akan mengurangi percabangan dari pencarian yang dilakukan dengan *forward checking*. Kelebihan dari *forward checking* adalah nilai dapat dijamin *consistent* dengan variabel sebelumnya, jadi pemeriksaan suatu *assignment* terhadap *assignment* sebelumnya sudah tidak diperlukan lagi.

Forward Checking adalah suatu algoritma yang berkerja dengan menghapus variabel yang sudah tidak valid lagi dari *domain*. Dengan melacak menggunakan *Forward Checking*, dalam setiap langkah akan diperiksa apakah setiap variabel masih memiliki domain yang legal, dan akan dapat diketahui lebih cepat apa hasil dari suatu pencarian, sehingga faktor percabangan dari pencarian dapat dikurangi.

	WA			NT			Q			NSW			V			SA		
Domain Awal	M	H	B	M	H	B	M	H	B	M	H	B	M	H	B	M	H	B
WA = Merah	**M**				H	B	M	H	B	M	H	B	M	H	B		H	B
Q = Hijau	**M**					B		**H**		M		B	M	H	B			B
V = Biru	**M**					B		**H**		M					**B**			

Gambar 2.7 : *Forward checking*, Pencarian pada *map-coloring* dengan *forward checking*. WA = merah yang pertama diisikan, kemudian *forward checking* menghapus merah dari *domain* yang bertetangga dengan WA yaitu NT dan SA. Setelah Q = hijau, hijau dihapus dari domain NT, SA, dan NSW. Setelah V = biru, biru dihapus dari domain NSW dan SA, menjadikan SA sudah tidak memiliki domain lagi. Dan sebelum *backtracking* dapat menemukan bahwa SA sudah tidak memiliki *domain* ketika akan melakukan pengisian pada SA, *Forward checking* sudah dapat memberitahukannya terlebih dahulu ketika melakukan assignment pada V.

2.7.3 Minimum Remaining Value

Minimum Remaining Value (MRV) adalah suatu teknik yang dikembangkan untuk menangani masalah besarnya kemungkinan gagal pada pencarian menggunakan CSP, MRV berkerja dengan memilih variabel yang memiliki *domain* legal paling sedikit (memiliki kemungkinan untuk membuat suatu *dead-end* paling besar) untuk diisikan terlebih dulu [RUS03]. Sehingga kemungkinan untuk terjadi suatu kegagalan akan lebih sedikit. Ini juga dibilang "*most constrained variable*" karena memilih variabel yang memiliki kemungkinan untuk membuat kesalahan sangat besar, maka MRV juga telah berhasil lebih lanjut memotong percabangan dari pencarian.

2.8 Active Database

Secara tradisional basis data hanya dipandang sebagai suatu tempat penyimpanan informasi yang dibutuhkan aplikasi, dan hanya diakses melalui program atau *interface*. *Database Management System* (DBMS) tradisional di sebut *passive* dalam arti perintah hanya dijalankan di dalam database jika diminta

user atau program. *Active database* merupakan pengembangan dari database yang memindahkan sifat *reactive* program ke dalam database.

Active database adalah sistem database yang dapat melakukan suatu aksi. Dalam *active database* sistem tidak hanya menyimpan data tetapi juga bertanggung jawab terhadap aksi dari sebuah *event* [SIL97]. *Active database* memiliki sebuah *active rules* yang menentukan kapan harus melakukan sebuah aksi, dan kapan aksi tersebut berakhir.

Active Database rules dikatakan *Event-Condition-Action* atau model ECA yaitu ketika mendeteksi sebuah *Event* dan memenuhi suatu *Condition (kondisi)* maka sebuah *Action (aksi)* dijalankan.

Jadi sebuah rule dalam ECA memiliki tiga komponen :

1. Event yang men-*trigger rule*.
2. Kondisi yang menentukan *action rule* apa yang harus diambil.
3. Aksi apa yang akan dilakukan.

Event didefinisikan dengan kejadian yang terjadi pada suatu waktu, dan bukan pada suatu periode waktu. Suatu *event* akan menjalankan rule yang menentukan suatu aksi tertentu bisa dijalankan atau tidak. Jika kondisi dipenuhi maka aksi dapat dijalankan, jika tidak maka tidak ada aksi yang dijalankan. Kondisi tersebut dapat berupa query dari database yang memberikan hasil, atau suatu prosedur yang memberikan nilai. Dan aksi yang dilakukan dapat meliputi operasi database seperti *retrieval, insertion, update,* atau *deletion* dari data. Dan juga bisa memanggil *stored procedure* atau *function* untuk mengeksekusi suatu transaksi.

Dalam sebuah active database ada hal-hal yang perlu diperhatikan yaitu :

1. *Termination*, suatu eksekusi dari aksi dapat membuat *event* lain terjadi. *Event* ini dapat merupakan suatu rule lain yang dijalankan. Jika hal ini berulang maka dapat menjadi *loop* yang tidak memiliki akhir.
2. *Priority*, jika beberapa *rule* ditrigger oleh *event* yang sama, maka harus dieksekusi berdasarkan urutannya rulenya.
3. *Error Handling*, jika eksekusi dari *rule* menghasilkan *error* maka suatu *error recovery* harus dijalankan.

BAB III

PERANCANGAN DAN IMPLEMENTASI PERANGKAT LUNAK

Pada bab ini akan dibahas mengenai Sistem Penjadwalan yang dibuat dalam tugas akhir ini. Sistem penjadwalan yang dibuat ditujukan untuk memenuhi kebutuhan Jurusan Teknik Informatika dalam membuat jadwal pada setiap semester. Sistem Penjadwalan ini dibuat dengan mengikuti acuan benar menurut aturan-aturan penjadwalan, dengan asumsi mahasiswa mengikuti semester yang seharusnya. Kemudian selanjutnya dalam bab ini akan diuraikan mengenai penerapan teori dan algoritma yang dikembangkan dalam bentuk perangkat lunak. Berikut ini akan dijelaskan mengenai lingkungan pembangunan Sistem penjadwalan, meliputi hardware, software, dan modul-modul yang digunakan.

3.1 Faktor-faktor yang mempengaruhi jadwal

Agar suatu jadwal dapat dibuat dengan mengikuti peraturan-peraturan penjadwalan . Ada beberapa faktor yang berpengaruh dalam pembentukan jadwal, yaitu :

1. Dosen

   Seorang dosen tidak dapat mengajar beberapa matakuliah pada jam yang sama. Selain itu, seorang dosen terkadang hanya dapat mengajar

pada jam-jam dan hari-hari tertentu saja, sehingga perlu untuk memesan jadwal khusus yang tidak dapat diganggu matakuliah yang lain.

2. Ruang

Mengingat jumlah ruang yang dimiliki sangat terbatas, maka perlu diperhatikan ruang yang tersedia agar tidak menggangu jalannya perkuliahan. Dan jadwal tidak akan keluar dari ruang yang sudah ditentukan.

3. Waktu

Waktu-waktu perlu untuk batasan berapa menit yang diperlukan untuk satu jam kuliah. Selain itu ada hari-hari dimana jam kuliah dibatasi sampai jam tertentu. Dengan batasan–batasan waktu ini, jadwal hanya akan berada pada waktu yang ditentukan.

4. Matakuliah

Mengingat setiap matakuliah memiliki semester matakuliah itu diajarkan. Maka perlu adanya aturan yang membatasi penjadwalan matakuliah, agar matakuliah itu sesuai dengan aturan-aturan penjadwalan.

3.2 Kriteria jadwal kuliah

Penjadwalan matakuliah adalah proses penempatan matakuliah kepada waktu, dimana penempatan matakuliah tersebut harus memenuhi syarat-syarat tertentu.

Kriteria jadwal yang baik memang tidak begitu jelas, karena banyak pihak yang terlibat. Sehingga setiap pihak pasti memiliki kriteria menurut

kepentingannya. Pada tugas akhir ini akan dibatasi kriteria sebuah jadwal yang ideal untuk membangun perangkat lunak.

Pada tugas akhir ini akan digunakan constraint, asumsi, dan rule sebagai berikut :

- Ada sejumlah matakuliah yang ditawarkan pada satu semester yang ingin dibuat jadwalnya.
- Setiap matakuliah dijadwalkan hanya satu kali (pada setiap pembuatan penjadwalan).
- Penjadwalan dibuat untuk menjadwalkan matakuliah dengan pengulangan selama satu minggu.
- Dalam satu waktu tidak boleh ada dosen yang mengajar di dua tempat.
- Sebuah ruang tidak boleh dipakai oleh lebih dari satu matakuliah pada waktu yang bersamaan.
- Matakuliah yang memiliki semester yang sama tidak boleh berada pada satu hari dan jam yang sama, kecuali matakuliah tersebut memiliki kode kuliah yang sama (berbeda kelas).
- Matakuliah pada semester 6, 7 dan 8 tidak boleh berada pada satu hari dan jam yang sama dengan matakuliah pilihan.
- Matakuliah yang memesan waktu dan ruang, hanya bisa ditempatkan pada waktu dan ruang tersebut.
- Jadwal hanya sampai hari jumat.
- Jadwal berada dalam batas waktu dan ruang yang ditentukan.
- Hari jumat jam ketiga dianggap selalu kosong.

Kebutuhan dalam penjadwalan di Jurusan Teknik Informatika ditujukan agar mahasiswa dapat mengambil matakuliah sesuai keinginannya, tetapi dalam prakteknya hal ini sangat sulit dipenuhi karena matakuliah yang diambil oleh mahasiswa sangat variatif, Karena itu penjadwalan lebih ditujukan untuk memenuhi aturan penjadwalan. Dan untuk menangani masalah yang dapat dialami oleh mahasiswa telah dibuat aturan-aturan dibawah ini yang dapat membantu. Kemudian dalam sistem ini juga akan dilakukan suatu optimalisasi yaitu mencari waktu tunggu yang paling kecil.

Aturan-Aturan tambahan yang dapat dipilih untuk dipergunakan :

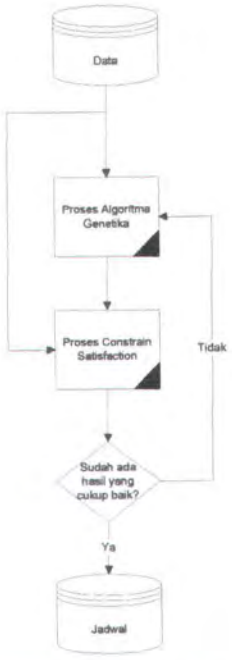
- Dalam satu hari tidak boleh ada jadwal yang memiliki semester sama lebih dari dua kali.
- Matakuliah yang sama dengan dosen yang berbeda harus terdapat pada waktu yang sama. Kecuali matakuliah yang sama dengan dosen yang sama boleh ditempatkan pada waktu yang berbeda.
- Matakuliah yang ditempatkan pada jam yang sama diharuskan memiliki selisih beberapa semester.
- Dosen dibatasi dalam satu hari hanya mengajar dua kali.
- Matakuliah pilihan dibatasi dalam satu hari hanya dua kali.

Dalam prakteknya sangat sulit untuk memenuhi semua constraint yang berada diatas, karena itu pilihan diatas menjadi optional. Dan ditentukan yang mana yang ingin dipergunakan.

3.3 Perancangan perangkat lunak

Ada tiga hal yang diperhatikan dalam perancangan perangkat lunak ini yaitu perancangan data, perancangan proses, dan perancangan antarmuka. Secara umum proses yang dilalui data dapat dilihat pada gambar 3.1. Untuk menjadikannya lebih sederhana, maka perancangan perangkat lunak ini dibagi ke dalam beberapa tahapan meliputi :

1. Perancangan Data.
2. Perancangan Proses.
3. Perancangan Antarmuka.



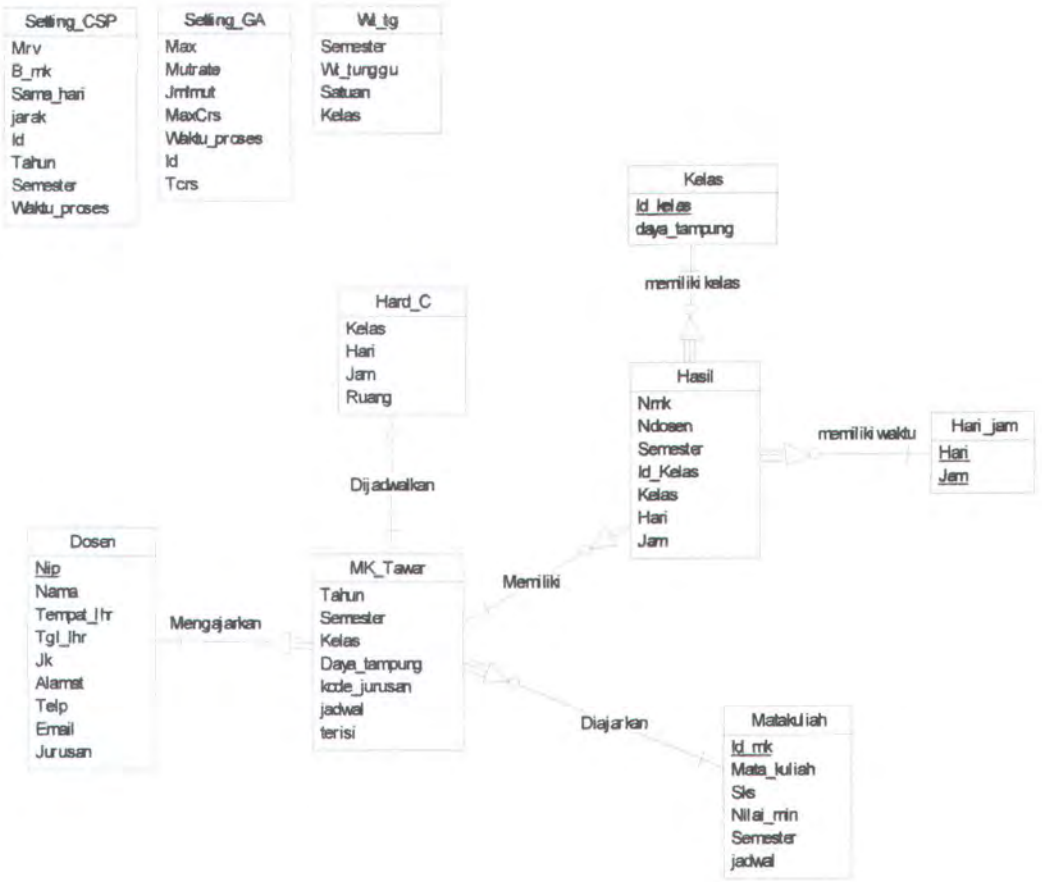
Gambar 3.1 Diagram urutan proses yang dilalui data

3.3.1 Perancangan data

Data-data yang digunakan dalam perangkat lunak ini dapat dibedakan menjadi tiga yaitu data master, data proses dan data keluaran.

3.3.1.1 Data master

Data master adalah data-data yang tersimpan pada database. Digunakan sebagai input pada perangkat lunak. Data-data untuk membuat penjadwalan ada yang mengambil dari FRS-online Jurusan Teknik Informatika ITS, adapun entitas-entitas yang diambil adalah entitas dosen, entitas matakuliah, dan entitas matakuliah tawar. Sedangkan data-data entitas lain yang digunakan, dibuat untuk membangun Sistem Penjadwalan ini. Data master yang diolah dalam sistem ditunjukan dalam gambar 3.2 :



Gambar 3.2 ER-D pada sistem penjadwalan.

Tabel mk_tawar	
id_mk	berisi kode matakuliah
tahun	tahun matakuliah tersebut diajarkan
semester	semester ganjil atau genap
kelas	kelas yang ditawarkan
daya_tampung	jumlah maksimal mahasiswa yang dapat mengambil matakuliah tersebut
Kode_jurusan	kode jurusan matakuliah tersebut
nip	nip dosen yang mengajar
terisi	mahasiswa yang mengambil matakuliah tersebut

Entitas matakuliah tawar adalah tabel yang berisi seluruh matakuliah yang ditawarkan pada tahun dan semesternya (ganjil/genap).

Data matakuliah ditawarkan diperlukan untuk mengetahui matakuliah-matakuliah apa saja yang harus untuk dibuat jadwalnya. Dari data matakuliah ditawarkan hanya akan diambil sebagian saja yang dipakai sebagai inputan, yaitu nip dosen yang mengajarkan, kode matakuliah, tahun, kelas dan daya tampung.

Tabel matakuliah	
id_mk	kode matakuliah
mata_kuliah	nama matakuliah
sks	jumlah sks matakuliah tersebut
nilai_min	nilai minimal untuk lulus
smt	semester matakuliah seharusnya diambil
jadwal	matakuliah ini diajarkan atau tidak

Entitas matakuliah adalah tabel yang berisikan data-data matakuliah yang tersedia pada suatu jurusan.

Data matakuliah diperlukan untuk mengetahui semester berapa matakuliah itu seharusnya diambil. Dengan melakukan query pada tabel matakuliah ditawarkan dan matakuliah maka akan didapat semester matakuliah itu seharusnya diambil.

Tabel hari_jam	
hari	Hari yang tersedia
jam	Jam yang tersedia

Entitas hari dan jam adalah tabel yang berisi hari dan jam yang mungkin untuk dibuat jadwalnya.

Data waktu diperlukan untuk mengetahui waktu yang tersedia dalam satu minggu untuk dibuat penjadwalannya.

Tabel hard_c	
id_mk	kode matakuliah.
nip	Nip dosen pengajar
smt	semester matakuliah ditawarkan
kelas	kode kelas matakuliah
hari	hari yang dipesan
jam	jam yang dipesan
ruang	ruang yang dipesan

Entitas Hard_C adalah tabel yang berisi data-data matakuliah yang memesan tempat.

Data hard constraint diperlukan untuk mengetahui matakuliah apa yang memesan waktu dan ruang.

Tabel kelas	
id_kelas	kode kelas
dy_tamp	jumlah banyak siswa yang dapat ditampung oleh kelas

Entitas kelas adalah tabel yang berisikan data kelas yang dimiliki suatu jurusan.

Data ruangan diperlukan untuk mengetahui berapa banyak tempat pada satu waktu yang bisa dipakai. Dan jumlah daya tampungnya.

Tabel hasil	
id_mk	kode matakuliah yang ditawarkan
nip	nip dosen yang mengajarkan matakuliah
Id_kelas	kelas matakuliah tersebut dijadwalkan
smt	semester matakuliah tersebut seharusnya diajarkan
hari	hari pelajaran tersebut dijadwalkan
jam	jam matakuliah tersebut dijadwalkan
nmk	nama matakuliah yang ditawarkan
ndosen	nama dosen yang mengajarkan
semester	semester matakuliah tersebut seharusnya diajarkan
tahun	tahun matakuliah ditawarkan
Kd_jur	kode jurusan matakuliah
kelas	kode kelas suatu matakuliah

Entitas hasil ada tabel yang berisikan data-data matakuliah yang sudah dijadwalkan.

Data hasil diperlukan untuk menampung jadwal yang sudah dibuat sistem penjadwalan.

Tabel dosen	
nip	nip dosen
nama	nama seorang dosen
tempat_lhr	tempat lahir dosen
tgl_lhr	tanggal lahir dosen
jk	jenis kelamin dosen
alamat	alamat rumah dosen
telp	no telepon dosen
email	alamat email dosen
Jurusan	jurusan tempat dosen mengajar

Entitas dosen berisi data-data mengenai dosen yang mengajar. Entitas dosen diperlukan untuk menentukan dosen yang mengajar pada suatu matakuliah yang ditawarkan.

Basis data diatas merupakan data utama yang akan diolah. Selain data diatas, diperlukan juga data masukan meliputi :

Tabel setting_ga	
max	merupakan jumlah maksimal populasi dalam algoritma genetika
mutrate	tingkat kelajuan mutasi
jmlmut	tingkat banyaknya mutasi
maxcrs	tingkat banyaknya crossover
waktu_pros	waktu lama maksimal algoritma genetic dijalankan
min	nilai fitness maksimal yang bias diterima
id	kode setting
tcrs	banyaknya crossover yang terjadi

Entitas setting_ga berisi data setting untuk algoritma genetika.

Entitas setting_ga diperlukan untuk mengatur jalannya proses algoritma genetika yang dipakai dalam sistem ini.

Tabel setting_csp	
mrv	menggunakan metode minimum remaining value
b_mk	membatasi agar satu hari hanya ada dua matakuliah untuk satu semester yang sama
sama_hari	mengharuskan satu matakuliah pada hari dan jam yang sama
jarak	membatasi suatu semester agar memiliki jarak dengan semester yang berada diatas dan dibawahnya, untuk tidak berada pada waktu yang sama
Tahun	tahun yang ingin dibuat jadwalnya
semester	semester yang ingin dibuat jadwalnya (ganjil / genap)
id	kode setting

Entitas setting_csp berisi data setting untuk *constraint satisfaction.*

Entitas setting_csp diperlukan untuk mengatur jalannya proses *constraint satisfaction* yang dipakai dalam sistem ini.

Tabel Wt_tg	
Smt	semester matakuliah diajarkan
Wt_tunggu	waktu tunggu
Satuan	satuan waktu tunggu
Kelas	kelas matakuliah ditawarkan

Entitas Wt_tg berisi data statistic hasil pengolahan algoritma genetika.

Entitas Wt_tg diperlukan untuk mengatur menyimpan hasil perhitungan fitness yang dihasilkan oleh algoritma genetika.

3.3.1.2 Data Proses

Data proses adalah data-data yang digunakan selama proses pembuatan jadwal. Data ini diolah dari data masukan dan digunakan untuk menghasilkan data keluaran. Data-data utama pada proses ini dibagi menjadi dua berdasarkan tahapan pemrosesannya:

1. **Pemrosesan Algoritma Genetika**

   1. Data kromosom.

      Kromosom adalah data yang dipakai diseluruh proses algoritma genetika. kromosom merupakan kumpulan gen. Kromosom ini memiliki suatu data yang akan dicari nilai fitnessnya dengan melihat hasil kombinasi dari setiap gen. Data kromosom ini diambil dari data matakuliah tawar dalam database.

      Data gen terdiri dari :

      - Nip, Berisi Nip dosen yang mengajarkan suatu matakuliah.
      - Id_mk, Kode matakuliah yang ditawarkan.
      - Smt, Semester seharusnya matakuliah tersebut diambil.
      - Kelas, Jenis kelas suatu matakuliah. (XA, XB, XR, A, B, C, XRA, XRB).
      - Jenis, tipe matakuliah (kosong - diajarkan, 1 - tidak diajarkan, 2 - matakuliah pilihan).

Table 3.1. Contoh representasi gen dalam bentuk table :

NIP	ID_MK	SMT	KELAS	JENIS
132103631	CI1422	6	C	

Data kromosom terdiri dari :

- Data, Berisi kumpulan gen yang berupa matakuliah yang ditawarkan pada semester ini.
- Fit, Merupakan nilai kualitas suatu kromosom.

Table 3.2. Contoh representasi kromosom dalam bentuk table :

NIP	ID_MK	SMT	KELAS	JENIS	FIT
132103631	CI1422	6	C		
132125674	CI1305	2	A		1.5
131285253	CI1305	2	B		
132306544	CI1305	2	C		
131996151	CI1205	8	A		
131996151	CI1205	8	B	2	
130816212	CI1307	2	A		
130816212	CI1307	2	B		
132298829	CI1307	2	C		
132125657	CI1414	4	A		
132296226	CI1414	4	B	1	

2. Pemrosesan Constraint Satisfaction

1. Data Variable.

Variable merupakan salah satu bagian *Constraint satisfaction* yang paling penting, yaitu suatu permasalahan yang harus diselesaikan dengan diisikan nilai-nilai dari domain sehingga semuanya terisi.

Data tersebut akan dipakai dalam data masukan *constraint satisfaction* berbentuk :

Data Variable terdiri dari kumpulan:

- Nip, berisi nomor induk dosen yang mengajar.

- Id_MK, berisi kode matakuliah.
- Smt, berisi semester matakuliah itu diajarkan.
- Kelas, berisi kode kelas matakuliah (XA, XB, XR, A, B, C, XRA, XRB).
- Jenis, tipe matakuliah (kosong - diajarkan, 1 - tidak diajarkan, 2 - matakuliah pilihan).

Table 3.3. Contoh representasi data variable :

NIP	ID_MK	SMT	KELAS	JENIS
132103631	CI1422	6	C	
132125674	CI1305	2	A	
131285253	CI1305	2	B	
132306544	CI1305	2	C	
131996151	CI1205	8	A	
131996151	CI1205	8	B	2
130816212	CI1307	2	A	
130816212	CI1307	2	B	
132298829	CI1307	2	C	
132125657	CI1414	4	A	
132296226	CI1414	4	B	1

2. Data Domain.

Data Domain merupakan bagian dari *constraint satisfaction* yang mengatur nilai-nilai yang boleh diisikan ke dalam variable.

Data Jadwal terdiri dari :

- Hari, hari matakuliah dapat diajarkan.
- Ruang, ruang yang dapat dipakai untuk mengajar.
- Jam, jam matakuliah tersebut dapat diajarkan.

Data Domain terdiri dari :

- Domain, merupakan kumpulan dari jadwal.

Table 3.4. Contoh representasi data domain :

HARI	ID_KELAS	JAM
1	1	1
1	2	1
1	3	1
1	4	1
1	5	1
1	6	1
1	1	2
1	2	2
1	3	2
1	4	2
1	5	2
1	6	2
1	1	3
1	2	3

3. Data solusi

Data jadwal merupakan tempat menyimpan solusi partial yang sudah ditemukan. Yang merupakan solusi kosong pada awalnya dan dengan semakin banyak variabel yang terisi maka akan semakin bertambah solusi partial. Sehingga pada akhirnya akan terbentuk solusi penuh yang merupakan hasil dari pencarian.

Data solusi terdiri dari kumpulan:

- No, nomor urutan pada variable
- Nip, berisi nomor induk dosen yang mengajar.
- Id_MK, berisi kode matakuliah.
- Smt, berisi semester matakuliah itu diajarkan.
- Kelas, berisi kode kelas matakuliah.
- Hari, hari matakuliah dapat diajarkan.
- Ruang, ruang yang dapat dipakai untuk mengajar.
- Jam, jam matakuliah tersebut dapat diajarkan.

- Jenis, tipe matakuliah (diajarkan, tidak diajarkan, matakuliah pilihan).

Table 3.5. Contoh representasi data solusi :

NIP	ID_MK	SEMESTER	KELAS	HARI	ID_KELAS	JAM
051100002	IF1416	4	B	1	2	1
132125674	IF1402	2	B	1	4	1
132303065	MU1126	8	B	1	6	1
132125657	IF1404	2	A	1	3	2

3.3.1.3 Data Keluaran

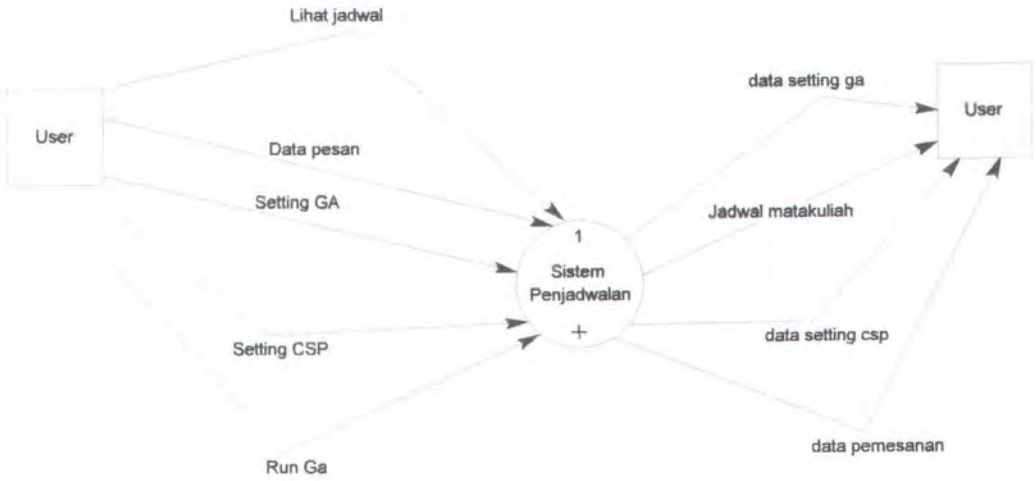
Data keluaran adalah informasi yang dihasilkan oleh perangkat lunak untuk pengguna. Pada tahapan ini akan didapatkan jadwal yang telah dioptimasi pada waktu tunggunya. Jadwal yang dikeluarkan merupakan hasil yang dianggap paling baik setelah proses selesai. Dengan kata lain adalah kromosom yang paling memiliki nilai fitness paling baik. Data keluaran ini berisi data solusi pada *constraint satisfaction problem* yang dimasukan ke dalam tabel hasil dan data nilai *fitness* dari solusi yang dimasukan ke dalam tabel wt_tg.

3.3.2 Perancangan Proses

Perancangan Proses digunakan untuk menggambarkan sejumlah proses terstruktur dalam sistem, berorientasikan pada aliran proses yang terjadi. Dalam perancangan proses kali ini digunakan Data Flow Diagram (DFD) yang dibuat dengan Power Designer 6.

3.3.2.1 DFD level 0

DFD level 0 (gambar 3.3) menjelaskan gambaran umum mengenai sistem yang akan dibuat. Pada diagram tersebut, hanya ada satu entitas luar yang terlibat dalam sistem ini yaitu user.

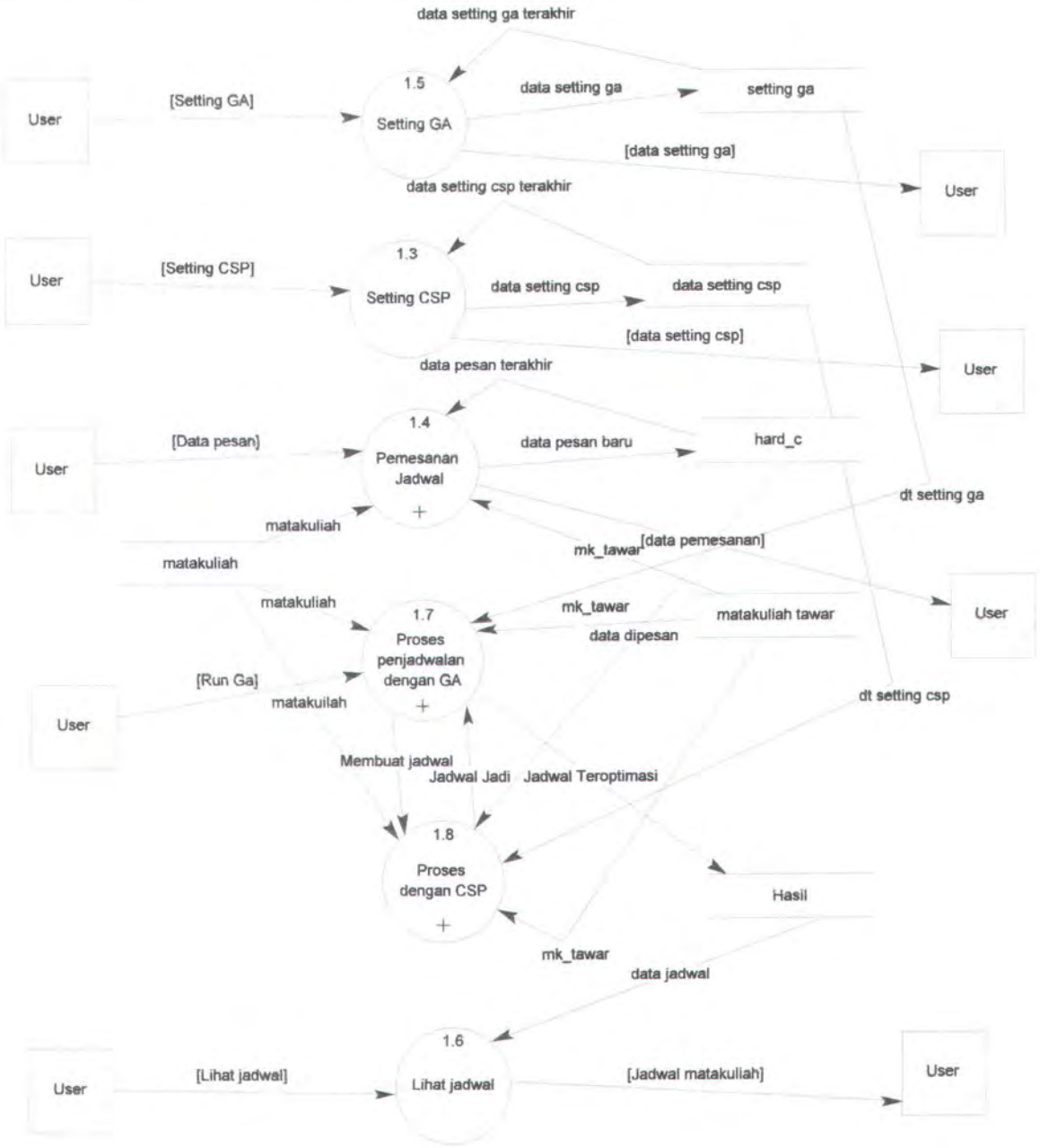


Gambar 3.3 DFD level 0

Terdapat beberapa proses yang bisa dilakukan oleh user dalam perangkat lunak ini yang diantaranya adalah:

1. Melakukan setting GA & CSP.
2. Melakukan pemesanan jadwal.
3. Mengeksekusi pembuatan Jadwal dengan menggunakan GA
4. Melihat Jadwal yang terbentuk.

3.3.2.2 DFD level 1 (Sistem Penjadwalan)



Gambar 3.4 DFD level 1

Pada DFD level 1 (gambar 3.4) ini terdapat 6 proses utama, yaitu:

1. Proses setting GA.

   Melakukan perubahan nilai setting algoritma genetika, agar algoritma genetika berjalan sesuai yang diharapkan oleh pengguna.

2. Proses setting CSP.

Melakukan perubahan nilai *setting* atau *constraint* pada CSP, agar jadwal yang terbentuk sesuai dengan keinginan pengguna. Jika dalam *setting* terjadi perubahan tahun atau semester maka secara langsung *active database* akan menghapus data pemesanan jadwal.

3. Proses pemesanan jadwal.

Melakukan pemilihan terhadap jadwal matakuliah yang memesan waktu dan tempat tertentu, setiap penambahan pemesanan akan divalidasi oleh *active database*.

4. Proses pembuatan jadwal dengan menggunakan GA.

Melakukan Proses Pembuatan jadwal dengan menggunakan GA sebagai pengatur arah pencarian nilai yang optimal.

5. Melakukan pembuatan jadwal dengan CSP.

Melakukan Proses Pembuatan jadwal dengan menggunakan GA.

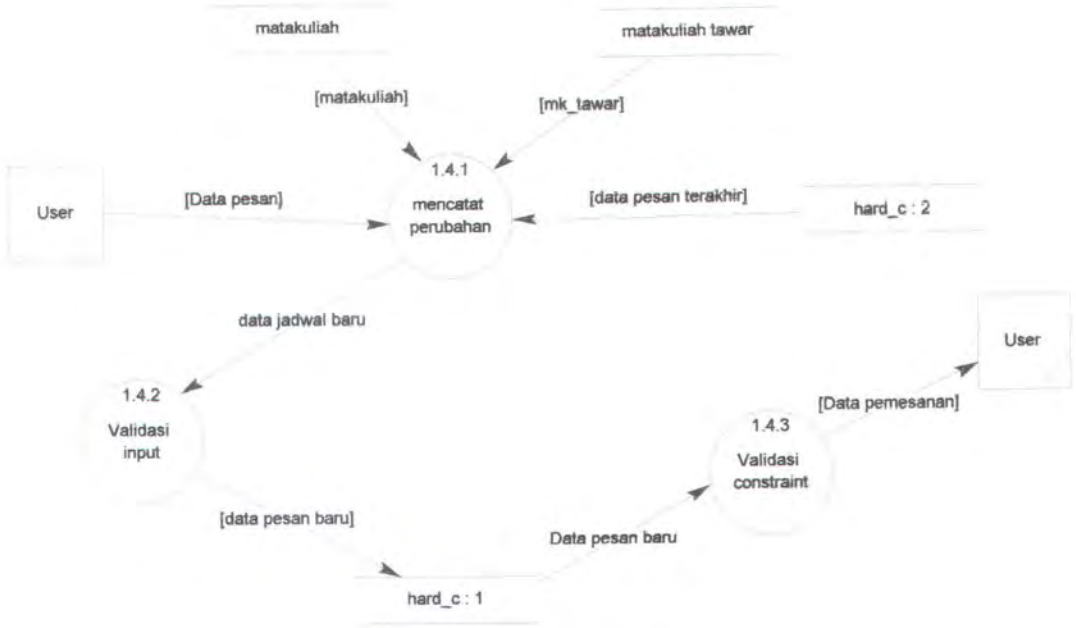
6. Lihat jadwal.

Melihat jadwal yang terbentuk.

3.3.2.3 DFD level 2

Pada aplikasi ini, proses pada DFD level 1 diturunkan ke DFD level 2 berikut ini:

3.3.2.4 Proses 1.4 (Pemesanan jadwal)



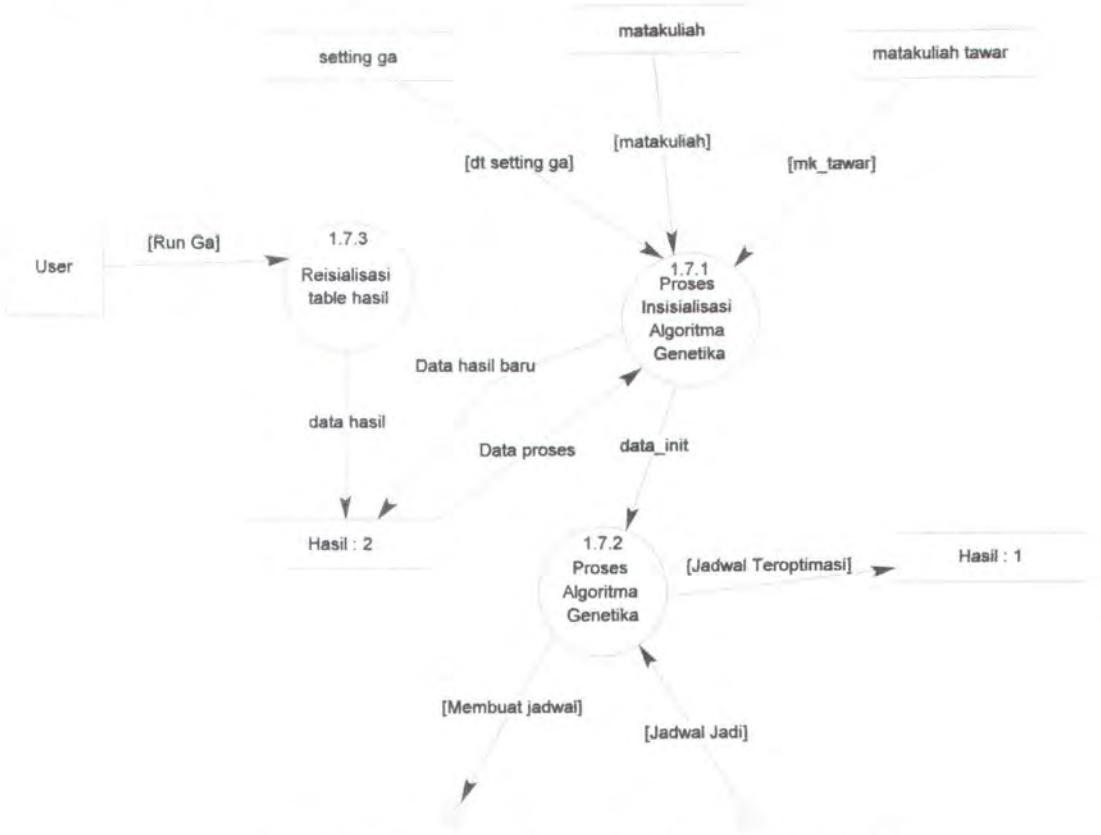
Gambar 3.5 DFD level 1

Pada tahap ini dapat dilakukan perubahan terhadap matakuliah yang dipesan, user dapat menambah ataupun mengurangi jumlah matakuliah. Dan matakuliah yang dipilih diambil dari jadwal matakuliah tawar yang nilai tahunnya dan semesternya diambil dari *setting* CSP. Matakuliah yang akan ditambahkan akan divalidasi melalui suatu *active database* sehingga matakuliah tersebut tidak akan melanggar aturan dasar penjadwalan

Secara umum tahap ini melakukan :

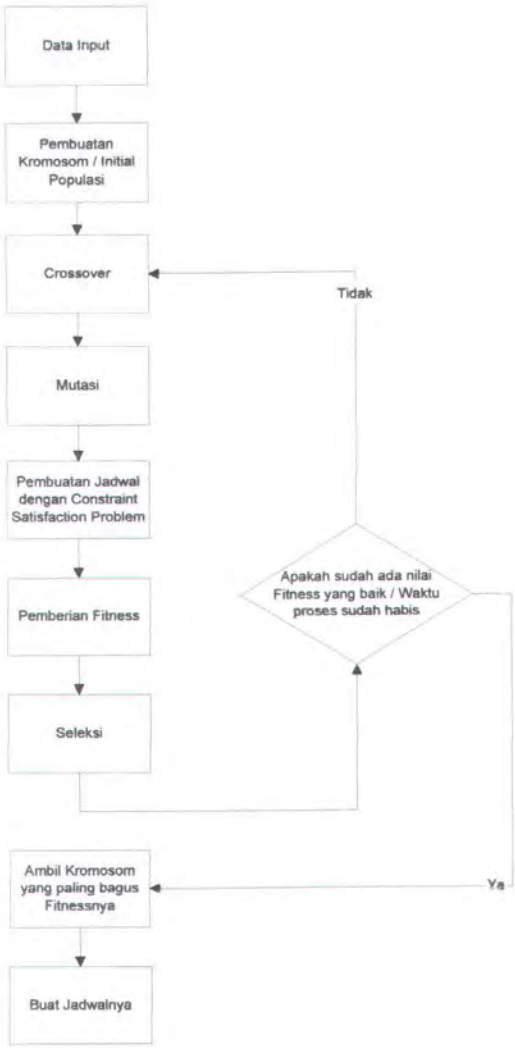
1. Pengambilan data jadwal yang berada di database dan belum dipesan.
2. Melakukan validasi input yang akan dimasukan ke dalam tabel hard_c.
3. Memvalidasi data melalui *active database* dengan cara menjalankan proses validasi *constraint*.
4. Mencatat perubahan yang terjadi ke database.

3.3.2.5 Proses 1.7 (Proses penjadwalan dengan GA)



Gambar 3.6 Proses 1.7 (penjadwalan dengan GA)

Tahap ini di mulai dengan mempersiapkan tabel hasil. Jika tabel hasil sudah diinisialisasi maka *active database* akan mentrigger prosedur GA untuk dijalankan. Kemudian pada tahap selanjutnya terjadi pembentukan kromosom pada algoritma genetika, kromosom dibentuk berdasarkan data kromosom (dijelaskan didata proses GA) yang datanya tersimpan di database. Setelah proses inisialisasi selesai maka proses mengambil setting GA untuk mengatur jalannya algoritma genetika sehingga sesuai dengan kebutuhan user. Kemudian algoritma genetika dijalankan untuk mendapatkan jadwal matakuliah yang optimal.



Gambar 3.7 Garis besar proses genetic algorithm

Secara umum proses inisialisasi melakukan :

1. Menginisialisasi table hasil, dengan menghapus data yang lama dan mengisikan data *default*.
2. Mengambil data setting GA dari database sebagai pengatur algoritma genetika.
3. Pengambilan data matakuliah tawar dan matakuliah dari database sebagai data kromosom.
4. Melakukan pembentukan kromosom sebagai *initial population.*
5. Melakukan proses *pre-processing* untuk memperkirakan apakah jadwal dapat dibuat. Proses ini melakukan

penghitungan jumlah data yang dibutuhkan untuk membuat jadwal dan membandingkannya dengan jumlah domain yang tersedia, jika domain tersebut mencukupi maka proses pembuatan jadwal dapat dilanjutkan, jika domain tidak mencukupi maka jadwal dianggap tidak dapat dibuat dan proses dibatalkan.

Model algoritma genetika yang akan digunakan untuk melakukan optimasi adalah sebagai berikut :

1. Seleksi.

   Pada seleksi, dilakukan penilaian atas nilai *fitness*. Sehingga *fitness* yang memiliki kualitas kromosom paling baik memiliki kemungkinan terpilih ke dalam generasi selanjutnya lebih besar. Seleksi yang dipakai disini adalah seleksi yang menggunakan metode *roulette wheel (*dijelaskan pada sub bab 2.6.2.1.1). Pada seleksi ini perlu diperhatikan jumlah maksimal populasi sebagai input. Agar populasi tidak menjadi terlalu besar dan memakan banyak waktu, dan agar populasi juga tidak menjadi terlalu kecil dan mengakibatkan kromosom terlalu mirip sehingga operasi kromosom tidak akan banyak berpengaruh.

2. Crossover.

   Crossover yang digunakan adalah penyilangan dua titik dengan permutasi. Pemilihan kromosom yang akan di *crossover* ditentukan oleh probabilitas yang ditentukan dalam setting algoritma genetika. Dan banyaknya gen yang ditukar juga mengikuti setting algoritma genetika. Di

dalam melakukan penyilangan setiap dua kromosom akan kemudian menghasilkan dua *offspring* baru hasil penyilangan.

Table 3.6. Contoh pengerjaan *crossover*

NIP	ID_MK	SMT
131285253	IF1402	2
132125674	IF1402	2
132137825	IF1403	1
132125657	IF1404	2
132296226	IF1404	2
130368610	IF1406	2
132256272	IF1406	2
131570363	IF1408	4
131633403	IF1408	4
132230429	IF1411	3
052100001	IF1412	4
131411100	IF1412	4
130816212	IF1414	4
132125657	IF1414	4
051100002	IF1416	4
132296226	IF1416	4
051100005	IF1502	6
132172210	IF1503	5
130532048	IF1504	6
051100001	IF1506	6
132206858	IF1506	6
131846108	IF1507	5
132163671	IF1509	5
132048148	IF1521	1
131933299	IF1522	2
131996151	IF1524	4
131996150	IF1525	5
131846108	IF1527	7
052100001	IF1529	1
051100005	IF1532	3
132048148	IF1537	8
131570363	IF1539	2
132303065	IF1545	7
131285253	IF1549	2
132163671	IF1550	3
132303065	MU1126	8
132163671	IF1410	4
132163671	IF1410	4

NIP	ID_MK	SMT
131570363	IF1408	4
131846108	IF1527	7
131411100	IF1412	4
052100001	IF1529	1
130368610	IF1406	2
132172210	IF1503	5
132163671	IF1509	5
132256272	IF1406	2
131633403	IF1408	4
132137825	IF1403	1
132125657	IF1404	2
131996151	IF1524	4
130816212	IF1414	4
132230429	IF1411	3
132206858	IF1506	6
051100005	IF1532	3
132163671	IF1410	4
132296226	IF1416	4
130532048	IF1504	6
131996150	IF1525	5
131285253	IF1549	2
132125674	IF1402	2
051100001	IF1506	6
132048148	IF1521	1
131933299	IF1522	2
132303065	IF1545	7
131570363	IF1539	2
131285253	IF1402	2
052100001	IF1412	4
131846108	IF1507	5
132048148	IF1537	8
051100002	IF1416	4
132303065	MU1126	8
132163671	IF1410	4
132296226	IF1404	2
132163671	IF1550	3
132125657	IF1414	4
051100005	IF1502	6

Jika terjadi pertukaran diantara dua kromosom di atas pada gen 1 sampai 8 maka hasil *crossover* menggunakan penyilangan dengan permutasi adalah dua kromosom sebagai berikut :

NIP	ID_MK	SMT
131570363	IF1408	4
131846108	IF1527	7
131411100	IF1412	4
052100001	IF1529	1
130368610	IF1406	2
132172210	IF1503	5
132163671	IF1509	5
132256272	IF1406	2
131633403	IF1408	4
132230429	IF1411	3
052100001	IF1412	4
132137825	IF1403	1
130816212	IF1414	4
132125657	IF1414	4
051100002	IF1416	4
132296226	IF1416	4
051100005	IF1502	6
132296226	IF1404	2
130532048	IF1504	6
051100001	IF1506	6
132206858	IF1506	6
131846108	IF1507	5
131285253	IF1402	2
132048148	IF1521	1
131933299	IF1522	2
131996151	IF1524	4
131996150	IF1525	5
132125674	IF1402	2
132125657	IF1404	2
051100005	IF1532	3
132048148	IF1537	8
131570363	IF1539	2
132303065	IF1545	7
131285253	IF1549	2
132163671	IF1550	3
132303065	MU1126	8
132163671	IF1410	4
132163671	IF1410	4

NIP	ID_MK	SMT
131285253	IF1402	2
132125674	IF1402	2
132137825	IF1403	1
132125657	IF1404	2
132296226	IF1404	2
130368610	IF1406	2
132256272	IF1406	2
131570363	IF1408	4
131633403	IF1408	4
131411100	IF1412	4
052100001	IF1529	1
131996151	IF1524	4
130816212	IF1414	4
132230429	IF1411	3
132206858	IF1506	6
051100005	IF1532	3
132163671	IF1410	4
132296226	IF1416	4
130532048	IF1504	6
131996150	IF1525	5
131285253	IF1549	2
131846108	IF1527	7
051100001	IF1506	6
132048148	IF1521	1
131933299	IF1522	2
132303065	IF1545	7
131570363	IF1539	2
132163671	IF1509	5
052100001	IF1412	4
131846108	IF1507	5
132048148	IF1537	8
051100002	IF1416	4
132303065	MU1126	8
132163671	IF1410	4
132172210	IF1503	5
132163671	IF1550	3
132125657	IF1414	4
051100005	IF1502	6

3. Mutasi.

Mutasi dilakukan setelah operasi kromosom ini dilakukan dengan menukar gen secara random. Dalam proses ini perlu diperhatikan berapa besarnya tingkat mutasi dan berapa kemungkinannya. Jika suatu mutasi banyak maka antara suatu generasi ke generasi selanjutnya akan kehilangan kemiripan dan pencarian akan menjadi random. Tetapi jika

terlalu sedikit maka kromosom akan menjadi terlalu mirip dan kromosom baru akan muncul terlalu lama di populasi. Contoh dalam table 3.7(a) pengerjaan mutasi dengan urutan gen yang dimutasikan : (12-22), (18-11), (19-10), (2-33), (29-32), (26-2), (12-22), (9-33), (18-25), (8-20), (17-13), (22-29), (16-12), (11-9), (37-35). Akan menghasilkan table 3.7(b).

Table 3.7. Contoh pengerjaan mutasi

NIP	ID_MK	SMT
131285253	IF1402	2
132125674	IF1402	2
132137825	IF1403	1
132125657	IF1404	2
132296226	IF1404	2
130368610	IF1406	2
132256272	IF1406	2
131570363	IF1408	4
131633403	IF1408	4
132230429	IF1411	3
052100001	IF1412	4
131411100	IF1412	4
130816212	IF1414	4
132125657	IF1414	4
051100002	IF1416	4
132296226	IF1416	4
051100005	IF1502	6
132172210	IF1503	5
130532048	IF1504	6
051100001	IF1506	6
132206858	IF1506	6
131846108	IF1507	5
132163671	IF1509	5
132048148	IF1521	1
131933299	IF1522	2
131996151	IF1524	4
131996150	IF1525	5
131846108	IF1527	7
052100001	IF1529	1
051100005	IF1532	3
132048148	IF1537	8
131570363	IF1539	2
132303065	IF1545	7
131285253	IF1549	2
132163671	IF1550	3
132303065	MU1126	8
132163671	IF1410	4
132163671	IF1410	4

(a)

NIP	ID_MK	SMT
131285253	IF1402	2
131996151	IF1524	4
132137825	IF1403	1
132125657	IF1404	2
132296226	IF1404	2
130368610	IF1406	2
132256272	IF1406	2
051100001	IF1506	6
132172210	IF1503	5
130532048	IF1504	6
132125674	IF1402	2
132296226	IF1416	4
051100005	IF1502	6
132125657	IF1414	4
051100002	IF1416	4
131411100	IF1412	4
130816212	IF1414	4
131933299	IF1522	2
132230429	IF1411	3
131570363	IF1408	4
132206858	IF1506	6
131570363	IF1539	2
132163671	IF1509	5
132048148	IF1521	1
052100001	IF1412	4
132303065	IF1545	7
131996150	IF1525	5
131846108	IF1527	7
131846108	IF1507	5
051100005	IF1532	3
132048148	IF1537	8
052100001	IF1529	1
131633403	IF1408	4
131285253	IF1549	2
132163671	IF1410	4
132303065	MU1126	8
132163671	IF1550	3
132163671	IF1410	4

(b)

4. Fitness.

Pada Fitness yang dilakukan adalah proses pemberian nilai yang mewakili kualitas dari kromosom. Hal pertama yang dilakukan proses ini adalah pembuatan jadwal sesuai kromosom yang dipilih, dengan memprosesnya dengan *Constraint Satisfaction Problem*, dan dari hasilnya akan diberi nilai fit.

Fungsi fitness yang dibuat merupakan rata-rata waktu tunggu mahasiswa dalam satu minggu, sehingga semakin kecil waktu tunggu maka akan semakin baik. Ditunjukan dengan rumus sebagai berikut :

$$f(fit) = \frac{\sum_{i=1}^{n} \frac{R(i)}{n} + \sum_{j=1}^{m} \frac{X(j)}{m}}{2}$$

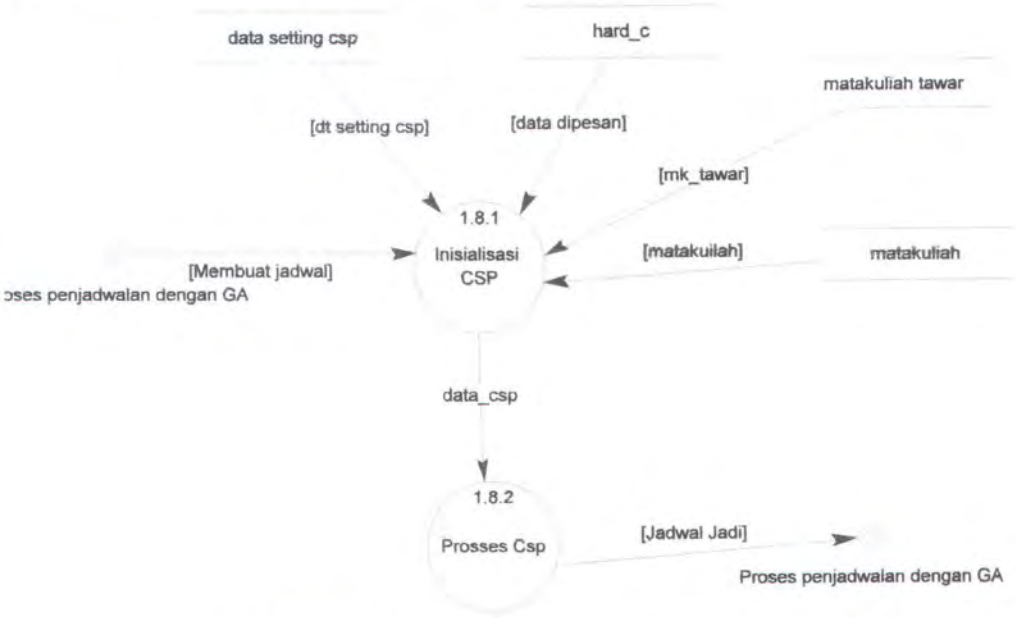
Dimana R(i) adalah waktu tunggu mahasiswa reguler dalam satu hari, dan X(j) adalah waktu tunggu mahasiswa ekstensi dalam satu hari. n adalah jumlah data mahasiswa reguler dalam satu minggu, m adalah jumlah data mahasiswa ekstensi dalam satu minggu. *f(fit)* adalah fungsi fitness.

Adapun proses yang terjadi dalam algoritma genetika adalah seperti yang terlihat dalam gambar 3.7, yaitu :

1. Inisialisasi algoritma genetika, melakukan persiapan data-data yang dibutuhan oleh algoritma genetika.
2. Melakukan pemilihan kromosom untuk dilakukan *crossover* dan mutasi. Dan kemudia hasilnya ditambahkan ke dalam populasi.
3. Melakukan pembuatan jadwal dari setiap data kromosom yang baru, dengan menggunakan *constraint satisfaction*.
4. Dari setiap data baru dicari nilai waktu tunggu rata-rata mahasiswa.

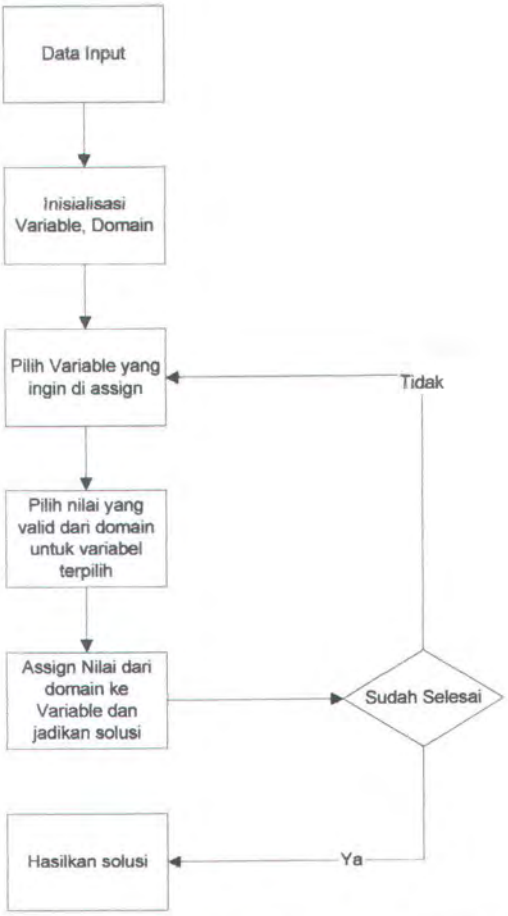
5. Melakukan seleksi, jika nilai *fitness* yang yang dicari belum tercapai dan waktu belum habis kembali ke langkah 2.
6. Ambil kromosom dengan nilai *fitness* terbaik sebagai solusi dan buat jadwalnya.

3.3.2.6 Proses 1.8 (Proses dengan CSP)



Gambar 3.8 Proses 1.8 (penjadwalan dengan CSP)

Proses pertama yang diambil adalah data setting CSP yang dipakai untuk mengatur pembentukan jadwal, setelah itu terjadi inisialisasi *Constraint Satisfaction Problem*, yaitu pembentukan solusi awal yang dibentuk dari data pesan jadwal yang sudah dibuat. setelah data pesan jadwal valid maka akan dilakukan suatu pembuatan jadwal dengan menggunakan CSP.



Gambar 3.9 Garis besar proses constraint satisfaction

Secara umum proses inisialisasi CSP melakukan :

1. Mengambil setting CSP dari database untuk menentukan *constraint* dan metode apa saja yang dipakai.
2. Data matakuliah tawar diberikan dari algoritma genetika dan data tersebut menjadi variable yang harus diselesaikan.
3. Mengambil data hard_c untuk inisialisasi variable yang dipesan terlebih dahulu sebelum melakukan penjadwalan pada variable yang lain.

Model *Constraint Satisfaction* yang digunakan untuk menyelesaikan masalah penjadwalan adalah sebagai berikut :

1. Penelusuran dengan Backtracking (BT).

   Algoritma *backtracking* yang digunakan disini menggunakan *Constructive methods*, yang berarti mengembangkan solusi partial sedikit demi sedikit dengan tetap menjaga *consistency*, sehingga mencapai *consistent complete assignment*.

2. Forward Checking (FC).

   Forward checking berkerja dengan membuat suatu tabel yang menyatakan nilai *domain* yang masih tersedia untuk setiap *variable*. Kemudian pada tabel ini dihilangkan domain yang sudah tidak boleh diambil lagi.

3. Minimum Remaining Value (MRV).

   Minimum remaining value dapat berkerja bersamaan dengan FC dengan memilihkan variable yang diisikan yaitu variable yang memiliki domain paling sedikit, karena variable yang domainnya paling sedikit mempunyai kesempatan gagal lebih besar. sehingga dapat memotong percabangan *tree* yang tidak perlu, dan mempercepat waktu pembuatan jadwal.

Adapun proses yang terjadi dalam *constraint satisfaction* adalah seperti yang terlihat dalam gambar 3.9, yaitu :

1. Inisialisasi, melakukan persiapan data-data yang dibutuhan oleh *Constraint Satisafaction.*
2. Memilih variabel yang ingin diisikan, jika melakukan pemilihan dengan menggunakan MRV maka diambil variable yang memiliki domain paling sedikit, jika tidak menggunakan MRV akan diambil secara berurutan sesuai urutan pada variabel.
3. Memilih nilai yang masih valid dari domain untuk variable yang dipilih.
4. Melakukan assignment dari variabel dan domain ke dalam solusi. Kemudian mengurangi domain valid dari setiap variabel berdasarkan constraint yang dimiliki variabel yang diassign.
5. Jika solusi sudah terbentuk maka lanjutkan ke langkah 6. Jika solusi belum ketemu maka kembali ke langkah 2.
6. Jika setiap variable sudah terisi maka solusi sudah terbentuk, kemudian CSP akan memberikan hasilnya kembali ke algoritma genetika untuk dievaluasi.

3.3.3 Perancangan antarmuka

Dalam pembuatan antarmuka diperlukan aplikasi yang memudahkan pengguna untuk dapat melihat tampilan yang diinginkan serta memberikan kenyamanan dalam melihatnya. Dengan memperhatikan kemudahan akses data maka perancangan antarmuka yang dibuat adalah aplikasi antarmuka berbasis web.

Dalam merancang aplikasi antarmuka ini harus diperhatikan hal-hal yang diperlukan oleh pengguna dalam pembuatan antarmuka, antara lain :

- Terdapat suatu mekanisme untuk mengontol jalannya Algoritma genetika dan *Constrain Satisfaction.*
- Terdapat suatu cara untuk suatu matakuliah memesan waktu-waktu tertentu.
- Adanya kemampuan untuk menampilkan isi data jadwal dalam format tertentu yang diinginkan.

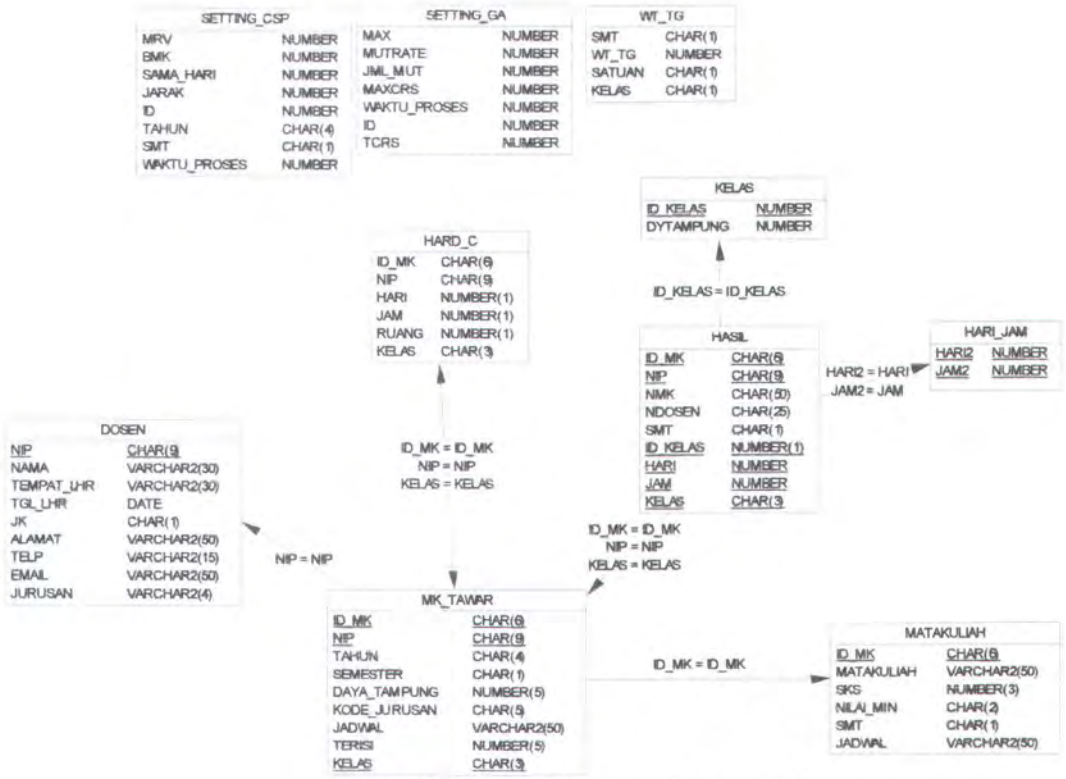
3.4 Implementasi Perangkat Lunak

Setelah melewati proses perancangan, dilakukan pembuatan perangkat lunak. Tahap-tahap pengimplementasian rancangan data, proses dan antarmuka akan dijelaskan dalam sub bab berikut.

3.4.1 Implementasi Data

Data yang diimplementasikan terdiri atas data master, data proses dan data keluaran sebagaimana telah diterangkan pada bab sebelumnya. Berikut ini merupakan pengelompokan data berdasarkan urutan prosesnya.

3.4.1.1 Data Master



Gambar 3.10 *Physical data model*

Berdasarkan *Conceptual data model* pada gambar 3.2 yang terdapat pada bab 3.3.1.1 maka deskripsi table *physical data model* yang dibentuk adalah seperti pada gambar 3.10. Seluruh penjelasan tabel dan attribute dapat dilihat pada bab 3.3.1.1

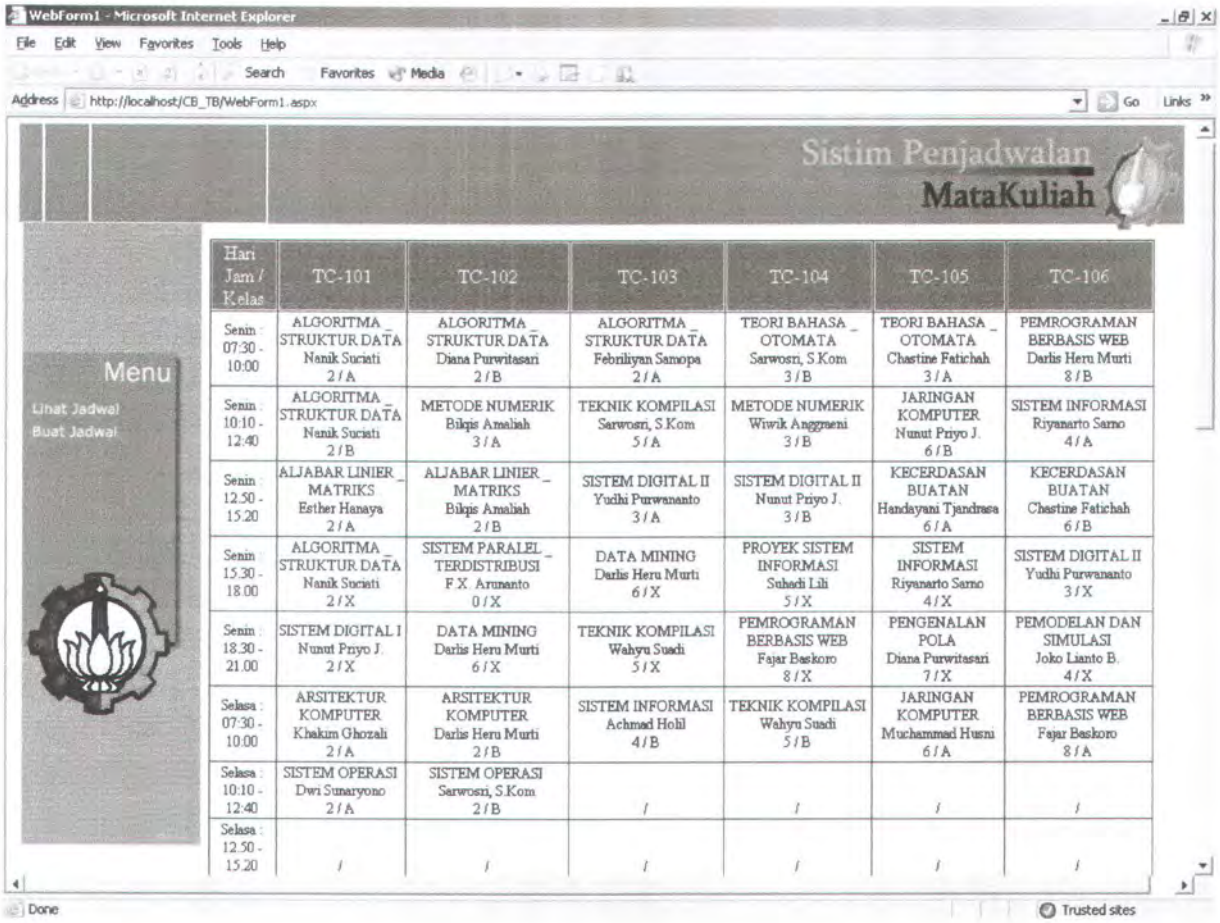
3.4.1.2 Data Keluaran

Hasil dari keseluruhan proses berupa jadwal yang sudah jadi yang setiap matakuliah yang ditawarkan memiliki waktu dan ruangan untuk perkuliahan, tersimpan dalam database tabel jadwal. Dimana antar setiap matakuliah yang ditawarkan tersebut tidak terjadi kesalahan dalam aturan penjadwalan sehingga

suatu matakuliah tidak bisa diajarkan. Adapun hasil dari keseluruhan proses adalah jadwal yang memiliki waktu tunggu yang paling sedikit.

3.4.2 Implementasi Antarmuka dan Perangkat Lunak

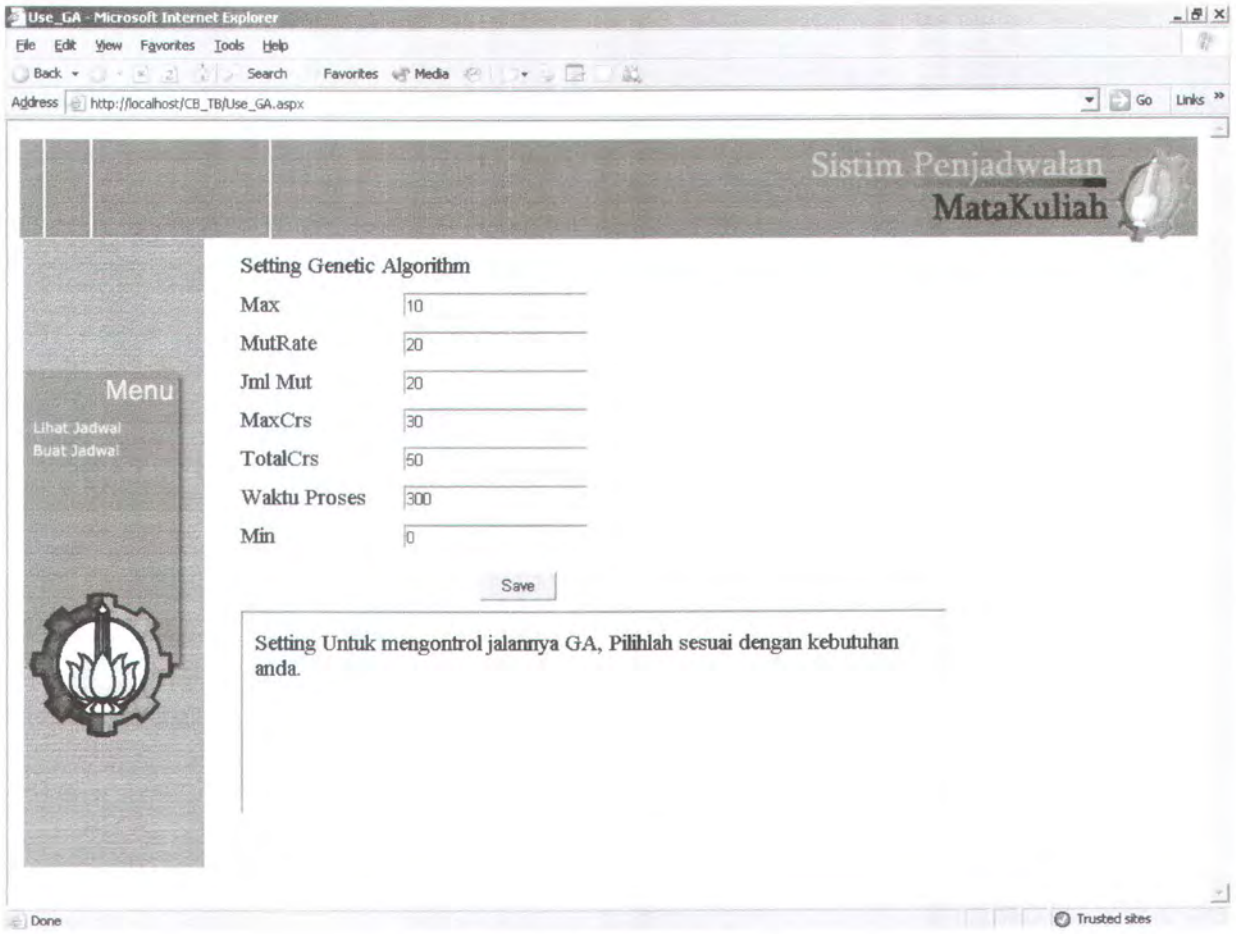
Pada bagian sebelumnya telah didefinisikan perancangan antarmuka berdasarkan proses yang berlangsung. Untuk merepresentasikan kebutuhan itu, dirancang form utama perangkat lunak dapat dengan pengimplementasian sebagai berikut:



Hari Jam / Kelas	TC-101	TC-102	TC-103	TC-104	TC-105	TC-106
Senin : 07:30 - 10:00	ALGORITMA _ STRUKTUR DATA Nanik Suciati 2 / A	ALGORITMA _ STRUKTUR DATA Diana Purwitasari 2 / B	ALGORITMA _ STRUKTUR DATA Febriliyan Samopa 2 / A	TEORI BAHASA _ OTOMATA Sarwosri, S.Kom 3 / B	TEORI BAHASA _ OTOMATA Chastine Fatichah 3 / A	PEMROGRAMAN BERBASIS WEB Darlis Heru Murti 8 / B
Senin : 10:10 - 12:40	ALGORITMA _ STRUKTUR DATA Nanik Suciati 2 / B	METODE NUMERIK Bilqis Amaliah 3 / A	TEKNIK KOMPILASI Sarwosri, S.Kom 5 / A	METODE NUMERIK Wiwik Anggraeni 3 / B	JARINGAN KOMPUTER Nunut Priyo J. 6 / B	SISTEM INFORMASI Riyanarto Sarno 4 / A
Senin : 12:50 - 15:20	ALJABAR LINIER _ MATRIKS Esther Hanaya 2 / A	ALJABAR LINIER _ MATRIKS Bilqis Amaliah 2 / B	SISTEM DIGITAL II Yudhi Purwananto 3 / A	SISTEM DIGITAL II Nunut Priyo J. 3 / B	KECERDASAN BUATAN Handayani Tjandrasa 6 / A	KECERDASAN BUATAN Chastine Fatichah 6 / B
Senin : 15:30 - 18:00	ALGORITMA _ STRUKTUR DATA Nanik Suciati 2 / X	SISTEM PARALEL _ TERDISTRIBUSI F.X. Arunanto 0 / X	DATA MINING Darlis Heru Murti 6 / X	PROYEK SISTEM INFORMASI Suhadi Lili 5 / X	SISTEM INFORMASI Riyanarto Sarno 4 / X	SISTEM DIGITAL II Yudhi Purwananto 3 / X
Senin : 18:30 - 21:00	SISTEM DIGITAL I Nunut Priyo J. 2 / X	DATA MINING Darlis Heru Murti 6 / X	TEKNIK KOMPILASI Wahyu Suadi 5 / X	PEMROGRAMAN BERBASIS WEB Fajar Baskoro 8 / X	PENGENALAN POLA Diana Purwitasari 7 / X	PEMODELAN DAN SIMULASI Joko Lianto B. 4 / X
Selasa : 07:30 - 10:00	ARSITEKTUR KOMPUTER Khakim Ghozali 2 / A	ARSITEKTUR KOMPUTER Darlis Heru Murti 2 / B	SISTEM INFORMASI Achmad Holil 4 / B	TEKNIK KOMPILASI Wahyu Suadi 5 / B	JARINGAN KOMPUTER Muchammad Husni 6 / A	PEMROGRAMAN BERBASIS WEB Fajar Baskoro 8 / A
Selasa : 10:10 - 12:40	SISTEM OPERASI Dwi Sunaryono 2 / A	SISTEM OPERASI Sarwosri, S.Kom 2 / B	/	/	/	/
Selasa : 12:50 - 15:20	/	/	/	/	/	/

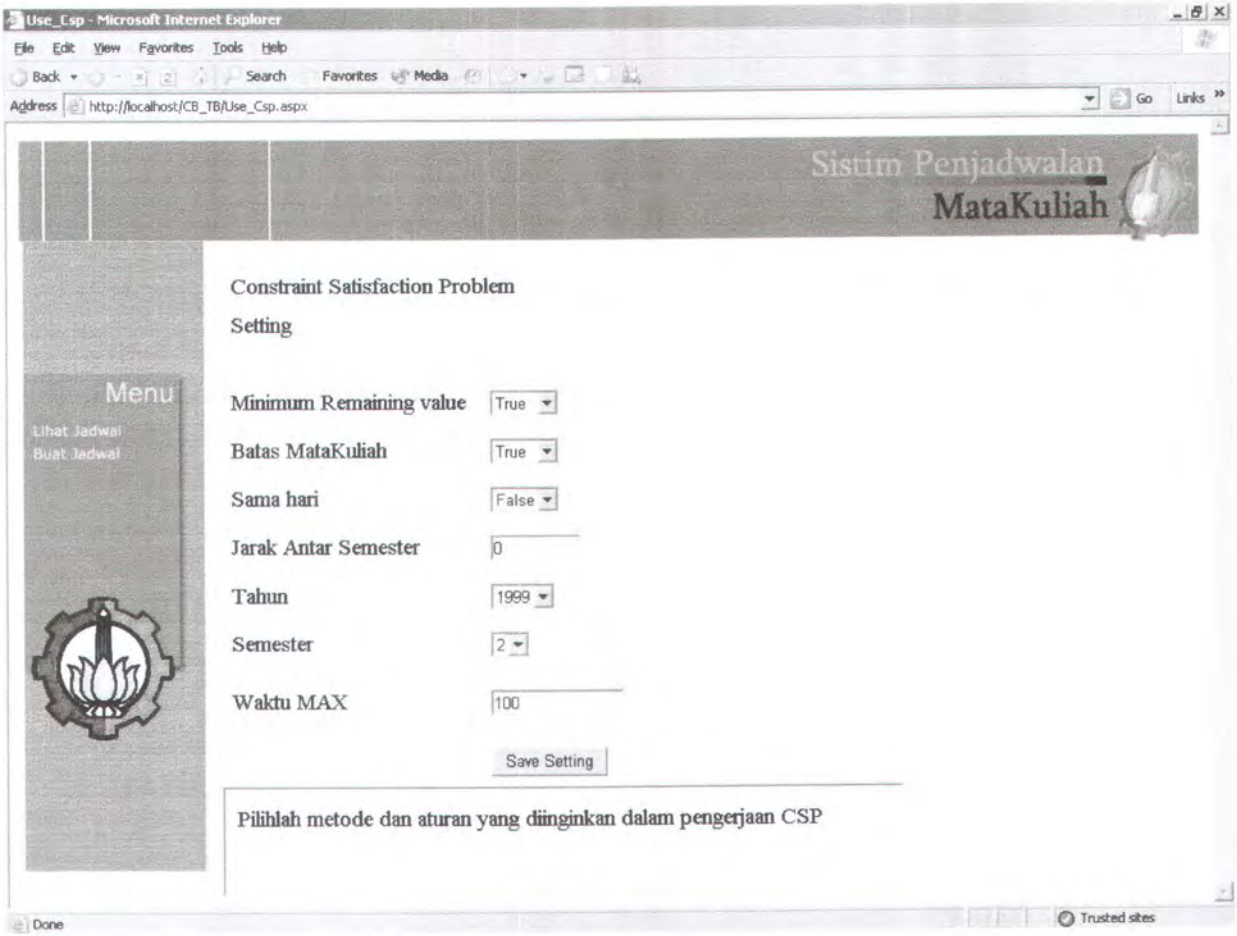
Gambar 3.11 Form halaman depan

Form pada Gambar 3.11 ini adalah halaman utama dari form penjadwalan. Form ini digunakan untuk menampilkan jadwal yang sudah dibuat.



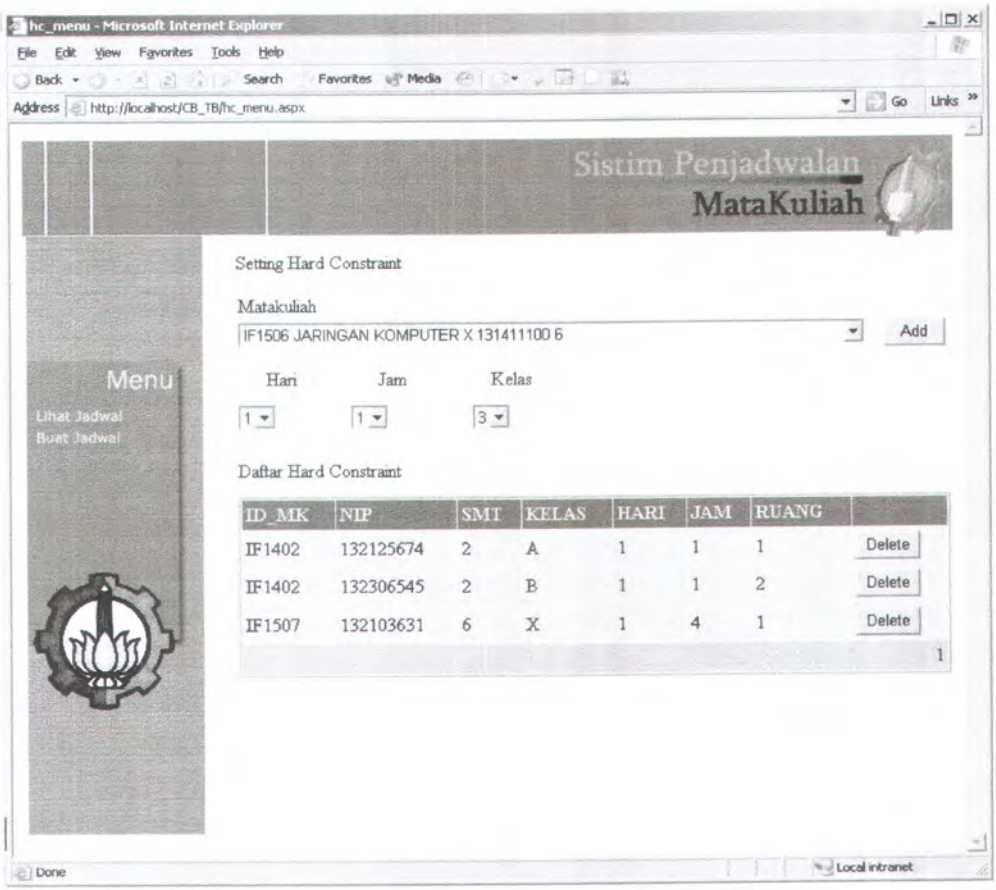
Gambar 3.12 Form setting GA

Form pada Gambar 3.12 ini adalah setting untuk GA. Dalam form ini user dapat mengisi data-data setiap field agar penjadwalan dapat berjalan sesuai kebutuhan user.



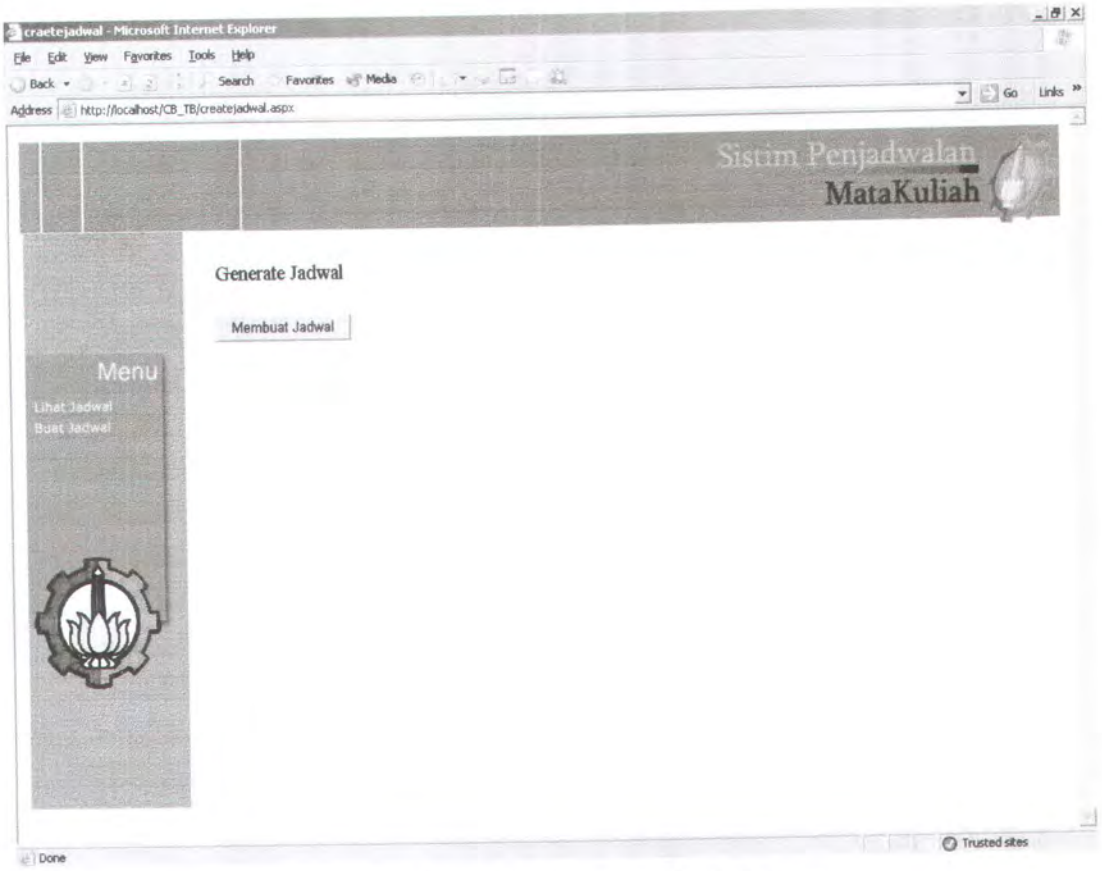
Gambar 3.13 Form setting CSP

Form pada Gambar 3.13 ini adalah setting untuk CSP. Dalam form ini diberikan pilihan untuk user agar dapat memilih metode dan aturan-aturan khusus pada constraint satisfaction problem, yang dibutuhkan agar hasil jadwal sesuai dengan keinginan.



Gambar 3.14 Form pemesanan jadwal.

Form pada Gambar 3.14 ini adalah form untuk melakukan pemesanan jadwal. pada form ini dapat diset matakuliah-matakuliah yang memesan waktu tertentu pada jadwal yang akan dibuat.



Gambar 3.15 Form membuat jadwal.

Form pada Gambar 3.15 ini adalah form untuk menjalankan proses pembuatan jadwal. Pada form ini user dapat memilih untuk membuat jadwal yang berusaha dioptimalisasi dengan bantuan GA atau hanya menggunakan CSP saja.

3.4.3 Implementasi Sistem Penjadwalan

Implementasi proses utama yang terjadi pada Sistem Penjadwalan terdiri atas:

3.4.3.1 Proses pemesanan jadwal

1. Pengambilan data jadwal

Pada aplikasi ini data yang diambil berasal dari database, pembacaan data dilakukan dengan potongan program ini

```
SELECT MT.ID_MK, MT.NIP, MATAKULIAH.SMT, MT.KELAS, MATAKULIAH.MATA_KULIAH,
       MATAKULIAH.JADWAL
FROM
       (SELECT MR.*
       FROM MK_TAWAR MR, SETTING_CSP ST
       WHERE MR.TAHUN=ST.TAHUN AND MR.SEMESTER=ST.SEMESTER AND ST.ID=1) MT
       JOIN MATAKULIAH ON(MT.ID_MK=MATAKULIAH.ID_MK)
WHERE (MATAKULIAH.JADWAL IS NULL OR MATAKULIAH.JADWAL LIKE '%2%')AND
NOT EXISTS
       (
       SELECT HC.ID_MK,HC.NIP,HC.SMT,HC.KELAS FROM HARD_C HC
       WHERE MT.ID_MK=HC.ID_MK AND MT.NIP=HC.NIP AND MT.KELAS=HC.KELAS
       )
```

2. Memvalidasi perubahan yang terjadi

Terdapat dua pemeriksaan yang dilakukan oleh *active database* yang pertama adalah sebelum data ditambahkan ke dalam database dilakukan pemeriksaan apakah data yang diambil dari matakuliah tawar tersebut valid.

```
CREATE OR REPLACE TRIGGER BEFORE_INSERT
BEFORE INSERT
ON HARD_C
REFERENCING NEW AS NEW OLD AS OLD
FOR EACH ROW
DECLARE
          TYPE HC_CONST IS RECORD
          (
               ID_MK     HARD_C.ID_MK%TYPE,
               NIP       HARD_C.NIP%TYPE,
               SMT       HARD_C.SMT%TYPE,
               KELAS     HARD_C.KELAS%TYPE
               );
               TYPE CONST IS TABLE OF HC_CONST INDEX BY BINARY_INTEGER;
               DATA CONST;
               I NUMBER;
               J NUMBER;
               FLAG BOOLEAN := FALSE;
               FLAG2 BOOLEAN := FALSE;
BEGIN
       SELECT ID_MK,NIP,SMT, BULK COLLECT INTO DATA FROM
       (SELECT MR.*
       FROM MK_TAWAR MR, SETTING_CSP ST
       WHERE MR.TAHUN=ST.TAHUN AND MR.SEMESTER=ST.SEMESTER AND ST.ID=1)
       LEFT OUTER JOIN MATAKULIAH USING (ID_MK);

        IF (:NEW.NIP IS NULL) THEN
```

```
            SELECT COUNT(*) INTO J FROM HARD_C WHERE :NEW.ID_MK=HARD_C.ID_MK
            AND :NEW.KELAS = HARD_C.KELAS;
        ELSE
            SELECT COUNT(*) INTO J FROM HARD_C
            WHERE :NEW.ID_MK=HARD_C.ID_MK AND :NEW.KELAS = HARD_C.KELAS AND
            :NEW.NIP=HARD_C.NIP;
        END IF;

        I := DATA.FIRST;
        WHILE (I IS NOT NULL) AND (FLAG = FALSE) LOOP
            IF (:NEW.ID_MK = DATA(I).ID_MK) AND (:NEW.KELAS = DATA(I).KELAS)
                AND (:NEW.NIP IS NULL) THEN
                :NEW.NIP := DATA(I).NIP;
                :NEW.SMT := DATA(I).SMT;
                FLAG := TRUE;
            ELSE
                IF (:NEW.ID_MK = DATA(I).ID_MK) AND (:NEW.KELAS =
                    DATA(I).KELAS) AND (:NEW.NIP = DATA(I).NIP) THEN
                    :NEW.SMT := DATA(I).SMT;
                    FLAG := TRUE;
                END IF;
            END IF;
            I := DATA.NEXT(I);
        END LOOP;

        IF (FLAG = FALSE) THEN
            raise_application_error(-20101, 'DATA TIDAK ADA DALAM KONTRAKULIAH
            TAHAP PADA TAHUN YANG DITENTUKAN');
        END IF;
        IF (J>0) THEN
            raise_application_error(-20101, 'DATA SUDAH ADA DALAM HARD
            CONSTRAINT');
        END IF;
        IF (:NEW.HARI IS NULL) OR (:NEW.JAM IS NULL) OR (:NEW.RUANG IS NULL) THEN
            raise_application_error(-20101, 'HARUS ISI WAKTU, TEMPAT, DAN
            RUANG');
        END IF;
END;
```

Pemeriksaan yang kedua adalah sesudah data ditambahkan ke dalam database dilakukan pemeriksaan apakah ada *constraint* yang dilanggar. Pemeriksaan ini dilakukan menggunakan metode *Constraint Satisfaction*. Dan jika ada data pemesanan jadwal yang melanggar *constraint* maka data tidak dimasukan ke dalam pemesanan.

```
CREATE OR REPLACE TRIGGER AFTER_INSERT
AFTER INSERT
ON HARD_C
REFERENCING NEW AS NEW OLD AS OLD
DECLARE
        VAR MAIN.VARIABLE;
        TFC MAIN.FC;
        HASIL MAIN.SOLUTION;
        I NUMBER;
        TEMP MAIN.ISI_JAD;
        TYPE X IS TABLE OF NUMBER INDEX BY BINARY_INTEGER;
        J X;
BEGIN
        SELECT HCO.NIP,HCO.ID_MK,HCO.SMT,99,HCO.KELAS,99,MK.JADWAL
            BULK COLLECT INTO VAR
```

```
        FROM [illegible] HCO LEFT OUTER JOIN [illegible] MK ON(HCO.ID_MK=MK.ID_MK);
        SELECT SAMA_HARI BULK COLLECT INTO J FROM [illegible];

        TFC := ISI(VAR);
        HC([illegible],VAR,TFC,0,0,0,0);
END;
```

3.4.3.2 Pemrosesan dengan GA

1. Inisialisasi table hasil

Inisialisasi table hasil dilakukan untuk mempersiapkan table yang akan menampung hasil dari pengolahan, setelah selesai inisialisasi di table ini maka *active database* akan berkerja untuk menjalankan Algoritma genetika. Di perlihatkan dengan potongan program ini :

```
CREATE OR REPLACE TRIGGER GIRI.AFT
AFTER DELETE
ON [illegible]
REFERENCING NEW AS NEW OLD AS OLD
DECLARE
          [illegible]          .INDIVIDU;
          START_WK CHAR(5);
          END_WK CHAR(5);
          elapsed_time NUMBER(5) := 0;
          MET NUMBER := 1;
          TAHUN CHAR(4);
          GJL_GNP CHAR(1);
          THARI NUMBER;
          TJAM NUMBER;
          TID NUMBER;
BEGIN
               BERSIHJADF2(1);
               DBMS_OUTPUT.PUT_LINE(THARI||' '||TJAM||'  '||TID);
                SELECT TAHUN INTO TAHUN FROM [illegible];
                SELECT SEMESTER INTO GJL_GNP FROM [illegible];
                SELECT TO_CHAR(SYSDATE,'SSSSS') INTO START_WK FROM sys.dual;
                  (MET,TAHUN,GJL_GNP);
                SELECT TO_CHAR(SYSDATE,'SSSSS') INTO END_WK FROM sys.dual;
                elapsed_time := END_WK - START_WK;
                DBMS_OUTPUT.PUT_LINE('ETA : '||elapsed_time);
       END ;
```

2. Inisialisasi kromosom

Pada aplikasi ini data yang diambil berasal dari database. Pembacaan data dan pembentukan kromosom dapat dilakukan oleh potongan program ini :

```
SELECT MAX, MUTRATE, JMLMUT,MAXCRS, WAKTU_PROS, MIN,TCRS INTO SETT FROM
GIRI.[illegible];
```

```
SELECT MT.NIP,MT.ID_MK,MK.SMT,MT.TAHUN,MT.KELAS,MT.DAYA_TAMPUNG,MK.JADWAL BULK
COLLECT INTO POP(1).DATA
        FROM MK_TAWAR MT,MATAKULIAH MK
        WHERE (MT.TAHUN = TAH) AND (MT.SEMESTER = SEM) AND (MT.ID_MK=MK.ID_MK) AND
(MK.JADWAL IS NULL OR MK.JADWAL =2);
```

3. Pre-Processing

Pre-Processing dilakukan dengan melakukan penghitungan jumlah domain yang dibutuhkan oleh kromosom. Hal yang dilakukan adalah menghitung jumlah domain yang dibutuhkan pada setiap *constraint* yang ada, kemudian pada setiap constraint dihitung apakah domain dapat mencukupi semua kebutuhan. Proses tersebut dapat dilihat dari potongan program berikut :

```
SELECT SMT,SUM(JML) BULK COLLECT INTO INI
FROM
        (
        SELECT ID_MK,SMT,MAX(TOT) AS JML
        FROM
                (
                SELECT DISTINCT MT.ID_MK,NIP,SMT,COUNT(*) AS TOT
                FROM MK_TAWAR MT LEFT OUTER JOIN MATAKULIAH MK
                ON(MK.ID_MK=MT.ID_MK)
                WHERE TAHUN=2006 AND SEMESTER=2 AND (MK.JADWAL!=1 OR
                MK.JADWAL IS NULL) AND MT.KELAS NOT LIKE 'RX%'
                GROUP BY MT.ID_MK,NIP,SMT
                ) ASD
        GROUP BY ID_MK,SMT
        )
GROUP BY SMT;
TAMP := INI;
I := INI.FIRST;
WHILE (I IS NOT NULL) LOOP
        J := TAMP.FIRST;
        WHILE (J IS NOT NULL) LOOP
                IF (TO_NUMBER(INI(I).SMT) >= (TO_NUMBER(TAMP(J).SMT)-2)
                AND TO_NUMBER(INI(I).SMT) <= (TO_NUMBER(TAMP(J).SMT)+2)
                AND I!=J) THEN
                        INI(I).JML := INI(I).JML + TAMP(J).JML;
                END IF;
                J:= TAMP.NEXT(J);
        END LOOP;
        DBMS_OUTPUT.PUT_LINE(INI(I).SMT||' = '||INI(I).JML);
        IF (MAXJAM-(INI(I).JML) <= 0) THEN
           RAISE_APPLICATION_ERROR(-20101, 'TEMPAT JAM REG TIDAK CUKUP ');
        END IF;
        I := INI.NEXT(I);
        END LOOP;
```

```
SELECT SUM(JML) INTO TEMP
FROM
        (
        SELECT ID_MK,SMT,MAX(TOT) AS JML
        FROM
                (
                SELECT DISTINCT MT.ID_MK,NIP,SMT,COUNT(*) AS TOT
```

```
                FROM MK_TAWAR MT LEFT OUTER JOIN MATAKULIAH MK
                ON(MK.ID_MK=MT.ID_MK)
                WHERE TAHUN=2006 AND SEMESTER=2 AND (MK.JADWAL!=1 OR
                MK.JADWAL IS NULL) AND MT.KELAS LIKE 'AXB%'
                GROUP BY MT.ID_MK,NIP,SMT
                ) ASD
        GROUP BY ID_MK,SMT
        );
SELECT COUNT(*) INTO MAXKELAS FROM (SELECT DISTINCT ID_KELAS FROM KELAS);
TEMP := TEMP/2;
SELECT SMT,SUM(JML) BULK COLLECT INTO INI
FROM
        (
        SELECT ID_MK,SMT,MAX(TOT) AS JML
        FROM
                (
                SELECT DISTINCT MT.ID_MK,NIP,SMT,COUNT(*) AS TOT
                FROM MK_TAWAR MT LEFT OUTER JOIN MATAKULIAH MK
                ON(MK.ID_MK=MT.ID_MK)
                WHERE TAHUN=2006 AND SEMESTER=2 AND (MK.JADWAL!=1 OR
                MK.JADWAL IS NULL) AND MT.KELAS LIKE 'AXB%' AND MT.KELAS NOT
                LIKE 'AXB%'
                GROUP BY MT.ID_MK,NIP,SMT
                ) ASD
        GROUP BY ID_MK,SMT
        )
GROUP BY SMT;
TAMP := INI;
I := INI.FIRST;
WHILE (I IS NOT NULL) LOOP
        J := TAMP.FIRST;
        WHILE (J IS NOT NULL) LOOP
                IF (TO_NUMBER(INI(I).SMT) >= (TO_NUMBER(TAMP(J).SMT)-2)
                AND TO_NUMBER(INI(I).SMT) <= (TO_NUMBER(TAMP(J).SMT)+2)
                AND I!=J) THEN
                        INI(I).JML := INI(I).JML + TAMP(J).JML;
                END IF;
                J:= TAMP.NEXT(J);
        END LOOP;
        DBMS_OUTPUT.PUT_LINE(INI(I).SMT||' = '||(INI(I).JML+TEMP));
        IF (MAXJAM-(INI(I).JML) <= 0) THEN
           RAISE_APPLICATION_ERROR(-20101, 'TEMPAT JAM EXT TIDAK CUKUP ');
        END IF;
        I := INI.NEXT(I);
        END LOOP;
```

4. Crossover

Crossover dilakukan dengan melakukan pertukaran pada urutan kromosom tertentu pada dua kromosom yang terlibat. Hal pertama yang dilakukan adalah memilih kromosom-kromosom mana dari populasi yang akan dikenai proses *crossover*, pemilihan tersebut dilakukan dengan cara merandom nilai, kemudian jika nilai yang dirandom masuk dalam probabilitas seleksi, maka kromosom tersebut akan dipilih untuk di *crossover*-kan.

```
CREATE OR REPLACE FUNCTION CROSSOVER(INPOP MAIN_GA.POPULASI, INRATE NUMBER,P1
NUMBER,P2 NUMBER)RETURN MAIN_GA.POPULASI IS
        POPTEMP MAIN_GA.POPULASI;
BEGIN
        POPTEMP := PILIH_PASANGAN(INPOP,INRATE);
        POPTEMP := SILANG(POPTEMP,P1,P2);
        RETURN POPTEMP;
END;
/
```

5. Mutasi

Mutasi dilakukan dengan cara mengubah urutan gen di kromosom. Pertama dilakukan adalah memilih suatu kromosom dengan cara menyeleksinya dengan probabilitas. Pemilihan tersebut dilakukan dengan cara merandom nilai, kemudian jika nilai yang di-*random* masuk dalam probabilitas seleksi, maka kromosom tersebut akan dipilih untuk dilakukan mutasi. Dalam melakukan mutasi ada jumlah berapa banyak suatu gen dirubah urutannya.

```
CREATE OR REPLACE FUNCTION MUTATE(INPOP MAIN_GA.POPULASI, INRATE NUMBER,MAXMUT
NUMBER,INMUT NUMBER)RETURN MAIN_GA.POPULASI IS
        INDIV MAIN_GA.POPULASI := INPOP;
BEGIN
        INDIV := PILIH_IND(INDIV,INRATE,MAXMUT);
        INDIV := RUBAH(INDIV,INMUT);
        RETURN INDIV;
END;
/
```

6. Pembuatan jadwal dengan CSP

Pembuatan jadwal dengan CSP dilakukan untuk mendapatkan suatu jadwal, yang dapat dipakai untuk menentukan kualitas suatu kromosom. Pembuatan jadwal ini dilakukan dengan cara memanggil fungsi yang menjalankan CSP (akan dijelaskan pada bagian CSP).

```
HASIL :=
MULAIGA(MINRV,TAH,SEM,SETT.B_MK,SETT.SAMA_HARI,POP(I).DATA,SETT.JARAK,SETT.BATAS);
```

7. Pemberian Fitness

Pemberian fitness dilakukan dengan cara mengelompokan data dari jadwal yang sudah dibuat ke dalam data rata-rata waktu tunggu suatu semester per hari. Kemudian dikelompokan menjadi rata-rata waktu tunggu per hari. Kemudian diubah menjadi rata-rata waktu tunggu keseluruhan.

```
        IF (INPUT.COUNT != 0) THEN
                I := REDUCE.FIRST;
                WHILE (I IS NOT NULL) LOOP
                        IF (REDUCE(I).JENIS LIKE '%2%') OR NOT(REDUCE(I).JENIS IS
NULL) THEN
                                REDUCE.DELETE(I);
                        END IF;
                        I := REDUCE.NEXT(I);
                END LOOP;

                FOR PASS IN 1..2 LOOP
                        TAMP := REDUCE;
                        I := TAMP.FIRST;
                        WHILE (I IS NOT NULL) LOOP
                                IF (TAMP(I).KELAS LIKE '%X%') AND
NOT(TAMP(I).KELAS LIKE '%XB%') AND (PASS = 1)THEN
                                        TAMP.DELETE(I);
                                ELSIF NOT(TAMP(I).KELAS LIKE '%X%')  AND (PASS =
2) THEN
                                        TAMP.DELETE(I);
                                END IF;
                                I := TAMP.NEXT(I);
                        END LOOP;
                        DATA := DATAK;

                        I := TAMP.FIRST;
                        WHILE (I IS NOT NULL) LOOP
                                THARI := TAMP(I).HARI;
                                TJAM  := TAMP(I).JAM;
                                TSMT  := TAMP(I).SMT;
                                TIDMK := TAMP(I).ID_MK;
                                TSKR  := 0;
                                WHILE (I IS NOT NULL) LOOP
                                        IF (TAMP(I).HARI=THARI) AND
(TAMP(I).SMT=TSMT) AND (TAMP(I).ID_MK=TIDMK) THEN
                                                IF (TSKR < TAMP(I).JAM) THEN
                                                        TSKR := TAMP(I).JAM;
                                                END IF;
                                                TAMP.DELETE(I);
                                        END IF;
                                        I := TAMP.NEXT(I);
                                END LOOP;
                                DATA(THARI)(TSMT)(TSKR) := 0;
                                I := TAMP.FIRST;
                        END LOOP;

                        RATA :=0;
                        BAGIH := 0;
                        TOTALH := 0;
                        I := DATA.FIRST;
                        WHILE (I IS NOT NULL) LOOP
```

```
                        J := DATA(I).FIRST;
                        TOTALS := 0;
                        BAGIS := 0;
                        WHILE (J IS NOT NULL) LOOP
                                K := DATA(I)(J).FIRST;
                                SBLM := 0;
                                BAGI := 0;
                                SKR := 0;
                                TOTAL := 0;
                                JARAK := 0;
                                WHILE (K IS NOT NULL) LOOP
                                        SKR := K;
                                        IF (SBLM != 0) THEN
                                             JARAK := SKR - SBLM;
                                             TOTAL := TOTAL + JARAK;
                                             BAGI := BAGI + 1;

                                        END IF;
                                        SBLM := SKR;
                                        K := DATA(I)(J).NEXT(K);
                                END LOOP;
                                IF BAGI = 0 THEN
                                      BAGI := 1;
                                END IF;
                                TOTALS := TOTALS + TOTAL / BAGI;
                                BAGIS := BAGIS + 1;
                                J := DATA(I).NEXT(J);
                        END LOOP;
                        TOTALH := TOTALH + TOTALS / BAGIS;
                        BAGIH := BAGIH + 1;
                        I := DATA.NEXT(I);
                END LOOP;
                IF (BAGIH != 0) THEN
                      RATA := TOTALH / BAGIH;
                ELSE
                      RATA := 99;
                END IF;
                IF (PASS = 1) THEN
                      RATAR := RATA;
                ELSE
                       RATA := (RATAR+RATA)/2;
                END IF;
        END LOOP;
ELSE
        RATA := 99;
END IF;
RETURN RATA;
```

8. Seleksi

Pada tahap seleksi, dilakukan dengan cara memilihnya berdasarkan probabilitas. Dimana yang memiliki nilai fitness terbesar memiliki kemungkinan terbesar untuk terpilih. Hal pertama yang dilakukan adalah mencari nilai probabilitas suatu kromosom. Dengan cara membagi nilai fitness dengan total seluruh nilai fitness. Kemudian dibandingkan dengan suatu nilai random yang akan menentukan kromosom itu terpilih atau tidak. Hal ini dilakukan sampai

jumlah maksimal suatu kromosom terpenuhi atau kromosom sudah tidak ada yang bisa dipilih lagi. Karena suatu kromosom tidak boleh dipilih lebih dari satu kali.

```
    POP := SORTPOPDESC(POP);

    J := 1;
    B := POP.FIRST;
    IF (B IS NOT NULL) THEN
            MAXII(J) := POP(B);
            J := J + 1;
            POP.DELETE(B);
    END IF;

    TOTAL1 := 0;
    B := POP.FIRST;
    WHILE (B IS NOT NULL) LOOP
            TOTAL1 := POP(B).FIT + TOTAL1;
            B := POP.NEXT(B);
    END LOOP;

    TOTAL := 0;
    B := POP.FIRST;
    WHILE (B IS NOT NULL) LOOP
            TOTAL := (TOTAL1/POP(B).FIT) + TOTAL;
            B := POP.NEXT(B);
    END LOOP;
   B := POP.FIRST;
WHILE (POP.COUNT > 0) AND (J <= MAXI) LOOP
            MAXS := DBMS_RANDOM.VALUE(0,100);
            K := ((TOTAL1/POP(B).FIT)/TOTAL)*100;
            IF (K >= MAXS) THEN
                    MAXI(J) := POP(B);
                    J := J + 1;
                    POP.DELETE(B);

                    TOTAL1 := 0;
                    B := POP.FIRST;
                    WHILE (B IS NOT NULL) LOOP
                            TOTAL1 := POP(B).FIT + TOTAL1;
                            B := POP.NEXT(B);
                    END LOOP;

                    TOTAL := 0;
                    B := POP.FIRST;
                    WHILE (B IS NOT NULL) LOOP
                            TOTAL := (TOTAL1/POP(B).FIT) + TOTAL;
                            B := POP.NEXT(B);
                    END LOOP;
                    B := POP.FIRST;
            ELSE
                     B := POP.NEXT(B);
            END IF;
            IF (B IS NULL) THEN
                    B := POP.FIRST;
            END IF;
    END LOOP;
    RETURN MAXI;
```

3.4.3.3 Pemrosesan dengan CSP

1. Inisialisasi variable

Pada aplikasi ini data yang diambil berasal dari database. Pada CSP ini ada dua macam inisialisasi, yaitu inisialisasi CSP yang dilakukan dalam proses GA atau inisialisasi untuk proses CSP. Yang membedakannya adalah sumber darimana CSP ini dipanggil. Jika dipanggil untuk memenuhi pembuatan jadwal untuk GA, maka disebut dilakukan dalam proses GA. Dan jika dipanggil untuk membuat jadwal hanya dengan CSP saja, maka disebut dilakukan untuk proses CSP.

Pembacaan pembentukan variable untuk proses CSP dapat dilakukan oleh potongan program ini :

```
SELECT MT.NIP,MT.ID_MK,MK.SMT,MT.TAHUN,MT.KELAS,MT.DAYA_TAMPUNG,MK.JADWAL BULK
COLLECT INTO VAR FROM MT,MATAKULIAH MK
        WHERE (MT.TAHUN = TAH) AND (MT.SEMESTER = SEM) AND (MT.ID_MK=MK.ID_MK) AND
(MK.JADWAL IS NULL OR MK.JADWAL =2);
```

Sedangkan pada pembentukan variable untuk proses GA tidak ada proses inisialisasi karena variable di-*passing* dari proses GA.

Untuk pembentukan domainnya dilakukan dengan cara :

```
        TEMPREG := INIT('REG');
        TEMPEXT := INIT('EXT');
        TEMPXR  := INIT('XR');
        C := VAR.FIRST;
        WHILE (C IS NOT NULL) LOOP
                IF (VAR(C).KELAS LIKE '%XR%') THEN
                        TABLEFC(C) := TEMPXR;
                ELSIF (VAR(C).KELAS LIKE '%X%') THEN
                        TABLEFC(C) := TEMPEXT;
                ELSE
                        TABLEFC(C) := TEMPREG;
                END IF;
                C := VAR.NEXT(C);
        END LOOP;
        RETURN TABLEFC;
```

2. Pilih variable yang ingin diisikan

Pada pemilihan variable yang ingin diisikan ini ada dua cara yang bisa dilakukan. Pertama adalah dengan memilih suatu variable yang akan diisikan secara berurutan dari awal variable sampai akhir.

Cara kedua adalah dengan cara menerapkan metode MRV, yaitu memilih yang memiliki kemungkinan gagal paling besar (variable yang memiliki domain paling sedikit) untuk diisi terlebih dahulu. Berikut ini adalah modul untuk pemilihan dengan MRV :

```
   WHILE (I IS NOT NULL) LOOP
                J := INSOL(I).NO;
                IF (J IS NOT NULL) THEN
                        IF TEMPVAR.EXISTS(J) THEN
                                TEMPVAR.DELETE(J);
                        ELSE
                                raise_application_error(-20101, 'DATA SOLUTION DAN
VARIABLE TIDAK SAMA');
                        END IF;
                ELSE
                        raise_application_error(-20101, 'DATA SOLUTION DAN VARIABLE
TIDAK SAMA JUGA');
                END IF;
                I := INSOL.NEXT(I);
   END LOOP;
   I := TEMPVAR.FIRST;
   WHILE (I IS NOT NULL) LOOP
                IF (MINI > INDOM(I).COUNT) THEN
                       MINI := INDOM(I).COUNT;
                       NO := I;
                END IF;
                I := TEMPVAR.NEXT(I);
   END LOOP;
RETURN NO;
```

3. Pilih nilai dari domain

Pada pemilihan nilai dari domain untuk diisi ke variable ada dua cara yang bisa dipakai.

Pertama adalah dengan memilih suatu domain yang masih valid secara berurutan dari awal domain sampai akhir.

Yang kedua adalah jika diinginkan agar setiap matakuliah yang sama namun berbeda kelas harus memiliki waktu yang sama. maka pertama harus dilakukan pemeriksaan apakah suatu jadwal dapat menampung seluruh matakuliah tersebut pada waktu yang bersamaan. Jika bisa maka diambil nilai tersebut sebagai solusi. Jika tidak bisa maka pencarian dianggap gagal. Maka modul yang dipakai adalah :

Modul untuk menghitung total domain yang dibutuhkan :

```
        J:= TEMPVAR.FIRST;
        WHILE (J IS NOT NULL) LOOP
                IF (TEMPVAR(J).ID_MK = ID_MK) THEN
                        IF (KELAS LIKE '%XR%') AND (TEMPVAR(J).KELAS LIKE '%XR%')
THEN
                                                TOTAL := TOTAL + 1;
                        ELSIF (KELAS LIKE '%X%') AND (TEMPVAR(J).KELAS LIKE '%X%')
THEN
                                        TOTAL := TOTAL + 1;
                        ELSE
                                TOTAL := TOTAL + 1;
                        END IF;
                END IF;
                J := TEMPVAR.NEXT(J);
        END LOOP;
        RETURN TOTAL;
```

Modul yang dipakai untuk memilih waktu :

```
        SELECT COUNT(*) INTO MAXJAM FROM (SELECT DISTINCT JAM FROM HARI_JAM);
        SELECT COUNT(*) INTO MAXKLS FROM (SELECT DISTINCT ID_KELAS FROM KELAS);
        MAXHARI := MAXKLS * MAXJAM;

        I := TEMP.FIRST;
        WHILE (I IS NOT NULL) LOOP
          IF JML_TOT.EXISTS(TRUNC((I-1)/MAXKLS)+1) THEN
          JML_TOT(TRUNC((I-1)/MAXKLS)+1) := JML_TOT(TRUNC((I-1)/MAXKLS)+1) + 1;
          ELSE
                JML_TOT(TRUNC((I-1)/MAXKLS)+1) := 1;
          END IF;
          I := TEMP.NEXT(I);
        END LOOP;

        I := JML_TOT.FIRST;
        WHILE (I IS NOT NULL) AND (OK) LOOP
                IF (JML  <= JML_TOT(I)) THEN
                        OK := FALSE;
                        OUTPUT := I;
                END IF;
                I := JML_TOT.NEXT(I);
        END LOOP;

        IF (OK) THEN
                raise_application_error(-20101, 'DATA TIDAK DAPAT DITEMPATKAN,
SOLUSI TIDAK MUNGKIN ADA...');
```

```
        END IF;

        HARI := TRUNC((OUTPUT-1)/MAXJAM)+1;
        JAM := TRUNC((OUTPUT-1)MOD MAXJAM)+1;

        OUTPUT := (HARI-1)*MAXHARI + (((JAM-1)*MAXKLS)+1);
        DBMS_OUTPUT.PUT_LINE('AWAL OUT : '||OUTPUT);
        IF INDOM.EXISTS(OUTPUT) THEN
                DBMS_OUTPUT.PUT_LINE('NGAK PAKE NEXT '||OUTPUT);
                RETURN OUTPUT;
        ELSE
               OUTPUT := TEMP.NEXT(OUTPUT);
                DBMS_OUTPUT.PUT_LINE(' MASUK NEXT HR/JAM :'||HARI||', '||JAM||'
GAK NILAI :'||OUTPUT);
                IF (HARI = TRUNC((OUTPUT-1) / MAXHARI) + 1) AND (JAM =
(TRUNC((OUTPUT-1) / (MAXKLS))MOD MAXJAM) + 1) THEN
                        RETURN OUTPUT;
                ELSE
                        raise_application_error(-20101, ' DATA TIDAK VALID
LAGI.....');
                        END IF;
        END IF;

        RETURN OUTPUT;
```

4. Assign nilai untuk menjadi solusi

Pada proses *assignment* ini dilakukan suatu penyesuaian terhadap domain, berdasarkan domain dan variable yang terpilih. Sehingga suatu variable tidak mungkin melanggar domain. Kemudian setelah itu dilakukan proses pengisian variable dari domain dan menjadi solusi.

Modul untuk penyesuaian domain :

```
SELECT COUNT(*) INTO MAXJAM FROM (SELECT DISTINCT JAM FROM HARI_JAM);
  SELECT COUNT(*) INTO MAXKELAS FROM (SELECT DISTINCT ID_KELAS FROM KELAS);
   MAXHARI := MAXJAM * MAXKELAS;
  HARI := TRUNC((Y-1) / MAXHARI) + 1;
  JAM := (TRUNC((Y-1) / (MAXKELAS))MOD MAXJAM) + 1;
  KELAS := ((Y-1) MOD MAXKELAS) + 1;
  K := ((HARI-1)*MAXHARI)+(((JAM-1)*MAXKELAS)+1);

-- MENANGANI MASALAH WAKTU YG SUDAH DI ASSIGN BIAR TIDAK DIPAKAI VARIABLE LAIN...
FOR I IN 1..INVAR.COUNT LOOP
        IF (TEMP(I).EXISTS(Y)) THEN
                IF ((X != I)) THEN
                        TEMP(I).DELETE(Y);
                ELSE
                        IF ((X = I)) THEN
                                TEMP := HAPUSBERSIH(TEMP,I,Y);
                        END IF;
                END IF;
        END IF;
 END LOOP;


-- MENANGANI MASALAH SATU KODE KULIAH AGAR SATU HARI DAN JAM YANG SAMA..
```

```
IF (SM_HR = TRUE) THEN
        IF (INVAR(X).KELAS LIKE '%K%') THEN
                FOR I IN 1..INVAR.COUNT LOOP
                        IF (INVAR(X).ID_MK = INVAR(I).ID_MK) AND (INVAR(I).KELAS
LIKE '%K%') AND (X != I )THEN
                                DBMS_OUTPUT.put_line('ID MK X '||INVAR(X).ID_MK||'
SAMA DENGAN : '||INVAR(I).ID_MK|| ' ' || X ||' = '|| I);
                                TEMP := SAMAHARI(TEMP,JAM,HARI,I,MAXJAM,MAXKELAS);
                        END IF;
                END LOOP;
        ELSE
                FOR I IN 1..INVAR.COUNT LOOP
                        IF (INVAR(X).ID_MK = INVAR(I).ID_MK) AND NOT(INVAR(I).KELAS
LIKE '%K%') AND (X != I ) THEN
                                DBMS_OUTPUT.put_line('ID MK A '||INVAR(X).ID_MK||' SAMA
DENGAN : '||INVAR(I).ID_MK|| ' ' || X ||' = '|| I);
                                TEMP := SAMAHARI(TEMP,JAM,HARI,I,MAXJAM,MAXKELAS);
                        END IF;
                END LOOP;
        END IF;
END IF;

-- MENANGANI NIP YANG BERADA PADA SATU WAKTU

 FOR I IN 1..INVAR.COUNT LOOP
        J := K;
        IF NOT(TEMP(I).EXISTS(J)) THEN
                J := TEMP(I).NEXT(J);
        END IF;
        WHILE ((J IS NOT NULL) AND (HARI = TRUNC((J-1) / MAXHARI) + 1) AND (JAM =
(TRUNC((J-1) / (MAXJAM+1))MOD MAXJAM) + 1)) LOOP
                IF ((INVAR(I).NIP = INVAR(X).NIP) AND (X != I)) THEN
                        TEMP(I).DELETE(J);
                END IF;
                J := TEMP(I).NEXT(J);
        END LOOP;
END LOOP;

-- MENAGANI MASALAH SEMESTER PADA SATU WAKTU YANG SAMA
 FOR I IN 1..INVAR.COUNT LOOP          -- LOOP UNTUK SETIAP VARIABLE
        J := K;
        IF NOT(TEMP(I).EXISTS(J)) THEN
                J := TEMP(I).NEXT(J);
        END IF;

        WHILE ((J IS NOT NULL) AND (HARI = TRUNC((J-1) / MAXHARI) + 1) AND (JAM =
(TRUNC((J-1) / (MAXJAM+1))MOD MAXJAM) + 1))LOOP
                IF ((INVAR(I).SMT = INVAR(X).SMT) AND (X != I)) THEN
                        IF (NOT(INVAR(I).ID_MK = INVAR(X).ID_MK)) THEN
                                TEMP(I).DELETE(J);
                        END IF;
                END IF;
                J := TEMP(I).NEXT(J);
        END LOOP;
END LOOP;

--MENAGANI MASALAH SEMESTER 7,8 TIDAK BOLEH SAMA DENGAN 0..
FOR I IN 1..INVAR.COUNT LOOP -- LOOP UNTUK SETIAP VARIABLE
        J := K;
        IF NOT(TEMP(I).EXISTS(J)) THEN
                J := TEMP(I).NEXT(J);
        END IF;

        WHILE ((J IS NOT NULL) AND (HARI = TRUNC((J-1) / MAXHARI) + 1) AND (JAM =
(TRUNC((J-1) / (MAXJAM+1))MOD MAXJAM) + 1))LOOP
                IF ((((INVAR(I).SMT ='7' OR INVAR(I).SMT ='8') AND
INVAR(X).SMT='0')) OR (((INVAR(X).SMT ='7' OR INVAR(X).SMT ='8') AND
INVAR(I).SMT='0')) AND (X != I)) THEN
                        IF (NOT(INVAR(I).ID_MK = INVAR(X).ID_MK)) THEN
                                TEMP(I).DELETE(J);
```

```
                    END IF;
                END IF;
                J := TEMP(I).NEXT(J);
        END LOOP;
 END LOOP;

 -- MENANGANI MASALAH JARAK ANTAR SEMESTER.. YANG BERJARAK 1 DARI SEMESTER
SEKARANG.. UNTUK MENGATASI YANG MENGULANG..
 IF (JARAK = TRUE) THEN
        FOR I IN 1..INVAR.COUNT LOOP
                J := K;
                IF NOT(TEMP(I).EXISTS(J)) THEN
                        J := TEMP(I).NEXT(J);
                END IF;
                WHILE ((J IS NOT NULL) AND (HARI = TRUNC((J-1) / MAXHARI) + 1) AND
(JAM = (TRUNC((J-1) / (MAXJAM+1))MOD MAXJAM) + 1))LOOP
                        IF (((INVAR(I).SMT = INVAR(X).SMT+1) OR (INVAR(I).SMT =
INVAR(X).SMT-1) ) AND (X != I)) THEN
                                IF (NOT(INVAR(I).ID_MK = INVAR(X).ID_MK)) THEN
                                        TEMP(I).DELETE(J);
                                END IF;
                        END IF;
                        J := TEMP(I).NEXT(J);
                END LOOP;
        END LOOP;
  END IF;

  RETURN TEMP;
```

Modul untuk assign :

```
        IF (I IS NULL) THEN
                I := 1;
        END IF;
        TEMP(I).NO := NUM;
        TEMP(I).NIP := INISI.NIP;
        TEMP(I).ID_MK := INISI.ID_MK;
        TEMP(I).SMT := INISI.SMT;
        TEMP(I).KELAS := INISI.KELAS;
        TEMP(I).HARI := INJAD.HARI;
        TEMP(I).RUANG := INJAD.RUANG;
        TEMP(I).JAM := INJAD.JAM;
        TEMP(I).JENIS := INISI.JENIS;
        RETURN TEMP;
```

BAB IV

UJI COBA DAN EVALUASI HASIL

Dalam bab ini akan dibahas uji coba pada sistem penjadwalan terhadap data FRS-online. Uji coba dilakukan dalam tiga skenario. Yang pertama mengenai pembuatan jadwal dengan *Constraint Satisfaction Problem*, yang kedua terhadap optimasi waktu tunggu menggunakan algoritma genetika, dan yang ketiga mengenai pencarian nilai setting algoritma genetika.

4.1 Lingkungan Uji Coba

Uji coba terhadap sistem yang telah dibuat dilakukan dengan menggunakan lingkungan dengan spesifikasi sebagai berikut:

Spesifikasi sistem:

- Microsoft Windows server 2003
- Oracle 9.2.0.2.1
- Quest® Software, Toad 7.6
- Microsoft Visual Studio.NET 2003
- ASP.NET

Spesifikasi hardware:

- Athlon XP 1.3 MHz
- RAM 384 MB
- 40 GB Hard disk

4.2 Skenario Uji Coba

Pada bagian uji coba ini diberikan beberapa skenario yang ditujukan untuk mengetahui fungsionalitas dari perangkat lunak yang dibuat. Terdapat 3 skenario yang diujikan, yaitu:

1. Kemampuan *constraint satisfaction* membuat jadwal, Kemampuan *constraint satisfaction* menangani pemesanan jadwal pada waktu dan kelas tertentu.
2. Kemampuan algoritma genetika untuk mendapatkan jadwal yang minimal dengan mengoptimasi waktu tunggu.
3. Mencari kombinasi untuk setting algoritma genetika.

Pada semua uji coba digunakan constraint dasar sebagai berikut :

- Tidak ada satu ruangan yang diisikan dua kali dalam satu waktu yang sama.
- Tidak ada nip yang sama pada hari dan jam yang sama.
- Tidak ada semester yang sama pada waktu yang sama.

4.2.1 Uji Coba Skenario 1

Dengan permintaan memesan kelas sebagai berikut :

Tabel 4.1. Pemesanan matakuliah tahun 2006 semester genap

Id_mk	NIP	SMT	Kelas	Hari	Jam	Ruang
CI1307	130816212	2	A	4	1	1
CI1504	051100002	3	XRA	5	4	1
CI1205	131996151	8	B	3	1	1
CI1422	51100013	6	XA	5	5	1

Dan Constraint Tambahan yang digunakan:

- Membatasi matakuliah pada semester yang sama hanya muncul dua kali pada satu hari.
- Matakuliah dengan kode matakuliah yang sama harus ada pada hari yang sama, kecuali jika dosennya sama.
- Jarak antar semester dengan semester berikutnya pada jam yang sama harus lebih besar dari jarak yang diinputkan.
- Dosen yang sama dalam satu hari hanya mengajar dua kali.

Diberikan matakuliah pada tahun 2006 semester genap dengan data sebagai berikut :

Tabel 4.2. Pemesanan matakuliah tahun 2006 semester genap

ID_MK	KELAS	NIP	NAMA	MATA_KULIAH
CI1205	A	131996151	Joko Lianto B.	ETIKA PROFESIONAL
CI1305	C	132306544	Nunut Priyo J.	STRUKTUR DATA
CI1305	B	131285253	F.X. Arunanto	STRUKTUR DATA
CI1305	A	132125674	Nanik Suciati	STRUKTUR DATA
CI1421	B	51100016	Ahmad Khoirul Bashori	BASIS DATA LANJUT
CI1421	A	11111	Daniel O. Siahaan	BASIS DATA LANJUT
CI1307	B	130816212	Esther Hanaya	MATEMATIKA DISKRIT
CI1412	C	132309748	Ary Mazharuddin Shiddiqi	JARINGAN KOMPUTER
CI1412	A	051100001	Royyana Muslim I	JARINGAN KOMPUTER
CI1412	B	131411100	Muchammad Husni	JARINGAN KOMPUTER
CI1424	A	132163671	Dwi Sunaryono	PEMROGRAMAN API
CI1414	A	132125657	Victor Hariadi	METODE NUMERIK
CI1414	C	132172210	Yudhi Purwananto	METODE NUMERIK
CI1414	B	132296226	Bilqis Amaliah	METODE NUMERIK
CI1424	B	132163671	Dwi Sunaryono	PEMROGRAMAN API
CI1501	XRA	130532048	Handayani Tjandrasa	PERENCANAAN STRATEGIS SUMBER DAYA PERUSAHAAN
CI1502	XRA	132125674	Nanik Suciati	ANALISIS CITRA DAN VISI KOMPUTER
CI1423	XA	132298829	Chastine Fatichah	PENGENALAN POLA
CI1205	XA	131996151	Joko Lianto B.	ETIKA PROFESIONAL
CI1202	A	132306544	Nunut Priyo J.	ORGANISASI KOMPUTER
CI1202	B	132309748	Ary Mazharuddin Shiddiqi	ORGANISASI KOMPUTER
CI1409	A	132306430	Darlis Heru Murti	STATISTIKA UNTUK KOMPUTASI 2
CI1423	A	132085802	Rully Soelaiman	PENGENALAN POLA
CI1202	C	132309748	Ary Mazharuddin Shiddiqi	ORGANISASI KOMPUTER

CI1409	B	132306430	Darlis Heru Murti	STATISTIKA UNTUK KOMPUTASI 2
CI1423	B	132085802	Rully Soelaiman	PENGENALAN POLA
CI1203	B	051100003	Arif Bramantoro	PEMROGRAMAN BERBASIS WEB
CI1203	A	132163671	Dwi Sunaryono	PEMROGRAMAN BERBASIS WEB
CI1203	C	51100016	Ahmad Khoirul Bashori	
CI1422	B	132103631	Siti Rochimah	ANALISIS DAN DESAIN BERORIENTASI OBYEK
CI1422	A	51100013	Isye Ariesanti	ANALISIS DAN DESAIN BERORIENTASI OBYEK
CI1512	XRA	132256272	Waskitho Wibisono	KOMPUTASI CLIENT-SERVER
CI1516	XRA	131285253	F.X. Arunanto	KOMPUTASI PARALEL
CI1421	XA	51100016	Ahmad Khoirul Bashori	BASIS DATA LANJUT
CI1205	B	131996151	Joko Lianto B.	ETIKA PROFESIONAL
CI1409	C	132306430	Darlis Heru Murti	STATISTIKA UNTUK KOMPUTASI 2
CI1422	C	132103631	Siti Rochimah	ANALISIS DAN DESAIN BERORIENTASI OBYEK
CI1410	A	132306543	Imam Kuswardayan	SISTIM INFORMASI
CI1410	B	132309747	Umi Laili Yuhana	SISTIM INFORMASI
CI1410	C	132309747	Umi Laili Yuhana	SISTIM INFORMASI
CI1521	XRA	132048148	Suhadi Lili	ARSITEKTUR PERANGKAT LUNAK
CI1523	XRA	131570363	Riyanarto Sarno	REKAYASA SISTEM
CI1424	XA	132306543	Imam Kuswardayan	PEMROGRAMAN API
CI1307	A	130816212	Esther Hanaya	MATEMATIKA DISKRIT
CI1307	C	132298829	Chastine Fatichah	MATEMATIKA DISKRIT
CI1411	A	051100003	Arif Bramantoro	REKAYASA PERANGKAT LUNAK
CI1411	B	132230429	Fajar Baskoro	REKAYASA PERANGKAT LUNAK
CI1411	C	132230429	Fajar Baskoro	REKAYASA PERANGKAT LUNAK
CI1522	XRA	132048148	Suhadi Lili	POLA-POLA PERANCANGAN
CI1513	XRA	132303065	Wahyu Suadi	DESAIN DAN MANAGEMENT
CI1504	XRA	051100002	Irfan Subakti	INTELIGENSIA MESIN
CI1422	XA	51100013	Isye Ariesanti	ANALISIS DAN DESAIN BERORIENTASI OBYEK

Menghasilkan data jadwal keluaran sebagai berikut :

Hari Jam / Kelas	TC-101	TC-102	TC-103	TC-104	TC-105	TC-106
enin 07.30 - 10.00	ETIKA PROFESIONAL Joko Lianto B E / A	STRUKTUR DATA Nanot Pinyo J 2 / C	STRUKTUR DATA F.X. Arunanto 2 / B	STRUKTUR DATA Nanik Suciati 2 / A	BASIS DATA LANJUT Ahmad Khoirul Basheri 6 / B	BASIS DATA LANJUT Daniel O Siahaan 6 / A
enin 10.10 - 12.40	MATEMATIKA DISKRIT Esther Hanaya 2 / B	JARINGAN KOMPUTER Ary Mazharuddin Shiddiqi 4 / C	JARINGAN KOMPUTER Royyana Muslim I 4 / A	JARINGAN KOMPUTER Muhammad Husni 4 / B	PEMROGRAMAN API Dwi Sunaryono 6 / A	/
enin 12.50 - 15.20	METODE NUMERIK Victor Hariadi 4 / A	METODE NUMERIK Yudie Purwanto 4 / C	METODE NUMERIK Bilqis Amaliah 4 / B	PEMROGRAMAN API Dwi Sunaryono 6 / B	/	/
enin 15.30 - 18.00	PERENCANAAN STRATEGIS SUMBER DAYA PERUSAHAAN Hendrawan Tjandrasa 6 / XRA	ANALISIS CITRA DAN VISI KOMPUTER Nanik Suciati 6 / XRA	/	/	/	/
enin 18.30 - 21.00	PENGENALAN POLA Chastine Fatichah 6 / XA	ETIKA PROFESIONAL Joko Lianto B 6 / XA	/	/	/	/
Selasa 07.30 - 10.00	ORGANISASI KOMPUTER Nanot Pinyo J 2 / A	ORGANISASI KOMPUTER Ary Mazharuddin Shiddiqi 2 / B	STATISTIKA UNTUK KOMPUTASI 2 Darlis Heru Murti 4 / A	PENGENALAN POLA Rully Soelaiman 6 / A	/	/
Selasa 10.10 - 12.40	ORGANISASI KOMPUTER Ary Mazharuddin Shiddiqi 2 / C	STATISTIKA UNTUK KOMPUTASI 2 Darlis Heru Murti 4 / B	PENGENALAN POLA Rully Soelaiman 6 / B	/	/	/
Selasa 12.50 - 15.20	PEMROGRAMAN BERBASIS WEB Arif Bramantoro 4 / B	PEMROGRAMAN BERBASIS WEB Dwi Sunaryono 4 / A	PEMROGRAMAN BERBASIS WEB Ahmad Khoirul Basheri 4 / C	ANALISIS DAN DESAIN BERORIENTASI OBYEK Siti Rochimah 6 / B	ANALISIS DAN DESAIN BERORIENTASI OBYEK Inye Arseanti 6 / A	/
Selasa 15.30 - 18.00	KOMPUTASI CLIENT-SERVER Waskitho Wibisono 6 / XRA	KOMPUTASI PARALEL F.X. Arunanto 7 / XRA	/	/	/	/
Selasa 18.30 - 21.00	BASIS DATA LANJUT Ahmad Khoirul Basheri 6 / XA	/	/	/	/	/
Rabu 07.30 - 10.00	ETIKA PROFESIONAL Joko Lianto B E / B	STATISTIKA UNTUK KOMPUTASI 2 Darlis Heru Murti 4 / C	ANALISIS DAN DESAIN BERORIENTASI OBYEK Siti Rochimah 6 / C	/	/	/
Rabu 10.10 - 12.40	SISTIM INFORMASI Imam Kuswardayan 4 / A	SISTIM INFORMASI Umi Laili Yuhana 4 / B	/	/	/	/
Rabu 12.50 - 15.20	SISTIM INFORMASI Umi Laili Yuhana 4 / C	/	/	/	/	/
Rabu 15.30 - 18.00	ARSITEKTUR PERANGKAT LUNAK Suhadi Lili 6 / XRA	REKAYASA SISTEM Riyanarto Sarno 6 / XRA	/	/	/	/
Rabu 18.30 - 21.00	PEMROGRAMAN API Imam Kuswardayan 6 / XA	/	/	/	/	/
Kamis 07.30 - 10.00	MATEMATIKA DISKRIT Esther Hanaya 2 / A	MATEMATIKA DISKRIT Chastine Fatichah 2 / C	REKAYASA PERANGKAT LUNAK Arif Bramantoro 4 / A	REKAYASA PERANGKAT LUNAK Fajar Baskoro 4 / B	/	/
Kamis 10.10 - 12.40	REKAYASA PERANGKAT LUNAK Fajar Baskoro 4 / C	/	/	/	/	/
Kamis 12.50 - 15.20	/	/	/	/	/	/
Kamis 15.30 - 18.00	POLA-POLA PERANCANGAN Suhadi Lili 6 / XRA	DESAIN DAN MANAGEMENT Wahyu Susih 6 / XRA	/	/	/	/
Kamis 18.30 - 21.00	/	/	/	/	/	/
Jumat 07.30 - 10.00	/	/	/	/	/	/
Jumat 10.10 - 12.40	/	/	/	/	/	/
Jumat 12.50 - 15.20	/	/	/	/	/	/
Jumat 15.30 - 18.00	INTELIGENSIA MESIN Irfan Subakti 7 / XRA	/	/	/	/	/
Jumat 18.30 - 21.00	ANALISIS DAN DESAIN BERORIENTASI OBYEK Inye Arseanti 6 / XA	/	/	/	/	/

Gambar 4.1 Solusi tahun 2006 dan semester genap

Tabel 4.3. Tabel solusi tahun 2006 semester genap

ID_MK	KELAS	NDOSEN	NMK	HARI	JAM	ID_KELAS
CI1205	A	Joko Lianto B.	ETIKA PROFESIONAL	1	1	1
CI1305	C	Nunut Priyo J.	STRUKTUR DATA	1	1	2
CI1305	B	F.X. Arunanto	STRUKTUR DATA	1	1	3
CI1305	A	Nanik Suciati	STRUKTUR DATA	1	1	4
CI1421	B	Ahmad Khoirul Bashori	BASIS DATA LANJUT	1	1	5
CI1421	A	Daniel O. Siahaan	BASIS DATA LANJUT	1	1	6
CI1307	B	Esther Hanaya	MATEMATIKA DISKRIT	1	2	1
CI1412	C	Ary Mazharuddin Shiddiqi	JARINGAN KOMPUTER	1	2	2
CI1412	A	Royyana Muslim I	JARINGAN KOMPUTER	1	2	3
CI1412	B	Muchammad Husni	JARINGAN KOMPUTER	1	2	4
CI1424	A	Dwi Sunaryono	PEMROGRAMAN API	1	2	5
CI1414	A	Victor Hariadi	METODE NUMERIK	1	3	1
CI1414	C	Yudhi Purwananto	METODE NUMERIK	1	3	2
CI1414	B	Bilqis Amaliah	METODE NUMERIK	1	3	3
CI1424	B	Dwi Sunaryono	PEMROGRAMAN API	1	3	4
CI1501	XRA	Handayani Tjandrasa	PERENCANAAN STRATEGIS SUMBER DAYA PERUSAHAAN	1	4	1
CI1502	XRA	Nanik Suciati	ANALISIS CITRA DAN VISI KOMPUTER	1	4	2
CI1423	XA	Chastine Fatichah	PENGENALAN POLA	1	5	1
CI1205	XA	Joko Lianto B.	ETIKA PROFESIONAL	1	5	2
CI1202	A	Nunut Priyo J.	ORGANISASI KOMPUTER	2	1	1
CI1202	B	Ary Mazharuddin Shiddiqi	ORGANISASI KOMPUTER	2	1	2
CI1409	A	Darlis Heru Murti	STATISTIKA UNTUK KOMPUTASI 2	2	1	3
CI1423	A	Rully Soelaiman	PENGENALAN POLA	2	1	4
CI1202	C	Ary Mazharuddin Shiddiqi	ORGANISASI KOMPUTER	2	2	1
CI1409	B	Darlis Heru Murti	STATISTIKA UNTUK KOMPUTASI 2	2	2	2
CI1423	B	Rully Soelaiman	PENGENALAN POLA	2	2	3
CI1203	B	Arif Bramantoro	PEMROGRAMAN BERBASIS WEB	2	3	1
CI1203	A	Dwi Sunaryono	PEMROGRAMAN BERBASIS WEB	2	3	2
CI1203	C	Ahmad Khoirul Bashori	PEMROGRAMAN BERBASIS WEB	2	3	3
CI1422	B	Siti Rochimah	ANALISIS DAN DESAIN BERORIENTASI OBYEK	2	3	4
CI1422	A	Isye Ariesanti	ANALISIS DAN DESAIN BERORIENTASI OBYEK	2	3	5
CI1512	XRA	Waskitho Wibisono	KOMPUTASI CLIENT-SERVER	2	4	1
CI1516	XRA	F.X. Arunanto	KOMPUTASI PARALEL	2	4	2
CI1421	XA	Ahmad Khoirul	BASIS DATA LANJUT	2	5	1

		Bashori				
CI1205	B	Joko Lianto B.	ETIKA PROFESIONAL	3	1	1
CI1409	C	Darlis Heru Murti	STATISTIKA UNTUK KOMPUTASI 2	3	1	2
CI1422	C	Siti Rochimah	ANALISIS DAN DESAIN BERORIENTASI OBYEK	3	1	3
CI1410	A	Imam Kuswardayan	SISTIM INFORMASI	3	2	1
CI1410	B	Umi Laili Yuhana	SISTIM INFORMASI	3	2	2
CI1410	C	Umi Laili Yuhana	SISTIM INFORMASI	3	3	1
CI1521	XRA	Suhadi Lili	ARSITEKTUR PERANGKAT LUNAK	3	4	1
CI1523	XRA	Riyanarto Sarno	REKAYASA SISTEM	3	4	2
CI1424	XA	Imam Kuswardayan	PEMROGRAMAN API	3	5	1
CI1307	A	Esther Hanaya	MATEMATIKA DISKRIT	4	1	1
CI1307	C	Chastine Fatichah	MATEMATIKA DISKRIT	4	1	2
CI1411	A	Arif Bramantoro	REKAYASA PERANGKAT LUNAK	4	1	3
CI1411	B	Fajar Baskoro	REKAYASA PERANGKAT LUNAK	4	1	4
CI1411	C	Fajar Baskoro	REKAYASA PERANGKAT LUNAK	4	2	1
CI1522	XRA	Suhadi Lili	POLA-POLA PERANCANGAN	4	4	1
CI1513	XRA	Wahyu Suadi	DESAIN DAN MANAGEMENT	4	4	2
CI1504	XRA	Irfan Subakti	INTELIGENSIA MESIN	5	4	1
CI1422	XA	Isye Ariesanti	ANALISIS DAN DESAIN BERORIENTASI OBYEK	5	5	1

Dari data yang diberikan diatas terlihat bahwa data input tahun 2006 semester genap berhasil dibuat jadwalnya. Dari hasil yang terbentuk terbukti bahwa *constraint-constraint* dasar yang digunakan dalam percobaan ini semuanya dapat terpenuhi.

Hasil dari percobaan ini menunjukan bahwa pemesanan jadwal dapat terpenuhi sesuai yang diharapkan, dan hasil dari pengolahan jadwal tetap dapat mempertahankan constraint agar sesuai aturan yang ditentukan.

Hasil dari penjadwalan diatas menunjukan bahwa pemesanan tempat dan constraint tambahan juga dapat terpenuhi dengan baik. Dari contoh jadwal yang terbentuk dengan sesuai aturan constraint maka dapat disimpulkan bahwa jadwal

sudah terbentuk dengan baik. Dan semua constraint dalam percobaan ini tidak ada yang terlanggar.

4.2.2 Uji Coba Skenario 2

Dari data tahun 2006 semester genap dilakukan percobaan dengan melihat nilai fitness yang dihasilkan pada setiap generasi:

Dengan setting algoritma genetika sebagai berikut :

- Maksimum populasi : 10
- Kemungkinan terjadinya mutasi : 20%
- Banyaknya terjadinya mutasi : 20%
- Kemungkinan terjadinya crossover : 30%
- Jumlah data yang disilangkan : 50%
- Waktu maksimal pelaksanaan algoritma genetika : 300 detik
- Nilai fitness yang dicari : 0 (berarti dilakukan sampai waktunya habis).

Menghasilkan fitness sebagai berikut :

Tabel 4.4. Statistik tiap generasi

GENERASI	FITNESS TERBAIK	FITNESS TERBURUK	FITNESS RATA-RATA
1	0,501190476	0,501190476	0,501190476
2	0,501190476	0,501190476	0,501190476
3	0,501190476	0,501190476	0,501190476
4	0,501190476	0,501190476	0,501190476
5	0,501190476	0,501190476	0,501190476
6	0,501190476	0,601190476	0,53452381
7	0,501190476	0,601190476	0,53452381
8	0,476190476	0,601190476	0,528452381
9	0,476190476	0,601190476	0,513452381
10	0,43452381	0,601190476	0,494285714
11	0,419642857	0,648809524	0,490892857
12	0,392857143	0,648809524	0,485059524
13	0,392857143	0,501190476	0,448035714
14	0,392857143	0,601190476	0,455357143
15	0,392857143	0,601190476	0,44702381
16	0,392857143	0,476190476	0,426190476
17	0,392857143	0,476190476	0,426190476
18	0,392857143	0,476190476	0,417857143
19	0,392857143	0,392857143	0,392857143

Dan constraint tambahan sebagai berikut :

- Membatasi matakuliah pada semester yang sama hanya muncul dua kali pada satu hari.
- Matakuliah dengan kode matakuliah yang sama harus ada pada hari yang sama
- Jarak antar semester dengan semester berikutnya pada jam yang sama harus lebih besar dari jarak yang diinputkan.

Dengan setting algoritma genetika sebagai berikut :

- Maksimum populasi : 10
- Kemungkinan terjadinya mutasi : 20%
- Banyaknya terjadinya mutasi : 20%
- Kemungkinan terjadinya crossover : 30%
- Jumlah data yang disilangkan : 50%
- Waktu maksimal pelaksanaan algoritma genetika : 300 detik
- Nilai fitness yang dicari : 0 (berarti dilakukan sampai waktunya habis).

Tabel 4.5. Statistik tiap generasi

GENERASI	FITNESS TERBAIK	FITNESS TERBURUK	FITNESS RATA-RATA
1	0,375	0,375	0,375
2	0,375	0,375	0,375
3	0,375	0,375	0,375
4	0,375	0,375	0,375
5	0,375	0,375	0,375
6	0,375	0,375	0,375
7	0,375	0,375	0,375
8	0,375	0,375	0,375
9	0,375	0,375	0,375
10	0,375	0,375	0,375
11	0,375	0,402777778	0,388888889
12	0,361111111	0,402777778	0,383333333
13	0,208333333	0,402777778	0,362847222
14	0,208333333	0,516666667	0,383611111
15	0,208333333	0,572222222	0,391944444
16	0,208333333	0,572222222	0,391944444
17	0,208333333	0,516666667	0,363333333
18	0,208333333	0,516666667	0,360555556

19	0,208333333	0,405555556	0,346666667
20	0,208333333	0,405555556	0,324722222
21	0,180555556	0,488888889	0,296944444
22	0,180555556	0,488888889	0,297777778

Dari percobaan diatas terlihat dalam tabel 4.4 dan tabel 4.5 bahwa waktu tunggu mahasiswa semakin lama semakin sedikit dengan ini menunjukan optimasi berjalan sesuai harapan.

4.2.3 Uji Coba Skenario 3

Dari data tahun 2006 dan semester genap dilakukan evaluasi dengan melihat nilai fitness yang dihasilkan melalui percobaan. Disimpulkan bahwa setting yang sebaliknya dipakai agar mendapatkan hasil yang cukup baik :

Setting algoritma genetika sebagai berikut :

- Maksimum populasi : 20
- Kemungkinan terjadinya mutasi : 45%
- Banyaknya terjadinya mutasi : 40%
- Kemungkinan terjadinya crossover : 50%
- Jumlah data yang disilangkan : 50%

Nilai Uji coba di atas didapatkan dengan melakukan 150 percobaan. Percobaan dilakukan dengan cara merubah nilai setting sedikit demi sedikit dan mengevaluasi nilainya, sehingga di dapatkan nilai setting yang dapat menunjukan rata-rata hasil yang baik.

Walaupun nilai-nilai hasil uji coba diatas ini semakin tidak efektif dengan semakin banyaknya aturan tambahan Constraint Satisfaction yang digunakan, hal

ini dikarenakan dengan semakin banyaknya aturan yang digunakan akan semakin mengurangi kombinasi jadwal yang mungkin terjadi. Sehingga perubahan didalam kromosom algoritma genetika akan semakin tidak efektif.

4.3 Evaluasi Hasil Uji Coba

Dari uji coba yang dilakukan terhadap perangkat lunak yang dibuat, maka dapat dilihat bahwa perangkat lunak telah berkerja dengan baik dan dapat menjalankan semua fungsi sesuai dengan yang diharapkan.

Dari uji coba skenario pertama terlihat bahwa perangkat lunak dapat membuat jadwal yang tidak memiliki jadwal yang bentrok, sehingga memenuhi semua constraint yang ditujukan dalam pembuatan jadwal pada skenario ini. jadwal juga dapat memenuhi semua pemesanan yang dilakukan dan masih tidak memiliki jadwal yang bentrok, sehingga memenuhi constraint dan pemesanan yang ditujukan dalam pembuatan skenario ini. sehingga tujuan dari penjadwalan dengan constraint satisfaction sudah dapat terpenuhi.

Dari uji coba skenario kedua terlihat bahwa algoritma genetika dapat melakukan optimasi pada waktu tunggu dan mencari nilai waktu yang memiliki waktu tunggu paling sedikit.

Melalui Uji coba skenario ketiga telah dihasilkan nilai kombinasi setting algoritma genetika yang dianggap baik. Hal ini disimpulkan melalui 150 percobaan.

BAB V
PENUTUP

5.1 Kesimpulan

Berdasarkan analisis hasil uji coba perangkat lunak maka dibuat kesimpulan sebagai berikut :

1. Melalui percobaan skenario 1 dengan mengolah jadwal pada tahun 2006 semester genap, Perangkat lunak Sistem Penjadwalan mampu mengolah data matakuliah tawar dan menghasilkan jadwal. Tanpa ada constraint yang terlanggar. Aplikasi juga dapat menangani pemesanan jadwal pada waktu tertentu. Adanya perangkat lunak dapat membantu pembuatan jadwal, dalam hal membuat jadwal yang memiliki aturan-aturan seperti constraint yang dapat digunakan dalam tugas akhir ini.
2. Melalui melakukan percobaan skenario 2 dengan melihat statistik yang dihasilkan oleh algoritma genetika pada matakuliah tawar tahun 2006 semester genap Aplikasi dapat melakukan Optimasi dalam hal mencari waktu tunggu yang minimal.
3. Nilai setting algoritma genetika yang di tunjukan dalam skenario 3, dianjurkan digunakan untuk mencari solusi. karena setelah melalui percobaan, diketahui nilai-nilai tersebut dapat menghasilkan hasil yang memiliki rata-rata nilai lebih optimal.

5.2 Saran

Berdasarkan hasil evaluasi yang dilakukan terhadap sistem, ada beberapa saran yang perlu dipertimbangkan dalam pengembangan aplikasi ini, yaitu :

1. Aplikasi dapat dikembangkan agar dapat memenuhi constraint-constraint lain yang tidak terdapat disini.
2. Aplikasi *Constraint Satisfaction Problem* dapat dikembangkan dengan mencoba menggunakan metode algoritma selain yang digunakan didalam Tugas Akhir ini. Agar memiliki waktu proses yang lebih cepat.

DAFTAR PUSTAKA

[AHO87]	Aho, Alfred v., John E. Hopcroft, Jeffrey D. Ullman, *"Data Structure and Algorithm"*, Addison-Wesley, 1987.
[HOL]	Holland, John. *"Genetic Algorithm"*, http://www.econ.iastate.edu/tesfatsi/holland.GAIntro.htm
[KAR98]	Karr, Charles L., L. Michael Freeman, *"Industrial Application of Genetic Algorithm"*, CRC Press LLC, 1998.
[KTI]	http://kti.ms.mff.cuni.cz/%7Ebartak/constraints/index.htm
[KUS05]	Kusumadewi Sri, Hari Purnomo, *"Penyelesaian masalah Optimasi dengan Teknik-teknik Heuristik"*, Graha Ilmu, 2005.
[LEV03]	Levitin Anany, *"Introduction to the design & Analysis of Algorithm"*, Addison-Wesley, 2003.
[MAD03]	Madsen Jeppe Nejsum, *"Methods for Interactive Constraint Satisfaction"*, Department of Computer Science University of Copenhagen, February 2003.
[MEL99]	Melanie Mitchell, *"An Introduction to Genetic Algorithms"*, Bradford Book The MIT Press, 1999.
[RUS03]	Russell Stuart J. and Peter Norvig, *Artificial Intelligence A Modern Approach Second Edition*, Prentice Hall, 2003.
[SIL97]	Silberschats Abraham, Henry F. Korth, S. sudarshan, *"Database System Concepts Third Edition"*, McGraw-Hill, 1997.
[TSA99]	Tsang Edward, *"A Glimpse of Constraint Satisfaction"*, Artificial Intelligence Review 13: 215–227, 1999.
[WIKI]	http://en.wikipedia.org/wiki/Combinatorial_search, http://en.wikipedia.org/wiki/Heuristic_%28computer_science%29, http://en.wikipedia.org/wiki/Constraint_satisfaction
[WIN92]	Winston, *"Artificial Intelligence 3rd Edition"*, Addison Wesley, 1992.