RANCANG BANGUN SISTEM PENJADWALAN PERKULIAHAN DAN UJIAN AKHIR SEMESTER DENGAN PENDEKATAN ALGORITMA GENETIKA

Tesis
untuk memenuhi sebagian persyaratan
mencapai derajat Sarjana S-2 Program Studi
Magister Sistem Informasi

Sam'ani
24010410400046

PROGRAM PASCASRAJANA
UNIVERSITAS DIPONEGORO
SEMARANG
2012

ABSTRAK

Penjadwalan perkuliahan dan ujian akhir semester pada suatu perguruan tinggi adalah kegiatan rutin tiap semester dan merupakan suatu proses untuk menerapkan *event* yang berisi komponen mata kuliah dan kelas pada *time slot* yang berisi komponen waktu dan ruang. Permasalahan yang sering terjadi dalam kegiatan penjadwalan adalah terjadinya bentrok antara jadwal yang satu dengan yang lain. Selain itu adanya permintaan waktu larangan dosen untuk mengajar. Salah satu metode untuk menyelesaikan permasalahan tersebut dengan menggunakan algoritma genetika yang bekerja melalui seleksi alam dan genetika. Terdapat 8 (delapan) *prosedure* algoritma genetika untuk penyelesaian permasalahan pada penelitian ini. *Prosedure* teknik pengkodean menggunakan *string bit / varchar*, populasi awal dan kromosom secara acak (*random*), fungsi *fitness* untuk meminimalkan jumlah bentrok antar jadwal, metode seleksi *roulette-wheel*, pindah silang satu titik potong (*one-point crossover*), mutasi pengkodean nilai, *elitisme* dan kondisi selesai bila iterasi maksimum telah tercapai. Data yang digunakan adalah data perkuliahan semester gasal dan genap tahun akademik 2010/2011 program studi Diploma III Manajemen Informatika STMIK Palangkaraya. Hasil *output* dari sistem berupa susunan penjadwalan perkuliahan dan ujian akhir semester dalam format *file Microsoft Excel*. Dari 3 (tiga) kali pengujian terhadap data yang dilakukan terhadap 5 – 10 generasi dan populasi serta probabilitas pindah silang dan mutasi yang berbeda, didapatkan hasil terbaik dengan semua nilai *fitness* tiap generasi bernilai 1 dan waktu tercepat adalah pada jumlah generasi 5, populasi 5, probabilitas pindah silang 25% dan mutasi 2%.

Kata Kunci : Sistem Penjadwalan, Algoritma Genetika

ABSTRACT

Timetabling for lecture and final examination on a university is a routine activity that happened every semester and a process to apply event that consisted of lecturing and class components on a time slot that consisted of time and space components. Problems that often occurred on timetabling is a crash between one timetabling with another. In addition there are request time prohibition lecturer to teach. A method to solve that problem is by using genetic algorithm that worked through natural selection and genetics. There are 8 (eight) genetic algorithm procedures for solving problems in this research. Encoding techniques procedure using bit string/varchar, initial population and chromosomes randomly, fitness function to minimize crash between one timetabling with another, roulette-wheel selection method, one-point crossover, encoding the value of mutation, elitism and condition of the iteration is complete when the maximum has been reached. The data used is lecture of data odd and even semesters the bachelor department of Information Management STMIK Palangkaraya for the year 2010/2011. The output of the system is the arrangement of timetabling for lecture and final examination in a Microsoft Excel file format. From 3 (three) data tests that had been done on five to ten generation and population and also probability of cross over and different mutation, best result was acquired with fitness score of every generation is one and the fastest time was on sum of generation of five, population of five, cross over probability of twenty five percent and mutation of two percent.

Keywords: Timetabling system, Genetic algorithm

BAB I
PENDAHULUAN

1.1 Latar Belakang

Dalam sistem akademik perguruan tinggi, penjadwalan merupakan pekerjaan rutin yang dilakukan setiap semester. Ada dua penjadwalan yang sering dijumpai pada perguruan tinggi yaitu penjadwalan perkuliahan dan ujian, baik teori maupun praktikum.

Proses penjadwalan adalah suatu proses untuk menerapkan *event* yang berisi komponen mata kuliah, dosen, kelas dan semester pada *time slot* yang berisi komponen waktu dan ruang. Jika menggunakan sistem manual maka masalah ini membutuhkan waktu proses yang cukup lama untuk pencarian solusinya, terlebih lagi bila ukuran permasalahan semakin besar dengan bertambahnya jumlah komponen dan tetapan atau syarat yang ditentukan oleh institusi tempat jadwal tersebut di gunakan.

Selama proses, banyak aspek yang harus dipertimbangkan untuk memperoleh jadwal kuliah dan ujian yang optimal. Oleh karena itu perlu ditetapkan suatu batasan yang menjadi acuan dalam proses penyusunan jadwal kuliah dan ujian.

Berbagai aspek yang berkaitan dalam penjadwalan kuliah dan ujian tersebut dan harus dilibatkan dalam pertimbangan diantaranya :

1) Adanya permintaan dosen yang bersangkutan tidak bisa mengajar pada waktu tertentu
2) Tidak boleh adanya jadwal kuliah dan ujian yang saling bentrok antar dosen, kelas, ruang ataupun waktu perkuliahan

Sistem penjadwalan kuliah dan ujian di beberapa institusi perguruan tinggi sampai saat ini masih dilakukan secara manual, yaitu dengan pencarian blok-blok atau kolom-kolom mana saja yang masih kosong, kemudian menempatkan jadwal pada blok atau kolom tersebut. Jadwal yang dihasilkan dengan cara seperti ini memerlukan waktu yang cukup lama dan cenderung mengabaikan berbagai aspek tersebut. Sehingga jadwal kuliah dan ujian yang sudah dibuat seringkali perlu

dilakukan perbaikan lagi. Oleh karena itu perlu dikembangkan suatu sistem penjadwalan kuliah dan ujian yang dapat mengakomodasi berbagai aspek yang menjadi pertimbangan diatas.

Ada beberapa metode dan algoritma yang sering digunakan dalam menyelesaikan masalah penjadwalan baik perkuliahan maupun ujian akhir semester, yang masing-masing memiliki keunggulan. Salah satu metode dan algoritma tersebut adalah Algoritma Genetika yang akan diterapkan pada penelitian ini.

Algoritma Genetika telah banyak diaplikasikan untuk penyelesaian masalah dan permodelan dalam bidang teknologi, bisnis, dan *entertainment*, seperti optimasi penjadwalan, pemrograman otomatis, *machine learning*, model ekonomi, model sistem imunisasi, model ekologis, interkasi antara evolusi dan belajar (Suyanto, 2005).

Cukup banyak penelitian yang mendukung kehandalan algoritma genetika untuk penyelesaian masalah penjadwalan, seperti pada masalah penjadwalan ujian akhir semester (Arogundade dkk, 2010) dan penjadwalan perkuliahan (Jain dkk, 2010).

Dengan menggunakan Algoritma Genetika dibuat sebuah sistem penjadwalan perkuliahan sekaligus ujian akhir semester yang optimal dengan memperhatikan berbagai aspek yang menjadi pertimbangan dan memiliki waktu proses yang lebih cepat dibanding manual.

1.2 Perumusan Masalah

Perumusan masalah dalam penelitian ini adalah bagaimana menyelesaikan masalah bentrokan yang sering terjadi pada sistem penjadwalan perkuliahan dan ujian akhir semester suatu institusi pendidikan dan disertai dengan adanya permintaan waktu larangan mengajar dosen dengan menggunakan pendekatan algoritma genetika.

1.3 Batasan Masalah

Adapun batasan ruang lingkup dari penelitian yang akan dibahas ini adalah sebagai berikut :

1) Masalah yang diteliti adalah yang berhubungan dengan sistem penjadwalan perkuliahan dan ujian akhir semester satu jurusan pada institusi perguruan tinggi.
2) Penjadwalan dibatasi hanya untuk mata kuliah teori dan praktikum tanpa kerja praktek dan tugas akhir.
3) Jumlah mahasiswa dalam satu kelas lebih kecil atau sama dengan jumlah kapasitas daya tampung ruang atau laboraturium perkuliahan.
4) Pemecahan permasalahannya dengan menggunakan pendekatan algoritma genetika.
5) Membangun aplikasi sistem penjadwalan perkuliahan dan ujian akhir semester.
6) Sistem yang dibuat mengambil studi kasus penjadwalan perkuliahan dan ujian akhir semester ganjil dan genap tahun 2010/2011 pada Jurusan Diploma III Manajemen Informatika STMIK Palangkaraya.

1.4 Keaslian Penelitian

Beberapa penelitian tentang sistem penjadwalan perkuliahan yang telah dilakukan sebelumnya antara lain :

1) Sistem penjadwalan perkuliahan dengan pendekatan algoritma genetika telah diterapkan pada Universitas Devi Ahilya, Indore . Sistem penjadwalan yang dibuat terdiri dari 4 semester yaitu semester 1 sampai dengan 4 dan tidak mencantumkan waktu larangan dosen mengajar. Aplikasi dibangun dengan menggunakan bahasa pemrograman C# dan *SQL Server 2000* sebagai databasenya (Jain dkk, 2010).
2) Demikian juga sistem penjadwalan ujian akhir semester menggunakan pendekatan algoritma genetika dengan jumlah mahasiswa 8000 orang dan 437 jadwal yang dilaksanakan selama 22 hari untuk semester

pertama telah digunakan pada fakultas pertanian Universitas Abeokuta Nigeria. Aplikasi yang dibangun menggunakan bahasa pemrograman Java (Arogundade dkk, 2010).

Perbedaan dengan penelitian yang akan dilakukan ini antara lain :

1) Objek penelitian

   Objek kedua penelitian tersebut terbatas hanya pada semester tertentu saja yakni penelitian pertama untuk mahasiswa semester 1 dan penelitian kedua mahasiswa semester 1 sampai 4 pada suatu jurusan. Sedangkan penelitian yang akan dilakukan untuk keseluruhan mahasiswa pada satu program studi perguruan tinggi. Dan terdapat waktu permintaan larangan mengajar dosen.

2) Ruang lingkup penelitian

   Ruang lingkup penelitian yang akan dilakukan ini akan membahas tentang kedua sistem penjadwalan pada satu program studi perguruan tinggi yaitu perkuliahan dan ujian akhir semester.

3) Perangkat lunak

   Perangkat lunak yang dipergunakan pada penelitian ini adalah bahasa pemrograman *Delphi Embarcadero RAD Studio* dan database *MySQL*.

1.5 Manfaat Penelitian

Manfaat penelitian ini diharapkan dapat memberikan kontribusi dan acuan serta pertimbangan bagi pengelolaan sistem penjadwalan perkuliahan dan ujian akhir semester pada suatu institusi perguruan tinggi.

1.6 Tujuan Penelitian

Tujuan penelitian ini adalah membuat sistem penjadwalan perkuliahan dan ujian akhir semester pada suatu intitusi perguruan tinggi dengan hasil penjadwalan yang mempunyai susunan bervariasi dan waktu proses yang lebih cepat menggunakan pendekatan Algoritma Genetika.

BAB II
TINJAUAN PUSTAKA

2.1 Tinjauan Pustaka

Sistem penjadwalan perkuliahan dengan pendekatan algoritma genetika telah diterapkan pada Universitas Devi Ahilya, Indore . Sistem penjadwalan yang dibuat terdiri dari 4 semester yaitu semester 1 sampai dengan 4 dan tidak mencantumkan waktu larangan dosen mengajar. Terdapat 7 kali perkuliahan setiap harinya dan 1 kali tatap muka mempunyai durasi selama 50 menit. Aplikasi penjadwalannya dibangun menggunakan bahasa pemrograman C# dan *SQL Server 2000* sebagai *database*nya. Sistem yang dihasilkan telah membuat penjadwalan perkuliahan yang efektif dengan menggunakan algoritma genetika (Jain dkk, 2010).

Demikian juga sistem penjadwalan ujian akhir semester menggunakan pendekatan algoritma genetika dengan jumlah mahasiswa 8000 orang dan 437 jadwal yang dilaksanakan selama 22 hari untuk semester pertama telah digunakan pada fakultas pertanian Universitas Abeokuta Nigeria. Aplikasi yang dibangun menggunakan bahasa pemrograman Java dan diuji coba dengan PC IBM prosessor Pentium IV 1 GHz, memory 512 MB dan monitor SVGA. Sistem yang dibuat telah dapat menghasilkan penjadwalan ujian yang efektif dan efesien (Arogundade dkk, 2010).

Algoritma genetika dapat digunakan untuk menghasilkan nilai maksimum luas *coverage area* dan biaya minimum operasional penempatan armada kapal TNI AL di kawasan timur Indonesia. Sistem yang dibuat diuji dengan penempatan armada kapal sebanyak 27 buah untuk ditempatkan pada 28 pangkalan yang ada. Hasil yang didapatkan adalah luas *coverage area* 1.942.929 Mil2 dan biaya operasional Rp. 2.853.447.000 dengan nilai *fitness* terbaik 6,6330. Jika dibandingkan dengan data lapangan yang ada yaitu total luas area yang harus diamankan sekitar 1,688,765 Mil2 dengan *budget* biaya Rp. 5.000.000.000 maka hasil yang didapatkan lebih efektif dan efesien (Hozari dkk, 2010).

Penjadwalan ujian mata kuliah Seminar dengan Teknik Inferensi *Boolean Satisfiability* (SAT) telah diterapkan pada Jurusan Teknik Informatika Universitas Katolik Parahyangan, Bandung. Mata kuliah Seminar adalah mata kuliah wajib yang harus diambil oleh mahasiswa sebelum skripsi yang berupa proposal penelitian. Jadwal ujian mata kuliah Seminar ini tidak ditetapkan di awal semester, melainkan seminggu sebelum masa ujian dimulai. Untuk prototipenya dikembangkan dengan menggunakan bahasa C++. Program yang dihasilkan telah diuji dengan data pada Semester Genap 2005/2006 dan berhasil memberikan solusi yang benar (Cecilia, 2008).

2.2 Landasan Teori

2.2.1 Pengertian Algoritma Genetika

Algoritma genetika merupakan evaluasi atau perkembangan dunia komputer dalam bidang kecerdasan buatan (*artificial intelligence*). Kemunculan algoritma genetika ini terinspirasi oleh teori Darwin dan teori-teori dalam ilmu biologi, sehingga banyak istilah dan konsep biologi yang digunakan dalam algoritma genetika, karena sesuai dengan namanya, proses-proses yang terjadi dalam algoritma genetika sama dengan apa yang terjadi pada evaluasi biologi.

Algoritma genetika adalah suatu algoritma pencarian yang berbasis pada mekanisme seleksi alam dan genetika. Algoritma genetika merupakan salah satu algoritma yang sangat tepat digunakan dalam menyelesaikan masalah optimasi kompleks, yang sulit dilakukan oleh metode konvensional.

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

Menurut (Suyanto, 2005) algoritma genetika telah banyak diaplikasikan untuk penyelesaian masalah dan pemodelan dalam bidang teknologi, bisnis dan *entertainment* seperti:

1) Optimasi

   Algoritma Genetika untuk optimasi *numeric* dan optimasi kombinatorial seperti *Traveling Salesman Problem* (TSP), perancangan *Intergrated Circuit* atau IC [LOU93], *job shop scheduling* [GOL91], optimasi video, dan suara.

2) Pemograman otomatis

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

3) Machine learning

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

4) Model Ekonomi

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

5) Model Sistem Imunisasi

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

6) Model Ekologis

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

7) Interaksi antara Evolusi dan Belajar

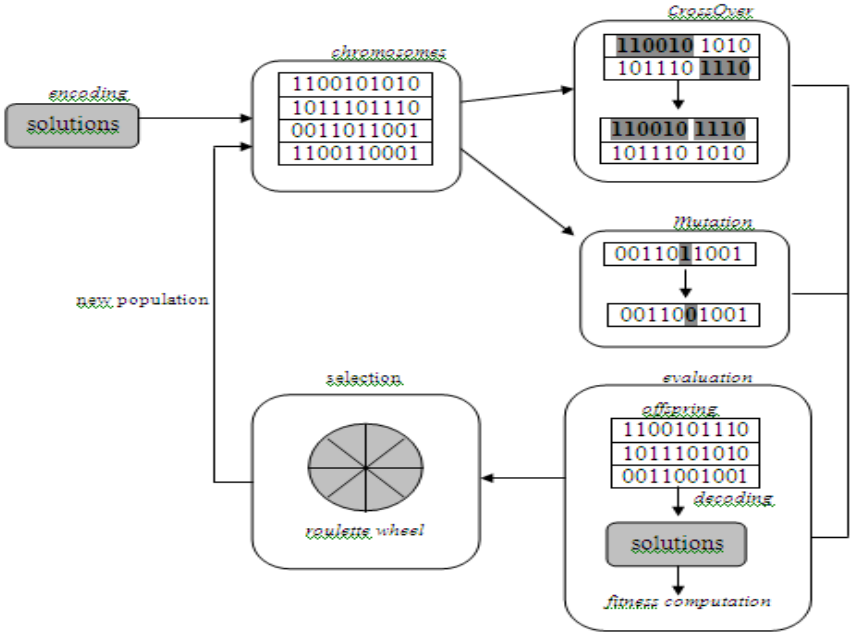
   Algoritma genetika telah digunakan untuk mempelajari bagaimana proses belajar suatu individu bisa mempengaruhi proses evolusi suatu *species* dan sebaliknya.

Ada 3 keuntungan utama dalam mengaplikasikan Algoritma Genetika pada masalah-masalah optimasi (Widodo, 2012) :

1) Algoritma Genetika tidak memerlukan kebutuhan matematis banyak mengenai masalah optimasi.
2) Kemudahan dan kenyamanan pada operator-operator evolusi membuat Algoritma Genetika sangat efektif dalam melakukan pencarian global.
3) Algoritma Genetika menyediakan banyak fleksibelitas untuk digabungkan dengan metode *heuristic* yang tergantung domain, untuk membuat implementasi yang efisien pada masalah-masalah khusus.

2.2.2 Struktur Umum Algoritma Genetika

Menurut (Suyanto, 2005) struktur umum algoritma genetika dapat dilihat pada gambar 2.1 :



Gambar 2.1 Struktur Umum Algoritma Genetika

Keterangan :

Dalam menyelesaikan suatu permasalahan, algoritma genetika diawali dengan menginisialisasikan himpunan solusi yang dibangkitkan secara acak (*random)*. Himpunan solusi ini disebut populasi (*Population*). Setiap individu pada populasi disebut kromosom (*Chromosom*), yang menggambarkan sebuah solusi dari suatu masalah yang akan diselesaikan. Sebuah kromosom dapat dinyatakan dalam simbol *string*, misalnya kumpulan *string bit*. Dalam sebuah populasi, setiap kromosom akan dievaluasi dengan menggunakan alat ukur yang disebut dengan *fitness* (tingkat kesesuaian). Nilai *fitness* ini digunakan untuk mencari dua kromosom (yang memiliki nilai *fitness* yang sesuai) dari sebuah populasi yang akan dijadikan sebagai kromosom induk untuk melakukan regenerasi. Kromosom induk ini akan melakukan regenerasi melalui pindah silang (c*rossover*) dan melakukan mutasi (*mutation*) yang akan menghasilkan kromosom baru (o*ffspring*). Pindah silang (c*rossover*) dilakukan dengan cara menggabungkan dua kromosom induk dengan menggunakan operator pindah silang (c*rossover*). Sedangkan mutasi hanya berlaku pada sebuah kromosom, dan kromosom ini akan mengalami suatu perubahan (misalnya : 11**1**01100 menjadi 11**0**01100 pada string bit).

Hasil dari pindah silang dan mutasi ini (o*ffspring*) akan di evaluasi dengan menggunakan alat ukur yang disebut *fitness* (tingkat kesesuaian). Kemudian akan dilihat apakah o*ffspring* ini merupakan solusi yang optimal atau belum. Jika optimal maka o*ffspring* ini lah jawabannya. Jika tidak, maka o*ffspring* ini akan diseleksi (*selection*) lagi dengan menggunakan salah satu metode seleksi. O*ffspring* yang lulus seleksi akan menjadi populasi yang baru dan akan melakukan regenerasi lagi, sedangkan yang tidak lulus seleksi akan dibuang. Regenerasi akan berhenti jika jumlah iterasi telah terpenuhi dan ditemukannya solusi optimal dari permasalahan yang diselesaikan.

2.2.3 Istilah dalam Algoritma Genetika

Terdapat beberapa definisi penting dalam Algoritma Genetika yang perlu diperhatikan, yaitu :

1) *Genotype* (Gen), sebuah nilai yang menyatakan satuan dasar yang membentuk suatu arti tertentu dalam satu kesatuan gen yang dinamakan kromosom. Dalam algoritma genetika, gen ini bisa berupa *biner, float, interger* maupun karakter, atau kombinatorial
2) *Allele*, merupakan nilai dari gen
3) Individu atau kromosom, gabungan gen-gen yang membentuk nilai tertentu dan merupakan salah satu solusi yang mungkin dari permasalahan yang diangkat.
4) Populasi, merupakan sekumpulan individu yang akan diproses bersama dalam satu siklus proses evalusi.
5) Generasi, menyatakan satu siklus proses evolusi atau satu iterasi di dalam algoritma genetika.

2.2.4 Komponen-komponen Algoritma Genetika

Terdapat beberapa komponen dalam algoritma genetika (Suyanto, 2005) yaitu :

1) Skema Pengkodeaan

   Teknik pengkodean adalah bagaimana mengkodekan gen dari kromosom, gen merupakan bagian dari kromosom. Satu gen akan mewakili satu variabel. Agar dapat diproses melalui algoritma genetik, maka alternatif solusi tersebut harus dikodekan terlebih dahulu kedalam bentuk kromosom. Masing-masing kromosom berisi sejumlah gen yang mengodekan informasi yang disimpan didalam individu atau kromosom.

   Gen dapat direpresentasikan dalam bentuk : *bit*, bilangan *real*, *string*, daftar aturan, gabungan dari beberapa kode, elemen permutasi, elemen program atau representasi lainnya yang dapat diimplementasikan untuk operator genetika.

2) Membangkitkan Populasi Awal dan Kromosom

Membangkitkan populasi awal adalah proses membangkitkan sejumlah individu atau kromosom secara acak atau melalui *procedure* tertentu. Ukuran untuk populasi tergantung pada masalah yang akan diselesaikan dan jenis operator genetika yang akan diimplementasikan. Setelah ukuran populasi ditentukan, kemudian dilakukan pembangkitan populasi awal.

Teknik dalam pembangkitan populasi awal pada penelitian ini menggunakan metode *random seach*, pencarian solusi dimulai dari suatu titik uji tertentu secara acak Titik uji tersebut dianggap sebagai alternatif solusi yang disebut sebagai populasi.

3) Nilai *Fitness*

Suatu individu dievaluasi berdasarkan suatu fungsi tertentu sebagai ukuran performansinya. Didalam evolusi alam, individu yang bernilai *fitness* tinggi yang akan bertahan hidup. Sedangkan individu yang bernilai *fitness* rendah akan mati. Pada masalah optimasi, solusi yang akan dicari adalah memaksimumkan fungsi *h* (dikenal sebagai masalah maksimasi) sehingga nilai *fitness* yang digunakan adalah nilai dari fungsi *h* tersebut, yakni $f = h$ (di mana f adalah nilai *fitness*). Tetapi jika masalahnya adalah meminimalkan fungsi *h* (masalah minimasi), maka fungsi *h* tidak bisa digunakan secara langsung. Hal ini disebabkan adanya aturan bahwa individu yang memiliki nilai *fitness* tinggi lebih mampu bertahan hidup pada generasi berikutnya. Oleh karena itu nilai *fitness* yang bisa digunakan adalah $f = 1/h$, yang artinya semakin kecil nilai *h*, semakin besar nilai *f*. Tetapi hal ini akan menjadi masalah jika *h* bisa bernilai 0, yang mengakibatkan *f* bisa bernilai tak hingga. Untuk mengatasinya, *h* perlu ditambah sebuah bilangan yang dianggap kecil [0-1] sehingga nilai *fitness*nya menjadi :

$$f = \frac{1}{(h + a)} \qquad (1)$$

dengan a adalah bilangan yang kecil dan bervariasi [0-1] sesuai dengan masalah yang akan diselesaikan.

4) Seleksi

Pembentukan susunan kromosom pada suatu populasi baru biasanya dilakukan secara proporsional sesuai dengan nilai *fitness*nya.Suatu metode seleksi yang umumnya digunakan adalah *roulette-wheel*. Metode seleksi dengan mesin roulette ini merupakan metode yang paling sederhana dan sering dikenal dengan nama *stochastic sampling with replacement*. Cara kerja metode ini adalah sebagai berikut:

a) Hitung total *fitness* semua individu
b) Hitung probabilitas seleksi masing-masing individu
c) Dari probabilitas tersebut, dihitung jatah interval masing-masing individu pada angka 0 sampai 1
d) Bangkitkan bilangan random antara 0 sampai 1
e) Dari bilangan random yang dihasilkan, tentukan urutan untuk populasi baru hasil proses seleksi.

5) Pindah Silang (*Crossover*)

Salah satu komponen yang paling penting dalam algoritma genetika adalah pindah silang atau *crossover*. Sebuah kromosom yang mengarah pada solusi yang baik dapat diperoleh dari proses memindah-silangkan dua buah kromosom. Pindah silang juga dapat berakibat buruk jika ukuran populasinya sangat kecil. Dalam suatu populasi yang sangat kecil, suatu kromosom dengan gen-gen yang mengarah pada solusi terbaik akan sangat cepat menyebar ke kromosom-kromosom lainnya. Untuk mengatasi masalah ini digunakan suatu aturan bahwa pindah silang hanya bisa dilakukan dengan suatu probabilitas tertentu, artinya pindah silang bisa dilakukan hanya jika suatu bilangan *random* yang dibangkitkan kurang dari probabilitas yang ditentukan tersebut. Pada umumnya probabilitas tersebut diset mendekati 1. Pindah silang yang paling sederhana adalah pindah silang satu titik potong (*one-point crossover*). Suatu titik

potong dipilih secara acak *(random)*, kemudian bagian pertama dari orangtua 1 digabungkan dengan bagian kedua dari orangtua 2 seperti terlihat pada tabel 2.1 :

Tabel 2.1 Pindah Silang pada Algoritma Genetika

	β_1				β_2				β_3			
Orang tua 1	**0**	**0**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**
Orang tua 2	1	1	0	0	0	0	0	0	0	0	0	0
	g1			g4	g5			g8	g9			g12
Anak 1	**0**	**0**	0	0	0	0	0	0	0	0	0	0
Anak 2	1	1	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**

6) Mutasi

Mutasi merupakan proses mengubah nilai dari satu atau beberapa gen dalam suatu kromosom. Mutasi ini berperan untuk menggantikan gen yang hilang dari populasi akibat seleksi yang memungkinkan munculnya kembali gen yang tidak muncul pada inisialisasi populasi. Metode mutasi yang digunakan adalah mutasi dalam pengkodean nilai. Proses mutasi dalam pengkodean nilai dapat dilakukan dengan berbagai cara, salah satunya yaitu dengan memilih sembarang posisi gen pada kromosom, nilai yang ada tersebut kemudian dirubah dengan suatu nilai tertentu yang diambil secara acak.

Contoh:

Kromosom sebelum mutasi : 1 **3** 4 **7** 6

Kromosom sesudah mutasi : 1 **2** 4 **8** 6

7) *Elitisme*

Karena seleksi dilakukan secara acak *(random)*, maka tidak ada jaminan bahwa suatu individu bernilai *fitness* tertinggi akan selalu terpilih. Kalaupun individu bernilai *fitness* tertinggi terpilih, mungkin saja individu tersebut akan rusak (nilai *fitness*nya menurun) karena proses pindah silang. Untuk menjaga agar individu bernilai *fitness* tertinggi tersebut tidak hilang selama evolusi, maka perlu dibuat satu atau beberapa salinannya. Prosedur ini dikenal sebagai *Elitisme*.

2.2.5 Penjadwalan

Penjadwalan adalah penempatan sumber daya (*resource*) dalam satu waktu. Penjadwalan mata kuliah dan ujian akhir semester merupakan persoalan penjadwalan umum dan sulit yang tujuannya adalah menjadwalkan pertemuan dari sumber daya. Sumber daya yang dimaksud adalah dosen pengasuh mata kuliah, mata kuliah, ruang kuliah, kelas mahasiswa, dan waktu perkuliahan (Setemen, 2008).

Terdapat batasan/persyaratan *(constraints)* dalam penyusunan penjadwalan mata kuliah dan ujian akhir semester. *Constraint* sendiri merupakan suatu syarat tidak boleh terjadi pelanggaran terhadap kendala yang ditetapkan agar dapat menghasilkan susunan penjadwalan yang baik. Beberapa *constraint* tersebut, yaitu :

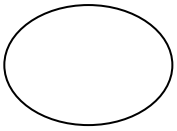
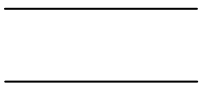
a) Dosen tidak boleh dijadwalkan lebih dari satu kali pada waktu yang bersamaan
b) Satu kelas dan ruang tidak boleh dijadwalkan lebih dari satu kali pada waktu yang bersamaan.

Jika terjadi pelanggaran terhadap kendala yang ditetapkan maka akan diberikan suatu nilai penalti atau hukuman antara 0 sampai 1 untuk setiap pelanggaran. Semakin kecil jumlah pelanggaran yang terjadi solusi penjadwalan yang dihasilkan akan semakin baik.

2.2.6 Desain Model Aplikasi

Desain model menggunakan pendekatan fungsional yang direpresentasikan menggunakan Diagram Arus Data (DAD) untuk menunjukkan secara fisik alur proses dan data pada program yang dibuat. Diagram yang menggunakan notasi-notasi untuk menggambarkan arus dari data sistem disebut dengan DAD. Notasi-notasi DAD dilihatkan pada tabel 2.2. (Jogiyanto, 1999).

Tabel 2.2 Simbol-simbol DAD

Notasi	Arti dan Keterangan
	***External entity*** (kesatuan luar) atau *boundary* (batas sistem) merupakan kesatuan (entity) di lingkungan luar sistem yang memberi *input* dan menerima *output*
	Data flow (arus data) yang mengalir diantara proses, simpanan data, dan kesatuan luar
	**Process (**Proses**)** merupakan arus data yang masuk ke proses menghasilkan arus data keluar dari proses
	**Data store** (simpanan data) yang menunjukkan nama file.

2.2.7 Model Pengembangan Perangkat Lunak *Waterfall*

Dalam penelitian ini akan digunakan model pengembangan perangkat lunak model *Waterfall*, model ini dipilih dengan alasan untuk membangun sistem ini dibutuhkan beberapa tahap yang berbeda yang merupakan sebuah pendekatan kepada perkembangan perangkat lunak yang sistematik dan sekuensial mulai pada tingkat dan kemajuan sistem pada seluruh analisa, perancangan, implementasi dan pengujian (Astuti, 2011).

1) **Analisa.** Proses pencarian kebutuhan diintensifkan dan difokuskan pada software. Untuk mengetahui sifat dari program yang akan dibuat, maka para software *engineer* harus mengerti tentang domain informasi dari software, misalnya fungsi yang dibutuhkan, *user interface*.
2) **Perancangan**. Proses ini digunakan untuk mengubah kebutuhan-kebutuhan diatas menjadi representasi ke dalam bentuk software

sebelum *coding* dimulai. Desain harus dapat mengimplementasikan kebutuhan yang telah disebutkan pada tahap sebelumnya. Seperti 2 (dua) aktivitas sebelumnya, maka proses ini juga harus didokumentasikan sebagai konfigurasi dari software.

3) **Implementasi.** Untuk dapat dimengerti oleh mesin, dalam hal ini adalah komputer, maka desain tadi harus diubah bentuknya menjadi bentuk yang dapat dimengerti oleh mesin, yaitu ke dalam bahasa pemrograman melalui proses *coding*. Tahap ini merupakan implementasi dari tahap desain yang secara teknis nantinya dikerjakan oleh programmer.
4) **Pengujian**. Sesuatu yang dibuat haruslah diuji cobakan. Demikian juga dengan software. Semua fungsi-fungsi software harus diujicobakan, agar software bebas dari kesalahan *(error)*, dan hasilnya harus benar-benar sesuai dengan kebutuhan yang sudah didefinisikan sebelumnya.

BAB III
METODE PENELITIAN

3.1 Bahan Penelitian

Bahan utama dari penelitian ini antara lain sebagai berikut :

1) Objek Penelitian

   Objek dalam penelitian ini menggunakan data-data yang berkaitan dengan sistem penjadwalan perkuliahan dan ujian akhir semester tahun akademik 2010/2011 pada program Studi Diploma III Manajemen Informatika STMIK Palangkaraya.

2) Metode Pengumpulan Data

   a) Observasi

      Observasi merupakan pengumpulan data yang dilakukan dengan mengamati sistem penjadwalan perkuliahan dan ujian akhir semester pada satu program studi perguruan tinggi.

   b) Wawancara

      Teknik pengumpulan data ini dilakukan dengan bertatap muka langsung kepada sumber dengan melakukan tanya jawab mengenai data yang akan diambil.

   c) Tinjauan Pustaka

      Tinjauan Pustaka merupakan teknik pengumpulan data yang dilakukan dengan media buku-buku pedoman yang berhubungan dengan pembuatan sistem ini diantaranya studi literatur algoritma genetika untuk menyelesaikan permasalahan penjadwalan perkuliahan dan ujian akhir semester pada satu program studi perguruan tinggi.

3.2 Alat Penelitian

Sebagai alat yang digunakan pada penelitian ini menggunakan Acer Aspire One Pro notebook, Intel (R) Atom (TM) N270 @ 1,60 Ghz, RAM 1 GB, sistem operasi menggunakan Windows XP SP3. Untuk aplikasi algoritma

genetika dibangun dengan bahasa pemrograman *Delphi* dari *Embarcadero RAD Studio* dan database *MySQL*.

3.3 Jalan Penelitian

Aplikasi penelitian ini dibuat berdasarkan pengembangan perangkat lunak air terjun (*waterfall*). Tahapan-tahapannya yaitu analisa kebutuhan, perancangan (*design*), implementasi dan pengujian (Astuti, 2011).

1) Analisa Kebutuhan

   Tahap ini untuk mengumpulkan data yang diperlukan sebagai bahan masukan *(input)* untuk membuat aplikasi penjadwalan dengan algoritma genetika yaitu data mata kuliah, data dosen, data kelas, data ruang, data *timeslot* dan data larangan dosen. Proses untuk mengolah data *input* adalah dengan algoritma genetika. *Output* yang dihasilkan sesuai dengan apa yang diharapkan.

2) Perancangan (*Design*)

   Tahap ini terdiri dari 3 bagian yaitu permodelan proses dan data bertujuan untuk merancang diagram arus data (DAD), *entity relationship diagram (ERD)* dan tabel *database*, serta perancangan *user interface* bertujuan untuk merancang *interface*/tampilan *input* dan *output* sistem pada layar dengan menggunakan prinsip-prinsip *GUI (Graphical User Interface)* yang mudah dipahami oleh pengguna sistem.

3) Implementasi

   Mengimplementasikan rancangan sistem ke dalam modul program (*coding program*). Pada proses ini akan mengkonversikan perancangan ke dalam kegiatan operasi *coding* dengan menggunakan bahasa pemograman tertentu yang dilandasi pada penggunaan algoritma genetika untuk proses penyusunan jadwal perkuliahan dan ujian.

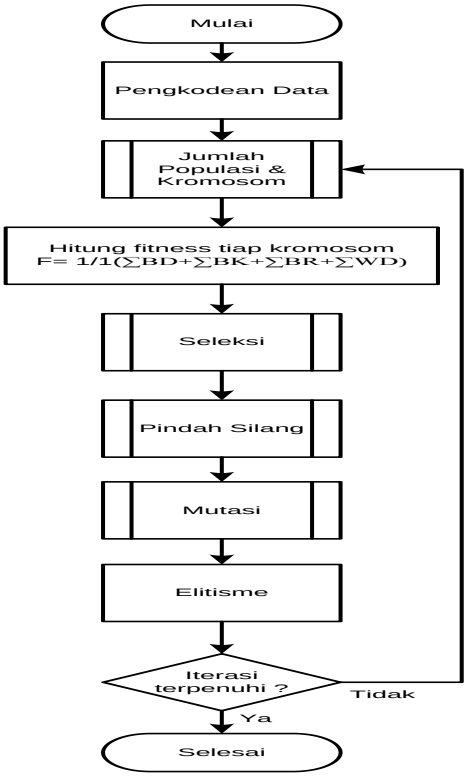
4) Pengujian

   Menguji apakah aplikasi telah siap digunakan dan berfungsi dengan baik. Proses pengujian dilakukan pada logika internal untuk

memastikan semua pernyataan sudah diuji. Pengujian eksternal fungsional untuk menemukan kesalahan-kesalahan dan memastikan bahwa *input* akan memberikan hasil yang aktual sesuai yang dibutuhkan. Pengujian pada penelitian ini mengambil studi kasus penjadwalan perkuliahan dan ujian akhir semester tahun 2010/2011 program Diploma III Manajemen Informatika STMIK Palangkaraya.

3.3.1 Penjadwalan dengan Algoritma Genetika

Gambar 3.1 memperlihatkan diagram alir algoritma genetika secara umum pada penelitian ini :



Gambar 3.1 *Flowchart* penjadwalan algoritma genetika

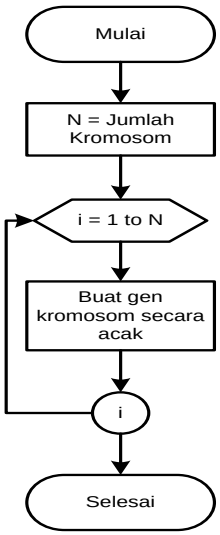
1) Teknik Pengkodean

Teknik pengkodean adalah bagaimana mengkodekan gen dari kromosom. Masing-masing kromosom berisi sejumlah gen yang mengkodekan informasi yang disimpan didalam kromosom.

Pada penelitian ini menggunakan teknik pengkodean dalam bentuk *string bit / varchar* yang dipergunakan dalam pemrograman genetika.

2) Menentukan populasi awal dan Inisialisasi kromosom

Menentukan populasi awal adalah proses membangkitkan sejumlah kromosom secara acak (*random*). Kromosom menyatakan salah satu alternatif solusi yang mungkin. Kromosom dapat dikatakan sama dengan individu. Ukuran populasi tergantung pada masalah yang akan diselesaikan. Setelah ukuran populasi ditentukan, kemudian dilakukan pembangkitan populasi awal dengan cara melakukan inisialisasi solusi yang mungkin kedalam sejumlah kromosom. Panjang satu kromosom ditentukan berdasarkan permasalahan yang diteliti. *Flowchart*nya pada gambar 3.2 :



Gambar 3.2 *Flowchart* pembentukan kromosom

Pada penelitian tentang penjadwalan ini solusi yang akan dihasilkan adalah menentukan waktu dan ruang untuk perkuliahan. Panjang satu kromosom adalah gabungan gen berdasarkan jumlah dari seluruh mata kuliah dan kelas yang ditawarkan pada semester aktif. Satu gen berisi informasi waktu dan ruang untuk satu matakuliah dan kelas. Sebagai contoh untuk inisialisasi pembentukan kromosom, misalkan ada sebaran mata kuliah pada tabel 3.1, sebaran waktu pada tabel 3.2 dan sebaran ruang yang tersedia pada tabel 3.3.

Tabel 3.1 Contoh sebaran mata kuliah

No	Id MK	Nama MK	Id Dosen	SKS	SMT	KLS
1	M01	Algoritma I	1	2	1	A
2	M02	Sistem Operasi	4	3	1	A
3	M03	Kalkulus	3	2	3	A
4	M04	Sistem Basis Data	4	3	3	A

Tabel 3.2 Contoh sebaran waktu

Index Waktu	Hari	Waktu
T1	Senin	07.00 – 08.45
T2	Senin	09.00 – 10.45
T3	Selasa	07.00 – 08.45
T4	Selasa	09.00 – 10.45

Tabel 3.3 Contoh ruang yang tersedia

Id Ruang	Nama Ruang
1	Ruang A
2	Ruang B

Terdapat permintaan dosen D03 tidak bisa mengajar pada hari senin jam 09.00. Diasumsikan dalam satu populasi yang terbentuk berjumlah 4 kromosom sesuai dengan jumlah mata kuliah dan kelas yang ada serta masing-masing kromosom memiliki 4 gen.

M02K01R02T3 M03K01R01T1 M01K01R02T2 M04K01R01T3
M01K01R01T2 M03K01R02T4 M04K01R02T4 M02K01R01T3
M01K01R02T1 M02K01R01T1 M03K01R01T2 M04K01R01T4
M04K01R02T4 M02K01R01T4 M01K01R02T1 M03K01R01T2

Urutan kode pada setiap gen mewakili kode mata kuliah, kode kelas, kode ruang dan *index* waktu. Penempatan urutan kode pada setiap gen dilakukan secara acak *(random)* berdasarkan suatu bilangan yang dibangkitkan secara acak *(random)* pula. Pada contoh bilangan tersebut merupakan jumlah dari seluruh mata kuliah dan kelas yang ditawarkan.

3) Fungsi *fitness*

Suatu individu dievaluasi berdasarkan suatu fungsi tertentu sebagai ukuran performansinya. Didalam evolusi alam, individu yang bernilai *fitness* tinggi yang akan bertahan hidup. Sedangkan individu yang bernilai *fitness* rendah akan mati.

Fungsi yang digunakan untuk mengukur nilai kecocokan atau derajat optimalitas suatu kromosom disebut dengan *fitness function.* Nilai yang dihasilkan dari fungsi tersebut menandakan seberapa optimal solusi yang diperoleh. Nilai yang dihasilkan oleh fungsi *fitness* merepresentasikan seberapa banyak jumlah persyaratan yang dilanggar, sehingga dalam kasus penjadwalan perkuliahan semakin kecil jumlah pelanggaran yang dihasilkan maka solusi yang dihasilkan akan semakin baik. Untuk setiap pelanggaran yang terjadi akan diberikan nilai 1. Agar tidak terjadi nilai *fitness* yang tak terhingga maka jumlah total semua pelanggaran akan ditambahkan 1.

$$F = \frac{1}{1 + (\sum BD + \sum BK + \sum BR + \sum WD)} \qquad (2)$$

Keterangan :
BD = Banyaknya bentrok dosen & mata kuliah
BK = Banyaknya bentrok kelas perkuliahan
BR = Banyaknya bentrok ruang yang digunakan
WD = Banyaknya waktu dosen yang dilanggar

Beberapa batasan yang digunakan dalam penyusunan penjadwalan ini adalah :

c) Dosen tidak boleh dijadwalkan lebih dari satu kali pada waktu yang bersamaan
d) Satu kelas dan ruang tidak boleh dijadwalkan lebih dari satu kali pada waktu yang bersamaan.
e) Dosen tidak boleh dijadwalkan pada waktu yang telah ditentukan oleh dosen yang bersangkutan.

Dari contoh yang ada akan menghasilkan nilai *fitness* sebagai berikut :

$$\textit{Fitness}\ \text{Kromosom 1} = \frac{1}{1 + (1 + 0 + 0 + 0)} = 0{,}5$$

$$\textit{Fitness}\ \text{Kromosom 2} = \frac{1}{1 + (0 + 0 + 1 + 0)} = 0{,}5$$

$$\textit{Fitness}\ \text{Kromosom 3} = \frac{1}{1 + (0 + 1 + 0 + 1)} = 0{,}33$$

$$\textit{Fitness}\ \text{Kromosom 4} = \frac{1}{1 + (1 + 0 + 0 + 1)} = 0{,}33$$

3) Seleksi

Pembentukan susunan kromosom pada suatu populasi baru dilakukan dengan menggunakan metode seleksi *roulette-wheel*. Sesuai dengan namanya, metode ini menirukan permainan *roulette-wheel* dimana masing-masing kromosom menempati potongan lingkaran pada *roulette-wheel* secara proporsional sesuai dengan nilai *fitness*nya. Kromosom yang memiliki nilai *fitness* lebih besar menempati potongan lingkaran yang lebih besar dibandingkan dengan kromosom bernilai *fitness* rendah.

Langkah pertama metode ini adalah dengan menghitung total nilai *fitness* seluruh kromosom seperti tabel 3.4 :

Tabel 3.4 Total Nilai *Fitness*

Kromosom	Nilai *fitness*
1	0,5
2	0,5
3	0,33
4	0,33
Total Nilai *Fitness*	1,66

Langkah kedua adalah menghitung probabilitas setiap kromosom dengan cara membagi nilai *fitness* tiap kromosom dengan total nilai *fitness*. Sehingga didapatkan hasil seperti tabel 3.5 :

Tabel 3.5 Probabilitas tiap kromosom

Kromosom	Probabilitas
1	0,5 / 1,66 = 0,301
2	0,5 / 1,66 = 0,301
3	0,33 / 1,66 = 0,199
4	0,33 / 1,66 = 0,199
Total Probabilitas	1

Langkah ketiga adalah menempatkan masing-masing kromosom pada interval nilai [0 – 1]. Dapat dilihat pada tabel 3.6.

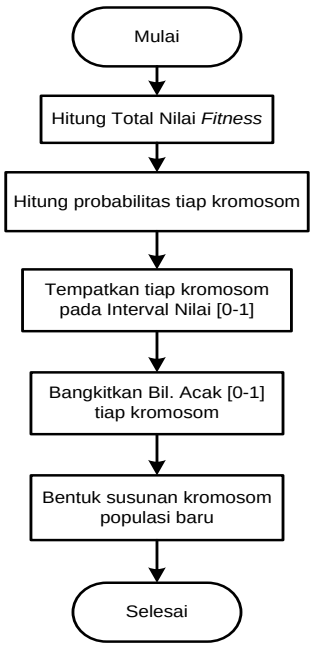
Tabel 3.6 Interval tiap kromosom

Kromosom	Interval Nilai
1	0 - 0,301
2	0,302 – 0,602
3	0,603 – 0,801
4	0,802 – 1

Untuk menentukan susunan populasi baru hasil seleksi maka dibangkitkan bilangan acak *(random)* antara [0 -1]. Dimisalkan bilangan yang dibangkitkan adalah [0,75 ; 0,25 ; 0,9 dan 0,5] maka susunan kromosom populasi baru hasil seleksi adalah :

M01K01R02T1 M02K01R01T1 M03K01R01T2 M04K01R01T4
M02K01R02T3 M03K01R01T1 M01K01R02T2 M04K01R01T3
M04K01R02T4 M02K01R01T4 M01K01R02T1 M03K01R01T2
M01K01R01T2 M03K01R02T4 M04K01R02T4 M02K01R01T3

Gambar 3.3 memperlihatkan *flowchart* seleksi :



Gambar 3.3 *Flowchart* seleksi

4) Pindah Silang (*CrossOver*)

Pindah silang *(CrossOver)* digunakan sebagai metode pemotongan kromosom secara acak *(random)* dan merupakan penggabungan bagian pertama dari kromosom induk 1 dengan bagian kedua dari kromoson induk 2 .

Pindah silang bisa dilakukan hanya jika suatu bilangan acak (*random)* yang dibangkitkan untuk kromosom kurang dari probabilitas pindah silang (Pc) yang ditentukan. Menurut (Suyanto, 2005) Pc umumnya diset mendekati 1, misalnya 0,5.

Metode pindah silang yang paling umum digunakan adalah pindah silang satu titik potong (*one-point crossover*). Suatu titik potong dipilih secara acak *(random)*, kemudian bagian pertama dari kromosom induk 1 digabungkan dengan bagian kedua dari kromosom induk 2. Bilangan acak *(random)* yang dibangkitkan untuk menentukan posisi titik potong adalah [1-N] dimana N merupakan banyaknya jumlah gen dalam satu kromosom.

Dimisalkan dari contoh yang ada nilai untuk kromosom 2 dan 4 kurang dari Pc yang ditetapkan serta bilangan acak *(random)* untuk posisi titik potong adalah pada posisi gen ke-2, maka proses pindah silangnya adalah :

Kromosom 2 = M02K01R02T3 | M03K01R01T1 M01K01R02T2 M04K01R01T3

Kromosom 4 = M01K01R01T2 | M03K01R02T4 M04K01R02T4 M02K01R01T3

Titik Potong

Hasil pindah silang kedua kromosom tersebut adalah :

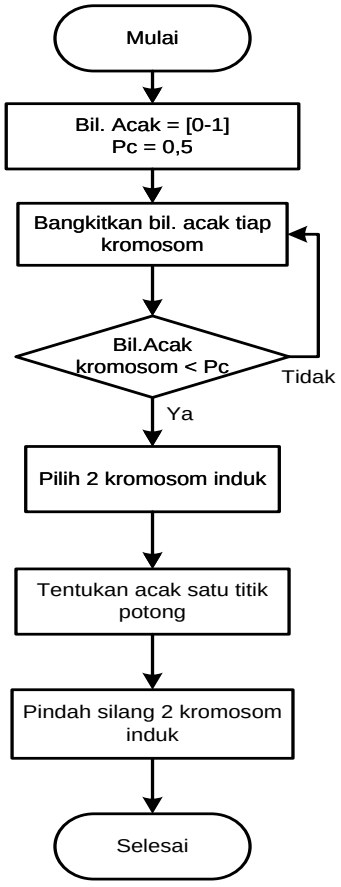
Kromosom 2 = M02K01R02T3 M03K01R02T4 M04K01R02T4 M02K01R01T3

Kromosom 4 = M01K01R01T2 M03K01R01T1 M01K01R02T2 M04K01R01T3

$$\textit{Fitness} \text{ kromosom 2 sesudah pindah silang} = \frac{1}{1 + (1 + 1 + 1 + 0)} = 0{,}25$$

$$\textit{Fitness} \text{ kromosom 4 sesudah pindah silang} = \frac{1}{1 + (1 + 1 + 0 + 0)} = 0{,}33$$

Gambar 3.4 merupakan *flowchart* pindah silang :



Gambar 3.4 *Flowchart* pindah silang

5) Mutasi

Proses mutasi adalah suatu proses kemungkinan memodifikasi informasi gen-gen pada suatu kromosom Perubahan ini dapat membuat solusi duplikasi menjadi memiliki nilai *fitness* yang lebih rendah maupun lebih tinggi daripada solusi induknya. Jika ternyata diperoleh solusi yang memiliki *fitness* yang lebih tinggi maka hal itulah yang diharapkan. Tetapi jika diperoleh solusi dengan nilai *fitness* yang lebih rendah maka bisa jadi pada iterasi berikutnya diperoleh solusi hasil mutasi yang lebih baik nilai *fitness*nya daripada solusi induknya. Untuk semua gen yang ada, jika bilangan acak *(random)* yang dibangkitkan kurang dari probabilitas mutasi (Pmut) yang telah ditentukan maka beberapa informasi gen akan dirubah dengan menggunakan metode pengkodean nilai. Pmut umumnya diset antara [0 – 1], misalnya 0,1 (Suyanto, 2005).

Untuk mendapatkan posisi gen yang akan dimutasi maka perlu dihitung jumlah total gen dalam satu populasi yaitu **Total gen = Jumlah gen dalam satu kromosom x Jumlah kromosom yang ada**. Berdasarkan contoh yang ada maka total gen adalah = 4 x 4 = 16. Probabilitas mutasi ditetapkan 0,1 maka diharapkan mutasi yang terjadi adalah : 0,1 x 16 = 1,6 = 2 gen yang akan mengalami mutasi. Selanjutnya dilakukan iterasi sebanyak jumlah total gen [0-16] dan membangkitkan bilangan acak untuk tiap iterasi antara [0-1]. Diasumsikan gen yang mendapatkan bilangan dibawah probabilitas mutasi adalah gen 2 dan 3. Informasi dalam gen yang akan dirubah adalah waktu perkuliahan, maka hasil mutasi pada kromosom tersebut adalah :

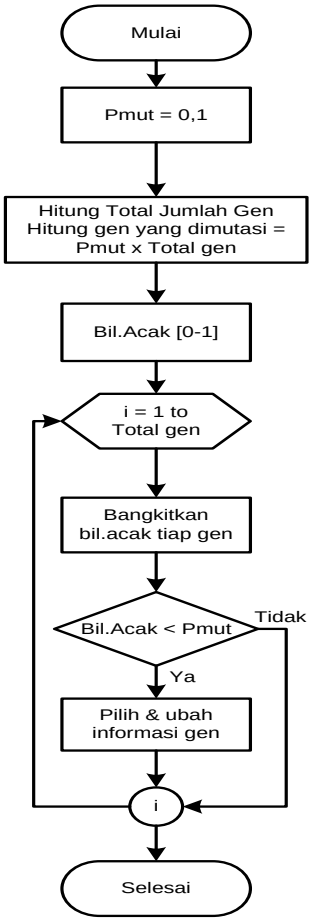
Sebelum mutasi = M01K01R02T1 M02K01R01T1 M03K01R01T2 M04K01R01T4
Sesudah mutasi = M01K01R02T1 M02K01R01T2 M03K01R01T3 M04K01R01T4

Sehingga akan menghasilkan susunan kromosom baru sebagai berikut :

M01K01R02T1 M02K01R01T2 M03K01R01T3 M04K01R01T4
M02K01R02T3 M03K01R01T1 M01K01R02T2 M04K01R01T3
M04K01R02T4 M02K01R01T4 M01K01R02T1 M03K01R01T2
M01K01R01T2 M03K01R02T4 M04K01R02T4 M02K01R01T3

Hasilnya kromosom 1 memiliki nilai *fitness* terbaik karena tidak terdapat pelanggaran yang telah ditetapkan dan merupakan solusi yang diinginkan.

Flowchart mutasi dapat dilihat pada gambar 3.5 :



Gambar 3.5 *Flowchart* mutasi

6) *Elitisme*

Proses ini adalah untuk membuat salinan *(copy)* individu bernilai *fitness* tertinggi agar tidak hilang selama proses evolusi.

7) Kondisi Selesai

Kondisi selesai yang dapat menghentikan proses algoritma genetika ini adalah jika jumlah generasi atau iterasi maksimum telah tercapai.

Untuk penjadwalan ujian akhir semester gen-gen yang membentuk kromosom lebih sedikit dibandingkan dengan penjadwalan kuliah. Panjang satu kromosom untuk penjadwalan ujian akhir semester berisi informasi mata kuliah, ruang ujian dan waktu ujian.

M02R02T1 M03R01T3 M01R02T2 M04R01T3
M01R01T2 M03R02T4 M04R02T4 M02R01T3
M01R02T1 M02R01T1 M03R01T2 M04R01T4
M04R02T1 M02R01T4 M01R02T1 M03R01T2

Fungsi *fitness* juga akan berbeda dengan penjadwalan perkuliahan, yaitu :

$$F = \frac{1}{1 + (\sum BS + \sum BR)} \qquad (3)$$

Keterangan :
BS = Banyaknya bentrok semester
BR = Banyaknya bentrok ruang yang digunakan

Terdapat dua batasan yang tidak boleh dilanggar yaitu :

a) Satu semester tidak boleh dijadwalkan lebih dari satu kali pada waktu bersamaan
b) Satu ruang tidak boleh dijadwalkan lebih dari satu kali pada waktu yang bersamaan.

Sedangkan tahapan proses algoritma genetika lainnya seleksi, pindah silang, mutasi dan kondisi selesai tidak berbeda dengan proses penjadwalan perkuliahan.

3.3.2 Perancangan Sistem

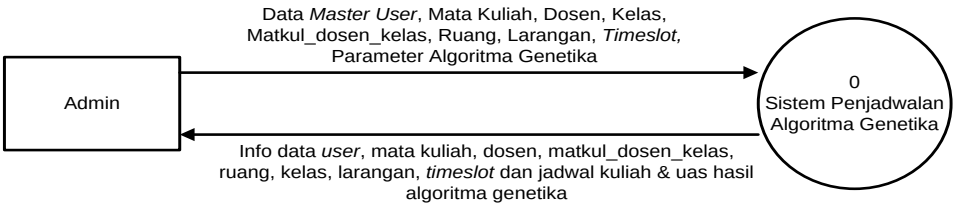
Tahap ini terdiri dari 3 bagian yaitu perancangan permodelan proses, data dan *user interface.*

3.3.2.1 Permodelan Proses

Permodelan proses berupa pemodelan fungsi yang digambarkan dengan Diagram Arus Data (DAD).

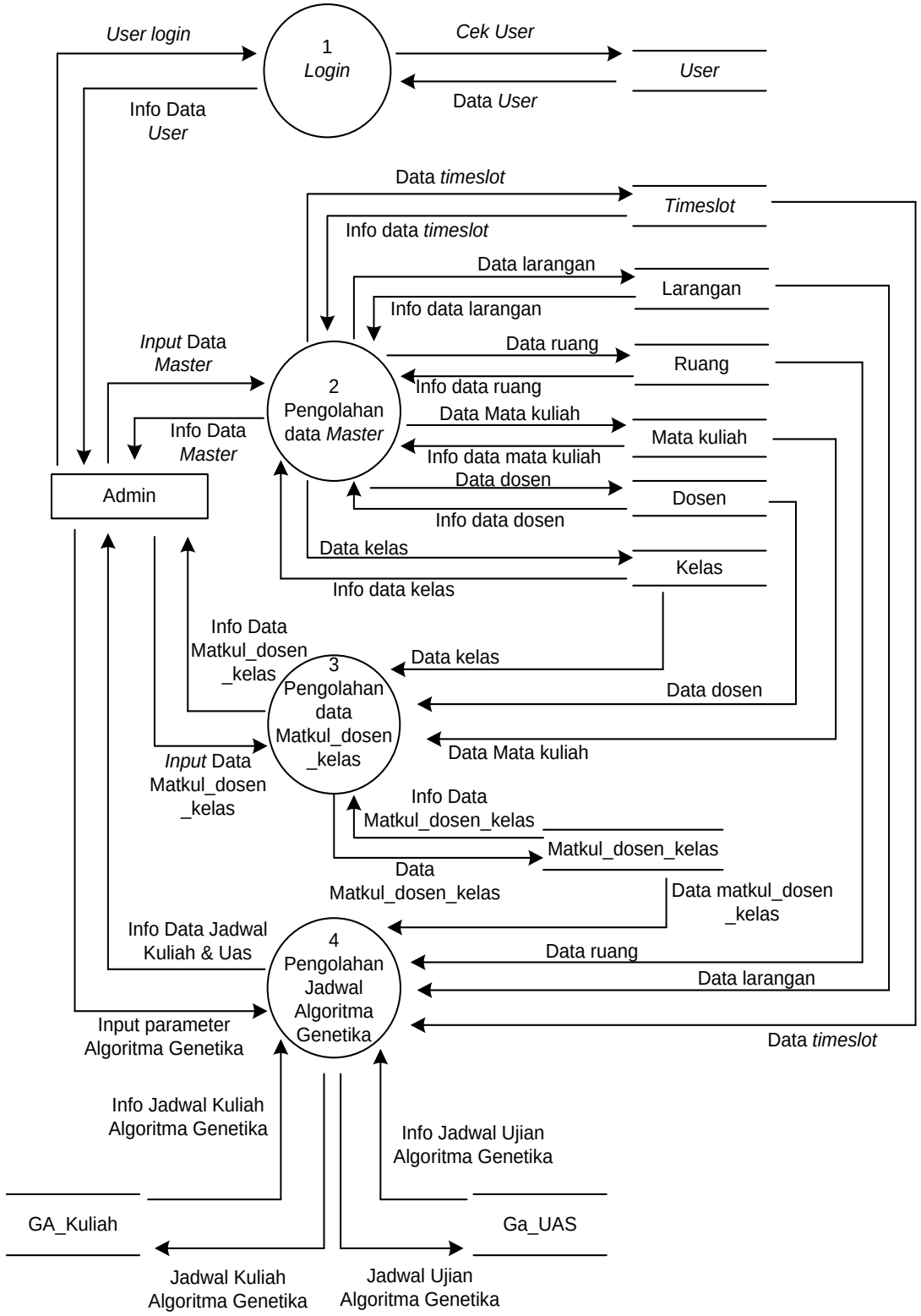
1) Diagram Konteks (DAD level 0)

Diagram konteks sistem penjadwalan merupakan level paling awal dari suatu DAD. Di dalam *context diagram* sistem penjadwalan ini terdapat satu entitas admin penyusun jadwal. Admin memberikan data mata kuliah, dosen, kelas, matkul_dosen_kelas, ruang, larangan, *timeslot* dan parameter algoritma genetika ke dalam sistem untuk menghasilkan penjadwalan hasil proses algoritma genetika. Terlihat pada gambar 3.6



Gambar 3.6 Diagram Konteks

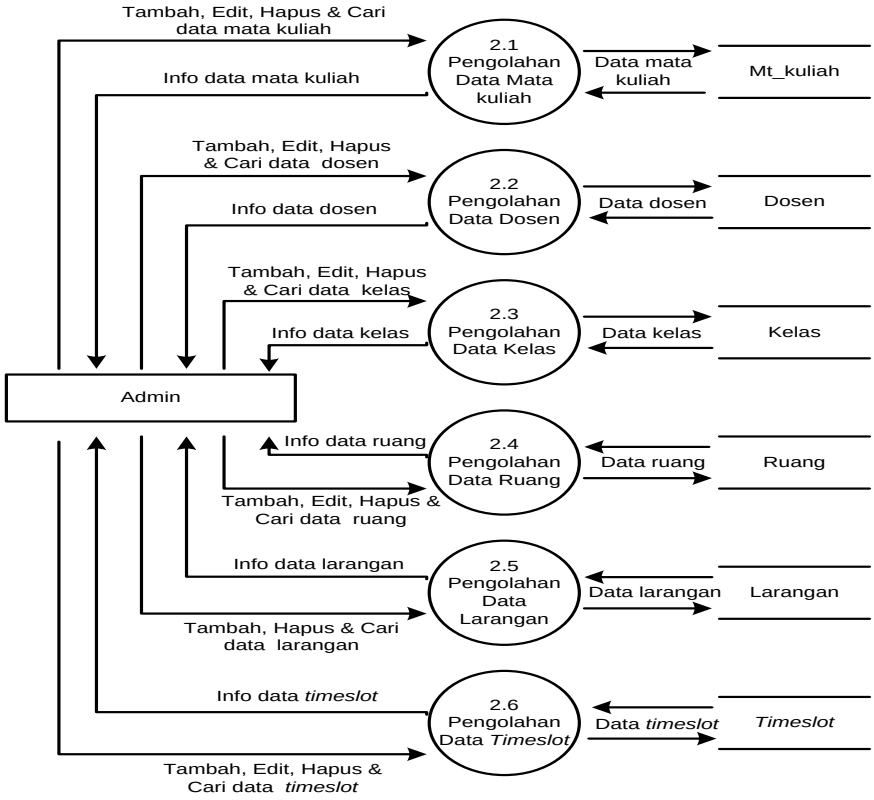
2) Diagram Arus Data Level 1 Sistem Penjadwalan Algoritma Genetika
   Diagram Arus Data Level 1 Sistem Penjadwalan Algoritma Genetika terdiri dari 4 proses, pertama proses *login* untuk admin dengan memasukkan *user name* dan *password* yang telah tersimpan dalam sistem. Proses kedua adalah pengolahan data-data *master* mata kuliah, dosen, kelas, ruang, larangan dan *timeslot*. Proses ketiga adalah proses untuk pengolahan data penggabungan data matakuliah, dosen dan kelas yang diampu oleh dosen. Proses keempat merupakan proses pengolahan jadwal dengan algoritma genetika yang diambil dari tabel mata kuliah, dosen, kelas, matkul_dosen_kelas, ruang, larangan dan *timeslot* disertai dengan peng*input*an paramater algoritma genetika oleh admin. Data hasil penjadwalan akan disimpan dalam tabel GA_Kuliah dan GA_Ujian. Ditunjukkan pada gambar 3.7



Gambar 3.7 DAD Level 1 Sistem Penjadwalan Algoritma Genetika

3) Diagram Arus Data Level 2 Proses 2 Pengolahan Data

Pada Diagram Arus Data Level 2 Proses 2 Pengolahan Data *Master* terdapat 6 proses yaitu pengolahan data mata kuliah, pengolahan data dosen, pengolahan data kelas, pengolahan data ruang, pengolahan data larangan dan pengolahan data *timeslot*. Masing-masing pengolahan data memiliki menu tambah, edit, hapus dan cari data. Terlihat pada gambar 3.8

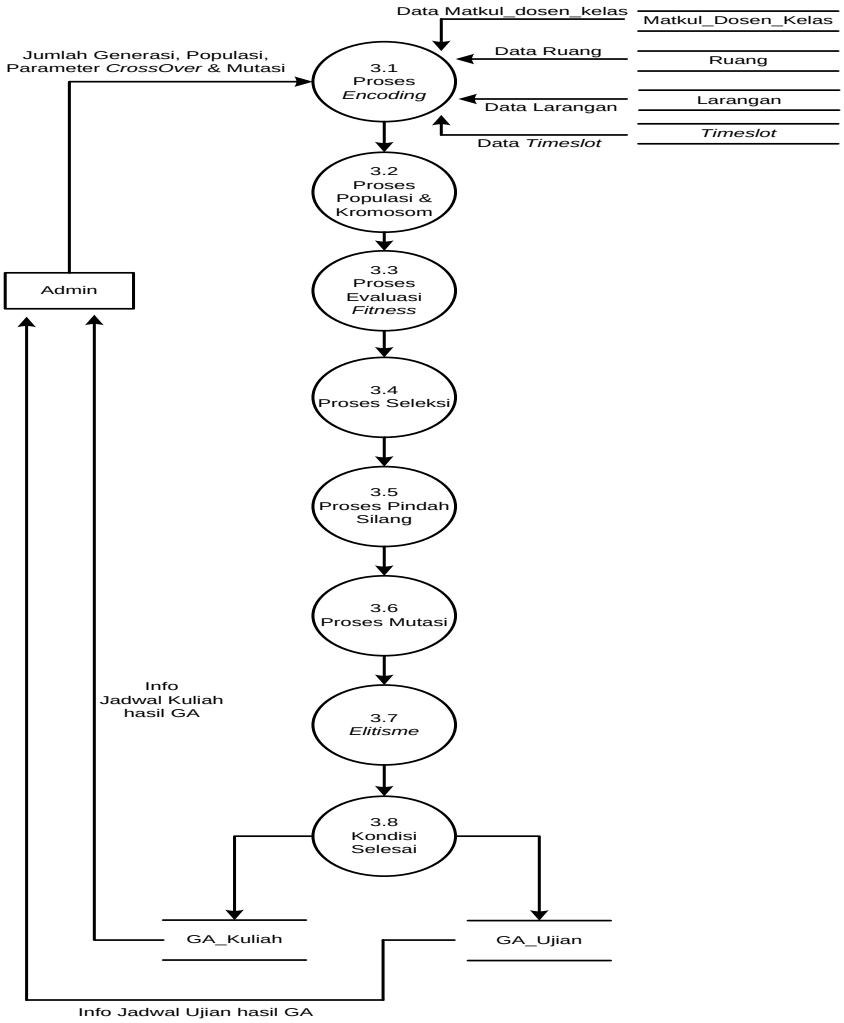


Gambar 3.8 DAD Level 2 Proses 2 Pengolahan Data

4) DAD Level 2 Proses 4 Pengolahan Jadwal Algoritma Genetika

Diagram Arus Data Level 2 Proses 4 Pengolahan Jadwal Algoritma Genetika memiliki 8 proses. Admin meng*input*kan jumlah generasi, populasi, probabilitas *crossover* dan mutasi. Proses *Encoding* mengambil data dari tabel matkul_dosen_kelas, larangan dan *timeslot*. Selanjutnya diproses untuk membentuk populasi dan

kromosom. Evaluasi *fitness* untuk menilai seberapa bagus solusi individu yang dihasilkan. Proses seleksi merupakan proses pembentukan susunan kromosom baru. Proses pindah silang adalah proses memindah silangkan gen dari 2 kromosom induk.. Proses mutasi mengganti atau merubah gen dari kromosom. Proses *elitisme* untuk menyimpan kromosom terbaik dari satu generasi. Kondisi selesai merupakan suatu kondisi yang digunakan untuk menghentikan proses pengulangan algoritma genetika. Ditunjukkan pada gambar 3.9



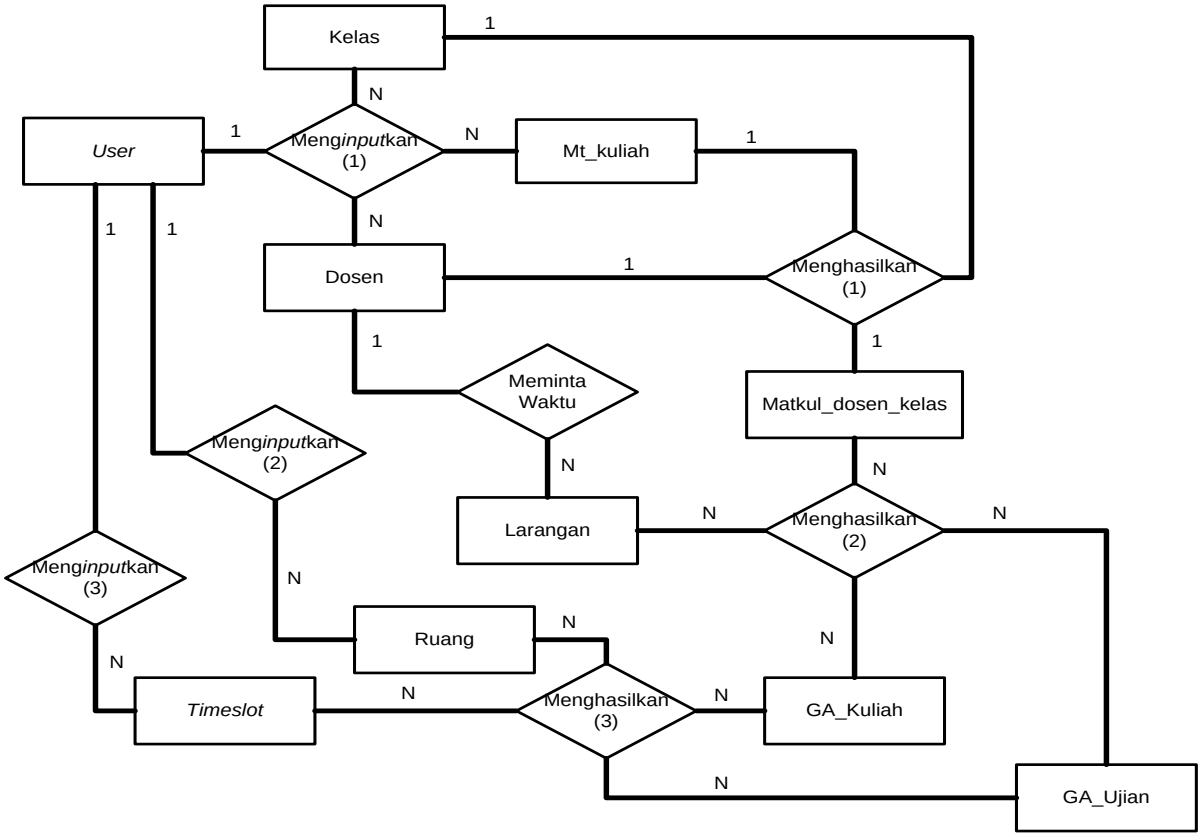
Gambar 3.9 DAD Level 2 Proses 4 Pengolahan Jadwal Algoritma Genetika

3.3.2.2 Permodelan Data

Terdapat 2 perancangan permodelan data yang digunakan yaitu perancangan *database* konseptual *(Entity Relationship Diagram)* dan fisik.

1) Perancangan *database* konseptual *(Entity Relationship Diagram)*

   Entity Relationship Diagram (ERD) merupakan gambaran mengenai berelasinya antar entitas yang digunakan dalam sistem. Gambar 3.10 menunjukkan perancangan *ERD* pada penelitian ini *:*



Gambar 3.10 Perancangan *ERD*

Keterangan :

User	: Id_user, Username, Password
Dosen	: Id_dosen, Nm_dosen
Mt_Kuliah	: Id_matkul, Nm_matkul, SKS, Semester, Is_praktikum
Ruang	: Id_ruang, Ruang, Is_lab
Kelas	: Id_kelas, Kelas

Matkul_dosen_kelas : Id_md, Id_matkul, Id_dosen, Id_kelas
Larangan : Id_larangan, Id_dosen, Id_time, Semester
Timeslot : Id_time, hari, waktu
GA_Kuliah : Id_best_kul, Num_gen_kul, kromosom_kul, fitness_kul
GA_Ujian : Id_best_uas, Num_gen_uas, kromosom_uas, fitness_uas

2) Perancangan *database* fisik

Terdapat 10 tabel *database* yang digunakan, yaitu :

a) Nama tabel : *User*

Kunci Utama : Id_dosen

Fungsi : Untuk menyimpan data dan *password user*

Tabel 3.7 Struktur tabel *User*

No	Nama Field	Type dan Panjang Field	Keterangan
1	Id_user	Integer (10)	Nomor id *user*
2	Username	Varchar (50)	Nama *user*
3	Password	Varchar (32)	*Password user*

b) Nama tabel : Dosen

Kunci Utama : Id_dosen

Fungsi : Untuk menyimpan data dosen

Tabel 3.8 Struktur tabel dosen

No	Nama Field	Type dan Panjang Field	Keterangan
1	Id_dosen	Integer (11)	Nomor id dosen
2	Nm_dosen	Varchar (50)	Nama dosen

c) Nama tabel : Mt_Kuliah

Kunci Utama : Id_matkul

Fungsi : Untuk menyimpan data mata kuliah

Tabel 3.9 Struktur tabel Mt_Kuliah

No	Nama Field	Type dan Panjang Field	Keterangan
1	Id_matkul	Integer (10)	Nomor id mata kuliah
2	Nm_matkul	Varchar (50)	Nama mata kuliah
3	SKS	tinyint (4)	Sks mata kuliah
4	Semester	tinyint (4)	Semester
5	Is_Praktikum	enum('Y','N')	Type mata kuliah

d) Nama tabel : Ruang
Kunci Utama : Id_ruang
Fungsi : Untuk menyimpan data ruangan perkuliahan

Tabel 3.10 Struktur tabel Ruang

No	Nama Field	Type dan Panjang Field	Keterangan
1	Id_ruang	Integer (10)	Nomor id ruang
2	Ruang	Varchar (30)	Nama ruang
3	Is_Lab	enum('Y','N')	Type / Jenis ruang

e) Nama tabel : Kelas
Kunci Utama : Id_kelas
Fungsi : Untuk menyimpan data kelas

Tabel 3.11 Struktur tabel Kelas

No	Nama Field	Type dan Panjang Field	Keterangan
1	Id_kelas	Integer (10)	Nomor id kelas
2	Kelas	Varchar (20)	Nama kelas

f) Nama tabel : Matkul_Dosen_Kelas
Kunci Utama : Id_md
Kunci Tamu : Id_matkul, Id_dosen dan Id_kelas
Fungsi : Untuk menyimpan data gabungan mata kuliah, dosen pengajar dan kelas yang diajar.

Tabel 3.12 Struktur tabel Matkul_Dosen_Kelas

No	Nama Field	Type dan Panjang Field	Keterangan
1	Id_md	Integer (10)	Nomor id mata kuliah & dosen
2	Id_matkul	Integer (10)	Nomor id mata kuliah
3	Id_dosen	Integer (10)	Nomor id dosen
4	Id_kelas	Integer (10)	Nomor id kelas

g) Nama tabel : GA_Kuliah
Kunci Utama : Id_gen_kul
Fungsi : Untuk menyimpan data jadwal perkuliahan hasil proses algoritma genetika

Tabel 3.13 Struktur tabel GA_Kuliah

No	Nama Field	Type dan Panjang Field	Keterangan
1	Id_best_kul	Integer (10)	Kode id generasi
2	Num_gen_kul	tinyint (4)	Nomor generasi proses algoritma genetika
3	Kromosom_kul	Varchar (800)	Susunan kromosom
4	Fitness_kul	double	Nilai fitness kromosom

h) Nama tabel : GA_Ujian
Kunci Utama : Id_gen_uas
Fungsi : Untuk menyimpan data jadwal ujian hasil proses algoritma genetika

Tabel 3.14 Struktur tabel GA_Ujian

No	Nama Field	Type dan Panjang Field	Keterangan
1	Id_best_uas	Integer (10)	Kode id generasi
2	Num_gen_uas	tinyint (4)	Nomor generasi proses algoritma genetika
3	Kromosom_uas	Varchar (800)	Susunan kromosom
4	Fitness_uas	double	Nilai fitness kromosom

i) Nama tabel : Larangan
Kunci Utama : Id_larangan
Fungsi : Untuk menyimpan data permintaan waktu ketidaksanggupan dosen mengajar

Tabel 3.15 Struktur tabel Larangan

No	Nama Field	Type dan Panjang Field	Keterangan
1	Id_larangan	Integer (10)	Nomor id larangan dosen
2	Id_dosen	Integer (10)	Nomor id dosen
3	Id_time	Integer (10)	Nomor id waktu larangan
4	Semester	enum('gasal','genap')	Semester gasal atau genap

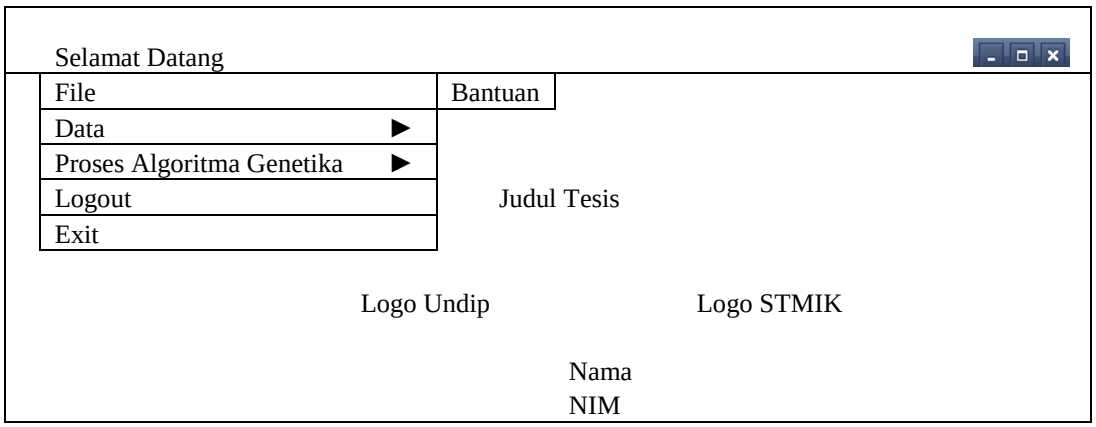
j) Nama tabel : *Timeslot*
Kunci Utama : Id_time
Fungsi : Untuk menyimpan data hari dan jam perkuliahan

Tabel 3.16 Struktur tabel *Timeslot*

No	Nama Field	Type dan Panjang Field	Keterangan
1	Id_time	Integer (10)	Nomor id waktu
2	Hari	enum('Senin','Selasa','Rabu', 'Kamis','Jumat','Sabtu')	Hari perkuliahan
3	Waktu	Varchar(15)	Waktu/Jam kuliah

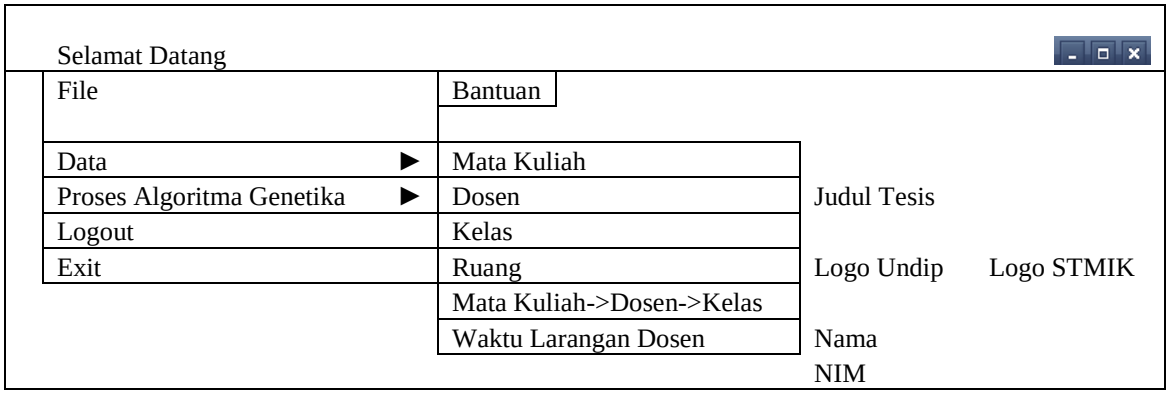
3.3.2.3 Perancangan *User Interface*

Pada aplikasi penjadwalan ini terdapat satu *form* utama yang terdiri dari File dan Bantuan. File memiliki 4 menu yaitu menu Data, Proses Algoritma Genetika, *Logout* dan *Exit*. Sedangkan Bantuan berisi Panduan Penggunaan dan Tentang informasi program. Gambar 3.11 merupakan rancangan *form* utama :

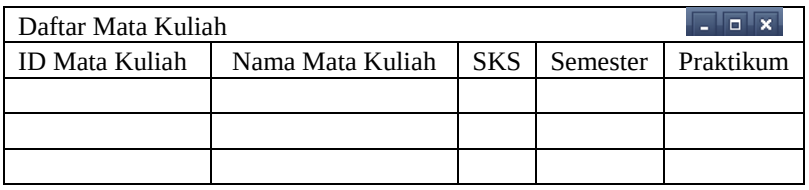


Gambar 3.11 Rancangan *form* utama

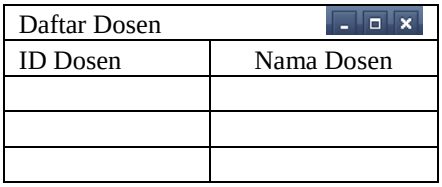
Form Menu Data pada gambar 3.12 merupakan *form* menu untuk meng*input*kan data-data yang diperlukan. *Form* Menu ini terdiri dari 6 (enam) *form* submenu data yaitu *form* submenu data mata kuliah pada gambar 3.13, submenu data dosen pada gambar 3.14, submenu daftar kelas pada gambar 3.15, submenu daftar ruang pada gambar 3.16, submenu relasi mata kuliah, dosen dan kelas pada gambar 3.17 serta submenu waktu larangan dosen mengajar pada gambar 3.18.



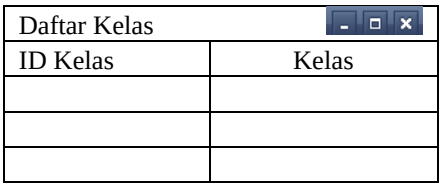
Gambar 3.12 Rancangan *form* Menu Data



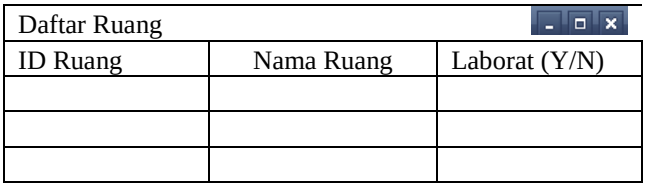
Gambar 3.13 Rancangan *form* submenu data mata kuliah



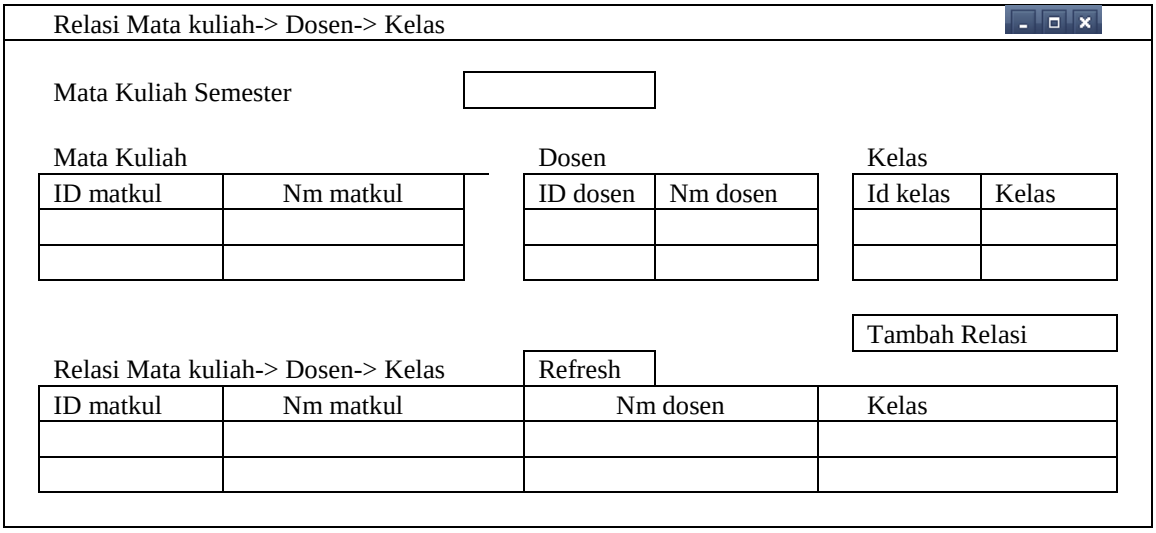
Gambar 3.14 Rancangan *form* submenu data dosen



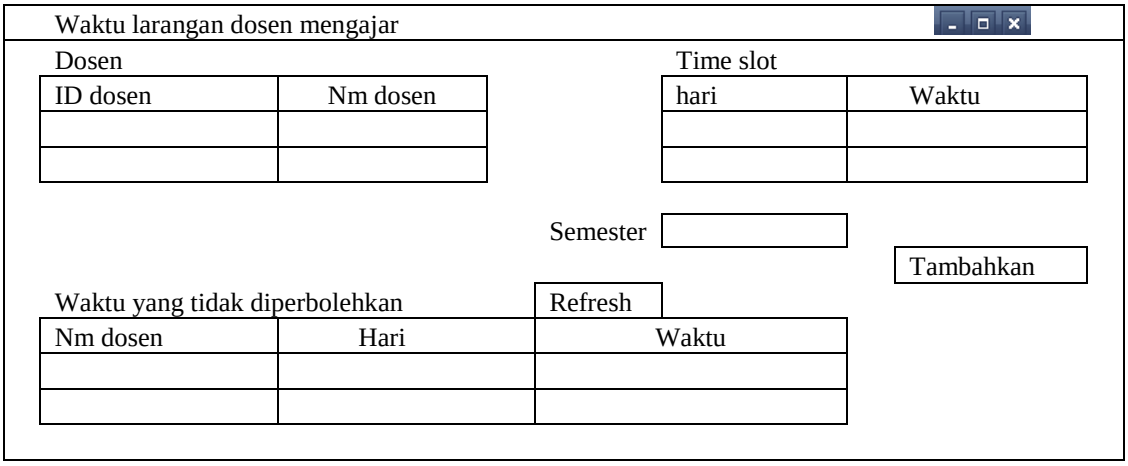
Gambar 3.15 Rancangan *form* submenu daftar kelas



Gambar 3.16 Rancangan *form* submenu daftar ruang

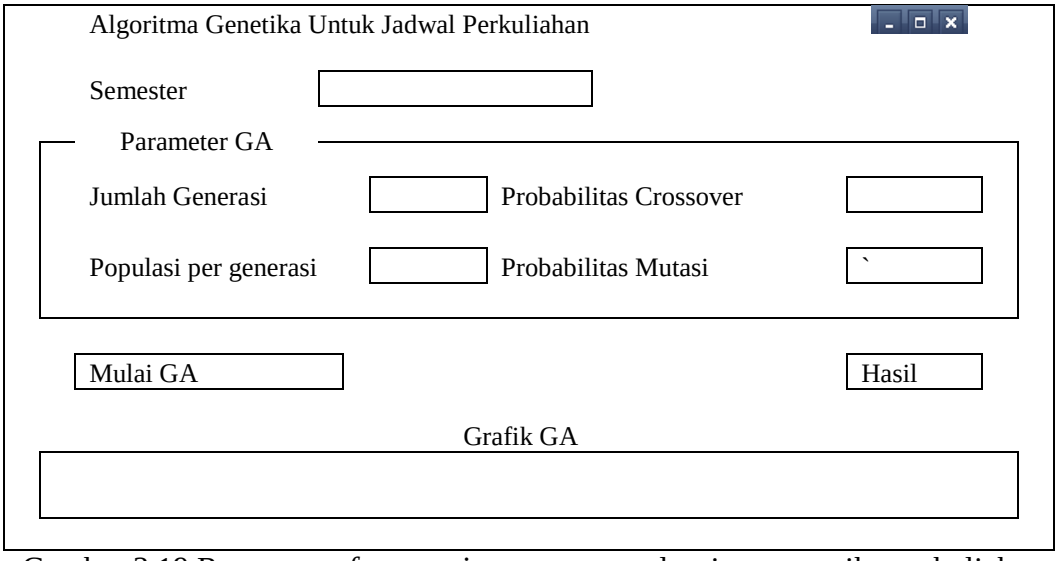


Gambar 3.17 Rancangan *form* submenu relasi mata kuliah, dosen dan kelas

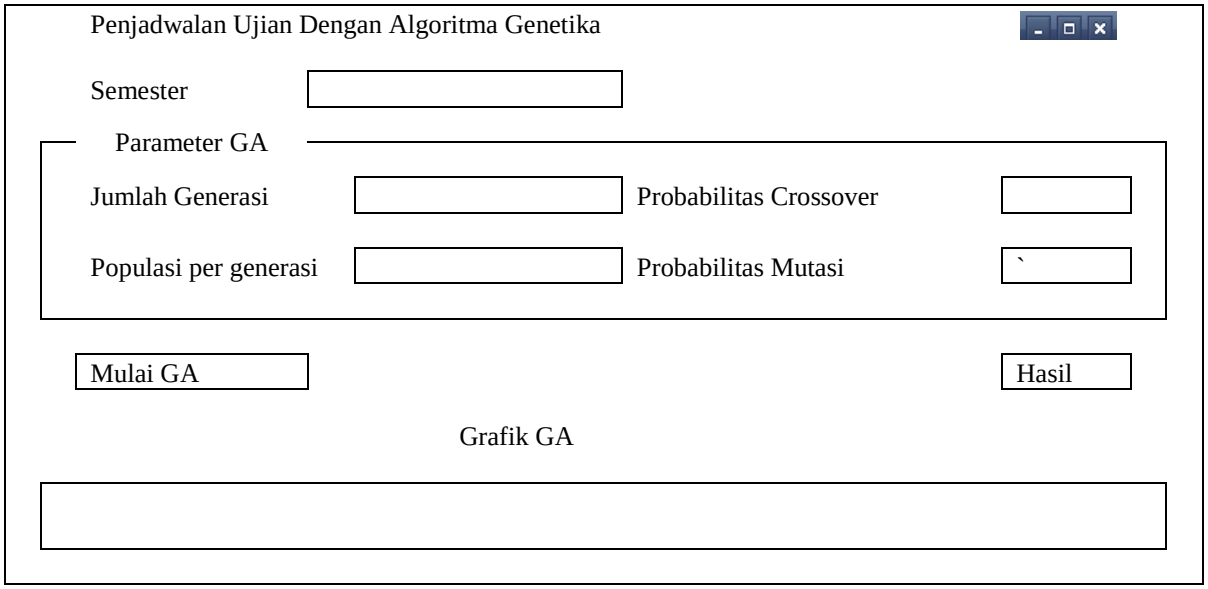


Gambar 3.18 Rancangan *form* submenu waktu larangan dosen mengajar

Form Menu Proses Algoritma Genetika merupakan *form* yang digunakan untuk memproses data-data yang telah di*input*kan sebelumnya dengan menggunakan parameter-parameter algoritma genetika yaitu parameter jumlah generasi atau perulangan, banyaknya populasi, probabilitas *crossover* dan mutasi. Rancangan *form setting* parameter algoritma genetika untuk pemrosesan jadwal perkuliahan dapat dilihat pada gambar 3.19 dan ujian akhir semester dapat dilihat pada gambar 3.20.



Gambar 3.19 Rancangan *form setting* parameter algoritma genetika perkuliahan



Gambar 3.20 Rancangan *form setting* parameter algoritma genetika ujian

Gambar 3.21 memperlihatkan perancangan *form output* hasil proses algoritma genetika baik perkuliahan maupun ujian akhir semester. Tombol Buat Jadwal dari Kromosom terpilih untuk menyimpan jadwal dan rekapitulasi penggunaan ruang yang dihasilkan kedalam format file *Microsoft Excel.*

Gambar 3.21 Rancangan *form output* hasil proses algoritma genetika

3.4 Kesulitan-kesulitan

Kesulitan yang dihadapi selama penelitian ini adalah :

1) Perbedaan hasil analisa dengan pengujian

   Algoritma genetika bekerja berdasarkan suatu bilangan acak *(random)* yang dibangkitkan pada masing-masing solusi sehingga akan

menyebabkan hasil analisa akan berbeda dengan hasil pengujian dengan menggunakan aplikasi yang dibangun.

2) Implementasi

   Tahapan implementasi *(coding)* merupakan tahapan tersulit untuk membangun suatu sistem yang menggunakan algoritma genetika karena diperlukan ketelitian dan kehati-hatian untuk masing-masing prosedur agar dapat menghasilkan solusi yang ingin dicapai.